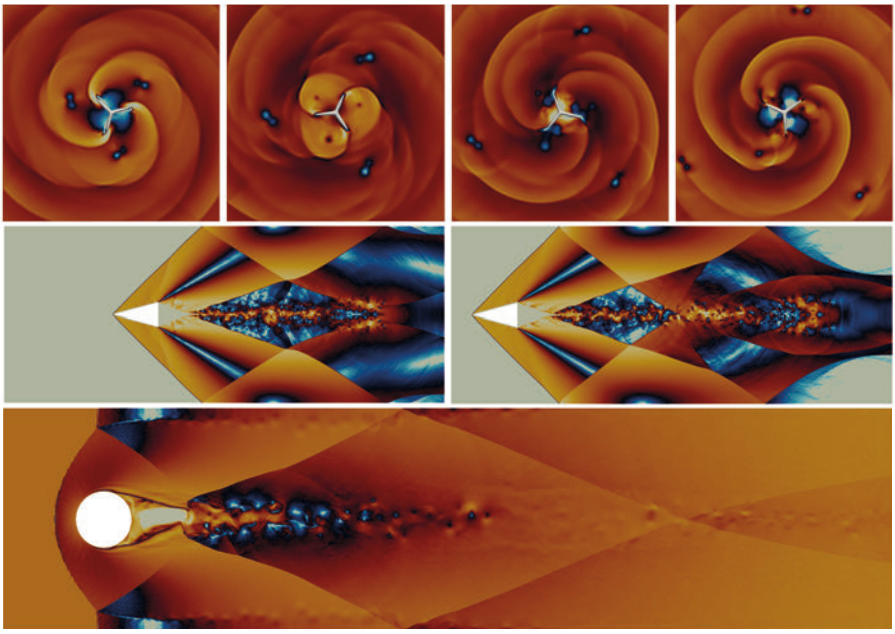
Simulation Techniques in Siegen
Vol. 6

Neda Ebrahimi Pour

Efficient high-order simulation of aeroacoustics from rigid body motion on massively parallel systems



universi
UNIVERSITÄTSVERLAG SIEGEN

Efficient high-order simulation of aeroacoustics from rigid body motion on massively parallel systems

DISSERTATION
zur Erlangung des Grades eines Doktors
der Ingenieurwissenschaften

vorgelegt von

Neda Ebrahimi Pour M.Sc.

eingereicht bei der Naturwissenschaftlich-Technischen Fakultät der
Universität Siegen
Siegen 2021

Betreuerin und erste Gutachterin
Prof. Dr.-Ing. Sabine Roller
Universität Siegen

Zweiter Gutachter
Prof. Hiroyuki Takizawa, Ph.D.
Tohoku Universität

Tag der mündlichen Prüfung
27. Juli 2021

Simulation Techniques in Siegen / STS

Edited by Sabine Roller and Harald Klimach

Vol. 6 (2021)

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the Internet at **http://dnb.d-nb.de**.

Diss. Universität Siegen, 2021
DOI: https://doi.org/10.25819/ubsi/10034

Simulation Techniques in Siegen Vol. 6 / STS Vol. 6 (2021)
Editors: Sabine Roller and Harald Klimach

1st edition univers*i* – Universitätsverlag Siegen 2021

Printing: UniPrint, Universität Siegen
printed on wood- and acid-free paper

univers*i* – Universitätsverlag Siegen
Am Eichenhang 50
57076 Siegen
Germany
info@universi.uni-siegen.de
www.uni-siegen.de/universi

ISBN: 978-3-96182-105-1

Acknowledgement

I want to take the opportunity to thank those who had a significant impact on this thesis and my personal development throughout my time as a doctoral candidate. First of all, I would like to express my gratitude to my supervisor Prof. Dr.-Ing. Sabine Roller, for her honest and direct advice and her understanding and motivation throughout the last years. Thank you Sabine, for always having the right words for me and your contribution to my professional and personal growth. My thesis further benefited from long discussions with my colleague Dr.-Ing. Harald Klimach, who unconditionally supported me throughout my research. Thank you Harald, for always making time for me. I would also like to thank Dr.-Ing. Kannan Masilamani and Nikhil Anand for our fruitful discussions and their support. Then to the members of the STS group, I am incredibly thankful for the time I spent with them, especially the group activities that were memorable for me. Finally, I would also like to thank my colleague Peter Vitt for all the technical support.

Let me also mention my dear mentor, a dear friend, Prof. Dr. Florina Ciorba, who always encouraged me to push myself to the next level to achieve my goals. Thanks, Florina, for all your support and encouragement. Further, I would like to thank my colleague Amin Totounferoush, whom I enjoyed working with. I spent one month at the Cyber Science Center at Tohoku University in Sendai, Japan, and want to thank Prof. Hiroyuki Takizawa and Prof. Ryusuke Egawa for making this research stay possible. I am thankful to Prof. Hiroyuki Takizawa for being part of my doctoral committee. Thanks to all the group members, who warmly welcomed me and helped me to discover the Japanese cuisine. I treasure those moments.

I am grateful to the doctoral committee Prof. Dr.-Ing. Thomas Carolus and Prof. Dr. Roland Wismüller for their support.

I want to express my unfailing gratitude to all my family members for their endless love and encouragement during my doctoral studies. I dedicate this thesis to my grandparents and parents, who diligently support me through all journeys in my life. I am grateful and fortunate to have sisters

whom I can lean on and share my thoughts. Thanks to all of you for caring for me and putting a smile on my face, whenever things did not go well.

Finally, I thankfully acknowledge the financial support of the priority program 1648 – Software for Exascale Computing of the German Research Foundation and the provided computing time on the Supermuc-NG system at the Leibniz Supercomputing Centre (LRZ) in Munich and on the Hawk system at the High-Performance Computing Center (HLRS) in Stuttgart.

Abstract

The numerical simulation of physical problems involves capturing various phenomena occurring simultaneously at different scales in a single simulation. For example, considering aeroacoustic problems, the noise generating flow and the propagation of the sound waves in the far-field need to be taken into account. The increasing computational capacities and the development of modern supercomputers allow for more detailed studies of complex multi-physics and multi-scale problems. In this work, the sound generation by moving obstacles and its propagation up to the far-field is of particular interest. For this purpose, a high-order Discontinuous Galerkin method is utilized to discretize the fluid dynamic equations. These high-order methods are exceptionally efficient as they only require a few degrees of freedom to represent smooth solutions. Therefore, they are often deployed for, e.g., the acoustics far-field, where a homogenous flow field can be found. From the computational perspective on modern high-performing architectures, few degrees of freedom are an exceptional advantage, with memory bandwidth being a bottleneck on modern systems. Additionally, the ratio between communication and computation is minimal due to the loose connection of computational elements at their respective interfaces, which is an additional advantage of these methods, when considering distributed and massively parallel computing systems.

However, the high-order representation of complex geometries has been a critical limitation for their application in various fields. The representation of geometrical shapes has to be appropriate to preserve the quality of the numerical solution, which has been discretized with high-order. Incorporating meshing techniques such as body-fitted meshes might not be robust in the workflow for the simulation. They are required to withstand different scenarios that are common, such as scenarios with general complex geometries inside the simulation domain. They can become expensive in computation when involving simulations with multiple geometries and even more when geometries can move. In these cases, the embedded method, also known as immersed boundary method, provides a promising prospect. In this work, the Brinkman penalization technique is applied to model multiple complex and moving geometries.

Moving rigid bodies are common in engineering applications. The sound emitted due to the motion and the flow disturbance by geometries is of particular interest, as awareness of environmental impact in society has grown in recent years. Therefore, predicting the produced noise is a common responsibility in different fields, such as the design of wind turbines. These simulations have a complex nature and require an efficient strategy to facilitate them feasibly. Therefore we deploy the partitioned coupling approach, where the complex and large simulation domain is decomposed into smaller subdomains. Each subdomain is configured such that the occurring phenomena can be precisely captured. It results in an efficient strategy allowing for the simulation of various scales and physics, such as the large-scale simulation in this work. The simulation includes the motion of an airfoil and the induced noise that spreads over a large domain.

Zusammenfassung

Bei der numerischen Simulation physikalischer Ereignisse sind oft viele unterschiedliche Phänomene involviert, die gleichzeitig auftreten, jedoch unterschiedliche räumliche und zeitliche Skalen haben können. Ist die Schallausbreitung von Interesse, so muss sowohl die schallgenerierende Strömung als auch die Schallausbreitung ins Fernfeld berücksichtigt werden. Durch die Weiterentwicklung von Supercomputern und die vorhandenen Rechenkapazitäten ist es möglich, detalierte Simulationen, die mehrere Skalen und physikalische Phänomene beinhalten, zu berücksichtigen.

In dieser Arbeit wird die Schallerzeugung durch bewegte Geometrien betrachtet, ebenso die Schallausbreitung ins Fernfeld. Um die numerische Simulation mit hoher Genauigkeit und effizient zu realiseren, wird ein Verfahren hoher Ordnung (Discontinuous Galerkin) für die numerische Diskretisierung der Strömungsgleichungen herangezogen. Das Verfahren erlaubt, neben effizienter Berechnung der Strömungsgleichungen, auch eine hohe Genauigkeit der Lösung. Besonders für glatte Lösungen kann das Verfahren seine Vorteile ausspielen. Dabei kann, verglichen mit Verfahren niedriger Ordnung, die selbe Genauigkeit der Lösung mit weniger Freiheitsgraden erzielt werden. Glatte Lösungen sind u.a. im akustischen Fernfeld vorzufinden, wo ein homogenes Strömungsfeld vorliegt. Aufgrund der wenigen Freiheitsgrade ist das Verfahren hoher Ordnung besonders vorteilhaft und erlaubt sehr effiziente Berechnungen auf modernen Supercomputern, deren Speicherbandbreite ein limitierender Faktor darstellt. Des Weiteren wird durch das angewendete Verfahren lediglich zwischen den direkten Nachbarn kommuniziert. Somit ist das Verhältnis zwischen Kommunikation und der eigentlichen Berechnung sehr gering gehalten. Dies ist besonders wichtig und von Bedeutung, wenn Simulationen auf massiv parallelen Rechensystemen ausgeführt werden.

Die Repräsentation von komplexen Geometrien gilt jedoch als Limitierung für Verfahren hoher Ordnung und wird somit kaum für die Darstellung von Geometrien herangezogen. Die Modellierung der Geometrie mit einer niedrigen Verfahrensordnung würde die Vorteile der hohen Verfahrensordnung im Strömungsfeld reduzieren und dazu führen, dass die hohe

Genauigkeit der Lösung nicht mehr gegeben ist. Das Heranziehen von Vernetzungsmethoden, wie z.B. körperangepassten Gittern, erlauben oft keinen robusten Arbeitsablauf, um alle komplexen Szenarien, wie z.B. die Bewegung einer komplexen Geometrie, zu erlauben. Des Weiteren kann das Verwenden von mehreren komplexen Geometrien und insbesondere ihre Bewegung zu Problemen bei der Generierung von körperangepassten Gittern, führen. Bei solchen Problemstellungen bieten eingebettete Methoden eine Möglichkeit, um auch mit Verfahren hoher Ordnung komplizierte Geometrien und deren Bewegung zu modellieren. In dieser Ausarbeitung wird zur Modellierung von Geometrien, das Brinkman Penaliserungsverfahren herangezogen.

Bewegte Bauteil sind im Ingenieurbereich üblich. Umströmte, bewegte Objekte verursachen Störungen der Strömung, die auch Schall erzeugen. Dabei ist es wichtig, bereits in der Entwicklungsphase von z.B. Windturbinen, die Geräuschentwicklung auf ein Minimum zu reduzieren. Die numerische Simulation solcher Anwendungen ist komplexer Natur und setzt eine effiziente Strategie voraus, um das große Simulationsgebiet durch numerische Simulation zu verwirklichen. Dazu wird in dieser Arbeit die Strategie der partitionierten Kopplung verwendet, um eine effiziente Simulation zu ermöglichen. Das Simulationsgebiet wird dabei in Teilgebiete unterteilt und jedes dieser Gebiete so konfiguriert, das auftretende physikalische Phänomene durch die Numerik erfasst werden. Dies ermöglicht physikalische Phänomene numerisch korrekt zu simulieren und den Rechenaufwand zu reduzieren. In dieser Arbeit wird u.a. der durch die Bewegung eines Tragflügels induzierter Schall simuliert.

Contents

Nomenclature

Symbols

β	Shock angle
β	Shock angle
β_e	Exact shock angle
β_e	Exact shock angle
β_n	Numerical shock angle
β_n	Numerical shock angle
$\boldsymbol{F}$	Flux function
$\boldsymbol{f}(\boldsymbol{v})$	Convective flux in x-direction
$\boldsymbol{f}(\boldsymbol{v_\mu})$	Viscous flux in x-direction
$\boldsymbol{F}^*$	Numerical flux function
$\boldsymbol{g}(\boldsymbol{v})$	Convective flux in y-direction
$\boldsymbol{g}(\boldsymbol{v_\mu})$	Viscous flux in y-direction
$\boldsymbol{h}(\boldsymbol{v})$	Convective flux in z-direction
$\boldsymbol{h}(\boldsymbol{v_\mu})$	Viscous flux in z-direction
$\boldsymbol{I}$	Identity matrix
$\boldsymbol{M}$	Mass matrix
$\boldsymbol{m}_u$	Momentum
$\boldsymbol{n}$	Outer surface normal
$\boldsymbol{S}$	Stiffness matrix
$\boldsymbol{u}$	Fluid velocity
$\boldsymbol{v}$	State vector
$\boldsymbol{M^f}$	Face lifting matrix
$\boldsymbol{U_G}$	Velocity of the geometry
χ	Masking function
Δt	Time step
η	Viscous permeability
η_T	Thermal permeability
γ	Heat capacity ratio, isentropic expansion factor
κ	Thermal conductivity
λ	Volume viscosity
μ	Dynamic viscosity
ν	Kinematic viscosity
ψ	Test function
ρ	Density
ρ'	Density perturbation
ρ_B	Background density
ρ_B	Background density
τ	Viscous stress tensor
θ	Deflection angle
θ	Deflection angle
ζ	diffusion constant
A_s	Reference surface
c	Speed of sound
C_D	Drag coefficient
C_L	Lift coefficient
c_p	Specific heat capacity
c_v	Heat capacity at constant volume
d	Spatial dimension
E	Energy
e	Specific internal energy
F_D	Drag force
F_L	Lift force
h	Element size
h_w	Height of the wedge
Ma	Mach number
p	Pressure
p'	Pressure perturbation
p_B	Background pressure
p_{ref}	The lowest sound a healthy ear can hear

p_{RMS}	Root mean square of the sound pressure
Pr	Prandtl number
R	Specific ideal gas constant
Re	Reynolds number
S_{BL}	Area of boundary layer separation
SPL	Sound pressure level
T	Temperature
T_G	Temperature of geometry
U	Characteristic velocity
u'	Velocity perturbation
u_B	Background velocity
v_{FS}	Free stream velocity

Acronyms

O	Scheme order
ODE	Ordinary differential equations
PDE	Partial differential equations
rhs	Right hand side
RK	Explicit Runge-Kutta method
APES	Adaptable Poly-Engineering Simulator
CFL	Courant-Friedrichs-Lewy condition
DG	Discontinuous Galerkin
DNS	Direct numerical simulation
FT	Fourier Transformation
HLRS	High-Performance Computing Center Stuttgart
IMEX	Implicit-mixed-explicit time-stepping scheme
LB	Load balancing
LRZ	Leibniz Rechenzentrum
MPI	Message Passing Interface
nDoF	Degrees of freedom

Notation

Before the introduction into this work and the technical discussions, let us first have a look into the consistent notation throughout this thesis. We review the basic notation of variables and operators used in this work. For $\mathbb{R}^d$ we apply subscripts to differentiate between spatial directions, i.e a specific point $\mathbf{x}$ in $\mathbb{R}^d$ is described by the d-tuple

$$\mathbf{x} = (x_1, \ldots, x_d)^T .$$

Vectors and vector-valued functions are denoted in bold, e.g. in $\mathbb{R}^d$ this results in

$$\mathbf{c} = (c_1, \ldots, c_d)^T .$$

The notation for the dyadic product $\otimes$ of two vectors $\mathbf{c} and \mathbf{e}$ is specified as

$$\mathbf{c} \otimes \mathbf{e} = \mathbf{c} \cdot \mathbf{e}^T = \begin{pmatrix} c_1 e_1 & \cdots & c_1 e_d \\ \vdots & & \vdots \\ c_d e_1 & \cdots & c_d e_d \end{pmatrix} .$$

Further we denote matrices and tensors in bold as well.

$$\mathbf{d} = \begin{pmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{pmatrix}$$

Throughout this thesis we always clearly distinguish between previously mentioned vector notation and tensors. We define for the contraction : of two tensors $\mathbf{a}, \mathbf{b}$ as

$$\mathbf{a} : \mathbf{b} = \sum_j \sum_i a_{ji} b_{ji}.$$

Partial derivatives are defined by

$$\partial_t f = \frac{\partial f}{\partial t}.$$

With the nabla operator all partial derivatives for a specific variable are collected, i.e.

$$\nabla = (\partial_{x_1}, \ldots, \partial_{x_d})^T .$$

The notation for the gradient of a scalar function g is defined as

$$grad(g) = \nabla g = (\partial_{x_1} g, \ldots, \partial_{x_d} g)^T .$$

Further we consider the divergence of a vector-valued function as

$$div(g) = \nabla \cdot g = \sum_{i=1}^{d} \partial_{x_i} g_i.$$

The Jacobi matrix of a vectorial function in

$$\mathbf{g} : \mathbb{R}^d \to \mathbb{R}^n$$

is defined as

$$\nabla \mathbf{g} = \begin{pmatrix} \partial_{x_1} g_1 & \cdots & \partial_{x_d} g_1 \\ \partial_{x_1} g_2 & \cdots & \partial_{x_d} g_2 \\ \vdots & & \vdots \\ \partial_{x_1} g_n & \cdots & \partial_{x_d} g_n \end{pmatrix} .$$

1. Introduction

Numerical simulations became an essential utility in many engineering applications. They provide detailed insights into occurring physics and enable modifications and improvements of devices in the early design process. With the increasing popularity of simulations, the demand for sustainable and efficient simulation strategies has become even more critical when enabling complex simulations. In the last decades, computational architectures have rapidly evolved. According to Moore's law, the computational power doubles every two years [70], even though this statement has been modified due to the changes in the evolution of hardware [45]. This progression in computational power offers the capabilities to solve large simulations and permits the incorporation of more aspects of the real world. However, the increasing complexity in the modeling necessitates being adapted to the available computing resources to allow for their efficient exploitation. For large-scale, particularly multi-scale and nonlinear problems, numerical methods might suffer efficiency and robustness. Considering a typical engineering problem such as fluid-structure-acoustics interaction, we encounter varying scales that must be resolved appropriately. The question arises: How to enable such complex problems from the numerical perspective and, more importantly, how to efficiently allow for these complex simulations. This thesis presents the strategy to enhance efficient and accurate large-scale simulations for rigid body motion induced aeroacoustics on massively parallel systems. The main focus is dedicated to the efficient modeling of complex geometries and their motion.

1.1. Motivation

The simulation of real world problems is of interest in many research fields. Focusing on engineering problems, numerical simulations have become an essential tool for providing insights into different physical phenomena. In addition, they help optimize devices to reduce, e.g., noise pollution, which has gained importance with the worldwide energy transition, leading to energy production through natural resources such as wind turbines. Wind turbines emit noise when producing energy, which can impact human health and may cause health issues for people living close to wind farms.

Often, health issues are exposed as i.a., sleep disturbance, headaches, tinnitus, or even cardiac arrhythmia [4]. According to the German government, wind turbines have to be located 1 km away from neighborhoods. It is intended to increase acceptance by the public. However, due to the increasing awareness of climate change and the governmental aim to reduce the CO_2 emission by 2030, the discussion arose to remove the 1 km policy. Moreover, the 1 km rule was criticized as available space would not be used efficiently [92]. Considering the current valid rule and its possible removal, the demand for appropriate noise reduction measures becomes unavoidable to increase the acceptance of these devices in neighborhoods.

In the context of wind turbines, numerical simulations can help to e.g., optimize rotor blades to reduce noise pollution. Due to the rotation of the blades, a swishing sound appears. However, such simulations involve various scales and have a complexity that needs to be addressed appropriately. More importantly, the requirements for numerical simulations are accurate solutions, thus the adequate representation of the sound source and the sound propagation. Furthermore, numerical simulations have to be efficient in computation, allowing for the realization of complex problems involving multi-physics and multi-scales. Wind turbines are an excellent example of such a complex problem, where the transition from sound generation to its propagation is computationally demanding as different physics and scales are involved. **(i)** Viscous flow around the geometry (rotor blade) that is dominated by small scales such as vortices and **(ii)** acoustic scales that have large wavelengths and can be found away from the geometry.

The first challenge, therefore, is adequate and efficient geometry modeling. Considering the geometrical shape of such wind turbines, especially the shape of the rotor blades, that are curved, the question arises, how they can be numerically represented. Hence the appropriate modeling and the efficient incorporation of geometries in the simulation domain are fundamental. A more realistic numerical simulation of such an engineering problem involves the geometrical shape and its motion, resulting in higher complexity of the numerical simulation, which can become troublesome to realize. The numerical approximation must provide accurate results and preserve numerical stability, while the efficient computation must be maintained.

For numerical simulations, often a 2^{nd} order scheme (low-order) is used to simulate the flow field and the geometrical representation, requiring tiny computational elements, to capture the flow features and to allow for

an appropriate numerical representation of the geometry. Additionally, when its motion and deformation are required, the computational mesh has to be adapted accordingly to capture all changes in the flow field and the motion of the geometry. Therefore, it can become troublesome as the computational mesh requires special treatment, and the efficient computation can be negatively affected as the memory requirements for the computation increase.

High-order methods, on the other hand, allow for very accurate numerical flow simulations. Moreover, they can represent smooth solutions with few degrees of freedom. Thus they require low memory bandwidth for the computation. Therefore, they allow efficient computation as large computational elements can be deployed. The high-order method is especially beneficial for representing long traveling waves due to the numerical scheme's small dissipation and dispersion errors. However, they are seen as a critical limitation when it comes to the representation of complex geometries. The description of curved geometries has to match the discretization scheme. An option to attain a high-order geometry description is to use deformed computational elements at the geometry interface. Unfortunately, though, curved elements are challenging to deal with and are prone to numerical instabilities. For flow simulations with moving geometries, curved elements can even be prohibitive. An efficient and promising approach to represent geometries using a high-order scheme can be achieved by embedding geometrical information into the fluid dynamic equations. As a result, both the flow field and the geometry can be discretized with the same underlying high-order scheme while allowing for efficient numerical computation.

The second challenge we need to overcome is the noise propagation of such wind turbines. Depending on their size, the noise can spread over hundreds of meters. Simulating such a large computational domain requires an enormous amount of computational resources. The increasing computational power and the upcoming exascale era make such multi-physics and multi-scale problems numerically feasible. However, they require efficient strategies for their realization. A typical scenario of such a multi-physics and multi-scale problem is the interaction of fluid-structure and acoustics. In the case of wind turbines, rotating blades interact with the surrounding flow and induce noise due to turbulent flow. The aim of the ExaFSA project [37], which was part of the German priority program SPPEXA [86], was to realize such large-scale and complex simulations, where sound is generated through fluid-structure interaction and propagated across a

large area. However, solving such complex problems with only one set of equations and the same spatial discretization in the entire domain is unfeasible.

We know that away from the geometry, viscous effects do not play a role anymore, and further away, only acoustic waves are present, then the fluid-structure and acoustics problem can be spatially separated. Thus, the overall complex computational domain can be decomposed into smaller subdomains, such as solving smaller tasks. Subdomains exchange information at joint interfaces, the so-called coupling interfaces. The decomposition and the coupling of the subdomains provide a dedicated numerical treatment of each of them. This strategy enables solving complex and large problems more efficiently, such that compute-intensive approximations are used where it is necessary and simplified as soon as specific physical phenomena can be represented with simpler equations.

1.2. Related work

In recent years significant investigations for coupled simulations have been conducted. In this work, we distinguish between **(i)** the coupling of modeling terms with the conservation equations to model the geometry, referred to as the embedded or immersed boundary method. **(ii)** The decomposed of the simulation domain into subdomains that are coupled together through a coupling approach, referred to as partitioned coupling. For both cases, a short overview of recent activities in research is given in the following sections. A more detailed review is presented in the respective chapters of this work.

1.2.1. Embedded method

The modeling methods for geometry representation have been advanced with many scientific contributions. Different techniques have been proposed to facilitate simulations with complex geometries and allow for precise geometry modeling. A wide range of applications and methods employed to realize fluid-structure interaction (FSI) is presented in [12]. Generally, the FSI problem can be classified according to two criteria [48]. The first one is based on how the physical equations of fluid and structure are coupled together and solved. The second criterion is according to the mesh utilized to discretize the simulation domain. We focus here on the second criterion. More details on the first criterion can be found in Chapter 7.

The second criterion, that is, according to the mesh, can be distinguished between the body-fitted mesh methods and the embedded methods. If the mesh is fitted towards the geometry (body-fitted), the geometrical interface is considered a physical boundary condition. Due to the geometry's movement or deformation, re-meshing or mesh adaptation is required as the solution advances. In case an embedded method is deployed, the boundary location and the geometry interface are implied as constraints in the equations to be solved, allowing for computational meshes that are preserved throughout the simulation [48]. This method does not require an adaptation of the mesh, even though the motion of the geometry is desired. Thus, it provides an efficient method to model arbitrary complex geometries for numerical simulations.

The embedded method was first proposed by Peskin in 1977 [73], who represented the structure through a boundary in the fluid dynamic equations to be solved. The fluid-structure force was explicitly computed and later fed into the fluid motion equations [72]. The disadvantage of this method was the structural response to the fluid domain, which was not accordingly modeled. Therefore, this method was extended, where a finite volume was used to advance the modeling of complex geometries [32].

The modeling approach incorporated in this work considers the embedded approach, where a finite volume is deployed to model complex moving geometries. We need to emphasize that most studies with the embedded approach are based on incompressible flows. However, as this method has grown in popularity, more investigations were conducted for compressible flow simulations of aeroacoustic problems e.g., in [49], [67] or [57]. To the best of our knowledge, this method has not been incorporated before in the literature for compressible flow simulations combined with moving geometries. Furthermore, this method has not been used with the high-order discretization method deployed in this work.

1.2.2. Partitioned coupling

Due to the complexity of fluid and acoustics interaction and the inherent multi-physics and multi-scale properties, various theories have been developed and investigated to enable such simulations in recent years. In this section, a brief overview of the realization of fluid-acoustic interaction is provided. The numerical simulation of aeroacoustics has become important in the strive of noise reduction. Investigations are not only restricted

to the near-field acoustics but also include the far-field. However, the simulation of near-field and far-field can become very compute-intensive. Therefore various investigations have been dedicated to the numerical simulation of both fields in an efficient manner. A comprehensive strategy that has proven to be very efficient is the partitioned coupling approach. This method decomposes the complex simulation domain and solves each according to the physical requirements. Whereas the decomposed subdomains can be connected through volume or surface coupling. In the case of volume coupling, volume information is exchanged; thus, coupling domains coincide and are often used in acoustics analogies. The incompressible Navier-Stokes equations are solved, source terms (e.g., pressure and velocity fluctuations) extracted, and provided to the acoustics solver. For this approach, e.g., the Lighthill analogy [63] is utilized, which is used, e.g., in the solver, *FASTEST* [58, 65]. It allows to compute the flow field first and, in a later step, the acoustics perturbation. Hence, both fluid and acoustics domains can be separately computed. However, a disadvantage of this approach is the uni-directional coupling of the fluid flow to the acoustics far-field, where information is transferred from the fluid domain to the acoustics domain, but not vice versa. Moreover, more memory is required for the computation since volume information needs to be exchanged between the subdomains.

Instead of separately computing source terms and utilizing the volume coupling, a more efficient approach is the surface coupling. Again, the simulation domain is decomposed but coupled together only at the surfaces of the respective domains. Thus information is exchanged in two-dimensional space (at the surfaces). In the flow field, the compressible Navier-Stokes equations are solved. Hence information about the fluctuation in, e.g., pressure is available and does not necessitate to be separately computed. Assuming a homogenous field is encountered in the far-field, thus only the acoustics propagation, simplified equations can be considered and solved to capture the occurring physics. This approach permits efficient and inexpensive computations, as **(i)** less memory and communication is needed between the domains, due to the surface coupling and **(ii)** the acoustics information is available in the equations to be solved. It does not require the additional computation of source terms [76].

More details on the partitioned coupling can be found in Chapter 7.

1.3. Aim of this work

This work aims to enable the coupled simulation of rigid body motion induced aeroacoustics. In the first part of this work, we examine the geometry representation and its validation in our numerical solver, where the embedded method is deployed for the modeling. Different scenarios are studied to address common geometrical constraints that are typical for engineering applications. Investigations cover various applications in different spatial directions, from a simple reflection of an acoustics pulse up to the interaction of shocks and shock formation. Additionally, flat, curved, and sharp geometrical shapes are investigated. A thorough search of the relevant literature does not yield an extensive examination of the embedded method used in this work. Afterward, we concentrate on the coupling strategy, where the complex simulation domain is decomposed into subdomains connected through joint coupling interfaces. Two coupling approaches are analyzed to enable the data exchange between them. In one case, an external black-box coupling approach is applied, where each subdomain receives an independent executable and is treated as a black-box. In the other case, the coupling approach is integrated into the simulation framework. Here each solver is seen as a library, and one executable is used for all subdomains. The second coupling approach has the advantage of exploring the knowledge of the internal data structure. The main focus is devoted to the accuracy of the solution and the optimization of the coupling strategy. Finally, all findings are used to achieve the goal of this work: The numerical simulation of a 3-field coupled problem with a moving geometry, which disturbs the fluid flow, and noise is generated due to turbulent flow and propagates up to the far-field.

1.4. Outline

The outline of this work is as follows: In the first chapter, the relevant fluid dynamic equations are revisited. Afterward, the numerical discretization methods in space and time are introduced. The high-order Discontinuous Galerkin method is utilized in space and the explicit Rung-Kutta method for the time discretization. Chapter 4 gives an overview of the simulation framework used to accomplish the numerical simulations in this thesis. In Chapter 5 the embedded method is presented, used to model arbitrary geometries with the same discretization method as the flow domain. Afterward, the geometry modeling is validated through several well-studied test cases from the literature. Hereafter the coupling strategy and challenges associated with that method are studied, and enhancements in terms

of efficient computation are introduced (cf. Chapter 7). In Chapter 9 numerical results of different geometrical shapes and different numerical challenges for the moving geometry are presented. Later, simulation results of a 3-field coupled simulation with a moving airfoil are presented and used to investigate the acoustics far-field (cf. Chapter 10). Finally, all findings are summarized and a conclusion drawn in Chapter 11.

2. Fluid dynamic equations

The goal of this work is the simulation of aeroacoustic problems that are governed by compressible flows. These can be described by the compressible Navier-Stokes equations. They often can be simplified in parts of the simulation domain as some physical phenomena do not play a role anymore. They are described by the equations to be solved but are not relevant in those parts. Hence the fluid dynamic equations to be solved can be reduced in complexity and computational effort.

In this chapter, the focus is devoted to the governing equations used to describe the fluid motion. In the Sections 2.1, 2.2 and 2.3 the field of continuums mechanics is revisited, where the governing equations of fluid dynamics will be explained. In Chapter 3 the numerical method to discretize those governing equations is introduced and explained in more detail.

2.1. Compressible viscid Navier-Stokes equations

Generally, compressible fluids can be described by the compressible viscid Navier-Stokes equations, given in three equations. They define the conservation of mass, momentum, and energy. For a Newtonian fluid, they are given by [62]

$$\partial_t \rho + \nabla \cdot \boldsymbol{m_u} = 0 \tag{2.1a}$$

$$\partial_t \boldsymbol{m_u} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{m_u} + p\boldsymbol{I}) - \nabla \boldsymbol{\tau} = \rho \boldsymbol{f} \tag{2.1b}$$

$$\partial_t E + \nabla \cdot (\boldsymbol{u}\,((E+p))) - \nabla \cdot (\boldsymbol{\tau u} + \gamma \nabla T) = -\boldsymbol{m_u} \cdot f, \tag{2.1c}$$

with $\boldsymbol{\tau}$ being the viscous stress tensor, that is defined as

$$\boldsymbol{\tau} = \mu \left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T \right) - \lambda (\nabla \cdot \boldsymbol{u}) \boldsymbol{I}. \tag{2.2}$$

In Eq. (2.1a) the conservation of mass, Eq. (2.1b) the conservation of momentum and lastly Eq. (2.1c) the conservation of energy is presented. Thereby, the density is given by ρ, the velocity by $\boldsymbol{u}$, the momentum with $\boldsymbol{m_u} = \rho \cdot \boldsymbol{u}$, p the pressure, $\boldsymbol{I}$ being the identity matrix and E the energy.

The dynamic viscosity of the fluid is given by μ, while λ is related to the volume viscosity and is given as $\lambda = 2\mu/3$ for the ideal gas phase. Further, the temperature is given with T and the thermal conductivity of the fluid with γ. On the right hand side (rhs) of Eq. (2.1b) and Eq. (2.1c) a force vector is imposed on the fluid, which can contain e.g gravitation or any other desired source term, acting on the fluid. The equation system is closed by considering the equation of state for the ideal gas

$$p = \rho RT = (\gamma - 1)\left(E - \frac{\rho \boldsymbol{u}^2}{2}\right), \tag{2.3}$$

where γ denotes the heat capacity ratio c_p/c_v and R (cf. Eq.(2.4)) the ideal gas constant.

$$R = c_p - c_v = (\gamma - 1)c_v \tag{2.4}$$

Additionally, the absolute temperature of the fluid can be calculated according to

$$c_v T = e, \tag{2.5}$$

where e is the internal energy and is defined as $e = (E - \rho \boldsymbol{u}^2/2)/\rho$.

For the characterization of the fluid motion, dimensionless numbers are helpful parameters to describe the property of the flow. The Mach number (Ma) provides a ratio between the speed of the fluid motion and the speed of sound. Further the Ma indicates how large the impact of compressibility effects are on the flow.

$$Ma = \frac{\|u\|}{c} \tag{2.6}$$

With c being the speed of sound defined as

$$c = \sqrt{\gamma \frac{p}{\rho}}. \tag{2.7}$$

Another dimensionless number is the Reynolds number (Re), which is interpreted as the ratio between inertial to viscous forces within the fluid. Hence flows with a high Re are dominated by convection, while a low Re indicates a flow dominated by dissipation

$$Re = \frac{\rho U L}{\mu} \tag{2.8}$$

with U being the characteristic velocity and L the characteristic length of the flow, respectively. Lastly, we consider the dimensionless Prandtl

number which is specified as

$$Pr = \frac{c_p \mu}{\gamma} = \frac{c_v \gamma \mu}{\gamma}. \tag{2.9}$$

Pr denotes the relation between momentum viscosity and thermal conductivity. It describes the ratio of the momentum boundary layer thickness to the thermal boundary layer thickness. Further dimensionless numbers can be found e.g. in [62].

In some investigations, it might be helpful to use the primitive variables. Those primitive variables can be determined by transformation of the conservative variables (ρ, $\boldsymbol{m_u}$ and E). The primitive variables are obtained from the conservative variables as

$$\rho = \rho \tag{2.10a}$$

$$\boldsymbol{u} = \frac{\boldsymbol{m_u}}{\rho} \tag{2.10b}$$

$$p = (\gamma - 1)\left(E - \frac{\rho \boldsymbol{u}^2}{2}\right). \tag{2.10c}$$

In this work, we mainly consider for the discussion of the numerical results the primitive variables. However, from an implementational perspective, the conservative form as stated is used.

2.1.1. Boundary conditions

For the viscous compressible Navier-Stokes the following boundary conditions are considered:

- *Subsconic inflow:* Dirichlet conditions are applied for $\boldsymbol{u}$ and ρ.
- *Subsonic outflow:* A Dirichlet condition for pressure p is prescribed along the boundary.
- *Supersonic inflow:* At this Dirichlet boundary all primitive quantities (ρ ,$\boldsymbol{u}$ and p) are imposed, while all flow information is propagated downstream.
- *Supersonic outflow:* The flow information is propagated downstream, i.e. no quantity is prescribed along this boundary.

- *Primitives:* All primitive variables are applied, resulting in Dirichlet conditions. Additional conditions are imposed for the gradients of all quantities, i.e. $n \cdot \nabla p = 0$, $n \cdot \nabla \rho = 0$ and $n \cdot \nabla \boldsymbol{u} = 0$.
- *No-slip-wall, isothermal:* Dirichlet conditions for $\boldsymbol{u}$ in all spatial directions as well as for the temperature, i.e. imposing $\boldsymbol{u} = 0$ and the initial temperature $T = T_0$ along the wall.
- *No-slip-wall, adiabatic:* Dirichlet boundary conditions are imposed, which are similar to the isothermal wall condition. However, a Neumann condition is prescribed for the temperature with $n \cdot \nabla T = 0$ along the wall.
- *Slip-wall condition:* Dirichlet condition is used on the normal velocity direction $\boldsymbol{n} \cdot \boldsymbol{u}$.

Variables not prescribed at the boundaries are extrapolated.

2.2. Compressible inviscid Euler equations

For an inviscid fluid, the Navier-Stokes equations can be simplified to the inviscid Euler equations. Hereby the viscous effects are neglected and the temperature is constant. The equations defined in Eq. (2.1) can be simplified to [62]

$$\partial_t \rho + \nabla \cdot \boldsymbol{m_u} = 0 \tag{2.11a}$$

$$\partial_t \boldsymbol{m_u} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{m_u} + p\boldsymbol{I}) = p\boldsymbol{f} \tag{2.11b}$$

$$\partial_t E + \nabla \cdot (\boldsymbol{u}((E + p))) = -\boldsymbol{m_u} \cdot f. \tag{2.11c}$$

In Eq. (2.11) again the conservation of mass, momentum, and energy are provided. To close the equation system from the mathematical perspective, the equation of state for the ideal gas Eq. (2.3) is used.

2.2.1. Boundary conditions

Boundary conditions mentioned in 2.1.1 can be employed for the inviscid compressible Euler equations, as well. However, as the viscous terms are not present for the Euler equations, gradients are not needed here.

- *Primitives:* All primitive variables are applied, resulting in Dirichlet conditions as in Section 2.1.1. However, the gradients are not required and do not have to be computed.

A further simplification of the inviscid Euler equations results in the linearized Euler equations. They are of interest for e.g. aeroacoustic problems, where the propagation of pressure waves and their impact on the surroundings is of interest. In the following section, they are introduced and described in more detail.

2.3. Linearized Euler equations

For some problems, non-linear effects do not have any impact on the solution e.g. for aeroacoustics, and therefore can be neglected in the computation of the solution. For that, the viscid Euler equations can be linearized (cf. Eq. (2.11)) around a constant background. These equations are attractive when e.g. the propagation of pressure waves in a larger area is of interest, which is the case in the acoustics far-field. The computation of these equations is inexpensive, as non-linear or viscid terms do not play a role anymore. For the linearization around a constant background, the state variables are split into a constant background and a perturbation part, denoted with a subscript c and $'$, respectively. With that, the conservation of density, momentum, and energy can be rewritten as [88]

$$\partial_t\ \rho' + \nabla \cdot \underbrace{\left(\boldsymbol{u}_c \rho' + \rho_c \boldsymbol{u}'\right)}_{:=\boldsymbol{m}_{\boldsymbol{u}}} = 0 \tag{2.12a}$$

$$\partial_t\ \boldsymbol{u}' + \nabla \cdot \left(\boldsymbol{u}_c \boldsymbol{u}' + \frac{1}{\rho_c} p'\right) = 0 \tag{2.12b}$$

$$\partial_t\ p' + \nabla \cdot \left(\boldsymbol{u}_c p' + \gamma p_c \boldsymbol{u}'\right) = 0. \tag{2.12c}$$

In Eq. (2.12) the multiplication of different perturbations is neglected as they are small scales and result in even smaller ones. Additionally, the time derivatives of the constant background variables can be eliminated as the background is, as mentioned, time-independent.

2.3.1. Boundary conditions

Boundary conditions mentioned in Section 2.1.1 can also be considered for the linearized Euler equations. However, the variables prescribed at the boundaries are the perturbations of all quantities, as the background has predefined values.

In the next section, the discretization method used to discretize the fluid

dynamic equations is explained. Afterward, the focus is devoted to the time integration.

3. Discretization in space and time

This chapter is devoted to the numerical discretization in space and time of the fluid dynamics equations mentioned in Chapter 2. We consider the Discontinuous Galerkin Finite Element method. The significant advantage of this method is the high locality. At the same time, the volumetric information within each element is tightly coupled. The interaction between the elements occurs only on the element interfaces. Hence only planar information has to be exchanged, resulting in a small ratio of communication to computation. The inherent property of the high locality of this method makes it well suited when running on modern distributed parallel computing architectures. By restricting the numerical scheme to cubical elements for the computational mesh, operations can be carried out through a dimension by dimension strategy in those elements, eligible for efficient parallel execution. An additional benefit of this method is its high-order accuracy, which is very attractive for many applications in different scientific areas. For a smooth solution, the high-order discretization can compute the numerical solution of a given quality with fewer degrees of freedom compared to a method that uses a lower order scheme. This benefit can also be taken into consideration for parallelization and large-scale applications when considering memory bandwidth. Using fewer degrees of freedom results in a reduced amount of memory consumption, which is a bottleneck on modern computing systems. Hence, this method is well suited for today's supercomputer architectures, which provide faster computation (floating-point operations) but have less memory bandwidth per compute unit. We aim for efficient large-scale simulations by fully deploying the capacities of this method. For the realization of the simulations, we first discretize the partial differential equations (PDE) in space, which results in ordinary differential equations (ODE) in time.

This chapter is structured as follows: In the first section, we recall the Discontinuous Galerkin method for hyperbolic problems, where the inviscid Euler equations are considered. Hereafter, we revisit the discretization of the viscous part of the parabolic mixed hyperbolic Navier-Stokes equations. Afterward, we focus on the time integration used in this work.

3.1. Spatial discretization - Discontinuous Galerkin method

The Discontinuous Galerkin Finite Element method has gained popularity as a numerical method for hyperbolic conservation laws. The user community has intensively increased, as this method is well suited for a highly parallel and distributed execution. Reed and Hill [80] were the first to introduce this method by solving the steady-state neutron transport equations. Consequently, this method was further investigated and developed (cf. e.g [26] and [25]) resulting in the utilization of this method in various areas of research. From linear to nonlinear problems, from low- to high-order equations, and from conservative to non-conservative formulations. Interested readers are referred to [46] and [74] for a full review regarding this discretization method.

3.1.1. Compressible inviscid Euler equations

In this section, the variational formulation of the Discontinuous Galerkin Finite Element method for the inviscid Euler equations is derived and explained in more detail. This method can be seen as a hybrid method, combining the high-order representation within elements from the Finite Element and the connection via fluxes known from the Finite Volume Method. To derive the variational formulation of the Discontinuous Galerkin method, we first consider the conserved quantities represented by the vector $\boldsymbol{v}$. With the physical flux function $\boldsymbol{F}$ being a vectorial function, we obtain the following equation

$$\partial_t \boldsymbol{v} + \nabla \boldsymbol{F}(\boldsymbol{v}) = 0, \tag{3.1}$$

with appropriate initial and boundary conditions. At first, we neglect any source term and require the rhs to be zero. The vector $\boldsymbol{v}$ contains all conserved quantites and the flux function is given as $\boldsymbol{F}(\boldsymbol{v}) = (\boldsymbol{f}(\boldsymbol{v}), \boldsymbol{g}(\boldsymbol{v}), \boldsymbol{h}(\boldsymbol{v}))^T$.

$$\boldsymbol{v} = \begin{pmatrix} \rho \\ \rho u_x \\ \rho u_y \\ \rho u_z \\ \rho E \end{pmatrix} \tag{3.2}$$

For a three dimensional problem $\boldsymbol{f}(\boldsymbol{v})$, $\boldsymbol{g}(\boldsymbol{v})$ and $\boldsymbol{h}(\boldsymbol{v})$ are the convective fluxes in all three directions. In the Euler equations these fluxes are defined

as:

$$\boldsymbol{f}(\boldsymbol{v}) = \begin{pmatrix} \rho u_x \\ \rho {u_x}^2 + p \\ \rho u_x u_y \\ \rho u_x u_x \\ (\rho E + p) u_x \end{pmatrix}, \boldsymbol{g}(\boldsymbol{v}) = \begin{pmatrix} \rho u_y \\ \rho u_y u_x \\ \rho {u_y}^2 + p \\ rho u_y u_z \\ (\rho E + p) u_y \end{pmatrix}, \tag{3.3}$$

$$\boldsymbol{h}(\boldsymbol{v}) = \begin{pmatrix} \rho u_z \\ \rho u_z u_x \\ \rho u_z u_y \\ \rho {u_z}^2 + p \\ (\rho E + p) u_z \end{pmatrix}$$

where ρ, u_x, u_y, u_z, E and p denote the density, the velocity in all three directions, the total energy and the pressure, respectively. The first step to achieve the variational formulation of the conservation laws, is to consider the above equation (Eq.(3.1)) and multiply it by a test function ψ

$$\partial_t \boldsymbol{v} \psi + \nabla \boldsymbol{F}(\boldsymbol{v}) \psi = 0. \tag{3.4}$$

Afterward, Eq. (3.4) is integrated over the domain Ω, and integration by parts is applied, leading to the weak formulation.

$$\int_\Omega \partial_t \boldsymbol{v} \cdot \psi d\Omega + \oint \boldsymbol{F} \psi \cdot \boldsymbol{n} dS - \int_\Omega \boldsymbol{F}(\boldsymbol{v}) \cdot \nabla \psi d\Omega = 0 \tag{3.5}$$

Here dS denotes the surface integral and $\boldsymbol{n}$ the normal vector at the surface. A discrete formulation of the above equation is obtained by dividing the domain Ω into m non-overlapping and closed elements, which is given by $T = \Omega_j | i = 1, 2, 3, ..., m$, so that $\Omega = \cup_{j=1}^{m} \Omega_j$ and $\Omega_j \cap \Omega_i = \emptyset \forall j \neq i$. Defining a finite element consisting of discontinuous polynomial functions of degree $p \geq 0$ by considering

$$P^p = \{ f \in [L^2(\Omega)]^p \}. \tag{3.6}$$

With this formulation, we are allowed to write the approximate solution $\boldsymbol{v}_h(\boldsymbol{x}, t)$ within each element considering a polynomial function of degree p as

$$\boldsymbol{v}_h(\boldsymbol{x}, t) = \sum_{j=1}^{p} \hat{v}_j \phi_j, \quad \psi_h(\boldsymbol{x}) = \sum_{j=1}^{p} \hat{w}_j \phi_j. \tag{3.7}$$

The expansion coefficients given with $\hat{v}_j$ and $\hat{w}_j$ denote the degrees of freedom of the solution and the test function respectively. In the next step Eq. (3.5) is used, with a sum of integrals over the elements Ω_j instead of the entire domain.

$$\sum_{i=1}^{m} \partial t \int_{\Omega_j} \psi_h \boldsymbol{v}_h d\Omega + \oint_{\partial\Omega_j} \psi_h \boldsymbol{F}^* \cdot \boldsymbol{n} dS - \int_{\Omega_j} \nabla\psi_h \cdot \boldsymbol{F}(\boldsymbol{v}_h) d\Omega = 0, \quad \forall \psi_h \in P^p \tag{3.8}$$

Since the element support is strongly local, the flux term on the element surface is not uniquely defined. Therefore the physical flux $\boldsymbol{F}$ is replaced by a numerical flux function $\boldsymbol{F}^*(\boldsymbol{v_h^-}, \boldsymbol{v_h^+}, \boldsymbol{n})$, with $\boldsymbol{v_h^-}$ and $\boldsymbol{v_h^+}$ representing the interior and exterior traces at the surface of the element, respectively. Hence the numerical flux connects the neighboring elements, resulting in a weak link between them. Considering the numerical flux function, the expansion coefficients, and rearranging Eq. (3.8), we obtain the final weak formulation as

$$\sum_{i=1}^{m} \partial t \int_{\Omega_j} \sum_{j}^{p} \psi_h \hat{v}_j \phi_j d\Omega = \int_{\Omega_j} \nabla\psi_h \cdot \boldsymbol{F}(\boldsymbol{v}_h) d\Omega - \oint_{\partial\Omega_j} \psi_h \boldsymbol{F}^* \cdot \boldsymbol{n} dS = 0, \quad \forall \psi_h \in P^p. \tag{3.9}$$

Rewriting Eq. (3.9) to a matrix-vector notation results in

$$\partial_t \hat{\boldsymbol{v}}(t) = \boldsymbol{M}^{-1} \cdot \left(\boldsymbol{S} \cdot \hat{\boldsymbol{F}}(\hat{\boldsymbol{v}}) - \boldsymbol{M}^F \cdot \hat{\boldsymbol{F}}^*(\hat{\boldsymbol{v}}) \right), \tag{3.10}$$

where $\boldsymbol{M}$ denotes the mass matrix, which is element local, and M^{-1} the inverse of it, that can be computed element by element. $\boldsymbol{S}$ and $\boldsymbol{M}^F$ are the stiffness and the face mapping matrix, respectively. The functions $\hat{\boldsymbol{F}}$ and $\hat{\boldsymbol{F}}^*$ denote the modal expansion coefficient counterparts of the physical flux $\boldsymbol{F}$ and the numerical flux $\boldsymbol{F}^*$ in each case. Different numerical approaches are available for the numerical flux, and the simplest might be the Lax-Friedrich scheme. The reader is referred to [46] for more detailed information about suitable numerical fluxes for the high-order Discontinuous Galerkin method. The formulation in Eq. (3.10) can be solved in time with any time-stepping method, e.g., the Runge-Kutta method that is used in this work. In Section 3.2 this time discretization method is described in more detail.

The difference between the Euler and Navier-Stokes equations are majorly

the viscous terms, which are neglected for the Euler equations. In the next paragraph, the discretization of the viscous part of the Navier-Stokes equations is shown.

3.1.2. Compressible viscous Navier-Stokes equations

This section is devoted to the discretization of the viscous terms of the Navier-Stokes equations. We recall Eq, (2.1) for this purpose.

$$\partial_t \rho + \nabla \cdot \boldsymbol{m_u} = 0 \tag{3.11a}$$

$$\partial_t \boldsymbol{m_u} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{m_u} + p\boldsymbol{I}) = \nabla \cdot \underbrace{\left(\mu \left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T\right) - \lambda(\nabla \cdot \boldsymbol{u})\boldsymbol{I}\right)}_{:=\boldsymbol{\tau}} \tag{3.11b}$$

$$\partial_t E + \nabla \cdot (\boldsymbol{u}\,((E+p))) = \nabla \cdot (\boldsymbol{\tau u} + \kappa \nabla T) \tag{3.11c}$$

In this section we are only interested in the viscous terms of Eq.(3.11) and rewrite the viscous fluxes in three-dimensional space through the vectors

$\boldsymbol{f}(\boldsymbol{v_\mu})$, $\boldsymbol{g}(\boldsymbol{v_\mu})$ and $\boldsymbol{h}(\boldsymbol{v_\mu})$.

$$\boldsymbol{f}(\boldsymbol{v_\mu}) = \begin{pmatrix} 0 \\ (2\mu+\lambda)\frac{\partial u_x}{\partial x} + \lambda\left(\frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z}\right) \\ \mu\left(\frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y}\right) \\ \mu\left(\frac{\partial u_z}{\partial x} + \frac{\partial u_x}{\partial z}\right) \\ (\boldsymbol{\tau}\cdot\boldsymbol{u})_x + \kappa\partial_x T \end{pmatrix},$$

$$\boldsymbol{g}(\boldsymbol{v_\mu}) = \begin{pmatrix} 0 \\ \mu\left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right) \\ (2\mu+\lambda)\frac{\partial u_y}{\partial y} + \lambda\left(\frac{\partial u_x}{\partial x} + \frac{\partial u_z}{\partial z}\right) \\ \mu\left(\frac{\partial u_z}{\partial y} + \frac{\partial u_y}{\partial z}\right) \\ (\boldsymbol{\tau}\cdot\boldsymbol{u})_y + \kappa\partial_y T \end{pmatrix}, \tag{3.12}$$

$$\boldsymbol{h}(\boldsymbol{v_\mu}) = \begin{pmatrix} 0 \\ \mu\left(\frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x}\right) \\ \mu\left(\frac{\partial u_y}{\partial z} + \frac{\partial u_z}{\partial y}\right) \\ (2\mu+\lambda)\frac{\partial u_z}{\partial z} + \lambda\left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y}\right) \\ (\boldsymbol{\tau}\cdot\boldsymbol{u})_z + \kappa\partial_z T \end{pmatrix}$$

As previously introduced, we collect all unknowns ($\rho, \boldsymbol{m}_u$ and E) in the vector $\boldsymbol{v}$ (cf. Eq. (3.2)). For the discretization we then rewrite Eq. (3.12) as

$$\begin{aligned} \partial_t \boldsymbol{v} = \sum_{k=1}^{d} \partial_{x_k} (&\nu_{k,1}(\rho, \boldsymbol{u}, E)\cdot\partial_{x_1}\boldsymbol{v} \\ &+ \nu_{k,2}(\rho, \boldsymbol{u}, E)\cdot\partial_{x_2}\boldsymbol{v} + \nu_{k,3}(\rho, \boldsymbol{u}, E)\cdot\partial_{x_3}\boldsymbol{v}) \end{aligned} \tag{3.13}$$

with d being the dimension of the problem and $\nu_{k,l}$ the viscosity tensor, more details can be found in full length in [98]. Rewriting Eq. (3.13) we obtain

$$\partial_t \boldsymbol{v} = \nabla\cdot(\nu(\rho, \boldsymbol{u}, E)\cdot\nabla\boldsymbol{v}) = \sum_{k=1}^{d} \partial_{x_k}\left(\sum_{l=1}^{d} \nu_{k.l}(\rho, \boldsymbol{u}, E)\cdot\partial_{k_l}\boldsymbol{v}\right). \tag{3.14}$$

For further simplification and to obtain a first order equation system, we split Eq. (3.14) into

$$\begin{aligned} \partial_t \boldsymbol{v} &= \nabla \cdot \sigma \text{ with} \\ \sigma &= \nu(\rho, \boldsymbol{u}, E) \cdot \nabla \boldsymbol{v}, \end{aligned} \tag{3.15}$$

where $\boldsymbol{v} = (\rho, \boldsymbol{u}, E) \ : \mathbb{R}^d \to \mathbb{R}^{d+2}$ resulting in five first order equations in three dimensional space to be solved, while the tensor σ is of second rank $\sigma \ : \mathbb{R}^d \to \mathbb{R}^{(d+2)\times d}$. To re-obtain the variational formulation for the viscous parts in the momentum and energy equations, we again multiply those equations by test functions ψ_h for a single element Ω_j. By summation over all elements we obtain the following formulation

$$\sum_{j=1}^{m} \partial t \int_{\Omega_j} \psi_h \boldsymbol{v} d\Omega = \sum_{j=1}^{m} \int_{\Omega_j} (\nabla \cdot \sigma) \psi_h d\Omega \tag{3.16a}$$

$$\sum_{j=1}^{m} \int_{\Omega_j} \sigma : \psi_h d\Omega = \sum_{j=1}^{m} \int_{\Omega_j} \nu(\rho, \boldsymbol{u}, E) \nabla \boldsymbol{v} : \psi_h d\Omega. \tag{3.16b}$$

Considering integration by parts for the rhs of Eq. (3.16) we end up with

$$\sum_{j=1}^{m} \partial t \int_{\Omega_j} \psi_h \boldsymbol{v} d\Omega = \sum_{j=1}^{m} \oint_{\partial\Omega_j} (\sigma \cdot \boldsymbol{n}) \psi_h dS - \int_{\Omega_j} \sigma : \nabla \psi_h d\Omega \tag{3.17a}$$

$$\begin{aligned} \sum_{j=1}^{m} \int_{\Omega_j} \sigma : \psi_h d\Omega = \sum_{j=1}^{m} \int_{\Omega_j} \boldsymbol{v} \cdot (\nu^T(\rho, \boldsymbol{u}, E) \psi_h \boldsymbol{n}) d\Omega \\ - \int_{\Omega_j} \boldsymbol{v} \nabla (\nu^T(\rho, \boldsymbol{u}, E) \psi_h) d\Omega. \end{aligned} \tag{3.17b}$$

Introducing numerical fluxes $\boldsymbol{u}^*$ and $\boldsymbol{\sigma}^*$ on the element surface and applying again integration by parts, we obtain Eq. (3.18).

$$\sum_{j=1}^{m} \partial t \int_{\Omega_j} \psi_h \boldsymbol{v} d\Omega = \sum_{j=1}^{m} \oint_{\partial\Omega_j} (\sigma^* \cdot \boldsymbol{n}) \psi_h dS - \int_{\Omega_j} \sigma : \nabla \psi_h d\Omega \tag{3.18a}$$

$$\begin{aligned} \sum_{j=1}^{m} \int_{\Omega_j} \sigma : \psi_h d\Omega = \sum_{j=1}^{m} \int_{\Omega_j} (\boldsymbol{v^*} - \boldsymbol{v}) \cdot (\nu^T(\rho, \boldsymbol{u}, E) \psi_h \boldsymbol{n}) d\Omega \\ + \int_{\Omega_j} \boldsymbol{v} \nabla (\nu^T(\rho, \boldsymbol{u}, E) : \psi_h) d\Omega \end{aligned} \tag{3.18b}$$

Defining $\psi_h = \nabla\psi_h$ [46] allows to couple the equations in Eq. (3.18), resulting in

$$\begin{aligned}\sum_{j=1}^{m} \partial t \int_{\Omega_j} \psi_h \boldsymbol{v} d\Omega = \sum_{j=1}^{m} &- \int_{\Omega_j} (\nu(\rho, \boldsymbol{u}, E)\nabla\boldsymbol{v}) : \psi_h d\Omega \\ &+ \oint_{\partial\Omega_j} (\sigma^* \cdot \boldsymbol{n})\psi_h dS \\ &- \oint_{\partial\Omega_j} (\boldsymbol{v^*} - \boldsymbol{v}) \cdot (\nu^T(\rho, \boldsymbol{u}, E)\psi_h \boldsymbol{n}) dS.\end{aligned} \tag{3.19}$$

With $\boldsymbol{u}^*(\boldsymbol{u}^-, \boldsymbol{u}^+)$ and $\sigma^*(\boldsymbol{u}^-, \nabla\boldsymbol{u}^-, \boldsymbol{u}^+, \nabla\boldsymbol{u}^+)$, where $-$ and $+$ indicate again the interior and exterior traces at the surface of the element, respectively. As in Section 3.1.1 different numerical flux approaches may be applied. The interested reader is referred to e.g. [46]. We can now bring together the convective and diffusive part of the discretized Navier-Stokes equations resulting in

$$\begin{aligned}\sum_{j=1}^{m} \partial t \int_{\Omega_j} \psi_h \boldsymbol{v} d\Omega = \sum_{j=1}^{m} &- \int_{\Omega_j} (\nu(\rho, \boldsymbol{u}, E\nabla\boldsymbol{v}) : \psi_h d\Omega \\ &+ \oint_{\partial\Omega_j} (\sigma^* \cdot \boldsymbol{n})\psi_h dS \\ &- \oint_{\partial\Omega_j} (\boldsymbol{v^*} - \boldsymbol{v}) \cdot (\nu^T(\rho, \boldsymbol{u}, E)\psi_h \boldsymbol{n}) dS \\ &+ \int_{\Omega_j} \nabla\psi_h \cdot \boldsymbol{F}(\boldsymbol{v}) d\Omega - \oint_{\partial\Omega_j} \psi_h \boldsymbol{F}^* \cdot \boldsymbol{n} dS.\end{aligned} \tag{3.20}$$

For a more detailed overview on the discretization of the viscous terms, the reader is referred to e.g. [98]. We simplify our afore shown equations, by rewriting them again in matrix-vector notation

$$\partial_t \hat{\boldsymbol{v}}(t) = \boldsymbol{M}^{-1} \cdot rhs(\hat{\boldsymbol{v}}(t), t), \tag{3.21}$$

where the right hand side $\boldsymbol{rhs}$ is defined as

$$\boldsymbol{rhs}(\hat{\boldsymbol{v}}(t), t) = \left(\boldsymbol{S} \cdot \hat{\boldsymbol{F}}(\hat{\boldsymbol{v}}(t)) - \boldsymbol{M}^F \cdot \hat{\boldsymbol{F}}^*(\hat{\boldsymbol{v}}(t))\right). \tag{3.22}$$

The computational effort to perform the operations in Eq. (3.21) is dependent on the choice of basis functions. In our studies, we consider

the Legendre polynomials, where we can use the inherent property of orthogonality of the Legendre polynomials for the mass matrix, resulting in a diagonalized mass matrix that can be trivially inverted. Hence, the multiplication with M^{-1} can be calculated in $\mathcal{O}(m+1)$ operations. The stiffness matrix is, however, occupied and therefore not trivially computed. But, due to the recursive property of the Legendre polynomials, the calculation can be performed in $\mathcal{O}(m+1)$ operations as well.

We now move on and focus on the discretization in time to fully discretize the conservation equations.

3.2. Explicit time discretization - Runge-Kutta method

This section deals with the discretization in time to complete the previous section, which was devoted to the discretization in space using the high-order Discontinuous Galerkin method. Generally, different kinds of ordinary differential equation solvers can be applied to Eq. (3.21). In this work, the explicit Runge-Kutta method (RK) is used since this method is well studied in theory as well as in combination with the Discontinuous Galerkin method (e.g. [74], [24] and [25]). The classical fourth-order Runge-Kutta method advances the system from a time t to $t+\Delta t$ through four substages in time, shown in Eq. (3.23). Each substage involves the evaluation of spatial discretization. One might assume that with increasing substages, higher accuracy in time can be achieved, with the order of accuracy being equivalent to the number of substages. However, this is only true to a certain extent, known as the Butcher barrier [20]. The Butcher barrier states that the number of substages grows faster than the convergence rate after more than four substages. Thus for a higher number of substages, there exists no explicit Runge-Kutta method that allows for a convergence rate as high as the number of substages. We, therefore, restrict our solver to the classical Runge-Kutta method with four substages leading to a fourth-order convergence in time

$$\hat{\boldsymbol{v}}^1 = \boldsymbol{M}^{-1} \cdot \boldsymbol{rhs}(\hat{\boldsymbol{v}}(t), t), \qquad \hat{\boldsymbol{v}}^a = \hat{\boldsymbol{v}}(t) + \frac{\Delta}{2}\hat{\boldsymbol{v}}^1 \tag{3.23a}$$

$$\hat{\boldsymbol{v}}^2 = \boldsymbol{M}^{-1} \cdot \boldsymbol{rhs}(\hat{\boldsymbol{v}}^a, t + \frac{\Delta t}{2}), \qquad \hat{\boldsymbol{v}}^b = \hat{\boldsymbol{v}}(t) + \frac{\Delta t}{2}\hat{\boldsymbol{v}}^2 \tag{3.23b}$$

$$\hat{\boldsymbol{v}}^3 = \boldsymbol{M}^{-1} \cdot \boldsymbol{rhs}(\hat{\boldsymbol{v}}^b, t + \frac{\Delta t}{2}), \qquad \hat{\boldsymbol{v}}^c = \hat{\boldsymbol{v}}(t) + \Delta t\hat{\boldsymbol{v}}^3 \tag{3.23c}$$

$$\hat{\boldsymbol{v}}^4 = \boldsymbol{M}^{-1} \cdot \boldsymbol{rhs}(\hat{\boldsymbol{v}}^c, t + \Delta t) \tag{3.23d}$$

$$\hat{\boldsymbol{v}}(t+\Delta t) = \hat{\boldsymbol{v}}(t) + \frac{\Delta t}{6}(\hat{\boldsymbol{v}}^1 + 2(\hat{\boldsymbol{v}}^2 + \hat{\boldsymbol{v}}^3) + \hat{\boldsymbol{v}}^4). \tag{3.23e}$$

Eq. (3.23e) provides the final stage, where the intermediate substages are weighted accordingly to advance the time step. To ensure numerical stability of the explicit Runge-Kutta Discontinuous Galerkin method, the Courant-Friedrichs-Lewy constant (CFL) is deployed. For parabolic problems, it can be computed according to

$$\zeta \frac{O^4 \cdot \Delta t}{h^2} < \mathrm{CFL}_p, \qquad \mathrm{CFL}_p \sim \frac{O^4}{h^2}. \tag{3.24}$$

ζ denotes the diffusion constant, $O = p + 1$ the scheme order and h the element size of the mesh. For hyperbolic problems the CFL is defined as

$$\alpha \frac{O^2 \cdot \Delta t}{h} < \mathrm{CFL}_h, \qquad \mathrm{CFL}_h \sim \frac{O^2}{h}, \tag{3.25}$$

where α represents the upper bound of the speed of the wave propagation. Eq. (3.24) and Eq. (3.25) demonstrate very well the influence of the scheme order on the CFL condition. For hyperbolic problems, the time step size reduces quadratically with increasing scheme order. However, for parabolic equations, the time step size is influenced by the scheme order with the power of four.

3.3. Numerical oscillations - Gibbs oscillation

High-order accuracy requires the smoothness of the solution, and therefore high-order discretization is mainly applied to problems where the smoothness of the solution can be ensured. However, in the case high-order is used for problems with discontinuities solutions, the Gibbs phenomenon occurs [22]. The Gibbs phenomenon destroys the pointwise convergence of the solution. Nevertheless, global convergence is still possible.

The Gibbs phenomenon occurs when the Fourier series of a piecewise continuous differentiable periodic function encounters a jump, a discontinuity. The xth partial sum of the Fourier series reacts with an oscillatory behavior in the vicinity of the jump. This can cause an increase in the maximum of the partial sum when compared to the original function. The maximum is known as an overshoot, which does not vanish when adding more terms to the Fourier sum, but will eventually reach a finite limit [22].

The phenomenon that appears in any smooth function series, not only

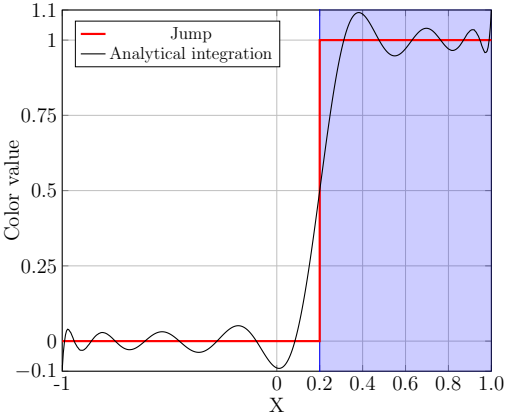


Figure 3.1.: Analytical integration of a discontinuity (red line and blue area). Oscillations are visible around the discontinuity, with over- and undershoots, representing the Gibbs oscillations around a discontinuity.

in Fourier expansions, is illustrated in Figure 3.1 for a Legendre series, approximating a step function. The discontinuity is located at $x = 0.2$ and highlighted by a red line. The blue area marks the part of the domain where the step function is not zero. We compute the expansion coefficients of the Legendre series for this special case by analytical integration. High-frequency oscillations appear with dominant peak deviations from the approximated step function in the vicinity of the discontinuity. We discuss the influence and properties of numerical integration, required for arbitrary functions in Section 5.3.

Many methods have been investigated and introduced to address this occurrence. One method is the h/p adaptivity, where the resolution is increased through a finer mesh (h adaptivity), while a lower scheme order is used (p adaptivity) near strong discontinuities [29]. Another approach is to introduce additional dissipation in the vicinity of the discontinuity to smear and achieve a smoother solution [71]. However, the mentioned methods reduce the accuracy of the solution to first order close to the discontinuity [98].

Let us move one step back and focus on identifying the origin of this

phenomenon. According to Gottlieb et al. [42] the Gibbs phenomenon exists, where point values need to be recovered by a function from its respective expansion coefficients. To allow a better understanding of this issue, we consider the Fourier series. In order to achieve a high-order accuracy, we need to have a sufficiently smooth function $f(x)$ to be approximated. This is an optimal condition for high-order polynomial interpolation. Assuming a problem is given with $2N+1$ Fourier coefficients f_{fk} of a function $f(x)$, with fk being $-N \leq fk \leq N$. Considering $f(x)$ is defined for $-1 \leq x \leq 1$, the classical Fourier sum can be constructed as [42]

$$f_N(x) = \sum_{k=-N}^{N} f_{fk} e^{ik\pi x}. \tag{3.26}$$

This is a simple way to reconstruct the function $f(x)$, when $f(x)$ is smooth as well as periodic. In the case $f(x)$ is analytic and periodic, it is well-known that the Fourier series converges exponentially with [42]

$$\max_{-1\leq x\leq 1} |f(x) - f_N(x)| \leq e^{qN}, \qquad \text{with} \quad q > 0. \tag{3.27}$$

In the case that $f(x)$ is not smooth or periodic, then $f_N(x)$ is not a reasonable approximation for the function $f(x)$ as it results in a poor convergence rate. Two features may be noticed here

1. There exists an overshoot close to the discontinuity, that does not vanish or decrease with increasing N, hence

   $$\max_{-1\leq x\leq 1} |f(x) - f_N(x)| \sim O(1)$$

   does not tend to zero.

2. The convergence rate becomes slow in the vicinity of the discontinuity x_0

   $$|f(x_0) - f_N(x_0)| \sim O\left(\tfrac{1}{N}\right)$$

This phenomenon is known as the Gibbs phenomenon, where oscillations appear around the point x_0. The Gibbs phenomenon gives the impression to disable the high-order pointwise convergence for discontinuous functions.

Intensive research has been dedicated to ENO (Essentially Non-Oscillatory), or the WENO (Weighted Essentially Non-Oscillatory) techniques [85]. These approaches aim to recover a non-oscillatory solution through weighted

interpolation or the modification of the state close to the non-smooth solution. The mentioned methods work as expected and increase the stability of the numerical simulation. But, unfortunately, they tend to reduce the accuracy of the solution to first order in the vicinity of the discontinuity [98]. At first glance, it seems impossible to obtain highly accurate local point values from the global modal coefficients for piecewise smooth functions. However, it can be shown that the high-order information is available in the modal coefficients, which can be considered during post-processing. Special techniques can be deployed to recover the exponential convergence rate from the modal coefficients. The argumentation is that the moments of the solution are maintained in the high-order information [42], even though a thoroughly investigated proof for most numerical methods is not available. The high-order information needs to be extracted through appropriate post-processing techniques [98].

The method used in this work to deal with the Gibbs phenomenon is filtering the modal expansion, where an appropriate modal filter in the simulation framework is used. As a result, the modal expansion is replaced by a filtered modal expansion, where either a polynomial or an exponential filter is used to reduce high oscillations. To ensure that strong oscillations around the point of discontinuity are damped, the filter needs to be chosen strong enough. This technique is also known as spectral viscosity filtering. Further, it needs to be mentioned that modal filtering allows a relatively inexpensive computational method when the solution of the simulation is available as expansion coefficients of a Legendre or Chebyshev series. Hence this filtering method addresses high oscillations throughout the simulation and post-processing to obtain a high-order and stable method.

An additional filtering method that is applied in this work is the so-called co-volume filtering. It was initially meant to increase the explicit time step size in the high-order Discontinuous Galerkin method and was introduced by Warburton et al. [95]. However, this technique can further be deployed to remove numerical oscillations from the solution, e.g., at the boundaries of the elements, as the spectral filtering loses its effect there. Zudrop [98] additionally applied the spectral viscous filtering to achieve improved filtering results for the numerical solution. This filter is utilized on the original element and a virtual element, which is shifted by half of the element size. The objective of the co-volume filtering is to diminish the high gradients on the co-volume mesh elements. Afterward, the gradients of the solution on the co-volume mesh are evaluated and projected back to the original mesh. Through the additional spectral filtering on the left and

right virtual element, a better filtering result can be achieved [98]. These filtering methods are used, to stabilize the numerical simulation and keep high numerical oscillations in a reasonable frame. Further information regarding these filtering techniques can be found in [98].

Conclusion In this chapter, the high-order Discontinuous Galerkin method is introduced, used to discretize hyperbolic and parabolic equations. After a short introduction into the discretization of the inviscid Euler equations, the discretization of the diffusive terms of the Navier-Stokes equations were presented. Afterward, the discretization method in time was introduced, where the fourth-order Runge-Kutta method was considered. Using four substages results in a fourth-order accuracy in time. Furthermore, as the explicit Runge-Kutta method is deployed, the CFL condition is used as stability criteria. Finally, we recalled the Gibbs phenomenon, numerical oscillations that have no physical nature and require to be treated appropriately, e.g., through special filtering techniques.

In the following chapter, the numerical framework to realize the numerical simulations in this work is introduced.

4. Numerical Framework

This chapter introduces and further develops the numerical framework to realize the simulations conducted in this work. The in-house *APES* (**A**daptable **P**oly-**E**ngineering **S**imulator) framework is end-to-end parallelized and tailored for large-scale simulations on today's supercomputing systems. Details on the simulation framework *APES* can be found in [81] and [54], an overview is given in Section 4.1. Afterward, the high-order Discontinuous Galerkin solver *Ateles* is presented in Section 4.1.1, used for the realization of the flow simulation in this work. Further, in Section 4.1.2 the integrated coupling approach in the simulation framework is revisited, enabling efficient computation of multi-scale problems. Additionally, the black-box library *preCICE* is briefly introduced in Section 4.2, that is later used to compare to the white-box coupling approach *APESmate* (cf. Section 4.1.2), where advantages and disadvantages are highlighted (see Section 7.3.3).

The main contribution of this work is the motion of complex geometries (cf. Chapter 5) in the solver *Ateles* and the simulation of flow-induced noise, where the motion of a rigid body disturbs the flow field and noise is generated and transported to the far-field. Further, load balancing challenges and the method deployed to treat load imbalances, especially in the presence of geometries, are addressed.

4.1. *APES* simulation framework

The simulation framework *APES* is designed for parallel execution on today's supercomputing systems. It is well suited for large-scale simulations in different areas of research, mainly for fluid dynamic problems. Such highly parallel and scalable frameworks are crucial to address computationally demanding large-scale simulations such as the direct numerical simulation of aeroacoustics. With the upcoming era of exascale computing, intensive research has been devoted to the development of software packages that are capable of efficient execution on a large number of CPUs.

The *APES* framework is massively parallel and provides scalable solvers

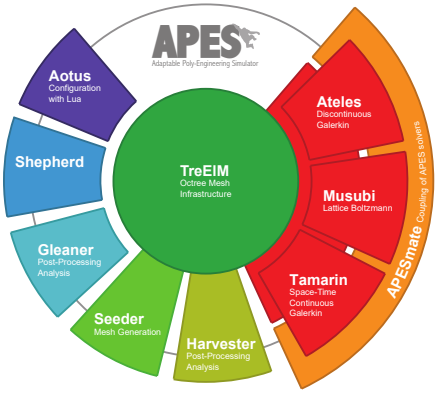


Figure 4.1.: Massively parallel simulation framework *APES*, suited for large-scale simulations.

and pre- and post-processing algorithms. An overview of this powerful framework is given in Fig. 4.1. All parts in *APES* share the common data structure called TreElM (**Tre**e based **El**emental **M**esh) [55]. The domain decomposition for parallel execution relies on a space-filling curve, allowing to maintain data locality. All solvers in *APES* are stencil-based and require neighbor information for the computation. Due to the tree-based data structure of the computational mesh and the locality of the data, the communication during the simulation can be reduced, which is an essential factor for efficient parallel execution. This property is, in particular, beneficial for the high-order Discontinuous Galerkin solver *Ateles* that relies on direct neighbor information. The computational mesh is generated with the mesh generator *Seeder* [68].

An important feature of *TreElM*, that is relevant for this work, is the load balancing algorithm *SPartA* [79] [44]. The *SPartA* algorithm is as *TreElM*, based on space-filling curves and individual weights, that provide the actual load per element. The computation of such element-wise weights allows an a priori re-partitioning of the computational mesh according to the actual workload. In order to enable efficient computation and reduce the computational effort for our simulations in this work, we extended the weight measurements for the computation of geometries and multilevel meshes in the solver *Ateles* [36]. This practice enables to capture the computational effort precisely and provides a simulation framework that

allows for a better load balancing on massively parallel systems.

Harvester is the in-house post-processing tool that converts binary files written during the simulation by the solver to a format that can be visualized. For the visualization, a variety of possibilities are provided by this tool, such as different canonical shapes, to measure specific variables at a specific location of the simulation domain or the transformation of the binary files to vtks used to visualize parts or the entire simulation domain at a specific simulation time. It can be executed in serial and parallel, allowing the visualization of different data sizes. The post-processing tool is closely related to the solver used, as it includes solver-specific information, such as the polynomial representation for the solver *Ateles*. A highlight of *Harvester*, especially for the solver *Ateles*, is the adaptive mesh refinement based on the polynomial solution. Regions in the simulation domain are better resolved with higher quality, according to the variational solution of the polynomials.

4.1.1. High-order Discontinuous Galerkin solver - *Ateles*

Ateles is based on the high-order modal Discontinuous Galerkin method (cf. Section 3.1). The solver is based on polynomial representation inside the elements and flux computation between the element faces. This results in a strong dependency of the data within the elements and a relatively loose dependency between them. The discretization in time is according to the explicit Runge-Kutta method (cf. Section 3.2). In the presence of geometries in the flow simulation, the implicit-mixed-explicit Runge-Kutta method is deployed (see Section 5.7). Thus the stability of the Runge-Kutta method is connected to the CFL condition, as previously mentioned (cf. Eq. (3.24) and Eq. (3.25)).

The choice of a high-order numerical scheme has several benefits when compared to low-order schemes. These are:

- Assuming a smooth solution and a high-order scheme, this yields a high-order convergence rate. Consequently, a high-order approximation results in high accuracy of the solution with fewer degrees of freedom (nDoF). Less nDoF has the same meaning as a smaller amount of memory, which is favorable when considering today's supercomputing systems.
- It is well known that the high-order scheme provides a low dissipation and dispersion error [47], which is beneficial when considering the

propagation of acoustic waves over a long distance (acoustic far-field).

Legendre polynomials are used as basis functions for the modal high-order Discontinuous Galerkin discretization. Due to their inherent orthogonality properties and recursive definition, they can be fastly evaluated, as in the mass and stiffness matrix. Another advantage of the Legendre polynomials are the numerical flux computation for linear problems that can be realized quadrature free, thus in polynomial function space (modal).

Apart from linear problems, nonlinear problems are the main target of numerical simulations, with operations, that require a representation in nodal space, as a computation in modal space might not be possible or more involved in realization. The transformation of the modal information is required for, e.g., nonlinear flux computation or geometry representation. Nodal information is then needed in each time step to update the fluxes or, in the case of the moving geometry, to identify the location of the geometry (cf. Section 5.3). In these cases, a modal to nodal transformation is necessary. Different methods may be applied for the transformation, e.g., the fast polynomial transformation (FPT) [5] or the direct projection using numerical quadrature, namely L2 projection. More information about these methods can be found in [7, 98].

Further, the solver *Ateles* allows for geometries and their motion in the simulation domain. Geometrical constraints are embedded in the fluid dynamic equations to be solved by utilizing the same discretization method as the flow field. In this work, the motion of geometries in the simulation domain has been enabled. More about the used technique can be found in Chapter 5. Additionally, a different set of equations are implemented in the solver: the Maxwell equations (electrodynamics), the compressible Navier-Stokes equations, and the inviscid Euler equations (fluid motion). More information about the solver can be found in [54, 59, 98].

4.1.2. Integrated coupling approach - *APESmate*

The coupling approach *APESmate* [60, 68] is fully integrated in the simulation framework *APES*. It allows the coupling of different solvers in the framework. The coupling tool supports both surface, and volume coupling [68]. In surface coupling, data is exchanged at arbitrary exchange points at the boundaries (coupling interface), the so-called coupling points. For the high-order Discontinuous Galerkin solver *Ateles*, the coupling points are the same as the integration points that are non-equidistantly distributed

inside an element (cf. Section 5.3). During the initialization process, each domain provides information of point coordinates, where values are expected to be exchanged. Those point coordinates are directly evaluated by the data-providing domains using the polynomial representation in the solver *Ateles*. Through a global communicator, data is exchanged at dedicated coupling points at each synchronization time step. The so-called synchronization time step is reached when all domains involved in the coupling have completed a predefined time step, that is, in the current implementation, a fixed one for all coupling domains. Thus, *APESmate* can be deployed for a wide range of applications such as, e.g., the fluid and acoustics coupling or multi-component flow and the electro-dynamic field coupling [68]. Even though different domains and solvers are coupled, this coupling approach has only one executable.

4.2. External coupling approach - *preCICE*

The black-box coupling approach *preCICE* is an open library that enables the coupling of arbitrary solvers with each other [19, 91]. The idea behind the black-box approach is to use input and output data of involved solvers at the coupling interfaces between the respective coupling domains. Numerical solvers exchange data at arbitrary coupling points via the coupling approach. Those point positions are, in the first instance, the only information provided to *preCICE*. Thus, *preCICE* has no information about the numerical discretization method of each solver. Though only point positions are required for the data exchange, the coupling approach needs to interpolate the data before providing them to the respective coupling domain. Therefore, different interpolation methods are available that can be used according to the needs of each solver with respect to the accuracy of the solution. More about the interpolation methods and the accuracy of the solution for the high-order solver *Ateles* can be found in Section 7.3. For the data mapping, this approach provides conservative and consistent mapping, depending on the value to be exchanged, e.g., displacement or pressure. To allow the connection between solvers and *preCICE* an adapter, e.g., Fortran or C++ interface, is required. Furthermore, this coupling approach enables implicit and explicit coupling, including Quasi-Newton waveform iteration [82] for implicit time stepping. Information about recent improvements of this approach can be found in [65, 78]. In addition, to address load imbalances, this coupling tool provides a data-based load balancing method based on linear regression. It can be used to balance the workload between the coupling domains [89].

5. Embedded boundary method

The high-order Discontinuous Galerkin method has, as mentioned previously, outstanding advantages and is well suited for highly accurate numerical simulations on high-performing computing (HPC) systems. This method is often used to simulate aeroacoustic problems due to its low dissipation and dispersion error. Thus, information can travel over large distances and still be captured with high accuracy.

In many engineering applications, simulations are complex and involve geometries, hence boundaries that need to be modeled appropriately. In high-order methods, the boundary requirements are even tighter, as a high-order representation is required to maintain the benefit of the high-order accuracy. Even though the Discontinuous Galerkin method is also applied to unstructured meshes, the high-order representation at the boundaries is often challenging. There exist some techniques for the discontinuous method based on, e.g., isoparametric elements [46], or the matched derivative technique [30].

All these methods show significant limitations, in particular for complex geometries. The challenge we encounter here is, when using a high-order method, it is preferred to exploit the benefits of this method by using relatively large elements for the computational mesh. However, when geometries are involved in the simulation domain, we need to use small elements close to the geometry to capture the flow features in those regions sufficiently well or utilize super-parametric elements to have curved physical elements. Combining large elements for the mesh and considering curved boundaries is complicated and involves mapping strategies and different kinds of corner cases. The main limitation that appears is linked to the geometry that needs to be represented. At some point, it becomes impossible to find a super-parameterization for a sufficiently complex surface. Lastly, this might even result in cases where the generation of a mesh might not be possible.

This section deals with a different approach. Instead of changing the computational mesh, the geometry is embedded in the simulation domain,

i.e., represented as an additional function on the computational mesh. A significant benefit is that a Cartesian mesh can be deployed without changing the shape of the elements close to the geometry while maintaining the algorithm as simple as possible. The embedded method is well suited for different equations but needs to be adjusted to the needs of the equation system. In this work, this approach is utilized for the compressible Navier-Stokes equations (cf. Eq. (2.1)) as well as for the compressible inviscid Euler equations (Eq. (2.11)). The idea behind this approach is to incorporate additional terms (penalization terms) wherever the geometry is embedded [73]. The numerical solution is enforced to a specific solution and the fluid dynamic equations are further solved inside the geometry. Consequentially, fluid flow is also present inside the geometry. The penalization terms add source terms to the conservation law that are discontinuous in space. Therefore the numerical solution might be affected by the Gibbs phenomenon (cf. Section 3.3). However, as previously mentioned in Section 3.3 those oscillations can be addressed during post-processing, allowing for highly accurate simulation results.

This method is presented in more detail in the next section, with further background information about this approach. Further, the penalized compressible Navier-Stokes equations are presented.

5.1. State of the art - Geometry representation

If we consider the geometry representation more generally, we can categorize the representation into two methods based on the mesh. The first one is the well-known body-fitted mesh method [87], where the geometry is included in the mesh, and the mesh is fitted towards the geometry. With that, the boundary-layer near curved surfaces can be resolved accordingly. This method is well suited when geometries do not move and are located at the exact location throughout the simulation time. However, if geometries can move or even deform, this method tends to become very costly. The mesh needs to be adjusted to the motion of the geometry. Further, the state variables might need to be interpolated from one fitted mesh to another in each iteration. To overcome the procedure of re-meshing, the second method, the embedded method [69] can be considered, which is widely used to model complex geometries. The significant benefit of this method is that the geometry is not directly linked to the mesh and, therefore, not included in the mesh generation procedure. Hence it is independent of the mesh itself in the first place. Furthermore, a simple Cartesian mesh can

be deployed for the simulation. This method is relatively inexpensive in the generation and well suited for parallelization and high-performance computing.

In this work, we consider the embedded method as its advantages are crucial, referring to the Brinkman penalization method. This method models the geometry as an artificial porous material but tunes its parameters to obtain an essentially solid body. First investigations with this volume penalization method were carried out by Arquis et al. [32]. They imposed additional penalization terms to the momentum equations. The main idea behind this approach is to model a complex and arbitrary solid body by modeling the geometry as a porous material with a porosity ϕ and permeability terms approaching zero. The boundary conditions can be forced to a specific precision while maintaining the numerical method and the computational mesh. Further, the error estimation of the solution can be given in terms of the penalization terms [9]. Kevlahan et al. [52] applied this method for incompressible flows and ran simulations with non-moving and moving geometries. They considered a pseudo-spectral method for their work.

Liu and Vasilyev investigated the Brinkman penalization further and extended their investigation to the compressible Navier-Stokes equations. They used a wavelet method for the discretization of the equations and provided an extensive study in terms of dependencies of this method from the porosity, and permeability terms [67]. They chose for their studies the viscous permeability η to be defined as $\eta = \alpha\phi$ and the thermal permeability as $\eta_T = \alpha_T\phi$ with α and α_T being scaling factors. With the chosen relations, a modeling error of $\mathrm{O}(\eta^{1/2}\phi)$ for resolved boundary layers and $\mathrm{O}((\eta/\eta_T)^{1/4}\phi^{3/4})$ for unresolved boundary layers were obtained. It needs to be emphasized that since inside the porous material fluid flow is considered, a boundary layer is found along the geometry surface outside the geometry and inside, where the fluid is penalized to predefined values. As the state inside the material evolves, thus the penalization occurs; a boundary layer can be found inside the porous material. The resolved and unresolved boundary layer definitions by Liu et al. refer to the boundary layer inside the porous material. In their studies, they revealed that the porosity has a larger impact on the error of the solution. However, the error can be influenced by the choice of the porosity and the permeability. Hence the modeling error can be reduced by a sufficiently small permeability factor η, and η_T [67], too. In further investigations, Komatsu et al. [57] noticed that the equations provided by Liu et al. [67] were not Galilean

invariant. Hence they corrected the conservation equations with additional terms.

Keeping the outcomes of Liu et al. in mind, Anand, EP et al. [8] further investigated in this direction while using a high-order modal Discontinuous Galerkin method for the discretization of the fluid dynamic equations. They ascertain that a small value for the porosity leads to stability issues as it implies strong time-step restriction in their explicit time-stepping scheme. Furthermore, with imposing ϕ in the mass conservation equation, the eigenvalues of the hyperbolic system changed, affecting the stability detrimentally. Their investigations showed how this method could be used efficiently by tuning the permeability terms to a tiny factor. As shown in the equations for the error bounds provided above, the permeability and porosity can influence the error. They introduced a scaling factor β and revealed that with sufficiently small permeability terms, the porosity ϕ can be set to 1.0, eliminating this term in the mass equation. With an implicit-mixed-explicit (IMEX) time-stepping scheme (see Section 5.7), the source terms from the penalization can be efficiently computed implicitly. At the same time, the remaining part of the equation is solved in a straightforward explicit approach. With that, they propose to define the viscous permeability η to be $\beta^2 \cdot \phi^2$ and the thermal permeability η_T to $0.4\beta \cdot \phi$, with an expected modeling error of $\beta^{1/4} \cdot \phi^{3/4}$. With a scaling factor of $\beta \leq 10^{-6}$, they were able to achieve the best numerical results in terms of conservation of momentum and energy.

In the following the Brinkman penalization technique, for the compressible Navier-Stokes equations is introduced, while the outcomes of Anand, EP et al. [8] are taken into consideration, with $\phi = 1.0$ and $\beta \leq 10^{-6}$, since the same numerical scheme is used.

5.2. Volume penalization method - Brinkman penalization

As already mentioned, the Brinkman technique is a volume penalization method, where the geometry is embedded in the equation system to be solved. This modeling strategy aims to model the geometry as an artificial porous material with specific properties. The governing Navier-Stokes equations for the fluid and the penalized Navier-Stokes equations are simultaneously solved to model the porous material.

$$\partial_t \rho + \nabla \cdot \boldsymbol{m_u} = -\left[\frac{1}{\phi} - 1\right] \chi \nabla \cdot \boldsymbol{m_G} \quad (5.1a)$$

$$\partial_t \boldsymbol{m_u} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{m_u} + p\boldsymbol{I}) - \nabla\tau = p\boldsymbol{f} - \frac{\chi}{\eta} \cdot (\boldsymbol{u} - \boldsymbol{u_G}) \quad (5.1b)$$
$$- \boldsymbol{u}\left[\frac{1}{\phi} - 1\right]\chi\nabla \cdot \boldsymbol{m_G}$$
$$\partial_t E + \nabla \cdot (\boldsymbol{u}\,((E+p))) - \nabla \cdot (\tau u + \kappa\nabla T) = -\boldsymbol{m_u} \cdot f \quad (5.1c)$$
$$- \frac{\chi}{\eta_T}(T - T_G)$$
$$- \frac{\chi}{\eta}\,(\boldsymbol{u} - \boldsymbol{u_G}) \cdot \boldsymbol{u}$$
$$- \frac{|\boldsymbol{u}|^2}{2}\left[\frac{1}{\phi} - 1\right]\chi\nabla \cdot \boldsymbol{m_G}$$

An extension of the incompressible Brinkman penalization method is required to realize simulations of compressible viscous flows with geometries. As a result, the conservation laws, namely the mass, momentum, and energy equations, have to be penalized. Taking this into account, we obtain Eq. (5.1), where the penalization terms are added to the rhs of the equations and act as source terms. The term $\boldsymbol{m_G} = (\boldsymbol{u} - \boldsymbol{u_G})\rho$ represents the momentum, with the difference between the fluid velocity $\boldsymbol{u}$ and the velocity of the geometry $\boldsymbol{u_G}$. The variable T_G indicates the temperature of the geometry, which is considered to be constant (isothermal). The parameters η and η_T denote the viscous and thermal permeability of the geometry, respectively, while ϕ is the porosity of the geometry. The mask function χ is defined as

$$\chi(x,t) = \begin{cases} 1 & \text{if } x \in G_i \\ 0 & \text{otherwise.} \end{cases} \quad (5.2)$$

Here G_i denotes the area covered by the porous material, which can be a composition of more than one area. The function χ in Eq. (5.2) refers to whether the penalization terms have to be computed or can be neglected.

In Eq. (5.1), in all equations the porous material is considered. However, as previously mentioned, we consider ϕ to be 1.0, which leads to a more simplified mass, momentum, and energy equation, as a small permeability (viscous and thermal) term is used to keep the modeling error small as possible. By setting the porosity to 1.0, the mass, momentum, and energy equation can be rewritten as

$$\partial_t \rho + \nabla \cdot \boldsymbol{m_u} = 0 \quad (5.3a)$$

$$\partial_t \boldsymbol{m_u} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{m_u} + p\boldsymbol{I}) - \nabla\tau = p\boldsymbol{f} - \frac{\chi}{\eta} \cdot (\boldsymbol{u} - \boldsymbol{u_G}) \quad (5.3b)$$

$$\begin{aligned} \partial_t E + \nabla \cdot (\boldsymbol{u}\,((E+p))) - \nabla \cdot (\tau u + \kappa \nabla T) &= -\boldsymbol{m_u} \cdot f \\ &\quad - \frac{\chi}{\eta_T}(T - T_G) \\ &\quad - \frac{\chi}{\eta}\,(\boldsymbol{u} - \boldsymbol{u_G}) \cdot \boldsymbol{u}. \end{aligned} \quad (5.3c)$$

It is known as a monolithic approach, where the geometrical constraints are treated as source terms in the fluid dynamic equations and solved simultaneously. Forces by the flow field have no impact on the obstacles in this model, as shown in Eq. (5.3). Hence only the fluid is influenced by the artificial porous material but not vice versa. Further, it has to be noticed that conservation is not maintained within the obstacles due to the introduced source terms. However, as already investigated by Liu et al. and Anand, EP et al., the impact of conservation violations on the rest of the domain can be eradicated with sufficiently small boundary layers inside the porous material. Thus by defining the porosity or in our case the permeability to a small value. The modeling error is kept small enough to maintain conservation of mass, momentum, and energy. Through intensive investigations in Chapter 6 this will be again shown in this work, where we prove that obtained simulation results are in excellent agreement with known benchmarks from literature, indicating that conservation is maintained when exploring our modeling scheme.

Introducing this modeling method in our Discontinuous Galerkin scheme, we need to be aware that a discontinuity is introduced into the system to be solved due to the masking function. This might be challenging for the numerical scheme, as the masking function jumps from 0 to 1 throughout the simulation. Hence, the question arises, how the modal Discontinuous Galerkin scheme can represent the geometry, thus the discontinuity. Furthermore, the explicit time-stepping scheme is restricted to tiny time step sizes to allow numerical stability due to the additional source terms in the fluid dynamic equations. This is due to the permeability terms, which are defined to be very small. The numerical scheme ends with drastic dynamics from the source terms since they quickly damp the state inside the porous material and need to be resolved in the explicit time integration. Thus, the small permeabilities are only feasible when treated with an implicit time integration. Therefore, an implicit-mixed-explicit time-stepping scheme (IMEX) is utilized to allow larger time steps for the simulation (cf. Section 5.7).

In the next section, the geometry representation in our high-order Discontinuous Galerkin scheme will be introduced in more detail, where we explicitly concentrate on how the masking function χ is evaluated.

5.3. Representation of the masking function χ in high-order Discontinuous Galerkin

As mentioned in Section 5.1 the embedded method is well suited to represent a complex geometry in high-order, where we aim for large computational elements to reduce memory consumption for our simulations. In this section, the geometry modeling in high-order Discontinuous Galerkin is investigated in more detail when deploying the Brinkman penalization method (cf. Section 5.2). Furthermore, the numerical scheme has to handle additional discontinuities due to the masking function introduced by the modeling method. Finally, the solver is based on the modal high-order Discontinuous Galerkin scheme [98], hence all conservation quantities are stored as Legendre polynomials (cf. Chapter 3). Consequently, the representation of the geometry needs to be in the function space of the Discontinuous Galerkin solver, which are polynomials, primarily the Legendre polynomials, in our case. These polynomials have the advantages of building an orthogonal basis with a weight of one on the interval of [-1, 1]. Figure 5.1 provides an overview of the Legendre polynomials of up to a polynomial degree of 5. Further, the Legendre polynomials can be computed according to the three-term recurring relation as

$$L_j(x) = \frac{2j-1}{j} x \; L_{j-1}(x) - \frac{j-1}{j} L_{j-2}(x), \quad \text{with } j > 1. \tag{5.4}$$

With the zero-mode $L_0(x) = 1$, representing the integral mean of the polynomial (see Figure 5.1 blue constant line) and $L_1(x) = x$ being the first mode of the polynomial. The higher polynomials can be constructed by Eq. (5.4). Please keep in mind that the zero-mode is the only one that has a non-zero integral mean, while all higher modes are mean-free in the interval of the Legendre polynomials. The geometry introduces a discontinuity, a jump at its interface (see Eq. (5.2)), where the masking function χ changes from a value of 0 (outside the geometry) to 1 (inside the geometry). Hence, a step function needs to be projected to the polynomial

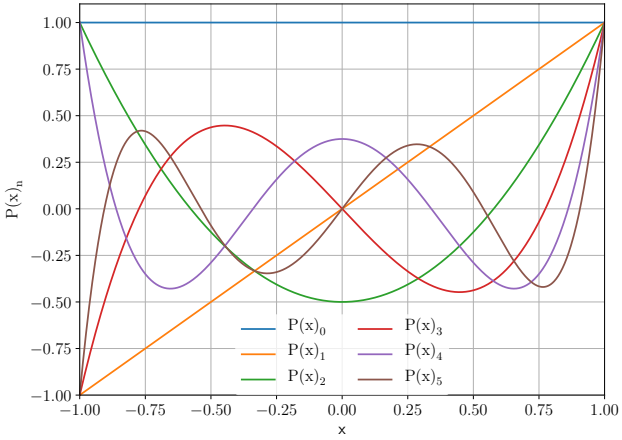


Figure 5.1.: Different curves represent the Legendre polynomials of up to a polynomial degree of 5, in the interval of [-1, 1].

space while a suitable expansion

$$P_n(x) = \sum_{j=0}^{n} c_j L_j(x) \tag{5.5}$$

has to be found, which approximates Eq. (5.2) appropriately. The expansion coefficients c_j have to be computed, in order to obtain $P_n(x)$. Therefore, the inherent property of the Legendre polynomials, of orthogonality to the inner product

$$\langle f, g \rangle = \int_{-1}^{1} f(x)g(x)dx \tag{5.6}$$

is considered. In our case the $f(x)$ is replaced by the masking function $\chi(x)$ and $g(x)$ by the Legendre polynomials $L_j(x)$, in order to obtain the

expansion coefficients c_j.

$$\begin{aligned} c_j = \langle \chi(x), L_j(x) \rangle &= \int_{-1}^{1} \chi(x) \cdot L_j(x) dx \\ &\approx \sum_{i=1}^{m} w_i \cdot \chi(x_i) \cdot L_j(x_i) \end{aligned} \tag{5.7}$$

Here m is the number of integration points and w_i a weight value for the numerical integration.

Considering the mentioned equations above, we can represent the masking function in the same function space as all conserved quantities, allowing for its appropriate approximation and representation with high-order polynomials. We now continue the investigation of its evaluation in the numerical scheme in the following section.

5.4. Evaluation of the masking function χ in the numerical scheme

In order to evaluate the masking function χ, thus the geometry, the workflow is defined to be as follows:

- **Identifying** the elements where the geometry can be located
- **Evaluating** the masking function χ at integration points (nodal data)
- **Converting** the nodal data of χ to modal (polynomial modes)

The solver defines geometries as a space-time function, allowing them to change location over time. To reduce the computational effort, we first **identify** computational elements that have geometry properties. More on the purpose of the identification procedure can be found in Section 5.5. The next step to be addressed here is the **evaluation** of the masking function χ at the integration points (Chebyshev nodes). The **evaluation** of the masking function χ is done in each time step, if the motion of the geometry is intended. As the evaluation can hardly be illustrated in more dimensions, we concentrate on visualizing the problem in a one-dimensional space.

In Figure 5.1a the target masking function χ (thick red line) and the geometry interface (blue line) are shown. The geometry has a specific

thickness, covering a certain area in the simulation domain (blue area). The orange line in the figure depicts the expansion series obtained with analytical integration, and the black line the numerical integration of the step function (masking function). The solution for the analytical solution can be again obtained through Eq. (5.5) and Eq. (5.7). In Eq. (5.7), the integral limits can be split according to the jump function, resulting in

$$\begin{aligned} c_j = \langle \chi(x), L_j(x) \rangle &= \int_{-1}^{1} \chi(x) \cdot L_j(x) dx \\ &= \int_{-1}^{Jump} \chi(x) \cdot L_j(x) dx \\ &+ \int_{Jump}^{1} \chi(x) \cdot L_j(x) dx, \end{aligned} \tag{5.8}$$

where the integral limit $Jump$ indicates the location of the discontinuity in the simulation domain. Generally, it is not as straightforward as in this simple one-dimensional test case to obtain the solution. Thus, numerical integration is required, where the target function χ has to be evaluated by a finite number of integration points, in our case the Chebyshev nodes. The quality of the solution (numerical integration) is strongly coupled with the number of Chebyshev nodes, which can be given as

$$x_{cn} = \cos\left(\frac{2 \cdot cn - 1}{2N} \cdot \pi\right) \qquad cn = 1, ..., N. \tag{5.9}$$

The number of nodes is even more significant for a moving geometry, where the geometry changes its location from one time step to the other. The minimal distance between the Chebyshev nodes is at the element boundary (in the interval of the Legendre polynomials). It is proportional to N^{-2}, resulting in smaller distances between the nodes with increasing polynomial degree. In the case the geometry moves, the most significant error from the point distribution is introduced by the largest distance between the Chebyshev nodes, which is in the middle of an element and proportional to N^{-1}.

We need to keep in mind that the solver is based on the modal Discontinuous Galerkin method. Thus the state of the simulation is always stored in polynomial function space. Therefore, nodal information (point data) always have to be **converted** to modal information (cf. numerical approximation in Figure 5.1). Since the Discontinuous Galerkin solver uses the Legendre modes c_j (cf. Eq. 5.5), that need to be obtained from the

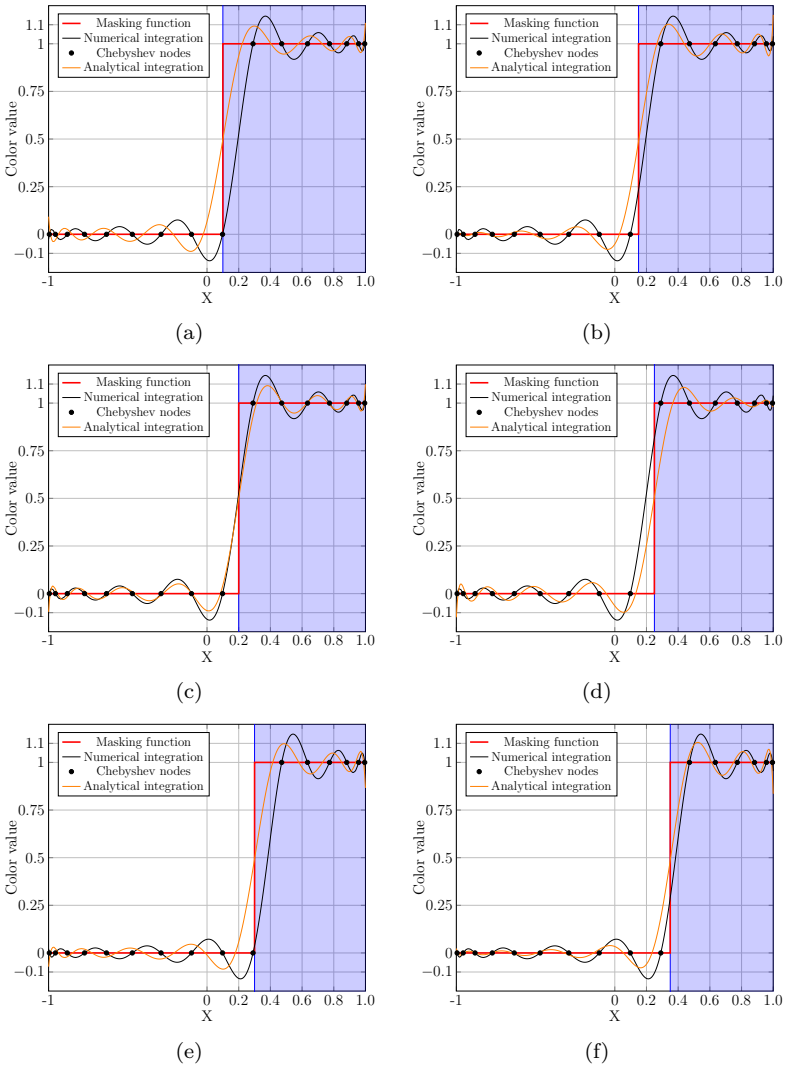


Figure 5.1.: Polynomial representation of the moving wall (geometry). Geometry interface moves and is therefore located at different locations. In (a) at $x = 0.1$, (b) at $x = 0.15$, (c) at $x = 0.20$, (d) at $x = 0.25$, (e) at $x = 0.30$ and in (f) at $x = 0.35$.

information of the Chebyshev nodes, a fast polynomial transformation is applied according to [6]. The **evaluation** of χ can be undertaken at each Chebyshev node x_{cn}, that are at the same time, as already mentioned, the integration points for the numerical approximation. In Figure 5.1 the Chebyshev nodes of $N = 16$ are indicted by black dotes, resulting in a polynomial degree of 15. It is of importance to mention here that the position of the geometry interface is of significant importance for its representation. If the jump is positioned between two neighboring nodes, a variation in this area does not result in a different numerical integration. This effect is an aliasing error, as insufficient integration points are used to represent the geometry appropriately. Further, this can cause instability in the numerical scheme as the higher order terms are misinterpreted. They are mirrored on the lower order terms, leading the Discontinuous Galerkin scheme to become unstable [46].

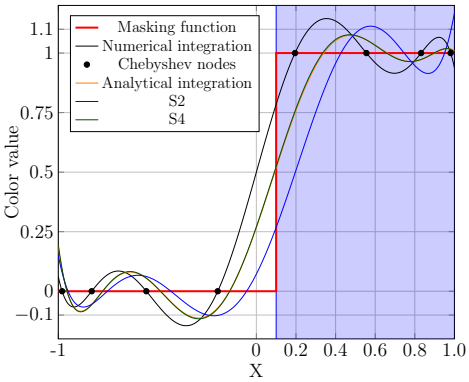


Figure 5.2.: Comparison: Analytical integration and numerical integration without and with over-integration to improve the numerical solution. S2 and S4 indicate the over-integration factor of 2 and 4, respectively.

We can observe this behavior between, e.g., Figure 5.1a and Figure 5.1b as the shifting of the geometry interface results in the same representation since it is still located between the same two neighboring Chebyshev nodes. The worst case of the geometry representation in the numerical scheme is depicted in Figure 5.1a and Figure 5.1e, where the geometry interface is exactly on the Chebyshev node, with a more significant distance to the

neighboring node. Hence resulting in the representation of the geometry with the largest error when compared to the analytic integration. The most desirable scenario for the geometry representation would be its location strictly between two integration points as in Figure 5.1c, where the discontinuity can be captured as precisely as the analytic integration. Thus, it is significant to know how the aliasing effect can be prevented

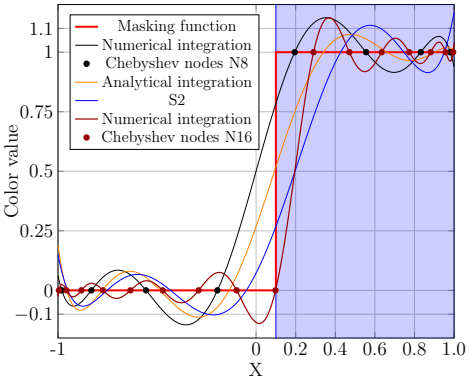


Figure 5.3.: Comparison: Analytical integration and numerical integration without and with over-integration to improve the numerical solution. The numerical integration without over-integration is represented by the black line and the respective Chebyshev nodes by the black dots. Numerical integration with over-integration of factor 2 (S2 curve), resulting in an approximation with 16 integration points. Lastly, the numerical integration for a polynomial degree of 15 without over-integration (rust-brown) is shown.

and the numerical approximation improved to preserve better numerical solutions, hence a better representation of the geometry interface. That is especially important when the geometry moves and is represented at a different location in the simulation domain. As shown in Figure 5.1 the motion of the geometry between two integration points always results in an identical representation. The advance of the jump out of this area results in a different representation of the geometry and a sudden change in the numerics. The geometry's continuous motion is only captured by the numerical scheme step-by-step when the geometry moves and can be represented between a "new pair" of Chebyshev nodes.

To address this drawback and improve the accuracy of the numerical integration, we consider more Chebyshev nodes (integration points) N, which is also known as over-integration [53]. It provides a smaller distance between the Chebyshev nodes and, therefore, a more precise representation of the geometry interface. In Figure 5.2 we exemplarily illustrate how over-integration can help to improve the numerical solution. With an over-integration factor of 2 and 4 indicated by S2 and S4. Over-integration can help to address this issue and control the error for the geometry representation. An over-integration factor of 2 (S2) is, as illustrated in Figure 5.2 not enough to improve the solution sufficiently, even though the amplitude of the Gibbs oscillations decrease. However, Figure 5.2 also demonstrates that an over-integration of 4 is suitable to approximate the solution and reduce the error sufficiently to achieve a solution that is in agreement with the analytical integration. An over-integration of 4 has the same meaning as using four times more integration points to represent the geometry for the numerical integration (cf. Figure 5.1). We obtain a new polynomial representation of the masking function through over-integration, representing the discontinuity with higher quality and avoiding aliasing error.

In Figure 5.3 different curves are depicted to exemplary provide an idea how the over-integration influences the solution. Again curves provided show the numerical approximation (black) with the respective Chebyshev nodes (N8), the analytical approximation (orange), and the numerical solution when considering an over-integration of two (S2). Further, the curve (rust-brown) and the respective Chebyshev nodes for a polynomial degree of 15 are considered to compare the difference between over-integration and the direct use of the respective polynomial to obtain an improved representation for the geometry. From Figure 5.3 it is apparent that through over-integration, the numerical integration is improved. However, the result does not provide the same representation as directly using a polynomial degree of 15 for the numerical integration. Thus, when considering over-integration, the higher modes represent the geometry with higher accuracy, but those higher modes are removed from the solution after numerical integration.

We now recall the previous test case from Figure 5.1 and rerun the same setup by considering an over-integration factor of 4 for the numerical integration. In all subfigures provided in Figure 5.4 we can observe improved solutions for the numerical integration when considering over-integration,

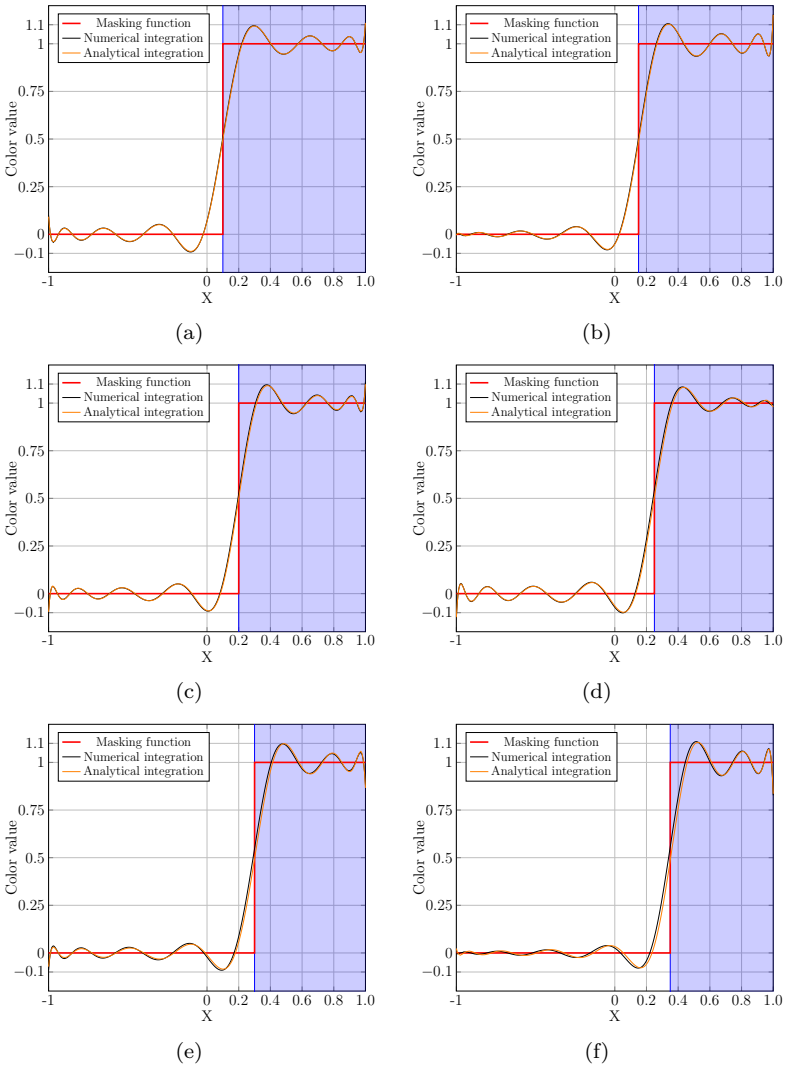


Figure 5.4.: Polynomial representation of the moving wall (geometry). The geometry interface moves and therefore is located at different locations, while considering an over-integration factor of 4. In (a) at $x = 0.1$, (b) at $x = 0.15$, (c) at $x = 0.20$, (d) at $x = 0.25$, (e) at $x = 0.30$ and in (f) at $x = 0.35$.

even in the case where χ is located at an unfavorable location, a significant improvement of the numerical integration can be achieved. Further, we need to recall that due to the introduced discontinuity by the masking function, the Gibbs phenomenon appears and needs to be taken into consideration. It is, as mentioned in Section 3.3 done throughout the post-processing procedure.

5.5. Over-integration - Cost estimation

Due to the motion of the geometry, it can be represented from one time step to the other differently. As previously mentioned, this introduces a small error, which can be reduced through over-integration. However, over-integration is coupled with increasing computational cost. As the porous material is defined as a space-time function and during initialization we **identify** elements that might have geometry properties; we consider those elements to introduce the over-integration for the masking function. Hence the additional computational effort is only for χ and only in those elements, where $\chi \neq 0$. E.g., a rotating object only requires a limited area of the simulation domain. Hence, in that region, χ is expected, and more integration points are necessary to approximate the porous material appropriately. The shape (a box) allows limiting the over-integration only where required and exclusively for evaluating χ.

We consider a small test case to show how this simple approach can help to reduce the computational cost. For our investigation, we investigate four scenarios to define where over-integration for χ is required. In the first scenario (All: No shape), we do not define any shape where the geometry (χ) might be. Hence the solver needs to check the entire simulation domain for the masking function. Further, for this scenario, over-integration is deployed for all operations, e.g., the expensive flux computation and boundary elements. In the second case, we only define the over-integration for the masking function (Mat: No shape); again, we do not define a shape here and require the solver to apply over-integration only for χ. Nevertheless, the solver also checks the entire simulation domain for the masking function and applies the over-integration, wherever χ is 1.0. In the third scenario, we predefine a shape, bounding the geometry motion in that area. The solver only needs to check for χ in that specific area of the simulation domain and apply over-integration. Again we define over-integration in the entire domain (All: Shape-based) for this case. Though, over-integration is done for all terms. However, outside of the

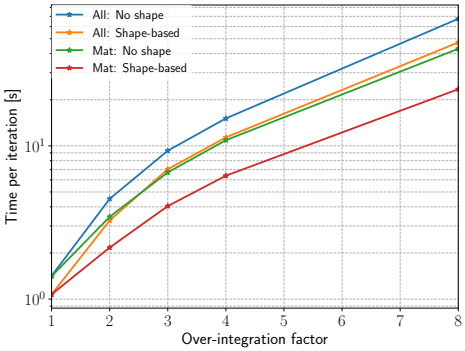


Figure 5.5.: Comparison: Compute time per iteration, for four different scenarios, over-integration in the entire domain (All) and just for the porous material (Mat). We distinguish between shape-based and no shape, to restrict the area, where the porous material is or might be.

shape, no material evaluation needs to be done, as the material properties are seen as constant. Lastly, for the fourth test case, we again define the shape for the geometry and only require over-integration to evaluate χ (Mat: Shape-based). Thus, the solver needs to check the state of χ in the predefined shape and apply over-integration for the masking function.

We consider a small test case, where a wall initially covers half of the domain and moves during runtime from its original location to the right side of the domain (similar to Figure 5.4). As we are only interested in the computational cost, any test case is suitable here. However, this test case provides a good example, as half of the domain is considered to be the porous material, and over-integration is done in that area of the simulation domain. We solve the two-dimensional Euler equations for 100 iterations and keep track of the computational time while neglecting the initialization time. We use a scheme order of $O(8)$ and 1024 elements along the length and one element in height, essentially a one-dimensional simulation domain. For this study an over-integration of $1, 2, 3, 4$ and 8 is utilized. As shown in the previous section, it is more likely that an over-integration of up to 3 or 4 is sufficient for our simulations to enable an adequate geometry representation and reduce the distance between the Chebyshev nodes. Thus, the over-integration factors are selected according

to this assumption. In contrast, a factor of 8 is chosen to show how the computational cost behaves for an even higher over-integration factor.

Results for this study are depicted in Figure 5.5. An over-integration only in the shape (shape-based) can considerably reduce the computational cost per iteration compared to the over-integration in the entire domain without considering the shape for the masking function (no shape). Comparing the curves with and without shape for over-integration (blue and orange curves), we can observe that the computational effort can be reduced with a predefined shape for the masking function. The solver only needs to check the state of the masking function inside the shape and apply the over-integration for χ. Further, it is clear that considering over-integration for χ, where it is needed, can further reduce the computational effort. Comparing the green and red curves (Mat), we can reduce the computational time per iteration by roughly 40% in the case of an over-integration factor of 3. Comparing the blue line (All, no shape) and the red line (Mat, shape-based), the reduction in computational time is even higher, with 56% for an over-integration factor of 3. However, we need to notice that from test case to test case, and the different sizes of the predefined shape for the masking function and the increase in complexity of the dimensions, the computational effort varies for the over-integration. But from the results in Figure 5.5 we can conclude that the best results in terms of computational cost can be achieved employing the shaped-based over-integration only for the masking function χ.

5.6. Specification of the masking function χ

In the previous sections, we explained our method of representing geometries with a one-dimensional problem in space to understand the used concept. We now extend our previous example and have a closer look into problems in two-dimensional space, where the modeling of geometries can become tangled. As previously mentioned, geometries are defined as a space-time function that enables the geometrical motion throughout the simulation. Depending on the complexity of the geometry, there might exist an analytical function to describe its surface. It can, e.g., be a cylinder defined by its radius or a rectangle defined by the length of its edges. Apart from simple geometries, we are more interested in realizing simulations with complex geometries that often have a complex surface, such as, e.g., turbine blades, which have different curvatures and can hardly be described by a simple function. In order to allow for simulations

with complex geometries, where the geometrical surface can not be easily defined, we use a polygonial definition.

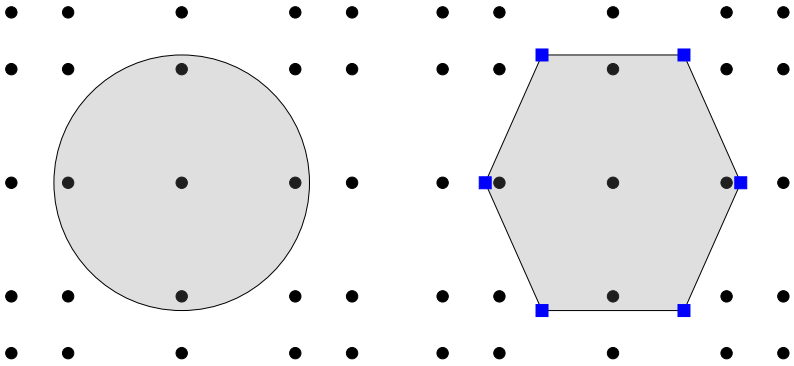


Figure 5.6.: Representation of the geometry interface in the solver. (a) Through an analytical function and in (b) through a list of points, that are connected to each other (segments).

The polygon is defined by a list of vertices with linear segments connecting them. Polygons can approximate curved surfaces if a sufficient number of vertices are provided to describe the geometry surface. Thus, this option enables the representation of arbitrary surfaces. In Figure 5.6 the geometry representation with both options is provided. In one case the analytical function for the geometry representation (cylinder) is provided (cf. Figure 5.6a). In the other case, we assume an analytical function is not available, and a list of six points (blue squares) is used to define the surface of the cylinder geometry (see Figure 5.6b). The provided figure considers one computational element and a scheme order of $O(5)$, resulting in five integration points per direction (black dots) representing the masking function χ. If an analytical function is available, the cylinder can be represented with increasing spatial resolution as good as the analytical solution. However, if we provide a list of points, the geometry can be resolved as well as the segments that represent the surface when increasing the spatial resolution. In order to have a detailed description of the geometrical surface, more vertices are required to have a better representation of a curved surface. The influence of the number of vertices

to represent the geometry surface on the computational cost is studied in more detail in Section 6.7.2.

We now continue with the time integration to address the time restriction caused by the penalty terms introduced by the modeling method. As mentioned in Section 5.1, due to tiny values of the permeability terms, the explicit time-stepping scheme is restricted to small time steps. In order to allow for larger time steps and enable a feasible computation, an implicit-mixed-explicit time-stepping scheme is deployed.

5.7. Implicit-mixed-explicit Runge-Kutta method

Due to the additional terms in the equations Eq. (5.3), they become more challenging to be handled, as an additional amount of numerical stiffness by the permeability terms is introduced to the system to be solved. Therefore a pure explicit approach in time, as shown in Eq. (3.23) is unfeasible from the computational perspective, as the time step is restricted to a tiny one to ensure stability. Hence a hybrid time-stepping approach is implemented to address this drawback efficiently. The implicit-mixed-explicit (IMEX) Runge-Kutta time integration [3] allows to deal with this issue, by discretizing the additional penalization terms in an implicit approach. The equations in Eq. (5.1) can be split into two parts, one that is explicitly solved and one which is implicitly computed [8], that results in

$$\partial_t \rho = C_\rho^\xi + C_\rho^\iota \tag{5.10a}$$

$$\partial_t \boldsymbol{m_u} = C_{\boldsymbol{m_u}}^\xi + C_{\boldsymbol{m_u}}^\iota \tag{5.10b}$$

$$\partial_t e = C_e^\xi + C_e^\iota. \tag{5.10c}$$

Where C denotes the rhs of the equation, with the superscript ι and ξ representing the implicit and explicit part, respectively. The subscripts ρ, $\boldsymbol{m_u}$ and e denote the variables that have to be conserved. The implicit part is chosen to be the penalization terms in our numerical scheme of Eq. (5.10), as they are spatial derivative-free and can be solved pointwise. They can be written as

$$C_\rho^\iota = 0 \tag{5.11a}$$

$$C_{\boldsymbol{m_u}}^\iota = -\frac{\chi}{\eta}\left(\boldsymbol{u} - \boldsymbol{u_G}\right) \tag{5.11b}$$

$$C_e^\iota = -\frac{\chi}{\eta_T}(T - T_G) - \frac{\chi}{\eta}(\boldsymbol{u} - \boldsymbol{u_G}) \cdot \boldsymbol{u}. \tag{5.11c}$$

In the first step, we only consider the implicit part of the equations, shown in Eq. (5.11).

$$\partial_t \rho = 0 \tag{5.12a}$$

$$\partial_t \boldsymbol{m_u} = -\frac{\chi}{\eta} \cdot (\boldsymbol{u} - \boldsymbol{u}_G) \tag{5.12b}$$

$$\partial_t E = -\frac{\chi}{\eta_T}(T - T_G) - \frac{\chi}{\eta}(\boldsymbol{u} - \boldsymbol{u_G}) \cdot \boldsymbol{u} \tag{5.12c}$$

To obtain a discretized equation system that needs to be solved, we first apply the Euler backward scheme to Eq. (5.12)

$$\frac{\rho(t + \Delta t) - \rho(t)}{\Delta t} = 0 \tag{5.13a}$$

$$\frac{\boldsymbol{m_u}(t + \Delta t) - \boldsymbol{m_u}(t)}{\Delta t} = -\frac{\chi}{\eta}(\boldsymbol{u}(t + \Delta t) - \boldsymbol{u_G}) \tag{5.13b}$$

$$\begin{aligned}\frac{E(t + \Delta t) - E(t)}{\Delta t} = &-\frac{\chi}{\eta_T}(T(t + \Delta t) - T_G) \\ &- \frac{\chi}{\eta}(\boldsymbol{u}(t + \Delta t) - \boldsymbol{u_G})\,\boldsymbol{u}(t + \Delta t).\end{aligned} \tag{5.13c}$$

From Eq. (5.13a) it is apparent, that $\rho(t + \Delta t)$ yields to $\rho(t + \Delta t) = \rho(t)$. This simplification can be used to derive from Eq. (5.13b) an explicit expression for the velocity $\boldsymbol{u}(t + \Delta t)$. We can achieve the explicit formulation for velocity, when considering the momentum $\boldsymbol{m_u}$ being

$$\frac{\boldsymbol{m_u}(t + \Delta t) - \boldsymbol{m_u}(t)}{\Delta t} = \frac{\rho(t + \Delta t)\boldsymbol{u}(t + \Delta t) - \rho(t)\boldsymbol{u}(t)}{\Delta t}. \tag{5.14}$$

Eq. (5.13b) can be reformulated to

$$\frac{\rho(t + \Delta t)\boldsymbol{u}(t + \Delta t) - \rho(t)\boldsymbol{u}(t)}{\Delta t} = \frac{\chi}{\eta}(\boldsymbol{u}(t + \Delta t) - \boldsymbol{u_G}), \tag{5.15}$$

obtained by reordering Eq. (5.15) according to $\boldsymbol{u}$.

$$\boldsymbol{u}(t + \Delta t) = \frac{\rho(t)\boldsymbol{u}(t) - \frac{\chi \Delta t}{\eta}\boldsymbol{u_G}}{\rho(t + \Delta t) - \frac{\chi \Delta t}{\eta}} \tag{5.16}$$

Considering the simplification in density ρ, Eq. (5.16) can be further simplified to

$$\boldsymbol{u}(t+\Delta t) = \frac{\rho(t)\boldsymbol{u}(t) - \frac{\chi\Delta t}{\eta}\boldsymbol{u_G}}{\rho(t) - \frac{\chi\Delta t}{\eta}}. \tag{5.17}$$

Finally, the temperature T can be explicitly determined, where the relation between energy and temperature can be written as $E = \frac{1}{2}\rho\boldsymbol{u}\boldsymbol{u} + \frac{p}{\gamma-1} = \frac{1}{2}\rho\boldsymbol{u}\boldsymbol{u} + \rho c_v T$. With this formulation and Eq. (5.13c) we can obtain the explicit formulation of the energy as

$$\begin{aligned}\frac{E(t+\Delta t) - E(t)}{\Delta t} &= \frac{\frac{\rho(t+\Delta t)\boldsymbol{u}^2(t+\Delta t)}{2} + c_v\rho(t+\Delta t)T(t+\Delta t)}{\Delta t} \\ &\quad - \frac{\frac{\rho(t)\boldsymbol{u}^2(t)}{2} - c_v\rho(t)T(t)}{\Delta t} \\ &= -\frac{\chi}{\eta_T}\left(T(t+\Delta t) - T_G\right) \\ &\quad - \frac{\chi}{\eta}\left(\boldsymbol{u}(t+\Delta t) - \boldsymbol{u_G}\right)\boldsymbol{u}(t+\Delta t).\end{aligned} \tag{5.18}$$

From Eq. (5.18) the temperature can be determined as

$$\begin{aligned}T(t+\Delta t) = &\frac{\frac{\chi\Delta t}{\eta_T}T_G - \frac{\chi\Delta t}{\eta}\left(\boldsymbol{u}(t+\Delta t) - \boldsymbol{u_G}\right)\boldsymbol{u}(t+\Delta t)}{c_v\rho(t) + \frac{\chi\Delta t}{\eta_T}} \\ &+ \frac{-c_v\rho(t)T(t) + \frac{\rho(t)}{2}(\boldsymbol{u}^2(t) - \boldsymbol{u}^2(t+\Delta t))}{c\rho(t) + \frac{\chi\Delta t}{\eta_T}},\end{aligned} \tag{5.19}$$

where $\boldsymbol{u}(t+\Delta t)$ can be computed according to Eq. (5.17). These formulations enable solving the implicit part of the time integration scheme, in an explicit strategy, without introducing much more computational effort. However, the demonstrated Euler backward method is only 1^{st} order in time. A third-order Runge-Kutta method is deployed, where the diagonally implicit Runge-Kutta [3] method is used. It includes three explicit and four implicit substages. This strategy provides a third-order accurate result in time and is L-stable. The substages can be computed when considering matrix-vector notation, as shown below.

Substage 1 - Explicit:

$$\hat{\boldsymbol{h}}^1 = \boldsymbol{M}^{-1} \cdot \boldsymbol{C}^{\boldsymbol{\xi}}(\hat{\boldsymbol{u}}(t + c_1\Delta t), t + c_1\Delta t) \tag{5.20a}$$

Substage 1 - Implicit (embedded part):

$$\begin{aligned}\hat{\boldsymbol{u}}^1(t+c_2\Delta t) = \hat{\boldsymbol{u}}(t) &+ \alpha_{1,1}\Delta t \boldsymbol{M}^{-1}\boldsymbol{C}^{\boldsymbol{\iota}}(\hat{\boldsymbol{u}}^1(t+c_2\Delta t)) \\ &+ \Delta t \alpha_{2,1}\hat{\boldsymbol{h}}^1\end{aligned} \tag{5.20b}$$

Substage 2 - Explicit:

$$\hat{\boldsymbol{h}}^2 = \boldsymbol{M}^{-1}\cdot\boldsymbol{C}^{\boldsymbol{\xi}}(\hat{\boldsymbol{u}}^1(t+c_2\Delta t), t+c_2\Delta t) \tag{5.20c}$$

Substage 2 - Implicit (embedded part):

$$\begin{aligned}\hat{\boldsymbol{u}}^2(t+c_3\Delta t) = \hat{\boldsymbol{u}}(t) &+ \alpha_{2,1}\Delta t \boldsymbol{M}^{-1}\boldsymbol{C}^{\boldsymbol{\iota}}(\hat{\boldsymbol{u}}^1(t+c_2\Delta t)) \\ &+ \alpha_{2,2}\Delta t \boldsymbol{M}^{-1}\boldsymbol{C}^{\boldsymbol{\iota}}(\hat{\boldsymbol{u}}^2(t+c_3\Delta t)) \\ &+ \bar{\alpha}_{3,1}\Delta t \boldsymbol{h}^1 + \bar{\alpha}_{3,2}\Delta t \boldsymbol{h}^2\end{aligned} \tag{5.20d}$$

Substage 3 - Explicit:

$$\hat{\boldsymbol{h}}^3 = \boldsymbol{M}^{-1}\cdot\boldsymbol{C}^{\boldsymbol{\xi}}(\hat{\boldsymbol{u}}^2(t+c_3\Delta t), t+c_3\Delta t) \tag{5.20e}$$

Substage 3 - Implicit (embedded part):

$$\begin{aligned}\hat{\boldsymbol{u}}^3(t+c_4\Delta t) = \hat{\boldsymbol{u}}(t) &+ \alpha_{3,1}\Delta t \boldsymbol{M}^{-1}\boldsymbol{C}^{\boldsymbol{\iota}}(\hat{\boldsymbol{u}}^1(t+c_2\Delta t)) \\ &+ \alpha_{3,2}\Delta t \boldsymbol{M}^{-1}\boldsymbol{C}^{\boldsymbol{\iota}}(\hat{\boldsymbol{u}}^2(t+c_3\Delta t)) \\ &+ \alpha_{3,3}\Delta t \boldsymbol{M}^{-1}\boldsymbol{C}^{\boldsymbol{\iota}}(\hat{\boldsymbol{u}}^3(t+c_4\Delta t)) \\ &+ \bar{\alpha}_{4,1}\Delta t \boldsymbol{h}^1 + \bar{\alpha}_{4,2}\Delta t \boldsymbol{h}^2 + \bar{\alpha}_{4,3}\Delta t \boldsymbol{h}^3\end{aligned} \tag{5.20f}$$

Substage 4 - Explicit:

$$\hat{\boldsymbol{h}}^4 = \boldsymbol{M}^{-1}\cdot\boldsymbol{C}^{\boldsymbol{\xi}}(\hat{\boldsymbol{u}}^3(t+c_4\Delta t), t+c_4\Delta t) \tag{5.20g}$$

The update rule is given as:

$$\begin{aligned}\hat{\boldsymbol{u}}(t+\Delta t) = \hat{\boldsymbol{u}}(t) &+ \beta_1\Delta t \boldsymbol{M}^{-1}\boldsymbol{C}^{\boldsymbol{\iota}}(\hat{\boldsymbol{u}}^1(t+c_2\Delta t)) \\ &+ \beta_2\Delta t \boldsymbol{M}^{-1}\boldsymbol{C}^{\boldsymbol{\iota}}(\hat{\boldsymbol{u}}^2(t+c_3\Delta t)) \\ &+ \beta_3\Delta t \boldsymbol{M}^{-1}\boldsymbol{C}^{\boldsymbol{\iota}}(\hat{\boldsymbol{u}}^3(t+c_4\Delta t)) \\ &+ \bar{\beta}_1\Delta t \hat{\boldsymbol{h}}^1 + \bar{\beta}_2\Delta t \hat{\boldsymbol{h}}^2 + \bar{\beta}_3\Delta t \hat{\boldsymbol{h}}^3 + \bar{\beta}_4\Delta t \hat{\boldsymbol{h}}^4\end{aligned} \tag{5.20h}$$

The coefficients α, $\bar{\alpha}$, β, $\bar{\beta}$ and c are given in the Runge-Kutta tableau in [98] that are used for this work as well, while $\boldsymbol{C}^{\boldsymbol{\iota}}$ and $\boldsymbol{C}^{\boldsymbol{\xi}}$ denote the the penalization operation and the discrete hyperbolic/parabolic operation. It has to be noticed that each implicit substage requires a transformation of the modal (polynomial information) to nodal (pointwise information) and vice versa, as the material parameters are calculated pointwise. More information regarding the transformation from modal to nodal and vice versa can be found in Chapter 3 and [98].

Conclusion The Brinkman volume penalization method is utilized to model the geometry for numerical simulations in our high-order Discontinuous Galerkin solver. We presented how our implicit-mixed-explicit time-stepping scheme allows overcoming the additional numerical stiffness introduced by the penalty terms. Further, the geometrical representation in our high-order numerical scheme was introduced, where the masking function χ was evaluated at the integration points to identify the geometry and its respective location. We illustrated a one-dimensional problem that the geometry is differently represented in each position due to the non-equidistant point distribution of the integration points. The most significant error in the geometry modeling is caused by the largest distance between the integration points. They can be found in the middle of an element. Over-integration was introduced to reduce the error in the modeling caused by the point distribution. Using more integration points to approximate the masking function, thus the geometry improves the modeling and reduces the error from the point distribution. This method yet increases the computational effort; however, it can be reduced by appropriate measures. The over-integration is only done in the first place for the approximation of the masking function. Additionally, a limiting shape (box) is introduced to reduce the computational effort further, as only inside the shape the masking function is evaluated, and the over-integration incorporated. The shape defines the area where the geometry is and might be during the simulation when it moves. E.g., in cases where the geometry only rotates around a particular center, it will only require a specific area. This practice allows to reduce the computational effort further and use over-integration where it is indispensable.

In the following chapter, the validation of the embedded method to arbitrary model geometries and their motion in our high-order scheme is presented employing test cases known from the literature. They involve shocks as well as curved boundaries and sharp edges. Those investigations demonstrate that the embedded method used to model geometries can

preserve shocks while the underlying Discontinuous Galerkin scheme stays stable. Furthermore, we can highlight that this method is also suitable for modeling curved and more complex boundaries.

6. Validation of the moving geometry

In this section, the motion of the geometry, modeled as an artificial porous material, is validated through different test cases that demonstrate different numerical challenges. Test cases deal, among others, with shocks, even though it is known that high-order poses some challenges for shock capturing due to strong discontinuity and the resulting Gibbs oscillations (see Section 3.3). This chapter aims to validate the porous material used to model moving geometries. Moreover, investigations in this chapter verify that more accurate solutions can be obtained when compared to a low-order scheme (O(2)), even though an additional discontinuity (geometry) is introduced into the simulation domain.

The first test case deals with an acoustic pulse that moves throughout the simulation time towards a moving wall and is reflected to a predefined location in the simulation domain. The second test case focuses on a shock that travels towards a non-moving wall and is reflected to a predefined location. This test case is later compared to a shock that moves towards a moving wall and is reflected. Afterward, we discuss a moving piston, where a shock is formed ahead of the piston. Exact solutions exist for the shock and piston test case in the literature, used for comparison. Lastly, we investigate curved boundaries using a moving cylinder and a moving wedge test case. In the case of the moving cylinder, an exact solution is not available; therefore, simulation results are compared to the already validated non-moving cylinder test case. For the wedge test case, an exact solution can be found in the literature. With these investigations, we want to demonstrate that: **(i)** The porous material acts as a solid geometry through low permeability for the modeling terms. **(ii)** Obtained results are in excellent agreement with known exact solutions. Further, numerical stability is maintained even though the step function χ introduces a further discontinuity. **(iii)** A high-order scheme still enables more accurate solutions, even in the vicinity of strong discontinuities.

6.1. Convergence study - Acoustic pulse

To investigate the ability of the porous material in order to model solid moving geometries, we apply a simple test case. We start with the L2 error convergence study that Anand, EP et al. [8] have used. However, this test case is slightly modified, where a moving wall replaces the fixed wall. All other flow properties, as well as the simulation domain, are kept the same as in [8].

Test case description: The one-dimensional simulation domain has a length l of 1 unit length. From $x = 0.475$ to $x = 1.0$ the computational domain is covered by the modeled wall. The pulse is located at $x = 0.225$, we expect the pulse at the end of the simulation to be at $x = 0.25$ comparatively to [8], and the wall at $x = 0.5$. The wall speed is defined to be 0.05 unit speed. The time step is controlled by the CFL condition with a Courant factor of 0.25 and the inviscid Euler equations are solved. The Courant factor is chosen to be a moderate value, as the velocity of the geometry is not considered in the CFL computation, thus a smaller value is chosen to ensure stability of the numerical simulation.
Initial conditions We prescribe the background pressure p_B to be $1/\gamma$, with $\gamma = 1.4$ being the isothermal coefficient. The background density ρ_B has a value of 1, resulting in a speed of sound of $c = 1$. The background velocity u_B is defined to have the same speed as the moving wall. The pulse is given in terms of deviations from the background values (density ρ', pressure p' and velocity u'), with a maximal deviation of 10^{-3} (cf. Eq. (6.1)). Thus at $t = 0$ the density is $\rho_B + \rho'$, the pressure is $p_B + p'$ and the velocity is $u_B + u'$.

$$\rho' = u' = p' = 10^{-3} exp\left[-\ln 2 \frac{(x - 0.225)^2}{0.004} \right] \tag{6.1}$$

Boundary conditions At the left boundary (Dirichlet boundary condition) p_B, ρ_B and u_B are prescribed. The right boundary is defined as subsonic outflow, where p_B is defined.

In acoustics theory, the reflection is perfectly symmetric, and the reflected pulse maintains its shape and size. For this linear test case, the analytical solution can be easily computed and deployed to investigate the quality of the numerical solution, considering the modeled porous wall. Furthermore, it allows for the analysis of the amplitude and the phase shift of the reflected pulse. Generally, the analytical solution can be used

as a reference for comparison with the numerical solution. However, as nonlinear equations are solved, and tiny errors are obtained, the analytical solution is unsuitable. This is due to the nonlinearity that is not considered in the analytical solution of the linear equation. Therefore the analytical solution deviates from the nonlinear behavior, restricting its suitability for the convergence investigation. For the computation of the L2 error, a highly resolved solution is used as a reference, for our convergence study. The reference is computed in a smaller domain of size 0.5 unit length, with an isothermal wall boundary condition at the right end of the domain. The same temperature is prescribed for the wall boundary condition. For the simulation, a polynomial degree of 255 and 120 elements for the computational mesh are used, which results in a well-resolved solution obtained by a convergence study.

Simulation results for different scheme orders and 48 elements are presented in Figure 6.1 (h refinement). The black line represents the reference solution using a scheme order of $O(256)$, all other curves depict the solution for $O(2)$ up to $O(64)$. Obviously, with higher scheme orders, the solution resembles more and more the reference solution. However, when considering a lower scheme order, e.g., $O(2)$ or $O(4)$, a high dissipation and dispersion error (smaller amplitude and a large phase shift) is noticeable. Starting from $O(16)$ the numerical solution resembles the reference solution, where a smaller phase shift can be observed, and the error in the amplitude is significantly reduced. For a scheme order of $O(32)$ and $O(64)$, the solution agrees with the reference solution. To allow for a more detailed investigation of the quality of the solution, a convergence analysis is conducted. We are interested in the L2 error, which can be computed according to

$$|\text{L2 error}| = \sqrt{\sum_{j=1}^{n} |x_j - y_j|^2}, \tag{6.2}$$

where x is the numerical solution, y the high resolved reference solution, and n the number of measurement points. Point values are obtained by evaluating the polynomial representation at determined point positions in the simulation domain. The investigated area is from $x = 0$ to $x = 0.5$, hence only the flow field. Our convergence study is conducted for the last time step of each simulation, where the pulse has reached its final location. Measurement points only need to be evaluated during post-processing and only for the last time step. To avoid aliasing effects, the number of measurement points is three times more than the number of integration points

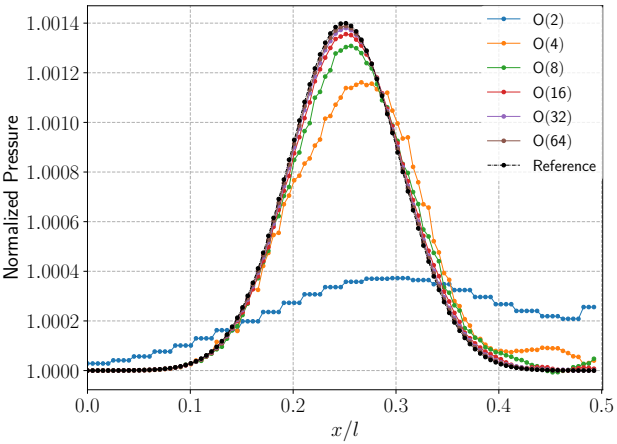
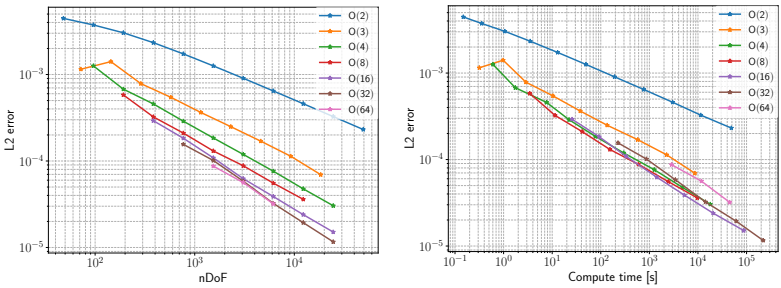


Figure 6.1.: Comparison: Curves represent simulation results for different scheme orders, compared to the reference solution. The simulation domain is discretized with 48 elements.

per element. In Figure 6.2 the convergence study is shown for the variable pressure. Figure 6.2a presents the error over the number of degrees of freedom (nDoF); hence the memory requirement for the simulation. Figure 6.2b shows the L2 error against the respective time to solution, executed on a single node with 48 cores on SuperMUC-NG supercomputing system at Leibniz Rechenzentrum (LRZ). The investigation starts with 48 elements in each data series, the leftmost point in both subplots. For the following points in our study, the element count is continuously increased by a factor of 2. We consider a wide range of scheme orders starting from $O(2)$ up to $O(64)$. We do not reach in Figure 6.2 a spectral convergence as expected for smooth solutions. The error decreases only linearly with increasing scheme order. However, the high-order discretization is still advantageous when considering the required memory consumption. In terms of computation, we can observe a noticeable improvement of the computational effort from $O(2)$ to $O(3)$ and $O(4)$. Nevertheless, the computational effort increases again for a higher order than $O(16)$. Even though, the timings depend on the system the simulations are executed on, however this observation holds true also on other computing systems.



(a) (b)

Figure 6.2.: Curves represent different scheme orders, while dots indicate the L2 error for different element counts. (a) L2 error over the number of degrees of freedom (nDoF) and (b) L2 error over the computational time. Simulations are conducted on one single node with 48 cores, using the computing system Supermuc-NG.

There exist two decisive contributions that diminish the convergence order for our simulations. The first one is related to the Gibbs phenomenon that introduces oscillations due to the discontinuity of the masking function χ (cf. Section 3.3). This error source can be limited by utilizing a re-projection method, e.g., recommended in [42]. However, this is not used in this work. The second error source is related to the numerical integration, hence the Chebyshev nodes and their respective distribution, as previously discussed in Section 5.3. The inaccuracy coming from the integration points can be reduced by over-integration. However, as mentioned in the respective Section 5.3, the node distribution is not equidistant, and most nodes are found at the element corners. Here the distance towards the element corners decreases with 2^{nd} order when using over-integration, while in the middle of the element, this is only of 1^{st} order. Thus the error from the numerical integration is dominated by the largest distance between the nodes, which only decreases with 1^{st} order and can be found in the middle of an element. As the wall moves and is represented differently, the most significant error dominates our L2 error solution. Even though the convergence rate drops to 1^{st} order, the solution in the smooth part of the simulation domain is still of high-order as shown in [8] and [77], allowing to capture the flow field with high accuracy.

Table 6.1.: Comparison of different scheme orders with the same nDoF of 49, 152.

Scheme order	L2 error $\times 10^{-4}$	Computational time [s] $\times 10^{3}$
O(2)	3.26396895	11.407
O(4)	0.30412917	17.889
O(32)	0.11621092	61.132

Though the error drops for h and p refinement linearly. However, we can observe that high-order schemes can still exhibit their advantage by providing a small L2 error with fewer degrees of freedom (nDoF) as in Figure 6.2a.

Considering the computational cost in more detail, we can observe that with increasing scheme order, the time to solution increases as well. It is due to the explicit time-stepping scheme and the stability criterion (cf. Section 3.2). Thus, the time step size decreases with increasing scheme order while the number of iterations increases to achieve the predefined simulation time. However, when compared to the error achieved with a higher scheme order, the computational time to attain a small error of e.g., 10^{-4}, is linked to fewer nDoF as well as computational time. This behavior can be observed up to $O(16)$, which is obviously cheaper in computation when compared to $O(2)$, $O(3)$, $O(4)$ and in some point also to $O(8)$.

In order to compare the solution of different scheme orders with the same nDoF, in Table 6.1 the L2 error and the computational time to reach the predefined simulation time for the cases $O(2)$, $O(4)$ and $O(32)$, are provided. We compare the solutions with the same nDoF of 49, 152 per variable, with 24, 576 elements for $O(2)$, 12, 288 elements for $O(4)$ and 1, 536 elements for $O(32)$. Even though all three cases have the same number of nDoF, the solutions are different compared to the reference solution. For O(2), the deviation from the exact solution is severe (cf Figure 6.3). It is expected, as the low-order scheme tends to smear the solution due to numerical dissipation, which diminishes the pulse's amplitude and cannot capture the pulse appropriately. For $O(32)$, we can observe an excellent agreement with the reference solution (cf. Figure 6.3a and Figure 6.3b), this is due to the inherent properties of the high-order scheme,

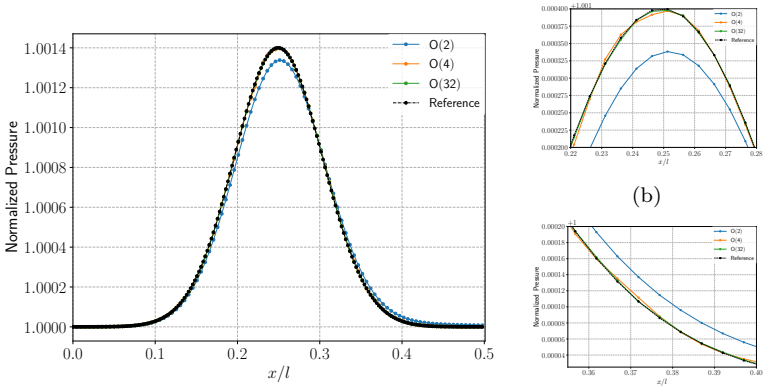


Figure 6.3.: Comparison of the reflected pulse with the reference solution, for the same nDoF of 49, 152. Curves represent the solution for the scheme orders $O(2)$, $O(4)$ and $O(32)$, with 24, 576, 12, 288 and 1, 536 elements, respectively. Pressure curves are normalized by the background pressure p_B. (b) and (c) present a zoom-in into the maxima and the lower right of the pulse solution.

that allows low dispersion and dissipation error. In the case of $O(4)$, the solution is significantly improved when compared to $O(2)$; however, at the maxima (cf Figure 6.3a) as well as on the right lower part of the pulse (cf. Figure 6.3b) we can observe some deviation from the reference solution again. Comparing the L2 error provided in Table 6.1, we can confirm our perception from Figure 6.3a, where the error is smaller for $O(32)$, when compared to $O(2)$ or $O(4)$. Though the desired small error with $O(32)$ comes with a higher computational cost, one might assume that $O(32)$ is not reasonable for the computation, as the computational effort is roughly four times higher, compared to $O(4)$. However, we need to emphasize that with a scheme order of $O(32)$, the error is almost three times smaller than for $O(4)$. To achieve the same error as for the case of $O(4)$, yet, with a scheme order of $O(32)$, the required time to solution is roughly the same. However the required nDoF, would only be 24, 576 per variable, thus less memory is needed (cf. Figure 6.2) compared to $O(4)$. On the other hand, if the same error as obtained for $O(32)$ is desired for a scheme order of $O(4)$,

a higher resolution for the computational mesh is required (3×). It leads to a higher computational effort and a higher memory consumption (393, 216 nDoF per variable) compared to $O(32)$. Even though, in Table 6.1 the simulation with $O(32)$ might give the impression that it is too expensive to be considered, we need to keep in mind that this only holds at first glance.

We can conclude from this test case that we can attain numerical solutions, which are in excellent agreement with the reference solution using high-order schemes. Even though strong discontinuities (moving wall) are adverse for high-order methods. Additionally, we confirmed that a higher accuracy of the solutions could be obtained when using a high-order scheme. In the next section, we examine more complex test cases, which are well known to pose challenges for high-order schemes, specifically shocks, a further discontinuity added in the simulation domain.

6.2. Shock capturing - Shock-wall interaction

For many engineering applications, the spontaneous formation of shocks is unavoidable; even though conservation quantities are smooth, they pose challenges for the numerical scheme. The major problem with shocks and high-order discretizations are, as explained previously, the Gibbs oscillations. However, as mentioned in Section 3.3 there exist different techniques to address this issue throughout runtime and post-processing. With that, it is possible to also keep a closer look into shocks when considering a moving geometry in the simulation domain, and the discretization of the fluid domain is done with high-order. The question that needs to be answered here is whether the chosen approach, namely the Brinkman penalization, can capture shocks appropriately. For a more detailed examination, a shock test case is investigated. The shock is located at a predefined location and travels over time towards a wall modeled as a porous material. The shock is reflected by the wall and travels back to its original position.

6.2.1. Interaction of a shock wave with a non-moving wall

The first investigation is simulated with a non-moving wall that is later taken into account for comparison with the moving wall test case and the exact solution. For this investigation, we neglect in the first place the viscosity of the Navier-Stokes equations, resulting in the inviscid Euler equations, comparable to the exact solution. This test case has already been investigated by Piquet et al. [75]. For comparison, we consider the same test case while the high-order Discontinuous Galerkin scheme is

Table 6.2.: Description of the shock state

Speed of sound at region 1	c_1	1.0
Shock Mach number	Ma_s	1.2
Velocity of the shock	u_s	1.2
Density downstream	ρ_1	1.0
Pressure downstream	p_1	γ^{-1}
Velocity downstream	u_1	0.0
Isentropic coefficient	γ	1.4

utilized, studying different polynomial degrees. Even though a high-order scheme is not ideal for shock representation, this test case should explain how we can deal with the discontinuity in our numerical scheme.

Test case description: The simulation is realized in a computational domain of has a length of $l = 1$ unit length, where the wall is located at $x = 0.5$. The wall covers half of the domain ($x \in [0.5, 1]$), modeled as a porous material, with properties resembling a solid wall. The initial location of the shock is at $x = 0.25$. In order to investigate the influence of the spatial discretization on the solution obtained, we examine the following configurations $256, 512, 1,024$, and $2,048$ elements (n) in total ($\Delta x = 1/n$) and a scheme order (O) of 32, 16, 8 and 4, respectively. With these configurations, the nDoF is kept the same by reducing the number of elements while increasing the scheme order simultaneously.
Initial conditions Initially, the state inside the domain is prescribed according to the values defined in Table 6.2 downstream and the values upstream are computed according to the Rankine-Hugoniot conditions.
Boundary conditions At the left boundary, we prescribe the primitive variables pressure, density, and velocity corresponding to the values upstream. The right boundary is defined as an outflow, where the pressure downstream is prescribed.

In Table 6.2 the simulation setup is given for the validation. Downstream values describe the state in front of the shock, denoted by 1 (cf. Figure 6.4 *Region* 1). Variables in *Region* 2, hence the state after the shock, are denoted by 2. This information can be computed by the Rankine-Hugoniot conditions, considering the shock Mach number Ma_s. With that, we obtain

$$\frac{\rho_2}{\rho_1} = \frac{\gamma + 1}{\gamma - 1 + 2{Ma_s}^{-2}} \tag{6.3}$$

for the ratio between the densities ρ_1 and ρ_2 before and after the shock, respectively. Further, we take the pressure ratio

$$\frac{p_2}{p_1} = \frac{2\gamma {Ma_s}^2 - (\gamma - 1)}{\gamma + 1}, \tag{6.4}$$

to calculate the relation between the pressures p_2 and p_1. These relations can then be used, to finally obtain the ratio between the pressures p_3 and p_2 of the reflected shock wave (cf. Eq. (6.5)) [14].

$$\frac{p_3}{p_2} = \frac{Ma_s^2(3\gamma - 1) - 2(\gamma - 1)}{2 + Ma_s^2(\gamma - 1)} \tag{6.5}$$

For the velocity u_{rs} of the reflected shock wave, Eq. (6.6) [41] is taken into account.

$$u_{rs} = \frac{1}{Ma_s}\left(1 + \frac{2({Ma_s}^2 - 1)}{(\gamma + 1)/(\gamma - 1)}\right)c_1 \tag{6.6}$$

Considering a shock wave velocity of $Ma_s = 1.2$ and Eq. (6.5), the pressure ratio (p_3/p_2) across the shock has a value of 1.47826087.

According to [98], the motion of the shock wave from one element to another causes oscillations. Those oscillations remain in the elements, even though the shock wave has passed through them. In Figure 6.4 and Figure 6.5 we try to reproduce the observation of [98], where results of the shock test case are shown. In this case, the modeled wall at $x = 0.5$ is replaced by an isothermal wall boundary condition. This measure is taken to avoid any influence by the modeled wall and to ensure that the moving shock wave purely causes those oscillations. In Figure 6.4 the initial condition is shown, with the shock positioned at $x = 0.25$. The solution after 0.094 simulation time is presented in Figure 6.5. The shock wave has moved from its initial position, and oscillations are visible behind the shock that moves in the opposite direction. Further, a regular pattern can be recognized. Oscillations remain in the elements, even though the shock wave has already passed through them. In the direct neighborhood of the shock, oscillations match with the element interfaces, e.g., from 0.35742 to 0.36133 (two elements). In contrast, oscillations away from the shock indicate a regular pattern, a plateau, that becomes further flat over time as the oscillations lose strength and level out (cf. Figure 6.5b). Further, in Figure 6.5b, the dash lines indicate the element interfaces. In order to remove those remaining oscillations from the solution, Zudrop [98]

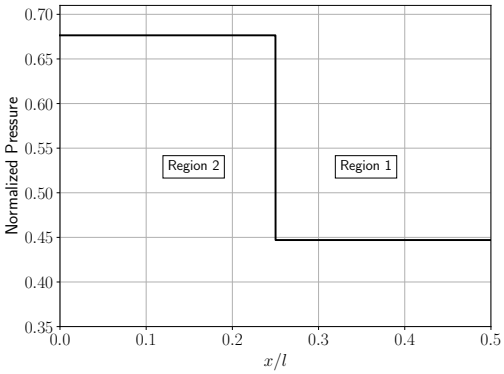
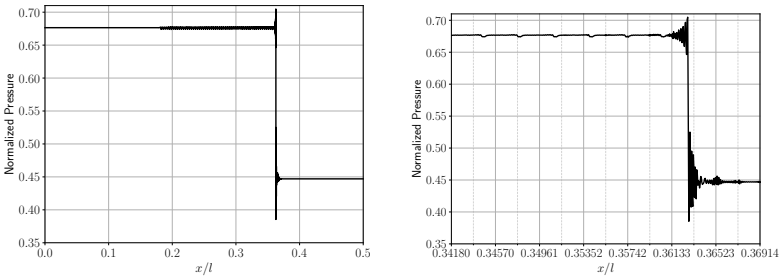


Figure 6.4.: Initial condition of the predefined shock wave in pressure that moves during runtime towards an isothermal wall boundary condition and is reflected. The regions denote the state, with *Region* 1 describing the downstream and *Region* 2 the upstream state of the flow.

recommended using the co-volume filtering during runtime. In the ensuing investigations, we consider this recommendation and use the co-volume filtering for our simulations in this section. A weak filter of $O(32)$ is used, which is sufficient to smoothen the numerical solution and remove remaining oscillations in the elements that are introduced by the moving shock.

Figure 6.6 depicts exemplary the reflected shock wave for different spatial discretizations. Due to the co-volume filtering, we can no longer observe oscillations coming from the shock wave. The curves represent the solution of $O(8)$ and 1024 elements and $O(4)$ with $2,048$ elements. Both have the same number of nDoF, while the last curve for $O(8)$ and $2,000$ elements shows the solution for a higher computational effort. As previously mentioned, the exact solution for the pressure ratio across the shock is 1.47826087. Table 6.3 presents the normalized pressure ratio (p_3/p_2) and the error in percentage between the numerical solution and the exact one (Error in p_3/p_2 in %) close to the reflected shock. Furthermore, the difference in the shock location as initially set and the position after the reflection (Δx: shock position) is shown for all configurations. Obviously, with increasing scheme order, while keeping the number of nDoF the same,

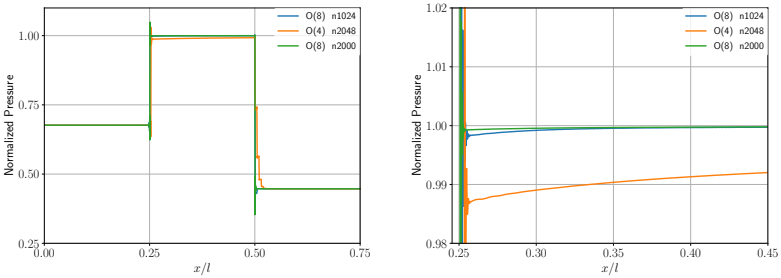


(a) (b)

Figure 6.5.: Visualization of the moving shock wave towards an isothermal wall boundary condition for $t = 0.094$. Due to the strong discontinuity (shock), oscillations remain behind the shock. In (a) the normalized shock wave pressure after movement and in (b) a zoom-in of the area close to the shock is presented.

the error in the numerical solution of the pressure ratio and the shock position is reduced, even though a discontinuous solution is present.

The obtained results are in agreement with the outcomes of [75] for $O(16)$ and 512 elements. From the zoom-in in Figure 6.6b it is apparent that the plateau after the shock is not completely flat but instead has a slope, which asymptotically attains the exact solution. Besides the lowest scheme order of $O(4)$, all configurations investigated indicate the plateau, even though it remains slightly off from the exact value. The remaining error $\mathbf{min_{error}}$ is presented in Table 6.3 as well. This error has a value of 0.0129 %, which is in a reasonable range. The results in Table 6.3 were obtained for a wall located at the edge of an element. As previously discussed, the positioning of the geometry, thus the wall, is essential for its representation in the numerical scheme. Therefore, we investigate the influence of the numerical integration when the wall is located in the middle of a computational element, which is, as previously mentioned, the case with the most significant error coming from the integration points. In Table 6.4 the solutions for the reflected shock wave are shown. The difference between Table 6.3 and Table 6.4 is the positioning of the wall interface, which is in the case of Table 6.4, in the middle of an element. Again we can observe that the error obtained is reduced when increasing the scheme order, while all configurations have the same nDoF. Further, it can be noticed that



(a) (b)

Figure 6.6.: Curves represent different spatial discretizations, considering a variation in the scheme orders and elements. (a) Normalized pressure of the reflected shock wave and (b) zoom-in of the reflected shock.

the error in the pressure ratio is reduced, in the cases with fewer element counts, when compared to the results in Table 6.3, where the wall is located at the element edge. The reason is an additional element introduced here, resulting in a smaller element size. Further, we can observe that the error corresponding to the shock location is considerably larger. Due to the property of the integration points that have a more significant distance in the middle of an element and represent the wall interface, the error is more prominent here.

Table 6.3.: Comparison: Results for the porous material located at the **edge** of the element

Test case	p_3/p_2	Error in p_3/p_2 in [%]	$\Delta x \cdot 10^{-4}$
n2048, O(4)	1.46053873	1.19885086	32.0161
n1024, O(8)	1.47642541	0.12416375	13.0319
n512, O(16)	1.47700446	0.08499256	8.1828
n256, O(32)	1.47714175	0.07570497	7.6228
n2000, O(8)	1.47740998	0.05755990	7.0317
$\mathbf{min_{error}}$	1.47806952	0.012944346	−−

Table 6.4.: Comparison: Results for the porous material located in the **middle** of an element.

Test case	p_3/p_2	Error in p_3/p_2 in [%]	$\Delta x \cdot 10^{-4}$
n2049, O(4)	1.44333420	2.36268639	25.6373
n1025, O(8)	1.47333865	0.33297335	21.5178
n513, O(16)	1.47750687	0.05100577	15.4564
n257, O(32)	1.47751832	0.05023153	13.0052
$\mathbf{min_{error}}$	1.47801754	0.01646083	$--$

In Figure 6.7 the different test cases are shown. Each curve represents a different location of the modeled wall compared to a wall boundary condition located at the same place as the modeled wall. The simulation results are compared to each other for the configuration of $O(16)$. From

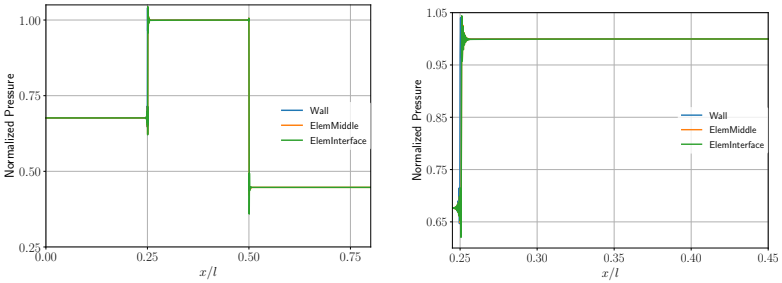


Figure 6.7.: Curves represent the different locations of the porous material in the element and the solution when using a wall boundary condition. The location of the porous material is in one case at the element edge (ElemInterface) and the other at the middle of an element (ElemMiddle). (a) Normalized pressure of the reflected shock wave and (b) zoom-in of the front area of the shock.

Figure 6.7 we can observe that obtained solutions are comparable when modeling the wall as a porous material and when using the reference wall boundary condition. Furthermore, we can recognize that the well-known

Gibbs oscillations in the vicinity of the discontinuities are introduced. However, apart from that, the shock is well sustained by the underlying Discontinuous Galerkin scheme. It is unavoidable that oscillations remain inside the porous material (see Figure 6.7b). Nevertheless, we need to emphasize that the information inside the material is out of interest. In both Figure 6.7a and Figure 6.7b we can observe an over- and undershoot. Since the modeled wall is represented in polynomial space and, according to the Gibbs phenomenon, a deviation of max. 9% is allowed to maintain the physical correctness of the solution [22], we investigate the over- and undershoot in more detail. With the material interface located in the middle of an element, the overshoot is around 2.052%, and the undershoot around 8.639%. Placing the material interface at the element interface results in 1.821% and 7.681% over- and undershoot, respectively.

With the outcomes from this test case, we can continue and set up the same simulation while moving the wall, modeled as a porous material according to a predefined velocity. The question, which arises here is, whether we can reproduce our results from the non-moving porous wall and obtain solutions, which are comparable to the solutions from this section. It needs to be noticed here, that due to the movement of the wall in the next test case, the complexity of this test case increases, as the discontinuity moves and changes its location throughout the simulation time.

6.2.2. Interaction of a shock wave with a moving wall

We rerun the same test case as in Section 6.2.1, however, we now move the wall with a predefined velocity. Wall and shock position are, for this purpose, slightly shifted to the left at the initial condition, though the distance between shock wave and wall remains the same as in the non-moving case. Furthermore, the simulation time is predefined such that at the end of the simulation, the shock wave and the wall have the same final position as in the non-moving case.

Test case description: The simulation domain is the same as in Section 6.2.1, however the wall position is at $x = 0.494340$ and the shock position at $x = 0.244340$, consequently the porous material, that resembles a solid wall, covers a larger area of the simulation domain ($x \in [0.494340, 1]$) than in Section 6.2.1. Again, we keep track of the shock wave position after its reflection and compare it at the end of the simulation time, with the exact position at $x = 0.25$. As described in Section 5.3 in order to represent the wall accordingly, we consider over-integration. In this case,

Table 6.5.: Description of the shock state

Speed of sound downstream	c_1	1.0
Shock Mach number	Ma_s	1.2
Shock speed	u_s	1.2
Density downstream	ρ_1	1.0
Pressure downstream	p_1	γ^{-1}
Velocity downstream	u_1	0.012
Isentropic coefficient	γ	1.4
Wall speed	w_s	0.012

an over-integration factor of 3 is used.
Initial conditions Initially we prescribe again the same values as in Section 6.2.1, but for this test case, we need to consider also the background velocity, which has the same velocity as the wall movement to avoid the formation of an additional shock. Values are defined downstream according to Table 6.5 and upstream according to the Rankine-Hugoniot conditions computed below.
Boundary conditions For the left boundary, we prescribe the modified primitive variables pressure, density, and velocity, considering the values upstream. Finally, along the right subsonic outflow boundary, the pressure downstream is defined.

In Table 6.5 the state downstream (denoted by 1) is presented. With the shock Mach number Ma_s we can obtain for the pressures p

$$\frac{p_2}{p_1} = 1 + \frac{2\gamma(Ma_s - \frac{w_s}{u_s})^2 - 1}{\gamma + 1} \tag{6.7}$$

and for the relation of the densities ρ.

$$\frac{\rho_2}{\rho_1} = \frac{1 + \frac{\gamma+1}{\gamma-1} \cdot \frac{p_2}{p_1}}{\frac{\gamma+1}{\gamma-1} + \frac{p_2}{p_1}}. \tag{6.8}$$

Considering these relations, the exact relation of the pressures p_3 and p_2 of the reflected shock can be computed according to [14], while additionally, the speed of the wall w_s needs to be incorporated.

$$\frac{p_3}{p_2} = \frac{(M_s - \frac{w_s}{c_1})^2(3\gamma - 1) - 2(\gamma - 1)}{2 + (M_s - \frac{w_s}{c_1})^2(\gamma - 1)} \tag{6.9}$$

To compute the speed of the reflected shock wave, Eq. (6.10) [84] is used.

$$u_{rs} = u_2 + c_2 \cdot \sqrt{1 + \frac{\gamma + 1}{2\gamma} \cdot \left(\frac{p_3}{p_2} - 1 \right)} \tag{6.10}$$

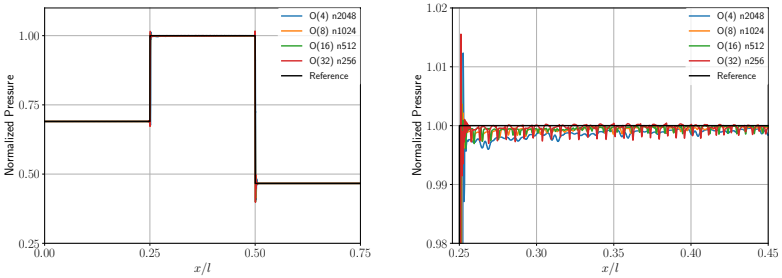
Where c_2 is defined as $\sqrt{\frac{\gamma p_2}{\rho_2}}$ and u_2 is computed according to

$$u_2 = w_s \cdot (\frac{\rho_1}{\rho_2}) + u_s \cdot \left(1 - \frac{\rho_1}{\rho_2} \right). \tag{6.11}$$

From Eq. (6.9) we are able to compute the exact solution for the pressure relation p_3/p_2, that is 1.449111, used to compare with the pressure relation obtained from the numerical simulation.

In Figure 6.8 the simulation results for the same test case as above are shown. They are filtered (spectral and co-volume filter) during post-processing to remove numerical oscillations from the solution (see Section 3.3). As mentioned above, we consider over-integration for the representation of χ and compare again the numerical solution with the exact solution known from the Rankine-Hugoniot conditions. We are interested in the pressure relation as well as the location of the shock. In Figure 6.8 we can observe oscillations in the solution when compared to the solution of the non-moving wall in Figure 6.7. Though, those oscillations were addressed through co-volume filtering, yet, oscillations in Figure 6.8b have a different nature. They are caused by the motion of the discontinuity of the masking function. As previously mentioned, the wall is not uniformly represented (Chebyshev nodes), resulting in remaining oscillations as the discontinuity of the masking function moves through the simulation domain. However, those remaining oscillations are small and become smaller in amplitude with increasing integration points, as the discontinuity can be precisely represented.

This behavior is shown in Figure 6.9, where different over-integration factors are shown for the case of $O(32)$ and 256 elements. Curves represent solutions when using a factor S of 1, 2, 3 and 4. It is apparent that with an increasing number of points, the amplitude of the oscillations can be significantly reduced. For a factor of 3 and 4, the amplitude is the smallest; however, both solutions do not provide a remarkable difference. It justifies our choice for an over-integration factor of only 3 for the simulations conducted in this section. For an over-integration of 4, the ratio between

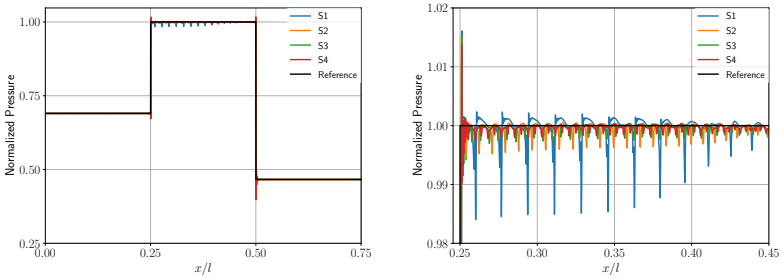


(a) (b)

Figure 6.8.: Different curves represent different discretizations for different scheme orders and number of elements. (a) Normalized pressure of the reflected shock wave and (b) zoom-in of the reflected shock.

improvement of the solution and computational effort to obtain the solution is not compatible. For the factors of 2 and 3 or 1 and 2, the amplitude of those oscillations becomes smaller. Furthermore, with an increasing over-integration factor, the over- and undershoot can be reduced as well (cf. Figure 6.9a and Figure 6.9b). We can further observe in Figure 6.8b that with increasing scheme order and larger elements, the amplitude of the oscillations increases for the solution of the pressure. However, when compared to Figure 6.9b for $S1$ or $S2$, they are less intense and have a higher frequency. In Table 6.6 the solutions for the pressure relation and the location of the shock are shown. Again, the over- and undershoot close to the shock are investigated (cf. Figure 6.8a). Those over- and undershoots are traced back to the Gibbs phenomenon (cf. Section 3.3), and require to be in the range of 9% deviation, as mentioned in the previous section.

From Table 6.6 we can again observe that even though all test cases have the same nDoF ($8,192$ per variable), with increasing scheme order, the error in pressure becomes smaller, compared to the exact solution. Additionally, in all cases, the error is below 0.05% and therefore in excellent agreement with the exact solution. This behavior is also observed for the shock location. Only in the case of $O(16)$, this trend is not maintained due to the location of the wall. Here, the shock reflection occurs when the wall is in the middle area of an element. The distance between the integration



(a) (b)

Figure 6.9.: Comparison: Curves represent the numerical solution for $O(32)$ and 256 elements, for different over-integration factors S. The factor is varied from 1 to 4. (a) Normalized pressure of the reflected shock wave and (b) zoom-in of the reflected shock.

Table 6.6.: Comparison: Numerical solution for the interaction of the shock and the moving wall **with** an over-integration factor of 3.

Test case	p_3/p_2	Error in p_3/p_2 in [%]	$\Delta x \cdot 10^{-4}$
O(4), n2048	1.44845372	0.04538827	6.17999817
O(8), n1024	1.44847062	0.04422237	4.95925779
O(16), n512	1.44858607	0.03625527	5.56962798
O(32), n256	1.44871493	0.02736308	4.34888760

points is the largest, and therefore also the error in the modeling (see Section 5.3).

Furthermore, we need to emphasize that for the case $O(4)$, the wall is also located in the middle area of an element when the shock wave is reflected. However, noticeable is that we can overcome this unfavorable representation with a higher scheme order and achieve a much smaller error, such as in the case of $O(16)$. When comparing both cases, namely $O(4)$ and $O(16)$, the error in the shock location is 0.247% and 0.22% respectively, and the error in the pressure relation is 0.045% and 0.036% for $O(4)$ and $O(16)$. Thus with a higher order, we are able to reduce the modeling error further, even when the representation of the geometry (here

wall) is in an unfavorable location inside an element. The undershoot of the smallest scheme order $O(4)$ and the largest $O(32)$ are given with 1.66% and 2.56%. For the overshoot, a value of 0.10% in the case of $O(4)$ and 1.44% for $O(32)$ are obtained.

Comparison Considering both cases, the moving and non-moving wall, we were able to observe several findings, to better understand the modeling of the embedded method in our high-order numerical scheme. We are aware that high-order approximation poses some challenges for strong discontinuities, as in our case shocks. However we were able to demonstrate, that assimilable results can be obtained for the interaction of a shock wave with a moving and a non-moving wall. In the case the boundary moves (moving wall), the numerical scheme requires over-integration to approximate the geometry accordingly. It results in an even more accurate numerical solution, compared to the case of a non-moving wall (cf. Table 6.3 and Table 6.6). Further, in the case of the moving wall, our observation showed a less steep slope behind the shock when compared to the non-moving wall, which is due to the smearing of the solution as the wall moves. It needs to be highlighted that the same degrees of freedom are used for the different simulation setups to investigate the shock-wall interaction. Therefore it is expected to obtain the same quality of the solution. However, we were able to show that with increasing polynomial degree and decreasing mesh resolution (element count), the error of the solution can be significantly reduced. Thus, when using a high-order scheme for our simulations that deal with shocks, the solution is improved, and no penalty in terms of accuracy of the solution is expected.

6.3. Shock formation - Moving piston

In the previous section, we investigated a predefined shock wave that is reflected by a moving wall (cf. Section 6.2.2). In this section, we study the formation of a shock due to the abrupt motion of an embedded geometry. We consider a case that has an exact solution for comparison. The test case deals with a moving piston located inside the simulation domain, with the fluid initially being at rest. Due to the motion of the piston, a shock is formed ahead of the piston. Behind the piston, the formation of rarefaction can be observed. Since this is a one-dimensional problem and the exact solution neglects the viscosity, we consider a one-dimensional simulation domain, where we solve the compressible Euler equations. This problem is well suited to demonstrate that conservation is maintained. If

a violation of conservation is existing, no shock will form ahead of the piston, or it will have a wrong speed that deviates from the exact solution.

Test case description: The simulation domain is of length $l = 1.0$ unit length, with a piston of thickness 0.04 unit length located at $x = 0.4$ inside the simulation domain. For the pressure p_1 we assume a value of 1 and the density ρ_1 is set to 1. The velocity u_1 of the fluid is defined as 0.0. The piston moves throughout the simulation with Mach $M_p = 0.4$ until it reaches its predefined location after a simulation time of $t = 0.0008$. Again the time step is controlled by the CFL condition with a Courant factor of 0.2, and an over-integration factor of three is used to approximate the piston appropriately.
Initial conditions Initially the fluid velocity is 0.0, the pressure and the density are prescribed.
Boundary conditions At the left boundary, the primitive boundary condition, and along the right boundary, a subsonic outflow is defined, where the initial pressure is given.

Due to the sudden motion of the piston, the gas can be distinguished ahead of the piston into two regions, namely *Region 1* and *Region 2* (cf. Figure 6.10). Whereas *Region 1* lies to the right of the shock, here the gas remains in rest and is undisturbed with density, pressure, and velocity being ρ_1, p_1 and u_1, respectively. *Region 2*, being the region between the shock and the piston, the gas is co-moving with the moving piston; hence its velocity is $u_2 = v_p$ with v_p representing the velocity of the piston. The quantities in this region are given with density ρ_2, pressure p_2 and the velocity u_2. From Eq. (6.4) and Eq. (6.3) we can again determine the conditions downstream of the shock, while replacing Ma_s with the piston Mach number $M_p = v_p/c_1$ with c_1 being the downstream speed of sound. In order to obtain the exact solution for the rarefaction behind the piston, that is enclosed by the *head* and *tail* (cf. Figure 6.10 *Region Fan*), being the characteristics of the speed, we can determine the density ρ_{Fan}, pressure p_{Fan} and velocity u_{Fan} in that area with respect to the following equations [88]:

$$\rho_{Fan} = \rho_1 \left(\frac{2}{\gamma+1} + \frac{\gamma-1}{(\gamma+1)\cdot c_1} \left[u_1 - \frac{x}{t} \right] \right)^{\frac{2}{\gamma-1}} \tag{6.12a}$$

$$p_{Fan} = p_1 \left(\frac{2}{\gamma+1} + \frac{\gamma-1}{(\gamma+1)c_1} \cdot \left[u_1 - \frac{x}{t} \right] \right)^{\frac{2\gamma}{\gamma-1}} \tag{6.12b}$$

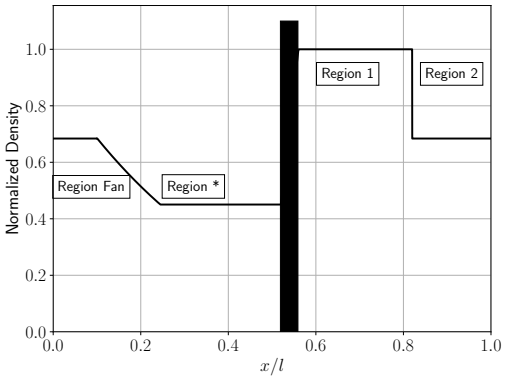


Figure 6.10.: Normalized density of the exact solution. Different regions are calculated according to different equations to determine the exact solution for the entire problem.

$$u_{Fan} = \frac{2}{\gamma+1}\left(c_1 + \frac{\gamma-1}{2}\cdot\left[u_1 - \frac{x}{t}\right]\right). \tag{6.12c}$$

Behind the rarefaction (cf. Figure 6.10 *Region**) a constant state is expected, which can be computed according to

$$\rho^* = \rho_1 \cdot \left(\frac{p^*}{p_1}\right)^{\frac{1}{\gamma}} \tag{6.13a}$$

$$p^* = p_1 \cdot \left(\frac{c^*}{c_1}\right)^{\frac{2\gamma}{\gamma-1}} \tag{6.13b}$$

$$u^* = u_1 - \left(\frac{2c_1}{\gamma-1}\cdot\left[\left(\frac{p^*}{p_1}\right)^{\frac{\gamma-1}{2\gamma}} - 1\right]\right). \tag{6.13c}$$

With $c^* = c_1 - \frac{v_p\cdot(\gamma-1)}{2}$ we can solve all equations in Eq. (6.13). The region *Region 1* in Figure 6.10 can again be determined with respect to the Rankine-Hugoniot conditions (cf. Section 6.2.1). More information and details about the equations can be found in [88]. For our investigation, we consider five cases, namely $O(2)$ with 4096 elements, $O(4)$ with 2048 elements, $O(8)$ with a total number of 1024 elements, $O(16)$ with 512 elements and lastly 0(32) with 256 elements. All test cases have the same

number of nDoF to allow for a comparative study. The Figure 6.11 depicts the normalized density, Figure 6.12 the normalized pressure and Figure 6.13 the normalized velocity for all configurations. It is apparent that with $O(2)$ and $O(4)$, the solution is not as precise as for $O(8)$, $O(16)$ or $O(32)$. For all quantities, the higher order can reproduce the characteristic properties of this test case accordingly, even though all test cases have the same number of nDoFs. Thus expecting all configurations to have comparable results. However, as shown in Section 6.2.1 and 6.2.2, we can achieve a much higher accuracy of the solution, even for non-smooth problems, with respect to the higher order, e.g., $O(16)$ or $O(32)$.

In Figure 6.11 we can observe strong oscillations between the piston interface and the shock, which are more dominant for this quantity than for pressure or velocity. Thus, density is a transport quantity and not penalized. Hence, oscillations appear for this quantity in higher strength when compared to velocity or pressure. However, the mentioned oscillations are also apparent for pressure and velocity. They are mainly due to two decisive factors. The first one is related to the shock, which moves throughout runtime from one element to another, while oscillations remain inside the elements, where the shock had been before (cf. Section 6.2.1). The second one is due to the motion of the piston (discontinuity in the masking function) and the representation of the masking function by non-equidistantly distributed integration points. As a result, oscillations appear, that remain in the solution and do not vanish as the discontinuity in the masking function moves. However, they stay small in amplitude. Oscillations induced by the shock can be addressed through an appropriate filter, in our case, the co-volume filter during runtime, as previously used in Section 6.2.1. However, using filters during runtime influences the solution. Some phenomena can be adequately captured, even though a filter is applied, e.g., the shock wave. On the other hand, filtering might have a more decisive influence on other phenomena, as it is applied everywhere in the domain with the same strength. In the case of the piston, our observations revealed that the shock position is accurately predicted when applying co-volume filtering throughout the simulation as in Section 6.2.1. However, the rarefaction, hence the information behind the piston, tends to be more sensitive. Therefore, filtering in this area results in a decrease in the numerical solution's accuracy compared to the exact solution.
Therefore, we consider no filtering during the simulation to be as close as possible to the exact solution but apply filtering during post-processing to remove numerical oscillations that are not of interest to the investigation. As a result, the remaining oscillations in the solution do not destroy

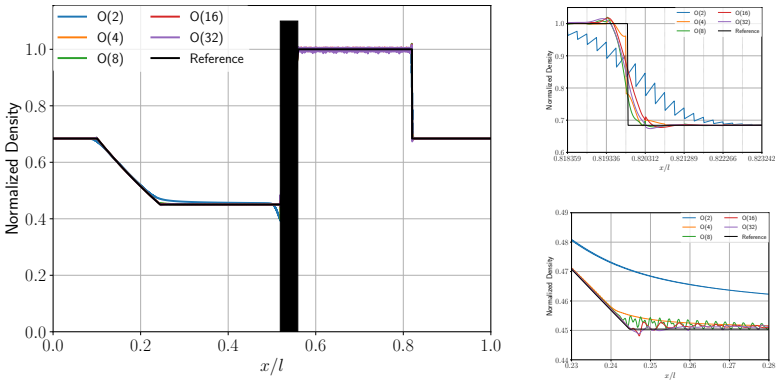


Figure 6.11.: Comparison of the numerical solution with the exact solution for the moving piston test case. The normalized density is presented for different scheme orders and mesh resolutions, resulting in the same nDoFs of 8,192 per variable. The upper right image depicts the shock position's zoom-in, and the lower left image is the zoom-in of the rarefaction behind the piston.

the solution and the characteristics of this test case (shock and rarefaction).

In Figure 6.11, Figure 6.12 and Figure 6.13 we can observe, that the rarefaction behind the piston can be solved more appropriately with increasing scheme order. For $O(2)$, the numerical solution is far from the exact solution, and for $O(4)$, the transition from linear decrease to constant state occurs not at the correct location. However, the highest scheme orders, namely $O16$ and $O(32)$, can reproduce this transition more precisely (cf., e.g., Figure 6.12 lower right). Furthermore, we can perceive the the benefit of the high-order scheme, as the smooth part of the solution (expansion fan), is precisely predicated with increasing scheme order. Additionally, the shock position can be captured with a higher scheme order more accurately, which is apparent for all quantities (e.g., cf. Figure 6.13 upper right).

We continue the investigation of this test case employing h and p refinement. The expectations are that with an increasing number of mesh elements (h) or increasing scheme order (p), thus a higher spatial resolu-

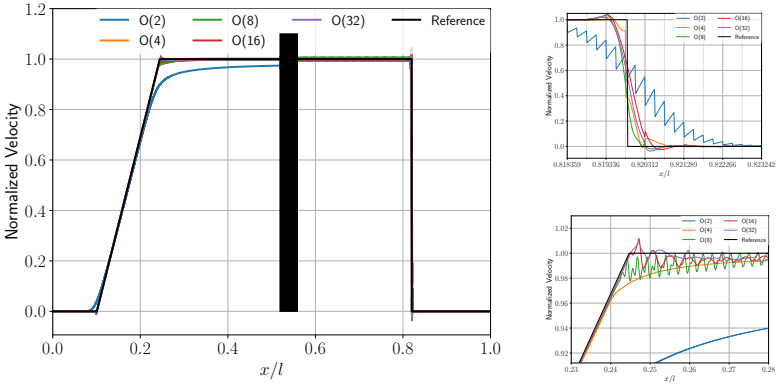


Figure 6.12.: Comparison of the numerical solution with the exact solution for the moving piston test case. The normalized velocity is presented for different scheme orders and mesh resolutions, resulting in the same nDoFs of 8,192 per variable. The upper right image depicts the shock position's zoom-in, and the lower left image is the zoom-in of the rarefaction behind the piston.

tion, the shock position, and the rarefaction area behind the piston can be captured more precisely when compared to the exact solution. In Figure 6.14 the numerical solution for this test case is given for the quantity pressure. Each curve represents one mesh refinement level, starting with 256 elements and ending with 4,096 elements for a predefined scheme order of $O(16)$. We can observe that with a higher mesh resolution, the shock position can be captured accordingly by a steeper gradient in pressure close to the shock (cf Figure 6.14 upper right). Behind the piston, the rarefaction is also precisely predicated. 512 elements can capture the transition from a smooth linear drop to a constant pressure value. With higher mesh resolution, the oscillations in the constant pressure area become smaller and closer to the exact solution (cf. Figure 6.14 lower right). Further, we can observe that the solution at the rarefaction is first below the expected solution with increasing mesh resolution. However, away from the transition area, the solution is slightly above the exact solution. This is due to the sudden motion of the piston and the oscillations of the polynomials used to represent the solution. This oscillatory behavior

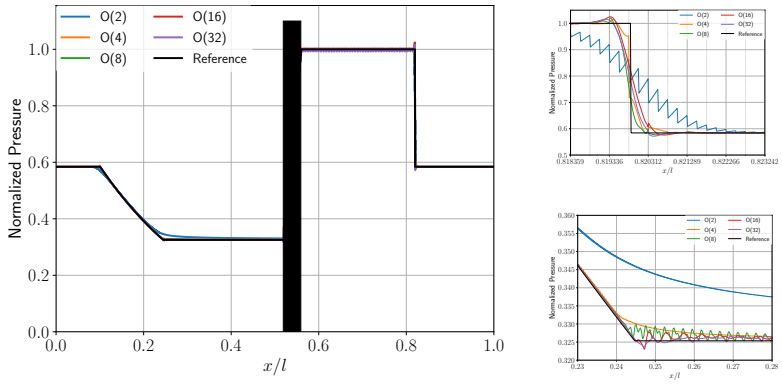


Figure 6.13.: Comparison of the numerical solution with the exact solution for the moving piston test case. The normalized pressure is presented for different scheme orders and mesh resolutions, resulting in the same nDoFs of 8,192 per variable. The upper right image depicts the shock position's zoom-in, and the lower left image is the zoom-in of the rarefaction behind the piston.

vanishes as soon as the system has leveled off. However, this oscillatory behavior is clearly around the exact solution, and eventually, the solution is in the mean in agreement with the expected solution.

For the p refinement study, we keep the computational mesh for all runs the same while changing the scheme order. We consider a mesh of 1,024 elements in total and the scheme orders of $O(2)$, $O(4)$, $O(8)$, $O(16)$ and $O(32)$. We again zoom in into the area where the shock is expected and the area behind the piston (rarefaction). In Figure 6.15 results are shown for the normalized pressure. For the scheme orders $O(8)$, $O(16)$, and $O(32)$, the numerical results are, as expected, from the previously shown results, the most accurate ones. The highest scheme order, namely $O(32)$, provides the most accurate shock position and rarefaction prediction. For $O(4)$, we can observe the different element interfaces and the jump of the representing polynomial, which agrees with the Discontinuous Galerkin scheme. The interfaces are indicated by the solid lines in Figure 6.14 (upper right).

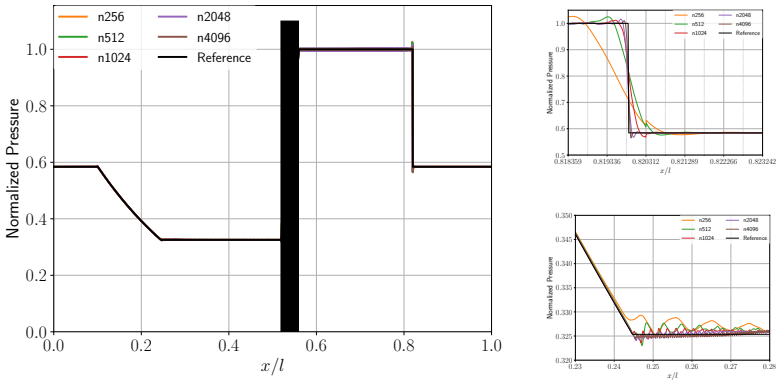


Figure 6.14.: Comparison of the numerical solution with the exact solution for the moving piston test case. The normalized pressure is presented for a h refinement with a scheme order of $O(16)$. Curves represent the different mesh resolutions using different element counts for the computational mesh. The upper right image depicts the shock position's zoom-in, and the lower left image is the zoom-in of the rarefaction behind the piston.

Form this test case; we can conclude that the shock formation and its position are in excellent agreement with the exact solution. Furthermore, we can ensure that conservation is maintained, as, in all quantities, the shock position and the rarefaction are correctly predicted. We also proved that the shock position could be captured with higher accuracy when using a higher scheme order for the discretization of the equations to be solved. Further, we illustrated that even though we are aware that high-order is not suitable for strong discontinuities, we demonstrated that utilizing high-order does not provide any disadvantages for our simulations. We can still capture the expected phenomena with higher scheme orders, with better quality, compared to lower scheme orders.

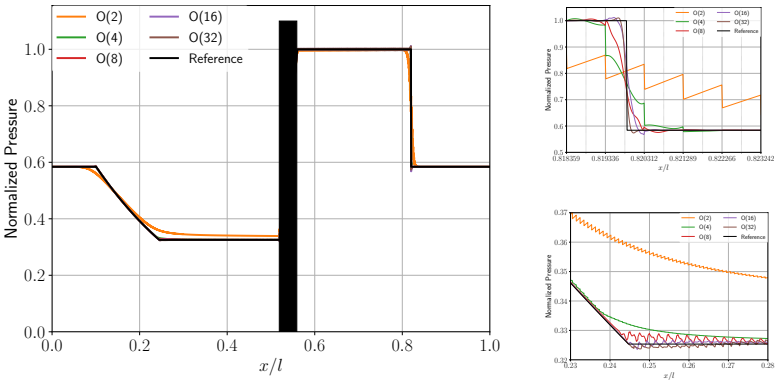


Figure 6.15.: Comparison of the numerical solution with the exact solution for the moving piston test case. The normalized pressure is shown for a p refinement with $1,024$ mesh elements and different scheme orders. Curves represent the solution for different scheme orders. The upper right image depicts the shock position's zoom-in, and the lower left image the zoom-in of the rarefaction behind the piston.

6.4. Curved boundary - Moving cylinder

In the previous sections, investigations were devoted to one-dimensional test cases. In this part, we extend our investigation to a problem in two-dimensional space. We examine the motion of a cylinder. The challenge in this test case is the geometrical surface, which is curved. The question that needs to be answered is, how the modeling method can handle curved moving boundaries. Therefore we first investigate the over-integration for this complex case. Afterward, we study a h refinement to demonstrate that the convergence of the solution can be achieved compared to the reference solution. It needs to be emphasized that the solutions obtained are compared to the non-moving case (reference). The non-moving geometry representation has been validated in [8, 34]. It allows ensuring comparability of both solutions as an exact solution is not available.

Test case description: We consider the compressible Euler equations for the simulations. The simulation domain is of size [1.0 $\times$ 1.0] unit

length, with a cylinder of diameter $d = 0.25$ unit lenght located inside the domain. The pressure and density are set to 1. In the case of the *non-moving* cylinder, the flow is streamed into the simulation domain with Mach 0.5. For the *moving* cylinder, the fluid enters the domain with Mach 0.25. The cylinder motion is prescribed with Mach 0.25, resulting in a relative Mach of 0.5. The time step is controlled by the CFL condition with a Courant factor of 0.25. The simulation time is set to 0.04; for the non-moving case, the cylinder is located at the same position, where the moving cylinder will end up at the end of the simulation time. Results are compared to each other at the end of the simulation time.
Initial conditions The velocity in x-direction, the pressure, and the density are defined.
Boundary conditions Along the right, upper and lower boundary, a subsonic outflow is prescribed. At the left boundary, a subsonic inflow is defined. For the moving cylinder, the velocity in positive x-direction is prescribed with Mach 0.25. Finally, for the non-moving cylinder, the fluid is streamed into the domain with Mach 0.5.

For the h refinement examination, we consider a scheme order of $O(16)$. Our study is carried out using different mesh resolutions, where the coarsest mesh has only 64 elements in total, hence 8 per spatial direction. The finest mesh contains 256×256 computational elements for this investigation. In order to prove that also for the two-dimensional case, an over-integration factor of 3 is sufficient; we run simulations using the moving case, where we consider in total 256 elements for the mesh and an over-integration factor of $1, 2, 3$ and 4. In Figure 6.16 the simulation results for the different over-integration factors are presented. It is evident that a factor of one does not yield the desired accuracy, rather an aliasing flaw, while a factor of 3 and 4 are in very good agreement with each other. Only at the maxima (stagnation point) we can observe a difference between the factor of 3 and 4, which is insignificantly small. The computational time, to complete the required simulation time, is for the over-integration factor $1, 2, 3$ and 4, 213.700 s, 613.035 s, 1325.00 s and 2381.00 s, respectively. Comparing the ratio between the computational effort and the accuracy of the obtained solution, we can conclude that an over-integration factor of 3 is sufficient. Therefore, we consider for all following simulations in this section an over-integration factor of 3. It needs to be emphasized that 360 measurement points are used around the cylinder geometry interface during post-processing to keep track of the pressure close to the surface of the cylinder. The measurement points are shifted by 0.5% away from the cylinder surface. This practice allows avoiding

the measurement inside the numerical boundary layer around the cylinder.

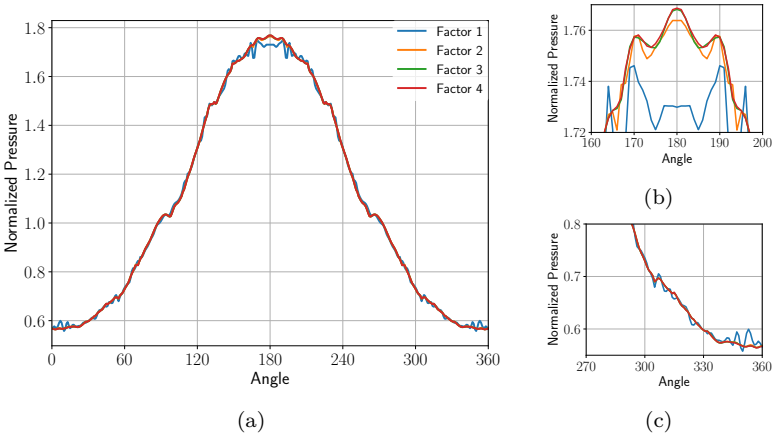


Figure 6.16.: Curves represent the solution for the different over-integration factors used for the numerical approximation. (a) Surface pressure of the cylinder geometry, (b) zoom-in into the stagnation area (maximum) and (c) zoom-in into the area behind the cylinder.

In Figure 6.17 and Figure 6.18 we examine a h refinement study to investigate the difference in the behavior of the moving cylinder when compared to the non-moving geometry, using a finer mesh. Therefore, we employ different refinement levels for the mesh. Starting from level $L5$ to $L10$, where the mesh of $L5$ contains 8×8 elements (coarsest), $L6$ has 16×16 elements, $L7$ has 32×32 elements, $L8$ contains 64×64 elements, level $L9$ in total 128×128 elements and level $L10$, 256×256 elements (finest). With increasing mesh resolution the pressure at the stagnation point decreases as the representation of the cylinder becomes more precise. As a result, the pressure curves of the moving and non-moving cylinder resemble each other properly, starting from level $L8$. Additionally, Figure 6.17b and Figure 6.18b depicts a zoom-in into the stagnation area in front of the cylinder. Here a high pressure value (maximum) is expected due to the compression of the fluid. The pressure curves become smoother with increasing resolution, too, as the geometry and the flow domain can be represented with higher accuracy. In Figure 6.17c and Figure 6.18c

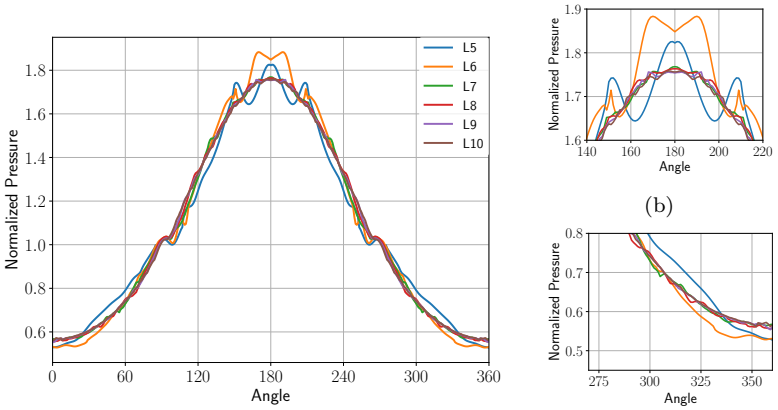


Figure 6.17.: Solution of the **moving** cylinder for different mesh refinement levels L over the angle (in degree). (a) Surface pressure of the cylinder geometry, (b) zoom-in into the stagnation area (maximum), and (c) zoom-in into the area behind the cylinder.

the area behind the cylinder is presented. Here the lowest value for the pressure is encountered due to the expansion of the fluid flow. From Figure 6.17b and Figure 6.18b it is apparent that with an increasing mesh resolution, the difference between the moving and non-moving cylinder decreases further. For $L5$ and $L6$, we can observe the highest deviation of the solution between the moving and non-moving geometry, which is due to the poor resolution. From $L7$ on, we can observe how close all solutions become. In Figure 6.19 only the solutions from $L8$ up to $L10$ are provided to analyze the moving and non-moving cylinder better.

Figure 6.19b and Figure 6.19c illustrate the pressure behavior in front of the cylinder and behind the cylinder, respectively. We can observe very dominant peaks for $L8$ for the non-moving cylinder around the stagnation area. Those peaks are not noticeable for the same case when the cylinder moves. This is due to the motion of the cylinder, resulting in an averaging effect coming from the different wall locations, which have a smoothening effect on the solution. We can perceive in all cases a smoother solution

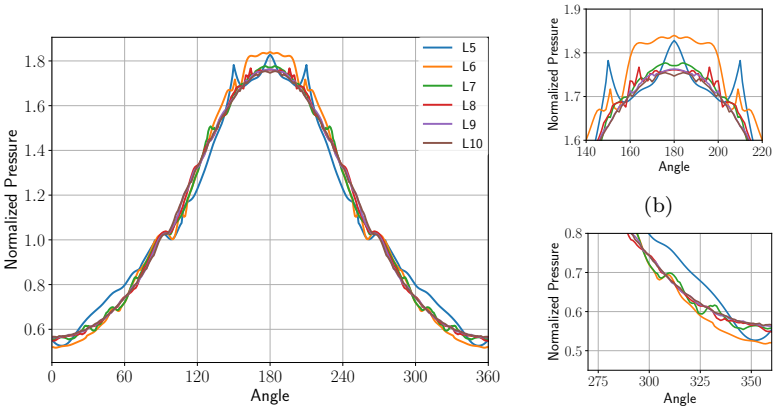


Figure 6.18.: Solution of the **non-moving** cylinder for different mesh refinement levels L over the angle (in degree). (a) Surface pressure of the cylinder geometry, (b) zoom-in into the stagnation area (maximum), and (c) zoom-in into the area behind the cylinder.

for the moving case compared to the non-moving one. Further, in all cases both, the moving and the non-moving cylinder resemble each other with increasing mesh resolution. In Figure 6.20 the solutions for $L10$ are presented. We can recognize a small deviation of the moving solution from the non-moving in the stagnation area (cf. Figure 6.20b). However, the difference is insignificantly small. Examining the solution behind the cylinder, we can also observe here an excellent agreement of both solutions (cf. Figure 6.20c). It needs to be pointed out that a small deviation in the solution can be expected, as in one case, the fluid is streamed with Mach 0.5, and the cylinder is at a predefined position (non-moving case). However, in the moving case, the flow is streamed into the domain with Mach 0.25, and the cylinder moves with Mach 0.25. Due to the motion, the cylinder is represented slightly differently, while in the case of the non-moving cylinder, the cylinder is all time represented with the same set of integration points. But again, even though the representation differs for the moving cylinder in each time step, the solution of the non-moving and moving case is in excellent agreement with each other and demonstrates

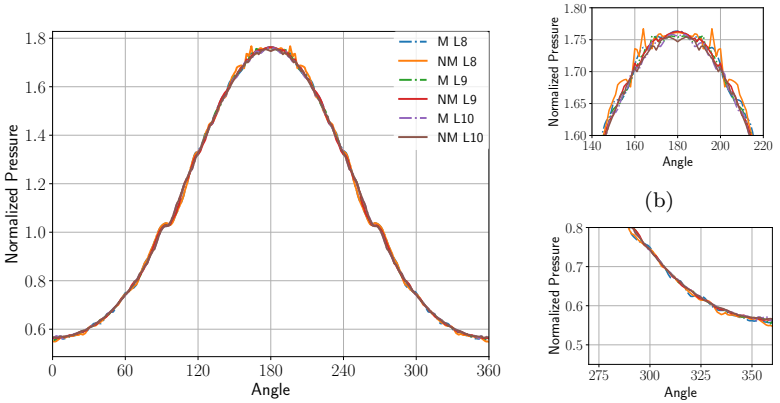


Figure 6.19.: Comparison of the solution for the moving M and non-moving cylinder NM with the mesh refinement levels $L8$ up to $L10$ over the angle (in degree). (a) Surface pressure of the cylinder geometry, (b) zoom-in into the stagnation area (maximum), and (c) zoom-in into the area behind the cylinder.

that both converge towards the same solution with finer mesh resolution.

In Figure 6.21 the absolute normalized difference between the moving and non-moving cylinder for the levels $L8$, $L9$, and $L10$ is shown. With increasing refinement levels, the difference between the curves becomes smaller. The most significant difference between the moving and non-moving cylinder can be found between an angle of 120 and 240, thus the front area of the cylinder. That is again due to the different representation of the moving cylinder and the smearing of the numerical solution. The difference is still below 1% in the case of $L10$, resulting in an insignificantly slight deviation. Furthermore, the decrease of the difference between the moving and non-moving cylinder with increasing mesh resolution indicates that both cases converge towards the same solution.

In Figure 6.22 the pressure is shown in the background. The pressure contours are colored by the velocity magnitude for the case of $L10$. On the left, the solution for the non-moving, and on the right, the moving cylinder is shown. The pressure contour lines (white/ gray) of both solutions are

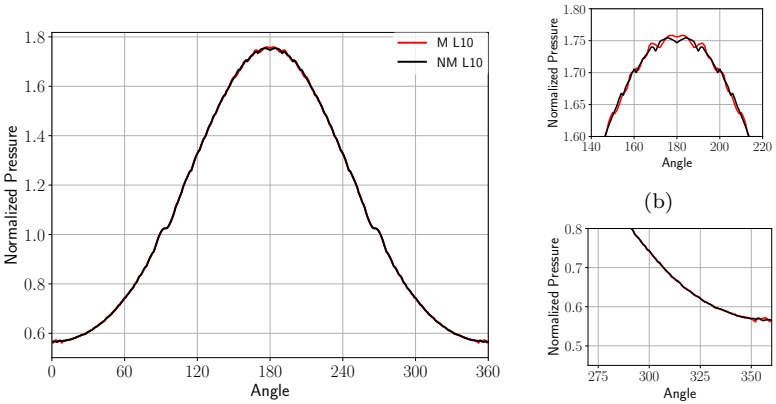


Figure 6.20.: Comparison of the solution for the moving M and non-moving cylinder NM with a mesh refinement level of $L10$ over the angle (in degree). (a) Surface pressure of the cylinder geometry, (b) zoom-in into the stagnation area (maximum), and (c) zoom-in into the area behind the cylinder.

in excellent agreement. Only in tiny areas, e.g., behind the cylinder, the solution slightly differs (cf. Figure 6.21), yet, this is insignificantly small. The color distribution of the contour lines are, as expected, higher in the non-moving case (left) compared to the moving cylinder, as the fluid is streamed into the domain in one case with a two times higher Mach number than in the other case.

According to Bernoulli's law, an increase in pressure results in the decrease of the velocity in that area; consequently, we expect everywhere, where the pressure has a high value, a low velocity, and vice versa. This behavior can be well recognized in Figure 6.23. The velocity at the stagnation point is the lowest (≈ 0.0), while the pressure has a maximum. For both cases, namely the non-moving cylinder (V - NM) and the moving case (V - M) this behavior can be recognized. Both curves match very well and overlap. It needs to be noticed that the velocity magnitude shown in Figure 6.23 is relative to the motion of the geometry.

With this test case, we were able to confirm that the utilized penal-

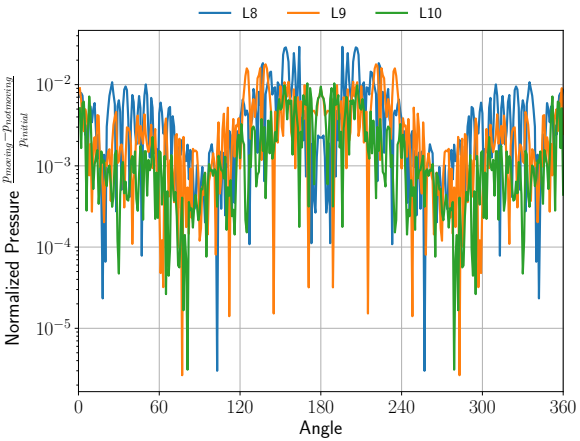


Figure 6.21.: Absolute difference of the numerical solutions for the moving and non-moving cylinder. For the mesh resolutions, $L8$, $L9$, and $L10$.

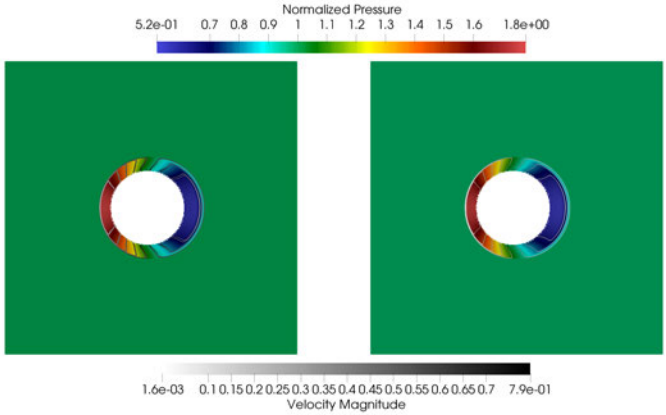


Figure 6.22.: Comparison of the non-moving cylinder (left) and the moving cylinder (right) for the mesh $L10$. The pressure is shown in the background, the pressure contours around the cylinder are colored by the velocity magnitude.

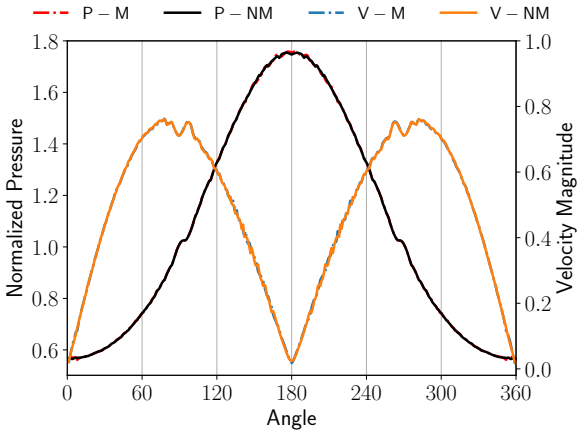


Figure 6.23.: Comparison of the non-moving cylinder (NM) and the moving cylinder (M) for the mesh $L10$. The pressure is shown for the non-moving (P - NM) and the moving case (P - M). The relative velocity is presented with V - NM and V - M for the non-moving and moving cylinder, respectively.

ization method is well suited for curved geometries. However, due to the motion of the geometry, it is represented throughout the simulation with a different set of integration points. Therefore the solution slightly differs from the non-moving geometry. However, we demonstrated that both cases converge towards the same solution, as shown for the different mesh resolutions.

6.5. Sharp boundary - Supersonic moving wedge

In the second two-dimensional test case, we examine a moving wedge that has been extensively studied in literature and has an exact solution. The purpose of this test case is to confirm that sharp geometrical surfaces can be accordingly modeled when deploying the Brinkman penalization method. The test case is about a wedge located close to the right boundary and moves with a predefined supersonic speed towards the left boundary. The fluid is initially at rest. Due to the abrupt motion of the wedge, an oblique shock wave is formed at its nose. The angle of the shock can be

analytically determined and compared to the numerical solution. A time series of the simulation results for the different configurations are presented in Section A.3.

Test case description: The simulation domain is shown in Figure 6.24. The length of the domain L is 10 m, and the height H is defined to be 4 m. In order to ensure the correctness of the simulation results, the exact

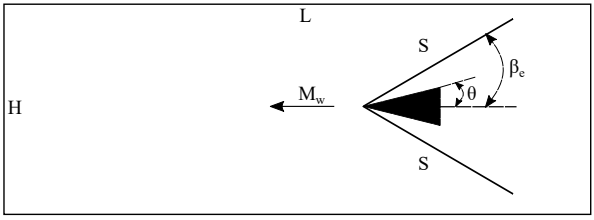


Figure 6.24.: Sketch of the simulation domain with a moving wedge.

solution for the shock angle β_e is considered. It can be computed according to [51]:

$$\tan\theta = \frac{2}{\tan\beta_e}\frac{{M_w}^2\sin^2\beta_e - 1}{{M_w}^2(\gamma + \cos(2\beta_e)) + 2} \tag{6.14}$$

With θ being the deflection angle, β_e the shock angle, γ the isentropic coefficient, and M_w the Mach number with which the wedge moves. The deflection angle is set to be $\theta = 15^\circ$ resulting in an exact shock angle of $\beta_e = 45.433^\circ$ with respect to $\gamma = 1.4$, the height of the wedge ($h_w = 1.0\ m$) and the wedge Mach $M_w = 2.0$. The investigation is carried out with different scheme orders such as $O(5)$, $O(6)$, $O(8)$ and $O(10)$. The time is controlled by the CFL condition with a Courant factor of 0.3. The final simulation time is determined to be 0.2 s. The computational mesh is discretized in all cases with 163,840 elements, resulting in an element size of 0.015625 m. In order to stabilize the simulation and avoid the domination of numerical oscillations, a weak co-volume filter with a filtering order of $O(28)$ is used throughout the computation. Finally, the shock angle is computed and compared with the exact solution at the end of the simulation.

Initial conditions Initially, the state of the simulation domain is set to be $[\rho, u, v, p] = [1.4\ kg/m^3,\ 0.0,\ 0.0,\ 400\ Pa]$.

Boundary conditions Along the upper and lower boundary slip walls are prescribed. At the left boundary an inflow, and at the right boundary, a

supersonic outflow is defined.

Before discussing the numerical results in this section, it is worth to mention, that the numerical shock angles were manually measured by fitting two lines towards the computed oblique shock. Hence, the presented numerical results of the shock angles are subject to sampling error. In order to provide a better comparison of the numerical and analytical solutions, lines are included in the figures, reflecting the exact solution. In Figure 6.25 and Figure 6.26 simulation results for the different runs are exhibited for the last time step (0.2 s). Red lines at the front of the wedge indicate the exact shock angle from the analytical solution. All figures provide very accurate results that improve with increasing scheme order. Not only the shock angle of the numerical solution β_n converges towards the exact shock angle β_e, but also further physical features of this test case can be captured, e.g., smaller vortices. An important observation is the sharpness of the shock that accordingly improves with increasing scheme order from Figure 6.25a to Figure 6.26b. In Figure 6.25a a slight deviation of the numerically obtained shock angle from the exact solution can be recognized. However, this deviation is less than 1.72%, with β_n being the numerical solution, that is 46.1233°. Further, flow features such as the shock-wall interaction and the interaction of the shock with vortices in the middle of the domain are captured, respectively. The Prandtl-Meyer expansion waves are physically resolved at the rear edges of the wedge geometry and can be found in all investigated test cases.

From Figure 6.25a to Figure 6.26b we can notice the oblique shock, which can be precisely predicted, but also the sharpness of the shock, which is more accurately represented with increasing scheme order. The solution of $O(6)$ (cf. Figure 6.25b) can predict the shock angle with an angle of β_n 45.7910° very close to the exact solution, with a deviation of less than 1%. In the case a scheme order of $O(8)$ (cf. Figure 6.26a) is used, the shock angle β_n is 45.3590°, with a difference of 0.03% from the exact solution. For the case a scheme order of $O(10)$ is used, the oblique shock angle can be perfectly predicted and is in excellent agreement with the exact solution, with a value of $\beta_n = 45.3445°$. Other than the sharpness of the shock, all numerical solutions provide very detailed insights into the occurring physics. E.g., the reflection of the shock at the wall, small scales that become more apparent with increasing scheme order, and the instability behind the wedge, which first resembles a ray but breaks into small scales with increasing scheme order. Please keep in mind that the inviscid Euler equations are solved here. Thus the physical viscosity is

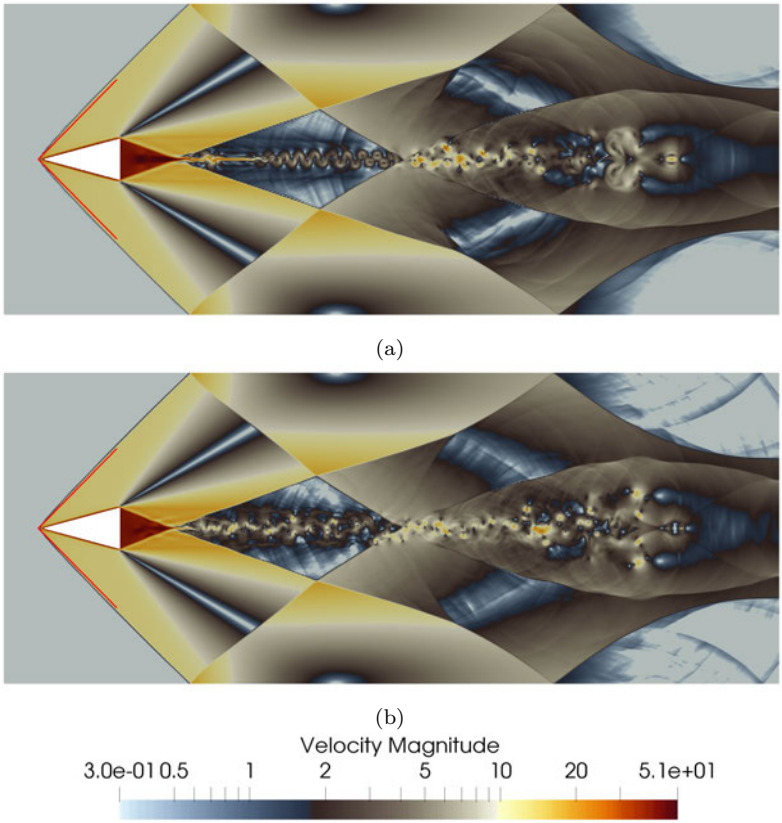


Figure 6.25.: Comparison: Supersonic flow over a supersonic translating wedge for different scheme orders and $M_s(2)$–$\theta(15^\circ)$–β_e (45.344°). Red lines indicate the exact solution. In (a) the scheme order of $O(5)$ with $\beta_n(46.1233^\circ)$ and in (b) the scheme order of $O(6)$ with $\beta_n(45.7910^\circ)$, is shown.

neglected, resulting in a Reynolds number approaching infinity. However, the numerical viscosity is present, which becomes smaller with increasing scheme order, thus a higher order approximation of the solution. In the case of $O(6)$, $O(8)$, and $O(10)$, the numerical viscosity is decreased due to the high-order approximation. Consequently, the flow becomes more tur-

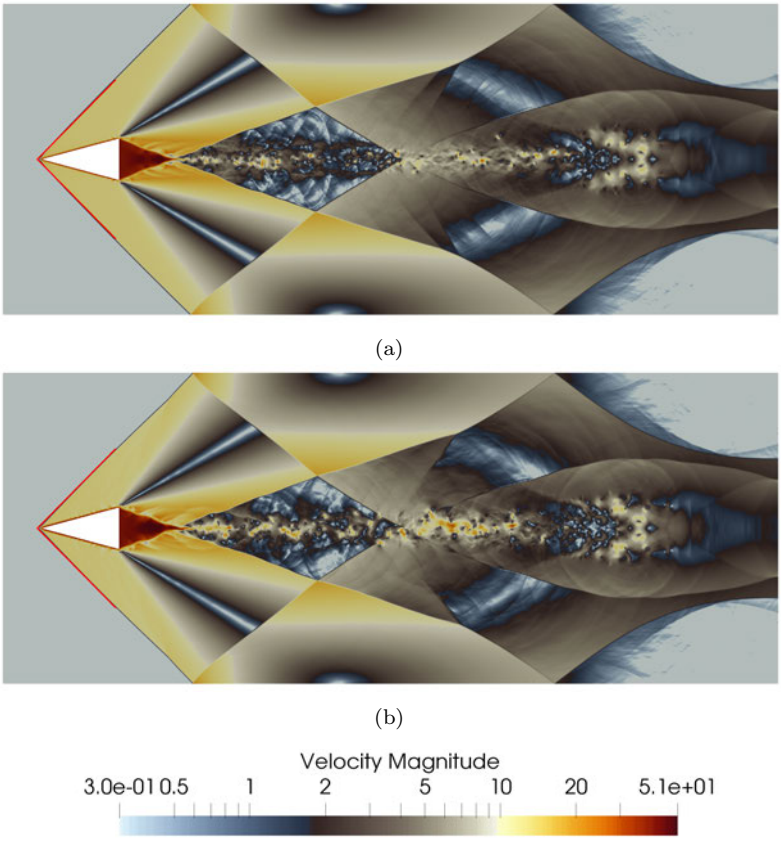


Figure 6.26.: Comparison: Supersonic flow over a supersonic translating wedge for different scheme orders and $M_s(2)$—$\theta(15°)$—β_e (45.344°). Red lines indicate the exact solution. In (a) the scheme order of $O(8)$ with $\beta_n(45.3590°)$ and in (b) the scheme order $O(10)$ with $\beta_n(45.3445°$, is presented.

bulent (chaotic), resulting in a solution with more small-scale phenomena in the wake flow when compared to $O(5)$.

This test case confirmed that the underlying high-order scheme can ac-

cordingly treat supersonic motions of modeled geometries and handle sharp boundaries that are modeled with high accuracy. The modeling approach allows for equivalent results when compared to the exact solution. We demonstrated two crucial challenges in the numerical simulation our scheme is capable of: **(i)** The supersonic motion of the geometry and **(ii)** the modeling of sharp geometries. They are challenging in the realization processes and prone to numerical instabilities. However, those cases are of relevance for many engineering applications and need to be realized appropriately.

More results and investigations on the embedded method can be found in Chapter 9 and in [77].

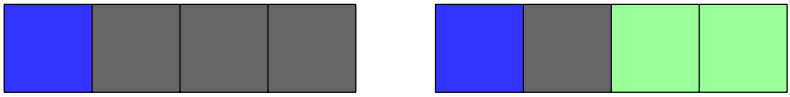
6.6. Reduced computation inside the geometry

As mentioned in Section 5.1 the major disadvantage of the embedded method is the computation inside the geometry, which is not desired. Due to the penalization method, the fluid dynamic equations are computed inside the geometry, too. This unnecessary computation leads to a waste of computational resources. It would be beneficial to cut out those elements from the computational domain to reduce the computational effort. However, this requires changes to the mesh in the first place, which impedes the parallel execution. Thus, It may involve communication to maintain proper adjacency and may not be easily convertible, as it also counters some of the benefits of this method, especially in the context of moving geometries. A simple solution in high-order discretization is to reduce the scheme order locally while maintaining the mesh. This procedure can be achieved by reducing the number of modes, a so-called mode reduction.

One of the most compute-intensive operations in high-order Discontinuous Galerkin method is the physical flux computation. Consequently, tackling this operation can decrease the computational cost. Only the integral mean, which is the zero mode in a Legendre expansion, is computed in elements covered by the geometry. Thus only the zero mode is used to compute the physical flux for those elements instead of calculating all higher modes. The criteria based on which the solver decides to apply this feature is the masking function χ. The solver distinguishes between elements that are inside the geometry and those that are outside (see Eq. (5.2)). The final decision is made by using the state of the element and the neighbors. If the geometry covers all neighboring elements, the

fluxes can be computed according to the mode reduction feature for the specific element. Examining the neighbors ensures that the accuracy of the solution is maintained in the boundary layer region inside the porous material, and the computation is consistent.

In Figure 6.27 a small test case is shown to demonstrate when this feature is applied. The simulation domain is discretized with four elements, one fluid element (blue) and three elements (gray) that are covered by a geometry (see Figure 6.27a). The mode reduction can only be applied to the last two elements, which are covered by the geometry (cf. Figure 6.27b in green) as they fulfill all criteria. In contrast, the first element in the geometry, the physical flux computation, needs to be done concerning all higher modes. This practice is of importance, as the boundary layer inside the geometry has to be resolved appropriately. A decrease in the polynomial degree in this region would badly affect the wall modeling and cause instabilities during runtime. The question arises: how much can the computational cost



(a) (b)

Figure 6.27.: Mode reduction feature for elements inside the geometry reduces the computational cost for elements covered by the geometry. (a) Computation of the physical flux with a high polynomial degree in the entire domain. The simulation domain contains one fluid element (blue) and three elements covered by the geometry (gray). (b) Use of mode reduction, resulting in two elements that can be reduced in computation (green) due to the reduced approximation of the physical flux.

be reduced when using this feature and if the physical result is still valid. Further, we need to ensure that physical results are still valid and in agreement with the case, where no mode reduction is applied. Therefore, further investigation is conducted with a simple two-dimensional test case, a Gaussian pulse, that travels from an initial position towards a wall modeled as a porous material. Then, the pulse is reflected and moves to its original position. The inviscid compressible Euler equations are solved for this test case.

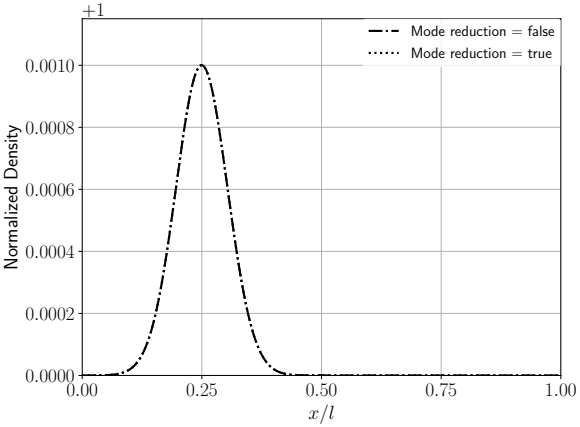


Figure 6.28.: Comparison: Solution of the reflected pulse for the density, normalized by the background density ρ_B. Curves represent the numerical solution, when mode reduction is deactivated (false) and activated (true).

Test case description: The simulation domain has a length of 1 unit length. From 0.5 to 1.0, the domain is covered by the modeled wall. The non-dimensionalized value for the background pressure is $1/\gamma$, with $\gamma = 1.4$ being the isothermal coefficient and for the background density is 1, resulting in a value of $c = 1$ for the speed of sound. A scheme order of $O(64)$ is used, while the mesh contains 64 elements. The pulse is defined for all quantities (density, pressure, and velocity) with an amplitude of 0.001 (cf. Section 6.1). It is initially located at $x = 0.25$. The time step size is controlled by the CFL condition with a Courant factor of 0.25.
Initial conditions Initially, for all quantities, the pulse is prescribed as for the test case in Section 6.1.
Boundary conditions Along the left boundary, the primitive variables (density, pressure, and velocity) are prescribed, and on the right, a subsonic outflow is defined.

In Figure 6.28 the normalized density is depicted over the length of the domain. The figure illustrates the solution after reflection at the modeled wall for the case, when mode reduction is activated (true) and in the case, it is not used (false). It is apparent that both results are in agreement, and

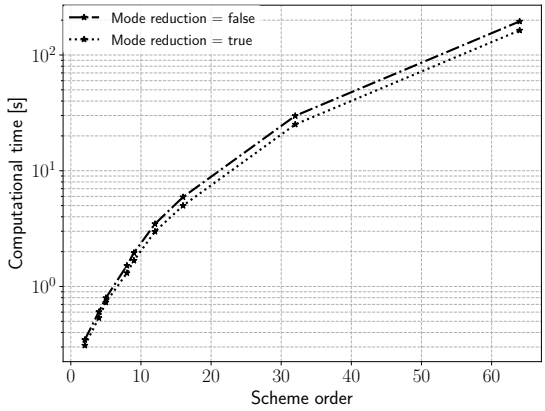


Figure 6.29.: Comparison: Computational time for 100 iterations, when the feature mode reduction is not activated (dashed line) and when the solver makes use of it (dotted line).

no disadvantages in terms of loss of accuracy of the solution need to be expected when applying the mode reduction feature. In terms of reduced compute time, we compare the computational time of 100 iterations when mode reduction is applied and when it is not used. The compute time investigated here is the pure computational time, hence without initialization time. In Figure 6.29 the computational time over the scheme order is shown. The test case is the same as described above. However, the scheme order is increased, starting from $O(2)$ up to $O(64)$. The simulation was sequentially executed to avoid load imbalances. The mode reduction feature implies load imbalances when the feature is used during runtime. From the shown figure, we can observe that with increasing scheme order, the savings in the computational time increases as well when the feature is applied. The computational time can be reduced by 10.67% up to 16.37% for the scheme orders of $O(2)$ to $O(64)$. However, it is crucial to keep in mind that the reduction in computational time depends on the number of elements covered by the geometry and the scheme order used. This feature permits the application area of the embedded method for larger geometries that can cover several elements in the computational mesh. Thus, this method can decrease the computational effort for those applications to reduce unnecessary computation and allow for more feasible numerical simulations.

6.7. Scalability and computational cost of the embedded method

This section investigates the scalability and the additional computational effort required when considering moving geometries in the simulation domain. We first examine the scalability of the solver when incorporating the penalization terms for a test case in two-dimensional space. We expect that the scalability of the solver is maintained, as the communication pattern remains unchanged. Afterward, we mainly focus on the additional computational cost due to the evaluation of the masking function throughout the simulation. Therefore a simple test case in three-dimensional space is used, where the compressible viscous Navier-Stokes equations and the compressible inviscid Euler equations are solved. This investigation aims to provide an estimation for the additional costs due to the presence of moving geometries. The expectation is that the evaluation of χ has a minor impact on the computational effort when solving the Navier-Stokes equations. However, when solving the Euler equations, we anticipate a more significant impact on the computational cost.

6.7.1. Scalability of the embedded method

In this part, we examine the scalability of the embedded method in our numerical scheme. Due to the motion of geometries, the masking function χ has to be evaluated throughout the simulation. It might be seen as a bottleneck and might influence the scalability of the numerical simulation when geometries are present in the simulation domain and can even move.

To investigate the scalability of the solver, in the case geometries are present in the simulation domain, a strong scaling measurement is conducted. The same test case as in Section 6.5 is utilized, with a computational mesh that contains $163,840$ elements. The scheme order is $O(8)$, and the inviscid Euler equations are solved. Simulations are executed on Supermuc-NG, with up to 320 nodes, resulting in $15,360$ cores in total. The simulation is completed after 100 iterations, where the initialization time is neglected and only the computational time is measured.

In Figure 6.30 the measurement for the mentioned test case is shown. The dotted line represents the ideal scaling for this problem, and the dashed line presents the actual compute time per iteration. The scaling behavior is in excellent agreement with the ideal scaling for the investigated node count. For the scaling measurement, the number of allocated nodes

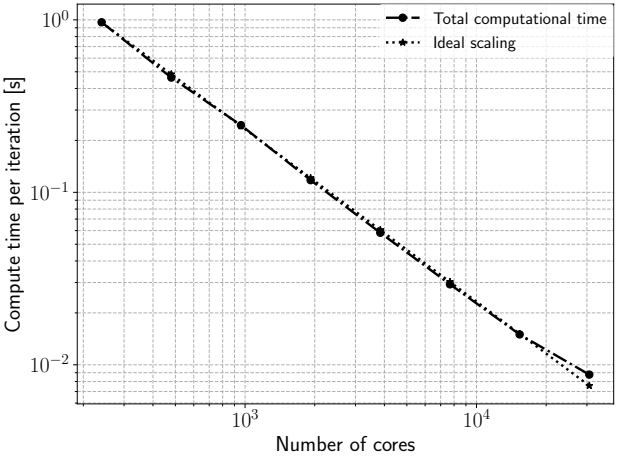


Figure 6.30.: Strong scaling measurement of the solver for a flow simulation with a moving wedge (cf. Section 6.5), with up to 320 nodes (48 cores per node) on Supermuc-NG computing system. The dotted line represents the ideal scaling and the dashed line the actual compute time required per iteration.

was doubled for each subsequent data point in Figure 6.30, starting from 5 nodes up to 320 nodes. The excellent performance of the solver was expected, as it is highly parallel, and the introduced method to model geometries does not negatively affect the solver's scalability. Furthermore, as previously mentioned, the high-order Discontinuous Galerkin method allows for little communication between the computational elements, as only the flux information needs to be communicated between the element faces. Nevertheless, the information inside the element is tight and remains local. Therefore the introduced modeling terms do not change the communication pattern, allowing to maintain the scalability of the solver.

6.7.2. Computational cost of the embedded method

In the previous section, we illustrated the scalability of the solver, which is maintained, even when incorporating the Brinkman penalization terms, to model geometries. An important question that needs to be answered here is how the computational effort is affected by moving geometries in

the simulation domain. To answer this question, we carry out several simulations, where we once solve the fluid dynamic equations without the Brinkman penalization terms and once where those terms are incorporated. As already mentioned, our study includes the compressible Navier-Stokes equations and Euler equations since those equations are primarily used in this work.

To model arbitrary complex geometries, the solver requires a list of points for its interface identification. Each list of points is evaluated at each integration point. This practice is required to obtain the state of the masking function χ (inside or outside of the polygon (geometry)). The more complex the geometry becomes, the more points are necessary to define the geometrical surface to represent, e.g., corners or curved areas. Consequentially, our investigation is mainly dedicated to the variation of the number of vertices for the polygon (geometry) computation. The primary purpose of this work is to simulate flows with complex and moving geometries.

Test case description: The simulation domain is of size $[4 \times 4 \times 4]$ unit length with an element length of 0.125 unit length, resulting in $32,768$ computational elements in total. The three-dimensional domain is once solved by the compressible Navier-Stokes equations with and once without the Brinkman penalization terms. The same procedure is also done for the inviscid Euler equations. The fluid dynamic equations are discretized with a scheme order of $O(7)$. The time is controlled by the CFL condition with a Courant factor of 0.3. The solver needs to identify the state of the masking function for all integration points $(11,239,424)$ inside the domain. We ensure not to have different types of elements that might result in a different workload in the simulation domain. If the modeling terms are considered, we introduce only one vertex, which serves as a moving geometry. The velocity of the geometry (vertex) is set to be 0.01 only in x-direction. This practice is essentially needed, as the solver has to identify the position of the geometry throughout the simulation time, which influences the computational time further. The examined test case is close to the large simulation conducted in this work.
Initial conditions The velocity is defined to be 0.0. The pressure and density are defined to be 1.
Boundary conditions All boundaries are defined to be periodic; thus, the domain has an infinite length in all spatial directions.

In Figure 6.31 the strong scaling measurement for all four test cases

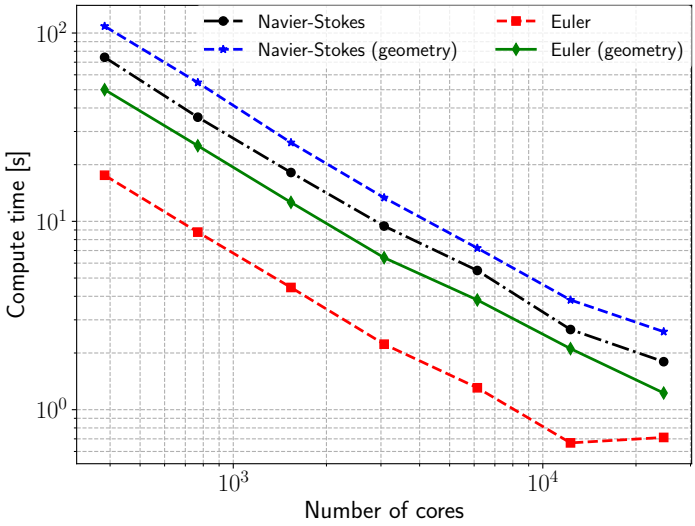


Figure 6.31.: Strong scaling measurement and comparison of the computational time between the cases, when no geometry is defined (Navier-Stokes and Euler), and when geometry is present (Navier-Stokes (geometry) and Euler(geometry)). Scaling measurements were conducted on Supermuc-NG with 8 nodes (384 cores) up to 512 nodes (24,576).

is presented. The strong scaling investigation was conducted on Supermuc-NG, from 8 compute nodes (384 cores) to 512 compute nodes (24,576 cores). The number of compute nodes is doubled for the subsequent runs. Thus, all shown compute times are always for 100 iterations. Examining the red curve, representing the case where the original Euler equations are solved and no penalization terms considered, we can observe how the computational effort reduces with increasing node count. Though for the last data point, the curve is flat and does not yield any further reduction in compute time. At this point, communication dominates the computational effort, and no further benefits can be achieved when using more compute nodes. Each core receives in the case of 512 nodes 1.33 elements. Since each core requires at least one complete element for the computation, some cores receive one while others process two. Furthermore, the computa-

tional cost, when compared to the other three curves, is the lowest, as **(i)** viscous terms are neglected for these equations and **(ii)** no Brinkman penalization terms are incorporated. The green curve illustrates the case when the modeling terms are computed as well. The computational cost increases considerably, yet, we can observe how well the computational cost is reduced with increasing node counts. When comparing the red curve (Euler) and the green curve (Euler (geometry)), our observation provides a better scaling for the green curve when compared to the red curve for the last node count. This behavior is due to the increased computation inside the elements, as the masking function needs to be evaluated. Hence the computational effort dominates compared to the communication time.

The black (Navier-Stokes) and blue (Navier-Stokes (geometry)) curves manifest similar behavior. Focusing on the black curve, we notice that the curve does not become flat for the last data point compared to the red curve (Euler). This behavior is due to the expensive gradient computation for these equations that dominate the computation. Therefore, even for the last data point, a reduction in computational time can be observed when increasing the number of compute nodes. Furthermore, we notice that the introduced modeling terms do not significantly impact the computational cost when comparing the black and blue curves. The gradient computation is still the dominating part of these equations, which is advantageous compared to the Euler equations. The physical viscosity needs to be taken into account for simulations of real-world problems, and incorporating the Brinkman terms in the Navier equations does not influence the computation as much as it does for the Euler equations.

Comparing the scaling and the compute time each test case requires to attain the predefined 100 iterations for the simulation, we need to emphasize that: **(i)** The original Euler equations are the cheapest (red curve), followed by **(ii)** the Euler equations with penalization terms (green curve) and **(iii)** the original Navier-Stokes equations (black curve), and lastly **(iv)** the Navier-Stokes equations with Brinkman terms (blue curve). Keeping this order in mind, we obtain from Figure 6.31 the ratio between Euler and Euler (geometry), of about 3 and between Navier-Stokes and Navier-Stokes (geometry), of approximately 1.5. We can also give the relation in computational cost between the Navier-Stokes and Euler case, which is nearly 3.5 times more expensive than the Euler equations. To conclude the outcome of this examination, we obtain the following ratio when comparing all four test cases: Euler: Euler (geometry): Navier-Stokes: Navier-Stokes (geometry) = 1 : 3: 3.5: 5. Remember that this ratio is

only valid for this particular test case with the used scheme order and the number of mesh elements. However, the mentioned ordering will stay valid for different orders and mesh resolutions.

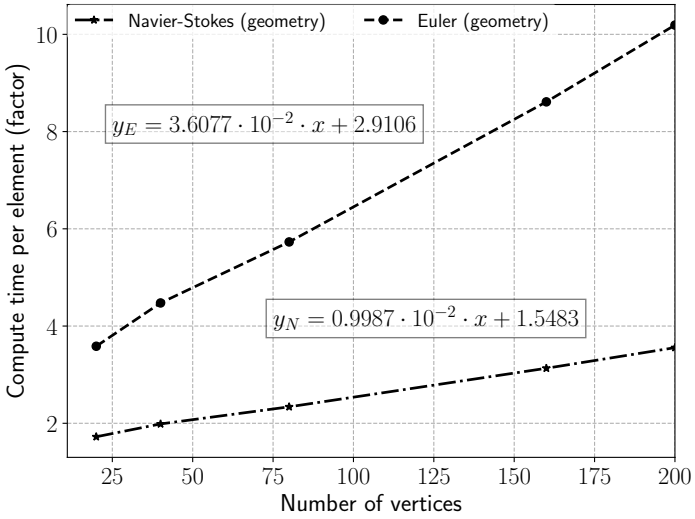


Figure 6.32.: Computational time per element (factor) for simulations with geometries compared to simulations without geometries. Vertices represent the geometrical surface and can vary in number, depending on the complexity of the geometries. Linear fits (y_N and y_E) show how the computational effort increases with an increasing number of vertices for this particular test case.

In this work, we are particularly interested in the modeling of arbitrary complex geometries that can move. To model such geometries, which are polygons, the solver requires a list of points used to define the geometrical boundaries (cf. Section 5.6). Thus, the computational effort is influenced by the number of points (vertices) provided to the solver. Therefore depending on the number of vertices, the computation at each integration point also increases when evaluating the masking function. The solver has to do the following procedure at each integration point: **(i)** Determining the interface of the polygon, and **(ii)** determining the state of the masking

Table 6.7.: Computational time per element for different number of vertices to represent the geometrical surface. In (a), the compute time per element for the compressible viscous Navier-Stokes equations and (b) for the compressible inviscid Euler equations are presented. The last column in (a) and (b) present the compute time per element for the respective equations when no geometry is present (reference).

Vertices	Compute time per element [s]
20	1.2482
40	1.4395
80	1.6946
160	2.2693
200	2.5764
–	0.7246

(a)

Vertices	Compute time per element [s]
20	0.7254
40	0.9056
80	1.15962
160	1.7427
200	2.0622
–	0.2024

(b)

function χ at the integration point. To give an overview on how the number of vertices influences the computational cost, we consider the test case configuration aforementioned, namely the Navier-Stokes (geometry) and the Euler (geometry) case (cf. Figure 6.31). For comparison, the cases without geometries (Navier-Stokes and Euler) are used. In Figure 6.32 the computational effort against the number of vertices is presented. As previously shown, the presence of geometries in the computational domain is more noticeable for the Euler equations from the computational perspective compared to the case where the Navier-Stokes equations are solved (cf. Figure 6.31). This trend can be further observed when increasing the number of vertices to represent the geometrical surface. When solving the Navier-Stokes equations, the computation roughly increases by a factor of 1.55, determined through the equation y_N (cf. Figure 6.32). For the Euler equations the computational cost has a constant factor of roughly 2.91 per element (cf. Figure 6.32), noticeable from equation y_E in the shown figure, that increases by nearly 1% per vertex (slope). In Table 6.7 the computational cost per element is given, used to calculate the factor, with which the computational effort increases, illustrated in Figure 6.32. In Table 6.8a the compute time per element for the Navier-Stokes equations is shown, while in Table 6.8b the respective times for the Euler equations

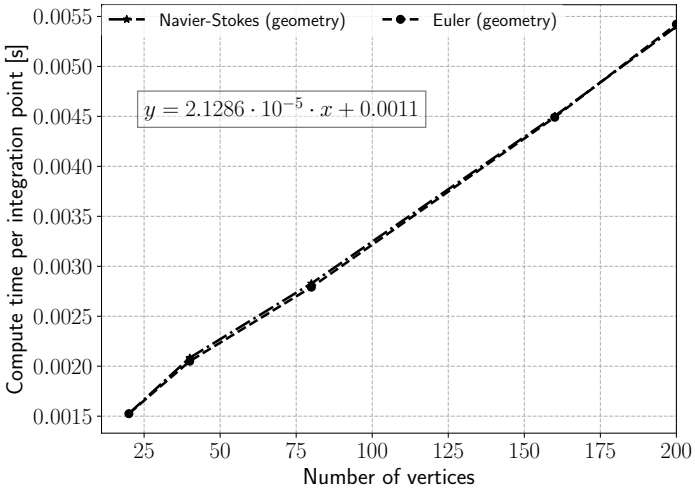


Figure 6.33.: Computational time per integration point for simulations with geometries, when compared to the simulations without geometries. Vertices represent the geometrical surface and can vary in number, depending on the complexity of the geometries. Linear fit (y) shows how the computational effort increases.

are presented. The last column in each table provides the compute time per element for the cases where no geometry is present, which serves as the reference time for the investigation shown in Figure 6.32. Please note that provided timings are for 100 iterations, as previously mentioned.

We now break down the problem further and investigate the computational effort per integration point. It allows to give a more general rule, independent from the scheme order or the number of elements used in the computational mesh. In Figure 6.33 the computational cost per integration point is depicted against the number of vertices used to model complex geometries. We know that the evaluation of the masking function χ is independent of the equations solved, but it depends on the number of integration points used to solve the problem. Therefore we expect, that the computational cost per integration point is the same for both cases (Navier-Stokes (geometry) and Euler (geometry)). From Figure 6.33

it is apparent that this is indeed the case, and both curves increase in computation with an increasing number of vertices. Both have the same slope and result in the same computational effort for one integration point compared to the number of vertices used to describe the geometrical surface. Hence through the given equation in Figure 6.33 we can approximate the additional computational effort for the simulation when geometries are present depending on the number of vertices. However, we need to emphasize that the computational time used to provide this equation is dependent on the system the measurements were done and might deviate when running on a different system.

Assuming 500 vertices to define the geometry and a computational mesh of $100,000$ elements, while the scheme order is $O(6)$, we can predict the additional computational effort using the equation provided in Figure 6.33 for 100 iterations. With the mentioned scheme order and the number of elements, we obtain $21,600,000$ integration points in the computational domain. The computational effort per integration point for 500 vertices is circa $0.0117\ s$. Multiplying this number with the total number of integration points, we obtain an additional computational time to evaluate the geometry of roughly $253,649\ s$. The equation provided in Figure 6.33 is as mentioned for 100 iterations. In Eq. (6.15), the equation per iteration is provided to allow for a better generalization of the prediction model. Our equation can be rewritten as

$$y_{inter} = 2.1286 \cdot 10^{-7} \cdot x + 1.1000 \cdot 10^{-5}, \tag{6.15}$$

with y_{inter} being the time per iteration and integration point and x the number of vertices. Thus, we can estimate the computational effort before executing our simulations. In Section 9.2.1 a more complex simulation is presented, where we prove that our prediction model accordingly works to predict the additional compute time for the simulation with complex geometries. The compute time per integration point from Figure 6.33 is presented in Table 6.9. In Table 6.10a the compute time per integration point for the Navier-Stokes equations is presented and in Table 6.10b the respective timings for the Euler equations. Both tables provide the compute time after subtraction of the compute time from the reference compute time per integration point.

Table 6.9.: Computational time per integration point for a different number of vertices to represent the geometrical surface. In (a), the compute time per integration point for the compressible viscous Navier-Stokes equations and (b) for the compressible inviscid Euler equations are presented. The last row in (a) and (b) presents the compute time per integration point for the respective equations when no geometry is present (reference).

Vertices	Compute time per integration point [s]
20	0.0015
40	0.0021
80	0.0028
160	0.0045
200	0.0054
–	0.0021

(a)

Vertices	Compute time per integration point [s]
20	0.0015
40	0.0021
80	0.0028
160	0.0045
200	0.0054
–	0.0006

(b)

Conclusion In this chapter, the validation of the Brinkmann penalization technique in our high-order Discontinuous Galerkin scheme was presented. We were able to prove that the mentioned technique can deal with **(i)** arbitrary complex shapes such as geometries with curved surfaces or sharp corners by introducing additional terms in the conservation equations to be solved. **(ii)** The approximation of the flow domain and the embedded geometry is with the same discretization method. **(iii)** Even though strong discontinuities are not desirable for high-order discretization, our approach (up to the investigated scheme order of $O(32)$) was able to provide more accurate solutions for the same number of degrees of freedom compared to a low-order scheme. Further, we studied through a test case from the literature that conservation of mass, momentum, and energy is maintained when modeling the geometry as an artificial porous material. In [34] we have further examined the boundary layer when solving the compressible Navier-Stokes equations for our simulations. Our investigations involved straight and curved boundaries. Obtained results demonstrated that the Brinkman penalization technique utilized in our high-order Discontinuous Galerkin solver also provides the expected results in the boundary layer region compared to solutions from the literature.

Additionally, through a strong scaling measurement, we were able to confirm that the solver scales further as expected, even when incorporating the penalization terms, to enable the motion of geometries. We further explained how geometries influence the computational effort for the compressible Navier-Stokes equations and the Euler equations. Our investigations revealed that the computational effort more noticeably increases for the Euler equations (factor of 2.91) than the Navier-Stokes equations (factor 1.55). Here the gradient computation is still the dominant part of the computation. The solver expects a list of points (vertices) to represent the geometrical boundaries accordingly (e.g., sharp interfaces). Therefore, the computational effort increases with the increasing number of vertices, as the masking function χ needs to be evaluated for the list of vertices at each integration point. In order to predict the additional computational effort, we introduced a linear equation that allows determining the additional computational cost a priori.

Furthermore, through our investigation, we pointed out that the relation between Euler: Euler (geometry): Navier-Stokes: Navier-Stokes (geometry) is roughly equal to $1 : 3 : 3.5 : 5$ for the computational cost in this particular test case. Even though this ratio is only valid for the used setup yet, the ordering, when using the mentioned equations with and without the penalization terms, remains valid, even for, e.g., a different scheme order. This outcome is of importance for later consideration of large-scale simulations (cf. Chapter 8) and their load balancing for efficient computation.

7. Multi-scale problems - An efficient strategy

The simulation of fluid-structure-acoustics (FSA) interaction is a typical engineering problem involving multi-scales and multi-physics. We can distinguish between two sound sources: structurally induced noise and flow-induced noise. Different scales and physics are involved, and each has to be appropriately resolved. These kinds of simulations demand a high amount of computational power. Thus the use of massively parallel supercomputers is inevitable. Since the sound effects in the far-field are the target of our simulations, a direct numerical simulation of a FSA problem becomes too expensive in realization when solving the entire problem with the same equations and spatial discretization. Assuming the far-field is far enough from the sound source, where a homogeneous background flow can be expected, we can restrict the simulation in this area of the computational domain to the propagation of acoustic waves. As a result, linear equations can be solved, allowing to reduce the computational effort accordingly. Furthermore, we can assume that far away from the sound source, viscous effects do not play a role anymore and can be neglected, as only nonlinear phenomena are present. Therefore an appropriate strategy is required to facilitate such complex simulations feasibly.

Generally, we can decompose such a FSA problem into three spatially separable domains, where each is solved with a tailored configuration to ensure a dedicated physical treatment. Thus, we end up with a setup where we have to deal with three domains, each considering a different set of equations and spatial discretization. In this context, we can take advantage of the high-order Discontinuous Galerkin method, where we can utilize the properties of this method for the acoustic far-field and consider simplified equations, a higher scheme order, and a coarser mesh. A higher order provides the benefits of low dissipation and dispersion error, ensuring that acoustics information can travel with high accuracy up to the far-field and requires a smaller amount of memory for the computation. Additionally, in areas where viscous effects can be neglected, the mesh does not have to be as fine as close to the structure, where small scales

are dominating. The decomposition of such a complex problem however, introduces new challenges that are i.a.: **(i)** Multi-scales in space and time (small scales close to the structure and acoustic waves in the far-field). **(ii)** Different spatial discretization (h/p refinement) for each domain. **(iii)** Load balancing challenges due to unequal computational workload per domain depending on the occurring physics. Engineering applications of such complex problems can be found in various technical systems, such as wind turbines, fans in air conditioners, turbines, or airfoil design. Therefore, an efficient strategy is required to address those challenges and realize these complex engineering problems properly.

7.1. State of the art - Coupled problems

The interaction of fluid and structure (FSI) plays a prominent role in many engineering applications and other scientific fields. However, a comprehensive study of such complex problems remains challenging due to their inherent nature of strong nonlinearity and multidisciplinary [11, 31]. Due to the recent advances in computer architecture and the computational power of new supercomputers, sophisticated and complex simulations have become more feasible. With that, advances in many scientific fields can be observed, where FSI simulations have become possible for many complex problems. FSI applications are among others in the field of aerodynamics [97], magneto-hydrodynamic flows [43], hemodynamics [28] and sedimentation [94]. Furthermore, many more applications in the medical field have been investigated, e.g., the interaction of blood flow and stenosed artery. The procedure of these FSI problems can be categorized into a monolithic approach and a partitioned approach.

In the context of the monolithic approach, the treatment of the structure and fluid dynamics is as a single large system of equations that must be solved simultaneously. The interface conditions are kept implicit for the computation of the solution. It is possible to achieve a more accurate solution applying this approach when considering multidisciplinary problems. However, this requires an immense amount of computational resources [13, 40]. The opposite strategy is the partitioned approach, where fluid and structure are treated as different computational fields, solved by different numerical algorithms, and have different spatial resolutions. Therefore the flexibility is higher when choosing suitable solvers for each of the domains. However, this approach has its challenges and limitations, including stability, accuracy, and efficiency between domains to be coupled.

In [17] numerous monolithic as well as partitioned approaches have been proposed and investigated in more detail. Investigations in literature were mostly restricted to FSI simulations, and the acoustic near-field due to the expensive computation [66].

Schwarzkopff [83] moved a step further and investigated in his work the acoustic far-field, where he enabled the surface coupling of the flow domain and the acoustics far-field using ghost elements for the coupling. They provide the necessary information for the numerical scheme and allow the coupling of different numerical methods. Based on his work, Utzmann [93] contributed to a more generalized approach for the coupling of the flow domain and the acoustics far-field. He enabled the data exchange at the Gauss integration points in the previously introduced ghost elements in his work. This change provided a more efficient coupling and a more generalized approach to allow the coupling of different numerical methods. A step further and a more flexible method is based on the black-box coupling. Each domain is treated as a black box with arbitrary surface data and solved by a dedicated numerical method. The coupling tool is responsible for the data exchange between the different domains, e.g., [65]. Various tools are available that support this kind of black-box couplings, such as *MpCCI* [2] developed by the Fraunhofer department SCAI or *preCICE* [18] developed as a joint project by the University of Stuttgart and the University of Munich.

Both approaches have been further developed in the past years. However, *preCICE* has shown to be more efficient in parallel execution when compared to *MpCCI*. The coupling tool *MpCCI* is based on client-server communication, thus limiting parallel efficiency. At the same time, *preCICE* employs point-to-point communication, which is efficient for large-scale simulations on HPC systems. More information on the coupling approaches can be found on their respective pages. Due to the mentioned reason, we utilize *preCICE* for our investigations in the following section, where simulation results are compared to the white-box tool *APESmate*.

As introduced in Chapter 5, in this work, we consider a one-directional coupling of fluid and structure. The geometrical properties are embedded in the fluid dynamic equations and simultaneously solved in a monolithic manner (cf. Chapter 5). Due to the presence of the geometry, the fluid flow is disturbed and eventually becomes turbulent, while noise is induced that propagates to the far-field (flow-induced noise). We are particularly interested in the simulation of the far-field, which has become even more critical

in recent years in the strife for noise reduction. Thus, an efficient approach is indispensable to realize such compute-intensive problems. Therefore a partitioned approach is utilized to couple the flow- and acoustic field. In the upcoming section, we focus on the coupling strategy we apply in this work to couple the flow field and the acoustics far-field, highlighting the advantages and limitations of the used approach.

7.2. Partitioned coupling

Considering the simulation of fluid-structure and acoustics interaction, the complexity of the simulation increases compared to an FSI simulation. However, these interactions are common in many engineering applications. They have to be enabled in a feasible practice as numerical simulations have become indispensable in, e.g., the design optimization process of devices. Intensive research has been conducted in recent years in order to address these complex simulations (cf. Section 7.1). Often, the question arises whether the partitioned coupling approach is suitable compared to the monolithic approach in terms of computational efficiency and the accuracy of the solution. In the partitioned approach, the computational domain is decomposed into subdomains according to the occurring physics. All subdomains are connected to the corresponding neighboring subdomain via a coupling tool. This methodology gives each subdomain certain independence regarding the set of equations to be solved and the spatial resolution. Furthermore, the partitioned approach allows for efficient computation when considering the increasing complexity of problems and available compute power. Thus, it can accentuate its advantages, especially when the acoustics far-field is of interest.

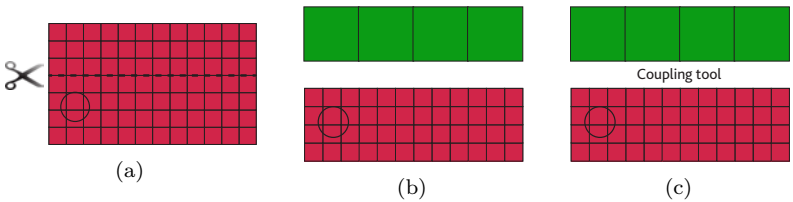


Figure 7.1.: Decomposition of the large simulation domain into smaller subdomains coupled together through a coupling approach for multi-scale simulations.

The principle of the partitioned coupling is explained in Figure 7.1. Each subdomain is solved with a different set of equations and spatial resolution (cf. Figure 7.1b), and connected at a joint interface via a coupling approach (cf. Figure 7.1c). The expensive equations (marked in red in Figure 7.1) are only solved in the subdomain, where the geometry (cylinder) is located. It requires a finer mesh to capture small scales close to the geometry. Away from the geometry and the small scales, the fluid dynamic equations can be simplified according to the physics (highlighted in green in Figure 7.1b) and a coarser mesh used.

In order to confirm, that the idea of the partitioned coupling allows the realization of complex numerical simulations more efficiently, we investigated this approach in terms of accuracy and computational efficiency compared to the monolithic approach for a small test case, that can be still realized monolithically. In this work we investigate the accuracy of the solution and refer to [35, 36, 60, 65] for analysis of the computational efficiency. The partitioned approach is based on the decomposition of the entire simulation domain into subdomains, that are connect to enable the data exchange at their boundary interface (cf. Figure 7.1c). This strategy is realized through a coupling approach, a so-called white-box and a black-box approach. The white-box approach is within the same simulation framework as the solver. Thus it knows the internal data-structure. On the other hand the black-box approach, that does not know the solver used for the simulations, expects interface definitions, where the domains are coupled together. Thus, this results in an approach, which is less flexible in terms of coupling different solvers (white-box), but can provide accurate results (see Section 7.3.3), while the black-box approach is more flexible in terms of coupling different solvers, however, the accuracy of the solution may suffer, when using this approach (cf. Section 7.3.1).

The next sections are devoted to the two coupling approaches, namely the black-box approach *preCICE* (cf. Section 4.2) and the white-box approach *APESmate* (cf. Section 4.1.2), where their advantages and disadvantages in terms of accuracy of the solution are studied in more details.

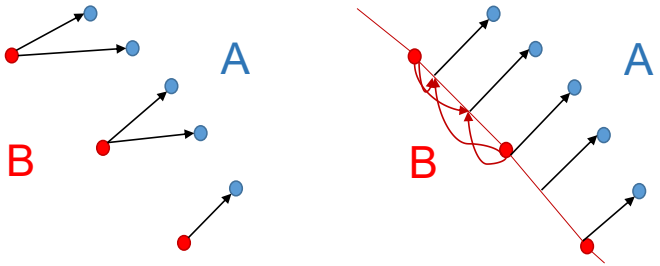
7.3. Quality of the solution - Data mapping

The major concern, when deploying partitioned coupling is the loss of accuracy, due to the decomposition and the different numerical treatment of

each subdomain. In this section we demonstrate, how we can maintain the quality of the solution with both coupling approaches. We first investigate the black-box coupling approach *preCICE*, surveying the different interpolation methods provided by this approach. Afterward, we examine the method used by the white-box approach *APESmate*. Lastly, we analyze the numerical solutions obtained by both coupling approaches with the monolithic solution, through a test case, a Gaussian density pulse.

7.3.1. Interpolation

preCICE is, as previously mentioned, a black-box coupling approach, allowing the coupling of different solvers, considering them as a black-box. This approach requires the point positions at dedicated coupling boundaries, at which data has to be exchanged. Those points are called coupling points and can have an arbitrary distribution. *preCICE* provides different interpolation methods for the data exchange, which are reviewed in the following. The task of the coupling approach is to provide values throughout the simulation at requested coupling point positions. Therefore, interpolation methods are used to compute those values from point values of one domain to the other and vice versa, resulting in the update of the state at the boundaries at predefined time steps. The accuracy of the solution at the coupling boundaries is of importance for the overall solution. Therefore, it is of interest to know which interpolation method can maintain the accuracy of the solution close to the monolithic solution, even though a non-matching coupling interface (cf. Figure 7.4) increases the challenge in preserving the overall accuracy.



(a) (b)

Figure 7.2.: Sketch of the (a) Nearest-Neighbor method and (b) Nearest-Projection interpolation method provided by *preCICE* [18]

The easiest applicable interpolation method to use is the Nearest-Neighbor (NN) method (see Figure 7.2a) [18]. The solver merely needs to provide the point positions of the coupling points and the variables to be exchanged. Coupling from one subdomain (B) to the other (A), the closest neighboring point on B is detected, while the value is copied to the requested point on A. Though this method is straightforward to apply, the major disadvantage is its accuracy. For example, in the case of Figure 7.2a more than one point of domain A is close to one of the coupling points in domain B; this results in the same point values for all those points in A from domain B. Therefore, this method is only valid in terms of accuracy if both coupling interfaces have the same number of points and the same point distribution, thus matching interfaces (cf. Figure 7.4a). The Nearest-Projection (NP) method (see Figure 7.2b), in contrast, is more involved and provides more accurate solutions as it is a 2^{nd} order accurate method. This method searches in domain B for the closest neighboring points, required for the points in A, while the closest point is projected from B to A and vice versa [18]. However, in order to use this method, the solver needs to provide additional connectivity information (cf. Figure 7.3). For a two-dimensional simulation, the coupling interface is only a line. Hence *preCICE* requires, in this case, additional edge information at the coupling interface. In the case of a three dimensional case, additional triangle information (see Figure 7.3b) are necessary. Please keep in mind that the coupling points in our case are the integration points of the underlying high-order Discontinuous Galerkin scheme (cf. Figure 7.3a), which are non-equidistantly distributed inside the element (cf. Section 5.3). To avoid additional implementation, while still using a 2^{nd} order interpolation, the Radial-Basis-Function method (RBF) can be considered. This method does not rely on any neighboring information and is internally implemented in *preCICE* as

$$g(x) = \sum_{i=1}^{N_B} \gamma_i \cdot \phi_{rbf}(||x - x_i||) + q(x). \tag{7.1}$$

It creates for the mapping a global interpolant on B, which is evaluated on domain A. The bases are radially symmetric basis functions, centered at the coupling points of domain B. To ensure the correct interpolation of constant and linear functions, an additional 1^{st} order global polynomial $q(x)$ for the bases is taken into consideration. In Eq. (7.1), the variable ϕ_{rbf} gives the user the flexibility to select between different basis functions. However, according to [18], the quality of the solution is comparable. In this work, we consider the Gaussian function, which requires the computation

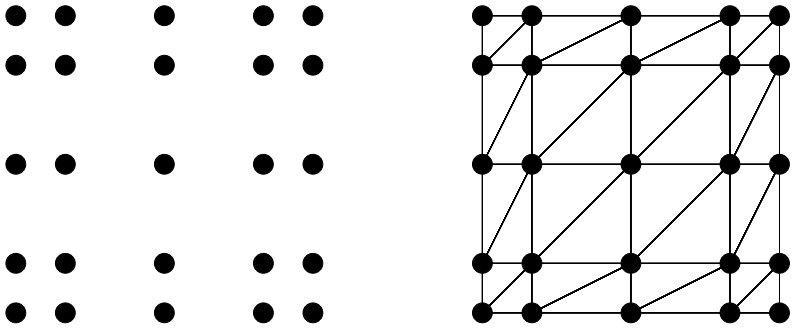


Figure 7.3.: Sketch of (a) the Discontinuous Galerkin point distribution, (b) the connectivity information (edge and triangle) for the NP interpolation method for a three-dimensional test case, provided to the black-box approach.

of the shape-parameter s by the user.

$$s = \frac{\sqrt{-\ln(10^{-9})}}{m \cdot h} \tag{7.2}$$

In Eq. (7.2), m defines the number of coupling points to be covered by the Gaussian function, while h is the average distance between the points. We need to emphasize that the distribution of the coupling points is essential for the quality of the solution and the convergence rate of the linear equation system solved for this method. As previously discussed and shown, e.g., in Figure 7.3a the coupling points are the same as the integration points and therefore non-equidistantly distributed inside an element. The points are denser in the element's corners, while the points have a more significant distance in the center of an element. The choice of the shape-parameter has a notable impact on the convergence rate of the equation system that has to be solved. A high value of m can improve the accuracy of the solution; however, it has an oppositional effect on the equation system to be solved, as the convergence rate increases as well, resulting in a high condition number [64]. For our computations, h is defined according to the largest distance between the integration points, the middle of an element. This practice allows covering all points inside

the element. The computation of the distance h is according to

$$h_{max} = \left[\cos\left(\frac{2\cdot(n_h+1)-1}{2\cdot nO}\cdot\pi\right) - \cos\left(\frac{2\cdot n_h-1}{2\cdot nO}\cdot\pi\right)\right]\cdot\frac{dx}{2}. \quad (7.3)$$

Here nO is the scheme order, n_h the used scheme order divided by two, and dx the size of the computational element at the coupling interface. Another drawback we face here is that not equidistantly distributed points result in instability of the convergence of the matrix to be solved. In contrast, equidistant points can solve this issue [64]. However, in the context of a high-order Discontinuous Galerkin method, this method would require additional effort (equidistant points) to achieve accurate simulation results. Keeping these challenges in mind, it is apparent that a different method is necessary to overcome these challenges while maintaining the accuracy of the solution. Therefore in the next section, we discuss the white-box coupling approach *APESmate*, which uses a different method to allow for accurate data exchange.

7.3.2. Data mapping by evaluation

The coupling approach *APESmate* (cf. Section 4.1.2) is integrated into the simulation framework *APES* (cf. Section 4.1); hence it has access to solver specific data and the underlying common data structure used by all solvers. Consequently, it can enable the data exchange at an arbitrary set of exchange points for the high-order Discontinuous Galerkin solver. Therefore non-equidistant point distribution is not a drawback for the coupling, as the polynomial representation can be evaluated at requested point coordinates by each coupling domain. This approach, therefore, allows maintaining the high-order accuracy of the solution. More information on this coupling tool can be found in [59, 68].

To investigate the quality of the solution, we examine the error of the solution compared to the solution obtained from a classical monolithic run. For the black-box approach *preCICE*, we review all three interpolation methods mentioned in Section 7.3.1. We aim to identify a well-suited method for multi-scale simulations with non-equidistant coupling points as in the context of the high-order Discontinuous Galerkin method. Further, it needs to be mentioned that some results in the following section have been published in [35].

7.3.3. Error investigation

Due to the decomposition and the different treatment of each subdomain, a significant concern of the partitioned coupling approach is the accurate data exchange between the subdomains. Furthermore, since different scheme orders can be used for each of the domains, the accuracy of the solution is limited to the lowest scheme order used. Therefore, it is even more important to maintain the solution's accuracy over the different subdomains. Thus, a small academic test case is used to investigate the accuracy of the coupled simulation for both coupling approaches. In the first step, the entire simulation domain is monolithically solved and later used as a reference for the investigation. In the next step, the monolithic domain is divided into two subdomains and solved with different spatial resolutions and scheme orders while solving the same set of equations. For the investigation, a density pulse is used that travels from the left side of the domain towards the right boundary. Therefore, the simulation domain is divided into two parts for the coupled scenario, namely the left and right domains. They are coupled together through the coupling approaches and exchange data at their respective coupling points, the Chebyshev nodes.

Test case description: A 4×4 unit plane is used for the simulation domain, which is decomposed into two subdomains for the coupled scenario. For all test cases, the inviscid compressible Euler equations are solved. The amplitude of the pulse is defined to be 1.0, with a halfwidth of 0.316. The pressure p and density ρ are 1.0 and 1.0, respectively. The velocity is prescribed as $\vec{v} = [12.5, 0.0, 0.0]$. The pulse travels due to advection in positive x-direction from the left to the right domain while crossing the coupling interface for the coupled scenario. For the examination, three different test cases are used. In test case (a), both domains have the same scheme order and mesh resolution, hence a matching coupling interface (boundary interface). In test case (b), the left domain is kept as (a), while the right domain has a coarser mesh and a higher scheme order, resulting in a non-matching scenario. Finally, test case (c) has the same configuration again in the left subdomain as in (a) while the right domain has an even coarser mesh and the same scheme order as in (b), thus an increasing non-matching coupling interface.

In Table 7.1 the setup of all three test cases are shown; it provides an overview of the different setups of each test case. Test cases (b) and (c) are essential for the investigation and provide insights into the accuracy of the solution. Furthermore, they accentuate how well the numerical solution

can be preserved. In Figure 7.4 the point distribution for each test case is

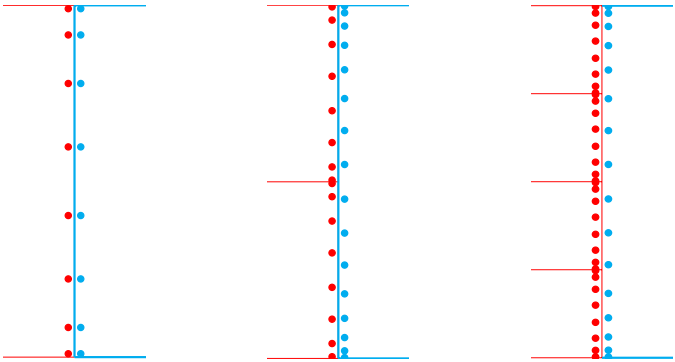


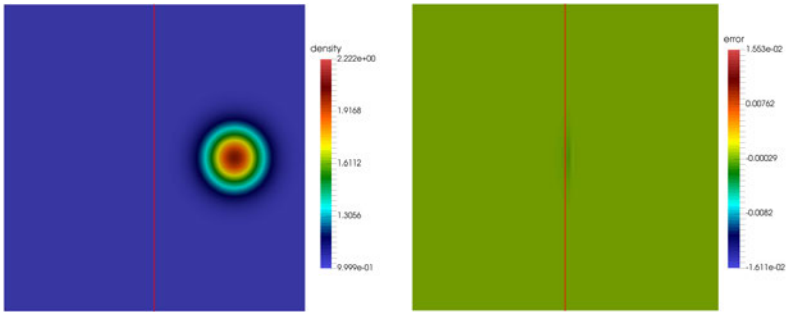
Figure 7.4.: Point distribution in elements, when using the Discontinuous Galerkin method: (a) Matching test case (a), (b) non-matching test case (b), and (c) non-matching test case (c).

shown. For test case (a), both domains have the same point distribution (cf. Figure 7.4a), in (b) the element size of the right domain is two times bigger than the left one (see Figure 7.4b) and in (c) the right domain covers four elements of the left domain (cf. Figure 7.4c). Furthermore, the point distribution is non-equidistant; points are more concentrated at the element's corners than the middle part. Therefore, the interpolation error for the data mapping is dominated by the distance of the point distribution [33, 35].

As the investigated test case is small enough, we can obtain a monolithic simulation, running the entire test case in a single domain (non-split) and using this as a reference to investigate the coupling error. We split the domain into halves while keeping the same element size and scheme order. I.e., all differences between split and monolithic simulation results are due to the coupling error (test case (a)). In the following, we then change the settings in the right domain to adopt the scheme order and element size to the needs of the domain (test cases (b) and (c)), resulting in additional errors due to the non-matching conditions at the coupling interface. For the monolithic simulation, the configuration is the same as for test case (a). The error is obtained from the difference between the solution from

Table 7.1.: Three test cases for the investigation of the interpolation methods

	Test case (a) matching		Test case (b) non-matching		Test case (c) non-matching	
	left	right	left	right	left	right
Number of elements	512	512	512	256	512	128
Number of coupling points	128	128	128	128	128	64
Scheme order	8	8	8	16	8	16



(a) (b)

Figure 7.5.: (a) Monolithic solution from the numerical simulation and (b) error in the solutions, when compared to the exact solution.

the simulation and the analytical solution [88]. Therefore, the error should provide an approximate error we can compare with later when computing the L2 error in Table 7.3. In the middle area of the simulation domain ((cf. Figure 7.5b)), a remaining error can be observed. This error is due to the location of the pulse at the beginning of the simulation. The pulse was located too close to the left boundary, and oscillations appear that remain in the domain. However, this error does not influence the coupled scenario and the comparison of the different methods. Moreover, since they appear in the monolithic scenario (reference) and the coupled case, this error is diminished in the solution when the difference between reference and the coupled test cases is determined. The main focus of this section is devoted

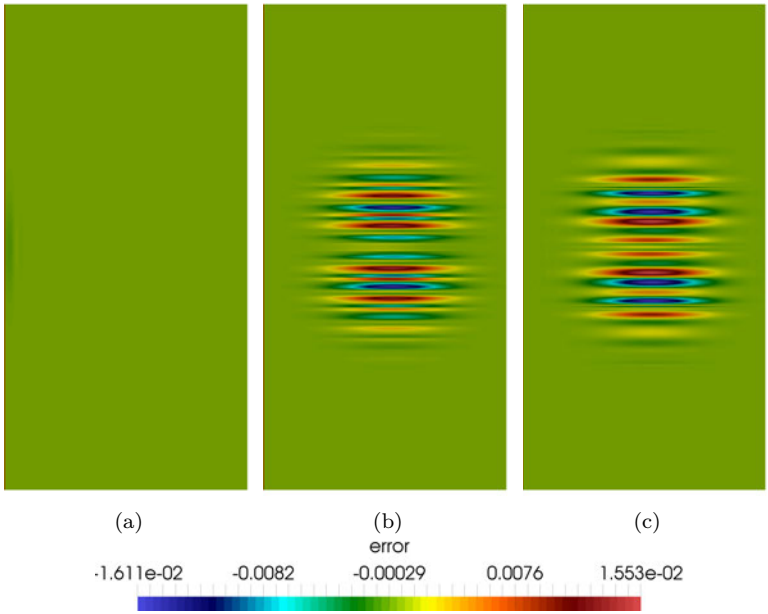


Figure 7.6.: Error for the NN method provided by the black-box approach *preCICE* compared to the monolithic approach (reference). (a) Matching interface (test case (a)), (b) non-matching interface (test case (b)) and (c) non-matching interface (test case (c)).

to the error investigation introduced by the interpolation/evaluation of the coupling approaches; therefore, the left subdomain is neglected for the visualization. Interested readers may consider [35] for the visualization of the left subdomain. In [35] it was already explained that the difference between the monolithic approach and the coupled simulation could be found in the right subdomain. This is due to the density pulse that crosses the coupling interface. Hence the coupling approaches have to interpolate/evaluate the solution from one subdomain to the other. As can be observed from Figure 7.6 the error in the solution drastically grows with increasing mismatched coupling interfaces for the interpolation method NN. In Table 7.3 the L2 error for all three test cases are calculated using the analytic solution as a reference.

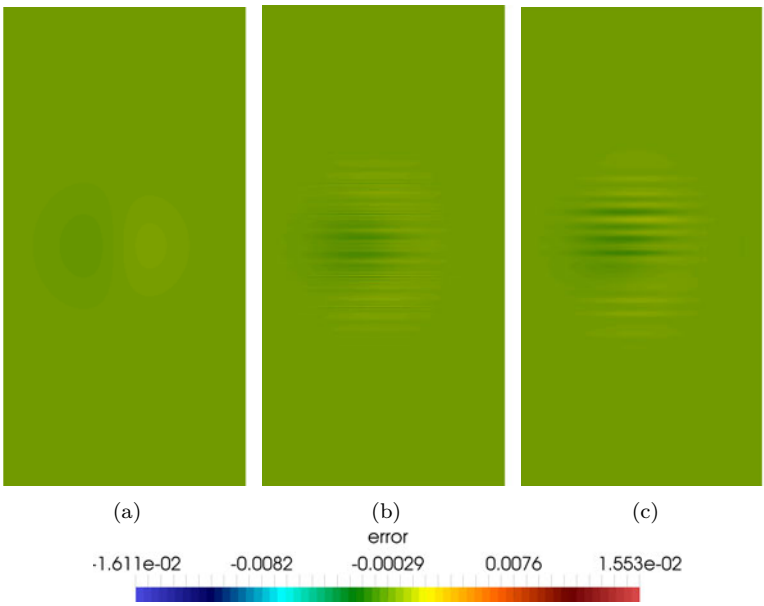


Figure 7.7.: Error for the NP interpolation method provided by the black-box approach *preCICE* compared to the monolithic approach (reference). (a) Matching interface (test case (a)), (b) non-matching interface (test case (b)) and (c) non-matching interface (test case (c)).

For the NN method, the L2 error increases such that the error of the interpolation method dominates the solution; hence it is not suited for coupled simulations with non-matching coupling interfaces. For the matching test case (a) (cf. Figure 7.6a), the error is, as expected very small, since both coupling interfaces have the same number of coupling points, scheme order, and element size. Therefore, this method is not suitable for non-matching coupling interfaces, as it is only 1^{st} order accurate and not able to reconstruct the solution accurately. Though, this method is eliminated for further investigations.

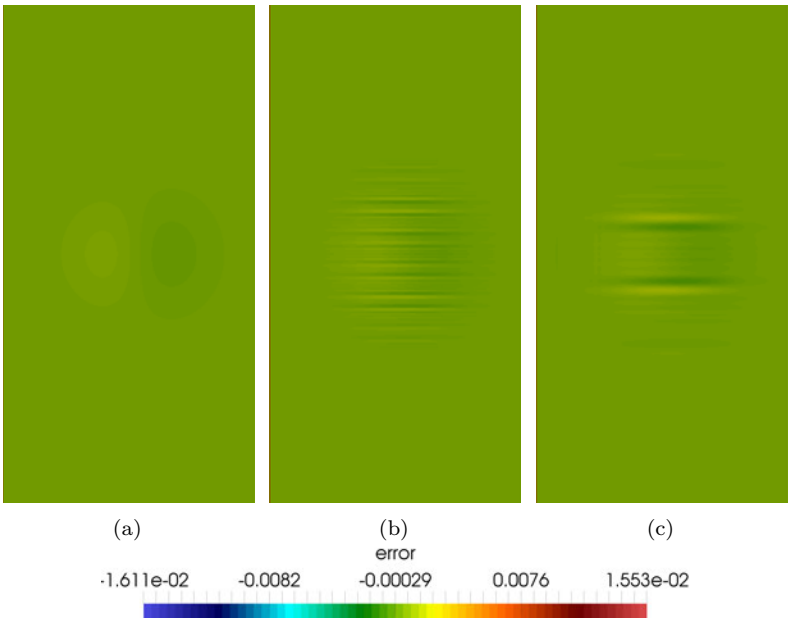


Figure 7.8.: Error for the RBF interpolation method provided by the black-box approach *preCICE* compared to the monolithic approach. (a) Matching interface (test case (a)), (b) non-matching interface (test case (b)) and (c) non-matching interface (test case (c)).

The NP method is as mentioned in Section 7.3.1 a 2^{nd} order accurate interpolation method. Therefore, the expectation for this method is to provide a smaller error compared to the NN method. As shown in Table 7.3 the L2 error for this method is smaller than for NN. However, on the other hand, an increasing L2 error can be observed when the non-matching degree at the coupling interface increases, e.g., in the case of the test case (c). The error, when compared to the monolithic approach, is depicted in Figure 7.7, also here the increasing error from test case (a) to (b) and from (b) to (c) can be observed.

The third interpolation method we investigate is the RBF method. Even though the user does not have to provide neighborhood information; how-

Table 7.2.: Shape-parameter for non-equidistant point distribution for the RBF interpolation method

Test case	h_{max}		m		s	
	left	right	left	right	left	right
a	0.0244	0.0244	4	4	46.642	46.642
b	0.0244	0.0245	4	3	46.642	61.936
c	0.0244	0.0245	4	2	46.642	46.452

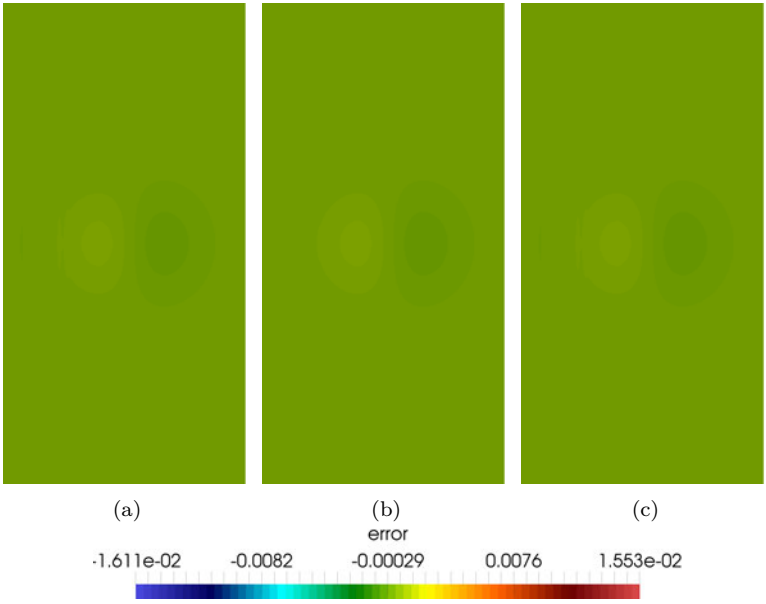


Figure 7.9.: Error of the evaluation provided by the white-box approach *APESmate*, compared to the monolithic approach. (a) Matching interface (test case (a)), (b) non-matching interface (test case (b)) and (c) non-matching interface (test case (c)).

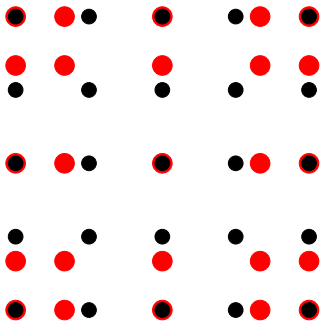


Figure 7.10.: Equidistant points (black dots) are provided to *preCICE* for the interpolation to provide data on non-equidistant points (red dots) requested by the solver.

ever, the shape-parameter has to be provided to the coupling approach. In Eq. (7.2) the shape-parameter is defined. This method is challenging as an equation system needs to be solved by the coupling tool, while the convergence rate depends on the shape parameter itself. Hence, the user needs to examine the shape-parameter a priori to find a suitable one, allowing the equation system's convergence for each test case. Figure 7.8 illustrates the error, which increases with increasing mismatching coupling interfaces. In Table 7.2 the different shape-parameters for the different test cases are presented. The parameter h_{max} determines the largest distance between the integration points, which are the same as the coupling points. This distance is set to be the maximum, as points in the middle area of an element have a larger distance and need to be taken into account. Finally, in Table 7.3 the L2 error for this method is presented. The error is when compared to NN and NP smaller yet presents an increase for the test cases (b) and (c), while the error is smaller when compared to the error of the 2^{nd} order NP interpolation method (cf. Figure 7.8).

For comparison, we simulate the same test cases applying the white-box approach *APESmate*, where no interpolation but an evaluation of the underlying polynomial representation is used to exchange data at pre-defined coupling points. With that, this approach can maintain the overall accuracy, and the error can be kept small for all three test cases, even when the non-matching at the coupling interface increases from test case (a) to (b) and from (b) to (c). Figure 7.9 illustrates the error when compared to

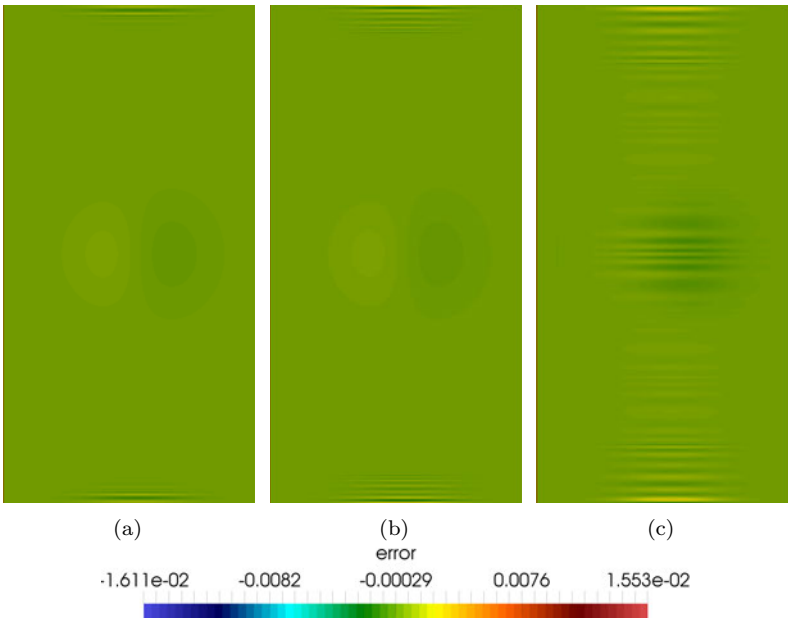


Figure 7.11.: Error for the RBF interpolation method when using equidistant points for the interpolation provided by the black-box approach *preCICE* compared to the monolithic approach. a) Matching interface (test case (a)), (b) non-matching interface (test case (b)) and (c) non-matching interface (test case (c)).Equidistant points (black dots) are provided to *preCICE* for the interpolation to provide data on non-equidistant points (red dots) requested by the solver.

the monolithic run for all three test cases. This method allows the direct evaluation of the underly polynomial at requested coupling points. In Table 7.3 the L2 error is shown, all simulation results provide outstanding results; even though the coupling interface considerably changes, the error is smaller for all three test cases compared to the interpolation methods provided by the black-box approach.

To conclude all outcomes up to here, we can summarize that the interpolation methods provided by the black-box approach *preCICE* lack

the desired accuracy for the solution for coupled simulations when the coupling points are non-equidistantly distributed. However, the withe-box approach *APESmate* can maintain the overall accuracy of the solution, even for those test cases, where all *preCICE* interpolation methods resulted in a significantly larger L2 error. Further, with the coupling tool *preCICE*, we can only achieve an accuracy of the solution of up to 2^{nd} order. With the coupling approach, *APESmate*, the accuracy of the solution is only dependent on the scheme order used in the domains.

In order to reduce the error for the coupled scenario, when using *preCICE*, the conclusion of Lindner et al. [64] is considered, where equidistant points are provided to the coupling approach for the interpolation instead of the non-equidistant points. Due to this change, the solver has additional work since two sets of points must be provided to *preCICE*. The equidistant points have the same amount of points as the non-equidistant ones but are equally distributed. They are used, as mentioned, only for the interpolation, while the second set of points are used to receive data from *preCICE*, as the solver requires values at the non-equidistant coupling points (see Figure 7.10). Figure 7.11 presents the results when using

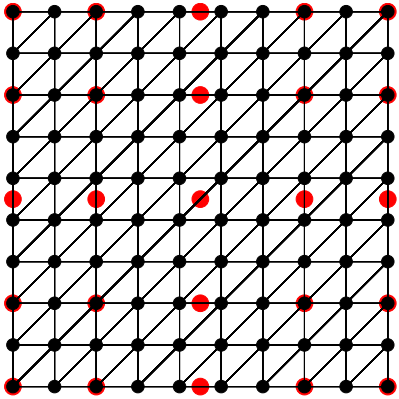


Figure 7.12.: Equidistant point distribution is used to improve the interpolation with *preCICE*. More points are considered (an over-sampling factor of two) to reduce oscillations and decrease numerical error. Red dots indicate the original non-equidistant Discontinuous Galerkin integration points, for which data is requested.

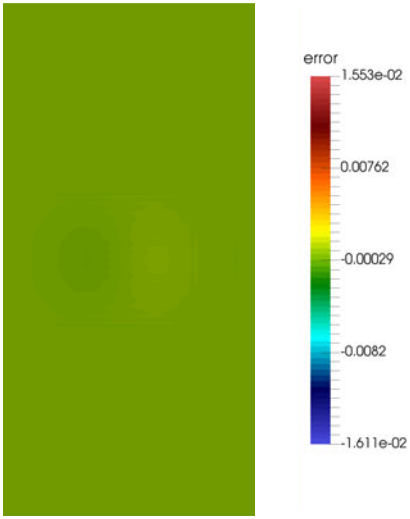


Figure 7.13.: Error for the NP interpolation method, when using two times more equidistant points for the interpolation (oversampling), compared to the monolithic approach.

equidistant coupling points for the interpolation in *preCICE*, while still requesting values for the solver at non-equidistant points. Therefore the solver needs to evaluate the polynomial representation of the state at equidistant points for the interpolation with *preCICE*. This practice does not provide satisfactory numerical solutions. The different test cases clearly show how the quality of the solution is dominated by oscillations appearing in the upper and lower area of the domain. Those oscillations become more apparent with increasing non-matching coupling interface, hence the change in the number of coupling points from test case (a) to (b) and from (b) to (c). This phenomenon (oscillations) is well known and referred to as Rung's phenomenon [38]. It is observed when using equidistant points to construct high-order polynomials, as in this test case. To reduce oscillations further, more equidistant points are used for the interpolation. This practice results in more points near the element's corners; hence a better approximation can be achieved. Since the RBF method with equidistant points is more complex and challenging when,e.g., considering the computation of a suitable shape-parameter, the NP method

is investigated further, where more equidistant points for the interpolation are used.

The NP method only needs the connectivity information and does not have to solve an equation system, which is more attractive in terms of usability and computational efficiency. For example, in Figure 7.12 is shown how we try to improve the solution through oversampling, thus providing *preCICE* only for the interpolation two times more equidistant points (black dots). Nevertheless, the solver requires point values at non-equidistant points (red dots) from *preCICE*. Figure 7.13 depicts exemplary the improvement of the error when considering the interpolation method NP with equidistant points while using two times more equidistant points for the interpolation (referred to as oversampling) for the test case (c), where the non-matching level of the coupling interface is the largest among the investigated cases. The computed L2 error in density is closer to the solution of the white-box approach *APESmate*, with $1.279 \cdot 10^{-4}$. In [33] the error behavior for equidistant and non-equidistant points has been examined in more detail. Interested readers are referred to this publication for more information.

Table 7.3 provides an overview of all investigated methods and the L2 error for each test case. When comparing the error with the monolithic approach, it is apparent that only the evaluation method provided by the white-box approach *APESmate* can produce for all test cases a small error, even when considering the increasing non-matching coupling interface from test case (a) to (b) and from (b) to (c). In contrast, all interpolation methods provided by the black-box approach *preCICE* produces a higher L2 error with increasing non-matching coupling interface. It is also clearly shown that providing equidistant points for the RBF method does not significantly improve the solution's quality for this particular test case. This method is dependent on the selection of the shape-parameter. In [33] the RBF method with equidistant points showed for that particular test case better simulation results than the RBF interpolation with non-equidistant points. Therefore, it is difficult to conclude whether this method provides sufficiently accurate results for the coupled scenario. Furthermore, the compute time varies from test case to test case as a convergence of the system needs to be obtained. Though for the NP method with equidistant points, the quality of the solution is significantly improved for the investigated test cases in [33] as well as for the test case considered here. For an oversampling factor of two with equidistant points, the solution of the NP method is improved when considering, e.g., test case (c). The L2 error drops from $1.716 \cdot 10^{-4}$ to $1.522 \cdot 10^{-4}$ (see Table 7.3) for this interpolation

method. Therefore the error is smaller than for both investigated RBF methods with and without equidistant points. It is challenging to improve the quality of the solution using oversampling for the RBFs as well, as with an increasing number of points, the condition of the equation system to be solved becomes worse, too [64].

Table 7.3.: Comparison of the L2 error with respect to the analytical solution for the different methods

$\times 10^{-4}$	(a)	(b)	(c)
Nearest-Neighbour	0.642	15.711	41.875
Nearest-Projection	0.642	1.252	1.716
Radial-Basis-Function: Non-Equidistant Points	0.642	1.124	1.861
Radial-Basis Function: Equidistant Points	1.150	2.885	2.788
APESmate	0.642	0.901	1.279
Monolithic		**0.423**	

From the L2 error, we can conclude that when applying the black-box approach *preCICE*, the NP method with equidistant points and an oversampling of those points for the interpolation can provide comparable results as for the white-box approach *APESmate*. For more information, the interested reader is referred to [33]. A general overview of the connectivity information of the NP method is shown in Section A.1. The outcomes of our investigation in this section were used for a FSA coupled simulation, where the Finite-Volume solver *FASTEST* [58] and the high-order Discontinuous Galerkin solver *Ateles* were coupled via the coupling approach *preCICE*. The FSI part for this simulation was realized by *FASTEST*, while the acoustics far-field was simulated by *Ateles*. Additional challenges regarding the simulated test case and the coupling of different solvers are discussed in [65].

In this work, the in-house solver *Ateles* is used for all simulations; therefore, the in-house coupling approach *APESmate* is utilized for coupled simulations. The white-box coupling approach can maintain the overall accuracy of the solution regardless of the point distribution. In the following chapter, we focus on the challenging part of partitioned coupling, namely load balancing, that can be foreseen and needs to be appropriately addressed due to the different workload of each subdomain.

Conclusion In this chapter we demonstrated, that partitioned coupling can maintain the overall solution compared to the classical monolithic approach. We investigated two coupling approaches, namely the black-box approach *preCICE* and the white-box approach *APESmate*. The quality of the solution is comparable to the monolithic approach, using the white-box coupling approach. We improved the solution considerably for the black-box coupling tool *preCICE*, utilizing more and equidistant coupling points for the interpolation. In addition, we highlighted that the interpolation method NP provides comparable solutions in terms of quality compared to the white-box approach's evaluation method. However, when deploying the solver *Ateles*, which is part of the simulation framework *APES*, applying the white-box coupling approach is advantageous, as it can maintain the quality of the solution without additional information provided by the user.

8. Load balancing - Coupled multi-scale problems

The goal of load balancing (LB) is to reduce performance overhead and maximize parallel performance. Thereby, the workload is ideally distributed among available processes such that the available computing power is used to its full capacity. Load balancing particularly is of importance for highly parallel applications and is critical for efficient usage of resources for computation [10]. However, even slight imbalances of the workload can cause severe performance and a scalability penalty. In terms of numerical simulations, especially in coupled simulations, they typically involve communication between subdomains. The imbalances become even worse due to the communication and data exchange at predefined synchronization time steps. Further, the decrease in performance is dependent on the load imbalance pattern. For example, a single process that has an overload of work will significantly reduce the overall performance. This is due to the waiting time of all other processes involved, which need to wait until the communication can continue again through synchronization. On the other hand, a less severe pattern might be when a few processes are underloaded, which will have a more negligible impact on the overall performance. In general, load imbalance can be classified into static and dynamic. Dynamic load imbalance occurs if the workload distribution varies over time. This is often the case when, e.g., adaptive meshes or the adaptation of equations is required throughout the simulation time. On the other hand, a workload that is independent of time, thus when it is constant, can be addressed through static load balancing (SLB) [16]. Since all simulations in this work do neither change the computational mesh nor the scheme order of the discretization method or the equations to be solved inside each subdomain, we consider SLB to address load imbalance for our simulations.

In the context of coupled simulations, we can distinguish between two sources of load imbalance. Load imbalance due to different workload of each subdomain referred to as **inter-subdomain** and load imbalance occurring inside each subdomain (**intra-subdomain**). The intra-subdomain load imbalance is further increased due to the coupling and the respective

coupling elements and the elements covered by the geometry. Both levels of load imbalance have to be addressed to enable efficient computation and the scalability of the coupling strategy.

8.1. Intra-subdomain

Intra-subdomain load imbalance can cause losings in the overall performance of a coupled simulation; different factors influence it. For example, considering a discretized computational domain with several elements, each of those elements might have a different workload due to the numerical scheme to be solved. If those elements are distributed among several processes with ideally the same workload in a parallel distributed computation, then load imbalances do not occur. Thus, we would expect perfect performance and scalability of the highly parallel and distributed simulation. However, we are aware that this is impossible without appropriate measures to enable an even workload distribution among available processes, especially when the computational elements' workload is diverse. The workload of each element is determined by, e.g., its position in the simulation domain (boundary element) and whether specific properties are prescribed to it (source terms/ boundary conditions).

Here, we discuss several influencing factors that cause load imbalance inside a subdomain. The focus is especially devoted to coupling elements and the source terms for geometries inside the domain and the feature of mode reduction (cf. Section 6.6). We can distinguish in the first place between three different workloads inside the subdomain: **(i)** Elements inside the geometry can be approximated with a reduced polynomial degree due to the mode reduction feature (reduced elements). **(ii)** Fluid elements, that only solve the conservation equations (non-coupling elements). **(iii)** Coupling elements at the boundaries have additional workload as they communicate with the coupling approach (coupling elements) and have to compute the conservation equations additionally. Figure 8.1 illustrates the different types of elements in the simulation domain. The mode reduction feature can reduce the computational effort inside the geometry. This feature allows approximating the solution inside the geometry with only 1^{st} order, as those elements are not of interest for the numerical solution (highlighted in green). Those elements are the most inexpensive ones in the computational domain. Elements located at the domain boundary and those located inside the domain (colored in orange) are more expensive in computation due to the high-order approximation. The most expensive

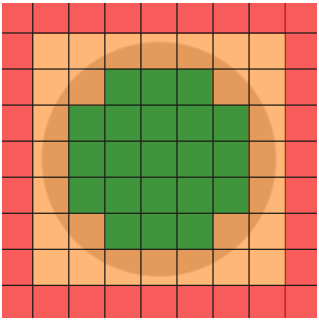


Figure 8.1.: Elements with different workloads inside a subdomain: 1. Green elements with reduced costs inside the geometry when using mode reduction. 2. Orange elements computing the physical equations only. 3. Red elements at the coupling interface have to compute the fluid dynamics equations and communicate with the coupling approach.

elements are those involved in the coupling (marked in red). These types of elements need to solve the conservation equations with a high-order approximation and communicate with the coupling approach. Please keep in mind that different element types might have a different workload, e.g., typical boundary elements again elements that are not coupling elements or the transition from a fine element to a coarser one in the context of multilevel meshes. The workload of other elements is also considered in our examination but not shown in the sketch above.

In coupled simulations, an essential so-called synchronization step at a predefined time is executed. During synchronization, coupled subdomains exchange data and update time step information with their respective coupling partners at their boundary interfaces. The synchronization time step needs to be ideally reached by all processes at the same time. Considering processes that have less workload than other processes, they would need to wait and sit idle until processes with a larger workload complete their respective work.

Figure 8.2 presents an example of workload distribution among four processes, indicated by the different columns. The color indicates the different workloads of each element. In this small example, each process has to

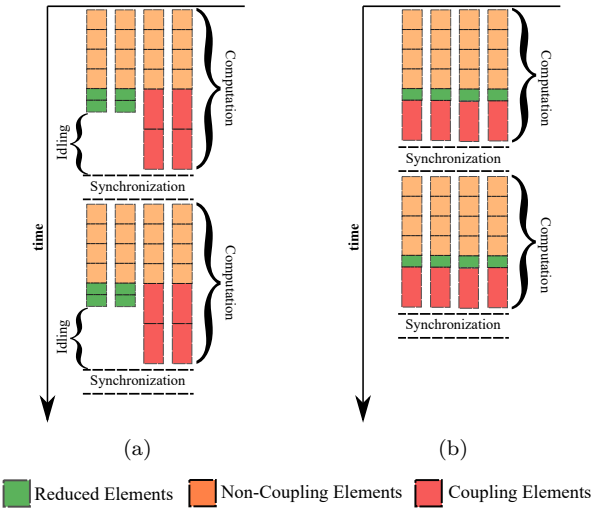


Figure 8.2.: Comparison: (a) Uneven and (b) even workload distribution among 4 processes: 1. Green elements with reduced costs inside the geometry when using mode reduction. 2. Orange elements computing only the conservation equations. 3. Red elements at the coupling interface, compute the conservation equations and communicate with the coupling approach.

compute six elements. The time axis indicates the amount of time required to compute each element per process.

In Figure 8.2a the element distribution among all process is illustrated. Two processes have only elements with a reduced computational effort and elements with a higher computational cost but are not involved in the coupling (non-coupling elements). However, the other two processes have expensive elements to compute, namely coupling elements and non-coupling elements. With this distribution, it is obvious that those two processes without coupling elements have less workload for the computation. They can complete the computation of a time step much faster. Thus, they reach the synchronization time step sooner, resulting in idling time of those processes until the other two processes complete the respective time step. Figure 8.2b demonstrates the ideal workload distribution between the processes. All processes receive, i.a., one coupling element for the

computation. This results in an equally distributed workload, where all processes reach the synchronization time step at a shorter time, and the overall computational effort can accordingly be reduced.

In order to reduce the load imbalance on the intra-subdomain level, we focus in the next section on the LB strategy, which we pursue in our simulation framework, to avoid load imbalances, namely the SPartA algorithm.

8.2. SPartA algorithm

To enable load balancing (LB) for our simulations, we utilize the SPartA algorithm [44, 79], which is implemented in the common data structure $TreElM$ of the $APES$ simulation framework. This algorithm aims to move elements from one process to another to achieve the same or similar workload per process. SPartA is based on space-filling curves to obtain a one-dimensional ordering of all elements among partitions. Individual weights are used that provide information about the actual load of each element inside the domain. The algorithm is well suited for simulations with fixed configurations, such as non-changing computational meshes or equations. Timers (time dimension) are placed around compute-intensive routines to determine the individual weights of each element. The actual workload can be determined with high accuracy as those dominant and time-consuming routines can be captured. They are the main target of the `MPI_Wtime` timers. Routines included in the measurements are, i.a., the compute-intensive physical flux computation and the projection onto the test function. Additionally, timers are used for elements involved in the coupling [59], for multilevel meshes, and for the computation of elements, where geometry is defined [36]. Timers are primarily placed in element loops to allow for precise measurements of the element weights.

In a parallel execution on $nProc_{total}$ processes, we refer to each by its rank in the range of $0 \leq rank < nProc_{total}$. In order to balance the workload, the prefix sum needs to be determined.

$$prefix(N) = \sum_{k=0}^{N-1} w_k \tag{8.1}$$

In Eq. (8.1) the $prefix(N)$ is the workload for N elements and per definition $prefix(0) = 0$. The total workload is the sum of w_k. An optimal

Table 8.1.: Example for the SPartA algorithm, using 16 elements and 4 ranks

Rank before LB	R0				R1				R2				R3			
Weight of elements	5	3	1	2	4	6	1	3	1	3	1	1	1	9	1	1
Prefix sum	0	5	8	9	11	15	21	22	25	26	29	30	31	32	41	42
Optimal range	$0 < 10.75$				$10.75 - 21.5$				$21.5 - 32.25$				> 32.25			
Rank after LB	R0				R1				R2				R3			

workload per process is defined as

$$w_{opt} = \frac{w_{global}}{nProc_{total}}. \tag{8.2}$$

Assuming the workload of each element is captured precisely, then w_{opt} is the optimal weight per process, and w_{global} is the total weight for all elements over all processes using `MPI_Allreduce`. Re-distributing the workload according to Eq. (8.2), each rank should have elements for the computation, that are in the prefix range of $[rank \cdot w_{opt}, (rank + 1) \cdot w_{opt}]$ [79]. In Table 8.1 a small example with 16 elements and four ranks is shown. In this example, the total weight is 43, resulting in an optimal work distribution over all four processes with 10.75. Considering this prefix value for each rank, the new positions for partitioning can be locally determined. In the next step, each process needs to loop over all elements and compare the element weight against the prefix value to identify the new destination of each element, hence keeping the element or moving it to the following process. Before balancing the load, the maximum workload per process was 14, and on process R1, resulting in an imbalance of $\frac{14}{10.75} \approx 1.302$ after re-distribution, this can be reduced to $\frac{10}{10.75} \approx 0.930$.

Applying this approach for our simulations, we first dump a weight file into the disk after a successful run. This practice allows capturing the workload of each element after an actual run. Afterward, the simulation is restarted using SPartA for re-partitioning. SPartA reads out those weight files for each subdomain and re-distributes the elements inside each subdomain across available processes [36]. This procedure is only done at the beginning of the simulation, thus only once for each simulation.

As mentioned previously, besides the intra-subdomain load imbalance, we need to address a different level of load imbalance. The second source of load imbalance for coupled simulations is the so-called inter-subdomain load imbalance explained in the upcoming section.

8.3. Inter-subdomain

In coupled simulations, the different numerical treatment of each subdomain through different equations, scheme order, and mesh resolution lead to individual workload for each subdomain. An example can be the coupling of two subdomains, where one subdomain solves the nonlinear flow equations such as the Euler equations (cf. Eq. (2.11)) and the equations are discretized with a high-order polynomial representation. The second domain is discretized with an even higher polynomial degree and solves the linearized Euler equations (see Eq. (2.12)). The different treatment of each domain leads to a different workload for each of them. Load imbalances are unavoidable if each subdomain does not receive the appropriate amount of processes for the computation. Assuming each subdomain's configuration is fixed and does not change over the simulation time, such as the computational mesh, the polynomial degree, or the equations to be solved, then SLB is sufficient to address load imbalances introduced by the different numerical treatments of each subdomain. Furthermore, an appropriate processes distribution among all subdomains ensures an approximate same computational time to complete a time step (synchronization).

The worst scenario for the inter-subdomain load imbalance is when one subdomain does not receive the expected amount of processes for the computation, resulting in the idling of all other subdomains. Thus their respective processes are unavoidable. Thus, load imbalance on this level is severe, as it is a barrier to the coupling approach's scalability.

To assign the correct number of processes to each subdomain and avoid inter-subdomain load imbalance, we consider the following equation to re-distribute the processes accordingly.

$$nProc_i = \frac{w_i}{\sum_{i=2}^{k} w_i} \cdot p_{total} \tag{8.3}$$

Where $nProc_i$ is the number of processes to be used for each subdomain, w_i the the timing needed for each subdomain individually, obtained from the sum of the weights per subdomain and p_{total} the total number of processes available for the coupled simulation. With that, we can ensure to minimize the idling time between the subdomains; interested readers are referred to [44, 59, 79] for more information.

8.4. Optimization of the gradient computation

As aforementioned, elements involved in the coupling have a more significant workload than other elements in the subdomain. The computational effort can increase even further depending on the values to be exchanged at the coupling interface. For example, assuming the coupling of two subdomains, where an inner and an outer subdomain is defined (see Figure 8.3) and coupled at their joint interfaces. The inner subdomain is solved with the compressible Navier-Stokes equations, while the outer subdomain solves the compressible inviscid Euler equations. The inner domain has all boundary elements involved in the coupling since the outer domain surrounds it. Since the outer domain solves the inviscid Euler equations,

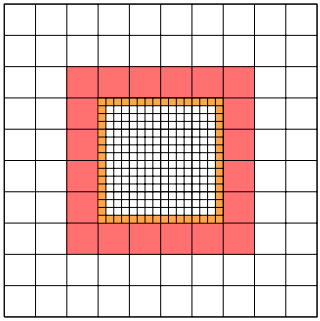


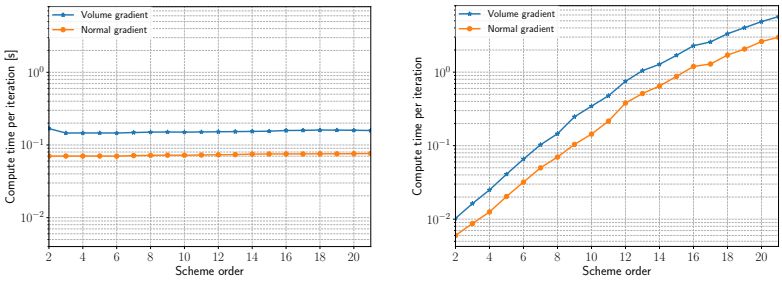
Figure 8.3.: Coupling two subdomains, the inner subdomain is solved for the compressible Navier-Stokes equations and the outer subdomain for the compressible inviscid Euler equations. The computational cost for the coupling elements is different. The outer domain (red) has to compute additional gradient values requested by the inner subdomain. Elements in the inner domain (orange) only have to provide the state variables for the outer domain.

it requires only the conservative variables from the inner subdomain. However, the inner subdomain solves the compressible Navier-Stokes equations; hence, it depends on the conservative variables and their gradients. Thus the outer subdomain needs to compute the gradients for coupling elements of the inner subdomain (at the coupling interface). Therefore, the outer domain's coupling elements are more compute-intensive than those of the inner domain. Considering a three-dimensional simulation and primitive

variables, this results in nine components for the velocity gradient, three components for the density gradient, and three components for the pressure gradient. In total, the outer subdomain needs to compute fifteen additional variables besides the five primitive state variables. These gradients have to be computed in each time step. However, we need to emphasize that only the gradients in the normal direction are of interest; not all components are required when, e.g., one direction has periodic boundaries or is not involved in the coupling.

Therefore, we improve the access pattern of the gradients from volume-data to gradient information per direction. Krupp [59] carried out an intensive examination on the coupling elements and the computation of the gradients. However, her investigations were based on the old implementation; thus, the gradient information was computed for all spatial directions. They are then later internally extracted by the solver to obtain the gradients in the normal direction. More on her investigation can be found in [59]. With the new implementation in this work, only gradients in the normal direction to the coupling interface must be computed; thus, they can be directly accessed. This change allows decreasing the number of gradients to be computed from the previous fifteen to only ten, when boundaries are, e.g., periodic. In this work, coupled simulations have periodic boundary conditions in the z-direction. This change can further decrease the computational effort. In order to investigate this new implementation in terms of computational time, a simple test case is used, where an inner subdomain with $2 \times 16 \times 1$ elements and an outer subdomain with $6 \times 20 \times 1$ elements is simulated. The mesh elements have the same size for both subdomains. In the outer domain, $2 \times 16 \times 1$ elements are removed in the middle of the domain to provide the space for the inner domain. This three-dimensional test case has four coupling interfaces that have to communicate with the coupling tool. The inner subdomain is solved with the compressible Navier-Stokes equations and the outer domain with the inviscid Euler equations. Simulations were executed on a single process (sequential) for each subdomain.

The investigation involves two studies. In the first study, the scheme order of the outer domain is kept the same. However, at the same time, it is changed in the inner subdomain, resulting in more coupling points in the inner domain, hence more points, where the outer domain must provide additional gradient information. For the second examination, the inner subdomain always solves the same scheme order, while the scheme order is varied for the outer domain. This change results in the same



(a) (b)

Figure 8.4.: Comparison of the computational time per iteration of the gradient computation. In (a), the scheme order of the inner domain (Navier-Stokes) is changed, resulting in more coupling points. The outer domain has a scheme order of $O(8)$. In (b), the scheme order in the outer domain (Euler) is changed, while for the inner domain, a scheme order of $O(4)$ is used.

number of coupling points, for which the outer domain has to provide the gradient information. However, the changing scheme order of the outer subdomain affects the gradient computation. Therefore, the scheme order is varied, and the gradient computation is done with the high-order scheme.

In Figure 8.4 the variation of the scheme order, hence the computational time per iteration over the different number of coupling points for the gradients, is depicted. In the case where the scheme order is changed for the inner domain (cf. Figure 8.4a), the scheme order of the outer domain is defined to be $O(8)$. Furthermore, in the case the scheme order of the outer domain is varied (cf. Figure 8.4b), the inner subdomain maintains a scheme order of $O(4)$. The scheme order varies from $O(2)$ to $O(21)$, while the scheme order is increased for each subsequent run for both investigations, respectively. In the Figure 8.4a the computational cost per iteration for the gradient calculation is shown. The blue line depicts the gradient computation in all directions (volume gradient), and the orange line the computation only in the normal direction (new implementation). In both cases, the computational effort is constant, independent from the increasing number of coupling points; thus, the scheme order. As the number of coupling points at the outer domain is the same, the innermost subdomain must provide information for a predefined list of coupling points. However,

the outer domain needs to provide more point information as the number of coupling points increases in the inner domain with increasing scheme order. However, the point evaluation is done with the same scheme order in the case of the outer domain. Comparing the time needed to compute one iteration, the curve representing the gradient computation in the normal direction is roughly by a factor of two faster when compared to the old implementation, where the volume data was computed (gradients in all directions).

In the second examination, the scheme order is varied in the outer domain. Figure 8.4b illustrates both curves, once where the gradient computation is only in the normal direction and once where the gradient is computed in all spatial directions. The computational cost rises for both cases with increasing scheme order (cf. Figure 8.4a and Figure 8.4b). However, in the case of the gradient computation only in the normal direction, the computational effort is reduced by two. Noticeable is that the computational cost is almost constant in the case where the inner domain is varied. However, if the outer domain is varied, the computational effort increases with increasing scheme order. This behavior is due to the evaluation of the polynomial at the requested points. The polynomial degree in the outer domain increases in Figure 8.4b, hence the evaluation requires more time for the computation of the respective gradients. that have to be additionally computed by the outer domain. In the case presented in Figure 8.4a though, the evaluation of the coupling points is done with the same polynomial degree by the outer domain. Therefore the computation is constant as the requested point values for the inner domain are always computed with the same scheme order.

8.5. Load balancing for a 3-field coupled simulation

Through the SPartA algorithm, the load inside the subdomain can be re-balanced to avoid intra-subdomain load imbalance. As previously mentioned in Section 8.1, the workload per element inside the domain might differ. Therefore, a three-dimensional test case is used in this section to demonstrate how load imbalances can be reduced through the LB method exploit in this work. We consider a 3-field coupled simulation, with an airfoil profile acting as a sound source and decomposing the simulation domain into three subdomains.

Test case description: The configuration of the 3-field coupled sim-

ulation is presented in Table 8.2, with the different number of elements and scheme orders used in each of the subdomains. The innermost subdomain solves the compressible Navier-Stokes domain, with a moderate scheme order and a fine mesh, to capture small scales around the airfoil profile (NACA0012). Next, the outermost subdomain solves the linearized Euler equations with the highest scheme order used for this test case. Finally, the middle domain solves the compressible Euler equations. All lengths

Table 8.2.: Small setup for 3-field coupled simulation

	Innermost	Middle	Outermost
Domain length [x, y, z]	[4, 2, 2]	[12, 6, 2]	[12, 3, 2]
Number of elements	94516	8192	1152
Scheme order	4	6	9
nDof	30,245,120	8,847,360	4,199,040

have been normalized by the chord length of the airfoil profile. In the innermost subdomain, a jet-inlet (cf Section 10) is defined that injects the airfoil geometry. It is exactly located in the middle of the domain and has a diameter of 0.5 unit length. The kinematic viscosity μ is set to $1.49 \cdot 10^{-5}$. The velocity at the inflow is linearly ramped and reaches a value of $\vec{v} = [0.1 \cdot \sqrt{(\gamma \cdot p/\rho)}, 0.0, 0.0]$ after a simulation time of 0.75. The isothermal coefficient γ is 1.4, the background pressure p_B and the density ρ_B have a value of 1.0.
Initial conditions The pressure and density value are initially set for all subdomains. In the outermost subdomain, the perturbation is defined to be 0.0 for all quantities.
Boundary conditions Along the left boundary of the innermost subdomain, the jet-inlet is prescribed, with inflow boundary conditions. At the left boundaries of the middle and the outermost subdomain primitive boundary conditions are defined. The upper, lower and the right boundary of the outermost subdomain are outflows. Along the right boundary of the middle domain outflow boundary condition is prescribed as well. All other domain boundaries are involved in the coupling. Physical results of this test case can be found in [36].

The purpose of this test case is to show that due to the additional improvements introduced in this work, namely the mode reduction and the gradient computation only in the normal direction to the coupling interface, the workload inside the domains has increased in diversity, thus has different

element types. As a result, load imbalance can be foreseen and is severe for the scalability and the performance of such large-scale simulations. Therefore the test case introduced before should help to demonstrate that load imbalance on the intra-subdomain level is a barrier for the scalability of large-scale simulations on HPC systems. However, the computational cost and thus the load imbalance can be reduced through the LB strategy deployed in this work.

To demonstrate that our load balancing strategy can properly address load imbalances, two cases are examined. In one case, the scalability measurements are performed using intra-subdomain and inter-subdomain load balancing. For the second case, only the inter-subdomain load balancing is used, where each subdomain receives a dedicated number of processes. As the workload inside the subdomains differs, depending on the elements and the location, we anticipate better scalability when the intra-subdomain load imbalance can be addressed through the SPartA algorithm. The scalability measurements were performed on the SuperMUC-NG system, with 10, 20, 40, 80, 160, 320 and 640 nodes, each node is equipped with 48 cores. The runtime is defined to be 100 iterations. To balance the workload on both levels, weights were first written out and used to re-balance the workload. Afterward, the re-balanced simulation is rerun. IIn Figure 8.5 both cases are depicted. In the case we only distribute the total number of cores among the subdomains depending on their workload (cf. Figure 8.5a), the computational effort is higher compared to the case where intra- and inter-subdomain load balancing (cf. Figure 8.5b) is addressed. In the case only inter-subdomain load balancing is used, the load inside each subdomain is not appropriately distributed, leading to idling time inside the subdomains, indicated by the gaps between the curves of each subdomain. Considering only inter-subdomain load balancing shown in Figure 8.5a, we can recognize that load imbalances introduce, as awaited, a barrier for the scalability. With increasing node count, the curve, presenting the total computational time, deviates more and more from the ideal scaling as load imbalance increases and becomes severe for the scalability. Furthermore, the vertical distance between the curves becomes larger when compared to Figure 8.5b, where intra- and inter-subdomain load balancing is used. The large distance indicates the load imbalance, hence the idling time between the subdomains. While the innermost and middle domains have a similar computational time, the outermost domain is much faster in computation, and therefore, again, idling of the respective cores can be predicted.

In Figure 8.5b we can observe a similar performance of the total time with

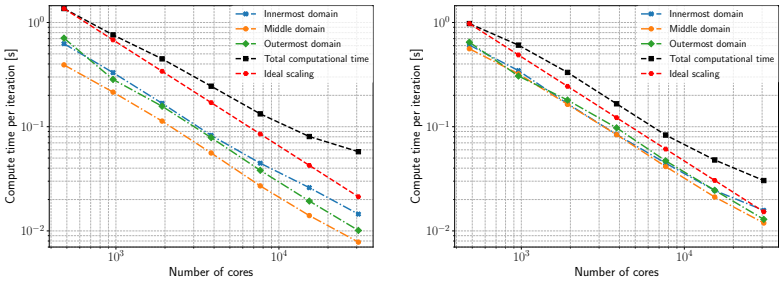


Figure 8.5.: Comparison: Load balancing (LB) for a 3-field coupled simulation. Runs were executed on SuperMUC-NG from 10 up to 640 nodes by doubling the node count for each subsequent data point. (a) Load balancing only between the subdomains (inter-subdomain). (b) Load balancing for intra- and inter-subdomain, considering the SPartA algorithm in the simulation framework.

increasing core count when compared to the ideal compute time; thus, the coupled simulation scales further as expected. Additionally, from the different curves representing the scalability of each subdomain, we can perceive in Figure 8.5b, where both levels of load imbalance are addressed, that the computational time per iteration of all three subdomains is approximately the same. Thus, load imbalances are appropriately reduced (see Figure 8.5b) by utilizing the load balancing approach in the simulation framework. Further, the vertical distance between the curves is smaller compared to Figure 8.5a. Comparing the total computational time in Figure 8.5a and Figure 8.5b e.g., 80 nodes, the total compute time differs by around 32%. At the same time, obviously, for the simulation, where both levels of load imbalance are addressed, the computation is faster. Using two times more nodes for this simulation does not provide the desired speed up (2×). We need to recall that the LB approach we utilize allows relocating elements along the space-filling curve; hence it has only one degree of freedom for the relocation (cf. Section 8.2). Furthermore, coupling elements are more compute-intensive and have a higher workload, even though they might be distributed on different processes. However, they can still be a barrier for the scalability when a distribution is reached that allows no further relocation. Please remember that the coupling also has a certain workload

Table 8.3.: Strong scalability measurements: Core distribution among the 3-field coupled subdomains

	Inter-subdomain LB		
Total core (node) count	Innermost	Middle	Outermost
480 (10)	465	13	2
960 (20)	930	25	5
1920 (40)	1860	50	10
3840 (80)	3720	100	20
7680 (160)	7440	200	40
15360 (320)	14880	400	80
30720 (640)	29760	800	160

included in the total compute time. Therefore we do not expect the sum of the three curves (innermost, middle, and outermost) to result in the total computational time. The total computational time includes e.g. the waiting time between the subdomains and the evaluation of the point values.

In Table 8.3 the core distribution is provided for each run. The required core ratio between the subdomains is kept the same; as neither the mesh nor the equations to be solved change over runtime in each subdomain. From the core distribution shown in Table 8.3, we can recognize that the innermost subdomain requires most of the cores for each run, which is approximately 97% of the total core count. Only 3% are used for the middle and outermost subdomain, with 2.6% and 0.4%, respectively. Even though the innermost subdomain covers the smallest volume in this 3-field configuration, it includes most mesh elements since an airfoil structure is located there and viscous effects play a dominant role, requiring an appropriate resolution.

Conclusion Partitioned coupling has shown in many studies to be an efficient strategy to reduce the computational effort of complex multi-scale simulations. However, this strategy requires an appropriate handling of load imbalance. The decomposition of the simulation domain into subdomains and the different treatment of each of them results in different workloads. This leads to two sources of load imbalance, namely the intra-subdomain and inter-subdomain load imbalance. Furthermore, each subdomain certainly has load imbalance inside the subdomain, as, e.g.,

coupling elements are more compute-intensive due to the communication with the coupling tool or elements that are covered by geometry and have a different workload. Further load imbalance inside and between the subdomains (intra- and inter-subdomain) is severe as it acts as a barrier for computational efficiency. With the LB algorithm used in this work, we can address this bottleneck for coupled multi-scale simulations using timers that capture the workload per element through actual runs on the computing system. This approach is adequate for the simulations in this work, since neither the mesh inside each subdomain nor the scheme order or the equations to be solved change throughout the runtime. Furthermore, we were able to show that idling of processes due to inappropriate load balancing inside the domain can be tackled using the SPartA algorithm implemented in the simulation framework. In the context of large-scale coupled simulations, the load balancing can become even more elaborate when different executables are used for each subdomain. We used the white-box approach in this work, which uses only one executable for the entire coupled simulation. The number of executables is essential when addressing workload imbalances and the distribution of allocated cores among subdomains. On some HPC systems, sharing one node by several executables might not be supported (e.g., Supermuc-NG). Thus, each subdomain has to receive complete nodes, even though this is not required or might increase the load imbalance further. This is, e.g., comparable to our investigation in Section 8.5, where the outermost subdomain needed less than one node for the load balancing procedure.

9. Application examples - Complex moving geometries

This chapter is dedicated to numerical results for different simulation setups. First, simulation results with the geometry modeled as an artificial porous material are presented. The focus is on arbitrary shapes of geometries and their motion. Further, we extend our investigation from Chapter 6 and present further challenging test cases, among others, the rotation of an airfoil profile or the collision of spheres. Finally, we demonstrate how the numerical method, namely the high-order Discontinuous Galerkin method, can deal with different numerical challenges. We must emphasize that all test cases are configured to preserve computational efficiency for all configurations. Therefore for all cases, we consider large computational elements and a high-order scheme, which allows for fewer memory requirements on the computational system compared to a low-order scheme with fine mesh elements. It further reduces the unnecessary computation inside the geometry. Afterward, we exemplary provide performance analysis of two test cases, where physical results are shown.

9.1. Numerical results - Moving geometry

In this section, different numerical challenges are presented when modeling moving geometries. Examinations are first dedicated to simulations in two-dimensional space, where the supersonic motion of a cylinder test case is shown, and the solution is compared to the literature. Afterward, the rotation of an airfoil profile is presented, the preparation for the coupled simulation in Chapter 10. We then move forward and extend the third dimension in space. Here the collision of two spheres is examined in more detail. After that, solutions of three spheres are shown; this test case exemplifies the additional capabilities of the modeling method, where it is insignificant how many geometries are involved for the simulation; they can all be modeled by the utilized penalization approach. Lastly, a test case common in the engineering field, a rotating fan, is presented. It allows covering the case where a device is a composition of more than one geometry.

9.1.1. Supersonic flow - Moving cylinder

The formation of shocks in engineering applications is a common phenomenon. Therefore, it is important that in numerical simulations, the underlying scheme can deal with those strong discontinuities. They cause abrupt changes in the state, and stability issues may occur. For high-order approximation, the challenge is even more difficult to overcome, as we consider the polynomial series to represent the state. It leads to high oscillations of the polynomials, which are unavoidable and have to be appropriately addressed. This test case covers the following numerical challenges: **(i)** Moving cylinder with supersonic speed, **(ii)** shock-wall interaction and **(iii)** shock-vortices interaction.

Test case description: A cylinder with diameter $d = 0.25 \cdot H$ is located inside the simulation domain, which has the dimensions of $[L \times H]$ $[4.0 \times 1.0]$ unit length, with L being the length of the domain and H the height of the domain. The element size is defined to be $H/128$ resulting in a total number of 65,536 elements. The polynomial approximation is of degree five, and the CFL condition controls the time step, which is fixed to a factor of 0.3. The geometry is located near the outflow boundary. The cylinder moves with Mach 1.5 from the outflow towards the inflow throughout the simulation. It leads to an upstream motion of the cylinder, and the fluid is streamed into the domain with Mach 1.5. The relative Mach number is 3.0 for the simulation. Pressure and density have a value of 1, respectively. We consider a weak co-volume filter of $O(24)$ during runtime to control oscillations.
Initial conditions Initially, the fluid has a velocity in positive x-direction of Mach 1.5; the pressure and density are prescribed with the given values.
Boundary conditions Along the upper and lower boundaries, slip walls are defined. At the left boundary, a supersonic inflow, and on the right boundary, a supersonic outflow is imposed.

In Figure 9.1 the density is presented for the entire domain. The position of the cylinder at different stages in time is shown in Figure 9.1a up to Figure 9.1e. The initial values can be observed ahead of the bow shock, while behind the shock, density and pressure increase. Behind the cylinder, the density is decreased due to the supersonic speed of the geometry. The pressure attained a value that is lower than the atmospheric pressure. Furthermore, in Figure 9.1a the flow behind the cylinder resembles a light bulb, which is due to the sudden motion of the cylinder. Additionally, the flow pattern is influenced by the slip wall boundary conditions at the

upper and lower boundaries, where the flow is reflected in the normal direction (cf. Figure 9.1d or Figure 9.1e). Along the upper and lower slip walls, a vortex street can be recognized, known as Richtmyer-Meshkov instability, that is well resolved as a high-order scheme is used. Furthermore, the results confirm that the numerical scheme can deal with strong discontinuities and supersonic motions. The geometry represented here is well modeled, and its motion can be seen as physically accurate when considering all appearing phenomena expected from this test case. These results are in agreement with the report of P. Hu et al. [50]. They also used the Discontinuous Galerkin method to discretize the fluid dynamic equations while considering a level-set approach for modeling the geometry. Comparing the outcomes of our work and those of P. Hu et al., we can observe that our simulation results are well resolved, with appearing small scales that are not apparent in their outcomes. However, in [23] the same test case is used, there we can recognize small scales as in our case, as they use a very fine computational mesh for the simulation. All physical phenomena occurring in this test case can be reproduced and compared to both works [23, 50]. In Figure 9.2 the velocity magnitude for the same test case is shown. Again, we can discern how well small scales are resolved. The physical phenomena are as expected, e.g., the high velocity magnitude behind the cylinder due to the supersonic motion of the cylinder.

We can conclude that this test case provides different numerical challenges, which are troublesome for many algorithms and may cause stability issues, resulting in not realizable simulations. The challenges can be seen as **(i)** high-order spatial discretization and supersonic flows, **(ii)** supersonic motion of the geometry and **(iii)** strong discontinuities (shocks). High-order methods are mostly avoided for simulations with strong discontinuity, as they imply the Gibbs phenomenon. The results show that the utilized high-order discretization can simulate these kinds of problems despite the mentioned challenges.

The simulation of supersonic flows is therefore challenging as the high velocity can provoke stability issues, resulting in small mesh elements (increasing resolution) and tiny time steps. However, due to appropriate co-volume filtering, the overall solution can be smoothed by maintaining high oscillations of the underlying scheme under control and therefore stabilizing the simulation without further refinement in space. This is also the case for the motion of the geometry, which implies a sudden change in the flow field. Furthermore, due to the high-order used for the spatial discretization of the fluid dynamic equations, the different scales can be

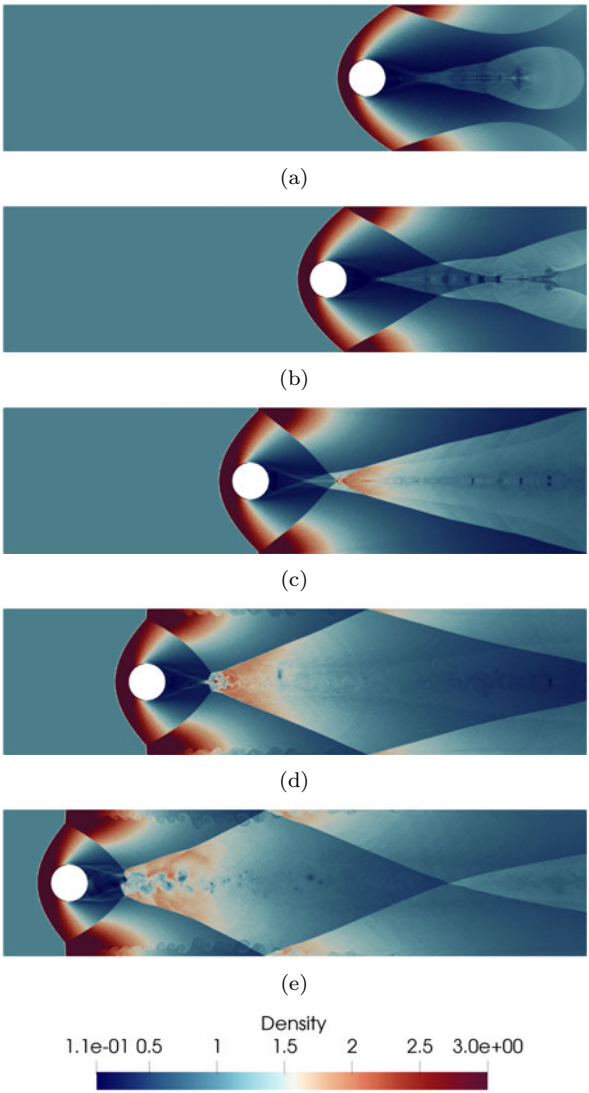


Figure 9.1.: Cylindric geometry moves from an initial location near the outflow towards the inflow with a relative Mach of 3.0. The movement is captured showing the density after a simulation time of (a) 0.35, (b) 0.5, (c) 0.8, (d) 1.2 and (e) 1.5.

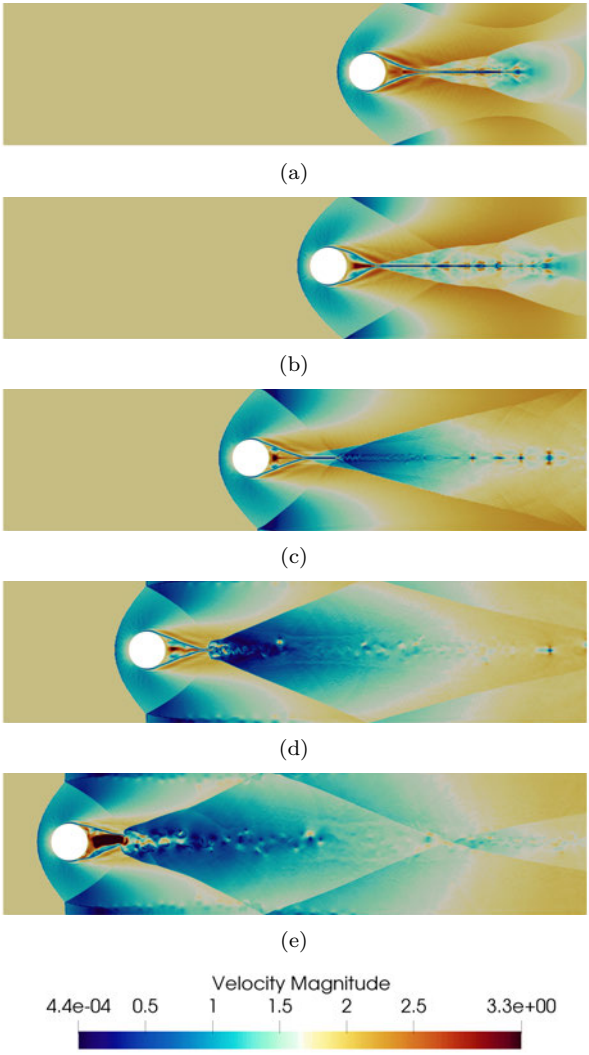


Figure 9.2.: Cylindric geometry moves from an initial location near the outflow towards the inflow with a relative Mach of 3.0. The movement is captured showing the velocity after a simulation time of (a) 0.35, (b) 0.5, (c) 0.8, (d) 1.2 and (e) 1.5.

captured with high resolution as well as the shock structure. Hence we can conclude that even though we consider supersonic flow, the movement of the geometry upstream, and high-order polynomials for the approximation, we can still achieve a detailed insight into occurring physical phenomena.

9.1.2. Rotating airfoil

The next test case is about a rotating airfoil of type NACA0012, where the ability of the embedded boundary method for more complex geometries is further examined. The fluid is at rest, while this changes with the rotational motion of the airfoil.

Test case description: The airfoil profile NACA0012 has a chord length of c and is located with its leading-edge at P(0.0, 0.0). Initially, the angle of attack is 0.0 for the airfoil. The domain size is of size $[8.0c \times 8.0c]$. The pressure has a value of 1, and the density is 1. The velocity is 0.0 in all spatial directions. We use a polynomial degree of five; the mesh is refined with an element size of $c/32$. The spatial resolution is chosen to be fine enough to capture small scales produced by the rotating airfoil. The rotational motion of the airfoil is anti-clockwise, with a rotational speed of $2\pi fr$, with $r = c$ and the frequency $f = 6.283 \cdot 1/t$. The time step size is controlled by the CFL condition with a Courant factor of 0.5.
Initial conditions The fluid is at rest, and the pressure and density are defined as mentioned before.
Boundary conditions The boundary conditions along the domain boundaries are set to Dirichlet boundary conditions, prescribing all quantities in primitive variables. It allows avoiding stability issues of the numerical simulation, as the formation of vortices is expected, which will eventually travel to the outer boundaries without dissipating since physical viscosity is neglected.

As previously mentioned, this test case is only a showcase. We want to demonstrate that our approach can be used for modeling more complex geometries and their motion, common for different engineering applications. In the time series of the simulation results (see Figure 9.3), the Schlieren image (density gradient) is shown. We can observe how the fluid, initially in rest, is influenced by the rotating geometry and causes the formation of vortices. Since this test case is simulated in two-dimensional space and no physical viscosity is present, vortices travel throughout the simulation domain without dissipating. Therefore vortices keep their shape or increase in size when merging with other vortices.

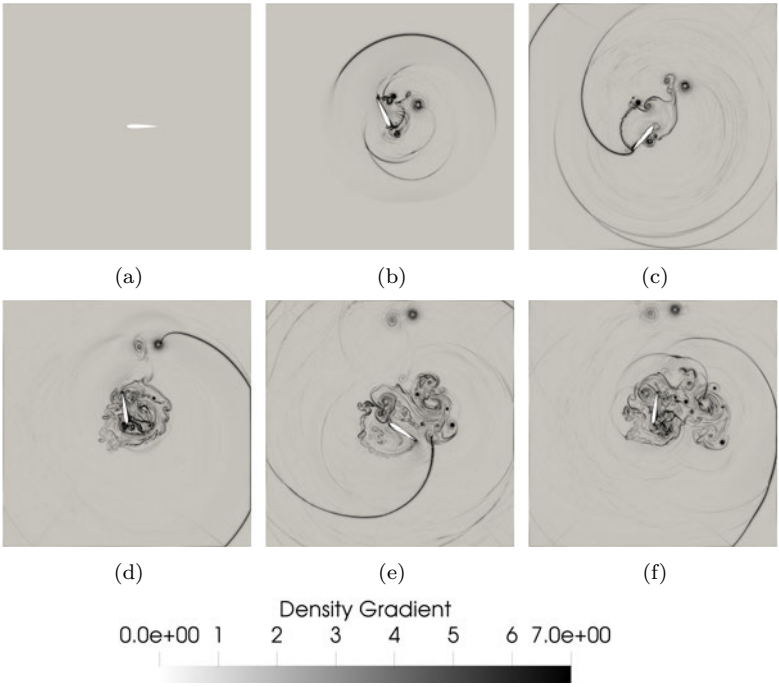


Figure 9.3.: NACA0012 profile modeled as an embedded geometry rotates in the simulation domain, where the fluid is at rest at the beginning of the simulation. The rotational motion is captured by showing the Schlieren image (density gradient) after a simulation time of (a) 0.0. (b) 0.2, (c) 4.0, (d) 8.0, (e) 12.0 and (f) 14.0.

Since the boundary conditions are set to Dirichlet for all quantities, vortices are trapped inside the domain. They can not leave the simulation domain when reaching the outer boundaries (see Figure 9.3f). Further, due to the rotational motion of the airfoil, occurring vortices are influenced by the rotating airfoil, which can be noticed in, e.g., Figure 9.3d or Figure 9.3f. Keeping track of this change, we can recognize how most vortex structures are concentrated around the geometry, as the airfoil prevents their motion. The strongest vortices (black) are due to the first movement

of the airfoil, causing the sudden disturbance of the flow field that was at rest (cf. Figure 9.3a). Furthermore, we can observe that the trailing edge of the geometry causes perturbations resulting in strong waves traveling towards the boundaries. Again the strongest perturbation is caused due to the motion at the beginning of the simulation. As the boundary conditions try to fulfill the predefined state, reflections appear that influence the fluid motion further (see, e.g., Figure 9.3c upper boundary). The mentioned reflections are not of importance for this test case, as the purpose is different, but it can be addressed through sponge layers (cf. Section A.4) that absorb those reflections at the boundaries.

9.1.3. Collision of two moving spheres

In Section 9.1.1 and Section 9.1.2 we presented different challenges that are common in engineering applications, first the formation of shocks and second the rotation of devices (in our case, airfoil profile). To further investigate the embedded geometry modeling, an additional aspect is its suitability for colliding objects. It might be challenging, as the geometries reach their respective interfaces and part ways according to a predefined function. Further, this is from the numerical perspective not always easily practicable, as, in the area where the geometries interact, high compression of the fluid occurs, leading to high density and pressure gradients. Therefore a three-dimensional simulation is examined, where two spheres move with a cosine function and initially have no distance to each other. Throughout the simulation, both spheres move away from each other, reach a maximum position and move back to their respective origin, where they again touch.

Test case description: The simulation domain has a size of $[L \times H \times W] = [4.0 \times 4.0 \times 1.0]$ unit length. The computational elements have a size of $W/16$ resulting in 65,536 elements in total. Both spheres are located in the middle of the domain, with their contacting interface being at $x = 0.0, y = 0.0$ and $z = 0.0$. The original center position of the first sphere is at $\mathrm{C}_1(0.0, 0.2, 0.0)$, the second sphere is located at $\mathrm{C}_2(0.0, -0.2, 0.0)$ (cf. Figure 9.4d). The translational movement is described by a cosine function $Y(t) = Y_0 + A \cdot \cos(2\pi \cdot t)$ in y-direction, and its time derivative gives the velocity of the spheres. Where Y_0 is the shift of the sphere position in y-direction at $t = 0$ and A the amplitude. The amplitude is defined to be 0.1 and the simulation time is 10 unit time. The diameter of each sphere is $0.4 \cdot W$. The Euler equations are discretized with a scheme order of $O(8)$. The high-order scheme allows to model the geometries accordingly and

capture flow features with high accuracy. A co-volume filter of $O(24)$ is applied to keep numerical oscillations small enough to maintain stability. The time step size is controlled by the CFL condition and a Courant factor of 0.3.
Initial conditions At time $t = 0.0$ the velocity in all spatial directions is set to be 0.0, while the density and pressure are prescribed with a value of 1.0, respectively.
Boundary conditions The boundary conditions are defined to Dirichlet conditions, where all primitive variables are defined. The z-direction is defined to be periodic. The choice of primitive boundaries allows avoiding stability issues when vortices travel towards them over the simulation time. Integrating a sponge layer at the outer boundaries would tackle the stability issue and absorb vortices at the boundaries. However, this measure requires a larger computational domain to place the artificial sponge layers (cf. Section A.4). As this section aims to give a small overview of the application possibilities of the modeling method introduced in this work, the Dirichlet boundary conditions are used.

In Figure 9.4 the motion of both spheres away from each other and towards each other is shown for different simulation times. The figures illustrate the different positions of the spheres throughout the simulation. In Figure 9.4a, Figure 9.4e and Figure 9.4i both spheres are moving from their initial position, towards their maximum position, which they reach in Figure 9.4b, Figure 9.4f and Figure 9.4j. Afterward, both geometries move again back to their original position, that can be observed in Figure 9.4c, Figure 9.4g and Figure 9.4k, which they finally reach in Figure 9.4d, 9.4h and Figure 9.4l. This is also the point, where both spheres touch each other. Zoom-in into the contact region between the two spheres at $t = 9.8$ (left) and $t = 10$ (right) for the variable density. White lines indicate the computational mesh. Due to the motion of the spheres, waves can be recognized that move towards the outer boundaries. In the position where both spheres touch each other, the fluid is strongly compressed and squeezed out of the contact area. The density and pressure end up with strong gradients in that region (cf. Figure 9.5 and Figure 9.6 right zoom-in). However, the numerical scheme stays stable and can excellently control this situation. In Figure 9.5 and Figure 9.6 a close-up of the contact area is presented. The numerical solution of density and pressure is shown on the left side of each figure after $t = 9.8$, on the right for $t = 10$, respectively. White lines in the background indicate the computational mesh. The pressure and density values between the two spheres are low as both geometries move towards each other. The fluid in between can ”escape” from this

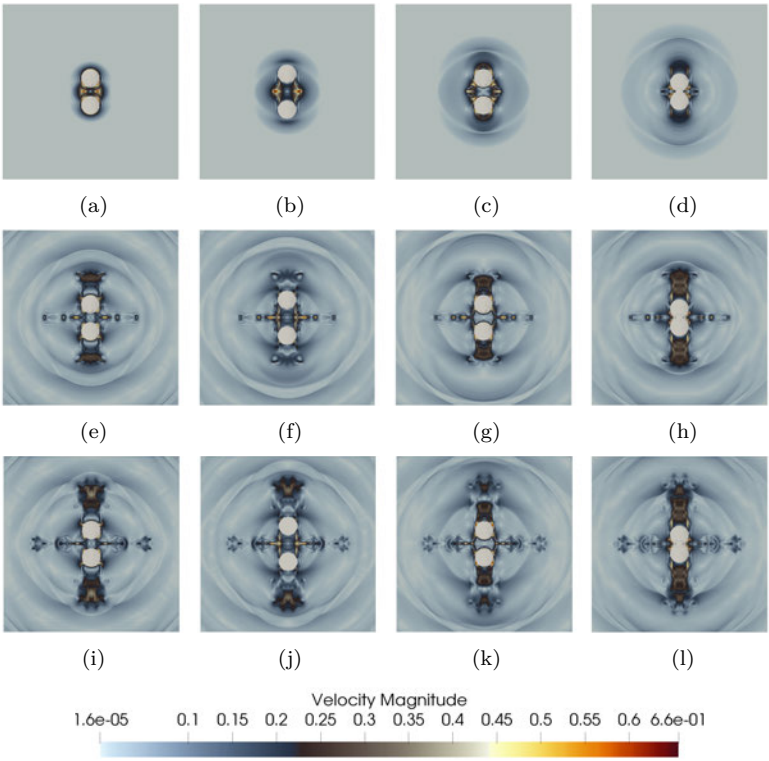


Figure 9.4.: Interaction of spheres, modeled as an embedded porous geometry. Translation movement according to a sine function, with the fluid initially being at rest. The sinusoidal movement is captured by showing the velocity magnitude after a simulation time of (a) 0.25. (b) 0.5, (c) 0.75, (d) 1.0, (e) 5.25, (f) 5.5, (g) 5.75, (h) 6.0, (i) 9.25, (j) 9.5, (k) 9.75 and (l) 10.0. The central point of the upper cylinder is at C_1(0.0, 0.2, 0.0) and the central point of the lower cylinder is at C_2(0.0, -0.2, 0.0) during initialization.

situation by moving to the left and right sides. A closer examination of the contact area, e.g., in Figure 9.5 (the figure on the right), reveals that the contact point is approximated with limited accuracy. While this inaccuracy

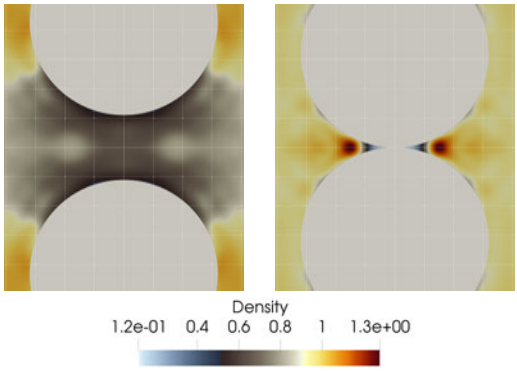


Figure 9.5.: Zoom-in into the contact region between the two spheres at $t = 9.8$ (left) and $t = 10$ (right) for the variable density. White lines indicate the computational mesh.

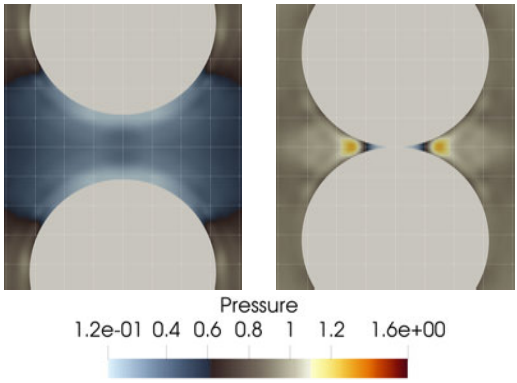


Figure 9.6.: Zoom-in into the contact region between the two spheres at $t = 9.8$ (left) and $t = 10$ (right) for the variable pressure. White lines indicate the computational mesh.

becomes apparent in a close-up solution inspection, it does not lead to numerical instability. Thus, this setup illustrates nicely how the embedded boundary enables a straightforward high-order discretization of arbitrary geometries. Small features, like gaps or cavities, are gracefully taken care of, with a resolution up to the accuracy of the employed discretization. Due to the collision of the spheres, small jet-like phenomena appear that migrate towards the outer boundary.

This example in three-dimensional space presented the simulation result of a typical scenario in the engineering field, the collision of two objects. The underlying high-order numerical method maintained stability, and the utilized modeling method allowed for appropriate modeling of both spheres. Even though we used a high-order of $O(8)$, no stability issues were encountered. Moreover, the high-order Discontinues Galerkin scheme can handle collision of geometries accordingly, which faces high pressure and density gradients at the contact interface. Furthermore, the geometries are adequately modeled, and the high-order scheme nicely provides the simulation results.

9.1.4. Collision of three moving Spheres

In the former section, a three-dimensional simulation with two spheres was examined. The next test case presented here involves three spheres. The geometries are located close to each other so that they get in contact. Throughout the simulation, they move in different directions until a predefined maximum position is reached, where they again take the same path to return to their respective original location. The simulation domain is comparable to the previous test case shown in Section 9.1.3.

Test case description: The three-dimensional test case has a length of $L = 1$ unit length, a height of $H = 1$, and a width W of 2.6 unit length. The first sphere is located at $\mathrm{C}_1(-0.1, 0.0, 0.0)$, the second at $\mathrm{C}_2(0.0, 0.45, 0.0)$ and the third one at $\mathrm{C}_3(0.0, 0.0, -0.45)$. In each spatial direction, we have located one moving sphere. The first sphere moves in x-direction and has a diameter of $L/10$, the second sphere moves in y-direction and has the same diameter as the third sphere, which moves in the z-direction and has a diameter of $W/5$. A cosine function controls the motion of the geometries, that is defined as $C_0 + A \cdot \cos(2\pi t)$ with A being the amplitude, which is 0.1 and C_0 is the displacement in each direction. The first sphere is shifted by -0.1 in x-direction, the second sphere by 0.45 in y-direction and the third sphere by -0.45 in z-direction. The velocity

is prescribed to be the time derivative of the displacement function. The compressible Euler equations are discretized with a scheme order of $O(8)$. For the computational mesh, an element size of $L/64$ is defined, resulting in $98,304$ elements in total. The time step is controlled by the CFL condition with a Courant factor of 0.3. The problem is simulated for 10 time units. A co-volume filter of $O(24)$ is used to keep numerical oscillations small.
Initial conditions Initially, the fluid is at rest, and the spheres are in contact. Further, the density and the pressure are defined to be 1.0 and 1.0, respectively.
Boundary conditions Along the upper, lower, left, and right boundary, the Dirichlet boundary conditions are applied, where all primitive variables are defined. Along the width of the domain, we prescribe periodic boundary conditions to allow for an endless array of motion of the geometry.

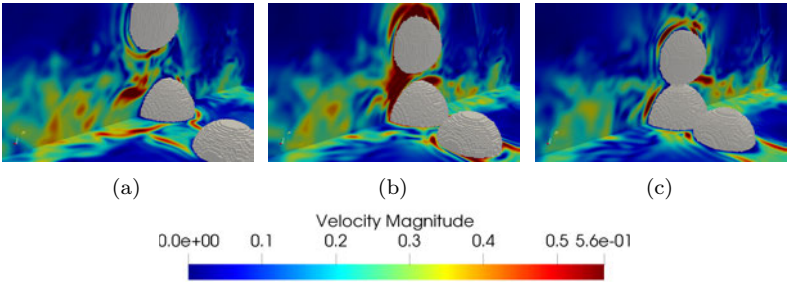
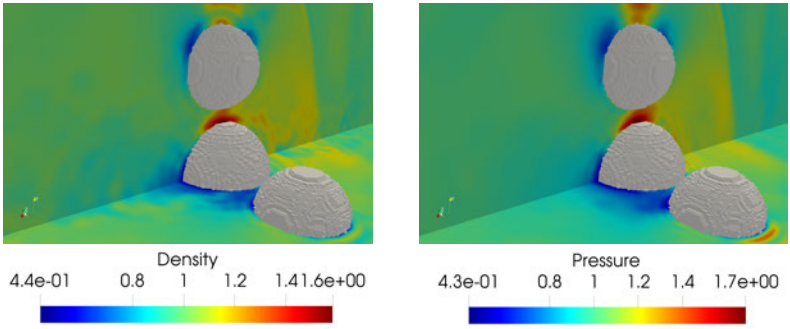


Figure 9.7.: Collision of three spheres with different starting positions. In the background, the velocity magnitude is depicted. In (a), the maximum position of the spheres, the largest distance between the geometries after 9.5, is presented. (b) The spheres move towards their original location after 9.8 simulation time, and in (c), the spheres have reached their respective original position after 10 of simulation time.

Figure 9.7 illustrates three different positions of the spheres during the simulation. The first image presents the position of each sphere when reaching the predefined maximum position (cf. Figure 9.7a), which is also the largest distance between the spheres. In Figure 9.7b the spheres move again towards each other, between the geometries, we can observe how the fluid is pressed out of that region, noticeable through the high velocity value. The spheres return to their original location at $t = 10$, depicted



(a) (b)

Figure 9.8.: Collision of three spheres at $t = 9.8$, when the spheres move towards each other. In the background, in (a) the density and in (b), the pressure is shown.

in Figure 9.7c. Here we can observe that the fluid close to the contact region has a high velocity value. It previously was between the geometries and has been pressed out, resulting in high velocity magnitude. Further in all figures in Figure 9.7 we notice how the fluid that initially was at rest has been disturbed by the motion of the spheres, resulting in vortex structures (cf. Figure 9.9). In Figure 9.8 the solution for density and pressure are shown in the background for the simulation time $t = 9.8$. For both variables, we can observe high values in the vicinity of the contact area between the middle and upper spheres. As the spheres move towards each other, the upper sphere moving downwards and the middle sphere moving from the right to the left side of the domain, the fluid is compressed between the surfaces of the upper and middle sphere. This results in high pressure and density values in that region (cf. Figure 9.8b and Figure 9.8a). With the motion of the middle geometry, the fluid is directed by the geometry to the left (cf. Figure 9.7b). In Figure 9.9 the Q-criterion for a value of 7.0 is shown. The velocity magnitude colors the contours. The high velocity magnitude can be found between the geometries, where the fluid is compressed. Away from the geometries, small and larger contours are visible, caused by the motion of the spheres, as the fluid initially was at rest. Further, throughout the simulation time, the spheres move ten times towards and away from each other, resulting in further disturbance of the fluid flow. This test case demonstrated how well the numerical

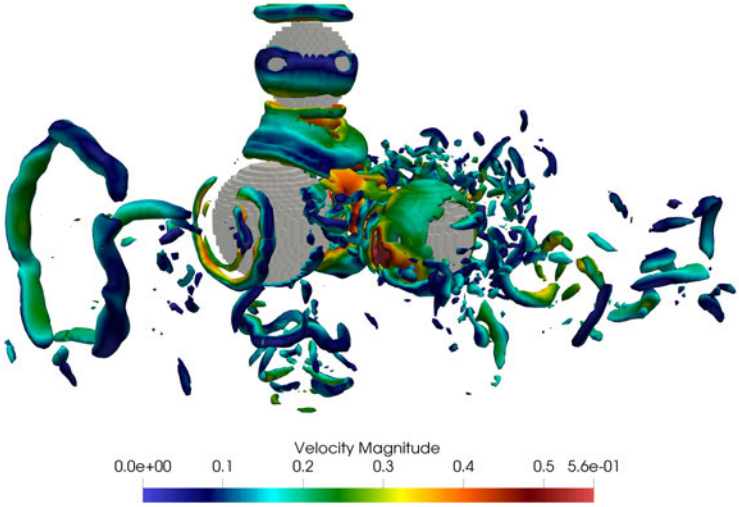


Figure 9.9.: Contour plot of the Q-criterion for a value of 7.0, colored by the velocity magnitude after 9.6 simulations time.

scheme could deal with multiple geometries that move and collide. The main concern of such a simulation is the numerical stability, which the solver excellently handled.

9.1.5. Rotating fan

The previously shown simulation results were simulated with the compressible inviscid Euler equations. We now include the physical viscosity and solve the compressible Navier-Stokes equations for a rotating fan with three NACA0012 airfoil profiles. The purpose of this test case is to show that the numerical scheme can deal with complex geometries, which are a composition of more than one geometry and can move. This test case is especially of interest, as many engineering devices are composed of different geometrical parts.

Test case description: The simulation domain has a length of $L = 10$, a height H of 10 and a width W of 2 unit length. The computational

domain is discretized with $102,400$ cubical mesh elements, with an element size of $W/16$. The fluid dynamic equations are discretized with a scheme order of $O(6)$. The chord length c of the airfoil is defined as $L/10$ and is extruded in the z-direction with respect to the width of the domain. The arrangement of the three blades of the fan is as follows: Initially, the first blade is positioned at an angle of 90, the second blade at 210 and lastly, the third blade at 330. The fan rotates counter-clockwise with a frequency of $6.283 \cdot 1/t$ around the center point P(0,0,1). The Reynolds number has a value of $67,114$ with respect to the chord length and the Mach number is 0.1. The rotational speed of the fan, is $2\pi fr$, with $r = c$.
Initial conditions The background pressure is 1, the density is 1, and the fluid is initially at rest.
Boundary conditions At almost all boundaries of the domain, Dirichlet boundaries are defined, where the background pressure and density are prescribed. Further, all gradients of the state variables are predefined to 0.0. The width of the simulation domain has periodic boundaries.

In Figure 9.10 the time evolution of the rotating fan for the simulation times of 5, 10, 15 and 20 is shown from Figure 9.10a to Figure 9.10d, respectively. The pressure is normalized by the background pressure and is shown as a color field in Figure 9.10. White streamlines illustrate the velocity pattern in the vicinity of the rotating geometry. For example, in Figure 9.10a three strong vortices appear that is due to the initial motion of the fan. Those vortices are still visible in Figure 9.10d, which have moved towards the boundaries and are far away from the obstacle. At the tip of the blades, the meeting point of the highest and lowest pressures can be found. Pressure waves are nicely presented and propagate through the computational domain with a spiral pattern. New waves disturb the outgoing pressure waves as from Figure 9.10b to Figure 9.10d. In the vicinity of the fan, the flow is captured in circulation areas confined by the blades, which is noticeable through shown streamlines in that area. Thus, the numerical scheme wsell represents the geometries, and the observed behavior agrees with the expectations from the physical perspective, e.g., high pressure values at the tip of the blades. Further, the modeling method utilized in this work allows for the composition of more than one geometry without any restrictions from the modeling perspective, and the underlying high-order scheme maintains numerical instability.

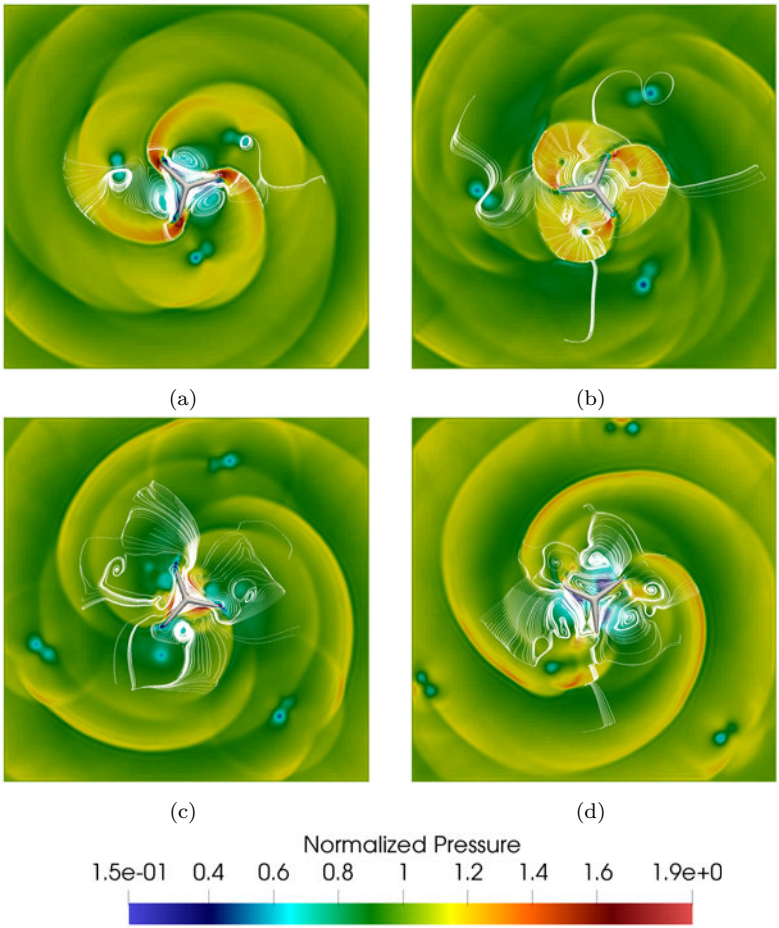


Figure 9.10.: Rotation of a fan composed by three NACA0012 airfoil profiles. The pressure is normalized by the background pressure. The rotational motion is captured after: (a) 5, (b) 10, (c) 15 and (d) 20 unit time. The normalized pressure is shown in the background. The streamlines (white) illustrate the velocity flow pattern.

9.2. Performance results - Moving geometry

This section is dedicated to the performance study of the previously shown test cases. Our investigation includes the rotating fan test case (see Section 9.1.5) and the collision of three spheres (cf. Section 9.1.4). Both test cases are in three-dimensional space, while in the first test case, the compressible Navier-Stokes equations are solved, and in the second test case, the inviscid Euler equations. Due to the presence of the geometries, the computational elements have different workloads. They, therefore, require a dedicated strategy to allow for efficient computation and the reduction of load imbalance. As mentioned in Section 8.2 the SPartA algorithm is deployed to address load imbalances inside the simulation domain. Simulations are carried out on German's national supercomputing system Supermuc-NG, hosted by the Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften in Munich. Each simulation is executed for 100 iterations to obtain the computational weights for the load balancing with SPartA. They are dumped to the disc, and the simulation is restarted for the predefined iterations. The weights are the summation of the computational time per element for 100 iterations to level out jitters.

9.2.1. Rotating fan

A detailed description of this test case and physical results are presented in Section 9.1.5. This test case has, as previously mentioned, a computational mesh that contains in total $102,400$ elements. A box of size $L \times H \times W = [1.125 \times 1.125 \times 1.125]$ unit length is used to limit the search space for the masking function χ (cf. Section 5.5) by the solver. The box allows for efficient computation, as only in the predefined area the solver is required to search and update the state of the moving geometry throughout the simulation time. Consequentially, only in that area, additional computation is necessary. The number of elements that are covered by the box is $4,608$, with 18 elements in x-direction, 16 elements in y-direction and 16 elements in z-direction, resulting in $995,328$ degrees of freedom per variable. Only the fluid dynamic equations are solved for the rest of the domain, resulting in $97,792$ elements, with $21,123,072$ degrees of freedom per variable in total. The area outside of the box has roughly $21\times$ more degrees of freedom. In Figure 9.11 the computational weights for the elements in the simulation domain are exemplary shown. Figure 9.11a illustrates the entire simulation domain and Figure 9.11b a zoom-in into the area where the fan is present and the limiting box is located. Further, white contour lines indicate the location of the fan geometry inside

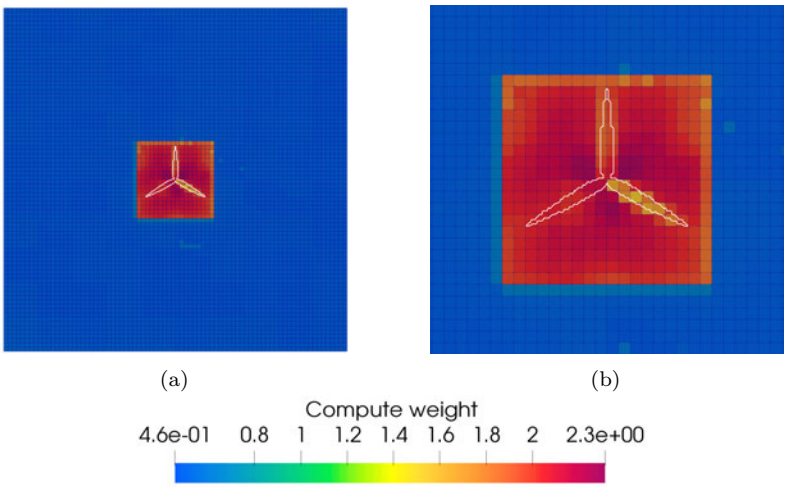


Figure 9.11.: Computational weights are presented for the rotating fan. In (a), the entire computational domain is shown, where white lines indicate the contour of the fan. In (b), a zoom-in into the location of the fan is provided.

the simulation domain. We need to emphasize that the representation of the fan is based on polynomials as required by the numerical scheme used in this work. For visualization purposes, we consider voxelization to allow for further analysis. Therefore the odd appearing fan blade at an angle of 90 is only a post-processing artifact compared to the other two blades.

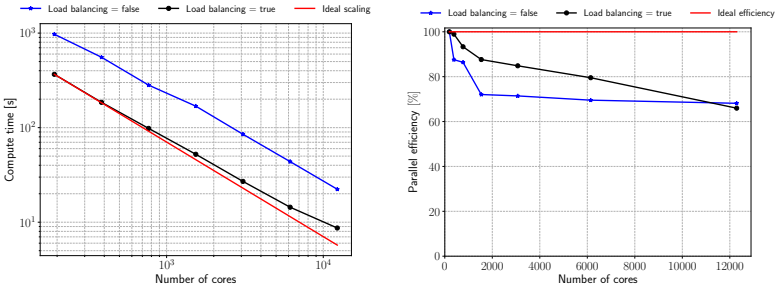
From the color scale, we can observe that elements outside the box have a similar compute time (weight). That is also the case for elements inside the box, even though they are slightly diverse. Elements solving only the fluid dynamic equations (blue) require a compute time of roughly 0.5 s for 100 iterations, while elements embraced by the predefined box need around 4$\times$ more time for the computation.

Assuming the computational weights are not provided in this section, then we can estimate the computational effort per element inside the box, using Eq. (6.15). Our test case is a polygon, where the surface of the fan is represented through three blades. Each blade has its list of

vertices for the surface representation. The list includes 64 vertices for one blade, which results in a computational time of $7.3869 \cdot 10^{-3}$ s for all vertices (3 blades) per integration point for 100 iterations. Considering one element, that has a total number of $6^3 = 216$ integration points, then each element has an additional computational time of approximately $7.3869 \cdot 10^{-3}$ $s \cdot 6^3 = 1.5956$ s, thus we can estimate the required compute time as 1.5956 $s + 0.5$ $s = 2.0956$ s for elements, that are covered by the limiting box. Comparing our estimation and the weights shown in Figure 9.11, we can emphasize that our estimation is in excellent agreement with the actual computational time from the simulation.

In Figure 9.11b one element has a lower computational weight (yellow) inside the fan wing at an angle of 210, which is due to the mode reduction feature introduced in Section 6.6. It allows avoiding expensive computation inside the geometry, where it is not of interest. However, as previously mentioned, we intend to use significantly large computational elements and a high-order scheme to allow for efficient and accurate computation. Therefore the test cases introduced in Section 9.1 are configured with the aim of efficient computation, where large elements are used and a high-order scheme utilized. One might expect to find the same reduced element in the wing on the left side at an angle of 330. However, we need to keep in mind that the integration points represent the fan. Any tiny shift in its positioning on the computational mesh can break the symmetry and result in a slightly different geometry representation. The element is no longer entirely inside the respective element but has a tiny part of the geometry boundary. By that, it is no longer qualified for the mode reduction feature.

In Figure 9.12 the strong scaling measurement and the parallel efficiency are presented. The investigation starts with 4 compute nodes (192 cores) up to 256 nodes (12,288 cores), where the node count is always doubled for the following data point. Three curves are shown in the figures, representing the ideal scaling (red), the strong scaling without load balancing (blue), and the scaling with load balancing (black). The load balancing strategy utilized in this work can adequately reduce the load imbalance. The computational effort is reduced by around 33% compared to where no load balancing is used for this particular test case. Furthermore, the computational effort still decreases with an increasing number of cores (black curve). After the third data point, the black curve deviates from the ideal scaling curve with a doubling core count, yet the computational time decreases further.



(a) (b)

Figure 9.12.: Strong scaling and parallel efficiency measurements for the rotating fan test case with 4 nodes, up to 256 nodes on Supermuc-NG. Each compute node is equipped with 48 cores. In (a) the strong scaling measurement and in (b) the parallel efficiency is shown.

To explain the reason for this deviation, we consider a small example, shown in Table 9.1. Our example takes 16 computational elements into account. Some have a weight of 1, while others of 4, as in our test case, with elements inside the box roughly 4× more expensive in the computation. Furthermore, the 16 elements are distributed among 8 ranks (cores) for the computation. Through the summation of all element weights, the total computational effort is 28. If no load balancing strategy is incorporated for the simulation, the computational elements are equally distributed among available ranks. However, not all elements have the same workload. This practice will affect the performance badly, as R2 and R5 have all compute-intensive elements to process (cf. Table 9.1 second row). This is comparable to our observation in Figure 9.12b, where the efficiency is badly affected (cf. blue curve), as the computational elements are not accordingly distributed. However, with load balancing (LB), the prefix is for each rank 28/8. Thus the total sum of the weights is divided by the number of ranks, resulting in 3.5. The SPartA algorithm can re-distribute the elements according to the prefix, yet, it needs to follow the space-filling curve for the re-distribution. It allows for one degree of freedom for the relocation of the elements among available ranks. The last row in Table 9.1 presents the new distribution after LB. We can observe that

Table 9.1.: Example of the SPartA algorithm, using 16 elements and 8 ranks

Rank before LB	R0		R1		R2		R3		R4		R5		R6		R7	
Weight of elements	1	1	1	1	4	4	1	1	1	1	4	4	1	1	1	1
Prefix sum	0	1	2	3	4	8	12	13	14	15	19	23	24	25	26	27
Optimal range	$0 < 3.5$				$3.5 - 7.0$	$7.0 - 10.5$	$10.5 - 14.0$		$14.0 - 17.5$		$17.5 - 21.0$	$21.0 - 24.5$	> 24.5			
Rank after LB	R0				R1	R2	R3		R4		R5	R6	R7			

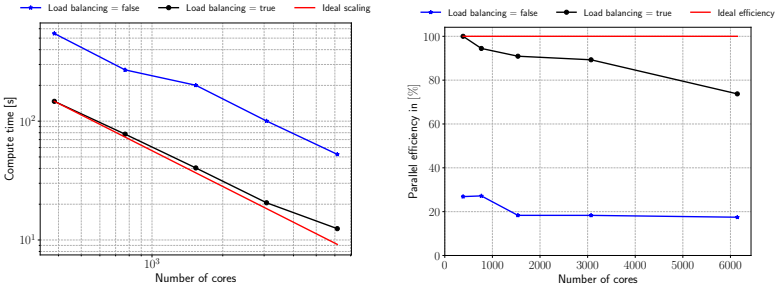
the algorithm tries to re-distribute the elements such that each rank has a similar workload. However, we can also recognize the limitation of our load balancing strategy due to the restriction given by the space-filling curve. Some ranks still have less workload to process. We further need to emphasize that each rank has to process at least one computational element.

In our example, all ranks have less workload when compared to the defined prefix. For example, R1 has only one element as well as R2. Suppose the number of ranks is further increased, it is impossible to find a distribution that further reduces the computational effort, as those compute-intensive elements already occupy a rank on their own. Furthermore, as we restrict the SPartA algorithm to follow the space-filling curve, we have a specific redistribution limitation. However, we must emphasize that even though this limitation exists, the imbalance can be accordingly reduced, as shown in Figure 9.12. Thus the deviation from the ideal scaling can be explained by the limitation we encounter when redistributing the computational elements and only allowing for relocation according to the space-filling curve. Further, we can observe from Figure 9.12b that the parallel efficiency is much higher compared to the case where no LB is used (blue curve). The efficiency drops for the case with LB (black curve), but we can still achieve a parallel efficiency of over 80% for up to $6,144$ cores. With $12,288$ cores, the efficiency drops to 65%, which is much higher compared to the case where no LB is used. The parallel efficiency there (blue curve) attains a value of less than 30%. We need to emphasize that the reference time used to compute the parallel efficiency is the first data point of the balanced test case. Since this test case is too large to be executed sequentially, the first balanced data point serves as reference for both test cases (no LB and with LB).

We now continue our performance investigation with a second test case, the collision of three spheres, as shown in Section 9.1.4.

9.2.2. Collision of three moving Spheres

In Section 9.1.4 we already have shown the physical results of this test case. We now continue our investigation in terms of performance examination. Again the strong scaling behavior, as well as the parallel efficiency, is studied (cf. Figure 9.13). For efficient computation, the search space of the masking function χ is again limited to a box of size $L \times H \times W = [0.7 \times 0.85 \times 0.85]$ unit length, thus 12 elements in the x-direction, 14 in the y-direction and 14 elements in the z-direction. The computational domain contains $98,304$ elements in total, where $2,352$ elements are inside the box used to find and update the location of the masking function. Thus, $95,952$ elements only have to compute the fluid dynamic equations and, therefore, have a smaller workload than the $2,352$ elements that additionally need to compute the masking function. Elements inside the



(a) (b)

Figure 9.13.: Strong scaling measurement and parallel efficiency for the collision of three spheres, using 8 nodes up to 128 nodes on Supermuc-NG. Each compute node is equipped with 48 cores. In (a) the strong scaling measurement and in (b) the parallel efficiency is shown.

box approximately make 2.45% compared to elements that are outside of the box. In Figure 9.13a the strong scaling measurement is presented, starting with 8 nodes (384 cores), up to 128 nodes ($6,144$ cores). Obviously, without LB (blue curve), the computational effort is significantly higher than the case where LB is considered (black line). With load balancing measures, the computational cost for 100 iterations can be reduced by around 29%, compared to the case where no load balancing is used. Thus, even though the scaling behavior is not equal to the ideal scaling, it is

very close and provides exceptional savings in the computation compared to the case where no load balancing is used. The same behavior can be observed for the parallel efficiency (cf. Figure 9.13b), with an efficiency of over 70% for 128 nodes. However, the parallel efficiency for the case, where no load balancing is used, is worse and results in less than 20 %. Thus the load balancing strategy utilized in this work allows for the decrease of load imbalance and efficient computation.

Conclusion In this chapter, we presented two- and three-dimensional simulation results of curved geometries in motion. The setups are specifically chosen to highlight possible troublesome scenarios for the high-order discretization. Presented simulation results are in excellent agreement with physical expectations and highlight how well the deployed method can overcome different challenges. Furthermore, due to the presence of geometries and the limiting box used to identify the masking function and reduce unnecessary computation, the workload inside the simulation domain is diverse. Elements inside the box have a more significant workload, while elements outside the box are less compute-intensive. It results in load imbalance that can be accordingly addressed through the SPartA algorithm utilized in this work. Finally, we demonstrated how well the computational cost could be reduced by employing strong scaling measurements and parallel efficiency plots. In both explained cases, the decrease in computational time was more than 28% and the efficiency over 60% for the highest node count.

With the outcomes of this and the previous section, we continue our investigation. We realize a 3-field coupled simulation with a moving airfoil that perturbs the fluid flow throughout the simulation time and acts as a sound source.

10. Numerical results - Coupled 3-field simulation

In this section, numerical solutions for a coupled multi-scale problem are presented. We investigate the test case where an airfoil varies its angle of attack throughout the simulation time and perturbs the fluid flow. The primary purpose of this study is to capture the flow-induced noise due to the jet-inflow and the geometry's motion. Moreover, we aim to demonstrate that the coupled simulation is well suited to produce meaningful physical results and efficiently computes multi-scale problems. The coupled simulation was executed on the hawk system hosted by the High-Performance Computing Center (HLRS) in Stuttgart, Germany.

Test case description: The simulation domain is decomposed into three subdomains. First, in the innermost domain, the compressible viscous Navier-Stokes equations are solved. Here an airfoil profile of type NACA0012 is located. Next, the fluid dynamic equations are simplified away from the viscous effects, and the compressible inviscid Euler equations are solved in the middle domain. Finally, away from all nonlinear effects, the equations can be further simplified. Here the outermost subdomain is located, and the linearized Euler equations are solved. A sketch of this coupled simulation is shown in Figure 10.1. The width of all subdomains is considered to be 4 m, with periodic properties at the boundaries. The Mach number is set to 0.4 and the Reynolds number Re is $10,790,418$, with respect to the chord length and the kinematic viscosity $\mu = 1.532 \cdot 10^{-5}\ m^2/s$. The time step Δt is predefined to $1.5 \cdot 10^{-6}\ s$ for all subdomains.

Innermost subdomain: Navier-Stokes domain This domain solves the Navier-Stokes equations. A NACA0012 airfoil is located with its center leading edge at P(0.0, 0.0, 0.0). It has a chord length of 1 m, a wingspan of 4 m (infinity, due to periodicity) and a maximum thickness of 0.12 m. The airfoil moves throughout the simulation time with a predefined sinus function, that is defined as $\alpha(t) = A \cdot \sin(2\pi \cdot t)$. The amplitude A has a value of 0.213, and the time is given with t. We consider, as previously

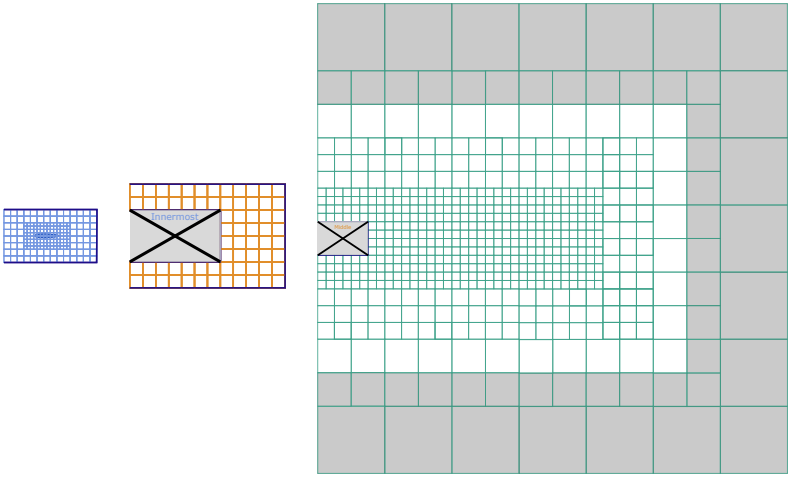


Figure 10.1.: Sketch of the 3-field coupled simulation: Decomposition of the problem size into smaller subdomains. (a) The innermost subdomain includes a NACA0012 airfoil and solves the compressible Navier-Stokes equations. (b) In the middle subdomain, the Euler equations are solved, and a coarser mesh is used. (c) The outermost subdomain solves the linearized Euler equations and has an even coarser computational mesh. Different mesh levels are highlighted through more significant elements. Gray areas close to the boundaries indicate the location of the sponge layers. Dark blue lines mark the boundaries involved in the coupling.

mentioned, an implicit-mixed-explicit (IMEX) time-stepping scheme for the discretization in time, where the stability of the numerical scheme is dependent on the CFL condition, which can be computed according to Eq. (3.24). From Eq. (3.24), we know that the time step is $\sim h^2/O^4$ with O signifying the scheme order and h the element length of the computational mesh. A moderate scheme order of $O(4)$ is used, enabling a reasonable simulation time with larger time steps. Small mesh elements are chosen to capture small-scale phenomena. This practice permits larger time steps, as the element size only affects the stability criterion by a power of 2.

Further, we need to emphasize that the previously mentioned time step size of $1.5 \cdot 10^{-6}$ s for the coupled simulation is restricted by the stability of this subdomain.

The mesh of this subdomain is the finest when compared to the middle and outermost subdomains. It consists of two different element lengths. Close to the airfoil geometry, the elements are chosen to be smaller, being 1/32 of the chord length of the airfoil. It allows resolving the boundary layer at the geometry interface. Away from the airfoil, the mesh elements become larger by a factor of 2. This subdomain has a size of [-2, -2, -2] x [4, 2, 2] m (cf. Figure 10.1a). The smaller cubical elements have an edge length of 0.03125 m and the coarser ones a length of 0.0625 m. In total this subdomain contains $511,488$ elements, with $16,128$ elements involved in the coupling (coupling elements), resulting in a total number of $258,048$ coupling points. The Prandtl number Pr is 0.716, with a thermal conductivity of 0.02587 $W/(mK)$ and a specific heat of 1003.691 $J/(kgK)$. The ideal gas constant R is 287.078 $J/(kgK)$ and the temperature T is initially 20 $°C$.

Initial conditions Initially, the fluid is at rest with $\vec{v} = [0.0, 0.0, 0.0]$ m/s. The background pressure is 101325 Pa and the background density is 1.204 kg/m^3.

Boundary conditions The fluid is streamed into the simulation domain from the left boundary through a jet-inflow. The fluid velocity is ramped over a time of 0.15 s into the domain using a sine function $sin(\pi/4 \cdot t)$. The fluid velocity that enters the simulation domain through the jet is defined as

$$u_x = \hat{u}_x \cdot 0.5 \cdot \left(1 + \tanh\left[\frac{r_0}{2d}\right]\right) \tag{10.1}$$

with $\hat{u}_x = Ma \cdot c$, whereas c is the speed of sound and has a value of 343.25 m/s and r_0 is the radius of the jet with a size of 0.5 m. The jet center is all time located at J(0.0, 0.0, 0.0) and the momentum thickness is predefined as $d = \frac{r_0}{20}$. The velocity in y- and z-direction is defined to be 0.0 m/s. The density is adapted to the ramped inflow velocity, according to the Crocco-Busemann relation [15]

$$\rho = \hat{\rho} \cdot \left(1 + 0.5 \cdot (\gamma - 1) \cdot Ma^2 \cdot \frac{u_x}{\hat{u}_x} \cdot \left(1 - \frac{u_x}{\hat{u}_x}\right)\right)^{-1}. \tag{10.2}$$

Apart from the left boundary, all other boundaries are involved in the surface coupling, where the coupling partner provides the required variables and vice versa. The boundary in the z-direction is, as previously mentioned, defined as periodic.

Middle subdomain: Euler domain The middle subdomain solves the compressible inviscid Euler equations. The domain size is defined to [-2, -10, -2] x [47, 10, 2] m, where an area of [-2, -2, -2] x [4, 2, 2] m is removed from the simulation domain, since the innermost subdomain is located there (cf. Figure 10.1b gray crossed box). With that both subdomains coincide only at their interfaces (cf. Figure 10.1b). This domain is located between the innermost and the outermost subdomain. The primary purpose of this subdomain is to allow for smooth transport of the information from the innermost subdomain up to the outermost subdomain, as both domains (innermost and outermost subdomain) require different configurations due to the different scales that have to be resolved in each of them, respectively. An abrupt change of the physical equations and the spatial configuration from the innermost subdomain directly to the outermost can result in numerical instability and visible discontinuities at the coupling interface.

The mesh is coarser in this domain while the scheme order is increased. Thus, as shown in Section 6.1, we can attain with the same number of degrees of freedom a numerical solution that has a smaller error for higher scheme orders compared to the case, where a low-order scheme (e.g., $O(2)$) is used. In [33] we further showed for a 2-field coupled scenario that in order to achieve a similar approximate accuracy in the solution with a scheme order of $O(4)$ and a computational element of length 0.0625 m a higher scheme order of $O(6)$ can be thought of, with corresponding mesh elements of size 0.125 m. Even though in both mentioned investigations, the test cases were different from the test case introduced in this chapter, we can still use those observations for our test case to determine a spatial discretization that allows us to capture scales of interest and maintain the computational efficiency for the middle subdomain. As mentioned, to have a similar resolution in the innermost and middle subdomain, the middle domain can be resolved with a scheme order of $O(6)$ and mesh elements with an edge length of 0.125 m. Hence, elements that are two times larger compared to the coarse elements of the innermost subdomain. However, this configuration would lead to a setup that resolves small scales further, which were necessary for the innermost subdomain but have a minor influence on the evolution of the far-field acoustics. As our aim is the efficient computation of such extensive simulations, and we know

that large scales significantly contribute to the evolution of the acoustics far-field, the spatial resolution is chosen to be not as high as in the innermost subdomain. The scheme order in this domain is $O(6)$, and an element length of 0.25 m is used, resulting in a level jump of 4 compared to the innermost subdomain. This practice allows for efficient computation, as only scales of interest are resolved, and the memory consumption of this subdomain is reduced. The mesh has a total number of $186,368$ elements. Around 960 elements are coupled to the innermost subdomain, thus maintaining communication with the coupling tool, resulting in a total of $34,560$ coupling points. Further, this subdomain has $36,160$ coupling elements to the outermost subdomain, with $1,301,760$ coupling points at the interface.

Initial conditions The background pressure and the density are prescribed with the same values as for the innermost subdomain. The fluid is also here initially at rest.

Boundary conditions At the left boundary, Dirichlet boundary conditions are defined, with the same pressure and density values as for the initial condition. The fluid is at rest. In z-direction, periodic boundaries are defined. All other boundaries are coupling interfaces and are involved in the coupling procedure. The upper, lower, and right boundaries are coupled to the outermost subdomain. While inside the domain, it is coupled to the innermost subdomain.

Outermost subdomain: Linearized Euler domain In the outermost subdomain, the linearized Euler equations are applied. They are inexpensive in the computation, as they can be computed quadrature-free, and only at the coupling interfaces, a transformation of modal data to nodal data and vice versa is necessary (cf. Section 4.1.1). Therefore, we can utilize a high-order of $O(9)$ to discretize the equations, which offers an accurate transport of acoustic pressure waves and efficient computation. The choice of high-order allows using even larger elements for the computational mesh compared to the innermost subdomain. The numerical scheme provides low dissipation and dispersion error, offering the accurate transportation of acoustic information up to the far-field. The simulation domain is of size [-2, -100, -2] x [198, 100, 2] m, while again [-2, -8, -2] x [47, 8, 2] m is cut out (cf. Figure 10.1c gray crossed box), to enable the surface coupling to the middle domain. The mesh contains $127,758$ elements in total, while $2,280$ elements are involved in the coupling to the middle subdomain, thus in total $184,680$ coupling points. The computational

mesh contains multiple mesh levels for the simulation domain. Close to the outer boundaries, the mesh becomes coarser. Here the computational elements are the largest with a length of 2 m and 4 m. Everywhere else, the mesh elements have a length of 1 m (cf. Figure 10.1c). Close to the outer boundaries, the elements become even coarser, leading to only one element in the z-direction. They have an element length of 4 m. This practice is utilized to increase the influence of the numerical viscosity to smear the numerical solution before reaching the outer boundaries. Hence, we try to reduce the possibility of reflections by outgoing acoustic waves.

Additionally, sponge layers are positioned in the vicinity of all outer boundaries of this subdomain to reduce the possibility of further reflections [39] (cf. Section A.4). The sponge layers act as source terms on the rhs of the conservation equations. It forces the solution to the desired value, reducing the magnitude of the density, pressure, and velocity before reaching the boundaries. A linear damping function is used for the sponge, which slowly reduces the fluctuations' magnitude to zero. The strength of the sponge is computed with the speed of sound, divided by the width of the sponge, and multiplied by the total simulation time. The width is 14 m, and the resulting strength of the sponge is 24.5, with respect to the speed of sound $c = 343.25\ m/s$ and the simulation time of $t = 1\ s$.

Initial conditions The pressure and the density are prescribed as background values, likewise for the innermost subdomain. Initially, all perturbation values of the state variables are defined to be 0.0.

Boundary conditions In z-direction, periodic boundaries are defined. Finally, outflow boundary conditions are prescribed at all other outer boundaries, where the pressure perturbation is 0.0, while all other values are extrapolated. The coupling interfaces are only in the inner part of the domain, where the middle subdomain is located. Sponges are positioned at the upper, lower, and right boundaries to suppress reflections.

To allow initial shock waves to leave the computational domain and enable the simulation to level out, a startup computation with reduced spatial scheme order is used. This startup phase is run for 1 s, which is equal to one period of the used sine function for the motion of the airfoil. A scheme order of $O(2)$ is used, and a time step size of $8.0 \cdot 10^{-6}\ s$. This practice is used to save computational resources, as not only do the initial shock waves need to leave the simulation domain, but it is also helpful, as flow needs to evolve. The simulation is restarted after 1 s and the previously

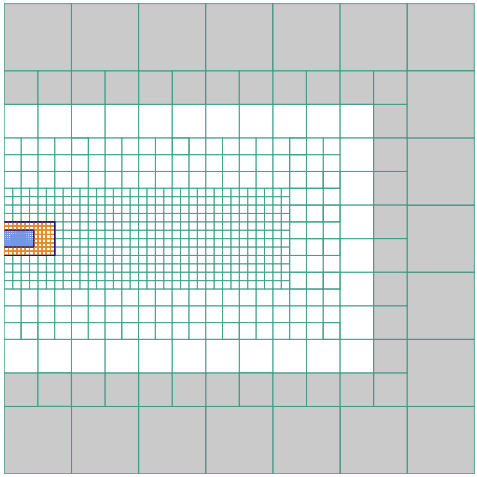


Figure 10.2.: A Sketch of the 3-field coupled simulation: Composition of all subdomains (innermost, middle, and outermost) to the large simulation domain.

mentioned scheme orders are used for the respective subdomains. Consequently, the computational time can be reduced accordingly, requiring about 2.9 Mio. core-h for the entire simulation, where the startup phase needed 300.000 core-h and the simulation with the high-orders further 2.6 Mio. core-h for another 1 s. We need to emphasize that the simulation would have required 5.2 Mio. core-h, in the case, the startup phase would also have been simulated with a high-order scheme. In the following, the simulation results are discussed in more detail. We need to emphasize that the startup phase is neglected and the physical results shown are only for the times where the high-order scheme is deployed. Therefore, the results shown in the following sections are per our definition starting from $t = 0.0$ s. The focus is first devoted to the results of the innermost subdomain that we will analyze in more detail. Afterward, the middle domain is attached to the innermost subdomain, and obtained results are further discussed. Hereafter, the outermost subdomain is attached to the middle domain for further investigation. Lastly, the coupling interfaces are shown for critical areas in the simulation domain. Please keep in mind that the simulation is executed all-time with all three subdomains and only for better analysis detached.

10.1. Innermost subdomain: Compressible Navier-Stokes

In the innermost subdomain, the moving airfoil is positioned. The fluid is initially at rest; however, the flow field is disturbed due to the jet stream and the moving airfoil. In Figure 10.3 the velocity in the innermost subdomain at different simulation times is shown. The time series also presents the motion of the trailing edge, first downwards and later upwards. A complete period requires 1 s simulation time. We need to emphasize that due to the airfoil's motion, we can capture the change in the flow field for various angles of attack of the airfoil. In Figure 10.3a the initial position of the airfoil is shown. The simulation has been restarted after the initial shock waves have left the simulation domain, and the scheme order is increased from $O(2)$ (startup phase) to the higher order of $O(4)$. The airfoil moves according to the predefined sine function throughout the simulation time, first in a downwards motion (cf. Figure 10.3b and Figure 10.3c). Figure 10.3c illustrates the position of the trailing edge after 0.25 s, where a maximum angle of attack of 12.2° is attained, before it moves back upwards, towards its original position in Figure 10.3e. At $t = 0.5$ s the airfoil is finally back at its initial position, while being still in motion in upwards direction (cf. Figure 10.3f and Figure 10.3g). In Figure 10.3g an angle of attack of -12.2° is attained. The angles of attack in Figure 10.3b and Figure 10.3f are 8.6° and -8.6°, respectively. We need to emphasize that the motion of the airfoil is continuously according to the predefined sine function, and in Figure 10.3 we always capture an instantaneous angle of attack of the airfoil as well as the flow field. The time series presents how the jet flow is disturbed by the moving airfoil. At the nose, the stagnation point can be identified, noticeable in all figures in Figure 10.3. Thus, the jet flow is divided into an upwards and a downwards stream. Behind the airfoil, the flow field is further disturbed by the trailing edge, causing a downstream wake that is turbulent. The velocity magnitude has decreased considerably away from the geometry towards the end of the simulation domain compared to the flow entering the domain.

Due to the changing angle of attack of the airfoil, the stagnation point is slightly shifted upwards in the case of downwards motion and shifted downwards when the trailing edge moves upwards (cf. e.g., Figure 10.3b and Figure 10.3g). Furthermore, the flow is no longer laminar close to the geometrical surface due to the inflow and the geometry's presence, which changes its angle of attack throughout the simulation. Thus, we can observe in the time series how the transition from the laminar boundary layer to the turbulent boundary layer occurs, depicted in, e.g., Figure

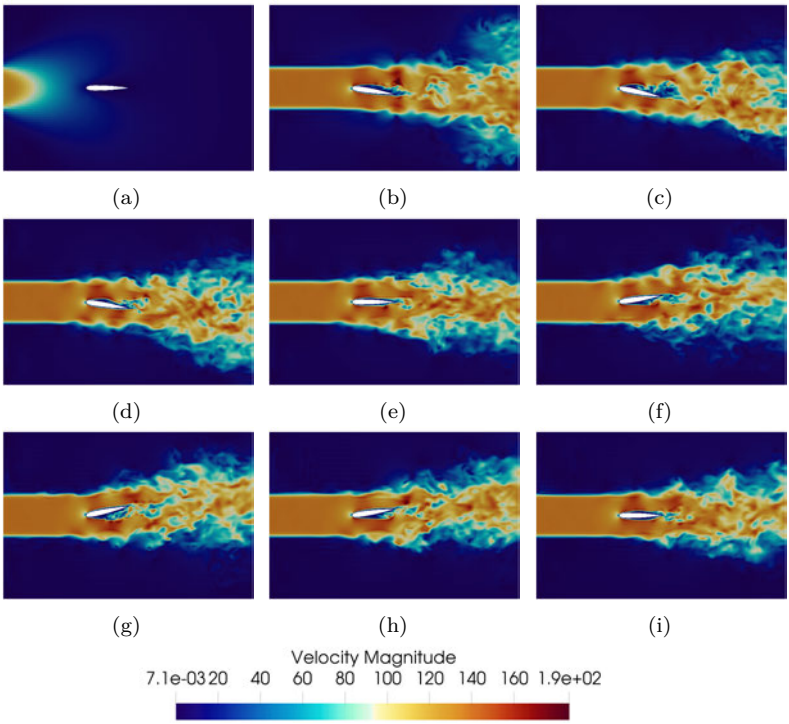


Figure 10.3.: The time series of the innermost subdomain for the quantity velocity magnitude in the negative z-normal. The velocity magnitude and the different angle of attacks of the airfoil are captured after (a) 0.0 s (0°), (b) 0.125 s (8.6°), (c) 0.25 s (12.2°), (d) 0.375 s (8.6°), (e) 0.5 s (0°), (f) 0.625 s (−8.6°), (g) 0.75 s (−12.2°), (h) 0.875 s (−8.6°) and (i) 1.0 s (0°).

10.3c. The trailing edge moves downwards, and the angle of attack changes from originally 0.0° to 12.2°. The fluid particles that move on the top surface of the airfoil encounter a change in pressure. First, from high pressure in front of the airfoil to low pressure over the surface of the airfoil. Lastly, back to the high pressure behind the airfoil profile. The fluid moves from low to high pressure (adverse pressure gradient), resulting in flow

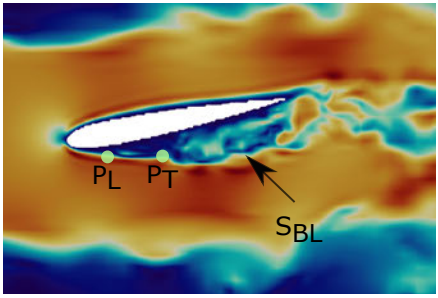


Figure 10.4.: Zoom-in of Figure 10.3h ($t = 0.75\ s$) at the boundary layer region, close to the airfoil surface. The point P_L depicts the region up to the laminar boundary layer. The area between P_L and P_T presents the approximate area, where the transition from laminar to turbulent boundary layer occurs and the point where separation of the boundary layer occurs (P_T). S_{BL} indicates the area where flow separation has happened.

separation. In this case, the pressure gradients become large enough to overcome the fluid inertial forces, and the flow separates from the airfoil surface. As the pressure gradient increases with the increasing angle of attack, flow separation occurs earlier for higher angles of attack of the airfoil. This behavior is noticeable in Figure 10.3, where the largest angle of attack (12.2°) causes flow separation, which happens much earlier than for an angle of 0.0° or 8.6°. The early flow separation can also be observed in Figure 10.3g.

In Figure 10.4 a close-up of the velocity field close to the airfoil is illustrated. Different areas are highlighted by points, indicating the change of the boundary layer behavior. P_L indicates the end of the laminar boundary layer and the transition to the turbulent boundary layer. P_T marks the separation of the turbulent boundary layer (S_{BL}). In this area, the fluid flow is reversed, resulting in a recirculation region.

This can be properly observed through streamlines, shown in Figure 10.5. Figure 10.5a and Figure 10.5c. They illustrate the streamlines for the entire domain when the airfoil has reached its final position in downwards and upwards direction, respectively. Figure 10.5b and Figure 10.5d present

a close-up of the upper and lower surface of the airfoil. In Figure 10.5c at the jet-inflow vortices can be observed. They are close to the boundary of the jet-inflow, where the flow is reduced in motion due to the surrounding fluid at rest and the domain boundary. Furthermore, the previously mentioned boundary layer separation and the reverse flow behavior can be identified at the lower surface of the airfoil, in the respective zoom-in in Figure 10.5d. The streamlines close to the geometrical surface are not attached anymore. Comparing Figure 10.5b and Figure 10.5d, both images provide very similar flow fields, with the flow pattern being turbulent. Please note that polynomial series represent simulation results, and for visualization, they are voxelized during post-processing. Thus the airfoil geometry appears rough. However, this does not influence the actual solution and is only a post-processing artifact.

A more detailed examination of the pressure is provided in Figure 10.6. The pressure field is shown for the same point in time as in Figure 10.3. We can observe high pressure values at the airfoil nose, where the flow velocity is close to 0, and the pressure has reached a high value (cf. Figure 10.6b, Figure 10.6f and Figure 10.6i). Furthermore, high pressure values are present at the trailing edge of the airfoil, which are among others responsible for the disturbance of the fluid flow and the generation of noise (cf. Figure 10.6g or Figure 10.6d). This behavior is well known from various investigations, numerical simulations, and experimental studies. The high pressure values arise around the moving airfoil, as the fluid is compressed in the case of the downwards motion, close to the airfoil's lower surface (pressure surface), and the case of an upwards motion at the upper surface of the geometry. Consequently, at the suction surface, the pressure is much lower compared to the surrounding flow.

We continue our examination by tracing the change in lift and drag around the geometrical surface. Measurement points (200) are equidistantly distributed around the airfoil to capture the change in pressure. The profile is tracked at the center in the z-direction ($z = 0.0$) during post-processing. Pressure values are used to compute the lift coefficient C_L and the drag coefficient C_D, respectively.

$$F_L = C_L \cdot \frac{\rho_B}{2} \cdot {v_{FS}}^2 \cdot A_s \tag{10.3a}$$

$$F_D = C_D \cdot \frac{\rho_B}{2} * \cdot {v_{FS}}^2 \cdot A_s \tag{10.3b}$$

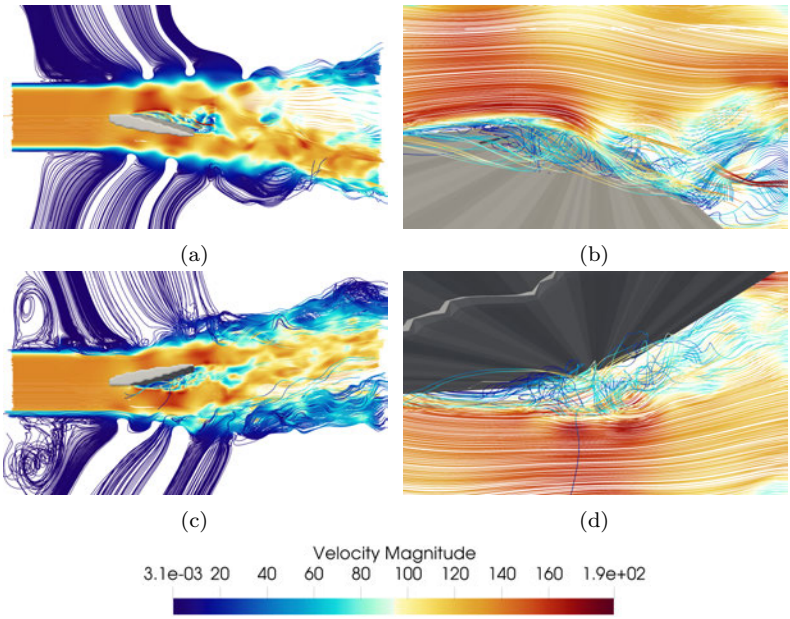


Figure 10.5.: Streamlines colored by the velocity magnitude at a $t = 0.25\ s$ and (c) $t = 0.75\ s$, when the airfoil geometry has an angle of attack of $12.2°$ and $-12.2°$, respectively. Zoom-ins of the boundary layer area are depicted in (b) and (d) for the mentioned simulation times.

Eq.(10.3) [51] provides the necessary equations for the computation of the respective coefficients. Here F_L and F_D denote the lift and drag forces, v_{FS} the free stream velocity, ρ_B the background density and A_s the reference surface of the airfoil.

In Figure 10.7a the pressure coefficient C_P is presented for the angles of attack of $0.0°$ and $12.2°$. As expected at the nose of the airfoil (stagnation point), both curves have a pressure value of around 1.0. In the case of an angle of attack of $0.0°$ starting from $x/l = 0.2$ the C_P value fluctuates around an approximate value of 0.0. We need to emphasize that the airfoil moves throughout the simulation time. Thus, the flow field is perturbed all time, and the dynamic of the motion influences the flow field further.

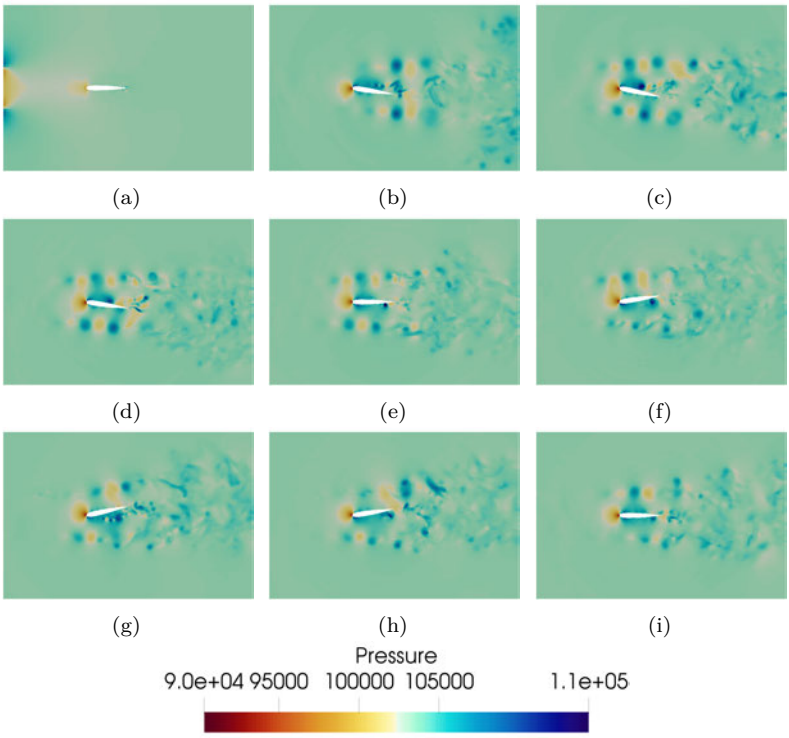
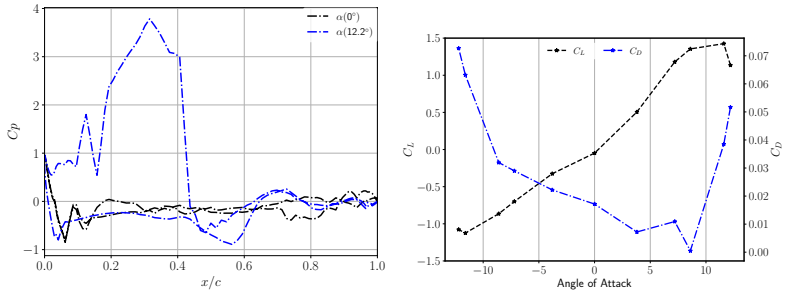


Figure 10.6.: Time series of the innermost subdomain for the variable pressure is shown in negative z-normal. The pressure is captured after (a) 0.0 s, (b) 0.125 s, (c) 0.25 s, (d) 0.375 s, (e) 0.5 s, (f) 0.625 s, (g) 0.75 s, (h) 0.875 s and (i) 1.0 s.

Comparing the curve representing an angle of attack of 0.0° and 12.2°, we recognize some differences, especially in the first half of the airfoil geometry. In the case of an angle of attack of 12.2° up to approximately $x/l = 0.42$, a high pressure coefficient can be identified. Due to the high pressure gradients in that area, the pressure coefficient attains a value of 3.8. The boundary layer separation occurs in this area, as previously shown, and the lift coefficient and the drag coefficient react oppositional (cf. Figure 10.7b).



(a) (b)

Figure 10.7.: In (a), the pressure coefficient C_p is presented for an angle of attack of 0.0° and 12.2°. In (b), the angle of attack is shown against the global C_L and global C_D. 200 measurement points are equidistantly positioned around the airfoil at coordinate $z = 0.0$.

In Figure 10.7b both C_L and C_D are plotted against the angle of attack. Since the airfoil moves, we can capture different angles of attack throughout the simulation time. For our investigation shown in Figure 10.7b, we consider the upwards motion of the airfoil. Thus, when the airfoil moves from the maximum position at an angle of 12.2° up to an angle of −12.2°. The C_D values are represented by the blue curve and the C_L values by the black line. With increasing angle of attack (from 0° up to 12.2°), the lift coefficient increases simultaneously. Up to an angle of 11.2°, this inclination can be observed. However, for an angle of attack of 12.2° the lift coefficient decreases. The decrease of the lift coefficient at 12.2° is in agreement with the previously mentioned flow separation shown in Figure 10.4 and Figure 10.5. At the same time, the C_D value increases, attaining a higher value for this angle of attack. From this point on, no lift is generated anymore, and an even higher angle of attack will not correspond to a higher lift coefficient. The same behavior can be observed for negative angles of attack, where the lift coefficient decreases with increasing angles of attack as the airfoil moves upwards. Here again, at an angle of −12.2° the curve has an oppositional behavior compared to, e.g., an angle of attack of −11.2°. It is again as shown in Figure 10.5 due to the flow separation, where the lift is no longer generated. The C_D has a high value for this angle of attack as supposed. One might wonder

why the C_L value does not attain a value of 0.0 at an angle of attack of $0°$, which is a little bit off compared to the expected value. We need to emphasize that the tiny offset is due to the motion of the geometry that continuously moves throughout the simulation time.

From previous studies we know, that flow separation occurs between an angle of attack of $10.0°$ and $15.0°$ depending on the Reynolds number. Moreover, the solutions for negative and positive angles of attack are evident that the solution obtained correlates with the expected flow behavior, which is in agreement with observations by [1, 21, 61, 90], who conducted investigations for Reynolds numbers of up to $3 \cdot 10^6$. However, we need to keep in mind that the experimental data and simulations conducted in those mentioned publications were always for a non-moving airfoil with a predefined angle of attack. Thus the dynamic of the motion of the airfoil was not considered in their investigation.

All figures presented in this section demonstrated that the simulation results agree with physical expectations known from the literature. In the following, the middle subdomain is attached to the innermost subdomain for further analysis.

10.2. Innermost and middle subdomain: Compressible Navier-Stokes and Euler

In this section, the transition of the fluid flow from the innermost subdomain to the middle subdomain is of interest. In Figure 10.8 both domains are attached at their respective coupling interfaces, and the velocity field is shown. The velocity is captured for the simulation times of 0.125 s, 0.25 s, 0.375 s, 0.5 s, 0.625 s, 0.75 s, 0.875 s and 1.0 s. In none of the shown snapshots, a discontinuity at the coupling interfaces is visible. The transition occurs smoothly from the innermost to the middle subdomain. Additionally, changes in the flow pattern are captured precisely by the middle domain. Due to the varying angle of attack of the airfoil, the flow field is correspondingly affected. For example, in Figure 10.8a and Figure 10.8b the trailing edge of the airfoil moves downwards until finally attaining an angle of attack of $12.2°$. Thus, the expectation is that the jet is also directed in this direction. This foreseen behavior is observable as the flow evolves and moves further along the length of the middle domain.

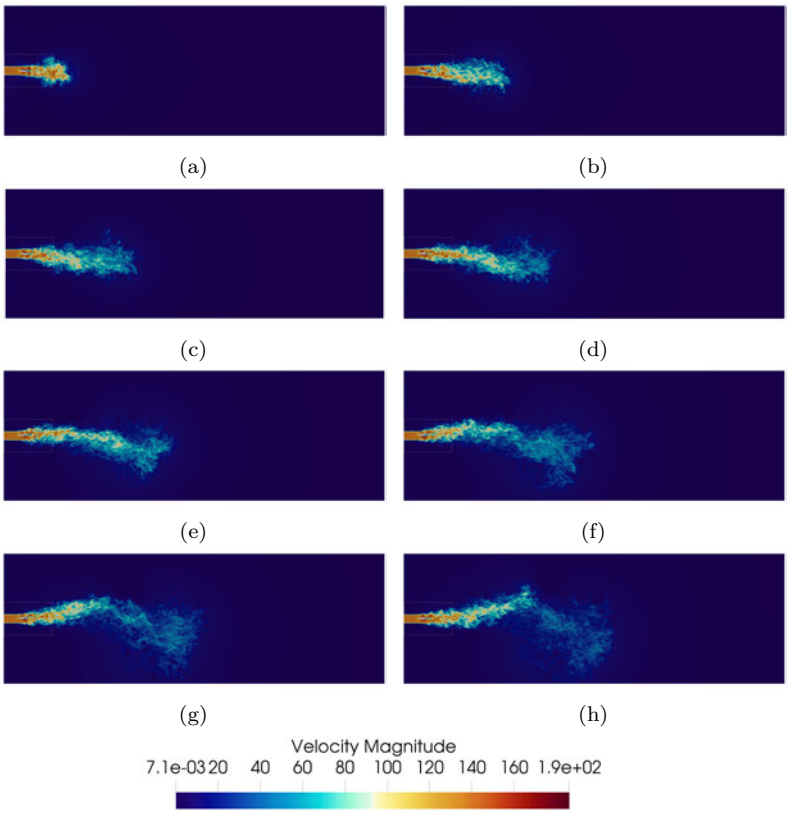


Figure 10.8.: Time series of the innermost and middle subdomain for the quantity velocity is shown in negative z-normal. The velocity is shown after (a) 0.125 s, (b) 0.25 s, (c) 0.375 s, (d) 0.5 s, (e) 0.625 s, (f) 0.75 s, (g) 0.875 s and (h) 1.0 s.

Figure 10.8c and Figure 10.8d illustrate the velocity field when the airfoil moves to its initial location. The flow field in the vicinity of the geometry changes according to the motion of the airfoil. Large eddies can be recognized, which travel from the innermost to the middle subdomain. Vortices directed downwards due to the downwards moving airfoil, e.g., at $t = 0.25$ s, continue their path in the respective direction. In Figure 10.8e

and Figure 10.8f the angle of attack has changed in upwards direction. Thus the jet is guided accordingly, resulting in upwards moving vortices close to the airfoil. Vortices generated due to the initial motion of the airfoil have now traveled a specific distance (cf. Figure 10.8g downwards traveling vortices). In addition, they have formed a vortex cluster that moves further in a downwards direction, towards the lower coupling interface of the middle subdomain (cf. Figure 10.8h). Since the inviscid Euler equations are solved in the middle subdomain and a higher scheme order is used, vortex structures are transported and maintained due to low dissipation of the scheme.

Figure 10.9 presents the pressure field for the innermost and middle subdomain for the same simulation times as in Figure 10.8. Close to the geometry, the highest pressure values are encountered, as previously mentioned in section 10.1. Additionally, pressure waves can be observed that travel from the innermost subdomain to the middle domain. Further a large vortex cluster can be identified, that moves towards the outer boundaries of the middle domain (cf. Figure 10.9f, Figure 10.9g and Figure 10.9h). Besides the left boundary, all other subdomain boundaries are involved in the coupling with the outermost subdomain, where only the linearized Euler equations are solved. Hence the expectation might be to encounter stability problems at the coupling interface due to simplifying the fluid dynamic equations in that subdomain. Moreover, due to the vortex cluster, significant changes in the flow field at the coupling interface might be expected as vortices approach the lower boundary interface. A more detailed investigation of the outermost subdomain will be presented in the next section. The transition from the middle to the outermost subdomain will be further discussed in Section 10.4.

In Figure 10.10 and Figure 10.11 the Q-criterion for the simulation times of $t = 0.625\ s$ and $t = 1.0\ s$ are depicted. A clip in normal z-direction is used to provide a more detailed insight into the flow field (cf. Figure 10.10b and Figure 10.11b). An additional zoom-in at the coupling interface presents a comprehensive view of the area, where the flow leaves the innermost subdomain and enters the middle domain (cf. Figure 10.10c and Figure 10.11c). In Figure 10.10a and Figure 10.11a vortex structures of different sizes can be found. Moreover, the expansion of the fluid flow along the width of the domain can be observed. Vortices that are away from the airfoil have, as expected, a lower velocity, while along the jet stream, they have a higher magnitude (cf. Figure 10.10b and Figure 10.11b). From Figure 10.11a and Figure 10.10a, no noticeable difference in the solution

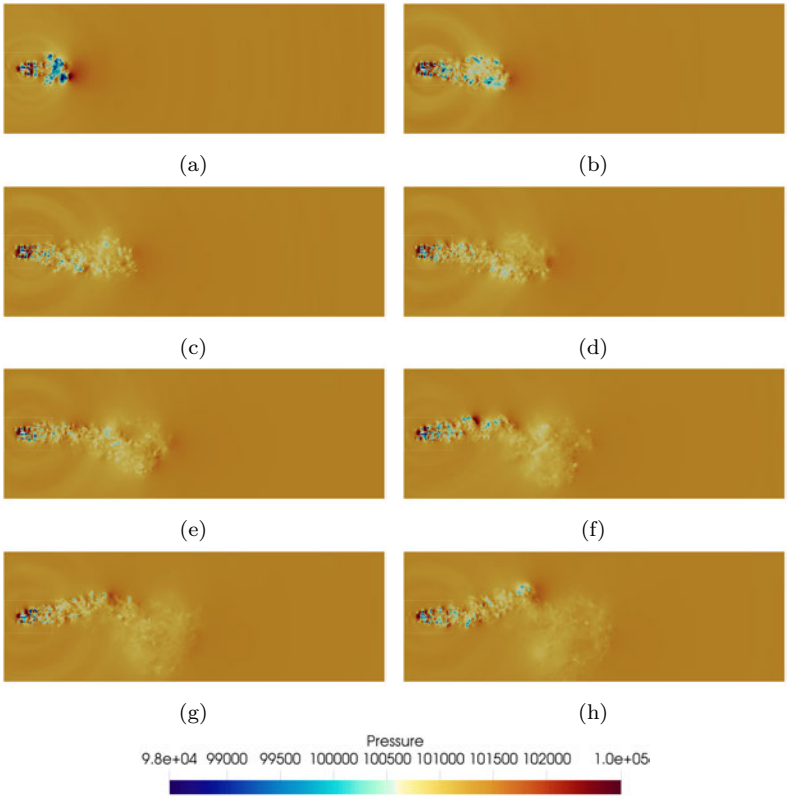


Figure 10.9.: Time series of the innermost and middle subdomain for the variable pressure is shown in negative z-normal. The pressure field is shown after (a) 0.125 s, (b) 0.25 s, (c) 0.375 s, (d) 0.5 s, (e) 0.625 s, (f) 0.75 s, (g) 0.875 s and (h) 1.0 s.

is visible, indicating a smooth transition from the innermost subdomain to the middle domain, where a different configuration for the simulation domain is used. A distinction between the subdomains is hardly visible.

Considering the zoom-in into the area, where both domains are coupled together, and information transferred from one side to the other and

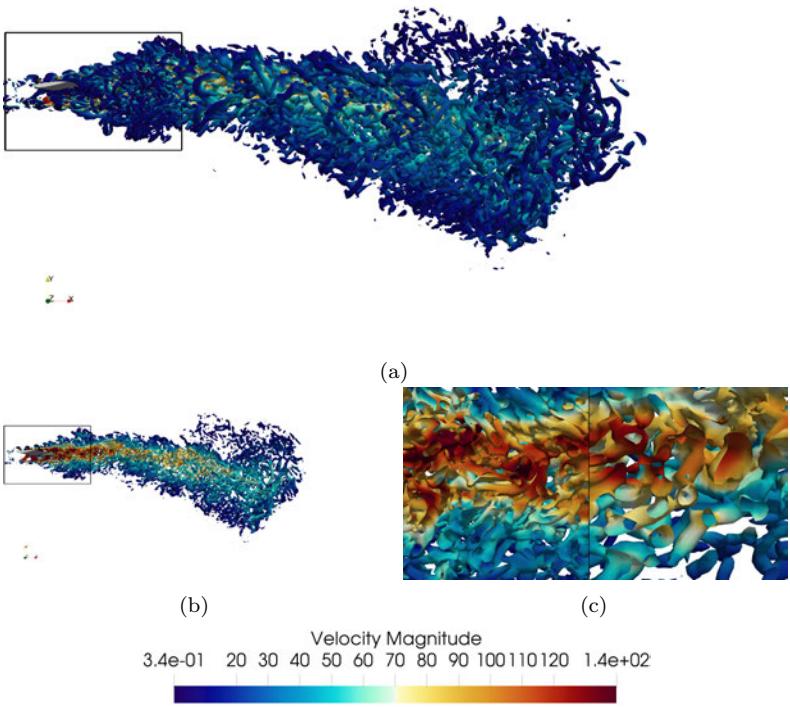


Figure 10.10.: Q-criterion of 4000 colored by the velocity magnitude for the simulation time $t = 0.625\ s$. The angle of attack of the airfoil is $-8.6°$. In (a) the Q-criterion is depicted for both subdomains and in (b) a vertical clip along the z-normal is used to illustrate further the transition from the innermost subdomain to the middle subdomain. A black frame indicates the innermost subdomain. In (c), a zoom-in of (b) at the coupling interface is shown. The coupling interface is marked through a vertical line.

vice versa, both appear very similar, even though a different scheme order and mesh resolution, as well as equations, are applied. However, a closer consideration of Figure 10.11c and Figure 10.10c portrays slight differences

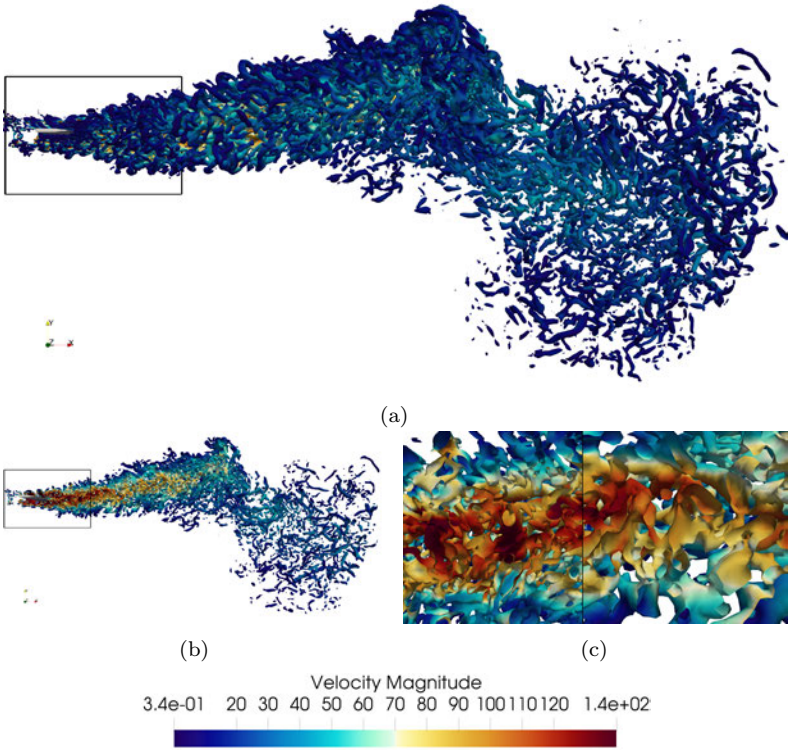


Figure 10.11.: Q-criterion of 4000 colored by the velocity magnitude for the simulation time $t = 1.0\ s$. The angle of attack of the airfoil is $0°$. In (a), the Q-criterion is depicted for both subdomains. In (b), a vertical clip along the z-normal further illustrates the transition from the innermost subdomain and the middle subdomain. A black frame indicates the innermost subdomain, and in (c), a zoom-in of (b) at the coupling interface is shown. The coupling interface is marked through a vertical line.

at the coupling interface. In the innermost subdomain, smaller vortices appear due to the physical viscosity and the resulting energy cascade, where large eddies break down into smaller ones until dissipation occurs

and kinetic energy is transferred to heat. Those small vortices reach the coupling interface but are not present in the middle subdomain. This loss is due to the lower resolution in the middle domain, which cannot capture the more minor scales (vortices). However, those vortices are not of interest, as only large eddies with a higher energy level have a remarkable impact on the flow field and the evolution of the acoustics far-field. Therefore, those scales are resolved and further transported. Resolving those smaller scales would only lead to their transportation to the middle domain, as physical viscosity is neglected. Moreover, they would require a higher computational effort as a higher spatial resolution would be necessary. Hence choosing the resolution according to the larger scales allows for more efficient computation and the possibility to resolve scales that are relevant for the problem to be solved.

Other than that, no noticeable differences between the innermost subdomain and the middle domain can be observed. Thus no indication of any wrong physical representation of any phenomena can be identified. Remarkable is that only a close-up, hence a high zoom-in level, was able to highlight this neglectable difference at the coupling interface.

10.3. Innermost, middle and outermost subdomain: Navier-Stokes, Euler and Linearized Euler

We now continue and attach the last subdomain for further examinations of the flow field. The outermost subdomain is the largest domain of all three subdomains. However, it has fewer elements when compared to the other two subdomains. Here, we are only interested in the transport of acoustic waves. Due to the underlying high-order Discontinuous Galerkin method, small perturbations can be transported over a large area with low dissipation and dispersion error. On the other hand, a low-order scheme would require a much finer mesh in order to be able to transport that information up to the far-field as low-order schemes are very dissipative and would smear the solution considerably. A scheme order of $O(9)$ is used in this subdomain, while the finest elements have a uniform element length of merely 1 m.

Figure 10.12 presents the velocity perturbation in the simulation domain over the time. The velocity field is captured form $t = 0.125\ s$ up to $t = 1.0\ s$ using $0.125\ s$ intervals. The velocity scaling is adjusted to the outermost subdomain to capture small perturbations, which are very tiny compared

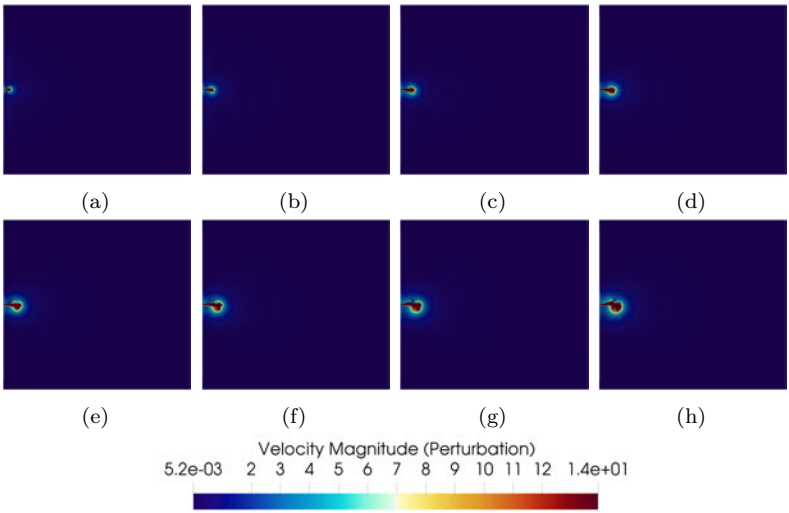


Figure 10.12.: The perturbation of the velocity is shown for different simulation times. The velocity magnitude in m/s is captured after (a) 0.125 s, (b) 0.25 s, (c) 0.375 s, (d) 0.5 s, (e) 0.625 s, (f) 0.75 s, (g) 0.875 s and (h) 1.0 s.

to the innermost or middle subdomain. In Figure 10.12a, Figure 10.12b, Figure 10.12c and Figure 10.12d the evolution of the velocity field over the time is shown. Vortices are visible, which travel from the innermost subdomain to the middle domain, as shown in the previous section. Due to the motion of the airfoil geometry, the flow is accordingly directed downwards and upwards. Therefore in Figure 10.12e, Figure 10.12f, Figure 10.12g and Figure 10.12h a high value for the velocity perturbation can be observed close to the lower boundary of the middle domain, in the vicinity of the respective outermost subdomain (cf. e.g., Figure 10.8h). From the previous section we know, that in Figure 10.12h the vortex cluster has reached the coupling interface, noticeable through the high velocity value in that region.

As it is hardly visible to distinguish the different subdomains and the occurring physics close to the outermost subdomain, a close-up of the coupling interface is presented in Figure 10.13. The flow field is shown for different simulation times to demonstrate how the flow features appear

close to the coupling interface. From Figure 10.13a to Figure 10.13d, it is apparent, how as mentioned before, the vortex cluster moves towards the coupling interface. However, the transition from the middle to the outer subdomain is smooth, and no discontinuity can be recognized in the outermost subdomain. A discontinuity at the coupling interface would indicate that the solution is not approximated accordingly. Hence a false physical representation for the solution is used. However, this is not the case at the coupling interface; we can assume that the solution has not yet advanced, and this information has not propagated into the outermost subdomain. Furthermore, in this case, nonlinear effects would have been further transported to the outermost subdomain. As a result, we might encounter stability issues, as the linearized equations cannot capture the dynamics of these nonlinear phenomena. Eventually, they will approximate the solution incorrectly and deviate from the physically correct solution. Further, we need to emphasize that linearization is only allowed when the perturbation is tiny. This requirement is satisfied for our simulations as the information has not propagated into the outermost subdomain. Therefore the obtained results are still meaningful and physically correct. However, suppose the simulation would be restarted for a longer simulation time. In that case, the middle subdomain necessitates being extended further in the y-direction to avoid vortices in the vicinity of the coupling interface to the outermost subdomain.

We need to emphasize that information at the coupling interface is always exchanged in both directions, e.g., from the middle to the outermost domain and vice versa. Thus, the propagation of vortices in the outermost subdomain and its false representation can result in physically wrong values exchanged at the coupling interface from the outermost subdomain to the middle domain.

In Figure 10.14 the pressure perturbation for different points in time is illustrated. From Figure 10.14a up to Figure 10.14h the pressure perturbation in all three subdomains for the simulation times of $t = 0.125\ s$ up to $t = 1.0$ in an interval of $0.125\ s$ is illustrated. In the far-field (outermost subdomain) a homogenous flow field is encountered. Furthermore, cylindrical waves can be observed that travel towards the outer boundaries generated by the jet-inflow and the presence of the airfoil. Both jet-inflow and airfoil disturb the flow field and cause, among others, those waves. Close to the boundaries, the waves are smeared due to the larger mesh elements. Further, the sponge layers actively damp the pressure perturbation close to the outer boundaries to avoid reflections by the outflows. Figure

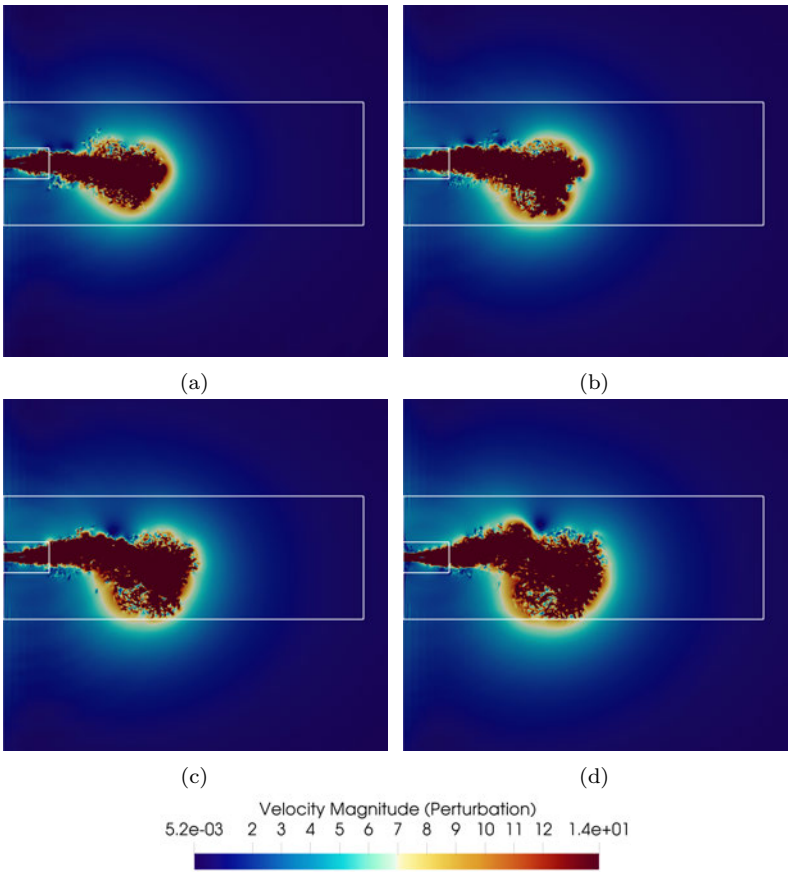


Figure 10.13.: Zoom-in into the region close to the coupling interface between the middle and the outermost subdomain. The perturbation of the velocity magnitude is presented after(a) 0.625 s, (b) 0.75 s, (c) 0.875 s and (d) 1.0 s. White frames highlight the coupling interfaces.

10.14c up to Figure 10.14h demonstrate that sponge layers act as expected since no reflection of the outgoing pressure waves can be observed. Finally, in Figure 10.15 a zoom-in close to the coupling interface is shown, where

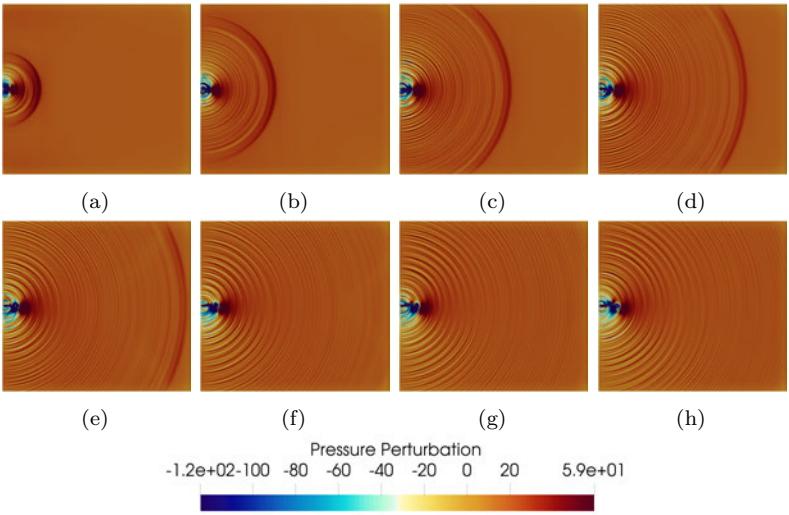


Figure 10.14.: The transport of pressure waves is shown for different simulation times. Results are presented after (a) 0.125 s, (b) 0.25 s, (c) 0.375 s, (d) 0.5 s, (e) 0.625 s, (f) 0.75 s, (g) 0.875 s and (h) 1.0 s.

pressure waves are transported from the middle domain to the outermost subdomain. The transition of the waves from one domain to the other is smooth and precise. Moreover, in both shown simulation times in Figure 10.15a and Figure 10.15b the pressure waves are transported as foreseen.

From Figure 10.14 we know that the pressure perturbation attains a value of up to roughly 60 Pa. Moreover, we know that linearization is only allowed if the perturbation is sufficiently small compared to the atmospheric pressure. The pressure perturbation is approximately 0.06 % of the background pressure (101325 Pa) for our test case. Therefore the change of the fluid dynamic equations to the linearized Euler equations in this subdomain is from the physical perspective allowed and valid.

For further analysis and to give an overview of how strong those pressure waves are, thus how high the sound pressure level (SPL) might be, we continue our investigation by utilizing sound pressure level plots. Thus the

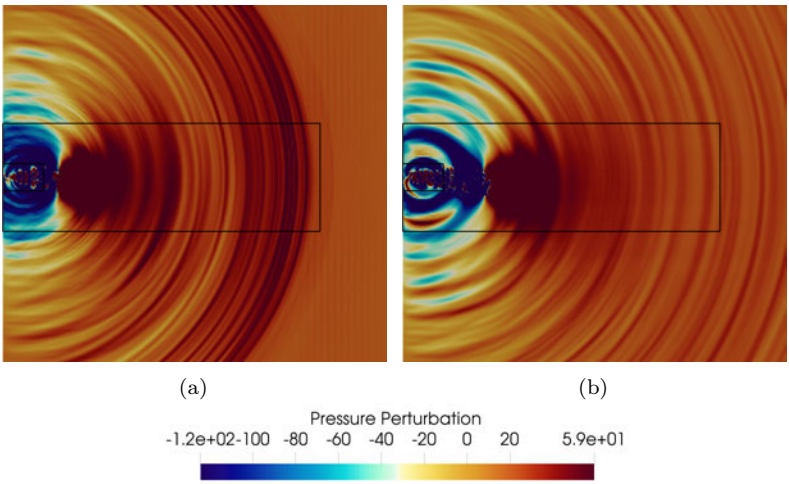


Figure 10.15.: Zoom-in into the region, where pressure waves are transported from the innermost subdomain to the middle and finally to the outermost subdomain. The pressure perturbation is presented after (a) 0.125 s and (b) 0.25 s. Black lines highlight the coupling interface.

sound level is provided in decibel (dB) against the frequency spectra in Hertz (Hz). Four microphones are positioned in the outermost subdomain, capturing the pressure perturbation in each iteration, thus resulting in over 400.000 data values for the spectral analysis for each microphone. The microphones are located at $M_1(65.0, 60.0, 0.0)$, $M_2(65.0, 60.0, 0.0)$, $M_3(180.0, 30.0, 0.0)$ and $M_4(180.0, -30.0, 0.0)$. Thus the distance from the microphones to the sound source (airfoil leading edge) is 63.0 m and 178.0 m in x-direction, respectively. The microphones are located away from the absorbing sponges to avoid any influences by them. The time signal and the frequency spectra are illustrated in Figure 10.16, where the sound pressure level is plotted over the frequency and, in the case of the time signal, the pressure perturbation over time. For the computation of the sound pressure level Eq. (10.4) is used.

$$SPL = 20 \cdot \log_{10}\left(\frac{p_{RMS}}{p_{ref}}\right) \tag{10.4}$$

Where p_{RMS} is the root mean square of the pressure and p_{ref} is the reference pressure, which has a value of $2 \cdot 10^{-5}$ Pa with a sound frequency of 1 kHz. This is the lowest hearing threshold of a young and healthy ear [27].

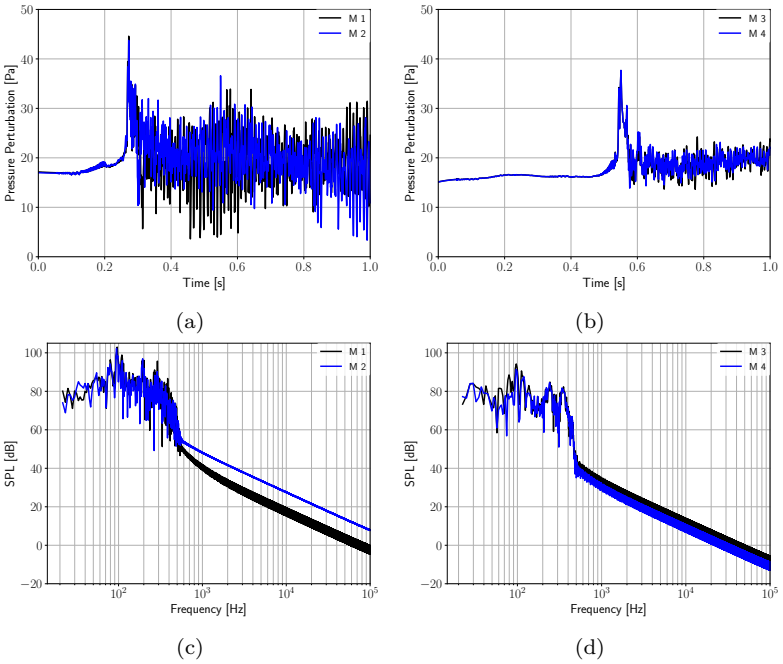


Figure 10.16.: Measurement results of four microphones (M_1, M_2, M_3 and M_4) positioned at different locations in the outermost subdomain. In (a), the pressure perturbation over time for M_1, and M_2 and in (c), the corresponding frequency range and the pressure level are shown. In (b) the pressure perturbation over time for M_3 and M_4 and (d) the respective frequency range and the pressure level are depicted.

From the time signal in Figure 10.16a and Figure 10.16b the recorded signal is first relatively constant, that is due to pressure waves that have not yet reached the respective microphones. Since the pressure waves move with the speed of sound, it requires some time until they finally reach the

respective microphones. However, after some time, a very dominant peak and a fluctuating behavior of the pressure perturbation can be observed (pressure waves). The strong peak represents the abrupt motion of the airfoil and its representation with higher order after the startup phase. When comparing Figure 10.16a and Figure 10.16b we notice that the first strong pressure wave, indicated by the highest peak in the spectra, arrives at M_1 and M_2 earlier, as the positioning is closer to the airfoil, compared to M_3 and M_4. On the other hand, M_3 and M_4 (cf. Figure 10.16b) record the strong wave at a later point in time, as they are even further away from the geometry and the jet-inflow.

For the Fourier Transformation (FT), we neglect those areas and only include for M_1 and M_2 the time from 0.4 s to 1.0 s and in the case of M_3 and M_4 the simulation time of 0.6 s to 1.0 s. Consequently, the influence of the first pressure wave is removed. The respective FT spectra are shown for the different microphones in Figure 10.16c and Figure 10.16d. The recorded sound, thus the sound pressure level for the first two microphones, is, as expected, higher. Here, a pressure level of up to 100 dB can be captured. The measuring points M_3 and M_4 can record the respective signal with up to 95 dB. However, starting from a frequency of approximately 500 Hz (cut-off frequency) the sound pressure level decreases drastically as the numerical resolution is too low to provide more information on those higher frequencies. We need to emphasize that the slight deviation of M_1 and M_2, and M_3 and M_4 is due to the motion of the airfoil. It influences the flow field according to the direction of its motion.

For the microphones M_1 and M_2 the approximated mean sound pressure level is 80 dB, for the microphones M_3 and M_4 the mean value is roughly 70 dB (cf. Figure 10.16c and Figure 10.16d). Comparing those sound levels to daily occurrences, we can associate 70 dB and 80 dB to typical traffic jams and a standard vacuum cleaner. Humans' hearing system can recognize the sound from 20 Hz up to 20.000 Hz. According to the World Health Organization, with a frequency of 500 Hz and a sound pressure level between 41 dB and 60 dB, the noise can cause moderate impairment. A sound level between 61 dB and 80 dB in this frequency range can even result in severe impairment [96]. In both frequency spectra shown, the mean audible range is above 60 dB. Considering the inverse square law, we know that a reduced sound pressure level of 6 dB is obtained by doubling the distance to the noise source. Assuming the simulation domain is larger than shown in this work, roughly 1.5 km, the sound pressure level measured at that distance is approximately 42 dB, thus still

in the range that can impact a human's hearing system.

In recent years research has been conducted to reduce noise pollution further by deploying shape optimizations in the design process and special measures, e.g., serration at the trailing edge of airfoils. Even though no further improvements on the airfoil design have been incorporated in this work, the shown results provide first numerical solutions and demonstrate that the utilized techniques, namely the embedded method and the coupling strategy, enable such complex and large-scale simulations efficiently.

In the following the coupling interfaces of the different subdomains are investigated in more detail.

10.4. Coupling interfaces of the 3-field coupled simulation

The previously discussed examination of the 3-field coupled simulation is continued in this section, where mainly the coupling interfaces are investigated. Moreover, the decomposition practice and the change of the fluid dynamic equations in each subdomain are further studied.

We first begin with the innermost subdomain, where the compressible Navier-Stokes equations are solved. Additionally, an airfoil is located in this subdomain. It requires, in any case, the consideration of physical viscosity. Therefore, there is no question on the equations to be solved in this subdomain. Then, however, the question arises, where a simplification is allowed, thus where the physical viscosity in the fluid dynamic equations can be neglected. The simplification is permitted when the viscous fluxes are reduced considerably compared to the area where it is mandatory, thus close to the airfoil, as shear stresses at the geometrical boundary are present and have to be physically captured. Therefore we focus on the second derivative of the velocity (Δv), which has to be small enough, indicating that physical viscosity can be neglected and simpler fluid dynamic equations can be solved. Thus, we first visualize the innermost subdomain, where the second derivative of the velocity is of importance. In Figure 10.17 two different views of the innermost domain are illustrated, one in z-normal (Figure 10.17b) and one that allows a border view (Figure 10.17a), allowing to also keep track of the coupling interface to the middle domain. From Figure 10.17a it is apparent that the second derivative of the velocity is the highest close to the airfoil. Away from the geometry, the quantity loses in magnitude. Close to the coupling interface (cf. Figure 10.17b right

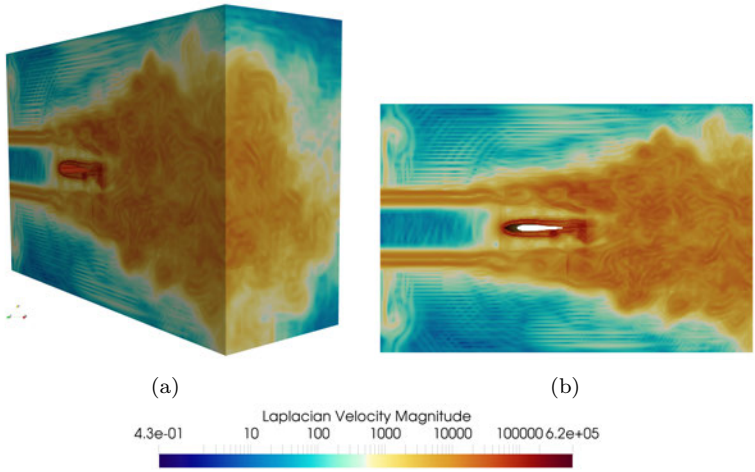


Figure 10.17.: Velocity magnitude of the 2^{nd} derivative of the velocity, an indicator for the simplification of the fluid dynamic equations.

end of the subdomain) the magnitude is already reduced by a factor of over 50. The flow behind the airfoil has reached a comparable velocity magnitude similar to the flow above the airfoil. Further, in the figure, we can observe oscillations (similar to waves) due to the state's polynomial representation. However, those oscillations do not affect occurring phenomena and can be removed through filtering in the post-processing procedure. Moreover, they are only visible, as the 2^{nd} derivative of the velocity is shown, which presents small details, only noticeable in a logarithmic scale. In the previous sections, we already demonstrated that the transition of the fluid flow from the innermost subdomain to the middle subdomain happens very precisely, without any loss of information, which is essential for the evolution of the far-field.

The innermost and the middle subdomain are connected through three coupling interfaces: One at a y-z plane (right side) and two at a x-z plane (up and down). At those boundaries, information is exchanged. However, as explained in the previous sections, the coupling interface at the right end of the innermost subdomain is of interest since most vortices have to cross this coupling interface to reach the middle domain. Furthermore,

the lower coupling interface connecting the middle subdomain and the outermost domain is crucial as vortices move from the middle domain towards this interface. Therefore those two interfaces are further examined for the quantities: Velocity, pressure, and density. In Figure 10.18 the

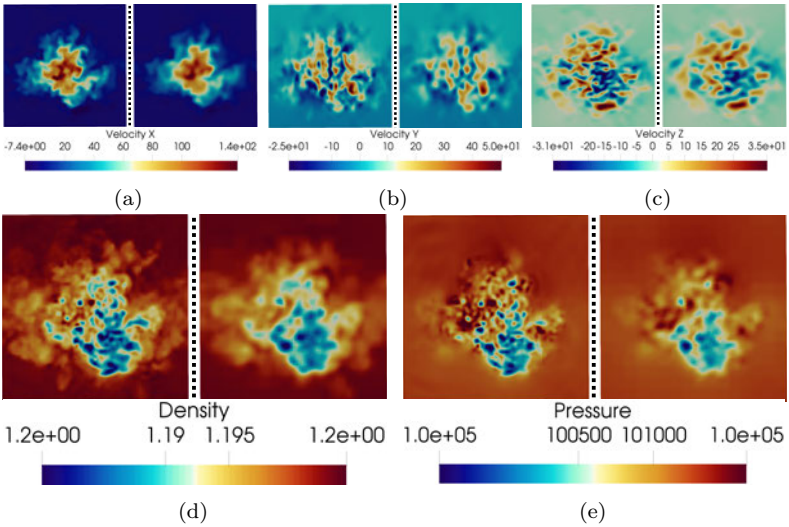


Figure 10.18.: Coupling interface between the innermost and middle subdomain at the right boundary of the innermost subdomain. In (a) the velocity in x-direction, in (b) the velocity in y-direction, in (c) the velocity in z-direction, in (d) the density, and (e) the pressure is shown. In each subfigure on the left, the coupling interface of the innermost subdomain and on the right the corresponding coupling interface of the middle subdomain is depicted. Black dashed lines separate the domains for visualization purposes. Slices are captured for $t = 1.0\ s$.

coupling interface between the innermost and the middle subdomain; thus, the right end of the innermost subdomain is shown. On the left side of the subfigures, the innermost interface is presented. On the right, the interface of the middle subdomain is depicted. A black dashed line between both domains separates the images for comparison. Figure 10.18a, Figure 10.18b and Figure 10.18c show the velocity components in all three spatial

directions. In all figures, the velocity pattern of the innermost subdomain resembles the corresponding interface at the middle domain. As previously mentioned, the resolution in the middle domain is slightly reduced than for the innermost subdomain as only large scales are considered for the evolution of the far-field. Therefore only large-scale phenomena cross the interface, while smaller ones can not be captured. In the case of the density and pressure, illustrated in Figure 10.19d and Figure 10.18e respectively, we can observe the same behavior as for the velocity. Here smaller scales are smeared, while larger ones are adequately captured.

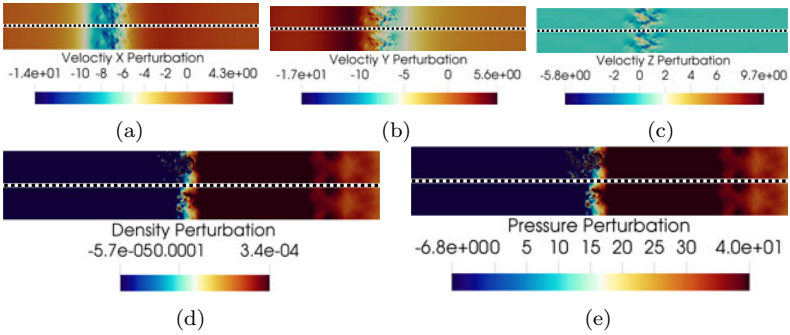


Figure 10.19.: Coupling interface between the middle and outermost subdomain at the lower boundary of the middle subdomain. The velocity perturbation is shown for (a) the velocity in x-direction, (b) the velocity in y-direction, (c) velocity in z-direction, (d) density, and (e) pressure. In each subfigure on the upper side, the coupling interface of the middle subdomain, and on the lower side, the corresponding coupling interface of the outermost subdomain is depicted. Black dashed lines separate both images. Slices are captured for $t = 1.0\ s$.

In Figure 10.19 the coupling interface at the middle and outermost subdomain is depicted for the velocity, density and pressure perturbation. As vortices move towards the lower coupling boundary of the middle domain, this interface is of interest. Only the linearized equations are solved in the outermost subdomain, and therefore, the dynamic of nonlinear phenomena is neglected. Though, the expectation might be, as nonlinear information

travels from the middle subdomain to the outermost subdomain towards the end of the simulation time, that those nonlinear phenomena cause **(i)** discontinuity at the coupling interface and **(ii)** stability issues of the numerical simulation. However, none of those mentioned points have occurred for the shown quantities. Nonlinear flow features have only reached the coupling interface and have not yet advanced into the respective simulation domain. In Figure 10.19a, Figure 10.19b and Figure 10.19c the velocity perturbation in all spatial directions are presented. As vortices exist at the coupling interface, they are transported from the middle subdomain to the outermost subdomain (cf. Figure 10.19b), accordingly. However, as the outermost subdomain solves the linearized Euler equations, it cannot capture the dynamics of nonlinear effects. Nevertheless, we can also not detect any flaw in the solution at the interface. For density and pressure (cf. Figure 10.19d and Figure 10.19e), though, slight differences can be observed, which are not apparent for the velocity components. Those differences are due to the vortices that travel from the middle subdomain towards the outermost subdomain. However, due to the linearization, the occurring dynamic can not be similarly represented as in the middle domain.

10.5. Performance

We now investigate the scalability of the 3-field coupled problem on the Hawk supercomputing system. Measurements were conducted from 16 compute nodes up to 256 nodes, each equipped with 128 cores. In addition, intra- and inter-subdomain load balancing are deployed to distribute the workload among allocated processes equitable (cf. Section 8). In Figure 10.20 the strong scaling measurement is presented. The dash-dotted line highlights the ideal scaling, while the dashed line presents the actual compute time per iteration. The dashed line curve includes the compute time of each subdomain, the evaluation, the data exchange at the coupling points, and the waiting time between the subdomains to complete one iteration (synchronization). The computational resources are distributed according to the ratio of 92% : 3% : 5% for the innermost, the middle, and the outermost subdomain. The innermost subdomain requires most computational resources due to the costly compressible Navier-Stokes equations and the computational domain with the highest amount of mesh elements. Furthermore, the moving airfoil geometry is located there as well. The middle domain is when compared to the outermost subdomain, less compute-intensive. It has more elements than the outermost subdomain

and is involved in the surface coupling with the innermost and outermost subdomain. However, the outermost subdomain solves a higher scheme order and is required to solve additional source terms on the *rhs* of the fluid dynamic equations (sponge layer).

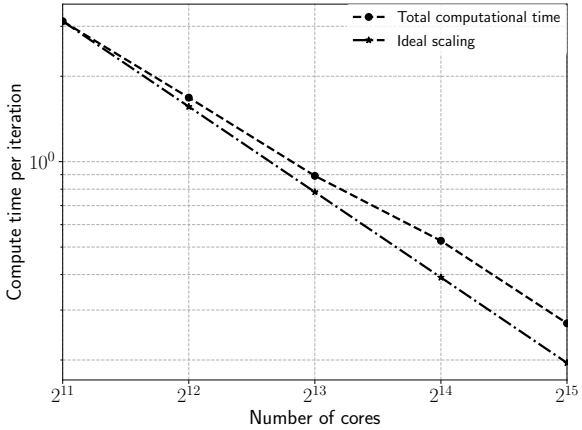


Figure 10.20.: Strong scaling measurement of the 3-field coupled simulation on Hawk supercomputing system, hosted by the High-Performance Computing Center in Stuttgart, Germany. Measurement includes up to 256 nodes, each equipped with 128 cores.

From Figure 10.20 we can observe how the total compute time (dashed line) scales with increasing core count. Even though it does not strictly follow the ideal scaling, the efficiency for the last data point with 256 nodes is approximately 74%. Furthermore, in section 6.7.1, we confirmed the scaling of the solver, where we present a perfect strong scaling behavior. The decisive difference between the measurement here and the measurement shown in Section 6.7.1 is the introduced coupling strategy. As already explained in Section 8.1 coupling elements have a higher workload, as they have to compute the fluid dynamic equations and are involved in the coupling procedure. Therefore, e.g., coupling elements at the domain corner are more compute-intensive than other coupling elements as they usually have three coupling interfaces.

Furthermore, coupling elements in the middle subdomain that provide boundary information to the innermost subdomain have a different workload than coupling elements between the middle and the outermost subdomain. Therefore, due to the higher workload and the fact that coupling elements are only present at the outer boundaries of each subdomain, they can limit the scalability of the coupled simulation. This limitation is due to the small number of those elements compared to those that only compute the fluid dynamic equations. With increasing cores, coupling elements are distributed among available compute resources, and at some point, a distribution is reached, where each coupling element is on one core. When reaching such a distribution, the scaling of the coupled simulation is eventually limited, and no further reduction in computational time can be achieved with increasing core count.

In our case, in the innermost subdomain, only $16,128$ elements are involved in the coupling compared to the total number of elements in the computational mesh (511,488 elements); those elements make less than 3.15% of the total number of mesh elements. On the other hand, in the middle domain, 960 elements are included in the coupling procedure to the innermost subdomain and $36,160$ elements to the outermost subdomain. However, the coupling elements to the innermost domain are more compute-intensive, as they need to additionally compute the gradients of the state variables for the innermost subdomain. Thus it is not surprising that the scaling is limited due to the compute-intensive elements.

Conclusion In this section, we presented a large-scale 3-field coupled simulation. The large simulation domain is decomposed into three subdomains, each configured with a dedicated configuration for the spatial discretization and solved with different fluid dynamic equations. Presented results are in agreement with expected physical phenomena. In the innermost subdomain, the compressible Navier-Stokes equations are solved. Moreover, an airfoil geometry is located inside this subdomain, which moves throughout the simulation time. The lift and drag coefficients are examined in detail, which can be captured for various angles of attacks due to the airfoil motion. With the changing angle of attack, the jet flow is also disturbed, resulting in vertex structures at the trailing edge of the airfoil. They travel from the innermost subdomain towards the middle subdomain, where they can be further captured accordingly, as the inviscid Euler equations are solved. Due to the jet-inflow and the airfoil structure, cylindrical pressure waves can be identified that move with the speed of sound from the innermost up to the outermost subdomain. In the out-

ermost subdomain, the inexpensive linearized Euler equations are solved. Microphones located in this subdomain capture the pressure perturbation used to provide information on the sound pressure level. Even though the microphones' location is far from the jet-inflow and the airfoil geometry, they captured perturbations precisely, indicating that pressure waves have been transported accordingly due to the high-order scheme. As a result, pressure waves become more significant over time (higher frequency) and move towards the outer boundaries of the outermost subdomain, where artificial sponge layers absorb them. Thus, they damp the state values to a predefined target state and avoid reflections at the boundaries.

Furthermore, we illustrated that the coupling approach acts as expected. The exchange of information at the coupling interfaces agrees with each other, highlighting the accurate information transfer and transport over the respective coupling interfaces. Presented results also confirm that the different subdomains can hardly be distinguished without indication of the coupling interfaces. Finally, we performed a strong scaling measurement, where we pointed out that with increasing node count, the compute time can be considerably decreased. However, our measurement deviates from the ideal scaling due to the limitation of the utilized load balancing approach.

11. Conclusion

This work presented an efficient method to model moving geometries in compressible flows utilizing the high-order Discontinuous Galerkin method. We introduced a straightforward method to model moving geometries with the same discretization method as for the fluid dynamic equations, namely the Brinkman penalization method. This volume penalization technique is an embedded or immersed boundary method, where the geometrical constraints are incorporated in the fluid dynamic equations to be solved. It allows the separate treatment of geometries from the computational mesh, chosen as a simple Cartesian grid. This procedure is desirable in the case of moving geometries as the mesh remains unchanged over the simulation time and necessitates to be generated only once a priori. Consequently, no special treatment of the moving geometry in the computational mesh and additional mesh adaptation are required due to its motion. We demonstrated how this method is incorporated in our numerical scheme and realized in polynomial function space. In addition, further optimization is introduced to reduce the computational cost for the geometry throughout the simulation.

With the obtained knowledge, we validated this method using known one-dimensional and two-dimensional benchmarks from literature. Simulations are conducted from the reflection of an acoustics pulse to the formation of shocks, from subsonic flows to supersonic flows, and from straight to curved and sharp geometrical boundaries. In all cases, the numerical solutions are in excellent agreement with available exact solutions. Furthermore, we exposed in our studies that with the same number of degrees of freedom, the numerical solution has a higher accuracy when utilizing a high-order scheme with fewer computational elements than a low-order scheme and a fine mesh (more elements). This observation holds, even in the vicinity of shocks, where the error convergence order breaks down. Conducted simulations with sharp and curved moving geometries presented, how this method can provide exact solutions, even when the mesh is not adapted towards the geometrical boundary. Often engineering applications are a composition of more than one component and may include the collision of different geometries. Our studies also incorporated

these scenarios, where the geometry is a composition of more than one object and moves throughout the simulation time. Additionally, the collision of multiple geometries is presented. These studies evidenced that the modeling method and the high-order discretization can deal with complex applications, independent of the geometry composition and its motion. Furthermore, we presented a prediction model to estimate the additional computational effort for complex moving geometries (polygons). Thus, it allows estimating the required computational time before an actual run on HPC systems.

For the efficient computation of multi-scale problems, especially when the simulation of the acoustics far-field is of interest, we utilize a partitioned coupling approach. The large simulation domain is decomposed into smaller subdomains, according to the occurring physics in the respective areas of the flow field. Each subdomain receives a dedicated configuration that permits capturing occurring physics accordingly. The spatial discretization of the domains is controlled by the computational mesh and the polynomial approximation. To further enhance the efficiency of this method, each subdomain is solved by a dedicated set of equations. It permits to neglect terms in the fluid dynamic equations that do not play a role from the physical perspective. For example, this is the case in the acoustics far-field, where a homogeneous flow field is encountered. Thus, only wave propagation occurs, viscous effects and nonlinear phenomena can be ignored. Therefore, a simplified set of equations is sufficient to specify the physics there. Examinations conducted in this work for the partitioned approach include the quality of the solution at the coupling interface while comparing two coupling approaches. Coupling tools enable the data exchange at the subdomain boundaries. The white-box approach is part of the simulation framework used in this work and has knowledge about the underlying scheme. The black-box approach, on the other hand, has less knowledge about the coupling domains. Therefore, coupled simulations using the white-box approach permit a high accuracy for the numerical solution, which is close to the classical monolithic approach (no coupling). In the case of the black-box approach, additional features are required to attain a similar solution error compared to the white-box approach. However, due to the flexibility of the black-box approach, the coupling of different solvers is possible.

To achieve more efficient computation for the coupled simulation, improvements are introduced to reduce the data exchange. Additionally, load balancing is used to address load imbalances occurring on the intra- and

inter-subdomain levels. With this, the varying workload of the geometry and the computational mesh is considered, which can be finer or coarser in some areas (multi-level) of the simulation domain. Thus, we can achieve savings in the computation for the coupled simulation through the mentioned appropriate measures, allowing for even more efficient simulations. Lastly, we showed a large-scale simulation, where we investigated the near-field, the turbulent flow, and the acoustics far-field. Even though the simulation domain is decomposed, the transition from one subdomain to the other is smooth, with no observable degradation at the coupling boundaries.

This work can be continued in different directions. First investigations may be conducted to a better re-projection method for the post-processing to capture shocks more precisely up to the point of discontinuity. Here the Gegenbauer re-projection or the Pade approximation can be considered. However, these methods imply additional parameters for post-processing. Furthermore, the geometry modeling can be further extended by incorporating the full interaction of fluid and structure. It requires including the flow response on the geometry, enabling the structure to capture deformations due to the fluid flow. Additionally, it is worth further investigating the update of the geometry during run time. The geometry position was tracked and updated in each substage of the time-stepping scheme in this work. This practice is, of course necessary, to maintain stability and the accuracy of the solution. However, in the case the motion of the geometry is small, it might be possible to reduce the computational effort in our case to 1/4. Hence the position of the geometry is only tracked once per time step.

Improvements in the direction of partitioned coupling can be the coupling in time that necessitates being investigated and enabled. Different phenomena often have different scales in space and time and have to be resolved accordingly. Due to the a priori decomposition of the simulation domain, e.g., the viscous/inviscid nonlinear domain might be chosen larger or smaller than required. Therefore an adaptive approach is helpful, where the overall domain is not decomposed. At the same time, the solver utilizes an adaptive approach to distinguish when, e.g., nonlinear effects can be neglected, and the linearization of the equations is allowed. It enables more efficient computation of the simulation domain and a more immediate transition to solve the linearized equations as soon as physically allowed. Indicators might be, e.g., energy or momentum. Thus the change in the conserved quantities, as presented, e.g., in [56].

In this work, we presented an overview of possible applications to model geometries and efficiently computed multi-scale problems. The methods used in this work can be deployed in different research areas. We demonstrated how the high-order approach is suited for modeling moving geometries, utilizing the same discretization method as the fluid flow, and how it only introduces little numerical dissipation. Thus, it can be deployed for, e.g., compressible turbulence investigations or aeroacoustic noise generation of transonic or even supersonic flows. The application area of high-order discretization methods, the embedded method, and the partitioned coupling is diverse and enables the investigation of various fluid dynamic problems.

A. Appendix

A.1. Multi-scale simulations - Partitioned coupling with *preCICE*

As mentioned in Section 7.3.3 in order to achieve accurate solutions from the numerical simulation, when considering the black-box coupling approach $preCICE$, different interpolation methods can be used for the data exchange. This section shows how the connectivity information for the Nearest-Projection method is generated and provided to the coupling tool for the interpolation. In Section 7.3 we already mentioned that in order to couple the solver with the coupling tool $preCICE$, an API (application programming interface) is used. It is required to allow for communication between the coupling tool and the solver. Through the interface, they can exchange information such as the coordinates of the coupling points, at which data is sent and received. Furthermore, the interface is essential since the solver *Ateles* is written in Fortran programming language, while $preCICE$ is based on C. Usually, only coupling points are communicated between the solver and the coupling tool. However, as discussed in Section 7.3.3 to obtain accurate simulation results at the coupling interface and increase the accuracy of the solution, we consider the 2^{nd} order interpolation method Nearest-Neighbor. For this interpolation, not only a list of coupling points is required, but also connectivity information. Therefore, the coupling points are connected for the connectivity, and a mesh based on triangular mesh elements is built. The coupling tool then uses the mesh for interpolation purposes of the coupling variables that have to be provided to the solver.

In Algorithm 1 the connectivity for the interpolation method is created. Neighboring elements are connected through edges. For example, to create triangular elements, the four edges that build a rectangular element are divided into two triangular elements through a diagonal edge. Two for-loops are used to capture both interface directions. As for three-dimensional simulations, a coupling interface is only a two-dimensional plane.

Algorithm 1 Generate connectivity information for Nearest-Projection

```
1: for i = 1, nCouplingPointsPerDir do
2:     for j = 1, nCouplingPointsPerDir do
3:         if i ≤ nCouplingPointsPerDir
4:            & j ≤ nCouplingPointsPerDir -1 then
5:             Create the vertical edges using two neighboring
6:             points and give them an ID
7:         end if
8:         if  ≤ nCouplingPointsPerDir
9:            & i ≤ nCouplingPointsPerDir -1 then
10:            Create the horizontal edges using two neighboring
11:            points and give them an ID
12:        end if
13:        if i ≤ nCouplingPointsPerDir -1
14:           & j ≤ nCouplingPointsPerDir -1 then
15:            Create the diagonal edges using two vertices
16:            and give them an ID
17:            Create triangles using three edgeIDs (horizontal,
18:            vertical and diagonal)
19:        end if
20:    end for
21: end for
```

Afterward, the information on the edges must be provided to *preCICE* for each element face. Thus the information is gathered element-wise and provided to the coupling tool. Finally, *preCICE* needs the vertexIDs, which are the same as the coupling points. In order to create the respective edges, both vertexIDs and the respective edgeIDs has to be provided (cf. Algorithm 2).

Algorithm 2 Provide connectivity information to *preCICE*

```
1: if Dimension > 1 thens
2:     for iFace = 1, nFaces do
3:         for iEdge = 1, nEdges do
4:             Call the preCICE interface and provide the
5:         meshID, FirstVertexID, SecondVertexID and the EdgeID
6:         end for
7:     end for
8: end if
9: if Dimension > 2 then
10:    for iFace = 1, nFaces do
11:        for iEdge = 1, nTriangles do
12:            Call the preCICE interface and provide the
13:        meshID, FirstEdgeID, SecondEdgeID and the ThirdID
14:        end for
15:    end for
16: end if
```

The meshID is used to identify all information belonging to the respective coupling domain. Each subdomain has its unique meshID, with that *preCICE*, can gather and identify information belonging to a certain domain.

A.2. Reduced computation inside the geometry - Code

In Section 6.6 we introduced how the disadvantage of the embedded method is tackled using the reduced computation inside elements that are covered by the geometry. They only use the integral mean for the approximation of the solution. The following steps are conducted in the solver, in order to **(i)** identify elements that are covered by the geometry, **(ii)** check the state of the neighboring elements, and **(iii)** reduce the order for the approximation of the physical flux to the integral mean only.

First, elements that are covered by the geometry have to be identified. Therefore inside the routine, where the masking function χ is evaluated, we examine the state of the function in each element that has no constant material property. When the geometry is present, an element has no constant state, thus $\chi \neq 0$. The state of the masking function χ is determined in each of those elements. For example, if the state of χ is 1.0, the element is tagged as a potential candidate for the reduced approximation. Afterward,

the neighboring elements are identified, and their state is confirmed. If both neighboring elements for a one-dimensional problem have $\chi = 1.0$, then the element is supposed to be used for reduced computation. Otherwise, the reduction is not utilized during computation (cf. Algorithm 3).

Algorithm 3 Check the state of the masking function of each element

```
for iElem = 1, nElems do
    Check the state of χ for this element
    if State of χ = 1.0 then
        Tag element as potential candidate for reduced computation
    end if
end for
for iElem = 1, nElems do
    Element is tagged with χ == 1
    for iNeighbor= 1, nNeighbors do
        Check the state of the neighboring elements
        if Neighboring element has χ = 1.0 then
            Neighboring element is inside the geometry
        end if
    end for
    The computational element is surrounded by elements that are inside
        the geometry. Therefore, it can be reduced in computation.
end for
```

The during computation of the physical flux, the solver, considers the tagged elements and proceeds with only computing the integral mean (first mode) for those identified elements.

A.3. Sharp boundary - Supersonic moving wedge

In this section, the time series of the wedge test case, previously discussed in Section 6.5 is presented. A wedge is close to the right boundary and moves with a predefined supersonic speed towards the left boundary. The fluid is initially at rest. An oblique shock wave is formed at the nose of the wedge. The angle of the shock can be analytically determined and compared to the numerical solution. More details on the configuration of this test case can be found in Section 6.5. All figures depict the solutions for different scheme orders and at simulation times of 0.05 s, 0.1 s, 0.15 s,

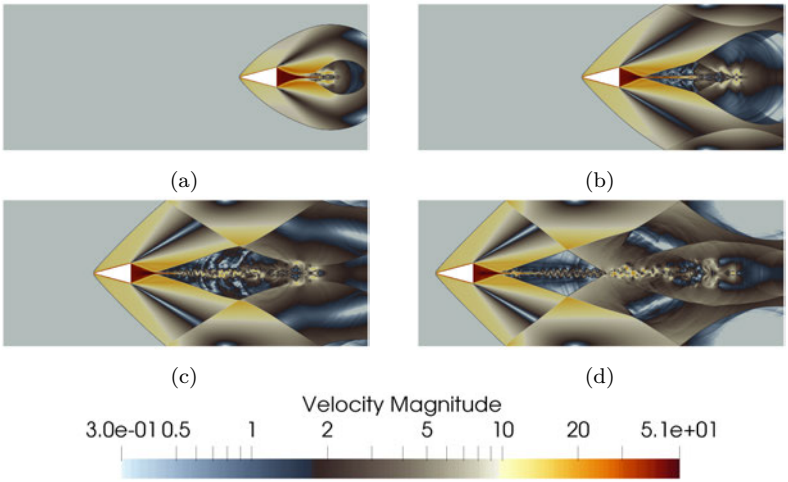


Figure A.1.: Time evolution of a supersonic translating wedge for scheme order $O(5)$ and $M_w(2) - \theta(15^\circ)$. The numerical solution is shown after: (a) 0.05 s, (b) 0.10 s, (c) 0.15 s and (d) 0.2 s.

and 0.2 s. In Figure A.1 the time series for the solution with a scheme order of $O(5)$ is provided. Figure A.1a depicts the solution after 0.05 s. Due to the wedge's supersonic motion, the flow field, which was initially at rest, is disturbed. Furthermore, the formation of an oblique shock ahead of the wedge can be recognized. From Figure A.1b to Figure A.1d the interaction of the shock with the upper and lower wall can be observed. Additionally, in all figures, the formation of vortices of different scales is noticeable.

The figures are shown in Figure A.2 illustrate the time evolution of the solution for a scheme order of $O(6)$. Again the shock-wall interaction can be recognized as well as the shock formation at the wedge nose. Compared to Figure A.1 more and smaller scales are resolved, observable in the middle of the domain. Further, as mentioned in Section 6.5 the shock angle is captured more precisely in the case of $O(6)$ than for $O(5)$. Furthermore, due to the higher scheme order, the spatial resolution is improved. Thus more integration points are available to represent the geometry with higher accuracy. It positively influences the wedge modeling, and the oblique

shock is sharper at the nose, as the wedge can be represented more precisely. Comparing the solutions of $O(6)$ with those of $O(8)$ illustrated in

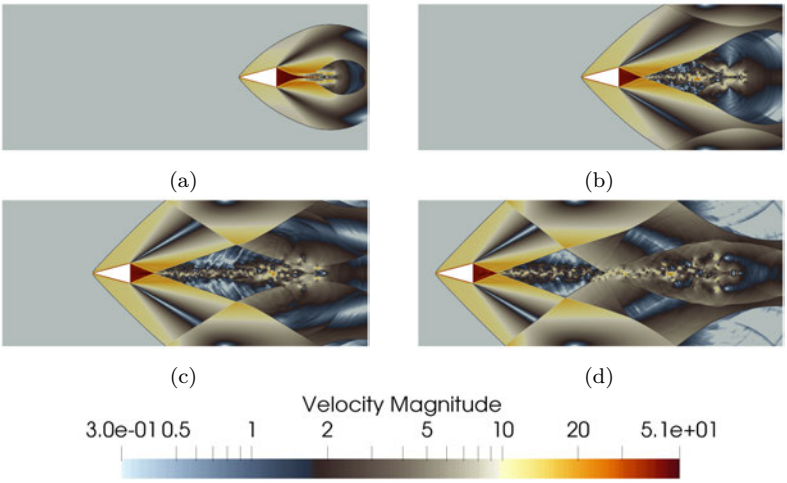


Figure A.2.: Time evolution of a supersonic translating wedge for scheme order $O(6)$ and $M_w(2)$ - $\theta(15°)$. The numerical solution is shown after: (a) 0.05 s, (b) 0.10 s, (c) 0.15 s and (d) 0.2 s.

Figure A.3, we can recognize that not only more features of the flow are resolved (e.g., smaller vortices) but also the sharpness of the shock is improved. In Figure A.3d the sharpness of the oblique shock can be observed. Furthermore, the previous curvy form of the shock is now improved and provides only a neglectable deviation from the exact solution as shown in Section 6.5. Lastly, in our examination, the solutions for the simulation with a scheme order $O(10)$ are shown in Figure A.4. The last configuration allows capturing the exact solution by the numerical solution (shock angle). Thus, the sharpness of the shock is perfectly predicted. Moreover, all characteristics of this problem are resolved likewise the previously shown solutions of the scheme orders $O(5)$, $O(6)$ and $O(8)$.

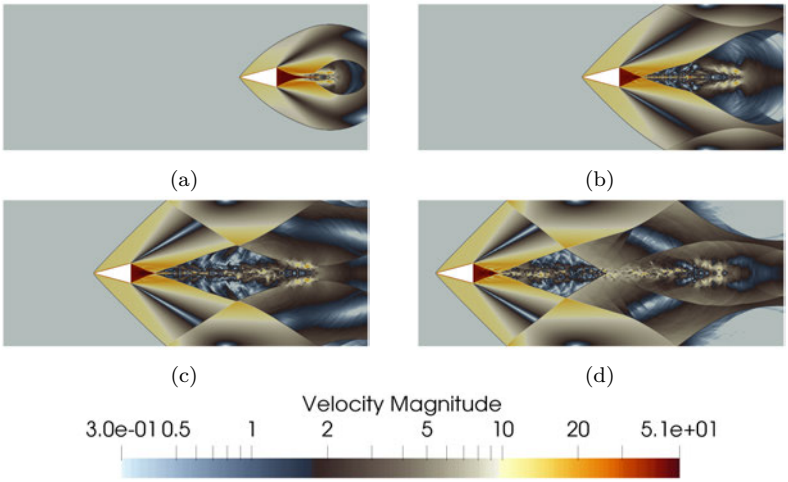


Figure A.3.: Time evolution of a supersonic translating wedge for scheme order $O(8)$ and $M_w(2) - \theta(15^\circ)$. The numerical solution is shown after: (a) 0.05 s, (b) 0.10 s, (c) 0.15 s and (d) 0.2 s.

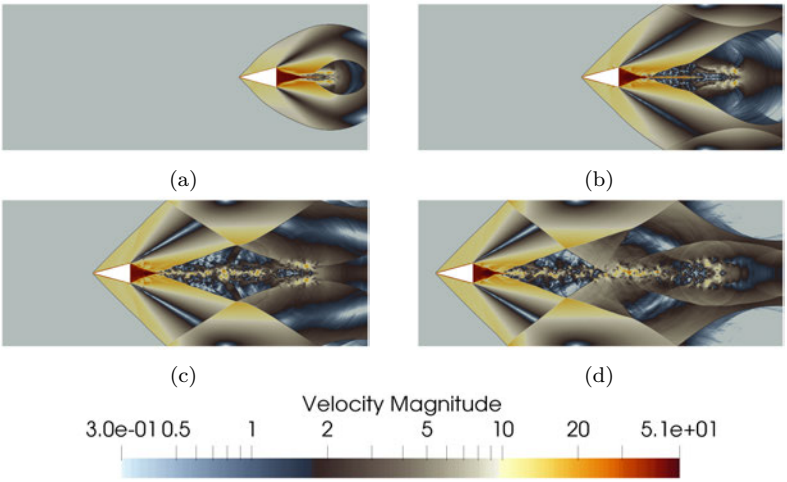


Figure A.4.: Time evolution of a supersonic translating wedge for scheme order $O(10)$ and $M_w(2) - \theta(15^\circ)$. The numerical solution is shown after: (a) 0.05 s, (b) 0.10 s, (c) 0.15 s and (d) 0.2 s.

A.4. Sponge - Reduced boundary reflections

Due to the limitation of the simulation domain by boundary conditions, reflections or even numerical instability caused by the outgoing information occur at the boundaries. In order to avoid this scenario for our large-scale coupled simulation and prevent outgoing pressure waves in the outermost domain from being reflected into the simulation domain, we consider sponge layers. Sponge layers can be seen as artificial areas in the simulation domain, placed in the vicinity of inflow or outflow boundaries to enforce the information close to the boundaries to a particular state before reaching the boundary itself. It allows reducing the intensity of the state variables, thus the reflection of, e.g., acoustics waves back into the simulation domain. The sponge layer used in this work was introduced by J.B. Freund [39]. He investigated the sponge layer and presented promising results. It is available in the high-order solver *Ateles* used in this work and can be utilized by all equations implemented in the solver.

The working principle of the sponge layer is given in Eq. (A.1), where the linearized Euler equations are presented. In addition, a damping function σ is introduced on the rhs of the conservation equations. It is used to reduce the intensity of the state variables to a predefined one. Thus the sponge layer acts as a source term in the conservation equations.

$$\partial_t\ \rho' + \nabla \cdot \underbrace{\left(\boldsymbol{u}_c\rho' + \rho_c\boldsymbol{u}'\right)}_{:=\boldsymbol{m_u}} = -\boldsymbol{\sigma} \cdot (\rho' - \rho'_{target}) \tag{A.1a}$$

$$\partial_t\ \boldsymbol{u}' + \nabla \cdot \left(\boldsymbol{u}_c\boldsymbol{u}' + \frac{1}{\rho_c}p'\right) = -\boldsymbol{\sigma} \otimes (\boldsymbol{u}' - \boldsymbol{u}'_{target}) \tag{A.1b}$$

$$\partial_t\ p' + \nabla \cdot \left(\boldsymbol{u}_c p' + \gamma p_c \boldsymbol{u}'\right) = -\boldsymbol{\sigma} \cdot (p' - p'_{target}). \tag{A.1c}$$

A target state ($_{target}$) is defined, to which the state is slowly damped in space over time, depending on the function σ utilized. This can be either a linear decrease to the target state or, e.g., a quadratic one. The damping function depends on the width of the sponge layer and the damping factor (σ_I). The damping factor defines how strong the sponge layer is. If this factor is chosen too strong, then the outgoing information will be reflected into the computational domain before reaching the domain's boundaries, which is counterproductive. Therefore the damping factor has to be chosen accordingly to deploy its benefits and avoid reflections of outgoing information.

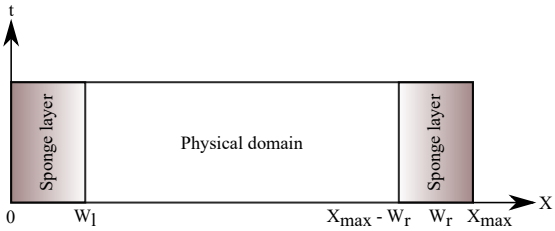


Figure A.5.: Principle of the sponge layer for a one-dimensional test case. Simulation domain with two boundary conditions on the right and left sides. In the vicinity of each boundary, a sponge layer is located to avoid reflections by the boundaries.

Our investigation has shown that: **(i)** The width of the sponge layer has to be thick enough to damp the state to the desired value. On the other hand, a too-thin layer does not allow appropriate damping. Thus the state has to be damped down in a short interval, causing strong gradients and resulting in strong oscillations of the polynomial representation. **(ii)** The strength of the sponge, hence the damping factor can be calculated according to the width of the sponge layer and the speed of sound. It allows achieving reflection-free results. Further, the choice of the damping exponent (α_d) allows to determine how fast the outgoing information is reduced in intensity before reaching the boundaries (cf. Eq. (A.2). An exponent of 1 results in a linear decrease. Thus the state is slowly reduced in intensity, while a quadratic one results in a faster intensity reduction. For simplicity reasons and a better understanding of the principle of the sponge layer, a small one-dimensional test case is shown in Figure A.5. On both sides of the domain, a sponge layer is positioned before the boundaries, located between the physical domain. Both sponge layers have a width of W_l and W_r, respectively. The sponge region is physically not relevant for the solution. However, it helps to overcome reflections or even instabilities caused by strong nonlinearities, e.g., vortices at the boundaries.

Corresponding to Figure A.5, a linear function is used to damp the solution to a target state in one-dimensional space, which is computed as

$$\sigma(x) = \begin{cases} \sigma_{Il} \cdot \left(\frac{W_l - x}{W_l}\right)^{\alpha_d} & 0 \leq x < W_l \\ 0 & W_l \leq x < X_{max} - W_r \\ \sigma_{Ir} \left[\frac{x-(x_{max}-W_r)}{W_r}\right]^{\alpha_d} & X_{max} - W_r \leq x \leq X_{max}. \end{cases} \tag{A.2}$$

Here W_l and W_r define the width of the left and right sponge layer, respectively. X_{max} determines the maximum length of the domain and σ_{Il} and σ_{Ir} the damping factor that defines how strong the sponge layer is. From Eq. (A.2), it is apparent that the sponge has its maximum in the damping process close to the boundaries of the domain, where the end part of the sponge can be found. In the area facing the physical domain, the strength of the sponge is the lowest. It is designed such that the information that travels to the boundary loses in amplitude before reaching it. More on this method can be found in [39].

Bibliography

[1] I.H. Abbott and A.E. Von Doenhoff. *Theory of Wing Sections, Including a Summary of Airfoil Data*. Dover Books on Aeronautical Engineering Series. Dover Publications, 1959. ISBN: 9780486605869. URL: https://books.google.de/books?id=DPZYUGNyuboC.

[2] R. Ahrem et al. "MpCCI A tool for coupling CFD with other disciplines". In: *5th World Conference in Applied Fluid Dynamics. Simulation and Visualisation 2001 Conference; SIM 2001. Proceedings of the Conference*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001.

[3] R. Alexander. "Diagonally Implicit Runge–Kutta Methods for Stiff O.D.E.'s". In: *SIAM Journal on Numerical Analysis* 14.6 (1977), pp. 1006–1021. ISSN: 0036-1429. DOI: 10.1137/0714068. (Visited on 07/27/2016).

[4] J Almeida, M Alves-Pereira, and P Nossa. "The impact of wind turbine noise on health". In: *European Journal of Public Health* 29.Supplement_1 (2019). ISSN: 1101-1262. DOI: 10.1093/eurpub/ckz034.060. URL: https://doi.org/10.1093/eurpub/ckz034.060.

[5] Bradley K. Alpert and Vladimir Rokhlin. "A Fast Algorithm for the Evaluation of Legendre Expansions". In: *SIAM J. Scientific Computing* 12 (1991), pp. 158–179.

[6] Bradley K. Alpert and Vladimir Rokhlin. "A Fast Algorithm for the Evaluation of Legendre Expansions". In: *SIAM Journal on Scientific and Statistical Computing* 12.1 (1991), pp. 158–179. DOI: 10.1137/0912009.

[7] Nikhil Anand, Harald Klimach, and Sabine Roller. "Dealing with Nonlinear Terms in a Modal High-Order Discontinuous Galerkin Method". In: *Sustained Simulation Performance 2016*. Ed. by Michael M. Resch et al. Cham: Springer International Publishing, 2016, pp. 43–59. ISBN: 978-3-319-46735-1.

[8] Nikhil Anand et al. "Utilization of the Brinkman Penalization to Represent Geometries in a High-Order Discontinuous Galerkin Scheme on Octree Meshes". In: *Symmetry* 11.9 (2019), pp. 11–26. DOI: 10.3390/sym11091126.

[9] Philippe Angot, Charles-Henri Bruneau, and Pierre Fabrie. "A penalization method to take into account obstacles in incompressible viscous flows". In: *Numerische Mathematik* 81.4 (1999), pp. 497–520. DOI: 10.1007/s002110050401.

[10] Steve Ashby et al. "The opportunities and challenges of exascale computing". In: *Summary Report of the Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee* (2010), pp. 1–77.

[11] H. F. Bauer. "Morand, H. J.-P.; Ohayon, R.: Fluid Structure Interaction. Applied Numerical Methods. Chichester etc., John Wiley & Sons; Paris etc., Masson 1995. VIII, 212 pp., F 320,?. ISBN 2-225-84682-0". In: *ZAMM - Journal of Applied Mathematics and Mechanics* 76.7 (1996), pp. 376–376. DOI: 10.1002/zamm.19960760705. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/zamm.19960760705. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/zamm.19960760705.

[12] Yuri Bazilevs, Kenji Takizawa, and Tayfun Tezduyar. *Computational Fluid-Structure Interaction: Methods and Applications*. 2013. ISBN: 9780470978771. DOI: 10.1002/9781118483565.

[13] Yuri Bazilevs, Kenji Takizawa, and Tayfun E Tezduyar. *Computational Fluid-Structure Interaction: Methods and Applications*. John Wiley & Sons, 2012.

[14] G. Ben-Dor, O. Igra, and T. Elperin, eds. *Handbook of Shock Waves*. Academic Press, 2001.

[15] Julien Berland, Christophe Bogey, and Christophe Bailly. "Numerical study of screech generation in a planar supersonic jet". In: *Physics of Fluids - PHYS FLUIDS* 19 (2007). DOI: 10.1063/1.2747225.

[16] D. Böhme, F. Wolf, and M. Geimer. "Characterizing Load and Communication Imbalance in Large-Scale Parallel Applications". In: *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops, PhD Forum*. 2012, pp. 2538–2541.

[17] Hans-Joachim Bungartz, Miriam Mehl, and Schäfer Michael. *Fluid structure interaction II*. Springer, 2010.

[18] Hans-Joachim Bungartz et al. "Partitioned Fluid-Structure-Acoustics Interaction on Distributed Data – Coupling via preCICE". In: *Software for Exa-scale Computing – SPPEXA 2013-2015*. Ed. by Hans-Joachim Bungartz, Philipp Neumann, and E. Wolfgang Nagel. Springer Berlin Heidelberg, 2016.

[19] Hans-Joachim Bungartz et al. "preCICE – A Fully Parallel Library for Multi-Physics Surface Coupling". In: *Computers and Fluids* (2015).

[20] J. C. Butcher. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*. USA: Wiley-Interscience, 1987. ISBN: 0471910465.

[21] Neil Cameron and D. Carter. "Rotorblade trailing edge flap failure modes and effects analysis". In: *35th European Rotorcraft Forum 2009, ERF 2009* 1 (2009), pp. 134–148.

[22] Horatio Scott Carslaw. *Introduction to the Theory of Fourier's Series and Integrals*. New York: Macmillan, 1921.

[23] Arnab Chaudhuri, Abdellah Hadjadj, and Ashwin Chinnayya. "On the use of immersed boundary methods for shock/obstacle interactions". In: *Journal of Computational Physics* 230.5 (2011), pp. 1731–1748. DOI: 10.1016/j.jcp.2010.11.016.

[24] Bernardo Cockburn, Suchung Hou, and Chi-Wang Shu. "The Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws. IV: The Multidimensional Case". In: *Mathematics of Computation* 54.190 (1990), p. 545. DOI: 10.2307/2008501.

[25] Bernardo Cockburn and Chi-Wang Shu. "TVB Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws II: General Framework". In: *Mathematics of Computation* 52.186 (1989), p. 411. DOI: 10.2307/2008474.

[26] S. Scott Collis and Kaveh Ghayour. "Discontinuous Galerkin Methods for Compressible DNS". In: *Volume 2: Symposia, Parts A, B, and C* (2003). DOI: 10.1115/fedsm2003-45632.

[27] National Research Council. *Low-Frequency Sound and Marine Mammals: Current Knowledge and Research Needs*. Washington, DC: The National Academies Press, 1994. ISBN: 978-0-309-05025-8. DOI: 10.17226/4557. URL: https://www.nap.edu/catalog/4557/low-frequency-sound-and-marine-mammals-current-knowledge-and-research.

[28] Paolo Crosetto et al. “Parallel Algorithms for Fluid-Structure Interaction Problems in Haemodynamics”. In: *SIAM Journal on Scientific Computing* 33.4 (2011), pp. 1598–1622.

[29] Andreas Dedner and Robert Klöfkorn. “A Generic Stabilization Approach for Higher Order Discontinuous Galerkin Methods for Convection Dominated Problems”. In: *Journal of Scientific Computing* 47.3 (2010), 365?388. DOI: 10.1007/s10915-010-9448-0.

[30] Shaozhong Deng. “On the immersed interface method for solving time-domain Maxwell’s equations in materials with curved dielectric interfaces”. In: *Computer physics communications* 179.11 (2008), pp. 791–800. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2008.07.001. URL: https://europepmc.org/articles/PMC2886521.

[31] E.H. Dowell and K.C. Hall. “Modeling of fluid-structure interaction”. In: *Annual Review of Fluid Mechanics* 33.1 (2001), pp. 445–490. ISSN: 0066-4189. DOI: 10.1146/annurev.fluid.33.1.445. URL: http://www.annualreviews.org/doi/abs/10.1146/annurev.fluid.33.1.445.

[32] J.P. Caltagirone E. Arquis. “Sur les conditions hydrodynamiques au voisinage d’une interface milieu fluide-milieu poreux: application a‘ la convection naturelle”. In: *C.R. Acad. Sci. Paris II 299* (1984), pp. 1–4.

[33] Neda Ebrahimi Pour and Sabine Roller. “Error investigation for coupled simulations using Discontinuous Galerkin method for discretisation”. In: *In Proceedings of ECCM VI / ECFD VII*. Glasgow, UK, 2018.

[34] Neda Ebrahimi Pour et al. “Brinkman Penalization and Boundary Layer in High-Order Discontinuous Galerkin”. In: *Sustained Simulation Performance 2019 and 2020*. Ed. by Michael M. Resch et al. Springer International Publishing, 2021. ISBN: 978-3-030-68049-7.

[35] Neda Ebrahimi Pour et al. “Coupled Simulation with Two Coupling Approaches on Parallel Systems”. In: *Sustained Simulation Performance 2017*. Ed. by Michael M. Resch et al. Springer International Publishing, 2017, pp. 151–164. DOI: 10.1007/978-3-319-66896-3_10.

[36] Neda Ebrahimi Pour et al. “Load Balancing for Immersed Boundaries in Coupled Simulations”. In: *Sustained Simulation Performance 2018 and 2019*. Ed. by Michael M. Resch et al. Springer International Publishing, 2019, pp. 185–201. DOI: 10.1007/978-3-030-39181-2_15.

[37] *ExaFSA*. http://www.sppexa.de/general-information/projects-phase-2.html#MEHL2.

[38] Bengt Fornberg and Julia Zuev. “The Runge Phenomenon and Spatially Variable Shape Parameters in RBF Interpolation”. In: *Comput. Math. Appl.* 54.3 (2007), pp. 379–398. ISSN: 0898-1221. DOI: 10.1016/j.camwa.2007.01.028. URL: http://dx.doi.org/10.1016/j.camwa.2007.01.028.

[39] J. Freund. “Proposed inflow/outflow boundary condition for direct computation of aerodynamic sound”. In: *AIAA Journal* 35 (1997), pp. 740–742.

[40] Michael W Gee, Ulrich Küttler, and Wolfgang A Wall. “Truly monolithic algebraic multigrid for fluid-structure interaction”. In: *International Journal for Numerical Methods in Engineering* 85.8 (2011), pp. 987–1016.

[41] E. Glazer et al. “Velocity scaling of a shock wave reflected off a circular cylinder”. In: *Physical Review E* 83.6 (2011). DOI: 10.1103/physreve.83.066317.

[42] David Gottlieb and Chi-Wang Shu. “On the Gibbs Phenomenon and Its Resolution”. In: *SIAM Review* 39.4 (1997), 644?668. DOI: 10.1137/s0036144596301390.

[43] D.g.e. Grigoriadis, S.c. Kassinos, and E.v. Votyakov. “Immersed boundary method for the MHD flows of liquid metals”. In: *Journal of Computational Physics* 228.3 (2009), 903?920. DOI: 10.1016/j.jcp.2008.10.017.

[44] Daniel F. Harlacher et al. “Dynamic Load Balancing for Unstructured Meshes on Space-Filling Curves”. In: *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*. 2012, pp. 1661–1669.

[45] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach*. 5th ed. Amsterdam: Morgan Kaufmann, 2012. ISBN: 978-0-12-383872-8.

[46] Jan S. Hesthaven and Tim Warburton. "Nodal Discontinuous Galerkin Methods". In: *Texts in Applied Mathematics* (2008). DOI: 10.1007/978-0-387-72067-8.

[47] Jan S. Hesthaven and Tim Warburton. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. 1st. Springer Publishing Company, Incorporated, 2007. ISBN: 9780387720654.

[48] Gene Hou, Jin Wang, and Anita Layton. "Numerical Methods for Fluid-Structure Interaction — A Review". In: *Communications in Computational Physics* 12.2 (2012), pp. 337–377. DOI: 10.4208/cicp.291210.290411s.

[49] M.S. Howe, M.S. Howe, and M.J. Ablowitz. *Acoustics of Fluid-Structure Interactions*. Cambridge Monographs on Mechanics. Cambridge University Press, 1998. ISBN: 9780521633208. URL: https://books.google.de/books?id=xsE2UID0uakC.

[50] Pei-Ching Hu, Joseph E. Flaherty, and Mark S. Shephard. "Solving Fluid / Rigid body Interaction Problem by a Discontinuous Galerkin Level Set Method". In: 2005.

[51] John David Anderson (jr). *Fundamentals of Aerodynamics* -. New York: McGraw-Hill, 2010. ISBN: 978-0-070-70012-3.

[52] Nicholas K.-R Kevlahan and Jean-Michel Ghidaglia. "Computation of turbulent flow past an array of cylinders using a spectral method with Brinkman penalization". In: *European Journal of Mechanics - B/Fluids* 20.3 (2001), pp. 333–350. DOI: 10.1016/s0997-7546(00)01121-3.

[53] Robert M. Kirby and George Em Karniadakis. "De-aliasing on non-uniform grids: algorithms and applications". In: *Journal of Computational Physics* 191.1 (2003), 249?264. DOI: 10.1016/s0021-9991(03)00314-0.

[54] Harald Klimach. "Parallel Multi-Scale Simulations with Octrees and Coupled Applications". PhD thesis. RWTH Aachen University, 2016.

[55] Harald G. Klimach et al. "Distributed Octree Mesh Infrastructure for Flow Simulations". In: *ECCOMAS 2012 - European Congress on Computational Methods in Applied Sciences and Engineering, e-Book Full Papers*. Ed. by J Eberhardsteiner. 2012.

[56] Harald Klimach and Sabine Roller. "Adaptive equation simplification in fluid simulations". GAMM Activity Group CSE, Workshop 2018, University of Siegen. 2018. URL: https://www.mb.uni-siegen.de/nm/workshops/gamm-cse-2018/paper/gamm-cse-2018.pdf.

[57] Ryu Komatsu, Wakana Iwakami, and Yuji Hattori. "Direct numerical simulation of aeroacoustic sound by volume penalization method". In: *Computers & Fluids* 130 (2016), pp. 24–36. DOI: 10.1016/j.compfluid.2016.02.016.

[58] M. Kornhaas, M. Schäfer, and D.C. Sternel. "Efficient numerical simulation of aeroacoustics for low Mach number flows interacting with structures". In: *Computational Mechanics* 55 (2015), pp. 1143–1154. DOI: 10.1007/s00466-014-1114-1.

[59] Verena Krupp. "Efficient coupling of fluid and acoustic interaction on massively parallel systems". PhD thesis. Universität Siegen, 2021. ISBN: 978-3-96182-104-4.

[60] Verena Krupp et al. "Efficient Coupling of Fluid and Acoustic Interaction on Massive Parallel Systems". In: *Sustained Simulation Performance 2016* (2016), pp. 61–81. DOI: 10.1007/978-3-319-46735-1_6.

[61] C. L. Ladson. "Effects of independent variation of Mach and Reynolds numbers on the low-speed aerodynamic characteristics of the NACA 0012 airfoil section". In: 1988.

[62] L. D. Landau and Lifshits E. M. *Fluid mechanics*. Elsevier Butterworth-Heinemann, 2012.

[63] M. J. Lighthill. "On Sound Generated Aerodynamically. II. Turbulence as a Source of Sound". In: *Proceedings of the Royal Society of London Series A* 222.1148 (1954), pp. 1–32. DOI: 10.1098/rspa.1954.0049.

[64] Florian Lindner, Miriam Mehl, and Benjamin Uekermann. "Radial basis function interpolation for black-box multi-physics simulations". In: *VII International Conference on Computational Methods for Coupled Problems in Science and Engineering*. Ed. by M. Papadrakakis, B. Schrefler, and E. Onate. submitted. 2017, pp. 1–12.

[65] Florian Lindner et al. "ExaFSA: Parallel Fluid-Structure-Acoustic Simulation". In: *Software for Exascale Computing - SPPEXA 2016-2019*. Ed. by Hans-Joachim Bungartz et al. Cham: Springer International Publishing, 2020, pp. 271–300. ISBN: 978-3-030-47956-5.

[66] G. Link et al. "A 2D finite-element scheme for fluid-solid-acoustic interactions and its application to human phonation". In: *Computer Methods in Applied Mechanics and Engineering* 198.41-44 (2009), pp. 3321–3334.

[67] Qianlong Liu and Oleg V. Vasilyev. "A Brinkman penalization method for compressible flows in complex geometries". In: *Journal of Computational Physics* 227.2 (2007), pp. 946–966. DOI: 10.1016/j.jcp.2007.07.037.

[68] Kannan Masilamani. "Framework for coupled simulation of electrodialysis processes". PhD thesis. RWTH Aachen University, 2020. ISBN: ISBN 978-3-96182-077-1.

[69] Rajat Mittal and Gianluca Iaccarino. "Immersed Boundary Methods". In: *Annual Review of Fluid Mechanics* 37.1 (2005), pp. 239–261. DOI: 10.1146/annurev.fluid.37.061903.175743.

[70] Gordon E. Moore. "Cramming more components onto integrated circuits". In: *Electronics* 38.8 (1965).

[71] Per-Olof Persson and Jaime Peraire. "Sub-Cell Shock Capturing for Discontinuous Galerkin Methods". In: *44th AIAA Aerospace Sciences Meeting and Exhibit* (2006). DOI: 10.2514/6.2006-112.

[72] Charles Peskin. "Peskin, C.S.: The immersed boundary method. Acta Numerica 11, 479-517". In: *Acta Numerica* 11 (2002), pp. 479–517. DOI: 10.1017/S0962492902000077.

[73] Charles S. Peskin. "Numerical analysis of blood flow in the heart". In: *Journal of Computational Physics* 25.3 (1977), pp. 220–252. ISSN: 0021-9991. DOI: 10.1016/0021-9991(77)90100-0.

[74] Daniele Antonio Di Pietro and Alexandre Ern. "Mathematical Aspects of Discontinuous Galerkin Methods". In: *Mathematiques et Applications* (2012). DOI: 10.1007/978-3-642-22980-0.

[75] A. Piquet, O. Roussel, and A. Hadjadj. "A comparative study of Brinkman penalization and direct-forcing immersed boundary methods for compressible viscous flows". In: *Computers & Fluids* 136 (2016), pp. 272–284. DOI: 10.1016/j.compfluid.2016.06.001.

[76] Neda Ebrahimi Pour. *Investigation of Coupled Fluid-Structure-Acoustic (FSA) Interaction*. Accessed 09-09-2020. 2020. URL: https://www.gauss-centre.eu/results/computational-and-scientific-engineering/article/investigating-coupled-fluid-structure-acoustic-fsa-interaction/.

[77] Neda Ebrahimi Pour et al. "Compressible flow simulation with moving geometries using the Brinkman penalization in high-order Discontinuous Galerkin". In: *Adv. Model. Simul. Eng. Sci.* 8.1 (2021), p. 10. DOI: 10.1186/s40323-021-00195-4. URL: https://doi.org/10.1186/s40323-021-00195-4.

[78] *preCICE*. https://www.precice.org.

[79] Jiaxing Qi. "Efficient Lattice Boltzmann Simulations on Large Scale High Performance Computing Systems". PhD thesis. RWTH Aachen University, 2017.

[80] W.h. Reed et al. "TRIPLET: a two-dimensional, multigroup, triangular mesh, planar geometry, explicit transport code". In: (1973). DOI: 10.2172/4397037.

[81] Sabine Roller et al. "An Adaptable Simulation Framework Based on a Linearized Octree". In: *High Performance Computing on Vector Systems 2011*. Ed. by Michael Resch et al. Springer Berlin Heidelberg, 2012, pp. 93–105. ISBN: 978-3-642-22244-3. DOI: 10.1007/978-3-642-22244-3_7.

[82] Benjamin Rüth et al. "Quasi-Newton Waveform Iteration for Partitioned Surface-Coupled Multi-Physics Applications". In: *International Journal for Numerical Methods in Engineering* (2020). DOI: 10.1002/nme.6443.

[83] T. Schwartzkopff. "Finite-Volumen Verfahren hoher Ordnung und heterogene Gebietszerlegung für die numerische Aeroakustik". PhD thesis. Universität Stuttgart, Institut für Aerodynamik und Gasdynamik, 2005.

[84] Ascher H. Shapiro. *The dynamics and thermodynamics of compressible fluid flow*. 1953.

[85] Chi-Wang Shu and Stanley Osher. "Efficient implementation of essentially non-oscillatory shock-capturing schemes". In: *Journal of Computational Physics* 77.2 (1988), 439?471. DOI: 10.1016/0021-9991(88)90177-5.

[86] *SPPEXA*. http://www.sppexa.de.

[87] Joe F Thompson, Zahir U.a Warsi, and C Wayne Mastin. "Boundary-fitted coordinate systems for numerical solution of partial differential equations A review". In: *Journal of Computational Physics* 47.1 (1982), pp. 1–108. DOI: 10.1016/0021-9991(82)90066-3.

[88] E.F Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*. Springer, 2009.

[89] Amin Totounferoush et al. “A new load balancing approach for coupled multi-physics simulations”. In: *In Proceedings of IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. (Rio de Janeiro, Brazil, May 20–24, 2019). 2019. DOI: 10.1109/IPDPSW.2019.00115.

[90] *Turbulence Modeling Resource*. https://turbmodels.larc.nasa.gov/naca0012_val.html. Accessed: 2021-01-14.

[91] Benjamin Walter Uekermann. “Partitioned Fluid-Structure Interaction on Massively Parallel Systems”. Dissertation. München: Technische Universität München, 2016.

[92] *Umweltbundesamt*. https://www.umweltbundesamt.de/themen/mindestabstaende-bei-windenergieanlagen-schaden-der.

[93] Jens Utzmann. “A domain decomposition method for the efficient direct simulation of aeroacoustic problems”. PhD thesis. Universität Stuttgart, Institut für Aerodynamik und Gasdynamik, 2008.

[94] Jin Wang and Anita Layton. “Numerical Simulations of Fiber Sedimentation in Navier-Stokes Flows”. In: *COMMUNICATIONS IN COMPUTATIONAL PHYSICS Commun. Comput. Phys* 5 (2009), pp. 61–83.

[95] T. Warburton and T. Hagstrom. “Taming the CFL Number for Discontinuous Galerkin Methods on Structured Meshes”. In: *SIAM Journal on Numerical Analysis* 46.6 (2008), 3151?3180. DOI: 10.1137/060672601.

[96] *WHO (2004) Global burden disease*. http://www.who.int/healthinfo/global_burden_disease/GBD_report_2004update_full.pdf?ua=1. Accessed: 2021-01-12.

[97] Weiwei Zhang, Yuewen Jiang, and Zhengyin Ye. “Two Better Loosely Coupled Solution Algorithms of CFD Based Aeroelastic Simulation”. In: *Engineering Applications of Computational Fluid Mechanics* 1.4 (2007), 253?262. DOI: 10.1080/19942060.2007.11015197.

[98] Jens Zudrop. “Efficient Numerical Methods for Fluid- and Electrodynamics on Massively Parallel Systems”. PhD thesis. RWTH Aachen University, 2015.

List of Figures

List of Tables

Simulation Techniques in Siegen
Vol. 6

This work presents an efficient strategy to facilitate large-scale simulations of aeroacoustics induced by rigid body motion. A Discontinuous Galerkin method, providing low dispersion and dissipation errors, is used to propagate the flow-induced acoustics. An embedded boundary method is deployed to model complex moving geometries. Both, flow field and geometry modeling can use the same discretization method. The combination with a partitioned coupling approach allows for efficient massively parallel simulations of multiple scales. The modeling method and the partitioned coupling are thoroughly investigated and used in a detailed simulation of a real-world three-dimensional engineering problem of a moving airfoil that disturbs the fluid flow, resulting in acoustic waves.

Neda Ebrahimi Pour studied mechanical engineering at the University of Siegen. From 2016-2019, she worked on the ExaFSA project in the priority program SPPEXA of the German Research Foundation (DFG). Her primary research in the project was dedicated to the efficient realization of coupled simulations concerning fluid-structure-acoustics interaction. Until early 2021 she worked as a doctoral candidate at the chair of Simulation Techniques and Scientific Computing at the University of Siegen. Since 2021, she has been a research associate at the German Aerospace Center (DLR) at the Institute of Software Methods for Product Virtualization in Dresden.

The series ***Simulation Techniques in Siegen*** presents contributions to the field of scientific computing with a focus on the utilization of large-scale computing systems for highly resolved simulations. Applications, as well as numerical methods and their efficient implementation on modern supercomputers, are investigated and described.

ISBN: 978-3-96182-105-1

universi
UNIVERSITÄTSVERLAG SIEGEN