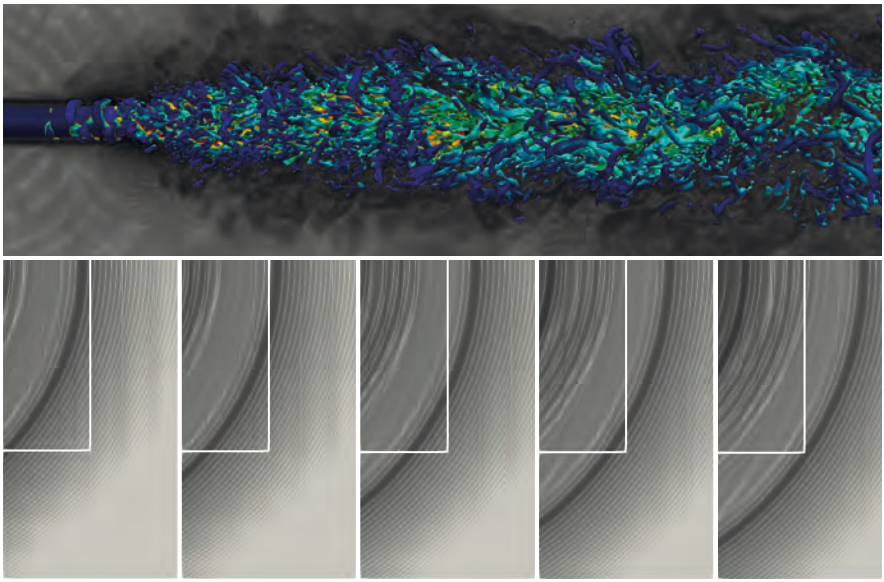
Simulation Techniques in Siegen
Vol. 5

Verena Krupp

Efficient coupling of fluid and acoustic interaction on massively parallel systems



universi
UNIVERSITÄTSVERLAG SIEGEN

Efficient coupling of fluid and acoustic interaction on massively parallel systems

DISSERTATION
zur Erlangung des Grades eines Doktors
der Ingenieurwissenschaften

vorgelegt von

Dipl.-Ing. (FH) Verena Krupp 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. Marek Behr, Ph.D.
RWTH Aachen

Tag der mündlichen Prüfung
27. August 2021

Simulation Techniques in Siegen / STS

Edited by Sabine Roller and Harald Klimach

Vol. 5 (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/10033

Simulation Techniques in Siegen Vol. 5 / STS Vol. 5 (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: 978-3-96182-104-4

Danksagung

Diese Dissertation wäre in ihrer jetzigen Form nicht ohne die Hilfe vieler Menschen möglich gewesen. An dieser Stelle möchte ich diesen Personen meinen Dank aussprechen.

Mein besonderer Dank gilt meiner Doktormutter Prof. Dr.-Ing. Sabine Roller, für die Möglichkeit das Thema dieser Dissertation zu bearbeiten und in der Arbeitsgruppe STS eine so gute Zeit erleben zu können. Ohne unsere Diskussionen, gerade in der letzten Phase dieser Arbeit, wäre ich aus der ein oder anderen Sackgasse nicht herausgekommen. Außerdem möchte ich mich bei Prof. Marek Behr, Ph.D herzlich für seinen Einsatz als Zweitgutachter bedanken. Zudem möchte ich mich bei Harald Klimach im Besonderen bedanken: ohne die unzähligen Diskussionen und Vorschläge wäre diese Arbeit nicht da wo sie jetzt ist. Des Weiteren haben seine Kommentare und das kritische Hinterfragen maßgeblich zur Qualität meiner Publikationen und Implementierungen beigetragen.

Darüber hinaus geht mein Dank an das komplette ExaFSA Projekt des DFG Schwerpunktprogramms “Software for Exascale Computing” (1648) für die Zusammenarbeit. Vor allem die produktive Zusammenarbeit mit Benjamin Uekermann im Rahmen der Implementierung von *preCICE* in *Ateles* hat mir große Freude bereitet.

Für eine großartige und produktive Arbeitsumgebung möchte ich mich bei allen (ehemaligen) Mitgliedern der Arbeitsgruppe STS der Uni Siegen bedanken: es war eine super Zeit! Besonders die Whiteboard-Diskussionen, Kaffee-Sessions und die Zeit im gemeinsamen Büro mit Nikhil Anand möchte ich nicht missen. Auch bedanken muss ich mich bei Peter Vitt, denn ohne ihn wäre die Implementierung und insbesondere meine Programmierfähigkeiten nicht da wo sie heute sind. Großer Dank gilt auch Kannan Masilamani mit dem ich gemeinsam an *APESmate* arbeiten durfte.

Es gibt noch viele Namen auf der Liste und Menschen, die mich bereichert haben. Gaby und Lena, ohne eure Unterstützung wäre mein Studium in Aachen so nicht möglich gewesen. Andreas, danke für dein Mentoring im letzten Jahr. Es hat mir immer wieder geholfen das Ziel nicht aus den Augen zu verlieren und meine Zeit sinnvoll einzusetzen.
Abschließend, gibt es eine ganz besondere Person, die alle meine Höhen und Tiefen mit dieser Dissertation miterlebt und mir stets den Rücken freigehalten hat - danke Stefan!

Zusammenfassung

Mehrskalenprobleme, wie die Schallerzeugung durch Strömung und deren reine Schallausbreitung, sind für die Industrie von wachsender Bedeutung, z.B. für die Reduktion der Schallemission von Windenergieanlagen. Generell können sowohl Strömung als auch akustische Wellenausbreitung durch die selben physikalischen Gleichungen beschrieben werden. Die Beschreibung des reinen Wellentransports kann jedoch wesentlich vereinfacht werden, was in einem geringeren Berechnungsaufwand resultiert. Des Weiteren basieren Strömungsphänomene auf sehr kleinen räumlichen Skalen, wohingegen die akustische Wellenausbreitung auf großen räumlichen Skalen stattfindet. Vor allem dieser Skalenunterschied macht die numerische Berechnung solcher Mehrskalenproblemen anspruchsvoll. So ist eine Simulation des gesamten Gebietes mit der Auflösung, welche durch die sehr kleinen Strömungsskalen vorgegeben ist, aufgrund der hohen Kosten und des hohen Energieverbrauches mit heutigen Computerressourcen nicht durchführbar. Im Rahmen dieser Arbeit wird eine partitionierte Kopplung mittels Oberflächen als Ansatz zur effizienten Simulation solcher Probleme auf massiv parallelen Supercomputern entwickelt. Dabei wird das gesamte Gebiet in kleinere Gebiete einzelner physikalischer Phänomene aufgeteilt und die Interaktion dieser Gebiete durch einen bidirektionalen Datenaustausch an den Rändern realisiert. Die separate Behandlung einzelner Phänomene ermöglicht nicht nur die Nutzung von auf die jeweilige Physik abgestimmter Verfahren (Ordnung, Gitterauflösung, Gleichungen), sondern auch verschiedener numerischer Löser. Diese Herangehensweise birgt aber auch numerische Herausforderungen an den Kopplungsrändern: so sind für einen konsistenten Datenaustausch an den Kopplungsrändern im Falle unterschiedlicher räumlicher Auflösung direkte Datenauswertung oder effiziente Interpolationsmethoden notwendig. Im Rahmen dieser Arbeit werden zwei verschiedene Ansätze zur partitionierten Kopplung von Strömungs- sowie Akustikgebieten im Hinblick auf Qualität der Lösung und Performanz implementiert und untersucht: ein *black-box* und ein *white-box* Ansatz. Dabei verspricht ein *white-box* Ansatz höhere Effizienz, da löserinterne Verfahren zur Datenevaluierung verwendet werden können. Ein *black-box* Ansatz hingegen, zeichnet sich durch Flexibilität in der Wahl der numerischen Löser sowie schneller Umsetzbarkeit aus. Dieser Anspruch auf Flexibilität geht allerdings einher mit einem beschränkten Zugriff auf löserinterne Informationen sowie potenziellen Geschwindigkeitseinbußen. Zudem arbeitet ein solcher Ansatz an den Kopplungsrändern ausschließlich mit Punktwerten, was zur Folge hat, dass externe Interpolationsmethoden zum konsistenten Datenaustausch verwendet werden müssen, was erwar-

tungsgemäß weniger effizient ist als löserinterne Methoden. Daher wird in dieser Arbeit auch die Performanz beider Kopplungsansätze betrachtet und eine optimale Lastverteilung zwischen den einzelnen physikalischen Gebieten erarbeitet. Es wird an einem Beispiel aus der Industrie, einem 3D Freistrahl mit hoher Reynoldszahl gezeigt, dass mit dem beschriebenen Verfahren die Berechnung komplexer Mehrskalenprobleme in angemessener Zeit durch eine effiziente Nutzung der heutigen Computerressourcen ermöglicht wird.

Abstract

Multi-scale problems like the generation of sound in a flow field and its sound wave propagation in the far field have become increasingly important in the design phase of industrial devices: One example is noise reduction of aircrafts or wind turbines. Although the generation of sound as well as its propagation can both be described by the same governing equations, wave propagation is a linear phenomenon and its equations can be simplified, which results in less computational effort. Additionally, the generation of sound in a flow field occurs at small spatial scales, while its propagation in the far field has to be observed on a large spatial scale. These large differences in scales are particularly challenging for numerical simulations. Resolving the entire domain with the high resolution that is required for the small scales of the flow domain is impossible due to the vast computational demand. In this thesis, we propose a partitioned coupling approach for the efficient simulation of such problems on massively parallel supercomputers. In partitioned coupling the physical domain is split into smaller subdomains, each covering a different physical phenomenon. Their interaction is realized by exchanging data at the joint coupling interface. Subsequently, these subdomains can be solved with numerical methods, resolutions, and equations tailored to the local physical requirements. Even different solvers can be used. However, this approach also holds numerical challenges at the coupling interface: E.g. for a consistent data exchange in case of different spatial resolutions, direct data evaluation, or efficient interpolation methods are necessary. Within this work, two different approaches of partitioned coupling are implemented and compared: A *black-box* and a *white-box* approach. The *black-box* approach is characterized by a flexible choice of numerical solvers which allows for a wide range of different applications. Its generality comes with limited access to information inside each solver and, therefore, with a potential loss of performance. However, a *black-box* approach only acts on point data at the coupling interface and therefore requires external interpolation methods for a consistent coupling in space which is expected to be less efficient than solver-internal data mapping. In contrast, the *white-box* approach is fully integrated within one numerical framework. Accordingly, it can access solver-internal data mapping methods which promises better numerical results. This tight integration allows for the exploitation of knowledge about internal data structures and, therefore, yields performance benefits. On the other hand, it comes with less flexibility. Both strategies will be compared with respect to quality of data mapping at the coupling interface as well as performance on modern supercomputers. In order to achieve the best performance,

the optimal load balancing strategy for a coupled setup is investigated. The benefits of the partitioned coupling approach are demonstrated on an industrial application of a 3D free-stream jet with a high Reynolds number showing that a multi-scale problem can be simulated using today's compute resources.

Contents

Nomenclature

Symbols

$c_p \backslash c_v$	Specific heat
d	Dimension
e	Energy density
$\boldsymbol{F^i}$	Flux function in i-direction
$\boldsymbol{F^{\mu i}}$	Diffusive flux function in i-direction
$\boldsymbol{U}$	State vector
h	Grid/Element size
$\boldsymbol{I}$	Identity matrix
$\mathcal{L}$	Refinement level of octree mesh
lu	Length unit
m	Polynomial degree
n	Number of coupling points
$\mathcal{O}$	Numerical order of a scheme
p	Polynomial order
R	Ideal gas constant
Re	Reynolds number
rhs	Right hand side
T	Temperature
tu	Time unit
v_i	Velocity in direction of i
$\boldsymbol{v}$	Vector of velocities

Greek Symbols

γ	Adiabatic exponent
κ	Thermal conductivity
λ	Bulk viscosity
$\rho\boldsymbol{v}$	Momentum
μ	Dynamic viscosity
Ω	Domain
ψ	Test function
ρ	Density
τ	Viscous stress tensor
Υ	Tessellation of the domain Ω

Other Symbols

$\otimes$	Dyadic product
L_2	Direct projection via numerical quadrature
$nElems$	Total number of elements
$nVars$	Number of variables
∂_i	Partial derivative with respect to i, e.g. time, x-, y-, z-direction
$\nabla\cdot$	Scalar product

Acronyms

ADER	Arbitrary high order using DERivatives
APES	Adaptable Poly-Engineering Simulator
CFL	Courant-Friedrich-Levy
DAA	Direct Aero-acoustics
DG	Discontinuous Galerkin
DNS	Direct Numerical Simulations
DoF	Degrees of freedom
EE	Euler equations
FD	Finite Difference
FE	Finite Elements
FPT	Fast Polynomial Transformation
FV	Finite Volume

HPC	High Performance Computing
LEE	Linearized Euler equations
MpCCI	Mesh-based parallel Code Coupling Interface
MPI	Message Passing Interface
NN	Nearest Neighbor mapping
NP	Nearest Projection mapping
NSE	Navier-Stokes equations
ODE	Ordinary differential equation
PDE	Partial differential equations
preCICE	Precise Code Interaction Coupling Environment
RBF	Radial Basis Bunctions
RK	Runge-Kutta
SFC	Space-filling Curve

1. Introduction

Acoustic noise is all around us in "our daily lives" stemming from a variety of sources that generate sound: Ranging from aircraft jet engines over wind turbines and air-conditioners to the fans of laptops. A prominent example are wind turbines: Their rotor blades move through the air, which leads to turbulent flow and vortices near the blades. The vortices themselves generate sound waves which are subsequently transported through the air and can be perceived by humans and wild life, even over long distances. In the last decades the amount of noise generated by humans has drastically increased and the term "noise pollution" has been coined. Noise pollution usually describes environmental, unwanted noise, such as the sound from traffic, or the occupational noise by industrial machinery. This noise, stemming from various sources, impacts the well-being of people and wildlife [1]. As a recent example, the impact of wind turbines has been studied [2–4]. With increasing awareness of noise pollution and its consequences, the reduction of acoustic sound has become more and more important in the design phase of industrial devices. Here, simulations of fluid dynamic processes have become a very important tool: They provide deeper insights into the sound generation of technical devices and help, for instance, during the design of rotor blades of a wind turbine to optimize their shape with the objective of noise reduction. On a large scale, simulations can also help to optimize the location of wind turbines close to villages with larger buildings where different sound waves and echoes can influence each other.

In general, the simulation of fluid dynamic processes is a very computationally demanding task: The phenomena that can occur in flows, especially noise generation and its propagation, cover a wide range of scales. With the availability of increasing computational resources it becomes feasible to include more and more physical effects or physical scales into a single simulation. The most common computer systems for the parallel execution of simulations are so-called supercomputers. Research and computation on such massively parallel systems are related to the field of High Performance Computing (HPC). The forthcoming exascale era for supercomputers promises immense computational power and allows to simulate a new range of multi-physics and multi-scale problems that

were previously unfeasible.

This thesis establishes an efficient approach for a direct numerical simulation of sound generation and its propagation. To make it feasible on supercomputers, a so-called partitioned coupling approach is used to reduce the computational cost. To understand why coupling is a promising approach and what the challenges are, we start with an explanation of the physics of acoustic noise. Afterwards, partitioned coupling is described and which approaches exist. Once this has been clarified, we address the numerical discretization and how to best approximate each physical phenomenon. This introduction part concludes with a section on HPC, since it is a key aspect of this work.

Physics of acoustic noise The physics of acoustic noise is a fluid dynamic process that comprises the generation of sound and its propagation. Flow at high Reynolds numbers or around physical structures causes small vortices that generate sound. These sound waves are subsequently transported via the fluid over long distances. The flow is governed by small scale structures carrying large amounts of energy, while acoustics is dominated by large wavelengths, but small amounts of energy. Hence, fluid acoustic dynamics is a multi-scale problem where the small scales of the vortices must be resolved **and** the large scales of acoustic waves must be treated appropriately. The sound waves are propagated over large distances, which is often called the acoustic far field. This propagation is a linear phenomenon. Considering the numerical discretization, small grid cells are required to capture the small scales of sound generation. For the spatial discretization of the acoustic far field there are three points to consider: First, large grid cells can be used since large wavelength have to be resolved; second, the numerical scheme should not dampen the solution since this would lead to decreasing wave amplitudes with distance; and third, the numerical scheme should not introduce a phase error.

Sound generation and its propagation can be described by the general compressible flow equations – the Navier-Stokes equations. Solving these full equations and resolving even the smallest eddies is called a direct numerical simulations (DNS). Of course, such simulations are computationally expensive and, when aiming to extend the computational domain up to the large acoustic far field, are infeasible within reasonable time on today's supercomputers. There are two ways to reduce the computational demand of aero-acoustic simulations: Modeling or tailoring governing

equations to a physical problem. A classical approach for modeling are large eddy simulations (LES). Here, instead of resolving all small scales, only large eddies are resolved by a discretization and an additional model is added for the small scales. Tailoring the governing equations to a physical problem is, in particular, beneficial for a multi-scale problem where different physical phenomena occur simultaneously. Depending on the physical problem, it is possible to separate the phenomena and save computational cost by only computing the equations which are actually required. Looking at the physical problems of flow and acoustics separately, the following difference can be stated: Acoustics-generating flow is a compressible, viscid phenomenon which is non-linear. Considering a turbulent flow, the viscous part of the Navier-Stokes equations is crucial for the transition from laminar to turbulent flow and the dissipation due to viscosity drives the energy cascade. The acoustics are described by wave transportation which is a linear problem without viscosity. Hence, using the full Navier-Stokes equations to compute the acoustic wave propagation is not necessary. While the full equations are valid, assuming non-viscosity and linearity, they can be simplified and some terms are even negligible. This decreases the numerical and, thus, the computational demand which is particularly helpful for the large distances of the acoustic far field.

As previously described, both phenomena, acoustic generation and acoustic propagation, occur in spatially separate partitions of the simulation domain. Hence, one approach is to split the overall domain into a flow subdomain and an acoustic subdomain that are coupled via boundary conditions. In these subdomains, different governing equations can be solved and different numerical approaches best suited for the individual physics can be used when exploiting partitioned coupling.

Partitioned coupling Partitioned coupling is based on the idea that an entire computational domain of a multi-physics or multi-scale simulation can be split into subdomains, where in each only single physics need to be considered. To realize their interactions, these subdomains are then connected via a coupling approach.

In general, there are two different kinds of coupling: Volume coupling and surface coupling, where the terms “volume” and “surface” refer to the region where information is exchanged, respectively. In case of volume coupling, the subdomains have to partially or in total overlap which results in a common volume. In this common volume, the coupling information is

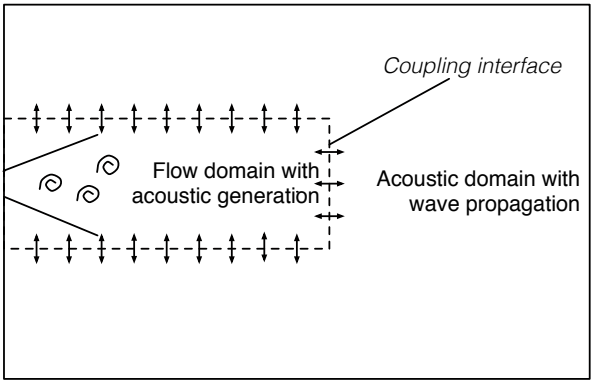


Figure 1.1.: Sketch of a coupled setup for an areo-acoustic jet. The dotted line indicates the coupling interface between the flow domain and the acoustic domain and the arrows depict exchange positions where coupling variables like density, velocity, and pressure are exchanged.

exchanged. Typically, the information of one subdomain is requested by the other subdomains as source terms. In contrast, for surface coupling the subdomains have a joint surface and do not overlap. At this joint coupling surface, the coupling information is exchanged. Thereby, the coupling data is used as boundary condition of the requesting subdomain. Figure 1.1 shows a typical scenario of a coupled fluid-acoustic simulation using surface coupling: We split the entire domain into a smaller flow subdomain and a surrounding acoustic subdomain. With that, the original multi-scale domain is divided into a subdomain with small scales and large scales, respectively. As depicted by the arrows in Figure 1.1, interactions are realized by exchanging the values of the coupling variables at the "coupling interface". In this way, appropriate numerical schemes can be applied in each subdomain. Partitioned coupling opens up the possibility to use different equations, different discretization in space as well as in time. Even coupling across different machines is possible [5]. In this work, we focus on enabling large-scale aero-acoustic simulations by efficient coupling of different equations (e.g. Navier-Stokes equations, Euler equations, Linearized Euler equations) as well as different numerical discretizations on parallel systems.

Another aspect to consider is the direction of coupling: Unidirectional

and bidirectional coupling; in other words one-way or two-way coupling. Unidirectional means that the information is only is requested by one subdomain and provided by the other. Unidirectional coupling is a common method for aero-acoustic simulations, where a flow solver computes acoustic source terms that are fed to the acoustic domain in which the acoustic wave propagation is computed subsequently. Here, no feedback is provided to the flow domain. Bidirectional coupling, in contrast, denotes that coupling information is requested and provided by both subdomains (as illustrated by the two-directional arrows in Figure 1.1). This becomes important when feedback or bilateral interaction is required, e.g. influence of an echo, or for fluid-structure interaction where the deformation of a geometry is influencing the flow.

In order to enhance the quality as well as the performance of multi-scale simulations, numerical schemes and equations that have been tailored to physical phenomena are required. Partitioned coupling allows to combine different numerical schemes and equations in one simulation. Therefore, the coupling approaches should not adversely affect the quality of the coupled simulation. Regarding the example of surface coupling, the coupling data is used as boundary condition of the requesting subdomain and hence "low quality" data influences the solution of this subdomain. Besides quality, also the performance of coupling approaches plays a role: To understand the performance challenges, it is important to point out that both domains are computed in parallel. The requesting subdomain requires coupling data at the coupling interface at synchronization points. If the data is not yet provided by the other subdomain, the requesting subdomain has to wait. To avoid waiting times, the delivering subdomain should provide the data immediately. Here, two points are important: Firstly, a good load balancing between the subdomains, so that they are done with their individual work at the same time during execution, and secondly, the coupling approach must not pose a bottleneck (e.g. in communication or interpolation).

We distinguish two different groups of coupling approaches: *black-box* and *white-box* approaches. Generally speaking, these approaches differ in data that is exposed during simulation and the code that drives the simulation. The term *black-box* describes coupling approaches that consider subdomains as black boxes without information about their inside. Typically, *black-box* coupling tools are flexible with respect to the applied solvers. If no application programming interfaces is readily available, it is simple to implement an adapter. This flexibility implies that *black-box* coupling tools exclusively work on input and output data, independent of

the applied numerical discretization. In contrast, the quality and the performance of a coupled simulation can be improved for dedicated cases when insights of the subdomains can be used, i.e. when opening the black boxes. We call this *white-box* approaches. For example, a *white-box* approach can be designed in such a way that it uses data from the numerical scheme of the subdomains. *White-box* approach further implies that the coupling approach is designed in such a way that it fits to the desired solvers in the subdomains and to the desired application. That means, solver and coupling tool are designed together to achieve the best framework for the desired application. This involves more implementation effort and code design than using a *black-box* approach, but is expected to gain efficiency and accuracy.

To look more closely at coupling challenges, let us consider two subdomains **A** and **B**: **B** requires data at specific points on the coupling interface. In order to properly realize an interaction, the second subdomain **A** has to provide data at these requested points. However, the points where subdomain **A** is able to provide data depends on its numerical discretization. When these sets of points – requesting and providing – coincide, we use the term *matching* coupling interface. While the data can be exchanged without additional work in this case, these points do not generally coincide at the coupling interface. When **A** provides data at a different set of points, we talk about *non-matching* coupling interfaces. Here, additional work for data mapping is required. There are two different options: evaluation or interpolation. With the former option, the numerical solver is requested to generate data at the desired points with additional effort. For example, a higher-order method based on polynomials, e.g. the Finite Element scheme or the Discontinuous Galerkin method, can evaluate their polynomials at any point. In contrast, interpolation methods take the provided data and interpolate the data at the requested points. Such external methods can be costly and might reduce the quality of the solution. *Black-box* approaches typically act exclusively on geometric data of the subdomain without internal information which forces them to use external interpolation methods, while *white-box* approaches can exploit all insights of the solvers.

Comparing *black-box* and *white-box* approaches, it is obvious that *black-box* approaches are more flexible in terms of connecting them to new solvers. Even commercial solvers might have application programming interface (API) to a *black-box* coupling tool. But *black-box* coupling tools require coupling schemes that can deal with arbitrary point data. Using external interpolation data for *non-matching* coupling interfaces can reduce quality

and performance. In contrast, *white-box* approaches achieve higher quality of the solution and can be more performant, since they can exploit internal information. *White-box* approaches, however, require more implementation effort and are limited to the specific scheme that they have been established for. Typically, *black-box* approaches require linking individual solvers to a coupling tool which can impair their handling on supercomputers. The *white-box* approach usually results in a single application to handle. Whether to chose a *black-box* or a *white-box* approach depends on the explicit multi-physical or multi-scale problem and the general conditions.

Numerical discretization After this brief introduction to coupling, we will focus on how to compute individual subdomains. For aero-acoustic problems, the physics suggests to split the domain into a small flow and a large acoustic subdomain.
After the generation of acoustic waves in the flow field, the waves enter the acoustic field and travel over long distances. In order to simulate the wave transportation as correctly as possible, the applied numerical scheme has to minimize:

- The dissipation error, that leads to incorrect amplitudes of the waves and would dampen them and
- the dispersion error, that results in phase errors of the waves.

A high-order scheme yields low numerical dissipation and dispersion errors [6]. It is also possible to use a lower-order scheme when refining the computational grid to counter the dissipation error. But a lower-order scheme will still yield a higher dispersion error. Hence, the low dispersion error of a high-order scheme is better suited for the computation of acoustic wave propagation over long distances. Additionally, a high-order scheme shows high convergence rates in case of smooth solutions. This means that such schemes provide high accuracy with only a few degrees of freedom (*DoF*), which equates to lower computational demand. Furthermore, fewer *DoF* also require smaller amounts of memory, which is essential since memory is an expensive resource and can limit scalability. Specifically for the simulation of a large acoustic far field, resource optimization is necessary. A promising candidate is the high-order Discountinous Galerkin (DG) method: It is a numerical method to solve partial differential equations and is applicable to a broad set of problems. It is based on a polynomial representation within each computational element and a flux calculation between elements. Because this method only requires data from direct neighbors, it is also well suited for parallel computations.

For the flow subdomain, a numerical scheme that is capable of handling high Reynolds numbers and resolving the small scales of vortices is required. Typically, a finite volume scheme with a fine computational grid is used. The DG scheme, however, is also suitable for this domain since the order of the polynomial function can be decreased and a lower-order scheme is constructed. Gassner presented the DG method for unsteady compressible Navier-Stokes equations [7]. Similar to a Finite Volume scheme, the DG scheme is capable of handling shocks. Zudrop has presented a DG approach which recovers high order solutions by post-processing even in the presence of discontinuities [8]. Additionally, the locality of the DG scheme makes it attractive for the use-cases discussed in this thesis.

High performance computing Nowadays, most large scale computing systems are massively parallel. Thus, approaches to simulate aero-acoustic problems and acoustic noise propagation should operate efficiently on such architectures. To this end, a framework that provides scalable solvers as well as pre- and post-processing tools is required. Here, the term "end-to-end parallel" was established over the last years for a tool chain without serial bottlenecks. Concerning exascale computing, we need software that is capable of running efficiently on a large number of CPUs. Furthermore, such large-scale simulations generate a large amount of data that cannot be processed serially. Thus, it is crucial that each tool of a tool chain is highly parallel. For coupled problems, this does not only hold for the individual solvers, but also for the coupling tools itself.

One especially important point is load imbalances. Load imbalances occur when computational work is not equally distributed over all processing units so that some units do more work than others. This usually results in waiting time for the processing units that are doing less work. At a synchronization point, they have to wait until the processing units that are doing more work have finished. Considering a coupling approach, the solvers of individual subdomains have to compute different equations with different numerical discretizations. Because of this, the workload of the solvers is different which leads to imbalances. The goal of load balancing approaches is to equally distribute the work over all processing units to avoid waiting times and increase the overall performance.

1.1. State of the art

For simulating aero-acoustic problems, including noise generation and its propagation, several approaches are known. Generally speaking, for each physical phenomenon there are direct simulation approaches as well as modeling techniques. That is, computationally demanding direct numerical simulations are available even for acoustics-generating flow, which compute the sound together with its fluid dynamic source field. One could also use a turbulence model in which only the dynamically important flow scales are resolved and the effects of smaller scales are modeled. For the acoustic far field, one can use acoustic analogies based on source terms in the flow domain, approximate the solution to the homogeneous wave equation based one the Kirchhoff integral theorem [9], or direct simulation by solving simplified equations such as the Linearized Euler equations. Aiming for a direct simulation of both, flow dynamics and acoustic propagation, results in vast computational cost. One idea to circumvent this is to only compute the equations that are strictly necessary to describe physical phenomena. As stated before, noise generation and its propagation occur in spatially separate partitions of the simulation domain. Hence, partitioned coupling helps to reduce the computational demand by a) only solving the equations required to describe the phenomena in a subdomain and b) using the best-suited numerical scheme for the respective phenomena. Wang et al provide a recent overview of the computational prediction of flow-generated sound including direct simulation of flow and acoustics, acoustic analogies, and turbulent flow modeling [10].

A prominent approach for aero-acoustic simulation are **acoustic analogies**. A classical approach is Lighthill's analogy [11]: It is based on the decomposition of variables into acoustic and non-acoustic components. Lighthill starts from the compressible Navier-Stokes equations and derives the inhomogeneous acoustic wave equation in which source terms describe the acoustic sources. With the introduction of assumptions these source terms become independent of the acoustic variables and linearized equations for the propagation of the acoustic waves in a homogeneous, resting fluid can be derived. Another analogy extending Lighthill's approach is the Ffowcs Williams–Hawkings analogy [12]. All analogies represent a unidirectional coupling, where the acoustic wave propagation never influences the flow computation. Acoustic analogies are also used to determine acoustic sources from the incompressible Navier-Stokes equations and feed these sources into a solver for wave propagation [13]. The authors of [10] summarize that remarkable care is required in evaluating the source terms

to make accurate computational predictions of the far-field sound. Also, other studies have critically discussed such acoustic analogies: Fedorchenko presents a critical analysis of main theoretical approaches for sound generation for the case of inviscid gas flow [14]. He reveals that the definition of aero-dynamic sound sources, as described by Lighthill's acoustic analogy, contains evident defects and models cannot be adapted to be physically correct. Tam presents examples in [15] where, in each case, the acoustic analogy theory identifies the wrong acoustic source. This study points out that quadrupole source terms are not unique and change depending on the choice of variables to characterize acoustic waves. Acoustic analogies have been an important tool to reduce computational demand and to make aero-acoustic simulations possible. They are, however, only applicable for dedicated applications where their assumptions hold. In several cases [15], for example, an acoustic field influencing a flow field, this is not guaranteed.

With the increase of computational power of supercomputing facilities, direct aero-acoustics (DAA) simulations of the far field have become possible. Instead of using analogies, the governing equations are numerically solved, as the name suggests. Hence, DAA can replaced acoustic analogies. Besides the computation of the far field, the acoustics-generating flow has to be computed: Direct numerical simulation (DNS) are one approach. Here, the full governing equations are numerically solved without applying additional models. For acoustics-generating flow, the compressible viscous Navier-Stokes equations are used. Without a model, the whole range of spatial and temporal scales of the flow must be resolved. For high Reynolds numbers and turbulent flow, these scales are very small: The smallest dissipative scales are called Kolmogorov microscales and depend on the kinematic viscosity and the rate of kinetic energy dissipation [16]. Increasing Reynolds numbers result in smaller scales which require a higher resolution and, hence, computational demand. Therefore, DNS of turbulent flow might be infeasible for some applications due to computational restrictions. In such cases, turbulent flow models can be applied, which use mathematical model to predict the effects of turbulence: Two popular streams are Large-eddy simulations (LES) and RANS (Reynolds-averaged Navier–Stokes)-based models [16]. Large-eddy simulation are based on the idea that the large eddies of a flow are dependent on the geometry while the smaller scales are universal. This allows for solving the large eddies in a simulation explicitly and implicitly accounting for the small eddies by using a subgrid-scale model. The first published subgrid-scale model is the Smagorinsky–Lilly model developed by Smagorinsky [16]. An approach connected to LES is the Variational Multiscale approach

which is a technique for a priori separation of scales and an essential mathematical framework for the construction of the subgrid-scale model [17]. The second group of turbulence model is based on the Reynolds-averaged Navier–Stokes equations (RANS) which are the time-averaged equations of motion for fluid flow. The idea is that an instantaneous quantity can be decomposed into its time-averaged and fluctuating quantities. To close the averaged equations, the non-linear Reynolds stress term requires additional modeling [16]. However, turbulence models are based on assumptions which have to hold for the respective applications. This is particularly challenging for the occurrence of multiple phenomena.

For aero-acoustic applications, by using partitioned coupling approach, it is possible to enable DNS for the acoustics-generating turbulent flow (without modeling assumptions) and DAA for the far field propagation. When considering this, however, it is important to use bidirectional coupling to fully realize the interaction. The combination of partitioned coupling and best-suited numerical schemes per phenomena makes the computational demand manageable: In this work we split the entire domain in a flow and a acoustic subdomain and leverage a lower-order DG method for the flow domain and a higher-order DG scheme in the acoustic domain. The next section outlines the state of the art of numerical discretization and emphasizes the DG method, particularly, for the acoustic far field.

Numerical discretization Multiple approaches for spatial discretization with different goals and advantages have been implemented. To solve the governing equations of flow (Navier-Stokes equations, Euler equations) in space, various numerical methods have been established. The three most widely used ones are Finite Difference (FD), Finite Volume (FV), and Finite Element (FE) [18]. A rather new method that became popular over the last decades is the Discontinuous Galerkin (DG) method [19–21], which combines advantages of the FV and the FE method. It uses higher-order representations within elements from the FE method as well as the allowance of discontinuity between elements where it uses the flux calculation of the FV method. Hence, DG maintains a good approximation quality which is only dependent on neighboring elements. In contrast to the FE method, there is no need to solve a global linear equation system in DG. Thus, with its locality and the lack of a global equation system, DG offers great potential for massively parallel programming and high performance computing. A large number of different variants of the DG method are known which typically differ only in the choice of basis functions,

e.g. nodal scheme [6] or modal scheme [8]. A comprehensive collection of mathematical details of the DG method is given in [22]. Hartmann presents a method to deal with the viscous parts of the Navier-Stokes equations [23]. All of the aforementioned discretization techniques can provide values at arbitrary points in coupling: The FD method computes data at the nodes of the discretization mesh and works on point values; the FV method computes the integral mean values and, therefore, the value is constant within a volume and thus assessable at every point; the DG method exploits polynomial representations within the elements which can be evaluated to obtain values at any point.

Another approach to solve PDEs are spectral methods [6, 24]. These methods are closely related to FE and DG, since they are based on similar ideas like basis functions and polynomial representations. Classical spectral methods are global approaches that are operating in frequency space. Therefore, the solution is based on information from the entire spatial domain. Even with this intrinsic lack of locality, parallelized implementations have been proposed [24]. Spectral methods converge exponentially and have excellent error properties when the solution is smooth, but underperform when handling discontinuities like shocks [25]. However, several aspects of spectral methods can be leveraged in the DG methods, for instance, spectral filters [8].

After this introduction into numerical discretization in space, we will focus on the discretization in time. A classical approach for solving partial differential equations (PDEs) is the method of lines where time and space discretization are treated separately of each other [26]. By first discretizing the spatial variables, a PDE is transformed into an ordinary differential equation (ODE) in time. To evolve an ODE in time, an ordinary differential equation integrator is required. Multiple ODE solvers are presented in [27]. Here, a general classification can be done: *implicit* or *explicit* methods. Since *implicit* schemes are based on previous and current time information, a linear equation system needs to be solved in each iteration. While this might be more involving, typically, *implicit* schemes are unconditionally stable which means that large timesteps are allowed. In contrast, *explicit* schemes do not involve any linear equation system since they are only based on the variables of the previous timestep. They are, however, only conditionally stable and have a restriction on the timestep size, the so-called CFL condition [28]. The stability criterion as well as the timestep size depends on the combination of time and space discretization. Classical *explicit* time integration methods are offered by

the family of explicit Runge-Kutta methods (RK) [29]. These methods achieve high-order time integration by using multiple substeps (sub-stages) within each timestep [28]. The classical fourth-order Runge-Kutta method uses four sub-stages to advance to the next timestep. Using the explicit Runge-Kutta method in combination with DG was introduced by Cockburn and Shu [19–21, 30]. Zudrop presents the implementation of the RK integration for DG in the numerical framework *APES* [8]. In general, it should be stated that each sub-stage requires the evaluation of a spatial discretization. However, increasing the order of the time integration by using corresponding numbers of sub-stages only holds up to a certain limit, the so-called Butcher barrier [31]. This barrier states that for orders higher than four, there are no explicit RK methods with the number of sub-stages corresponding to the convergence order, i.e. disproportionate more sub-stages are necessary to compute the timestep for an desired order. Hence, up to fourth-order, the explicit RK method is an efficient and straightforward time integration method. Another popular class of time integration methods are the so-called strong-stability-preserving (SSP) time-stepping schemes. These high-order time discretization methods preserve the strong stability properties of first-order Euler timestepping and have proven to be useful in solving hyperbolic partial differential equations [32].

The DG solver *Ateles*[1] will be used and extended in this work. A second-order SSP RK method as well as a fourth-order RK method in time is available in *Ateles*.

Partitioned coupling As DNS of acoustic-generating flow, particularly combined with a DAA of the far field, are computational demanding, the idea of partitioned coupling for such problems is not new: Several authors have proposed to use surface coupling for aero-acoustic applications to couple a non-linear compressible flow domain to an acoustic far field [33–35]. In literature, the term *domain decomposition* is used as an alternative to *partitioned coupling*. For example Schwarzkopff couples via so-called ghost elements which then provide information for the numerical methods [36]. This allows for coupling different numerical schemes in space as well as in time. Based on this, Utzmann established a coupling mechanism entirely based on the exchange of data in the Gauss integration points of the ghost elements [37], which was then parallelized and optimized for large-scale simulation by Klimach [5]. These are examples for *white-box* coupling, where the coupling approach is implemented in one integrated tool.

[1] `https://www.apes-suite.org/pages/ateles`

On the other hand, *black-box* coupling tools are developed to couple individual *black-box* solvers. They are often used for multi-physics simulations like fluid-structure interaction. A widely used tool is *MpCCI* [38]. *MpCCI*[2] abbreviates "Mesh-based parallel Code Coupling Interface" and is developed at the Frauenhofer Institue for Algorithms and Scientific Computation (SCAI). It can be used for fluid-structure interaction, coupling climate models, fluid-electro combinations, among others. The *MpCCI* software is offered as a commercial product. Another open-source coupling tool is the coupling library *preCICE* (Precise Code Interaction Coupling Environment) [39]. Details that go beyond the scope of [39] are described in [40] and [41]. *preCICE* is constructed as library which allows a minimally invasive integration into existing solvers. In [42] and [43], development and achievements of *preCICE* working on distributed data are presented. The idea is to establish a fully parallel point-to-point concept for the communication of coupling data. As a *black-box* approach, *preCICE* works purely on geometric data at the coupling interface. For *non-matching* interfaces, *preCICE* provides two standard interpolation methods: Projection-based mapping and radial basis function interpolation. Additionally, for implicit coupling, which is not part of this work but a key feature of *preCICE*, efficient solvers for fixed-point equations derived from coupling conditions are implemented in *preCICE*. Flexibility is the key benefit of using a coupling tool like *preCICE*. The application programming interface (API) is concise and enables easy coupling of individual solvers. The advantages, however, are clouded by disadvantages intrinsic to *black-box* approaches: A performance decrease is accepted in favour of generality. Additionally, the quality of the overall simulation can diminish for *non-matching* coupling interfaces when using external interpolation methods with a lower order than in the subdomains. Using a high-order interpolation can be disproportionately expensive, particularly in contrast to solver-internal methods. Another aspect, compared to single application tools is the user experience on massively parallel architectures. For coupling with *preCICE*, different executables of the solvers involved in coupling must be linked with *preCICE* and all executables must be started individually, but simultaneously. On today's supercomputers, setting up such a coupled job is more involved than a single application: When starting several executables, the correct binding of MPI ranks to the corresponding executable needs to be taken into account to avoid running multiple ranks on the same CPU. Hence, porting software, establishing

[2] `http://www.mpcci.de/en/mpcci-software/mpcci-couplingenvironment.html`

the correct binding of MPI ranks, and compiling the job script on a supercomputer is more challenging compared to running a single application. Nevertheless, *preCICE* was successfully applied in the ExaFSA ("Exascale Simulation of Fluid-Structure-Acoustics Interactions")project, which is part of the German priority program SPPEXA[3] and has tackled challenges of the simulations of fluid-structure-acoustics interactions. In this context, results using the finite volume solver OpenFOAM[4] (for the flow) coupled to the DG solver *Ateles* (for the acoustic far field), have been published in [44, 45]. Further results using the finite volume solver FASTEST[5] coupled with the DG solver *Ateles* have been reported in [46].

High performance computing With the increased computational power of today's supercomputer, researchers invest a lot of effort to efficiently use them. Here, we focus on the state of the art in high performance computing directly related to this thesis. The aforementioned SPPEXA programme abbreviates "Software for Exascale Computing" and indicates the importance of the forthcoming exascale era for supercomputers. It investigates the new possibilities of exascale compute systems and their challenges [47, 48]. The project "ExaFSA" presented its work on supercomputers in Germany in [49].

As mentioned before, a parallel solver on its own is not enough to solve large-scale problems. An "end-to-end parallel" simulation tool chain without bottlenecks and memory issues, working directly on massively parallel systems, is required. One open-source simulation toolbox is *DUNE* ("Distributed and Unified Numerics Environment") [50], which was also part of the SPPEXA programme. It is a modular software toolbox for parallel solving of PDEs and is based on the separation of data structures and algorithms by abstract interfaces. Hence, it provides a generic mesh interface, so that one numerical scheme can efficiently work on different mesh implementations [51]. *DUNE* supports the implementation of methods like Finite Elements (FE), Finite Volumes (FV), and also Finite Differences (FD). Recently, a DG scheme with *DUNE* was published [52]. Another promising "end-to-end parallel" simulation tool chain is *APES* ("Adaptable Poly-Engineering Simulator"), developed by the *STS* (Simulation Techniques and Scientific Computing) group at the University of Siegen. *APES* provides scalable solvers, Lattice-Boltzmann method and DG, as well as

[3] `http://www.sppexa.de`
[4] `http://www.openfoam.org`
[5] `https://www.hkhlr.de/fastest`

pre- and post-processing tools. It is based on the common "Tree-based Elemental Mesh library" *TreElM* that provides the functionality to act on distributed parallel octree meshes. Klimach presents the advantages of octree based data structure for mesh generation in flow applications [5]. In *TreElM*, a space filling curve is used for spatial ordering of the mesh information and implements a fast connectivity search. With the connectivity already in the mesh generation, the neighbor information required for a numerical solver can be found a priori. Hence, *APES* provides all building blocks for an *white-box* coupling approach, like the octree representation of the mesh, a distributed neighborhood search, communication routines, parallel IO, and time measurements. The high order DG solver *Ateles*, which is part of *APES*, shows great performance on today's supercomputers [8].

For the aspect of balancing loads in HPC, a wide variety of partitioning and load balancing algorithms have been developed. Teresco et al review several partitioning algorithms, along with their strengths and weaknesses for various PDE applications and present that the effectiveness of partitioning and load balancing algorithms depend on the characteristics of the application [53]. For load balancing algorithms based on weights, which mirror the actual workload, the appropriate measurement of these weights has to be determined for each application individually. A common approach are re-partitioning methods which use weights to indicate the workload and re-distribute the work accordingly. For numerical algorithms based on grid cells, the weights are typically measured per grid cell and then the grid cells are re-distributed accordingly. A popular option for the partitioning of meshes is provided by the *ParMETIS* library [54]. *ParMETIS* ("Parallel Graph Partitioning and Fill-reducing Matrix Ordering")[6] is an MPI-based parallel library and is developed at the Department of Computer Science & Engineering at the University of Minnesota. It uses a graph-based algorithm for the following functions: Graph partitioning, mesh partitioning, graph repartitioning, partition refinement, and matrix reordering. Harlacher et al argue that with an increasing number of processes, graph-based partitioning algorithms seem to reach their scalability limits. In particular, one scalability issue arises due to the required memory [55]. An alternative to graph-based partitioning is offered by partitioning methods that are based on space-filling curves. An example of such a method is *SPartA* [55] which is available in *Ateles*. *SPartA* is based on space-filling curves and individual weights which mirror the actual load

[6] `http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview`

per element. Using such weights and calculating a priori the additional cost for re-partitioning, *SPartA* is an efficient method that is tailored to the requirements of a static mesh.

1.2. Aim of this work

This work deals with large-scale aero-acoustic simulations including fluid-acoustic interaction without using modeling techniques, but DNS and DAA. To make such costly simulations feasible on today's supercomputers, we propose to use a partitioned coupling approach that reduces the numerical cost in applicable regions. Bidirectional coupling is applied for full interaction between subdomains. When using a coupling approach to enable simulations for a specific scenario, like fluid-acoustic interaction, it has to be verified that the coupling interface does not influence the solution. Once we have shown this, we can straightforwardly extend the idea of partitioned coupling to an n-field coupling with n subdomains. For an aero-acoustic problem, this means that the entire domain can be separated into three individual subdomains: A viscous flow domain with vortices, an inviscous flow and a linear acoustic far field. Hence, three different equations, three different numerical schemes with different grid sizes and numerical orders can be chosen. In this thesis, a 3D subsonic free-stream jet with high Reynolds number is investigated by employing such a 3-field coupling.

We will establish a *black-box* and a *white-box* approach within the simulation framework *APES* and provide a direct comparison in terms of numerical as well as performance results. The difference in terms of quality of the solution will be investigated with an academic testcase with analytical solution, while the performance difference will be analyzed with the 3D subsonic free-stream jet.

However, the usage of coupling itself does not necessarily promise a feasible simulation on massively parallel systems: Load balancing between subdomains is essential to avoid idle processors and to use computational resources efficiently. As introduced before, a coupled simulation has a higher risk for imbalances due to individual workloads. Hence, This work investigates the origin of load imbalances and evaluates load balancing strategies for 3-field coupling. Typically, load balancing strategies are based on measurements of workload, so-called weights, which are used for re-distributing the workload. This means that the quality of a load balancing strategy is limited by the quality of these measurements. How to measure these weights appropriately is a fundamental issue for individual

applications. Here, a *white-box* approach is advantageous since all insights of coupling as well as the solver characteristics can be leveraged for a correct measurement of workload.

In a partitioned coupling approach, we compute the subdomains individually. For the acoustics-generating flow domain, we choose the high-order DG solver *Ateles* that allows to set the numerical order arbitrarily and shows great performance on today's supercomputer [8]. For the far field, we use *Ateles* as well after extending it for the computation of acoustic wave propagation. Hence, we present the coupling of a DG scheme with a DG scheme where each has a tailored grid as well as numerical order. As already explained, the high-order DG method is able to provide evaluation methods which can be used by a *white-box* approach in case of non-matching coupling interfaces. As indicated, we consider a *white-box* approach to be the best option regarding the quality and performance of an aero-acoustic simulation using a high-order DG in the far field. Thus, as part of this work, the *white-box* coupling approach *APESmate* based on *APES* was developed. During development, the main focus was on the integration of the DG solver *Ateles*. For the solution of a coupled simulation it is not only essential how data is communicated, but foremost which data is communicated. To exchange the correct data, the polynomial evaluation at arbitrary points has been implemented in *Ateles* where a scalable implementation was considered as crucial (Appendix A). The points for this evaluation have to be identified and required information like process unit and variable type has to be exchanged. Gathering of all static information as well as mapping of process units should be done during initialization while the evaluation is processed during simulation. We chose the simulation tool chain *APES* with its open source DG solver *Ateles* and its open source library *TreElM*, which provides all building blocks to implement a parallel coupling approach. The availability of sources is a necessity for a good load balancing in large-scale scenarios, where a *white-box* approach can use all the insights of schemes as well as the code for an accurate measurement of weights.

To compare the *white-box* approach, a *black-box* approach using the *black-box* coupling library *preCICE* is also developed with *Ateles*. Once, both coupling approaches, *preCICE* with *Ateles* and *APESmate* with *Ateles*, have been established and validated, the 3D free-stream jet with 3-field coupling can be investigated. To observe the wave propagation in the entire far field, the simulation has to run sufficiently long. To find the most efficient approach, load balancing plays a significant role. The

weights for the load balancing strategy are directly connected to a specific scenario and, hence, investigations with the 3D jet have to be conducted beforehand: This is done for *APESmate* and *preCICE*, respectively, to determine the appropriate load balancing strategy. We will take a closer look at the measured weights for the *white-box* approach *APESmate* that uses polynomial evaluation of the solver for the data mapping. For the *black-box* tool *preCICE* it is not possible to attribute the coupling workload per element, since the coupling work is done per processing unit. With the appropriate load balancing strategy for the *black-box* approach *preCICE* and the *white-box* approach *APESmate*, the performance is compared. The final step is the large-scale simulation of the 3D free-stream jet with an efficient coupling approach. Now, the numerical results can be investigated, where we will take a closer look at the coupling interfaces. Here, we first analyze a point in simulation time for which the coupling interfaces between three different governing equations (viscous flow, inviscid flow, acoustic) are chosen appropriately. Consciously progressing the simulation of the free-stream jet further in time, the choice of coupling interfaces becomes non-optimal and viscous flow phenomena/properties are reaching into the inviscid flow subdomain. Here, we can investigate the influence of imperfect coupling interface onto the sound generation and the acoustic far field.

In the following, instead of using the term “black-box approach”, we use the term “multi-solver approach” since when considering the solvers, and even the coupling tool, as black boxes, is results in a coupling of multiple solver. The term “white-box approach”, in contrast, will be replaces by “integrated approach”, since in case of “opening” the black boxes and integrated insights to obtain a better solution anyway, we are aiming for a fully integrated implementation in order to achieve the best performance.

1.3. Outline

The upcoming chapter presents the governing equations of the single physics and their discretization. This is followed by a chapter about coupling theory that discusses the coupling tasks, static load balancing and the numerical DG method as well as the governing equations in the context of coupling. The theory part is followed by a chapter on the numerical framework. Here, the individual components of the coupled simulation, namely *APES* (Section 4.1), *Ateles* (Section 4.2) and both coupling approaches, *preCICE* (Section 4.3) as well as *APESmate* (Section 4.4), are presented. The latter

two sections do not only examine how the coupling tasks are implemented but also present a performance evaluation. Chapter 5 exhibits and discusses the coupling results. First, both presented approaches are evaluated with an academic testcase (Section 5.1). Subsequently, the 3-field coupling of a 3D free-stream jet is investigated in Section 5.2. This section starts with a look at the numerical results at a point in simulation time where the choice of coupling interfaces are valid, followed by a detailed description of this setup. Since load balancing of coupled simulation is crucial for large scale simulations, this section also includes the investigation of static load balancing strategies on the massively parallel system SuperMuc Phase 1 for both approaches. With a strategy for well-balanced coupled simulations, we perform the 3D free-stream jet for a large simulation time resulting in a scenario where the defined coupling interface are not optimal anymore. Hence, we conclude this section with an investigation of an imperfect choice of coupling interfaces and the influence on the acoustic far field. After the numerical results with the focus on the data mapping in space, we briefly investigate time-consistent coupling and presents preliminary results to tackle this challenge. This work is concluded with a summary and short outlook in Chapter 7.

2. Governing equations of fluid dynamics and their discretization

In this chapter, the governing equations of fluid dynamics are described. The governing equations are the mathematical models that describe real-world phenomena and translate them into a language that can be used by numerical methods.
After recapitulating the leading partial differential equations in Section 2.1, their discretization is presented in Section 2.2. In this work, we use the explicit Runge-Kutta (RK) method (Section 2.2.2) for evolving in time, whereas for the spatial discretization a higher-order Discontinuous Galerkin (DG) method (Section 2.2.1)is employed.

2.1. Governing equations of fluid dynamics

Fluid dynamics are characterized by partial differential equations (PDEs). These partial differential equations for flow dynamics are typically based on conservation laws which concern the conservation of mass (depicted by ρ), momentum (specified by $\rho\,\boldsymbol{v}$) and energy (referred as e). For ease of use, conservation laws are stated in a compact notation, defining a vector $\boldsymbol{U} = (\rho, \rho v_x, \rho v_y, \rho v_z, e)$ of the conserved variables and the flux functions $\boldsymbol{F^x(U)}$, $\boldsymbol{F^y(U)}$, $\boldsymbol{F^z(U)}$ in x-, y-, and z-direction respectively, which depend on the vector of conserved variables $\boldsymbol{U}$. The equation for that notation reads as

$$\partial_t \boldsymbol{U} + \partial_x \boldsymbol{F^x(U)} + \partial_y \boldsymbol{F^y(U)} + \partial_z \boldsymbol{F^z(U)} = rhs. \tag{2.1}$$

The expression $\partial_{t,x,y,z}$ denotes the partial derivative with respect to t, x, y, z respectively. In the following, rhs marks right-hand side $rhs_{t,x,y,z}$. Using $\nabla\cdot$ to express the scalar product with the gradients and combining the individual flux functions to the flux $\boldsymbol{F}$ consisting of $\boldsymbol{F^x}$, $\boldsymbol{F^y}$, $\boldsymbol{F^z}$, the short notation

$$\partial_t \boldsymbol{U} + \nabla \cdot \boldsymbol{F}(\boldsymbol{U}) = rhs \tag{2.2}$$

can be obtained. In this work, the compressible Navier-Stokes equations (NSE) are used for the general flow, the Euler equations (EE) as simplifica-

tion for inviscid flow, and the further simplified Lineraized Euler equations (LEE) are used for acoustic far field propagation.

2.1.1. Navier-Stokes equations

In general, compressible fluids are described by the compressible Navier-Stokes equations. The first conservation law is the conservation of mass ρ and it is described as

$$\partial_t \rho + \nabla \cdot (\rho \boldsymbol{v}) = 0 \tag{2.3}$$

where ρ defines the density, $\boldsymbol{v}$ is the velocity vector and no sources or sinks of mass are applied. The second conservation law describes the balance of momentum

$$\partial_t (\rho \boldsymbol{v}) + \nabla \cdot (\rho \boldsymbol{v} \otimes \boldsymbol{v}) = -\nabla p + \nabla \cdot \boldsymbol{\tau}, \tag{2.4}$$

where the pressure is denoted by p and the viscous stress tensor by $\boldsymbol{\tau}$. $\otimes$ is the dyadic product of two vectors. Any external source terms/forces are omitted for brevity.
Energy is the third quantity to be conserved and the corresponding equation is

$$\partial_t e + \nabla \cdot ((e + p)\boldsymbol{v}) = \nabla \cdot (\boldsymbol{\tau}\boldsymbol{v} + \kappa \nabla T), \tag{2.5}$$

e is energy density which is the sum of the kinetic and internal energy of the fluid, κ denotes the thermal conductivity of the fluid and T the temperature.
For a Newtonian fluid, the stress tensor $\boldsymbol{\tau}$ can be formulated as

$$\boldsymbol{\tau} = \mu(\nabla \otimes \boldsymbol{v} + (\nabla \otimes \boldsymbol{v})^T) - (\lambda \nabla \cdot \boldsymbol{v})\, \boldsymbol{I}, \tag{2.6}$$

where μ is the dynamic viscosity and λ the bulk viscosity of the fluid. In an ideal gas, the assumption $\lambda = 2\mu/3$ is valid [56]. $\boldsymbol{I}$ denotes the identity matrix.

To fully describe the system we consider the equation of state for ideal gas

$$p = \rho R T = (\gamma - 1)\left(e - \frac{\rho \boldsymbol{v} \cdot \boldsymbol{v}}{2}\right) \tag{2.7}$$

where R is the ideal gas constant and γ the adiabatic exponent $\gamma = c_p/c_v$ with the specific heats c_p and c_v dependent on the fluid. This equation yields a relation between pressure p and energy density e.

Rewriting the Navier-Stokes equations (2.3)-(2.5) in the compact notation

defined in (2.1), the vector of conserved variables is

$$\boldsymbol{U} = \begin{pmatrix} \rho \\ \rho v_x \\ \rho v_y \\ \rho v_z \\ e \end{pmatrix} \tag{2.8}$$

where the velocity $\boldsymbol{v}$ is split into the entries for each dimension. The flux functions $\boldsymbol{F^x}$, $\boldsymbol{F^y}$, $\boldsymbol{F^z}$ are defined as following

$$\boldsymbol{F^x}(\boldsymbol{U}) = \begin{pmatrix} \rho v_x \\ \rho v_x^2 + p \\ \rho v_x v_y \\ \rho v_x v_z \\ v_x(e+p) \end{pmatrix}, \boldsymbol{F^y}(\boldsymbol{U}) = \begin{pmatrix} \rho v_y \\ \rho v_y v_x \\ \rho v_y^2 + p \\ \rho v_y v_z \\ v_y(e+p) \end{pmatrix}, \boldsymbol{F^z}(\boldsymbol{U}) = \begin{pmatrix} \rho v_z \\ \rho v_z v_x \\ \rho v_z v_y \\ \rho v_z^2 + p \\ v_z(e+p) \end{pmatrix}. \tag{2.9}$$

For the Navier-Stokes equations, the right-hand side of Equation (2.1) consists of the viscous part. Applying the expression for the stress tensor (2.6), the diffusive fluxes of the right-hand side $\boldsymbol{F^{\mu x}}$, $\boldsymbol{F^{\mu y}}$, $\boldsymbol{F^{\mu z}}$ can be rewritten into vectorial form like

$$\boldsymbol{F^{\mu x}}(\boldsymbol{U}, \nabla \boldsymbol{U}) = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ (\boldsymbol{\tau} \cdot \boldsymbol{v})_x + \kappa \, \partial_x T \end{pmatrix}, \boldsymbol{F^{\mu y}}(\boldsymbol{U}, \nabla \boldsymbol{U}) = \begin{pmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ (\boldsymbol{\tau} \cdot \boldsymbol{v})_y + \kappa \, \partial_y T \end{pmatrix},$$

$$\boldsymbol{F^{\mu z}}(\boldsymbol{U}, \nabla \boldsymbol{U}) = \begin{pmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ (\boldsymbol{\tau} \cdot \boldsymbol{v})_z + \kappa \, \partial_z T \end{pmatrix}. \tag{2.10}$$

Inserting these vectors into the compact notation, the Navier-Stokes equations can be stated as

$$\begin{aligned} \partial_t \, \boldsymbol{U} + \partial_x \, \boldsymbol{F^x}(\boldsymbol{U}) + \partial_y \, \boldsymbol{F^y}(\boldsymbol{U}) + \partial_z \, \boldsymbol{F^z}(\boldsymbol{U}) = \\ \partial_x \, \boldsymbol{F^{\mu x}}(\boldsymbol{U}, \nabla \boldsymbol{U}) + \partial_y \, \boldsymbol{F^{\mu y}}(\boldsymbol{U}, \nabla \boldsymbol{U}) + \partial_z \, \boldsymbol{F^{\mu z}}(\boldsymbol{U}, \nabla \boldsymbol{U}). \end{aligned} \tag{2.11}$$

The flux functions depend on the state and the diffusive fluxes functions depend on the state vector as well as on its gradients. Note that the Navier-Stokes equations have hyperbolic as well as parabolic parts.

2.1.2. Euler equations

Inviscid compressible flow is governed by the Euler equations which are obtained from the Navier-Stokes equations (2.3)-(2.5) by neglecting the diffusive terms. Hence, the compact notation is obtained from Equation (2.11) by omitting the right hand side, which results in

$$\partial_t \, \boldsymbol{U} + \partial_x \, \boldsymbol{F^x}(\boldsymbol{U}) + \partial_y \, \boldsymbol{F^y}(\boldsymbol{U}) + \partial_z \, \boldsymbol{F^z}(\boldsymbol{U}) = 0 \tag{2.12}$$

using the vector of conserved variables (2.8) and the flux vectors (2.9). Without diffusive terms, this partial differential equation is purely hyperbolic. It governs the dynamics of a compressible fluid and hence, allows for, e.g. the appearance of shocks.

Since all equations above are derived from the conservation laws, they are formulated in conserved variables, but can also be stated in primitive variables. With Equation (2.7) the transformation from conservative $\boldsymbol{U} = (\rho, \rho v_x, \rho v_y, \rho v_z, e)^T$ to primitive variables $\boldsymbol{U}_{prim} = (\rho, v_x, v_y, v_z, p)^T$ is

$$\begin{aligned} \rho &= \rho \\ v_x &= \frac{\rho v_x}{\rho} \\ v_y &= \frac{\rho v_y}{\rho} \\ v_z &= \frac{\rho v_z}{\rho} \\ p &= (\gamma - 1)(e - \frac{\rho}{2}(v_x^2 + v_y^2 + v_z^2) \end{aligned} \tag{2.13}$$

where v_x, v_y, v_z is the velocity in x-, y-, z-direction respectivly. The Euler

equations written in primitives variables are

$$\begin{aligned}
&\partial_t \rho + v_x \partial_x \rho + v_y \partial_y \rho + v_y \partial_z \rho + \rho(\partial_x v_x + \partial_y v_y + \partial_z v_z) = 0 \\
&\partial_t v_x + v_x \partial_x v_x + v_y \partial_y v_x + v_z \partial_z v_x + \frac{1}{\rho}\partial_x p = 0 \\
&\partial_t v_y + v_x \partial_x v_y + v_y \partial_y v_y + v_z \partial_z v_y + \frac{1}{\rho}\partial_y p = 0 \\
&\partial_t v_z + v_x \partial_x v_z + v_y \partial_y v_z + v_z \partial_z v_z + \frac{1}{\rho}\partial_z p = 0 \\
&\partial_t p + v_x \partial_x p + v_y \partial_y p + v_z \partial_z p + \gamma p(v_x + v_y + v_z) = 0.
\end{aligned} \tag{2.14}$$

2.1.3. Linearized Euler equations

If non-linear effects do not have an impact and can be neglected, compressible flows governed by the Euler equations (2.12) can be linearized around a constant background state.
Only perturbations of the flow are considered as variables in the equation and thus the flow can be linearized around the constant background flow. For this, we split the variables into the constant background denoted by the subscript 0 and the perturbation denoted with the superscript $'$, e.g. $\rho = \rho_0 + \rho'$. By inserting this in the conservation laws written in their primitive form (2.14) and apply the assumption that the perturbation quantities are much less that the background flow ($\rho' \ll \rho_0$), the products of small quantities (e.g. rho', v', p') can be neglected since their multiplication ($v' \cdot p'$) becomes insignificant. Further, the time derivatives of constant background drop out.
The following Linearized Euler equations are obtained

$$\partial_t\ \rho' + \nabla \cdot \left(\boldsymbol{v}_0 \rho' + \rho_0 \boldsymbol{v}'\right) = 0 \tag{2.15a}$$

$$\partial_t\ \boldsymbol{v}' + \nabla \cdot \left(\boldsymbol{v}_0 \boldsymbol{v}' + \frac{1}{\rho_0} p'\right) = 0 \tag{2.15b}$$

$$\partial_t\ p' + \nabla \cdot \left(\boldsymbol{v}_0 p' + \gamma\ p_0\ \boldsymbol{v}'\right) = 0\,. \tag{2.15c}$$

The vector of primitive perturbation is $\boldsymbol{U}'_{prim} = (\rho', v'_x, v'_y, v'_z, p')^T$ and the according flux functions are

$$\boldsymbol{F^x}(\ \boldsymbol{U}'_{prim}) = \begin{pmatrix} v_{x0}\rho' + \rho_0 v'_x \\ v_{x0}v'_x + \frac{1}{\rho_0}p' \\ v_{x0}v'_y \\ v_{x0}v'_z \\ v_{x0}p' + \gamma p_0 v'_x \end{pmatrix}, \boldsymbol{F^y}(\ \boldsymbol{U}'_{prim}) = \begin{pmatrix} v_{y0}\rho' + \rho_0 v'_y \\ v_{y0}v'_x \\ v_{y0}v'_y + \frac{1}{\rho_0}p' \\ v_{y0}v'_z \\ v_{y0}p' + \gamma p_0 v'_y \end{pmatrix},$$

$$\boldsymbol{F^z}(\ \boldsymbol{U}'_{prim}) = \begin{pmatrix} v_{z0}\rho' + \rho_0 v'_z \\ v_{z0}v'_x \\ v_{z0}v'_y \\ v_{z0}v'_z + \frac{1}{\rho_0}p' \\ v_{z0}p' + \gamma p_0 v'_z \end{pmatrix}. \tag{2.16}$$

The according Jacobi matrices of the flux function are

$$\boldsymbol{J^x} = \frac{\partial \boldsymbol{F^x}}{\partial \boldsymbol{U}'_{prim}} = \begin{pmatrix} v_{x0} & \rho_0 & 0 & 0 & 0 \\ 0 & v_{x0} & 0 & 0 & \frac{1}{\rho_0} \\ 0 & 0 & v_{x0} & 0 & 0 \\ 0 & 0 & 0 & v_{x0} & 0 \\ 0 & \gamma p_0 & 0 & 0 & v_{x0} \end{pmatrix},$$

$$\boldsymbol{J^y} = \frac{\partial \boldsymbol{F^y}}{\partial \boldsymbol{U}'_{prim}} = \begin{pmatrix} v_{y0} & 0 & \rho_0 & 0 & 0 \\ 0 & v_{y0} & 0 & 0 & 0 \\ 0 & 0 & v_{y0} & 0 & \frac{1}{\rho_0} \\ 0 & 0 & 0 & v_{y0} & 0 \\ 0 & 0 & \gamma p_0 & 0 & v_{y0} \end{pmatrix}, \tag{2.17}$$

$$\boldsymbol{J^z} = \frac{\partial \boldsymbol{F^z}}{\partial \boldsymbol{U}'_{prim}} = \begin{pmatrix} v_{z0} & 0 & 0 & \rho_0 & 0 \\ 0 & v_{z0} & 0 & 0 & 0 \\ 0 & 0 & v_{z0} & 0 & 0 \\ 0 & 0 & 0 & v_{z0} & \frac{1}{\rho_0} \\ 0 & 0 & 0 & \gamma p_0 & v_{z0} \end{pmatrix}.$$

Since the Jacobi matrices (2.17) only depends on the background state, the Linearized Euler equations can be defined in their primitive formulation as

$$\partial_t \boldsymbol{U}'_{prim} + \boldsymbol{J^x}\partial_x\ \boldsymbol{U}'_{prim} + \boldsymbol{J^y}\partial_y\ \boldsymbol{U}'_{prim} + \boldsymbol{J^z}\partial_x\ \boldsymbol{U}'_{prim} = 0. \tag{2.18}$$

Note, that for the Linearized Euler equations the notation in (2.18) is also the conservative form, due to the constant matrices. Linearizing the Euler equations simplifies the PDE, but this is only allowed in regions with non-linear terms can be neglected which is true for small perturbations.

2.2. Numerical discretization

After introducing the equations, we now present the numerical methods to solve them with a focus on the methods used in this work. All presented partial differential equations (PDEs) are time as well as space dependent. The Discontinuous Galerkin scheme, presented in Section 2.2.1, is used in space and the Runge-Kutta methods, described in Section 2.2.2, is used in time.

2.2.1. Discretization in space: Discontinuous Galerkin

In this section, we derive the semi-discrete form of the DG method. For detailed derivation and numerical details on the modal DG scheme implemented in our numerical solver *Ateles*, please refer to Zudrop [8]. A general derivation can be found in Hartmann [23].
We start with the short notation of the conservation law (Equation (2.2)). Depending on the governing equations, this equation consists of convective parts defined by the left-hand side $\nabla \cdot \boldsymbol{F}(\boldsymbol{U})$ and viscous parts on the right-hand side. Since the viscous part is only relevant for the viscous compressible Navier-Stokes equations, we first concentrate on the convective part and derive the DG discretization. For generality, we combine the convective fluxes $\boldsymbol{F^x}$, $\boldsymbol{F^y}$, $\boldsymbol{F^z}$ to the generic flux $\boldsymbol{F}$.

To derive the variational formulation of the conservation laws, the first step is the multiplication by the so-called test function ψ:

$$\partial_t \boldsymbol{U}\psi + \nabla \cdot \boldsymbol{F}(\boldsymbol{U})\psi = 0. \tag{2.19}$$

Subsequently, we integrate over the domain Ω and, by using integration by parts, the following weak formulation is obtained:

$$\int_\Omega \partial_t \boldsymbol{U}\psi d\Omega - \int_\Omega \boldsymbol{F}(\boldsymbol{U}) \cdot \nabla\psi d\Omega + \int_{\partial\Omega} \boldsymbol{F}\psi \cdot \boldsymbol{n} dS = 0, \qquad \forall\psi, \tag{2.20}$$

where dS denotes the surface integral and n the corresponding vector normal to the surface.
The discrete variational formulation is obtained by considering a tessellation of the domain Ω into N closed, non-overlapping elements given by $\Upsilon =$

$\{\Omega_i | i = 1, 2, \ldots, N\}$, such that $\Omega = \cup_{i=1}^{N} \Omega_i$ and $\Omega_i \cap \Omega_j = \emptyset, \forall i \neq j$. Within the element Ω_i the DG method uses a polynomial representation to approximate the solution. We define a finite element space comprising of discontinuous polynomial functions of degree $m \geq 0$ given by

$$P^m = \{f \in [L^2(\Omega)]^m\}. \tag{2.21}$$

With the above definition we can write the approximate discrete solution $\boldsymbol{U}_h(\boldsymbol{x}, t)$ within each element using a polynomial function of degree m

$$\mathbf{U}_h(\mathbf{x}, t) = \sum_{k=1}^{m} \hat{u}_k \phi_k, \qquad \psi_h(\mathbf{x}) = \sum_{k=1}^{m} \hat{v}_k \phi_k, \tag{2.22}$$

where the expansion coefficients $\hat{u}_k$ and $\hat{v}_k$ denote the degrees of freedom of the approximated solution and of the test function, respectively. Notice, that there is no global continuity requirement for $\mathbf{U}_h$ and ψ_h in the previous definition.
Splitting the integrals in Equation (2.20) into a sum of integrals over elements Ω_i, we obtain the space-discrete variational formulation

$$\sum_{i=1}^{N} \left(\partial_t \int_{\Omega_i} \mathbf{U}_h \psi_h d\Omega - \int_{\Omega_i} \mathbf{F}(\mathbf{U}_h) \cdot \nabla \psi_h d\Omega + \int_{\partial\Omega_i} \boldsymbol{F}^* \cdot \psi_h \mathbf{n} dS \right) = 0, \forall \psi_h. \tag{2.23}$$

Due to element-local support of the numerical representation, the flux term is not uniquely defined at the element interfaces. Therefore, the flux function $\boldsymbol{F}$ in the surface integral is replaced by a numerical flux function $\mathbf{F}^*(\mathbf{U}_h^-, \mathbf{U}_h^+, \mathbf{n})$, where $\mathbf{U}_h^-$, $\mathbf{U}_h^+$ are the interior and exterior traces at the element face in the direction normal to the interface $\mathbf{n}$. Hence, the numerical flux only couples the direct neighboring elements.
Rearranging the terms and using the expansion coefficients (Equation (2.22)) leads to the space-discrete formulation: Find the coefficients $\hat{u}_k$ such that

$$\begin{aligned} &\sum_{i=1}^{N} \partial_t \int_{\Omega_i} \sum_{k=1}^{m} \hat{u}_k \phi_k \psi_h \; d\Omega = \\ &\qquad \sum_{i=1}^{N} \left(\int_{\Omega_i} \mathbf{F}(\mathbf{U}_h) \cdot \nabla \psi_h d\Omega - \int_{\partial\Omega_i} \boldsymbol{F}^* \cdot \psi_h \mathbf{n} dS \right), \forall \psi_h. \end{aligned} \tag{2.24}$$

The handling of numerical flux is well known from the FV schemes: At each element interface a Riemann problem exists. For the numerical scheme,

the exact solution of the Riemann problem is not required; it is sufficient to use an approximate Riemann solver instead. One approximation is the the Lax-Friedrich flux [57]:

$$\boldsymbol{F}^*(\boldsymbol{U}^-, \boldsymbol{U}^+) = \frac{\boldsymbol{F}(\boldsymbol{U}^-) + \boldsymbol{F}(\boldsymbol{U}^+)}{2} + \frac{c}{2}(\boldsymbol{U}^- \boldsymbol{n} - \boldsymbol{U}^+ \boldsymbol{n}), \tag{2.25}$$

with c the local speed of sound of the interior and exterior trace. A detailed explanation of the Riemann problem and its solution can be found in [57]. We utilize the Lax-Friedrich flux in this work, when solving the Euler equations. Looking at coupling with a domain that solves the Navier-stokes equations, this does not pose an issue since the domain with the Euler equations is typically solved with a higher order. For the DG scheme it holds true that the higher the order of the scheme is, the lower the impact of the numerical flux calculation is [58].

In case of the viscous Navier-Stokes equations, as already mentioned, there are convective and viscous parts. The presented derivative of the DG scheme is valid for the convective part described with flux functions $\boldsymbol{F^x}, \boldsymbol{F^y}, \boldsymbol{F^z}$ and the numerical flux function $\boldsymbol{F^*}$. The *rhs* consists of the viscous flux functions. The derivation of the DG scheme for viscous fluxes can be found in Hartmann [23] and Zudrop [8]. To concentrate on the viscous parts, we neglect the convective part for now and Equation (2.11) becomes

$$\partial_t \, \boldsymbol{U} = \partial_x \, \boldsymbol{F^{\mu x}}(\boldsymbol{U}, \nabla \boldsymbol{U}) + \partial_y \, \boldsymbol{F^{\mu y}}(\boldsymbol{U}, \nabla \boldsymbol{U}) + \partial_z \, \boldsymbol{F^{\mu z}}(\boldsymbol{U}, \nabla \boldsymbol{U}). \tag{2.26}$$

With the introduction of a viscosity tensor $\nu_{ij} \; \forall i, j = 1, ..., d$, the equation can be rewritten to

$$\partial_t \, \boldsymbol{U} = \nabla \cdot (\nu(\boldsymbol{U}) \cdot \nabla \boldsymbol{U}). \tag{2.27}$$

To obtain a first-order system, the equation can be rephrased to

$$\partial_t \, \boldsymbol{U} = \nabla \boldsymbol{\sigma} \tag{2.28a}$$

$$\boldsymbol{\sigma} = \nu(\boldsymbol{U}) \nabla \boldsymbol{U}, \tag{2.28b}$$

with the auxiliary tensor $\boldsymbol{\sigma} : \mathbb{R}^d \to \mathbb{R}^{(d+2) \times d}$.

To achieve the variational formulation, test functions are defined as polynomial functions of the introduced tessellation Υ:

$$\psi_h \in [P^m(\Upsilon)]^{d+2}, \qquad \overline{\psi}_h \in [P^m(\Upsilon)]^{(d+2)\times d}. \tag{2.29}$$

Note that the first function is the vector-valued polynomial function known from the convective parts and the second is a tensor-valued polynomial function. Multiplying the first-order system with the test functions and considering a single element Ω_i results in

$$\partial_t \int_{\Omega_i} \boldsymbol{U}_h \psi_h \; dV = \int_{\Omega_i} (\nabla \boldsymbol{\sigma}_h) \psi_h \; dV \tag{2.30a}$$

$$\partial_t \int_{\Omega_i} \boldsymbol{\sigma}_h \cdot \overline{\psi}_h \; dV = \int_{\Omega_i} (\nu(\boldsymbol{U}) \nabla \boldsymbol{U}_h) \cdot \overline{\psi}_h \; dV. \tag{2.30b}$$

Integration by parts and summation over all elements N yields

$$\sum_i^N \partial_t \int_{\Omega_i} \boldsymbol{U}_h \psi_h \; dV = \sum_i^N \left(- \int_{\Omega_i} \boldsymbol{\sigma}_h \cdot \nabla \psi_h dV + \int_{\partial\Omega_i} (\boldsymbol{\sigma}_h \cdot \boldsymbol{n}) \cdot \psi_h dS \right) \tag{2.31a}$$

$$\sum_i^N \partial_t \int_{\Omega_i} \boldsymbol{\sigma}_h \cdot \overline{\psi}_h \; dV = \sum_i^N \left(\int_{\Omega_i} \boldsymbol{U}_h \nabla \cdot (\nu^T(\boldsymbol{U}) \overline{\psi}_h) \; dV + \int_{\partial\Omega_i} \boldsymbol{U}_h \cdot (\nu^T(\boldsymbol{U}) \overline{\psi}_h \cdot \boldsymbol{n}) \; dS \right). \tag{2.31b}$$

The next step is to introduce the numerical flux functions $\boldsymbol{u}^* : \mathbb{R}^d \to \mathbb{R}^{d+2}$ and $\boldsymbol{\sigma}^* : \mathbb{R}^d \to \mathbb{R}^{(d+2)\times d}$ and, in contrast to the convective derivation, a second integration by parts is additionally applied

$$\sum_{i=1}^{N} \partial_t \int_{\Omega_i} \boldsymbol{U}_h \psi_h \, dV = \sum_{i=1}^{N} \Bigg(- \int_{\Omega_i} \sigma_h \cdot \nabla \psi_h \, dV + \int_{\partial\Omega_i} (\boldsymbol{\sigma}^* \cdot \boldsymbol{n}) \cdot \psi_h \, dS \Bigg) \tag{2.32a}$$

$$\sum_{i=1}^{N} \partial_t \int_{\Omega_i} \sigma_h \cdot \overline{\psi}_h \, dV = \sum_{i=1}^{N} \Bigg(\int_{\Omega_i} (\nu(\boldsymbol{U}_h) \nabla \boldsymbol{U}_h) \cdot \overline{\psi}_h \, dV + \int_{\partial\Omega_i} (\boldsymbol{u}^* - \boldsymbol{U}_h) \cdot (\nu^T(\boldsymbol{U}) \overline{\psi}_h \cdot \boldsymbol{n}) \, dS \Bigg). \tag{2.32b}$$

By setting the tensor-valued polynomial function $\overline{\psi}_h = \nabla \psi$ in Equation (2.32), the auxiliary variable σ_h can be eliminated and the variational equation is obtained

$$\sum_{i=1}^{N} \partial_t \int_{\Omega_i} \boldsymbol{U}_h \psi_h \, dV = \sum_{i=1}^{N} \Bigg(- \int_{\Omega_i} (\nu(\boldsymbol{U}_h) \nabla \boldsymbol{U}_h) \cdot \nabla \psi_h \, dV - \int_{\partial\Omega_i} (\boldsymbol{u}^* - \boldsymbol{U}_h) \cdot (\nu^T(\boldsymbol{U}_h) \nabla \psi_h \cdot \boldsymbol{n}) \, dS + \int_{\partial\Omega_i} (\boldsymbol{\sigma}^* \cdot \boldsymbol{n}) \cdot \psi_h \, dS \Bigg). \tag{2.33}$$

The viscous numerical flux $\boldsymbol{u}^*, \boldsymbol{\sigma}^*$ of the viscous part are chosen according to the Symmetric Interior Penalty Discontinuous Galerkin (SIPG) [8]

discretization to

$$\boldsymbol{u}^*(\boldsymbol{U}^-, \boldsymbol{U}^+) = \frac{\boldsymbol{U}^- + \boldsymbol{U}^+}{2}$$

$$\boldsymbol{\sigma}^*(\boldsymbol{U}^-, \nabla\boldsymbol{U}^-, \boldsymbol{U}^+, \nabla\boldsymbol{U}^+) = \frac{\nu(\boldsymbol{U}^-)\nabla\boldsymbol{U}^- + \nu(\boldsymbol{U}^+)\nabla\boldsymbol{U}^+}{2} \tag{2.34}$$

$$- C_{IP}(\boldsymbol{U}^-\boldsymbol{n} - \boldsymbol{U}^+\boldsymbol{n}).$$

The penalty parameter is set to

$$C_{IP} = C\frac{(m+1)(m+d)}{d}\frac{\partial\Omega_i}{\Omega_i}, \tag{2.35}$$

where $C \geq 1$ is required for stability, $\partial\Omega_i$ is the surface area, and Ω_i is the volume of the element.

Combining the variational formulation of the convective terms (Equation (2.24)) and the just derived variational formulation of viscous terms (Equation (2.33)), the Variational Interior Penalty Discontinuous Galerkin formulation of the viscous compressible Navier-Stokes equations is given by: Find $\boldsymbol{u_h} \in [P^m(\Upsilon)]^{d+2}$ such that for all $\phi_h \in [P^m(\Upsilon)]^{d+2}$

$$\begin{aligned}\sum_{i=1}^{N} \partial_t \int_{\Omega_i} \boldsymbol{U}_h\psi_h \, dV = \sum_{i=1}^{N}\Bigg(&- \int_{\Omega_i} (\nu(\boldsymbol{U}_h)\nabla\boldsymbol{U}_h)\cdot\nabla\psi_h \, dV \\ &- \int_{\partial\Omega_i} (\boldsymbol{u}^* - \boldsymbol{U}_h)\cdot(\nu^T(\boldsymbol{U}_h)\nabla\psi_h\cdot\boldsymbol{n}) \, dS \\ &+ \int_{\partial\Omega_i} (\boldsymbol{\sigma}^*\cdot\boldsymbol{n})\cdot\psi_h \, dS \\ &+ \int_{\Omega_i} \boldsymbol{F}(\boldsymbol{U}_h)\cdot\nabla\psi_h dV - \int_{\partial\Omega_i} (\boldsymbol{F}^*\cdot\boldsymbol{n})\cdot\psi_h dS\Bigg)\end{aligned} \tag{2.36}$$

holds true. The SIPG flux defined by Equation (2.34) is used for $\boldsymbol{\sigma}^*$ and $\boldsymbol{u}^*$, and $\boldsymbol{F}^*$ is the numerical flux of the convective part.
How boundaries are imposed with the help of the numerical flux functions can be found in Zudrop [8]. In this work when solving the Navier-Stokes equations, the HLL Riemann solver is used for $\boldsymbol{F}^*$ [57].

The previously obtained ordinary differential equation is the starting point for the time integration methods presented in the next section.

2.2.2. Discretization in time: Runge-Kutta

Runge-Kutta methods (RK) are classical *explicit* time integration methods [29]. These methods achieve high-order time integration by using multiple stages within each timestep to advance to the next timestep $t + \Delta t$ [28]. That is, a fourth-order Runge-Kutta method uses four sub-stages and a second-order Runge-Kutta method uses two sub-stages.

Re-arranging Equation (2.36) to matrix-vector notation yields

$$\partial_t \boldsymbol{U}_h = \boldsymbol{M}^{-1} \cdot \left(\boldsymbol{S} \cdot \boldsymbol{F}(\boldsymbol{U}_h(t)) - \boldsymbol{M}^{\boldsymbol{f}} \cdot \boldsymbol{F}^*(\boldsymbol{U}_h(t)) \right), \tag{2.37}$$

where $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{S}$ the stiffness matrix, and $\boldsymbol{M^f}$ the so-called face lifting matrix. $\boldsymbol{F}$ and $\boldsymbol{F^*}$ are flux and numerical flux functions, respectively. It can be further shortened to

$$\partial_t \boldsymbol{U_h} = \boldsymbol{M}^{-1} \cdot \boldsymbol{rhs}(\boldsymbol{U}_h(t), t). \tag{2.38}$$

The explicit RK method advances the system (2.38) from t to the next timestep $t + \Delta t$ by solving

$$\boldsymbol{U}_h(t + \Delta t) = \boldsymbol{U}_h(t) + \Delta t \sum_{i=1}^{s} b_i \boldsymbol{U}_{hi} \tag{2.39}$$

where b_i are coefficients from the Butcher tableau [27] and $\boldsymbol{U}_{hi}$ the sub-stages. For the second order it results in

$$\boldsymbol{U}_h(t + \Delta t) = \boldsymbol{U}_h(t) + \Delta t(\boldsymbol{U}_{h1} + \boldsymbol{U}_{h2}), \tag{2.40}$$

and for the fourth order it is

$$\boldsymbol{U}_h(t + \Delta t) = \boldsymbol{U}_h(t) + \frac{\Delta t}{6}(\boldsymbol{U}_{h1} + 2(\boldsymbol{U}_{h2} + \boldsymbol{U}_{h3}) + \boldsymbol{U}_{h4}), \tag{2.41}$$

where the sub-stages are

$$\boldsymbol{U}_{h1} = \boldsymbol{M}^{-1} \cdot \boldsymbol{rhs}(\boldsymbol{U}_h(t), t), \qquad \boldsymbol{U}_h^A = \boldsymbol{U}_h(t) + \frac{\Delta t}{2}\boldsymbol{U}_{h1}, \tag{2.42a}$$

$$\boldsymbol{U}_{h2} = \boldsymbol{M}^{-1} \cdot \boldsymbol{rhs}(\boldsymbol{U}_h^A, t + \frac{\Delta t}{2}), \quad \boldsymbol{U}_h^B = \boldsymbol{U}_h(t) + \frac{\Delta t}{2}\boldsymbol{U}_{h2}, \tag{2.42b}$$

$$\boldsymbol{U}_{h3} = \boldsymbol{M}^{-1} \cdot \boldsymbol{rhs}(\boldsymbol{U}_h^B, t + \frac{\Delta t}{2}), \quad \boldsymbol{U}_h^C = \boldsymbol{U}_h(t) + \Delta t \boldsymbol{U}_{h3}, \tag{2.42c}$$

$$\boldsymbol{U}_{h4} = \boldsymbol{M}^{-1} \cdot \boldsymbol{rhs}(\boldsymbol{U}_h^C, \Delta t). \tag{2.42d}$$

To construct the two sub-stages $\boldsymbol{U}_{h1}$ and $\boldsymbol{U}_{h2}$, which are the only ones

required for the second-order method, the midpoint rule is applied by using one midpoint at $t + \frac{\Delta t}{2}$.

As mentioned in the introduction (Chapter 1), explicit time integrations are only conditionally stable. For hyperbolic conservation laws solved with the explicit RK scheme combined with the DG discretization, the stability limit is given by

$$\alpha \cdot \frac{p^2 \cdot \Delta t}{h} \leq \text{CFL}, \tag{2.43}$$

where α denotes an upper bound on the global wave propagation speed and *CFL* denotes the Courant-Friedrich-Levy constant, which is independent of gridsize h and numerical order in space p of the DG scheme. Hence, the timestep Δt is limited by h/p^2 and results in a very small timestep for very high-order schemes. This timestep becomes even smaller for parabolic equations. Here, the stability limit is

$$\nu \cdot \frac{p^4 \cdot \Delta t}{h^2} \leq \text{CFL}, \tag{2.44}$$

with the diffusion constant ν. Thus, the timestep Δt is limited by h^2/p^4.

3. Partitioned coupling

As described in the introduction (Chapter 1), the term *partitioned coupling* refers to the splitting of a computational domain into subdomains such that each subdomain can be treated with different PDEs and numerical techniques that handle the physical characteristics best. The main motivation for coupled simulations is to enable the solution of computationally expensive problems: Using best-suited techniques for each subdomain can decrease the computational demand, in contrast to solving the entire problem monolithically. It can also improve the quality of the solution, for example, by using a high-order scheme with low dissipation and dispersion errors for wave propagation over long distances. When dividing a domain, interactions between subdomains must be realized as well: In case of surface coupling this can be achieved with boundary conditions. In literature, the term *domain decomposition* is used as an alternative to *partitioned coupling*, for instance by Schwarzkopff [36]. In this work, however, the term partitioned coupling is used in accordance with the DFG-funded ExaFSA project [47, 48].

In this chapter, we first describe the **general** coupling tasks that a coupling tool has to undertake. We establish two different implementations for partitioned coupling: A multi-solver approach using the coupling library *preCICE* and the integrated approach *APESmate* as part of the *APES* framework. Details of these will be elaborated in the corresponding sections of Chapter 4. Subsequently, in Section 3.2 static load balancing for coupled simulations is described since one goal of this work is the efficient computation on many compute cores. Here, we discuss how idling processes and imbalances in a coupled setup can be avoided. The following section (Section 3.3) addresses special features of the Discontinuous Galerkin (DG) method in the coupling context. This chapter is closed with a section about fluid dynamics in the context of coupling.

3.1. Coupling tasks

When coupling multiple subdomains with individual solvers, the following major tasks need to be tackled:

- Steering of individual solvers,
- communication of coupling data at interfaces,
- data mapping in time, and
- data mapping in space.

These tasks should be handled efficiently in parallel by a coupling approach and will be discussed in the following.

3.1.1. Steering of individual solvers

For a coupled simulation, individual solvers need to be steered by the coupling tool in such a way, that the solvers provide coupling data at the points in time and in space, where these coupling data are required. In this work, two implementations are established, a multi-solver and an integrated approach. Concerning these implementations, steering is done in two different ways: Figure 3.1 depicts both implementations on

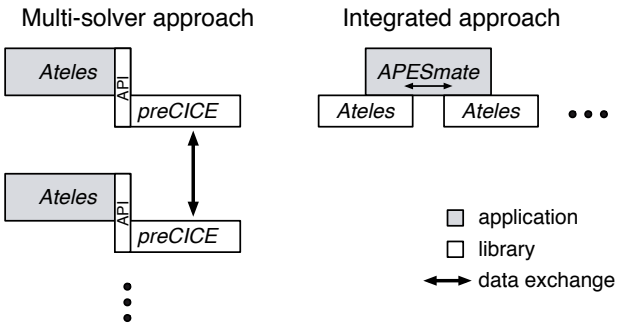


Figure 3.1.: Overview of steering individual solvers of the coupled simulation in both established coupling approaches.

an abstract level. To use the multi-solver approach as shown on the left side of Figure 3.1, an external coupling library is required. This, in turn, implies the availability of an application programming interfaces (API) to the coupling tool that can be accessed by individual solver. In such an approach, the coupled simulation is controlled by calls from a solver to the corresponding library and their return values. For example, the solver asks the library if the simulation is done or should continue, and has corresponding procedures implemented. For each subdomain, the individual solvers,

e.g. two instances of *Ateles*, are started and the coupling library, e.g. *preCICE*, is called internally to take care of the aforementioned coupling tasks.

Running individual executables in a coupled mode is more involving on supercomputers: For example, control mechanisms are required to finalize both solvers correctly in case one of them reports an instability. Also, binding the correct number of MPI-ranks to each individual executable to avoid running multiple MPI-ranks on the same core needs to be taken into account. For the multi-solver approach with individual executables, these must be linked to the coupling library during compilation and the correct MPI-ranks must be chosen when submitting jobs. Merging multiple executables into one executable is also possible, but this implies more modifications to the *black-box* solver and a larger API: The solver, for instance, must be capable of using a provided MPI communicator for internal communication.

In contrast, the integrated approach (right side of Figure 3.1), where the coupling concept is implemented in the same framework as the numerical solvers, only has one single executable with all required solvers. Here, the solvers are incorporated as a library. Since the coupling tool has access to the entire data structures of the solvers, the steering is more direct. The coupling tool on the highest hierarchy level (Figure 3.1) has the control of the simulation. For example, it checks for the synchronization step and triggers the solver to run until the next synchronization step or to finalize its subdomain. Within a single timestep, the solver has no further interaction with the coupling tool. Of course, this direct steering is only feasible if the solver and the coupling tool are implemented within the same framework. The benefit of the integrated approach is the handling of one executable instead of several ones on a supercomputer. Porting software, establishing the correct binding of MPI-ranks to the individual subdomains, e.g. to avoid running multiple MPI-tasks on one CPU, and compiling the job script is greatly simplified when running a single application.

3.1.2. Communication of coupling data

A coupling is based on the exchange of data at defined coupling points at the interface at defined synchronization steps. Such data are coupling variables such as density, velocity, and pressure for flow simulations, and are defined by the user in the corresponding coupling configuration (file). They are requested by one subdomain from the other subdomain at the coupling interface. The definition of coupling points, as well as the synchro-

nization step, will be discussed in the next sections. The challenge is that parallel processes of each subdomain can be arbitrarily distributed. Furthermore, only the processes that accommodate elements at the coupling interface, henceforth referred to as coupling elements, should be involved in the communication between subdomains. These processes establish a point-to-point communication for data mapping between neighboring coupling element. Figure 3.2 sketches a setup with two subdomains A and B, where arrows illustrate the required communication. R_{Ai} and R_{Bi} depict the processes arbitrarily distributed on the subdomains. Here the processes R_{A1}, R_{A5} and R_{A11}, R_{A20} on subdomain A should communicate directly with the processes R_{B3} and R_{B35} on subdomain B, respectively

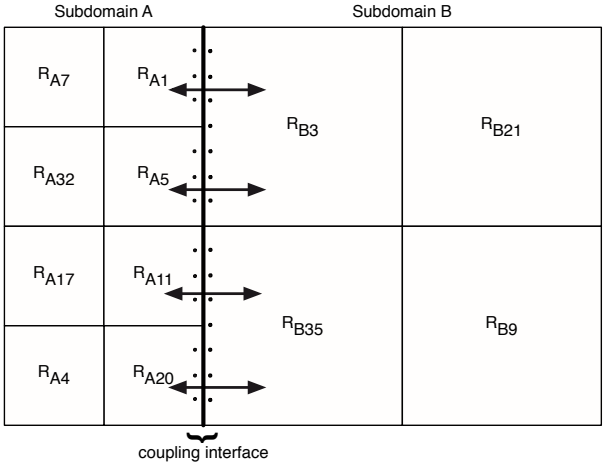


Figure 3.2.: Sketch of two subdomains A and B where the corresponding processes R_{Ai} and R_{Bi} at the interface need to communicate (blue arrows) the coupling data between each other. For the sake of visualization, the points are slightly shifted away from the interface.

The focus is on the efficient communication of these data from the coupling points of the subdomain A to the coupling points of the second subdomain B and vice versa. Since we are aiming for large-scale simulations on massively parallel systems and even prospective exascale systems,

communication must not pose a bottleneck. To establish communication at every synchronization step, the communication infrastructure must be initialized first as discussed below.

Initialization of communication: In this phase, the communication relations between the processes, which contain coupling elements and, therefore, participate in the coupling, have to be established. This is closely related to data mapping between different coupling points at the coupling interfaces, since only the processes with coupling points are included in the corresponding data mapping and should talk to each other. This communication should be as direct as possible. However, depending on the implementation of the coupling approach, this initialization is more or less challenging as discussed below.
Communication at synchronization steps: This part of the communication is even more important than the communication during initialization, since it happens at every synchronization step during the simulation time. The communication should be significantly faster than the computation of the individual subdomains. Therefore, after establishing the communication relations between individual processes, the actual exchange of data should be in a direct way. All information should be available locally to a process to avoid expensive communication, e.g. *all-to-all*. This *point-to-point* communication is sometimes referred to as *M:N* communication indicating M processes on one side exchange with N processes on the other side.

Realization of both steps, *Initialization of communication* and *Communication at synchronization steps*, depends on the coupling approach and its implementation. Naturally, *Initialization of communication* is more simple for the integrated approach. The integrated approach is a single application that uses a single MPI communicator. For the multi-solver approach, employing a coupling library to establish a *point-to-point* communication is not trivial. As mentioned in the previous Section 3.1.1, the multi-solver approach starts individual applications that are connected via the coupling library. Hence, different MPI communicators are involved. The coupling library treats solvers as black boxes and, therefore, only acts on input and output data from them. Nevertheless, communication must not contradict exascale computing paradigms. Furthermore, memory might be a limited resource, thus, for instance gathering all information on the root processes during initialization to realize the mapping is not possible for very large interfaces. After initialization, *point-to-point* communication, i.e. direct communication between two processes sharing a coupling interface, is similar for both implementations in terms of available

local information on every MPI-rank. Of course, *APESmate* communicates within one MPI communicator, whereas *preCICE* needs to invoke sockets or MPI-ports to communicate across different MPI communicators. More detailed descriptions of the individual implementations can be found in Section 4.3.3 and Section 4.4.3, respectively.

3.1.3. Data mapping in time

Data mapping in time is one of the required coupling tasks when subdomains use different time integration methods or different physical timestep sizes due to different mesh discretizations or multi-scale phenomena. When exactly a synchronization in time happens is testcase and solver-dependent. Whether explicit or implicit time integration is used, for instance, will influence the size of the simulation timestep, since explicit time integration is limited by the so-called *CFL* condition. In the following, we will explain various concepts concerning the data mapping in time for coupled simulations: Sub-cycling, multi-stage time integration, and iterative coupling.
Sub-cycling: Choosing a synchronization timestep that is larger than the individually required timesteps of the numerical solvers, will result in sub-cycling. Figure 3.3 demonstrates such a configuration where the

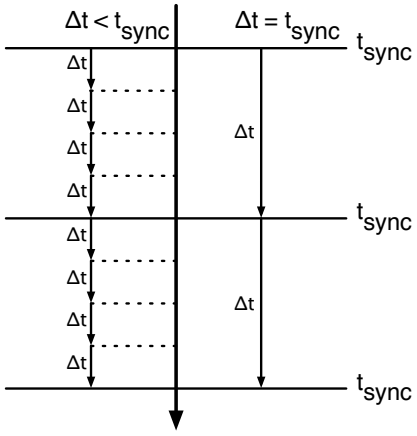


Figure 3.3.: Sub-cycling of one subdomain. The left subdomain has a smaller timestep Δt than the synchronization timestep t_{sync} which leads to sub-cycling of this subdomain. The solid horizontal lines indicate data exchange at each synchronization timestep t_{sync}. Dotted lines depict data exchange required for the sub-cycling left subdomain.

left subdomain has a smaller timestep Δt than the chosen synchronization timestep t_{sync}. Hence, the left subdomain requires four small timesteps Δt until it reaches the synchronization timestep. Dotted lines indicate points

in time, where the left subdomain requires data from the right subdomain. For consistent time integration, it is important to ensure that the required data at the interface is provided at those sub-cycling timesteps as well. It either needs to be ensured by the coupling tool by providing a high-order extrapolation in time or the solver needs to provide a corresponding time integration method. This is, for example, offered by the ADER time integration method [59]. In this work, we do not allow sub-cycling (due to adaptive time-stepping) of one solver to avoid inconsistent coupling in time. Allowing sub-cycling (adaptive timestepping) can improve the performance of multi-scale simulations which implies using individual timestep sizes tailored to the individual physics and, therefore, optimal usage of the individual numerical schemes. Furthermore, we do not consider limitations on the numerical order of the solution due to inconsistent coupling in time, which of course can reduce the overall order of the numerical scheme.
Multi-stage time integration: Another challenge is the time-consistent coupling of multi-stage time integration schemes. As presented in Section 2.2.2, we use the explicit two-stage Runge-Kutta (RK) method. This method constructs two sub-stages via a midpoint in the timestep which leads to a similar problem as for the sub-cycling in Figure 3.3: In the synchronization at the midpoint of the RK timestep one subdomain needs data at the coupling interface, where the other subdomain does not provide it. This can be circumvented if both subdomains use the same time integration method and the coupling tool supports the exchange of data at the midpoint of an RK timestep. This, however, contradicts the concept of coupling *black-box* solvers, while it is possible for an integrated approach.
Iterative coupling: In case of implicit time integration in a solver, where the solver requires data of the old as well as the new timestep to evolve in time, iterative coupling is required: This is, performing an iterative process of solver evaluations in every timestep until convergence of coupling values at the interface at this timestep is achieved. Iterative coupling is also called implicit coupling and different iterative solvers are available [60, 61]. Such an iterative coupling is typically more stable, but costly and is often considered to couple multi-physics phenomena like fluid-structure interaction. More details about implicit coupling approaches and their realization are addressed by Gatzhammer in [40]. Implicit coupling could be helpful regarding sub-cycling and time-consistent coupling when using multi-stage time integration.

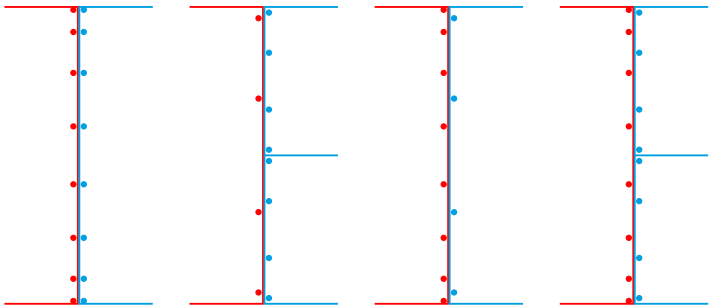
When exploiting explicit time integration methods, where only data of previous timesteps are required to advance in time, only one iteration is required, i.e. an iterative coupling with only one iteration is used to

compute the new timestep. Such a coupling, where only one iteration of the solver evaluation is performed, is also referred to as explicit coupling. In case an explicit time integration is used, for some coupling tools it is a pre-requisite to exchange the initial condition at each coupling interface since the solver requires valid boundary conditions at the initial step.

For the simulation of fluid-acoustic interaction in this work, we use explicit time integration and a fixed synchronization timestep for all subdomains. This fixed timestep is set to the minimum of all individual timesteps limited by the individual CFL condition of each subdomain and equation system. Multi-stage time integration will be investigated during load balancing investigations.

3.1.4. Data mapping in space

The interaction between different subdomains is realized by exchanging coupling variables at coupling interfaces. These data are exchanged at defined points on the coupling interface. Each subdomain requests data at individual coupling points that depend on the mesh discretization and the order of the numerical scheme. Typically, these coupling points in space do not coincide at the coupling interface: For instance, two neighboring subdomains can use different meshes, different approximation orders, or schemes with e.g. different integration points. In such cases, one subdomain requires data at specified points, but the second subdomain operates on data at different points. Figure 5.15 depicts the integration points for DG schemes of different order and mesh size. A matching grid as shown in Figure 3.4a is only achieved when using the same grid size h and polynomial order p in each subdomain. n is the number of integration points which is a function of h and p. All other examples in Figure 3.4b - Figure 3.4d present non-matching interfaces. In Figure 3.4b, both subdomains have fourth-order discretization, but the computational grid of the right subdomain is twice as fine as the left subdomain, which leads to twice the number of integration points at the right interface n_R. Figure 3.4c presents an example where the same computational grid in both subdomains is chosen, but different orders, i.e. left subdomain is eighth-order and right subdomain is fourth-order, respectively, which results in twice the number of integration points at the left interface n_L. Finally, Figure 3.4d shows an interface with the same number of integration points ($n_L = n_R$) but still obtaining a non-matching grid due to different orders and mesh resolutions. Here, the left subdomain has a coarser grid but an eighth-order DG scheme in space, and the right subdomain has two-fold finer grid, but only fourth-order



(a) Matching: $h_L = h_R$, $p_L = p_R$, $n_L = n_R$

(b) Non-matching: $h_L > h_R$, $p_L = p_R$, $n_L < n_R$

(c) Non-matching: $h_L = h_R$, $p_L > p_R$, $n_L > n_R$

(d) Non-matching: $h_L > h_R$, $p_L > p_R$ $n_L = n_R$

Figure 3.4.: Example of matching and non-matching integration points at the coupling interface when using DG. At least one element per subdomain is shown exemplarily. Grid size h and/or the polynomial order p, and, therefore, the number of integration points n vary. For illustration purpose, the coupling points are shifted to the left and right of the coupling interface, in reality the points are lying directly on the interface. Please note that integration points for the DG scheme are not equidistant within the element.

DG scheme. This is due to the non-equidistant distribution of integration points in DG schemes. Furthermore, when coupling different discretization methods in space, e.g. an FV scheme with a DG scheme, we typically face non-matching coupling interfaces. Therefore, good data mapping for non-matching integration points at interfaces is required and will influence the overall coupling accuracy.

Especially for high-order discretizations in one subdomain, a good data mapping in space is important: Low-order data at the interface can impair the error convergence of individual solvers. However, high-order data mapping can be achieved by solver-internal methods to obtain the values at arbitrary coupling points or by providing additional data points resulting in higher effort. Additionally, interpolation methods can be expensive. It can be discussed up to which numerical order data mapping is useful: Assuming the coupling of a lower-order discretization with a higher-order

discretization, it can be unnecessarily costly to provide higher-order data to the lower-order discretization. Assuming *black-box* coupling, where a coupling tool only acts on input/output data, an external interpolation method within the coupling tool is typically used. Of course, acting purely on point data, information on the specific discretization is lost. In case of matching interfaces, a requesting subdomain can directly use the data that the other solver is working on. Here, no additional effort is required, but the coupling tool cannot influence the quality of the data. For non-matching interfaces, additional work in form of matching techniques is required, for which the coupling tool has several options: a) Use the provided data and apply an external interpolation method. This option is flexible and the coupling tool can influence the quality by choosing appropriate mapping methods; it is expensive though. b) Use solver-internal information, e.g. the DG polynomials to obtain the values at the requested points. This yields less additional effort than external interpolation methods since the polynomial is already constructed, and the quality of the solution (from the solver) is maintained. However, this is only possible if the coupling tool has access to the solver-internal data. A *white-box* approach can avoid this issue by leveraging the internally known information of the scheme to provide high quality point values, i.e. using the aforementioned option a). For example, Discontinuous Galerkin is a numerical scheme that can provide data at arbitrary points in space (see Section 3.3).

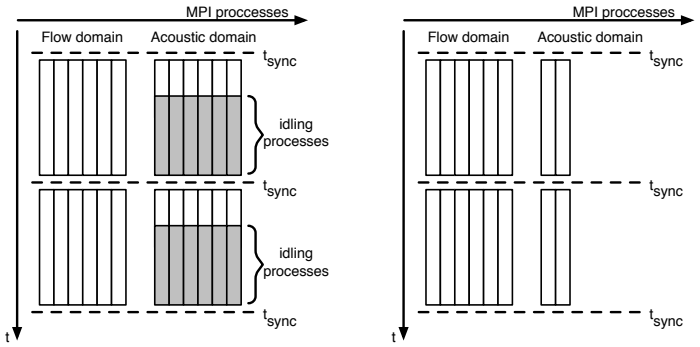
3.2. Static load balancing

Coupled simulations on massively parallel systems often result in load imbalances due to different workloads of each subdomain. Therefore, balancing these loads is an important topic for coupled simulations running on massively parallel systems. The available resources should be used efficiently, which implies that the workload across the processes must be evenly distributed. Static balancing in this context means that imbalances introduced during execution are not considered. We assume that the load distribution does not vary during the execution since the coupling setup (e.g. location of the coupling interface) does not change over time. With respect to partitioned coupling, we can differentiate load balancing **between** the individual subdomains and the load balancing **within** each subdomain. We first discuss the load balancing between the subdomains due to different workloads and afterwards elaborate on load imbalances within subdomains, originating from the additional workload at the coupling interfaces.

3.2.1. Load balancing between subdomains

Different equations, spatial domains, and numerical discretizations lead to different workloads of individual subdomains. For example, solving a non-linear flow with around 1000 fine elements and lower-order has a different workload than a linearized acoustic field with 500 coarse elements and a higher order. If the subdomains are not properly distributed according to their workload then the different workloads will lead to imbalances in the coupled simulation, even if a perfect load balance within the individual subdomains would be given. We always assume parallel coupling, which means the subdomains are solved in parallel. A static load balancing can be achieved by choosing an appropriate number of processes for each subdomain, s.t. its computation takes approximatively the same time, by assuming no adaptive time-stepping, a fixed coupling interface, and no sub-cycling, Figure 3.5 shows an example of a non-linear flow domain coupled with a linear acoustic domain. Here, the workload of the acoustic domain is only $\frac{1}{3}$ of the flow domain. In Figure 3.5a the same number of MPI-processes, i.e. 6, per domain are used which results in idle time for all 6 processes of the acoustic domain. But only $\frac{1}{3}$ of the 6 MPI-processes are required to solve the acoustic domain in the same computational time. Hence, Figure 3.5b presents a perfect load balancing using the appropriate number of MPI-processes, i.e. only 2 processes in the acoustic domain. Instead of using 12 MPI-processes in total only 8 MPI-processes are enough to solve this setup in the same computational time and avoid idling processes. Choosing the number of processes according to the actual load requires a strategy that can be based on heuristics or weights that mirror the workload. Such weights naturally depend on the solver and the physical regime (equation system) to solve, but also on the computer system, e.g. memory cache size. Furthermore, when using the DG solver *Ateles*, the weights depend on the chosen spatial order. Accordingly, some experience and preliminary investigations are required.

Applying a dynamic load balancing between two subdomains is even more challenging. Here, "dynamic" means the re-partitioning of all processes between the subdomains, i.e. moving some MPI-processes from subdomain A to subdomain B during execution. Re-partitioning of processes at runtime using an integrated coupling approach (e.g. *APESmate*) might be simpler to implement than for a multi-solver approach that uses an external coupling library (e.g. *preCICE*). Since the integrated approach is a single application, a dynamic LB mechanism designed for one application can be applied. In detail, this comprises a) the determination of the compute



(a) Same number of MPI-processes per subdomain yielding load imbalances and idling processes

(b) Appropriate number of processes

Figure 3.5.: Example of load balancing between a flow domain and an acoustic domain. a) Same number of MPI-processes in both subdomains yield to idling processes in the acoustic subdomain; b) adapted number of processes in the acoustic domain which avoids idling processes.

weights by measuring runtimes, b) the re-distribution of the domain onto a different number of MPI-processes according to the weights, and c) the update of the domain-specific MPI communicators. In contrast, utilizing an external coupling library results typically in separate applications coupled via the library. In this case, the dynamic load balancing is more involving, e.g. exchanging MPI-processes between the subdomains might imply to store results intermediately and to restart the coupled simulation with different processor counts due to different MPI communicators. Dynamic load balancing gains importance when the workload per subdomain changes during computation due to adaptive time-stepping or dynamic changes of coupling interfaces. Dynamic load balancing is not part of this work but should be considered in the future.

3.2.2. Load balancing within a subdomain

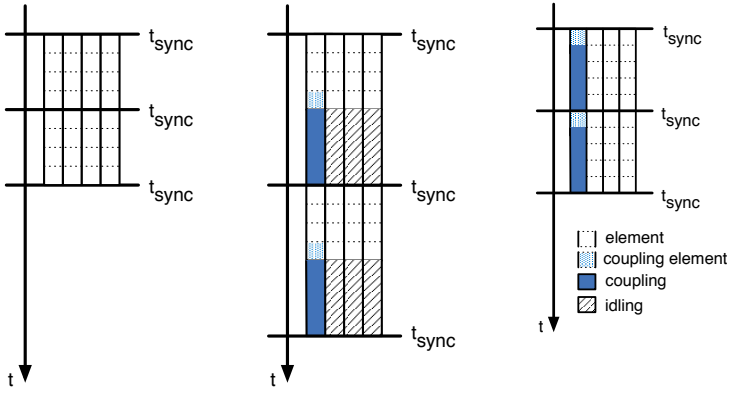
Several factors influence the workload and, therefore, the load balancing of a simulation. Considering a computational domain discretized into several elements and distributed over processes, an ideal load balancing is

achieved when all processes have the same workload. The workload of a single element depends on the numerical scheme but also additional work like applying boundary conditions or source terms. Here, we discuss the load imbalances within a subdomain introduced by additional coupling load at the coupling interfaces. Elements at this coupling interface are under additional load: With the integrated approach *APESmate*, such elements need to provide coupling data at coupling points, which can be costly depending on the chosen discretization (refer to Section 3.3). In case of the multi-solver approach these coupling elements have to "write to" the black-box tool and "read from" it. An element located at a coupling interface is involved in the numerical computation of the equation ("compute workload") as well as in coupling ("coupling workload"). Such an element will be referred to as "coupling element " throughout this thesis.

During a parallel simulation, synchronization steps across all processes are required, e.g. to exchange local timestepping information after completing a timestep or the simulation status. At such a synchronization step, processes with less workload have to wait (idle) for processes with more workload to complete their computation. For an efficient load balancing, the compute and coupling loads of a coupling element must not be separated by a synchronization point, so that both workloads can be considered jointly for re-partitioning.

Figure 3.6 presents common examples for load balancing scenarios within a subdomain for single-stage time integration. In each scenario, 4 MPI-processes are depicted by solid lines and 16 elements by dotted lines. In Figure 3.6a, an ideal balancing without coupling elements is shown. Each process computes 4 elements and, therefore, has the same workload per timestep. The second example (Figure 3.6b) shows the same element distribution, i.e. 4 elements per MPI-process, but includes 1 coupling element (blue dotted). This coupling element has additional work (blue blocks) while the other 3 processes are idling (shaded blocks) at the synchronization timestep. Here, we can see that large imbalances in workload yield a suboptimal load balancing in a subdomain and result in idling processes. Figure 3.6c presents the same setup but with a sensible load balancing: Here, the loads are distributed according to the total load per element, i.e. computation + coupling load: 1 expensive coupling element on 1 rank and 5 elements on each of the 3 other ranks. This re-partitioning can be done with specific libraries like ParMETIS[1] or other re-partitioning methods that use weights to indicate the load per element and re-distribute

[1] `http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview`



(a) Ideal LB yielding same number of elements on each processes without any element involved in coupling

(b) Same number of elements on each processes with one element involved in coupling.

(c) Using re-balancing mechanics resulting in good load balancing with one element involved in coupling.

Figure 3.6.: Example of load balancing within a subdomain that a) does not have any coupling elements and therefore ideal balancing is achieved by the same number of elements per rank; b) has the same number of elements per rank with one coupling element which leads to idling ranks while coupling; c) has one coupling element but rebalanced load, which results in same computation times and avoids idling ranks.

accordingly. In *APES*, we use the parallel re-partitioning algorithm *SPartA* [55], which is described in Section 4.2.1. Figure 3.6 also highlights that a good load balancing results in an overall smaller computation time.

The compute load can be partitioned down to one element per process in *Ateles*, which naturally limits the maximum number of processes. For example in Figure 3.6 this means 16 ranks. Inhomogeneous loads of individual elements limit the sensible number of processes to be utilized even further: This is the case in Figure 3.6 where the cost of the single coupling element (blue dotted + blue) is assumed to be extraordinary high, e.g. 5 times the workload of any other element. The re-partitioning algorithm tries to distribute elements such that each partition has a similar load. In this example, the re-partitioning results in 1 rank with the coupling

element and 3 more ranks with 5 elements each (Figure 3.6c). The high load of the single coupling element located on 1 rank determines how much load must be located on the other ranks to minimize idling times. Therefore, the sensible number of processes to utilize in this example is 4. Using more ranks does not reduce the overall runtime any further, since it is determined by the high load of one element that cannot be further distributed. In general, we have to consider the ratio of the maximum load and the average load of elements. This ratio determines the number of average-load elements that can be solved in the same time on one rank while a single high-load element is solved. In our example we have 15 compute only elements and 1 coupling element, and we denote load by the arbitrary unit l: Each compute element has $1l$, while the single coupling element has $5l$. The overall load is $20l$. Locating the coupling element on one rank, the remaining load of $20l$-$5l = 15l$ (the compute elements) needs to be distributed on a sensible number of ranks. The load ratio between coupling and average is $\frac{5}{1}$, which means that while the coupling element is solved, 5 compute elements can be solved on one rank. Considering the remaining load of 15, $\frac{15}{5} = 3$ ranks are required. The total number of sensible ranks can be computed as

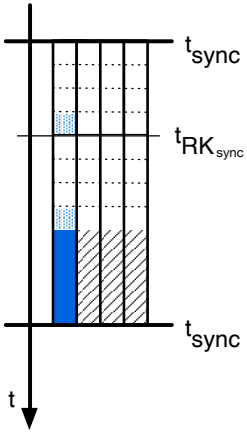
$$\begin{aligned} N_{sensible} &= N_{remaining} + N_{max} \\ &= \frac{W_{total} - W_{max}}{W_{max}} + \frac{W_{max}}{W_{max}} \\ &= \frac{W_{total} - W_{max} + W_{max}}{W_{max}} \\ &= \frac{W_{total}}{W_{max}} \end{aligned} \tag{3.1}$$

W denote loads (="weight"), N the number of processes, and subscripts the kind of loads, respectively. With this, we obtain $N_{sensible} = 4$ for our example. Hence, inhomogeneous workload of elements within one subdomain can limit the sensible number of processes and must be considered.

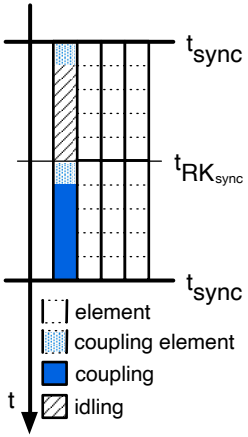
Load balancing within a subdomain is a general concept and applies to the multi-solver as well as the integrated coupling approach. However, for the multi-solver approach with an external library the load balancing inside of the coupling tool is not addressed in this work.

So far, we have only looked at single-stage time integration (Figure 3.6). Compared to this, multi-stage time integration methods utilize multiple sub-stages within one timestep to increase the order in time. As presented in Section 2.2.2, the RK method is such a multi-stage time integration method. Here, second-order is assumed where the midpoint rule is applied

by using one midpoint at $t + \frac{\Delta t}{2}$ to construct the two sub-stages. Typically, in a coupled simulation coupling data is exchanged at the end of a complete timestep, which implies that the coupling data is only available at that point in time. A second-order RK timestep has one synchronization within the subdomain and one synchronization between subdomains at the end, while a first order timestep has just one synchronization between subdomains at the end. Please keep in mind, that load can only be balanced between synchronization points.



(a) Load balanced according to the midpoint of the 2nd order RK method leading to idling processes at the end of the complete timestep



(b) Load balanced including the coupling load yielding one idling process at the midpoint of the timestep of the 2nd order RK method

Figure 3.7.: Load balancing examples with multi-stage time integration. t_{RK} depicts **synchronization steps** at midpoints within the second-order RK method **within** a subdomain and t_{sync} illustrates **synchronization steps** at complete timesteps **between** subdomains.

Figure 3.7 presents the load balancing examples within a subdomain when using multi-stage time integration. Here, t_{RKsync} depicts **synchronization steps** at midpoints and t_{sync} illustrates **synchronization steps** at "complete" timesteps. t_{RKsync} synchronizes **within** a subdomain, while t_{sync} synchronizes **between** subdomains. The same exemplary scenario as

in Figure 3.6 is chosen: 4 MPI-ranks with 16 elements including 1 coupling element. In Figure 3.7a, only the compute load is balanced, which results in a balancing according to the the midpoints of the second-order RK method. At the end of the complete timestep, the coupling data need to be provided by the single coupling element and idling ranks appear. Figure 3.7b gives an example, where the load of computation **and** coupling is balanced. This implies that 1 rank only has the coupling element while each of the other 3 ranks has 5 elements without coupling. Due to the different compute load, the rank with the coupling element is idling at the midpoint while the other ranks are computing the 3 additional elements. For the synchronization step t_{sync} at the end of the timestep, the coupling data needs to be provided and, thus, all ranks require the same amount of time which avoids idling ranks. Hence, using a multi-stage approach in a coupled setup poses a conflict when coupling data is only provided at the end of a complete RK step: Either there are idling ranks in at the end of the timestep while coupling (and only the compute load is balanced), see Figure 3.7a; Or coupling and compute load are considered together and balanced accordingly, which results in idling ranks at every midpoint in the timestep for processes with coupling load, see Figure 3.7b. Providing coupling data at every midpoint in a timestep could circumvent this conflict but it needs to be supported by the coupling tool and the solvers. Furthermore, we do not consider re-partitioning of elements for each sub-stage since the cost of re-partitioning would be larger than the benefit of avoiding idling ranks. Typically, the number of coupling elements is considerably lower than the number of elements without additional coupling work. Therefore, we consider compute and coupling load together and accept a few idling processes in synchronization at the midpoint of the timestep.

In general, the coupling loads introduce load imbalances within a subdomain due to additional work for some elements of the partitions. Therefore, imbalances should be minimized by, for instance, additional mechanisms like re-partitioning. Re-partitioning, however, impacts single-stage time integration methods and multi-stage time integration methods differently. Additionally, all synchronization points between the computation step and the coupling step should be avoided to enable the joint balancing of compute and coupling workload. A good load balancing in the subdomains will avoid idling processes and reduce the overall computation time.

3.3. Discontinuous Galerkin in the context of coupling

In this work, we are using a high-order Discontinuous Galerkin (DG) method to solve the flow field as well as the acoustic far field. For the acoustic far field in particular, which is computationally expensive due to its spatial dimension, we are aiming for a high order in space. Using a higher order allows for the mesh to be significantly coarser which reduces the degrees of freedom, while still maintaining a reasonably small error. The grid size h and the numerical order in space p can be chosen in such a way that the timestep t limited by the *CFL* condition $t \sim h/p^2$, becomes sufficiently large. When coupling different physical regimes, e.g. an acoustic domain with a flow domain, we ensure the timestep t in both domains to be equal to ease the data mapping in time (see Section 3.1.3). Using the CLF condition Equation (2.43) it follows

$$t_a = \mathrm{CFL}_a \, \alpha_a \, \frac{h_a}{p_a^2}$$

$$t_f = \mathrm{CFL}_f \, \alpha_f \, \frac{h_f}{p_f^2}$$

where the subscript a defines acoustic parameters and f flow parameters. α denotes an upper bound on the global wave propagation speed. Assuming the parameters CFL_a and CFL_f as well as α_a and α_f to be fixed and forcing $t_a = t_f$ lead to

$$\frac{h_a}{p_a^2} = \frac{\mathrm{CFL}_f}{\mathrm{CFL}_a} \, \frac{\alpha_f}{\alpha_a} \, \frac{h_f}{p_f^2} \, .$$

With this equation, grid sizes and orders can be chosen accordingly.

Additionally, the high-order DG scheme provides low dispersion and low dissipation error, which is in particular suited for linear acoustic wave propagation over long distances. Since the general method is already described in Section 2.2.1, we focus on the implication and benefits in the coupling context in the following. Two major challenges are addressed: the choice of the coupling points at the coupling interface and the evaluation method of the DG scheme at arbitrary points in space.

3.3.1. Coupling points

As presented in Section 3.1.4, the interaction between two domains is realized via boundary conditions. There, coupling data needs to be exchanged at the coupling interface. This is done at specific coupling points. Figure 5.15 (of Section 3.1.4) shows several examples of individual coupling points at coupling interfaces. Here, we discuss which coupling points can be chosen in the DG method, in particular when using a high order.

Our solver *Ateles* implements the modal DG scheme (see Section 2.2.1): A nodal representation is provided via a transformation based on numerical integration at fixed integration points. *Ateles* implements the Gauss-Legendre integration [62] or Gauss-Chebyshev [63] integration. More details on the modal implementation and the modal-to-nodal transformations can be found in Section 4.2 and in [64]. Here, we focus on Gauss-Legendre integration points as example. These integration points are defined in the interval $[-1, 1]$ and are the zeros of the Legendre polynomial, i.e. the point x_i is the i-th zero of the Legendre polynomial. The number of Gauss points n depends on the polynomial degree m and is $n = m + 1$. Figure 3.8 presents the first six Legendre polynomials and illustrates their zeros x_i. These reference positions on the interval $[-1, 1]$ are then transformed into physical coordinates.

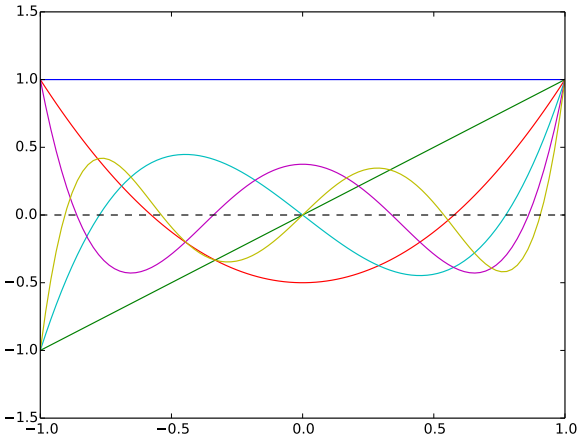


Figure 3.8.: Legendre polynomials for polynomial degree $m = 0...5$.

These Gauss points are used as coupling points at the coupling interface, which are the red and blue dots in Figure 5.15. A DG scheme of order p result in a polynomial degree of $m = p - 1$. For a coupling element at the coupling interface there are $n = m + 1$ coupling points per direction. That is, for a 2D coupling face of a 3D coupling element there are n points in x-direction and n points in y-direction, resulting in n^2 coupling points.

As seen in Figure 3.8, the Gauss points feature a non-equidistant distribution. Actually, they are more concentrated towards the interval boundaries -1 and 1. Referring again to Figure 5.15, the one subdomain (red) requires data at its Gauss points (red dots) while the other subdomain (blue) requires data at its own Gauss points (blue dots). When using an external library, e.g. *preCICE*, each solver sends these points to the coupling tool along with its mesh connectivity. An external interpolation is subsequently done by the coupling tool based purely on geometric data. The presented non-equidistant distribution of the coupling points is challenging for this interpolation method, in case of non-matching coupling interfaces (Section 3.1.4 and Section 4.3.4). Using an integrated approach, e.g. *APESmate*, the challenge of non-matching interfaces remains, but solver-specific evaluation methods based on its numerical scheme in space can be used. In the next section we address the interpolation method based on the polynomial representation in the DG scheme.

3.3.2. Data mapping via polynomial evaluation

Data mapping in space between different coupling points at the coupling interface is a key task of a coupling tool. The Discontinuous Galerkin method is a numerical method that provides a continuous solution in space within each element which can be exploited to provide data at arbitrary points in space within this element. This is the most attractive property of the DG method in the context of coupling and is presented in the following.

In the DG method, there is a polynomial representation of the solution within an element. The approximated solution of the state vector $\mathbf{U}(x, y, z, t)$ within an element can be formulated by using a polynomial function of degree m as

$$\mathbf{U}(x, y, z, t) = \sum_{ijk=1}^{(m+1)^3} \hat{u}_{ijk}(t) L_i(x) \cdot L_j(y) \cdot L_k(z) \tag{3.2}$$

where $\hat{u}_{ijk}$ are the modal coefficients and the corresponding iterator is $ijk = i + j \cdot (m+1) + k \cdot (m+1)^2, \forall i, j, k \in \{0, \dots m\}$. L_i, L_j, L_k are the chosen Legendre functions in the dimension i, j, k. The Legendre polynomials have a recursive definition and the i-th polynomial is defined by

$$
\begin{aligned}
L_i(x) &:= \frac{(2i-1)xL_{i-1} - (i-1)L_{i-2}}{i}\,, \\
L_1(x) &:= x\,, \\
L_0(x) &:= 1\,.
\end{aligned}
$$

The polynomial order is $p = m + 1$. With this recursive definition and Equation (3.2) the solution at an arbitrary point (x, y, z) in space up to the order of the chosen polynomial can be computed. The polynomial evaluation in Equation (3.2) is a evaluation of a three-dimensional polynomial. In case of coupling, coupling data for an arbitrary number of coupling points are requested. This number depends on the numerical scheme of the other coupling subdomain (see previous section Section 3.3.1). For each coupling point we compute this three-dimensional evaluation to obtain the coupling data. In general, the cost of the polynomial evaluation depends on the polynomial order p, and the number of requested coupling points n. For the three-dimensional evaluation the resulting cost is $n \cdot p^3$. Projecting the polynomial from the volume to the coupling face is a promising optimization step, that yields a two-dimensional polynomial. Subsequently, a two-dimensional evaluation can be done for all points on this coupling face. This optimization step reduces the evaluation cost from $n \cdot p^3$ to $p^3 + n \cdot p^2$ per coupling face.

As highlighted in the next section, Section 3.4, in case of coupling to a flow domain where the Navier-Stokes equations are solved, the state variables as well as their gradients have to be provided at the coupling points. This is additional work in the coupling context and the impact of the gradient evaluation should be minimized. Therefore, an important optimization step of the implementation of this gradient evaluation is presented in Appendix A. Without this optimization step, coupling to a Navier-Stokes domain would be highly computationally demanding and would decrease the performance benefits of the partitioned coupling approach.

Looking at interpolation methods based on geometric data, e.g. the radial basis functions, a global interpolator is built first by using an

equation system that needs to be solved; then the interpolator needs to be evaluated. The cost of the polynomial evaluation depends on its degree. If the polynomial degree is the same, the evaluation cost of a DG polynomial, a RBF polynomial or a Legendre polynomial can be considered be similar. Compared to nearest neighbor injection or nearest projection, polynomial evaluation is clearly more expensive. Which quality of the different interpolation methods is needed depends on the expected coupling data, e.g. matching coupling points at the coupling interface only need first-order mapping, whereas for highly non-matching points a higher-order interpolation should be used. The benefit of direct evaluation gains importance when using a higher order in a subdomain, e.g. in the acoustic far field. Coupling a high-order solution using a low-order interpolation method for the data mapping reduces the overall numerical order and can lead to instabilities. However, there could be cases where a higher order is coupled to a lower order, where the lower order is chosen due to stability reasons or special characteristics of an application. In such cases, it can be sufficient to map data in space with a reasonably lower order to reduce the coupling cost. Limiting the polynomial order in the DG scheme for the direct evaluation of the coupling data, depending on the coupling scenario, could be a promising optimization step in the future.

3.4. Fluid dynamics in the context of coupling

Partitioned coupling opens up the possibility to solve different equations in individual subdomains. For fluid dynamics, the governing equations are presented in Section 2.1. These equations (Equation (2.11), Equation (2.12), Equation (2.15)) describe different physical phenomena of fluid dynamics but are based on the same state vectors.This is beneficial for coupling different equations where coupling variables need to be exchanged. Hence, it is important to know which variables to exchange between subdomains and how to compute them.

In general, the variables of the state vector $\boldsymbol{U}$ have to be exchanged. Here, the conservative state vector $\boldsymbol{U} = (\rho, \rho v_x, \rho v_y, \rho v_z, e)$ as well as the primitive state vector $\boldsymbol{U_{prim}} = (\rho, v_x, v_y, v_z, p)$ are available, where Equation (2.13) defines the transformation. When coupling the same equations, the same type of state vector should be used to avoid additional computation: For our description of the equations this means that we exchange the conservative state vector for Navier-Stokes equations and the Euler equations, and the primitive state vector for Linearized Euler

equations. Even when coupling different equations, Navier-Stokes with Euler, we exchange the conservative state vector since both equations are formulated in conservative variables. Additionally, the gradient of the state vector is required to solve the Navier-Stokes equations as indicated in Equation (2.11). In case of coupling to a subdomain with Navier-Stokes equations, the delivering subdomain has to provide the gradients of the state vector $\boldsymbol{U}$ at the coupling interface for the Navier-Stokes subdomain. In this work, the Linearized Euler equations is formulated in primitive variables (Equation (2.15)). When coupling Euler equations with Linearized Euler equations, the solver of the Euler subdomain has to convert the conservative state variables to the primitives variables by using Equation (2.13) to provide them to the Linearized Euler subdomain and vice versa.

One challenging question of coupling different equations is the location of the coupling interface. In case a physical phenomenon requires diffusive and viscous effects, computing the Navier-Stokes equations is necessary. This is, for instance, the case when computing a shear layer or vorticies. Considering turbulent flow, the viscosity influences the transition from laminar to turbulent flow. Viscous effects are responsible for vorticies to dissipate and trigger the energy cascade. Regarding the Navier-Stokes equations (2.11), the stress tensor (including viscosity) only acts on the gradient of the velocity. Hence, the gradient of the velocity can be used as a measurement for the influence of viscosity in a flow. When using DG to solve the Navier-Stokes equations, the viscous numerical flux $\boldsymbol{u}^*, \boldsymbol{\sigma}^*$ of Equation (2.34) is computed and can, hence, be used as an indicator of the local influence of viscosity. When viscous effects can be neglected it is legitimate to switch from Navier-Stokes to Euler equations. A switch between Euler and Linearized Euler equations is only valid when linearization of the flow is allowed. Linearization assumes that flow quantities can be split into background flow and perturbation, and that the perturbation quantities are much smaller than the background flow. This is the case, for example, when no vortices occur in the flow. Hence, the vorticity can be used as a measure of non-linearity of a flow. Furthermore, the perturbation magnitude can be an indicator. Schwartzkopff describes the coupling between Euler and Linearized Euler equations in a conservative way, that is, the flux functions at the coupling interface are constant [36]. In such a case the propagation speed at the interface changes from non-linear $\mathbf{u} + \sqrt{\gamma p/\rho}$ to linear $\mathbf{u}_0 + \sqrt{\gamma p_0/\rho_0}$, where the subscript "0" denotes the constant background quantities. The change in propagation

speed can be interpreted as two materials with different propagation speed where the coupling interface is the material-border. Schwartzkopff uses the fundamentals of optics, like the angles of incidence and refraction, to show that this can lead to reflections. Hence, to avoid reflection the variance of local speed of sound can be taken as indicator for the switch between equations.

For the studies in this thesis, we choose the location of coupling interfaces according to these considerations. The main focus is the development of partitioned coupling approaches. Solving the computationally demanding physics, like Navier-Stokes equations, only in the narrowed space where it is actually required will improve the speed-up of the partitioned coupling approach for a dedicated testcase.

4. Numerical framework

In this chapter, we present the numerical framework that is further developed with this work. The end-to-end parallel simulation framework *APES* (Section 4.1) with the high-order Discontinuous Galerkin solver *Ateles* (Section 4.2) forms the basis. Detailed descriptions of *APES* can be found in [5, 65]. For further explanations of the high-order solver *Ateles* and implementation details the reader might refer to Zudrop [8]. In the context of this work, i.e. to solve aero-acoustic simulations, the DG solver *Ateles* is extended to compute the Linearized Euler equations (Section 2.1.3). To enable a multi-solver coupling approach, calls to the API of the external coupling library *preCICE* are implemented into *APES*. Details about *preCICE* and how the multi-solver approach handles coupling are described in Section 4.3 and [39–41]. Subsequently, the integrated coupling approach *APESmate* and its approach to the coupling tasks are presented in Section 4.4. The development and implementation of *APESmate* are joint work together with Kannan Masilamani and the first results are published in [66]. Additionally, an important optimization step within the implementation of *Ateles* for the evaluation of coupling data is discussed in Appendix A. In the context of high performance computing, the performance on a massive parallel system (SuperMUC, LRZ, Munich) for both coupling approaches are presented.

4.1. Simulation framework *APES*

APES is an end-to-end parallel simulation framework that is designed to take advantage of massively parallel systems available in supercomputing today. As such, it provides scalable solvers as well as pre- and post-processing tools. Figure 4.1 gives a schematic overview of the *APES* framework. All computational components of *APES* are implemented based on the common mesh library *TreElM* [67], which is developed and distributed as open-source software[1]. The *TreElM* library relies on an octree representation of computational meshes and provides a distributed neighborhood search within them. A space-filling curve is used for the

[1] `https://osdn.net/projects/apes/scm/hg/treelm/`

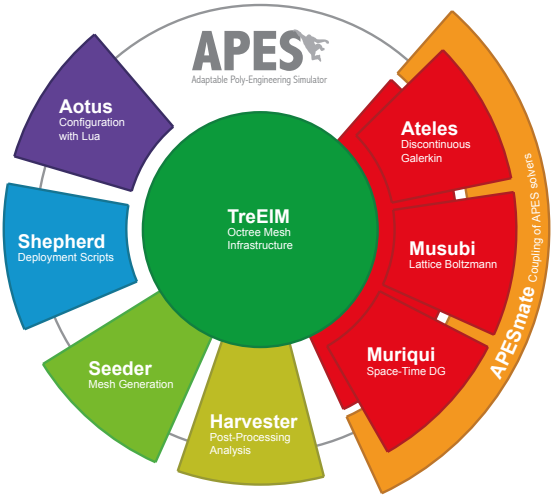


Figure 4.1.: Schematic organization of the *APES* framework.

domain decomposition of the octree mesh, which maintains data locality. Since stencil-based solvers require neighborhood information during computation, the octree structure of the mesh and its locality of the data help to reduce communication during computation which is essential for parallel efficiency. Even the high-order Discontinuous Galerkin solver *Ateles* only requires data from direct neighbors such that it can perfectly exploit the locality property. An crucial feature of *TreElM* for this work is the load balancing algorithm *SPartA* [55], which is further described in Section 4.2.1. *Ateles* requires a computational meshes in the octree format for operation. that is generated using the mesh generator *Seeder* [68]. *Seeder* also provides geometries in a high-order representation of their surfaces in the octree meshes, which is required for high-order solvers like *Ateles* as low-order surface representations would diminish the accuracy of their solution [69].

The post-processing tool *Harvester* reads binary output data from the aforementioned solvers and converts them into a multitude of visualization formats, in parallel. Hence, binary data is output during computation and we can post-process the results in a subsequent step using a different number of processes. Usually, visualization tools like ParaView or TotalView working on cell or point data, and the visualization of polynomial data

that *Ateles* is using is not intuitive. Therefore, *Harvester* is closely related to the numerical solver as it includes special treatment of the polynomial representation of *Ateles*, e.g. a higher refined visualization mesh represents better the features of the polynomial solution. A convenient feature of *Harvester* is the adaptive post-processing where the visualization mesh is based on the actual solution of the polynomial and not just on the Cartesian mesh. Hence, areas where the polynomials vary more are better resolved and not unnecessary memory is consumed.

4.2. High-order Discontinuous Galerkin solver *Ateles*

The high-order solver *Ateles*[2] uses a modal DG method in space (Section 2.2.1), while classical explicit RK method are employed in time (Section 2.2.2). Hence, simulations with *Ateles* are limited by the so-called *CFL condition* described in Equation (2.43) and Equation (2.44). The DG scheme is a numerical method to solve partial differential equations and is applicable to a broad set of problems. It is based on a polynomial representation within each element and a flux calculation between elements. Hence, there is a strong linking of data within each element and only a loose linking between elements via their surfaces. The choice of the polynomial degree defines the spatial discretization order so that by choosing a high degree a high-order method is constructed.

A high-order scheme has several advantages: First, it shows high convergence rates in case of smooth solutions. Hence, a high-order approximation provides high accuracy with a limited number of degrees of freedom (*DoF*). Fewer *DoF* equate to smaller amounts of memory, which is essential since memory is an expensive resource that can limit scalability. Secondly, a high-order scheme yields low numerical dissipation and dispersion errors [6], which is advantageous for approximating a wave propagation over long distances in the acoustic far field. We choose the Legendre Polynomials as basis functions of the DG scheme in *Ateles*. Exploiting such a modal basis has numerical as well as computational reasons. Orthogonality and the recursive definition of Legendre Polynomials lead to the fast evaluation of mass and stiffness matrices while showing good conditions. Moreover, the numerical flux of a linear problem can be directly evaluated in modal space. The cubical elements provided by the mesh generator *Seeder* can be computed without expensive transformations to a reference element. Then, using a tensor product formulation in the DG scheme, a dimension-by-

[2] `https://www.apes-suite.org/pages/ateles`

dimension algorithm can be deployed. However, some operations require a nodal representation as they can not be computed in the modal space: a) The non-linear flux computation for Euler and Navier-Stokes equations in every timestep; b) applying boundary conditions; c) applying initial conditions or source terms to the PDE. Therefore, different modal-to-nodal transformations have been implemented: The fast polynomial transformation (FPT) by Alpert and Rokhlin [70], direct projection via numerical quadrature (also called L_2 projection), and a spherical harmonic transform using fast multipole method as implemented in the FXTPACK library. Details on each method and investigations for hybrid parallelization can be found in [64].

Whether we consider linear or non-linear equations influences the workload per timestep due to this modal-to-nodal transformation. For linear equations without the need for transformation, the work per timestep is $\mathcal{O}(nElems \cdot p^d)$, where $nElems$ is the total number of elements, d is the number of dimensions, and p denotes the spatial order. In case of a cubic domain, increasing p or refining the mesh size h (to increase $nElems$) will result in the same additional costs. For non-linear equations, where a transformation is required, the workload is impacted by the cost of the transformation. Using direct numerical projection together with the dimension-by-dimension algorithm reduces the number of iterations from $\mathcal{O}(nElems \cdot p^{2d})$ to $\mathcal{O}(nElems \cdot d \cdot p^{d-1} \cdot p^2) = \mathcal{O}(nElems \cdot p^{d+1})$ [64]. Using the fast polynomial transformation, this complexity can also be recovered for smaller polynomial degrees p, whereas for large degrees this projection shows an asymptotic behavior of $\mathcal{O}(nElems \cdot p^d \log p)$. This asymptotic behavior is also promised for the fast multipole method for sufficiently high orders. Hence, in our implementation, the high-order scheme implies an increased computational cost for non-linear equations, but for linear systems like the acoustic equations, a modal scheme keeps the computational effort per DoF constant over increased spatial orders and solves them efficiently. When coupling a linear system like the Linearized Euler equations, the modal-to-nodal transformation is required to obtain the coupling points at the coupling interface (see Section 3.3.1). For non-linear equations, e.g. Navier-Stokes and Euler equations, this transformation is performed anyway.

In our implementation of modal DG, the spatial order of the scheme and, thus, the polynomial degree p can be chosen arbitrarily [6]. The number of degrees of freedom (DoF) per element is computed as $DoF = (p+1)^d \cdot nVars$, where $nVars$ is the number of conservative variables in the equation system.

The numerical resolution is dictated by the physical scale that should be resolved. However, applying high-order methods to problems with discontinuous solutions or shocks might lead to unphysical oscillations, the so-called Gibbs phenomena [8]. In particular for non-linear terms, these arbitrary oscillations can grow to the point where they impair the pointwise solution and convergence. Thus, for some problems it is reasonable to choose a lower order and a more refined mesh. Secondly, the spatial order depends on the parallelization and the system to run on.

While the volumetric data within each element is tightly linked and many operations require different access patterns to the data and the interaction between elements happens purely on the faces. Thus, only planar data on the faces need to be exchanged between elements. With this, the scheme offers a natural separation into two levels of parallelism, which we exploit by using OpenMP parallelism for operations within elements and MPI communication between elements. In general, *Ateles* can scale down to a single element per MPI-rank. However, computing only one element per CPU reduces the options for distributing the elements and, therefore, limits the potential of optimal load balancing. Furthermore, the DG method exhibits a strong data coupling within an element due to the polynomial representation. To get the most out of this data locality, the size of an element and the order of the polynomial p should be chosen so that the data fits into the cache of the system. Since the cache is far smaller than shared memory, several elements should be computed per MPI-rank for optimal usage of the resources.

To overcome stability issues, different modal filters can be applied. This is another advantage of the modal scheme since oversampling, which is used for de-aliasing and co-volume stabilization, is easy to apply. More details on stabilization approaches can be found in Zudrop [8]. The polynomial representation of the DG method also has an advantage in the coupling context. For the data exchange at the coupling interface, the polynomial representation can be evaluated at any point on the surface up to a chosen order of the method. In a multi-solver approach, where solvers are treated as *black-box*, the quadrature points of the polynomial on the coupling surface are utilized as coupling points, as explained in Section 3.3.1.

The DG method can be used to solve a large set of partial differential equations. In *Ateles* the following set of equations were available prior to this work: Flow (compressible Navier-Stokes and inviscid Euler equations), electrodynamics (Maxwell equations), as well as Heat equation. For the

purposes of this work, *Ateles* was extended to solve the Linearized Euler equations.

4.2.1. Load balancing in *Ateles*

Detailed information about the load balancing algorithm *SPartA* can be found in [55]. We will summarize the main points for the reader: An arbitrary mesh with *nElems* elements shall be distributed equally among P processes. In other words, the number of elements per process shall be balanced. The workload of element might be differ due to different task of an element, for example an element that is involved in boundary condition. Therefore, each element is assigned a weight W_i, where i is the global index of the element, which corresponds to the actual workload during simulation. *SPartA* uses these weights to re-distribute the elements such that the workload per process is balanced instead of the number of elements per process. The ideal workload per process, W_{opt}, is

$$W_{opt} = \frac{W_{sum}}{P} \tag{4.1}$$

where $W_{sum} = \sum_{i=1}^{N} W_i$ denotes the total workload. Each process has a unique rank identifier r, $0 \leq r < P$. To identify which elements has to be moved to which processes, the prefix dependency on the element is introduced

$$prefix(I) = \sum_{i=0}^{I-1} W_i \tag{4.2}$$

where $0 < I \leq nElems$ and $prefix(0) = 0$, by definition. After re-partitioning, each process should have the elements with the $prefix(r)$ with the range $[r \cdot W_{opt}, (r+1) \cdot W_{opt}]$. By comparing $prefix(I)$ with the target range, each process can determine to which process each of its elements has to be moved. An important aspect is, that this re-partitioning procedure follows the space-filling curve (SFC), which is used in *Ateles*, for the domain decomposition of the octree mesh to maintain data locality. To identify the new partitions *SPartA* uses so-called splitters, to mark the ideal splitting positions along the SFC to match $prefix(I)$ with aforementioned target ranks. The ideal splitter are multiple of W_{opt}. This algorithm is implemented in the octree mesh library *TreElM*. Examples of this procedure are presented in [55].

In order for this approach to work, solvers have to provide weights per

element to *SPartA*. The weights calculation for the LBM solver *Musubi* is discussed in [71]. Here, we present the timings-based calculation of weights for *Ateles*. Please note, that initialization routines are explicitly not included in the weights. By default, *Ateles* distributes elements equally among processes. To appropriately re-partition these elements, the workload per element has to be determined and weights have to be computed per element. To determine the workload per element, the compute kernel for all elements on a process is clocked. Hence, the weight for the computation is $W_{comp} = \frac{T_{comp}}{N_p}$, where T_{comp} is the timing for the compute kernel and N_p the number of elements per process. As discussed earlier, partitioned coupling introduces additional work for the elements which are involved in coupling. In order to express this workload per element, coupling point evaluation weights W_{eval} are introduced, which are measured directly per element. For both implemented approaches, *APESmate* and *preCICE*, evaluating coupling data is the same routine except for the number of coupling points. So, we measure the polynomial evaluation for state and gradient evaluation to obtain coupling point evaluation weights. For the multi-solver approach with *preCICE*, there are additional library calls like `read_from_precice`, `write_to_precice` and `advance_precice` (see Section 4.3.2) that cannot be resolved per element, since it is a single library call per process. Nonetheless, the element weights in *Ateles* can be calculated as

$$W_i = \begin{cases} W_{comp} + W_{eval}, & \text{if element is a coupling element} \\ W_{comp}, & \text{otherwise} \end{cases}$$

As described in Section 3.2.2, computational and coupling workload should be considered together for a balanced simulation. The user has to provide weights to *SPartA* via a so-called weight file. When configured accordingly, solvers output their weights after simulation so that this file can be used as starting distribution for the next simulation.

SPartA preforms a reasonable LB when considering elements with similar workloads but it is limited with $\frac{W_{comp}}{W_{opt}}$, where W_{opt} depends on the number of processes. In case of a coupled simulation, the weights of coupling elements can be significantly higher though. Figure 4.2 described different scenarios of re-partitioning when high-weight elements occur: The black lines is the vector of elements along the SFC and S defines the splitter positions based on W_{opt} for the partitioning. Each box is an element where the size along the line defines the weight of this element. Gray boxes are elements with an outstanding load. In scenario **a)**, all elements

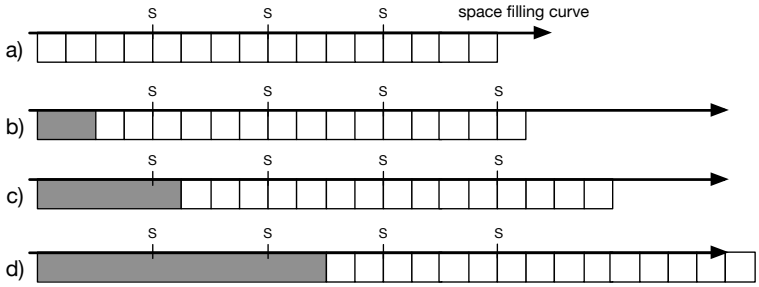


Figure 4.2.: Schematic of splitting with *SPartA*.

have the same load and *SPartA* can easily partition the SFC at the shown splitter positions. **b)** Assuming one gray element with $W_i < W_{opt}$, this element still fits between two splitters and even two more elements can be located onto that partition. In case of **c)**, the gray element has a load which is slightly higher than the optimal weight W_{opt} for this number of partitions. It is not optimal, but *SPartA* can locate only this element on a partition. This "overshoot" can be compensated with the cheaper elements of the next partition. An even worse case is, when several gray elements are located back to back in the SFC vector: This, however, should be avoided with the chosen Z-curve in *TreElM* [5]. The last scenario **d)** illustrates the case that the gray element is twice as expensive as the optimal weight, $W_i > 2W_{opt}$. This results in one element spreading over two splitter locations which hamper the re-partitioning of the SFC.

Besides load balancing within a subdomain with *SPartA*, we uses these weights for load balancing between subdomains, i.e. to chose the number of processes per subdomain based on actual work. Based on the sum of the weights, we approximate the different workload between the subdomains. The entire procedure is described for a large-scale simulation in Section 5.2.4.

4.3. Multi-solver approach: *preCICE*

For the multi-solver approach, we focus on using the solvers as *black-boxes* which means that a solver's input and output values are only accessible via their provided interfaces. Therefore, the tasks of the coupling library,

i.e. *steering* individual solvers, *communication of data*, and accurate *data mapping* between non-matching interfaces, are more challenging. We first give an overview of the coupling library *preCICE* [39], which is followed by a description of the treatment of the aforementioned coupling tasks. Then, a summary of available interpolation methods is presented. For more details on interpolation, the reader is referred to [72] and [42]. Afterwards, a brief investigation of interpolation methods in the high-order solver *Ateles* is done, where Section 5.1 adds mode details. Finally, a performance investigation is presented.

4.3.1. Overview

The open-source coupling library *preCICE*[3] offers approaches for the coupling tasks presented in Section 3.1, while allowing for a minimally invasive integration into existing solvers [39]. Additionally, for implicit coupling, which is not part of this work but is a key feature of *preCICE*, efficient solvers for fixed-point equations derived from coupling conditions are available. The major tasks of the coupling device need to work efficiently and should be scalable for distributed computations. In [42] and [43], developments and achievements of *preCICE* working on distributed data are presented. [40] and [41] give an even more detailed description of *preCICE*. Flexibility is the key benefit of using a coupling tool like *preCICE*. The application programming interface (API) is concise and enables easy coupling of individual solvers. Additionally, it implements several sophisticated coupling methods, which are required to improve numerical stability and accuracy at the coupling interface. The advantages are only clouded by the decrease in parallel performance due to the generality of a *black-box* approach.

4.3.2. Steering of individual solvers

The steering of individual solvers by *preCICE* is done via an API through which the solver and coupling library interact. The main steering procedures that need to be called from within a solver are listed in Code 4.1.

[3] `www.precice.org`

Code 4.1: Main steering procedures of the *preCICE* API.

```
def: initialize_precice()
def: initializeData_precice()
def: advance_precice(computedTimestepLength)
def: finalize_precice()
```

The *initialize* procedure sets up data structures and master communication channels. Afterwards, the coupling meshes are communicated and, if required, the re-partitioning of the meshes and initialization of *point-to-point* communication is performed. Moreover, *initialize* returns the maximum timestep length for the solver to reach the next synchronization timestep. Subsequently, *initializeData* can be used to communicate initial conditions from one subdomain to another. This is important for solvers with explicit time integration since each subdomain should start with valid conditions at the coupling boundaries. If this routine is not used, default "zero" boundary conditions are assumed. *advance* is called after the computation of every timestep. Here, *preCICE* applies mappings schemes, communicates the coupling data, and computes fixed-point acceleration techniques for iterative coupling methods. Additionally, the time steering is performed by taking the last timestep as the input argument. *preCICE* checks for reaching the synchronization timestep or indicates sub-cycling of one solver. The return value indicates, again, the next maximum timestep size or the remaining timestep in the case of sub-cycling. Finally, *finalize* tears down data structures and closes communication channels.

Other coupling routines, which are essential but do not belong to classical steering, are *reading from* and *writing to preCICE*. Both routines interact with the local *preCICE* slaves to read and write values at the local coupling points, respectively. This does not imply communication, which is only done in the *advance* procedure. A further auxiliary procedure is *isCouplingOngoing* which allows to trigger the end of the simulation by the coupling library and to return a logical flag which is read by the solver. Code 4.2 sketches the pseudo code of *Ateles* augmented with calls to the *preCICE* API.

Code 4.2: Pseudo code of the solver *Ateles* extended to call the *preCICE* API.

```
initialize_ateles()
dt_cfl = get_cfl_timestep()
dt_precice = initalize_precice()

if (precice_actionisrequired):
        write_to_precice(initial condition)
        initializeData_precice()

dt = min(dt_cfl, dt_precice)
t=0
while t < t_max:
        read_from_precice()
        data = solve_timestep()
        write_to_precice(data)
        advance_precice(dt)
        determine_end_by_ateles
        determine_end_by_precice
        t = t + dt

finalize_precice()
finalize_ateles()
```

To enable iterative coupling with fix-point iterations in *preCICE* the API needs further control procedures. Please refer to [41], as this is not implemented in this work.

For coupling with *preCICE*, the different executables for the solvers involved in the coupling must be linked with *preCICE* and all executables must be simultaneously started. Starting such a job on supercomputers is more involved: When starting several executables, MPI-ranks have to be correctly bound to the corresponding executable to avoid running multiple ranks on the same CPU. Furthermore, concepts and scripts to check if one simulation might have crashed need to be established to maintain a consistent state and to finalize properly. Hence, porting software, establishing the correct binding of MPI-ranks, and compiling job scripts on a supercomputer is more challenging compared to running a single application.

4.3.3. Communication of coupling data

As already mentioned, efficient *point-to-point* communication is a prerequisite for large-scale simulations. Here, we consider the initialization phase and general communication in each synchronization step from the perspective of the external coupling library *preCICE* and focus on basic concepts and outcomes. More details and performance tests can be found in [42].

The major challenge is to establish the communication relations between the processes of the subdomains. Assuming the *black-box* approach, *preCICE* only works on geometric data as input/output data from each subdomain. For the initialization of the communication, *preCICE* exploits a *Master-Slave* communication concept: All coupling interfaces of individual subdomains are exchanged via their master processes. Subsequently, the coupling interface is re-partitioned to the corresponding ranks according to data mapping, i.e. each slave receives its portion of the interface to be able to do the data mapping.

The algorithmic steps for one-way coupling with two subdomains A and B with the corresponding coupling interfaces Γ_A and Γ_B (as noted in [42]) are:

I. The master rank of B gathers coupling interface Γ_B from all ranks of B.

II. Γ_B is communicated from the master rank of B to the master rank of A.

III. The master rank of A broadcasts Γ_B to all ranks of A.

IV. Each rank of A filters Γ_B according to its partition of Γ_A and a defined mapping between both meshes.

V. Master rank of A gathers distribution information from all ranks of A.

Hence, after broadcast II and gathering distribution information V, the master rank of A holds two partitioning descriptions of the coupling interface Γ_B, i.e. all coupling surface mesh points and the corresponding processes in subdomain A. But with step IV every rank of A knows which coupling points it has to communicate with whom. For all further communications, an M:N communication (built from multiple 1:N communications) between the ranks of A and B is established. These 1:N communications

are implemented either with TCP/IP (based on `Boost.Asio`[4]) or with MPI-2.0 (or later) ports, which can be chosen at run-time. For two-way coupling this algorithm is also executed to communicate distribution information from subdomain A to the ranks of subdomain B. While the initialization still relies on global operations, all necessary information is completely local afterwards and the communication in every *advance* step is a purely *point-to-point* between the processes. One major drawback of the *Master-Slave* concept is high memory consumption due to the exchange of entire coupling interfaces with distribution information. Even with the minimal storage requirements of surface coupling, this contradicts exascale computing. Hence, this poses a major bottleneck that needs to be addressed in future versions.

4.3.4. Data mapping in time

preCICE provides several fix-point iteration schemes for iterative coupling (sometimes referred to as implicit coupling). This means performing an iterative process of solver evaluations in every timestep until convergence of coupling values at the interface is achieved: This might allow for time-consistent coupling. However, when iterative coupling is not used, *preCICE* does not provide higher-order time representations. Therefore, in this work, we do not allow for sub-cycling of any participant which means that we have to force all subdomains to use the smallest timestep. In most cases, the synchronization timestep in the *preCICE* configuration is chosen so that it provides the smallest timestep.

4.3.5. Data mapping in space: (Interpolation) methods

As mentioned, *preCICE* is acting purely on geometric data. When coupling non-matching interfaces as depicted in Figure 5.15, mapping between coupling points becomes necessary. *preCICE* provides two standard mapping methods: Projection-based mapping and radial basis function interpolation. The two solvers for subdomains A and B provide the interpolation values $U_A \in \mathbb{R}^{N_A}$ on the coupling interface Γ_A and $U_B \in \mathbb{R}^{N_B}$ on the coupling interface Γ_B, respectively. A general interpolation from Γ_B to Γ_A can be written as

$$U_A = H^{AB} U_B \ ,$$

with the mapping matrix $H^{AB} \in \mathbb{R}^{N_A \times N_B}$. There are two different kinds of mappings: *Consistent* and *conservative* [73], where only the first mapping

[4] `www.boost.org`

is considered in this work. A *consistent* mapping denotes that the entries of every row sum up to 1, which guarantees an exact mapping of constant functions. *Conservative* mapping, in contrast, defines the column-sum to be equal to 1. In the following we briefly summarize the interpolation methods available in *preCICE*. More information can be found in [42, 72].

4.3.5.1. Projection-based mapping

There are two different projection-based mapping methods available in *preCICE*: Nearest neighbor (NN) and nearest projection (NP). While the first mapping is first-order, the latter one is second-order if the projection distance from one mesh to the other is much smaller than the mesh width. In practice, this typically holds [41].

For the NP mapping, the solver needs to provide mesh connectivity information on the coupling interface. This is not the case for the NN mapping. Figure 4.3 depicts a schematic view of both projection-based

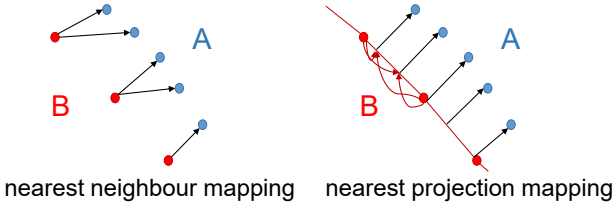


Figure 4.3.: Schematic view of the two projection-based mapping methods provided in *preCICE* in a two-dimensional case: Nearest neighbor mapping and nearest projection mapping. Arrows denote data transfer between points of B and A.

mapping methods: The coupling interfaces Γ_A and Γ_B with multiple coupling points are depicted in *blue* and *red*, respectively. The NN mapping is a first-order mapping where the closest neighbor point among all Γ_B points is found and the respective value is copied to the Γ_A point. Thus, the NN mapping is purely injective. NP in contrast, does a linear interpolation on the Γ_B points based on the provided connectivity information from B before sending this value to the corresponding Γ_A point (blue). NN is useful for coupling matching interfaces, where the coupling points coincidence and a higher-order interpolation could introduce numerical errors and would be unnecessarily expensive. For non-matching interfaces where connectivity

information on the coupling interface is provided, NP mapping on the other hand can be suitable since no further tuning parameters are required.

4.3.5.2. Radial Basis Function interpolation

A more general mapping method, without the need for any mesh connectivity information, is the second-order accurate Radial Basis Function (RBF) interpolation method. Here, the influence of all coupling points depends on their distance from the interpolated point. To map values from Γ_B to Γ_A, a global interpolant on a point on Γ_B is generated and then evaluated at the coupling points on Γ_A.
The interpolant $a : \mathbb{R}^3 \to \mathbb{R}$, reads

$$a(x) = \sum_{i=1}^{N_B} \gamma_i \cdot \varphi(\|x - x_i\|) + g(x) \ ,$$

with the radial basis functions φ centered at the (coupling) point x_i of Γ_B. There are several types of radial basis functions implemented in *preCICE*, e.g.

$$\begin{aligned} \text{Gaussian} \quad & \varphi = \exp\left(-(s\|x\|)^2\right), \\ \text{Thin Plate Splines} \quad & \varphi = \|x\|^2 \log\left(\|x\|\right), \\ \text{Multiquadrics} \quad & \varphi = \sqrt{s^2 + \|x\|^2}, \end{aligned}$$

where s is the so-called shape parameter and $\|x\|$ is the Euclidean distance of the evaluation point from the origin of the basis function. These types have global support, but also basis functions with local support are available. Decreasing the support radius should improve the solver's convergence, but decrease the approximation quality. To ensure the exact interpolation of constant and linear functions this basis is enriched with a global linear function $g(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$ (3D case). The interpolant $a(x)$ and the linear global function $g(x)$ imply the interpolation condition

$$a(x_i) = \omega_i^B \quad \forall i = 1 \dots N_B$$

for the functions coefficients $\gamma_i \in \mathbb{R}$ and $\beta_i \in \mathbb{R}$ where ω_i^B denotes the respective value at point x_i on Γ_B; as well as the polynomial condition

$$\sum_{i=1}^{N_B} \gamma_i \cdot q(x^i) = 0$$

for every polynomial q. In combination the following matrix notation can be derived

$$\left(\begin{array}{c|c} 0 & Q^T \\ \hline Q & P \end{array}\right) \left(\begin{array}{c|} \beta \\ \hline \gamma \end{array}\right) = \left(\begin{array}{c|} 0 \\ \hline \omega \end{array}\right) , \tag{4.3}$$

where $P \in \mathbb{R}^{N_B \times N_B}, P_{i,j} = \varphi\left(\|x_i - x_j\|_2\right)$ and where the ith row of $Q \in \mathbb{R}^{N_B \times 4}$ looks like $(1\, x_{i,1}\, x_{i,2}\, x_{i,3})$ [42].

After solving this system, the interpolant can be evaluated at the point y_j on Γ_A by

$$\omega_j^A = a(y_j) = \sum_{i=1}^{N_B} \gamma_i \varphi(\|y_j - x_i\|_2) + g(y_j) \;\; \forall j = 1 \ldots N_A \; . \tag{4.4}$$

Solving the system (4.3) on distributed data, *preCICE* uses *PETSc*[5], a toolkit for parallel solution of scientific applications that provides solvers for linear systems.

In this work, we will only consider Gaussian basis function interpolation in *PETSc* [74]. The basis function can heavily influence the condition of the mapping matrix (4.3). Therefore, the function as well as the shape should fit to the distribution of points on the coupling interfaces. The condition number of the mapping matrix increases exponentially with a [75, 76]. Solving the system, the condition of the matrix plays an important role in linear system solvers like *GMRES*, *ILU*, or *Jacobi*. The condition of the system matrix needs to be sufficient, so that the system can be solved in an acceptable amount of iteration steps. The full behavior of the linear solver of *PETSc* and the shape parameter of the Gaussian RBFs for the coupling of non-matching interfaces is still under investigation. A more detailed study [77] points out that a non-equidistant point distribution leads to instability and poor convergence properties of the system. As presented in Section 3.1.4, the DG scheme provides non-equidistant points at the interface. Hence, the radial basis functions are a sub-par choice for coupling non-matching interfaces as shown in next Section 4.3.5.3. One circumvention would be to provide equidistant points for the interpolation from the solver to *preCICE* and use a solver-internal interpolation from non-equidistant to equidistant points. This might stabilize the RBF system but would lead to a higher computational effort on the solver side [78].

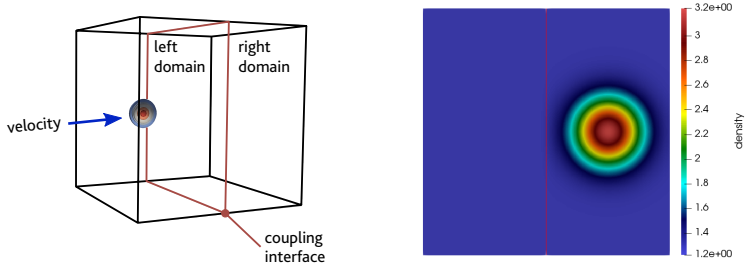
[5] `https://www.mcs.anl.gov/petsc/`

In this work the following equation for choosing the shape parameter for Gaussian basis functions is applied:

$$s = \frac{\sqrt{-\ln 10^{-9}}}{np \cdot \Delta x}, \tag{4.5}$$

where np is the number of points included in the interpolation at one specific coupling point and Δx is the distance of the coupling points. Due to the non-equidistant points at the interface, we use the maximum distance of coupling points to calculate the corresponding shape parameter s as suggested in [79].

4.3.5.3. Comparison of the interpolation methods



(a) Sketch of simple showcase with Gaussian density distribution travels from left to the right with constant speed.

(b) Monolithic solution of Gaussian pulse in density (xy-plane) at time $t = 0.008$ s; red line depicts location of coupling interface.

Figure 4.4.: Testcase of traveling Gaussian pulse in density.

Here, we illustrate briefly the difference between first-order NN, second-order NP, and second-order RBF interpolation utilizing Gaussian basis functions: Our testcase is a 4×4 plane that is split orthogonally to the x-axis into two equally sized halves. The subdomain on the left is initialized with a Gaussian distribution for density and the flow is set to travel with a constant velocity from the left to the right subdomain. We use the Euler equations in 2D with the material parameters: Thermal conductivity $\lambda = 1.5625e^{-2}$, isentropic coefficient $\gamma = 1.4$, and specific gas constant $R = 280.0$. The initial conditions of the flow are $\rho = 1.225$, $\mathbf{v} = \begin{bmatrix}250.00.0\end{bmatrix}^T$, and $p =$100 000. Initially, the Gaussian distribution in

density is located in the middle of the left domain and the amplitude of the pulse is set to 2.0 and a halfwidth of 0.2. The setup is depicted in Figure 4.4a. In the following investigation we use the same equations in both subdomains to exclude the influence from two different physics. Additionally, we use a bidirectional coupling to allow interactions in both directions to be as close as possible to the monolithic approach. For a correct interpretation of the coupling results, we compare against the monolithic solution with the same numerical discretization. Figure 4.4b presents the monolithic solution for the depicted scenario of the Gaussian density pulse at $t = 0.008\,\mathrm{s}$. It is an xy-plane where the red line illustrates the location of the coupling interface for the following coupled simulations. The amplitude of the density as well as the shape of the Gauss pulse are well preserved.

In total, four different interface configurations with three different combinations of matching and non-matching interfaces are tested, where h specifies the grid size and $\mathcal{O}$ the numerical order in space:

(o) Matching interfaces
left/right domain: 4096 points ($h = 0.5$, 64 elements at interface, $\mathcal{O}(8)$)

(i) Non-matching interfaces: Same number of coupling points at the coupling interface
left domain: 4096 points ($h = 0.5$, 64 elements at interface, $\mathcal{O}(8)$)
right domain: 4096 points($h = 0.25$, 256 elements at interface, $\mathcal{O}(4)$)

(ii) Non-matching interfaces: Different number of coupling points where the left interface is better resolved than the right interface
left domain: 6400 points ($h = 0.5$, 64 elements at interface, $\mathcal{O}(10)$)
right domain: 1024 points ($h = 0.5$, 64 elements at interface, $\mathcal{O}(4)$)

(iii) Non-matching interfaces: Different number of coupling points where the right interface is better resolved than the left interface
left domain: 1024 points ($h = 0.5$, 64 elements at interface, $\mathcal{O}(4)$)
right domain: 6400 points ($h = 0.5$, 64 elements at interface, $\mathcal{O}(10)$)

First (o) we maintain the same numerical resolution (same grid size h as well as same numerical order in space $\mathcal{O}$) to ensure coinciding coupling

points at the interface. To illustrate the influence of the different mapping methods, the investigation is done with non-matching interfaces according to configurations (i)-(iii). (ii) and (iii) are different since in this dedicated testcase the information is only traveling by convection in one direction, i.e. from the left domain to the right domain. Therefore, a different distribution of the coupling points on the coupling interface can influence the quality of the mapping method. Since it is guaranteed that all discretizations resolve the physical testcase correctly the errors stem from the chosen mapping method. All configurations are done with nearest neighbor (NN), nearest projection (NP), and Radial Basis Functions (RBF) interpolation utilizing Gaussian functions. Results (o)-(iii) with NN mapping are presented in

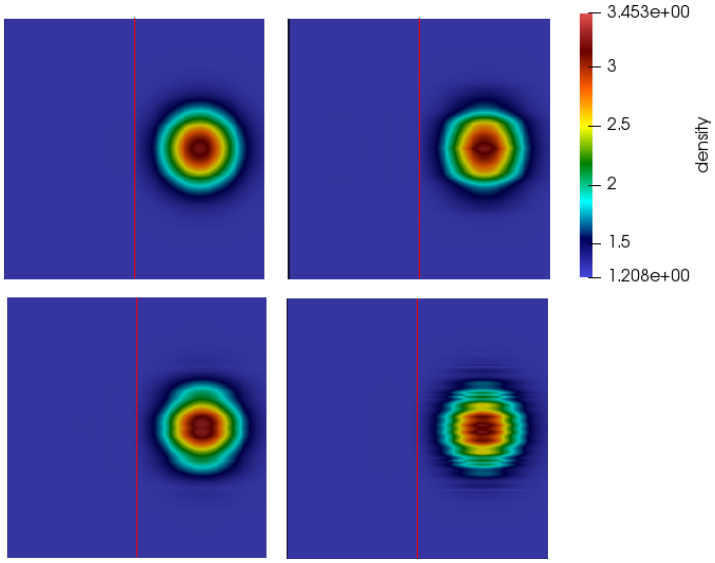


Figure 4.5.: NN mapping results (xy-plane) for configuration (o) top left, (i) top right, (ii) bottom left and (iii) bottom right showing the density; red lines denote the coupling interface.

Figure 4.5. As described in Section 4.3.5.1, NN mapping only copies data from the closest point on the mapping interface without interpolation. Therefore, the following results can be interpreted regarding the spatial location of the coupling points. Coupling matching interfaces (o), the NN mapping is similar to the monolithic simulation and does not show any

further numerical error. However, coupling non-matching interface, you can see that the different configurations (i) - (iii) lead to different results with a distorted pulse after passing the coupling interface. Maintaining the same number of points (i) leads to an elongated, drawn-out pulse in x-direction which indicates that around the origin of the pulse there are too few points on the right interface originating from the lower order. As stated in Section 3.3, in the DG method the number of coupling points at the interface is order-dependent and the points concentrate towards the element corners. Here, on the left $\mathcal{O}(8)$-interface the points are closer to the center of the Gaussian pulse than in the right $\mathcal{O}(4)$-interface, which leads to the pulse getting spread. Looking at (ii), the distortion of the pulse is slightly different: The density pulse is not heavily distorted in x-direction, instead the circular shape of the pulse is not preserved. Moreover, it looks like a split pulse center in the y-direction. In this configuration, the left interface has more points than the right interface. Here, a few points on the right interface can choose nearest neighbors among many points on the left interface. There is, however, still some kind of compression on the pulse in the y-direction. The last setup (iii), turns out to be the the worst configuration of coupling points for the depicted showcase: Since the right domain has 6 times more points on the interface than the left domain, the information from the left domain is duplicated onto the points of the right domain. This explains for example the multiple splitting of the center of the pulse and the overall distribution after passing the coupling interface. This investigation shows the immense influence of the location of the coupling points on the coupling interfaces. Hence, it can be stated that first-order NN mapping is only sufficient for matching interfaces with coinciding coupling points.

Next, we investigate the NP mapping for all configurations and show the results in Figure 4.6. For all configurations, the amplitude as well as shape of the density pulse after the coupling interface at $t = 0.008$s are well preserved. Just for configuration (iii), it looks like the shape is minimally elongated in x-direction. A deeper validation with NP mapping is done in Section 5.1. For the matching interfaces (o), the NP mapping is equal to NN mapping but with added computational effort.

Since we are using Gaussian basis functions, we have to determine suitable shape parameters s. This parameter needs to suite the corresponding interface from which the values are mapped onto the other interface. The shape parameter s is chosen in a way that a) as many mapping points as possible are used to improve the quality of the solution, and b) still

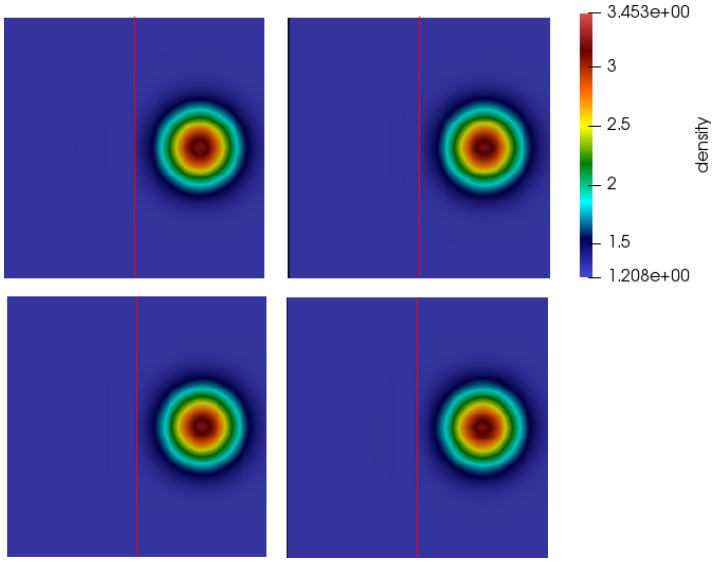


Figure 4.6.: NP mapping results (xy-plane) for configuration (o) up left, (i) up right, (ii) down left and (iii) down right showing density (a) and error(b); red line denotes the coupling interface.

achieve good condition of the system matrix (4.3) to solve the system in an acceptable number of linear solver iterations. The chosen shape parameters according to Equation (4.5) are listed in Table 4.1.

The number of points included in the interpolation *np* of Equation (4.5) is chosen to be as large as possible to still achieve reasonable convergence of the linear solvers of *PETSc*. The unsatisfying results for RBF interpolation are shown in Figure 4.7: Only the matching configuration (o) preserves the shape as well as the amplitude of the density pulse. All non-matching configurations show very distorted pulses. The worst configuration is (iii), where the shape of the pulse is not preserved at all. This scenario shows the influence of the interpolation at non-matching interfaces. In contrast to the NN mapping, where the influence of the mapping only depends on the position of the coupling points on the interface, the RBF interpolation is more involving. The choice of the shape parameter highly influences the quality of the interpolation results. Hence, a smaller shape parameter could improve the results, but for the given point distributions no smaller

shape parameter s	coupling points	grid size h	order $\mathcal{O}$	np	configurations
23.334	4096	0.5	8	2	(o) left / right (i) left
23.791	4096	0.25	4	2	(i) right
7.930	6400	0.5	10	3	(ii) left (iii) right
58.2	1024	0.5	4	1	(ii) right (iii) left

Table 4.1.: Shape parameters for different configurations.

shape parameter yield to a convergence of the linear solvers provided by *PETSc*.

Moreover, when using the modal DG scheme, we work with a non-equidistant point distribution at the interface: This is challenging for the interpolation and it is hard to pick the correct shape parameter, as most investigations are done on uniform point distributions [74, 80]. One option would be to utilize basis functions with global support since they don't require addition parameters, like the shape parameter. However, in these configurations, running simulations on distributed data and, therefore, using the linear solvers provided by *PETSc*, no converging, global basis function was found. Another approach would be to provide equidistant coupling points to the coupling library *preCICE*. This could increase the quality of the RBF interpolation but also increases the coupling cost in *Ateles* for providing these points. Results of the investigation of equidistant coupling points with *Ateles* and *preCICE* with RBFs can be found in [78]. Please note that the presented results are preliminary: More work on radial basis functions and parallel coupling of non-matching coupling interfaces is done at the Institute for Parallel and Distributed Systems at the University of Stuttgart [72]. However, the intention of this section was to show that coupling *black-box* solvers is not trivial. In particular, data mapping in space, where the coupling library only works on point clouds requires special treatment. In this investigation, we have only considered the numerical results of the interpolation methods and did not address their performance in a coupled approach.

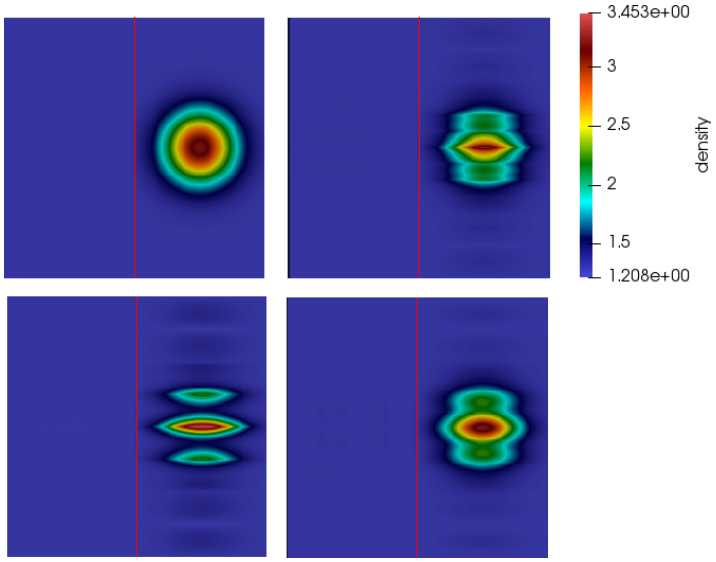


Figure 4.7.: RBF interpolation utilizing Gaussian basis function results (xy-plane) for configuration (o) top left, (i) top right, (ii) bottom left and (iii) bottom right showing the density; red lines denote the coupling interface.

Here, we briefly summarize these first conclusions for data mapping in space using *preCICE*:

- For matching interfaces always use NN mapping.
- For non-matching interfaces never use NN mapping.
- If connectivity information is provided by the solver, NP is an easy-applicable mapping with good results for non-matching interfaces.
- RBF interpolations show a promising theory, but from the user perspective it is hard to find a good basis function with suitable shape parameter s which provides a good converge of the linear system solver. These parameters are also highly dependent on the points distribution at the coupling interface.
- RBF will lead to higher computational cost due to additional computation in solving a linear system.

A thorough investigation on the data mapping in space with *preCICE* is performed in Section 5.1 where the multi-solver coupling approach is further analyzed.

4.3.6. Performance

In this section, we present the performance of multi-solver coupling with *preCICE* with NN mapping on the SuperMuc Phase 1 IBM system at LRZ, Munich. The first performance results of *preCICE* with the DG solver *Ateles* are were presented in [81] with more recent investigations on the coupling library and its performance in [41–43]. SuperMuc Phase 1 IBM system comprises a total of 9216 nodes on 18 islands with 2 Sandy Bridge-EP Xeon E5-2680 8-core processor per node resulting in 147 456 cores. The nodes are connected with Infiniband FDR10. The testcase for this scaling analysis is a 3D Gaussian pulse (similar to the one described in the previous section) with a total problem size of 8192 elements: The computational domain is split equally into a *left* and a *right* subdomain with 4096 elements each. The total problem size of 8192 elements is chosen

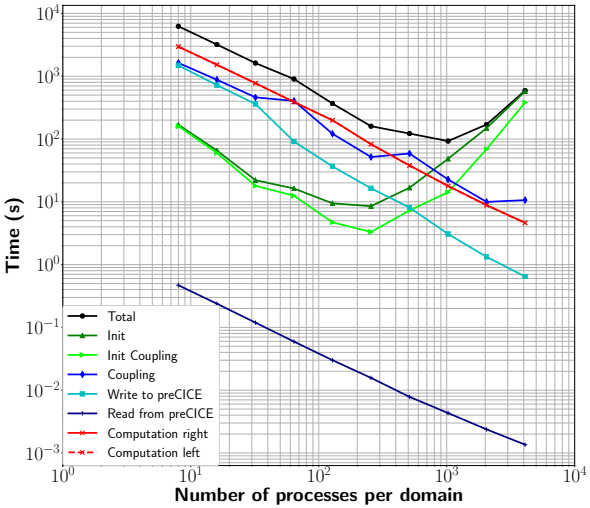


Figure 4.8.: Strong scaling using *preCICE* up to one island (= 8192 MPI ranks) on the SuperMuc Phase 1 IBM system at LRZ, Munich. Total runtime is split into individual subroutines.

such that there is at least one element per core and the polynomial order $\mathcal{O}$ is set to fit into node memory, i.e. $\mathcal{O}(20)$. The simulations are run for 100 timesteps. The *DoF* per element is 40 000, resulting in 163 840 000 *DoF* for a problem size of 4096 elements per domain. Since it is a matching coupling interface, the NN mapping is chosen for the data mapping in space. We only consider MPI parallelism for this testcase. Figure 4.8 shows strong scaling up to a single island, i.e 512 compute nodes or 8192 cores. The total run time is an averaged results to include minimum and maximum results on different processes.

The total runtime is split into the individual routines: *Initialization*, *Coupling* and *Computation*. To uncover the influence of the coupling on the whole *Initialization*, the specific coupling initialization work *Init Coupling* is measured additionally. The *Coupling* line depicts the pure coupling processes of *preCICE*. To provide data to *preCICE* and read it from *preCICE*, in every timestep two extra routines *Write to preCICE* and *Read from preCICE* are measured. For *Computation*, the measurements of the left and the right domain are plotted individually which show exactly the same behavior. The routines *Coupling*, *Write to preCICE*, *Read from preCICE*, and *Computation* show good scalability. These are the routines in the simulation phase after initialization is done. *Initialization*, in contrast, scales roughly until 128 processes per domain but takes more and more time for more processes. Looking at *Init Coupling*, it is visible that *Initialization* mainly consists of *Init Coupling*. Hence the scalability of the initialization of *preCICE* heavily impacts the initialization of the multi-solver approach. The bad scalability is mainly due to gathering and broadcasting of the complete surface meshes in the re-partitioning step explained in Section 4.3.3. The initialization phase is challenging for a coupling library which connects *black-box* solvers and works purely on input/output data This is already presented in [42]. Ideas to overcome this are presented in [43], and work in progress is reported in [72]. Beyond 1024 processes per domain, the influence of the *Initialization* phase outweighs the other phases, so that the solver *Total* doesn't scale anymore. The immense influence of the initialization phase is due to the very short simulation phase for this scaling test of 100 timesteps. Nevertheless, the initialization of the coupling tool *preCICE* as well as the solver *Ateles* should be reconsidered and optimized since all runtime phases scale well. For good performance of the coupling approach in general, the most expensive routine should be *Computation*, which is the case. Unfortunately, *Write to preCICE*, which provides coupling values to *preCICE*, is also very expensive. This routine is expected to have optimization potential and should be considered in

future software updates.

4.4. Integrated approach: *APESmate*

In this section we present the integrated coupling approach *APESmate*. This approach is implemented in the *APES* framework (presented in Section 4.1). Since the idea of integrated coupling is similar to a multi-solver approach, the same coupling tasks have to be accomplished. However, an integrated approach has several benefits from the performance point of view. First, we present a general overview of the integrated approach, discuss its potential and benefits, and follow up with implementation principles for each individual coupling task described in Section 3.1. Subsequently, a performance analysis is done.

4.4.1. Overview

The fully integrated coupling approach *APESmate* within the *APES* framework is based on the *TreElM* library so that it has full access to octree data structures, solver data, as well as parallel features. This tight integration allows for the exploitation of knowledge about internal solver data and, therefore, yields performance benefits accompanied with some limitations: This approach is less flexible since only numerical solvers within the *APES* infrastructure can be coupled. Besides the performance benefits, another advantage is solver-specific data mapping procedures. In contrast to an external coupling library that is most likely working on pure geometric data, the integrated approach can access solver specific procedures to provide coupling data directly. For example in the DG solver *Ateles*, the coupling tool can evaluate the polynomial on the required coupling surface points as described in Section 3.3. Exploiting solver specific routines is superior with respect to quality of data mapping. This is of particular importance when tethering high-order solvers like *Ateles* where direct numerical evaluation of polynomials is available. Conceptually, the solvers are invoked as a library by the coupling application so that only a single application must be handled.

Assuming no adaptive time-stepping and no change of the coupling interface, the same static load balancing based on heuristics, as presented in Section 3.2, can be applied. This is similar to the multi-solver approach but dynamic load balancing can be deployed easier. From the user's perspective, the handling of an integrated approach with one executable is facilitated. Development and implementation are joint work together with

Kannan Masilamani: More details and further implementation principles of *APESmate* can thus be found in [82].

4.4.2. Steering of individual solvers

The integrated coupling approach *APESmate* combines all included solvers into one application and invokes a solver as a library. The *steering* of the coupled simulation is direct by accessing data structures explicitly instead of providing and returning information from a library. Code 4.3 sketches the pseudo code for *APESmate*.

Code 4.3: pseudo code for *APESmate*.

```
def initialize apesmate:
        load domain_distribution
        load domain configurations
        initialize individual domains
        initialize communication

def compute apesmate:
        while t < t_max
                synchronize_domains
                dt_sync = max(domain_dt)
                solve individual domains until dt_sync
                t = t + dt_sync

def finialize apesmate:
        finalize individual domains
```

The *domain distribution* is the partitioning of the subdomains onto processes as defined in a configuration file. The user can choose between defining the number of MPI-ranks per domain or specifying the fraction of all MPI-ranks per domain. Using the fraction parameter guarantees that all domains sum to unity. *Loading* and *initialization* of the individual domain are solver-specific and the corresponding routines are directly called by *APESmate*. The initialization phase for communication will be more elaborated in Section 4.4.3. *synchronize_domains* depicts the part where coupling variables are evaluated and exchanged between subdomains at the beginning of each synchronization step. In the integrated approach, no external synchronization timestep needs to be selected: *APESmate* defines the next synchronization step internally by taking the maximum timestep size of all domains. A solver then iterates with its own timestep

limitation until it reaches this synchronization point. On a technical level, this allows for sub-cycling without additional implementation effort.

4.4.3. Communication of coupling data

The communication of coupling data at the coupling interface itself as well as its initialization is done without any global communication. More importantly, only *point-to-point* communication is used. In a fully integrated approach, the communication can be realized directly: All components are implemented in a single application which then distributes processes according to a configuration file. Starting with a global communicator, each subdomain gets its own MPI sub-communicator for subdomain-internal communication, e.g. communication of fluxes within each timestep. Therefore, the global communicator is only used for subdomain to subdomain communication. The elements of the computational domain are equally distributed over processes. The challenge is to connect these processes which have to couple points and values with each other. This is done in the initialization step. Let's assume that there are two coupling domains A and B: First, all coupling requests, i.e. all coupling points of domain A, are locally gathered. Exploiting the space-filling curves in *APES*, every process can identify on which process a specific leaf of the tree is located, i.e. every process knows on which process a defined element is located [5]. Using this property, each process of domain A sends its requested points to any process of the requested domain B. The process of domain B subsequently converts the points into the corresponding elements and identifies the processes in domain B that host these elements. The requested process of A receives this information and for the next communication, it can directly ask the corresponding process on domain B for coupling values. This first communication is done in a round-robin fashion, i.e. rank 1 of domain A asks rank 1 of domain B which is the correct rank for *point-to-point* communication. The implementation invokes several steps, such as: Using of communication buffers between the domains; storing of requested point coordinates at the initialization phase for reuse; applying an offset bit which shifts the coupling points from the surface towards the element to uniquely identify the corresponding elements; mapping between global MPI communicator; and establishing the domain-specific MPI sub-communicators.

Additionally to establishing the direct correspondence, i.e. to whom to talk to during simulation, the information "what is to be exchanged at which coupling point" is sent to the target process. This ensures that

after the initialization phase, each coupling process knows which data to provide to which process, minimizing unnecessary communication. During the simulation, coupling values have to be evaluated and exchanged at each synchronization step. This is done in the *synchronize domains* step in Code 4.3. The solver-specific evaluation is described in Section 4.4.5. As established, the communication is purely *point-to-point* and done via the global communicator.

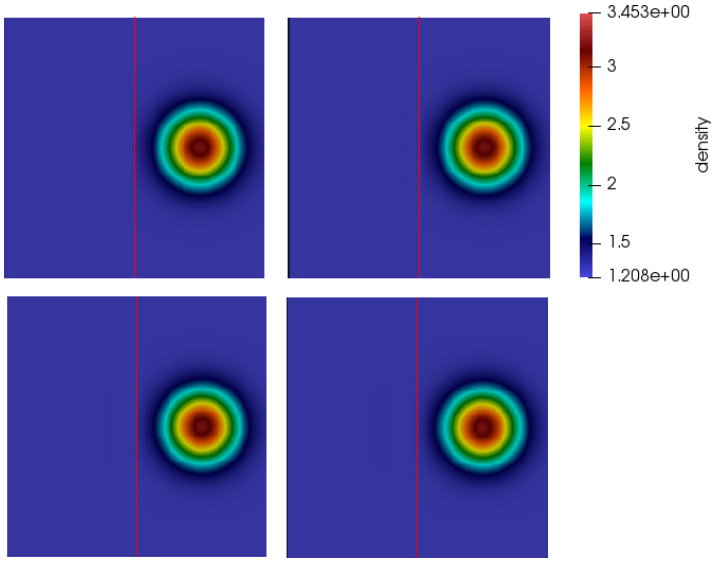
4.4.4. Data mapping in time

As already mentioned in the general description of this coupling task in Section 3.1.3 and similar to the multi-solver approach - consistent time stepping is challenging for the integrated approach as well. Here, we do not have to provide a synchronization timestep explicitly as it is required for the coupling library *preCICE*. As aforementioned, *APESmate* defines the next synchronization step as the maximum of all domain-specific timesteps so that a solver iterates with its own timestep until it reaches this synchronization step. In *Ateles*, this individual timestep is either a fixed timestep dictated by the user or an adaptive timestep identified by the *CFL* stability condition described in Equation (2.43) and Equation (2.44). However, if the synchronization timestep is larger than the individual timestep, this can result in time-inconsistent coupling, as already explained in Section 3.1.3. The same holds true for multi-step time integration: Using explicit multi-step time integration, the solver needs to provide data at each sub-step of the time integration method. This can be realized with *APESmate* when coupling the same solver, the same time integration, and the same timestep size. At the time writing this thesis, sub-cycling and multi-step time interpolation are inconsistent in time which will be more discussed in Section 6. In the scope of this work, we used fixed timestepping to guarantee that sub-cycling does not happen.

4.4.5. Data mapping in space

Within the fully integrated coupling approach, the application can access solver specific data as well as its data mapping procedure. Obviously, the solver needs to provide these procedures. Using the high-order DG solver *Ateles*, arbitrary points can be obtained through direct evaluation of the polynomial representation in the high-order scheme itself. Hence, coupling non-matching grids with different numerical resolutions, as shown in Figure 5.15, does not require additional data mapping. This is a key benefit compared to the multi-solver approach: In particular for very high

orders, a second-order NP or RBF mapping of the multi-solver approach might be insufficient, while a low order interpolation can decrease the quality of the overall solution.
Similar to the investigation on data mapping in space using the multi-solver approach with *preCICE* in Section 4.3.5.3, we examine the data mapping in space using *APESmate* on the same testcase configurations. Figure 4.9 presents the results for the configurations (o)-(iii) as defined



(a) Density pulse at $t = 0.008$s

Figure 4.9.: Coupling scenarios with *APESmate* using internal data evaluation at the coupling points. Results (xy-plane) for configuration (o) up left, (i) up right, (ii) down left, and (iii) down right showing density (a) and error (b); red line denotes the coupling interface.

in Section 4.3.5.3. All scenarios show similar results of the density pulse and shape while the amplitude of the pulse is well preserved. A complete validation of *APESmate* in 2D and 3D is presented in Section 5.1.

In the case of coupling solvers other than *Ateles* within *APESmate*, e.g. the Lattice-Boltzmann solver *Musubi*, which does not provide a polynomial representation of the solution, the solver is required to provide an interpo-

lation method using its data representation and mathematical formulation. Even if interpolation is necessary though, it is done by the data-providing solver and can make use of all the knowledge of its data structures. In general, *APESmate* is designed in a way so that surface coupling as well as volume coupling can be realized to increase the range of applications: For instance, the coupling of multi-component flow and an electro-dynamic field [82, 83].

4.4.6. Performance

In this section, the performance of the integrated approach *APESmate* is investigated. These results are published in [66] and presented here for completeness as well as comparison to *preCICE*. Hence, the same scaling testcase as well as the same Supercomputer as for the performance measure-

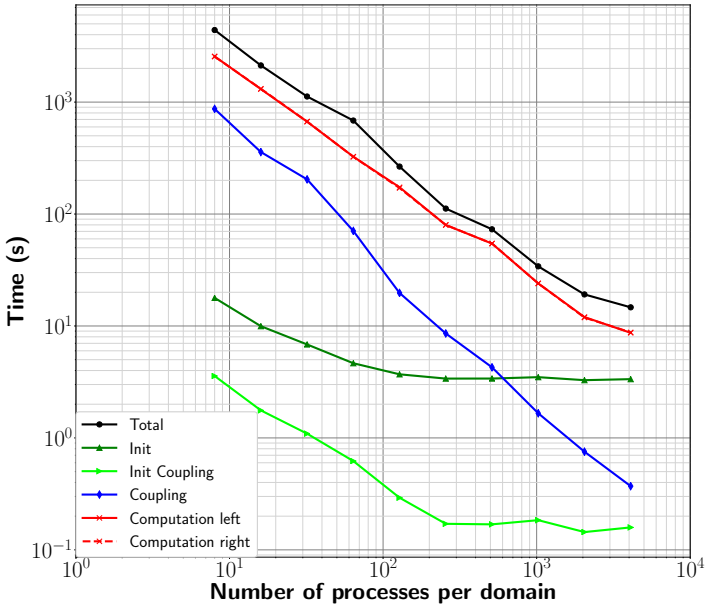


Figure 4.10.: Strong scaling using *APESmate* up to one island (= 8192 MPI-processes) on the SuperMuc Phase 1 IBM system. The total runtime is split into individual subroutines.

ments of the coupling tool *preCICE* in Section 4.3.6 are utilized. Strong scaling up to a single island is presented in Figure 4.10 where the number of processes per subdomain is on the x-axis and the total run time in seconds is on the y-axis. The total runtime is split into the individual routines: *Initialization*, *Coupling* and *Computation*. To investigate the influence of coupling on the *Initialization*, the initialization of the coupling is measured individually as *Init Coupling*. *Coupling* includes the synchronization of subdomains which comprises solver specific data mapping, i.e. evaluation of the polynomial in *Ateles*, and the point-to-point communication of these values between the corresponding partitions. For *Computation* the measurements of the left and the right domain are plotted individually, which show the same behavior. The major phases during simulation, *Coupling* and *Computation*, are both scaling well. *Coupling* shows linear speedup and scales better than *Computation*. Moreover, the execution time of *Coupling* is about one order of magnitude lower than *Computation*. This is important because the coupling approach should not slow down the overall computation. This testcase is a nice showcase for performance, since both subdomains have the same number of elements and degrees of freedom to solve. Furthermore, the number of elements and the polynomial order are equal. Due to the same setup, there are no imbalances between the subdomains and the point-to-point communication in *APESmate* is optimized. There are still work imbalances in each subdomain due the additional work of the elements involved in coupling but imbalances of the whole domain are minimized. *Initialization* is scaling roughly up to 256 processes per domain, after which it flattens out. The same holds true for the initialization phase of the coupling approach *Init Coupling*. Additionally, *Init Coupling* is more than one magnitude of order smaller than *Initialization*. Hence, *Init Coupling* does not restrict the *Initialization* as it is done with the coupling library *preCICE* (Figure 4.8 in Section 4.3.6).

The presented results are obtained using sparse all-to-all communication. Using MPI _ALLTOALL communication, the initialization phase yields poor scalability. This is described in more detail in Appendix B. In the following, the initialization phase is more investigated.

4.4.6.1. Initialization of coupling

Figure 4.11 depicts the *Init Coupling* routine in the initialization phase of *APESmate* which is split further into two major routines: *Fill SpaceTime Function* and *Init Cpl Comm*. *Fill SpaceTime Function* gathers all main information, e.g. which subdomain requests which coupling values from

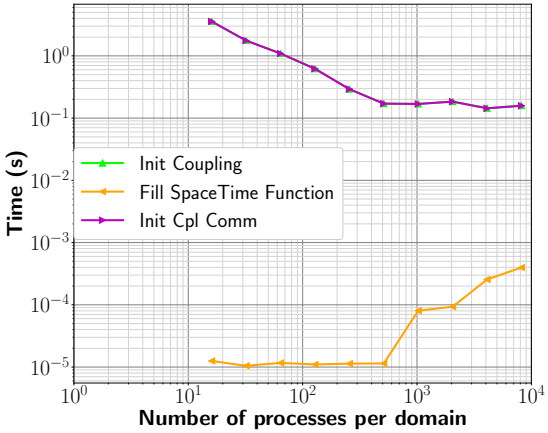


Figure 4.11.: Strong scaling using *APESmate* of the initialization of *APESmate*. The initialization has two major routine: *Fill SpaceTime Function*, which gathers all information about coupling variables from each subdomain, and *Init Cpl Comm*, which establishes the point-to-point communication between the subdomains.

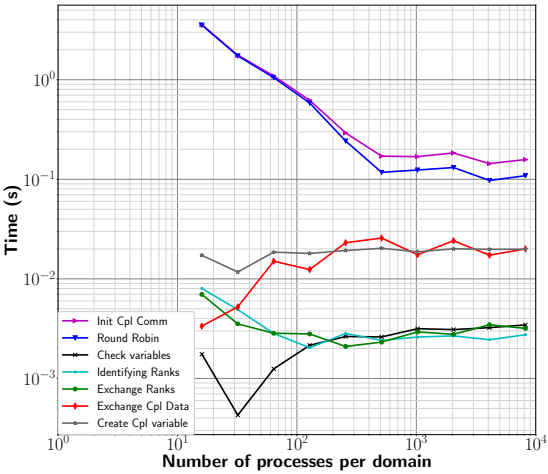


Figure 4.12.: Strong scaling of the initialization phase of the coupling communication *Init Cpl Comm* of *APESmate* split into individual substeps.

which other subdomain. *Init Cpl Comm* establishes the point-to-point communication between the MPI-processes of subdomains. As you can see, *Fill SpaceTime Function* does not scale at all but only requires a small amount of time for less than 1000 processes. The initialization phase is mainly dominated by the initialization of the coupling communication, *Init Cpl Comm*. Therefore this routine is further investigated in Figure 4.12. The major routine *Init Cpl Comm* is split into the individual subroutines:

Round Robin Communicate general information (coupling points and variable names) to the remote domain in round robin

Identify Ranks Identify source ranks ("ranks sending data") and target ranks ("ranks receiving data") from the round robin information to establish direct communication between source and target ranks

Check variables Check if the requested variable names are actually available on the remote domain

Exchange Ranks Send the information of the corresponding rank back to the requesting partition in round robin fashion

Exchange Cpl Data Send the requested coupling information to the correct targets which host the requested partition

Create Cpl variable Combine requested information to avoid multiple communication during a synchronization step of *APESmate*

As it can been seen in Figure 4.12, the initialization *Init Cpl Comm* is mainly dominated by the substep *Round Robin*. Up to 512 processes, *Round Robin* scales well, and then it flattens out. All other routines don't scale well, i.e. the time spent in each routine is almost constant as number of processes increases. However, the time spent in this subroutine is reasonably small and thus the initialization phase will be negligible for long simulation runs.

5. Coupling results

This chapter collect the results of a selection of coupled simulations. In the first part we use an **academic testcase** to investigate two coupling strategies: The integrated approach *APESmate* and the multi-solver approach utilizing *preCICE*. Partitioned coupling provides the flexibility to couple different physics by using different equations, different numerical resolutions, i.e. different grid resolutions and spatial order of the scheme, and different solvers. Since different resolutions lead to non-matching coupling interfaces, we investigate the influence of data mapping in space at coupling interfaces. First results of this, including a 2D scenario, have been published in [66]. In this chapter, we investigate a 3-field coupling where three physical regimes are considered in 3D: A viscous flow (Navier-Stokes equations), an inviscid flow (Euler equations), and an acoustic far field (Linearized Euler equations).

Once the aforementioned coupling strategies have been evaluated, a **realistic testcase** is investigated: A 3D subsonic free-stream jet. Here, we first analyze the numerical results of a 3-field coupled simulation with correctly chosen coupling interfaces. After the numerical results have been presented and the testcase setup has been described, we examine the load balancing in detail and *APESmate* and *preCICE* are compared in terms of performance. We complete the chapter with a section on how the choice of imperfect coupling interfaces influences acoustic wave propagation into a far field. The chapter is completed by the numerical results of this realistic testcase.

5.1. Gaussian distribution in pressure

In order to analyze the integrated and the multi-solver approach, we start by coupling the same physics and the same resolution in both domains. This means that we use the same equations and a matching coupling interface by using the same grid sizes as well as spatial orders in both subdomains. Once the implementation has been checked for all combinations and the coupling and its data exchange itself does not influence the solution, we change the resolution in the domains but keep the equations

the same. Hereby, we investigate the interpolation errors of *APESmate* and *preCICE*. Due to the multi-solver nature of *preCICE*, it has to use an external interpolation method and we chose the nearest projection (NP) mapping, according to Section 4.3.5.3, for non-matching interfaces. The integrated approach *APESmate* uses direct evaluation of polynomial representations in the DG scheme as presented in Section 4.4.5. We compare the solutions by plotting evolution over time and inspecting deviations in these figures and measure the relative error at positions close to the coupling interface. In order to provide a valid comparison of the relative error we compute the monolithic simulation for all governing equations. The monolithic simulations are performed with the resolution of the matching scenario, since a monolithic setup is limited to the same order of the numerical scheme. Subsequently, we investigate the coupling of different physics. Thus, we analyze all possible combinations of different equations which are meaningful, i.e. Navier-Stokes equations with Euler equations and Euler equations with Linearized Euler equations. Here, we start with matching interfaces and continue with non-matching interfaces. For non-matching interfaces, there will be again a difference in both approaches due to different interpolation techniques. The last step is the investigation of a 3-field coupling where three physical regimes are coupled: A viscous flow (Navier-Stokes equations), an inviscid flow (Euler equations), and an acoustic far field (Linearized Euler equations).

For brevity, we will use the abbreviations NSE for Navier-Stokes equations, EE for Euler equations and LEE for the Linearized Euler equation. Furthermore, we will refer to a domain according to the equations used in that domain, i.e. a domain employing the Euler equations as *Euler domain*.

We have chosen an academic testcase which is suitable for this investigation: Valid for all equations, physics influences all variables of the state vector, and an analytical solution is available. Hence, the academic testcase for the analysis is a 3D Gaussian pulse in pressure as illustrated in Figure 5.1. The different subdomains are referred to as *inner* and *outer*, respectively. The 3D acoustic pulse is initialized at time $t = 0$ with a Gaussian pressure distribution which is spreading spherically symmetric with respect to the origin of the pulse as described in [84] and sketched in Figure 5.1. All lengths are defined in length units *lu*. The 3D flow domain in which the pulse is located is a box of $20 \times 20 \times 20$ *lu* with a surrounding outer domain of size $60 \times 60 \times 60$ *lu*. The parameters for the different flow regimes are defined in Table 5.1.

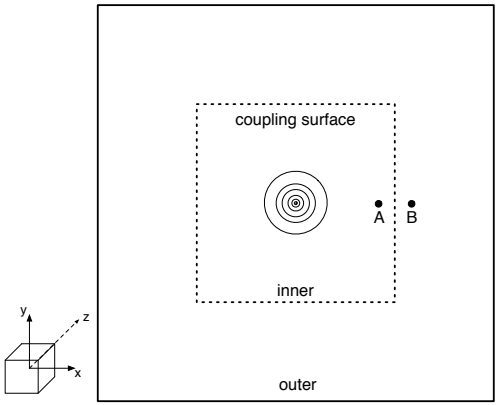


Figure 5.1.: 2D sketch of the 3D Gaussian pulse in pressure [origin at (0,0,0)], where a inner box is surrounded by an outer box. The • mark the measurement positions $A = (10 - 0.01, 0, 0)$ and $B = (10 + 0.01, 0, 0)$, close to the coupling interface at $x = 10$ (dashed line).

viscous flow (NSE)	dynamic viscosity	μ	$1.0\text{e}^{-6}\,\text{m}^2\,\text{s}^{-1}$
	thermal conductivity	λ	$1.5625\text{e}^{-2}\,\text{W}\,\text{m}^{-1}\,\text{K}^{-1}$
viscous/ inviscid flow (EE)	isentropic coefficient	γ	1.4
	specific gas constant	R	280.0
acoustics (LEE)	background density	ρ_0	1.0
	background velocity	$\mathbf{v}_0$	$\left[0.0, 0.0, 0.0\right]^T$
	background pressure	p_0	$1/\gamma$
	speed of sound	c	1.0

Table 5.1.: Parameters for the Gaussian pulse in pressure for all different equations: viscous flow (NSE), inviscid flow (EE) and acoustics (LEE).

For the inner domain, the initial condition is a Gaussian pressure distribution:

$$p = p_0 + p_{pulse} \cdot \left(-[(x+x_0)^2 + (y+y_0)^2 + (z+z_0)^2]/b \cdot \log(2)\right) \quad (5.1)$$

with amplitude of the pulse $p_{pulse} = 0.001$ and half width set to $b = 3$. The origin is $(x_0, y_0, z_0) = (0.0, 0.0, 0.0)$. The background for the flow is set to the background of the LEE domain defined in Table 5.1. When defining the inner domain to be an LEE domain, p_0 in Equation (5.1) is set to 0 instead of $1/\gamma$, since the Linearized Euler equations only solve for perturbations and the background state $p_0 = 1/\gamma$ is already defined with the Linearized Euler equations (2.15). In case the outer is an LEE domain, the initial condition $\left[\rho^a, \mathbf{v}^a, p^a\right]^T$ is specified to be $\mathbf{0}$, since at the start of the simulation, no acoustic perturbation should occur. The inner boundaries are always coupling interfaces where, as described in Section 3.4, the variables according to the equations are exchanged. When coupled to an NSE domain, also gradients in normal direction are exchanged. The outer boundaries are set to Dirichlet boundary conditions for all state variables where in viscous and inviscid flow the state is set to the background state and for acoustics the perturbations are set to zero. This can avoid stability issues which might occur when extrapolating these boundary conditions. For linear acoustic transportation, in particular, we did not face issues with reflections due to the type of boundary condition. The analytical solution for a Gaussian pressure distribution spreading spherically symmetric with respect to the origin (0.0, 0.0, 0.0) and the radial distance $r = \sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2}$ is :

$$p = p_0 + p_{pulse} \cdot \left[\frac{r - c\cdot t}{2\cdot r} \cdot \exp\left(-\log(2) \cdot \left(\frac{r-c\cdot t}{b}\right)^2\right) + \frac{r + c\cdot t}{2\cdot r} \cdot \exp\left(-\log(2) \cdot \left(\frac{r+c\cdot t}{b}\right)^2\right)\right] \quad (5.2)$$

with speed of sound defined by the material $c = \sqrt{\frac{\gamma \cdot p_0}{\rho_0}}$, or in acoustic $c = 1.0$. For this testcase, we assume that the perturbations are sufficiently small for the linearization to be applicable. Therefore, the analytical solution is valid for NSE and EE and deviations due to non-linearity and viscous effects are negligible.

As a first step, we check the implementation by keeping the physics as well as the resolution constant. To that end, we require a matching coupling interface. Subsequently, to investigate the data mapping in space

and, thus, the interpolation error of integrated and multi-solver approach, we require a non-matching coupling interface. Table 5.2 presents the used discretizations to obtain matching and non-matching coupling interfaces. For matching interface, we chose the grid size $h = 1$ and a numerical

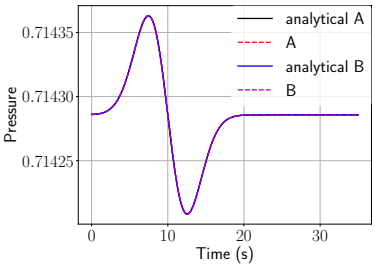
interface	domain	total number of elements	grid size h	numerical order p
matching	inner	8000	1	$\mathcal{O}(6)$
	outer	208 000	1	$\mathcal{O}(6)$
non-matching	inner	8000	1	$\mathcal{O}(6)$
	outer	1664	5	$\mathcal{O}(12)$

Table 5.2.: Discretizations for matching and non-matching coupling interfaces.

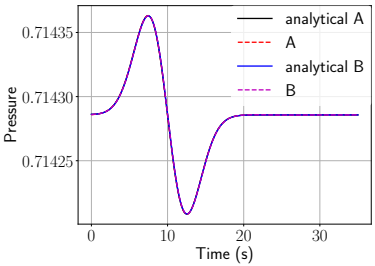
order of $\mathcal{O}(6)$. For non-matching interfaces, the effort in the outer domain is decreased by using a coarser mesh ($h = 5$ instead of $h = 1$), but a higher order in the DG scheme compared to the inner domain ($\mathcal{O}(12)$ instead of $\mathcal{O}(6)$). This leads to $DoF = 8\,640\,000$ in the inner domain and $DoF = 224\,640\,000$ (matching) and $DoF = 14\,376\,960$ (non-matching) in the outer domain, respectively. We decrease the number of DoF when using a higher order, since the DG scheme shows a similar convergence rate and, hence, fewer DoF are required for a high accuracy. For the coupling interface in the non-matching case we decrease the total number of coupling points from 14 400 to 2304. For each simulation, using *APESmate* as well as *preCICE*, the timestep is fixed and depends on the CFL condition of the explicit RK scheme together with the respective equation. For *preCICE*, nearest neighbor (NN) mapping is utilized for matching interfaces and NP mapping for non-matching interfaces as presented in Section 4.3.5.3.

5.1.1. 2-field coupling of same equations

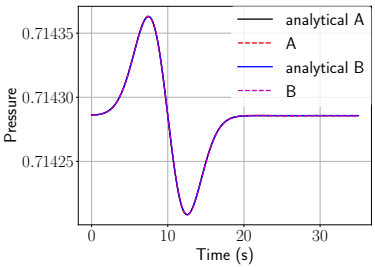
For the investigation on coupling of the same equations, we look at the temporal evolution of values at the specific points A and B close to the coupling interface (± 0.01) as sketched in Figure 5.1. We compare the analytical solution against the solution of the simulation and plot temporal evolution as well as look at the relative error. Due to the large number of combinations, we only show significant results in this chapter. All plots are presented in Appendix C. Figure 5.2 shows the result for coupling the



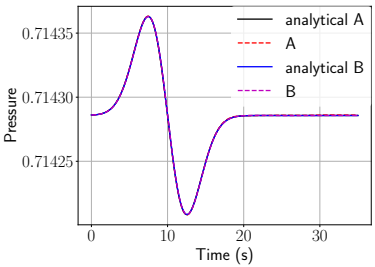
(a) NSE - NSE with *APESmate*, non-matching interface.



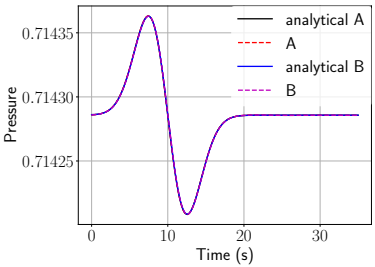
(b) NSE - NSE with *preCICE*, non-matching interface.



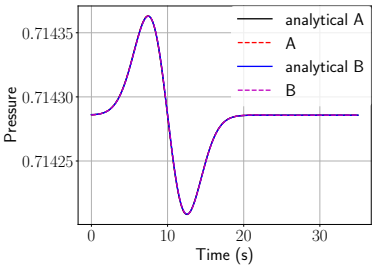
(c) EE - EE with *APESmate*, non-matching interface.



(d) EE - EE with *preCICE*, non-matching interface.



(e) LEE - LEE with *APESmate*, non-matching interface.



(f) LEE - LEE with *preCICE*, non-matching interface.

Figure 5.2.: Comparison of numerical and analytical result in both subdomains for coupling of **same** equations with non-matching interfaces for *APESmate* (left column) and *preCICE* (right column).

same equations with non-matching interfaces. Since matching interfaces are not as challenging as non-matching interfaces, we explicitly present the combination with non-matching interfaces. It depicts the analytical solution according to Equation (5.2) at positions A and B and the solution of the simulation at these positions. We can see that the solutions of the simulations match the analytical solution well and *APESmate* as well as *preCICE* are comparable. Please note, that the solution at position A and B coincides since the pulse is very broad and does not vary within 0.02 *lu*.

equations		A (Inner)		B (Outer)	
NSE		$3.948e^{-8}$		$3.972e^{-8}$	
EE		$4.070e^{-8}$		$4.092e^{-8}$	
LEE		$2.629e^{-9}$		$2.617e^{-9}$	
	interface	*APES-mate*	*pre-CICE*	*APES-mate*	*pre-CICE*
NSE - NSE	matching	$1.899e^{-7}$	$1.586e^{-7}$	$4.051e^{-9}$	$1.730e^{-8}$
NSE - NSE	non-matching	$1.537e^{-7}$	$1.600e^{-7}$	$2.605e^{-8}$	$5.138e^{-8}$
EE - EE	matching	$2.175e^{-7}$	$2.175e^{-7}$	$1.361e^{-8}$	$1.361e^{-8}$
EE - EE	non-matching	$2.221e^{-7}$	$6.666e^{-8}$	$3.795e^{-9}$	$3.422e^{-8}$
LEE - LEE	matching	$1.528e^{-7}$	$1.741e^{-7}$	$3.313e^{-8}$	$3.197e^{-8}$
LEE - LEE	non-matching	$1.897e^{-7}$	$1.060e^{-7}$	$3.625e^{-8}$	$1.087e^{-7}$

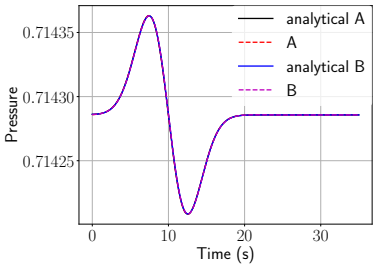
Table 5.3.: Relative error at simulation time $t = 7$ in pressure for positions A and B for coupling the **same** equations, first three rows are the monolithic simulations.

To investigate these results further, we look at the relative error in pressure measured at positions A and B at the simulation time of 7 *tu*, when the maximum pressure over simulation is reached. Table 5.3 shows this relative error for matching and non-matching interfaces as well as both coupling strategies. Furthermore, the relative error at the simulation time t=7 *tu* for a monolithic simulation is provided in the first three rows. The relative error is normalized to the analytical solution of Equation (5.2).

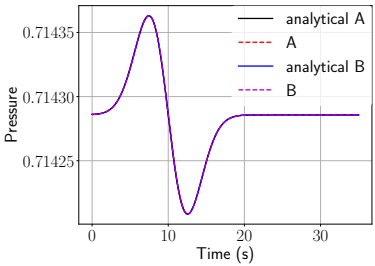
We can see that the relative error that occurs in all simulations is in the order of e^{-7} to e^{-8}, and one order of magnitude lower than for the monolithic approach. Both approaches, *APESmate* and *preCICE* with NN/NP mapping, yield similar results. The relative errors for monolithic with NSE and EE is one order of magnitude lower compared to the LEE. This can be explained with the fact that the presented analytical solution (Equation (5.2)) is the solution of the linear equations under the assumption that they are a good approximation for NSE and EE. This assumption only holds when the viscous and non-linear terms, including rounding errors, become zero. In Section 6, we will have a closer look at the time-consistent treatment of the coupling interface when using a second-order RK scheme, as it is done for this simulations.

5.1.2. 2-field coupling of different equations

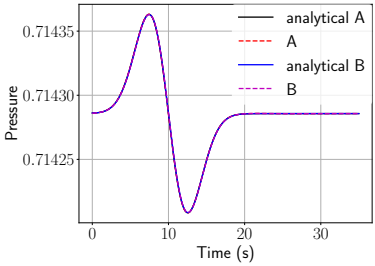
The next step is analyzing the coupling of different equations. Here, in a first step, only the equations are switched while grid size and order of the scheme are kept the same in both parts of the domain (= matching resolution). Hereby, the influence of different equations is investigated. Once this is done, we test the coupling of different equations having different resolutions in both subdomains yielding a non-matching coupling interface. In this section we only print selected figures; figures for all combinations of coupling can be found in Appendix C. When exploiting non-matching interfaces, we utilize the NP mapping for the multi-solver approach with *preCICE*. Figure 5.3 shows the results for coupling different equations. The first two figures present NSE–EE coupling with matching interface for *APESmate* (a) and *preCICE* (b). The figures (c) - (f) demonstrate the coupling with non-matching interface. These figures do not indicate a discrepancy between simulation results and analytic results. Table 5.4 shows the comparison of the results for matching and non-matching interfaces in terms of the relative error in pressure at simulation time 7 *tu* (maximum pressure at positions A and B over simulation time). Similar to the investigation of coupling same equations (Table 5.3), both approaches show the same behavior. One striking result is the simulation of EE - LEE with non-matching coupling interface using *APESmate*, where the relative error is two orders of magnitude lower.



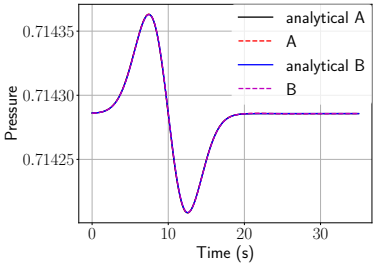
(a) NSE - EE with *APESmate*, matching interface.



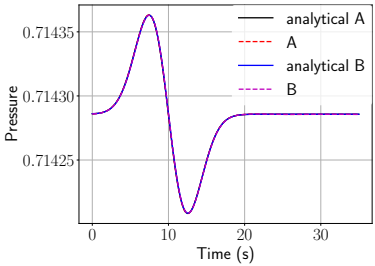
(b) NSE - EE with *preCICE*, matching interface.



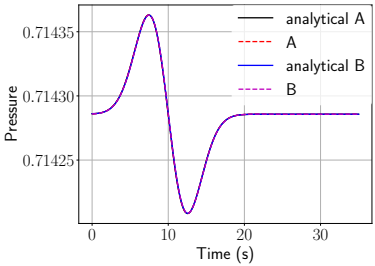
(c) NSE - EE with *APESmate*, non-matching interface.



(d) NSE - EE with *preCICE*, non-matching interface.



(e) EE- LEE with *APESmate*, non-matching interface.



(f) EE - LEE with *preCICE*, non-matching interface.

Figure 5.3.: Comparison of numerical and analytical result in both subdomains for coupling of **different** equations with non-matching and matching interfaces for *APESmate* (left column) and *preCICE* (right column).

equations		A (Inner)		B (Outer)	
NSE		$3.948e^{-8}$		$3.972e^{-8}$	
EE		$4.070e^{-8}$		$4.092e^{-8}$	
LEE		$2.629e^{-9}$		$2.617e^{-9}$	
	interface	*APES-mate*	*pre-CICE*	*APES-mate*	*pre-CICE*
NSE - EE	matching	$1.900e^{-7}$	$1.586e^{-7}$	$4.226e^{-9}$	$1.742e^{-8}$
NSE - EE	non-matching	$1.538e^{-7}$	$1.611e^{-7}$	$2.615e^{-8}$	$4.565e^{-8}$
EE - LEE	matching	$2.210e^{-7}$	$9.305e^{-8}$	$1.010e^{-8}$	$4.830e^{-8}$
EE - LEE	non-matching	$2.256e^{-7}$	$7.718e^{-8}$	$1.204e^{-10}$	$3.585e^{-8}$

Table 5.4.: Relative error at simulation time $t = 7$ in pressure for positions A and B when coupling **different** equations, first three rows are the monolithic simulations.

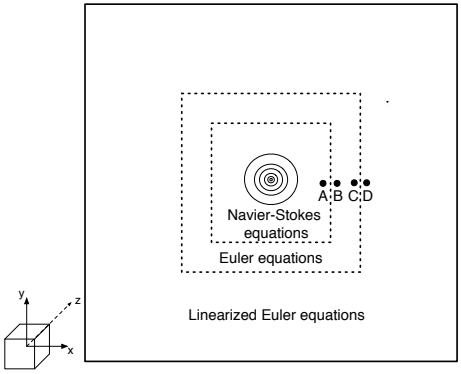


Figure 5.4.: 2D sketch of 3-field coupling for the 3D Gaussian pulse [origin at (0,0,0)]. The dotted lines illustrate coupling interfaces. The • mark the measurement positions A, B, C, and D, close to the coupling interfaces at $x = 10$ and $x = 15$ (dashed line).

5.1.3. 3-field coupling of different equations

In this last section, we investigate a 3-field coupling of the 3D Gaussian pulse with matching as well as non-matching interfaces. The goal is to couple three different equations: NSE, EE, and LEE. Once a 3-field cou-

pling is validated, it enables an efficient fluid-acoustic simulation.

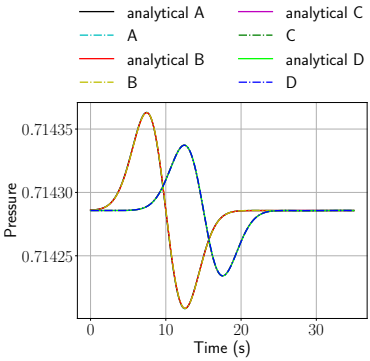
Figure 5.4 illustrates the sketch of this 3-field coupling. The complete domain ($60 \times 60 \times 60$ *lu*) is split into an inner ($20 \times 20 \times 20$ *lu*) subdomain, surrounded by the middle ($30 \times 30 \times 30$ *lu*) subdomain, which is surrounded by the outer ($60 \times 60 \times 60$ *lu*) subdomain.The inner domain solves the Navier-Stokes equations, then the Euler equations are computed, surrounded by the acoustic far field approximated by the Linearized Euler equations. The same parameters for the different flow regimes as in the previous investigations are used (Table 5.1). For this setup we have four measuring positions: $A = (10 - 0.01, 0, 0)$, $B = (10 + 0.01, 0, 0)$, $C = (15 - 0.01, 0, 0)$, and $D = (15 + 0.01, 0, 0)$, where A is located in the Navier-Stokes domain, B and C in the Euler domain and D lies in the Linearized Euler domain. Table 5.5 defines the discretizations for matching

interface	domain	total number of elements	grid size h	numerical order p
matching	inner	8000	1	$\mathcal{O}(6)$
	middle	19 000	1	$\mathcal{O}(6)$
	outer	189 000	1	$\mathcal{O}(6)$
non-matching	inner	8000	1	$\mathcal{O}(6)$
	middle	1216	2.5	$\mathcal{O}(10)$
	outer	1512	5	$\mathcal{O}(12)$

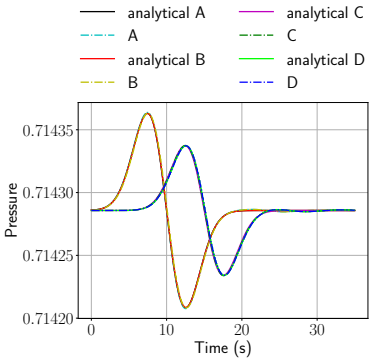
Table 5.5.: Discretizations for matching and non-matching coupling interfaces for 3-field testcase.

and non-matching coupling interfaces for the 3-field testcase NSE–EE–LEE. For the non-matching setup, the inner domain as well as the outer domain are similar to the previous investigations. For the middle domain we chose a grid to be between $h = 1$ and $h = 5$ and a medium order of $\mathcal{O}(10)$. This decreases the number of coupling points at the coupling interface by a factor of 3 between each coupling interface.

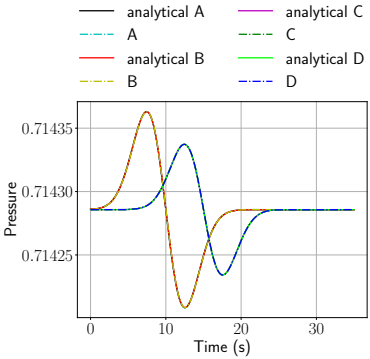
Figure 5.5 presents the simulation results for matching and non-matching interfaces for both coupling approaches. The amplitude of the pressure pulse decreases over distance. In contrast to the previous section, we have a distance of 5 *lu* between the interfaces. Hence the solutions at A/B



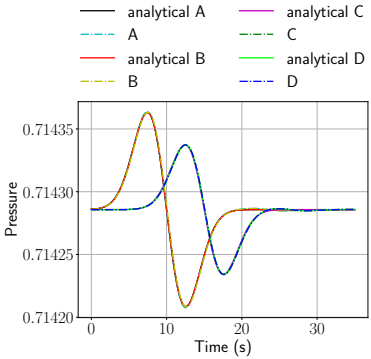
(a) NSE - EE - LEE with *APESmate*, matching interface.



(b) NSE - EE - LEE with *preCICE*, matching interface.



(c) NSE - EE - LEE with *APESmate*, non-matching interface.



(d) NSE - EE - LEE with *preCICE*, non-matching interface.

Figure 5.5.: Comparison of numerical and analytical results for the coupling of three different equations with matching interfaces and non-matching interfaces at measuring points A, B, C, and D.

and C/D do not coincide and the time evolution show two pulses with a decreased amplitude. The solutions at all measured points show good agreement with the analytical solution. Table 5.6 presents the relative error at simulation time of 7 *tu* for position A/B and 13 *tu* for position C/D for both coupling approaches and both coupling interfaces. At the simulation time $t = 13$, the maximum of the pressure has reached the measuring positions C/D. The first three rows are the corresponding results for the monolithic simulations. All relative errors of the monolithic simulations are in the range of e^{-6} to e^{-9}. Compared to the presented investigations with two coupling domains (Table 5.3 and Table 5.4), the relative error is one order of magnitude higher. Although we have two coupling interface now, the error at the boundary in the outermost subdomain is not significantly lower. The error of the monolithic simulations at the measuring points also varies with two orders of magnitude.

			A (Inner)	B (Middle)	C (Middle)	D (Outer)
NSE			$3.948e^{-8}$	$3.972e^{-8}$	$9.892e^{-9}$	$8.582e^{-9}$
EE			$4.070e^{-8}$	$4.092e^{-8}$	$7.861e^{-9}$	$6.547e^{-9}$
LEE			$2.629e^{-9}$	$2.617e^{-9}$	$3.945e^{-10}$	$4.754e^{-10}$
NSE - EE - LEE	interface	coupling approach				
	matching	*APESmate*	$1.899e^{-7}$	$4.207e^{-9}$	$1.187e^{-7}$	$1.793e^{-7}$
		preCICE	$9.339e^{-7}$	$3.421e^{-7}$	$1.007e^{-6}$	$1.322e^{-6}$
	non-matching	*APESmate*	$1.604e^{-7}$	$1.305e^{-8}$	$9.131e^{-8}$	$1.356e^{-7}$
		preCICE	$1.028e^{-6}$	$2.034e^{-7}$	$8.186e^{-7}$	$8.584e^{-7}$

Table 5.6.: Relative error at simulation time $t = 7$ for position A/B and $t = 13$ for position C/D in pressure when coupling **three different** equations, first three rows are the monolithic simulations.

5.2. 3D subsonic free-stream jet

The simulation of a 3D subsonic free-stream jet is enabled by employing a 3-field coupling. The free-stream is expanded by a nozzle, where close to it viscous effects influence the flow leading to vortices. Farther away from the nozzle, where viscous terms can be neglected, inviscid Euler equations can be used to describe the physics. In the outermost region, where only acoustic waves are transported, the Linearized Euler equations are employed. For ease of reading, domains will be named after their respective equations in the following: *NSE subdomain* for Navier-Stokes equations, *EE subdomain* for Euler equations, and *LEE subdomain* for linearized Euler equations. First, we will look at numerical results of the free-stream jet and discuss the numerical testcase including the coupling setup. In a monolithic simulation with a state-of-the-art solver, the grid size can be varied, while a change in the order of the numerical scheme within a domain is often not possible. Thus, in order to compare the performance of the coupling approaches to a monolithic simulation, we setup the coupled scenario to use the same numerical order in each domain. With this monolithic-like setup **A**, we investigate static load balancing strategies on the massively parallel system SuperMuc Phase 1 for *APESmate* and *preCICE*. For the integrated approach *APESmate* we will take a closer look at the measured weights. With the appropriate load balancing strategy for both approaches, the performance is compared. Once, an appropriate load balancing strategy is found, we will adapt the the setup by tailoring the numerical discretization to the physical regimes to leverage all benefits of a partitioned coupling approach. According to the physics, we chose a low order with a very fine grid for the flow domain and a high order with a coarse mesh for the acoustic far field. This tailored setup for the numerical resolution will be called setup **B**. We will look at the performance of this tailored setup **B** with *APESmate* and compare it to a monolithic simulation as well as to the monolithic-like setup **A**. Since another important aspect of performance is the scalability of an approach, we will also present a strong scaling of the balanced *APESmate* approach for both setups. Subsequently, we will investigate the impact of choosing imperfect coupling interfaces on the acoustic wave propagation with the 3D subsonic free-stream jet.

5.2.1. Investigation of numerical setup

A 3D subsonic free-stream jet is an example for a scenario where acoustics are generated and acoustic waves are radiated. “Subsonic” indicates that the Mach number of the flow (Ma) is less than 1, that is, flow velocity

is smaller than the speed of sound everywhere. Typically, such jets are expanded from a nozzle: The nozzle flow itself is not discretized but modeled with the initial conditions as described in [85]. The noise generation of free stream jets has been studied [85–88].

The presented 3-field coupling is applied to the testcase of a subsonic 3D jet. The jet is separated into an NSE subdomain, close to the nozzle inlet where turbulent flow occurs. This is surrounded by the EE subdomain, and, where non-linear effects can be neglected, we introduce a surrounding LEE subdomain.

First, we have a general look at how such a free-stream develops within the NSE subdomain. Here, we also show parts of the EE subdomain, where white lines in the figures denote coupling interfaces between subdomains. Figure 5.6 illustrates the instantaneous density field at a simulation time

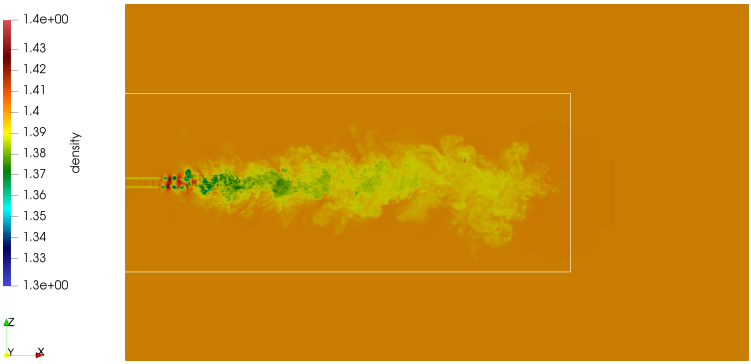
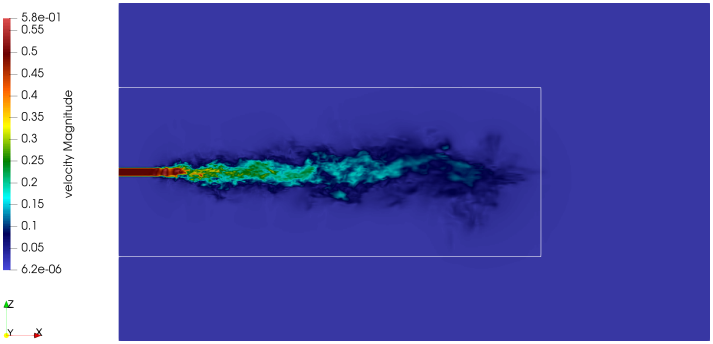


Figure 5.6.: Instantaneous density field of a 3D subsonic free-stream jet at $t = 120$.

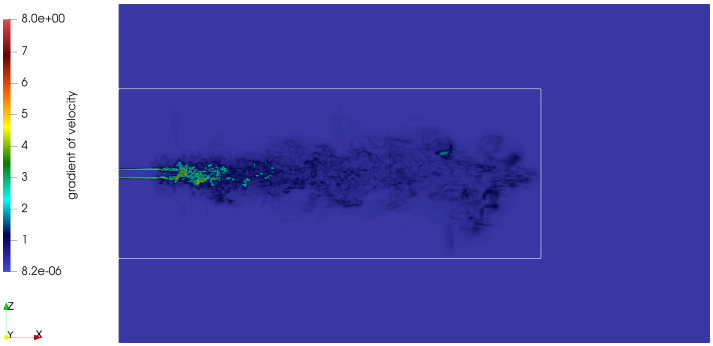
of 120 *tu*. *tu* defines time unit and will be used in the following as unit for the simulation time. On the left boundary, a nozzle inlet is modeled and the free-stream can be observed flowing in x-direction. Along the flow direction, a stable stream and its shear layer are visible for a limited extent. Outside of the stream we observe a fluid at rest. At the shear layer there is a high velocity inside the stream and low velocity outside.

Figure 5.7 illustrates the properties of the free-stream when considering velocity at the same point in time as the density field. Figure 5.7a presents the instantaneous velocity field where the shear layer is clearly visible. Almost immediately, however, the stream breaks up into turbulent flow. This depends on the viscosity of the fluid and the velocity of the flow. In the turbulent region vortices are generated which transmit sound waves. These vortices travel downstream, decay, and interact with each other. Such interactions create acoustic waves, which are radiated to the surrounding of the free-stream jet. Figure 5.7b illustrates the gradient of the velocity. The only term in the Navier-Stokes equations (2.11) that includes the stress tensor (2.6) with viscosity is the multiplication with the gradient of velocity. Hence, we can interpret the gradient of velocity as a measure for the influence of viscosity on the flow. We can see that viscosity is a major contributor to the collapse of the jet and the emergence of a turbulent region. Downstream, where the stream widens slightly, the gradient of the velocity is close to a value of 1-1.5. In the surrounding of the free-stream, where the fluid is at rest ($v = 0$), the viscosity has no impact. Hence, the gradient of velocity can indicate where we have to compute Navier-Stokes equations in order to capture the physical phenomena correctly. Vorticity is another variable related to velocity, and is presented in Figure 5.7c. The vorticity shows very similar features to the gradient of the velocity (Figure 5.7b). Hence, we can say that in regions where vorticity is not zero, viscosity and non-linearity effects play a role and computing the NSE or EE is required.

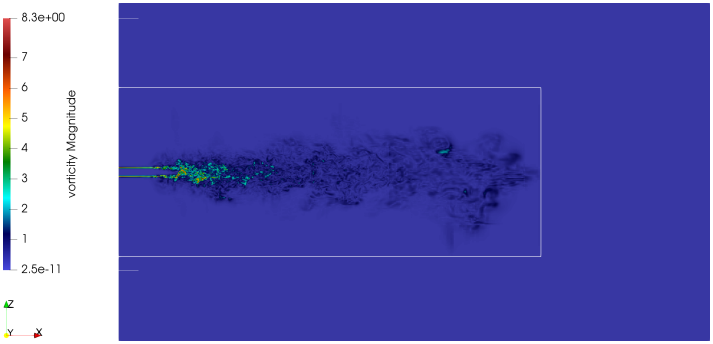
The physical phenomenon that we are ultimately interested in is the acoustic generation and wave propagation. Figure 5.8 presents the pressure field of the subsonic free-stream scaled to the flow as well as to the acoustics. Considering the flow scales in Figure 5.8a, the pressure field is completely located in the NSE domain, while in the acoustic scales the coupling interface is reached: In the acoustic scales in Figure 5.8b we can see acoustic waves radiating from the free-stream jet. At the inlet, the radius of the acoustic wave front is visible, and downstream we can see that waves with larger radii occur. This figure demonstrates that a large surrounding domain is required to capture acoustic waves entirely. Since we are interested in the acoustic far field, we will generously extend the computational domain in order to propagate the acoustic waves. To visualize them, Figure 5.9 presents a Schlieren visualization of the NSE and EE subdomain for $t = 120\,tu$. Compared to the figures before, we show an even larger excerpt of the EE subdomain to visualize the wave propagation. Schlieren visualizations are common practice to visualize the



(a) Velocity

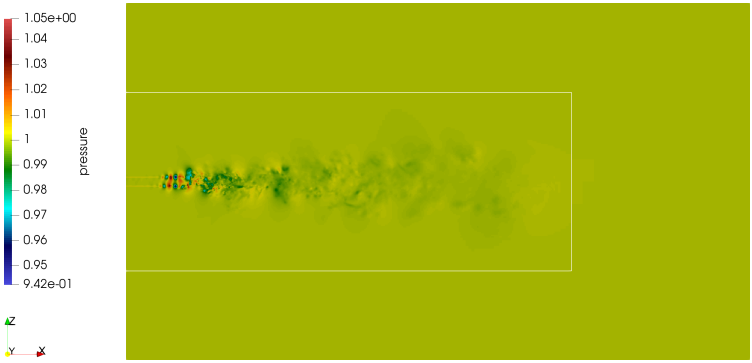


(b) Gradient of velocity

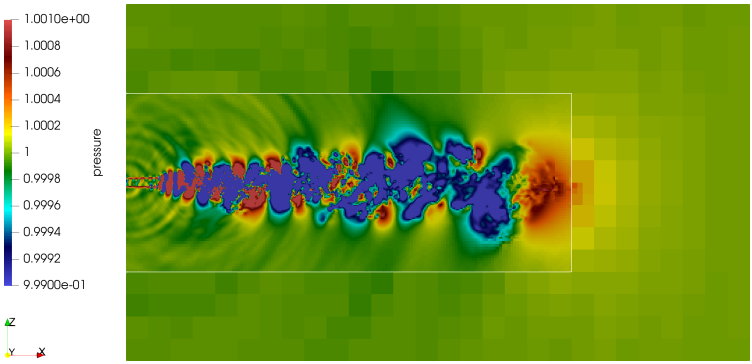


(c) Vorticity

Figure 5.7.: Velocity, gradient of velocity, and vorticity of the 3D subsonic free-stream jet at t=120.



(a) Scaled to flow



(b) Scaled to acoustics

Figure 5.8.: Pressure field of the 3D subsonic free-stream jet at $t = 120$.

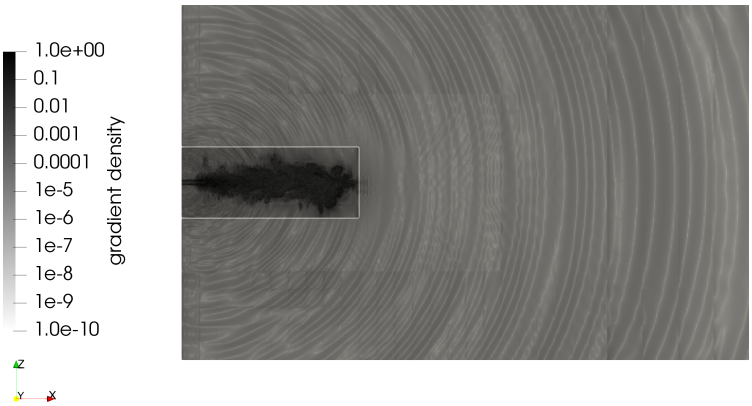


Figure 5.9.: Schlieren visualization (gradient of density) of the 3D subsonic free-stream jet at $t = 120$ surrounded by the Euler subdomain (extended clip). White lines indicate coupling interfaces.

flow of fluids and their acoustic waves. It shows the flow density gradient and, thus, how density varies. We use a gray color-scale to be as close as possible to common Schlieren flow photographies that are based on shadow patterns. The scale is logarithmic to cover the full range ($1.0e^{-10}$ to 1.0) of values.

Acoustic far field While we have looked at the acoustic generation in the turbulent region before, we now inspect the wave propagation into the acoustic far field. Figure 5.10 presents the Schlieren visualization at simulation time of 120 tu for almost the entire computational domain. For presentation's sake, the outermost subdomain is cropped so that only the inner region is visible. At this simulation time, acoustic waves have traveled through the EE subdomain and have already reached passed through half of the acoustic subdomain.

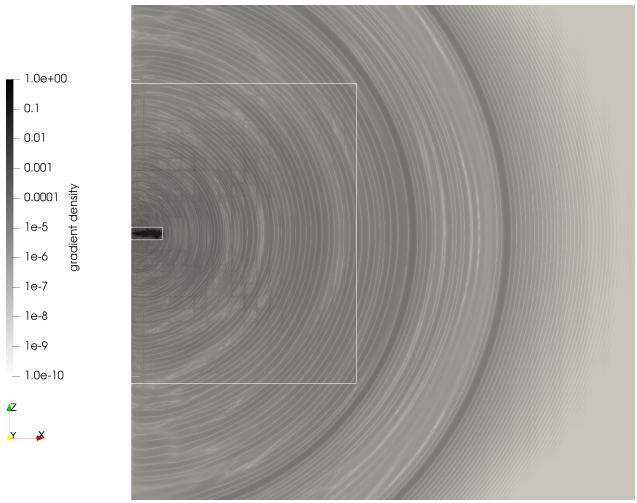
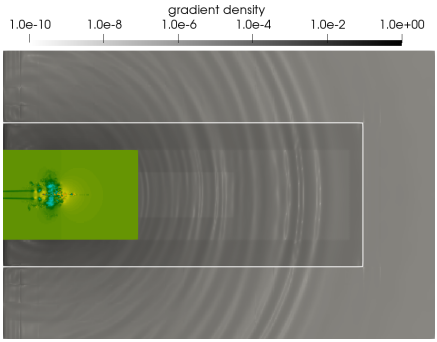
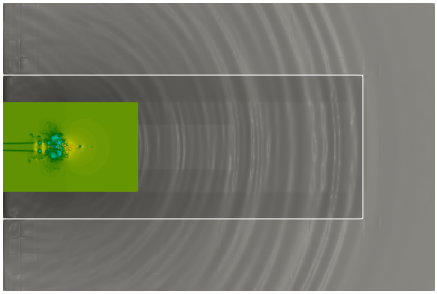


Figure 5.10.: Schlieren visualization of NSE, EE, and parts of LEE at $t = 120$.

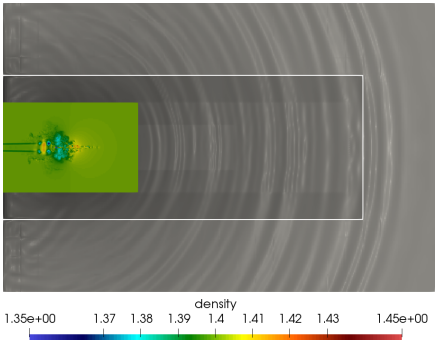
NSE – EE coupling interface In order to take a closer look at the NSE–EE coupling interface, Figure 5.11 presents acoustic waves traveling across the NSE–EE coupling interface at multiple points early in simulation time: Figure 5.11a shows the wavefront just before reaching the right coupling interface, Figure 5.11b when the first wave touches the right coupling interface, and Figure 5.11c after the wave has reached the coupling interface and is propagating over it. In these Schlieren visualizations, the internal level jumps of the NSE domain are also visible. From this figure we can derive how the jet evolves at early timestep and how acoustic wave generation starts early on. Please note, that for stability reason the inlet velocity is ramped up to $Ma = 0.5$ in the first $30\,tu$ of the simulation, which will be described later in the initial conditions of the testcase.



(a) t=24

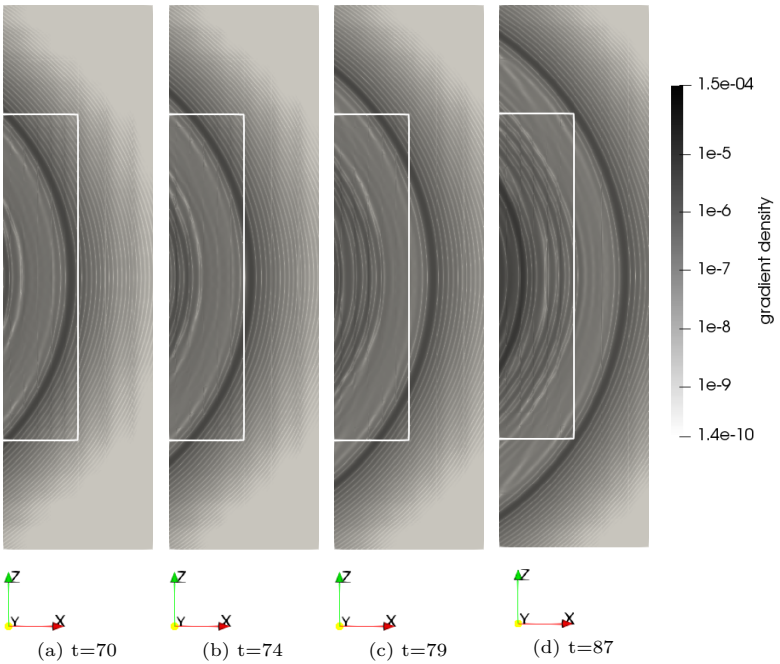


(b) t=25



(c) t=26

Figure 5.11.: Schlieren visualization (gradients of density in xz-plane) showing the acoustic waves traveling across the NSE–EE coupling interface at different timesteps. The evolution of the jet close to the inlet is illustrated with density field in rainbow color scale.



(a) t=70 (b) t=74 (c) t=79 (d) t=87

Figure 5.12.: Schlieren visualization (xz-plane) showing the acoustic waves traveling across the EE–LEE coupling interface (indicated by white lines) at different timesteps.

EE - LEE coupling interface Figure 5.12 presents the acoustic waves traveling across the EE-LEE coupling interface at the timestep before the first significant wave hits the coupling interface up to the timestep when the first waves have successfully propagated across this interface, and finally across the corners of the coupling interface. The value range of the logarithmic scale is narrowed compared to the previously shown Schlieren visualizations to focus on the data at the interface. The time evolution shows that the acoustic waves propagate without disturbance or artifacts from the coupling interface.

These figures emphasize that coupling until this point in simulation time ($t = 120$) and the chosen coupling interface are physically allowed. Now,

we will describe the testcase in detail, explain the coupling setup, and investigate the performance for such large scale simulations.

5.2.2. Testcase Setup

To decrease the computational demand for the simulation of such a large-scale scenario, we try to solve the physics only where required. In order to properly take into account the viscous effects, we need to compute the Navier-Stokes equations. When viscosity has no longer any influence, we can switch to Euler equations and solve for inviscid flow with non-linearity. Once only linear phenomena occur, we switch to the linearized Euler equations. Switching as soon as possible from non-linear to linear is promising in terms of computational demand: As described in Section 4.2, the solver *Ateles* is a modal DG solver that requires modal-to-nodal transformations in dedicated cases. In case of linear equations, the computation can be done entirely in modal space, except for applying boundary conditions where a nodal representation is required. The workload per timestep for a linear equation is $\mathcal{O}(nElems \cdot p^d)$, where d is the number of dimensions, $nElems$ the total number of elements and p the polynomial order. For non-linear equations, in contrast, a modal-to-nodal transformation is required for the flux computations. Depending on the transformation, L_2 or FPT, this results in a workload of $\mathcal{O}(nElems \cdot p^{d+1})$ or $\mathcal{O}(nElems \cdot p^d \log p)$, respectively. Nevertheless, the coupling interfaces are determined by the physics of the jet. Hence, we split the computational domain into an NSE, an EE, and an LEE domain: The viscous NSE domain is the smallest, elongated domain solving the bulk viscous flow. It is surrounded by the EE domain, that is almost twice as large in x- and four times in y- (and z-) direction, where non-linear inviscid flow is computed. The LEE domain is the outermost square and largest domain where only linear wave equations are simulated. Figure 5.13 depicts a 2D sketch of this setup that is not to scale. The black lines indicate coupling interfaces. Thus, we have a coupling interface between the NSE and the EE domain, as well as between the EE and the LEE domain, respectively. The outermost black dashed lines depict the outflow boundary of the LEE domain. The computational domain is $240 \times 240 \times 240$ *lu* in total. The NSE domain includes the regions where the gradient of the velocity is not zero, while the EE domain contains all non-linearities. To make sure that the coupling is physically allowed, we chose each domain to rather be too large than too small (see previous section). Next, we give all information of the simulation setup considering parameters, initial as well as boundary conditions, and the numerical discretizations.

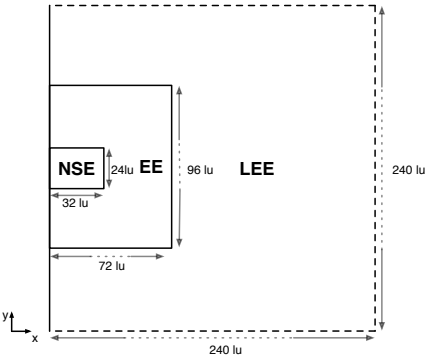


Figure 5.13.: 2D sketch of the coupling setup of free-stream jet for monolithic-like setup **A** (not to scale). Black lines indicate coupling interfaces, the outermost black dashed lines are outflow boundary conditions. Lengths in *lu*.

Material Table 5.7 defines the flow parameters for the different flow regimes.

Initial condition As initial condition, the flow is set at rest which is equal to the background state of the acoustic domain. The velocity of the jet at the inflow with the amplitude $\tilde{v}_x = Ma \cdot c$ is defined as

$$v_x = \tilde{v}_x \cdot 0.5 \cdot \left(1 + tanh\left(\frac{r_0 - |y - y_0|}{2d}\right)\right)$$

where the tanh function smoothes the pulse in y-direction. The jet radius is $r_0 = 0.1$ and the jet center is moved in y-direction by $1e^{-4}$ to induce asymmetry which accelerates the development of more vortices in the flow. The momentum thickness is $d = r_0/20 = 5e^{-3}$. The velocity in y-direction at the inlet is set to zero. To diminish the initial shock of the jet streaming into a fluid at rest, the inflow velocity is linearly ramped up to the full amplitude during the first 30 *tu* of the simulation. The density at the inflow is adapted to the inflow velocity by means of the Crocco-Busemann relation [86]:

$$\rho = \tilde{\rho} \cdot \left(1 + \frac{\gamma - 1}{2} \cdot Ma^2 \cdot \frac{v_x}{\tilde{v}_x} \cdot \left(1 - \frac{v_x}{\tilde{v}_x}\right)\right)$$

viscous flow (NSE)	dynamic viscosity	μ	$4.0\text{e}^{-6}\,\text{m}^2\,\text{s}^{-1}$
	thermal conductivity	λ	$5.92\text{e}^{-3}\,\text{W}\,\text{m}^{-1}\,\text{K}^{-1}$
	Prantl number	Pr	0.6621
	Reynold number	Re	12 500
viscous/ inviscid flow (NSE/EE)	isentropic coefficient	γ	1.4
	specific gas constant	R	280.0
	specific heat	c_p	980
	Mach number	Ma	0.5
acoustics (LEE)	background density	ρ_0	1.4
	background velocity	$\mathbf{v}_0$	$\left[0.0, 0.0, 0.0\right]^T$
	background pressure	p_0	1.0
	speed of sound	c	1.0

Table 5.7.: Parameters for the individual equations for the jet testcase.

Boundary conditions The black lines in Figure 5.13 mark all coupling interfaces that are either coupling NSE with EE or EE with LEE. According to Section 3.4, at the coupling interfaces all state variables are exchanged and for NSE the gradients in normal direction are communicated as well. The outer boundary conditions of the surrounding acoustic domain are Dirichlet boundaries where all perturbation is set to zero.

Numerical framework All simulations are performed with the DG solver *Ateles*. For the transformation from modal to nodal space, a fast polynomial transformation (FPT) is used with no extra dealiasing factor. The individual numerical resolutions in space are presented in Section 5.2.3 and Section 5.2.6. For the discretization in time, we use the explicit second-order Runge-Kutta method for all simulations. As coupling approach, we use the presented integrated approach *APESmate* and the multi-solver approach with *preCICE*. Considering *preCICE*, for non-matching coupling interfaces the NP mapping is chosen. In order to stabilize the monolithic simulation that computes NSE, we apply a positivity filter in the monolithic simulation. We also apply a positivity filter in the NSE domain in the coupled setup, whereas in the EE domain a weak modal cutoff filter with an order of 50 is sufficient and less costly. The LEE domain does not

need any stabilization.

5.2.3. Numerical resolution A: Monolithic-like setup

In the following, we compare a monolithic simulation against a coupled simulation while focusing on performance. First, we present a monolithic-like setup, called setup **A**. Since it is not possible to change the order of the numerical scheme between different regions in a monolithic simulation, we use the same numerical order everywhere. In order to have the monolithic and coupled simulations as similar as possible, we also use the same order in every subdomain for the coupled setup. We set the order to $\mathcal{O}(8)$ as a compromise between a low order for the flow domain but a high order for the acoustic propagation in the far field.

Solving the entire domain with a very fine computational grid, which is

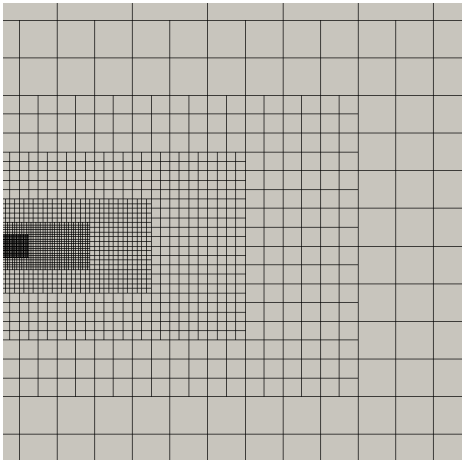


Figure 5.14.: Multi-level mesh ($\mathcal{L}(12)$ up to $\mathcal{L}(7)$) of the viscous flow, showing mesh level jumps where neighboring elements are coarsened or refined by a factor of two.

required for the viscous flow close to the nozzle exit, is not feasible due to vast computational cost. This cost can be reduced by coarsening the mesh in regions of the domain where it is physically allowed, e.g. far away from the nozzle. Therefore, most numerical solvers support varying grid sizes. We call such a mesh with different grid sizes within one domain "multi-level mesh" [5]. For *Ateles*, which is based on a cartesian octree grid, the mesh level $\mathcal{L}$ indicates the grid size. Between neighboring elements, the grid size can be coarsened or refined by a factor of two which equals to a jump of one mesh level. For the monolithic simulation, we use such a multi-level mesh

	equation	domain size [lu]	number of elements	mesh levels $\mathcal{L}$	element size h	spatial order $\mathcal{O}$
mono	NSE	240×240^2	137 964	12−5	$6.25e^{-2}$−8.0	8
coupled	NSE	32×24^2	102 180	12−7	$6.25e^{-2}$−2.0	8
	EE	72×96^2	10 080	6	4.0	8
	LEE	240×240^2	25 704	5	8.0	8

Table 5.8.: Individual domain and numerical discretization specifications of monolithic-like setup **A** where the numerical resolution of monolithic and coupled setups are equal. Lengths are measured in normalized length units, lu.

with $\mathcal{L}(12) - \mathcal{L}(5)$ to reduce the computational cost. This mesh contains five level jumps concentrated in the region of viscous flow, and seven jumps in total. Figure 5.14 shows the mesh for the viscous flow and these five mesh level jumps. For the coupled simulation, we want to achieve exactly the same computational discretization in element size h (= mesh level $\mathcal{L}$) as for the monolithic simulation. Hence, we use the same $\mathcal{L}(12) - \mathcal{L}(7)$ multi-level mesh in the NSE domain (Figure 5.14), where we introduce a coupling interface from the NSE to the EE domain at the level jump from 7 to 6, and at the level jump from 6 to 5, we introduce a coupling interface from EE to LEE, respectively. Table 5.8 shows the individual domain and discretization specifications of the monolithic-like setup **A**, for the monolithic, as well as for the coupled simulation. The number of elements for the monolithic simulation matches the sum of elements across all subdomains in the coupled one. All domains use the same spatial order $\mathcal{O}$and the number of degrees of freedom (DoF) for monolithic and coupled are identical with 353 187 840. Thus, the monolithic and coupled simulations differ in the following points: The replacement of two level jumps by a (non-matching) coupling interface, and the simulation of NSE in the entire domain in contrast to NSE–EE–LEE in individual subdomains.

As mentioned, we use the explicit second-order Runge-Kutta scheme in time. For the coupled simulation, the timestep of each domain is fixed to the smallest timestep of all domains: With the given flow parameters, the parabolic timestep (Equation (2.44)) of the NSE domain limits the

timestep to be less than $1.61e^{-4}$ s. The CFL condition of the EE domain requires a timestep less than $2.08e^{-2}$ s. Hence, the timestep of the NSE domain limits the timestep in the coupled setup. This is equal to the monolithic setup where the whole domain must be solved with the smallest timestep.

Since we keep the computational discretization (order $\mathcal{O}$ and element size h) the same for monolithic and coupled, with the monolithic-like setup **A** we investigate the influence of the different equations in terms of computational costs. These investigations will be done with the integrated approach *APESmate* and the multi-solver approach *preCICE*. A good load balancing strategy for the coupled scenario is a prerequisite for a large scale simulation without wasting computational resources.

5.2.4. Investigations of load balancing

In this section we analyze load balancing (LB) by looking at the aforementioned 3D jet testcase. We focus on a coupled simulation which has a great potential to benefit from LB: The individual workloads for computation and coupling steps for the monolithic-like setup **A** using the integrated coupling approach *APESmate* are investigated in detail. With *APESmate*, we also compare a single-stage to a multi-stage time integration. Subsequently, we look into load balancing with the multi-solver coupling approach *preCICE*. Finally, we compare the performance of the two coupling approaches to find the configuration which simulates the 3D jet test case most efficiently. All investigations and timings were done on SuperMuc Phase 1, IBM system at LRZ, Munich.

5.2.4.1. Load balancing with *APESmate*

In a coupled simulation we can observe two types of load imbalances (as described in Section 3.2): Load imbalances **between** subdomains that are due to different workloads from solving different equations, resolutions, or orders of the numerical scheme. Therefore, the available computational resources have to be distributed across subdomains according to their individual workload. The second type of load imbalances are imbalances **within** a subdomain that originate from boundary treatment, i.e. physical boundaries as well as coupling interfaces. Elements that have to treat boundaries are more expensive than inner elements, and coupling interfaces are even more expensive than the physical boundaries, especially where additional gradients need to be computed (e.g. for coupling to NSE).

Both types of imbalances, between and within subdomains, influence each other: Higher imbalances within a subdomain increase the run time of this subdomain which in turn can result in imbalances between the subdomains. In the following, we will investigate the load to find an optimal partitioning of computational resources for a coupled simulation.

Therefore, we investigate the factors that contribute to the execution time of an individual element: The numerical computation of equations; boundary treatment in general and coupling work in particular; communication of coupling data; communication of flux data; waiting at synchronization points; waiting in peer-to-peer communication. We identify the following reasons why a coupled simulation requires time:

(i) Load for actual numerical computation: Computation of physical flux, projection onto test functions, multiplication with inverted mass matrix, etc.,

(ii) load for coupling point evaluation,

(iii) load for communication (internal communication to neighboring elements as well as exchange of coupling data), and

(iv) waiting (also known as idling).

For this analysis, we have to distinguish between the actual load of a process **(i)-(iii)** and the idling of a process **(iv)**. The sum of loads **(i)** will be called "compute load". Time required for coupling point evaluation **(ii)** will be called "coupling point evaluation load". The LB algorithm *SPartA* re-distributes **elements** based on weights, which should obviously mirror the workload. In Section 4.2.1 we have described that measured timings of individual routines are used as weights in *Ateles*. Therefore, weights will be denoted in seconds in the following. To determine the compute weights, W_{comp}, the averaged workload **(i)**, i.e. excluding communication, is measured. For the point evaluation weights, W_{eval}, the evaluation of the polynomial **(ii)** is measured. Please note that each coupling element is involved in computation like all "inner" compute elements and, thus, has a non-zero compute weight W_{comp} as well. The loads **(i)** and **(ii)** represent the actual computation and evaluation task and, thus, are independent of the number of MPI-processes. Hence, the sum of the compute weights W_{comp} as well as the evaluation weights W_{eval} over all processes is constant for the setup.

In order to be able to balance load well, synchronization points should be avoided, since it is only possible to balance load in between them. Since synchronization points often cannot be entirely avoided, the weights

for the LB algorithm have to represent the load between them. Hence, an important conclusion from Section 3.2.2 is that load balancing that considers computation and evaluation jointly is only beneficial when these are not separated by any sort of synchronization.

Compute weights To examine the compute load per subdomain, we sum up the compute weights of all elements in a subdomain, $\sum_{i=1}^{nElems} W_{comp_i}$. The **sum of compute weights** for the subdomains of setup **A** are

	NSE	EE	LEE
sum of compute weights	61 922 s	2193 s	1893 s

,

which indicates that the NSE subdomain has the largest workload (a factor of 28 larger than EE and a factor of 33 larger than LEE), while EE and LEE show similar load with a minor 16% difference. The large difference is due to the fact that 74% of the total number of elements compute NSE, 7% compute EE, and 19% compute LEE, where the latter two are also less computationally demanding. Computing the LEE, in particular, is cheap since the equations can be evaluated fully in modal space, except for the boundary elements (see Section 4.2). In total, the compute load of the entire coupled simulation is 66 008 s.

These compute weights, W_{comp}, include internal level jumps. In case a computational domain contains level jumps (here: monolithic and NSE subdomain), they introduce additional internal load since interpolation between coarse and fine elements is required. Setup **A** is constructed such that two (of the seven) level jumps in the monolithic configuration correspond to the coupling interfaces in the coupled configuration. The workload of coupling interfaces is not included in W_{comp}, but will be investigated separately as W_{eval}. In the coupled setup, the remaining five level jumps are located exclusively in the NSE subdomain. Hence, the monolithic simulation has two more level jumps, resulting in a higher compute load. Thus, for a valid comparison we have to compare W_{comp} of the monolithic simulation with $W_{comp} + W_{eval}$ of the coupled simulation, which we will present in the paragraph about total weights.

Coupling point evaluation weights Besides compute load, there are elements at the coupling interface with additional load due to coupling. To obtain the load of this coupling point evaluation, additional weights are

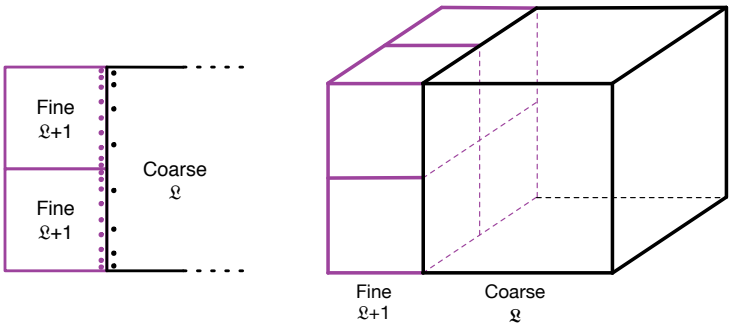
introduced. These weights, W_{eval}, measure the evaluation of the polynomial in the DG solver per element, as described in Section 3.3.2. Several factors influence the cost of polynomial evaluation:

a) The polynomial order,

b) the number of points to be evaluated, and

c) whether a state only or a state & gradient evaluation is done.

For setup **A**, all subdomains use the polynomial order $\mathcal{O}(8)$ so that no influence on W_{eval} due to **a)** is expected. The number of points to be evaluated, **b)**, varies for the coupling elements for two reasons: The number of requested points and the location of the coupling element. The number of requested points depends on the polynomial degree of the requesting subdomain (Section 3.3.1) and the number of "counter elements" that need to be served. "Counter elements" describe the elements in the opposite subdomain whose coupling points have to be served. For setup **A** the polynomial order p is constant and set to $p = 8$. Therefore, we have $n = p^2 = 64$ coupling points per element per coupling face for the 2D interface. The number of counter elements depends on the element size and the level jumps at the coupling interface. Figure 5.15 presents a coupling



(a) 2D sketch (b) 3D sketch

Figure 5.15.: Coupling interfaces with a one-level jump. Black dots illustrate coupling points (shifted to the left and right of the coupling interface for illustration purposes).

interface with a one-level jump. In 3D, one coarse element has to serve the

points of 4 elements (instead of 2 elements in 2D), and one fine element has to serve the points of $\frac{1}{4}$ elements (instead of $\frac{1}{2}$ elements in 2D). This results in one element having to provide coupling data for several elements or only a fraction of an element of the other subdomain. The black dots in Figure 5.15a illustrate coupling points that are shifted to the left and right of the coupling interface for illustration purpose, in reality the points are lying directly on the interface. With the presented different number of counter element to serve, a coupling element has to evaluate different number of coupling points which are listed in Table 5.9.

coupling interface	corner elements	edge elements	plane elements	coupling elements (total)	coupling points per coupling face per element	total number of coupling points for **state** evaluation	total number of coupling points for **gradient** evaluation
NSE–EE	4	100	700	804	16	14 592	0
EE–NSE	0	0	112	112	256	28 678	28 678
EE–LEE	4	156	1980	2140	16	36 864	0
LEE–EE	0	0	380	380	256	97 280	0

Table 5.9.: Element configuration at the coupling interfaces of setup **A**, classified into into corner, edge, and plane elements.

Considering the location of the coupling elements, we can state that a 3D coupling element can have one, two, or three coupling faces. In case an element has no coupling face, it is not a coupling element according to our definition. An element with one coupling face is, for instance, located at the plane of a 2D coupling interface. An element with two coupling faces is located at the edge of the 2D coupling interface, resulting in two faces that have to serve the other subdomain. In this case, the element has to serve twice the number of points compared to plane elements of the same coupling interface. An element with three coupling faces is located at one of the corners of the 2D coupling interface. Table 5.10 gives an overview of the number of total, compute and coupling elements per subdomain

subdomain	total elements	compute elements	coupling elements
NSE	102 180	101 376	804
EE	10 080	7828	2252
LEE	25 704	25 324	380

Table 5.10.: Number of total, compute and coupling elements at the coupling interfaces of setup **A**.

of setup **A**. Compute elements are "inner" elements without boundary treatment. Additionally, Table 5.9 presents the element configurations per coupling interface, where coupling elements are classified into corner, edge, and plane elements. The names of the coupling interface, e.g. NSE–EE, are chosen in such a way, that the first subdomain, e.g. NSE, is the subdomain that has to evaluate the coupling points and, thus, is the subdomain with the additional weight W_{eval}. The second subdomain, e.g. EE, is the subdomain that requests the coupling points and requires the coupling data.

Table 5.9 presents the number of requested points that require a gradient evaluation in addition to the state evaluation in the last column. For the jet setup, only the EE elements at the NSE interface have to evaluate state & gradient. This is the third factor **c)** that influences the load of a coupling element, and has the biggest impact.

The total evaluation weight of a subdomain depends on the number of coupling elements and the evaluation weight per element. With the setup **A** and the presented element configuration of Table 5.10 and Table 5.9, the **sum of coupling point evaluation weights** for each subdomain are

	NSE	EE	LEE
sum of coupling point evaluation weights	30 s	690 s	207 s

.

These evaluation weights show that the NSE subdomain has the lowest coupling load, while the load of the LEE subdomain is six times higher, and the EE subdomain has the highest load with its state & gradient evaluation and two coupling interfaces. This can be expected from Table 5.9: Considering only state evaluation NSE has the smallest number of points to be evaluated. LEE has 6.5 times the number of points of

NSE, which leads to a proportionally times higher evaluation load. For the EE domain, the number of points for state evaluation (EE–NSE **and** EE–LEE) is 4.5 times higher than NSE, which would result in an evaluation weight of around 135 s, but EE has 28 678 points that additionally require gradient evaluation. Usually, it can be assumed that the number of points effects the workload for a polynomial evaluation. This is true for the state evaluation, but not entirely for the gradient evaluation: As presented in Appendix A, we have developed an *optimized* implementation that is based on an element-wise evaluation. This means, that some parts of the gradient evaluation are performed once for an element and are reused for all coupling points located in this element. Hence, not only the number of coupling points but also the combination of points within elements influences the evaluation load (for a detailed analysis see Appendix A and Figure A.3).

The high evaluation weights of the EE subdomain are analyzed in more detail. Figure 5.16 illustrates the evaluation weights of the EE subdomain

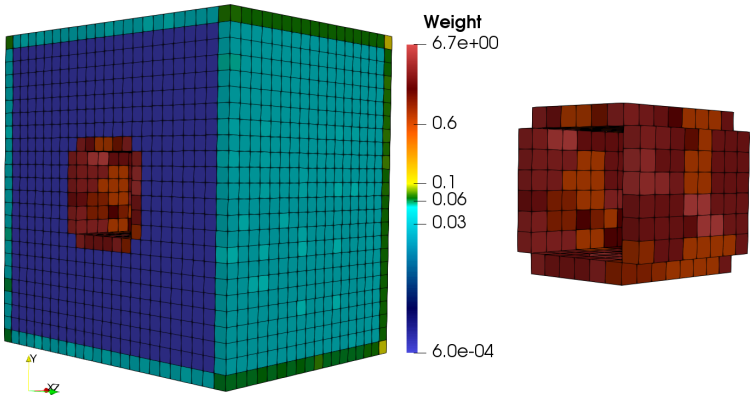


Figure 5.16.: Coupling weights of both EE coupling interfaces. Left shows the entire subdomain and right is zoomed to the EE–NSE interface which shows an increased evaluation load.

with logarithmic scale. On the left side, the entire EE subdomain is shown where the front "with the whole" is the yz-plane of the physical boundary of the EE subdomain. At the interface between EE and the surrounding LEE subdomain, the different weights for plane (one face), edge (two faces), and corner (three faces) elements can be distinguished: The weight for a

plane element with one coupling face are around 0.03 s (cyan color); the weight for an edge element with two coupling faces are around 0.06 s (green color); a corner element with three coupling faces shows a weight of around 0.1 s (yellow color). The dark blue elements are "inner"elements, that do not generate coupling load. The right side of Figure 5.16 shows only the coupling elements towards the inner NSE subdomain. Here, we see that the weight for state & gradient evaluation elements (red color) is by more than an order of magnitude higher than all other weights. Please note that the weights are based on timers and during performance measurements fluctuation of the timings were noticeable. All of the aforementioned points explain the high coupling costs of the EE subdomain.

Total weights In order to obtain the total load of a subdomain, we consider the sum of compute and coupling point evaluation load. Compared to the compute weight, the evaluation weight for NSE is significantly smaller at only 0.048%. For the EE subdomain, the evaluation weight is around 30% compared to the compute weight and for the LEE subdomain the evaluation weight is only 10% compared to the compute weight. The **sum of total weights** for the individual subdomains are

	NSE	EE	LEE
sum of total weights	61 952 s	2883 s	2100 s

.

The total weights of the coupled simulation sum up to 66 935 s. The monolithic simulation, in contrast, has a total weight of 83 250 s which is about 24% more expensive. As previously mentioned, the monolithic simulation has internal level jumps, where the coupled simulation uses coupling interfaces. Costs for internal level jumps are included in W_{comp} while costs of coupling evaluation are included in W_{eval}. Hence, we have to compare W_{comp} of the monolithic simulation with $W_{comp} + W_{eval}$ of the coupled simulation for a valid statement of loads. The conclusion is that the difference of 24% in load is due to the fact that the monolithic simulation computes NSE in the entire domain.

Table 5.11 summarizes the previously presented weights for the subdomains. Regrading the distribution of the total number of process across the subdomains, we have to consider the total weights. Hence,

	NSE	EE	LEE
sum of compute weights	61 922 s	2193 s	1893 s
sum of evaluation weights	30 s	690 s	207 s
sum of total weights	61 952 s	2883 s	2100 s

Table 5.11.: Weight distribution **between** the subdomains: Sum of weights for compute, coupling point evaluation, and total as sum of the subdomains. All weights in s.

the number of available processes should be distributed according to $\frac{\text{total weight of subdomain}}{\text{sum total weight of all subdomains}}$. This results in 92.6% of available processes for NSE, 4.3% of available processes for EE, and 3.1% of available processes for LEE subdomain and will be present in the following paragraph for load balancing.

Weight ranges within a subdomain With the presented weights per subdomain, we can balance the load between subdomains. As the loads per element differ for coupling and non-coupling elements, we will look at the load ranges within a subdomain in the following.
The compute weights per element do not differ within a subdomain, except for the elements involved in internal level jumps and physical boundaries, since all elements have to compute the same equation and numerical scheme. In the NSE domain, there are internal level jumps. The physical boundaries for NSE and EE do not have additional impact, since these equations anyway require nodal-to-modal transformations for non-linear terms. For the LEE domain, the physical boundary elements have to do an additional nodal-to-modal transformation. The **compute weights range** is

	NSE	EE	LEE
compute weights range	[0.55 s, 0.65 s]	[0.19 s, 0.21 s]	[0.06 s, 0.08 s]

.

Considering the coupling point evaluation weights, the range is larger: The minimum weight is zero, since all “inner”, non-coupling elements have no evaluation weight. The **coupling point evaluation weights range** in the coupled setup **A** are measured to

	NSE	EE	LEE
coupling point evaluation	[0.0 s, 0.1 s]	[0.0 s, 6.7 s] (EE–NSE)	[0.0 s, 0.55 s]
weights range		[0.0 s, 0.12 s] (EE–LEE)	

.

For the EE subdomain, two ranges are listed since the EE subdomain has two coupling interfaces, EE–NSE and EE–LEE, respectively. The maximum evaluation weights differ so significantly due to the already presented reasons: Mesh level jumps at the coupling interfaces (see Figure 5.15), the different element locations (see Table 5.9), and the additional gradient computation at the EE–NSE interface (see Table 5.9). In detail the maximum weights can be explained as follows:

NSE All NSE elements need to couple to a mesh that is one level coarser, which means that 1 NSE element serves only $\frac{1}{4}$ of the coupling points of one EE element. The NSE element with the largest coupling load is the corner element that has 3 times the coupling points to serve than the other NSE elements.

EE The EE subdomain has two coupling interfaces: At the EE–NSE interface the EE subdomain needs to provide the gradients of the coupling variables (red elements in Figure 5.16) and, additionally, couple to a one-level finer mesh s.t. 1 EE element has to serve 4 NSE elements. Therefore, these coupling elements have the highest evaluation weight within the EE subdomain. At the EE–LEE interface, there is a one-level jump and, thus, an EE element has to serve points of $\frac{1}{4}$ of an LEE element. According to Table 5.9, the EE–LEE interface has elements of different location categories, where the corner elements are the most expensive category among the EE to LEE elements.

LEE The LEE subdomain only has plane elements with one coupling face. Hence, all coupling elements have the same amount of coupling load of around 0.55 s. For completeness, all LEE elements couple to a one-level finer mesh resulting in 1 LEE element serves points of 4 EE elements.

With these facts, we can look at the total weights range within a subdomain. The largest share are, as expected, the coupling weights, since all elements have to perform calculations. The minimum total weight of all elements is the minimum compute weight for an “inner” element. While

the total weight is the sum of compute and coupling point evaluation weight, the **maximum** total weight over all elements is not necessarily the maximum of all compute weights and the maximum of all evaluation weight. For example, a coupling element with high evaluation load is typically not involved in a internal level jump which increases the compute load. The **total weights range** is

	NSE	EE	LEE
total weights range	[0.55 s, 0.73 s]	[0.21 s, 6.79 s]	[0.08 s, 0.6 s]

.

	NSE	EE	LEE
compute weights range	[0.55, 0.65]	[0.19, 0.21]	[0.06, 0.08]
coupling point evaluation	[0.0, 0.1]	[0.0, 6.7] (EE–NSE)	[0.0, 0.55]
weights range		[0.0, 0.11] (EE–LEE)	
total weights range	[0.55, 0.73]	[0.21, 6.79]	[0.08, 0.6]

Table 5.12.: Weight distributions **within** the subdomains: Minimum and maximum weights for compute, coupling point evaluation, and total. All weights in s.

Table 5.12 summarizes the previously presented minimum as well as maximum weights and, thus, the weight distribution the subdomains. As described in Section 3.2.2, the most expensive element, when compared to the averaged load, limits the sensible number of processes to be used. The LB algorithm tries to distribute the overall workload of the subdomains so that all partitions have the same workload. The ratio between coupling evaluation load and compute load in a subdomain is mainly determined by the ratio of coupling elements to non-coupling elements. For coupling elements, the type of polynomial evaluation is crucial since it increases the load of the coupling point evaluation. The limiting case is reached when we want to use more processes than the sensible number and no workload is left to be further distributed. For setup **A**, the limiting subdomain is the EE subdomain, due to the high cost of gradient evaluation with a

one-level jump and a ratio of coupling to compute elements of $\frac{1}{3}$. This limitation highly depends on the simulation setup.

Communication load The next type of load of an element that we focus on is communication. In general, a simulation does not necessarily involve communication: Communication is only required when distributing a problem across multiple processes to solve it in parallel. How much communication is required depends on the number of processes and the numerical approach. Often, communication load is not included in weights for load balancing since it is not an actual workload, but a necessity for parallel computation. Additionally, when measuring communication timings, it is hard to distinguish between actual communication and waiting in point-to-point communication. Therefore, we do not include communication load as weights in the LB algorithm.

Nevertheless, we will discuss where communication is required. **Within** a subdomain, each element communicates with its direct neighbors in order to exchange flux information during DG computation. Additionally, MPI-reduce communication takes place for all elements to exchange the local time step and simulation status. In case all elements have the same compute load, the waiting timings in communication are minimized and the communication load only depends on the message size. Since all communication elements within a subdomain, communicate the same kind of information, the communication load can be considered a constant offset. This offset is irrelevant for the LB algorithm since it only considers weights relative to each other. **Between** subdomains, coupling elements communicate with their counter element(s) to exchange coupling data. Here, the total communication load depends on the number of coupling elements and their coupling data which determine the message size. Hence, the message size of an element changes with the number of points to serve and whether it is state evaluation only or state & gradient evaluation. In particular, the number of messages changes whether it is a plane, edge or corner element: An edge element sends two messages of the same message size as a plane element; and a corner element sends three messages of the respective size. Nevertheless, we assume the point-to-point communication to be significantly smaller than the polynomial evaluation (depending on its order) and, therefore, we neglect this contribution to the imbalances. For coupling, it is important to minimize the communication during simulation which is done with a thought out initialization routine (see Section 4.4.3). The main aspect is, that all information that can be gathered a priori and

does not change during simulation are stored per process. Based on this, the number of messages during simulation can be reduced.

Load balancing to minimize waiting times After discussing the actual workload and the communication load, we look at the waiting times in coupled simulation and how they occur. Waiting times typically arise during communication and at synchronization points. As mentioned in the previous paragraph, we have communication **within** subdomains and coupling communication **between** the subdomains. Within a subdomain, waiting can occur in the point-to-point communication between elements which indicates bad load balancing within the subdomain. In the coupling step, waiting times can occur in the exchange of coupling data. For example, when an LEE element has to wait for an EE element to reach the coupling step to be able to receive the requested data. This is an indicator for bad load balancing between the subdomains. The other possibility for waiting times of an element is at synchronization points: When an element is done with its tasks after one timestep, there is a synchronization point in *Ateles* where all elements wait for each other (e.g. to synchronize the timestep, check of the simulation status, etc). An element without coupling load is faster and will reach this final synchronization point faster than elements with coupling load. Therefore, the goal of load balancing is to gather elements without coupling and elements with coupling onto a number of processes so that in sum each process needs about the same time to reach the final synchronization point. This is equivalent to avoiding, or at least minimizing, waiting times. We evaluate the waiting times at this synchronization point as indicative numbers to evaluate how well the LB within and between the subdomains is.

In order to find the load balancing strategy that minimizes waiting times, we investigate setup **A** on 4096 MPI-processes for 100 timesteps. Table 5.13 presents the execution times for different partitioning strategies with *APESmate* including the waiting times at the synchronization point. Please note that the execution time is presented as "wall-clock time" while the waiting times are given as "cpu time". That is, the waiting times are aggregates of the elapsed time over all coupling elements and all processes. Hence, it cannot be stated whether some elements are waiting for a long time, or if there are many elements waiting for a short time. However, a significantly large number indicates more waiting in sum and, thus, worse load balancing. For a better comparison, we list the the sum of waiting times ("cpu time") averaged over the total number of coupling elements as

Partitioning strategy	Subdomains	Processes	Execution time [s]	Waiting times [s] Sum	Waiting times [s] Element averaged
1)	NSE	3034		78 100	97.1
elements	EE	299	**99.6**	25 500	11.3
	LEE	763		73 100	192.4
2)	NSE	3791		217 000	269.9
total weights	EE	176	**114.8**	13 600	6.0
	LEE	129		243 000	639.5
3)	NSE	3791		80 300	99.9
total weights	EE	176	**80.3**	7250	3.2
+ *SPartA* LB	LEE	129		7580	19.9

Table 5.13.: Waiting and execution times (without initialization phase) for different partitioning strategies: 1) using number of elements per subdomain for the distribution of processes, 2) using weights for the distribution of processes, 3) using weights and re-partitioning with *SPartA* running 100 timesteps of setup **A** with *APESmate*. Execution times are "wall-clock time". Waiting times are "cpu-time".

"element averaged waiting time". Nevertheless, the most important times are the execution times as "wall-clock time" since this is the elapsed time between the first timestep of the simulation and reaching 100 timesteps (without initialization). We use three different partition strategies to distribute the total number of 4096 processes over the three subdomains (NSE, EE, LEE):

1) Partitioning **between** the subdomains according to the **number of elements per subdomain** and **within** subdomains according to the **number of elements**: According to the number of elements (102 180, 10 080, 25 704) each subdomain gets 74.1%, 7.3%, and 18.6% of the 4096 MPI-processes, respectively. Within the subdomains, the elements are equally distributed. As the costs per element differ strongly for NSE, EE, and LEE, as well as for coupling and non-coupling elements, this distribution cannot lead to a good load balancing between subdomains. Table 5.13 shows the high waiting times resulting from these imbalances.

In particular the LEE subdomain, with its cheap computation, results in high averaged waiting times.

2) Partitioning **between** subdomains according to **total weights per subdomain** and **within** a subdomain according to **number of elements**: As presented in Table 5.12, the total weight (computation + evaluation) per subdomain differ and according to the actual weights the NSE subdomain requires 92.6% of the resources, the EE and LEE subdomains 4.3% and 3.1%, respectively. Keeping the overall number of MPI-processes constant to 4096, the distribution of processes is 3791, 176, and 129, respectively. With this, the waiting times for NSE and LEE increase by a factor of 2.8 and 3.3, respectively, and the execution time escalates by 15% (Table 5.13). Considering each subdomain individually, the NSE subdomain should be faster since it has more processes, while the EE and the LEE subdomain take more computation time since they have less processes compared to *1)*. Coupling them, the overall execution time should decrease since load between the subdomains should be more balanced and smaller waiting times are to be expected. The problem, however, is that **within** the subdomain, the elements are still equally distributed and the high evaluation load of elements at the coupling interface results in waiting times for the other subdomains (to receive their coupling data). The missing key is load balancing within subdomains that leads to element distributions according to the workload, in particular in the EE subdomain.

3) Partitioning **between** the subdomains according to **total weights per subdomain** and **within** subdomain according to **total weights per elements** by applying *SPartA*: The aforementioned number of processes per subdomain is chosen. Table 5.13 shows that the waiting times for EE and LEE decreased by a factor of 3.5 and 9.6 compared to strategy *1)*, and that they are the smallest of all strategies. The waiting times for NSE increase slightly but we cannot determine how many elements are actually waiting. Within the NSE subdomain, the elements are distributed according to their weights which leads to a better balancing so that coupling elements can finish faster which results in less waiting of the EE subdomain. Compared to *2)* where the same number of processes is used, the waiting times decrease by a factor of 2.7, 1.9, and 32, respectively. This partitioning strategy *3)* shows the best overall performance of the coupled simulation. The execution time for 100 timesteps decreased by 30% compared to *2)* and by 19% compared to *1)*.

In contrast to the coupled simulation, the monolithic simulation needs

120 s for 100 timesteps, which is more than any of the coupled simulations. The execution time of the fastest coupled simulation with partitioning strategy *3)* is 34% less than the monolithic simulation.

Using a load balancing algorithm can prolong the initialization phase due to additional computations required for re-partitioning and, subsequently, the actual re-partitioning. Table 5.14 shows the time spent in the

Partitioning strategy	Initialization time of *APESmate* [s]
1) elements	4.885
2) total weights	5.805
3) total weights + *SPartA* LB	13.89

Table 5.14.: Time spent in the initialization phase of *APESmate* for the different partitioning strategies.

initialization phase of *APESmate* for the entire simulation. We observe that re-partitioning with the LB algorithm *SPartA* increases the time by a factor around 2.6. Considering large scale simulations, the initialization phase becomes negligible so that all aforementioned execution times exclude the time for the initialization process. An "iterative" use of *SPartA* where we restart the simulation with new weights after certain numbers of timesteps is evaluated in Appendix D.

Well-balanced coupled simulations with *APESmate* We propose the following workflow for setting up a well-balanced coupled simulation: First, use partitioning strategy *1)* to partition between the subdomains according to the number of elements per subdomain and within subdomains according to the number of elements. Run the simulation for a small number of timesteps to determine the set of weights per element. Subsequently, apply partitioning strategy *3)* to partition between the subdomainss according to the total weight per subdomain and within subdomains according to the total weight per element by applying *SPartA*. The partitioning strategy *2)* is not required as the presented results show. As mentioned before, the weights are based on timings that are highly dependent on the hardware setup (the machine, partition, CPUs). Hence, the weights do not necessarily translate between system so that they have to be measured for a new target system.

5.2.4.2. Single-stage vs. multi-stage time integration

All previously shown results were done with a second-order Runge-Kutta (RK) method which is a multi-stage time integration scheme (Section 2.2.2). As already discussed in Section 3.2.2, a multi-stage time integration utilizes two sub-stages to increase the order in time. These two sub-stages are constructed with a midpoint $t + \frac{\Delta t}{2}$ for each timestep. Hence, a second-order RK method has two computation steps each concluded by a synchronization step. A single-stage time integration like the first-order forward Euler scheme has only one computation step. The synchronization at the midpoint in the second-order RK method influences the load balancing within the subdomain. The load balancing based on total weights (computation + coupling) works well for single-stage integration, whereas it introduces imbalances at the synchronization step at the midpoint of the second-order method. As presented in Figure 3.7 in Section 3.2.2, when balancing according to total weights, coupling elements are idling at the end of the first sub-stage. Considering the overall simulation, this balancing strategy is superior to balancing according to the compute weights only, since it leads to idling of all compute elements at the end of the second sub-stage of the second-order RK scheme. It also results in smaller overall simulation time. We assume that the maximum of waiting times in the internal communication over all elements represents this idling of the coupling elements at the midpoint. Therefore, Table 5.15 presents these

time integration method	subdomain	maximum communication time [s]
forward Euler	NSE	32.2
	EE	3.3
	LEE	4.4
Runge-Kutta	NSE	62.8
	EE	10.7
	LEE	11.5

Table 5.15.: Maximum communication time for first-order forward Euler scheme and second-order Runge-Kutta scheme when balancing based on total weights (computation + coupling).

maximum communication times for the individual subdomains for the first-order forward Euler scheme and the second-order RK scheme. Ideally,

the internal communication of the second-order RK require twice as much time as the internal communication of the forward Euler scheme. Since data to communicate, partners to communicate with, and partition of elements are similar, the deviation can be explained by processes waiting for their communication partner to be ready to communicate. Table 5.15 shows that for the NSE subdomain the maximum communication time is almost doubled. For the EE subdomain the maximum communication time of the coupling element is a factor of 3.2 larger and for the LEE subdomain a factor of 2.6. This indicates idling processes in a balanced simulation with multi-stage time integration, as already predicted in Section 3.2.2.

5.2.4.3. Load balancing with *preCICE*

The multi-solver approach uses the external coupling library *preCICE* that works exclusively on geometry data of the coupling interfaces. However, most load balancing algorithms are based on **element** weights (as previously discussed). In order to distribute the compute and coupling workload jointly, both types of workload must be measured, which is hardly possible within *preCICE*: After computation, point data at the interface are provided by the solver *Ateles* to *preCICE*. Subsequently, the association between points and elements is lost and data mapping as well as communication within *preCICE* is done **per process**. While element weights for computation can be measured by the solver, the element weights for coupling cannot be measured. As described in Section 4.3.2, the solver has two extra calls to *preCICE* for writing data to and reading data from *preCICE* in every timestep. The routine *Write to preCICE* is expensive, as shown in Section 4.3.6. In *Ateles*, we measure the timings for these two extra calls to *preCICE* as **coupling weights** and include them in the **total weights** per element. Hence, the load balancing algorithm *SPartA* distributes according to total weights that are based on computation of the numerical scheme in *Ateles* and the solver-internal coupling cost due to *preCICE* calls. Table 5.16 presents execution times for different partitioning strategies similar to the investigations for *APESmate* (Table 5.13). The first line in the partitioning strategy describes the distribution between the subdomains and *SPartA* defines whether load balancing within the subdomains is used. Since we are running a multi-solver approach, each subdomain has its individual runtime which should be similar to each other to be efficient. Therefore, each run has three execution times and the maximum time is pivotal. Six different strategies are used, which already shows that load balancing is not that simple with *preCICE*.

Partitioning strategy	subdomain	processes	execution time [s]
1) elements	NSE	3034	**107.4**
	EE	299	99.8
	LEE	763	105.7
2) total weights	NSE	3795	**155.8**
	EE	172	137.6
	LEE	129	78.4
3) total weights	NSE	3795	**151.5**
+ *SPartA* LB	EE	172	149.9
	LEE	129	75.8
4) computation and	NSE	1816	163.5
coupling times	EE	1113	**165.8**
+ *SPartA* LB	LEE	1167	165.3
5) computation times	NSE	3979	**222.2**
+ *SPartA* LB	EE	69	205.3
	LEE	48	78.3
6) elements	NSE	3034	94.7
+ *SPartA* LB	EE	299	**94.9**
	LEE	763	93.5

Table 5.16.: Computation times (without initialization phase) of different partitioning strategies running 100 timesteps of setup **A** using the multi-solver approach *preCICE*. First line in the partitioning strategy describes the distribution **between** the subdomains, and *SPartA* is used for load balancing **within** the subdomains.

We start with the naive approach and distribute the total number of processes between the subdomains according to the number of elements in each subdomain. Next, we use the total weights per subdomain to distribute the processes according to the workload. Strategy *3)* maintains the same distribution of processes and utilizes *SPartA* for load balancing within each subdomain. The maximum execution time does not reduce as expected for the aforementioned strategies. Please note that strategies *1) - 3)* are similar to the load balancing with *APESmate*, except that the

measuring of the coupling weights differs due to the black box *preCICE*. Therefore, strategy *4)* partitions the processes according to computation times (including internal communication) and the coupling times (times of all processes per subdomain spent in *preCICE*) and uses *SPartA* for load balancing within the subdomains. This shifts processes from the NSE subdomain to the EE and LEE subdomains, which increases all execution times even further. In particular for the LEE subdomain, the execution time increases by more than a factor of two due to high waiting times in *preCICE*. Hence, we neglect coupling and strategy *5)* distributes processes between the subdomains exclusively according to computation times per subdomain, which leads to an even higher maximum time. Therefore, the final strategy *6)* falls back to distributing processes according to the number of elements per subdomain (strategy *1)*), but uses load balancing within the subdomains. It is the best run of all partitioning strategies. The timings for this run are 13% lower than for partitioning according to elements *1)* and all subdomains observe the same low execution time, thus no subdomains are idling or have more workload than others. For setup **A**, the number of elements appears to be a good indicator of the overall load even if the subdomains solve different equations. *SPartA* helps to reduce the load imbalances within the subdomains due to the solver-internal coupling load. Partitioning strategy *3)* (weights + *SPartA* LB) which is working good for *APESmate* does not improve the maximum timing since the coupling load per element in *preCICE* (data mapping + communication) is not properly measured and *SPartA* cannot distribute the coupling elements. In this work, we identify partitioning strategy *6)* to be the most efficient approach for coupled simulations with the multi-solver approach *preCICE*. For further enhancements, *preCICE* would need to incorporate load balancing or, at least, provide coupling weights per element s.t. the LB within the subdomain can work properly.

5.2.4.4. Comparison of *APESmate* and *preCICE*

To sum up, Table 5.17 presents the execution times (without initialization time) for a monolithic and a coupled simulation using *APESmate* and *preCICE* for setup **A**. The results of the most efficient partitioning strategies, as identified in the previous sections (see Table 5.13 and Table 5.16), are listed. The monolithic simulation was setup to not utilize *SPartA*, since using it has increased the simulation time by 5%. As presented in Section 4.2.1, compute weights are not explicitly measured per element, but are computed as the average weight of all elements on the same process. Hence, the load of elements involved in level jumps is underestimated,

	subdomain	processes	execution time [s]
monolithic		4096	120.2
preCICE	NSE	3034	94.71
	EE	299	94.97
	LEE	763	93.53
APESmate	NSE	3791	
	EE	176	80.3
	LEE	129	

Table 5.17.: Execution times (without initialization phase) of the most efficient partitioning (Table 5.13 and Table 5.16) of the setup **A** for monolithic, coupled simulation using *preCICE* and coupled simulation using *APESmate*.

while the load of the elements on the same process without level jump are overestimated. Therefore, the monolithic simulation does not benefit from *SPartA*. For future work, it is highly recommended to explicitly measure all weights per element. The multi-solver approach with *preCICE* is partitioned according to the number of elements per subdomains and balanced with *SPartA*. The *APESmate* run is partitioned according to the weights and also balanced with *SPartA*. The execution times confirm our previous predictions: Coupled simulations are faster than monolithic simulations and the integrated coupling approach is faster than the multi-solver approach using an external library. *APESmate* speeds up the simulation by a factor of 1.5 and *preCICE* by a factor of 1.2.

5.2.5. Scalability of setup A with *APESmate*

In this section we investigate the scalability of setup **A** with *APESmate*. To achieve a strong scalability, we use the identified best load balancing configuration and balance between the subdomains according to the total weights and use *SPartA* for LB within the subdomains. Based on the resulting distribution on 4096 processes, we distribute the processes according to Table 5.18. The construction of the weights mirrors the compute and polynomial evaluation load and, therefore, are independent of the number of processes.

total	NSE	EE	LEE
1024	948	44	32
2048	1896	88	64
4096	3791	176	129
8192	7582	352	258
12 288	11 373	528	387

Table 5.18.: Number of processes for strong scaling distributed of the individual subdomains for monolithic-like setup **A**.

The maximum number of processes that can be utilized is naturally limited by the number of elements in a setup, since one element cannot be distributed across multiple processes. As described in Section 3.2.2, the most expensive element, when compared to the average load, limits the sensible number of processes to be used. For the usage of *SPartA*, this is further described in Section 4.2.1. For setup **A**, we use 12 288 as maximum number of processes (Table 5.24) for the strong scaling. This is due to the EE subdomain, where coupling elements to NSE are expensive and the ratio between maximum total weight (6.79 s) and average total weight (0.28 s) is bad.

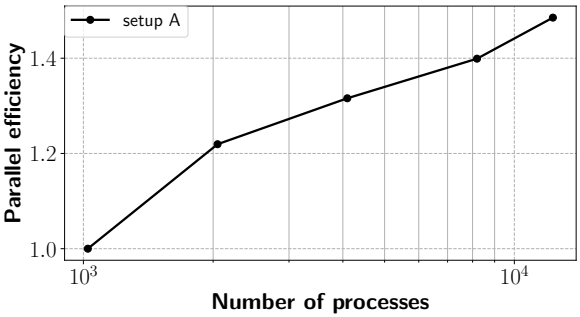


Figure 5.17.: Strong scaling for well-balanced setup **A** with *APESmate* measured for 100 timesteps.

Figure 5.17 presents the results of strong scaling for the monolithic-like

setup **A**. The simulations are performed for 100 timesteps, the consumed time excluding the initialization phase are measured, and the parallel efficiency is computed. The computation on the smallest number of ranks (1024) is used as reference, resulting in a parallel efficiency of 1. The setup shows a superlinear scaling which indicates that coupling in general does not pose a bottleneck.

5.2.6. Numerical resolution B: Tailored setup

Once an appropriate load balancing strategy has been found, we will adapt the setup by tailoring the numerical discretization to the physical regimes to exploit all benefits of a partitioned coupling approach. According to the physics, we choose a low order with a very fine grid for the viscous flow domain and a higher order with a coarse mesh for the acoustic far field. For the adaptation, we can use our knowledge about the physics of the 3D free-stream jet as well as the results from the previous Section 5.2.4.

For the tailored setup **B**, we focus on the coupled simulation. The monolithic simulation is adapted to be as close as possible to the coupled setup in order to allow for a performance comparison. Please note that the monolithic simulation will only be simulated for a limited period of simulation time to not unnecessarily consume computational resources. First, we narrow the NSE domain for the coupled simulation to a specific region, such that two different cases can be investigated: One where the turbulent jet only expands in the NSE subdomain ($t = 120$) and another example where the vorticty of the jet travels over the coupling interface into the EE subdomain ($t > 100$). Thus, we shrink the NSE domain in x-direction by a factor of 3.2 and in y-(and z-)direction by a factor of 6. As a consequence, the number of elements of the EE domain is increased, which improves the coupling to compute ratio within the EE domain. A limiting factor of setup **A** was the small timestep of the NSE subdomain which had to be used for all subdomains. This small timestep is due to the high order, $\mathcal{O}(8)$, in the NSE domain according to the CFL condition for parabolic equations (2.44), In order to increase the timestep, a second-order scheme in the viscous flow with an according grid resolution is preferable.

Similar to setup **A**, Table 5.19 summarizes the individual domain sizes and numerical specifications of **B**. In the following, we will discuss individual changes in detail. In contrast to the monolithic-like setup **A**, we do not limit the tailored setup **B** to use the same spatial order in every domain.

	equation	domain size $[lu]$	number of elements	mesh levels $\mathcal{L}$	element size h	spatial order $\mathcal{O}$
mono	NSE	240×240^2	8 833 832	14−7	$1.5625e^{-2}$−2.0	2
coupled	NSE	10×4^2	4 676 160	14−12	$1.5625e^{-2}$−$6.25e^{-2}$	2
	EE	72×96^2	35 688	9−6	0.5−4.0	8
	LEE	240×240^2	25 704	5	8.0	12

Table 5.19.: Individual domain and numerical specifications of setup **B** where the monolithic and the coupled setup are tailored to physical needs. Lengths are measured in normalized length units.

Instead, we adapt the spatial order to the physical phenomena: $\mathcal{O}(2)$ in the NSE domain and a high order $\mathcal{O}(12)$ for the acoustic wave propagation in the LEE domain. For the EE domain, we maintain the medium order $\mathcal{O}(8)$, similar to setup **A**. In the monolithic approach, we also use $\mathcal{O}(2)$ to benefit from a larger timestep as well. Accordingly, we have to adapt the grid size to achieve the required numerical resolution. Therefore, we refine the mesh in the entire domain to counteract the reduction in spatial order. In practice, this means increasing all mesh levels by two. Similarly, we refine the grid resolution for the NSE domain accordingly. Figure 5.18 presents a comparison of the meshes of the NSE subdomain for setup **A** and **B**. Please note, that the locations of level jumps in setup **A** and **B** are similar but not identical. Additionally, we have adjusted the mesh of setup **B** to the narrowed NSE domain. Since we have narrowed the NSE domain, the EE domain has to be enlarged accordingly and we shift jumps of the refinement levels from the former NSE domains to the EE domain. Close to the EE–LEE coupling interface, the grid resolution is unaffected, i.e. $\mathcal{L}(6)$. For the tailored setup, we do not restrict the mesh configuration of coupling interfaces to be one-level jumps only. Hence, the NSE domain has an $\mathcal{L}(12)$-mesh at the coupling interface, while the EE subdomain has a $\mathcal{L}(9)$-mesh at the NSE interface. This results in a three-level jump as presented in Figure 5.19.

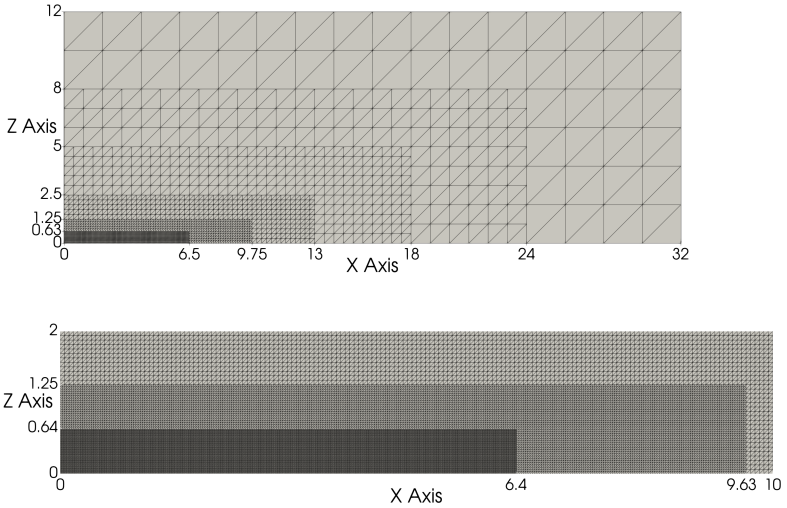


Figure 5.18.: Comparison of mesh of NSE domain for setup **A** (top) and setup **B** (bottom).

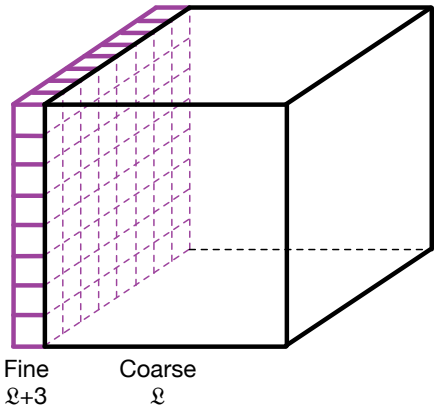


Figure 5.19.: Coupling interfaces with three-level jump.

coupling interface	corner elements	edge elements	plane elements	coupling elements (total)	coupling points per coupling face per element	total number of coupling points for **state** evaluation	total number of coupling points for **gradient** evaluation
NSE–EE	4	884	43 276	44 164	1	45 056	0
EE–NSE	0	0	468	468	256	119 808	119 808
EE–LEE	4	156	1980	2140	36	82 944	0
LEE–EE	0	0	380	380	256	97 280	0

Table 5.20.: Element configuration at the coupling interfaces of setup **B**, classified into into corner, edge, and plane elements.

Considering the number of coupling points, a coarse EE element has to serve 4 points to each of the 64 NSE elements, resulting in 256 coupling points to evaluate. In total, the number of coupling points increased by a factor of 4 since the number of EE coupling elements is increased as seen in Table 5.20. The EE–NSE interface has to provide expensive gradients as well. In this case, however, one EE elements has more coupling points located within it and, hence, can we exploit the element-wise implementation of the gradient evaluation. For the LEE domain, we are not limited to the medium order $\mathcal{O}(8)$ of the EE domain: Since we are interested in the acoustic far field, we maintain the grid resolution while increasing the spatial order to $\mathcal{O}(12)$, in order to simulate the wave propagation with higher quality than in the limited monolithic-like setup **A**. The higher order will increase the cost of the coupling point evaluation. The computational cost, however, does not increase drastically since the modal computation of linear equations scales well with a complexity of $\mathcal{O}(nElems \cdot p^3)$ in 3D (as presented in Section 4.2). With all this, setup **B** poses non-matching coupling interfaces.

Table 5.21 gives an overview of the number of total, compute and coupling elements per subdomain of setup **B**. Compared to setup **A**, especially

subdomain	total elements	compute elements	coupling elements
NSE	4 676 160	4 631 996	44 164
EE	35 688	33 080	2608
LEE	25 704	25 324	380

Table 5.21.: Number of total, compute and coupling elements at the coupling interfaces of setup **B**.

for the EE domain, the ration between compute to coupling elements is increased from 3.5 to 12. This is very beneficial for load balancing since more compute workload can be distributed and compensate the coupling workload.

	setup **A**	setup **B**
NSE	261 580 800	187 046 400
EE	25 804 800	91 361 280
LEE	65 802 240	222 082 560
Coupled	353 187 840	500 490 240
Monolithic	353 187 840	353 353 280

Table 5.22.: Degrees of freedom *DoF* for the monolithic-like setup **A** and the tailored setup **B**.

Table 5.22 provides an overview of the degrees of freedom (*DoF*) for both setups. The narrowed NSE domain with second order and finer grid results in 187 046 400 *DoF*, which is 30% less than setup **A**. Please note that this is misleading because of the highly decreased domain volume of setup **B**. Normalizing the *DoF* to volume, the NSE domain has a factor of 82 more *DoF* than in setup **A**. Considering the EE domain, the increased number of elements results in 3.5 times more *DoF*. The higher number of elements in the EE domain is beneficial for the coupled approach since it increases the computational costs for this domain. This results in a better computation to coupling ratio and improves the load balancing as described in Section 3.2.2. The LEE domain with $\mathcal{O}(12)$ enhances the *DoF* by a factor of 3.4. Hence, the coupled domain has 500 490 240 *DoF* which is by a factor of 1.5 larger than the monolithic simulation. In particular,

the acoustic LEE domain is much better resolved, as desired. In contrast, the degrees of freedom for the monolithic simulation is 353 353 280, and thereby close to the coupled setup **A**.

	NSE	EE	LEE
sum of compute weights	400877 s (6.5)	5904 s (2.7)	5748 s (3.0)
sum of evaluation weights	23 s (0.8)	916 s (1.3)	677 s (3.2)
sum of total weights	400900 s (6.5)	6820 s (2.4)	6425 s (3.1)

Table 5.23.: Weight distribution **between** the subdomains: Sum of weights for compute, coupling point evaluation, and total as sum of the subdomains for setup **B**. Numbers in brackets denote the fold change compared to setup **A**. All weights in s.

Table 5.23 summarizes the weight distribution **between** the subdomains for setup **B**. In total, the weights are 414 145 s and a factor 6.2 higher than for setup **A**. One striking fact is that the costly evaluation weights for the EE subdomain are only slightly increased in comparison to setup **A** (from 690 s to 916 s), while the number of points to evaluate is increased by a factor of 4. The maximum evaluation cost is thereby dropped from 6.0 s to 2.4 s. This was expected, since more coupling points are now located within one element that has to be evaluated. The compute load of the EE subdomain is about 2.7 times more expensive, which is due to 3.5 times more compute elements (of same order as setup **B**) . The NSE subdomain has a 6.5 time higher load while the *DoF* are only 1.3 times larger. This hints at a peculiarity of *Ateles*: Exploiting $\mathcal{O}(2)$ is more expensive than $\mathcal{O}(8)$. This is due to the fact that *Ateles* is optimized for higher order computations: The efficiency is reduced since small orders produce more but smaller messages instead of fewer but larger messages Furthermore, vectorization cannot be exploited for very small problem sizes. In contrast, the cost for polynomial evolution is reduced with this lower order. Looking at the LEE subdomain, the evaluation costs are increased due to the higher order, resulting in a factor of 3.2. The compute cost, with 3.375 times the *DoF*, is only increased about 3.1 times. Although we increase the *DoF* in LEE drastically, together with the enlarged EE subdomain EE and LEE now have around the same overall weight.

Similar to setup **A**, the timestep of each domain is fixed to the smallest timestep of all domains for the coupled simulations of setup **B**. For setup **A**, the limiting factor was the NSE timestep since a high order decreases the timestep drastically according to Equation (2.44). While tailoring the numerical resolution of setup **B**, we have tried to achieve $dt_{NSE} \overset{!}{\approx} dt_{EE}$. In the NSE domain, according to Equation (2.44), the second order and the four times smaller h results in a limiting NSE timestep dt_{NSE} of 2.58e^{-3}. The EE timestep, dt_{EE}, with an eight times smaller h but $\mathcal{O}(8)$ reduces to 2.6e^{-3}. Thus, both timesteps are almost equal and the NSE resolution is no longer limiting. Finally, the maximal timestep of the LEE due to the CFL condition is 3.4e^{-2}. Compared to setup **A** with $dt = 1.61e^{-4}$, the timestep of setup **B** is increased by factor 16. Hence, with this tailored setup a larger simulation time can be reached with an order of magnitude fewer timesteps is pivotal for **B**.

In summary, setup **B** exploits all benefits of a coupled approach: Exploiting different numerical orders in the individual subdomains ($\mathcal{O}(2)$, $\mathcal{O}(8)$, $\mathcal{O}(12)$) and arbitrary jumps of element sizes at coupling interfaces (internal mesh level jumps are restricted to one-level jumps). In particular using a low order and a fine grid for the NSE domain is beneficial due to the timestep restriction of the parabolic equations. Additionally, we increase the quality of LEE solution since we are interested in the acoustic far field. Next, we will look at the performance of this tailored setup **B** and compare it to the monolithic simulation of setup **B** as well as to the monolithic-like setup **A**. Due to compute time budget restrictions, the consideration of only one of the two approaches was possible. As Section 5.2.4 has shown the highest efficiency for the integrated approach, *APESmate* was chosen.

5.2.6.1. Scalability of setup B with *APESmate*

Here, we show the strong scaling of the tailored setup **B** with the integrated approach *APESmate*. For this setup we distribute the number of processes between the subdomain according to Table 5.24. Compared to setup **A**, the maximum number of processes for the meaningful usage of *SPartA* is not limited for setup **B** (since the maximum total weight in the EE domain is drastically decreased) and, thus, it is scaled up to 32 768 processes, which is the maximum number of cores per job available on SuperMUC.

Figure 5.20 presents the result of strong scaling for the tailored setup **B**. The simulations are performed for 100 timesteps, the consumed time

total	NSE	EE	LEE
1024	992	24	8
2048	1984	48	16
4096	3968	96	32
8192	7936	192	64
12 288	11 904	288	96
16 384	15 872	384	128
24 576	23 808	576	192
32 768	31 744	768	256

Table 5.24.: Number of processes for strong scaling distributed of the individual subdomains for tailored setup **B**.

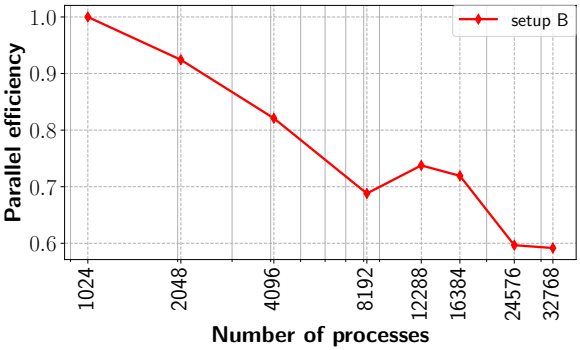


Figure 5.20.: Strong scaling for the tailored setup **B** with *APESmate* measured for 100 timesteps.

excluding the initialization phase is measured, and the parallel efficiency is computed. The computation on the smallest number of ranks (1024) is used as reference. The efficiency of the coupled simulation decreases down to 60%. For 12 288 ranks the efficiency of setup **B** increases slightly, and for 24 576 and 32 768 ranks the efficiency plateaus.

One conspicuousness is that the trend of the efficiency changes at 12 288 and 24 576 processes. The parallel simulation of the subdomains itself and

the coupling is based on exchanging data at the interface via point-to-point communication. On SuperMUC, one island consists of 512 nodes that comprise 8192 cores. This implies the usage of two islands for runs on more than 8192 cores and the usage of three islands for more than 16 384 cores. Using more than one island results in higher communication times due to limited bandwidth between islands. This influence of the communication is visible in Figure 5.20 at 12 288 and 24 576 processes. Regarding the distribution of the subdomain, it is very likely that the EE and LEE sudomain are located on one island while the NSE subdomain is spread over multiple islands. It might be counterintuitive that the scaling improves slightly when communicating over more than one island. But, this change in the communication introduces another layer of imbalances, which can compensate the imbalances of the subdomain to some extent.

We generally see that the coupling approach with three coupled subdomains scales. However, the scalability of coupled simulation has its limits: For high numbers of ranks, the problem size per subdomain per rank decreases and overhead can dominate the computation time. Additionally, expensive coupling elements cannot be further distributed across multiple ranks (one element per rank) and, thus, the scaling only shows benefits until most of the coupling elements are distributed. Looking at the difference between setup **A** and **B**, it appears that the issues of scaling a coupled 3-field simulation are concealed by the general configuration of setup **A**. For setup **A**, the element ratio of computation to coupling is worse than for **B**. Additionally, in the EE subdomain the coupling load at the NSE interface is increased. In the tailored setup **B**, where the benefits of coupling are exploited, the limits of scalability become visible. Here the element ratio between interface and computation is tuned. In particular for setup **B**, the ratios between evaluation cost and computation cost for EE subdomain (cheaper gradient evaluation) and for NSE subdomain (higher computation, lower evaluation) are tuned (Table 5.23).

We have started out with the assumption that we can reuse the weights measured in Section 5.2.4.1 for 4096 ranks, since the actual load does not depend on the number of ranks. This investigation, however, shows that this might not be entirely valid. In Section 4.2.1, we have described that the weight of computation is measured per process and not element (as it is done for coupling). The computation weight per element is then computed as $W_{comp} = \frac{T_{comp}}{N_p}$, where T_{comp} is the timing for the compute kernel and N_p the number of elements per process. Hence, the number of

processes influences the average computation weights. For future work, it is recommended to compute the weights once per individual distribution or to change the implementation so that the computation weights are not averaged but directly computed per element.

5.2.6.2. Load Balancing of setup B with *APESmate*

Analogously to the previous performance investigation for setup **A**, we will now consider setup **B**. Table 5.25 shows the results for the tailored setup when using the presented *workflow of well-balanced coupled simulation*. According to the total weights for the individual subdomains (Table 5.23), the number of processes between the subdomains are distrusted by 96%, 1.6% and 1.5%. To make it easier to compare results, timings of setup **A** are listed as well. Compared to setup **A**, we observe that the timing of the monolithic approach is by a factor of 7.5 higher, which is also observed for the NSE subdomain in the coupled simulation and is due to the fact that *Ateles* is not optimized for low orders. The coupled timings, in contrast, only increase by a factor of 4. Here the coupled simulation benefits from the tailored configuration and results in a speedup of 2.8 compared to monolithic.

		setup **B**		setup **A**	
	sub-domain	processes	execution time [s]	processes	execution time [s]
monolithic		4096	902.8	4096	120.2
		16 384	211.8		
APESmate	NSE	3965		3791	
	EE	67	318.4	176	80.3
	LEE	64		129	
	NSE	15 876			
	EE	256	92.9		
	LEE	252			

Table 5.25.: Computation times for **100 timesteps** (without initialization phase) of monolithic and coupled simulations of setup **B**. As comparison timings of setup **A** of Table 5.13 are listed.

This comparison is based on a fixed number of timesteps, while the major benefit of setup **B** is the **16 times larger** timestep. Therefore, Table 5.26 also includes execution times for 100 timesteps (like Table 5.25) as well as execution times until 1 *tu* of simulation time is reached. To reach the latter, the monolithic simulation of **B** is twice as fast as **A**, and the coupled simulation is four times faster. Furthermore, the acoustic far field (LEE subdomain) is expected to show a better quality of the solution.

	setup **B**			setup **A**		
		execution time for			execution time for	
	dt	$100 \cdot dt$	1 *tu*	dt	$100 \cdot dt$	1 *tu*
monolithic	$2.58e^{-3}$	902.8	3499.2	$1.61e^{-4}$	120.2	7465
APESmate	$2.58e^{-3}$	318.4	1234.1	$1.61e^{-4}$	80	4968.9

Table 5.26.: Execution times for **100 timesteps** (=dt) (similar to Table 5.25) and for reaching **1 *tu*** of simulation time for setup **A** and **B** on 4096 MPI-processes. All times without initialization phase and in s. (Setup **B** is limited to 12 288 processes, and thus, not listed).

This rather large test case is constructed for running on as many processes as possible: The number of processes can also influence the load balancing slightly, since the workload can be spread over more processes. Therefore, we also show the timings for running on up to two islands (16 384 processes) on SuperMUC, LRZ, Munich. Comparing the timings for 4096 and 16 384 processes for setup **B**, we observe a speedup of 3.4 for the coupled simulation. For the monolithic simulation we observe a superlinear behavior with a speedup of around 4.3. One explanation for the superlinearity of this strong scaling is that the monolithic simulation uses a low order of $\mathcal{O}(2)$ which can yield caching effects in the computation. Additionally, for the coupled simulation even when balancing between and within subdomains, there is more risk for for waiting times.

5.2.7. Investigation of imperfect choice of coupling interfaces

In this section, we enforce an imperfect coupling interfaces between viscous and inviscid flow in order to investigate the influence on the acoustic far field. To that end, we place the NSE–EE interface in the 3D jet too close to the jet assuming a final simulation time t. The coupling setup was design to fit optimal for $t = 120$ (Section 5.2.2) where the turbulent jet is still inside the NSE subdomain. When simulating this setup until $t = 250$, however, the turbulent jet with viscous effects travels over the coupling interface into the inviscid flow domain. This was done on purpose to investigate the obvious influence on the turbulent part and its impact on the development of acoustic waves. The results up to a simulation time of 120 are described in Section 5.2.1, where the tailored setup **B** is used and the partitioning from Section 5.2.6.2 is applied. The simulation is performed on 1024 nodes with 16 384 cores of SuperMUC. Since one island on SuperMUC consists of 512 cores, we have to use a minimum of two islands. Using two island exclusively can lead to long job waiting times in case the system is highly frequented. Hence, due to a high occupancy on the system we allow to distribute the MPI-processes on three islands to minimize job waiting times. To perform this simulation until a simulation time of 250 tu, several jobs have been submitted. Depending on the used CPUs and the network load, it took between 33 and 55 minutes to simulate a second in simulation time and write one restart file. In total, the full simulation has consumed more than 2 million CPU hours.

Navier-Stokes and Euler domain We consider the NSE and EE domain mostly together, since the free stream is expected to travel over the coupling interface. White lines always indicate coupling interface between different equations. Figure 5.21 shows the acoustic waves with a Schlieren visualization of an xz-plane at $y = 0$. Since we have to handle a multi-scale problem, the jet is shown on the large data range of the flow variable by the iso-contour of Ma=0.39. The Schlieren (gradient of density) indicate the shape of the free-stream and show turbulence. On the left-hand side, close to the inlet where the free-stream jet is stable and has pressure close to 1, we can see sound waves radiating from it. On the right-hand side at the coupling interface, the flow (in Schlieren representation) does not freely travel out of the NSE domain.

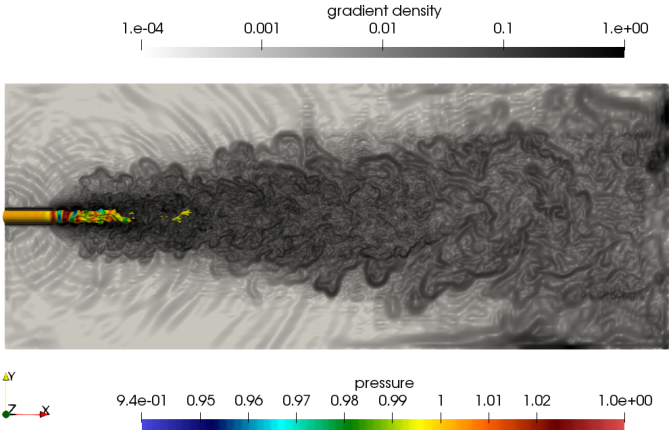
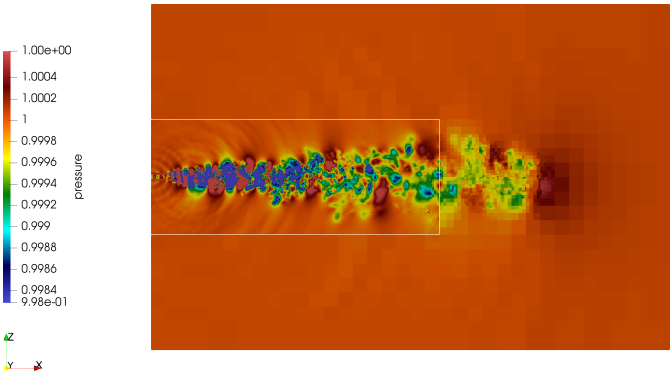


Figure 5.21.: Schlieren visualization (gradient of density in xz-plane) of NSE domain (gray scale) and iso-contour of Ma=0.39 colored by pressure (rainbow scale) at $t = 250$.

To investigate the NSE domain and its coupling further, we look at different properties of the flow: Figure 5.22 presents the pressure of the flow in the NSE domain with the interface to the EE domain. Scaling the pressure range to the values of NSE domain (Figure 5.22a), the pressure field looks nicely resolved in the NSE domain and values close to background ($p = 1$) have traveled over the coupling interface. Scaling the pressure range to the scales of the EE domain (Figure 5.22b), the structure is still preserved but already reaches into the EE domain. In Figure 5.22b at the break-up of the free-stream, the acoustic wave generation is nicely visible.
The next property of the free-stream that we consider is the velocity presented in Figure 5.23. At $t = 250$, small values of the velocity (below 0.1) have propagated into the EE domain. Here, we do not observe any strong artifacts but some oscillations ($5e^{-2}$ to $5e^{-8}$) at the bottom right coupling corner of the NSE domain. To investigate if viscous effects play a role, we look at the vorticity in Figure 5.24. The aforementioned oscillations in the velocity field are better visible in the vorticity plot. Vorticity of the flow has traveled over the coupling interface and into the EE domain where a vorticity of ≈ 1 can be observed. The main influence of the vorticity at $t = 250$, similar to $t = 120$, is at the beginning of the jet in the shear layer and the turbulent region close to the nozzle.

(a) Scaled to NSE



(b) Scaled to EE

Figure 5.22.: Pressure field of NSE and EE subdomain at $t = 250$.

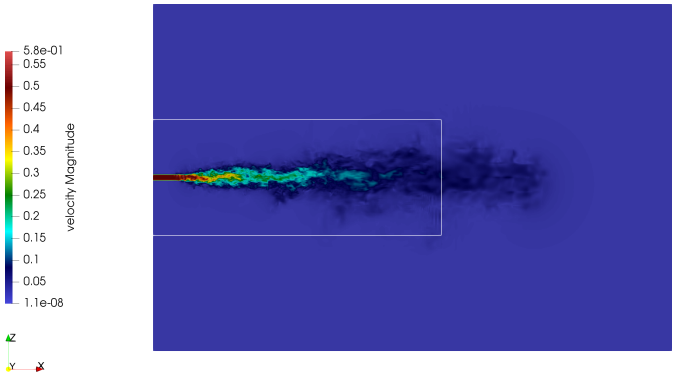


Figure 5.23.: Velocity of NSE and EE subdomain at $t = 250$.

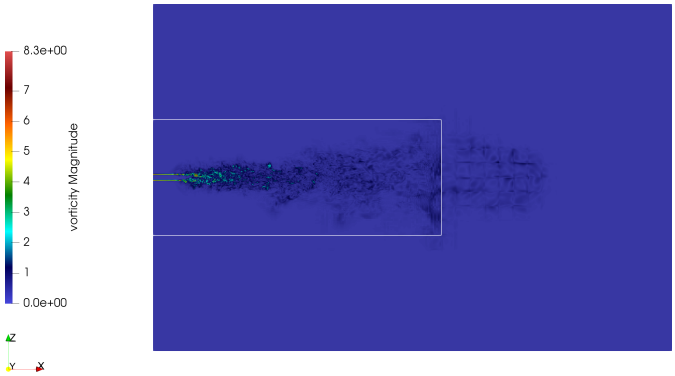
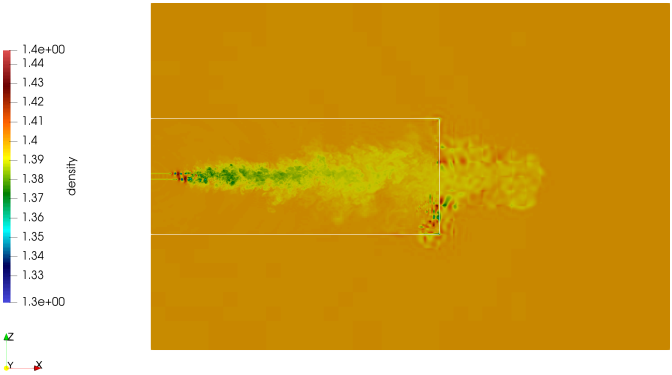
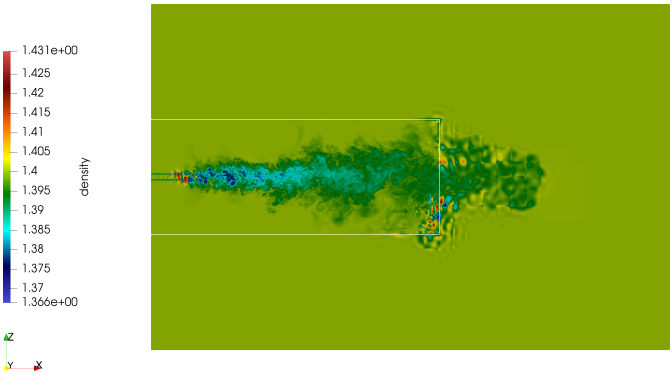


Figure 5.24.: Vorticity of NSE and EE subdomain at $t = 250$.



(a) Scaled to NSE



(b) Scaled to EE

Figure 5.25.: Density field of NSE and EE subdomain at $t = 250$.

Figure 5.25 illustrates the density field at the simulation time of 250 tu of the free-stream jet, scaled to both, the NSE (Figure 5.25a) and the EE domain (Figure 5.25b), respectively. Similar to previous properties, the free-stream has traveled into the EE domain. Both figures reveal issues in the density at the NSE–EE interface which were not there for $t = 120$.

As we are highly interested in acoustic wave propagation, we consider the Schlieren visualization in the NSE and EE subdomains at $t = 250$ in Figure 5.26. Even though we can identify some artifacts at the coupling interface, the acoustic waves are traveling over the coupling interface in x-direction and waves properly radiate into the EE domain. However, the acoustic waves are mainly generated within the first part of the free-stream jet where the jet evolves without the influence of internal level jumps and the coupling interface. In the higher-order EE domain, the "coarse" Cartesian grid of the simulation is visible where the polynomials show artifacts at element boundaries. In z-direction, small oscillations parallel to the coupling interface occur. As mentioned in Section 4.1, the visual-

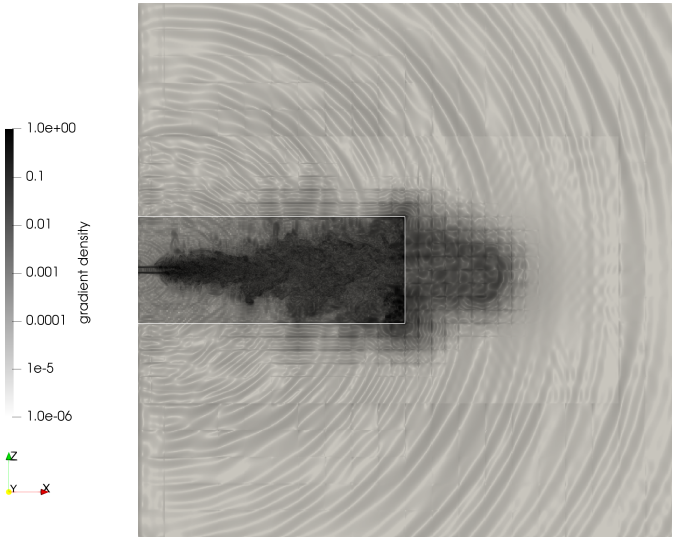


Figure 5.26.: Schlieren visualization of NSE and EE subdomain where EE is clipped to 20×10 lu at $t = 250$.

ization of polynomials is not implemented in common visualization tools. Therefore, we have to subsample the polynomial data (in this case of $\mathcal{O}8$) with additional points. This is not ideal for the polynomial representation because the integration points of the polynomials are not equidistant while the subsampling is.

Above, we have identified the following challenges: i) Turbulent flow reaches the coupling interface and, hence, switching from a viscous to an inviscid flow is not physically allowed. Here, the NSE domain should be elongated in x-direction, while this is not required for the z- and y-direction. ii) Artifacts of internal level jumps are visible in the vorticity field (and other properties) when scaling to small value ranges. This can indicate that the spatial resolution is not high enough. In addition to the internal level jumps, the numerical resolution (setup **B**) has a large jump in mesh resolution ($\mathcal{L}(12)$-$\mathcal{L}(9)$) and polynomial degree ($\mathcal{O}(2)$-$\mathcal{O}(8)$). In general, non-matching coupling interfaces with such large jumps in numerical order and grid resolution were investigated with the academic testcase. However, turbulent flow is a challenging testcase and the large difference in order and grid at the coupling interface can have a significant influence. iii) A final aspect is that the visualization of high order polynomials is not trivial and small oscillations of polynomials within the Schlieren visualization of the higher-order EE domain are visible (Figure 5.26). There, the Schlieren visualization with logarithmic scale presents the small scales of the flow. The "coarse" Cartesian grid of the simulation is visible and toward element interfaces the polynomials are better resolved. It can be stated that, even though the turbulence of the jet in x-direction is not perfectly resolved and travels over the coupling interface, acoustic waves are mainly generated close to the nozzle and that they nicely propagate. For future simulations of this testcase, the spatial resolution of the EE domain should be increased. After we have investigated the NSE domain jointly with the EE domain, we analyze the EE domain separately and look at the coupling interface to the LEE domain.

Figure 5.27 visualizes the entire EE domain at the simulation time 250 tu without the NSE domain, where in a region close to the NSE domain (20×10 lu from Figure 5.26) the density is shown in the rainbow colors and a Schlieren visualization is used to show the acoustic waves otherwise. Based on this figure, we can state that for the time evolution until 250 tu the EE domain could be reduced. As explained earlier, besides the physics, also the computational load should be taken into account for choosing the locations of coupling interfaces and switching as soon as

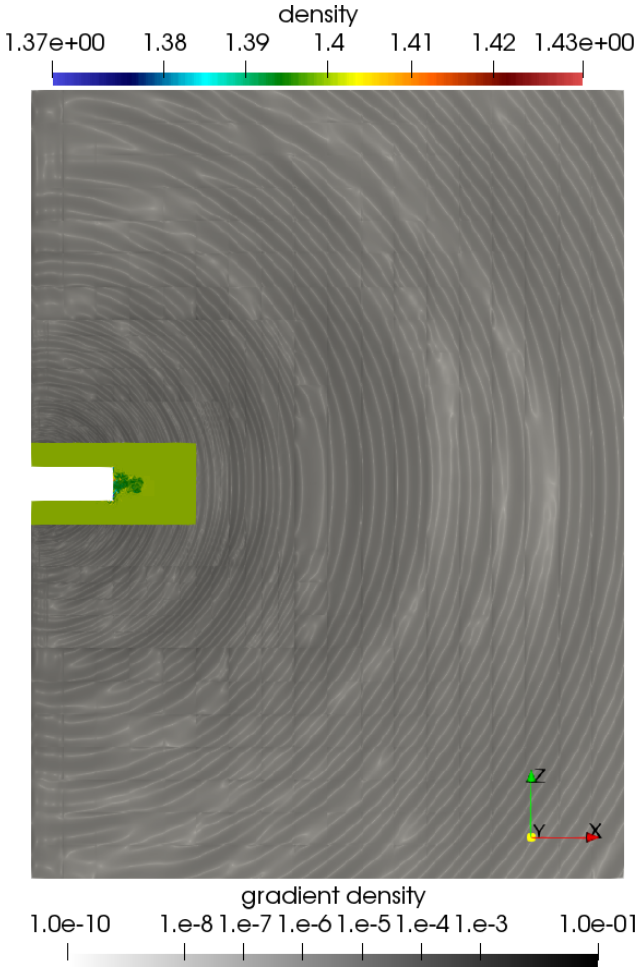


Figure 5.27.: The full EE domain at $t = 250$: Close to the NSE coupling interface the density is shown with the rainbow colors and further away the Schlieren flow is presented. The region for density visualization is clipped to 20×10 *lu*.

possible to LEE is preferable. Figure 5.27 emphasizes that, even though the turbulent jet has already traveled into the inviscid EE subdomain, the large acoustic scales are resolved and transported into the far field. Finally, we will investigate the LEE domain at $t = 250$.

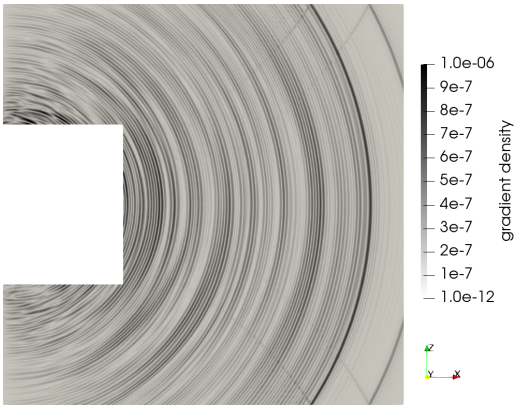


Figure 5.28.: Schlieren visualization (xz-plane) of the entire LEE domain at $t = 250$.

Linearized Euler domain Figure 5.28 shows the Schlieren visualization of the entire LEE domain at the simulation time 250 *tu*. In this visualization the color scale is not logarithmic, since the data range is smaller. In the Linearized Euler equations only the perturbations of the flow are considered as variables (see Section 2.1.3).

Acoustic far field The overall goal is the simulation of acoustic waves and their propagation into the acoustic far field. Figure 5.29 presents the acoustic far field of the entire coupled domain (NSE–EE–LEE), where the coupling interfaces are marked with white lines. It is a Schlieren visualization of xz-plane at y=0 (the middle of the jet), in which the gradient of the density is visualized. At this point in time the acoustic waves have reached the outer outlet boundary of the domain.

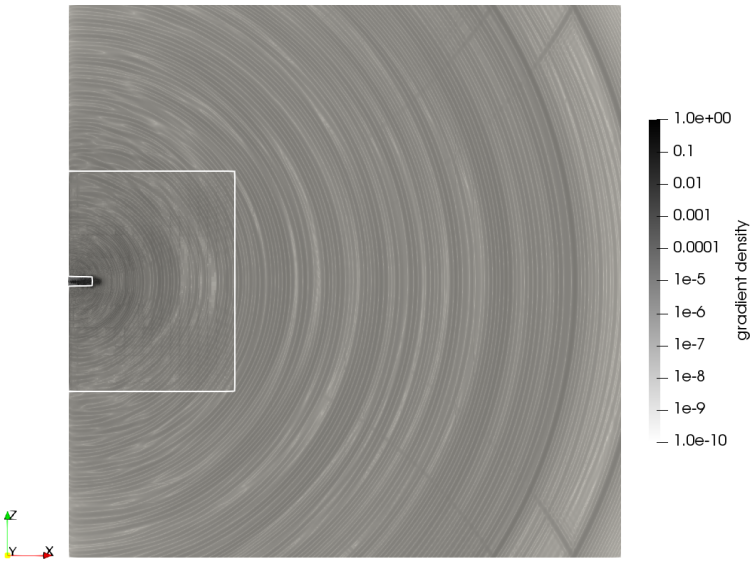


Figure 5.29.: Schlieren visualization (gradients of density) of xz-plane at $t = 250$ where the acoustic waves reach the outer boundary condition. The coupling interfaces are marked with white lines.

Summary To sum up, we presented the 3-field coupling for a 3D free-stream jet at $Re = 12500$ and $Ma = 0.5$, computed efficiently with the integrated coupling approach *APESmate*. While the coupling interfaces had beend constructed for evolving the simulation up to a simulation time of 120 tu, here we have investigated imperfect sound generation and the influence on the acoustic far field. At the final time (250 tu), the turbulent jet has fully propagated in the NSE domain and reaches over the coupling interface into the EE subdomin. That is, an unsuitable coupling interface between viscous and inviscid flow has been chosen. The acoustic far field has reached the outer boundary of the computational domain at this point in time. Even though the sound generation of turbulent jet is not perfectly simulated considering the coupling interface at $t = 250$, the acoustic waves are smoothly transported into the far field. One aspect is that the acoustic generation happens mainly at the beginning of the jet which is simulated fully in NSE domain. With the tailored setup **B**, we have aimed for a good resolution of the LEE domain.

Besides the physical aspects, computational aspects must be considered as well and the workload of a coupled simulation must be balanced. Therefore, the grid resolution plays a role as well as it increases the workload but leads to better results. As presented in the investigation of the NSE domain, the physics are not fully resolved and some artifacts like internal level jumps are visible. Also, in the Euler domain, the resolution close to the NSE domain should be increased and it would be beneficial to tailor the spatial order of the DG scheme to the refinement levels. For the given scenario and point in time, the computational domain of the EE domain could be reduced and the coupling interface to the LEE domain moved towards the jet: Switching as soon as possible from non-linear to linear equations with *Ateles* is preferable form the computational workload aspect.

6. Investigation of time-consistent coupling

In the previous chapter we have presented and analyzed the integrated as well as the multi-solver coupling approach. So far, we have exclusively investigated the influence of data mapping in space for those approaches. As introduced in Section 3.1.3, however, data mapping in time is important as well: In case both subdomains use a different timestep, the subdomain with the smaller timestep requires data at points in time where the other subdomain does not provide it. Without a method to circumvent this, the subdomain with the smaller timestep may use inconsistent data. In this work, we avoid this by using the same timestep in all subdomains which guarantees that data is available at each timestep. Unfortunately, the same challenges of time-consistent coupling occur when using multi-stage integration in time.

As discussed in Section 3.1.3, there is an inconsistent coupling in time when using a multi-stage time stepping approach without appropriately treating boundary conditions within a timestep. All simulations presented in this work are done with the explicit second-order Runge-Kutta (RK) method which requires two sub-stages. To construct the two sub-stages, a midpoint at $t + \frac{\Delta t}{2}$ is used. The equation for RK to advance the DG solution from t to the next timestep $t + \Delta t$ for second order is (combined Equation (2.40) and Equation (2.42a)):

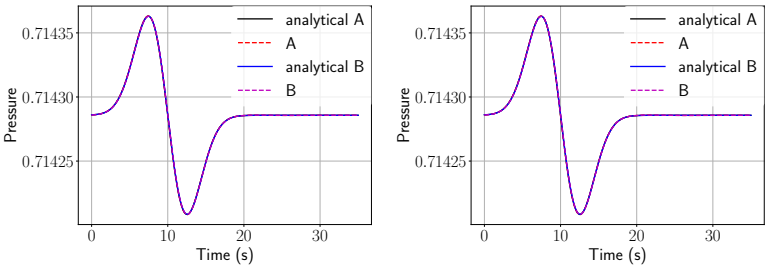
$$\boldsymbol{U}_h(t + \Delta t) = \boldsymbol{U}_h(t) + \tag{6.1a}$$

$$\Delta t \Bigg(\underbrace{\boldsymbol{M}^{-1} \cdot \boldsymbol{rhs}(\boldsymbol{U}_h(t), t)}_{\text{1. sub-stage: data at last complete timestep}} \tag{6.1b}$$

$$+ \underbrace{\boldsymbol{M}^{-1} \cdot \boldsymbol{rhs}(\boldsymbol{U}_h^A, t + \frac{\Delta t}{2})}_{\text{2. sub-stage: data at midpoint}} \Bigg). \tag{6.1c}$$

When only exchanging coupling data at a complete timestep, consistent data at the coupling interface is only available for the first sub-stage while the second sub-stage has to use "outdated" data of the latest complete RK

timestep. In case coupling data is exchanged at the midpoint as well, the second sub-stage can use consistent data at the coupling interface. Within a subdomain, all elements use second-order RK and the synchronization at midpoints is no problem so that flux data from neighbors can be used. Hence, the "effective" time discretization decreases to first order for the boundary treatment at the coupling interface. In the following we investigate the influence of this inconsistent boundary treatment by using the same academic testcase as presented in Section 5.1 with Linearized Euler equations (LEE) in both subdomains.

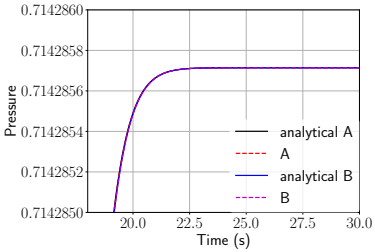


(a) LEE - LEE with *APESmate*, matching interface.

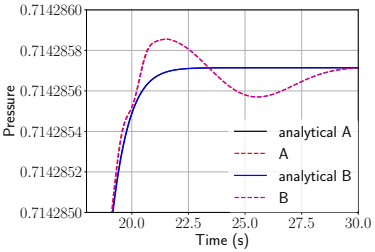
(b) LEE - LEE with *preCICE*, matching interface.

Figure 6.1.: Comparison of numerical and analytical result in both subdomains when coupling the **same** equations with **matching** interfaces for *APESmate* and *preCICE* at measurement positions $A = (10 - 0.01, 0, 0)$ and $B = (10 + 0.01, 0, 0)$, close to the coupling interface.

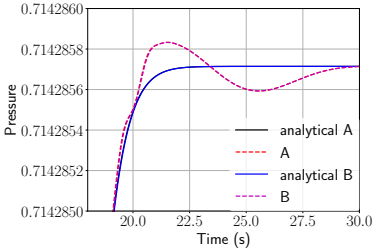
Figure 6.1 presents the results for coupling the same equations with matching interfaces at the measurement positions $A = (10 - 0.01, 0, 0)$ and $B = (10 + 0.01, 0, 0)$, close to the coupling interface. With this minimal setup, we exclude any influence from changing physics as well as data mapping in space. Even though the plots in Figure 6.1 show good general agreement, taking a closer look reveals issues at the coupling interface: Figure 6.2 shows close-ups of different simulations, illustrating oscillations that are not visible in a monolithic simulation with a single LEE domain of the same resolution (Figure 6.2a). The first observation comparing Figures 6.2c - 6.2b is that the oscillations have the same order of magnitude regardless of the type of interface or approach. Figure 6.2d with non-matching coupling interfaces shows slightly different oscillations at



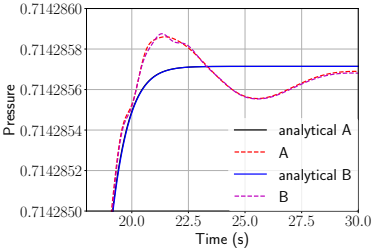
(a) Monolilthic simulation of LEE domain.



(b) LEE - LEE with *preCICE*, matching interface.



(c) LEE - LEE with *APESmate*, matching interface.

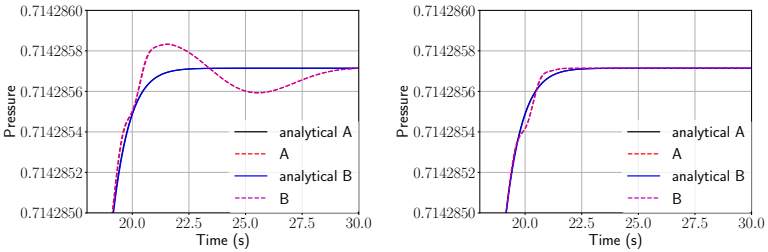


(d) LEE - LEE with *APESmate*, non-matching interface.

Figure 6.2.: Close-ups for $15s < t < 30$s and pressure range of $0.7142850 < p < 0.7142860$ of the comparison of monolithic simulation and different coupled simulations with numerical and analytical results in both subdomains.

the overshoot for measurement position B (after the coupling interface), which indicates that the data mapping in space has some impact.

To prove that the oscillations are due to the inconsistency at the boundary treatment, we check the influence of Δt: The inconsistency is expected to become smaller with decreasing Δt. Thus, we compare a simulation with $dt_{CFL} = 1.25e^{-2}$ and a significantly smaller timestep $dt_{small} = 1e^{-5}$ in Figure 6.3. Figure 6.3b shows that the magnitude of the oscillations is strongly decreased when using dt_{small}, which hints at the impact of the time-inconsistent coupling. While the main focus of this thesis is consistent coupling in space, we show a naive approach for time-consistent coupling



(a) $dt_{CFL} = 1.25e^{-2}$ (b) $dt_{small} = 1e^{-5}$

Figure 6.3.: Comparison of LEE - LEE coupling with *APESmate* and matching interface with a) the *CFL* timestep dt_{CFL} and b) a significantly smaller timestep dt_{small}.

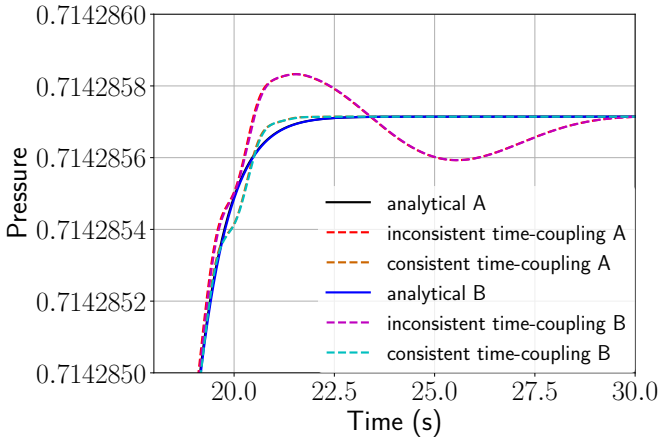


Figure 6.4.: Comparison of inconsistent to consistent time-coupling for LEE - LEE coupling with *APESmate* and matching interface.

with multi-stage timestepping to perform a preliminary investigation. To that end, we have enabled the integrated approach *APESmate* to exchange coupling data at the midpoint of the RK timestep. If both subdomains use the same time discretization, we can use it for a proof of concept to show the need tor time-consistent coupling. Figure 6.4 compares the results of inconsistent and consistent coupling in time, i.e. coupling at each complete RK timestep and coupling at each midpoint of the RK timestep. We can see that with coupling at each midpoint of the RK timestep, the oscillations decrease in a similar way to using $dt_{small} = 1e^{-5}$ (Figure 6.3b); the oscillations for $t > 22$s even vanish completely. The overshoot of the pressure at $20\text{s} < t < 23\text{s}$ decreases by a magnitude of 2e^{-7}. But, compared to the monolithic simulations the oscillations do not vanish entirely. Hence, for this academic testcase, there is no difference in time-consistent coupling and using a significantly smaller timestep $dt_{small} = 1e^{-5}$. So, in case time-consistent coupling is not possible, using a significant smaller timestep could circumvent the problem of inconsistent boundary treatment. Please note that we are looking at a strong close-up to the pressure range of 1e^{-6} which is only two orders of magnitude smaller than the accuracy of the discretization. As shown in Table 5.3 of the previous chapter, the monolithic solution for the LEE with the chosen discretization has a relative error of 1.218e^{-8} at measurement position B.

The presented investigation illustrates the influence of time-inconsistent coupling and the impact of inconsistent boundary treatment. For the usage of second-order RK time integration, we have presented preliminary results where data are exchanged at the midpoints of the second-order RK timestep: This shows increasing accuracy of the solution. The impact for different testcases is out of scope of this thesis. Typically, consistent boundary treatment is not as important for the overall simulation since boundary conditions happen "far away" from the desired phenomena. In a coupled simulation, this does not hold: In particular for the subdomain where the desired information are emitted from the coupling boundary, consistent treatment is important. In a flow-acoustic simulation, the acoustic waves travel from the flow domain into the acoustic domain and, thus, the impact of inconsistent boundary treatment is higher for the acoustic domain than for the flow domain. In case of bi-directional coupling, where feedback is desired, this is not entirely true trough. The impact of the inconsistent boundary treatment for the entire simulation and the testcase dependency should be further analyzed. Also the question whether to use a second or fourth order time integration and accept lower-order boundary treatment must be considered individually. For a fourth-order

RK, where the sub-stages are not only constructed by the midpoint rule, time-consistent coupling is more involving.

Consequently, time-consistent coupling and its realization should be addressed in future work: This could be achieved by coupling for each sub-stage with an thought-out strategy or by exploiting time integration schemes that construct high-order methods differently than multi-stage approaches. For instance, by using an ADER scheme [36, 59] that constructs time derivatives from spatial derivatives or a local time Galerkin scheme [89]. Another approach would be to use an extrapolation / reconstruction in time within the domain so that each domain solver can provide the required data itself. This challenge of time-consistent coupling is even bigger when allowing each subdomain to use its own timestep based on the individual physics to solve. Allowing adaptive timestepping can improve the performance of a coupled simulation, since not all subdomains are forced to use the same (small) timestep of one subdomain. Hence, the benefits of individual numerical schemes can be exploited even further.

7. Summary and outlook

In this final chapter, Section 7.1 provides a short summary of the presented work. This thesis has established two different approaches of partitioned coupling that we have realized within the end-to-end parallel simulation toolchain *APES*: The multi-solver approach with the coupling library *preCICE* and the integrated approach *APESmate*. In this context, the challenges of a partitioned coupling with respect to these approaches are discussed. The work shows that partitioned coupling enables the computation of large, multi-scale problems which are computationally too demanding with a monolithic approach. This is shown with a simulation of a 3D free-stream jet with acoustic noise generation on the massive parallel computing system SuperMuc Phase 1 IBM system at LRZ, Munich. With this promising result and the presented coupling approaches, multi-physics and multi-scale problems, which are unfeasible with a monolithic approach, can be examined. Ideas and future tasks are outlined in Section 7.2.

7.1. Summary

Aero-acoustic simulations have the potential to provide deeper insights into applications from various fields like the sound design of aircraft or wind turbines. It is the nature of such multi-scale simulations to exhibit a wide range of scales in space and time. Hence, they pose one of the most demanding computational tasks in engineering. Indeed, only with the increasing computational power of supercomputers and appropriate strategies larger simulations investigating interactions between multiple physics became feasible. Therefore, this work is concerned with partitioned coupling as an approach to enable simulations of fluid-acoustic interactions on massive parallel computing systems. To that end, we have presented the coupling of a high-order Discontinuous Galerkin with a low-order Discontinuous Galerkin method. We have used the scalable DG solver *Ateles* (4.2) which is part of the end-to-end parallel simulation toolchain *APES* (4.1). To represent the aero-acoustic problem, a 3D free-stream jet was picked.

For this work, two different coupling approaches were implemented: The

multi-solver approach with the coupling library *preCICE* (4.3) and the integrated approach *APESmate* within the simulation framework *APES* (4.4). Both coupling approaches enable direct aero-acoustic simulations with bidirectional coupling, but differ in their scaling on massive parallel systems and their quality of data mapping in space. The comparison highlights the trade-off between a coupling library with high flexibility and an integrated approach with several constraints on the solver but increased performance and appropriate data mapping in space. The multi-solver approach with *preCICE* enables the direct simulations of an aero-acoustic problem which is unfeasible with a monolithic simulation, but the speed up of the simulation is not as high as with *APESmate*. This is due to the fact that *APESmate* is a new implementation that is limited to solvers in *APES*, which enables circumventing external data interpolation. This yields a high scalability on massive parallel computing systems.

With respect to the coupling tasks: *steering*; *communication*; and *data mapping in space and time*; both approaches were compared. When *steering* individual solvers (3.1.1), *preCICE* has to deal with individual executables of solvers which need to be linked to it. In contrast, *APESmate* compiles to a single executable. Hence, porting the software, establishing the correct binding of MPI-ranks, and setting up the job script on a supercomputer is more challenging with *preCICE* compared to running a single executable for *APESmate*. To realize the interaction between different domains that are coupled, *communication* of coupling data is required (3.1.2). Using an efficient communication is essential: Both approaches establish *point-to-point* communication between the involved processes in their initialization phase. For *preCICE*, which is working on input/output data, this is more demanding which is consistent with the presented performance.
The last coupling tasks are the *data mapping in time* (3.1.3) and *data mapping in space* (3.1.3). We specifically did not focus on data mapping in time in this thesis, since we guarantee same timestepping in all coupled domains and no sub-cycling. For data mapping in space, the main difference is that *APESmate* utilizes mapping routines of the numerical solver while *preCICE* works exclusively on geometric data, which requires an external interpolation method. As described in Section 3.3.2, when using a DG method the polynomial evaluation can be used to provide data at arbitrary points in space. In other words, *preCICE* requires interpolation while *APESmate* uses an evaluation within the solver *Ateles*. We presented the available interpolation methods in *preCICE* in Section 4.3.5 and compared their solutions in Section 4.3.5.3. As reported in these sections, Radial Basis functions are promising but challenging for non-equidistant

coupling points at the coupling interface. Furthermore, we derived that it is advisable to use NN for matching coupling interfaces with *preCICE* and nearest projection mapping for non-matching interfaces. The quality of the *data mapping in space*, i.e. interpolation (*preCICE*) and evaluation (*APESmate*), was evaluated with an academic testcase and shows good agreement with the analytical solution for all combinations of interfaces and equations. As presented in Section 6, *data mapping in time* is an non-negligible aspect for multi-stage time integration methods, like higher-order Runge-Kutta, and will be further discussed in the outlook. Here, a 3-field coupling using three different physical subdomains, namely viscous flow with Navier-Stokes equations, inviscid flow with Euler equations and acoustic propagation with Linearized Euler equations is analyzed as well.

Another crucial aspect of partitioned coupling, and large-scale simulations in general, is load balancing (3.2): We discussed the load imbalances between subdomains (3.2.1) and within a subdomain (3.2.2) which are introduced by additional coupling work. The load balancing strategy in *Ateles* was presented which is beneficial for the multi-solver as well as the integrated approach. Furthermore, the load balancing for *APESmate* and *preCICE* was investigated for the large-scale scenario of the 3D jet and a workflow for well-balanced coupled simulation was identified in Section 5.2.4.1.

This large-scale scenario of the 3D jet was also used to investigate the spatial choice of coupling interfaces: First, the coupling interfaces were chosen to suit the physical phenomena and, subsequently, an imperfect choice of coupling interfaces was demonstrated. The latter one shows, that in our testcase an imperfect location of coupling interface between viscous and inviscid flow subdomain is not a determining factor for acoustic wave propagation into the far field: The sound is mainly generated at the break-up of the free-stream and large scales are transported over the interface without disturbance.

In short, partitioned coupling is beneficial for multi-scale as well as multi-physics problems, in particular, the presented large-scale scenario. Here, the integrated coupling approach *APESmate* requires $\approx 66\%$ the time of a monolithic simulation. In contrast, the multi-solver approach using the external library *preCICE* with NP mapping still requires $\approx 83\%$. Thus, the coupling library *preCICE* presents an accessible approach with a minimal-invasive API to the numerical solver to enable partitioned *black-box* coupling. In contrast, *APESmate* was developed from scratch and is

tailored for use with the solvers available in *APES*. An important factor of the performance increase is the optimization of the implementation in *APESmate*, which is described in Appendix A.

7.2. Outlook

The outlook structures ideas for future tasks according to topics concerning partitioned coupling in general, topics which focus on the integrated approach *APESmate*, and finally ideas for the coupling library *preCICE*.

7.2.1. Partitioned coupling

There are multiple aspects to look at in future: Time-consistent coupling, improvements concerning the general performance, and investigations of real-world scenarios which are made possible with the presented approaches.

One crucial aspect presented in this work is the time-consistent coupling when applying a multi-stage approach. The challenge is to provide appropriate coupling data at the coupling interface for intermediate timesteps. Coupling at each intermediate timestep shows increasing accuracy of the solution. One approach would be the usage of an extrapolation or reconstruction in time within the domain so that each domain solver can provide the required data itself. Exploiting a single-step scheme in time can circumvent this issue. For instance the ADER scheme [36, 59] or a local time Galerkin scheme [89]. Implicit coupling, which is a feature of the coupling library *preCICE*, could overcome this issue as well. Using adaptive timestepping to enable one coupling domain to have its own timestep based on the individual physics to solve, would improve the performance even more.

Considering the performance of coupled simulations, coupling introduces load imbalances within one domain, and every step that reduces these imbalances speeds up the simulation. This also improves the ratio of computation cost incurred by every element compared to coupling cost incurred by a small portion of elements, which can be exploited by a load balancing algorithm. While a static load balancing was presented in this thesis, dynamic load balancing would be interesting to cover cases where the workload changes at simulation time. For example, considering a moving structure or moving coupling interface would require dynamic load balancing.

The most seminal aspect is the possibility that the coupling approach opens up: Aero-acoustic noise generation of super- and transonic flows can be investigated in the future, e.g. noise of a planar jet [90] or noise of a coaxial jet [91]. By extending the DG solver *Ateles* with an immersed boundary approach to represent geometries within the Cartesian octree-mesh [92–94] even more challenges could be tackled, e.g. flow around a profile [95] or the flow around a moving geometry which introduces high noise.

An entirely different approach for fluid-acoustic simulations would be a scheme where the equation system changes locally. Considering the DG scheme, where elements are only loosely coupled to each other via the numerical flux, an element-wise change of the equation system is possible. With this, no additional coupling tools are required and, hence, additional communication can be avoided [96].

7.2.2. Integrated approach: *APESmate*

For the integrated coupling approach *APESmate*, several optimization ideas were sketched in this work. One promising aspect to consider for speed up is the gradient evaluation: In this work a point-wise evaluation is done, while it would be faster to do a face-wise evaluation. For a face-wise evaluation, a projection from the volume to the coupling face could yield a two-dimensional polynomial. Subsequently, for all points of the coupling face, a two-dimensional evaluation instead of a three-dimensional could be done. This optimization step would reduce the evaluation cost from $\mathcal{O}(n \cdot p^3)$ to $\mathcal{O}(p^3 + n \cdot p^2)$ per coupling face, where n is the number of coupling points and p the spatial order of the scheme. Another idea is the vectorization of the gradient evaluation, so that it is possible to evaluate the polynomial in one specific direction. This is beneficial, because the gradient data is typically required in the normal direction of the coupling face.

Besides reducing the coupling costs, knowing these costs in advance is a benefit for designing testcases accordingly. One potential direction is the investigation of a cost function for the polynomial evaluation of the DG method. The coupling costs depend on the order of the polynomial and the number of requested points. The order p is the more crucial factor since the complexity is $\mathcal{O}(p^3)$. In case a reliable cost function for state and gradient evaluation is known, the coupling cost can be predicted and used for load balancing within a subdomain instead of weights that are based

timings. Improving load balancing and therefore the weights in *Ateles* is offers great potential for future work: The calculation of weights per elements (4.2.1) can be fine-tuned to, for instance, better include the cost for internal level jumps or to measure the computation time per element explicitly instead of an average value of the partition. This could improve the load balancing of coupled simulation since the re-partition algorithm relies on the work for computation and coupling point evaluation.

7.2.3. Multi-solver approach: *preCICE*

One area for improvement is how the DG solver *Ateles* provides the coupling data to *preCICE*: Currently, the data mapping in space is exclusively done in *preCICE*. *Ateles* provides coupling data at the coupling points based on the numerical scheme and the modal-to-nodal transformations (Section 3.3.1). In the current implementation of *Ateles*, the computation of the DG-flux and the evaluation of coupling data are handled separately. For non-linear equations, however, nodal information could directly be provided by *Ateles* for reuse by *preCICE*, which would avoid an additional modal-to-nodal transformation.

Looking at the data mapping in space in general, the interpolation methods in *preCICE* could be improved: For a low order simulation in conjunction with a solver that provides mesh connectivity information, the second-order nearest-projection mapping is useful. For higher-order simulations, the general approach of Radial Basis functions sounds promising, but finding the optimal shape parameter is not trivial, especially for non-equidistant coupling points [72]. Hence, for the DG solver *Ateles* one idea is to convert its non-equidistant distributed points to equidistant points and provide these to the RBF interpolation of *preCICE*, for which results are discussed in [78].

A general challenge, independent of the numerical solver, is the performance on massive parallel systems. For highly parallel computations, the performance of the coupling library must not impact the overall performance. This holds true for *preCICE* except for the initialization phase. Another aspect of the initialization phase is the *Master-Slave* communication concept: The complete surface mesh of one coupling domain is communicated to the master rank of the other coupling domain. Therefore, the memory of the master rank is the limiting factor for the size of the coupling mesh. A workaround would be grant the master rank exclusive access to a CPU, which guarantees the largest available memory. However,

this depends on the computing system, the distribution of processes, and the coupling scenario. In this work, a static load balancing based on measured weights within the solver *Ateles* is presented. Another approach is presented in [97], where a load balancing model is used to approximate the appropriate number of processes per domain. Similar to *APESmate*, it is promising to work towards a more dynamic balanced workload during run time.

A. Optimization of polynomial evaluation in *Ateles*

In this section, we present a optimization step. Coupling via boundary conditions always implies additional work at the coupling interface which introduces load imbalances: This work should be minimized. The costs when coupling with *APESmate* are mainly due to of the polynomial evaluation of the coupling variables at the coupling points, as described in Section 3.3.2. Therefore, this appendix describes the implementation of polynomial evaluation and show analysis of two different implementations. Additionally, more optimization potential is identified. As mentioned in Section 3.4, the coupling variables differ depending on which equation system is to be coupled: For Euler or Linearized Euler equations **only** the state variables are required; for coupling to the Navier-Stokes equations the state variables **and** their gradients in the normal direction to the coupling interface have to be exchanged. The gradient evaluation is more costly than the state evaluation, which poses a bottleneck when coupling Navier-Stokes equations. Thus, an optimization of the gradient evaluation will accelerate the entire coupling process. The difference between the two evaluations can be explained when recapitulating the individual steps of the implementation.
For the **state** variable, the evaluation has the following steps:

I Find corresponding element of requested point

II Convert physical point coordinates to reference coordinates

III Get modal coefficients by evaluating 3D tensor product of Legendre polynomials at reference coordinates

IV Multiply modal coefficients with **state** vector to obtain exact value at requested point

To evaluate the **gradient** of state variables these steps are required:

I Find corresponding element of requested point

II Convert physical point coordinates to reference coordinates

A Get modal coefficients for gradients of complete element:

A.1 Get modal coefficients of state variable

A.2 Differentiate modal values for each direction

III Get modal coefficients by evaluating 3D tensor product of Legendre polynomials at reference coordinates

IV Multiply modal coefficients with **modal coefficients for the gradients** to obtain exact value at requested point

While steps **I** - **IV** are (almost) identical for both evaluations, the gradient evaluation requires an additional step: **A** implements the gradient computation of the full polynomial ending in all modal coefficients of the gradient for the corresponding element. Regarding the complexity, steps **I** - **IV** are of complexity $\mathcal{O}(p^3)$, with the polynomial order p. The additional steps (**A.1**) - (**A.2**) for gradients are of the same complexity $\mathcal{O}(p^3)$ which offers a great potential for improvements.

Implementing the steps of the polynomial evaluation as described above will lead to a point-wise evaluation and we will refer to it as *naive* implementation. In contrast to the point-wise evaluation, an *optimized* implementation utilizes an element-wise evaluation. As presented in Section 3.3, the coupling points at the coupling interface are chosen to be the integration points of the modal-to-nodal transformation, that is, for example, the Gauss-Legendre integration. Hence, there are multiple coupling points per coupling element. More precisely, there are p coupling points per direction for each coupling element. For coupling domains that use higher spatial order and typically have coarser elements, it is very likely that one element needs to serve multiple requested points. Therefore, the optimization step includes the sorting of the requested points per element at first and then evaluates them element-wise. Thereby the elemental modal information of steps (**A.1**) - (**A.2**) in the gradient evaluation can be reused. Another advantage of the element-wise evaluation is that steps **I** and **II** can be done during initialization and the corresponding element and coordinates can be stored. Step **I** is implemented as a binary search in the octree representation of the *TreElM* mesh [5] and has typically the complexity $\mathcal{O}(\log n)$, with the number of elements n. Hence, the influence of this $\mathcal{O}(\log n)$ operations during computation is negligible. Step **II** is a plain scaling of the coordinates since *Ateles* works exclusively on cubic elements. Here, the costs are quantified with the help of timers that measure the time spent in the evaluation routines. Due to the implementation, the measurements of the evaluation do not include the binary search **I** and, thus. it is not represented in the measured timings.

To investigate the benefit of the *optimized* implementation we use a plain testcase: A 3D computational domain is split into two domains, left and right, with the yz-plane as the coupling interface. Each domain is constructed to have 16 elements at this interface. Both domain have varying orders between 4 to 100 but the number of elements remains constant. Figure A.1 compares the *optimized* implementation (solid line) to the

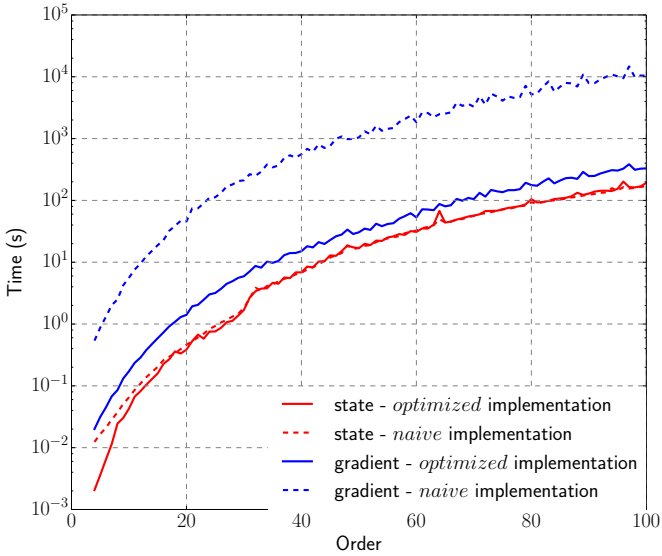


Figure A.1.: Comparison of the *naive* (dashed line) and the *optimized* (solid line) implementation for both evaluation.

naive one (dashed line). As expected, the optimization mainly improves the gradient evaluation whereas the state evaluation is not affected much. Only for orders $p < 20$ the optimization speeds up the state evaluation as well. On average, the *optimized* version is faster than the *naive* version by a factor of 34. For orders $p < 20$, the speedup is a little bit less. This can be explained by caching effects of the machine. The measurements were done on the SuperMuc IBM system at LRZ, Munich. For orders $p < 20$, the consumed memory fits into the L2 cache of the SuperMUC and the caching effects superpose the effects of the optimization. For the state evaluation, the speedup is 1.15 on average. Figure A.2 shows

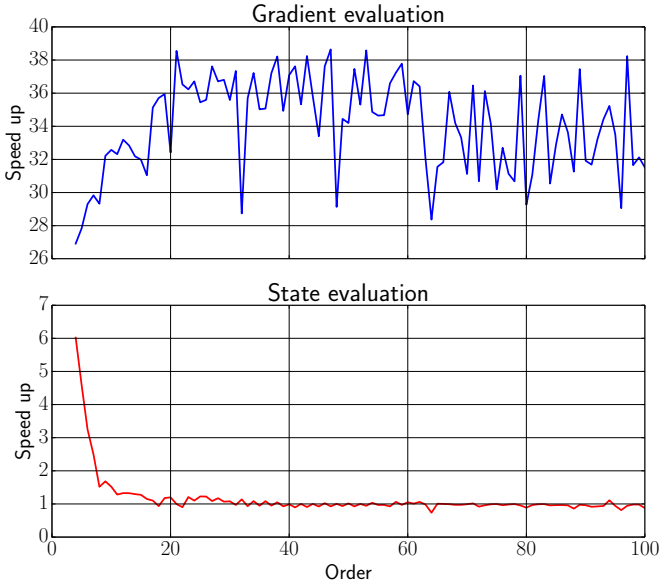


Figure A.2.: Speed up of the *optimized* implementation for the gradient (upper) and the state evaluation (lower).

that for small orders $p < 20$, the benefit is larger with a peak of 6 for a fourth-order evaluation. This indicates that for the caching region, the overhead for point-wise evaluation is rather large which is circumvented with the element-wise evaluation. Additionally, in Figure A.2 it can be seen that the speedup of the gradient evaluation for orders of 32, 48, 64, 80 and 96 is comparatively low.

Even with the optimization, there is clearly a difference between the evaluation of gradient variables and state variables, presented in Figure A.1. With the optimization, the difference between state and gradient evaluation reduces from a mean factor around 79 to a mean factor of 2.6. For sake of completeness, Figure A.3 illustrates the coupling costs for a typical Navier-Stokes and Euler coupling: One domain evaluates state variables (dashed red line) only, while the other domain evaluates "state+gradients" (solid grey line). Obviously, the "state+gradients" evaluation is the individual sum of the state evaluation and the gradient evaluations. By speeding up

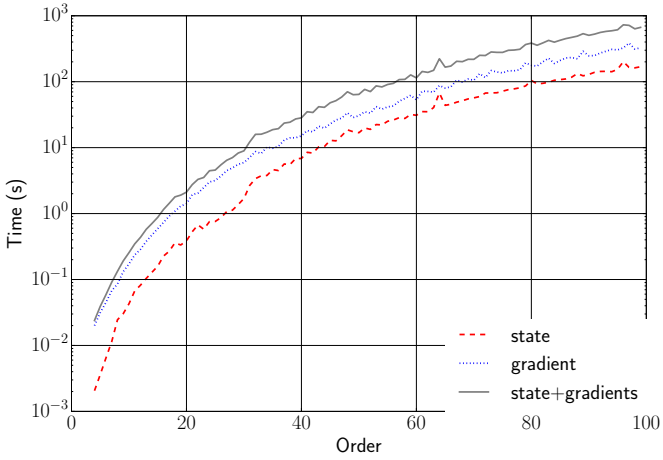


Figure A.3.: Comparison of both evaluations required for coupling to the Euler equations: state variables (dashed/red line) and to the Navier-Stokes equations: state variables + their gradients (solid/grey line) in the *optimized* version. Additional the pure gradient (dotted/blue line) evaluation is plotted.

of the gradient evaluation, the factor between state and "state+gradients" evaluation reduces on average from 80 to 4.6. Depending on the order, this difference decreases from a maximum factor of 146 to 11. As discussed in Section 3.2.1, different subdomains can have different workload, e.g. Navier-Stokes equations are more costly than Linearized Euler equations. Besides the load of solving the equation system, also the coupling cost contributes to the workload of a subdomain. Hence, with the presented optimization, the imbalances introduced by different coupling costs (due to state or gradient evaluation) on different subdomains are highly reduced.

There is an additional benefit to the element-wise implementation. Regarding real-world scenarios, there might be "bad location elements". "Bad location elements" are elements with increased coupling costs due to their location. There are two reasons for this: First, the number of requested points for such an element increases, e.g. when coupling different grid resolutions, such that one coupling element might need to evaluate points from four or more elements; second, an element needs to provide gradient

information for two or three normal directions, e.g. elements located on a corner or at an edge, which requires the costly element evaluation (step **A**) to be executed two or three times. By using an element-wise evaluation instead of a point-wise evaluation, the high cost of the element evaluation for the gradient computation is minimized to only one per element. This is particularly observable when providing coupling data for a larger number of requested points. Also, in case of gradient evaluation in two or three normal directions, this optimization speeds up the whole evaluation. Still, the costs of such elements are two or three times higher, which cannot be avoided. However, the optimization step reduces coupling costs of such a "bad location element" and, therefore, decreases the difference of coupling cost between all coupling elements (as well as of course all elements which are not involved in coupling). As described in Section 3.2.2, load imbalances within a subdomain can be introduced by different loads on the elements of these subdomains. With the optimization the cost difference between the coupling elements, the coupling elements at bad locations, and non-coupling elements is reduced. Hence, the load imbalances introduced by coupling are minimized which is essential for good load balancing.

In this appendix, we have presented an important optimization based on an element-wise implementation of the polynomial evaluation with the DG solver *Ateles*. As shown, this optimization step yields a speedup of up to 38 (on average 34) for the gradient evaluation and does not affect the state evaluation. With this speedup, the cost difference between coupling state variables and coupling state+gradient variables is reduced from 80 to 2.6. This is essential since we can not avoid to couple the gradients in addition to the state variables when coupling to the Navier-Stokes equations. Besides, with the shown optimization step, we minimized the impact of the coupling location which is relevant for realistic testcases in which, for example, corner elements exist. Minimizing the load imbalances within a domain enables real-world applications. With the optimization step the real-world scenarios presented in Section 5.2 coupled with *APESmate* shows faster computation times than a similar monolithic simulation. This optimization demonstrates an important step towards efficient coupling using the integrated approach *APESmate* that utilizes *Ateles*.

B. Performance of the initialization phase of *APESmate* using MPI_ALLTOALL

Section 4.4.6 has presented the performance of the initialization phase of *APESmate* when using a sparse all-to-all communication. In this appendix, we show why the usage of `MPI_ALLTOALL` instead of the sparse all-to-all is not advisable.

During the initialization phase the *point-to-point* communication for the exchange of coupling data in the simulation has to be established. For

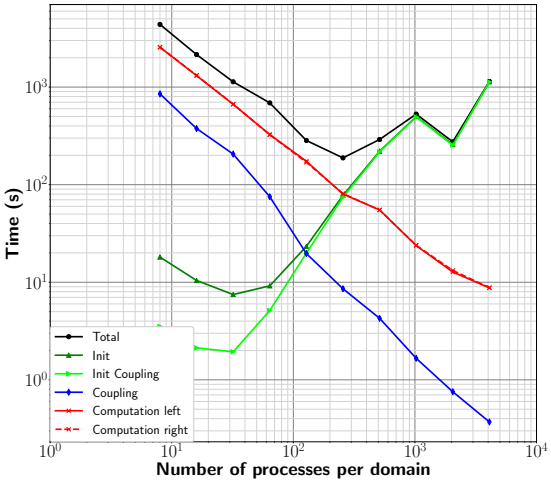


Figure B.1.: Strong scaling using *APESmate* and standard MPI_ALLTOALL for the initialization phase up to 1 island (= 8192 MPI ranks) on the SuperMuc Phase 1 IBM system. The total runtime is split into individual subroutines.

the establishment of point-to-point communication, one round of global communication is necessary. The main coupling information is communi-

cated via *Round Robin* so that each coupling process subsequently knows which data to provide to which process during simulation. Usually, the `MPI_ALLTOALL` routine provided by the MPI standard was used. We have observed a weak scaling of the initialization routines, which is shown in Figure B.1. For comparison, the same testcase as well as the same setup as for the strong scaling in Figure 4.10 of Section 4.4.6 are utilized.

The impact of the initialization of the coupling is so high that it influences the total runtime (for 100 timesteps). Actually, the weak scaling of the total computation time arises from the initialization of the coupling communication, namely from *Round Robin*, *Exchange Ranks* and *Exchange Cpl Data* subroutines. This is due to the dense `MPI_ALLTOALL` calls. With the MPI-3 standard [98], more non-blocking collectives like `MPI_IBARRIER`, `MPI_ISSEND`, `MPI_IRECV (MPI_ANY_SOURCE)` were introduced. With the help of those non-blocking collectives a **sparse** all-to-all communication [99] can be realized in which only a target rank and data to send need to be provided. This sparse communication is mainly useful in the case where there are only few actual communication partners in a large communicator. Figure B.2 shows the scaling behavior using sparse all-to-all

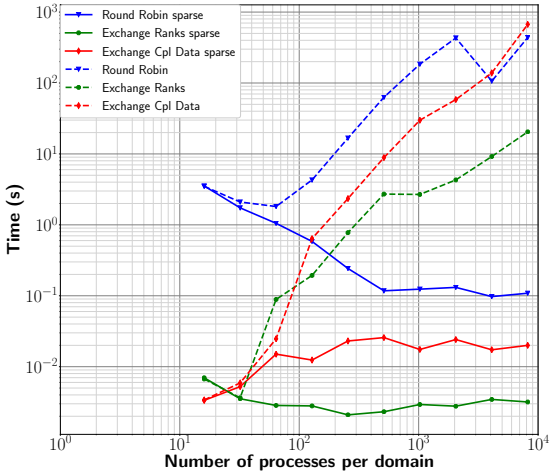


Figure B.2.: Strong Scaling of the costly initialization subroutines and comparison using dense MPI all-to-all communication (dashed lines) or a sparse version of all-to-all communication (solid lines).

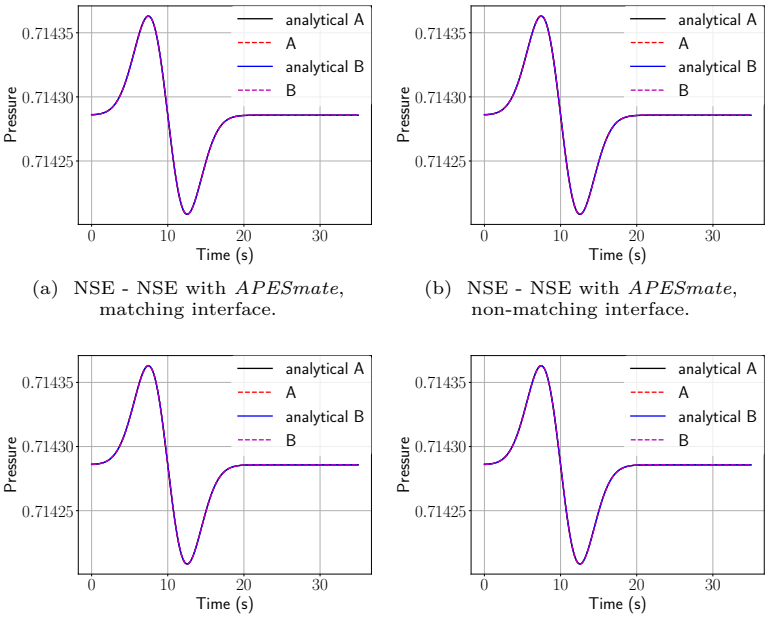
communication in comparison with the standard `MPI_ALLTOALL` communication. Dashed lines denote the dense `MPI_ALLTOALL` communication, while solid lines represent sparse all-to-all communication. We can see that all three subroutines benefit from the sparse all-to-all communication. With `MPI_ALLTOALL`, none of the subroutines scale but require increasing time with an increasing number of processes. With the sparse all-to-all communication, the timings for the subroutines reach a plateau at 512. For the `Round Robin`, the sparse communication has the most benefit and it shows good scalability for up to 512 processes.

So, the usage of `MPI_ALLTOALL` has an immense impact of the scalability of the initialization phase of *APESmate*. For simulations with *APESmate* on massive parallel systems the *sparse* all-to-all communication should be used.

This is configured with setting `use_sparse_ alltoall = true` in the lua configuration file of *APESmate*.

C. Figures for investigation of the Gaussian distribution in pressure

In this appendix, we present all figures for the investigation of the academic testcase described in Section 5.1. The testcase is the 3D Gaussian distribution in pressure. For the evaluation and discussion please refer to the previously mentioned section.



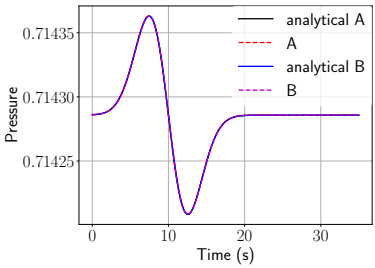
(a) NSE - NSE with *APESmate*, matching interface.

(b) NSE - NSE with *APESmate*, non-matching interface.

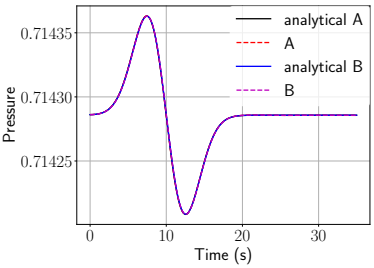
(c) NSE - NSE with *preCICE*, matching interface.

(d) NSE - NSE with *preCICE*, non-matching interface.

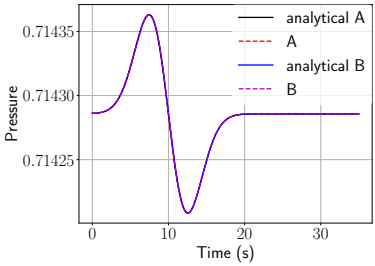
Figure C.1.: NSE - NSE coupling: Comparison of numerical and analytical results in both subdomains for coupling of **same** equations with matching interface (both subdomains $h = 1, \mathcal{O} = 6$) and non-matching interfaces (inner: $h = 1, \mathcal{O} = 6$; outer: $h = 5, \mathcal{O} = 12$).



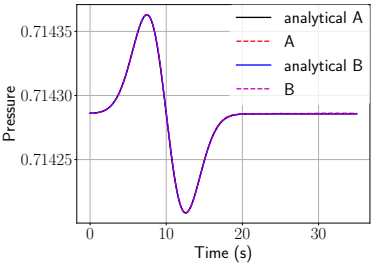
(a) EE - EE with *APESmate*, matching interface.



(b) EE - EE with *APESmate*, non-matching interface.

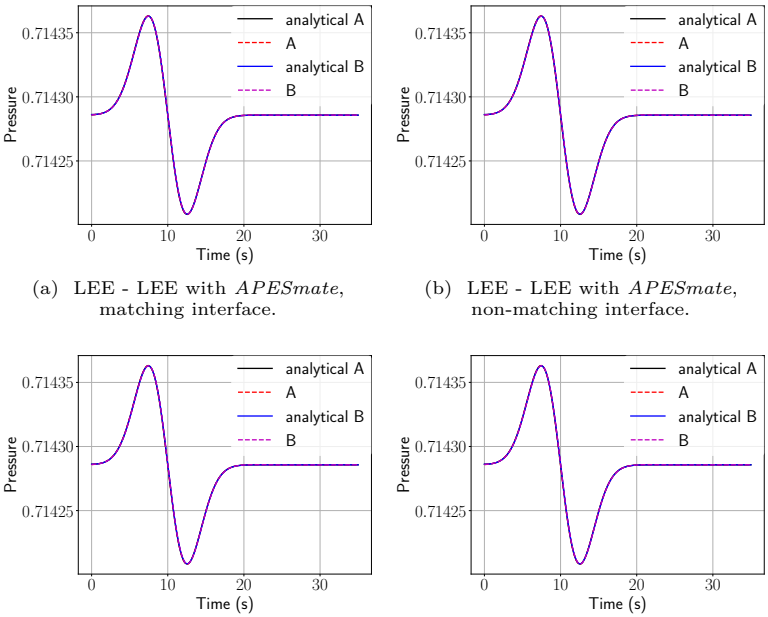


(c) EE - EE with *preCICE*, matching interface.



(d) EE - EE with *preCICE*, non-matching interface.

Figure C.2.: EE - EE coupling: Comparison of numerical and analytical results in both subdomains for coupling of **same** equations with matching interface (both subdomains $h = 1, \mathcal{O} = 6$) and non-matching interfaces (inner: $h = 1, \mathcal{O} = 6$; outer: $h = 5, \mathcal{O} = 12$).



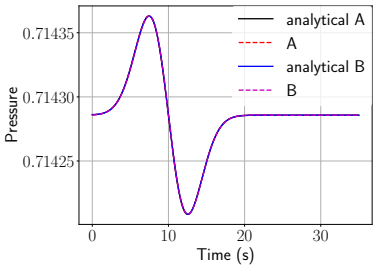
(a) LEE - LEE with *APESmate*, matching interface.

(b) LEE - LEE with *APESmate*, non-matching interface.

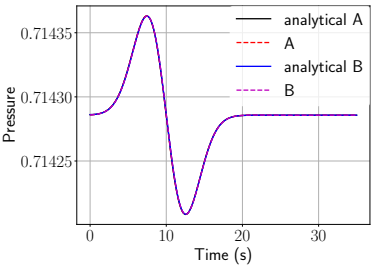
(c) LEE - LEE with *preCICE*, matching interface.

(d) LEE - LEE with *preCICE*, non-matching interface.

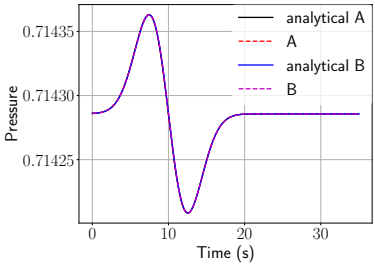
Figure C.3.: LEE - LEE coupling: Comparison of numerical and analytical results in both subdomains for coupling of **same** equations with matching interface (both subdomains $h = 1, \mathcal{O} = 6$) and non-matching interfaces (inner: $h = 1, \mathcal{O} = 6$; outer: $h = 5, \mathcal{O} = 12$).



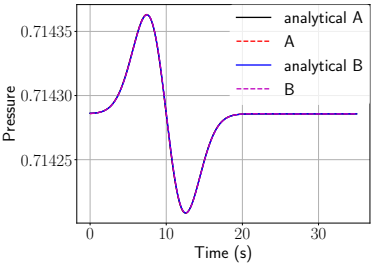
(a) NSE - EE with *APESmate*, matching interface.



(b) NSE - EE with *APESmate*, non-matching interface.

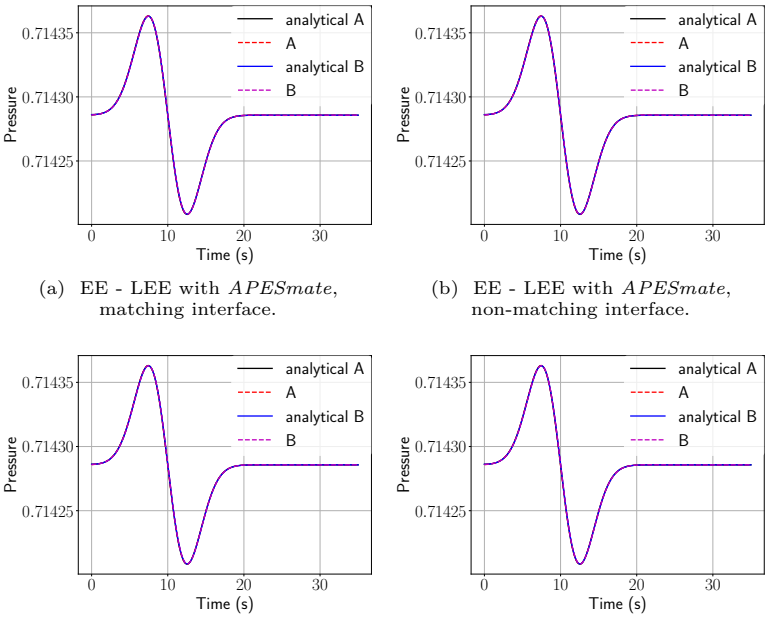


(c) NSE - EE with *preCICE*, matching interface.



(d) NSE - EE with *preCICE*, non-matching interface.

Figure C.4.: NSE - EE coupling: Comparison of numerical and analytical results in both subdomains for coupling **different** equations with matching interface (both subdomains $h = 1, \mathcal{O} = 6$) and non-matching interfaces (inner: $h = 1, \mathcal{O} = 6$; outer: $h = 5, \mathcal{O} = 12$).



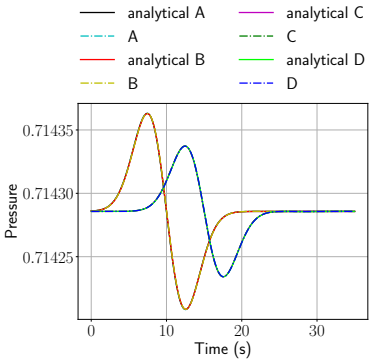
(a) EE - LEE with *APESmate*, matching interface.

(b) EE - LEE with *APESmate*, non-matching interface.

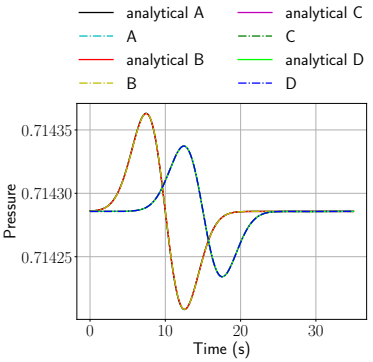
(c) EE - LEE with *preCICE*, matching interface.

(d) EE - LEE with *preCICE*, non-matching interface.

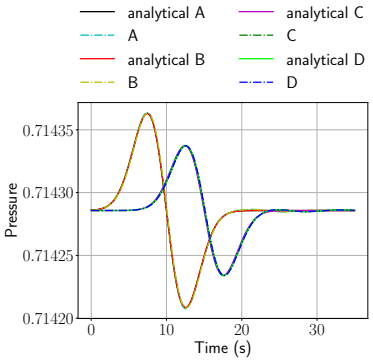
Figure C.5.: EE - LEE coupling: Comparison of numerical and analytical results in both subdomains for coupling **different** equations with matching interface (both subdomains $h = 1, \mathcal{O} = 6$) and non-matching interfaces (inner: $h = 1, \mathcal{O} = 6$; outer: $h = 5, \mathcal{O} = 12$).



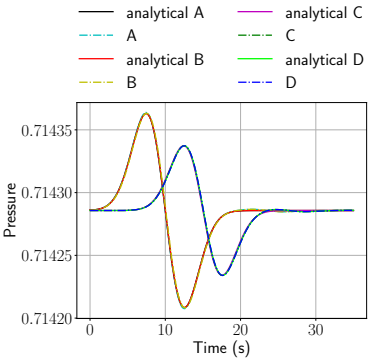
(a) NSE - EE - LEE with *APESmate*, matching interface.



(b) NSE - EE - LEE with *APESmate*, non-matching interface.



(c) NSE - EE - LEE with *preCICE*, matching interface.



(d) NSE - EE - LEE with *preCICE*, non-matching interface.

Figure C.6.: NSE - EE - LEE coupling: Comparison of numerical and analytical results for the coupling of **three different** equations with matching interface (both subdomains $h = 1, \mathcal{O} = 6$) and non-matching interfaces (inner: $h = 1, \mathcal{O} = 6$; middle: $h = 2.5$; $\mathcal{O} = 10$; outer: $h = 5, \mathcal{O} = 12$).

D. Iterative use of *SPartA*

Here, we investigate the use of the load balancing algorithm *SPartA* in an "iterative" way. This means we run the simulation of the jet setup **A** (from Section 5.2.3)) for a certain number of timesteps balanced with *SPartA*, measuring new weights, rerun the simulation for for a certain number of timesteps with *SPartA* etc. This re-distributes the loads, and thereby the elements, which can result in different communication patterns that might influence the overall performance. Table D.1 presents the computation

SPartA iteration	computation time [s]
1	80.3
2	79.3
3	80.0
4	81.3
5	76.6
6	77.2
7	79.1
8	84.4

Table D.1.: Iterative usage of the load balancing algorithm *SPartA*, computation time is without initialization time.

times (excluding the initialization time) for iterative usage of the load balancing algorithm *SPartA*. We see that using *SPartA* iteratively does not give any benefits. The timings spread closely around 80 s. This might be different for another simulation scenario, though, where the boundary elements and, hence, the communication patterns are strongly influenced by the balancing algorithm.

Bibliography

[1] R. Sordello et al. “Evidence of the environmental impact of noise pollution on biodiversity: a systematic map protocol”. In: *Environmental Evidence* 8.1 (2019), p. 8. DOI: 10.1186/s13750-019-0146-6.

[2] L. D. Knopper and C. A. Ollson. “Health effects and wind turbines: A review of the literature”. In: *Environmental Health* 10.78 (2011). DOI: 10.1186/1476-069X-10-78.

[3] E. Pedersen and K. Persson Waye. “Perception and annoyance due to wind turbine noise—a dose—response relationship”. In: *The Journal of the Acoustical Society of America* 116 (2005), pp. 3460–3470. DOI: 10.1121/1.1815091.

[4] I. van Kamp and F. van den Berg. “Health Effects Related to Wind Turbine Sound, Including Low-Frequency Sound and Infrasound”. In: *Acoustics Australia* 46.1 (2018), pp. 31–57. DOI: 10.1007/s40857-017-0115-6.

[5] H. Klimach. “Parallel Multi-Scale Simulations with Octrees and Coupled Applications”. PhD Thesis. RWTH Aachen, 2016.

[6] J. S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. 1st ed. Springer, 2007.

[7] G. Gassner. “Discontinuous Galerkin Methods for the Unsteady Compressible Navier-Stokes Equations”. PhD Thesis. Universität Stuttgart, 2009.

[8] J. Zudrop. “Efficient Numerical Methods for Fluid- and Electrodynamics on Massively Parallel Systems”. PhD Thesis. RWTH Aachen, 2015.

[9] F. Farassat and M. Myers. “Extension of Kirchhoff’s formula to radiation from moving surfaces”. In: *Journal of Sound and Vibration* 123.3 (1988), pp. 451–460. DOI: 10.1016/S0022-460X(88)80162-7.

[10] M. Wang, J. B. Freund, and S. K. Lele. “Computational Prediction of Flow-Generated Sound”. In: *Annual Review of Fluid Mechanics* 38.1 (2006), pp. 483–512. DOI: 10.1146/annurev.fluid.38.050304.092036.

[11] M. Lighthill. “On Sound Generated Aerodynamically I. General Theory”. In: *Proceedings of The Royal Society London A* 211 (1107 1952), pp. 564–587. DOI: 10.1098/rspa.1952.0060.

[12] J. E. F. Williams et al. “Sound Generation by Turbulence and Surfaces in Arbitrary Motion”. In: *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences* 264.1151 (1969), pp. 321–342. DOI: 10.1098/rsta.1969.0031.

[13] M. Kornhaas. “Effiziente numerische Methoden für die Simulation aero-akustischer Probleme mit kleinen Machzahlen”. PhD Thesis. Technische Universität Darmstadt, 2011.

[14] A. . Fedorchenko. “On Some Fundamental Flaws in Present Aeroacoustic Theory”. In: *Journal of Sound and Vibration* 232.4 (2000), pp. 719–782. DOI: 10.1006/jsvi.1999.2767.

[15] C. K. Tam. “Computational Aeroacoustics Examples Showing The Failure Of The Acoustic Analogy Theory To Identify The Correct Noise Sources”. In: *Journal of Computational Acoustics* 10.04 (2002), pp. 387–405. DOI: 10.1142/S0218396X02001607.

[16] S. B. Pope. *Turbulent Flows*. Cambridge University Press, 2000. DOI: 10.1017/CBO9780511840531.

[17] T. J. R. Hughes, L. Mazzei, and K. E. Jansen. “Large Eddy Simulation and the variational multiscale method”. In: *Computing and Visualization in Science* 3.1 (2000), pp. 47–59. DOI: 10.1007/s007910050051.

[18] S. Larsson and V. Thomée. *Partial Differential Equations with Numerical Methods*. 2003. DOI: 10.1007/978-3-540-88706-5.

[19] B. Cockburn and C.-W. Shu. “TVB Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws II: General Framework”. In: *Mathematics of Computation* 52.186 (1989), pp. 411–435.

[20] B. Cockburn, S.-Y. Lin, and C.-W. Shu. “TVB Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws III: One-dimensional Systems”. In: *Journal of Computational Physics* 84.1 (), pp. 90–113. DOI: 10.1016/0021-9991(89)90183-6.

[21] B. Cockburn, S. Hou, and C.-W. Shu. "The Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws IV: The Multidimensional Case". In: *Mathematics of Computation* 54.190 (1990), pp. 545–581.

[22] D. Di Pietro and A. Ern. *Mathematical Aspects of Discontinuous Galerkin Methods*. Springer Berlin Heidelberg, 2012. DOI: 10.1007/978-3-642-22980-0.

[23] R. Hartmann and P. Houston. "Symmetric Interior Penalty DG Methods for the Compressible Navier–Stokes Equations I: Method Formulation". In: *International Journal of Numerical Analysis & Modeling* 3 (2006), pp. 1–20.

[24] C. Canuto et al. *Spectral Methods: Evolution to Complex Geometries and Applications to Fluid Dynamics*. Springer Berlin Heidelberg, 2007. DOI: 10.1007/978-3-540-30728-0.

[25] W. S. Don. "Numerical Study of Pseudospectral Methods in Shock Wave Applications". In: *Journal of Computational Physics* 110.1 (1994), pp. 103–111. DOI: https://doi.org/10.1006/jcph.1994.1008.

[26] R. LeVeque. *Numerical Methods for Conservation Laws*. Lectures in Mathematics ETH Zürich, Department of Mathematics Research Institute of Mathematics. Birkhäuser Basel, 1990.

[27] J. C. Butcher. *Numerical Methods for Ordinary Differential Equations*. 2nd ed. John Wiley & Sons Ltd, 2008. DOI: 10.1002/9780470753767.ch1.

[28] J. H. Verner. "On deriving explicit Runge-Kutta methods". In: *Proceedings of the 1971 Conference on Applications of Numerical Analysis*. Springer Berlin Heidelberg, 1971, pp. 340–347. DOI: 10.1007/BFb0069470.

[29] W. Kutta. *Beitrag zur näherungsweisen Integration totaler Differentialgleichungen*. B.G Teubner, 1901.

[30] B. Cockburn and C.-W. Shu. "Runge–Kutta Discontinuous Galerkin Methods for Convection-Dominated Problems". In: *Journal of Scientific Computing* 16.3 (2001), pp. 173–261. DOI: 10.1023/A:1012873910884.

[31] J. C. Butcher. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*. USA: Wiley–Interscience, 1987. DOI: 10.5555/22730.

[32] S. Gottlieb, C.-W. Shu, and E. Tadmor. “Strong Stability-Preserving High-Order Time Discretization”. In: *SIAM Review* 43.1 (2001), pp. 89–112.

[33] T. Schwartzkopff and C.-D. Munz. “Direct Simulation of Aeroacoustics”. In: *Analysis and Simulation of Multifield Problems*. Springer Berlin Heidelberg, 2003, pp. 337–342.

[34] M. Borrel, L. Halpern, and J. Ryan. “Euler/Navier-Stokes couplings for multiscale aeroacoustic problems”. In: *20th AIAA Computational Fluid Dynamics Conference*. 2011. DOI: 10.2514/6.2011-3047.

[35] J. Ryan, L. Halpern, and M. Borrel. “Domain decomposition vs. overset Chimera grid approaches for coupling CFD and CAA”. In: *Proceedings of the 7th International Conference on Computational Fluid Dynamics (ICCFD7)*. 2012.

[36] T. Schwartzkopff. “Finite-Volumen Verfahren hoher Ordnung und heterogene Gebietszerlegung für die numerische Aeroakustik”. PhD Thesis. Universität Stuttgart, 2005.

[37] J. Utzmann. “A Domain Decomposition Method for the Efficient Direct Simulation of Aeroacoustic Problems by”. PhD Thesis. Universtität Stuttgart, 2008.

[38] W. Joppich and M. Kürschner. “MpCCI—a tool for the simulation of coupled applications”. In: *Concurrency and Computation: Practice and Experience* 18.2 (2006), pp. 183–192. DOI: 10.1002/cpe.913.

[39] H.-J. Bungartz et al. “preCICE – A fully parallel library for multi-physics surface coupling”. In: *Computers and Fluids* 141 (2016), pp. 250–258. DOI: 10.1016/j.compfluid.2016.04.003.

[40] B. Gatzhammer. “Efficient and Flexible Partitioned Simulation of Fluid- Structure Interactions”. PhD Thesis. Technische Universität München, 2014.

[41] B. Uekermann. “Partitioned Fluid-Structure Interaction on Massively Parallel Systems”. PhD Thesis. Technische Universität München, 2016.

[42] H.-J. Bungartz et al. “Partitioned Fluid–Structure–Acoustics Interaction on Distributed Data: Coupling via preCICE”. In: *Software for Exascale Computing – SPPEXA 2013-2015*. Springer International Publishing, 2016, pp. 239–266.

[43] A. K. Shukaev. "A Fully Parallel Process-to-Process Intercommunication Technique for preCICE". Master's Thesis. Technische Universität München, 2015.

[44] D. Blom et al. "On parallel scalability aspects of strongly coupled partitioned fluid-structure-acoustics interaction". In: *Proceedings of the 6th International Conference on Computational Methods for Coupled Problems in Science and Engineering*. CIMNE, 2015, pp. 556–565.

[45] D. Blom et al. "Partitioned Fluid-Structure-Acoustics Interaction on Distributed Data: Numerical Results and Visualization". In: *Software for Exascale Computing – SPPEXA 2013-2015*. Springer International Publishing, 2016, pp. 267–291.

[46] T. Reimann et al. "Multifield coupling of a fluid-structure-acoustics interaction problem in low-Mach number turbulent flow". In: *6th European Conference on Computational Mechanics (ECCM 6) / 7th European Conference on Computational Fluid Dynamics (ECFD 7)*. 2018.

[47] H.-J. Bungartz, P. Neumann, and W. Nagel, eds. *Software for Exascale Computing - SPPEXA 2013-2015*. Springer International Publishing, 2016. DOI: 10.1007/978-3-319-40528-5.

[48] H.-J. Bungartz et al., eds. *Software for Exascale Computing - SPPEXA 2016-2019*. Springer International Publishing, 2020. DOI: 10.1007/978-3-030-47956-5.

[49] H. Bijl et al. "ExaFSA–Exascale Simulation of Fluid-Structure-Acoustics Interactions". In: *InSiDE Magazine* (Autumn 2014).

[50] P. Bastian et al. *The Distributed and Unified Numerics Environment (DUNE)*.

[51] P. Bastian et al. "A generic grid interface for parallel and adaptive scientific computing. Part I: Abstract framework". In: *Computing* 82 (2008), pp. 103–119. DOI: 10.1007/s00607-008-0003-x.

[52] A. Dedner et al. "The DUNE-FEM-DG module." In: *Archive of Numerical Software* 5 (2017), No 1. DOI: 10.11588/ANS.2017.1.28602.

[53] J. Teresco, K. Devine, and J. Flaherty. "Partitioning and Dynamic Load Balancing for the Numerical Solution of Partial Differential Equations". In: *Numerical Solution of Partial Differential Equations on Parallel Computers*. 2006, pp. 55–88. DOI: 10.1007/3-540-31619-1_2.

[54] K. Schloegel, G. Karypis, and V. Kumar. "A Unified Algorithm for Load-balancing Adaptive Scientific Simulations". In: 2000, pp. 59–59. DOI: 10.1109/SC.2000.10035.

[55] D. F. Harlacher et al. "Dynamic Load Balancing for Unstructured Meshes on Space-Filling Curves". In: *Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*. 2012, pp. 1661–1669. DOI: 10.1109/IPDPSW.2012.207.

[56] L. D. Landau and E. M. Lifshitz. *Fluid Mechanics, Second Edition: Volume 6 (Course of Theoretical Physics)*. 2nd ed. Butterworth-Heinemann, 1987.

[57] E. F. Toro. *Riemann solvers and numerical methods for fluid dynamics*. 2nd ed. Springer, 1999.

[58] A. Beck, G. Gassner, and C.-D. Munz. "On the Effect of Flux Functions in Discontinuous Galerkin Simulations of Underresolved Turbulence". In: *Spectral and High Order Methods for Partial Differential Equations - ICOSAHOM 2012*. Springer International Publishing, 2014, pp. 145–155. DOI: 10.1007/978-3-319-01601-6_11.

[59] V. Titarev and E. Toro. "ADER schemes for three-dimensional nonlinear hyperbolic systems". In: *Journal of Computational Physics* 204.2 (2005), pp. 715–736. DOI: 10.1016/j.jcp.2004.10.028.

[60] F. Lindner et al. "A comparison of various quasi-Newton schemes for partitioned fluid-structure interaction". In: *Proceedings of 6th International Conference on Computational Methods for Coupled Problems in Science and Engineering*. 2015, pp. 1–12.

[61] D. Blom et al. "A Review on Fast Quasi-Newton and Accelerated Fixed-Point Iterations for Partitioned Fluid–Structure Interaction Simulation". In: *Advances in Computational Fluid-Structure Interaction and Flow Simulation: New Methods and Challenging Computations*. Springer International Publishing, 2016, pp. 257–269. DOI: 10.1007/978-3-319-40827-9_20.

[62] F. G. Lether. "On the construction of Gauss-Legendre quadrature rules". In: *Journal of Computational and Applied Mathematics* 4.1 (1978), pp. 47–52. DOI: http://dx.doi.org/10.1016/0771-050X(78)90019-0.

[63] C. Clenshaw and A. Curtis. "A method for numerical integration on an automatic computer." In: *Numerische Mathematik* 2 (1960), pp. 197–205.

[64] N. Anand, H. Klimach, and S. Roller. "Dealing with non-linear terms in a modal High-Order Discontinuous Galerkin Method". In: *Proceedings of the 2016 Joint Workshop on Sustained Simulation Performance, University of Stuttgart (HLRS) and Tohoku University*. Springer International Publishing, 2016.

[65] S. Roller et al. "An Adaptable Simulation Framework Based on a Linearized Octree". In: *High Performance Computing on Vector Systems 2011*. Springer Berlin Heidelberg, 2012, pp. 93–105. DOI: 10.1007/978-3-642-22244-3_7.

[66] V. Krupp et al. "Efficient Coupling of Fluid and Acoustic Interaction on Massive Parallel Systems". In: *Proceedings of the 2016 Joint Workshop on Sustained Simulation Performance, University of Stuttgart (HLRS) and Tohoku University*. Springer International Publishing, 2016, pp. 61–81. DOI: 10.1007/978-3-3199-46735-1_6.

[67] H. G. Klimach et al. "Distributed Octree Mesh Infrastructure for Flow Simulations". In: *Proceedings of the 2012 European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS)*. 2012.

[68] D. F. Harlacher et al. "Tree Based Voxelization of STL Data". In: *High Performance Computing on Vector Systems 2011*. Springer Berlin Heidelberg, 2012, pp. 81–92. DOI: 10.1007/978-3-642-22244-3_6.

[69] H. G. Klimach, J. Zudrop, and S. P. Roller. "Generation of high order geometry representations in Octree meshes". In: *PeerJ Computer Science* 1 (Nov. 2015), e35. DOI: 10.7717/peerj-cs.35.

[70] B. K. Alpert and V. 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.

[71] J. Qi. "Efficient Lattice Boltzmann Simulations on Large Scale High Performance Computing Systems". PhD Thesis. RWTH Aachen, 2017.

[72] F. Lindner. "Data Transfer in Partitioned Multi-Physics Simulations: Interpolation & Communication". PhD Thesis. Universität Stuttgart, 2019.

[73] A. de Boer, A. van Zuijlen, and H. Bijl. "Comparison of conservative and consistent approaches for the coupling of non-matching meshes". In: *Computer Methods in Applied Mechanics and Engineering* 197.49-50 (2008), pp. 4284–4297. DOI: 10.1016/j.cma.2008.05.001.

[74] R. Yokota, L. A. Barba, and M. G. Knepley. "PetRBF — A parallel O(N) algorithm for radial basis function interpolation with Gaussians". In: *Computer Methods in Applied Mechanics and Engineering* 199.25-28 (2010), pp. 1793–1804. DOI: 10.1016/j.cma.2010.02.008.

[75] S. Deparis, D. Forti, and A. Quarteroni. "A Rescaled Localized Radial Basis Function Interpolation on Non-Cartesian and Nonconforming Grids". In: *SIAM Journal on Scientific Computing* 36.6 (2014), A2745–A2762. DOI: 10.1137/130947179.

[76] M. Buhmann. "Radial basis functions". In: *Acta Numerica* 9 (2000), pp. 1–38.

[77] F. Lindner, M. Mehl, and B. Uekermann. "Radial Basis Function Interpolation for Black-Box Multi-Physics Simulations". In: *Proceedings of the VII International Conference on Coupled Problems in Science and Engineering*. CIMNE, 2017, pp. 50–61.

[78] N. Ebrahimi Pour et al. "Coupled Simulation with Two Coupling Approaches on Parallel Systems". In: *Proceedings of the 2017 Joint Workshop on Sustained Simulation Performance, University of Stuttgart (HLRS) and Tohoku University*. Springer International Publishing, 2017, pp. 151–164. DOI: 10.1007/978-3-319-66896-3_10.

[79] A. de Boer, A. H. van Zuijlen, and H. Bijl. "Comparison of the conservative and a consistent approach for the coupling of non-matching meshes". In: *European Conference on Computational Fluid Dynamics* (2006), pp. 1–19.

[80] M. Mongillo. "Choosing Basis Functions and Shape Parameters for Radial Basis Function Methods". In: *SIAM* (2011), pp. 190–209.

[81] H.-J. Bungartz et al. "Fluid-Acoustics Interaction on Massively Parallel Systems". In: *International Workshop on Computational Engineering CE 2014*. Lecture Notes in Computational Science and Engineering. Heidelberg, Berlin: Springer, 2015.

[82] K. Masilamani. "Coupled Simulation Framework to Simulation Electrodialysis Processes for Seawater Desalination". PhD Thesis. RWTH Aachen, Submitted in 2019.

[83] K. Masilamani, H. Klimach, and S. Roller. "Highly Efficient Integrated Simulation of Electro-Membrane Processes for Desalination of Sea Water". In: *High Performance Computing in Science and Engineering '13*. Springer International Publishing, 2013, pp. 493–508. DOI: 10.1007/978-3-319-02165-2_34.

[84] C. K. Tam. *Computational Aeroacoustics: A Wave Number Approach*. Cambridge Aerospace Series. Cambridge University Press, 2012.

[85] C. Bogey and C. Bailly. "Investigation of downstream and sideline subsonic jet noise using Large Eddy Simulation". In: *Theoretical and Computational Fluid Dynamics* 20.1 (2006), pp. 23–40.

[86] C. Bogey and C. Bailly. "Computation of a high Reynolds number jet and its radiated noise using large eddy simulation based explicit filtering". In: *Computers & Fluids* 35 (2006), pp. 1344–1358. DOI: 10.1016/j.compfluid.2005.04.008.

[87] C. K. Tam et al. "The Sources of Jet Noise: Experimental Evidence". In: *Journal of Fluid Mechanics* 615 (2008), pp. 253–292. DOI: 10.1017/S0022112008003704.

[88] S. Karabasov. "Understanding jet noise". In: *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences* 368 (2010), pp. 3593–3608. DOI: 10.1098/rsta.2010.0086.

[89] M. Dumbser et al. "A unified framework for the construction of one-step finite volume and discontinuous Galerkin schemes on unstructured meshes". In: *Journal of Computational Physics* 227.18 (2008), pp. 8209–8253. DOI: 10.1016/j.jcp.2008.05.025.

[90] J. Berland, C. Bogey, and C. Bailly. "Numerical study of screech generation in a planar supersonic jet". In: *Physics of Fluids* 19.7 (2007), p. 075105. DOI: 10.1063/1.2747225.

[91] D. Casalino and S. Lele. "Lattice-Boltzmann Simulation of Coaxial Jet Noise Generation". In: *Proceedings of the Summer Program, Center for Turbulence Research, Stanford, CA*. 2014, pp. 231–240.

[92] N. Ebrahimi Pour. "Efficient high-order simulation of aeroacoustics from rigid body motion on massively parallel systems". PhD thesis. Universität Siegen, 2021.

[93] N. 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). DOI: 10.3390/sym11091126.

[94] N. Ebrahimi Pour et al. "Load Balancing for Immersed Boundaries in Coupled Simulations". In: *Proceedings of the 2018 and 2019 Joint Workshops on Sustained Simulation Performance, University of Stuttgart (HLRS) and Tohoku University*. Springer International Publishing, 2019, pp. 185–201. DOI: 10.1007/978-3-030-39181-2_15.

[95] M. Ghasemian and A. Nejat. "Aerodynamic Noise Computation of the Flow Field around NACA 0012 Airfoil Using Large Eddy Simulation and Acoustic Analogy". In: *Journal of Computational Applied Mechanics* 46.1 (2015), pp. 41–50. DOI: 10.22059/jcamech.2015.53392.

[96] M. Gaiga. "Implementierung einer lokal linearisierten Euler-Gleichung in den modalen Discontinuous Galerkin Löser Ateles". Bachelor's Thesis. Universität Siegen, 2017.

[97] A. Totounferoush et al. "A new load balancing approach for coupled multi-physics simulations". In: *Proceedings of the 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 2019. DOI: 10.1109/IPDPSW.2019.00115.

[98] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard Version 3.0*.

[99] T. Hoefler and J. Träff. "Sparse collective operations for MPI". In: *Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing*. 2009, pp. 1–8. DOI: 10.1109/IPDPS.2009.5160935.

List of Figures

List of Tables

Simulation Techniques in Siegen
Vol. 5

This work presents an efficient coupling strategy to tackle the multi-scale problem posed by technical devices that radiate sound. Two different approaches, a flexible multi-solver approach (with the coupling library preCICE) and an optimized integrated approach (called APESmate), are established and compared in terms of quality, load balancing, and performance. Load balancing is a crucial aspect regarding simulations in the field of high performance computing where many processes are used in parallel and, in particular, when considering coupled simulation where different partitions are solving different physical phenomena and numerical discretization. The benefits of the partitioned coupling approach and good load balancing are demonstrated on an industrial application of a 3D free-stream jet with a high Reynolds number showing that a multi-scale problem can be efficiently simulated using today's computing resources.

Verena Krupp studied mechanical engineering at H-BRS and Simulation Sciences at the GRS/RWTH Aachen; 2013-2016 she worked in the DFG funded project exaFSA on the efficient coupling of fluid and acoustics. From 2013-2017 she was working in research at the chair for Simulation Techniques and Scientific Computing at the University of Siegen.

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: 978-3-96182-104-4

universi
UNIVERSITÄTSVERLAG SIEGEN