Implicit Discontinuous Galerkin Shock Tracking Methods for Compressible Flows with Shocks

Novel Cut-Cell Approach and Linear Solvers
Zur Erlangung des Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.)
Genehmigte Dissertation von Jakob Vandergrift aus Frankfurt am Main
Tag der Einreichung: 30.05.24, Tag der Prüfung: 05.08.24

1. Gutachten: Prof. Dr.-Ing. habil. Martin Oberlack
2. Gutachten: Prof. Dr. Jan Giesselmann
Darmstadt, Technische Universität Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Maschinenbau
Fachgebiet für
Strömungsdynamik

Implicit Discontinuous Galerkin Shock Tracking Methods for Compressible Flows with Shocks
Novel Cut-Cell Approach and Linear Solvers

Genehmigte Dissertation von Jakob Vandergrift

Tag der Einreichung: 30.05.24
Tag der Prüfung: 05.08.24

Darmstadt, Technische Universität Darmstadt

Für Maraike

Erklärungen laut Promotionsordnung

§ 8 Abs. 1 lit. c PromO

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§ 8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

§ 9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation – abgesehen von den in ihr ausdrücklich genannten Hilfen – selbstständig verfasst wurde und dass die „Grundsätze zur Sicherung guter wissenschaftlicher Praxis an der Technischen Universität Darmstadt“ und die „Leitlinien zum Umgang mit digitalen Forschungsdaten an der TU Darmstadt“ in den jeweils aktuellen Versionen bei der Verfassung der Dissertation beachtet wurden.

§ 9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 30.05.24

J. Vandergrift

Zusammenfassung

Numerische Strömungssimulationen (engl. computational fluid dynamics (CFD)) spielen in Industrie und Forschung eine wesentliche und ergänzende Rolle bei der Analyse kompressibler und von Verdichtungsstößen dominierter Strömungen. Genaue und robuste Simulationen solcher Strömungen, wie sie zum Beispiel in der Umgebung von Flugzeugen vorkommen, stellen eine große Herausforderung für die heutigen CFD-Verfahren dar. Numerische Methoden höherer Ordnung, wie diskontinuierliche Galerkin-Methoden (DG), liegen im Fokus der Forschung, da sie unter anderem eine hohe Genauigkeit pro Freiheitsgrad und exzellente parallele Skalierbarkeit aufweisen. Typischerweise sind diese Methoden jedoch nicht robust genug für Stöße und Kontaktdiskontinuitäten, da deren geringe numerische Dissipation zu unerwünschten Oszillationen und letzlich zum Abbruch der Simulation führen kann.

In den letzten Jahren wurde eine vielversprechende neue Kategorie robuster DG-Methoden entwickelt, sogenannte *implizite Stoßanpassungsmethoden* (engl. *implicit shock tracking (IST) methods*). Diese berechnen mittels numerischer Optimierung DG-Approximationen hoher Ordnung an stoßdominierte Strömungen, indem sie das Rechengitter an die Stöße anpassen. Die Stöße können durch Diskontinuitäten zwischen den angepassten Gitterelementen exakt dargestellt werden und ermöglichen es Basisfunktionen hoher Ordnung glatte Bereiche der Strömung hochgenau zu approximieren. Infolgedessen können auch auf groben Gittern genaue Approximationen berechnet werden. In dieser Arbeit werden zwei bedeutende Fortschritte in der Entwicklung von IST-Methoden vorgestellt.

Erstens, wird ein neues numerisches Verfahren durch die Integration von IST und *erweiterter DG-Methoden (XDG)* entwickelt. Anders als bei herkömmlichen IST-Methoden, entfallen hierbei aufwendige und teure Gitteroperationen. Diese neuartige *implizite XDG-Stoßanpassungsmethode* (XDG-IST) verwendet eine Level-Set-Funktion um Diskontinuitätsschnittstellen zu definieren, entlang derer Sprünge in den XDG-Basisfunktionen zugelassen sind. Die Schnittstellen werden mittels IST an die Stöße angepasst, sodass letztere exakt dargestellt werden können. Die Robustheit und Genauigkeit der XDG-IST-Methode für stationäre 2D und instationäre 1D Strömungen wird gezeigt und sie wird mit einer DG-Methode verglichen, die künstliche Viskosität zur Stabilisierung verwendet. Der Vergleich zeigt, dass die XDG-IST-Methode eine höhere Genauigkeit aufweist und für akustische 1D Wellen, die mit Stößen interagieren, als einzige Methode Konvergenz hoher Ordnung aufzeigt.

Zweitens, um IST auf großskalige Probleme anwenden zu können, wird eine Familie von Vorkonditionierern entwickelt. Diese sind speziell für das effiziente Lösen der linearen Systeme ausgelegt, welche in IST-Methoden vorkommen. Die entwickelten Vorkonditionierer kombinieren etablierte Techniken aus der restringierten Optimierung zum Lösen linearer Systeme mit bewährten Methoden innerhalb von DG-Verfahren. Die wirksamsten Vorkonditionierer werden anhand numerischer Studien ausgewählt, welche die Reaktion auf kritische IST-Parameter messen.

Abstract

Computational fluid dynamics (CFD) play an essential role in both industry and research for analyzing compressible flows dominated by shocks, enhancing experimental and theoretical studies. Achieving accurate and robust simulations of these shock-dominated flows, such as those encountered around airplanes, poses a significant challenge for contemporary CFD techniques. High-order numerical methods, such as discontinuous Galerkin (DG) methods, have received considerable attention because: they introduce minimal numerical dissipation, are highly accurate per degree of freedom, provide geometric flexibility, and exhibit excellent parallel scalability. However, these methods often struggle with robustness in scenarios involving shocks and contact discontinuities, as the high-order approximation of shocks and discontinuities can induce spurious oscillations, leading to the failure of numerical solvers.

In recent years, a novel category of numerical methods, termed *implicit shock tracking (IST) methods*, has been developed, utilizing numerical optimization to achieve high-order DG approximations of shock-dominated flow solutions while aligning the computational mesh with non-smooth features. These methods represent these features accurately through inter-element jumps and allow high-order basis functions to approximate smooth areas of the flow without requiring nonlinear stabilization. As a consequence, IST methods achieve accurate approximations of shock-dominated flows even on traditionally coarse meshes. This dissertation introduces two significant advancements for IST.

First, we integrate ideas from IST with extended DG (XDG) methods, introducing the *implicit XDG shock tracking (XDG-IST) method*. This innovative method utilizes a level set function to define discontinuity interfaces, which segment but do not deform the computational grid, thereby circumventing cumbersome mesh operations used in conventional IST methods. The approximation space is enriched by XDG basis functions, discontinuous at the interfaces, and the latter are aligned accurately with shocks using IST methodologies. We successfully apply the method to various test scenarios, including steady two-dimensional (2D) and unsteady one-dimensional (1D) inviscid flow problems. We show that the XDG-IST method is more accurate and the only method maintaining high-order convergence properties for 1D shock-acoustic-wave interaction problems, when compared to a traditional DG method employing artificial viscosity.

Second, aiming at extending the use of IST to large-scale problems, we present a *family of preconditioners* tailored for the saddle point linear systems that define the progression towards optimality in each iteration of the optimization solver. These preconditioners merge traditional constrained optimization techniques with widely-used methods for DG discretizations, including block Jacobi, block incomplete LU factorizations and P-multigrid. Comprehensive evaluations are conducted using two 2D inviscid compressible flow scenarios to assess the effectiveness of each preconditioner within this family, and their responsiveness to key IST parameters, singling out the best preconditioner in terms of performance.

Acknowledgments

This dissertation is the result of my work at the Chair of Fluid Dynamics over the last three years. I would like to express my gratitude to everyone who has contributed to and made the completion of this work possible.

First and foremost, I would like to thank Prof. Dr.-Ing. habil. Martin Oberlack for enabling me to start my PhD journey at the Chair of Fluid Dynamics, providing me with all the necessary infrastructure, and allowing me to attend four conferences and two research visits abroad. I am also grateful for his initial idea for my research, his never-ending optimism, and his continuous guidance.

I would also like to thank Prof. Jan Giesselmann for agreeing to co-referee this dissertation.

Furthermore, I am grateful to Dr.-Ing. Florian Kummer, my direct supervisor at the Chair of Fluid Dynamics. I would like to thank him for his continuous efforts, including weekly individual and group meetings, his helpful guidance, his research ideas, his assistance in navigating the BoSSS code, and the valuable comments he provided for our publication. I also would like to thank him for diligently maintaining the IT and server infrastructure at the institute, enabling the work of the whole group. Additionally, I am pleased to have had the opportunity to supervise the exercise corresponding to his DG course and to assist him in maintaining and improving the institute's infrastructure. I have greatly enjoyed our collaboration and have learned much from it.

Additionally, I would like to express my gratitude to Prof. Matthew Zahr for hosting me during two research visits at the University of Notre Dame and organizing three group social events. My research, strongly based on his and his group's prior work, would not have been possible without their contributions. Moreover, I am grateful for the opportunity to have worked together with Prof. Zahr on a joint project. I gained significant scientific knowledge and thoroughly enjoyed the entire process.

Also, I would like to thank Prof. Dr.-Ing. habil. Yongqi Wang, for his professional support and for organizing the teaching at the Chair of Fluid Dynamics. I will also fondly remember the support in all administrative aspects provided by Ruth Völker, her great effort in organizing Christmas parties, and all the pleasant conversations that momentarily took my mind off work.

A special thanks also goes out to all colleagues at the Chair of Fluid Dynamics, with whom I have tremendously enjoyed working and who have collectively created a pleasant atmosphere. I would like to thank Muhammed Toprak, Cat-Tuong Nguyen, Simon Görtz, Toni Dokoza, Lara De Broeck, and Schahin Akbari. I enjoyed the wonderful time we spent together at work and privately. Finally, I am grateful to all of my other colleagues who have contributed to the pleasant environment at the institute, even if they are not explicitly mentioned. Your presence has greatly enriched my time at the Chair of Fluid Dynamics.

This dissertation has been greatly improved by the excellent feedback from Schahin Akbari, Lara De Broeck, Andrej Brojatsch, Toni Dokoza, Houssem Ben Gozlen, Simon Görtz, Dr.-Ing. Florian Kummer, Chen Miao, Cat-Tuong Nguyen, Miltiadis Poursanidis, Felician Putz, Matthias Rieckmann, Martin Smuda, Muhammed Toprak, Maraike Vandergrift, and Prof. Matthew Zahr, for which I am very grateful.

During my time at the Chair of Fluid Dynamics I was also gifted with the possibility to supervise the master thesis of Taha Furkan Benli and to supervise Dipen Amipara as a student-worker. They both assisted me in code-development and I would like to thank them.

Finally, this journey would not have been possible without the support of my friends and family, to whom I want to dedicate this dissertation. In particular, I am deeply grateful to my wife for her unconditional support, her advice, and her open ears during all the ups and downs of my PhD journey. I would also like to thank my parents for their financial support during my student years, enabling me to focus on my studies. Lastly, I wish to express gratitude towards my mother and my brother for their never-ending belief in me.

Contents

List of Tables

List of Figures

List of Abbreviations

1D one-dimensional

2D two-dimensional

3D three-dimensional

AV artificial viscosity

BILU block incomplete LU factorization

BoSSS bounded support spectral solver

CAA computational aeroacoustics

CFD computational fluid dynamics

CG continuous Galerkin

CNS compressible Navier-Stokes

DG discontinuous Galerkin

DNS direct numerical simulation

DOF degree of freedom

DOFs degrees of freedom

EOC experimental order of convergence

FDM finite difference method

FEM finite element method

FVM finite volume method

HiOCFD5 5th International Workshop on High Order CFD Methods

HLL Harten-Lax-van Leer

HLLC Harten-Lax-van Leer-Compact

HOIST high-order implicit shock tracking

HPC high performance computing

IBM immersed boundary method

ILU incomplete LU factorization

IST implicit shock tracking

KKT Karush-Kuhn-Tucker

LIA linearized interaction analysis

MDF minimum discarded fill

MDG-ICE moving discontinuous Galerkin method with interface condition enforcement

PDE partial differential equation

SD sub-domain

SPD symmetric positive definite

SQP sequential quadratic programming

TVD total variation diminishing

WENO weighted essentially non-oscillatory

XDG extended discontinuous Galerkin

XDG-IST implicit XDG shock tracking

xFEM extended finite element method

List of Symbols

α	Step length
α_{min}	Minimal step length
a	Speed of sound
a_J	Average jump function
$\boldsymbol{A}$	Generic system matrix of linear system
$\tilde{\boldsymbol{A}}$	Generic preconditioner for system matrix
$\tilde{\boldsymbol{A}}_C$	Constrained preconditioner
$\tilde{\boldsymbol{A}}_{\text{AT}}$	Block anti-triangular constrained preconditioner
$\boldsymbol{A}^{(0)}$	Coarse level matrix (P-multigrid)
$\tilde{\boldsymbol{A}}_0$	Preconditioner with $\tilde{\boldsymbol{B}}_{\boldsymbol{yy}} = \boldsymbol{B}_{\boldsymbol{yy}}$ and $\tilde{\boldsymbol{J}}_{\boldsymbol{u}} = \boldsymbol{J}_{\boldsymbol{u}}$
$\tilde{\boldsymbol{A}}_{\text{BJ}}$	Preconditioner with $\tilde{\boldsymbol{B}}_{\boldsymbol{yy}} = \text{diag}(\boldsymbol{B}_{\boldsymbol{yy}})$ and $\tilde{\boldsymbol{J}}_{\boldsymbol{u}} = \tilde{\boldsymbol{J}}_{\text{BJ}}$
$\tilde{\boldsymbol{A}}_{\text{BILU}}$	Preconditioner with $\tilde{\boldsymbol{B}}_{\boldsymbol{yy}} = \text{diag}(\boldsymbol{B}_{\boldsymbol{yy}})$ and $\tilde{\boldsymbol{J}}_{\boldsymbol{u}} = \tilde{\boldsymbol{J}}_{\text{BILU}}$
$\tilde{\boldsymbol{A}}_{\text{BJ + ilu}(\boldsymbol{B}_{\boldsymbol{yy}})}$	Preconditioner with $\tilde{\boldsymbol{B}}_{\boldsymbol{yy}} = \text{ilu}(\boldsymbol{B}_{\boldsymbol{yy}})$ and $\tilde{\boldsymbol{J}}_{\boldsymbol{u}} = \tilde{\boldsymbol{J}}_{\text{BJ}}$
$\tilde{\boldsymbol{A}}_{\text{BILU + ilu}(\boldsymbol{B}_{\boldsymbol{yy}})}$	Preconditioner with $\tilde{\boldsymbol{B}}_{\boldsymbol{yy}} = \text{ilu}(\boldsymbol{B}_{\boldsymbol{yy}})$ and $\tilde{\boldsymbol{J}}_{\boldsymbol{u}} = \tilde{\boldsymbol{J}}_{\text{BILU}}$
$\tilde{\boldsymbol{A}}_{\text{0p0}}$	Preconditioner with $\tilde{\boldsymbol{B}}_{\boldsymbol{yy}} = \boldsymbol{B}_{\boldsymbol{yy}}$, $\tilde{\boldsymbol{J}}_{\boldsymbol{u}} = \boldsymbol{J}_{\boldsymbol{u}}$ and P-multigrid
$\tilde{\boldsymbol{A}}_{\text{BJp0}}$	Preconditioner with $\tilde{\boldsymbol{B}}_{\boldsymbol{yy}} = \text{diag}(\boldsymbol{B}_{\boldsymbol{yy}})$, $\tilde{\boldsymbol{J}}_{\boldsymbol{u}} = \tilde{\boldsymbol{J}}_{\text{BJ}}$ and P-multigrid
$\tilde{\boldsymbol{A}}_{\text{BILUp0}}$	Preconditioner with $\tilde{\boldsymbol{B}}_{\boldsymbol{yy}} = \text{diag}(\boldsymbol{B}_{\boldsymbol{yy}})$, $\tilde{\boldsymbol{J}}_{\boldsymbol{u}} = \tilde{\boldsymbol{J}}_{\text{BILU}}$ and P-multigrid
$\mathfrak{A}$	Sub-domain with $\varphi_s < 0$ and $\varphi_b < 0$
$\mathcal{A}$	Aggregation map
$\text{amp}_{\text{ex}}^{+}$	Pressure ampflification coefficient (fast acoustic wave)
$\text{amp}_{\text{ex}}^{-}$	Pressure reducation coefficient (reflected slow acoustic wave)
$\text{agg}_{\text{thrsh}}$	Aggregation threshold
$\text{arf}_k^{\boldsymbol{c}}$	Averaged reduction factor
B	Jacobian of the projected physical flux
B_x	Jacobian of the projected physical flux for space-only Euler equations
$\tilde{B}$	Linearization of B
$\tilde{\boldsymbol{B}}$	Approximation to $\boldsymbol{B}$
$\boldsymbol{b}$	Generic right-hand side of linear system
$\boldsymbol{b}^{(0)}$	Coarse level right-hand side (P-multigrid)
$\boldsymbol{B}_{\boldsymbol{uu}}$	Sub-block of Hessian approximation

$\boldsymbol{B}_{\boldsymbol{yy}}$	Sub-block of Hessian approximation
$\tilde{\boldsymbol{B}}_{\boldsymbol{yy}}$	Approximation to $\boldsymbol{B}_{\boldsymbol{yy}}$
$\boldsymbol{B}$	Hessian approximation
$\boldsymbol{B}_{\boldsymbol{u}\boldsymbol{\varphi}}$	Sub-block of Hessian approximation
$\boldsymbol{B}_{\boldsymbol{\varphi\varphi}}$	Sub-block of Hessian approximation
$\boldsymbol{B}_{\boldsymbol{uy}}$	Sub-block of Hessian approximation
$\mathfrak{B}$	Sub-domain with $\varphi_s > 0$ and $\varphi_b < 0$
β	Line search dampening factor

c_{smth}	Smoothing factor
c_{cmp}	Isothermal compressibility
$c_{\text{cmp}}^{\text{wtr}}$	Isothermal compressibility for water
$c_{\text{cmp}}^{\text{air}}$	Isothermal compressibility for air
$\vec{c}_1$	Center of circle (bow shock geometry)
$\vec{c}_2$	Center of circle (bow shock geometry)
$C^l(\Omega_0)$	Space of l-times continuously differentiable functions on Ω_0
$\tilde{\boldsymbol{C}}$	Matrix product $-\tilde{\boldsymbol{J}}_{\boldsymbol{u}}^{-1}\boldsymbol{J}_{\boldsymbol{y}}$
$\mathfrak{C}$	Sub-domain with $\varphi_s < 0$ and $\varphi_b > 0$

Ω	Physical domain of interest, computational domain
d	Dimension of domain of interest
Ω_x	Space-only domain
Ω_0	Refenence domain
$\partial\Omega$	Domain boundary
$\partial\Omega_L$	Left domain boundary
$\partial\Omega_R$	Right domain boundary
$\partial\Omega_B$	Bottom domain boundary
$\partial\Omega_T$	Top domain boundary
$\mathfrak{D}$	Sub-domain with $\varphi_s > 0$ and $\varphi_b > 0$
$\text{diag}(\cdot)$	Diagonal matrix of an input matrix

e_{vol}	Specific volume
e_{in}	Specific inner energy
e_{kin}	Specific kinetic energy
E	Specific total energy
ϵ_1	Reinitialization parameter
ϵ_2	Reinitialization parameter
ϵ_3	Reinitialization parameter
ϵ_4	Reinitialization parameter
err^+	Average deviation of pressure ampflification coefficient (fast acoustic wave)

f	Objective function
f_{ER}	Full enriched residual objective function
f_{NB}	Near-band-only enriched residual objective function
f_{RH}	Rankine-Hugoniot objective function

f_{err}	Error component of objective function
f_{msh}	Mesh-quality component of objective function
F	Flux vector
F_x	Spcace-only component of flux vector
$\bar{F}$	Transformed flux
$\boldsymbol{F}$	Residual function defining objective function f

g	Deformation gradient
$\boldsymbol{g}$	Jacobian of objective function (HOIST)
$\mathcal{G}$	Domain mapping
G	Mapping Jacobian
$\mathbb{G}$	Diffeomorphisms between physical and reference domain
$\hat{\gamma}$	Heat capacity ratio
γ	Regularization parameter
γ_{min}	Minimal value for regularization parameter
γ_{max}	Maximal value for regularization parameter
$\gamma^{\star}$	Updated regularization parameter
γ_{red}	Adaptation factor for regularization parameter
Γ	Edges
Γ_{int}	Internal edges
Γ_{ext}	Outer edges

H_{err}	Enthalpy error
H_{smth}	Smooth Heaviside function
H	Enthalpy
$\boldsymbol{H}_{\mathcal{L}}$	Hessian of Lagrange functional
$\boldsymbol{H}_{\mathcal{L}}^{\boldsymbol{u}\boldsymbol{\varphi}}$	$(\boldsymbol{u}-\boldsymbol{\varphi})$-Part of Hessian of Lagrange functional
$\boldsymbol{H}_f$	Hessian of objective function
$\bar{\mathcal{H}}$	Transformed numerical flux
$\mathcal{H}$	Numerical flux
$\mathcal{H}^{\text{cntrl}}$	Central numerical flux
$\mathcal{H}^{\text{up}}$	Upwind numerical flux
$\mathcal{H}^{\text{up}}_{\text{smth}}$	Smooth upwind numerical flux
$\mathcal{H}^{\text{riem}}$	Godunov's numerical flux
$\mathcal{H}^{\text{HLLC}}$	Harten-Lax-van Leer-Compact (HLLC) numerical flux
$\mathcal{H}^{\text{roe}}$	Roe numerical flux

$\text{ilu}(\cdot)$	Point-ILU of a matrix
$\boldsymbol{I}$	Identity matrix
$\boldsymbol{I}_{d'}$	Identity matrix of dimension d'
$\mathfrak{I}_s$	Interface of shock level set φ_s
$\mathfrak{I}_b$	Interface of boundary level set φ_b
$\mathfrak{I}$	Interface

$\jmath_{\text{smth}}$	Smooth jump function
$\boldsymbol{J}_{\boldsymbol{u}}$	Jacobian of residual function with respect to $\boldsymbol{u}$

$\tilde{\boldsymbol{J}}_{\boldsymbol{u}}$	Approximation to $\boldsymbol{J}_{\boldsymbol{u}}$
$\tilde{\boldsymbol{J}}_{\text{BJ}}$	Block-Jacobi preconditioner for $\boldsymbol{J}_{\boldsymbol{u}}$
$\tilde{\boldsymbol{J}}_{\text{BILU}}$	Block-ILU preconditioner for $\boldsymbol{J}_{\boldsymbol{u}}$
$\boldsymbol{J}$	Jacobian of residual function
$\boldsymbol{J}_{\boldsymbol{y}}$	Jacobian of residual function with respect to $\boldsymbol{y}$
$\tilde{\boldsymbol{J}}$	Approximation to $\boldsymbol{J}$

κ	Mesh penalty parameter
$\hat{k}$	Wavenumber
k_{term}	Parameter for termination
k_{min}^{P}	Minimum iteration threshhold for polynomial degree P
$\mathcal{K}_h^{X,\text{agg}}$	Aggregation mesh
$\mathcal{K}_{\text{ReInit}}$	Mesh cells to be reinitialized
$\mathcal{K}_h^{s}$	Sub-grid for shock level set
$\mathcal{K}_h$	Numerical mesh
$\tilde{\mathcal{K}}_h$	Deformed numerical mesh
$\mathcal{K}_h^{X}$	Numerical cut-cell mesh
$\mathcal{K}_h^{cc,0}$	Set of cut-cells for a level set
$\mathcal{K}_h^{cc,1}$	Near band set for a level set

l_{W}	Wave length in the context of single-period waves
$L^2(\Omega_0)$	Space of L^2-integrable functions on Ω_0
L	Length scale for regularization parameter adaptation
$\tilde{\boldsymbol{L}}$	Lower part of block-ILU decomposition ($\tilde{\boldsymbol{J}}_{\text{BILU}}$)
$\boldsymbol{\lambda}$	Lagrange multiplier
$\boldsymbol{\lambda}^*$	First-order optimality solution (Lagrange multiplier)
$\Delta\boldsymbol{\lambda}$	Optimization step (Lagrange multiplier)
Λ_x	Eigenvalue matrix of B_x
$\mathcal{L}$	Lagrange functional

m	Number of conservation laws
M	Merit function
M^{l_1}	l_1-Merit function
M^{l_2}	l_2-Merit function
$\boldsymbol{M}$	Mass matrix
μ	Weighting factor for line search
Ma	Mach number

n_x	x-Component of normal vector in physical domain
n_y	y-Component of normal vector in physical domain
$\vec{n}$	Normal vector
$\vec{n}_x$	Space-only component of space-time normal vector
n_t	Time component of space-time normal vector
$\vec{n}_\Gamma$	Edge normal field
$\vec{N}$	Normal vector in reference domain
N_K	Number of cells

N_v	Number of mesh nodes
$N_{\boldsymbol{x}}$	Number of concatenated mesh nodes
$N_{\boldsymbol{y}}$	Number of concatenated mesh nodes (parameterized mesh)
$N_{\boldsymbol{F}}$	Dimension of $\boldsymbol{F}$
$N_{\boldsymbol{z}}$	Dimension of $\boldsymbol{z}$
$N_{\boldsymbol{u}}$	Dimension of $\boldsymbol{u}$
$N_{\boldsymbol{\varphi}}$	Dimension of $\boldsymbol{\varphi}$
N_S	Number of spline interpolation points
N_P	Dimension of $(\mathcal{P}_P(K_e))^m$
$\hat{N}_P^e$	Dimension of $(\mathcal{P}_P(\mathcal{N}_e))^m$
$\mathcal{N}_e$	Set of neighoring elements of mesh-element K_e
$\tilde{\mathcal{N}}$	Shock-aware neighbor set

σ_2	Lower threshold for regularization parameter adaptation
σ_1	Upper threshold for regularization parameter adaptation
$\sigma_{\nu,+}$	Amplification parameter for time-step adaptation
$\sigma_{\nu,-}$	Reduction parameter for time-step adaptation

$\boldsymbol{\phi}$	Parametrization for the physical nodes
p	Pressure
P	Total polynomial degree of the DG/XDG flow solution
P_c	Total polynomial degree of the continous Galerkin space
P_s	Polynomial degree of shock level set φ_s
P_b	Polynomial degree of boundary level set φ_b
P'	Total polynomial degree of the test space
P_{end}	Final polynomial degree of the DG/XDG flow solution
$\boldsymbol{P}$	Full prolongation operator (P-multigrid)
$\boldsymbol{P}_e$	Selection matrix for $\boldsymbol{u}$-degrees of freedom (DOFs) (element K_e)
$\hat{\boldsymbol{P}}_e$	Selection matrix for $\boldsymbol{u}$-DOFs (nieghbors element K_e)
$\tilde{\boldsymbol{P}}$	MDF reordering for $\tilde{\boldsymbol{J}}_{\text{BILU}}$
$\boldsymbol{P}_{\boldsymbol{u}}$	$\boldsymbol{u}$-Component of prolongation operator $\boldsymbol{P}$
$\boldsymbol{P}_{\boldsymbol{y}}$	$\boldsymbol{y}$-Component of prolongation operator $\boldsymbol{P}$
$\mathcal{P}_P(K)$	Space of polynomial functions (on element K, degree P)
φ_s	Level set representing discontinuity
φ_b	Level set representing immersed boundary
φ_s^{DG}	DG shock level set
φ_s^{CG}	Continuous Galerkin (CG) shock level set
φ_s^{Gl}	Global shock level set
φ_s^{Sp}	Spline shock level set
$\boldsymbol{\varphi}$	Coefficients of shock level set φ_s
$\boldsymbol{\varphi}^*$	First-order optimality solution (level set)

$\boldsymbol{\varphi}^{\text{DG}}$	Coefficient vector of DG shock level set
$\boldsymbol{\varphi}^{\text{CG}}$	Coefficient vector of CG shock level set
$\boldsymbol{\varphi}^{\text{Sp}}$	Coefficient vector of spline shock level set
$\Delta\boldsymbol{\varphi}$	Optimization step (level set)
$\boldsymbol{\Phi}$	Basis of $\mathcal{V}_h^{P,m,X}$
$\boldsymbol{\Phi}^{\text{CG}}$	Basis vector for CG shock level set
$\boldsymbol{\Phi}^{\text{Gl}}$	Basis for global shock level set
proj_{P-1}	L^2-projection on XDG space of order $P-1$
$(\cdot)_L$	Value left to the shock
$(\cdot)_R$	Value right to the shock
$\lvert \cdot \rvert_{\text{smth}}$	Smooth absolute value
$(\cdot)_\infty$	Inflow/Free stream value
$(\cdot)_{\text{ref}}$	Reference value
$(\cdot)^\star$	Non-dimensional quantity
$\llbracket\cdot\rrbracket$	Jump operator
$(\cdot)^\circ$	Constant base flow
$(\cdot)'$	Perturbation of flow
$(\cdot)^{\text{in}}$	Inner trace of DG field
$(\cdot)^{\text{out}}$	Outer trace of DG field
$[\cdot]$	Index set for natural numbers
$(\underline{\cdot})$	Roe average
$(\cdot)^{\text{agg}}$	Agglomerated object
q^{in}	Correction factor for minimum eigenvalue wave
q^{out}	Correction factor for maximum eigenvalue wave
$\boldsymbol{Q}_e$	Selection matrix for $\boldsymbol{x}$-DOFs (element K_e)
$\boldsymbol{Q}$	Full restriction operator (P-multigrid)
$\boldsymbol{Q}_{\boldsymbol{y}}$	$\boldsymbol{y}$-Component of restriction operator $\boldsymbol{P}$
$\frac{D}{Dt}$	Material derivative in time
∇	Gradient
∇_x	Space-only component of gradient
∂_t	Time component of gradient, parital derivative in time
$\bar{\nabla}$	Transformed gradient on reference domain
$\nabla_{(\Delta\boldsymbol{z}_k)}$	Directional derivative in direction $\Delta\boldsymbol{z}_k$
$r_h^{P',P}$	DG/XDG residual form (P' test degree,P trial degree)
$r_K^{P',P}$	DG/XDG residual form on mesh element K
$\boldsymbol{r}^0$	XDG residual $(P=0)$
$\boldsymbol{r}$	Algebraic DG/XDG residual
$\boldsymbol{r}_e$	Elemental, algebraic residual (element K_e)
$\boldsymbol{R}$	Enriched algebraic DG/XDG residual
$\boldsymbol{R}_{\text{NB}}$	Near-band-only enriched residual
$\boldsymbol{R}_{\text{RH}}$	Near-band-only enriched residual
$\boldsymbol{R}_e$	Elemental, algebraic enriched residual (element K_e)
$\boldsymbol{R}_{\text{msh}}$	Elemental mesh-distortion
$\mathbb{R}_+$	Positive real numbers
$\mathbb{R}$	Real numbers

ρ	Density
$\rho'^{\text{ ent.}}$	Entropy wave perturbation of density
$\rho'^{\text{ ac.}}$	Acoustic wave perturbation of density

$\hat{s}$	Entropy
$\boldsymbol{s}$	Generic solution of linear system
$\boldsymbol{s}_{\text{ex}}$	Exact solution of linear system
$\tilde{\boldsymbol{s}}^{(0)}$	Coarse level solution (P-multigrid)
$\tilde{\boldsymbol{s}}$	Fine level solution (P-multigrid)
S^{in}	Speed of minimum eigenvalue wave
S^{out}	Speed of maximum eigenvalue wave
$S_\star$	Speed of contact wave
S	Spline of spline shock level set
$\mathfrak{s}$	Sub-domain
SD	Set of all sub-domains
sen	Shock sensor
$\text{sr}_k^{\boldsymbol{c}}$	Residual-norm skyline

t_{start}	Start time of $\mathcal{T}$
t	Time
t_{end}	End time of $\mathcal{T}$
$\tilde{t}$	Pseudo time
$\Delta\tilde{t}$	Pseudo time-step size
T	Temperature
τ	Adaptation factor for step length
θ_{wdg}	Angle of inclined plane
θ_{shk}	Angle of straight-sided shock wave (inclined plane)
$\theta_{\text{wdg,max}}$	Critical angle of inclined plane
$\theta_{\text{wdg,weak}}$	Angle of weak shock (inclined plane)
$\mathcal{T}$	Time interval of interest
tol	Tolerance

u_a	Advection field
u	Velocity in x-direction
$\vec{u}$	Velocity vector
u_w	Wave speed
$u_\perp$	Flow direction
u_{dr}	Dirichlet boundary condition for veloctiy
u_n	Velocity in normal direction
$\vec{u}^{\text{mir}}$	Mirrored velocity
u_{ex}	Exact solution to Burgers/advection equation
$\bar{U}$	Transformed conserved variables
U	Conserved variables
$\bar{U}$	Flow solution on the reference domain
$\bar{U}_h$	Transformed XDG approximation of conserved variables
$U_\perp$	Flow direction, evaluated at average of U^{in} and U^{out}

U^{ReInit}	Reinitialized XDG solution
$\boldsymbol{u}$	Coefficient vector of XDG/DG flow solution $\bar{U}$
$\boldsymbol{u}^*$	First-order optimality solution (flow)
$\tilde{\boldsymbol{u}}$	XDG flow solution of polynomial degree $P = 0$
$\Delta\tilde{\boldsymbol{u}}$	Time-step of flow solution (degree 0)
$\boldsymbol{u}_e$	$\boldsymbol{u}$-DOFs corresponding to mesh-element K_e
$\hat{\boldsymbol{u}}_e$	$\boldsymbol{u}$-DOFs corresponding to neighboring elements of mesh-element K_e
$\tilde{\boldsymbol{U}}$	Upper part of block-ILU decomposition ($\tilde{\boldsymbol{J}}_{\text{BILU}}$)
v	Velocity in y-direction
$\boldsymbol{v}$	Generic vector
$\boldsymbol{V}$	Eigenvalue matrix of B
$\boldsymbol{V}_x$	Right eigenvector matrix of B_x
$\mathcal{V}_h^{P,m,X}$	XDG function space (degree P, m components)
$\mathcal{V}_h^{P,m}$	DG function space (degree P, m components)
ν_{min}	Parameter for time-step adaptation
ν_{max}	Parameter for time-step adaptation
$\mathcal{W}_h^{P_c}$	CG space with degree P_c
ω	Oscillation frequency
x	Coordinate in space
x_{start}	Minial x value (Cartesian grid)
x_{end}	Maximal x value (Cartesian grid)
x_s	Shock position
$\tilde{x}$	Characteristic curve
$\hat{x}$	Mesh node in physical domain
$\vec{x}_{\text{min}}$	Candidate points for reconstructing the shock front
$\vec{x}^{\text{CG}}$	Nodes of CG shock level set
x_{W}	Wave position
$\hat{X}$	Transformed mesh node in reference domain
$\vec{X}$	Point in reference domain
$\boldsymbol{x}$	Concatenated mesh nodes
$\boldsymbol{x}_e$	Concatenated mesh nodes (element K_e)
$\hat{\boldsymbol{x}}_e$	Concatenated mesh nodes (neighbors of element K_e)
χ_s	Points reconstructing the shock front
χ_{seed}	Seeding points for reconstructing the shock front
χ_{cand}	Candidate points for reconstructing the shock front
y	Coordinate in space
$\boldsymbol{y}^*$	First-order optimality solution (parameterized mesh)
$\boldsymbol{y}$	Concatenated mesh nodes (parameterized mesh)
$\boldsymbol{z}$	Concatenation of $\boldsymbol{u}$ and $\boldsymbol{\varphi}/\boldsymbol{y}$
$\boldsymbol{z}^*$	First-order optimality solution (flow and level set/mesh)

Δz	Optimization step (flow and level set/mesh)

1 Introduction

1.1 Background and motivation

Commercial and affordable air travel is one of the major technological success stories of our modern world. The ability to explore different cultures around the globe connects humanity and enhances the exchange of ideas. Whether it is a flight to an academic conference to learn about research first-hand or a personal trip that brings new perspectives on life, aviation fosters the necessary cultural and technological development of humanity.

One long-time dream in commercial flight is increasing speed and reducing travel time between distant locations. Currently, commercial aircraft travel at subsonic speeds, that is, speeds below the speed of sound, due to regulatory barriers and high operating costs. One major challenge for supersonic (faster-than-sound) air travel is dealing with so-called shock waves.

The Concorde, a commercial plane that flew at 1,350 miles per hour (roughly twice the speed of sound), was retired in 2003. The main reason besides increased fuel consumption was the thunderous sonic boom it produced. This boom was so loud that many countries restricted aircraft to subsonic speeds over land (Benson, 2013), limiting the routes these planes could take. The sonic boom results from an abrupt pressure change caused by shock waves spreading away from the plane, reaching the ground, and passing through every listening ear.

To make supersonic air travel feasible again, new technological improvements are needed to reduce fuel consumption and take the 'bang out of the boom' (Council et al., 2002). This involves designing aircraft that produce a different shape of sonic boom with reduced intensity. A major building block needed for these improvements is the numerical analysis of compressible flows with shocks, which enables engineers to understand the noise associated with supersonic travel and to test new aircraft designs.

Compressible flows with shocks In the research area of fluid dynamics, flows are categorized into two main types based on their velocity relative to the speed of sound: incompressible and compressible. The Mach number, a critical dimensionless parameter, represents the ratio of the flow's speed to the speed of sound in that fluid. For flows where the Mach number is below 0.3, the assumption is that they are incompressible, i.e., they have a constant density. This assumption is justified because their density variation remains under 5%, and the effects of compressibility on the flow are minimal, thus making them negligible for practical purposes (Anderson, 2003). In contrast, when the Mach number exceeds 0.3, the flow enters the compressible regime. Compressible flows are present in a vast array of applications, such as in aircraft design (Raymer, 2012), the aerodynamics of airfoils (Kral, 1998), the dynamics within turbine stages of jet engines (Klapdor, 2011), natural gas pipelines (Menon and Menon, 2013), and the air movement in solar power plant chimneys (Schlaich et al., 2005). These applications span several industries, showing the importance of understanding and predicting the behavior of compressible flows.

Generally, compressible flows are classified by aerodynamicists into specific ranges of inflow Mach number Ma_∞, including subsonic ($\mathrm{Ma}_\infty < 0.8$), transsonic ($0.8 < \mathrm{Ma}_\infty < 1.2$), supersonic ($1.2 < \mathrm{Ma}_\infty < 5.0$), and hypersonic ($5.0 < \mathrm{Ma}_\infty$) flows (Anderson, 2003). Each of these flows may exhibit different physical phenomena. One of the most challenging phenomena in compressible flows is the formation of shock waves, which are sudden, drastic alterations in fluid properties such as density, pressure, and temperature, occurring within a very thin layer approximately 100 nanometers thick for air under standard conditions. Shock waves are present in all named flow categories except for subsonic flows, and predicting their exact shape and position is complex, making their study a significant area of research. For studying compressible flows with shocks, computational fluid dynamics (CFD) plays a crucial role by simulating and analyzing the complex behaviors and interactions of shock waves with various aircraft designs.

Computational fluid dynamics Over the past decades, CFD has emerged as an indispensable tool in both research and engineering, amplified by advancements and broader access to computational resources. CFD allows for the acquisition of data in areas where wind tunnel experiments are challenging or where theoretical methods are limited, offering faster results and lower costs. Among mathematical models employed in these simulations, the Euler equations are mainly used to describe compressible flows in which the viscous effects are negligible leading to a set of first-order nonlinear partial differential equations (PDEs). The task of discretizing these nonlinear PDEs has long been a focus of research with so-called low-order methods being the predominant techniques for many years. Recently, high-order methods have gained increased attention within the CFD community due to their promise of being both more accurate and computationally efficient than low-order methods (Wang et al., 2013).

High-order methods Numerical methods can be categorized by their discretization error, which is proportional to h^{P+1}, with h representing the grid's characteristic length scale and $P+1$ the order of the method. For some numerical methods P corresponds to the degree of polynomial basis functions used to approximate the flow solution and methods of order $P > 1$ are widely considered *high-order* within the CFD community (Wang et al., 2013). The advantage of high-order methods is that they typically introduce less numerical dissipation and, thus, have the potential to obtain more accurate solutions at lower total degrees of freedom (DOFs) (and computational time) than low-order methods. Additionally, high-order solutions can be achieved on coarse grids.

In many applications, the superior accuracy of high-order methods is highly sought after: In the field of computational aeroacoustics (CAA), it is crucial that broadband acoustic waves can travel long distances, without additional artificial dissipation or dispersion introduced by the numerical scheme. The CAA community largely favors high-order numerical methods for their exceptional precision and effectiveness (Wagner et al., 2007). Additionally, high-order techniques are vital for the precise resolution of unsteady vortices within vortex-dominated flows, significantly influencing the aerodynamic efficiency of aircraft (Wang et al., 2013).

Despite these advantages, low-order methods continue to dominate industrial CFD applications. This prevalence is due to their superior robustness and faster convergence to steady-state conditions, attributed to higher stabilizing numerical dissipation, and the lack of readily available robust high-order mesh generators (Wang et al., 2013). Particularly, scenarios featuring discontinuous flow features, such as shock waves in high-speed flows, pose significant challenges for high-order methods. These challenges manifest as oscillating approximations within cells containing discontinuities, known as the Gibbs phenomenon, and lead to a loss of global high-order convergence. In many cases, due to its minimal numerical diffusion which would dampen the oscillations, the high-order method fails to converge at all, leading to simulation failures. However, initiatives and developments like the European ADIMGA project (Kroll, 2010), the IDIHOM project (Kroll et al., 2015), and the FLEXI solver framework (Krais et al., 2021) have demonstrated the potential of high-order methods for compressible flow simulations, offering evidence that they can surpass the computational performance of traditional low-order approaches. These findings suggest a promising path for further exploration and adoption of high-order methods in complex fluid dynamics applications.

Discontinuous Galerkin methods In recent decades, a class of high-order methods named discontinuous Galerkin (DG) methods have surged in popularity for discretizing PDEs in the field of CFD. Known for their localized nature, applicability to arbitrary geometries on unstructured grids, and efficient parallelization capabilities, DG methods are very well suited for high performance computing (HPC) applications. They can be regarded as a generalization of the popular finite volume method (FVM), employing cell-local polynomial basis functions of arbitrary order P. By introducing numerical fluxes at the interfaces between elements, DG methods not only ensure the conservation of key physical quantities but also preserve the directionality of information propagation across the computational domain. This methodology not only facilitates compact discretization stencils, beneficial for parallelization, but also offers remarkable flexibility, particularly suitable for multi-domain and multi-physics simulations. DG methods can operate on arbitrary unstructured grids, avoiding the limitations imposed by structured Cartesian grids. Furthermore, hp-adaptivity enables local grid refinement and the variation of polynomial basis functions, providing enhanced accuracy without global continuity constraints. Notably, the highly local character of DG methods yields easily invertible block-diagonal mass matrices, contrasting with the computationally expensive global matrices often associated with finite element methods (FEMs). The method's compatibility with explicit time-integration schemes further enhances its parallelizability.

DG methods trace back over 50 years, with an extensive overview provided by Di Pietro and Ern (2012). The first attempt to approximate first-order PDEs with DG methods was conducted by Reed and Hill (1973) with their method for steady neutron transport, which was followed by an analysis by Lasaint and Raviart (1974). Subsequent advancements improved error estimates, especially for smooth solutions (Johnson and Pitkäranta, 1986). The method's applicability broadened as it was extended to approximate three-dimensional boundary-layer equations for incompressible fluid flows (Caussignac and Touzan, 1990). Concurrently, DG methods were expanded to time-dependent hyperbolic PDEs (Chavent and Cockburn, 1989), further refined using explicit Runge–Kutta schemes (Cockburn and Shu, 1991).

Over time, numerous contributions have been made applying DG methods to compressible flow scenarios, naming only a fraction (Cockburn, 1998; Fidkowski et al., 2005; Persson and

Peraire, 2006; Hartmann and Houston, 2008; Kroll, 2010; Kroll et al., 2015; Krais et al., 2021; Geisenhofer et al., 2019). In the context of compressible flows with shocks, many strategies have been devised to stabilize the DG scheme in the vicinity of shock waves, mainly categorized as *shock capturing* and *shock fitting/tracking* techniques, whereby a distinction is made between explicit and implicit shock fitting/tracking methods.

Shock capturing Shock capturing strategies, designed to manage discontinuities and mitigate oscillations in numerical solutions, employ a variety of techniques. These include limiters for achieving total variation diminishing (TVD) schemes (Cockburn et al., 1989), high-order reconstruction methods such as weighted essentially non-oscillatory (WENO) schemes (Shu and Osher, 1988; Harten et al., 1997), the incorporation of artificial viscosity (Persson and Peraire, 2006; Barter and Darmofal, 2010; Ching et al., 2019), and the local usage of low-order techniques (Beck et al., 2020; Mossier et al., 2022). In these approaches, the numerical discretization is tailored locally to handle discontinuities, especially in cells where solution oscillations are detected. For example, with artificial viscosity approaches, a second-order term is added locally to the equations at hand, leading to a smooth shock profile (Persson and Peraire, 2006).

Despite their practical effectiveness and relative ease of implementation, these approaches are not without limitations. A key issue with existing shock capturing methods is their dependence on the alignment of the mesh with shock waves. In simple scenarios, structured meshes can be intentionally crafted to align with shocks, significantly reducing numerical errors associated with shock capturing (Barter and Darmofal, 2010). However, when dealing with unstructured meshes that are not aligned with shocks, numerical inaccuracies in intense shock regions can be substantial. These errors can induce nonphysical fluctuations that propagate downstream towards the boundary layer, adversely affecting accuracy in other parts of the domain. Thus, shock capturing schemes can compromise the high-order convergence characteristics of the original method, highlighting a trade-off between implementation ease and maintaining high-resolution fidelity in complex flow situations.

Explicit shock tracking While shock capturing schemes incorporate sufficient numerical diffusion to stabilize solutions near shocks, explicit shock tracking/fitting schemes strive to overcome this by explicitly adjusting the computational mesh. This adjustment aims to satisfy the Rankine-Hugoniot jump conditions (Rankine, 1870; Hugoniot, 1887) across sharp shock fronts, by aligning the computational mesh's cell edges with the shock front. The cell's edges are treated as internal boundaries and jumps in numerical solutions between elements are leveraged by shock tracking methods. As a result, discontinuities can be accurately represented without further stabilization, making shock-fitting particularly attractive for direct numerical simulation (DNS) focused on capturing all relevant flow details, excluding the shock layer itself (Salas, 2010).

Historically, explicit shock tracking dates back to the 1940s. Early methodologies involved formulating a finite difference method (FDM) for the Euler equations on a static, rectangular reference domain, which is then mapped onto a time-varying physical domain. Starting with an initial shock location estimate, a time-marching procedure adjusts the reference domain to align with the shock position (Salas, 2010). Recently, such a method was applied to detonation problems, showcasing high-order convergence rates (Romick and Aslam, 2017). To mitigate the need for in-depth prior knowledge of the flow, ’floating’ shock tracking methods have been developed, which simplify the management of internal shock boundaries. These methods track the movement of shock intersections with the background mesh (Moretti and Valorani, 1988; Nasuti and Onofri, 1996). Another strategy still under current research (Assonitis et al., 2022; Assonitis et al., 2023) and sometimes also termed floating shock-fitting adapts the mesh directly to the shock geometry using the Rankine-Hugoniot conditions. Techniques like those involve shock point tracking and local computational grid re-meshing and are used to align mesh edges with the shock surface, while shock capturing addresses yet-to-be-fitted discontinuities (Paciorri and Bonfiglioli, 2009; Bonfiglioli et al., 2016).

However, the shock wave’s position is generally unknown and can change or develop over time, making meshing a challenging task. Furthermore, complex patterns can form due to interactions of different shock waves or reflections. Even though explicit shock tracking approaches showed promising results, they seem to be not easily generalizable for problems where the shock position and its topology are unknown. Also, early approaches focused on low-order schemes where shock capturing has a bigger relative advantage, making shock tracking play a minor role in the simulation of compressible flows to date (Trépanier et al., 1996; Baines et al., 2002).

Implicit shock tracking Recently, in the context of DG methods, so-called implicit shock tracking (IST) approaches have been developed and successfully applied to two-dimensional (2D) flows. These methods promise to remedy the problems associated with explicit shock tracking by not requiring any information about the shock or its topology. IST methods treat mesh coordinates as additional variables in the discretized conservation law equations and compute numerical solutions by implicitly aligning mesh edges with discontinuities. In the work by Corrigan et al. (2019c), the moving discontinuous Galerkin method with interface condition enforcement (MDG-ICE) is proposed. This approach augments the weak form of stationary conservation laws with a term enforcing the Rankine-Hugoniot jump conditions. In subsequent studies, the method was applied to three-dimensional (3D) (Corrigan et al., 2019b) and to viscous flows (Kercher et al., 2021; Ching et al., 2024), and a least squares formulation of the same method was presented by Kercher and Corrigan (2021). Furthermore, the high-order implicit shock tracking (HOIST) method was developed by Zahr and Persson (2018), where shock tracking of the mesh is achieved by solving a constrained optimization problem based on a general error indicator defined by an enriched DG discretization. This method was subsequently applied to reactive flows (Zahr and Powers, 2021), time-dependent problems were tackled (Shi et al., 2022; Naudet and Zahr, 2024), and additional robustness was investigated along with steady 3D cases (Huang and Zahr, 2022).

Even though IST methods are robust and accurate for various flow types, they have certain weaknesses. For time-dependent problems, IST methods, such as the one discussed by Shi et al. (2022), require cumbersome re-meshing when utilizing the method of lines strategy. During time evolution only certain parts of the mesh move with the shock, resulting in inadequately resolved regions upstream and excessively resolved regions downstream of the shock, especially without artificial mesh adaptation. Consequently, new developments focus on space-time strategies (Corrigan et al., 2019b; Naudet and Zahr, 2024), which present other challenges. For unsteady 3D problems, they require solving four-dimensional space-time problems. In such scenarios, mesh operations like element-splitting, fundamental to mesh-based IST methods, need to be carried out in four dimensions, which can become very inefficient (Naudet and

Zahr, 2024). To circumvent difficulties associated with mesh adaptation, so-called extended methods are used in other CFD domains (e.g., multi-phase flows) to track evolving interfaces on a fixed computational background mesh (Kummer et al., 2018; Smuda and Kummer, 2020; Rieckmann et al., 2024).

Extended discontinuous Galerkin methods The concept underlying extended methods, such as extended discontinuous Galerkin (XDG) methods, involves addressing scenarios where the solution exhibits discontinuities by enrichment of the approximation space with discontinuous basis functions and without adapting the computational grid. These additional basis functions are discontinuous at interfaces, which are usually defined explicitly or implicitly and allowed to move freely throughout the computational domain. This is especially useful for problems where solution discontinuities move in time, as extended methods circumvent the need for cumbersome re-meshing.

The first such method in the context of finite elements is presented by Mös et al. (1999). This extended finite element method (xFEM) was introduced for the simulation of crack growth in solid mechanics. In the context of DG, the first extended method traces back to the work by Bastian and Engwer (2009), which focuses on discretizing elliptic scalar model problems on complex-shaped domains. This approach involves utilizing cell-local, piecewise planar, triangular sub-cells for integrating weak forms in cut-cells. Later, this approach was applied to incompressible Navier-Stokes two-phase flows (Heimann et al., 2013). The first XDG method for incompressible steady two-phase flows, which employs a high-order approximation of the interface using a level set function alongside a quadrature technique tailored for implicitly defined domains, was introduced by Kummer (2017). The discretization strategy relies on the symmetric interior penalty method for the viscous flux, complemented by stabilization techniques to mitigate issues arising from small cut-cells, accomplished through cell agglomeration. For compressible two-phase flows the XDG method was then applied by Henneaux et al. (2020) to solve gas-liquid flows with phase transition.

Additionally, first steps towards an explicit XDG shock tracking method for supersonic compressible flows with shocks, separating the pre-shock state and the post-shock state by an implicitly defined interface, have been made by Geisenhofer et al. (2020). There, an explicit reconstruction procedure is employed to obtain a shock-aligned level set function from a shock capturing simulation. As the resulting interface is not accurate enough to stabilize the XDG method, an additional sub-cell correction algorithm is proposed for one-dimensional (1D) scenarios. However, the method has yet to be shown to generalize to 2D and 3D cases.

1.2 Research gap and objectives

Research gap As of today, no approaches that combine IST concepts with extended methods, in particular XDG methods, have been explored. Such an implicit XDG shock tracking (XDG-IST) method holds the potential to bypass the complex and often necessary re-meshing procedures needed for mesh-based IST when solving time-dependent problems (Shi et al., 2022) and could provide a generalizable solution to the inaccurate interface positions observed in the explicit XDG shock fitting approach (Geisenhofer, 2021).

Additionally, until now, research on IST has primarily been centered on its variational formulation (Corrigan et al., 2019b; Kercher and Corrigan, 2021; Kercher et al., 2021), the selection of appropriate objective and constraint functions (Zahr and Persson, 2018; Zahr and Persson, 2020), the development of robust solvers for optimization problems (Huang and Zahr, 2022), and the application of these methods across various scenarios (Zahr and Powers, 2021; Shi et al., 2022; Huang et al., 2023; Naudet and Zahr, 2024). However, minimal attention has been given to developing linear solvers for the linearized optimality system governing the search direction at each optimization step. So far, IST methods solely rely on direct sparse solvers, which are known to scale terribly for 3D problems. Efficient linear solvers are crucial for enabling IST methods to tackle large-scale problems, relevant for real-world applications.

Research objectives In view of these research gap mentioned above, the two objectives of this work are the following:

1. **Development of an implicit XDG shock tracking method**: The first objective is to establish an IST approach based on an XDG method. Unlike techniques that involve implicitly moving and deforming element edges, this method keeps the grid fixed. Level set functions will be used to track shock surfaces and implicitly adapt the approximation space, allowing for the accurate representation of arbitrarily positioned, discontinuous flow features. By doing so, we circumvent the need for re-meshing and addressing mesh operations related to ill-shaped cells (i.e., cell splitting).

2. **Development of linear solvers for implicit shock tracking**: The second objective is to create efficient linear solvers for the linearized optimality system, which determines the search direction at each optimization step of IST methods. This involves the development and testing of specialized preconditioners tailored to the complexities of the system matrix.

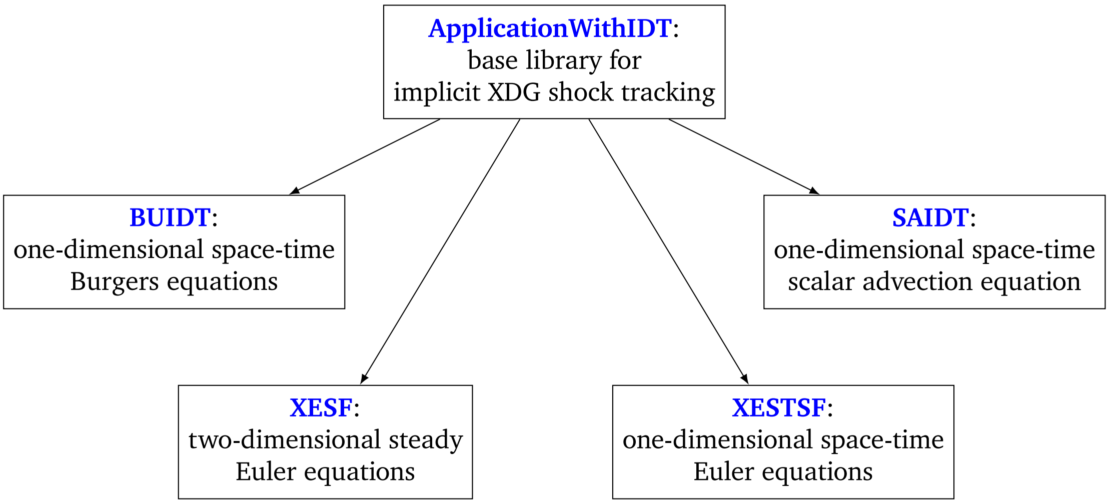

ApplicationWithIDT:
base library for
implicit XDG shock tracking
BUIDT:
one-dimensional space-time
Burgers equations
SAIDT:
one-dimensional space-time
scalar advection equation
XESF:
two-dimensional steady
Euler equations
XESTSF:
one-dimensional space-time
Euler equations

Figure 1.1: Illustrations of the implicit XDG shock tracking modules contributed to the software package BoSSS. Arrows indicate that the equation-specific modules (XESF, BUIDT, SAIDT, XESTSF) derive from the base library (ApplicationWithIDT). In this illustration the colored bold names are clickable, sending the reader to corresponding locations in the code repository.

Author's contributions In the light of the research objectives presented above the main contributions of this dissertation are:

1. **Novel implicit XDG shock tracking method**:

   (a) **Method development and validation**: In this work, we adapt the IST framework introduced by Zahr et al. (2020) for use with XDG methods, introducing the XDG-IST method. The approach revises the formulation as a constrained optimization problem, shifting from variable mesh nodes to a fixed Cartesian background grid and incorporates DOFs of a level set function as additional problem variables. Furthermore, we introduce robustness measures to our method which are tailored to the XDG context, incorporating XDG-specific stability measures and drawing insights from the work by Huang and Zahr (2022). The novel XDG-IST method is applied to several 2D problems with discontinuous solutions, including the 1D space-time Burgers, advection, and Euler equations, and the steady 2D Euler equations. The XDG-IST method is then tested using various objective functions, level set representations, and numerical fluxes. Further, we compare the XDG-IST method to a DG shock capturing method based on artificial viscosity (Müller et al., 2017; Krämer-Eis, 2017; Geisenhofer et al., 2019).

   (b) **Transparent open-source implementation**: During this research, the novel XDG-IST method has been integrated into an open-source CFD software, named bounded support spectral solver (BoSSS) (Kummer et al., 2024), contributing additional modules into the code base (see Figure 1.1). The software BoSSS serves as a platform for the development, evaluation, and application of numerical discretization schemes for PDEs in fluid dynamics, particularly focusing on XDG methods.

   Additionally, to ensure transparency and encourage reusability, this dissertation links concepts directly to their implementations in the BoSSS GitHub repository, enabling readers to easily navigate from theoretical discussions to the corresponding code segments. These links are embedded within the electronic version of the dissertation, providing direct access to the specific files and lines of code relevant to the discussed equations or test cases. Equations numbers, corresponding to equations or expressions featuring a link, are colored in blue, demonstrated by the following example

   $$x \mapsto \frac{1}{1+e^{-2s_\alpha x}}. \tag{1.1}$$

   Here, when clicking the expression (1.1) (see Remark 1), the reader is sent to a method encoding the smoothed Heaviside function. The method is defined inside a class encoding a numerical flux for the space-time advection equation.

   Furthermore, the studies presented in this work are accompanied by Jupyter notebooks, facilitating the reproduction of results. These notebooks provide the possibility to execute a series of snippets of BoSSS code inside a user-friendly graphical user interface. These code-snippets, when executed, can either run (multiple) simulations, extract simulation data from BoSSS databases, perform post-processing on the data, or create visual plots. Also, besides these code-blocks, we included markdown-blocks with descriptive text and equations. At the beginning of the presentation of a test case or a study, we provide a hyperlink to corresponding notebooks in the form of blue colored text (see links in Figure 1.1).

2. **Linear solvers for implicit shock tracking:** This dissertation introduces a set of preconditioners designed for linearized optimality systems used in sequential quadratic programming (SQP) solvers for constrained IST methods, with a particular focus on the HOIST method (Zahr et al., 2020; Huang and Zahr, 2022). These preconditioners, which are also adaptable to various objective functions and in particular to the XDG-IST method, are constructed by simplifying the constraint Jacobian using standard DG preconditioning techniques and omitting certain blocks of the Lagrangian Hessian. Additionally, a two-level P-multigrid technique is proposed which can be combined with each preconditioner. Extensive testing with two inviscid compressible flow problems assesses the effectiveness and sensitivity of each preconditioner to various parameters, including mesh and Hessian regularization, linearization state, and solution space resolution.

The content in this dissertation is heavily based on two articles (Vandergrift and Kummer, 2024; Vandergrift and Zahr, 2024). In each article, the author of this dissertation is the first author and has contributed significantly to the presented work. This includes sole responsibility for the actual implementation and evaluation of the methods and writing the initial drafts of the articles. The co-authors, Dr.-Ing. Florian Kummer and Professor Matthew J. Zahr served in advisory roles, respectively. They were instrumental in developing the original ideas, in assisting with their continuous refinement, and provided the necessary CFD codes that form the foundation of the implementation. Their contributions to the articles were mainly in the realms of initial conceptualization, proofreading, and making minor corrections.

Remark 1. *In our current configuration, we have not been able to devise a method that allows the displayed equation numbers within the text to be black. Hence, equations with a link feature blue reference numbers and those without feature black ones.*

1.3 Structure of the dissertation

The dissertation is structured into seven chapters, with the current chapter serving as an introduction. Chapter 2 briefly summarizes basic concepts and equations related to the theory of inviscid compressible flows with shocks. First, inviscid compressible flows are addressed by presenting a conservative formulation of the Euler equations (Section 2.1.1) followed by their non-dimensionalization (Section 2.1.2). Subsequently, an overview of shock wave phenomena is provided, including discussions of normal (Section 2.2.1) and oblique shock waves (Section 2.2.2). Further, an analysis of shock-acoustic-wave interactions for 1D flows is presented (Section 2.3).

Chapter 3 explores high-order XDG discretization techniques for a general transformed system of conservation laws, which incorporate domain deformations (Section 3.1). The exploration is followed by the system's discretization employing an XDG method (Section 3.2). Numerical flux functions employed in this study are discussed along with desired properties (Section 3.3). The chapter concludes with a discussion of specific conservation laws and associated boundary conditions considered in this research (Section 3.4).

In Chapter 4, the novel implicit XDG shock tracking method developed by Vandergrift and Kummer (2024) is presented. First, the formulation of the optimization problem at the heart of the method is examined (Section 4.1) and the SQP solver employed for its solution is described

(Section 4.2). The outline continues to explore the robustness measures implemented to ensure the stability and reliability of the XDG-IST method (Section 4.3), followed by details on the initialization and termination of the solver (Section 4.4). Subsequently, the full algorithm is presented (Section 4.5) and different level set discretization approaches employed within this research are discussed (Section 4.6).

In Chapter 5, numerical experiments are conducted to evaluate the performance and effectiveness of different variants of the XDG-IST method through various test cases. First, a series of test cases from four systems of conservation laws is presented to which the XDG-IST method is applied (Section 5.1). Second, results of numerical studies (Section 5.2) examining different variants of the XDG-IST method are shown, followed by a conclusion (Section 5.3).

Chapter 6 delves into the developed preconditioners for efficiently solving linear systems that arise in IST methods, focusing on those methods employing mesh deformations and in particular on the HOIST method developed by Zahr et al. (2020). This exploration begins with an overview of the HOIST method (Section 6.1) and progresses to examine the developed preconditioners (Section 6.2). Finally, results from detailed numerical experiments with these preconditioners are reported (Section 6.3), analyzing their performance against several key optimization solver parameters.

Chapter 7 concludes the dissertation and provides an outlook. For the research goals mentioned before, the key contributions of this work are listed and potential avenues for future research are explored, in particular for the XDG-IST method (Section 7.1) and the development of preconditioners for IST (Section 7.2). Lastly, the contributions are synthesized for both topics (Section 7.3).

2 Inviscid compressible flows with shocks

In this chapter, we provide a brief description of inviscid compressible flows with shocks and present analytical expressions for basic examples, which are used to validate and test the novel numerical methods introduced in the subsequent chapters. Compressible flows are prevalent in various industrial applications, such as jet engines and commercial aircraft. The conventional incompressibility constraint, assuming a constant density, becomes invalid in these flows due to significant density variations. In high-speed compressible flows, inertia forces dominate, allowing for the neglect of viscosity and the Euler equations are commonly employed as the foundational model for inviscid compressible flow.

The structure of this chapter is outlined as follows: The discussion begins with an introduction to inviscid compressible flows. Then, in Section 2.1, two conservative forms of the two-dimensional (2D) Euler equations are introduced as a mathematical model for compressible flows. Further, shock wave phenomena are discussed in Section 2.2 and, finally, Section 2.3 dives into the study of interactions between shock waves and acoustic waves in one-dimensional (1D) flows, utilizing the linearized Euler equations.

We start by briefly describing compressibilty and the flow regimes arising in compressible flows drawing from the textbook by Anderson (2003). We describe the flow of a fluid inside a space-time domain $\Omega = \Omega_x \times \mathcal{T}$ in terms of its density $\rho : \Omega \to \mathbb{R}_+$, its velocities $u, v : \Omega \to \mathbb{R}$ and its pressure $p : \Omega \to \mathbb{R}_+$, where the two-dimensional spatial domain is denoted by $\Omega_x \subset \mathbb{R}^2$ and the time interval by $\mathcal{T} := [t_{\text{start}}, t_{\text{end}}]$.

Compressible Flows Contrary to incompressible flows, where the fluids density ρ is considered constant ($\partial\rho/\partial t = 0$), compressible (real) flows inherently exhibit variable density. In order to define compressibility we examine a fluid element with unit mass, specific volume e_{vol} per unit mass, and corresponding density $\rho = 1/e_{\text{vol}}$. In the flow, this fluid element encounters pressure p, arising from neighboring fluid elements. An infinitesimal pressure increase, Δp, results in the compression of the fluid element by Δe_{vol}. Further assuming a constant temperature, the isothermal compressibility can be defined as

$$c_{\text{cmp}} = -\frac{1}{e_{\text{vol}}}\left(\frac{\Delta e_{\text{vol}}}{\Delta p}\right)_T . \tag{2.1}$$

Generally, gases possess a significantly higher compressibility than liquids. For instance, the isothermal compressibility is $c_{\text{cmp}}^{\text{wtr}} = 5{\times}10^{-10}$ m^2/N at 1 atm for water and $c_{\text{cmp}}^{\text{air}} = 5{\times}10^{-5}$ m/N at 1 atm for air under sea-level conditions, differing in five orders of magnitude. In most technical applications, compressible effects are assumed negligible for gas flows with speeds less than 30% of the local speed of sound a (defined in (2.2) for a perfect gas), given the low density variation.

Flow Regimes Compressible flows exhibit different flow regimes, categorized by respective ratios of the flow velocity $(u, v)^T$ in stream-wise direction and the local speed of sound

$$a = \sqrt{\frac{\hat{\gamma} p}{\rho}} \tag{2.2}$$

for an ideal gas. The local Mach number, defined as

$$\mathrm{Ma} = \frac{\sqrt{u^2 + v^2}}{a}, \tag{2.3}$$

distinguishes three flow regimes based on Mach number values: subsonic flow ($\mathrm{Ma} < 1$), sonic flow ($\mathrm{Ma} = 1$) and supersonic flow ($\mathrm{Ma} > 1$). Although the terms subsonic and supersonic technically denote speeds below and above the local speed of sound, respectively, plane aerodynamicists frequently employ these terms to describe specific ranges of inflow Mach values, denoted by $(\cdot)_\infty$ (see Table 2.1 and the work by Anderson (2003)). This is due to the fact, that each range may exhibit distinct physical phenomena. In this work, we mainly focus on supersonic flows ($\mathrm{Ma}_\infty \in (1.2 - 5.0)$) and all of these flow types, except subsonic flows, typically feature shock waves which are further discussed in Section 2.2.

Table 2.1: Classification of flight regimes

Regime	Subsonic	Transonic	Supersonic	Hypersonic
Ma_∞	<0.8	$0.8 - 1.2$	$1.2 - 5.0$	>5.0

2.1 The Euler equations

This discussion proceeds with the presentation of the Euler equations in two conservative forms: a form with dimensions (Section 2.1.1) and a dimensionless form (Section 2.1.2).

2.1.1 Conservative form

The two-dimensional Euler equations are comprised of conservation laws for mass, momentum, and energy, presented here in differential conservative form

$$\frac{\partial U}{\partial t} + \frac{\partial F_x^{\mathrm{eul}}(U)}{\partial x} + \frac{\partial F_y^{\mathrm{eul}}(U)}{\partial y} = 0, \tag{2.4}$$

where the conserved quantities $U : \Omega \to \mathbb{R}^4$, the convective fluxes $F_x^{\mathrm{eul}} : \mathbb{R}^4 \to \mathbb{R}^4$ and $F_y^{\mathrm{eul}} : \mathbb{R}^4 \to \mathbb{R}^4$ are given by the following expressions:

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}, \quad F_x^{\mathrm{eul}}(U) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ u(\rho E + p) \end{pmatrix}, \quad F_y^{\mathrm{eul}}(U) = \begin{pmatrix} \rho v \\ \rho v u \\ \rho v^2 + p \\ v(\rho E + p) \end{pmatrix}. \tag{2.5}$$

Furthermore, $\rho u, \rho v : \Omega \to \mathbb{R}$ are the momentums and $\rho E : \Omega \to \mathbb{R}_+$ is the total energy per volume. The total energy ρE itself is the sum of the inner energy $\rho e_{\text{in}} : \Omega \to \mathbb{R}_+$ and the kinetic energy $e_{\text{kin}} := \frac{1}{2}\rho(u^2 + v^2)$.

In order to close the equations (2.4), an equation of state must be chosen for the pressure. We confine our analysis to flow configurations where the perfect gas assumption holds true, which means that we neglect intermolecular forces, including Van der Waals forces. This assumption is applicable to a broad spectrum of research and engineering applications, as detailed by Anderson (2003), Section 1.4.1. Written in terms of the quantities introduced so far, the *ideal gas law* is expressed by

$$p = (\hat{\gamma} - 1)\rho e_{\text{in}} = (\hat{\gamma} - 1)(\rho E - e_{\text{kin}}), \tag{2.6}$$

where the heat capacity is assumed to be $\hat{\gamma} = 1.4$ for air at standard conditions.

2.1.2 Non-dimensionalization

It is customary to derive and employ a non-dimensional form of an equation to facilitate the transfer of results between small and large scales in experiments and numerical simulations. To obtain dimensionless parameters for inviscid compressible flows and a non-dimensional form of the Euler equations (2.4), we follow Müller (2014), and choose a reference length $l_{\text{ref}} \in \mathbb{R}$, a reference velocity $u_{\text{ref}} \in \mathbb{R}$, (here $(\cdot)_{\text{ref}}$ denote reference values) and introduce non-dimensional independent variables by

$$x^\star = \frac{x}{l_{\text{ref}}}, \quad y^\star = \frac{y}{l_{\text{ref}}}, \quad t^\star = \frac{u_{\text{ref}} t}{l_{\text{ref}}}. \tag{2.7}$$

Additionally, choosing a reference density $\rho_{\text{ref}} \in \mathbb{R}$, we define non-dimensional dependent variables

$$\rho^\star = \frac{\rho}{\rho_{\text{ref}}}, \quad p^\star = \frac{p}{\rho_{\text{ref}} u_{\text{ref}}}, \quad u^\star = \frac{u}{u_{\text{ref}}}, \quad v^\star = \frac{v}{u_{\text{ref}}}, \quad E^\star = \frac{E}{u_{\text{ref}}^2}, \tag{2.8}$$

where $(\cdot)^\star$ denotes the non-dimensional quantities. Further, we deduce the reference speed of sound and Mach number

$$a_{\text{ref}} = \sqrt{\frac{\hat{\gamma} p_{\text{ref}}}{\rho_{\text{ref}}}}, \quad \text{Ma}_{\text{ref}} = \frac{u_{\text{ref}}}{a_{\text{ref}}}. \tag{2.9}$$

This change of variables leaves the Euler equations completely unchanged (Müller, 2014), thus, without loss of generality, we set $l_{\text{ref}} = 1\text{m}$, $\rho_{\text{ref}} = 1\frac{\text{kg}}{\text{m}^2}$ and $u_{\text{ref}} = 1\frac{\text{m}}{\text{s}}$. From this point on, only the non-dimensional quantities will be referred to, while omitting the $(\cdot)^\star$ in the definition for simplicity.

2.2 Shock waves

This section serves as an introductory exploration into the theory of shock waves. A shock wave is a thin layer within the flow field that exhibits an abrupt and nearly discontinuous alteration in pressure, temperature, velocity and density. An illustrative example of its manifestation is found when a blunt body travels at supersonic speeds, giving rise to a shock wave ahead of it.

In this scenario, the emergence of a shock wave is a consequence of the sudden compression of air molecules encountered in the body's trajectory.

A notable attribute of shock waves is their inherent thinness, typically measured on the order of micrometers under standard air conditions. When employing the inviscid Euler equations for modeling, as done in this work, viscous forces are neglected and shock waves are conceptualized as true discontinuities, i.e., having zero thickness.

For simplified examples the jumps in the flow variables can be computed analytically and we will use the resulting expressions to construct test-cases for our numerical methods. We introduce relations for calculating the changes in flow quantities across a stationary normal shock wave in Section 2.2.1. These relations are extended for two-dimensional stationary oblique shock waves in Section 2.2.2. The content of the subsequent sections draws from the works by Anderson (2003) and Geisenhofer (2021).

2.2.1 Normal shock waves

We examine a normal shock wave, which can be characterized by a horizontal flow (i.e., $v = 0$) separated by a shock wave positioned orthogonal to the flow direction (see Figure 2.1). Across the shock most flow quantities feature a discontinuity, so we denote by $(\cdot)_L$ (i.e., ρ_L, u_L ...) the quantities on the supersonic side ($\mathrm{Ma}_L > 1$) of the shock, whereas the quantities on the subsonic side ($\mathrm{Ma}_R < 1$) are denoted as $(\cdot)_R$. Next, the aim is to calculate the post-shock values, given the pre-shock flow. To do so, the Rankine-Hugoniot jump conditions for stationary shocks Rankine, 1870; Hugoniot, 1887

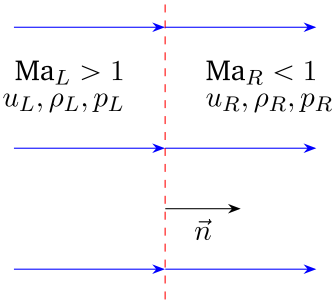


Figure 2.1: Schematic stationary shock wave (*dashed red line*) with pre- $(\cdot)_L$ and post-shock $(\cdot)_R$ variables.

$$\begin{aligned}
[\![\rho\vec{u}\cdot\vec{n}]\!] &= 0, && \text{(continuity equation)}, && (2.10a)\\
[\![\rho u\vec{u}\cdot\vec{n} + pn_x]\!] &= 0, && \text{(x-momentum equation)}, && (2.10b)\\
[\![\rho v\vec{u}\cdot\vec{n} + pn_y]\!] &= 0 && \text{(y-momentum equation)}, && (2.10c)\\
[\![(\rho E + p)\vec{u}\cdot\vec{n}]\!] &= 0, && \text{(energy equation)} && (2.10d)
\end{aligned}$$

can be used, where $\vec{n} := (n_x, n_y)^T \in \mathbb{R}^2$ is a normal unit vector orthogonal to the shock and $\vec{u} := (u, v)^T$ the velocity vector. These conditions ensure the conservation of mass, momentum and total energy across the shock. Further, $[\![\cdot]\!] = (\cdot)_L - (\cdot)_R$ denotes the jump operator (which is also defined for discontinuous Galerkin (DG) functions in (3.15)). In the examined case of the normal shock wave, we have horizontal flow ($v_L, v_R = 0$) and the normal vector can be chosen as $\vec{n} = (1, 0)^T$ so that the jump conditions simplify to

$$\begin{aligned} [\![\rho u]\!] &= 0, && (2.11a)\\ \left[\!\left[\rho u^2 + p\right]\!\right] &= 0, && (2.11b)\\ [\![(\rho E + p)u]\!] &= 0. && (2.11c) \end{aligned}$$

Together with the ideal gas law (2.6), we obtain a system consisting of four equations with four unknowns, i.e., the post-shock values ρ_R, p_R, E_R and u_R, which can be solved by simple algebraic manipulations. The solution (except for the Energy) can be nicely expressed in terms of the supersonic Mach number Ma_L, as done by Geisenhofer (2021), and writes:

$$\begin{aligned} \rho_R &= \frac{(\hat{\gamma}+1)\mathrm{Ma}_L^2}{2 + (\hat{\gamma}-1)\mathrm{Ma}_L^2}\rho_L,\\ u_R &= \frac{2 + (\hat{\gamma}-1)\mathrm{Ma}_L}{(\hat{\gamma}+1)\mathrm{Ma}_L^2}u_L,\\ p_R &= \left[1 + \frac{2\hat{\gamma}}{\hat{\gamma}+1}\left(\mathrm{Ma}_L^2 - 1\right)\right]p_L,\\ E_R &= \frac{p_R}{\rho_R(\hat{\gamma}-1)} + \frac{1}{2}u_R^2. \end{aligned} \tag{2.12}$$

Additionally, one can derive a solution for the subsonic Mach number Ma_R, temperature T_R and the entropy $\hat{s}_R$ by

$$\begin{aligned} \mathrm{Ma}_R &= \sqrt{\frac{1 + \frac{(\hat{\gamma}-1)}{2}\mathrm{Ma}_L^2}{\hat{\gamma}\mathrm{Ma}_L^2 - \frac{(\hat{\gamma}-1)}{2}}},\\ \frac{T_R}{T_L} = \frac{\hat{h}_R}{\hat{h}_L} &= \left[1 + \frac{2\hat{\gamma}}{\hat{\gamma}+1}\left(\mathrm{Ma}_L^2 - 1\right)\right]\left[\frac{2 + (\hat{\gamma}-1)\mathrm{Ma}_L^2}{(\hat{\gamma}+1)\mathrm{Ma}_L^2}\right]. \end{aligned} \tag{2.13}$$

We may conclude that pressure p, density ρ, internal energy E, Mach number Ma, temperature T and entropy $\hat{s}$ rise across a shock, while the velocity u decreases. The x-momentum is not affected as stated by the jump condition (2.11a).

2.2.2 Oblique shock waves

Oblique shock waves are a generalization of normal shock waves, appearing when supersonic flow is deflected by a straight-sided surface towards the main bulk of the flow. Figure 2.2 presents such a flow configuration. There, an inclined plane (wedge) with inclination angle θ_{wdg} is shown. The supersonic flow deflected by it, produces a straight-sided shock with angle θ_{shk}. In the following, we examine the relations between pre- and post-shock quantities, denoted by $(\cdot)_L$ and $(\cdot)_R$, and the angles of shock and wedge.

In the textbook by Anderson (2003), relations between the two angles $\theta_{\text{wdg}}, \theta_{\text{shk}}$ and the inflow Mach number Ma_L are given. The wedge angle θ_{wdg} can be expressed in terms of θ_{shk} and Ma_L by

$$\theta_{\text{wdg}} = 2\cot(\theta_{\text{shk}})\left[\frac{\mathrm{Ma}_L^2\sin^2(\theta_{\text{shk}}) - 1}{\mathrm{Ma}_L^2(\hat{\gamma} + \cos(2\,\theta_{\text{shk}})) + 2}\right]. \tag{2.14}$$

On the basis of (2.14), two fundamental aspects of oblique shocks can be observed:

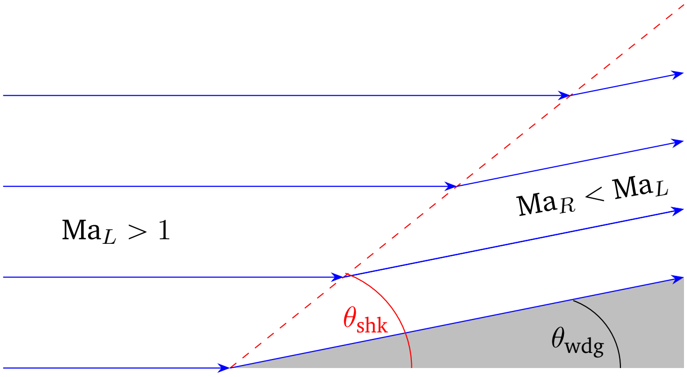
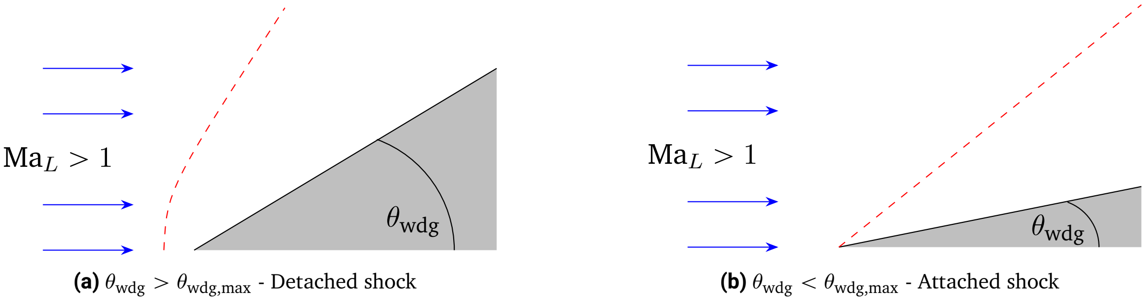


Figure 2.2: Schematic supersonic flow over an inclined plane (*grey*) with inclination angle θ_{wdg} producing an attached oblique shock (*dashed red line*) with angle θ_{shk}.



(a) $\theta_{\mathrm{wdg}} > \theta_{\mathrm{wdg,max}}$ - Detached shock

(b) $\theta_{\mathrm{wdg}} < \theta_{\mathrm{wdg,max}}$ - Attached shock

Figure 2.3: Two distinguished cases of supersonic flow over an inclined plane (*grey*) with inclination angle θ_{wdg} producing a shock (*dashed red line*). For inclination angle $\theta_{\mathrm{wdg}} > \theta_{\mathrm{wdg,max}}$ (*left*) a detached curved shock (*dashed red line*) is obtained, while in the other case (*right*) an attached oblique shock (*dashed red line*) is obtained.

1. Given an incoming Mach number Ma_L there exists a maximum deflection angle $\theta_{\mathrm{wdg,max}}$. If the plane is such that $\theta_{\mathrm{wdg}} > \theta_{\mathrm{wdg,max}}$, there is no straight-sided shock solution (as in Figure 2.3) and instead a detached, curved shock in front of the wedge is obtained.
2. For angles $\theta_{\mathrm{wdg}} < \theta_{\mathrm{wdg,max}}$ with a straight-sided shock, two possible shock angles $\theta_{\mathrm{shk,strong}} > \theta_{\mathrm{wdg,weak}}$, both satisfying (2.3), exist. The *weak* shock solution, obtained by $\theta_{\mathrm{wdg,weak}}$, exhibits less severe changes across the shock, features a mostly supersonic post-shock Mach number ($\mathrm{Ma}_R > 1$). According to Anderson (2003), this solution is favored in nature. The second one is called the *strong* shock solution accompanied by more severe changes across the shock and a subsonic post shock Mach number $\mathrm{Ma}_L < 1$. According to Anderson, it can be forced to occur by an increased downstream pressure.

Next, the computation of post-shock quantities for the oblique shock wave case ($\theta_{\mathrm{wdg}} < \theta_{\mathrm{wdg,max}}$) is examined. To obtain the expressions, the relations (2.12) and (2.13) for the normal shock wave can be applied when substituting the velocities u_L, u_R with the shock-normal velocities $u_{L,n}, u_{R,n}$ and the Mach numbers $\mathrm{Ma}_L, \mathrm{Ma}_R$ with their shock-normal equivalents $\mathrm{Ma}_{L,n}, \mathrm{Ma}_{R,n}$. The transformation between regular and shock-normal quantities can be done by

$$\mathrm{Ma}_{L,n} = \mathrm{Ma}_L \sin(\theta_{\mathrm{shk}}), \quad u_{L,n} = u_L \sin(\theta_{\mathrm{shk}}), \tag{2.15}$$

for the pre-shock values, and for the post-shock values by

$$\mathrm{Ma}_{R,n} = \mathrm{Ma}_R \sin(\theta_{\mathrm{shk}} - \theta_{\mathrm{wdg}}), \quad u_{R,n} = u_R \sin(\theta_{\mathrm{shk}} - \theta_{\mathrm{wdg}}), \tag{2.16}$$

as mentioned by Anderson (2003). As the derivation is straight-forward we do not include the resulting expressions here.

In conclusion, we have obtained analytical expressions for the post-shock flow variables for stationary normal and oblique shock waves. They will be useful in constructing test cases for the Euler equations (Sections 5.1.4 and 5.1.3). Further, we also aim to construct test cases for 1D shock-acoustic-wave interaction problems, which will be examined in the next section.

2.3 One-dimensional shock-acoustic-wave interaction

In the fields of fluid dynamics and aerospace engineering, the study of shock-acoustic interaction has remained a cornerstone of research over the years. Its practical implications extend across diverse technical domains, notably in unraveling phenomena such as transonic buffet effects. There, in the context of an airfoil encountering transonic flow, the shock wave positioned on the suction side of the wing undergoes self-sustained oscillations, leading to an unsteady flow characterized by fluctuations in lift and drag (Feldhusen-Hoffmann et al., 2018). Theoretical frameworks suggest that this instability originates from an acoustic feedback loop, wherein disturbances in the flow field downstream of the shock wave interact with upstream-propagating acoustic waves, forming a cohesive mechanism.

In this section, we discuss a simplified case of the interaction between 1D shock waves and acoustic waves by the means of a linearized interaction analysis (LIA). The presented expressions are later referenced when constructing test cases to validate the implicit XDG shock tracking (XDG-IST) method for the 1D space-time Euler equations. In Section 2.3.1, we present the linearization of the 1D Euler equations. Then, different possible acoustic wave types occurring in a fluid are examined as in Section 2.3.2. Lastly, we describe the phenomena resulting from the shock-acoustic-wave interaction in Section 2.3.3.

2.3.1 Linearized one-dimensional Euler equations

Flow decomposition In the simplified case of 1D shock waves, solutions of the interaction can be computed by the means of a LIA. There, the flow is assumed to consist of a steady- and an unsteady fluctuation component, the latter being usually magnitudes smaller than the steady component (a similar approach is used in the context of linear stability theory). In this sense, we assume a 1D flow ($v = 0, \Omega_x \subset \mathbb{R}$) and introduce its decomposition into a piece-wise constant base-flow (denoted as $(\cdot)^\circ$) and corresponding fluctuations (denoted by $(\cdot)'$), i.e.,

$$p(x,t) = p^\circ(x) + p'(x,t), \quad \rho(x,t) = \rho^\circ(x) + \rho'(x,t), \quad u(x,t) = u^\circ(x) + u'(x,t), \tag{2.17}$$

where $p', \rho', u' : \Omega \to \mathbb{R}$ and $p^\circ, \rho^\circ, u^\circ : \Omega_x \to \mathbb{R}$. The base flow is steady and constant on either side of the shock positioned at $x_s \in \Omega_x$, i.e.,

$$\psi^\circ(x) = \begin{cases} \psi_L^\circ & \text{if } x \leqslant x_s \\ \psi_R^\circ & \text{else} \end{cases}, \quad \psi \in \{p, \rho, u, a, \mathrm{Ma}\}, \tag{2.18}$$

where $p_L^\circ, \rho_L^\circ, \mathrm{Ma}_L^\circ, a_L^\circ, p_R^\circ, \rho_R^\circ, \mathrm{Ma}_R^\circ, a_R^\circ \in \mathbb{R}_+$ and $u_L^\circ, u_R^\circ \in \mathbb{R}$ are the constant pre- and post shock values (which can be derived from the expressions presented in Section 2.2.1).

Linearized Euler equations Under the assumption of a steady base flow with significantly small perturbations the Euler equations can be linearized. This is achieved by inserting the decomposition (2.17) into the 1D Euler equations and neglecting all terms of second order, i.e., products between two fluctuations or their derivatives. The resulting linearized Euler equations write as

$$\frac{\partial \rho'}{\partial t} + \rho^\circ \frac{\partial u'}{\partial x} + u^\circ \frac{\partial \rho'}{\partial x} = 0, \tag{2.19a}$$

$$\rho^\circ \frac{\partial u'}{\partial t} + \rho^\circ u^\circ \frac{\partial u'}{\partial x} + \frac{\partial p'}{\partial x} = 0, \tag{2.19b}$$

$$\frac{\partial (\rho E)'}{\partial t} + u^\circ \frac{\partial}{\partial x} \left((\rho E)' + p' \right) + \left((\rho E)^\circ + p^\circ \right) \frac{\partial u'}{\partial x} = 0. \tag{2.19c}$$

Additionally, a linearized equation of state and linearized jump conditions

$$[\![\rho^\circ u' + u^\circ \rho']\!] = 0, \tag{2.20a}$$

$$[\![2\rho^\circ u^\circ u' + u^{\circ 2} \rho' + p']\!] = 0, \tag{2.20b}$$

$$[\![((\rho E)' + p')u^\circ + ((\rho E)^\circ + p^\circ)u']\!] = 0, \tag{2.20c}$$

are obtained by the same procedure. The set of Equations (2.19a)-(2.20c) describes a system of linear partial differential equations (PDEs) for fluctuations.

2.3.2 Acoustic waves

Generally, in 1D flow, two decoupled families of waves exist (Grube, 2020). Entropy waves, solely governed by density fluctuations, and acoustic waves, determined by pressure fluctuations and exhibiting perturbations in pressure, density and velocity. Using the linearized Euler equations (2.19) as a basis, one can derive equations for the linear motion of acoustic waves in the fluid. One such derivation, which assumes the fluctuations to be isentropic and isothermal, is demonstrated in the work by Grube (2020) and results in a wave equation

$$\frac{D^2 p'}{Dt^2} - a^{\circ 2} \frac{\partial^2 p'}{\partial x^2} = 0, \tag{2.21}$$

governing the pressure perturbations for acoustic waves, where $\frac{D}{Dt} = \frac{\partial}{\partial t} + u^\circ \frac{\partial}{\partial x}$ denotes the material derivative. Equation (2.21) is solved by a superposition of two harmonic waves

$$p'(x,t) = \psi^+(x - (u^\circ + a^\circ)t) + \psi^-(x - (u^\circ - a^\circ)t), \tag{2.22}$$

one *fast* acoustic wave $\psi^+ : \mathbb{R} \mapsto \mathbb{R}$ moving with speed $u^\circ + a^\circ$, and a *slow* acoustic wave $\psi^- : \mathbb{R} \mapsto \mathbb{R}$ moving with speed $u^\circ - a^\circ$, both determined by initial conditions. Note, that in the supersonic regime $u^\circ > a^\circ$ holds and hence both waves move downstream, while in the subsonic regime the slow wave moves upstream. Additionally, the density fluctuations ρ' are decomposed into entropy fluctuations $\rho'^{\text{ ent.}}$ and acoustic density fluctuations $\rho'^{\text{ ac.}}$ (i.e., $\rho' = \rho'^{\text{ ent.}} + \rho'^{\text{ ac.}}$), where the latter can be computed from the pressure perturbations (due to the assumption of isothermal perturbations)

$$\rho'^{\text{ ac.}} = \frac{p'}{a^{\circ 2}}. \tag{2.23}$$

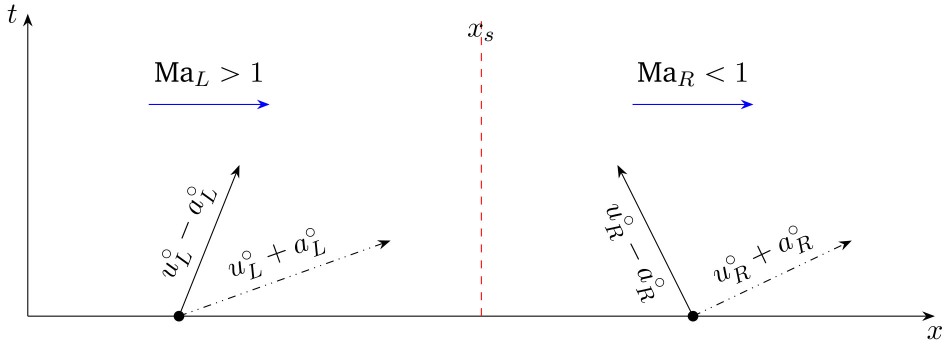


Figure 2.4: Schematic x-t-diagram for an 1D flow with the shock wave positioned at $x = x_s$ (*dashed red line*) and acoustic waves, represented by vectors originating from the x-axis. The acoustic waves have different speeds depending on the flow regime (indicated by Mach number Ma) and whether a slow acoustic wave (*solid line*) or a fast acoustic wave (*dash-dotted line*) is described.

Lastly, one can find the velocity fluctuations utilizing the linearized momentum equation (2.19b) by

$$\frac{Du'}{Dt} = \frac{1}{\rho^\circ}\frac{\partial p'}{\partial x}. \tag{2.24}$$

Incorporating both waves into the velocity results in

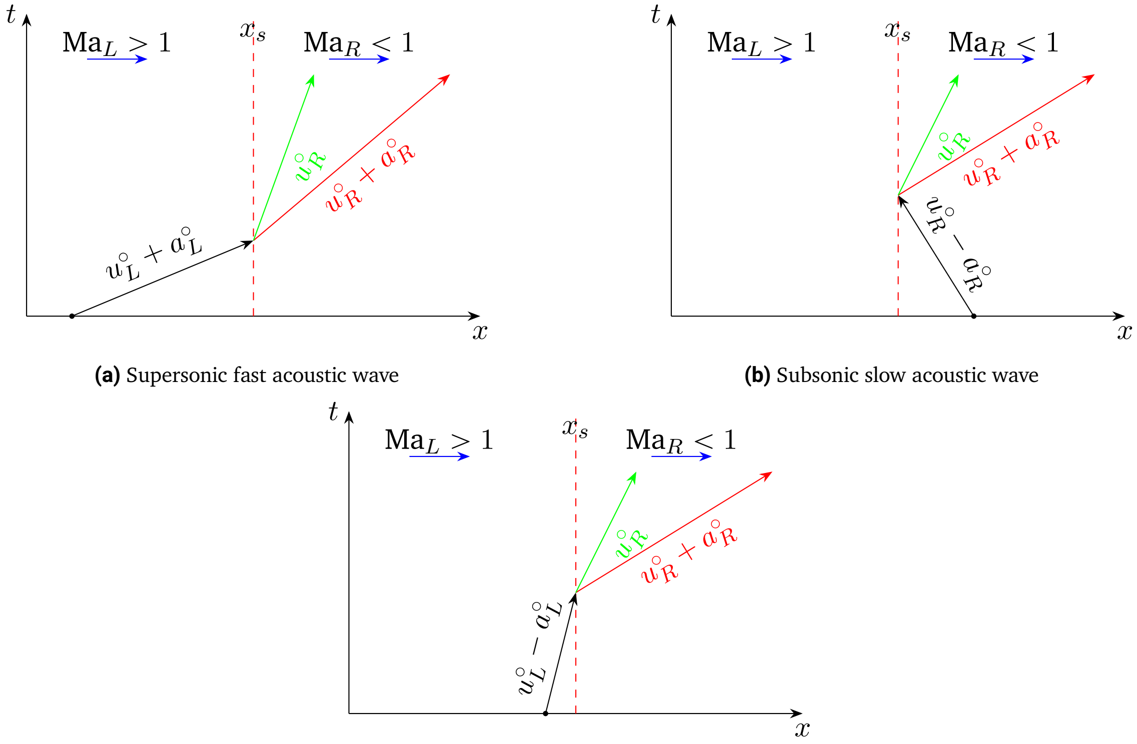
$$u'(x,t) = -\frac{1}{\rho^\circ a^\circ}\psi^+(x-(u^\circ + a^\circ)t) + \frac{1}{\rho^\circ a^\circ}\psi^-(x-(u^\circ - a^\circ)t). \tag{2.25}$$

In Figure 2.4, the four possible acoustic wave types (fast/slow, subsonic/supersonic) are depicted in an x-t-diagram, illustrating the speed and direction of propagation. The expressions (2.22), (2.23) and (2.25) generally describe 1D acoustic waves without taking into account effects resulting from the interaction with shock waves.

2.3.3 Linear shock-acoustic-wave interaction

In this section, we describe the interaction of acoustic waves with shocks in 1D scenarios. The interaction produces an additional acoustic wave and an entropy wave on the subsonic side, both moving downstream and we provide theoretical values for the amplitudes of the additional acoustic waves in the form of amplification/reduction factors (Landau and Lifshitz, 2013). Ultimately, we aim to construct test cases that can be used to validate the novel XDG-IST method. In numerical studies (see Section 5.1.4), we will test if the computed XDG shock tracking solutions align well with the physical phenomena described in the following.

An acoustic wave may propagate towards a shock either from the supersonic or the subsonic side and a linear solution of the interaction for the 1D case was initially obtained by Blokhintsev (1945) and Burgers (1946). Additionally, a more recent discussion can be found in the textbook by Landau and Lifshitz (2013). For all three possible configurations (see Figure 2.5) the interaction results in small variations in the shock's position, usually described in the form of a shock velocity $\frac{d}{dt}x_s : \mathcal{T} \to \mathbb{R}$. Moreover, the interaction produces an acoustic wave and an entropy wave on the subsonic side, both moving downstream. Next, we will examine two cases separately: a fast acoustic wave hitting the shock from the supersonic side (see Figure 2.5 (a)) and a slow acoustic wave hitting the shock from the subsonic side (see Figure 2.5 (b)).



(a) Supersonic fast acoustic wave

(b) Subsonic slow acoustic wave

(c) Supersonic slow acoustic wave

Figure 2.5: Schematic x-t-diagram for 1D shock-acoustic-wave interaction with the shock wave positioned at $x = x_s$ (*dashed red line*). Three cases are distinguished based on the incoming wave type and position: (a) fast acoustic wave (*left*) with speed $u^\circ_L + a^\circ_L$ hitting the shock from the supersonic side ($\mathrm{Ma}_L > 1$) and resulting in a transmitted fast acoustic wave (solid *red*) with speed $u^\circ_R + a^\circ_R$ as well as an entropy wave with speed u°_R on the subsonic side ($\mathrm{Ma}_R < 1$). (b) slow acoustic wave (*left*) with speed $u^\circ_R - a^\circ_R$ hitting the shock from the subsonic side ($\mathrm{Ma}_L > 1$) and resulting in a reflected fast acoustic wave (solid *red*) with speed $u^\circ_R + a^\circ_R$ as well as an entropy wave with speed u°_R on the subsonic side ($\mathrm{Ma}_R < 1$). 3) slow acoustic wave (*left*) with speed $u^\circ_L - a^\circ_L$ hitting the shock from the supersonic side ($\mathrm{Ma}_L > 1$) and resulting in a transmitted fast acoustic wave (solid *red*) with speed $u^\circ_R + a^\circ_R$ as well as an entropy wave with speed u°_R on the subsonic side ($\mathrm{Ma}_R < 1$).

In the case of downstream moving waves hitting the shock from the supersonic side, the interaction produces a transmitted acoustic wave and an additional entropy wave on the subsonic side. Specifically, for an 1D fast acoustic wave (see Figure 2.5 (a)), Burgers provides a solution in terms of a linear system, from which the pressure and density perturbations, p'_R, ρ'_R, unknown for the subsonic side, and the shock velocity $\frac{d}{dt}x_s$ can be computed. Essentially, Burgers provides a functional relationship f^+_{ac} (see Equations (22),(23),(24) in Burgers (1946))

$$\left\{p'_R, \rho'_R, \frac{d}{dt}x_s\right\} = f^+_{\text{ac}}(p'_L, p^\circ, \rho^\circ, u^\circ), \tag{2.26}$$

such that the unknown quantities can be determined from the incoming perturbation p'_L and the base flow quantities. In the textbook by Landau and Lifshitz (2013), §90, Problem 2, the amplification/reduction coefficient for the transmitted wave is derived directly in terms of Mach numbers and writes

$$\frac{\delta p'_R}{\delta p'_L} = \frac{(1 + \mathrm{Ma}^\circ_L)^2}{1 + 2\mathrm{Ma}^\circ_R + 1/\mathrm{Ma}^{\circ 2}_L}\left[1 - \frac{\hat{\gamma} - 1}{\hat{\gamma} + 1}\left(1 - \frac{1}{\mathrm{Ma}^\circ_L}\right)^2\right], \tag{2.27}$$

where $\delta p'_R, \delta p'_L \in \mathbb{R}$ are the amplitudes of the waves p'_R, p'_L.

In the case of a slow wave (see Figure 2.5 (b)) moving upstream and hitting the shock from the subsonic side, the interaction produces a reflected fast acoustic wave and an additional entropy wave, again both moving downstream on the subsonic side. Here, again a solution in terms of a functional relationship f^-_{ac} is provided by Burgers

$$\left\{p'^-_R, \rho'^-_R, \frac{d}{dt}x_s\right\} = f^-_{\text{ac}}(p'_R, p^\circ, \rho^\circ, u^\circ) \tag{2.28}$$

(see Equations (22a),(23a),(24a) in Burgers (1946)). The unknown reflected pressure and density perturbations p'^-_R, ρ'^-_R, and the shock velocity $\frac{d}{dt}x_s$ are computed from the prescribed acoustic wave p'_R and the base flow quantities. Also, for this case, the textbook by Landau and Lifshitz (2013), §90, Problem 1, provides the amplification/reduction coefficient for the reflected wave more conveniently in terms of Mach numbers

$$\frac{\delta p'^-_R}{\delta p'_R} = -\frac{1 - 2\text{Ma}^\circ_R + 1/\text{Ma}^{\circ 2}_L}{1 + 2\text{Ma}^\circ_R + 1/\text{Ma}^{\circ 2}_L}, \tag{2.29}$$

where $\delta p'_R, \delta p'^-_R \in \mathbb{R}$ are the amplitudes of the waves p'_R, p'^-_R respectively.

Lastly, note that for the slow wave hitting the shock from the supersonic side (see Figure 2.5 (c)) no solutions were found in the literature and are not included here.

To construct test cases from the presented expressions, we will consider acoustic waves prescribed by sinusoidal waves, which are further described in the following.

Sinusoidal waves Assuming that the incoming perturbations are described by sinusoidal waves, i.e., waves of form

$$p'(x,t) = \delta p' \sin(\hat{k}x - \omega t), \tag{2.30}$$

the solution provided by Burgers immediately dictates that all resulting perturbations are superpositions of sinusoidal waves, differing potentially only in frequency ω, wave number $\hat{k} \in \mathbb{R}$ or amplitude $\delta p' \in \mathbb{R}$. Inserting this form into the wave equation (2.21) gives the dispersion relation $\omega = \hat{k}u_w$. Here, the wave speed $u_w \in \{u^\circ \pm a^\circ, u^\circ\}$ depends on the type of disturbance (fast acoustic vs. slow acoustic vs. entropy wave) and the flow regime where the waves originates (super- vs subsonic side). The wave number $\hat{k}$ is prescribed for the incoming wave and changes for the waves resulting from the shock-interaction in such a way, that incoming and resulting wave are matched in time. Lastly, the amplitudes for the resulting waves and the shock velocity can be determined using (2.26) for a fast wave hitting the shock from the supersonic side and (2.28) for a slow wave hitting the shock from the subsonic side.

Concluding, we have introduced a framework to construct numerical test cases for 1D shock-acoustic-wave interaction problems. In particular, Equations (2.27) and (2.29) will be utilized to asses the accuracy in this context. Bases on this framework, we will construct concrete test cases and use these to validate the XDG shock tracking method as well as compare it to a shock capturing method in terms of accuracy (Chapter 5).

3 High-order discretization

In this chapter, we describe the high-order extended discontinuous Galerkin (XDG) discretization of a general transformed system of conservation laws, where domain deformations are represented explicitly. This unified discretization framework forms a basis for both implicit shock tracking (IST) methods considered in this work: the implicit XDG shock tracking (XDG-IST) method (Chapter 4) and the high-order implicit shock tracking (HOIST) method (Chapter 6).

The HOIST method employs a discontinuous Galerkin (DG) method to discretize the system of transformed conservation laws and uses domain deformations to adapt edges of computational cells to discontinuous flow features. The DG method is as a high-order extension of the finite volume method (FVM) and approximates solutions using piece-wise polynomial basis functions defined on the cells of a computational grid, hence allowing for discontinuities at cell edges.

The discretization technique we employ in the XDG-IST context is an XDG method. The XDG method is an extension to the DG method and utilizes interfaces to segment the computational domain into sub-regions. For computational cells intersected by the interfaces, the DG basis functions are replaced by XDG basis functions. The latter treat the interfaces as additional cell boundaries such that geometries (i.e., solid regions) or discontinuous features of the flow solution can be accurately represented. In the context of this research, the interfaces are described implicitly by the zero iso-contours of level set functions. Analogously to the HOIST method, the XDG-IST method adapts these interfaces to unknown discontinuous flow features by an optimization approach.

The chapter presents a discretization framework that accommodates both IST methods (HOIST, XDG-IST) and it is structured as follows: in Section 3.1, we introduce a general system of conservation laws which is transformed to a fixed reference domain. Then, the system's XDG discretization is detailed in Section 3.2 and we discuss different numerical flux functions in Section 3.3. Lastly, details on the specific conservation laws and associated boundary conditions considered in this work are given in Section 3.4.

3.1 Transformed system of conservation laws

System of conservation laws We consider a general system of m inviscid conservation laws

$$\nabla \cdot F(U) = 0 \quad \text{in } \Omega. \tag{3.1}$$

They are defined on a physical domain $\Omega \subset \mathbb{R}^d$ and subject to appropriate boundary conditions, where $U : \Omega \to \mathbb{R}^m$ is the solution of the system of conservation laws, $F : \mathbb{R}^m \to \mathbb{R}^{m \times d}$

is the flux function, $\nabla := (\partial_{x_1}, \dots, \partial_{x_d})$ is the gradient operator in the physical domain, and $\vec{x} := (x_1, \dots, x_d) \in \Omega$ are the coordinates. The boundary of the domain $\partial\Omega$ has outward unit normal $\vec{n} : \partial\Omega \to \mathbb{R}^d$. In general, the solution U may contain discontinuities. In this case, the conservation laws (3.1) hold away from the discontinuities and the Rankine-Hugoniot conditions (Majda, 1984) hold at the discontinuities. Further, we note that the formulation of the conservation law in (3.1) is sufficiently general to encapsulate steady conservation laws in a d-dimensional spatial domain and unsteady space-time conservation laws in a $(d-1)$-dimensional spatial domain, i.e., a d-dimensional space-time domain (see Remark 2).

Remark 2. *In the case of an unsteady conservation law the general system is usually written as*

$$\frac{\partial U}{\partial t} + \nabla_x \cdot F_x(U) = 0 \quad \text{in} \;\; \Omega_x \times \mathcal{T}, \tag{3.2}$$

where $\mathcal{T} : (t_{\mathit{start}}, t_{\mathit{end}}) \subset \mathbb{R}$ is a time interval, $\Omega_x \subset \mathbb{R}^{d-1}$ the space-only domain, $\vec{x} := (x_1, \dots, x_{d-1}, t) \in \Omega = \Omega_x \times \mathcal{T}$ are the space-time coordinates, $\nabla_x := (\partial_{x_1}, \dots, \partial_{x_{d-1}})$ the spatial gradient operator and $F_x : \mathbb{R}^m \to \mathbb{R}^{m \times (d-1)}$ the spatial physical flux. The unsteady system is related to the general formulation (3.1) *by*

$$F(U) = \begin{pmatrix} F_x(U) & U \end{pmatrix}, \quad \nabla = \begin{pmatrix} \nabla_x & \partial_t \end{pmatrix}. \tag{3.3}$$

In Section 3.4, we introduce all specific conservation laws considered in this work, giving details on the corresponding physical flux F and the boundary conditions.

Domain transformation One of the two implicit shock tracking methods considered in this work (the HOIST method detailed in Section 6.1) relies on mesh deformation to track the solution's inherent discontinuities. It is beneficial to address these deformations in the domain Ω before proceeding with discretization. To manage the anticipated mesh deformations caused by the shifting of nodal coordinates, the problem is transformed to a fixed reference domain $\Omega_0 \subset \mathbb{R}^d$. This approach is visualized in Figure 3.1. The transformation between Ω_0 and Ω is done by diffeomorphisms $\mathcal{G} \in \mathbb{G}$, where

$$\mathbb{G} := \{\mathcal{G} : \Omega_0 \to \Omega \mid \mathcal{G} : X \mapsto \mathcal{G}(X) \text{ is a diffeomorphism}\}, \tag{3.4}$$

mapping the reference domain Ω_0 to the physical domain Ω. Diffeomophisms $\mathcal{G}$ are defined as bijective and continuously differentiable functions having a continuously differentiable inverse $\mathcal{G}^{-1}$. We follow Zahr and Persson (2018) transforming the system of conservation laws on the physical domain Ω to a system on the reference domain Ω_0 for any $\mathcal{G} \in \mathbb{G}$

$$\bar{\nabla} \cdot \bar{F}(\bar{U}; G) = 0 \quad \text{in} \;\; \Omega_0. \tag{3.5}$$

Here, the solution of the transformed conservation law system is denoted by $\bar{U} : \Omega_0 \to \mathbb{R}^m$, the transformed flux function by $\bar{F} : \mathbb{R}^m \times \mathbb{R}^{d \times d} \to \mathbb{R}^{m \times d}$, the gradient operator on the reference domain by $\bar{\nabla} := (\partial_{X_1}, \dots, \partial_{X_d})$, where $\vec{X} := (X_1, \dots, X_d) \in \Omega_0$ are the reference coordinates, and the mapping Jacobian is denoted by $G : \Omega_0 \to \mathbb{R}$. The latter is defined by

$$G = \bar{\nabla}\mathcal{G}. \tag{3.6}$$

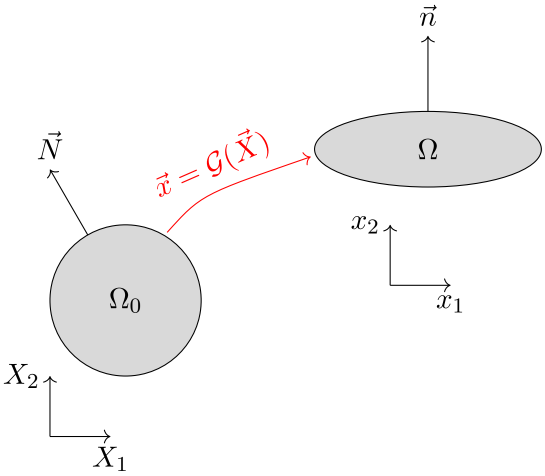


Figure 3.1: Mapping $\mathcal{G}$ between reference Ω_0 and physical domain Ω. Two coordinate systems $\vec{X} = (X_1, X_2)$, $\vec{x} = (x_1, x_2)$ and normal vectors $\vec{N}, \vec{n}$ are shown for each domain, respectively. (Figure adapted from Zahr et al. (2020))

The unit outward normal to the reference domain is denoted $\vec{N}: \partial\Omega_0 \to \mathbb{R}^d$ and the following relation to the unit normal $\vec{n} \in \mathbb{R}^d$ in the physical domain holds:

$$\vec{n} \circ \mathcal{G} = \frac{gG^{-T}\vec{N}}{\left\| gG^{-T}\vec{N} \right\|}, \tag{3.7}$$

where $g: \Omega_0 \to \mathbb{R}^{d\times d}$ is the deformation gradient defined by

$$g = \det G. \tag{3.8}$$

For any $\vec{X} \in \Omega_0$, the transformed and physical solution are related by

$$\bar{U}(\vec{X}) = U(\mathcal{G}(\vec{X})), \tag{3.9}$$

whereas the transformed flux is defined by

$$\bar{F}: (\bar{W}; G) \mapsto (\det G)F(\bar{W})G^{-T}. \tag{3.10}$$

Remark 3. *The transformation of the system of conservation laws, as introduced in this section, is only relevant to the mesh-based HOIST method (Chapter 6). In contrast, for the XDG shock tracking approach (Chapter 4), domain deformations are not utilized. There, we will assume $\mathcal{G} = Id$ (i.e., $\mathcal{G}$ is the identity mapping) and hence reference and physical domain are identical ($\Omega_0 = \Omega$). This equality then extends to all other quantities defined in this section.*

3.2 Discretization of transformed system of conservation laws

We elaborate on the discretization of the system of transformed conservation laws (3.5), extending the methodology outlined by Zahr et al. (2020) to XDG spaces. This marks a significant departure, as it effectively integrates a hybrid approach, bridging the nodal DG discretization employed in the work of Zahr et al. (2020) with the modal XDG discretization featured in the bounded support spectral solver (BoSSS) framework (Vandergrift and Kummer, 2024; Kummer, 2017).

While we present this discretization in a generalized manner, it's crucial to note that we do not employ this hybrid approach directly. Instead, we opt to work independently with their respective implementations. This choice stems from the fact that the actual implementation of the HOIST method, utilized in the development of IST preconditioners presented in Chapter 6, does not integrate cut cells. Conversely, the XDG shock tracking method implemented in the BoSSS framework does not support curved elements, further necessitating separate treatment.

In order to solve (3.5) numerically within the framework of Galerkin methods, the system of partial differential equations (PDEs) is multiplied by selected test functions and then integrated over the computational domain. Then, the expression is integrated by parts, a fundamental step in many Galerkin methods which helps in managing boundary terms and ensuring the correct application of boundary conditions. Further, the solution of (3.5) is approximated within a specifically chosen discrete trial function space (in this work the XDG space described in Sections 3.2.1 and 3.2.2). By selecting appropriate test functions, which are functions from the same or similar function spaces as the trial functions, the method ensures that the solution satisfies the differential equations in a weak sense. This means the solution may not necessarily satisfy the differential equations point-wise everywhere but does so in an integral sense over the domain.

This discretization process culminates in the derivation of algebraic residual forms (Section 3.2.3) that represent the discretized equations. These forms are critical for the subsequent solution of the system through numerical methods. Typically, the nonlinear residuals are tackled using iterative solvers, with the residuals serving as a measure of how close the current approximation is to satisfying the conservation laws. The goal is to iteratively adjust the solution within the discrete spaces until the residuals are minimized, indicating that an approximate solution to the transformed system of conservation laws has been found.

3.2.1 Basic Galerkin approximation spaces

Domain discretization The reference domain $\Omega_0 \subset \mathbb{R}^d$ is discretized into N_K distinct, possibly curved, non-overlapping computational elements K represented by the (background) mesh $\mathcal{K}_h$ defined by

$$\mathcal{K}_h := \{K_1, K_2, \dots, K_{N_K}\} \text{ s.t. } \bigcup\nolimits_{e \in [N_K]} K_e = \Omega_0. \tag{3.11}$$

In the chapter concerning XDG shock tracking it's assumed that the mesh elements are Cartesian. Conversely, for the HOIST method high-order, potentially curved elements are used.

Remark 4. *Note that we use* $[\cdot]$ *with* $[N] := \{1, 2, \dots, N\}$ *to denote index sets for natural numbers* $N \in \mathbb{N}$.

Further, the set of mesh edges is defined by

$$\Gamma := \Gamma_{\text{int}} \cup \Gamma_{\text{ext}} := \bigcup_{e \in [N_K]} \partial K_e, \tag{3.12}$$

where we differentiate between interior edges Γ_{int} and exterior edges Γ_{ext}.

Discontinuous Galerkin space On the basis of the discretized domain we define the DG approximation space

$$\mathcal{V}_h^{P,m} = \mathcal{V}_h^{P,m}(\mathcal{K}_h) := \left\{ \bar{V}_h \in \left(L^2(\Omega_0)\right)^m \;\middle|\; \bar{V}_h\big|_K \in (\mathcal{P}_P(K))^m\,,\ \forall K \in \mathcal{K}_h \right\}, \tag{3.13}$$

where $\mathcal{P}_P(K)$ is the space of (multi-variate-)polynomial functions of total degree at most $P \geqslant 0$ on the element K and $L^2(\Omega_0)$ the space of L^2-integrable functions on Ω_0. Functions $\bar{V}_h \in \mathcal{V}_h^{P,m}$ residing in this space are effectively characterized as coinciding with a vector of m mutually independent polynomials within each mesh cell $K \in \mathcal{K}_h$, while allowing for discontinuities on the inner cell boundaries Γ_{int}. This characteristic means that at points on these internal edges $\vec{x} \in \Gamma_{\text{int}}$ DG functions exhibit not a single value but rather an inner $(\cdot)^{\text{in}}$ and an outer trace $(\cdot)^{\text{out}}$

$$\bar{V}_h^{\text{in}} := \lim_{\epsilon \to 0^+} \bar{V}_h(\vec{x} - \epsilon\vec{n}), \quad \bar{V}_h^{\text{out}} := \lim_{\epsilon \to 0^-} \bar{V}_h(\vec{x} - \epsilon\vec{n}), \quad \forall \vec{x} \in \Gamma, \tag{3.14}$$

which is also illustrated in Figure 3.4. Additionally, the jump operator $[\![\cdot]\!]$ is introduced for ease of notation. It defines the difference between the inner and outer traces on internal edges Γ_{int} and is equal to the inner trace alone on the domain boundary $\partial\Omega$, that is,

$$[[\bar{V}_h]] := \begin{cases} \bar{V}_h^{\text{in}} - \bar{V}_h^{\text{out}} & \text{on } \Gamma_{\text{int}} \\ \bar{V}_h^{\text{in}} & \text{on } \partial\Omega \end{cases}. \tag{3.15}$$

Continuous Galerkin space To discretize the domain mappings expressed in (3.4), we adhere to the methodology outlined by Zahr et al. (2020). We introduce a globally continuous Galerkin (CG) space $\mathcal{W}_h^{P_c}$ of degree $P_c \geqslant 0$ by

$$\mathcal{W}_h^{P_c} = \mathcal{W}_h^{P_c}(\mathcal{K}_h) := \left\{ \mathcal{G}_h \in \left(C^0(\Omega_0)\right)^d \;\middle|\; \mathcal{G}_h|_K \in (\mathcal{P}_{P_c}(K))^d\,,\ \forall K \in \mathcal{K}_h \right\}, \tag{3.16}$$

whereas generally $C^l(\Omega_0)$ denotes the space of l-times continuously differentiable functions on Ω_0. Similar to the DG space, functions $\mathcal{G}_h \in \mathcal{W}_h^{P_c} \subset \mathcal{V}_h^{P_c,d}$ are characterized by a vector of d polynomials on cells $K \in \mathcal{K}_h$. However, a key distinction lies in the assumption of continuity along cell edges Γ. Consequently, for these globally continuous functions, we observe equal traces $\mathcal{G}_h^{\text{in}} = \mathcal{G}_h^{\text{out}}$, ensuring $[\![\mathcal{G}_h]\!] = 0$.

While $\mathcal{W}_h^{P_c}$ serves as a suitable space for discretizing domain mappings, it's essential to assure the admissibility of individual elements $\mathcal{G}_h \in \mathcal{W}_h^{P_c}$. In the work by Zahr et al. (2020), criteria are established to determine the admissibility of domain mappings. These aim at mitigating undesirable outcomes such as element inversion.

Admissible domain mappings In mesh-based implicit shock tracking (e.g., the HOIST method), the domain mapping transforming the computational background mesh $\mathcal{K}_h$ is discretized such that the mapped mesh approximates the boundaries of the physical domain to high-order. We summarize the approach presented by Huang and Zahr (2022) to describe how the domain mappings are determined by high-order mesh nodes, which eventually are moved to track flow features.

In the context of mesh-based implicit shock tracking, a nodal DG basis is utilized (Hesthaven and Warburton, 2008), that is, the mesh and the basis functions are described by a set of N_v nodes. Let $\{\hat{X}_i\} \subset \mathbb{R}^d$ represent the (ordered) nodes associated with the background mesh $\mathcal{K}_h$ and the CG-space $\mathcal{W}_h^{P_c}$, which may include high-order nodes. Consequently, as a polynomial, each element $\mathcal{G}_h \in \mathcal{W}_h^{P_c}$ can be uniquely determined by its effect on the nodes $\hat{X}_i$. This means that if $\{\Phi_i\}_{i=1}^{N_\mathrm{v}}$ forms a nodal basis of $\mathcal{W}_h^{P_c}$ which is linked to the nodes $\{\hat{X}_i\}_{i=1}^{N_\mathrm{v}}$ and satisfies $\Phi_i(\hat{X}_j) = \delta_{ij}$ (here δ_{ij} denotes the Kronecker-delta), then a map $\mathcal{G}_h$ can be defined by

$$\mathcal{G}_h(\,\cdot\,;\boldsymbol{x}):\ \Omega_0 \to \mathbb{R}^d, \qquad \mathcal{G}_h(\,\cdot\,;\boldsymbol{x}): X \mapsto \sum_{i=1}^{N_\mathrm{v}} \hat{x}_i \Phi_i(X), \tag{3.17}$$

where $\boldsymbol{x} \in \mathbb{R}^{N_{\boldsymbol{x}}}$ ($N_{\boldsymbol{x}} = dN_v$) represents the concatenation of $\{\hat{x}_i\}$. Here, the coefficients $\hat{x}_i \in \mathbb{R}^d$ correspond to the physical coordinates of the reference nodes $\hat{X}_i$ since they align with the mapping's effect at $\hat{X}_i$ based on the selected nodal basis.

We note that $\mathcal{G}_h(\,\cdot\,;\boldsymbol{x})$ is an element of $\mathbb{G}$, if $\mathcal{G}_h(\,\cdot\,;\boldsymbol{x})$ is a diffeomorphism, i.e., a continuously differentiable bijection with continuously differentiable inverse. Ensuring that $\mathcal{G}_h(\,\cdot\,;\boldsymbol{x})$ is a bijection from Ω_0 to Ω means that $\boldsymbol{x} \in \mathbb{R}^{N_{\boldsymbol{x}}}$ must be defined such that $\tilde{\mathcal{K}}_h := \mathcal{G}_h(\mathcal{K}_h;\boldsymbol{x})$ is a valid mesh of Ω, i.e., a partition of Ω into non-overlapping and non-inverted elements. As a polynomial, $\mathcal{G}_h(\,\cdot\,;\boldsymbol{x})$ is continuously differentiable for any $\boldsymbol{x} \in \mathbb{R}^{N_{\boldsymbol{x}}}$, so the deformed mesh $\tilde{\mathcal{K}}_h$ will inherit the property of having no gaps between elements from the reference background mesh $\mathcal{K}_h$. Therefore, $\boldsymbol{x}$ only needs to be restricted to ensure the elements of $\tilde{\mathcal{K}}_h$:

1. are not inverted (injectivity) and
2. conform to the boundary $\partial\Omega$ (surjectivity with respect to Ω).

Explicitly enforcing the first condition presents challenges, so it is integrated into the implicit shock tracking optimization problem (see Huang and Zahr (2022) for details). Furthermore, the second condition is directly incorporated into the definition of admissible mappings. To achieve this, a parametrization for the physical nodes is introduced by

$$\boldsymbol{\phi}:\ \mathbb{R}^{N_{\boldsymbol{y}}} \to \mathbb{R}^{N_{\boldsymbol{x}}}, \qquad \boldsymbol{\phi}: \boldsymbol{y} \mapsto \boldsymbol{\phi}(\boldsymbol{y}) \tag{3.18}$$

such that $\mathcal{G}_h(\mathcal{K}_h;\boldsymbol{\phi}(\boldsymbol{y}))$ conforms to $\partial\Omega$ for any $\boldsymbol{y} \in \mathbb{R}^{N_{\boldsymbol{y}}}$ that does not cause element inversion. For further details on the constructions of this parametrization the reader is referred to the work by Huang and Zahr (2022).

After having discussed the admissibility of domain mappings (only relevant to the HOIST method), we continue the path towards discretization by introducing XDG approximation spaces.

3.2.2 Extended discontinuous Galerkin approximation spaces

Domain decomposition As a basis of the XDG discretization we introduce two discretized level set functions

$$\varphi_b \in \mathcal{V}_h^{P_b,1}, \ \varphi_s \in \mathcal{V}_h^{P_s,1} \tag{3.19}$$

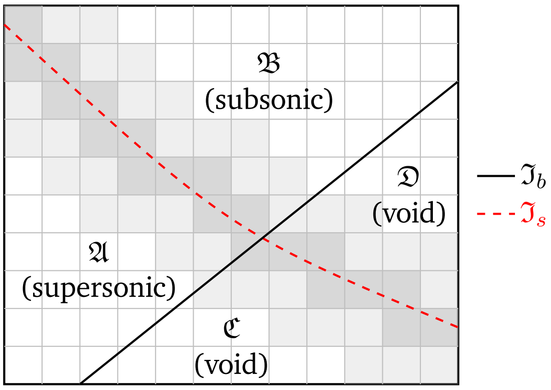


Figure 3.2: Exemplary cut-cell mesh given two interfaces $\mathfrak{I}_s, \mathfrak{I}_b$ (defined by φ_s and φ_b). The resulting sub-domains $\mathfrak{A}$,$\mathfrak{B}$,$\mathfrak{C}$,$\mathfrak{D}$ are depicted as defined in (3.20). Exemplary flow regimes which are used for supersonic flow configurations examined in this work (*supersonic, subsonic, void*) are assigned to each sub-domain. The set of non-empty cut-cells (3.22) (indicated by dark gray) and the near band (dark and light gray) are illustrated for the shock interface $\mathfrak{I}_s$.

associated with polynomial degrees $P_b, P_s \geqslant 1$, respectively. These level sets functions partition the domain Ω_0 into (level set dependent) sub-domains (SDs) $\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \mathfrak{D}$ (see Figure 3.2 for illustration) which are defined by

$$\mathfrak{A} = \mathfrak{A}(\varphi_s, \varphi_b) := \left\{\vec{x} \in \Omega_0 \;\middle|\; \varphi_s(\vec{X}) < 0 \;\cap\; \varphi_b(\vec{X}) < 0\right\}, \tag{3.20a}$$

$$\mathfrak{B} = \mathfrak{B}(\varphi_s, \varphi_b) := \left\{\vec{X} \in \Omega_0 \;\middle|\; \varphi_s(\vec{X}) > 0 \;\cap\; \varphi_b(\vec{X}) < 0\right\}, \tag{3.20b}$$

$$\mathfrak{C} = \mathfrak{C}(\varphi_s, \varphi_b) := \left\{\vec{X} \in \Omega_0 \;\middle|\; \varphi_s(\vec{X}) < 0 \;\cap\; \varphi_b(\vec{X}) > 0\right\}, \tag{3.20c}$$

$$\mathfrak{D} = \mathfrak{D}(\varphi_s, \varphi_b) := \left\{\vec{X} \in \Omega_0 \;\middle|\; \varphi_s(\vec{X}) > 0 \;\cap\; \varphi_b(\vec{X}) > 0\right\}, \tag{3.20d}$$

$$\mathrm{SD} = \mathrm{SD}(\varphi_s, \varphi_b) := \{\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \mathfrak{D}\}, \tag{3.20e}$$

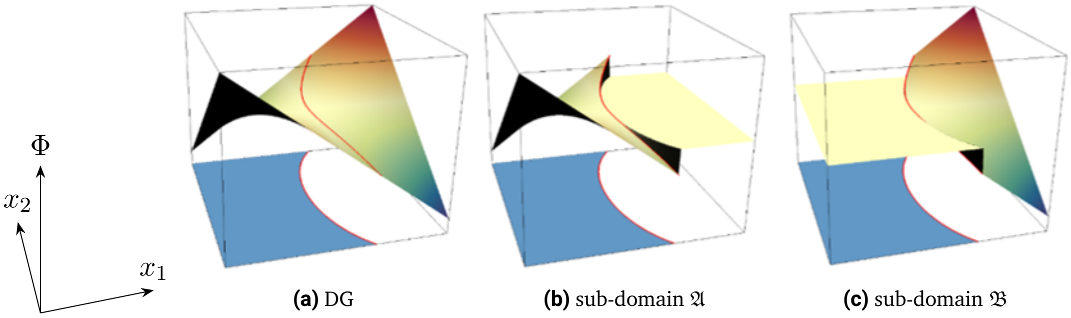
$$\mathfrak{I}_b := \mathfrak{I}(\varphi_b) := \left\{\vec{X} \in \Omega_0 \;\middle|\; \varphi_b(\vec{X}) = 0\right\}, \tag{3.20f}$$

$$\mathfrak{I}_s := \mathfrak{I}(\varphi_s) := \left\{\vec{X} \in \Omega_0 \;\middle|\; \varphi_s(\vec{X}) = 0\right\}. \tag{3.20g}$$

The sub-domains are separated by implicitly defined interfaces $\mathfrak{I}_s$, $\mathfrak{I}_b$ represented by the zero iso-contours of their respective level sets φ_s, φ_b. They may be thought of corresponding to different flow regimes or regions. For instance, in the case of supersonic flow, $\mathfrak{A}$ may correspond to the supersonic pre-shock region, while $\mathfrak{B}$ may correspond to the subsonic post-shock region and the shock may coincides with $\mathfrak{I}_s$. Subsequently, the sub-domains $\mathfrak{C}, \mathfrak{D}$ would correspond to a solid or (void) region behind a wall (represented by φ_b) where no flow occurs. In Figure 3.2, this specific decomposition is illustrated given a Cartesian two-dimensional (2D) mesh.

Remark 5. *Throughout this work, we refer to a level set function as level sets and vice versa.*

Cut-cells In the context outlined in (3.20), each cell K_e of the background grid is segmented into phase-cells $K_{e,\mathfrak{s}}$ by intersecting it with sub-domains, where $K_{e,\mathfrak{s}} := K_j \cap \mathfrak{s}$ and $\mathfrak{s}$ belongs to the set of sub-domains SD. This procedure forms the cut-cell grid $\mathcal{K}_h^X$ (illustrated in Figure



(a) DG **(b)** sub-domain $\mathfrak{A}$ **(c)** sub-domain $\mathfrak{B}$

Figure 3.3: Illustration depicting a two-dimensional, single-valued DG function ($P = 2$, $d = 2$) within a single cut-cell and the corresponding XDG functions for sub-domains $\mathfrak{A}$ and $\mathfrak{B}$, assuming the presence of one level set. The original DG function in (a) is cut along the interface (*red line*) to yield XDG functions for each respective sub-domain. The resulting functions for sub-domains $\mathfrak{A}$ (b) and $\mathfrak{B}$ (c) are depicted accordingly.

3.2), which is represented by

$$\mathcal{K}_h^X = \mathcal{K}_h^X(\varphi_s, \varphi_b) := \{K_{e,\mathfrak{s}} := K_e \cap \mathfrak{s} \mid K_e \in \mathcal{K}_h, \mathfrak{s} \in \mathrm{SD}\}. \tag{3.21}$$

Notably, a significant number of these phase-cells $K_{e,\mathfrak{s}}$ may be empty, specifically when the background cell K_e does not intersect with the sub-domain $\mathfrak{s}$. To identify the set of non-empty cut-cells $\mathcal{K}_h^{cc,0}$ related to a level set $\varphi_\star \in \{\varphi_s, \varphi_b\}$, the following definition is employed

$$\mathcal{K}_h^{cc,0}(\varphi_\star) := \left\{K_{e,\mathfrak{s}} \in \mathcal{K}_h^X \mid K_e \cap \mathfrak{I}(\varphi_\star) \neq \varnothing\right\}. \tag{3.22}$$

Additionally, the concept of a near band is introduced for each level set $\varphi_\star$, extending the non-empty cut-cells by those in close proximity. This near band, denoted $\mathcal{K}_h^{cc,1}$, is defined by

$$\mathcal{K}_h^{cc,1}(\varphi_\star) := \left\{K_{e,\mathfrak{s}} \in \mathcal{K}_h^X \;\middle|\; \exists K_{i,\mathfrak{s}} \in \mathcal{K}_h^{cc,0}(\varphi_\star) \text{ s.t. } \partial K_{i,\mathfrak{s}} \cap \partial K_e \neq \varnothing\right\}. \tag{3.23}$$

and illustrated in Figure 3.2.

Extended discontinuous Galerkin space Building on the cut-cell grid framework (3.21), we define the vector-valued XDG space as

$$\mathcal{V}_h^{P,m,X} := \mathcal{V}_h^{P,m}(\mathcal{K}_h^X) = \left\{\bar{V}_h \in \left(L^2(\Omega_0)\right)^m \mid \bar{V}_h|_K \in (\mathcal{P}_P(K))^m, \forall K \in \mathcal{K}_h^X\right\}. \tag{3.24}$$

Note that this space is an extension to the DG-space defined in (3.13), i.e., $\mathcal{V}_h^{P,m} \subset \mathcal{V}_h^{P,m,X}$. By transitioning from the background grid to the cut-cell grid, functions $\bar{V}_h \in \mathcal{V}_h^{P,m,X}$ are granted additional degrees of freedom (DOFs) in the cut-cells, allowing for further discontinuities across the interfaces $\mathfrak{I}_s, \mathfrak{I}_b$. Enriching basis functions with additional XDG basis functions for enhanced discontinuity handling across cut-cells is illustrated in Figure 3.3 for a single interface.

Remark 6. *An important remark to note is that if the level sets do not intersect the domain, implying $\Omega_0 \cap \mathfrak{s} = \Omega_0$ for one sub-domain $\mathfrak{s} \in SD$, then the cut-cell grid $\mathcal{K}_h^X$ reverts to the original background grid $\mathcal{K}_h$. Consequently, in such scenarios, the XDG space becomes equivalent to the DG space. This assumption will be used in the context of the HOIST method (Chapter 6).*

3.2.3 Residual forms

In this section, the nonlinear residual forms are derived which represent the discretized system of conservation laws.

Elemental weak form To obtain the XDG weak form, the system of m conservation laws (3.5) is multiplied by a test function and integrated over the reference domain. After integration py parts, the solution $\bar{U}$ is approximated in a trial space and the test functions are taken from a suitable test space. As the trial space, we opt for the XDG space $\mathcal{V}_h^{P,m,X}$ with polynomial degree P, while the test space considered is an XDG space $\mathcal{V}_h^{P',m,X}$ with $P' \geqslant P$. On an elemental level, the weak XDG form can be expressed as follows: given a mesh deformation $\mathcal{G}_h \in \mathcal{W}_h^{P_c}$, the objective is to find $\bar{U}_h \in \mathcal{V}_h^{P,m,X}$ such that for all test functions $\bar{V}_h \in \mathcal{V}_h^{P',m,X}$, the following condition holds

$$\int_{\partial K} \bar{V}_h^{\text{in}} \cdot \bar{\mathcal{H}}(\bar{U}_h^{\text{in}}, \bar{U}_h^{\text{out}}, \vec{N}_{h,K}; \bar{\nabla}\mathcal{G}_h)\, dS - \int_K \bar{F}(\bar{U}_h; \bar{\nabla}\mathcal{G}_h) : \bar{\nabla}\bar{V}_h \, dV = 0, \tag{3.25}$$

where $\vec{N}_{h,K} : \partial K \to \mathbb{R}^d$ represents the unit outward normal to element $K \in \mathcal{K}_h^X$, and $\bar{\mathcal{H}} : \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^d \times \mathbb{R}^{d\times d} \to \mathbb{R}^m$ denotes the numerical flux function associated with the reference flux $\bar{F}$. The numerical flux $\bar{\mathcal{H}}$ is essential as the surface integral in (3.25) necessitates the evaluation of the inviscid flux of the discrete solution $\bar{U}_h$, which is not single-valued on cell boundaries ∂K. Thus, the numerical flux is introduced as an approximation

$$\bar{\mathcal{H}}(\bar{U}_h^{\text{in}}, \bar{U}_h^{\text{out}}, \vec{N}_{h,K}; \bar{\nabla}\mathcal{G}_h) \approx \bar{F}(\bar{U}; \bar{\nabla}\mathcal{G}_h) \cdot \vec{N}_{h,K}. \tag{3.26}$$

In Section 3.3, further details on the design of the numerical fluxes employed in this work are discussed, along with essential requirements ensuring that the DG discretization maintains properties of consistency, conservativeness, and stability (Hesthaven and Warburton, 2008). Additionally, for exterior edges Γ_{ext}, $\bar{U}_h^{\text{out}}$ corresponds to a boundary state constructed to enforce the appropriate boundary condition. The construction of equation-specific boundary states is discussed in Section 3.4.

Residual form Using the weak form introduced in (3.25) we define the residual form $r_h^{P',P} : \mathcal{V}_h^{P',m,X} \times \mathcal{V}_h^{P,m,X} \times \mathcal{W}_h^{P_c} \times \mathcal{V}_h^{P_s,1} \to \mathbb{R}$ by

$$r_h^{P',P} : (\bar{V}_h, \bar{U}_h, \mathcal{G}_h, \varphi_s) \mapsto \sum_{K \in \mathcal{K}_h^X} r_K^{P',P}(\bar{V}_h, \bar{U}_h, \mathcal{G}_h, \varphi_s). \tag{3.27}$$

It is composed of a sum of elemental residual forms $r_K^{P',P} : \mathcal{V}_h^{P',m,X} \times \mathcal{V}_h^{P,m,X} \times \mathcal{W}_h^{P_c} \times \mathcal{V}_h^{P_s,1} \to \mathbb{R}$ given by

$$\begin{aligned} r_K^{P',P} : (\bar{V}_h, \bar{U}_h, \mathcal{G}_h, \varphi_s) \mapsto & \oint_{\partial K} \bar{V}_h^{\text{in}} \cdot \bar{\mathcal{H}}(\bar{U}_h^{\text{in}}, \bar{U}_h^{\text{out}}, \vec{N}_{h,K}; \bar{\nabla}\mathcal{G}_h)\, dS \\ & - \int_K \bar{F}(\bar{U}_h; \bar{\nabla}\mathcal{G}_h) : \bar{\nabla}\bar{V}_h \, dV \end{aligned} \tag{3.28}$$

where $K \in \mathcal{K}_h^X$ is a phase-cell belonging to the cut-cell mesh. Even though not indicated explicitly in the right-hand-side of Equation 3.28, the residual forms depend implicitly on both interfaces φ_s, φ_b (see Remark 7). The immersed boundary, represented by the corresponding φ_b, is assumed to be fixed in this work, hence we omit the dependence in the definition of $r_h^{P',P}$ and $r_K^{P',P}$.

Subsequently, we insert a basis for the test space ($\mathcal{V}_h^{P,m,X}$), trial space ($\mathcal{V}_h^{P,m,X}$) (with equal polynomial degrees $P' = P$), and domain mapping space ($\mathcal{W}_h^{P_c}$) into (3.27) reducing it to a system of nonlinear algebraic equations in residual form. We further choose an approximation space $\mathcal{C}_h^{P_s} \subset \mathcal{V}_h^{P_s,1}$ for the shock level set φ_s relating it to discrete DOFs $\boldsymbol{\varphi} \in \mathbb{R}^N_{\boldsymbol{\varphi}}$ (more details on the specific choice of $\mathcal{C}_h^{P_s}$ are discussed in 4.6). As a results, the algebraic form of the residual is given by

$$\boldsymbol{r} : \mathbb{R}^{N_{\boldsymbol{u}}} \times \mathbb{R}^{N_{\boldsymbol{x}}} \times \mathbb{R}^{N_{\boldsymbol{\varphi}}} \to \mathbb{R}^{N_{\boldsymbol{u}}}, \qquad \boldsymbol{r} : (\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{\varphi}) \mapsto \boldsymbol{r}(\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{\varphi}), \tag{3.29}$$

where $N_{\boldsymbol{u}} = \dim(\mathcal{V}_h^{P,m})$, $N_{\boldsymbol{x}} = \dim(\mathcal{W}_h^{P_c})$ and $N_{\boldsymbol{\varphi}} = \dim(\mathcal{C}_h^{P_s})$. Note that in this presentation $\bar{U}_h$ depends on the XDG coefficients $\boldsymbol{u}$ in the sense of

$$\bar{U} \approx \bar{U}_h = \boldsymbol{u} \cdot \boldsymbol{\Phi}, \tag{3.30}$$

where $\boldsymbol{\Phi}$ denotes a basis of $\mathcal{V}_h^{P,m,X}$. Further, $\mathcal{G}_h$ is determined from the nodes $\boldsymbol{x}$ as detailed in (3.17), and the domains of integration $K, \partial K$ are implicitly dependent on the level sets φ_b, φ_s (see Remark 7), the latter being defined from $\boldsymbol{\varphi}$.

Additionally, the XDG basis is composed of basis functions $\bar{\Phi}_{je} \in \mathcal{V}_h^{P,1,X}$ corresponding to a specific phase-cell $K_e \in \mathcal{K}_h^X$, and a polynomial mode $n \in [\dim(\mathcal{V}^{P,1}(K_e)]$. A single component $\boldsymbol{r}_e^{i,n} : \mathbb{R}^{N_{\boldsymbol{u}}} \times \mathbb{R}^{N_{\boldsymbol{x}}} \times \mathbb{R}^{N_{\boldsymbol{\varphi}}} \to \mathbb{R}$ of the algebraic residual is obtained by inserting a specific basis function into (3.27) and considering one component F_i of the physical flux:

$$\begin{aligned} \boldsymbol{r}_e^{i,n} : (\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{\varphi}) \mapsto & \oint_{\partial K_e} \bar{\mathcal{H}}_i(\bar{U}_h^{\text{in}}, \bar{U}_h^{\text{out}}, \vec{N}_{h,K}; \bar{\nabla}\mathcal{G}_h) \, [[\bar{\Phi}_{je}]] \; dS \\ & - \int_{K_e} \bar{F}_i(\bar{U}_h; \bar{\nabla}\mathcal{G}_h) \cdot \bar{\nabla}\bar{\Phi}_{je} \, dV. \end{aligned} \tag{3.31}$$

Enriched residual Building on the residual form introduced as (3.27), the algebraic enriched residual form is defined as follows:

$$\boldsymbol{R} : \mathbb{R}^{N_{\boldsymbol{u}}} \times \mathbb{R}^{N_{\boldsymbol{x}}} \times \mathbb{R}^{N_{\boldsymbol{\varphi}}} \to \mathbb{R}^{N'_{\boldsymbol{u}}}, \qquad \boldsymbol{R} : (\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{\varphi}) \mapsto \boldsymbol{R}(\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{\varphi}), \tag{3.32}$$

where $\boldsymbol{R}$ represents the enriched residual. This definition extends the standard residual form by utilizing a test function space of one degree higher, specifically $P' = P + 1$, with $N'_{\boldsymbol{u}} = \dim(\mathcal{V}_h^{P+1,m,X})$ denoting the dimension of this enriched space. The rationale behind using a higher-degree test function space is to enhance the sensitivity of the residual to features in the solution that are not well captured, such as non-aligned interfaces and oscillatory behaviors, a fact discussed in detail in (Zahr et al., 2020). By being more sensitive to these features, the enriched residual acts as a more effective error indicator, particularly useful in the context of implicit shock tracking methods and will be employed in the methods presented in Chapters 4 and 6 to construct an objective function for the optimization problem.

Remark 7. *The dependency of the residual forms on the level sets φ_s, φ_b is predominantly implicit and not immediately apparent. However, it affects the domains of integration ∂K and K, as well as $\bar{U}_h$ and the integrated test functions, all of which are contingent upon the level sets in the cut cells, i.e., $K = K(\varphi_s, \varphi_b)$ and $\bar{U}_h = \bar{U}_h(\varphi_s, \varphi_b)$.*

Remark 8. *Although the discretization is presented to simultaneously incorporate both, mesh deformations obtained by moving nodes and level set movement, in this work, we will only consider each of these separately. For mesh-based implicit shock tracking (Chapter 6), the level set will be assumed to not intersect the background grid (see also Remark 6), resulting in $\mathcal{K}_h^X = \mathcal{K}_h$ and $\mathcal{V}_h^{P,m,X} = \mathcal{V}_h^{P,m}$. Conversely, for implicit XDG shock tracking (Chapter 4), we will assume a fixed background grid (i.e., $\mathcal{G}_h = Id$) (see also Remark 3), ensuring that all transformed quantities are equivalent to the non-transformed ones.*

3.3 Numerical flux function

Following the overview regarding the spatial XDG discretization of a transformed system of conservation laws, we outline the fundamental characteristics of numerical fluxes, which serve as integral components ensuring stability and convergence in any DG method drawing from Di Pietro and Ern (2012), Section 3.2. Further insights into suitable numerical fluxes for hyperbolic conservation laws are available in the works by LeVeque (1992) and Toro (2009). We provide a comprehensive overview of the numerical fluxes employed in this study, starting by discussing desired properties for numerical fluxes in Section 3.3.1. Then, simple numerical fluxes, applicable to a broad range of conservation laws are presented in Section 3.3.2, followed by an introduction of numerical fluxes employed for the Euler equations in Section 3.3.3.

The XDG methods presented in this work require the computation of the residual form (3.27), which feature a boundary integral. Its computation (in a numerical sense) requires evaluation of the projected transformed flux $\bar{F}(\bar{U}) \cdot \vec{N}_{h,K}$ along quadrature nodes situated on cell edges ∂K, where the discrete state $\bar{U}_h$ is multi-valued and $\vec{N}_{h,K} : \partial K \to \mathbb{R}^d$ represents the unit outward normal to the element $K \in \mathcal{K}_h^X$ in the reference domain. The transformed numerical flux function $\bar{\mathcal{H}}$ approximates

$$\bar{\mathcal{H}}(\bar{U}_h^{\text{in}}, \bar{U}_h^{\text{out}}, \vec{N}_{h,K}, \mathcal{G}_h) \approx \bar{F}(\bar{U}) \cdot \vec{N}_{h,K}. \tag{3.33}$$

The numerical flux $\bar{\mathcal{H}}$ may be chosen differently for interior non-cut edges $\partial K \in \Gamma_{\text{int}} \backslash (\mathfrak{I}_s \cup \mathfrak{I}_b)$, cut-cell edges $\partial K \in \mathfrak{I}_s \backslash \Gamma_{\text{int}}$, $\partial K \in \mathfrak{I}_b \backslash \Gamma_{\text{int}}$, and exterior edges $\partial K \in \Gamma_{\text{ext}}$. For DG-methods, usually boundary conditions are prescribed by choosing an appropriate external state $\bar{U}_h^{\text{out}}$. For the specific conservation laws considered, these are introduced in Section 3.4.

Further, we shift our attention to the numerical flux function in the physical domain

$$\mathcal{H} : \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^d \to \mathbb{R}^m, \qquad \mathcal{H} : (U_h^{\text{in}}, U_h^{\text{out}}, \vec{n}_{h,K}) \mapsto \mathcal{H}(U_h^{\text{in}}, U_H^{\text{out}}, \vec{n}_{h,K}), \tag{3.34}$$

which approximates the projected physical flux

$$\mathcal{H}(U_h^{\text{in}}, U_h^{\text{out}}, \vec{n}_{h,K}) \approx F(U_h) \cdot \vec{n}_{h,K}. \tag{3.35}$$

The numerical flux $\bar{\mathcal{H}}$ in the reference domain is uniquely determined by the numerical flux in the physical domain by

$$\bar{\mathcal{H}}(\bar{U}_h^{\text{in}}, \bar{U}_h^{\text{out}}, \vec{N}_{h,K}, \mathcal{G}_h) = \left\| g G^{-T} \vec{N}_{h,K} \right\| \mathcal{H}(U_h^{\text{in}}, U_h^{\text{out}}, \vec{n}_{h,K}), \tag{3.36}$$

so in the remainder of this section, we will discuss the physical numerical flux $\mathcal{H}$ and it will be transformed to the reference domain according to (3.36).

3.3.1 General requirements

We discuss three desired properties (*stability*, *consistency* and *conservativity*) for the numerical flux $\mathcal{H}$ in the context of DG methods, following the textbook by Di Pietro and Ern (2012). Additionally, two properties desired in the case of implicit shock tracking are presented as described by Zahr et al. (2020): *consistency with Rankine-Hugoniot conditions* and *smoothness with respect to the normal*.

For the following discussion concerning the physical numerical flux, we use a simplified notation: given a specific quadrature node on a cell's edge, i.e., $\vec{x} \in \partial\mathcal{G}_h(K)$, for this section denote $\vec{n} = \vec{n}_{h,K}(\vec{x}) \in \mathbb{R}^d$, $U = U_h(\vec{x}) \in \mathbb{R}^m$, $U^{\text{in}} = U_h^{\text{in}}(\vec{x}) \in \mathbb{R}^m$ and $U^{\text{out}} = U_h^{\text{out}}(\vec{x}) \in \mathbb{R}^m$ (depicted in Figure 3.4).

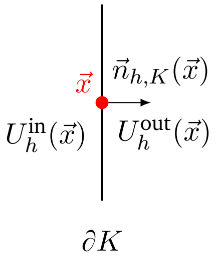


Figure 3.4: Illustration of an interface configuration with inner and outer trace for a function $U_h(\vec{x})$ and $\vec{x} \in \Omega$. $\vec{n}_{h,K}$ denotes the unit outward normal.

Stability It is desired for time dependent problems and states that the L^2-norm $\|U\|_{L^2(\Omega)}$ remains constant over time when assuming homogeneous Neumann boundary conditions. This is in particular true if the *E-flux* property is satisfied, which reads

$$\left(F(U^{\text{in}} + \epsilon(U^{\text{out}} - U^{\text{in}})) \cdot \vec{n} - \mathcal{H}(U^{\text{in}}, U^{\text{out}}, \vec{n})\right)(U^{\text{out}} - U^{\text{in}}) \geqslant 0, \quad \forall \epsilon \in [0,1]. \tag{3.37}$$

Consistency The numerical flux satisfies the *consistency* property if

$$\mathcal{H}(U, U, \vec{n}) = F(U) \cdot \vec{n}. \tag{3.38}$$

This property ensures agreement with the physical flux for continuous solutions and it also follows from the E-flux property for Lipschitz continuous numerical fluxes.

Conservativity When it is ensured that the total amount of the conserved quantities U on the domain Ω only changes due to fluxes across the domain boundary $\partial\Omega$, *conservativity* is upheld. This necessitates that the numerical flux satisfies

$$\mathcal{H}(U^{\text{in}}, U^{\text{out}}, \vec{n}) = -\mathcal{H}(U^{\text{out}}, U^{\text{in}}, -\vec{n}). \tag{3.39}$$

preserving physical conservation properties globally.

Apart from these three classical requirements, in the realm of implicit shock tracking two other conditions are favorable for numerical fluxes, as discussed in the work by Zahr et al. (2020).

Consistency with Rankine-Hugoniot conditions In the context of implicit shock tracking, one seeks to align mesh-edges with solution discontinuities and represent the jumps accurately. On an edge corresponding with the shock front, the numerical flux should be consistent with the Rankine-Hugoniot jump conditions for stationary shocks (see (2.10) for the Euler equations), i.e.,

$$F(U^{\text{in}}) \cdot \vec{n} = F(U^{\text{out}}) \cdot \vec{n} \implies \mathcal{H}(U^{\text{in}}, U^{\text{out}}, \vec{n}) = F(U^{\text{in}}) \cdot \vec{n} = F(U^{\text{out}}) \cdot \vec{n}. \tag{3.40}$$

This can be seen as an amendment to the classical consistency property which does not apply when the right and the left state are different due to a jump in the solution.

Smoothness with respect to the normal This property establishes that the numerical flux

$$\mathcal{H}(U^{\text{in}}, U^{\text{out}}, \vec{n}) \text{ is smooth with respect to variations in } \vec{n}. \tag{3.41}$$

It is beneficial for implicit shock tracking, as the solution is sought as a minimizer of an non-linear optimization problem, based on the residual forms. Numerical fluxes incorporating absolute values or Heaviside functions (i.e., in the form of *if-else*-statements) feature single isolated points of non-smoothness which often coincide with the precise points where a solution is discontinuous and satisfies the Rankine-Hugoniot property (3.40). To this end, an illustrative example for a linear advection problem and more details concerning the smoothness property can be found in (Zahr et al., 2020).

3.3.2 Simple numerical flux functions

Simple numerical flux functions that can be employed for a broad range of conservation laws are introduced in the following.

Central Flux The central flux naively averages the projected flux of both states, i.e.,

$$\mathcal{H}^{\text{cntrl}} : (U^{\text{in}}, U^{\text{out}}, \vec{n}) \mapsto \frac{F(U^{\text{in}}) \cdot \vec{n} + F(U^{\text{out}}) \cdot \vec{n}}{2}. \tag{3.42}$$

It satisfies both consistency properties (3.38), (3.40), the conservativity property (3.39) and the smoothness property (3.41). For linear advection (3.66) and the semi-discrete setting, one can show that the energy norm is conserved exactly (see Di Pietro and Ern (2012), Proposition 3.38), i.e., that the L^2-norm stays the same for all times, which renders the flux optimal in some sense. But typically this lack of numerical dissipation implies that in general for first-order conservation laws the stability property (3.37) is not satisfied.

Smoothed upwind flux In this work ,the classical upwind flux is defined for single-component problems (where $m = 1$) which can be written in terms of a problem-dependent flow direction $u_{\perp} : \mathbb{R} \to \mathbb{R}^d$, i.e., for which $F(U) = u_{\perp}(U)U$ (valid for space-time linear advection and Burgers equation). Then, denoting $U_{\perp} := u_{\perp}(\frac{U^{\text{in}}+U^{\text{out}}}{2})$, the classical upwind flux can be written as

$$\mathcal{H}^{\text{up}} : (U^{\text{in}}, U^{\text{out}}, \vec{n}) \mapsto \begin{cases} F(U^{\text{in}}) \cdot \vec{n} & \text{if } U_{\perp} \cdot \vec{n} \geqslant 0 \\ F(U^{\text{out}}) \cdot \vec{n} & \text{if } U_{\perp} \cdot \vec{n} < 0 \end{cases} \tag{3.43}$$

and it satisfies both consistency properties (3.38), (3.40), the conservativity property (3.39), the E-flux property (3.37), but fails to satisfy the smoothness property (3.41). Thus, following Zahr et al. (2020), we introduce a smoothed upwind flux by writing (3.43) in terms of a Heaviside function and replacing the latter by a smoothed variant

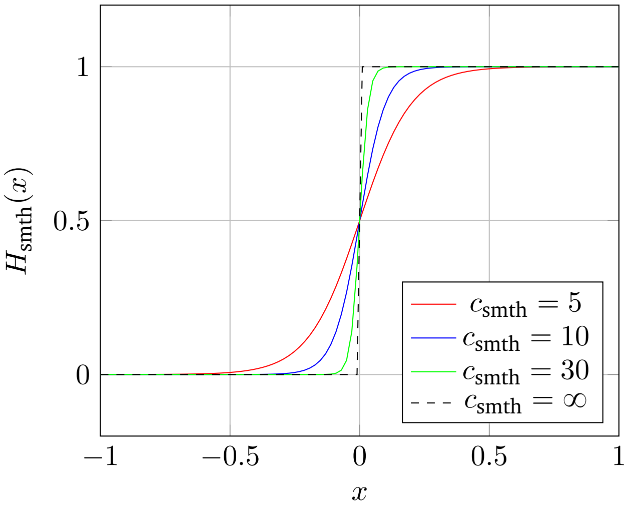


Figure 3.5: Illustration of the smoothed Heaviside function $H_{\text{smth}}(x)$ (Equation 3.45) for different smoothing values $c_{\text{smth}} \in \{5, 10, 30, \infty\}$. (Figure adapted from Zahr et al. (2020))

$$\mathcal{H}^{\text{up}}_{\text{smth}} : (U^{\text{in}}, U^{\text{out}}, \vec{n}) \mapsto (U_{\perp} \cdot \vec{n}) \left(U^{\text{in}} H_{\text{smth}}(U_{\perp} \cdot \vec{n}) + U^{\text{out}}(1 - H_{\text{smth}}(U_{\perp} \cdot \vec{n}) \right), \tag{3.44}$$

where the smoothed Heaviside function (illustrated in Figure 3.5) is defined by

$$H_{\text{smth}}(x) = \frac{1}{1 + e^{-2c_{\text{smth}}x}}, \tag{3.45}$$

and where $c_{\text{smth}} \in \mathbb{R}$ is a smoothing factor ($c_{\text{smth}} = 10$ chosen in this work). This modification ensures that the smoothness property (3.41) is recovered but leads to a violation of the Rankine-Hugoniot property (3.40). In their work, Zahr et al. (2020) show that, in the case of implicit shock tracking, this is a reasonable trade-off, facilitating the convergence of the optimizer.

3.3.3 Numerical flux functions for the Euler equations

The relatively simple numerical fluxes presented in Section 3.3.2 are not feasible in the context of discretizing the Euler equations. While the central flux (3.42) is not stable, the upwind flux (3.44) is introduced only for a specific problem set. Hence, in this section more sophisticated numerical fluxes are introduced. Their evaluation and construction is closely related to the approximate solution of local Riemann problems (Riemann, 1860) presented in the following.

Riemann problem The general Riemann problem concerns a setting where a piece-wise constant discontinuous initial value is imposed and it can be written as:
find a solution $V : \mathbb{R} \times \mathcal{T} \to \mathbb{R}^m$ that satisfies

$$\frac{\partial V(x,t)}{\partial t} + \frac{\partial}{\partial x}\left(F(V(x,t)) \cdot \vec{n}\right) = 0, \quad V(x, t_{\text{start}}) = \begin{cases} U^{\text{in}} & \text{if } x \leqslant 0 \\ U^{\text{out}} & \text{if } x > 0., \end{cases} \tag{3.46}$$

It offers, as outlined by Toro (2009), valuable insights into hyperbolic conservation laws and serves as the solution for these laws under basic initial conditions, encapsulating all relevant physical and mathematical properties. Particularly in the context of the Euler equations, it is akin to the shock tube problem in an infinitely long tube. The first Riemann solver for the Euler equations was proposed by Godunov and Bohachevsky (1959).

Godunov's method The method proposed by Godunov and Bohachevsky (1959) aims to find self-similar solutions $\tilde{V} : \mathbb{R} \to \mathbb{R}^m$ to the Riemann problem (3.46), i.e., $\tilde{V}(x/t) = V(x,t)$. Subsequently, the numerical flux function is defined by

$$\mathcal{H}^{\text{riem}} : (U^{\text{in}}, U^{\text{out}}, \vec{n}) \mapsto F(\tilde{V}(0)) \cdot \vec{n}, \tag{3.47}$$

i.e., the physical flux evaluated at the solution of the Riemann problem along the ray $x/t = 0$. Finding the solution $\tilde{V}$ is equation-dependent; for nonlinear systems, it often entails solving one or more nonlinear algebraic equations iteratively until a desired accuracy is obtained, rendering the evaluation of the numerical flux function computationally expensive. The Godunov flux (3.47) satisfies all properties except the smoothness property and because of its superior accuracy, it is often used as a reference for other numerical methods. On its basis, less accurate but more feasible approximate Riemann fluxes are constructed, which are explored in the following.

Harten-Lax-van Leer-Compact flux In this work, one of the two Godunov-type flux approximations utilized is the Harten-Lax-van Leer-Compact (HLLC) flux, representing an advanced upwind flux (3.43) tailored for the Euler equations. A detailed exposition of the HLLC approximate Riemann solver, specifically customized for the compressible Navier-Stokes (CNS) solver within the BoSSS framework, was presented by Krämer-Eis (2017). For brevity, this discussion provides a concise overview based on the work by Toro (2009) and the work by Krämer-Eis (2017) for one-dimensional (1D) flows.

The HLLC Riemann solver, pioneered by Toro et al. (1994), extends the foundational principles of the original Harten-Lax-van Leer (HLL) solver introduced by Harten et al. (1983). Both methodologies operate on the premise of approximating the requisite wave speeds, corresponding to waves that originate from the intial discontinuity of the 1D Riemann problem (3.46). While the HLL approach adopts a two-wave model, the HLLC method employs a three-wave model, enhancing the resolution of the intermediate wave. The two nonlinear waves (wave speeds S^{in} and S^{out}) representing the minimum and maximum eigenvalues of the quasi-linear Euler system and correspond to shock or rarefaction waves, respectively. The third wave corresponds to a contact discontinuity (wave speed $S_\star$) associated with the third

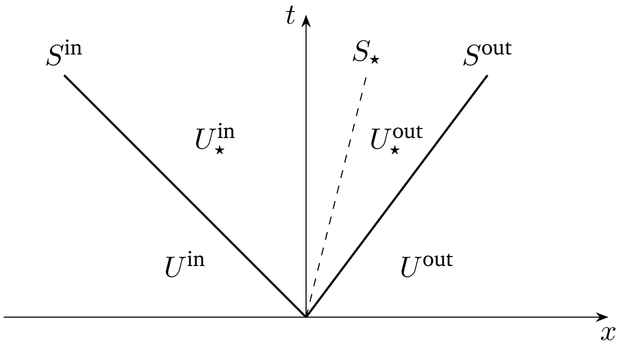


Figure 3.6: Illustration of the two nonlinear waves representing the maximum and minimum eigenvalues of the system (*solid thick lines*, wave speeds S^{in} and S^{out}, corresponding to shock or rarefaction waves), along with a contact wave (*dashed line*, wave speed $S_\star$) associated with the third eigenvalue of the problem and four constant states $U^{\text{in}}, U_\star^{\text{in}}, U^{\text{out}}, U_\star^{\text{out}}$ being separated, all of which characterize the one-dimensional Riemann problem (3.46). (Figure adapted from Krämer-Eis (2017))

eigenvalue of the problem. The mathematical formulation of the HLLC numerical flux, adapted from (Krämer-Eis, 2017), is given by

$$\mathcal{H}^{\text{HLLC}} : (U^{\text{in}}, U^{\text{out}}, \vec{n}) \mapsto \begin{cases} F(U^{\text{in}}) \cdot \vec{n} & \text{if } 0 < S^{\text{in}}, \\ \left(F(U^{\text{in}}) + S^{\text{in}}(U_\star^{\text{in}} - U^{\text{in}})\right) \cdot \vec{n}, & \text{if } S^{\text{in}} \leqslant 0 \leqslant S_\star, \\ \left(F(U^{\text{out}}) + S^{\text{out}}(U_\star^{\text{out}} - U^{\text{out}})\right) \cdot \vec{n} & \text{if } S_\star \leqslant 0 \leqslant S^{\text{out}}, \\ F(U^{\text{out}}) \cdot \vec{n} & \text{if } S^{\text{out}} < 0, \end{cases} \tag{3.48}$$

where $U^{\text{in}}, U_\star^{\text{in}}, U^{\text{out}}, U_\star^{\text{out}}$ are the four constant states being separated by the three waves (see Figure 3.6). Under the assumption of the ideal gas law prevalent in this work, Toro (2009) suggests to estimate the wave speeds depending on a pressure estimate in the star region (i.e., region in between the two non-linear waves $S^{\text{in}}, S^{\text{out}}$)

$$p_\star = \max\left(0,\ \frac{p^{\text{in}} + p^{\text{out}}}{2} - \frac{1}{2}\,[\![u]\!]\,\frac{\rho^{\text{in}} + \rho^{\text{out}}}{2}\frac{a^{\text{in}} + a^{\text{out}}}{2}\right). \tag{3.49}$$

Then, the wave speeds $S^{\text{in}}, S^{\text{out}}$ are estimated as

$$S^{\text{in}} = u^{\text{in}} - a^{\text{in}} q^{\text{in}} \quad \text{and} \quad S^{\text{out}} = u^{\text{out}} - a^{\text{out}} q^{\text{out}}, \tag{3.50}$$

with the wave-type dependent correction factor

$$q^\times = \left\{ \begin{array}{ll} 1 & \text{if} \quad p_\star \leqslant p^\times \\ \left(1 + \frac{\hat{\gamma}+1}{2\hat{\gamma}}\left(\frac{p_\star}{p^\times} - 1\right)\right)^{1/2} & \text{if} \quad p_\star > p^\times, \end{array} \right\} \tag{3.51}$$

where $q^\times \in \{q^{\text{in}}, q^{\text{out}}\}$. Additionally, the contact wave speed $S_\star$ is obtained from the assumed wave speeds $S^{\text{in}}, S^{\text{out}}$ and the known states U^{in}, U^{out} by

$$S_\star = \frac{[\![\rho u(S-u)]\!] - [\![p]\!]}{[\![\rho(S-u)]\!]}. \tag{3.52}$$

Lastly, the missing constant intermediate states $U_\star^{\text{in}}, U_\star^{\text{out}}$ can be found by

$$U_\star^\times = \rho^\times \left(\frac{S^\times - u^\times}{S^\times - S_\star}\right) \begin{pmatrix} 1 \\ S_\star \\ \frac{\rho^\times E^\times}{\rho^\times} + (S_\star - u)\left(S_\star + \frac{p^\times}{\rho^\times}\left(S^\times - u^\times\right)\right) \end{pmatrix}, \tag{3.53}$$

where $U_\star^\times \in \{U_\star^{\text{in}}, U_\star^{\text{out}}\}$ denotes the left and right states.

More details on the estimates are given in the corresponding paper by Toro et al. (1994).

The HLLC flux exhibits classical consistency as defined by property (3.38), yet it lacks consistency regarding the Rankine-Hugoniot conditions, as specified in (3.40). Consequently, this flux is unsuitable for mesh-based implicit shock tracking (Chapter 6), as it generates non-zero residuals even when the mesh is shock-aligned and the solutions adhere to the conservation laws. Hence, it is not advisable for fluxes across the shock interface $\mathfrak{I}_s$ in the context of XDG shock tracking, although it remains viable for regular cell boundaries in the XDG setting. Despite satisfying stability and conservativity properties (3.37) and (3.39) respectively, the HLLC flux fails to meet the smoothness requirement outlined in (3.41), owing to its *if-else* definition. One approach to restore smoothness involves employing smoothed Heaviside functions, similar to those utilized in the smooth upwind flux (3.44), although this strategy is not explored within the scope of this work.

Roe flux Another numerical flux function employed in this work is the approximate Riemann solver developed by Roe (1981). The following brief summary draws from the adaptation to higher dimensions described by Naudet and Zahr (2024). In Roe's approach the nonlinear Riemann problem in (3.46) is approximated with the following linearized Riemann problem: find $\hat{V} : \mathbb{R} \times \mathcal{T} \to \mathbb{R}^m$ that satisfies

$$\frac{\partial}{\partial t}\hat{V}(x,t) + \tilde{B}(U^{\text{in}}, U^{\text{out}}, \vec{n})\frac{\partial}{\partial x}\hat{V}(x,t) = 0 \quad \text{for } x \in \mathbb{R}, \qquad \hat{V}(x, t_{\text{start}}) = \begin{cases} U^{\text{in}} & x \leqslant 0 \\ U^{\text{out}} & x > 0. \end{cases} \tag{3.54}$$

Here, the linearization matrix $\tilde{B} : \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^d \to \mathbb{R}^{m\times m}$ is chosen such that for any $\vec{n} \in \mathbb{R}^d$ (unit length), the linearized problem is

- hyperbolic, i.e., $\tilde{B}(U^{\text{in}}, U^{\text{out}}, \vec{n})$ is diagonalizable with real eigenvalues,
- consistent with the original Riemann problem: $\tilde{B}(U^{\text{in}}, U^{\text{out}}, \vec{n}) \to B(U, \vec{n})$ as $U^{\text{in}}, U^{\text{out}} \to U \in \mathbb{R}^m$, where
$$B(U, \vec{n}) := \frac{\partial[F(U)\cdot\vec{n}]}{\partial U} \tag{3.55}$$
is the Jacobian of the projected physical flux,
- conservative at discontinuities: $[\![F(U)\cdot\vec{n}]\!] = \tilde{B}(U^{\text{in}}, U^{\text{out}}, \vec{n})\,[\![U]\!]$.

Then, Roe's numerical flux, $\mathcal{H}^{\text{roe}} : \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^d \to \mathbb{R}^m$, takes the form

$$\mathcal{H}^{\text{roe}} : (U^{\text{in}}, U^{\text{out}}, \vec{n}) \mapsto \frac{F(U^{\text{in}})\cdot\vec{n} + F(U^{\text{out}})\cdot\vec{n}}{2} + \frac{1}{2}|\tilde{B}(U^{\text{in}}, U^{\text{out}}, \vec{n})|\,[\![U]\!]\,, \tag{3.56}$$

where the absolute value of a matrix $\tilde{B} \in \mathbb{R}^{m\times m}$ is defined as $|\tilde{B}| := \boldsymbol{V}|\Lambda|\boldsymbol{V}^{-1}$, where $\tilde{B} = \boldsymbol{V}\Lambda\boldsymbol{V}^{-1}$ is the eigenvalue decomposition and the absolute value is applied component-wise to a diagonal matrix. The expression for the linearization matrix is

$$\tilde{B} : (U^{\text{in}}, U^{\text{out}}, \vec{n}) \mapsto B(\underline{U}(U^{\text{in}}, U^{\text{out}}, \vec{n}), \vec{n}), \tag{3.57}$$

where $\underline{U} : \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^d \to \mathbb{R}^m$ is a problem-specific Roe average. The Roe flux satisfies both consistency properties (3.38), (3.40), stability (3.37) and conservativity (3.39), but fails to satisfy the smoothness property (3.41). However, if the absolute value function is replaced by a smooth variant (illustrated in Figure 3.7)

$$|\cdot|_{\text{smth}} : x \mapsto x \tanh(c_{\text{smth}} x) \tag{3.58}$$

then smoothness (3.41) is satisfied, while the property (3.37) no longer holds. Again, this is a desirable trade-off for implicit shock tracking, due to requirements already discussed for the upwind flux in Section 3.3.2. Furthermore, it is well-known that the Roe flux can lead to entropy-violating rarefaction shocks, which are circumvented via entropy fixes that directly modify the eigenvalues of the matrix $\tilde{B}(U^{\text{in}}, U^{\text{out}}, \vec{n})$ near zero (sonic point), but theses fixes are not considered in this work.

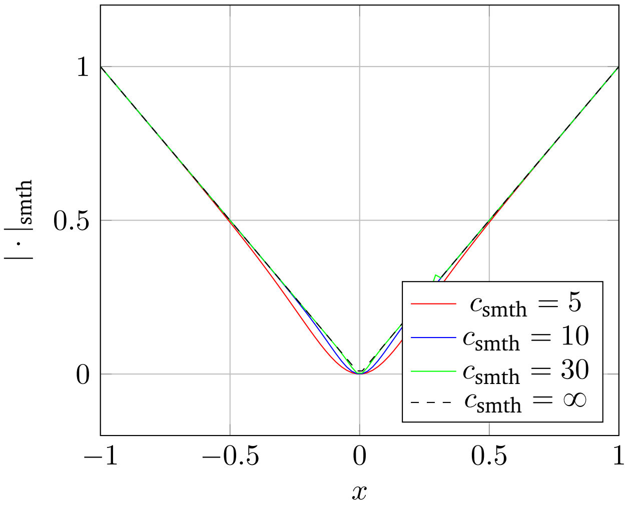


Figure 3.7: Illustration of the smoothed absolute value function $|\cdot|_{\text{smth}}$ (Equation 3.58) for different smoothing values $c_{\text{smth}} \in \{5, 10, 30, \infty\}$. (Figure adapted from Zahr et al. (2020))

In the course of this work Roe's numerical flux was implemented into the BoSSS framework for two problems: The two-dimensional steady Euler equations and the one-dimensional space-time Euler equations. In the following, concrete descriptions of the matrices $\boldsymbol{V}, \boldsymbol{V}^{-1}$ and $\tilde{B}(U^{\text{in}}, U^{\text{out}}, \vec{n})$ for both problems are given, drawing from the work by Naudet and Zahr (2024). To this end, we will express $\tilde{B}(U^{\text{in}}, U^{\text{out}}, \vec{n}) = B(\underline{U}, \vec{n})$ in terms of the Roe-averaged velocity $\underline{\vec{u}}$, enthalpy $\underline{H}$ and speed of sound $\underline{a}$, ($\underline{(\cdot)}$ denotes the Roe-average for a variable), i.e.,

$$B(\underline{U}, \vec{n}) \mapsto B_0(\underline{\vec{u}}, \underline{H}, \underline{a}, \vec{n}), \tag{3.59}$$

and where the Roe-averaged quantities are defined by

$$\underline{\vec{u}}(U^{\text{in}}, U^{\text{out}}) = \frac{\sqrt{\rho^{\text{in}}}\vec{u}^{\text{in}} + \sqrt{\rho^{\text{out}}}\vec{u}^{\text{out}}}{\sqrt{\rho^{\text{in}}} + \sqrt{\rho^{\text{out}}}}, \tag{3.60a}$$

$$\underline{H}(U^{\text{in}}, U^{\text{out}}) = \frac{\sqrt{\rho^{\text{in}}}H^{\text{in}} + \sqrt{\rho^{\text{out}}}H^{\text{out}}}{\sqrt{\rho^{\text{in}}} + \sqrt{\rho^{\text{out}}}}, \tag{3.60b}$$

$$\underline{a}(U^{\text{in}}, U^{\text{out}}) = \left((\hat{\gamma} - 1)\left(\underline{H} - \frac{1}{2}\underline{\vec{u}} \cdot \underline{\vec{u}}\right)\right)^{1/2}, \tag{3.60c}$$

and where the enthalpy is obtained by $H = (\rho E + p)/\rho$.

Roe flux for steady Euler equations For the steady (space-only) Euler equations the projected flux Jacobian $B_x(\underline{U}, \vec{n}_x) := B(\underline{U}, \vec{n})$, denoting $\vec{n}_x = \vec{n}$, can be written by

$$B_x(\underline{U}, \vec{n}_x) = \begin{bmatrix} 0 & (\vec{n}_x)^T & 0 \\ \left(\frac{\hat{\gamma}-1}{2}\right) \|\underline{\vec{u}}\|^2 \vec{n}_x - (\underline{\vec{u}} \cdot \vec{n}_x)\underline{\vec{u}} & (\underline{\vec{u}} \cdot \vec{n}_x)\boldsymbol{I}_{d'} + \underline{\vec{u}}(\vec{n}_x)^T - (\hat{\gamma} - 1)\vec{n}_x\underline{\vec{u}}^T & (\hat{\gamma} - 1)\vec{n}_x \\ \left(\left(\frac{\hat{\gamma}-1}{2}\right) \|\underline{\vec{u}}\|^2 - \underline{H}\right) (\underline{\vec{u}} \cdot \vec{n}_x) & \underline{H}(\vec{n}_x)^T - (\hat{\gamma} - 1)(\underline{\vec{u}} \cdot \vec{n}_x)\underline{\vec{u}}^T & \hat{\gamma}(\underline{\vec{u}} \cdot \vec{n}_x) \end{bmatrix}, \tag{3.61}$$

where $\boldsymbol{I}_{d'} \in \mathbb{R}^{d' \times d'}$ is the identity matrix (for the space-only Euler equations $d' = d$). Further, the eigenvalues $\Lambda_x :\ \mathbb{R}^m \times \mathbb{R}^d \to \mathbb{R}^{(d'+2)\times(d'+2)}$ are given by

$$\Lambda_x : (\underline{U}, \vec{n}_x) \mapsto \begin{bmatrix} (\underline{\vec{u}} \cdot \vec{n}_x) - \underline{a} & & \\ & (\underline{\vec{u}} \cdot \vec{n}_x)\boldsymbol{I}_{d'} & \\ & & (\underline{\vec{u}} \cdot \vec{n}_x) + \underline{a} \end{bmatrix}, \tag{3.62}$$

the right eigenvectors $\boldsymbol{V}_x :\ \mathbb{R}^m \times \mathbb{R}^{d'} \to \mathbb{R}^{(d'+2)\times(d'+2)}$ by

$$\boldsymbol{V}_x : (\underline{U}, \vec{n}_x) \mapsto \begin{bmatrix} 1 & (\vec{n}_x)^T & 1 \\ \underline{\vec{u}} - \underline{a}\vec{n}_x & (\underline{\vec{u}} - \vec{n}_x)(\vec{n}_x)^T + \boldsymbol{I}_{d'} & \underline{\vec{u}} + \underline{a}\vec{n}_x \\ \underline{H} - (\underline{\vec{u}} \cdot \vec{n}_x)\underline{a} & \underline{\vec{u}}^T + \left(\|\underline{\vec{u}}\|^2 /2 - (\underline{\vec{u}} \cdot \vec{n}_x)\right) (\vec{n}_x)^T & \underline{H} + (\underline{\vec{u}} \cdot \vec{n}_x)\underline{a} \end{bmatrix}. \tag{3.63}$$

and the left eigenvectors by

$$(\boldsymbol{V}_x(\underline{U}, \vec{n}_x))^{-1} = \frac{\hat{\gamma} - 1}{2\underline{a}^2} \begin{bmatrix} \|\underline{\vec{u}}\|^2 /2 + \frac{(\underline{\vec{u}} \cdot \vec{n}_x)\underline{a}}{\hat{\gamma}-1} & -\underline{\vec{u}}^T - \frac{\underline{a}}{\hat{\gamma}-1}(\vec{n}_x)^T & 1 \\ \frac{2\underline{a}^2}{\hat{\gamma}-1}(\vec{n}_x + \vec{n}_x - \underline{\vec{u}}) - \|\underline{\vec{u}}\|^2 \vec{n}_x & 2\vec{n}_x\underline{\vec{u}}^T + \frac{2\underline{a}^2}{\hat{\gamma}-1}(\boldsymbol{I}_{d'} - \vec{n}_x{\vec{n}_x}^T) & -2\vec{n}_x \\ \|\underline{\vec{u}}\|^2 /2 - \frac{(\underline{\vec{u}} \cdot \vec{n}_x)\underline{a}}{\hat{\gamma}-1} & -\underline{\vec{u}}^T + \frac{\underline{a}}{\hat{\gamma}-1}(\vec{n}_x)^T & 1 \end{bmatrix}, \tag{3.64}$$

such that $B_x = \boldsymbol{V}_x\Lambda_x{\boldsymbol{V}_x}^{-1}$ is the eigenvalue decomposition. The space-only Roe flux is obtained by plugging the decomposition into (3.56).

Roe flux for space-time Euler equations For the space-time Euler equations the Roe flux is obtained using the expression from the space-only case. In accordance with Naudet and Zahr (2024) we have

$$\tilde{B}(U^{\text{in}}, U^{\text{out}}, \vec{n}) = \boldsymbol{V}_x(\underline{U}, \vec{n}_x)|\Lambda_x(\underline{U}, \vec{n}_x)\|\vec{n}_x\| + n_t\boldsymbol{I}_m|\boldsymbol{V}_x(\underline{U}, \vec{n}_x)^{-1}, \tag{3.65}$$

where the space-time normal $\vec{n} = \begin{pmatrix}\vec{n}_x & n_t\end{pmatrix}^T$ is decomposed into its spatial and temporal parts. The space-only Roe flux is obtained by plugging the decomposition (3.65) into (3.56).

Remark 9. *The Roe flux is often augmented with an entropy fix (Harten and Hyman, 1983) to improve its stability and ensure physical correctness of the solution. However, even though its consideration is perfectly valid in this context, an entropy fix was not considered in this work as the resulting flux would not satisfy the Rankine-Hugoniot property 3.40 (Corrigan et al., 2019a).*

3.4 Considered conservation laws and boundary conditions

In this last section, we introduce four specific systems of conservation laws that have been considered in this work, namely: the 1D space-time advection equation in Section 3.4.1, the 1D space-time Burgers equation in Section 3.4.2, the 2D steady Euler equations in Section 3.4.3, and the 1D space-time Euler equations in Section 3.4.4. The systems are used to construct specific test cases that will be used to evaluate our numerical methods. We present the equations in the general form (3.1) for conservation law systems, which was the starting point of this chapter, and we discuss the boundary conditions employed.

3.4.1 One-dimensional space-time advection equation

We consider the space-time formulation of the linear advection equation

$$\frac{\partial u}{\partial t} + u_a(t)\frac{\partial u}{\partial x} = 0 \quad \text{in } \Omega = \Omega_x \times \mathcal{T}, \tag{3.66}$$

where $\Omega_x \subset \mathbb{R}$ is the spatial domain, $\mathcal{T} = [t_{\text{start}}, t_{\text{end}}] \subset \mathbb{R}$ the temporal domain, $u_a : \mathcal{T} \to \mathbb{R}$ represents a time-dependent advection, and $u : \Omega \to \mathbb{R}$ a velocity. This formulation is linked to the general form (3.1) by setting the conserved variables U and the corresponding physical flux F to

$$U = (u), \quad F(U) = (u_a(t)u \;\; u), \tag{3.67}$$

(hence $d = 2$ and $m = 1$) and considering a space-time setting (see Remark 2). In this work, Dirichlet boundary conditions $u_{\text{dr}} : \partial\Omega \to \mathbb{R}$ are solely weakly imposed using appropriate exact solutions for the whole boundary, i.e., setting

$$U^{\text{out}} = u_{\text{dr}}(x,t), \; \forall (x,t) \in \Gamma_{\text{ext}} \tag{3.68}$$

for all exterior edges Γ_{ext} in the discretized form (3.27). As the numerical flux, the smoothed upwind flux detailed in Equation 3.44 is employed. Additionally, exact solutions for this problem set can be derived by the method of characteristics.

Exact solution Let u be a solution of (3.66) and consider a curve $\tilde{x} : \mathcal{T} \to \mathbb{R}$ along which u is constant, i.e.,

$$\frac{d(u(\tilde{x}(t),t))}{dt} = 0 = \frac{\partial u}{\partial t} + u_a(t)\frac{\partial u}{\partial x}. \tag{3.69}$$

From the chain rule of differentiation, we have

$$\frac{d(u(\tilde{x}(t),t))}{dt} = \frac{\partial u}{\partial t} + \frac{d\tilde{x}}{dt}\frac{\partial u}{\partial x}, \tag{3.70}$$

and comparing (3.70) with the right-hand side of (3.69) yields that

$$\frac{d\tilde{x}}{dt} = u_a(t). \tag{3.71}$$

Equation 3.71 can be integrated to obtain the characteristic curves

$$\tilde{x}(t) = \int_{t_{\text{start}}}^{t} u_a(\hat{t})d\hat{t} + x_0 \tag{3.72}$$

along which the solution is constant. Here, $x_0 = \tilde{x}(0)$ is a constant resembling the starting point of the curve. Thus, we can determine the exact solution given an initial value $u_0 : \mathbb{R} \to \mathbb{R}$ by

$$u(x,t) = u_0(x - \int_{t_{\text{start}}}^{t} u_a(\hat{t})d\hat{t}). \tag{3.73}$$

In Figure 3.8, we have illustrated characteristic curves for a specific choice of u_a.

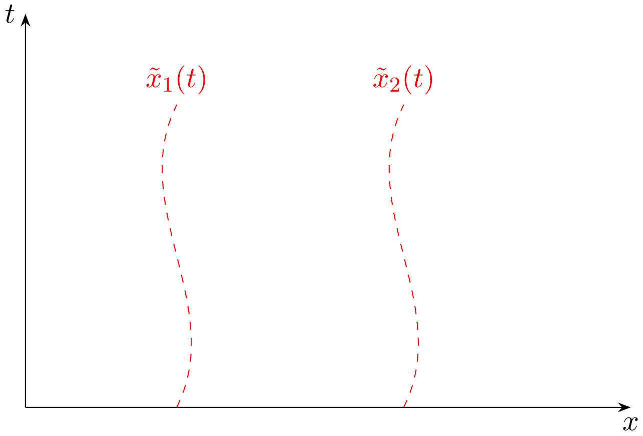


Figure 3.8: Illustration of two characteristic curves $\tilde{x}_1(t), \tilde{x}_2(t)$ (*red dashed lines*) for the linear space-time advection equation (3.66) where $u_a(t) = 3t^2 - 3t + 0.5$, resulting in the curve $\int_{t_{\text{start}}}^{t} u_a(\hat{t})d\hat{t} = t^3 - \frac{3}{2}t^2 + \frac{1}{2}t$.

The linear space-time advection equation serves as an 'easy' model problem for evaluating implicit shock tracking methods. By introducing discontinuities into the initial value, these discontinuities propagate in time along characteristic curves. This leads to the formation of curves where the solution becomes discontinuous, which are sought to be fitted by the implicit shock tracking method being investigated. In Section 5.1.1, we will present two specific test cases used to validate our XDG-IST method. For other examples in both two-dimensional and three-dimensional contexts, we refer the reader to the works by Corrigan et al. (2019a), Zahr et al. (2020), and Huang and Zahr (2022).

3.4.2 One-dimensional space-time Burgers equation

Consider the space-time formulation of the 1D Burgers equation, which can be expressed as

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0 \quad \text{in } \Omega = \Omega_x \times \mathcal{T}, \tag{3.74}$$

where $\Omega_x \subset \mathbb{R}$ and $u : \Omega \to \mathbb{R}$ is a velocity. By setting the conserved variables U and the corresponding physical flux F to

$$U = (u), \quad F : U \mapsto (\frac{1}{2}u^2 \;\; u), \tag{3.75}$$

the equation can be linked to the general form (3.1) (here $d = 2$ and $m = 1$), considering a space-time setting (see Remark 2). Like for the advection equation (3.4.1), boundary conditions are applied using solely an appropriate exact solution and the smooth upwind numerical flux (3.44) is employed.

While the linear advection equation discussed in Section 3.4.1 is relatively straightforward, the Burgers equation mentioned in Section 3.4.2 introduces non-linearity. Despite the possibility of constructing exact solutions for most configurations, this non-linearity poses a greater challenge for implicit shock tracking methods. Additionally, it allows for scenarios where a discontinuity emerges after a certain period, even if the initial value is continuous. Examples of such scenarios can be found in the works by Corrigan et al. (2019a), Zahr et al. (2020), and Huang and Zahr (2022).

3.4.3 Two-dimensional steady Euler equations

The steady two-dimensional Euler equations are obtained from their unsteady counterpart, already introduced in (2.4), by discarding the time derivative $\partial U/\partial t$ and write

$$\frac{\partial F_x^{\text{eul}}(U)}{\partial x} + \frac{\partial F_y^{\text{eul}}(U)}{\partial y} = 0, \tag{3.76}$$

where the conserved quantities $U : \Omega \to \mathbb{R}^4$ and the convective fluxes $F_x^{\text{eul}} : \mathbb{R}^4 \to \mathbb{R}^4$ and $F_y^{\text{eul}} : \mathbb{R}^4 \to \mathbb{R}^4$ are given by the following expressions

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}, \quad F_x^{\text{eul}}(U) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ u(\rho E + p) \end{pmatrix}, \quad F_y^{\text{eul}}(U) = \begin{pmatrix} \rho v \\ \rho v u \\ \rho v^2 + p \\ v(\rho E + p) \end{pmatrix}. \tag{2.5 repeated}$$

In order to link the equation system to the general form (3.1), we introduce the physical flux as

$$F : U \mapsto \begin{pmatrix} F_x^{\text{eul}}(U) & F_y^{\text{eul}}(U) \end{pmatrix}, \tag{3.77}$$

noting that $d = 2$ and $m = 4$. In the corresponding BoSSS module XESF many variants of numerical fluxes described above are implemented: the central flux (3.42), the Godunov flux (3.47), the HLLC flux (3.48), and the Roe flux (3.56). Furthermore, it is possible to choose different numerical fluxes for inner non-interface edges $\Gamma \backslash \mathfrak{I}_s \backslash \mathfrak{I}_b$, edges lying on the shock interface $\mathfrak{I}_s$ and edges lying on the immersed boundary interface $\mathfrak{I}_b$. For most test case considered in this work, the following combination is employed:

- The HLLC flux is used for non-interface edges while directly imposing boundary conditions for edges corresponding to the domain boundary (see next paragraph for details).
- The Godunov flux is utilized for edges corresponding to the shock interface $\mathfrak{I}_s$.
- An adiabatic slip wall condition is applied for the edges corresponding to the immersed boundary interface $\mathfrak{I}_b$ utilizing the HLLC flux.

Using this combination, the physical correctness of the Godunov flux ensures consistency with the Rankine-Hugoniot conditions on the shock interface (see Equation 3.40), while the rest of the edges are handled by the HLLC flux for computational efficiency.

Boundary conditions Boundary conditions are introduced in a weak sense by prescribing the external state U^{out} on external edges Γ_{ext}, needed for the evaluation of the numerical fluxes in Equation 3.27. The prescription is done in accordance with the treatment of boundary conditions for compressible flows for DG methods (Bassi and Rebay, 1997; Hartmann and Houston, 2008; Mengaldo et al., 2014) and follows the presentation by Geisenhofer (2021).

The boundary conditions are mainly characterized by the orientation of characteristics of the 1D Euler equations which are obtained by restriction to the points normal to the edge. The three characteristics are $u_n - a, u_n$, and $u_n + a$, where $\vec{n}$ denotes the unit inward normal, and $u_n = \vec{u} \cdot \vec{n}$ denotes the velocity in edge-normal direction. Depending on the numeric value of u_n and its sign, some characteristics may move into our out of the domain, having implications on the type of boundary condition that must be imposed.

The external states $U^{\text{out}}(U_\infty, U^{\text{in}})$ are defined in terms of (prescribed) free-stream conditions U_∞ and the inner values U^{in} on the edges of boundary cells. The boundary conditions considered in this work are:

- *Supersonic inlet* $(u_n > a)$*:* all three characteristics $(u_n - a > 0, u_n > 0, u_n + a > 0)$ point into the domain. Thus, the free stream conditions are imposed as a Dirichlet boundary condition, i.e.,

$$U^{\text{out}} = U_\infty. \tag{3.78}$$

- *Supersonic outlet* $(-u_n > a)$*:* all three characteristics $(u_n - a < 0, u_n < 0, u_n + a < 0)$ point out of the domain. Thus, no information travels into the domain and free boundary conditions are applied by choosing the inner values

$$U^{\text{out}} = U^{\text{in}}. \tag{3.79}$$

- *Subsonic inlet* $(0 < u_n < a)$*:* two characteristics $(u_n > 0, u_n + a > 0)$ point into the domain, one characteristic $(u_n - a < 0)$ leaves the domain. Here, free-stream conditions are imposed except for the pressure where the inner value is used

$$U^{\text{out}} = \begin{pmatrix} \rho_\infty \\ \rho_\infty \vec{u}_\infty \\ \frac{p^{\text{in}}}{\hat{\gamma}-1} + \frac{1}{2}\rho_\infty(\vec{u}_\infty \cdot \vec{u}_\infty) \end{pmatrix}. \tag{3.80}$$

- *Subsonic outlet* $(0 < -u_n < a)$*:* two characteristics $(u_n < 0, u_n - a < 0)$ point out of the domain, one characteristic $(u_n - a < 0)$ points into the domain. Here, free-stream conditions are applied for the pressure and inner values are used for the other quantities.
- *Adiabatic slip-wall* $(u_n = 0)$*:* defining a wall boundary condition in the context of the Euler equations (2.4), necessitates allowing a tangential velocity. This slip wall boundary condition is defined by introducing a mirrored velocity by $\vec{u}^{\text{mir}} = \vec{u}^{\text{in}} - 2(\vec{u}^{\text{in}} \cdot \vec{n})\vec{n}$, which imposes the same tangential component u_t in the external state as in the interior, i.e., $u_t^{\text{in}} = u_t^{\text{out}}$, while for the normal component we have ${u_n}^{\text{in}} = -{u_n}^{\text{in}}$ (Mengaldo et al., 2014). The adiabatic slip-wall boundary condition is thus expressed as:

$$U^{\text{out}} := \begin{pmatrix} \rho^{\text{in}} \\ \rho^{\text{in}}\left(\vec{u}^{\text{in}} - 2(\vec{u}^{\text{in}} \cdot \vec{n})\vec{n}\right) \\ \rho^{\text{in}} E^{\text{in}} \end{pmatrix}. \tag{3.81}$$

3.4.4 One-dimensional space-time Euler equations

The 1D space-time Euler equations can be obtained from the 2D Euler equations introduced in (2.4), by discarding the derivative in y-direction and considering a one-dimensional spatial domain $\Omega_x \subset \mathbb{R}$. They write

$$\frac{\partial U}{\partial t} + \frac{\partial F_x^{\text{eul}}(U)}{\partial x} = 0 \quad \text{in } \Omega = \Omega_x \times \mathcal{T}, \tag{3.82}$$

where the conserved quantities $U : \Omega \to \mathbb{R}^3$ and the convective flux $F_x^{\text{eul}} : \mathbb{R}^3 \to \mathbb{R}^3$ are redefined as the following expressions:

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho E \end{pmatrix}, \quad F_x^{\text{eul}}(U) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ u(\rho E + p) \end{pmatrix}. \tag{3.83}$$

In order to link the equation system to the general form (3.1), we introduce the space-time physical flux (in accordance with Remark 2) as

$$F : U \mapsto \begin{pmatrix} F_x^{\text{eul}}(U) & U \end{pmatrix}, \tag{3.84}$$

noting that in this case $d = 2$ and $m = 3$.

Boundary conditions For the 1D space-time Euler equations, we employ rectangular domains $\Omega = [x_{\text{start}}, x_{\text{end}}] \times [t_{\text{start}}, t_{\text{end}}]$ and describe the boundary conditions for their left, right, top and bottom boundaries. Here, for the left and right boundaries

$$\partial\Omega_L := \{(x_{\text{start}}, t) \mid t \in \mathcal{T}\}, \quad \partial\Omega_R := \{(x_{\text{end}}, t) \mid t \in \mathcal{T}\} \tag{3.85}$$

the same boundary conditions can be applied as for the two-dimensional steady Euler equations (see Section 3.4.3), discarding the y-momentum and allowing for a time-dependency. Conversely, the top and bottom boundaries

$$\partial\Omega_B := \{(x, t_{\text{start}}) \mid x \in [x_{\text{start}}, x_{\text{end}}]\}, \quad \partial\Omega_T := \{(x, t_{\text{end}}) \mid x \in [x_{\text{start}}, x_{\text{end}}]\} \tag{3.86}$$

are more restricted. On the bottom boundary $\partial\Omega_B$, an initial condition, usually in primitive variables $p_\infty, u_\infty, \rho_\infty : [x_{\text{start}}, x_{\text{end}}] \to \mathbb{R}$, must be prescribed in the form of a Dirichlet boundary. It can thus be regarded as a supersonic inflow boundary condition. The top boundary $\partial\Omega_T$ corresponds to the solution U at the end time t_{end}, which is usually unknown. Thus, here a free boundary condition is applied, which is essentially the supersonic outflow boundary condition discussed for the steady case.

4 Implicit XDG shock tracking

In this chapter, a novel implicit XDG shock tracking (XDG-IST) method is presented, drawing from and expanding the publication by Vandergrift and Kummer (2024). The approach integrates techniques from classical mesh-based implicit shock tracking, as introduced by Zahr and Persson (2018) and Corrigan et al. (2019a), and extended discontinuous Galerkin (XDG) methods (see Section 3.2.2), which are both particularly suited for handling partial differential equations (PDEs) with discontinuities. To avoid nonlinear stabilization through shock capturing (Müller et al., 2017; Geisenhofer et al., 2019), a constrained optimization problem is solved that aligns the zero iso-contour of a level set function with the discontinuities, hence allowing for their accurate representation in the XDG approximation spaces. The optimization problem is solved by employing an inexact, regularized, sequential quadratic programming (SQP) solver together with a line search algorithm. Moreover, a range of additional robustness strategies is devised (Huang and Zahr, 2022) to improve solver convergence. The solver framework presented in this chapter is linked to its actual implementation in the open-source software bounded support spectral solver (BoSSS) by utilizing hyperlinks in the electronic version of this thesis.

The structure of this chapter is as follows: The formulation of the optimization problem is examined in Section 4.1 and the SQP solver is described in Section 4.2. In Section 4.3, we present measures to enhance the robustness of the method. Further, details on the initialization and termination strategies of the solver are found in Section 4.4 and the full algorithm is summarized in Section 4.5. Lastly, different level set discretization approaches employed within the method are discussed in Section 4.6.

4.1 Shock tracking formulation

In this section, we present the formulation of the XDG-IST method. In Section 4.1.1, we start by introducing the nonlinear constrained optimization problem at its heart and examine three different possibilities of defining an objective function. Then, we describe the corresponding first-order optimality conditions employing Lagrange multipliers, defining first-order solutions to the optimization problem in Section 4.1.2.

The XDG-IST method, as described by Vandergrift and Kummer (2024), is a high-order technique that simultaneously computes both the coefficients $\boldsymbol{\varphi}$ of the discrete level set φ_s representing the shock and the coefficients $\boldsymbol{u}$ defining the discrete solution U_h of the conservation laws. By doing so, the XDG-IST method aligns the shock-interface represented by $\mathfrak{I}_s = \{\varphi_s = 0\}$ (see Section 3.2.2) with discontinuities. This process is accomplished using a full-space optimization formulation, where the optimization variables consist of the respective coefficients $\boldsymbol{\varphi}$ (level set) and $\boldsymbol{u}$ (flow solution). The method is formulated as a constrained optimization problem

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{u},\boldsymbol{\varphi}) \\ \text{subject to} \quad & \boldsymbol{r}(\boldsymbol{u},\boldsymbol{\varphi}) = 0, \end{aligned} \tag{4.1}$$

where $\boldsymbol{r}$ is the algebraic residual defined in (3.29) and $f : \mathbb{R}^{N_{\boldsymbol{u}}} \times \mathbb{R}^{N_{\boldsymbol{\varphi}}} \to \mathbb{R}$ an objective function (defined in the next section). It is assumed by construction that solutions $(\boldsymbol{u},\boldsymbol{\varphi})$ of (4.1) feature a discontinuity-aligned interface $\mathfrak{I}_{\mathfrak{s}}$ and simultaneously satisfy the discretized system of conservation laws $\boldsymbol{r}(\boldsymbol{u},\boldsymbol{\varphi}) = 0$.

In contrast to the high-order implicit shock tracking (HOIST) method (Zahr et al., 2020), which serves as an inspiration for the XDG shock tracking method, we employ a fixed background mesh. Hence, the dependency in the mesh nodes $\boldsymbol{x}$ for the algebraic residuals $\boldsymbol{r}$ and $\boldsymbol{R}$ is dropped throughout this chapter. Also, we assume no domain deformations, i.e., $\mathcal{G} = \boldsymbol{I}$. Thus, the reference domain and the physical domain are the same, and all transformed quantities equal the non-transformed quantities (see Remark 3).

4.1.1 The objective function

Addressing the central optimization problem of this study, the choice of the objective function is paramount, particularly for aligning the interface described by φ_s. This research explores three distinct strategies for choosing the function f. Each of these can be written within a least squares framework, that is, f is of the form

$$f(\boldsymbol{u},\boldsymbol{\varphi}) := \frac{1}{2}\boldsymbol{F}(\boldsymbol{u},\boldsymbol{\varphi})^T \boldsymbol{F}(\boldsymbol{u},\boldsymbol{\varphi}), \tag{4.2}$$

where $\boldsymbol{F} : \mathbb{R}^{N_{\boldsymbol{u}}} \times \mathbb{R}^{N_{\boldsymbol{\varphi}}} \mapsto \mathbb{R}^{N_{\boldsymbol{F}}}$ for some $N_{\boldsymbol{F}} \in \mathbb{N}$. We now proceed to describe each strategy separately, starting with the enriched residual.

Enriched residual The first strategy adopts the methodology by Zahr et al. (2020) utilizing the enriched residual ($\boldsymbol{F} = \boldsymbol{R}$) as detailed in Equation (3.32). This residual is characterized by trial functions that possess a polynomial degree of $P + 1$, leading to the following objective function

$$f_{\mathrm{ER}}(\boldsymbol{u},\boldsymbol{\varphi}) := \frac{1}{2}\boldsymbol{R}(\boldsymbol{u},\boldsymbol{\varphi})^T \boldsymbol{R}(\boldsymbol{u},\boldsymbol{\varphi}). \tag{4.3}$$

Within this context, the enriched residual $\boldsymbol{R}$ serves as a nuanced error indicator, exhibiting high sensitivity to solutions with unfitted discontinuities. This attribute is particularly underlined by Zahr and Persson (2018) and the effectiveness of this approach has been demonstrated in the mesh-based HOIST framework.

Near-band enriched residual The second approach utilizes an enriched residual confined to elements K_e in the near band (3.23), that is, $\boldsymbol{F} = \boldsymbol{R}_{\mathrm{NB}}$, where

$$\boldsymbol{R}_{\mathrm{NB}} : \mathbb{R}^{N_{\boldsymbol{u}}} \times \mathbb{R}^{N_{\boldsymbol{\varphi}}} \to \mathbb{R}^{N_{\boldsymbol{u}}^{\star}}, \qquad \boldsymbol{R}_{\mathrm{NB}} : (\boldsymbol{u},\boldsymbol{\varphi}) \mapsto \boldsymbol{R}_{\mathrm{NB}}(\boldsymbol{u},\boldsymbol{\varphi}). \tag{4.4}$$

The residual $\boldsymbol{R}_{\text{NB}}$ can be seen as a sub-vector of $\boldsymbol{R}$ which is confined to components corresponding to test functions supported within the near band of φ_s, denoted as $\mathcal{K}_h^{cc,1}(\varphi_s)$ and detailed in Equation (3.23). Here, $N_{\boldsymbol{u}}^{\star}$ represents the dimension of the discontinuous Galerkin (DG) space $\mathcal{V}_h^{P+1,m}(\mathcal{K}_h^{cc,1}(\varphi_s))$. The corresponding objective function is expressed as

$$f_{\text{NB}}(\boldsymbol{u}, \boldsymbol{\varphi}) := \frac{1}{2}\boldsymbol{R}_{\text{NB}}(\boldsymbol{u}, \boldsymbol{\varphi})^T \boldsymbol{R}_{\text{NB}}(\boldsymbol{u}, \boldsymbol{\varphi}). \tag{4.5}$$

The justification for limiting the objective function to only include elements K_e within the near band is based on the observation that elements $K_{e'}$ outside the near band $\mathcal{K}_h^{cc,1}(\varphi_s)$ region exhibit negligible sensitivity to minor modifications in the level set. This phenomenon arises because slight adjustments to the level set's degrees of freedom (DOFs), which alter the interface position, predominantly impact the integration domains of cut-cells and their immediate neighbors, rather than affecting distant elements $K_{e'}$.

Weak form from Rankine-Hugoniot conditions The third method for defining the objective function leverages the Rankine-Hugoniot jump conditions for stationary shocks, expressed as

$$[\![F(U) \cdot \vec{n}_\Gamma]\!] = 0, \tag{4.6}$$

where F represents the physical flux defined in Equation (3.1), U denotes the vector of conserved quantities, and $\vec{n}_\Gamma$ is the unit normal on cell edges. To construct a residual based on these conditions, we first establish a weak form similar to the discretization approach for the transformed system of conservation laws. This involves multiplying the Rankine-Hugoniot conditions by test functions V and integrating across all cell edges

$$\oint_\Gamma [\![F(U) \cdot \vec{n}_\Gamma]\!] \, V \, dS = 0. \tag{4.7}$$

By substituting a basis from both the trial space ($\mathcal{V}_h^{P,m,X}$) and an enriched test space ($\mathcal{V}_h^{P+1,m,X}$) into this weak form, we discretize it into a system of nonlinear algebraic equations in residual form, represented as

$$\boldsymbol{R}_{\text{RH}}(\boldsymbol{u}, \boldsymbol{\varphi}) = 0. \tag{4.8}$$

It's important to note that the dependency on $\boldsymbol{\varphi}$ is implicit, as mentioned in Remark 7, and dependencies on mesh nodes $\boldsymbol{x}$ and φ_b are not considered. The objective function derived from this residual is defined by

$$f_{\text{RH}}(\boldsymbol{u}, \boldsymbol{\varphi}) := \frac{1}{2}\boldsymbol{R}_{\text{RH}}(\boldsymbol{u}, \boldsymbol{\varphi})^T \boldsymbol{R}_{\text{RH}}(\boldsymbol{u}, \boldsymbol{\varphi}), \tag{4.9}$$

and it favors solutions satisfying the Rankine-Hugoniot conditions for stationary shocks across all cell edges (for cut-cells and non-cut-cells). This approach mirrors strategies used in mesh-based implicit shock tracking, such as the moving discontinuous Galerkin method with interface condition enforcement (MDG-ICE). It has been successfully applied to various scenarios (Corrigan et al., 2019a; Corrigan et al., 2019b; Corrigan et al., 2019c; Kercher and Corrigan, 2021; Kercher et al., 2021).

Choosing the objective function f_{RH} for the optimization problem (4.1) is rooted more firmly in physics compared to f_{ER} and f_{NB} by weakly enforcing the Rankine-Hugoniot jump conditions

(4.6). However, the extension to viscous problems is more straight-forward for the enriched residual based approaches, only requiring the treatment of second-order terms, which has been well-established in the DG setting (Zahr et al., 2020).

Remark 10. *Within mesh-based implicit shock fitting, in particular the HOIST method, the objective function is typically augmented by a term that penalizes undesirable mesh configurations, as detailed in Section 6.1.1. This penalization is based on comparing each mesh element against an idealized element standard, with penalties applied for deviations from this ideal. In contrast to the HOIST method, no reasonable equivalent penalizing undesired interface positions was found for the XDG framework, while we experimented with utilizing the curvature of the level set as a measure.*

4.1.2 First-order optimality system

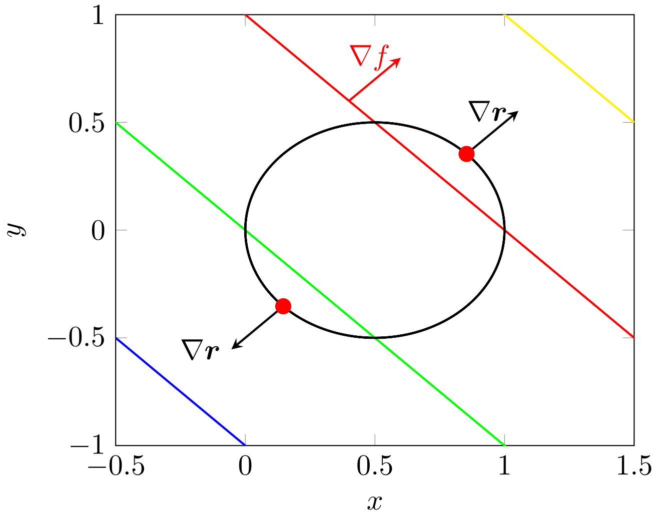


Figure 4.1: Illustration of a constraint optimization problem with iso-contours of an exemplary objective function $f(x,y) = x + y$ (*thick colored lines*) and the feasible set given by $\boldsymbol{r}(x,y) = (x-0.5)^2 + y^2 - 0.5^2 = 0$ (*thick black line*). The gradients of $\boldsymbol{r}$ at any point on the circle are normal to the circle, indicating the direction of fastest increase of $\boldsymbol{r}$. To stay on the constraint surface, one must move in a direction tangent to the circle. Two first-order solutions can be found visually, corresponding to the maximal and the minimal value (red points) of the corresponding constraint optimization problem. At these points, the gradients are parallel, satisfying $\nabla f = \boldsymbol{\lambda} \nabla \boldsymbol{r}$ for some $\boldsymbol{\lambda} \in \mathbb{R}$.

Non-linear constrained optimization problems, such as (4.1), are typically addressed by introducing the Lagrange functional $\mathcal{L} : \mathbb{R}^{N_u} \times \mathbb{R}^{N_\varphi} \times \mathbb{R}^{N_u} \to \mathbb{R}$, expressed as

$$\mathcal{L}(\boldsymbol{u}, \boldsymbol{\varphi}, \boldsymbol{\lambda}) = f(\boldsymbol{u}, \boldsymbol{\varphi}) + \boldsymbol{\lambda}^\top \boldsymbol{r}(\boldsymbol{u}, \boldsymbol{\varphi}). \tag{4.10}$$

The first-order optimality conditions are as follows: A pair $(\boldsymbol{u}^*, \boldsymbol{\varphi}^*) \in \mathbb{R}^{N_u} \times \mathbb{R}^{N_\varphi}$ is considered a first-order solution if there exists a $\boldsymbol{\lambda}^* \in \mathbb{R}^{N_u}$ such that

$$\nabla \mathcal{L}\left(\boldsymbol{u}^*, \boldsymbol{\varphi}^*, \boldsymbol{\lambda}^*\right) = 0. \tag{4.11}$$

Together with the constraint $\boldsymbol{r}(\boldsymbol{u}^*, \boldsymbol{\varphi}^*) = 0$, these are known as the Karush-Kuhn-Tucker (KKT) conditions (Kuhn and Tucker, 1951). They effectively imply that the objective gradient ∇f

must be a linear combination of the gradients of the residual when evaluated at a first-order solution. We illustrate this in Figure 4.1 for a simple and imaginary example (i.e., where f and $\boldsymbol{r}$ only depend on two variables x, y).

4.2 Sequential quadratic programming solver

In this section, we examine the iterative SQP solver for solving the optimization problem (4.1), following it's presentation by Vandergrift and Kummer (2024). Solutions to (4.1) are sought as solutions to the first-order optimality system obtained by the KKT conditions. The nonlinear optimality system is then linearized in each iteration to obtain the next solution update, employing a quasi-Newton approach (Section 4.2.1) and featuring a (regularized) Levenberg-Marquardt approximation of the Hessian involved (Section 4.2.2). Moreover, the SQP solver is modified to utilize an inexact line search method (Section 4.2.3) for globalization and to employ an adaptive regularization strategy (Section 4.2.4). Further, we discuss how newborn cut-cells are handled that arise when changing the interface position (Section 4.2.5).

4.2.1 Linearized optimality system

There are two ways of deriving the SQP method for finding solutions to the first-order optimality system (4.11). The first approach comes from applying Newton's method to solve Equation 4.11 and is described in the following. First, let us introduce a single variable for the coefficients by

$$\boldsymbol{z} := \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{\varphi} \end{pmatrix} \in \mathbb{R}^{N_{\boldsymbol{z}}}, \quad N_{\boldsymbol{z}} = N_{\boldsymbol{u}} + N_{\boldsymbol{\varphi}}. \tag{4.12}$$

Newton's method Applying Newton's method to (4.11), iteratively computes a sequence $(\boldsymbol{z}_0, \boldsymbol{\lambda}_0), (\boldsymbol{z}_1, \boldsymbol{\lambda}_1), ..., (\boldsymbol{z}_k, \boldsymbol{\lambda}_k)$ which eventually converges to a solution $(\boldsymbol{z}^*, \boldsymbol{\lambda}^*) = (\boldsymbol{u}^*, \boldsymbol{\varphi}^*, \boldsymbol{\lambda}^*)$. The solution is updated via $(\boldsymbol{z}_{k+1}, \boldsymbol{\lambda}_{k+1}) := (\boldsymbol{z}_k, \boldsymbol{\lambda}_k) + (\Delta\boldsymbol{z}_k, \Delta\boldsymbol{\lambda}_k)$. To obtain the next solution update $(\Delta\boldsymbol{z}_k, \Delta\boldsymbol{\lambda}_k)$, one typically solves the linear system

$$\boldsymbol{H}_{\mathcal{L}}(\boldsymbol{z}_k, \boldsymbol{\lambda}_k) \begin{pmatrix} \Delta\boldsymbol{z}_k \\ \Delta\boldsymbol{\lambda}_k \end{pmatrix} = -\nabla\mathcal{L}(\boldsymbol{z}_k, \boldsymbol{\lambda}_k) \tag{4.13}$$

in every Newton iteration. The system (4.13) features the Hessian $\boldsymbol{H}_{\mathcal{L}}(\boldsymbol{z}_k, \boldsymbol{\lambda}_k)$ of the Lagrangian $\mathcal{L}$ evaluated at the iterate $(\boldsymbol{z}_k, \boldsymbol{\lambda}_k)$ and it can be expressed as

$$\boldsymbol{H}_{\mathcal{L}}(\boldsymbol{z}_k, \boldsymbol{\lambda}_k) = \begin{pmatrix} \boldsymbol{H}_{\mathcal{L}}^{\boldsymbol{u}\boldsymbol{\varphi}}(\boldsymbol{z}_k, \boldsymbol{\lambda}_k) & \boldsymbol{J}(\boldsymbol{z}_k)^T \\ \boldsymbol{J}(\boldsymbol{z}_k) & 0 \end{pmatrix}, \tag{4.14a}$$

$$\text{where } \boldsymbol{H}_{\mathcal{L}}^{\boldsymbol{u}\boldsymbol{\varphi}}(\boldsymbol{z}_k, \boldsymbol{\lambda}_k) := \boldsymbol{H}_f(\boldsymbol{z}_k) - \sum_{j=1}^{N_{\boldsymbol{z}}} (\boldsymbol{\lambda}_k)_j \boldsymbol{H}_{\boldsymbol{r}^j}(\boldsymbol{z}_k), \tag{4.14b}$$

and where $\boldsymbol{H}_f$ denotes the Hessian of the objective function f, $\boldsymbol{H}_{\boldsymbol{r}^j}$ denote Hessians for each residual component $\boldsymbol{r}^j$, and $\boldsymbol{J}$ denotes the full Jacobian matrix of the residual $\boldsymbol{r}$

$$\boldsymbol{J} : \boldsymbol{z} \mapsto \left[\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}(\boldsymbol{u}, \boldsymbol{\varphi}) \quad \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{\varphi}}(\boldsymbol{u}, \boldsymbol{\varphi}) \right], \tag{4.15}$$

all evaluated at the iterate $\boldsymbol{z}_k$.

Sequential quadratic programming The linear system (4.13) is then reformulated to immediately solve for the Lagrange multiplier $\boldsymbol{\lambda}_{k+1}$. Examining the upper block of the system

$$\boldsymbol{H}_{\mathcal{L}}^{\boldsymbol{u}\boldsymbol{\varphi}}(\boldsymbol{z}_k, \boldsymbol{\lambda}_k)\Delta \boldsymbol{z}_k + \boldsymbol{J}(\boldsymbol{z}_k)^T \underbrace{(\Delta \boldsymbol{\lambda}_k + \boldsymbol{\lambda}_k)}_{\boldsymbol{\lambda}_{k+1}} = -\nabla f(\boldsymbol{z}_k), \tag{4.16}$$

allows for the following reformulation

$$\begin{pmatrix} \boldsymbol{H}_{\mathcal{L}}^{\boldsymbol{u}\boldsymbol{\varphi}}(\boldsymbol{z}_k, \boldsymbol{\lambda}_k) & \boldsymbol{J}(\boldsymbol{z}_k)^T \\ \boldsymbol{J}(\boldsymbol{z}_k) & 0 \end{pmatrix} \begin{pmatrix} \Delta \boldsymbol{z}_k \\ \boldsymbol{\lambda}_{k+1} \end{pmatrix} = - \begin{pmatrix} \nabla f(\boldsymbol{z}_k)^T \\ \boldsymbol{r}(\boldsymbol{z}_k) \end{pmatrix}. \tag{4.17}$$

Alternatively, the same linear system is obtained when computing the solution of the following quadratic program

$$\begin{aligned} \underset{\Delta \boldsymbol{z} \in \mathbb{R}^{N_{\boldsymbol{z}}}}{\text{minimize}} \quad & \nabla f(\boldsymbol{z}_k)^T \Delta \boldsymbol{z}_k + \frac{1}{2} \Delta \boldsymbol{z}_k^T \boldsymbol{B}^{(k)} \Delta \boldsymbol{z}_k \\ \text{subject to} \quad & \boldsymbol{r}(\boldsymbol{z}_k) + \boldsymbol{J}(\boldsymbol{z}_k) \Delta \boldsymbol{z}_k = 0, \end{aligned} \tag{4.18}$$

if $\boldsymbol{B}^{(k)} = \boldsymbol{H}_{\mathcal{L}}^{\boldsymbol{u}\boldsymbol{\varphi}}(\boldsymbol{z}_k, \boldsymbol{\lambda}_k)$ is chosen. Constructing the corresponding KKT conditions of (4.18) leads to the same system as in (4.17), which means that the above approach can be described as a process where quadratic problems are solved sequentially. This motivates the name *sequential quadratic programming* (SQP).

4.2.2 Approximation of the Hessian

Computing the Hessians $\boldsymbol{H}_f, \boldsymbol{H}_{\boldsymbol{r}^j}$ that are part of $\boldsymbol{H}_{\mathcal{L}}^{\boldsymbol{u}\boldsymbol{\varphi}}$ in Equation (4.14b) is a very challenging task because it involves second-order derivatives. These derivatives are usually very expensive to compute and commonly not available in most computational fluid dynamics (CFD) solvers. To circumvent this issue, we follow Zahr et al. (2020) and introduce a Levenberg-Marquardt approximation (see the work by Dennis and Schnabel (1996) for reference) to $\boldsymbol{H}_{\mathcal{L}}^{\boldsymbol{u}\boldsymbol{\varphi}}$. Specifically, it is replaced with a partially regularized approximation

$$\boldsymbol{H}_{\mathcal{L}}^{\boldsymbol{u}\boldsymbol{\varphi}}(\boldsymbol{z}_k) \approx \boldsymbol{B}^{(k)} = \boldsymbol{B}(\boldsymbol{z}_k, \gamma) := \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{z}}(\boldsymbol{z}_k)^T \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{z}}(\boldsymbol{z}_k) + \gamma \begin{pmatrix} 0 & 0 \\ 0 & \boldsymbol{I}_{\varphi} \end{pmatrix}, \tag{4.19}$$

where $\boldsymbol{I}_{\varphi}$ is the $(N_{\varphi} \times N_{\varphi})$-identity matrix, $\gamma \in \mathbb{R}_+$ a regularization parameter chosen adaptively (see Section 4.2.4), and $\boldsymbol{F} \in \{\boldsymbol{R}, \boldsymbol{R}_{\text{RH}}, \boldsymbol{R}_{\text{NB}}\}$ a residual function defining the objective function (Section 4.1.1). We regularize the components corresponding to DOFs of the level set since the resulting sub-matrix (see Equation 4.21c) is often singular. For instance, this may be the case for entries of $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{z}}$ corresponding to cells far away from the interface, which are insensitive to small level set perturbations. However, it can be assumed that the part belonging to the flow solution is invertible, hence no regularization is applied to its components (Zahr et al., 2020). Additionally, the remaining Hessians $\boldsymbol{H}_{\boldsymbol{r}^j}$ (only containing second-order derivatives) are entirely discarded. It can be assumed that, far away from the shock, they are dominated by the first-order terms. This observation is discussed by Zahr et al. (2020) in more detail.

The approximation can be written in more detail by

$$\boldsymbol{H}_{\mathcal{L}}^{\boldsymbol{u}\boldsymbol{\varphi}}(\boldsymbol{z},\boldsymbol{\lambda}) \approx \boldsymbol{B}(\boldsymbol{z},\gamma) := \begin{pmatrix} \boldsymbol{B}_{\boldsymbol{u}\boldsymbol{u}}(\boldsymbol{z}) & \boldsymbol{B}_{\boldsymbol{u}\boldsymbol{\varphi}}(\boldsymbol{z}) \\ \boldsymbol{B}_{\boldsymbol{u}\boldsymbol{\varphi}}(\boldsymbol{z})^T & \boldsymbol{B}_{\boldsymbol{\varphi}\boldsymbol{\varphi}}(\boldsymbol{z},\gamma) \end{pmatrix}, \tag{4.20}$$

where the sub-blocks are defined by

$$\boldsymbol{B}_{\boldsymbol{u}\boldsymbol{u}}(\boldsymbol{z}) = \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}(\boldsymbol{z})^T \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}(\boldsymbol{z}) \tag{4.21a}$$

$$\boldsymbol{B}_{\boldsymbol{u}\boldsymbol{\varphi}}(\boldsymbol{z}) = \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}(\boldsymbol{z})^T \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\varphi}}(\boldsymbol{z}) \tag{4.21b}$$

$$\boldsymbol{B}_{\boldsymbol{\varphi}\boldsymbol{\varphi}}(\boldsymbol{z},\gamma) = \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\varphi}}(\boldsymbol{z})^T \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\varphi}}(\boldsymbol{z}) + \gamma \boldsymbol{I}_{\boldsymbol{\varphi}}. \tag{4.21c}$$

The final linear system to be solved is

$$\begin{pmatrix} \boldsymbol{B}(\boldsymbol{z}_k,\gamma) & \boldsymbol{J}^T(\boldsymbol{z}_k) \\ \boldsymbol{J}(\boldsymbol{z}_k) & 0 \end{pmatrix} \begin{pmatrix} \Delta \boldsymbol{z}_k \\ \boldsymbol{\lambda}_{k+1} \end{pmatrix} = - \begin{pmatrix} \nabla f(\boldsymbol{z}_k)^T \\ \boldsymbol{r}(\boldsymbol{z}_k) \end{pmatrix}, \tag{4.22}$$

and it governs the next optimization step $\Delta \boldsymbol{z}_k$.

4.2.3 Inexact line search globalization

After computing the solution update $\Delta \boldsymbol{z}_k$ and the Lagrange multiplier $\boldsymbol{\lambda}_{k+1}$ from (4.22), the next SQP iterate is usually computed via

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \alpha_k \Delta \boldsymbol{z}_k, \tag{4.23}$$

where $\alpha_k \in (0,1]$ is a step length. It's important to note that setting α_k to 1, a strategy often employed in non-globalized SQP methods, usually does not yield the best outcomes. It potentially leads to sub-optimal convergence rates or even hinders convergence in the first place. The underlying reason is that the direction of the solution update $\Delta \boldsymbol{z}_k$ is derived from a local quadratic model, which, in turn, is an approximation of the original problem's second-order Taylor expansion around the current iterate. While theoretically, this update direction suggests a descent path, meaning that moving slightly in this direction is expected to reduce the objective function's value, the local nature of the Taylor approximation implies that its effectiveness diminishes with larger step sizes. In other words, choosing a larger α_k entails an inaccurate Taylor expansion and does not guarantee a decrease in the objective function, which may lead to sub-optimal outcomes.

A simple approach to find suitable step lengths α_k is named *inexact line search* or *backtracking* and is employed in this work (Nocedal and Wright, 2006). This strategy iteratively searches for the largest step length from a discrete set

$$\alpha_k \in \{1 = \tau^0, \tau^{-1}, \ldots, \alpha_{\text{min}}\}, \tag{4.24}$$

which satisfies the *condition of sufficient decrease*, given by

$$M_k(\alpha_k) \leqslant M_k(0) + \alpha_k \beta M_k'(0), \tag{4.25}$$

where $M_k : \mathbb{R} \to \mathbb{R}$ is a merit function assessing the solution's quality, $\tau \in (0,1)$ is the factor by which the step length is successively reduced, and $\beta > 0$ is a relaxation parameter to ensure flexibility in the search process.

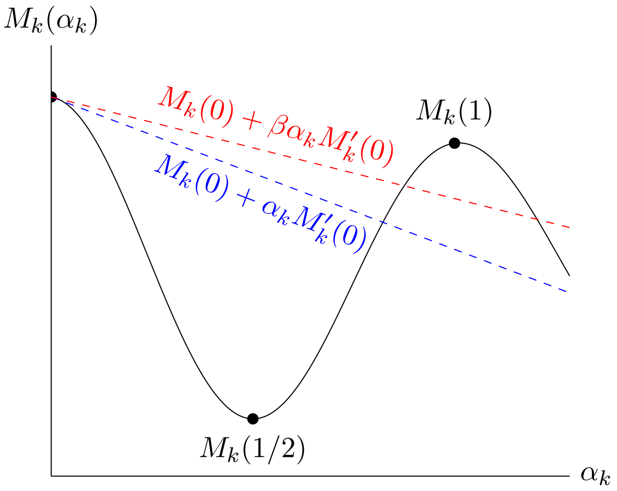


Figure 4.2: Illustrating the condition of sufficient decrease (4.25). The merit function M_k is plotted against α_k, together with its first-order approximation (*blue dashed line*) at $M_k(0)$ and its relaxed first-order approximation at the same point (*red dashed line*). All points on the curve below the latter satisfy the condition of sufficient decrease (i.e., $M_k(1/2)$ in the plot) while those above (i.e., $M_k(1)$) do not.

Note that the presented SQP method is an *unfeasible* method, which means that it allows iterates $\boldsymbol{z}_k$ to violate the constraint in (4.1), i.e., often $\boldsymbol{r}(\boldsymbol{z}_k) \neq 0$. Thus, the algorithm requires some additional means to asses the quality of the steps and iterates. These means are provided by the merit function M_k, combining the objective with a measure of constraint violation weighted by a parameter μ_k.

The condition of sufficient decrease (4.25) mandates that the actual reduction in the merit function value, which is the difference

$$M_k(0) - M_k(\alpha_k), \tag{4.26}$$

should be greater than a *predicted reduction* derived from a first-order Taylor expansion (relaxed by a parameter β), which is the difference

$$M_k(0) - (M_k(0) + \alpha_k \beta M_k'(0)). \tag{4.27}$$

Further, the merit function should ideally be *exact*, which means that any local solution of the optimization problem (4.1), should also be a local minimizer of M_k. For illustrative purposes, an example for a merit function and the condition of sufficient decrease is illustrated in Figure 4.2.

In this work, following Nocedal and Wright (2006), Chapter 15, two merit functions are employed. First, the canonical exact l^1-merit function

$$M_k^{l_1}(\alpha) := f(\boldsymbol{z}_k + \alpha\Delta\boldsymbol{z}_k) + \mu_k \|\boldsymbol{r}(\boldsymbol{z}_k + \alpha\Delta\boldsymbol{z}_k)\|_1 \tag{4.28}$$

is utilized, essentially summing the objective function value and the l^1-norm of the residual, where $\mu_k := 2\|\boldsymbol{\lambda}_{k+1}\|_\infty$ is a penalty parameter controlling the relative weighting of both terms. Secondly, we employ the non-smooth exact l^2-merit function

$$M_k^{l_2}(\alpha) := f(\boldsymbol{z}_k + \alpha\Delta\boldsymbol{z}_k) + \mu_k \|\boldsymbol{r}(\boldsymbol{z}_k + \alpha\Delta\boldsymbol{z}_k))\|_2. \tag{4.29}$$

While the considered parameters are adjustable, we explicitly set them to $\beta = 10^{-4}$, $\tau = 0.5$ and $\alpha_{\text{min}} = 10^{-8}$. In numerical experiments, we observed no significant difference between M^{l_1} and M^{l_2}. Hence, we use M^{l_1} in the remainder to be consistent with the HOIST method.

Remark 11. *Note that both, the l_1 and the l_2-merit function, are not differentiable at states $\boldsymbol{z}$ with $\boldsymbol{r}(\boldsymbol{z}) = 0$, because of the the absolute value from the l_1-norm and the root function in the l_2-norm, respectively. However, for the computation of (4.27), only a directional derivative in direction $(\Delta \boldsymbol{z}_k)$, denoted as $\nabla_{(\Delta \boldsymbol{z}_k)}$, of the corresponding norm is needed. For the absolute value it is derived in the work by Nocedal and Wright (2006) in Appendix A. Applying the presented formulas, we obtain for the l_1-case*

$$({M^{l_1}}_k)'(0) = \nabla f(\boldsymbol{z}_k) \cdot \Delta \boldsymbol{z}_k + \mu_k \|\boldsymbol{J} \Delta \boldsymbol{z}_k\|_1. \tag{4.30}$$

The l_2 case follows analogously and reads

$$({M^{l_2}}_k)'(0) = \nabla f(\boldsymbol{z}_k) \cdot \Delta \boldsymbol{z}_k + \mu_k \|\boldsymbol{J} \Delta \boldsymbol{z}_k\|_2. \tag{4.31}$$

Remark 12. *In certain scenarios, the objective function or residual generates an exception when evaluated at the new state $\boldsymbol{z}_{k+1}$ resulting from a step of length $1 \geqslant \alpha_k > \alpha^\star$. This exception may stem from nonphysical states (e.g. negative pressure/density) or a breach of the guidelines for managing newly formed cut-cells, the latter outlined in Section 4.2.5. In our actual implementation of the XDG-IST method, the step length is decreased for such scenarios and the line search algorithm is employed only within the admissible interval $(0, \alpha^\star)$.*

4.2.4 Adaptive regularization

Recall that the regularization parameter $\gamma_k \in [\gamma_{\text{min}}, \gamma_{\text{max}}]$ was key in regularizing the level set component of the system Matrix in (4.22). The selection of γ_k is intended to match the changes in the interface position, allowing for more regularization when the level set's update $\Delta \boldsymbol{\varphi}_k$ is significantly large ($\|\Delta \boldsymbol{\varphi}_k\| > \sigma_2 L$) and less when the update is notably small ($\|\Delta \boldsymbol{\varphi}_k\| < \sigma_1 L$). Following (Zahr et al., 2020), this flexibility is managed by the formula

$$\gamma_{k+1} = \min\{\max\{{\gamma^\star}_{k+1}, \gamma_{\text{min}}\}, \gamma_{\text{max}}\}, \quad \gamma_{k+1} = \begin{cases} {\gamma_{\text{red}}}^{-1}\gamma_k & \text{if } \|\Delta \boldsymbol{\varphi}_k\|_2 < \sigma_1 L \\ \gamma_{\text{red}}\gamma_k & \text{if } \|\Delta \boldsymbol{\varphi}_k\|_2 > \sigma_2 L \\ \gamma_k & \text{else.} \end{cases} \tag{4.32}$$

In this expression, σ_1, σ_2, and L are chosen based on the domain's specific length scales, with $\gamma_{\text{red}} > 1$ controlling how aggressively the adaptation is made. For our studies, we have selected $L = 1$, $\sigma_1 = 0.1$, $\sigma_2 = 0.01$, $\gamma_{\text{max}} = 1$, $\gamma_{\text{min}} = 10^{-8}$, and $\gamma_{\text{red}} = 1.5$.

4.2.5 Handling of newborn cut-cells

In the XDG shock tracking method, we compute XDG solutions iteratively, updating the flow and the level set coefficients $\boldsymbol{u}_k, \boldsymbol{\varphi}_k$ in each iteration. Usually, when updating the level set coefficients, the position of the shock interface ${\mathfrak{I}_s}^{(k)}$ changes, which may result in a status change of certain mesh cells $K \in \mathcal{K}_h$ (see Figure 4.3). In this section, we explain how our

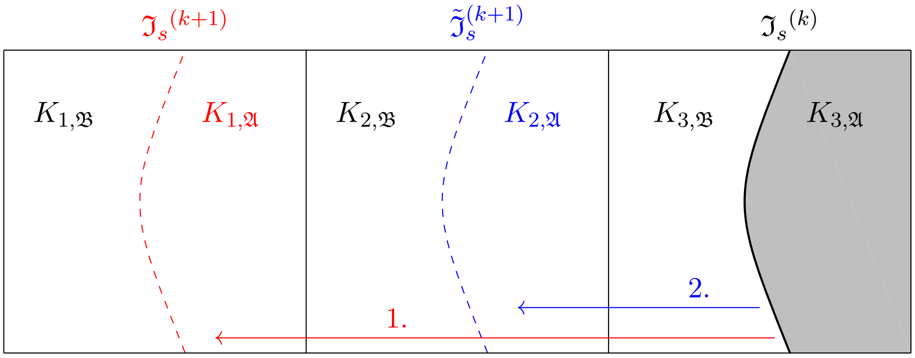


Figure 4.3: This example demonstrates how newly formed cut-cells are managed using a mesh of size 3×1 with a single level set $\varphi_s{}^{(k)}$ (its zero set $\mathfrak{I}_s{}^{(k)}$ is shown with a thick black line), which initially resides in cell K_3. Two separate updates, marked as 1. and 2., illustrate the movement of the interface to new positions $\mathfrak{I}_s{}^{(k+1)}$ and $\tilde{\mathfrak{I}}_s^{(k+1)}$, respectively. The first update (*red*) breaks Rule 1 (the interface must move only to adjacent cells) since K_1 is not an immediate neighbor of K_3, leading to its rejection by the solver. On the other hand, the second update (*blue*) adheres to the guidelines and is thus approved. (Figure adapted from Vandergrift and Kummer (2024))

approach deals with instances where the shock interface intersects a *newborn cell* (i.e., one that has not been intersected before the update). We want to ensure that the interface's movement remains consistent and continuous within our solution framework. Hence, we uphold two principal guidelines:

Rule 1. ***Make sure the interface moves only to neighbors****: Only cells adjacent to an already intersected cell are allowed to be intersected in the current iteration, as depicted in Figure 4.3. Updates that breach this rule are discarded and the step size α_k for the update is recalibrated (see Remark 12). This ensures that the interface progresses exclusively to adjacent cells.*

Rule 2. ***Extrapolate solution values for newborn cut-cells from the largest neighboring cell within the same sub-domain, as determined by volume****: A cell initially not intersected by any interface must be contained entirely in one sub-domain $\mathfrak{s}$. Once intersected, the cell is divided into cut-cells, one of them contained in another sub-domain $\mathfrak{s}'$. For this (newborn) cut-cell, we employ a strategy of solution extrapolation. This means that the solution values for these newborn cut-cells are derived from the largest neighboring cell within its sub-domain $\mathfrak{s}'$, based on volume.*

Example To demonstrate the implementation of the two specific rules for managing newly formed cells, consider an one-dimensional (1D) example for clarity. We have the domain $\Omega = [0, 3]$, which is segmented into three cells $K_i = (i-1, i)$ for $i \in \{1, 2, 3\}$, and is further divided into two sub-domains, $\mathfrak{A}$ and $\mathfrak{B}$. Here,

$$\mathfrak{A} = \mathfrak{A}(\varphi) := \{x \in \Omega \mid \ \varphi(x) < 0\} \tag{4.33}$$

and

$$\mathfrak{B} = \mathfrak{B}(\varphi) := \{x \in \Omega \mid \ \varphi(x) > 0\}\,. \tag{4.34}$$

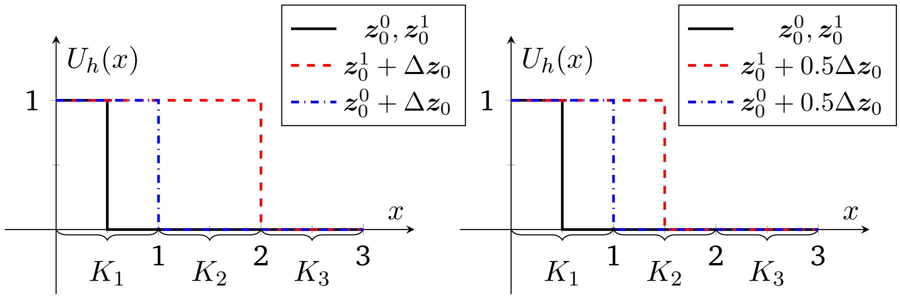


Figure 4.4: This figure demonstrates the process of applying an update, $\Delta \boldsymbol{z}_0$, to an one-dimensional XDG field $\boldsymbol{z}_0^\star$ across a grid consisting of three cells K_1, K_2, K_3. It highlights the method for managing newly created cut-cells. *Left*: The application of $\boldsymbol{z}_0^\star + \Delta \boldsymbol{z}_0$ shifts the interface from $x = 0.5$ to $x = 2.5$. This movement breaches Rule 1 by making the step length $\alpha = 1$ unacceptable. *Right*: Adjusting the update to $\boldsymbol{z}_0^\star + 0.5\Delta \boldsymbol{z}_0$ relocates the interface from $x = 0.5$ to $x = 1.5$, adhering to Rule 1 without violation. The solution within the newly formed cut-cell $K_{2,\mathfrak{B}} = (1, 1.5)$ is then extrapolated from its neighbor, utilizing $\boldsymbol{z}_1 = \boldsymbol{z}_0^1 + 0.5\Delta \boldsymbol{z}_0$ (red curve) instead of $\boldsymbol{z}_1 = \boldsymbol{z}_0^0 + 0.5\Delta \boldsymbol{z}_0$ (blue curve). (Figure adapted from Vandergrift and Kummer (2024))

We choose an initial level set $\varphi_0(x) = x + a$, defined globally with a single degree of freedom (DOF) $a \in \mathbb{R}$, initially set at $a = -0.5$. For the discretized solution, we construct an element U_h within the $(P = 0)$ XDG space through the definition of two coordinate vectors, $\boldsymbol{u}^0$ and $\boldsymbol{u}^1$

$$\boldsymbol{u}^\star = (u_i^\star)_{i\in[6]} = (\underbrace{\underbrace{1}_{K_{1,\mathfrak{A}}}, \underbrace{0}_{K_{1,\mathfrak{B}}}}_{\text{cut-cell}}, \underbrace{\underbrace{\star}_{K_{2,\mathfrak{A}}}, \underbrace{0}_{K_{2,\mathfrak{B}}}}_{\text{uncut-cell}}, \underbrace{\underbrace{0}_{K_{3,\mathfrak{A}}}, \underbrace{0}_{K_{3,\mathfrak{B}}}}_{\text{uncut-cell}})^T, \quad \boldsymbol{z}_0^\star = \begin{pmatrix} \boldsymbol{u}^\star \\ a \end{pmatrix}, \tag{4.35}$$

where $\star \in \{0, 1\}$ and alongside a current iteration vector $\boldsymbol{z}_0^\star$. The relationship between the coefficients $u_i^\star$ and the cut-cells $K_{l,\mathfrak{s}}$ is elaborated in Equation 4.35 (assuming $P = 0$, which implies a single basis function per cut-cell, essentially the indicator function specific to each cell). It's noteworthy that $K_{2,\mathfrak{A}}$ and $K_{3,\mathfrak{A}}$ are null sets, resulting in $u_3^\star$ and $u_5^\star$ having no impact on the actual value of $U_h(x)$ for $x \in \Omega$. Therefore, the value of $U_h(x)$ can be described as $\tilde{B} \cdot \boldsymbol{\Phi}(x) = \boldsymbol{u}^1 \cdot \boldsymbol{\Phi}(x)$, where $\boldsymbol{\Phi}(x)$ is the XDG basis vector. By considering an imaginary search direction $\Delta \boldsymbol{z}_0 = (0, \dots, 0, -2)^T$, exclusively updating the level set DOF, we proceed with a line search (as discussed in Section 4.2.3) that respects the previously mentioned rules. Implementing the search direction $\Delta \boldsymbol{z}_0$ (illustrated in the Figure 4.4) with $\alpha_0 = 1$ leads to

$$\boldsymbol{z}_1^\star = \boldsymbol{z}_0^\star + \Delta \boldsymbol{z}_0 = \begin{pmatrix} \boldsymbol{u}^\star \\ -2.5 \end{pmatrix}. \tag{4.36}$$

The update results in the level set being revised to $\varphi_1(x) = x - 2.5$, which designates cell K_3 as a cut-cell, alters the status of K_1 from a cut-cell to a non-cut-cell, and causes K_2 to switch sub-domains. However, this adjustment contravenes Rule 1 because the neighboring cell to K_3, cell K_2, was not a cut-cell prior to the update. As a result, the algorithm dismisses the step size $\alpha_0 = 1$ and evaluates a reduced step size $\tau\alpha_0 = 0.5$.

With the step size set to $\alpha = 0.5$, the updated level set becomes $\varphi_1(x) = x - 1.5$, making cell K_2 a cut-cell without breaching Rule 1, given that K_2 was adjacent to the previous cut-cell K_1.

Moreover, implementing Rule 2 is showcased as follows: without solution extrapolation, the discrete state $\boldsymbol{u}^\star \cdot \boldsymbol{\Phi}$ varies with respect to the value $\star$, since $K_{2,\mathfrak{A}}$ transitions from being an empty cell. Thus, the discrete state is represented as:

$$\boldsymbol{u}^{\star}(x)\cdot\boldsymbol{\Phi}(x)=\begin{cases}1 & \text{for } x\leqslant 1\\ \star & \text{for } 1<x\leqslant 1.5\\ 0 & \text{else.}\end{cases} \tag{4.37}$$

To ensure continuity within the solution space, solution values for newly formed cut-cells are determined by extending the polynomial solution from the largest neighboring cell (by volume) within the same sub-domain. In this scenario, regardless of its initial value, $u_3^{\star}=1$ is selected for the update with $\alpha_0=0.5$.

4.3 Robustness measures

Numerical experiments conducted have shown that, under specific conditions, implicit shock tracking methods may face challenges with convergence or become trapped in sub-optimal local minima (Huang and Zahr, 2022). In the case of the XDG shock tracking method presented above, such difficulties tend to intensify when working with small cut-cells or when the polynomial degree of the flow approximation is set above zero ($P>0$), particularly if the current iterate is far from the true solution. To counter these challenges and improve the stability of the method, we have implemented supplementary stabilization techniques, each tailored to mitigate different issues that could emerge during the computation.

In this section, drawing from the presentation by Vandergrift and Kummer (2024), we outline a series of robustness measures designed to enhance the reliability and stability of the XDG-IST method in focus. In Section 4.3.1, we discuss the strategy of cell agglomeration, a technique aimed at improving computational stability and efficiency by merging small computational cut-cells into their bigger neighbors. Additionally, we explore the process of local solution reinitialization in Section 4.3.2, which involves resetting the solution's state to ensure better convergence properties. Lastly, in Section 4.3.3, we introduce the P-continuation strategy, a methodology that gradually increases the complexity of the discretization by adjusting the polynomial degree of the XDG method.

4.3.1 Cell agglomeration

In numerical experiments, stagnation of the optimizer has been partially traced back to the presence of small cut-cells. As the background mesh is fixed, the ideal positioning of the interface can result in the formation of small cut-cells (small when compared to the volume of the uncut cell). Moreover, during optimization, the arbitrary locations of interfaces across intermediate iterations can further lead to the emergence of these small cells. The challenge with small cut-cells is that the entries of the solution vector $\boldsymbol{u}$ and related residual Jacobians $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}, \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}$ inversely correlate with cell size. Consequently, having a high discrepancy between cell volumes, amplifies the condition number of the linear system (4.22) and ultimately destabilizes the XDG shock tracking method. Further details on the effect of small cut-cells on stability and conditioning of immersed boundary methods can be found in the work by Prenter et al. (2022).

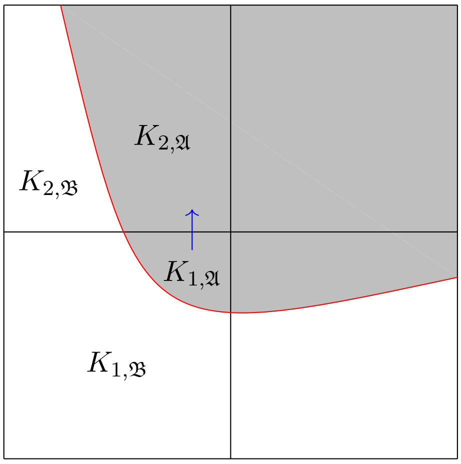


Figure 4.5: The figure illustrates the process of cell agglomeration applied to a 2×2 mesh intersected by a level set φ, with the zero set highlighted by a red line. It showcases the agglomeration of a relatively small cut-cell, denoted as $K_{1,\mathfrak{A}}$, into its largest neighboring cell $K_{2,\mathfrak{A}}$ in terms of volume, that resides within the same subdomain $\mathfrak{A}$. This merging process is visually represented by a blue arrow pointing from the smaller cell towards its larger neighbor. (Figure adapted from Vandergrift and Kummer (2024))

The technique of *cell agglomeration* is deployed to mitigate issues arising from small cut-cells. This method involves combining cut-cells $K_{j,\mathfrak{s}}$ when the ratio of their volume to the volume of the parent cell falls below a predefined threshold $\text{agg}_{\text{thrsh}} \in (0.1)$ (in this work chosen to be $\text{agg}_{\text{thrsh}} = 0.4$), i.e.,

$$\frac{|K_{j,\mathfrak{s}}|}{|K_j|} \leqslant \text{agg}_{\text{thrsh}}. \tag{4.38}$$

The selected cut-cells are then merged with their largest neighboring cell within the same sub-domain (illustrated in Figure 4.5 with a 2×2 mesh and an example of cell agglomeration), ensuring that cells across a discontinuity are not merged. This leads to the transformation of the initial cut-cell grid $\mathcal{K}_h^X$ into an agglomerated grid $\mathcal{K}_h^{X,\text{agg}}(\mathcal{K}_h^X, \mathcal{A})$, where $\mathcal{A}$ serves as the agglomeration map. This map is a subset of all pairs of neighboring cut-cells within the same domain that share an edge, that is

$$\mathcal{A} \subset \left\{ (K_{i,\mathfrak{s}}, K_{j,\mathfrak{s}}) \;\middle|\; \overline{K}_{i,\mathfrak{s}} \cap \overline{K}_{j,\mathfrak{s}} \neq \varnothing, (i,j) \in [J]^2, \mathfrak{s} \in \text{SD} \right\}. \tag{4.39}$$

Here, the relation $(K_{i,\mathfrak{s}}, K_{j,\mathfrak{s}}) \in \mathcal{A}$ indicates that $K_{i,\mathfrak{s}}$ is agglomerated into $K_{j,\mathfrak{s}}$. In the context of the BoSSS framework, *cell agglomeration* has already proven successful in various applications and the approach has been fundamental in the development of a specialized multigrid algorithm, further elaborated in the work by Kummer et al. (2021).

Within the framework of the XDG-IST method and considering an SQP iteration step with $\boldsymbol{z}_k$ as the starting point, the application of cell agglomeration unfolds through the following steps:

1. Compute the residuals and sensitivities on the original, non-agglomerated mesh $\mathcal{K}_h^X$ by

$$\boldsymbol{z}_k \mapsto \boldsymbol{r}(\boldsymbol{z}_k), \boldsymbol{F}(\boldsymbol{z}_k), \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{z}}(\boldsymbol{z}_k), \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{z}}(\boldsymbol{z}_k), \tag{4.40}$$

2. Generate the agglomeration map

$$(\boldsymbol{\varphi}_k, \varphi_b) \mapsto \mathcal{A}_k := \mathcal{A}_k(\boldsymbol{\varphi}_k, \varphi_b) \tag{4.41}$$

utilizing (4.38), which takes into account both the static boundary level set φ_b and the shock level set φ_s associated with the coefficients $\boldsymbol{\varphi}_k$.

3. Calculate the agglomerated state
$$\boldsymbol{z}_k \mapsto \boldsymbol{z}_k^{\text{agg}}, \tag{4.42}$$
requiring a basis-change from the original basis of $\mathcal{V}_h^{P,m,X}(\mathcal{K}_h^X)$ to the agglomerated basis of $\mathcal{V}_h^{P,m,X}(\mathcal{K}_h^{X,\text{agg}})$.

4. Derive the agglomerated residuals and sensitivities
$$\left(\boldsymbol{r}(\boldsymbol{z}_k), \boldsymbol{F}(\boldsymbol{z}_k), \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{z}}(\boldsymbol{z}_k), \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{z}}(\boldsymbol{z}_k)\right) \mapsto \left(\boldsymbol{r}^{\text{agg}}(\boldsymbol{z}_k^{\text{agg}}), \boldsymbol{F}^{\text{agg}}(\boldsymbol{z}_k^{\text{agg}}), \frac{\partial \boldsymbol{r}^{\text{agg}}}{\partial \boldsymbol{z}^{\text{agg}}}(\boldsymbol{z}_k^{\text{agg}}), \frac{\partial \boldsymbol{F}^{\text{agg}}}{\partial \boldsymbol{z}^{\text{agg}}}(\boldsymbol{z}_k^{\text{agg}})\right) \tag{4.43}$$
from their original versions through the aforementioned basis transformation. Here, the superscript $(\cdot)^{\text{agg}}$ denotes that the agglomerated basis is used for the residuals.

5. Assemble and solve the agglomerated version of the linear system (4.22) to identify the agglomerated inexact Newton step $\Delta \boldsymbol{z}_k^{agg}$.

6. For each line search iteration, calculate $\boldsymbol{z}_k^{\text{agg}} + \alpha \Delta \boldsymbol{z}_k^{\text{agg}}$ and the corresponding agglomerated residuals
$$\left(\boldsymbol{z}_k^{\text{agg}}, \alpha, \Delta \boldsymbol{z}_k^{\text{agg}}\right) \mapsto \left(\boldsymbol{r}^{\text{agg}}(\boldsymbol{z}_k^{agg} + \alpha \Delta \boldsymbol{z}_k^{\text{agg}}), \boldsymbol{F}^{\text{agg}}(\boldsymbol{z}_k^{\text{agg}} + \alpha \Delta \boldsymbol{z}_k^{\text{agg}})\right) \tag{4.44}$$
when evaluating the merit function M_k (see Equation 4.26) and its derivative (see Equation 4.27) for different values α.

7. Upon determining an appropriate step length α_k, update the agglomerated iterate to $\boldsymbol{z}_{k+1}^{\text{agg}} = \boldsymbol{z}_k^{\text{agg}} + \alpha_k \Delta \boldsymbol{z}_k^{\text{agg}}$ and subsequently map it back to the original, non-agglomerated mesh to achieve the updated iterate $\boldsymbol{z}_{k+1}$, i.e.,
$$\boldsymbol{z}_{k+1}^{\text{agg}} \mapsto \boldsymbol{z}_{k+1}. \tag{4.45}$$
This last step is achieved by an extrapolation from the agglomerated mesh $\mathcal{K}_h^{X,\text{agg}}$ to the non-agglomerated source mesh $\mathcal{K}_h^X$.

4.3.2 Solution reinitialization

In accordance with Huang and Zahr (2022) and following the presentation by Vandergrift and Kummer (2024), we adopt a local solution reinitialization procedure to further enhance robustness. Similar to Huang and Zahr (2022) in the context of mesh-based shock tracking, we also observed (in prior numerical experiments) the development of non-physical oscillations in the XDG solution during intermediate steps when employing polynomial degrees greater than zero. Such oscillations often lead to subpar search directions in the iterations that follow, causing the step sizes α_k to become excessively small.

To mitigate this, our reinitialization strategy is implemented as follows: it utilizes a version of the established Persson-Peraire shock sensor (see Persson and Peraire (2006)) to detect cells experiencing oscillatory behavior. In the identified cells, the XDG solution is reset locally to a constant value, computed as the average across a patch of neighboring elements around the oscillatory cell (illustrated in Figure 4.6).

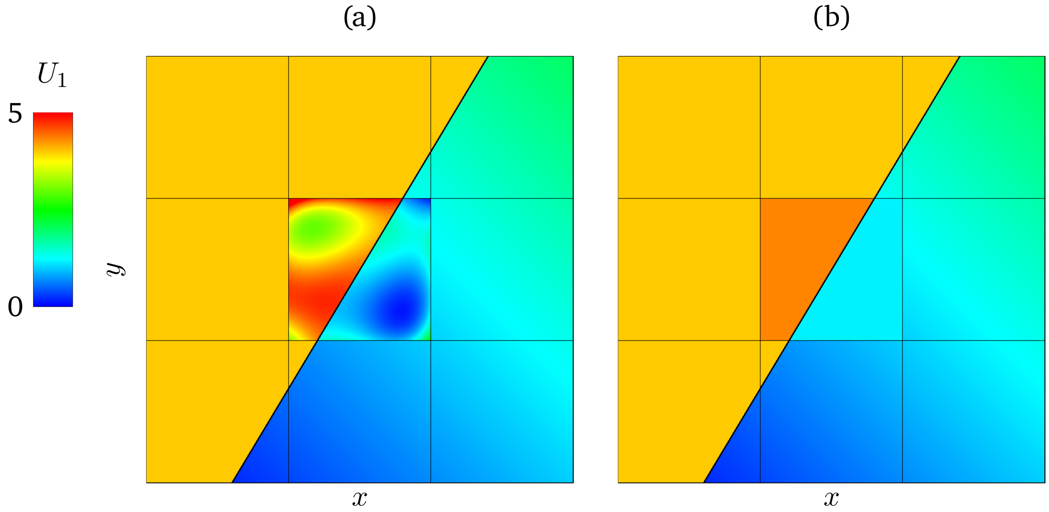


Figure 4.6: Illustration of the cell-local solution reinitialization procedure for a 3×3 mesh and a straight-sided interface. Panel (a) shows the solution U_1 featuring oscillatory cells in the center of the mesh. Panel (b) highlights the reinitialized solution U^{ReInit}, i.e., where the oscillatory cells are reset to constant values, determined by a patch of shock-aware neighbors.

Determining oscillatory cells In a first step, oscillatory cells are identified using a version of the Persson-Peraire shock sensor: For notational ease, let $U = U_h \in \mathcal{V}_h^{P,m,X}$ of polynomial degree $P > 0$ be the XDG solution and $K = K_{j,\mathfrak{s}} \in \mathcal{K}_h^X$ a cut-cell. We define the shock sensor $\text{sen} : \mathcal{K}_h^X \times \mathcal{V}_h^{P,m,X} \to \mathbb{R}$ for the first component $U_1 \in \mathcal{V}_h^{P,1,X}$ of U as:

$$\text{sen} : \ (K, U) \mapsto \log\left(\frac{\|U_1 - \text{proj}_{P-1}(U_1)\|_{L^2(K)}}{\|U_1\|_{L^2(K)}}\right). \tag{4.46}$$

Here, $\text{proj}_{P-1} : \mathcal{V}_h^{P,1,X} \to \mathcal{V}_h^{P-1,1,X}$ denotes the orthogonal L^2-projection

$$\text{proj}_{P-1} : \ U_1 \mapsto \text{proj}_{P-1}(U_1) \tag{4.47}$$

onto the XDG space $\mathcal{V}_h^{P-1,1,X}$ with polynomial degree $P-1$. The shock-sensor is commonly used in shock capturing strategies to identify cells with oscillating solutions in order to apply the local stabilization therein.

Using the sensor, we define the set of oscillatory cells by

$$\mathcal{K}_{\text{ReInit}}(U) := \left\{K \in \mathcal{K}_h^X \;\middle|\; \text{sen}(K, U) > \epsilon_1\right\}, \tag{4.48}$$

where $\epsilon_1 < 0$ (typically we choose $\epsilon_1 = -0.2$) is a threshold chosen depending on the problem and the polynomial degree.

Small step sizes result from excessive line search iterations and indicate stagnation of the optimization method. We follow Huang and Zahr (2022) who propose to extend the set $\mathcal{K}_{\text{ReInit}}(U)$ in such cases (5 or more line search iterations). In these scenarios, the set is redefined as

$$\mathcal{K}_{\text{ReInit}}(U) := \left\{K \in \mathcal{K}_h^X \;\middle|\; \text{sen}(K, U) > \epsilon_2 \max_{K \in \mathcal{K}_h^X} \text{sen}(K, U)\right\}, \tag{4.49}$$

where Huang and Zahr (2022) propose to set $\epsilon_2 = 10^{-2}$.

Shock-aware solution reinitialization Upon identifying the oscillatory cells $\mathcal{K}_{\text{ReInit}}(U)$ the XDG solution is reset on these to a constant value. This value is derived as an average of the current solution U across a selected group of neighboring cells $\tilde{\mathcal{N}}(K,U)$, which we define below. In line with Huang and Zahr (2022), not all neighboring cells are selected for averaging. The selection is restricted to cells residing on the same side of the interface. This distinction is critical for maintaining physical relevance, particularly near discontinuities or shocks.

To select neighboring cells that qualify for averaging, we utilize the average jump function $a_J : \mathcal{K}_h^X \times \mathcal{K}_h^X \times \mathcal{V}_h^{P,m,X} \to \mathbb{R}$ defined as

$$a_J : (K, K', U) \mapsto \frac{1}{|\partial K \cap \partial K'|} \int_{\partial K \cap \partial K'} [\![U_1]\!] \, dS, \tag{4.50}$$

where $[\![U_1]\!]$ represents the jump in U_1 across the common face of cells K and K'. The function a_J measures the magnitude of the discontinuity across cell boundaries and, thereby, assists in distinguishing between cells based on their relative positioning to the discontinuity or shock. Additionally, a_J is also helpful in selecting cells if the shock position does not coincide with the interface $\mathfrak{I}_s$ (usually the case during intermediate iterations).

Using the jump function a_J, the shock-aware neighbor set $\tilde{\mathcal{N}}(K,U)$ is defined as

$$\tilde{\mathcal{N}}(K,U) := \left\{ K' \in \mathcal{N}(K) \;\middle|\; |a(K,K',U)| \leqslant \epsilon_3 \right\}, \tag{4.51}$$

where again $\epsilon_3 = 10^{-2}$ is a threshold parameter. For cells within this set, each component U_i of the XDG solution is reinitialized by

$$U^{\text{ReInit}}{}_i|_K := \begin{cases} \frac{1}{|K \cup \tilde{\mathcal{N}}(K,U)|} \int_{|K \cup \tilde{\mathcal{N}}(K,U)|} U_i \, dV & \text{if } K \in \mathcal{K}_{\text{ReInit}}(U), \\ U_i & \text{else,} \end{cases} \tag{4.52}$$

for each component $i = [m]$. Reinitialization is not used when the current iteration is sufficiently close to meeting the constraints, specifically when:

$$\|\boldsymbol{r}(\boldsymbol{z}_k)\| \leqslant \epsilon_4, \tag{4.53}$$

where $\epsilon_4 = 10^{-2}$. Additionally, reinitialization is not used after a designated number of iterations to guarantee the asymptotic convergence of the method. When the polynomial degree P is increased as part of the P-continuation strategy (will be introduced in Section 4.3.2), the iteration limit for reinitialization is proportionally raised to accommodate the adjustments for the higher degrees.

Algorithm The detailed algorithm for re-initializing the solution in cells that exhibit oscillatory behavior is encapsulated in Algorithm 1. The solution might be reinitialized across a different set of cells $\mathcal{K}_{\text{ReInit}}(U)$. Depending whether it is triggered by excessive line search iterations (*isEL* $=$ *true*) or not (*isEL* $=$ *false*), the set of cells to be re-initialzied is defined in Equation (4.48) and (4.49) respectively.

Algorithm 1 Solution reinitialization on oscillatory cells (adapted from Vandergrift and Kummer (2024))

```
Input: XDG solution $U$, grid $\mathcal{K}_h^X$, parameters $\epsilon_1, \epsilon_2, \epsilon_3$ and logical isEL (indicating whether
   solution reinitialization is applied due to excessive line search)
Output: XDG solution $U^{\text{ReInit}}$ reinitialized on oscillatory cells
  Compute max shock sensor: $S_{\max} := \max_{K \in \mathcal{K}_h^X} \text{sen}(K, U)$
  Compute set of cells to be reinitialized $\mathcal{K}_{\text{ReInit}}(U)$:
  $\mathcal{K}_{\text{ReInit}}(U) := \varnothing$ (initialize)
  for $K \in \mathcal{K}_h^X$ do
     Compute shock sensor $\text{sen}(K, U)$
     if isEL and $\text{sen}(K, U) > \epsilon_2 S_{\max}$ then
        $\mathcal{K}_{\text{ReInit}}(U) := \mathcal{K}_{\text{ReInit}}(U) \cup K$ (add cell)
     else if $\text{sen}(K, U) > \epsilon_1$ then
        $\mathcal{K}_{\text{ReInit}}(U) := \mathcal{K}_{\text{ReInit}}(U) \cup K$ (add cell)
     end if
  end for
  Reinitialize solution:
  $U^{\text{ReInit}} := U$ (initialize)
  for $K \in \mathcal{K}_{\text{ReInit}}(U)$ do
     Compute shock-aware neighbors:
     $\tilde{\mathcal{N}}(K, U) := \varnothing$ (initialize)
     for $K' \in \mathcal{N}(K)$ do
        if $a_J(K, K', U) \leqslant \epsilon_3$ then
           $\mathcal{N}(K, U) := \mathcal{N}(K, U) \cup K'$
        end if
     end for
     Reinitialize solution cell-wise:
     Set $U^{\text{ReInit}}|_K := \frac{1}{|K \cup \tilde{\mathcal{N}}(K,U)|} \int_{|K \cup \tilde{\mathcal{N}}(K,U)|} U \, dV$
  end for
  Set $U = U^{\text{ReInit}}$
```

4.3.3 P-Continuation strategy

During our numerical tests, we found that, in situations needing significant adjustments to the level set, it is advantageous to start with a polynomial degree of $P = 0$ (for the flow solution) and then increment it progressively until the desired final polynomial degree P_{end} is reached. In this setting, the solution is sought in the XDG spaces

$$\mathcal{V}_h^{0,m} \to \mathcal{V}_h^{1,m} \to \ldots \to \mathcal{V}_h^{P_{\text{end}},m}, \tag{4.54}$$

starting on the left. This is especially useful when the initial shock interface (i.e., the initial guess) is not close to the true shock position. Our step-up approach may only increase the polynomial degree P if the solver surpasses the pre-set minimum SQP iteration thresholds $k_{\min}^P$ for the current polynomial degree P. In our work, we selected

$$\{k_{\min}^0 = 30,\ k_{\min}^1 = 30,\ k_{\min}^2 = 10,\ k_{\min}^3 = 10,\ k_{\min}^4 = 10,\ \ldots\} \tag{4.55}$$

for these minimum iterations. The polynomial degree is increased according to the termination conditions presented in Section 4.4.3. For the implementation of this strategy, one needs to transfer all dependent data structures to the incremented polynomial degree in the moment of incremental.

4.4 Solver initialization and termination

In this section, we discuss the initialization and the termination of the XDG-IST solver, i.e., how to choose an initial guess for the optimizer and how to decide whether the optimization process should be terminated. In Section 4.4.1, methods for constructing a first guess for the shock level set are discussed and, in Section 4.4.2, different approaches to select an initial guess for the flow solution U are introduced. Lastly,in Section 4.4.3, we present details on the termination criteria employed in the shock tracking framework.

4.4.1 Initialization of the shock level set

In this work, the initialization of the shock level set follows one of two approaches: Choosing an user-defined initial level set derived from a priori knowledge or employing a reconstruction procedure based on a shock capturing approach.

User-defined initial guess In the first approach, the user simply chooses a function φ_0 which is then projected onto the discretized level set. In this work, we use this approach for testing purposes, but one should note that it is unfeasible for many real-world problems, especially those lacking a priori knowledge of the solution. However, in the case of time-dependent problems deriving the initial guess from the initial conditions can be a valid choice. Specifically, the initial position of the shock and the corresponding shock speed at $t = 0$ can be used for this purpose.

Initial guess from shock capturing A more advanced approach to generate an initial guess for the level set leverages a shock capturing methodology. It is computationally more intensive, as it requires to run a full simulation. In our setup, we utilize the compressible Navier-Stokes (CNS) solver (Müller et al., 2017; Krämer-Eis, 2017; Geisenhofer et al., 2019), a DG-immersed boundary method (IBM) using artificial viscosity for stabilization, to obtain a high-order shock capturing flow solution (see Remark 13). This solution features a smoothed shock (due to the artificial viscosity) and we apply a reconstruction procedure developed by Geisenhofer (2021), which allows for the derivation of a sub-cell accurate estimate for the shock level set. The reconstruction procedure aims to reconstruct the shock front by identifying a set of points, $\chi_s := \{\vec{x}_i\} \subset \Omega$, through a three-step process:

1. Initialization with seed points χ_{seed} placed in all cells with artificial viscosity.
2. Generation of candidate points χ_{cand} from χ_{seed} by identifying sign changes in the Hessian of the density along the density gradient $\nabla\rho$ and by using a bisection method.

3. Clustering of χ_{cand} based on density and the degree of artificial viscosity, while filtering out points not associated with the shock.

The outcome is a collection of points suitable for creating a level set representation of the shock front. Further insights to the procedure and its application to a Mach 4 bow shock (Section 5.1.3) are detailed in the work by Geisenhofer (2021).

Having obtained a description of the shock in the form of points χ_s, an initial level set is constructed depending on the employed level set type. Using the spline-based level set approach (4.6.4), the defining spline can be constructed by interpolating the points χ_s. In the case of implicit level set representations, one can construct a signed distance function φ_0 from χ_s, being the distance to the closest point from χ_s by

$$\varphi_0(\vec{x}) = \text{sgn}(\rho(\vec{x}) - \rho(\vec{x}_{\text{min}}(\vec{x})))\|\vec{x} - \vec{x}_{\text{min}}(\vec{x})\|_2, \tag{4.56}$$

where the closest point is defined as

$$\vec{x}_{\text{min}}(\vec{x}) = \text{argmin}_{\vec{x}_i \in \chi_s} \|\vec{x} - \vec{x}_i\|_2. \tag{4.57}$$

In this work, this reconstruction procedure was used to obtain an initial guess for the level set for a Mach 4 bow shock (Section 5.1.3). However, this method's applicability beyond the demonstrated case, particularly the clustering aspect in complex flow scenarios, remains to be validated.

Remark 13. *The CNS solver implemented in the BoSSS framework (Müller et al., 2017; Krämer-Eis, 2017; Geisenhofer et al., 2019) is a DG-IBM solver tailored for solving the compressible Navier-Stokes and Euler equations. This solver incorporates a shock capturing technique for the Euler equations, utilizing artificial viscosity to manage shock waves effectively. Additionally, it is capable of handling immersed geometries through the use of a level set φ_b. To localize shocks, a shock sensor akin to the one detailed in* (4.46) *is employed. Upon detection of a shock within certain cells, a second-order diffusion operator is applied to these cells, smoothing the shock profile and enhancing the solver's stability.*

4.4.2 Initialization of flow solution

The initialization of the flow solution, i.e., determining the coordinate vector $\boldsymbol{u}_0$, can also be accomplished by different approaches. In this work, the following three have been implemented in the BoSSS framework:

1. User-defined initial guess

2. Pseudo-time evolution with a fixed level set to compute a $(P = 0)$ initial guess

3. Shock capturing solution

Next, all three approaches are described individually.

User-defined initial guess In the first approach, the user simply chooses a function for each flow variable (each component U_i of U), which is then projected onto the employed XDG space (after choosing an initial position for the level set and a polynomial degree). In this work, we use this approach for testing purposes, but one should note that it is impractical for many real-world problems without a priori knowledge of the solution. However, in the case of time-dependent problems, it is valid to choose the initial guess to be an extrusion of the initial condition in time.

Pseudo-time evolution The second approach refers to a method where the problem represented by the system of conservation laws (3.5) (including space-time conservation laws) is addressed by solving for a low-order solution through pseudo-time evolution. Given a fixed initial level set characterized by its coefficients φ_0, the objective is to find a solution $\tilde{\boldsymbol{u}}$ in the 0th order XDG space ($P = 0$) for the discretized problem

$$\boldsymbol{r}^0(\tilde{\boldsymbol{u}}) = \boldsymbol{r}^0(\tilde{\boldsymbol{u}}, \varphi_0) = 0, \tag{4.58}$$

where $\boldsymbol{r}^0$ represents the residual from (3.29) employing 0th order XDG test- and trial spaces ($P, P' = 0$). This setting corresponds to a classical finite volume method (FVM) on the cut-cell grid $\mathcal{K}_h^X$.

Directly solving the nonlinear Equation (4.58) (using for example Newton's method) often remains computationally challenging, due to the lack of a good initial guess. The utilized nonlinear solver may stagnate in local minima and, thus, fail to find the solution.

As a robust remedy, the solution $\tilde{\boldsymbol{u}}$ is sought by employing pseudo-time evolution, carried out until the residual is minimized to a certain threshold (e.g. $\boldsymbol{r}^0(\tilde{\boldsymbol{u}}) \approx 10^{-12}$) and, hence, the flow becomes (pseudo-) steady. In this setup, we also need to solve a system of nonlinear equations, but here, the controllable time-step governs the proximity of the current state and the state at the next pseudo-time. Hence, it is reasonable to assume that we will always find a small pseudo-time, such that the pseudo-time evolution problem is solvable by Newton's method. Concretely, we solve the system of ordinary differential equations

$$\boldsymbol{M}\frac{\partial \tilde{\boldsymbol{u}}}{\partial \tilde{t}} + \boldsymbol{r}^0(\tilde{\boldsymbol{u}}) = 0, \tag{4.59}$$

where $\boldsymbol{M}$ is the mass matrix resulting from the discretization of the pseudo-temporal operator, $\tilde{t}$ is a pseudo-time variable, and $\tilde{\boldsymbol{u}} = \tilde{\boldsymbol{u}}(\tilde{t})$ is assumed to be pseudo-time-dependent. Beginning with an user-defined initial guess $\tilde{\boldsymbol{u}}_0$, pseudo-time updates are computed, using an implicit time stepping scheme (in this work: implicit Euler) with adaptive time steps $\Delta\tilde{t}_k$. The resulting nonlinear equations are then solved using Newton's method (restricted to a maximum of 10 nonlinear iterations). Additionally, we employ a dog-leg approach (Pawlowski et al., 2008) for globalization. The solution is iteratively updated to $\tilde{\boldsymbol{u}}_{k+1} = \tilde{\boldsymbol{u}}_k + \Delta\tilde{\boldsymbol{u}}_k$, where $\Delta\tilde{\boldsymbol{u}}_k$ is the step resulting from the dog-leg method. Furthermore, the following time-step adaptation is employed

$$\Delta\tilde{t}_{k+1} = \begin{cases} \Delta\tilde{t}_k\sigma_{\nu,+} & \text{if } \nu \leqslant \nu_{\text{min}}, \\ \Delta\tilde{t}_k & \text{if } \nu_{\text{min}} < \nu \leqslant \nu_{\text{max}}, \\ \Delta\tilde{t}_k\sigma_{\nu,-} & \text{else,} \end{cases} \tag{4.60}$$

where $\nu = \max_{i \in N_{\boldsymbol{u}}} |(\Delta\tilde{\boldsymbol{u}}_k)_i|$, $\nu_{\text{min}} = 0.05$, $\nu_{\text{max}} = 0.1$, $\sigma_{\nu,+} = 2$, and $\sigma_{\nu,-} = 0.2$ are parameters controlling the amplification or reduction of the time-step.

Similar methods are utilized by other shock-tracking groups (Zahr et al., 2020; Huang et al., 2023) and prove to be a robust approach for obtaining the initial guess. Compared to the shock capturing approach, they are also more computationally efficient, due to the lower polynomial degree.

Initial guess from shock capturing A more advanced yet computationally intensive approach to generate an initial guess leverages a shock capturing methodology and allows for the construction of a high-order initial estimate. Specifically, we utilize the CNS solver (Müller et al., 2017; Krämer-Eis, 2017; Geisenhofer et al., 2019), a DG-IBM using artificial viscosity for stabilization, to obtain a high-order shock capturing flow solution (see Remark 13). In order to obtain a solution of high-quality, it is advisable to employ a fine mesh, especially around the shock. Such a mesh would minimize the area where artificial viscosity is applied.

4.4.3 Termination

One approach to terminate the algorithm (or increases the polynomial degree of the solution), is to wait until the norms of the residuals have diminished below a specified threshold. However, the selection of an one-size-fits-all threshold is fraught with difficulty; a threshold that's excessively low can cause the algorithm to run indefinitely, whereas one that's overly high may lead to a premature stop, hence not fully leveraging the capabilities of the high-order method. To navigate this issue, the algorithm proceeds until the solution stabilizes (in terms of residual norms $\|\boldsymbol{r}\|$, $\|\boldsymbol{F}\|$) across a number of predetermined steps, denoted as k_{term} (here set to $k_{\text{term}} = 8$). To monitor stability, we utilize a *residual-norm skyline*, represented by $\text{sr}_k^{\boldsymbol{c}} := \min_{\tilde{k} \leqslant k} |\boldsymbol{c}(\boldsymbol{z}_{\tilde{k}})|$ (with $\boldsymbol{c}$ being either $\boldsymbol{r}$ or $\boldsymbol{F}$), along with an averaged reduction factor defined as:

$$\text{arf}_k^{\boldsymbol{c}} := \frac{1}{k_{\text{term}}} \left(\sum_{\tilde{k}=k-k_{\text{term}}}^{k-1} \frac{\text{sr}_{\tilde{k}}^{\boldsymbol{c}},}{\max(\text{sr}_{\tilde{k}+1}^{\boldsymbol{c}}, 10^{-100})} \right), \tag{4.61}$$

applicable for $k \geqslant k_{\text{term}}$ and applied to both residuals. The criteria for terminating the method hinge on these conditions being met for both $\boldsymbol{r}$ and $\boldsymbol{F}$:

$$\big(k \geqslant k_{\text{term}}\big) \ \wedge \ \big(\text{sr}_k^{\boldsymbol{c}} \leqslant 10^{-5} + 10^{-5}\|\boldsymbol{u}_k\|_2\big) \ \wedge \ \big(\text{arf}_k^{\boldsymbol{c}} < 1.001\big). \tag{4.62}$$

Implementing these criteria aims to ensure that the nonlinear system is solved with a high degree of accuracy, taking into account the limitations imposed by floating-point computation. Moreover, the application of the skyline metric protects the algorithm from potential fluctuations near the minimum threshold.

4.5 Full algorithm

The complete XDG-IST method as described in the above sections is summarized in Algorithm 2. We have added hyperlinks to the algorithm (indicated by blue letters) so that the reader can navigate the implementation from here. Some parts of the algorithm, like the initialization, are implemented in the equation-specific solvers individually (see Figure 1.1 for an overview). Hence, as there are multiple implementations, no hyperlink is provided for the corresponding parts of the algorithm.

Algorithm 2 Implicit XDG shock tracking method

```
Input: control object defining the solver configuration and all SQP parameters
Output: shock-aligned XDG solution $\boldsymbol{u}^{\star}$ and level set $\boldsymbol{\varphi}^{\star}$
Initialize: grid, level sets, XDG fields, operators
Initial guess: compute $\boldsymbol{\varphi}_0$ and $\boldsymbol{u}_0$ ($P = 0$ solution) as detailed in Section 4.4
for $k = 0, 1, 2, \dots$ do
    System assembly: assemble the linear system (4.22)
    Agglomerate system: agglomerate linear system (Section 4.3.1)
    Solve system: solve (4.22) for $\Delta \boldsymbol{z}_k$ to obtain SQP search direction
    SQP update: determine $\alpha_k$ that satisfies (4.25) using line search and compute $\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \alpha_k \Delta \boldsymbol{z}_k$
    Update regularization/agglomeration: update $\mathcal{A}$ and compute $\gamma_{k+1}$ by (4.32)
    if convergence (4.62) then
        if $P < P_{\text{end}}$ then
            increase polynomial degree ($P = P + 1$)
        else
            terminate
        end if
    end if
    Reinitialize solution: in oscillatory cells the solution is reset to a constant value (Section 4.3.2)
end for
```

4.6 Discretization of the shock level set function

This section elaborates on how the shock level set function φ_s (aka. the shock level set) is discretized using different methods within this study. Previously, φ_s was loosely associated with a coefficient vector $\boldsymbol{\varphi} \in \mathbb{R}^{N_\varphi}$ without detailed explanation. We clarify this by detailing four specific discretization approaches:

1. The DG level set, discussed in Section 4.6.1.
2. The continuous Galerkin (CG) level set, covered in Section 4.6.2.
3. The global level set, presented in Section 4.6.3
4. The spline-based level set, described in Section 4.6.4.

Before we specify the discretization approaches, we discuss a few general requirements.

General requirements In the BoSSS framework, level sets are primarily discretized in the DG space $\mathcal{V}_h^{P_s,1}(\mathcal{K}_h)$ defined over the same mesh as the XDG solution. However, the shock level set implemented in this thesis exists and is optimized within its own space $\mathcal{C}_h^{P_s}$, yet is closely integrated with a distinct base DG level set used by BoSSS routines. Modifications to the shock level set are instantly projected to the base DG level set to ensure the BoSSS framework's data structures remain consistent. The methodologies outlined in this thesis are selected based on their ability to project onto the DG space without losing accuracy, i.e., $\mathcal{C}_h^{P_s} \subseteq \mathcal{V}_h^{P_s,1}$.

Moreover, the discretization strategy should focus on minimizing the DOFs for representing interfaces, which is crucial for efficiently assembling the linear system (4.22). This efficiency is important because the assembly process involves calculating the derivatives of the residuals $\boldsymbol{r}$ and $\boldsymbol{F}$ with respect to the level set coefficients $\boldsymbol{\varphi}$. These derivatives are computed through differentiation of integrals over surfaces/volumes/lines defined implicitly by the level set, observable in Equation (3.28). Lacking an analytical expression to calculate these derivatives, we employ central finite differences, that is,

$$\frac{\partial \boldsymbol{r}}{\partial \varphi_l} \approx \frac{\boldsymbol{r}(\boldsymbol{u}, \ldots, \varphi_{l-1}, \varphi_l + \epsilon, \varphi_{l+1}, \ldots) - \boldsymbol{r}(\boldsymbol{u}, \ldots, \varphi_{l-1}, \varphi_l - \epsilon, \varphi_{l+1}, \ldots)}{2\epsilon}, \tag{4.63}$$

where φ_l represents a component of $\boldsymbol{\varphi}$ and ϵ is a small positive value (commonly 10^{-8}). For each perturbation of one of these components, it is necessary to recalculate quadrature rules for cut-cells, due to a change in the interface position. In the BoSSS framework, the quadrature rules are computed using a costly algorithm based on nonlinear root-finding and developed by Saye (2015). Hence, the full process of computing the derivatives, despite its accuracy, is computationally demanding, especially with a high DOF count of the level set representation. This fact is evident in the performance measurements presented in Section 5.2.6 and when comparing different level set representations 5.2.1.

Additionally, allowing discontinuities or kinks in the interface across cell boundaries adversely affects the optimization process, often leading to stagnation. Hence, achieving a globally continuous or even differentiable interface is desired.

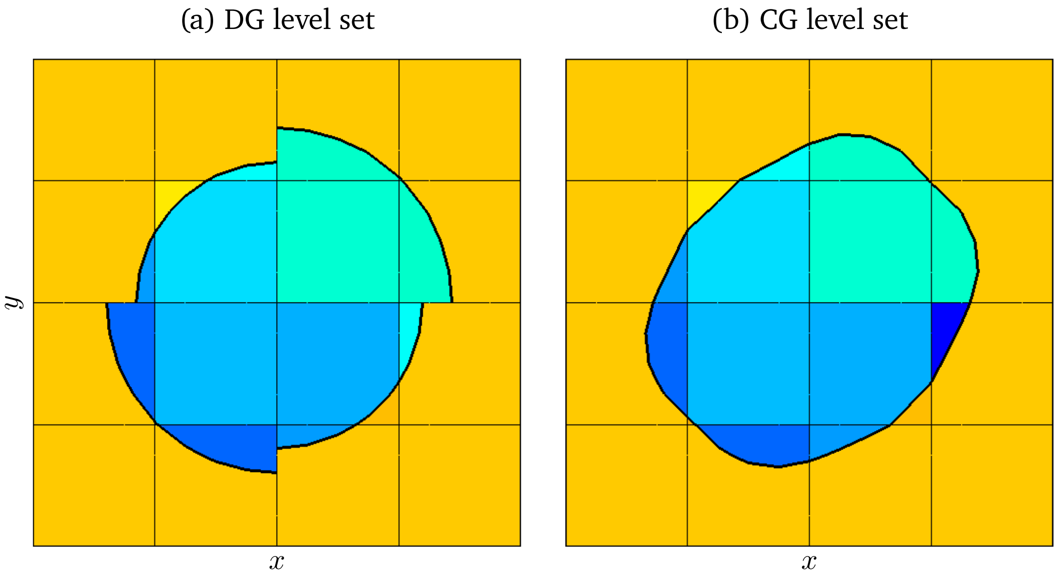


Figure 4.7: Illustration of two level set discretizations of the same level set function, a circle with different radius for each quadrant. Left (a), a DG level set is depicted, featuring discontinuities. Right (b), a CG level set is depicted with a continuous interface.

4.6.1 Discontinuous Galerkin level set

In the context of XDG methods, interfaces are typically described implicitly using level sets φ_s^{DG} within the DG space $\mathcal{V}_h^{P_s,1}(\mathcal{K}_h)$, which shares the computational mesh with the solution.

The first level set employed in our shock tracking framework is a DG level set, which is defined on a sub-mesh $\mathcal{K}_h^s$ of the original mesh $\mathcal{K}_h$ (decreasing the degrees of freedom of the level set). Each cell of the sub-mesh $\mathcal{K}_h^s$ can be expressed as a set union of cells in $\mathcal{K}_h$, i.e.,

$$K_s \in \mathcal{K}_h^s \Rightarrow \exists K_1, \dots, K_n \in \mathcal{K}_h \text{ s.t. } K_s = \bigcup_{i=1}^{n} K_i. \tag{4.64}$$

The resulting level set is represented by its DG coefficients $\boldsymbol{\varphi}^{\mathrm{DG}}$ corresponding to the DG space $\mathcal{V}_h^{P_s,1}(\mathcal{K}_h^s)$.

The DOFs of the DG level set are cell-local, i.e., each component of its coefficient vector $\boldsymbol{\varphi}^{\mathrm{DG}}$ is associated with a cell, and its alteration only influences the cut-cell structure in that specific cell, leading to a block-structure in the Jacobians $\partial \boldsymbol{r}/\partial \boldsymbol{\varphi}, \partial \boldsymbol{F}/\partial \boldsymbol{\varphi}$. However, this positive feature is outweighed by the fact that DG level sets exhibit discontinuous interfaces. Hence, DG level sets have a relative high DOF count compared to other spaces defined on the same mesh, which is due to the high dimensionality of $\mathcal{V}_h^{P_s,1}(\mathcal{K}_h^s)$. Additionally, there is the presence of redundancy, where scalar multiples of the same level set represent the same interface.

4.6.2 Continuous Galerkin level set

Another approach examined in this research are CG level sets. These are defined on the subset of globally continuous functions residing in the DG space, i.e., the CG space defined in (3.16). More specifically, they are constructed from a modal spectral element basis, described in detail in (Karniadakis and Sherwin, 2005). That is, for each cell, nodes

$$\vec{x}^{\mathrm{CG}} := \left\{ \vec{x}_i^{\mathrm{CG}} \;\middle|\; i \in [N_{\boldsymbol{\varphi}}] \right\} \subset \mathbb{R}^d \tag{4.65}$$

are placed on the corners, potentially the edges and inside the cell, merging nodes on common edges for neighboring cells. Then, the basis functions $\boldsymbol{\Phi}^{\mathrm{CG}} := (\boldsymbol{\Phi}_j^{\mathrm{CG}})_{j \in [N_{\boldsymbol{\varphi}}]}$ are determined by their function values at these nodes, which can be expressed by the Kronecker delta

$$\boldsymbol{\Phi}_j^{\mathrm{CG}}(\vec{x}_i^{\mathrm{CG}}) = \delta_{ij}. \tag{4.66}$$

Subsequently, the coordinate vector $\boldsymbol{\varphi}^{\mathrm{CG}}$ of a CG level set simply corresponds to the functional values at the nodes i.e.,

$$\varphi_s^{\mathrm{CG}}(\vec{x}_u) = \boldsymbol{\varphi}_i^{\mathrm{CG}} \text{ and } \varphi_s^{\mathrm{CG}} = \sum_{i=1}^{N_{\boldsymbol{\varphi}}} \boldsymbol{\varphi}_i^{\mathrm{CG}} \boldsymbol{\Phi}_i^{\mathrm{CG}}. \tag{4.67}$$

An illustration for a small 4×4 mesh can be found in Figure 4.7, where a CG level set is depicted alongside a DG level set. It depicts the same function, $\varphi(x,y) = x^2 + y^2 - r_q(x,y)^2$, featuring four different radii $r_q(x,y)$ for each quadrant of the circle, which is projected onto each representation. The DG level set captures the discontinuous interface accurately, while the CG representation inherently enforces continuity of the interface resulting in a different shape.

While for triangular meshes the procedure of constructing CG fields is comparably easy, for Cartesian meshes it involves major difficulties, especially for three-dimensional (3D) meshes or those employing hanging nodes. Also, CG level sets feature comparably many DOFs as DG level sets. As a difference, altering a DOF may change the interface not only in the corresponding cell but also in all neighboring cells, due to the continuity constraints. Lastly, CG level sets also introduce redundant DOFs, having no impact on the interface.

4.6.3 Global level set

A highly effective strategy for reducing the DOFs can be accomplished through the global level set approach. In this method, a set of globally analytical basis functions $\Phi_i^{\mathrm{Gl}} \in C^\infty(\Omega)$ is carefully selected

$$\boldsymbol{\Phi}^{\mathrm{Gl}} := \left(\Phi_i^{\mathrm{Gl}}\right)_{i \in [N_{\boldsymbol{\varphi}}]}, \tag{4.68}$$

and the resulting global level set is defined as a linear combination

$$\varphi_s^{\mathrm{Gl}} = \sum_{i=1}^{N_{\boldsymbol{\varphi}}} \boldsymbol{\varphi}_i^{\mathrm{Gl}} \Phi_i^{\mathrm{Gl}}. \tag{4.69}$$

This discretization approach accommodates various basis functions, making it particularly advantageous when there is prior knowledge of the discontinuity (e.g., straight-sided shocks). The DOFs of the level set are chosen independently of the mesh. Moreover, the resulting function space from this method maintains C^∞ continuity, thereby ensuring smooth shock interfaces and aligning well with optimization algorithms. Nonetheless, while this approach is beneficial for straightforward shock profiles, its applicability to complex scenarios may be limited and it is not used in the numerical experiments in this work.

4.6.4 Spline-based level set

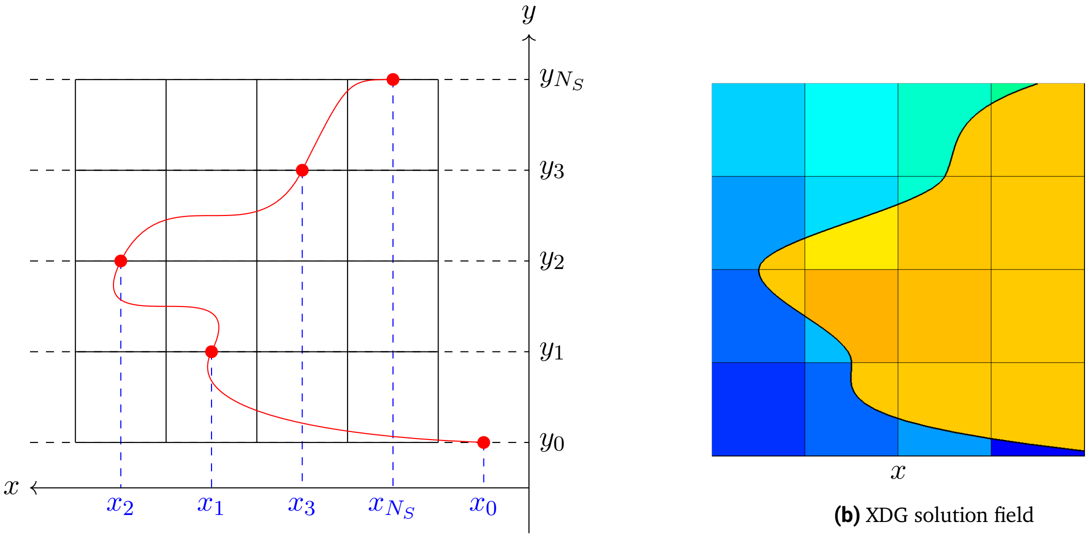
Next, the locality of the CG and DG level sets are combined with a low DOF and C^1-continuous representation of the interface. For this specialized *spline-based level set*, we parameterize the interface explicitly using C^1-continuous splines. The application of the spline-based level set is limited to two-dimensional (2D) conservation laws and Cartesian background grids. In the following description, we assume a space-only problem domain Ω with coordinates (x, y). However, this approach is analogously applicable to 1D space-time problems by replacing the y-coordinates with t-coordinates.

Given a sequence of interpolation points $y_0 < y_1 < \ldots < y_{N_S}$, corresponding to the y-coordinates of the vertices of Cartesian cells within a grid, along with associated values $\{x_0, x_1, ..., x_{N_S}\}$ and derivatives $\{x'_0, x'_1, ..., x'_{N_S}\}$ at these points, a cubic spline $S : [y_0, y_{N_S}] \to \mathbb{R}$ can be defined as follows:

$$S(y_i) = x_i, \;\; S'(y_i) = x'_i, \;\; S|_{[y_i, y_{i+1}]} \in \mathcal{P}_3([y_i, y_{i+1}]). \tag{4.70}$$

This cubic spline function is C^1-smooth over the interval $[y_0, y_S]$ and the corresponding coefficient vector $\boldsymbol{\varphi}^{\mathrm{Sp}}$ is the concatenation of th function values $(x_i)_i$ and the derivatives $(x'_i)_i$, i.e.,

$$\boldsymbol{\varphi}^{\mathrm{Sp}} := \begin{pmatrix} (x_i)_i \\ (x'_i)_i \end{pmatrix}, \text{with } N_{\boldsymbol{\varphi}} = 2N_S. \tag{4.71}$$



(b) XDG solution field

(a) Spline height function

Figure 4.8: *Left*: (a) Illustration of a cubic C^1-spline $S : [y_0, y_{N_S}] \to \mathbb{R}$ (*red line*) defined by interpolation points $\{y_0, y_1, \ldots, y_{N_S}\}$ and corresponding values $\{S(y_i) = x_i\}$ (*blue dashed line*) on a 4×4 mesh. Here, the derivatives $S'(y_i) = x_i'$ are not depicted. The corresponding level set is defined by $\varphi(x, y) = x - S(y)$. *Right*: (b) The (roughly) corresponding XDG solution field. (Left figure adapted from Vandergrift and Kummer, 2024)

Figure 4.8 illustrates such a spline for a 4×4 mesh. Further, it can be linked with the DG representation of the shock level set by projecting the function

$$\varphi_s^{\mathsf{Sp}}(x, y) = x - S(y) \qquad (4.72)$$

onto the DG space $\mathcal{V}_h^{P_s,1}(\mathcal{K}_h)$.

The spline-based level set representation requires a fixed number of two DOF per grid point in y-direction, significantly fewer than a DG or CG field of polynomial order three would entail (a concrete illustration is provided in a later section in Figure 5.23). Moreover, it explicitly represents the shock front, such that quadrature rules could be computed by an analytical formula and the Jacobians $\partial \boldsymbol{r}/\partial \boldsymbol{\varphi}, \partial \boldsymbol{R}/\partial \boldsymbol{\varphi}$ could be obtained with greater ease. Additionally, a huge benefit in terms of solver robustness was observed from the enforced C^1-continuity at the interface. Although this spline representation can be extended to any polynomial order (e.g., a continuous linear spline by omitting the derivative information), this work primarily employs cubic and linear splines.

Note that the approach above is limited to shocks that are graphs of 1D height functions. However, this approach is worth considering, since there are numerous flow instances that induce such shocks.

5 Numerical experiments for implicit XDG shock tracking

This chapter presents a comprehensive suite of numerical experiments designed to validate and scrutinize the implicit XDG shock tracking (XDG-IST) method introduced in this work. Through a series of test cases and in-depth studies, we explore the method's applicability, accuracy, robustness, and computational efficiency across a range of challenging flow scenarios.

Some of the results in this chapter have been published in a similar fashion by Vandergrift and Kummer (2024). But, our actual implementation of the XDG-IST method and the underlying code base have been modified slightly, due to continuing research. All the results in this chapter are created using a more recent version of the BoSSS code Kummer et al. (2024) and might slightly differ from the results presented in the publication by Vandergrift and Kummer (2024).

The chapter is outlined as follows: first, we define test cases (Section 5.1) for each system of conservation laws considered, featuring both one-dimensional (1D) space-time and space-only two-dimensional (2D) scenarios, and we provide corresponding illustrative results from applying the XDG-IST method. Following the presentation of test cases, we show results from comprehensive studies aimed at further evaluating and optimizing the method (Section 5.2). At the end, we conclude the chapter (Section 5.3).

5.1 Test cases

In this section, we present a series of test cases and apply the XDG-IST method to these. The presented results can be reproduced by the publicly available Jupyter notebooks (see Remark 14).

Specifically, Section 5.1.1 focuses on problems based on the 1D space-time advection equation, presenting results for simulations with straight-sided and curved shocks. In Section 5.1.2, problems arising for the 1D space-time Burgers equation are examined, also showcasing results for both straight-sided and curved shock configurations. Cases based on the 2D steady Euler equations are discussed in Section 5.1.3, where flow phenomena such as supersonic flow over an inclined plane and a bow shock are introduced. Lastly, Section 5.1.4 features examples for the 1D space-time Euler equations, highlighting the method's capability to accurately model shock-acoustic wave interactions.

Remark 14. *All of the computations underlying the presented results in this chapter can be reproduced by executing the corresponding Jupyter notebooks from the public BoSSS repository Kummer et al., 2024. The notebooks are stored under the path ./examples/ShockFitting with* *sub-folders corresponding to the specific equations, problems and studies. At the beginning of each test case, we also provide a hyperlink in the form of blue colored text, sending the reader to the corresponding notebook.*

5.1.1 One-dimensional space-time advection equation

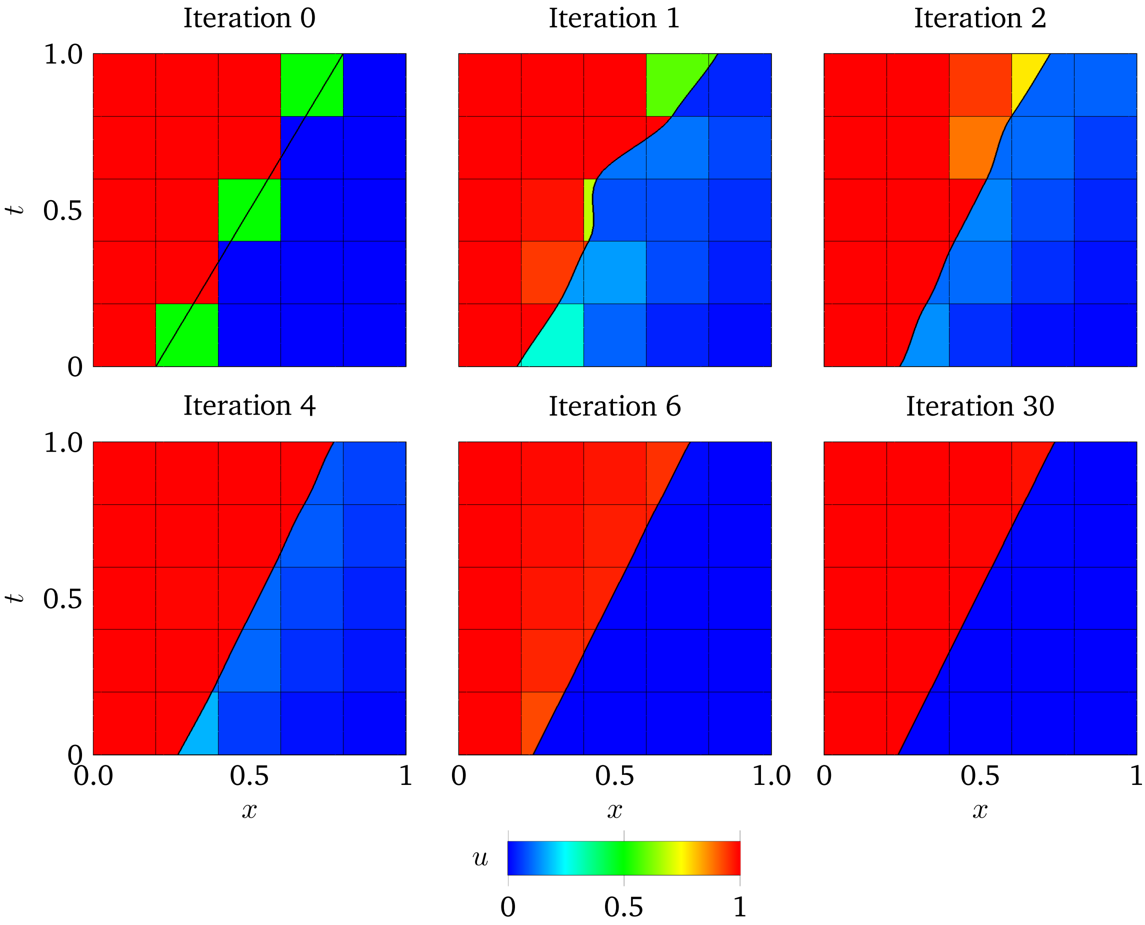


Figure 5.1: Plots of velocity u for selected XDG shock tracking iterations $\boldsymbol{z}_k$ ($k = 0, 1, 2, 4, 6, 30$, polynomial degree $P = 0$) for the space-time advection test case with a straight-sided shock. The straight-sided discontinuity is tracked by the SQP solver starting from an initial guess for the shock interface $\mathfrak{I}_s$ (*thick black line*), which is close to the correct position and implicitly defined by a cubic spline level set ($P_s = 3$). The XDG-IST solver converges to the solution of the scalar conservation law and successfully aligns the cubic spline level set with the discontinuity.

First, two problems for the space-time formulation of the linear advection equation, which has been introduced in Section 3.4.1, are presented. We consider a rectangular space-time domain $\Omega = [0, 1] \times [0, 1]$ and we introduce a discontinuity in the initial value

$$u(x, 0) = \begin{cases} 1 & \text{if } x < \frac{1}{4} \\ 0 & \text{else }, \end{cases} \tag{5.1}$$

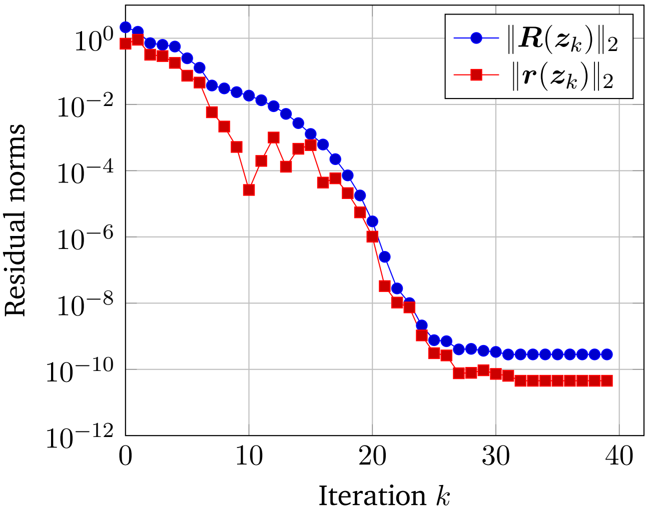


Figure 5.2: Optimization history of the XDG-IST method for the space-time advection test case (polynomial degree $P = 0$) with a straight-sided shock. Both residual norms $\|\boldsymbol{R}\|_2$, $\|\boldsymbol{r}\|_2$ converge rapidly after 30 iterations as the SQP solver successfully tracks the straight-sided discontinuity.

which will be advected in time in dependency of the advection field $u_a(t)$. We prescribe Dirichlet boundary conditions using the exact solution and examine two choices of the advection field, one resulting in a straight-sided shock and another one resulting in a curved shock.

Advection - straight-sided shock To simulate a straight-sided shock, the advection field is defined as a constant function over time, given by

$$u_a(t) = \frac{1}{2}. \tag{5.2}$$

(This configuration ensures the shock advances at a constant velocity of $1/2$.) Utilizing an XDG space of polynomial order $P = 0$ with a linear shock level set, the piece-wise constant space-time solution is exactly representable. The shock's location can then be represented by the exact level set function

$$\varphi_s(x,t) = x - \frac{1}{4} - \frac{1}{2}t. \tag{5.3}$$

This example serves as a test of the method's capability in solving a linear 2D problem with a simple solution, representable within the XDG space.

Applying the XDG-IST method with the full enriched residual objective function f_{ER}, we employ a 5×5 Cartesian mesh, polynomial degree $P = 0$, a cubic spline-based level set ($P_s = 3$) for the shock, and the upwind numerical flux, as defined in (3.43). The initial configuration of the level set is determined by projecting the arbitrarily chosen function

$$\varphi_0(x,t) = x - \frac{2}{5} - \frac{3}{5}t \tag{5.4}$$

onto the spline-based level set, with the initial solution (first guess) being a projection of the exact solution. Figure 5.1 illustrates selected states $\boldsymbol{z}_k$ at various stages during the optimization. Notably, the solver closely approximates the true solution after just three iterations, achieving full convergence after 30 iterations as indicated by the residual norms $\|\boldsymbol{r}(\boldsymbol{z}_k)\|_2, \|\boldsymbol{R}(\boldsymbol{z}_k)\|_2$ plateauing around 10^{-9} in Figure 5.2.

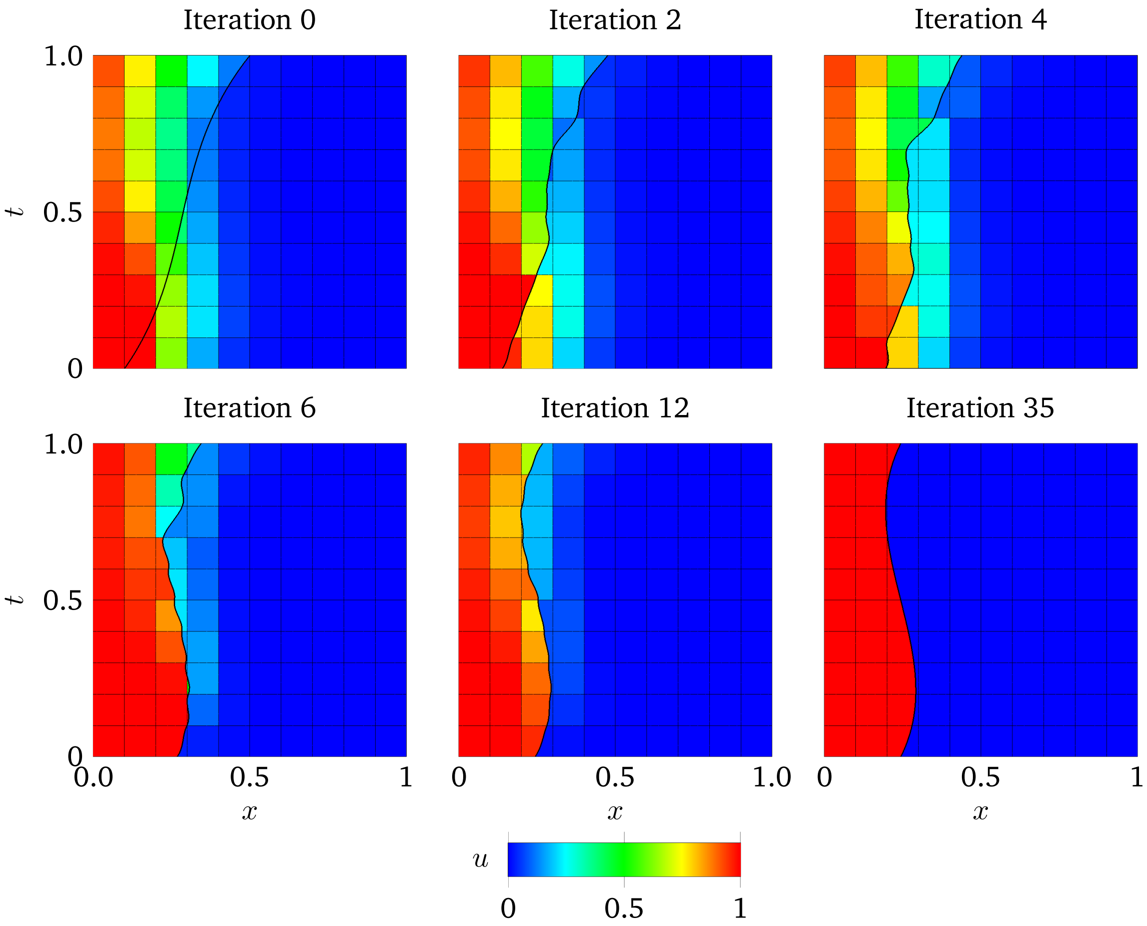


Figure 5.3: Plots of velocity u for selected XDG shock tracking iterations z_k ($k = 0, 2, 4, 6, 12, 35$, polynomial degree $P = 0$) for the space-time advection test case with a curved shock. The curved polynomial discontinuity is tracked by the SQP solver starting from an initial guess for the shock interface $\mathfrak{I}_s$ (*thick black line*), implicitly defined by a cubic spline level set ($P_s = 3$) which is not sub-cell accurate. Simultaneously, the solution of the scalar conservation law is obtained and the cubic spline level set is successfully aligned with the discontinuity.

Advection - curved shock For the curved shock scenario, the advection field is specified as a polynomial function in time

$$u_a(t) := 3t^2 - 3t + \frac{1}{2}. \tag{5.5}$$

This setup allows the representation of the piece-wise constant space-time solution exactly within an XDG space of order $P = 0$, employing a shock level set of degree $P_s = 3$. The shock's trajectory is described exactly by

$$\varphi_s(x,t) = x - t^3 + \frac{3}{2}t^2 - \frac{1}{2}t - \frac{1}{4}. \tag{5.6}$$

This example serves as a test of the method's capability in solving a linear 2D problem with a curved polynomial discontinuity, representable within the XDG space. Here, the initial guess for the shock level set is chosen more than one cell-distance away from the exact position, to make the case more challenging.

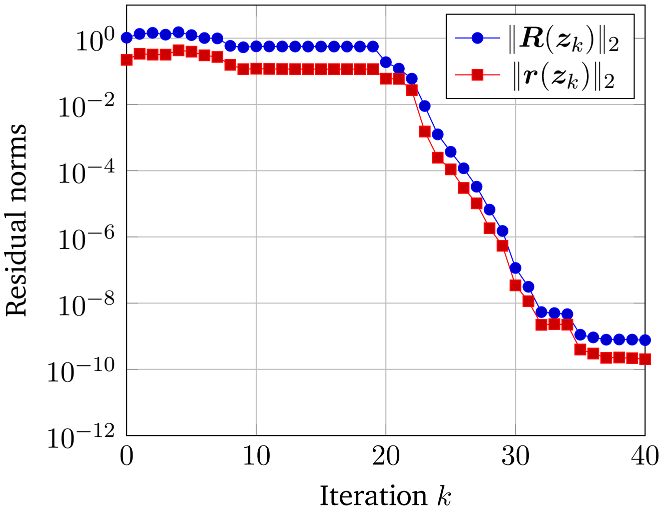


Figure 5.4: Optimization history of the XDG-IST method for the space-time advection test case with a curved shock (polynomial degrees $P = 0, P_s = 3$). Both residual norms $\|\boldsymbol{R}\|_2, \|\boldsymbol{r}\|_2$ converge rapidly after 35 iterations as the SQP solver successfully tracks the polynomial discontinuity.

We apply the XDG-IST method with the objective function f_{ER} on a 10×10 Cartesian mesh, a cubic spline-based level set, a $(P = 0)$ XDG space, and by incorporating the upwind flux from (3.43). The initial state of the level set is determined by projecting

$$\varphi_0(x,t) = x - \frac{7}{10}t^3 + t^2 - \frac{7}{10}t - \frac{1}{10}. \tag{5.7}$$

and the initial solution state is a projection of the exact solution. In Figure 5.3, we present selected optimization states $\boldsymbol{z}_k$, showing the solver's rapid approach to the actual solution and confirming its convergence after 35 iterations. The optimization process's residual norms $\|\boldsymbol{r}(\boldsymbol{z}_k)\|_2, \|\boldsymbol{R}(\boldsymbol{z}_k)\|_2$ stabilize at approximately 10^{-9}, as depicted in Figure 5.4.

5.1.2 One-dimensional space-time Burgers equation

Next, two problems for the space-time formulation of the Burgers equation, which has been introduced in Section 3.4.2, are presented. We construct two cases with exact solutions: one with a piece-wise constant solution and a straight-sided discontinuity, and a second with a curved discontinuity and a non-polynomial solution. In both cases, Dirichlet boundary conditions are imposed weakly using the exact solution.

Burgers - straight-sided shock For the straight-sided case a rectangular space-time domain $\Omega = [0,1] \times [0,1]$ and a piece-wise constant initial value with a discontinuity

$$u(x,0) = \begin{cases} \frac{3}{4} & \text{if } x < \frac{1}{4} \\ \frac{1}{4} & \text{else} \end{cases} \tag{5.8}$$

are chosen. Using the Rankine-Hugoniot jump conditions the shock speed can be determined to be $1/2$ and a straight-sided shock is obtained, representable by the level set

$$\varphi_s(x,t) = x - \frac{1}{4} - \frac{1}{2}t. \tag{5.9}$$

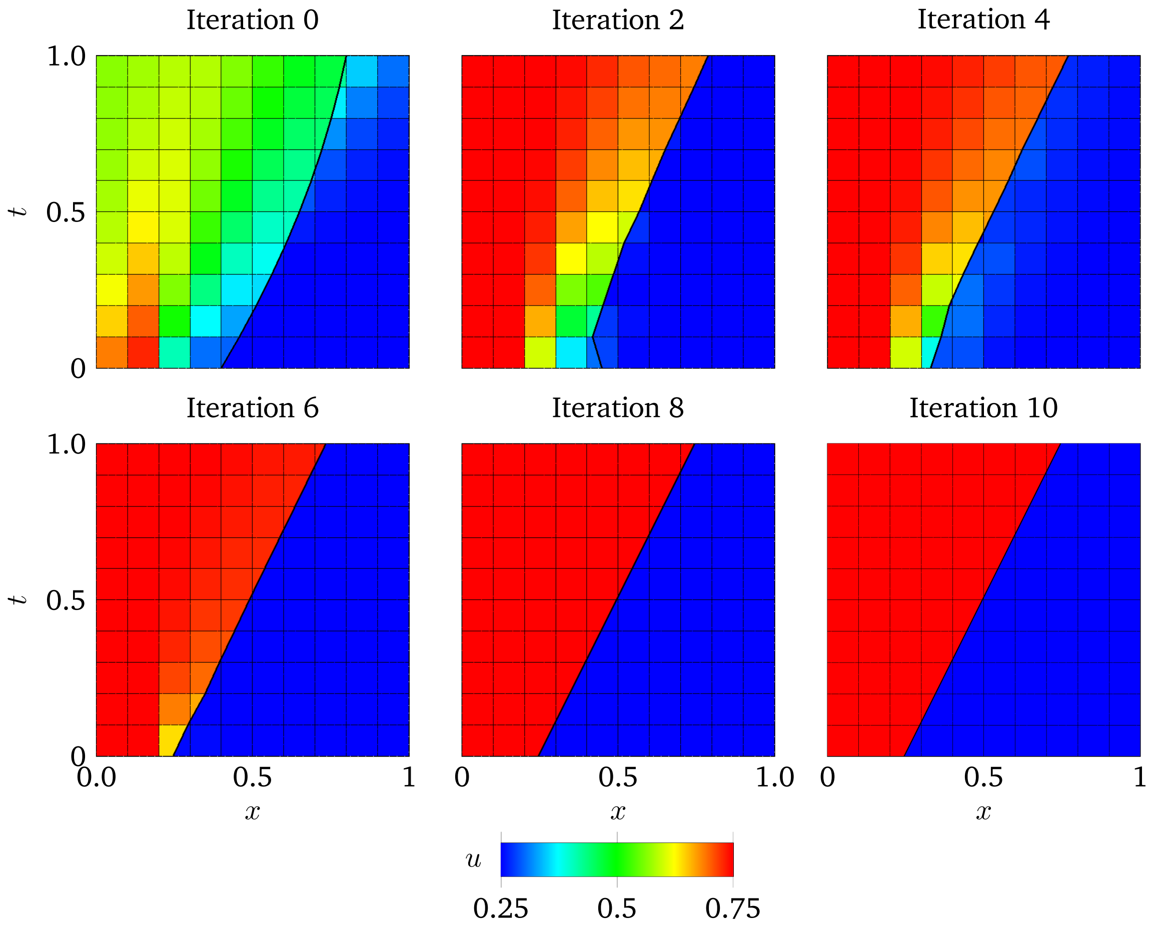


Figure 5.5: Plots of velocity u for selected XDG shock tracking iterations $\boldsymbol{z}_k$ ($k = 0, 2, 4, 6, 8, 10$, polynomial degree $P = 0$) for the space-time Burgers test case with a straight-sided shock. The straight-sided discontinuity is tracked by the SQP solver starting from an curved initial guess for the shock interface $\mathfrak{I}_s$ (*thick black line*), implicitly defined by a linear spline level set ($P_s = 1$). The XDG-IST solver converges the solution of the scalar conservation law and successfully aligns the linear spline level set with the discontinuity.

This example serves as a test of the methods capability in solving a nonlinear 2D problem with a simple solution, representable within the XDG space.

Analogously to the advection test case, we employ the XDG-IST method with the full enriched residual objective function f_{ER}, a 10×10 background mesh, a $P = 0$ extended discontinuous Galerkin (XDG) space, a linear spline-based level set ($P_s = 1$), and the upwind numerical flux defined in (3.43). The initial configuration of the level set is determined by projecting

$$\varphi_0(x,t) = x + \frac{1}{5}t^2 - \frac{3}{5}t - \frac{2}{5}, \tag{5.10}$$

onto the spline-based level set, with the initial solution being a projection of the exact solution. In Figure 5.5, we present selected optimization states $\boldsymbol{z}_k$, showing the solver's rapid approximation of the actual solution and underlining its convergence after remarkable 12 iterations. The residual norms $\|\boldsymbol{r}(\boldsymbol{z}_k)\|_2, \|\boldsymbol{R}(\boldsymbol{z}_k)\|_2$ stabilize during the optimization history at approximately 10^{-15}, as depicted in Figure 5.6.

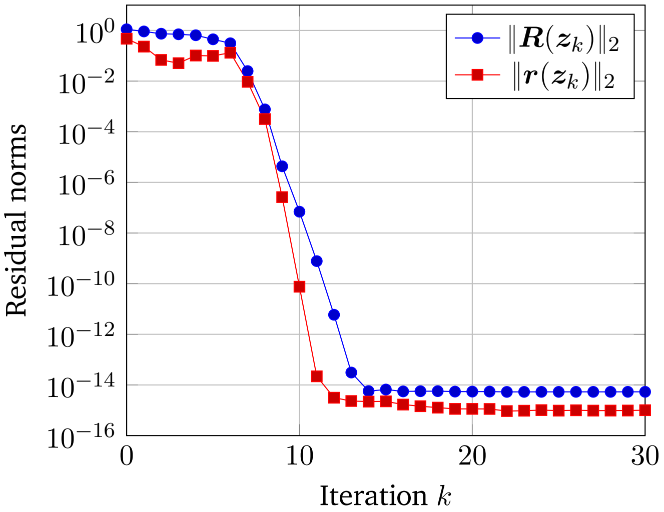


Figure 5.6: Optimization history of the XDG-IST method for the straight shock Burgers test case ($P = 0$). Both residual norms $\|\boldsymbol{R}\|_2, \|\boldsymbol{r}\|_2$ converge rapidly after 12 iterations while the SQP solver successfully tracks the straight discontinuity using a linear spline level set.

Interestingly, the residual of the XDG solution for the nonlinear Burgers test case is significantly lower compared to the residuals from the two linear advection test cases. This difference may stem from the type of spline-based level sets used; cubic spline-based level sets are employed in the advection cases, while a linear spline-based level set is used in the Burgers case. When the first advection test case was run with a linear spline-based level set, we observed residuals converging to approximately 10^{-15}. It is important to note, as mentioned in Section 4.6, that the level set Jacobians are computed using finite differences with an accuracy limit of 10^{-8}. This limitation likely affects the convergence accuracy of the derivatives $x_i{}'$ at the interpolation points y_i (defined in (4.70)), potentially explaining the observed discrepancies in residuals across different test cases.

Burgers - curved shock Next, we introduce a case for the Burgers equation with a curved shock and a non-polynomial solution. Following Huang and Zahr (2022), we consider the domain $\Omega = [-0.2, 1] \times [0, 1.2]$ and the following discontinuous initial condition

$$u(x,0) = \begin{cases} 4 & \text{if } x < 0 \\ 3(x-1) & \text{else.} \end{cases} \tag{5.11}$$

For this case, the exact solution is known to be

$$u_{\text{ex}}(x,t) = \begin{cases} 4 & \text{if } x < x_s(t) \\ \frac{3(x-1)}{1+3t} & \text{else,} \end{cases} \tag{5.12}$$

where the accelerating shock speed x_s is given by

$$x_s(t) = \frac{7}{4}\left(1 - \sqrt{1+3t}\right) + 4t. \tag{5.13}$$

This example serves as a test of the method's capability in solving a challenging nonlinear 2D problem where the solution cannot be precisely represented within the XDG space. Unlike the

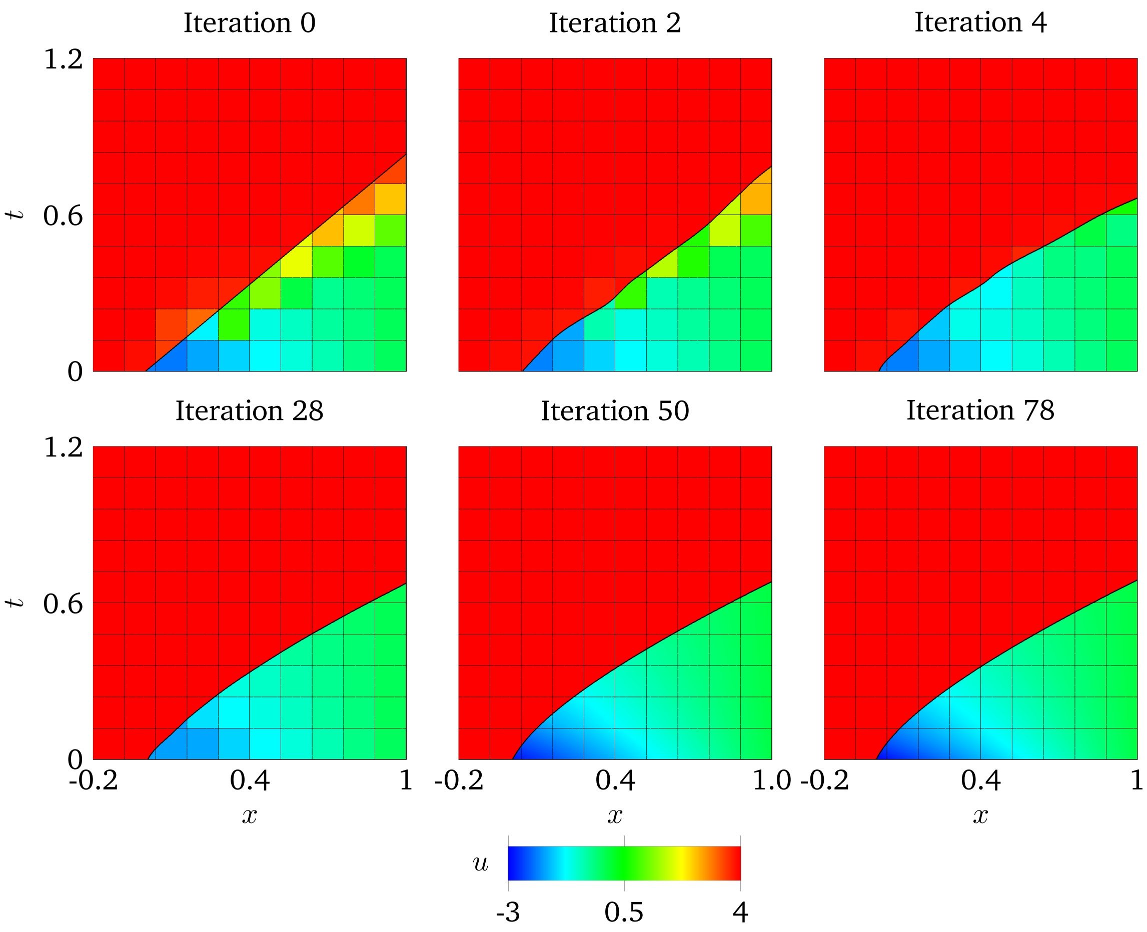


Figure 5.7: Plots of velocity u for selected XDG shock tracking iterations $\boldsymbol{z}_k$ (polynomial degrees $P = 0, 1, 2, 3$) for the accelerating shock burgers test case for $k = 0, 2, 4, 28, 50, 78$. Here, as an initial guess for the shock interface $\mathfrak{I}_s$ (*thick black line*), implicitly defined by a cubic spline level set ($P_s = 3$), a linear interface is employed and aligned to the discontinuity by the XDG-IST solver, simultaneously computing the solution of the conservation law. Here, a P-continuation strategy is employed gradually increasing the polynomial degrees.

previous cases, both the interface and the solution are non-polynomial, adding an extra layer of complexity.

The XDG-IST method is applied using a 10×10 background grid, the full enriched residual objective function f_{ER}, a cubic spline-based level set ($P_s = 3$), and the upwind numerical flux defined in (3.43). The polynomial degree is chosen as $P = 3$ and a P-continuation strategy (see Section 4.3.3) is employed. The initial guess for the level set is chosen to be

$$\varphi_0(x, t) = x - \frac{6}{5}t, \tag{5.14}$$

and the initial flow solution is computed using pseudo-transient continuation (see Section 4.4.2).

Selected XDG-IST iterations are depicted in Figure 5.7 and the corresponding optimization history of the residual norms $\|\boldsymbol{R}(\boldsymbol{z}_k)\|_2, \|\boldsymbol{r}(\boldsymbol{z}_k)\|_2$ is shown in Figure 5.8 showcasing success-

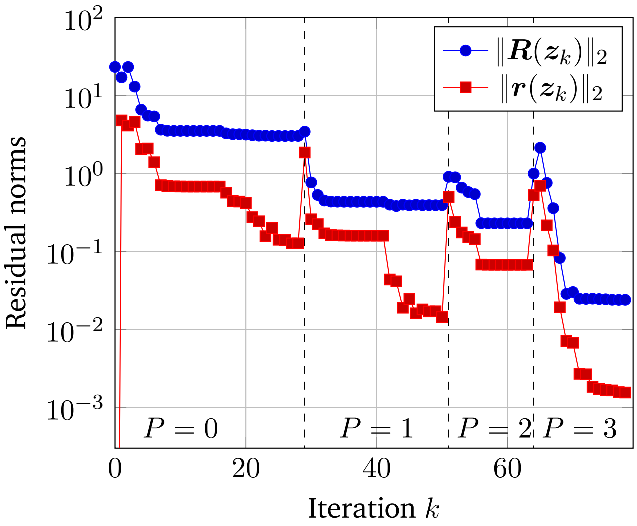


Figure 5.8: Optimization history of the XDG-IST method for the space-time Burgers test case with a curved shock, where a P-continuation strategy is employed (polynomial degree $P = 0, 1, 2, 3$) showing the residual norms $\|\boldsymbol{R}\|_2, \|\boldsymbol{r}\|_2$ for all SQP iterations. After converging the residual of an intermediate polynomial degree, the latter is increased, such that a jump in both residuals $\boldsymbol{R}, \boldsymbol{r}$ can be observed (*dashed lines*). We have $\boldsymbol{r}(\boldsymbol{z}_0) \approx 10^{-16}$ due to the initial value coming from pseudo-transient-continuation.

ful alignment of the XDG solution with the shock and convergence after 78 iterations to $\|\boldsymbol{r}(\boldsymbol{z}_k)\|_2 \approx 2 \times 10^{-3}$. Note, that the residuals are computed employing XDG spaces relative to the polynomial degree of the iterate $\boldsymbol{z}_k$ which increases during the optimization process due to the P-continuation. Hence, at iterations where the polynomial degree is increased the residuals exhibit an uptick, observable in Figure 5.8.

Note that the residual plateau reached in this case is significantly higher compared to the cases presented previously. This discrepancy arises because, unlike in the other cases, the solution for this case cannot be exactly represented within the XDG space used, due to the non-polynomial nature of both the shock and flow solutions. Even when projecting the analytical solution onto the XDG space, we obtained residuals in the same range, approximately 10^{-3}, as those computed by the shock tracking method. It is only through refining the mesh and significantly increasing the polynomial degree that the residuals gradually approach zero. This observation suggests that there may either be an inconsistency in the discretization (e.g., no solution exists that satisfies $\boldsymbol{r} \approx 10^{-16}$ for polynomial degrees $P > 0$), or that the solver, tasked with minimizing both the residual and the objective function, can not find a solution with $\boldsymbol{r} \approx 10^{-16}$ because doing so would simultaneously increase the objective function.

5.1.3 Two-dimensional steady Euler equations

Going further, we introduce two cases for two-dimensional stationary supersonic flows, i.e., problems based on the steady 2D Euler equations discussed in Section 3.4.3. In opposition to the other cases presented in this section, a secondary boundary level set φ_b is employed to represent geometries immersed into the flow field, eventually being responsible for the creation of shock waves.

We start the discussion by giving details on the setup considered for both simulations. Firstly, the full enriched residual objective function f_{ER} is utilized. Secondly, for the numerical flux function $\mathcal{H}$, employed for cell boundaries Γ, we use the following:

- For boundaries not coinciding with any interface, specifically $(\Gamma \backslash \mathfrak{I}_s) \backslash \mathfrak{I}_b$, the Harten-Lax-van Leer-Compact (HLLC) flux is employed. This approach is directly integrated with the imposition of specific boundary conditions at the domain's edges. These conditions entail the application of supersonic inlet conditions $(\rho_\infty, p_\infty, u_\infty, v_\infty)$ at the domain's left boundary, supersonic outlet conditions at the right boundary, and adiabatic slip wall conditions at both the top and bottom boundaries (see Section 3.4.3 for details on boundary conditions in the 2D Euler context).
- The Godunov flux method is specifically utilized for the edges that are part of the shock interface, denoted as $\mathfrak{I}_s$. (Varying this choice will be studied in Section 5.2.3)
- For the edges associated with the immersed boundary interface $\mathfrak{I}_b$ adiabatic slip wall conditions are uniformly applied.

The supersonic inlet conditions on the left domain boundary are chosen to be solely dependent on the free-stream Mach number Ma_∞, as we assume no inflow in y-direction and normalize pressure and density, i.e.,

$$\rho_\infty = 1, \; p_\infty = 1, \; u_\infty = \text{Ma}_\infty \sqrt{\hat{\gamma}}, \; v_\infty = 0. \tag{5.15}$$

To determine the precision of the numerical solutions, the computed enthalpy values are compared with the free-stream enthalpy H_∞, known for being constant in stationary inviscid compressible flows. The free-stream enthalpy is derived from the inflow conditions, namely the inflow density ρ_∞, pressure p_∞, and Mach number Ma_∞, using the formula:

$$H_\infty = \frac{\hat{\gamma}}{\hat{\gamma} - 1} \frac{p_\infty}{\rho_\infty} + \frac{1}{2} \text{Ma}_\infty^2 \hat{\gamma} \frac{p_\infty}{\rho_\infty}. \tag{5.16}$$

Variations from this established enthalpy benchmark are indicative of the numerical method's fidelity.

In pursuit of a quantifiable measure of accuracy, the normalized L^2-error for enthalpy across the fluid domain is determined by

$$H_{\text{err}} = \frac{1}{\|H_\infty\|_{L^2}} \sqrt{\int_\Omega (H - H_\infty)^2 dV}, \tag{5.17}$$

where H represents the enthalpy values derived from the numerical simulation. In the following, we also denote by $H_{\text{err}}(\boldsymbol{z}_k)$ the enthalpy error derived for the SQP iterate $\boldsymbol{z}_k$.

Euler - Mach 2 flow over inclined plane The first test case examined details a stationary supersonic Mach 2 flow over an inclined plane, a configuration previously investigated in Zahr et al. (2020). This scenario is set within a rectangular space-only domain $\Omega = [0, \frac{3}{2}] \times [0, 1]$, featuring supersonic flow impinging on an inclined plane at an angle $\theta_{\text{wdg}} > 0$. The uniform inflow, when interacting with the wedge's geometry, creates a solution characterized by piecewise constant flow parameters, with the resulting shock wave maintaining a linear trajectory.

In this particular setup, leveraging relations outlined in Section 2.2.2 enables the explicit calculation of both the shock wave's angle and the flow solution. Specifically, for an inflow Mach number $M_\infty = 2$ and a wedge angle $\theta_{\text{wdg}} = 10°$, the shock wave angle is determined to be approximately $\theta_{\text{shk}} \approx 39.31°$. In this setup, two level sets are utilized: a fixed one, denoted as φ_b, representing the immersed boundary and a variable one, denoted as φ_s, representing the shock. The wedge's geometry can be accurately represented by the zero iso-contour of the following linear level set

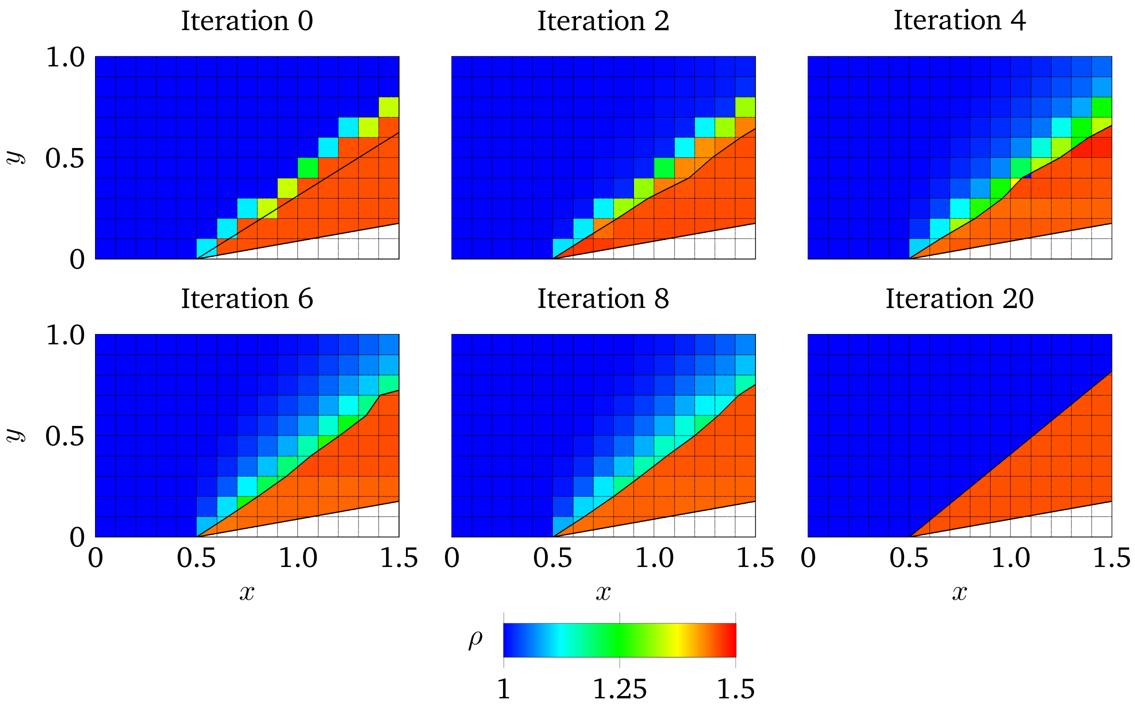


Figure 5.9: Selected XDG shock tracking iterations z_k ($k = 0, 2, 4, 6, 8, 20$, polynomial degree $P = 0$) for the density ρ (of a Mach 2 flow over an inclined plane (*indicated by white filling*) with angle $\theta_{\text{wdg}} = 10°$ for a 15×10 mesh, represented by an immersed boundary $\mathfrak{I}_b$ (*lower thick black line*, $P_b = 1$). The shock interface $\mathfrak{I}_s$ (*upper thick black line*) is represented by a linear spline level set φ_s ($P_s = 1$) and is converged by the SQP solver to the correct shock position in about 20 iterations.

$$\varphi_b(x, y) = \frac{1}{2} + \frac{y}{\tan(\frac{10°\pi}{180°})} - x. \tag{5.18}$$

Similarly, the exact shock would be approximately represented by the linear level set

$$\varphi_s(x, y) = \frac{1}{2} + \frac{y}{\tan(\frac{39.31°\pi}{180°})} - x. \tag{5.19}$$

The XDG-IST method is applied, utilizing a 15×10 background grid for the domain discretization and employing an initial projection of the exact solution (see Section 2.2.2 for details) for the initial guess. Due to the piece-wise constant nature of the flow, a polynomial degree of $P = 0$ is sufficient for representing the solution, along with linear level sets ($P_s = P_b = 1$). Thus, a linear spline-based level set with an initial angle of $32°$ is chosen, intentionally creating a configuration where the level set is not sub-cell accurate.

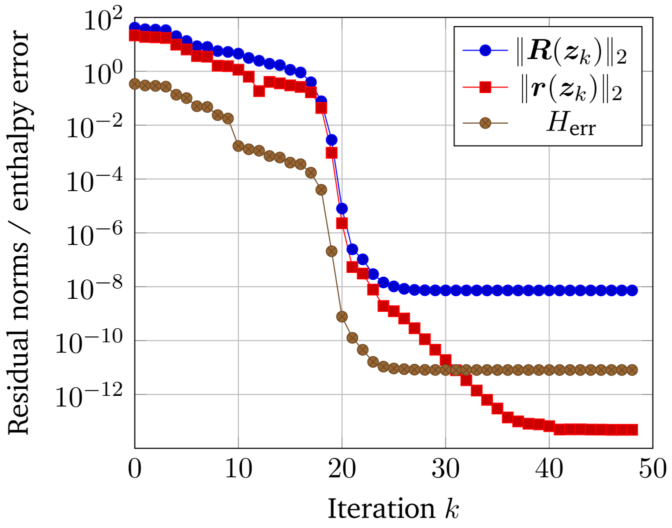


Figure 5.10: Optimization history of the XDG-IST method for the supersonic Mach 2 flow over an inclined plane $(P = 0)$ showing the residual norms $\|\boldsymbol{R}\|_2, \|\boldsymbol{r}\|_2$ and the normalized enthalpy error H_{err} across all SQP iterations. An convergence of both residuals and the enthalpy error can be measured after 25 iterations as the SQP solver tracks the straight discontinuity and computes the constant solution.

Remarkably, applying the XDG-IST method demonstrates impressive convergence, with the residual norms reaching approximately 1×10^{-12} after about 25 iterations. Furthermore, the enthalpy error decreases to around 10^{-11}, with the exact enthalpy $H_\infty = 6.3$ serving as the reference. Consequently, it can be assumed that the shock is accurately tracked, and the correct solution is computed with errors primarily attributed to rounding. For illustrative purposes, selected plots of solution states at different solver iterations are depicted in Figure 5.9 and the history of the residual norms and the enthalpy error is shown in Figure 5.10.

It is worth noting that this particular test case involves doubly cut-cells where the wedge and the shock intersect, as these cells pose challenges for the generation of quadrature rules. Recently developed algorithms by Saye (2022) and Beck and Kummer (2023), the latter of which is implemented in the BoSSS framework, have been instrumental in handling such cases. These algorithms build upon the same quadrature rules that are utilized for single cut-cells and have been introduced by Saye (2015).

Euler - Mach 4 bow shock The second test case is a supersonic flow impinging on a blunt body, marked by a curved geometry and a corresponding curved shock front, as outlined in the framework of the 5th International Workshop on High Order CFD Methods (HiOCFD5) (Murman, 2017). This scenario's geometry is a rounded rectangle, crafted from the segments of two circles centered at $\vec{c}_1 = (x_1, y_1) = (0, 0.5)$ and $\vec{c}_2 = (x_2, y_2) = (0, -0.5)$, each having a radius of 0.5. To complete the blunt body's form, a vertical line segment is introduced at $\{x = -0.5, y \in [-0.5, 0.5]\}$. A continuous level set function $\varphi_b \in C^1(\Omega)$ in its quadratic expression is employed to represent this geometry accurately

$$\varphi_b(\vec{x}) = \begin{cases} \|\vec{x} - \vec{c}_1\|_2^2 - 0.25 & \text{if } 0.5 \leqslant y \\ x^2 - 0.25 & \text{if } 0.5 < y \leqslant -0.5 \\ \|\vec{x} - \vec{c}_2\|_2^2 - 0.25 & \text{else,} \end{cases} \tag{5.20}$$

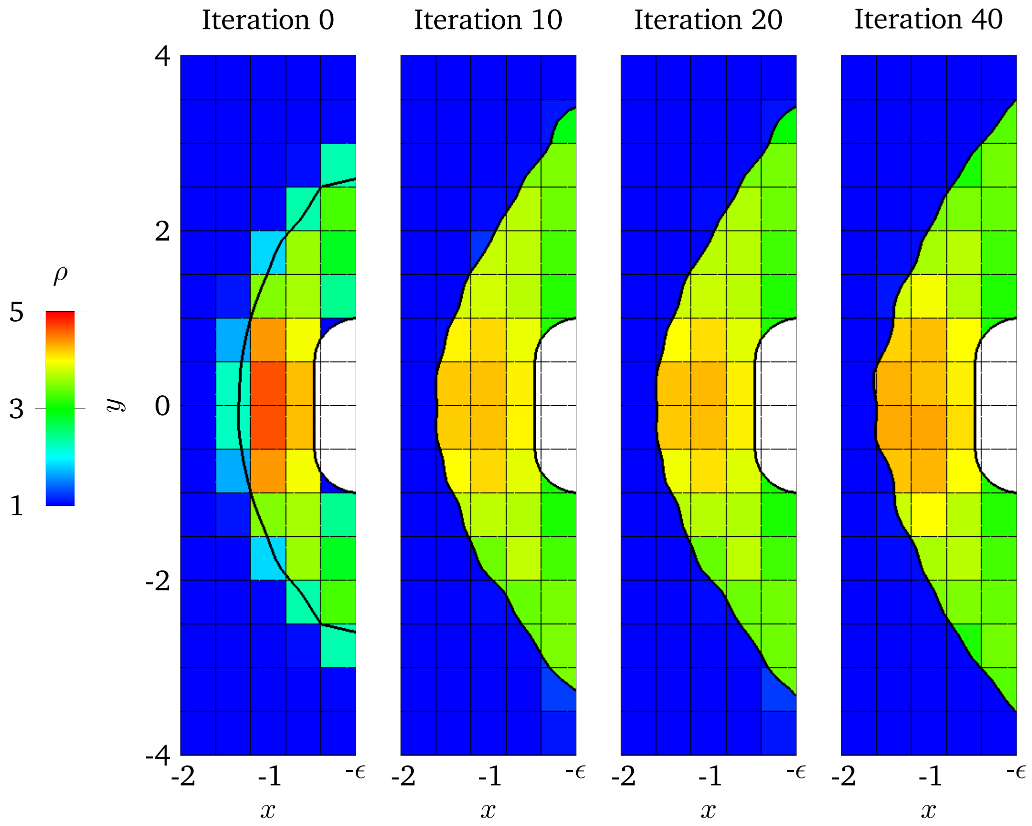


Figure 5.11: Selected XDG shock tracking iterations $\boldsymbol{z}_k$ ($k = 0, 10, 20, 40$, polynomial degree $P = 0$) for the density ρ of the Mach 4 bow shock for a 5×16 mesh. The blunt body (*white filling*) is represented by an immersed boundary $\mathfrak{I}_b$ (*right thick black line*) using a quadratic level set. Starting from an initial guess obtained by a reconstruction procedure, the shock interface $\mathfrak{I}_s$ (*left thick black line*), represented by a cubic spline level set ($P_s = 3$), and the XDG approximation are converged by the SQP solver (see Figure 5.12 for $P > 0$ iterations).

which achieves precise representation within a second-order discontinuous Galerkin (DG) space ($P_b = 2$), particularly when the mesh aligns with the horizontal lines at $y = 0.5$ and $y = -0.5$. The conditions at the inflow are defined as supersonic, with a Mach number $M_\infty = 4.0$. The blunt body's boundary is treated with adiabatic slip-wall conditions.

The computational domain is chosen as $\Omega = [-2 - \epsilon, -\epsilon] \times [-4, 4]$, introducing a minor translation ($\epsilon = 0.0025$) in the x-direction. This modification aims to avoid an exact alignment of the shock interface with the vertical portion of the blunt body at $x = -0.5$, promoting an optimal cut-cell configuration. For the shock interface $\mathfrak{I}_s$ representation, a cubic spline-based level set ($P_s = 3$) is utilized, ensuring an accurate representation of the curved shock.

The initial guess for the solution and the level set function is obtained from a reconstruction procedure based on a $P = 2$ shock capturing simulation, as proposed by Geisenhofer (2021) and outlined in Section 4.4.2. Additionally, we employ the P-continuation strategy, starting with $P = 0$ and progressively increasing the degree of the solution until $P_{\text{end}} = 3$.

We apply the XDG-IST method using a very coarse 5×16 mesh and a refined 10×32 mesh, with a cubic spline-based level set ($P_s = 3$) and employing the full enriched residual objective function f_{ER}. Selected iterations ($P = 0$) are illustrated in Figures 5.11 and 5.13. Notably,

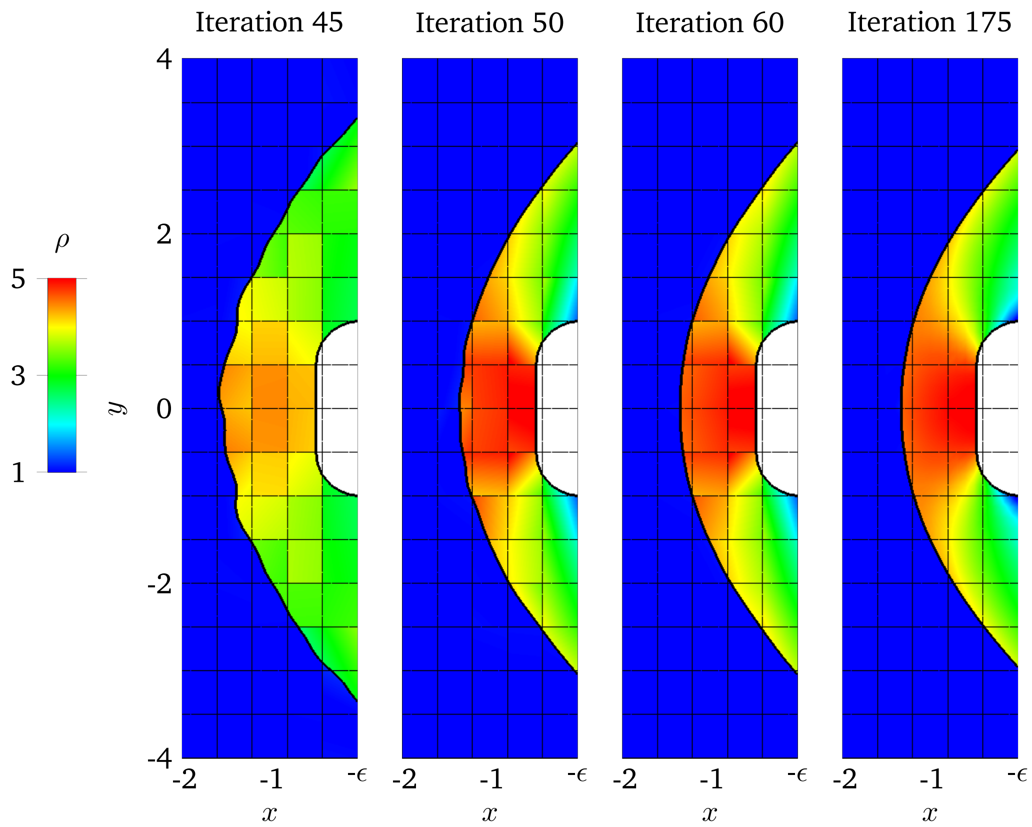


Figure 5.12: Selected XDG shock tracking iterations z_k ($k = 45, 50, 60, 175$, polynomial degree $P > 0$) for the density ρ of the Mach 4 bow shock for a 5×16 mesh. The blunt body (*white filling*) is represented by an immersed boundary $\mathfrak{I}_b$ (*right thick black line*) using a quadratic level set. Continuing the optimization history from Figure 5.11, the shock interface $\mathfrak{I}_s$ (*left thick black line*), represented by a cubic spline level set ($P_s = 3$), and the the XDG approximation are converged by the SQP solver.

during iterations where the solver increases the polynomial degree due to stagnation (e.g., $k = 40$ for 5×16), a visible discontinuity and a strongly overestimated shock front are observed. As the polynomial degree of the flow solution is increased (to $P > 0$), iterations shown in Figures 5.12 and 5.14, the shock front moves closer to the blunt body after several iterations and eventually reaches a visually well-resolved and converged solution. The optimization history for the 5×16 mesh is depicted in Figure 5.15, where a steady reduction in the enthalpy error is noted. In the converged state, a remarkably low enthalpy error of $H_{\text{err}} \approx 8.5 \times 10^{-4}$ is achieved. Figure 5.16 presents the optimization history for the refined 10×32 mesh, which, while similar to the coarse mesh case, displays lower residuals and enthalpy errors by approximately one order of magnitude during solver stagnation phases.

Similar to the curved Burgers case discussed earlier, the solver is unable to converge the residual norm to zero. However, we observe that refining the mesh lowers the value ultimately reachable for the residual norm. Remarkably, as will be presented in Section 5.2.3, switching the numerical flux at the interface from Godunov's flux to Roe's flux enables the optimizer to reach significantly lower residual levels, while maintaining comparable enthalpy errors for both approaches. Also in Section 5.2.5, we will see that the enthalpy errors obtained by our XDG method are superior to those obtained by a shock capturing simulation on a significantly finer mesh (factor 80).

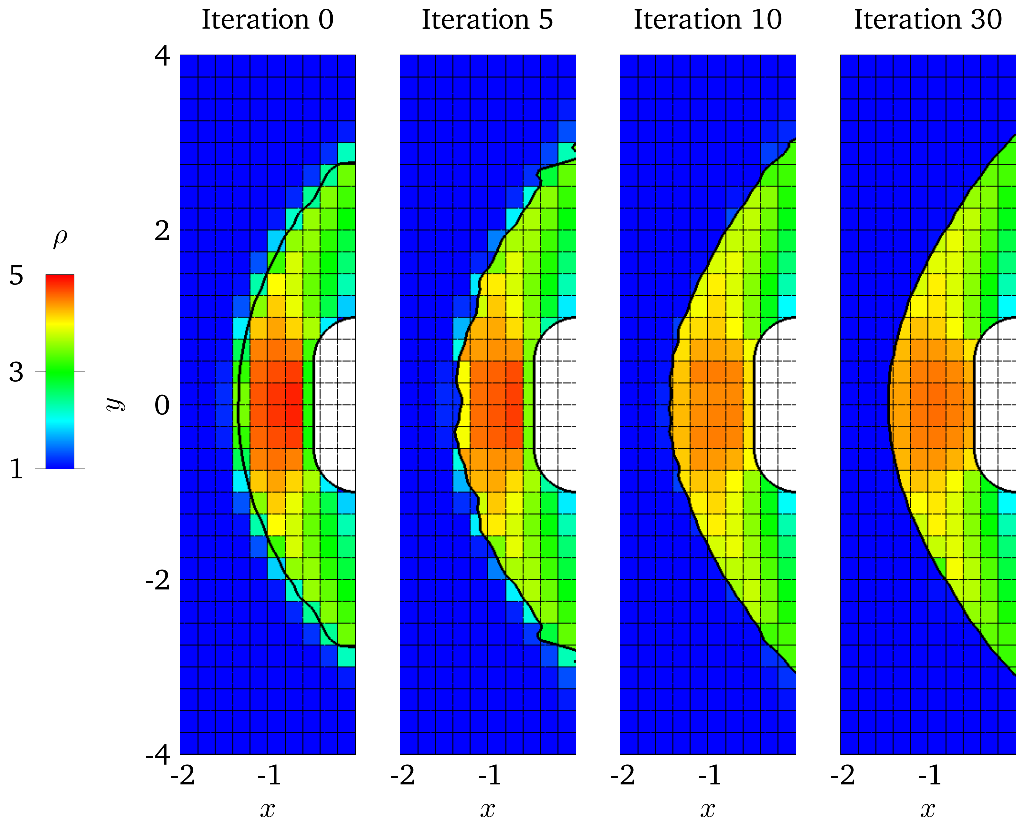


Figure 5.13: Selected XDG shock tracking iterations $\mathbf{z}_k$ ($k = 0, 5, 10, 30$, polynomial degree $P = 0$) for the density ρ of the Mach 4 bow shock for a 10×32 mesh. The blunt body (*white filling*) is represented by an immersed boundary $\mathfrak{I}_b$ (*right thick black line*) using a quadratic level set. Starting from an initial guess obtained by a reconstruction procedure, the shock interface $\mathfrak{I}_{\mathfrak{s}}$ (*left thick black line*), represented by a cubic spline level set ($P_s = 3$), and the XDG approximation are converged by the SQP solver (see Figure 5.14 for $P > 0$ iterations).

5.1.4 One-dimensional space-time Euler equations

Considering the space-time formulation of the 1D Euler equations (3.82), we aim to apply the XDG-IST method to 1D shock-acoustic-wave interaction problems. To this end, we consider three cases, each featuring an initially stationary normal shock wave, which is then impinged by an acoustic wave. Specifically, the three cases are: 1) a fast acoustic wave hitting the shock from the supersonic side of the domain, 2) a slow acoustic wave hitting the shock from the subsonic side of the domain, and 3) a slow acoustic wave hitting the shock from the supersonic side of the domain, with the corresponding theory already outlined in Section 2.3. By comparing the results of 1) and 2) with the analytical expressions for the amplification factors (2.27) and (2.29), we aim to demonstrate the method's capabilities to capture these physical phenomena correctly.

For all three cases, the initial setup is described by a case-dependent time interval $\mathcal{T}$, a space-time domain $\Omega = [0, 3] \times \mathcal{T}$, a normal shock located at $x_s \in [0, 3]$, and a constant base flow

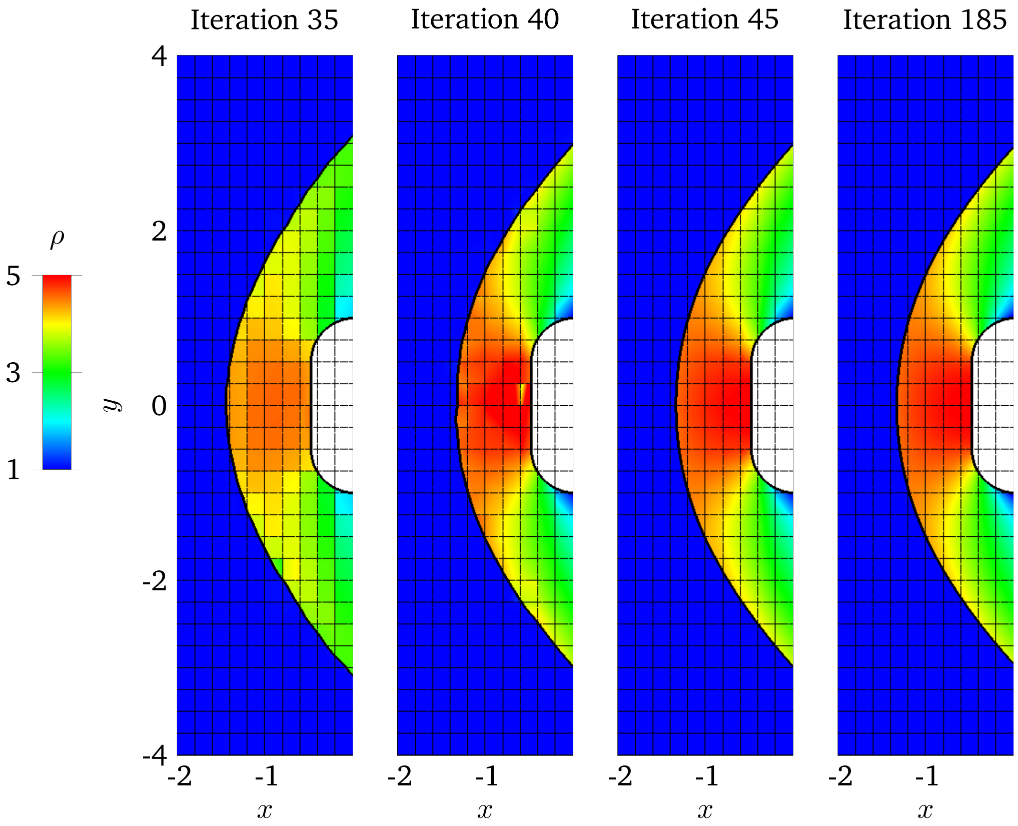


Figure 5.14: Selected XDG shock tracking iterations z_k ($k = 35, 40, 45, 185$, polynomial degree $P > 0$) for the density ρ of the Mach 4 bow shock for a 10×32 mesh. The blunt body (*white filling*) is represented by an immersed boundary $\mathfrak{I}_b$ (*right thick black line*) using a quadratic level set. Continuing the optimization history from Figure 5.13, the shock interface $\mathfrak{I}_s$ (*left thick black line*), represented by a cubic spline level set ($P_s = 3$), and the the XDG approximation are converged by the SQP solver.

(indicated by $(\cdot)^\circ$)

$$(\rho^\circ(x), u^\circ(x), p^\circ(x))^\top = \begin{cases} (\rho_L^\circ, u_L^\circ, p_L^\circ)^\top & \text{if } x < x_s, \\ (\rho_R^\circ, u_R^\circ, p_R^\circ)^\top & \text{else,} \end{cases} \tag{5.21}$$

with pre-shock conditions $(\cdot)_L$ given by

$$(\rho_L^\circ, u_L^\circ, p_L^\circ)^\top = \left(1, \sqrt{\hat{\gamma}}\,\mathrm{Ma}_L^\circ, 1\right)^\top, \tag{5.22}$$

and post-shock conditions $(\cdot)_R$ given by

$$\rho_R^\circ = \frac{(\hat{\gamma}+1)\,\mathrm{Ma}_L^{\circ 2}}{2 + (\hat{\gamma}-1)\,\mathrm{Ma}_L^{\circ 2}} \rho_L^\circ, \tag{5.23a}$$

$$u_R^\circ = \frac{2 + (\hat{\gamma}-1)\,\mathrm{Ma}_L^{\circ 2}}{(\hat{\gamma}+1)\,\mathrm{Ma}_L^{\circ 2}} u_L^\circ, \tag{5.23b}$$

$$p_R^\circ = \left[1 + \frac{2\hat{\gamma}}{\hat{\gamma}+1}\left(\mathrm{Ma}_L^{\circ 2} - 1\right)\right] p_L^\circ. \tag{5.23c}$$

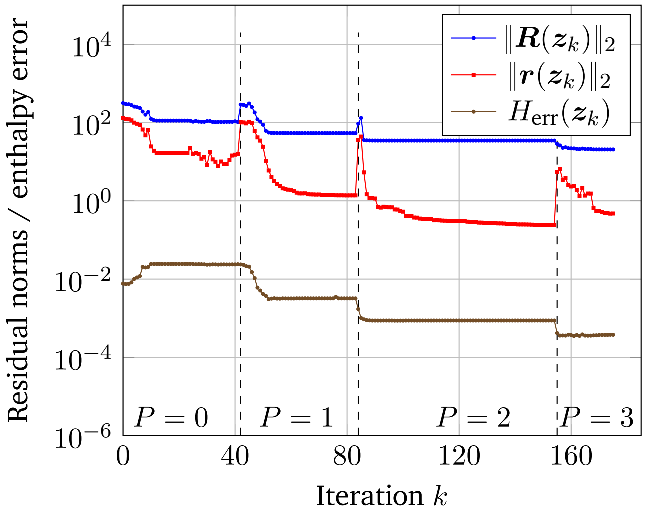


Figure 5.15: Optimization history of the XDG-IST method for the Mach 4 bow shock for the 5×16 mesh, employing a P-continuation strategy ($P = 0, 1, 2, 3$). The residual norms $\|\boldsymbol{R}\|_2$, $\|\boldsymbol{r}\|_2$ and the normalized enthalpy error H_{err} across all SQP iterations are shown. After stagnation of the residual of an intermediate polynomial degree, the latter is increased, such that a jump in both residuals $\boldsymbol{R}, \boldsymbol{r}$ can be observed (*dashed lines*). An overall decline of both residuals and the enthalpy error can be measured as the SQP solver tracks the non-polynomial discontinuity and solution.

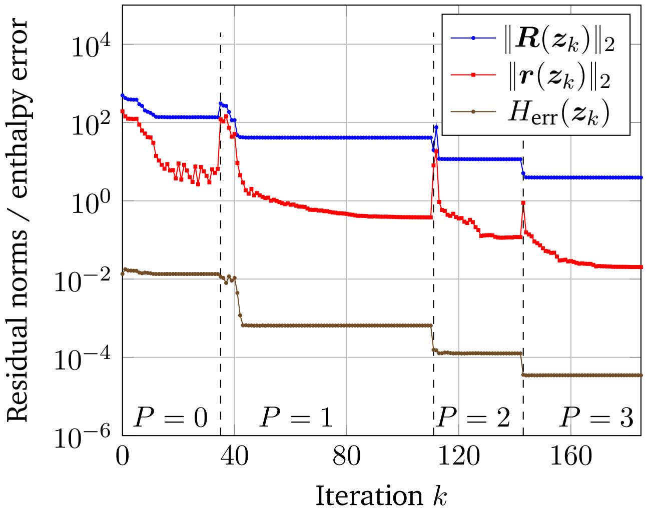


Figure 5.16: Optimization history of the XDG-IST method for the Mach 4 bow shock for the 10×32 mesh, employing a P-continuation strategy ($P = 0, 1, 2, 3$). The residual norms $\|\boldsymbol{R}\|_2$, $\|\boldsymbol{r}\|_2$ and the normalized enthalpy error H_{err} across all SQP iterations are shown. After stagnation of the residual of an intermediate polynomial degree, the latter is increased, such that a jump in both residuals $\boldsymbol{R}, \boldsymbol{r}$ can be observed (*dashed lines*). An overall decline of both residuals and the enthalpy error can be measured as the SQP solver tracks the non-polynomial discontinuity and solution.

The latter are computed by employing the normal shock relations outlined in Section 2.2.1. On top of this base flow initial perturbations (denoted by $(\cdot)'$) are added for each primitive variable $\psi \in \{p, \rho, u\}$

$$\psi'(x) = \begin{cases} \delta\psi' \sin(\frac{2\pi(x - x_\mathrm{W})}{l_\mathrm{W}}) & \text{if } x \in [x_\mathrm{W}, x_\mathrm{W} + l_\mathrm{W}], \\ 0 & \text{else}, \end{cases} \tag{5.24}$$

where $x_\mathrm{W} \in \mathbb{R}$ denotes the initial wave position and $l_\mathrm{W} \in \mathbb{R}$ a wave length (set to 0.8 for all cases). The pressure amplitude $\delta p'$ is chosen to be $\delta p' = 10^{-5}$, from which the amplitude $\delta\rho'$ for the density perturbations is computed by

$$\delta\rho' = \frac{\delta p'}{a^{\circ 2}}, \tag{5.25}$$

using the relation (2.23). Analogously, the velocity amplitude

$$\delta u' = \pm\frac{\delta p'}{\rho^\circ a^\circ} \tag{5.26}$$

is given by (2.25). For the latter, the choice of the sign determines whether a fast ($\mathrm{sgn}(\delta u') = -1$ → wave speed $= u^\circ + a^\circ$) or slow ($\mathrm{sgn}(\delta u') = +1$ → wave speed $= u^\circ - a^\circ$) acoustic wave is considered. A brief overview of acoustic waves can be found in Section 2.3.2 and a schematic overview for all three cases considered is given in Figure 2.5.

We use a Dirichlet boundary condition on the lower part of the domain boundary $\{(x, t) \in \Omega \mid t = 0\}$ imposing the initial condition

$$p(x, 0) = p^\circ(x) + p'(x), \; \rho(x, 0) = \rho^\circ(x) + \rho'(x), \; u(x, 0) = u^\circ(x) + u'(x). \tag{5.27}$$

In each case, a 61×61 mesh is employed to discretize the domain Ω and a single cubic spline-based level set ($P_s = 3$) is employed to represent the slightly oscillating shock front. Employing a P-continuation strategy, a ($P = 3$) XDG solution is computed using the implicit shock tracking method, together with the space-time Roe flux without entropy fix (3.56) and the objective function f_ER based on the full enriched residual and defined in (4.3). The spline-based level set is initialized as a stationary shock wave, i.e.

$$\varphi_0(x, t) = x - x_s. \tag{5.28}$$

In order to gauge the accuracy of the amplification factor obtained for the XDG approximation, we sample the pressure of the resulting XDG solution and compute the maximal/minimal value along specified rays in t-direction (e.g. $\{(x_i, t) \mid t \in \mathcal{T}\}$ with $x_i \in [0, 3]$ for a ray in t direction). The resulting amplification factors are then compared against their analytical counterparts. While this may not be the most optimal comparison (see Remark 15), it still gives a rough estimate of the quality of the numerical solution.

Remark 15. *Note, that by the nature of the DG method, which computes solutions based on cell and surface integrals, the precision of point-wise measures varies depending where the points evaluated are positioned relative to a cell and domain boundaries.*

Remark 16. *In the illustrations (Figures 5.17- 5.20), we do not show the shock interface $\mathfrak{I}_s$ since the shock movement is so marginal that it would not be visible in the depicted plots. However, note that a high-order DG scheme without any shock stabilization, e.g. employing artificial viscosity (AV) or implicit shock tracking (IST), fails to converge for these problems, even when mesh edges are initially aligned with the stationary shock at x_s.*

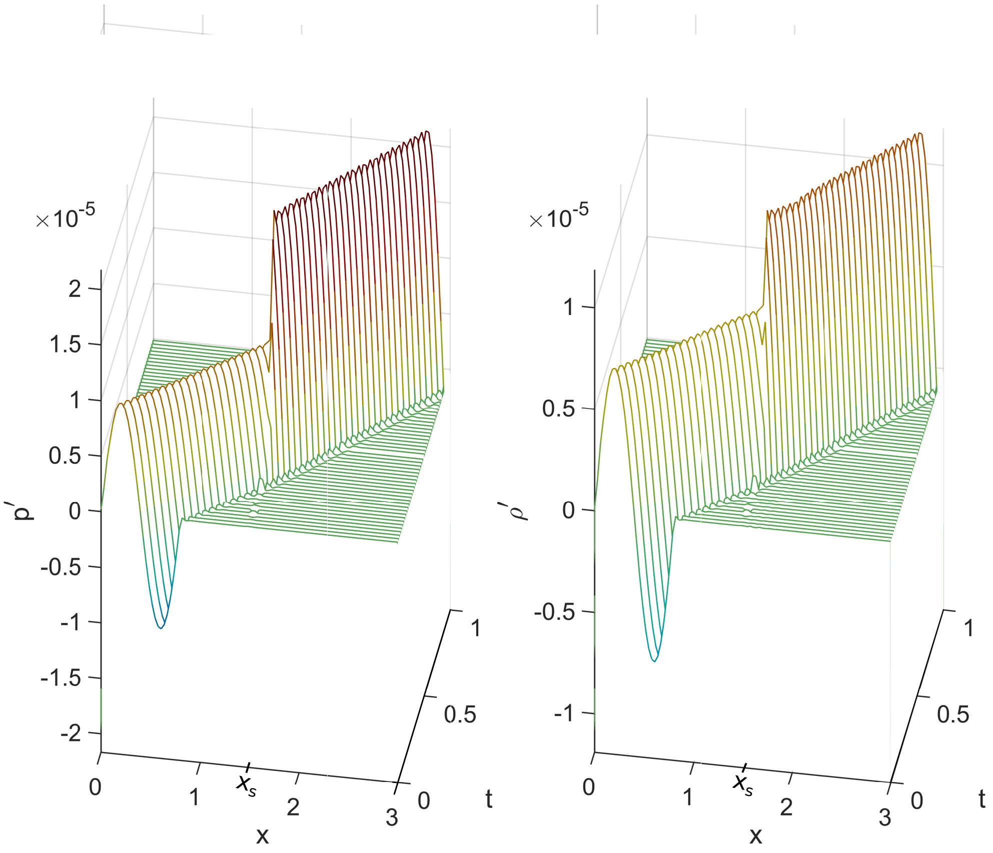


Figure 5.17: Waterfall plot of XDG shock tracking solution for an 1D space-time shock-acoustic-wave problem with a fast acoustic wave hitting the shock from the supersonic side and a shock initially located at $x_s = 1.5$. *Left*: Pressure perturbations p'. *Right*: Density perturbations ρ'.

Fast acoustic wave hitting the shock from the supersonic side For the first case we choose $\operatorname{sgn}(\delta u') = -1$ and position the initial perturbations at $x_\mathrm{W} = 0.0$ and the shock at $x_s = 1.5$. The wave length is chosen to be $l_\mathrm{W} = 0.8$. The amplification coefficient $\mathrm{amp}^+_\mathrm{ex}$ of the pressure perturbations across the shock can be computed using (2.27) and (2.13), i.e.,

$$\mathrm{amp}^+_\mathrm{ex} := \frac{\delta p'_R}{\delta p'_L} = \frac{(1+\mathrm{Ma}^\circ_L)^2}{1+2\,\mathrm{Ma}^\circ_R + 1/\mathrm{Ma}^{\circ 2}_L}\left[1 - \frac{\hat{\gamma}-1}{\hat{\gamma}+1}\left(1-\frac{1}{\mathrm{Ma}^\circ_L}\right)^2\right] \approx 2.154926. \qquad (5.29)$$

In Figure 5.17, we plot the converged XDG-IST solution without the shock interface (see Remark 16), showing pressure and density perturbations. There, it is observable that the pressure and density waves are both amplified by the interaction. However, the entropy wave, present in the density perturbations, is hidden by the plotting angle (see Remark 17). To asses the accuracy of the XDG approximation, we compare the maximal values $\max_{t\in\mathcal{T}}(p'(x_i,t))$ for $x_i \in [1.8, 2.7]$ to the analytical value in Figure 5.18 (*Left*). We observe a good agreement with the analytical value with a minimum precision of 2.5^{-2} and a maximum deviation of 1.18%.

The oscillations observable in Figure 5.18 (*Left*) can be attributed to the positioning of the maximum points found for each $x_i \in [1.8, 2.7]$. Depending on the relative positioning of a point

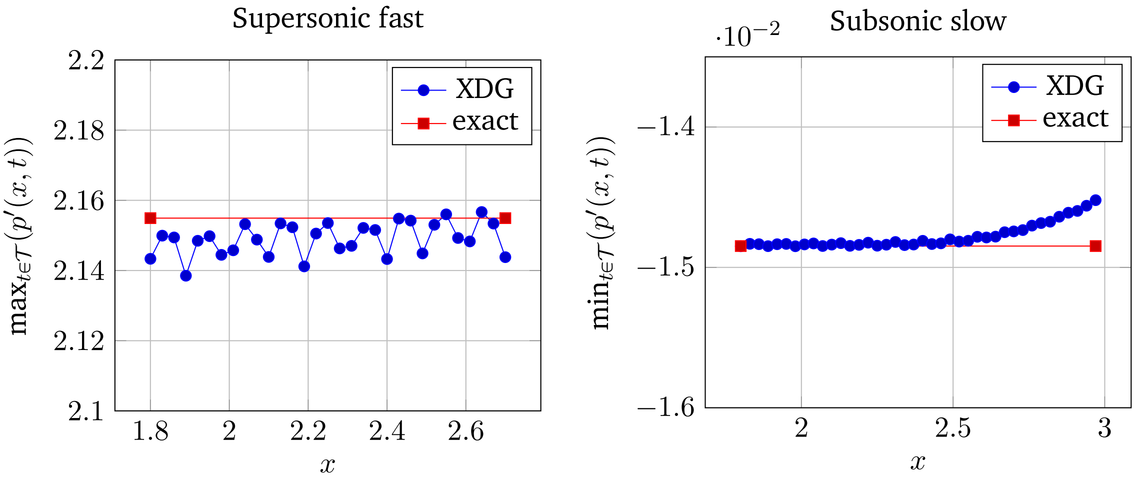


Figure 5.18: Comparison of pressure amplification/reduction coefficient resulting from the acoustic-shock-wave interaction. The analytical and XDG shock tracking solution are compared for two cases. *Left*: In the case of the fast acoustic wave hitting the shock from the supersonic side, the solution is evaluated at 100 × 100 sample points and the maximum $\max_{t\in\mathcal{T}}(p'(x_i,t))$ is taken for $x_i \in [1.8, 2.7]$. *Right*: In the case of the slow acoustic wave hitting the shock from the subsonic side, the solution is again evaluated at 100 × 100 sample points and the minimum $\min_{t\in\mathcal{T}}(p'(x_i,t))$ is taken for spatial points $x_i \in [1.8, 3.0]$.

inside a computational cell, DG methods may differ in accuracy at these points, being more accurate towards the center of a cell and less accurate towards the boundaries. The maximum points, for which the values $p'(x:i,t)$ are plotted in the figure, differ in terms of the relative positioning which therefor explains the oscillations.

Remark 17. *The entropy wave associated with the fast acoustic wave is approximately three orders of magnitude smaller than the amplified pressure perturbations, making it challenging to visualize both in the same plot. Given that the primary focus of this work is on acoustic waves, we have chosen not to include secondary plots for the entropy wave in this particular case. This decision is also supported by the fact that the entropy wave is observable in the other two cases presented in this study.*

Slow acoustic wave hitting the shock from the subsonic side For the second case we choose $\operatorname{sgn}(\delta u') = 1, l_{\mathrm{W}} = 0.8$ and position the initial perturbations at $x_{\mathrm{W}} = 0.9$ and the shock at $x_s = 0.5$. As the wave is located on the subsonic side of the shock we have $u_R^\circ - a_R^\circ < 0$ and, hence, the slow wave moves upstream and will be eventually reflected from the shock. The reduction coefficient $\mathrm{amp}_{\mathrm{ex}}^-$ for the reflected pressure perturbations can be computed using (2.29) and (2.13), which gives

$$\mathrm{amp}_{\mathrm{ex}}^- := \frac{\delta p_R'^-}{\delta p_R'} = -\frac{1 - 2\,\mathrm{Ma}_R^\circ + 1/\mathrm{Ma}_L^{\circ 2}}{1 + 2\mathrm{Ma}_R^\circ + 1/\mathrm{Ma}_L^{\circ 2}} \approx -0.014848. \tag{5.30}$$

This coefficient indicates, that the reflected wave is two magnitudes smaller than the original wave and is accompanied by a sign change. In Figure 5.19, the converged XDG-IST solution is depicted without the shock interface (see Remark 16), showing the pressure and density perturbations. There, the reflected wave in the pressure perturbations and a stronger entropy

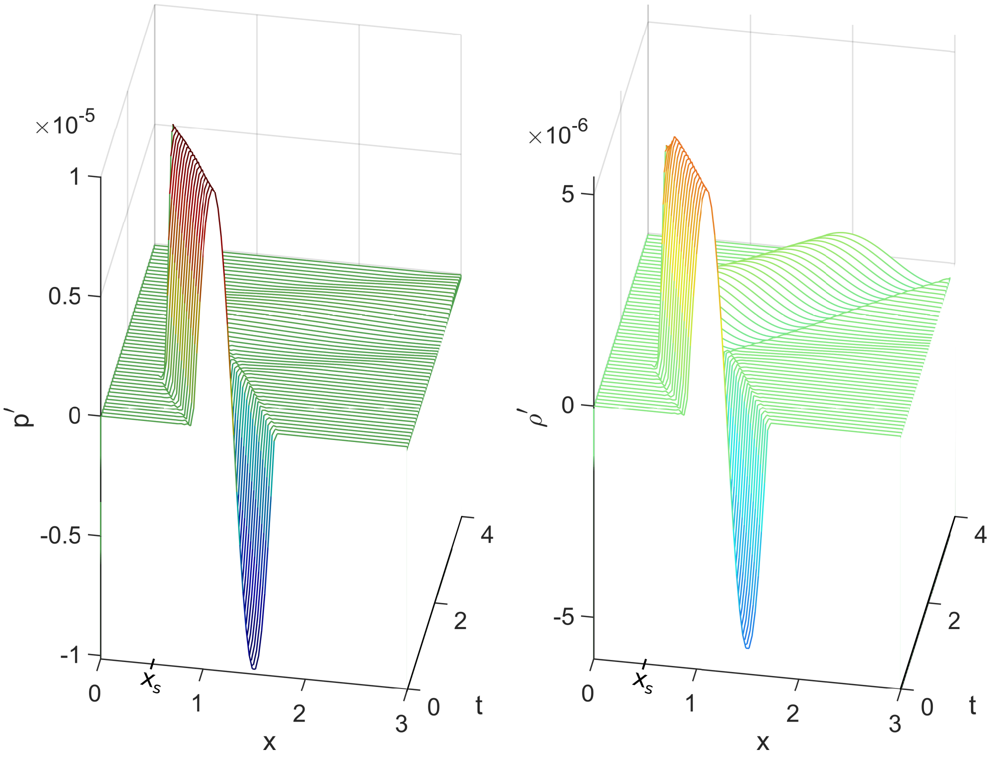


Figure 5.19: Waterfall plot of XDG shock tracking solution for an 1D space-time shock-acoustic-wave problem with a slow acoustic wave hitting the shock from the subsonic side and a shock initially located at $x_s = 0.5$. *Left*: Pressure perturbations p'. *Right*: Density perturbations ρ'.

wave in the density perturbations are visible. Furthermore, we compare the minimums (we choose the minimum due to the negative amplification/reduction factor) $\min_{t \in \mathcal{T}}(p'(x_i, t))$ for $x_i \in [1.8, 3.0]$ to the analytical value in Figure 5.18, which showcases a precision of roughly 5×10^{-4} and a maximum deviation of 2.2%. This deviation is almost two times higher than for the supersonic fast wave and can be attributed to a slight but steady loss of precision which is observed towards the upper right corner of the domain for the subsonic slow wave. A possible explanation, needing further investigation, could be an inconsistency in the boundary conditions at the upper right corner.

Slow acoustic wave hitting the shock from the supersonic side For the third case we choose $\text{sgn}(\delta u') = -1, l_\text{W} = 0.8$ and position the initial perturbations at $x_\text{W} = 0.5$ and the shock at $x_s = 1.5$. As the wave is located in the supersonic region we have $u_L^\circ - a_L^\circ > 0$ and, hence, the slow wave moves downstream. When hitting the shock it is transmitted to the subsonic side, resulting in a fast acoustic wave (speed $u_R^\circ + a_R^\circ$) and an entropy wave (speed u_R°) both moving downstream.

In Figure 5.20, the converged XDG-IST solution is depicted without the shock interface (see Remark 16), showing pressure and density perturbations. A reduction of the pressure amplitude across the shock is visible (*Left*) and we observe a relative strong entropy wave in the density

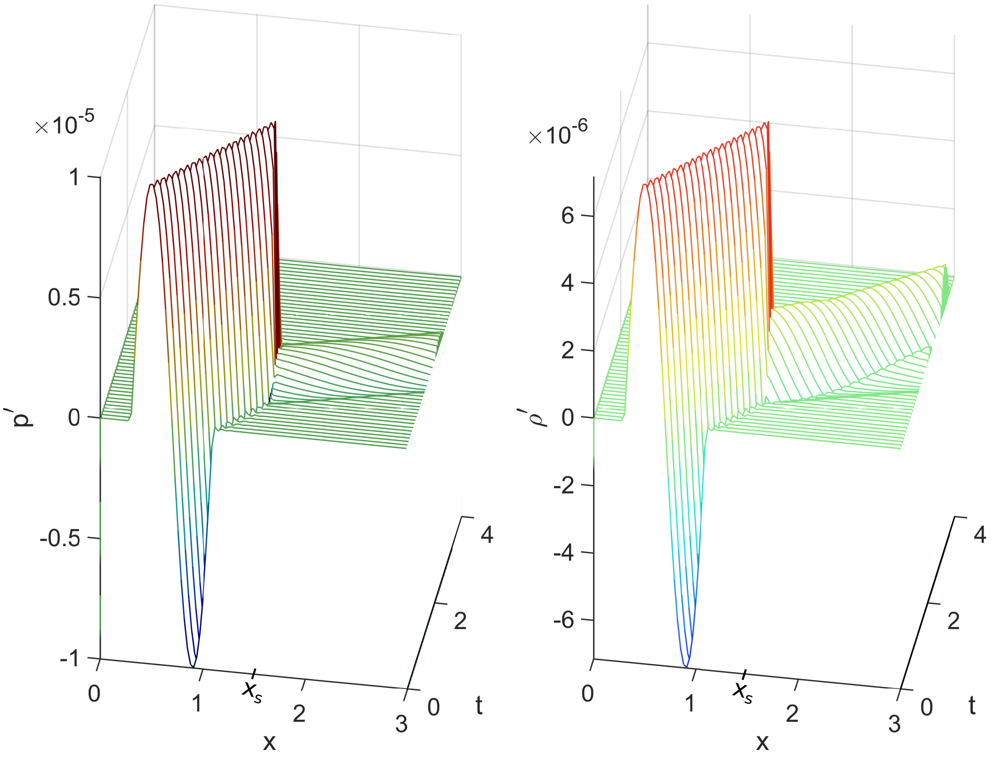


Figure 5.20: Waterfall plot of XDG shock tracking solution for an 1D space-time shock-acoustic-wave problem with a slow acoustic wave hitting the shock from the supersonic side and a shock initially located at $x_s = 1.5$. *Left*: Pressure perturbations p'. *Right*: Density perturbations ρ'.

perturbations (*Right*), both contrasting the results observed for the fast acoustic wave. As we haven not found any reference in the literature for this particular case, no comparison to an analytical solution is made. However, as we observed good agreement for the other two cases, it is most likely that the observed phenomena are close to the ones expected from theory. Further investigation would necessitate a derivation for the amplification/reduction factor and are left for future endeavors.

5.2 Numerical studies

This section presents results from a series of numerical studies testing various variants and aspects of the implicit XDG shock tracking method.

It is structured as follows: Section 5.2.1 offers a comparison of different level set discretizations, exploring their impact on the overall performance and accuracy of the method. The effectiveness of various objective functions is assessed in Section 5.2.2, with a focus on how they influence the optimization process and solution quality. In Section 5.2.3, a comparative analysis of numerical fluxes in the context of the 2D Euler equations is conducted to determine their effect on the method's ability to capture shock dynamics accurately. A convergence study in Section

5.2.4 quantifies the method's numerical convergence properties for the Mach 4 bow shock problem and a 1D shock-acoustic-wave interaction problem. Additionally, the XDG-IST method is compared to a shock capturing method in Section 5.2.5. Finally, Section 5.2.6 presents a performance study that evaluates the computational efficiency of the XDG-IST for a single Mach 4 bow shock test case.

5.2.1 Comparison of level set discretizations

We perform an in-depth assessment of the XDG-IST method by comparing three level set discretizations: a DG-, a continuous Galerkin (CG)-, and a spline-based discretization. These discretizations, which were previously detailed in Section 4.6, represent the solution discontinuity and have different features regarding flexibility, computational efficiency and continuity of the interface. Our study centers on the straight-sided shock test case for the space-time advection equation introduced in Section 5.1.1, where we employ the aforementioned discretizations within the XDG-IST framework using a polynomial degree of $P_s = 3$ for all three approaches.

In Figure 5.21, we illustrate the optimization history of the XDG-IST method, highlighting the evolution of both the classical and enriched residuals over successive iterations. It becomes evident that the spline-based level set delivers a superior convergence profile, achieving significantly lower residual magnitudes. In contrast, the DG and CG level sets exhibit signs of stagnation.

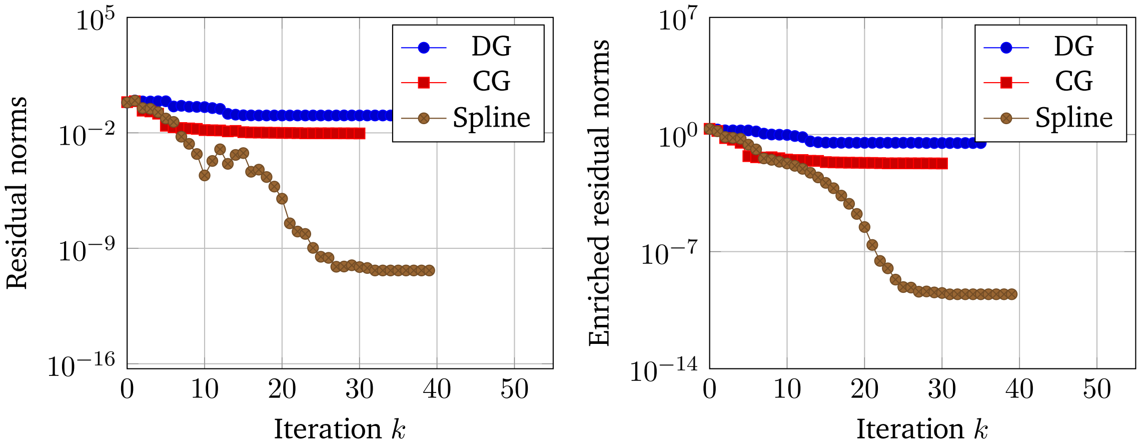


Figure 5.21: Comparative optimization history of classical residuals (*left*) and enriched residuals (*right*) across different level set discretizations applied in the simulation of the space-time advection equation.

The graphical progression of selected iterations, depicted in Figure 5.22, reveals that while the spline and the CG level set visually approximate the solution, it is only by examining the optimization history in Figure 5.21 that we observe the superior accuracy of the spline-based level set, with residuals minimized to the order of 10^{-10}. Despite the CG level set's close proximity to the exact solution, it encounters difficulties in fine-tuning the XDG approximation, as indicated by its higher residual values. The DG level set shows a tendency towards a chaotic interface evolution as early as the first iteration, likely due to the discontinuities permitted at the element boundaries including the interface.

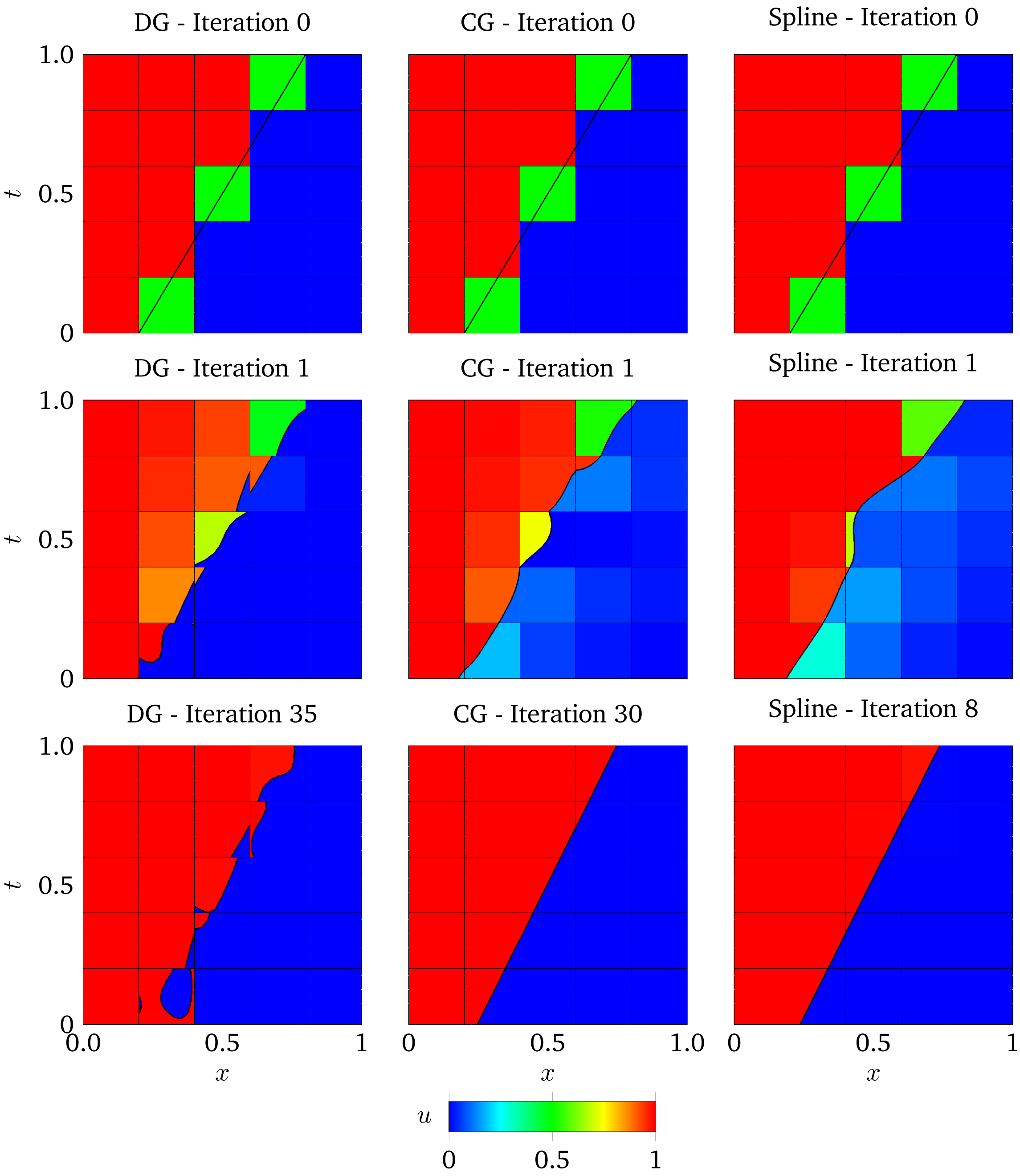


Figure 5.22: Plots of velocity u for selected XDG shock tracking iterations for the space-time advection test case ($P = 0$) with a straight shock, employing three distinct level set discretizations ($P_s = 3$): the first coloumn (*left*) shows iterations with a DG level set, the second (*middle*) features a CG level set and the last (*right*) a spline level set.

To visualize the computational costs associated by each level set discretization, we measure the number of degrees of freedom (DOFs) and the time to compute the Jacobian $\partial \boldsymbol{R}/\partial\varphi$ of the enriched residual with respect to the level set for a series of meshes with $2^i \times 2^i$ cells ($i \in \{1,2,3,4\}$). The results are shown in Figure 5.23. There, the relative advantage of the spline-based level set is vivid in terms of low total DOFs and resulting computational times associated with the computation of $\partial \boldsymbol{R}/\partial\varphi$. The scaling is also less, which is partially since the DOFs are only increased when refining the mesh in t-direction. Conversely, for the other two approaches, the DOFs increase each time a cell is added.

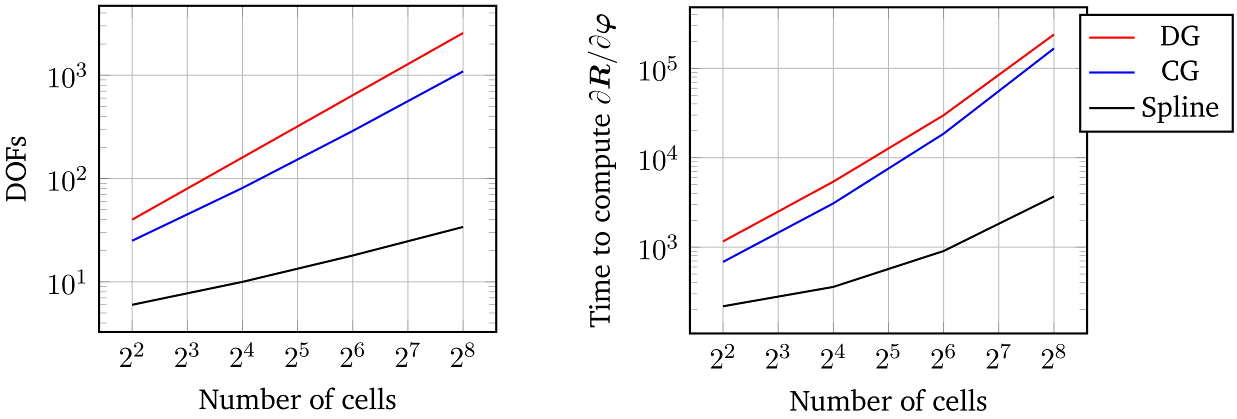


Figure 5.23: *Left*: Plot of DOFs for different level set discretizations as a function of the number of cells. *Right*: Plot of time to compute $\partial \boldsymbol{R}/\partial\varphi$ for different level set discretizations as a function of the number of cells.

In conclusion, this study demonstrates that, within the confines of the current implementation, the XDG-IST method achieves its most reliable and accurate performance when paired with the spline-based level set discretization.

5.2.2 Comparison of objective functions

Post evaluation of the level set discretizations, our analysis shifts towards assessing the impact of different objective functions on the performance of the XDG-IST method. Specifically, we focus on objective functions f_{ER}, f_{NB}, and f_{RH}, which are elaborated in Section 4.1.1. To this end, we apply the XDG-IST method to the Mach 2 wedge flow ($P = 0$) (Figure 5.9) and the Mach 4 bow shock (P=2) (Figure 5.14) test case as detailed above. In particular this means, that spline-based level sets are employed.

In the case of the wedge flow we show the optimization history in Figure 5.24, we observe a comparable convergence trend for the enthalpy error across all three objective functions. However, it is noted that the enriched residual-based objective function f_{ER} achieves a marginally lower residual magnitude, converging to approximately 10^{-14}, while the other functions plateau near 10^{-8}.

Examining the $P = 2$ bow shock case, Figure 5.25 presents a contrasting narrative. Here, only the enriched residual-based objective function f_{ER} demonstrates the abilities to converge the solution and significantly reduce the enthalpy error. The Rankine-Hugoniot-based objective

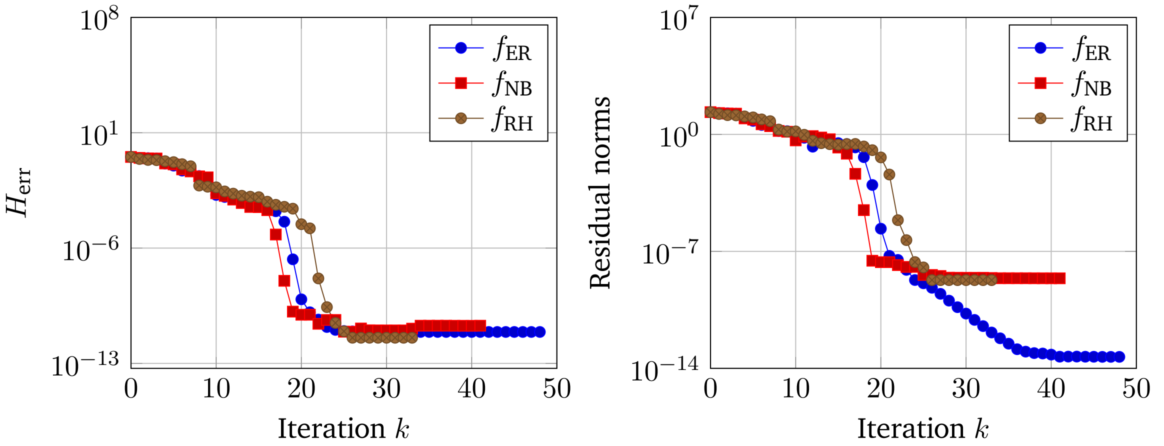


Figure 5.24: Comparative optimization history of enthalpy errors (*left*) and residuals (*right*) across different objective functions (f_{ER}, f_{NB}, f_{RH}) employed in the simulation of supersonic Mach 2 airflow over an inclined plane.

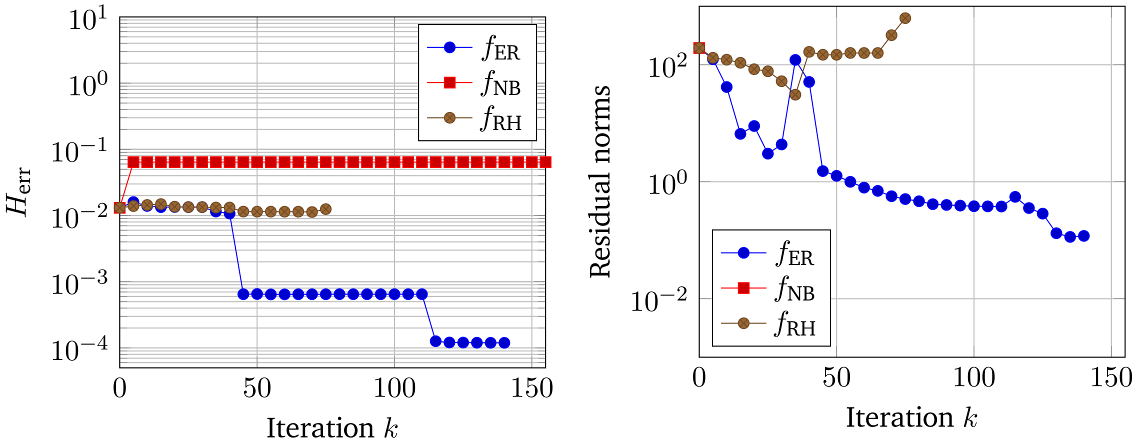


Figure 5.25: Comparative optimization history of enthalpy errors (*left*) and residuals (*right*) across different objective functions (f_{ER}, f_{NB}, f_{RH}) employed in the simulation of a supersonic Mach 4 bow shock. The residual for the constrained objective function f_{RH}, quickly becomes *NaN* and leads to a stagnation of the solver resulting in a constant enthalpy error, hence only the starting value is plotted.

function f_{RH} does not manage to lower the initial enthalpy error or the residual norms, particularly following the polynomial degree increase associated with P-continuation. Alarmingly, the objective function f_{NB} based on the enriched residual confined to the near band fails entirely, evidenced by a *NaN* appearing in the residual computation after just one iteration.

To conclude, the findings suggest that f_{ER} stands out as the most robust objective function, particularly for complex flow problems, indicating its superiority in terms of achieving accurate and stable solutions within the XDG-IST framework.

5.2.3 Comparison of numerical fluxes

Our focus now shifts to the analysis of different numerical fluxes utilized at the shock interface within the XDG discretization of the steady 2D Euler equations (see Section 3.4.3). In the discussion of test cases (Section 5.1.3), we employed the Godunov flux at the interface (3.47), renowned for its accuracy at the expense of increased computational demands. While in the 2D scenarios considered, this computational overhead is mitigated by the relatively sparse distribution of interface edges, the scalability of this approach to three-dimensional (3D) contexts is questionable due to the potentially prohibitive costs.

To evaluate the efficacy of alternative fluxes that promise computational efficiency, despite a trade-off with accuracy, we apply the XDG-IST method to the Mach 2 wedge flow and the 10×32 mesh resolution for the Mach 4 bow shock (Section 5.1.3). This investigation encompasses four numerical flux variants: the Godunov flux (3.47), the smoothed Roe flux without entropy fix (3.56), the HLLC flux (3.48), and the central flux (3.42).

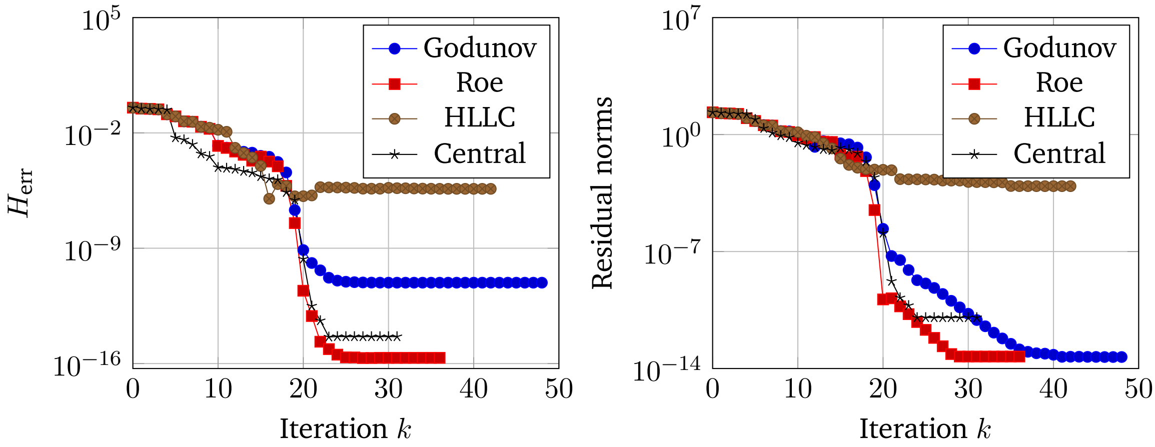


Figure 5.26: Comparative optimization history of enthalpy errors (*left*) and residuals (*right*) across different numerical flux functions (Godunov, Roe, HLLC, and Central) applied to the interface edges in the simulation of supersonic Mach 2 airflow over an inclined plane.

In Figure 5.26, the optimization histories for residuals and enthalpy errors are presented for the Mach 2 wedge flow and for each flux variant. The Roe flux without entropy fix demonstrates impressive convergence to machine precision for the enthalpy error, closely followed by the central flux which achieves an error magnitude of 10^{-14}. The Godunov variant presents an enthalpy error on the order of 10^{-11}, whereas the HLLC flux culminates at a less favorable error estimation of roughly 10^{-5}.

Conversely, the bow shock scenario, with the objective of converging the solution to a polynomial degree of $P = 2$, reveals distinct performance characteristics among the flux variants. As depicted in Figure 5.27, the central flux variant struggles, ultimately causing the termination of the solver after roughly 70 iterations due to a failure to converge. The HLLC flux variant, although successful during the initial polynomial degree increase, capitulates after subsequent increments, ceasing after 90 iterations. Notably, the Godunov flux secures an enthalpy error of 10^{-4} and reduces the residual to approximately 10^{-2}. The Roe flux, requiring more iterations, not only matches the enthalpy error of the Godunov flux but also succeeds in driving down

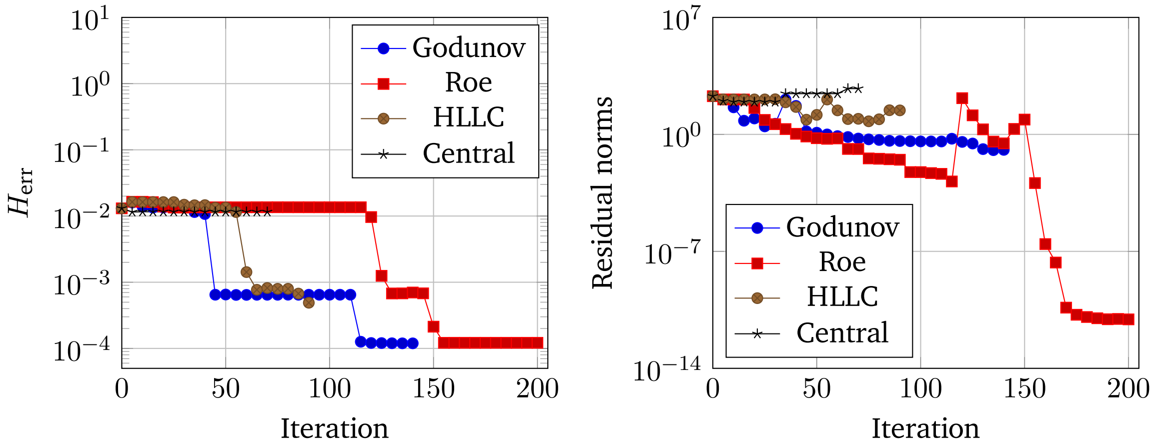


Figure 5.27: Comparative optimization history of enthalpy errors (*left*) and residuals (*right*) across different numerical flux functions (Godunov, Roe, HLLC, and Central) applied to the interface edges in the simulation of a supersonic Mach 4 bow shock.

the residual to a significantly lower level of 10^{-11}. The shortcomings of the central and HLLC fluxes, particularly in complex scenarios, align with theoretical expectations, where the central flux is notoriously unstable and the HLLC flux exhibits inconsistency with the Rankine-Hugoniot conditions. However, the Roe flux without entropy fix emerges as a viable contender to the Godunov flux. Our results suggest potential applicability across a broader spectrum of problems, though further studies are required to confirm its general performance.

Additionally, The persisting divergence in residual norms between the variants employing Godunov and Roe fluxes asks for further investigation, a query which remains unresolved within the limitations of this study. A possible explanation for this phenomenon relates to the Roe flux's smoothing, implemented to fulfill a specific criterion desired for optimization methods (see Section 3.3.1). This smoothing enhances the solver's capability to identify the residual-minimizing solution. Another plausible explanation pertains to the limitations inherent in the weak form constituted by the Godunov flux when applied within the existing XDG space. The under-resolved nature of the solution and interface, when employed alongside the Godunov flux, may preclude the attainment of a nonlinear residual *close-to-zero*, unless realized on an exceedingly refined mesh or at elevated polynomial degrees.

5.2.4 Convergence study

We test the convergence properties of the XDG-IST method, by conducting two spatial convergence studies: one for the Mach 4 bow shock test case and a second for the shock-acoustic-wave interaction problem with a fast acoustic wave hitting the shock from the supersonic side (see Remark 18).

Remark 18. *The most recent bounded support spectral solver (BoSSS) release Kummer et al., 2024 does not contain the notebooks corresponding to convergence studies for the shock-interaction-problem, as it was released before the completion of the studies. They were finished in close proximity to the submission date of this dissertation and, hence, we could not provide links to the notebooks.*

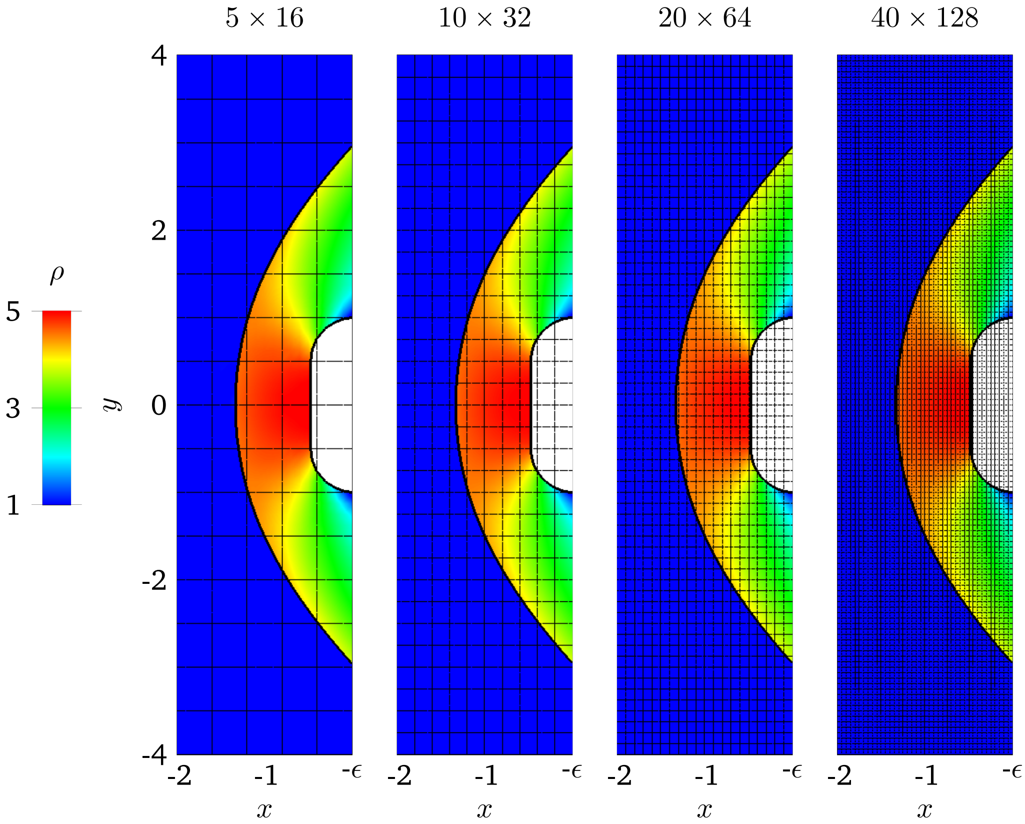


Figure 5.28: Density of converged XDG shock tracking iterations ($P = 3$) of the Mach 4 bow shock for four different meshes. The blunt body (*white filling*) is represented by an immersed boundary $\mathfrak{I}_b$ (*right thick black line*) using a quadratic level set and the shock front (*left thick black line*) by a cubic spline level set ($P_s = 3$).

Mach 4 bow shock For the bow shock study we use various polynomial degrees $P = 0, 1, 2, 3$ and discretize the computational domain Ω using equidistant Cartesian grids comprising 5×16, 10×32, 20×64, and 40×128 cells. As in the test cases presented in 5.1.3, we employ the objective function f_{ER} based on the enriched residual, a cubic spline-based level set ($P_s = 3$) and use the Godunov flux as the numerical flux at the shock interface. For illustrative purposes, we provide plots for the converged solutions for each mesh in Figure 5.28. To assess the accuracy of the numerical solutions, we use the enthalpy error as before.

The results of the spatial convergence study are presented in Figure 5.29, highlighting the method's convergence for different polynomial degrees. The estimated rates of convergence

Table 5.1: Experimental order of convergence (EOC) for different polynomial degrees P for the Mach 4 bow shock. The fixed domain is resolved with different mesh resolutions, and the enthalpy errors (see Figure 5.29) are computed using the XDG-IST method.

P	EOC
0	0.90
1	1.93
2	2.37
3	2.72

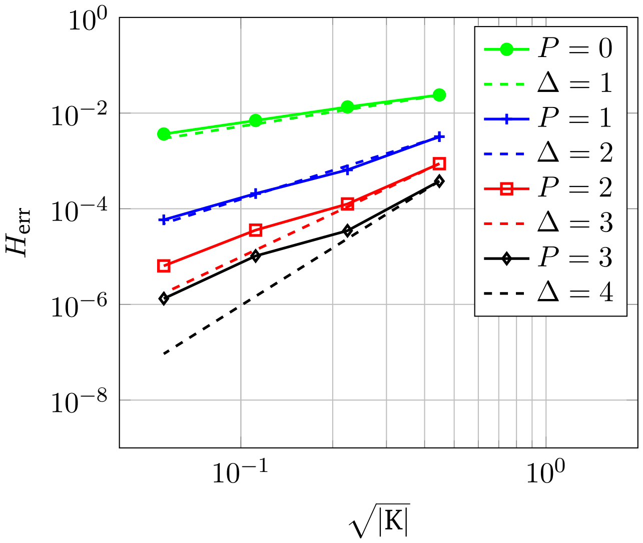


Figure 5.29: Convergence plot for normalized enthalpy error H_{err} for different polynomial degrees ($P = 0, 1, 2, 3$) for the Mach 4 bow shock on a fixed domain resolved with different mesh resolutions $\sqrt{|K|}$, where K denotes a cell. Here, the dashed lines (Δ) show the expected convergence slopes of a high-order method. (Figure adapted from Vandergrift and Kummer, 2024)

increase with polynomial degree, as depicted in Table 5.1. Although these rates are not optimal for a high-order method, they demonstrate a benefit from using higher-order approximations. A reason for the non-optimality could not be established in the course of this research and is left for future endeavors. A speculative explanation could be that the employed spline-based level set does not converge at the same speed as the flow solution and therefor starts to limit the convergence rate. As the grid is refined uniformly in x an y direction, the spline only gets more accurate from y-refinement.

Fast acoustic wave interacting with a shock For the second convergence study, we revisit the test case involving the 1D space-time Euler equations and where a fast acoustic wave hits a shock from the supersonic side, as described in 5.1.4. We employ a series of Cartesian grids comprising 16×16, 31×31, 61×61 and 121×121 cells and compute XDG solutions for polynomial degrees $P \in \{0, 1, 2, 3\}$. We assess the convergence quality by comparing the computed amplification factors $\max_{t \in \mathcal{T}}(p'(x_i, t))$ with the exact amplification factor $\text{amp}^{+}_{\text{ex}}$ defined in (5.29). Specifically, we use the average deviation

$$\text{err}^{+}(t) := |\max_{x \in [1.6, 3.0]} p'(x, t) - \text{amp}^{+}_{\text{ex}}|, \tag{5.31}$$

as our error metric for time $t = 0.8$ at which the acoustic wave has already crossed the shock. Here, $\max_{x \in [1.6, 3.0]} p'(x, t)$ is approximated by computing the maximum from the pressure perturbations evaluated at equally spaced 800 sample points (x_i, t), where $x_i \in [1.6, 3.0]$.

In the convergence plot (Figure 5.30), we display the errors for each combination and the corresponding EOCs are tabulated in Table 5.2. Notably, the EOC for the low-order ($P = 0$) approximation is highly non-optimal (0.06), contrary to the expectation of 1. This discrepancy is attributed to the heavy damping of the acoustic wave, characteristic of low-order schemes.

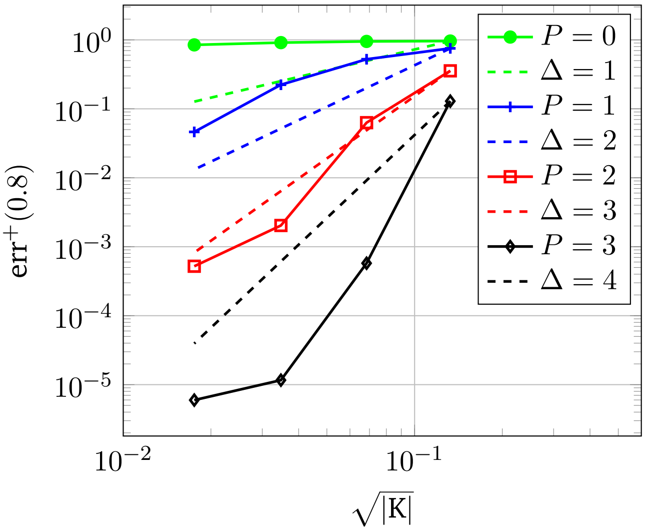


Figure 5.30: Convergence plot for the error in the pressure amplification factor $\mathrm{err}^+(t = 0.8)$ (5.31) for different polynomial degrees ($P = 0, 1, 2, 3$) for the 1D shock-acoustic-wave interaction problem with a fast acoustic wave hitting the shock from the supersonic side. We use a fixed domain resolved with different mesh resolutions $\sqrt{|K|}$, where K denotes a cell. Here, the dashed lines (Δ) show the expected convergence slopes of a high-order method.

Table 5.2: EOC for different polynomial degrees P for the 1D shock-acoustic-wave interaction problem where a fast acoustic wave hits the shock from the supersonic side. The fixed domain is resolved with different mesh resolutions, and the errors (see Figure 5.30) are computed using the XDG-IST method.

P	EOC
0	0.06
1	1.37
2	3.23
3	4.97

A similar, though less pronounced, effect is observed for the $P = 1$ approximation with an EOC of 1.37. However, the EOC values for the high-order approximations ($P = 2, 3$) even surpass the optimal convergence orders, showcasing the superiority of high-order methods in computational aeroacoustics (CAA) applications (Wagner et al., 2007). Notably, the EOC for the $P = 3$ approximation is even super-optimal (4.97).

5.2.5 Comparison with a shock capturing method

In this section, we compare the XDG-IST method to corresponding shock capturing solutions computed using the compressible Navier-Stokes (CNS) solver (see Remark 13). The CNS solver implements a DG immersed boundary method (IBM) for the unsteady 2D Euler equations, i.e., it uses a boundary level set φ_b to immerse boundaries into the flow field if needed. The CNS solver uses artificial viscosity (AV) for shock stabilization (hence, we abbreviate it as a DG-AV method). This means that the considered DG-AV method localizes cells in which the DG solution oscillates, augmenting the Euler equations with an artificial second-order diffusive term. By doing so, the shock is smoothed and the method is stabilized. For the CNS simulations we show in this section, we set the shock capturing parameters (not introduced in this manuscript) as in the work by Geisenhofer (2021), specifically $S_0 = 1.0 \times 10^{-3}$, $\epsilon_0 = 1.0$, $\kappa = 1.0$, and $\lambda_{c,\text{max}} = 15$.

One dimensional shock-acoustic-wave interaction

We start by comparing simulation results for the shock-acoustic-wave interaction problems discussed in Section 5.1.4. These are run using the DG-AV method with a polynomial degree of $P = 3$.

We want to ensure that the overall approximation error is not dominated by the temporal error, thus we employ a 4th order accurate explicit Runge-Kutta scheme for time integration. As the CNS solver is mainly confined to unsteady 2D computations, we setup a pseudo-2D simulation, where all flow variables will be constant along the y-direction (i.e., $\frac{\partial \cdot}{\partial y} = 0$). For this purpose, we employ the domain $\Omega = [0, 3] \times [0, 0.03]$ which is discretized using a 61×3 mesh. At the top and the bottom of the domain Ω slip wall boundary conditions are prescribed, whereas prescribing inflow boundary conditions (defined below) at the other two boundaries. As initial values for all primitive flow variables $\psi \in \{p, u, v, \rho\}$, we set up a smoothed stationary shock wave, that is

$$\psi(x, 0) = \psi_L^\circ(x) - j_{\text{smth}}(x - x_s)(\psi_L^\circ(x) - \psi_R^\circ(x)), \tag{5.32}$$

where the smooth jump function is defined by a tangent hyperbolicus

$$j_{\text{smth}}(x) = \frac{1}{2}\left(\tanh\left(\frac{xP}{2h}\right) + 1\right), \tag{5.33}$$

and where $h = \frac{3}{61}$ is the cell-size. The quantities $(\cdot)_L$ and $(\cdot)_R$ denote left/right values for a stationary Mach 1.5 shock wave and are chosen in accordance to (5.22) and (5.23a)-(5.23c), respectively. Further, we assume no velocity in y-direction, i.e., $v_L, v_R = 0$.

The initial value chosen, although being a close proximity, does not correspond to a stationary solution to the 2D Euler equations with artificial viscosity applied around the shock. The resulting mismatch will therefore produce perturbations moving downstream. To ensure that these artificial perturbations do not interfere with the acoustic waves studied, the simulation is run until $t = 10$ to reach a stationary state. Only then the acoustic wave is introduced in the form of supersonic inflow boundary conditions at the left or right boundary of the domain. For the fast acoustic wave hitting the shock from the supersonic side the inflow boundary conditions

$$\psi(0, t) = \begin{cases} \psi_L & \text{if } t < 10 \\ \psi_L + \psi'(t) & \text{else} \end{cases}, \qquad \psi(3, t) = \psi_R, \quad \psi \in \{p, u, v, \rho\} \tag{5.34}$$

are prescribed at the left boundary of the domain (pre-shock). For the slow acoustic wave hitting the shock from the subsonic side we prescribe

$$\psi(0, t) = \psi_L, \quad \psi(3, t) = \begin{cases} \psi_R & \text{if } t < 10 \\ \psi_R + \psi'(t) & \text{else} \end{cases}, \tag{5.35}$$

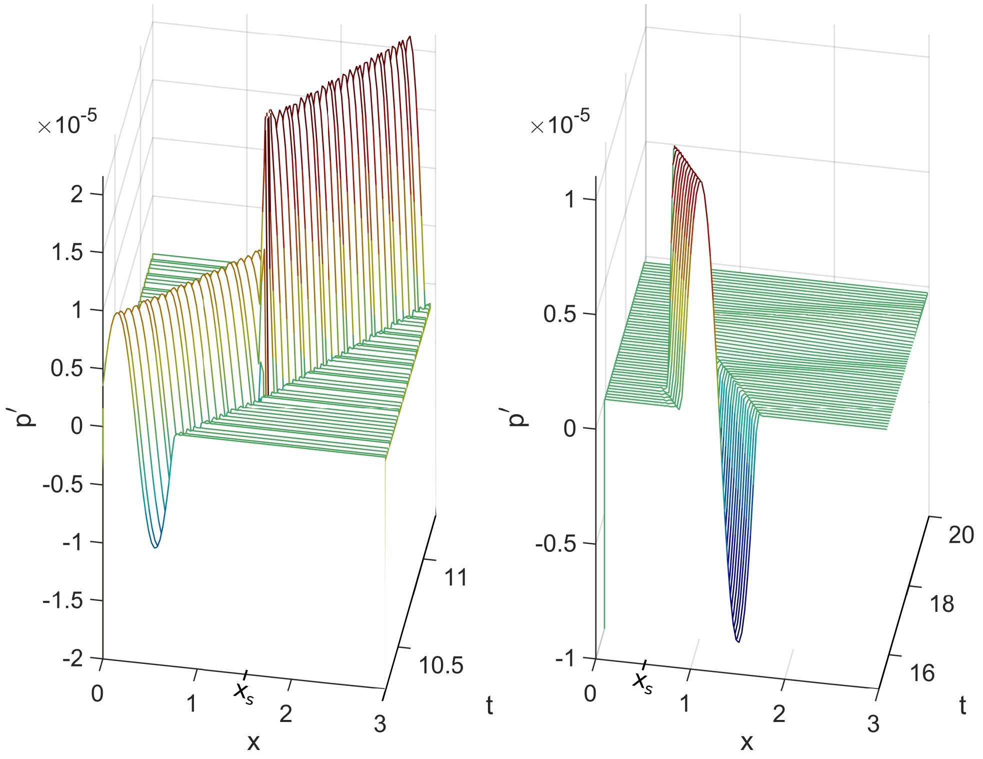


Figure 5.31: Waterfall plot of shock capturing solution (DG-AV) for an 1D shock-acoustic-wave problem. Pressure perturbations $p' = p - p^{\circ}$ for two cases are depicted: a fast acoustic wave hitting the shock from the supersonic side of the domain with the shock initially located at $x_s = 1.5$ (*left*) and a slow acoustic wave hitting the shock from the subsonic side of the domain with the shock initially located at $x_s = 0.5$ (*right*).

at the right domain boundary for $\psi \in \{p, u, v, \rho\}$. The perturbations are chosen as in Section 5.1.4 (with $v' = 0$). They only differ in the wave position x_{W}, which is chosen to be $x_{\mathrm{W}} = -0.8$ for the supersonic fast acoustic wave and $x_{\mathrm{W}} = 3.2$ for the subsonic slow acoustic wave.

In Figure 5.31, we present waterfall plots illustrating the pressure perturbations in the DG-AV solutions for both scenarios, demonstrating that both types of interaction phenomena are captured accurately on a qualitative level (also when compared to Figures 5.17 and 5.19). We then employ the sampling methodology outlined in Section 5.1.4 to quantify the amplification factors of the pressure, as specified in equations (2.27) and (2.29). The quantified results are displayed in Figure 5.32, which also compares them to the XDG-IST solution results (previously shown in Figure 5.18) and the theoretical analytical values.

The analysis shows that the DG-AV shock capturing method exhibits a maximum deviation of approximately 2.3% for the supersonic fast acoustic wave and 22.5% for the slow acoustic wave. In comparison, the XDG-IST method records deviations of 1.18% and 2.2% for the respective waves. As we have chosen the same spatial resolution and the same polynomial degree for both methods and have applied a 4th order accurate time integration scheme for the DG-AV method, our results indicate that our XDG-IST method outperforms the DG-AV method in terms of accuracy for 1D shock-acoustic-wave interactions. The difference for this

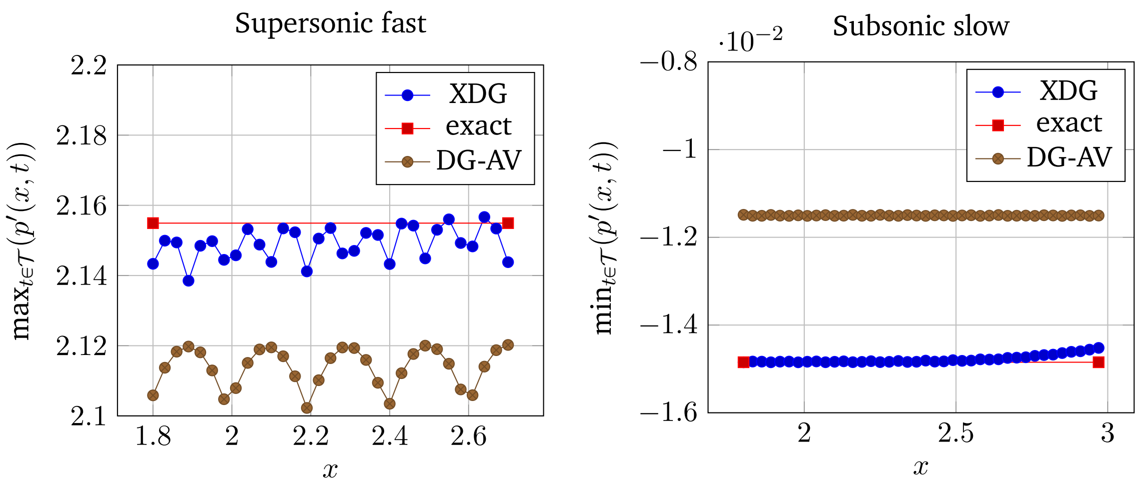


Figure 5.32: Comparison of the pressure amplification of an acoustic wave for the shock-acoustic-wave interaction problem. We compare an XDG shock tracking solution (*blue*), the analytical value (*red*) and a shock capturing (DG-AV) solution (*brown*). Two problem variants are shown. *Left*: A fast acoustic wave hitting the shock from the supersonic side. Here, the solution is evaluated at 100×100 equally-spaced sample points and the maximum $\max_{t\in\mathcal{T}} p'(x_i,t)$ is taken for $x_i \in [1.8, 2.7]$. *Right*: A slow acoustic wave hitting the shock from the subsonic side. Here, the solution is again evaluated at 100×100 equally-spaced sample points and the minimum $\min_{t\in\mathcal{T}} p'(x_i,t)$ is taken for $x_i \in [1.8, 3.0]$.

problem set can be most likely attributed to the conceptual advantage of the IST approach over the application of artificial viscosity. Note however, that with the XDG-IST method solving an implicit problem and the DG-AV method only using explicit time-stepping, we can expect that the computational time needed to compute a solution grows immensely faster for the XDG-IST method when adding more resolution.

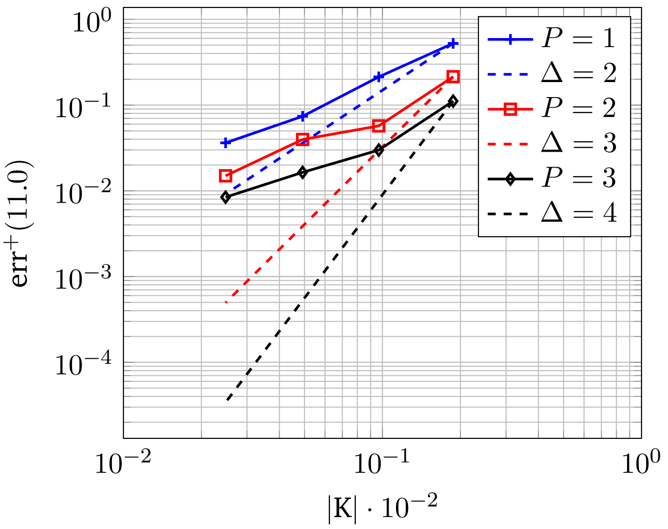


Figure 5.33: Convergence plot for the error in the pressure amplification factor $\mathrm{err}^{+}(11.0)$ (5.31) for the DG-AV method. Using three different polynomial degrees ($P = 1, 2, 3$) solutions to the 1D shock-acoustic-wave interaction problem with fast wave originating on the supersonic side are computed. We use a fixed domain resolved with different mesh resolutions $|\mathrm{K}| \cdot 10^{-2}$, where K denotes a cell. Here, the dashed lines (Δ) show the expected convergence slopes of a high-order method.

Secondly, we conduct a spatial convergence study similar to the one presented in Figure 5.30 using the CNS solver. For this purpose we compute $P = 1, 2, 3$ DG-AV solutions to the fast acoustic wave hitting the shock from the supersonic side for a series of four meshes refined in x-direction (16×3, 31×3, 61×3, 121×3). To measure the error in the pressure amplification factor, we use the metric defined in (5.31). The error is computed for time $t = 11.0$, because then the acoustic wave has crossed the shock (see Figure 5.31) and the amplification is measurable.

Table 5.3: EOC for different polynomial degrees P for the 1D shock-acoustic-wave interaction where a fast acoustic wave hits the shock from the supersonic side. The fixed domain is resolved with different mesh resolutions, and the errors (see Figure 5.33) using the DG-AV shock capturing method.

P	EOC
1	1.31
2	1.32
3	1.28

In Figure 5.33, we have depicted the resulting convergence plot together with high-order convergence slopes and we summarizes the resulting EOCs in Table 5.3. We observe that the DG-AV method shows approximately the same EOC for all three polynomial degrees. This means, that even high polynomial degrees exhibit low-order convergence rates for the DG-AV method. This observation is in contrast to the results we obtained for the XDG-IST method (see Table 5.2). There, we have obtained optimal convergence rates for the high polynomial degrees $P = 2, 3$. As we have employed a 4th-order Runge-Kutta scheme for the DG-AV method, its low-order convergence must be dominated by the spatial error and must therefore result from the artificial viscosity added to smooth the shock. Concluding, we have shown that our method outperforms the DG-AV method for the 1D shock-acoustic-wave interaction problem. Even more notably, among those two it is the only method capable of achieving high-order convergence, showcasing the merit of sharply representing the shock using implicit shock tracking.

2D Steady Euler equations

Next, we compare simulation results for the two 2D steady Euler test cases: the wedge flow and the bow shock (Section 5.1.3). For the DG-AV method we utilize a level set to represent the immersed boundary and set the cell-agglomeration threshold to $\text{agg}_{\text{thrsh}} = 0.3$. Further, we employ the explicit Euler scheme for time integration, computing time steps until a steady state is reached. We then compare the steady solutions (XDG and DG-AV) visually by using the enthalpy error H_{err} defined in (5.17) as an accuracy measure.

Mach 2 wedge flow For the Mach 2 airflow past a wedge with a 10° angle (Section 5.1.3) we use a 60×40 mesh and a polynomial degree $P = 1$ for the DG-AV shock capturing method. We integrate over time until $t_{\text{end}} = 2.0$ reaching a point where the solution is steady. Figure 5.34 shows the density and enthalpy profiles of the DG-AV solution at $t_{\text{end}} = 2.0$ as well as the XDG solution for a coarser 15×10 mesh. The shock capturing method diffuses the shock, reducing the accuracy near the shock and around the blunt body. In contrast, the XDG-IST precisely captures the discontinuity, allowing for a sharp representation of the shock front. The

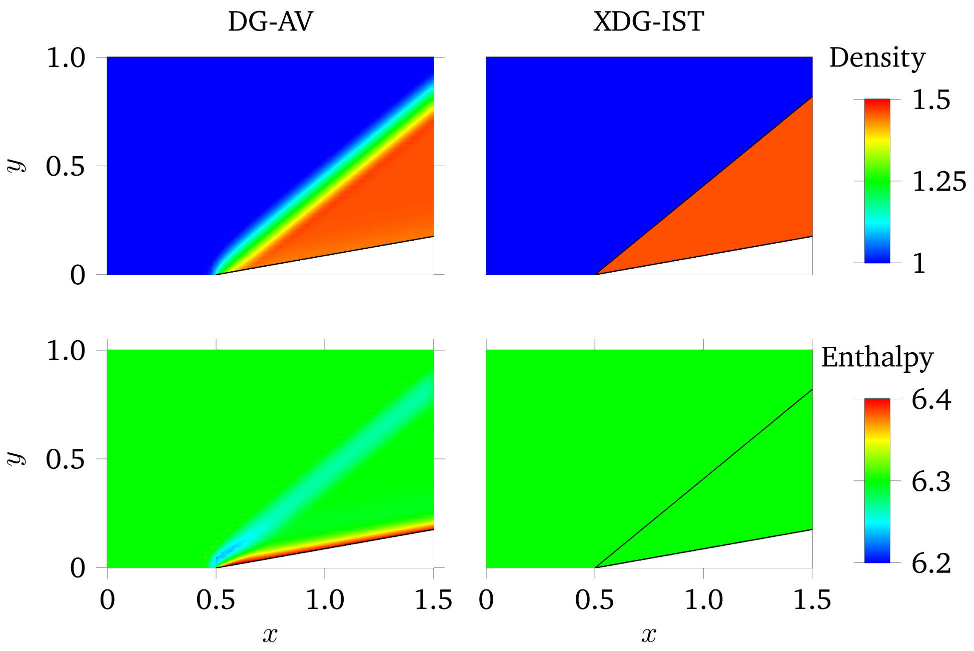


Figure 5.34: Illustrative comparison between a shock capturing (DG-AV) solution $(P = 1)$ of a Mach 2 wedge flow on a 60×40 mesh and an XDG shock tracking solution $(P = 0)$ on a coarse 15×10 mesh (grid lines are not depicted). The density ρ (*top*) and the enthalpy (*bottom*, theoretical value $H_\infty = 6.3$) are shown.

enthalpy error for the XDG method is approximately $H_{\text{err}} \approx 10^{-11}$, significantly lower than the shock capturing error of $H_{\text{err}} = 2 \times 10^{-1}$.

For this pathological case, a better shock capturing approximation could theoretically be achieved with a mesh adapted to the shock. Specifically, employing a refined region close to the shock would undoubtedly enhance the accuracy. Particularly for the straight-sided shock, where the position is known analytically, the mesh could even be designed such that no artificial viscosity is needed, i.e., with mesh edges fitted directly to the shock. Next, we move on to a more interesting case: the Mach 4 bow shock.

Mach 4 bow shock Revisiting the Mach 4 flow scenario over a rounded rectangle, as described in Section 5.1.3, we reuse the simulation settings from (Geisenhofer, 2021) with a fine 40×160 mesh and a polynomial degree of $P = 2$ for the shock capturing solution. There, a quadratic level set is used for the immersed boundary and the simulation runs up to $t_{\text{end}} = 16.0$ until it reaches a stationary solution. In Figure 5.35, we present the density ρ and the enthalpy H (theoretical value $H_\infty = 14.7$), for both the shock capturing (DG-AV) solution on the fine mesh and the $P = 2$ XDG-IST solution on the coarse 5×16 mesh (discussed in Section 5.1.3). Like in the previous case, the shock capturing method results in a smeared shock, reducing accuracy near the shock and the blunt body. Remarkably, the XDG solution, even though the mesh is coarsened by a factor of 80, improves the accuracy and achieves an enthalpy error of $H_{\text{err}} = 8.5 \times 10^{-4}$. The latter is markedly better than the shock capturing enthalpy error of $H_{\text{err}} = 1.8 \times 10^{-2}$.

Again, one can argue that the employed DG-IBM method is not well-suited for this problem. As

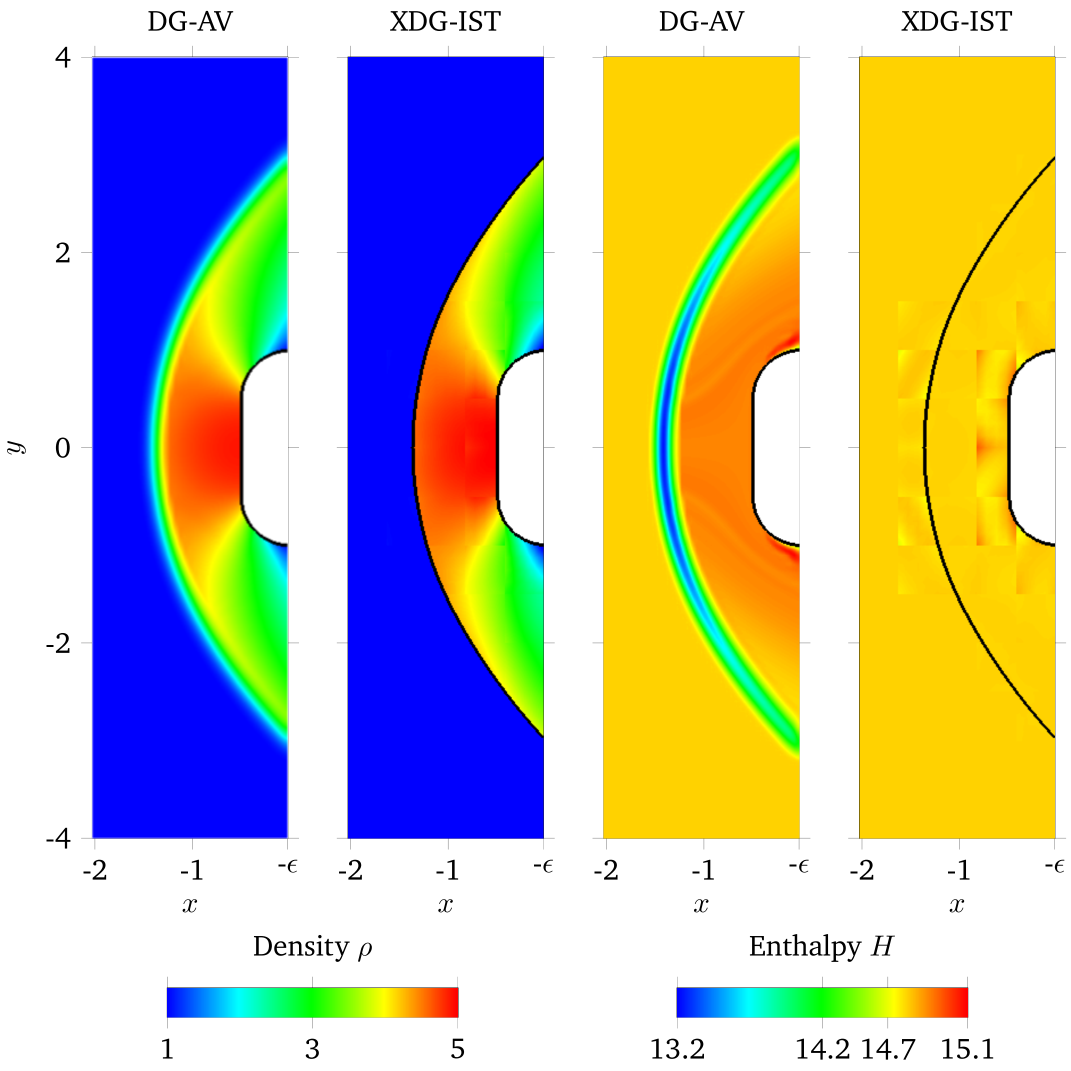


Figure 5.35: Illustrative comparison between a $(P = 2)$ shock capturing solution (DG-AV) of a Mach 4 bow shock on a 40×160 mesh and a $(P = 2)$ XDG shock tracking solution on a coarse 5×16 mesh (grid lines are not depicted). The density ρ (*left*) and the enthalpy H (*right*, theoretical value $H_\infty = 14.7$) are shown.

previously mentioned, the accuracy of the shock capturing (DG-AV) method could be enhanced by a mesh adapted to the shock. Specifically, employing anisotropically refined regions close to the shock and the blunt body, while coarsening the mesh in the constant pre-shock region, would optimize the mesh distribution. By doing so, one could minimize the size of the region where artificial viscosity is applied, resulting in more accurate approximations.

In conclusion, the XDG-IST method demonstrates a clear superiority over the considered DG-AV shock capturing method in terms of accurately approximating the steady flows examined. It is important to note, however, that altering shock capturing parameters or adopting a different shock capturing strategy may influence these results.

5.2.6 Performance study

In the last study, we evaluate the computational expenses associated with the XDG-IST method used together with the enriched residual-based objective function f_{ER}. For smaller scale problems, the primary computational load stems from assembling the linearized optimality system (4.22). This process predominantly involves the computation of Jacobians for the enriched residual $\boldsymbol{R}$ with respect to the solution vector $\boldsymbol{u}$ and the level set coefficients $\boldsymbol{\varphi}$. Additional procedures, such as line search, agglomeration, and reinitialization, though essential, contribute minimally to the total computational cost and their impact does not scale significantly with problem size.

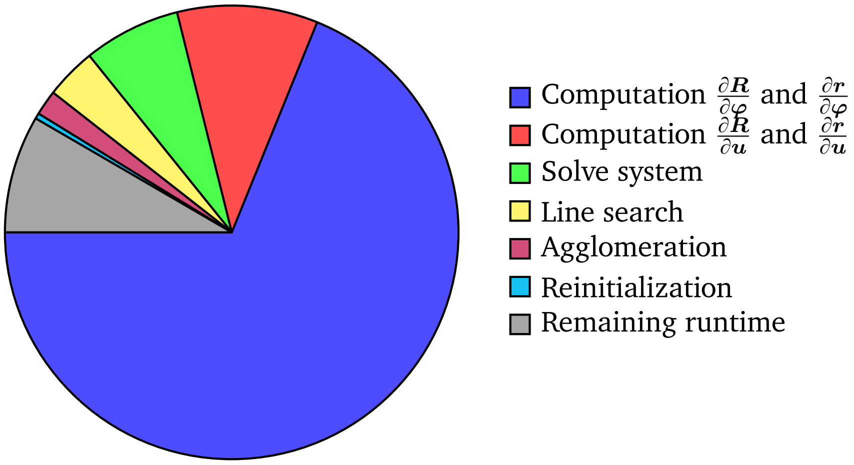


Figure 5.36: Runtime distribution of relevant sub-routines for the implicit XDG shock tracking solver for the Mach 4 bow shock test case on a 10×32 mesh. (Figure adapted from Vandergrift and Kummer, 2024)

This characteristic is illustrated in Figure 5.36 where we present the runtime distribution for critical subroutines involved in the XDG-IST for the Mach 4 bow shock scenario using a 10×32 mesh (as described in Section 5.1.3. It is apparent from the data that the computations of $\frac{\partial \mathbf{R}}{\partial \varphi}$ are particularly resource-intensive, consuming approximately two-thirds of the total computational effort for the given example. This significant demand originates primarily from the intricate computations involved in applying Saye quadrature rules (Saye, 2015) to each modified level set, a complexity previously discussed in Section 4.6. As a result, we constitute a clear need for a level set representation that allows for analytical derivative calculations.

Finally, it should be noted that while not the primary focus of this study, the performance of the algorithm could potentially be improved through more efficient implementation strategies.

5.3 Conclusion

In this section, we conclude the numerical results for the XDG-IST method which were presented in this chapter. First, results for the test cases are concluded in Section 5.3.1. Secondly, we draw conclusions from the numerical studies presented in Section 5.3.2.

5.3.1 Test cases

A series of test cases for the XDG-IST method was introduced and we presented numerical results from applying the method across four different systems of conservation laws: the 1D space-time Burgers equation, the 1D space-time advection equation, the 1D space-time Euler equations, and the steady 2D Euler equations. The XDG-IST method was applied using spline-based level sets and the full enriched residual-based objective function f_{ER}.

In the context of the linear 1D space-time advection equation, two test cases where presented featuring piece-wise constant flow solutions separated by a straight-sided and a curved discontinuity, respectively. We could show, that for both cases the discontinuities were perfectly tracked by the XDG-IST method, even when initializing the level set with a non-sub-cell accurate first guess. Also, the residual norms were minimized to at least 10^{-9}, showcasing the capabilities of our method to robustly solve a linear problem.

Similarly, for the nonlinear 1D space-time Burgers equation we developed two test cases with a straight-sided and a curved shock, with the curved test case featuring a non-polynomial flow solution. As before, the XDG-IST method was able to track the discontinuities and approximate the solution for both cases. While the residuals were minimized to 10^{-8} for the straight-sided case, the XDG-IST method converged to residuals around 10^{-2} for the curved test case. Hence, we demonstrated the ability of the XDG-IST method to solve these nonlinear problems accurately.

For the steady 2D Euler equations, we applied the XDG-IST method together with Godunov's numerical flux at the interface to two test cases: supersonic Mach 2 flow over an inclined plane and supersonic Mach 4 flow over a blunt body. The first case featured a straight-sided shock and a piece-wise constant solution, both perfectly tracked by our method. Also for the second case, when applying the XDG-IST method to a Mach 4 bow shock, we obtained remarkably well resolved solutions and low enthalpy errors on coarse meshes. These results showcased the methods capabilities of solving steady 2D supersonic flow problems.

Lastly, we applied the XDG-IST method to three 1D shock-acoustic-wave interaction problems in the context of the 1D space-time Euler equations. We presented waterfall plots of the XDG solutions showing the expected phenomena which result from the shock-acoustic-wave interaction. For two of the problems we then compared our results with the expected pressure amplification/reduction factor, showing good agreement with theory and deviations under 3%. Our results indicate the ability of the XDG-IST method to accurately capture the relevant physical phenomena for unsteady 1D shock-acoustic-wave interaction problems.

5.3.2 Numerical studies

Throughout our research, we evaluated various configurations of the XDG-IST method to enhance its effectiveness and efficiency. Specifically, we tested three distinct level set discretizations: the DG level set, the CG level set and the spline-based level set. Employing a straightforward space-time scalar advection problem featuring a straight-sided shock, we determined that the spline-based level set was uniquely capable of precisely tracking the shock while minimizing the residual. Additionally, analyzing the computational time needed by each approach revealed that the spline-based level set not only performed but also scaled best, indicating its suitability for middle-scale 2D applications.

We also investigated the impact of different objective functions (full enriched residual, enriched residual confined to the near-band, Rankine-Hugoniot conditions) on the XDG-IST method using test cases from the steady 2D Euler equations as benchmarks. Our findings suggest a clear preference for the full enriched residual-based objective function, especially evident in the bow shock scenario where it was the only approach that achieved convergence, thus demonstrating its superiority.

Further analysis involved comparing various numerical fluxes at the shock interface edges for the steady 2D Euler equations. We observed that the central flux was too unstable, and the HLLC flux is inconsistent with the shock solution, as anticipated. In contrast, the Roe flux without entropy fix, newly integrated into the BoSSS framework, proved to be a robust and efficient alternative to the traditionally used Godunov flux, supporting its use in this context.

A convergence study was conducted using the Mach 4 bow shock test case across four nested meshes and using the deviation from the known enthalpy as the error metric. Although the XDG-IST method did not achieve optimal convergence rates at polynomial degrees $P = 2$ and $P = 3$, high-order convergence was still observed, affirming the method's potential for accurate high-resolution simulations. Furthermore, we have conducted a second convergence study for a 1D shock-acoustic-wave problem, using the amplification factor of the acoustic wave to measure the error. Here, our results showcased super-optimal high-order convergence rates for the degrees $P = 2, 3$, while the convergence was degraded for $P = 0, 1$. Hence, we demonstrated the potential that high-order IST methods hold for the accurate simulation of shock-acoustic-wave interaction phenomena.

Additionally, we assessed the performance of the XDG-IST method against a traditional DG-IBM, implemented as the CNS solver within the BoSSS framework and which incorporates a shock capturing strategy. In evaluations involving 2D steady flows, the XDG method exhibited superior accuracy, notably achieving improved enthalpy error metrics on significantly coarsened meshes. This superiority was also evident in unsteady 1D shock-acoustic-wave interactions, where the XDG-IST method outperformed in terms of pressure wave amplification/reduction factors, particularly for the slow acoustic wave emerging on the subsonic side. Subsequently, we replicated the convergence study initially conducted for the XDG-IST method using the CNS solver. Remarkably, the CNS solver demonstrated low-order convergence rates across polynomial degrees $P = 1, 2, 3$. This underscores the conceptual advantage in terms of accuracy per DOF that IST methods have over traditional shock-capturing techniques, as IST methods maintain high-order convergence for shock-acoustic-wave phenomena. Note however, that IST methods are solving an implicit problem while shock capturing methods rely on explicit time-stepping. Thus, we can expect that the computational time needed to compute a solution grows immensely faster for the IST methods when adding more resolution.

Lastly, our comprehensive evaluation of computational performance highlighted that the majority of the computational time (>60%) of the XDG-IST method was devoted to calculating Jacobians of XDG residuals with respect to DOFs associated with the spline-based level set. This finding underscores the computational intensity associated with the level set DOFs, despite the presence of a greater number of DOFs in the flow solution, pointing to potential areas for optimization in future work.

6 Linear solvers for implicit shock tracking

In the realm of computational fluid dynamics (CFD), solving large systems of linear equations efficiently is crucial for modeling fluid behaviors accurately. These systems, especially in three-dimensional (3D) scenarios, are often very large due to the detailed discretization required, making traditional direct solving methods, like matrix factorization, impractical due to their high demands on memory and computational power. Instead, iterative linear solvers are preferred, using fewer resources and effectively speeding up computations. These solvers are particularly good at scaling up for larger problems, adapting well to parallel computing environments, which is essential for modern CFD simulations. However, iterative methods have to often be enhanced using preconditioning techniques, improving their efficiency by optimizing the condition number of the matrix. These preconditioners often have to be specifically designed for the method and problem considered.

In this chapter, a close adaption of the research presented in Vandergrift and Zahr (2024) (see also Remark 19), we propose a set of preconditioners designed specifically for linearized optimality systems used in sequential quadratic programming (SQP) solvers in implicit shock tracking methods. We particularly examine the mesh-based high-order implicit shock tracking (HOIST) method (Zahr et al., 2020; Huang and Zahr, 2022), which incorporates an enriched residual as the objective function. However, our preconditioners can be adapted to other objectives, such as the Rankine-Hugoniot conditions (Corrigan et al., 2019a), and to XDG-based shock tracking (Chapter 4). In SQP, advancing towards an optimal solution involves solving the linearized Karush-Kuhn-Tucker (KKT) conditions. Our proposed preconditioners are based on *constrained preconditioners* (Keller et al., 2000) that emulate the structure of original saddle point problems. These preconditioners have been effectively combined with methods like conjugate gradients and other Krylov subspace techniques to tackle nonlinear programming issues (Coleman and Verma, n.d.; Gould et al., 2001; Lukšan and Vlček, 2001; Dollar, 2007). Our approach simplifies the constraint Jacobian using standard discontinuous Galerkin (DG) methods such as block Jacobi and block incomplete LU factorization, selectively ignoring parts of the Lagrangian Hessian while approximating the remaining parts with standard techniques (Persson and Peraire, 2008; Biros and Ghattas, 2000). We also incorporate a two-level P-multigrid strategy that enhances any preconditioner in our family. Extensive tests on two non-viscous compressible flow problems assess each preconditioner's effectiveness and sensitivity to crucial factors like mesh quality, Hessian regularization, state of linearization, and resolution.

This chapter is organized as follows: Section 6.1 introduces the HOIST method and details the sparsity pattern of the linearized optimality system. Then, widely-used preconditioners for DG methods and how these are adapted into specialized matrix-based preconditioners for the implicit shock tracking linearized systems are discussed in Section 6.2. Finally, Section 6.3

reports detailed experiments with these preconditioners, analyzing their performance against several key optimization solver parameters.

Remark 19. *This chapter copies most of the content from the manuscript by Vandergrift and Zahr (2024) with minor modifications to integrate it into this thesis. The manuscript was primarily authored by the author of the present work, bearing the sole responsibility for the actual implementation and evaluation of the methods, creating visual representations, and writing the initial draft of the manuscripts. The co-author, Assistant Professor Matthew J. Zahr, served in an advisory role. He was instrumental in developing the original ideas, assisting with their continuous refinement, and providing the necessary CFD codes that form the foundation of the investigated shock tracking test cases. His contributions to the article were mainly in the realms of initial conceptualization, proofreading, and making minor corrections.*

6.1 The high-order implicit shock tracking method

In this section, we briefly describe the HOIST method, a mesh-based implicit DG shock tracking method originally developed by Zahr and Persson (2018), as we aim to developed specialized linear solvers for the corresponding linearized optimality systems dictating the next optimization update in each step. The HOIST method, which has been adapted to the implicit extended discontinuous Galerkin (XDG) shock tracking context in Chapter 4, uses numerical optimization to simultaneously compute a high-order approximation and align elements of the computational mesh with non-smooth features. Further, it employs an objective function based on an enriched residual and a mesh-quality term.

In this section we summarize relevant aspects of the HOIST method in Section 6.1.1 and sparsity structure of the linear optimality system, which needs to be solved efficiently in each iteration step, in Section 6.1.2.

6.1.1 High-order implicit shock tracking formulation and solver

In this section, we review the optimization formulation and the SQP solver (Huang and Zahr, 2022) on which the HOIST method is based, including the linear system that defines the SQP step. Here, we rely on concepts related to the discretization of the transformed system of conservation laws introduced in Chapter 3. Note, that in the case of the HOIST method, and thus in the remainder of this chapter, we assume that no interface (defined by a level set) is present inside the computational domain Ω (see Remark 8) and hence we drop the corresponding dependency of the residuals, that is

$$\boldsymbol{r}(\boldsymbol{u},\boldsymbol{x},\varphi)=\boldsymbol{r}(\boldsymbol{u},\boldsymbol{x}),\boldsymbol{R}(\boldsymbol{u},\boldsymbol{x},\varphi)=\boldsymbol{R}(\boldsymbol{u},\boldsymbol{x}). \tag{6.1}$$

Also, in this chapter, the discretization is conducted employing a DG space on the background grid, which is equivalent to an XDG space on a cut-cell grid when no interfaces are cutting the domain.

Remark 20. *As both methods are closely related, we reuse notation introduced for the implicit XDG shock tracking method. Note however, that some concepts might have a slightly different meaning in the mesh-based shock tracking context. For instance, instead of depending on the coefficients of the shock level set φ, relevant functions (e.g. $\boldsymbol{r}, \boldsymbol{R}, \mathcal{L}$) depend on the unconstrained mesh coordinates $\boldsymbol{y}$.*

HOIST formulation

The HOIST method (Zahr et al., 2020; Huang and Zahr, 2022) is a high-order technique that simultaneously computes both the discrete solution of the conservation law and the nodal coordinates of the mesh, aligning element faces with discontinuities. This process is accomplished using a fully discrete, full-space optimization formulation, where the optimization variables consist of the discrete flow solution and the nodal coordinates of the mesh.

With a boundary-preserving parameterization $\boldsymbol{\phi}$ of the mesh motion (introduced in 3.18) the HOIST method is formulated as: find the solution $(\boldsymbol{u}^\star, \boldsymbol{y}^\star)$ to the constrained minimization problem

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{u}, \boldsymbol{y}) \\ \text{subject to} \quad & \boldsymbol{r}(\boldsymbol{u}, \boldsymbol{\phi}(\boldsymbol{y})) = 0, \end{aligned} \tag{6.2}$$

where $f : \mathbb{R}^{N_{\boldsymbol{u}}} \times \mathbb{R}^{N_{\boldsymbol{y}}} \to \mathbb{R}$ is the objective function, $\boldsymbol{r} : \mathbb{R}^{N_{\boldsymbol{u}}} \times \mathbb{R}^{N_{\boldsymbol{y}}} \to \mathbb{R}^{N_{\boldsymbol{u}}}$ is the algebraic residual (defined in Section 3.2.3) and the nodal coordinates of the aligned mesh are $\boldsymbol{x}^\star = \boldsymbol{\phi}(\boldsymbol{y}^\star)$. Contrary to the XDG shock tracking context, the objective function is dependent on mesh variables (see Remark 20) rather than on level set coefficients and is composed of two terms as

$$f : (\boldsymbol{u}, \boldsymbol{y}) \mapsto f_{\text{err}}(\boldsymbol{u}, \boldsymbol{y}) + \kappa^2 f_{\text{msh}}(\boldsymbol{y}), \tag{6.3}$$

balancing alignment of the mesh with non-smooth features and the quality of the elements. Here, $\kappa \in \mathbb{R}_{\geqslant 0}$ is an adaptively chosen mesh penalty parameter to weight the two terms such that the first term is prioritized (Huang and Zahr, 2022). The mesh alignment term, $f_{\text{err}} : \mathbb{R}^{N_{\boldsymbol{u}}} \times \mathbb{R}^{N_{\boldsymbol{y}}} \to \mathbb{R}$, is taken to be the norm of the enriched DG residual

$$f_{\text{err}} : (\boldsymbol{u}, \boldsymbol{y}) \mapsto \frac{1}{2} \left\| \boldsymbol{R}(\boldsymbol{u}, \boldsymbol{\phi}(\boldsymbol{y})) \right\|_2^2 . \tag{6.4}$$

We also want to ensure that the elements of the discontinuity-aligned mesh are of high quality, which leads to the definition of the mesh distortion term, $f_{\text{msh}} : \mathbb{R}^{N_{\boldsymbol{y}}} \to \mathbb{R}$, as

$$f_{\text{msh}} : \boldsymbol{y} \mapsto \frac{1}{2} \left\| \boldsymbol{R}_{\text{msh}}(\boldsymbol{\phi}(\boldsymbol{y})) \right\|_2^2 , \tag{6.5}$$

where $\boldsymbol{R}_{\text{msh}} : \mathbb{R}^{N_{\boldsymbol{y}}} \to \mathbb{R}^{|\mathcal{K}_h|}$ is the elemental mesh distortion with respect to an ideal element (Zahr et al., 2020; Knupp, 2001; Roca et al., 2012).

To obtain the first-order optimality system of the implicit shock tracking formulation (6.2), we introduce the corresponding Lagrangian, $\mathcal{L} : \mathbb{R}^{N_{\boldsymbol{u}}} \times \mathbb{R}^{N_{\boldsymbol{y}}} \times \mathbb{R}^{N_{\boldsymbol{u}}} \to \mathbb{R}$, defined as

$$\mathcal{L} : (\boldsymbol{u}, \boldsymbol{y}, \boldsymbol{\lambda}) \mapsto f(\boldsymbol{u}, \boldsymbol{y}) - \boldsymbol{\lambda}^T \boldsymbol{r}(\boldsymbol{u}, \boldsymbol{\phi}(\boldsymbol{y})). \tag{6.6}$$

Then, the first-order optimality, or KKT, conditions state that $(\boldsymbol{u}^\star, \boldsymbol{y}^\star) \in \mathbb{R}^{N_{\boldsymbol{u}}} \times \mathbb{R}^{N_{\boldsymbol{y}}}$ is a first-order solution of the optimization problem in (6.2) if there exists $\boldsymbol{\lambda}^\star \in \mathbb{R}^{N_{\boldsymbol{u}}}$ such that the Lagrangian is stationary, i.e.

$$\nabla \mathcal{L}(\boldsymbol{u}^\star, \boldsymbol{y}^\star, \boldsymbol{\lambda}^\star) = 0. \tag{6.7}$$

Sequential quadratic programming solver

Next, we briefly describe the SQP solver (Huang and Zahr, 2022) for the optimization problem in (6.2). It is a full-space approach that aims to converge the DG solution $\boldsymbol{u}$ and the mesh $\boldsymbol{y}$ to their optimal values simultaneously. To this end, we define a new variable $\boldsymbol{z} \in \mathbb{R}^{N_{\boldsymbol{z}}}$ ($N_{\boldsymbol{z}} = N_{\boldsymbol{u}} + N_{\boldsymbol{y}}$) that combines the DG solution $\boldsymbol{u}$ and the unconstrained mesh coordinates $\boldsymbol{y}$ as

$$\boldsymbol{z} = (\boldsymbol{u}, \boldsymbol{y}), \tag{6.8}$$

and use $\boldsymbol{z}$ interchangeably with $(\boldsymbol{u}, \boldsymbol{y})$. For brevity, we introduce the following notation for the derivatives of the objective function, $\boldsymbol{g} : \mathbb{R}^{N_{\boldsymbol{z}}} \to \mathbb{R}^{N_{\boldsymbol{z}}}$, and the DG residual, $\boldsymbol{J} : \mathbb{R}^{N_{\boldsymbol{z}}} \to \mathbb{R}^{N_{\boldsymbol{u}}} \times \mathbb{R}^{N_{\boldsymbol{z}}}$, as

$$\boldsymbol{g} : \boldsymbol{z} \mapsto \begin{bmatrix} \dfrac{\partial f}{\partial \boldsymbol{u}}(\boldsymbol{u}, \boldsymbol{y})^T \\ \dfrac{\partial f}{\partial \boldsymbol{y}}(\boldsymbol{u}, \boldsymbol{y})^T \end{bmatrix}, \quad \boldsymbol{J} : \boldsymbol{z} \mapsto \begin{bmatrix} \boldsymbol{J}_{\boldsymbol{u}}(\boldsymbol{u}, \boldsymbol{y}) & \boldsymbol{J}_{\boldsymbol{y}}(\boldsymbol{u}, \boldsymbol{y}) \end{bmatrix}. \tag{6.9}$$

Here,

$$\boldsymbol{J}_{\boldsymbol{u}}(\boldsymbol{u}, \boldsymbol{y}) := \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}(\boldsymbol{u}, \boldsymbol{\phi}(\boldsymbol{y})), \quad \boldsymbol{J}_{\boldsymbol{y}}(\boldsymbol{u}, \boldsymbol{y}) := \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}}(\boldsymbol{u}, \boldsymbol{\phi}(\boldsymbol{y}))\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{y}}(\boldsymbol{y}) \tag{6.10}$$

denote the Jacobians with respect to $\boldsymbol{u}$ and $\boldsymbol{y}$, respectively. The SQP method in Huang and Zahr (2022) produces a sequence of iterates $\{\boldsymbol{z}_k\}_{k=0}^{\infty}$ such that $\boldsymbol{z}_k = (\boldsymbol{u}_k, \boldsymbol{y}_k) \to \boldsymbol{z}^\star = (\boldsymbol{u}^\star, \boldsymbol{y}^\star)$, where $(\boldsymbol{u}^*, \boldsymbol{y}^*)$ satisfies the first-order optimality conditions in (6.7). The sequence of iterates is generated as

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \alpha_k \Delta \boldsymbol{z}_k, \tag{6.11}$$

where the search direction $\Delta \boldsymbol{z}_k \in \mathbb{R}^{N_{\boldsymbol{z}}}$ is computed as the solution of the following quadratic program

$$\begin{aligned} \underset{\Delta \boldsymbol{z} \in \mathbb{R}^{N_{\boldsymbol{z}}}}{\text{minimize}} \quad & \boldsymbol{g}_k^T \Delta \boldsymbol{z} + \frac{1}{2} \Delta \boldsymbol{z}^T \boldsymbol{B}_k \Delta \boldsymbol{z} \\ \text{subject to} \quad & \boldsymbol{r}_k + \boldsymbol{J}_k \Delta \boldsymbol{z} = \boldsymbol{0} \end{aligned}, \tag{6.12}$$

$\boldsymbol{g}_k \in \mathbb{R}^{N_{\boldsymbol{z}}}$, $\boldsymbol{r}_k \in \mathbb{R}^{N_{\boldsymbol{u}}}$, and $\boldsymbol{J}_k \in \mathbb{R}^{N_{\boldsymbol{u}} \times N_{\boldsymbol{z}}}$ are the objective gradient, residual, and residual Jacobian, respectively, evaluated at $\boldsymbol{z}_k$

$$\boldsymbol{r}_k := \boldsymbol{r}(\boldsymbol{u}_k, \boldsymbol{\phi}(\boldsymbol{y}_k)), \qquad \boldsymbol{g}_k := \boldsymbol{g}(\boldsymbol{z}_k), \qquad \boldsymbol{J}_k := \boldsymbol{J}(\boldsymbol{z}_k), \tag{6.13}$$

$\boldsymbol{B}_k \in \mathbb{R}^{N_{\boldsymbol{z}} \times N_{\boldsymbol{z}}}$ is a symmetric positive definite (SPD) approximation to the Hessian of the Lagrangian at $\boldsymbol{z}_k$, and $\alpha_k \in \mathbb{R}_{>0}$ is the step length. The latter is computed by an inexact line search employing a standard l_1-merit function (Nocedal and Wright, 2006) and the first-order optimality conditions of the quadratic program lead to the following linear system of equations

$$\begin{bmatrix} \boldsymbol{B}_k & \boldsymbol{J}_k^T \\ \boldsymbol{J}_k & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{z}_k \\ \boldsymbol{\eta}_k \end{bmatrix} = - \begin{bmatrix} \boldsymbol{g}_k \\ \boldsymbol{r}_k \end{bmatrix}, \tag{6.14}$$

where $\boldsymbol{\eta}_k \in \mathbb{R}^{N_{\boldsymbol{u}}}$ are the Lagrange multipliers associated with the linearized constraint in (6.12). This linear system of size $2N_{\boldsymbol{u}} + N_{\boldsymbol{x}}$ must to be solved at each iteration k to compute the step $\Delta \boldsymbol{z}_k$ to update the DG solution and mesh (6.11). For large-scale applications with many DOFs, direct solvers are not a viable option as these systems are larger than standard DG system (size: $N_{\boldsymbol{u}}$). In the next section, we explore the structure of the linear system in (6.14) that will facilitate the development of efficient preconditioners and iterative linear solvers in Section 6.2.

Remark 21. *This SQP method (Zahr et al., 2020) proved not to be robust enough to handle complex problems, such as high Mach number flows with complex discontinuities, so several robustness measures were introduced (Huang and Zahr, 2022). These measures manipulate the state $\boldsymbol{z}_{k+1}$ after the SQP update only for a fixed number of iterations $M > 0$ (to ensure SQP convergence in the limit) and include 1) boundary-preserving, shock-aware element removal, 2) geometric curvature removal from inverted or ill-conditioned elements, and 3) elemental solution reinitialization; see Huang and Zahr (2022) for details. These operations have a small positive impact on linear solvers as they locally reduce sources of ill-conditioning, which can lead to abrupt (positive) changes in the performance of iterative solvers when comparing between different states (Section 6.3).*

Hessian approximation

Implicit shock tracking methods employ a Levenberg-Marquardt Hessian approximation (Corrigan et al., 2019b; Zahr et al., 2020) to define $\boldsymbol{B}_k$ (see Section 4.2.2 for a more detailed discussion in the context of implicit XDG shock tracking). To this end, $\boldsymbol{B}_k$ is expanded as

$$\boldsymbol{B}_k = \begin{bmatrix} \boldsymbol{B}_{\boldsymbol{uu},k} & \boldsymbol{B}_{\boldsymbol{uy},k} \\ \boldsymbol{B}_{\boldsymbol{uy},k}^T & \boldsymbol{B}_{\boldsymbol{yy},k} \end{bmatrix}, \tag{6.15}$$

where the individual components $\boldsymbol{B}_{\boldsymbol{uu},k} \in \mathbb{R}^{N_{\boldsymbol{u}} \times N_{\boldsymbol{u}}}$, $\boldsymbol{B}_{\boldsymbol{uy},k} \in \mathbb{R}^{N_{\boldsymbol{u}} \times N_{\boldsymbol{y}}}$, and $\boldsymbol{B}_{\boldsymbol{yy},k} \in \mathbb{R}^{N_{\boldsymbol{y}} \times N_{\boldsymbol{y}}}$ are defined as

$$\begin{aligned} \boldsymbol{B}_{\boldsymbol{uu},k} &:= \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}(\boldsymbol{z}_k)^T \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}(\boldsymbol{z}_k) \\ \boldsymbol{B}_{\boldsymbol{uy},k} &:= \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}(\boldsymbol{z}_k)^T \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{y}}(\boldsymbol{z}_k) \\ \boldsymbol{B}_{\boldsymbol{yy},k} &:= \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{y}}(\boldsymbol{z}_k)^T \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{y}}(\boldsymbol{z}_k) + \gamma_k \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{y}}(\boldsymbol{y}_k)^T \boldsymbol{D}_k \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{y}}(\boldsymbol{y}_k), \end{aligned} \tag{6.16}$$

where $\boldsymbol{F} : \mathbb{R}^{N_{\boldsymbol{u}}} \times \mathbb{R}^{N_{\boldsymbol{y}}} \to \mathbb{R}^{N'_{\boldsymbol{u}} + |\mathcal{K}_h|}$ is the residual function

$$\boldsymbol{F} : (\boldsymbol{u}, \boldsymbol{y}) \mapsto \begin{bmatrix} \boldsymbol{R}(\boldsymbol{u}, \boldsymbol{\phi}(\boldsymbol{y})) \\ \kappa \boldsymbol{R}_{\text{msh}}(\boldsymbol{\phi}(\boldsymbol{y})) \end{bmatrix}, \tag{6.17}$$

and $\boldsymbol{D}_k \in \mathbb{R}^{N_{\boldsymbol{x}} \times N_{\boldsymbol{x}}}$ is a SPD matrix constructed to regularize the mesh motion. The regularization parameter $\gamma_k \in \mathbb{R}_{\geqslant 0}$ is chosen adaptively (and analogously to (4.32)) during the optimization process and has a strong impact on the number of iterations needed for an iterative solver, which will be observed in the numerical experiments in Section 6.3.5.

6.1.2 Sparsity of linear optimality system

In this section, we analyze the sparsity of the linear optimality system in order to develop requirements for efficient linear solvers. We start by examining the sparsity of the discrete operators $\partial \boldsymbol{r}/\partial \boldsymbol{u}$, $\partial \boldsymbol{R}/\partial \boldsymbol{u}$, which form the basis of the full system.

Sparsity of discrete operators

The sparsity structure of the Jacobians of the DG residuals $\boldsymbol{r}$ and $\boldsymbol{R}$ is examined with respect to the variables $\boldsymbol{u}$ and $\boldsymbol{x}$, as they will be central to the shock tracking optimization method. For any element $K_e \in \mathcal{K}_h$, let $\boldsymbol{u}_e \in \mathbb{R}^{N_P}$ and $\boldsymbol{x}_e \in \mathbb{R}^{N_{\boldsymbol{x}}^e}$ denote the degrees of freedom (DOFs) of $\boldsymbol{u}$ and $\boldsymbol{x}$, respectively, associated with element K_e, where $N_P = m \dim[\mathcal{P}_P(K_e)]$ and $N_{\boldsymbol{x}}^e = d \dim[\mathcal{P}_{P_c}(K_e)]$. The elemental DOFs are related to the global DOFs via the selection matrices, $\boldsymbol{P}_e \in \{0,1\}^{N_{\boldsymbol{u}} \times N_P}$ and $\boldsymbol{Q}_e \in \{0,1\}^{N_{\boldsymbol{x}} \times N_{\boldsymbol{x}}^e}$, which are subsets of the identity matrix that extract selected rows from $N_{\boldsymbol{u}}$- and $N_{\boldsymbol{x}}$-vectors, respectively,

$$\boldsymbol{u}_e = \boldsymbol{P}_e^T \boldsymbol{u}, \qquad \boldsymbol{x}_e = \boldsymbol{Q}_e^T \boldsymbol{x}. \tag{6.18}$$

Furthermore, denote the DOFs corresponding to the neighbors of element K_e as $\hat{\boldsymbol{u}}_e \in \mathbb{R}^{\hat{N}_P^e}$, where $\hat{N}_P^e = m \dim[\mathcal{P}_P(\mathcal{N}_e)]$, and $\hat{\boldsymbol{P}}_e \in \{0,1\}^{N_{\boldsymbol{u}} \times \hat{N}_P^e}$ as the corresponding selection matrix such that

$$\hat{\boldsymbol{u}}_e = \hat{\boldsymbol{P}}_e^T \boldsymbol{u}. \tag{6.19}$$

Here, $\mathcal{N}_e \subset \mathcal{K}_h$ is the collection of elements neighboring (i.e., sharing a face with) element K_e.

With this notation, the elemental DG residuals, $r_{K_e}^{P,P}$ and $r_{K_e}^{P',P}$, can be written algebraically as

$$\boldsymbol{r}_e : \mathbb{R}^{N_P} \times \mathbb{R}^{\hat{N}_P^e} \times \mathbb{R}^{N_{\boldsymbol{x}}^e} \to \mathbb{R}^{N_P}, \qquad \boldsymbol{r}_e : (\boldsymbol{u}_e, \hat{\boldsymbol{u}}_e, \boldsymbol{x}_e) \mapsto \boldsymbol{r}_e(\boldsymbol{u}_e, \hat{\boldsymbol{u}}_e, \boldsymbol{x}_e) \tag{6.20}$$

$$\boldsymbol{R}_e : \mathbb{R}^{N_P} \times \mathbb{R}^{\hat{N}_P^e} \times \mathbb{R}^{N_{\boldsymbol{x}}^e} \to \mathbb{R}^{N_{P'}}, \qquad \boldsymbol{R}_e : (\boldsymbol{u}_e, \hat{\boldsymbol{u}}_e, \boldsymbol{x}_e) \mapsto \boldsymbol{R}_e(\boldsymbol{u}_e, \hat{\boldsymbol{u}}_e, \boldsymbol{x}_e). \tag{6.21}$$

The global residuals are formed by summing over all elements and assembling into the appropriate degree of freedom (DOF) as

$$\boldsymbol{r}(\boldsymbol{u}, \boldsymbol{x}) = \sum_{e=1}^{|\mathcal{K}_h|} \boldsymbol{P}_e \boldsymbol{r}_e(\boldsymbol{u}_e, \hat{\boldsymbol{u}}_e, \boldsymbol{x}_e) = \sum_{e=1}^{|\mathcal{K}_h|} \boldsymbol{P}_e \boldsymbol{r}_e(\boldsymbol{P}_e^T \boldsymbol{u}, \hat{\boldsymbol{P}}_e^T \boldsymbol{u}, \boldsymbol{Q}_e^T \boldsymbol{x}). \tag{6.22}$$

Direct differentiation leads to an expression for the Jacobian $\boldsymbol{J}_{\boldsymbol{u}}(\boldsymbol{u}, \boldsymbol{x}) \in \mathbb{R}^{N_{\boldsymbol{u}} \times N_{\boldsymbol{u}}}$ that exposes its block structure

$$\boldsymbol{J}_{\boldsymbol{u}}(\boldsymbol{u}, \boldsymbol{x}) := \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}(\boldsymbol{u}, \boldsymbol{x}) = \sum_{e=1}^{|\mathcal{K}_h|} \boldsymbol{P}_e \left(\frac{\partial \boldsymbol{r}_e}{\partial \boldsymbol{u}_e}(\boldsymbol{u}_e, \hat{\boldsymbol{u}}_e, \boldsymbol{x}_e) \boldsymbol{P}_e^T + \frac{\partial \boldsymbol{r}_e}{\partial \hat{\boldsymbol{u}}_e}(\boldsymbol{u}_e, \hat{\boldsymbol{u}}_e, \boldsymbol{x}_e) \hat{\boldsymbol{P}}_e^T \right), \tag{6.23}$$

where $\frac{\partial \boldsymbol{r}_e}{\partial \boldsymbol{u}_e}(\boldsymbol{u}_e, \hat{\boldsymbol{u}}_e, \boldsymbol{x}_e) \in \mathbb{R}^{N_P \times N_P}$, $\frac{\partial \boldsymbol{r}_e}{\partial \hat{\boldsymbol{u}}_e}(\boldsymbol{u}_e, \hat{\boldsymbol{u}}_e, \boldsymbol{x}_e) \in \mathbb{R}^{N_P \times \hat{N}_{\boldsymbol{u}}^e}$ are its matrix blocks. The matrix $\boldsymbol{J}_{\boldsymbol{u}}$ is a $|\mathcal{K}_h| \times |\mathcal{K}_h|$ block matrix with blocks of size $N_P \times N_P$. For an example two-dimensional mesh consisting of 9 elements (Figure 6.1), a polynomial degree of $P = 1$, and a single conservation law ($m = 1$), the sparsity of $\boldsymbol{J}_{\boldsymbol{u}}$ is shown in Figure 6.1. Using the same arguments, one can derive the sparsity pattern for the Jacobian $\partial \boldsymbol{R} / \partial \boldsymbol{u}$, the only difference being that the blocks $\frac{\partial \boldsymbol{R}_e}{\partial \boldsymbol{u}_e} \in \mathbb{R}^{N_{P'} \times N_P}$ have a different size to account for the additional constraints. The sparsity pattern of $(\partial \boldsymbol{R} / \partial \boldsymbol{u})^T$ for the exemplary mesh in Figure 6.1 is shown in Figure 6.2 with 9 blocks of size 3×6 coming from the enriched polynomial degree $P' = 2$.

Remark 22. *The elemental residuals do not depend on the neighboring nodes $\hat{\boldsymbol{x}}_e$ as the coupling to the neighboring elements is only due to trace values on the boundaries. For a more detailed discussion the reader is referred to the work by Wen and Zahr (2023).*

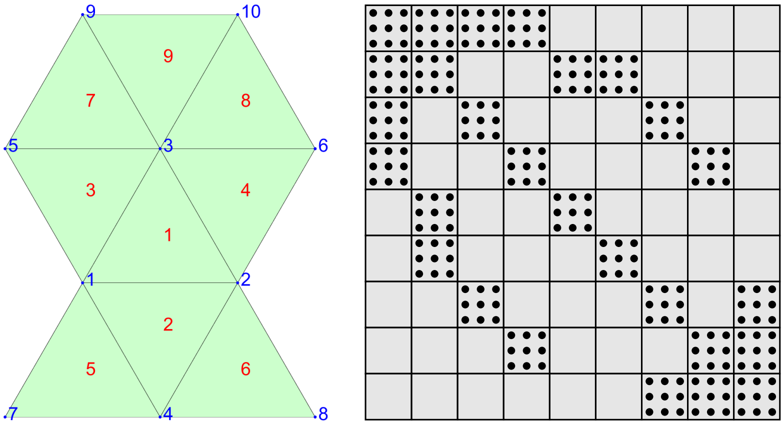


Figure 6.1: Example two-dimensional mesh (*left*) (10 nodes and 9 elements) and corresponding sparsity structure of $\boldsymbol{J}_{\boldsymbol{u}}$ (*right*) for a polynomial degree of $P = 1$ and a single conservation law ($m = 1$). This choice leads to 9 blocks of size 3×3 for $\boldsymbol{J}_{\boldsymbol{u}}$.

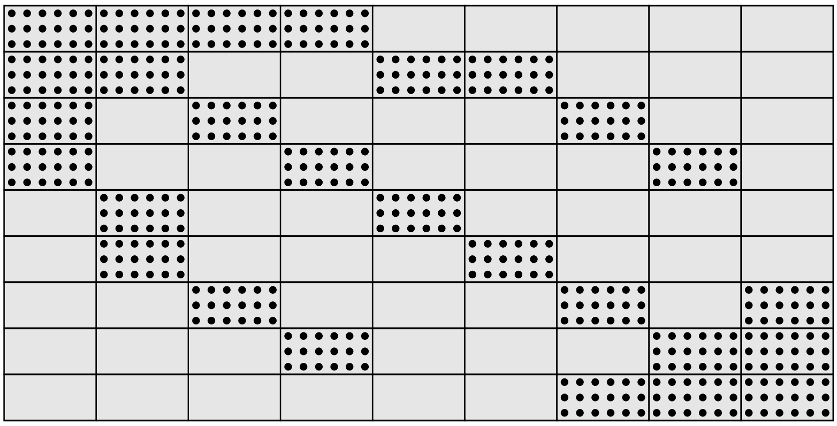


Figure 6.2: Sparsity structure of $(\partial \boldsymbol{R}/\partial \boldsymbol{u})^T$ for the mesh in Figure 6.1, polynomial degrees of $P = 1,\ P' = 2$ and a single conservation law ($m = 1$). This choice results in 9 blocks of size 6×3 for $\partial \boldsymbol{R}/\partial \boldsymbol{u}$.

Sparsity of full system

We detail the sparsity of the linear system (6.14) as it has significant implications for the design requirements of efficient preconditioners. From this point forward, we fix the state $\boldsymbol{z}_k$ and drop the k subscript on all terms. As we already examined the sparsity of $\boldsymbol{J}_{\boldsymbol{u}}$ we begin with $\boldsymbol{B}_{\boldsymbol{uu}}$. First, recall that $\partial \boldsymbol{R}_{\text{msh}}/\partial \boldsymbol{u} = 0$ by construction. Therefore, we can derive the block sparsity structure using the elemental decomposition of the DG Jacobian as follows

$$\begin{aligned}
\boldsymbol{B}_{\boldsymbol{uu}} &= \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}^T \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}} = \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u}}^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u}} \\
&= \left(\sum_{e=1}^{|\mathcal{K}_h|} \boldsymbol{P}'_e \left(\frac{\partial \boldsymbol{R}_e}{\partial \boldsymbol{u}_e} \boldsymbol{P}_e^T + \frac{\partial \boldsymbol{R}_e}{\partial \hat{\boldsymbol{u}}_e} \hat{\boldsymbol{P}}_e^T\right)\right)^T \left(\sum_{E=1}^{|\mathcal{K}_h|} \boldsymbol{P}'_E \left(\frac{\partial \boldsymbol{R}_E}{\partial \boldsymbol{u}_E} \boldsymbol{P}_E^T + \frac{\partial \boldsymbol{R}_E}{\partial \hat{\boldsymbol{u}}_e} \hat{\boldsymbol{P}}_E^T\right)\right) \qquad (6.24) \\
&= \sum_{e,E=1}^{|\mathcal{K}_h|} \left(\left(\boldsymbol{P}_e \frac{\partial \boldsymbol{R}_e}{\partial \boldsymbol{u}_e}^T + \hat{\boldsymbol{P}}_e \frac{\partial \boldsymbol{R}_e}{\partial \hat{\boldsymbol{u}}_e}^T\right) (\boldsymbol{P}'_e)^T \boldsymbol{P}'_E \left(\frac{\partial \boldsymbol{R}_E}{\partial \boldsymbol{u}_E} \boldsymbol{P}_E^T + \frac{\partial \boldsymbol{R}_E}{\partial \hat{\boldsymbol{u}}_e} \hat{\boldsymbol{P}}_E^T\right)\right).
\end{aligned}$$

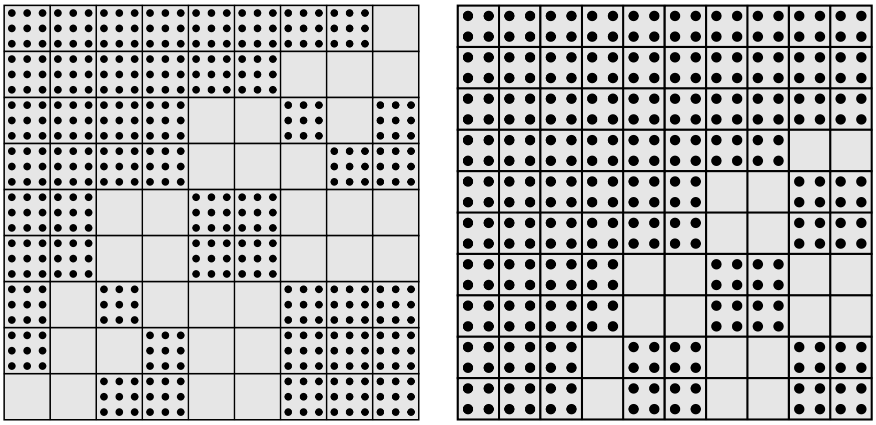


Figure 6.3: Sparsity structure of $\boldsymbol{B}_{\boldsymbol{uu}}$ (*left*) and $\boldsymbol{B}_{\boldsymbol{yy}}$ (*right*) (assuming no boundary constraints, i.e. $\phi(\boldsymbol{y}) = \boldsymbol{y}$) for mesh depicted in Figure 6.1 with polynomial degrees of $P = 1,\ P' = 2$ and a single conservation law ($m = 1$).

Due to the fact that $(\boldsymbol{P}_e')^T \boldsymbol{P}_E' = \boldsymbol{I}\delta_{eE}$ (because $\boldsymbol{R}$ is a DG residual), we finally obtain

$$\boldsymbol{B}_{\boldsymbol{uu}} = \sum_{e=1}^{|\mathcal{K}_h|} \left(\boldsymbol{P}_e \frac{\partial \boldsymbol{R}_e}{\partial \boldsymbol{u}_e}^T \frac{\partial \boldsymbol{R}_e}{\partial \boldsymbol{u}_e} \boldsymbol{P}_e^T + \boldsymbol{P}_e \frac{\partial \boldsymbol{R}_e}{\partial \boldsymbol{u}_e}^T \frac{\partial \boldsymbol{R}_e}{\partial \hat{\boldsymbol{u}}_e} \hat{\boldsymbol{P}}_e^T + \hat{\boldsymbol{P}}_e \frac{\partial \boldsymbol{R}_e}{\partial \hat{\boldsymbol{u}}_e}^T \frac{\partial \boldsymbol{R}_e}{\partial \boldsymbol{u}_e} \boldsymbol{P}_e^T + \hat{\boldsymbol{P}}_e \frac{\partial \boldsymbol{R}_e}{\partial \hat{\boldsymbol{u}}_e}^T \frac{\partial \boldsymbol{R}_e}{\partial \hat{\boldsymbol{u}}_e} \hat{\boldsymbol{P}}_e^T \right). \tag{6.25}$$

From this identity it can be deduced that $\boldsymbol{B}_{\boldsymbol{uu}}$ has an element based block structure like $\boldsymbol{J}_{\boldsymbol{u}}$, but with an extended (denser) sparsity pattern. The difference lies in the additional non-zero blocks due to the neighbor-neighbor interaction $\hat{\boldsymbol{P}}_e \frac{\partial \boldsymbol{R}_e}{\partial \hat{\boldsymbol{u}}_e}^T \frac{\partial \boldsymbol{R}_e}{\partial \hat{\boldsymbol{u}}_e} \hat{\boldsymbol{P}}_e^T$, which does not exist for $\boldsymbol{J}_{\boldsymbol{u}}$. This amounts to an increase of non-zero blocks by a factor of $1 + d(d+1)/(d+2)$ for a simplicial mesh (see Remark 23), which is expensive and memory-intensive (Table 6.1) and requires parallel communication of the blocks to form the product. Therefore, in the next section, we will avoid preconditioners that require explicitly forming $\boldsymbol{B}_{\boldsymbol{uu}}$. On the other hand, matrix-vector products of the form $\boldsymbol{B}_{\boldsymbol{uu}}\boldsymbol{v}$ can be performed efficiently as $\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u}}^T(\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u}}\boldsymbol{v})$, making it well-suited for use with an iterative (Krylov) solver. For illustrative purposes, the sparsity of $\boldsymbol{B}_{\boldsymbol{uu}}$ is shown in Figure 6.3 (*left*) for the same exemplary mesh as in Figure 6.1, and the substantial decrease in sparsity can be observed.

Remark 23. *Let us quantify the sparsity of $\boldsymbol{J}_{\boldsymbol{u}}$ relative to $\boldsymbol{B}_{\boldsymbol{uu}}$ for simplicial grids. Let m_1 and m_2 denote the number of non-zero blocks per row of $\boldsymbol{J}_{\boldsymbol{u}}$ and $\boldsymbol{B}_{\boldsymbol{uu}}$, respectively. For simplicity, we consider a row corresponding to an element sufficiently far from a boundary to avoid enumerating special cases. Because the eth block row of $\boldsymbol{J}_{\boldsymbol{u}}$ has a non-zero for each element neighboring K_e, we have $m_1 = d + 2$ (the block diagonal plus $d + 1$ neighbors). On the other hand, the eth block row of $\boldsymbol{B}_{\boldsymbol{uu}}$ has a non-zero for all neighbors of K_e and all neighbors of neighbors of K_e, which gives $m_2 = m_1 + d(d + 1)$ (each of the $d + 1$ neighbors of K_e adds at most d new neighbors). Thus, the ratio of non-zero blocks in $\boldsymbol{B}_{\boldsymbol{uu}}$ to those in $\boldsymbol{J}_{\boldsymbol{u}}$ is*

$$\frac{m_2}{m_1} = 1 + \frac{d(d+1)}{d+2}, \tag{6.26}$$

which is a significant factor (Table 6.1), especially considering the DG Jacobians themselves are already memory-intensive to form and store. This motivates our decision to avoid explicitly forming $\boldsymbol{B}_{\boldsymbol{uu}}$ in the proposed preconditioners in Section 6.2.3.

Table 6.1: Growth of block sparsity structure of $\boldsymbol{B}_{\boldsymbol{uu}}$ (m_2) relative to $\boldsymbol{J}_{\boldsymbol{u}}$ (m_1).

d	1	2	3	4
m_2/m_1	1.67	2.5	3.4	4.33

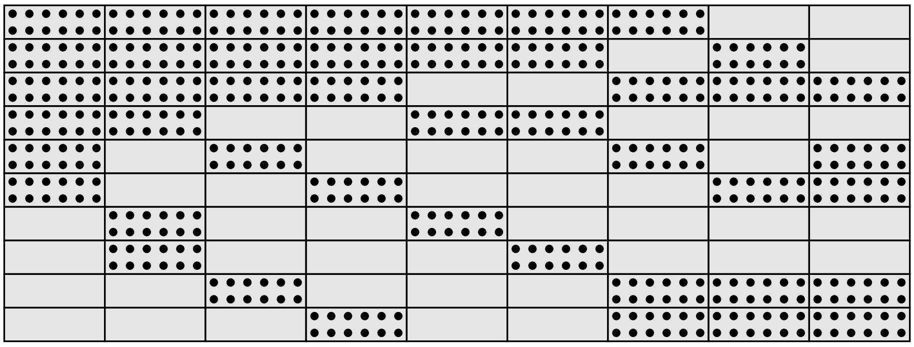


Figure 6.4: Sparsity structure of $(\partial \boldsymbol{R}/\partial \boldsymbol{x})^T$ for mesh depicted in Figure 6.1 with polynomial degrees $P=1,\ P'=2$ and a single conservation law $(m=1)$.

Next, we examine the sparsity of $\boldsymbol{B}_{\boldsymbol{uy}}$ and $\boldsymbol{B}_{\boldsymbol{yy}}$. First, we build the DG Jacobian with respect to $\boldsymbol{y}$ using the Jacobian of the $\boldsymbol{\phi}$ mapping

$$\frac{\partial}{\partial \boldsymbol{y}}\boldsymbol{R}(\boldsymbol{u},\boldsymbol{\phi}(\boldsymbol{y})) = \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}(\boldsymbol{u},\boldsymbol{\phi}(\boldsymbol{y}))\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{y}}(\boldsymbol{y}) = \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}(\boldsymbol{u},\boldsymbol{x})\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{y}}(\boldsymbol{y}). \tag{6.27}$$

Direct differentiation of (6.22) with respect to $\boldsymbol{x}$ (and replacing $\boldsymbol{r}$ with $\boldsymbol{R}$) exposes the assembled block structure of the enriched DG residual

$$\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}(\boldsymbol{u},\boldsymbol{x}) = \sum_{e=1}^{|\mathcal{K}_h|} \boldsymbol{P}'_e \frac{\partial \boldsymbol{R}_e}{\partial \boldsymbol{x}_e}(\boldsymbol{P}_e^T\boldsymbol{u}, \hat{\boldsymbol{P}}_e^T\boldsymbol{u}, \boldsymbol{Q}_e^T\boldsymbol{x})\boldsymbol{Q}_e^T = \sum_{e=1}^{|\mathcal{K}_h|} \boldsymbol{P}'_e \frac{\partial \boldsymbol{R}_e}{\partial \boldsymbol{x}_e}(\boldsymbol{u}_e, \hat{\boldsymbol{u}}_e, \boldsymbol{x}_e)\boldsymbol{Q}_e^T, \tag{6.28}$$

where $\frac{\partial \boldsymbol{R}_e}{\partial \boldsymbol{x}_e} \in \mathbb{R}^{N_{P'} \times N_{\boldsymbol{x}}^e}$. Note that because most nodes $\boldsymbol{x}_e$ are shared between two elements, it is not possible to obtain an elemental block structure for the columns (elemental block structure does exist for the rows). For illustrative purposes only, we assume no boundary constraints $(\boldsymbol{\phi}(\boldsymbol{y}) = \boldsymbol{y})$ and refer to Figure 6.1 for our exemplary mesh. The sparsity pattern of $(\partial \boldsymbol{R}/\partial \boldsymbol{x})^T$ is illustrated in Figure 6.4, revealing 9 block rows.

Going further, we obtain the following sparsity-block structure for $\boldsymbol{B}_{\boldsymbol{uy}}$ from the following identity

$$\begin{aligned}\boldsymbol{B}_{\boldsymbol{uy}} &= \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}^T \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{y}} = \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u}}^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{y}} \\ &= \left(\sum_{e=1}^{|\mathcal{K}_h|} \boldsymbol{P}'_e \left(\frac{\partial \boldsymbol{R}_e}{\partial \boldsymbol{u}_e}\boldsymbol{P}_e^T + \frac{\partial \boldsymbol{R}_e}{\partial \hat{\boldsymbol{u}}_e}\hat{\boldsymbol{P}}_e^T\right)\right)^T \left(\sum_{E=1}^{|\mathcal{K}_h|} \boldsymbol{P}'_E \frac{\partial \boldsymbol{R}_E}{\partial \boldsymbol{x}_E}\boldsymbol{Q}_E^T\right)\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{y}} \\ &= \sum_{e=1}^{|\mathcal{K}_h|}\left(\boldsymbol{P}_e \frac{\partial \boldsymbol{R}_e}{\partial \boldsymbol{u}_e}^T + \hat{\boldsymbol{P}}_e \frac{\partial \boldsymbol{R}_e}{\partial \hat{\boldsymbol{u}}_e}^T\right)\left(\frac{\partial \boldsymbol{R}_e}{\partial \boldsymbol{x}_e}\boldsymbol{Q}_e^T\right)\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{y}}.\end{aligned}$$

The structure of $\partial \boldsymbol{r}/\partial \boldsymbol{y}$ is identical to that of $\partial \boldsymbol{R}/\partial \boldsymbol{y}$ by repeating the above derivation. Because this is a (rectangular) off-diagonal term, the proposed preconditioners and linear solvers only require products with $\boldsymbol{B}_{\boldsymbol{uy}}$, which can be computed as $\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{u}}^T (\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}(\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{y}}\boldsymbol{v}))$ for any vector $\boldsymbol{v}$, so $\boldsymbol{B}_{\boldsymbol{uy}}$ never needs to be explicitly computed.

Lastly, we consider the structure of $\boldsymbol{B}_{\boldsymbol{yy}}$. From a simple application of the chain rule, we have

$$\boldsymbol{B}_{\boldsymbol{yy}} = \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{y}}^T \boldsymbol{B}_{\boldsymbol{xx}} \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{y}}, \tag{6.29}$$

where

$$\boldsymbol{B}_{\boldsymbol{xx}} = \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}} + \kappa^2 \frac{\partial \boldsymbol{R}_{\text{msh}}}{\partial \boldsymbol{x}}^T \frac{\partial \boldsymbol{R}_{\text{msh}}}{\partial \boldsymbol{x}} + \gamma \boldsymbol{D}. \tag{6.30}$$

Furthermore, from (6.28), we have

$$\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}} = \left(\sum_{e=1}^{|\mathcal{K}_h|} \boldsymbol{P}'_e \frac{\partial \boldsymbol{R}_e}{\partial \boldsymbol{x}_e} \boldsymbol{Q}_e^T \right)^T \left(\sum_{E=1}^{|\mathcal{K}_h|} \boldsymbol{P}'_E \frac{\partial \boldsymbol{R}_E}{\partial \boldsymbol{x}_E} \boldsymbol{Q}_E^T \right) = \sum_{e=1}^{|\mathcal{K}_h|} \left(\boldsymbol{Q}_e \frac{\partial \boldsymbol{R}_e}{\partial \boldsymbol{x}_e}^T \frac{\partial \boldsymbol{R}_e}{\partial \boldsymbol{x}_e} \boldsymbol{Q}_e^T \right), \tag{6.31}$$

which loses block structure once assembled because of the overlapping entries in $\boldsymbol{Q}_e$ for different elements. The sparsity of $\frac{\partial \boldsymbol{R}_{\text{msh}}}{\partial \boldsymbol{x}}^T \frac{\partial \boldsymbol{R}_{\text{msh}}}{\partial \boldsymbol{x}}$ is a subset of $\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}$ because each entry of the mesh distortion $\boldsymbol{R}_{\text{msh}}$ is defined individually for each element K_e and solely depends on the element nodes $\boldsymbol{x}_e$. The sparsity of the regularization matrix, $\boldsymbol{D}$, depends solely on its specific choice. We choose $\boldsymbol{D}$ as the linear elasticity (isotropic) stiffness matrix, with the elasticity modulus being inversely proportional to the volume of elements in the reference mesh. Thus, the sparsity of $\boldsymbol{D}$ is a subset of $\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}^T \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{x}}$, as it originates from the continuous finite element discretization of the elasticity equations. Finally, the mapping $\boldsymbol{\phi}$ determines the final structure of $\boldsymbol{B}_{\boldsymbol{yy}}$ and an illustrative example for the sparsity structure of $\boldsymbol{B}_{\boldsymbol{xx}}$ is provided in Figure 6.3 (*right*).

6.2 Iterative linear solvers and preconditioners

In this section, we introduce preconditioners for implicit shock tracking linearized systems, which are derived from successful preconditioners utilized for DG methods. We begin with a brief overview of Krylov iterative solvers (Section 6.2.1) and review commonly used preconditioners for DG discretizations (Section 6.2.2). Finally, we present the novel preconditioners for implicit shock tracking (Section 6.2.3).

6.2.1 Krylov solvers and preconditioning

In this work, we consider Krylov subspace methods for solving the linear system $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{b}$. Krylov methods only require the action of the matrix $\boldsymbol{A}$ on vectors, not the entire matrix itself, which minimizes storage cost. This is particularly advantageous for implicit shock tracking because it allows us to avoid explicitly forming all blocks of $\boldsymbol{B}$ (the regularized Lagrangian Hessian approximation).

On the other hand, Krylov methods rely on preconditioning, i.e., transformation of the system $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{b}$ to enhance its spectral properties. Left preconditioning is achieved by multiplying the linear system on the left by some non-singular matrix $\tilde{\boldsymbol{A}}^{-1}$ to yield

$$\tilde{\boldsymbol{A}}^{-1} \boldsymbol{A}\boldsymbol{s} = \tilde{\boldsymbol{A}}^{-1} \boldsymbol{b}, \tag{6.32}$$

which has the same solution as the original system. Here, $\tilde{\boldsymbol{A}} \approx \boldsymbol{A}$ is the preconditioner and must be inexpensive to apply its inverse to a vector $\tilde{\boldsymbol{A}}^{-1}\boldsymbol{v}$ to be practical. Generally, as the preconditioner approaches the original matrix $\boldsymbol{A}$, the number of Krylov iterations decreases while the associated costs increase (only one iteration is necessary if $\tilde{\boldsymbol{A}} = \boldsymbol{A}$). Finding a suitable preconditioner that balances the need for fewer Krylov iterations with increased costs per iteration requires a specialized solution tailored to the matrix structure, discretization method, and equations at hand. The most effective preconditioners require all or part of the matrix $\boldsymbol{A}$, which partially neutralizes the matrix-free benefits of Krylov methods. In Section 6.2.3, we will develop matrix-based preconditioners for implicit shock tracking that (1) build on established preconditioners for the DG system (Section 6.2.2) and (2) avoid forming the entire $\boldsymbol{B}$ matrix.

6.2.2 Preconditioners for discontinuous Galerkin methods

Two established matrix-based preconditioners for the DG system ($\boldsymbol{A} = \boldsymbol{J}_{\boldsymbol{u}}, \boldsymbol{b} = -\boldsymbol{r}(\boldsymbol{u})$) introduced in the work by Persson and Peraire (2008), are the block Jacobi preconditioner (Section 6.2.2) and the block incomplete LU factorization (BILU) preconditioner with minimum discarded fill (MDF) (Section 6.2.2). Both preconditioners utilize the block structure of the Jacobian matrix $\boldsymbol{J}_{\boldsymbol{u}}$ and are efficient in terms of computational cost and memory to form and apply. They will be building blocks for HOIST preconditioners.

Block Jacobi

The block Jacobi preconditioner is obtained by setting all blocks of the original matrix $\boldsymbol{A}$ off the diagonal to zero, which can be written compactly as

$$\tilde{\boldsymbol{J}}_{\mathrm{BJ}} := \sum_{e=1}^{|\mathcal{K}_h|} \boldsymbol{P}_e \frac{\partial \boldsymbol{r}_e}{\partial \boldsymbol{u}_e} \boldsymbol{P}_e^T. \tag{6.33}$$

This block diagonal preconditioner can be easily formed from $\boldsymbol{J}_{\boldsymbol{u}}$ and its inverse can be explicitly formed by inverting each $N_P \times N_P$ block as

$$\tilde{\boldsymbol{J}}_{\mathrm{BJ}}^{-1} = \sum_{e=1}^{|\mathcal{K}_h|} \boldsymbol{P}_e \left(\frac{\partial \boldsymbol{r}_e}{\partial \boldsymbol{u}_e} \right)^{-1} \boldsymbol{P}_e^T. \tag{6.34}$$

Because the size of each block is relatively small, a direct solver can be used. According to Persson and Peraire (2008) this preconditioner shows good performance in specific cases, but loses effectivity as the Reynolds number (when used for the compressible Navier-Stokes equations) or timestep (when used for transient problems) increases, and in the low Mach limit.

Block incomplete LU preconditioning with minimum discarded fill reordering

A more advanced preconditioner is the BILU factorization with MDF, which is achieved by performing an ILU0 factorization of the matrix $\boldsymbol{A}$ on the block level. This procedure involves limiting a standard LU factorization to maintain the sparsity structure of $\boldsymbol{A}$, i.e., any operation that would introduce new non-zero blocks (known as "fill in") are skipped. To optimize the performance of an incomplete LU factorization (ILU), it is augmented with an initial re-ordering of the matrix block rows to minimize fill-in. Readers are referred to Persson and Peraire (2008) for the complete algorithm and implementation details.

The preconditioner $\tilde{\boldsymbol{J}}_{\text{BILU}}$ is formed as $\tilde{\boldsymbol{P}}\tilde{\boldsymbol{J}}_{\text{BILU}} = \tilde{\boldsymbol{L}}\tilde{\boldsymbol{U}}$, where $\tilde{\boldsymbol{P}}$ is the MDF reordering permutation, $\tilde{\boldsymbol{L}}$ is a lower block-triangular matrix with the identity matrix along the diagonal, and $\tilde{\boldsymbol{U}}$ is an upper block-triangular matrix; both $\tilde{\boldsymbol{L}}$ and $\tilde{\boldsymbol{U}}$ that share the same sparsity pattern as $\boldsymbol{J}_{\boldsymbol{u}}$. Because of the complementary structure of $\tilde{\boldsymbol{L}}$ and $\tilde{\boldsymbol{U}}$, the matrix $\boldsymbol{A}$ can be mutated in-place into $\tilde{\boldsymbol{L}}$ (strict lower block triangle) and $\tilde{\boldsymbol{U}}$ (upper block triangle). To apply the inverse of $\tilde{\boldsymbol{J}}_{\text{BILU}}$ to a vector $\boldsymbol{w}$ ($\tilde{\boldsymbol{J}}_{\text{BILU}}^{-1}\boldsymbol{w}$), we must solve the system $\tilde{\boldsymbol{J}}_{\text{BILU}}\boldsymbol{v} = \boldsymbol{w}$. First, we multiply this equation by the permutation and substitute the ILU factorization

$$\tilde{\boldsymbol{P}}\tilde{\boldsymbol{J}}_{\text{BILU}}\boldsymbol{v} = \tilde{\boldsymbol{L}}\tilde{\boldsymbol{U}}\boldsymbol{v} = \tilde{\boldsymbol{P}}\boldsymbol{w}. \tag{6.35}$$

Then, we apply the usual forward-backward substitution process to solve for $\boldsymbol{v}$: first solve $\tilde{\boldsymbol{L}}\tilde{v} = \tilde{\boldsymbol{P}}\boldsymbol{w}$ for $\tilde{v}$ using block forward substitution, then solve $\tilde{\boldsymbol{U}}\boldsymbol{v} = \tilde{v}$ for $\boldsymbol{v}$ using block backward substitution. Because the block diagonal of $\tilde{\boldsymbol{L}}$ are identity matrices, forward substitution only requires matrix-vector products at the element level. On the other hand, backward substitution requires solving linear systems of size $N_P \times N_P$, which is usually performed with a direct solver because of the relatively small size. According to Persson and Peraire (2008) this preconditioner works effectively for a wide range of problems, particularly when combined with P-multigrid.

6.2.3 Preconditioners for implicit shock tracking

In this section, we introduce matrix-based preconditioners tailored for the HOIST linearized system in (6.14). These preconditioners are derived from *constrained preconditioners*, commonly employed for linear systems encountered in constrained optimization. We close the section with a summary of all preconditioners proposed and studied in this work. We are interested in efficient preconditioners that do not require formation of $\boldsymbol{B}_{\boldsymbol{uu}}$ or involve the inverse of $\boldsymbol{J}_{\boldsymbol{u}}$ or $\boldsymbol{B}_{\boldsymbol{yy}}$; however, we consider a suite of preconditioners to study what is lost by these requirements.

Constrained preconditioners

The system matrix, which must be solved at every iteration of the HOIST method, repeated here for reference

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{B} & \boldsymbol{J}^T \\ \boldsymbol{J} & \boldsymbol{0} \end{pmatrix}, \tag{6.36}$$

is a symmetric saddle-point matrix. Typically, matrices of this type are known to suffer from bad condition numbers and there exists a wide variety of preconditioners tailored to the specific scenarios where they arise (Benzi et al., 2005). In the realm of constrained optimization, where these saddle-point systems naturally emerge from first-order optimality conditions, a class of popular preconditioners known as *constrained preconditioners*, denoted $\tilde{\boldsymbol{A}}_C$, is commonly used

$$\tilde{\boldsymbol{A}}_C = \begin{pmatrix} \tilde{\boldsymbol{B}} & \tilde{\boldsymbol{J}}^T \\ \tilde{\boldsymbol{J}} & 0 \end{pmatrix}. \tag{6.37}$$

Here, $\tilde{\boldsymbol{B}} \approx \boldsymbol{B}$ and $\tilde{\boldsymbol{J}} \approx \boldsymbol{J}$ are approximations to the Hessian and constraint matrices. If $\tilde{\boldsymbol{J}} = \boldsymbol{J}$, the preconditioner is the coefficient matrix for a modified saddle-point problem with the same linearized constraint. Furthermore, $\tilde{\boldsymbol{B}}$ and $\tilde{\boldsymbol{J}}$ are generally chosen as such that $\tilde{\boldsymbol{A}}_C$ and $\tilde{\boldsymbol{B}}$ are invertible and $\tilde{\boldsymbol{B}}^{-1}$, $\tilde{\boldsymbol{J}}\tilde{\boldsymbol{B}}^{-1}\tilde{\boldsymbol{J}}^T$ are easy to compute. In this case, the inverse of $\tilde{\boldsymbol{A}}_C$ can be explicitly computed as

$$\begin{pmatrix} \tilde{\boldsymbol{B}} & \tilde{\boldsymbol{J}}^T \\ \tilde{\boldsymbol{J}} & 0 \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{I} & -\tilde{\boldsymbol{B}}^{-1}\tilde{\boldsymbol{J}}^T \\ 0 & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{B}}^{-1} & 0 \\ 0 & -(\tilde{\boldsymbol{J}}\tilde{\boldsymbol{B}}^{-1}\tilde{\boldsymbol{J}}^T)^{-1} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & 0 \\ -\tilde{\boldsymbol{J}}\tilde{\boldsymbol{B}}^{-1} & \boldsymbol{I} \end{pmatrix}. \tag{6.38}$$

However, we consider more restrictive approximations because of our desire to avoid formation of $\boldsymbol{B}_{\boldsymbol{uu}}$ and inverses of $\boldsymbol{J}$ and $\boldsymbol{B}_{\boldsymbol{yy}}$.

Block anti-triangular constrained preconditioner

We propose a class of preconditioners for the HOIST linearized system with

$$\tilde{\boldsymbol{B}} = \begin{pmatrix} 0 & 0 \\ 0 & \tilde{\boldsymbol{B}}_{\boldsymbol{yy}} \end{pmatrix}, \qquad \tilde{\boldsymbol{J}} = \begin{pmatrix} \tilde{\boldsymbol{J}}_{\boldsymbol{u}} & \boldsymbol{J}_{\boldsymbol{y}} \end{pmatrix}, \tag{6.39}$$

where $\tilde{\boldsymbol{B}}_{\boldsymbol{yy}}$ is an approximation to $\boldsymbol{B}_{\boldsymbol{yy}}$ and $\tilde{\boldsymbol{J}}_{\boldsymbol{u}}$ is an approximation to $\boldsymbol{J}_{\boldsymbol{u}}$. Substitution of these choices into the constrained preconditioner leads to a lower block anti-triangular matrix, denoted $\tilde{\boldsymbol{A}}_{\text{AT}}$,

$$\tilde{\boldsymbol{A}}_{\text{AT}} = \begin{pmatrix} 0 & 0 & \tilde{\boldsymbol{J}}_{\boldsymbol{u}}^T \\ 0 & \tilde{\boldsymbol{B}}_{\boldsymbol{yy}} & \boldsymbol{J}_{\boldsymbol{y}}^T \\ \tilde{\boldsymbol{J}}_{\boldsymbol{u}} & \boldsymbol{J}_{\boldsymbol{y}} & 0 \end{pmatrix} \tag{6.40}$$

that will be referred to as the *approximate block anti-triangular constrained preconditioner* in the remainder. The inverse of $\tilde{\boldsymbol{A}}_{\text{AT}}$ is

$$\tilde{\boldsymbol{A}}_{\text{AT}}^{-1} = \begin{pmatrix} \tilde{\boldsymbol{C}}\tilde{\boldsymbol{B}}_{\boldsymbol{yy}}^{-1}\tilde{\boldsymbol{C}}^T & \tilde{\boldsymbol{C}}\tilde{\boldsymbol{B}}_{\boldsymbol{yy}}^{-1} & \tilde{\boldsymbol{J}}_{\boldsymbol{u}}^{-1} \\ \tilde{\boldsymbol{B}}_{\boldsymbol{yy}}^{-1}\tilde{\boldsymbol{C}}^T & \tilde{\boldsymbol{B}}_{\boldsymbol{yy}}^{-1} & 0 \\ \tilde{\boldsymbol{J}}_{\boldsymbol{u}}^{-T} & 0 & 0 \end{pmatrix}, \tag{6.41}$$

where $\tilde{\boldsymbol{C}} := -\tilde{\boldsymbol{J}}_{\boldsymbol{u}}^{-1}\boldsymbol{J}_{\boldsymbol{y}}$. Furthermore, the action of $\tilde{\boldsymbol{A}}_{\text{AT}}^{-1}$ on a vector $\boldsymbol{v} = \begin{pmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \boldsymbol{v}_3 \end{pmatrix}^T$, is

$$\tilde{\boldsymbol{A}}_{\text{AT}}^{-1} \begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \boldsymbol{v}_3 \end{pmatrix} = \begin{pmatrix} \tilde{\boldsymbol{J}}_{\boldsymbol{u}}^{-1}(-\boldsymbol{J}_{\boldsymbol{y}}\tilde{\boldsymbol{B}}_{\boldsymbol{yy}}^{-1}(-\boldsymbol{J}_{\boldsymbol{y}}^T\tilde{\boldsymbol{J}}_{\boldsymbol{u}}^{-T}\boldsymbol{v}_1 + \boldsymbol{v}_2) + \boldsymbol{v}_3) \\ \tilde{\boldsymbol{B}}_{\boldsymbol{yy}}^{-1}(-\boldsymbol{J}_{\boldsymbol{y}}^T\tilde{\boldsymbol{J}}_{\boldsymbol{u}}^{-T}\boldsymbol{v}_1 + \boldsymbol{v}_2) \\ \tilde{\boldsymbol{J}}_{\boldsymbol{u}}^{-T}\boldsymbol{v}_1 \end{pmatrix}. \tag{6.42}$$

The current arrangement of the $\tilde{\boldsymbol{A}}_{\text{AT}}^{-1}\boldsymbol{v}$ shows the following sequence of operation is required to compute the product: 1) a linear solve of the form $\tilde{\boldsymbol{J}}_{\boldsymbol{u}}^T\boldsymbol{w}_1 = \boldsymbol{v}_1$, 2) matrix product of the form $\tilde{\boldsymbol{w}}_2 = \boldsymbol{J}_{\boldsymbol{y}}^T\boldsymbol{w}_1$, 3) a linear solve of the form $\tilde{\boldsymbol{B}}_{\boldsymbol{yy}}\boldsymbol{w}_2 = -\tilde{\boldsymbol{w}}_2 + \boldsymbol{v}_2$, 4) matrix product of the form $\tilde{\boldsymbol{w}}_3 = \boldsymbol{J}_{\boldsymbol{y}}\boldsymbol{w}_2$, and 5) a linear solve of the form $\tilde{\boldsymbol{J}}_{\boldsymbol{u}}\boldsymbol{w}_3 = -\tilde{\boldsymbol{w}}_3 + \boldsymbol{v}_3$. Hence, the product $\tilde{\boldsymbol{A}}_{\text{AT}}^{-1}\boldsymbol{v}$ requires three linear system solves with the matrices $\tilde{\boldsymbol{J}}_{\boldsymbol{u}}$, $\tilde{\boldsymbol{B}}_{\boldsymbol{yy}}$, and $\tilde{\boldsymbol{J}}_{\boldsymbol{u}}^T$, and two matrix products with $\boldsymbol{J}_{\boldsymbol{y}}$ and $\boldsymbol{J}_{\boldsymbol{y}}^T$. Thus, preconditioners of this form entirely circumvent the need to form $\boldsymbol{B}_{\boldsymbol{uu}}$ or invert $\boldsymbol{J}_{\boldsymbol{u}}^T\boldsymbol{J}_{\boldsymbol{u}}$, $\boldsymbol{J}_{\boldsymbol{y}}^T\boldsymbol{J}_{\boldsymbol{y}}$, $\boldsymbol{B}_{\boldsymbol{uu}}$.

Considered preconditioners

The effectivity and cost of the anti-triangular constrained preconditioner are determined by the approximations $\tilde{\boldsymbol{J}}_{\boldsymbol{u}}$ and $\tilde{\boldsymbol{B}}_{\boldsymbol{yy}}$. In this work, we consider three choices of $\tilde{\boldsymbol{J}}_{\boldsymbol{u}}$, including the standard DG preconditioners: (1) $\tilde{\boldsymbol{J}}_{\boldsymbol{u}} = \boldsymbol{J}_{\boldsymbol{u}}$, (2) $\tilde{\boldsymbol{J}}_{\boldsymbol{u}} = \tilde{\boldsymbol{J}}_{\text{BJ}}$, and (3) $\tilde{\boldsymbol{J}}_{\boldsymbol{u}} = \tilde{\boldsymbol{J}}_{\text{BILU}}$. We also study three choices for $\tilde{\boldsymbol{B}}_{\boldsymbol{yy}}$, including standard preconditioners for general sparse matrices: (1) $\tilde{\boldsymbol{B}}_{\boldsymbol{yy}} = \boldsymbol{B}_{\boldsymbol{yy}}$, (2) $\tilde{\boldsymbol{B}}_{\boldsymbol{yy}} = \text{diag}(\boldsymbol{B}_{\boldsymbol{yy}})$ (point Jacobi), and (3) $\tilde{\boldsymbol{B}}_{\boldsymbol{yy}} = \text{ilu}(\boldsymbol{B}_{\boldsymbol{yy}})$ (point ILU0), where diag$(\cdot)$ extracts the diagonal and ilu$(\cdot)$ is the ILU0 factorization. The combinations of these choices studied in this work are summarized in Table 6.2, including preconditioners combined with P-multigrid (described in the following sub-section). The preconditioner $\boldsymbol{A}_0$ that uses $\tilde{\boldsymbol{J}}_{\boldsymbol{u}} = \boldsymbol{J}_{\boldsymbol{u}}$ and $\tilde{\boldsymbol{B}}_{\boldsymbol{yy}} = \boldsymbol{B}_{\boldsymbol{yy}}$ is not practical as the action of the preconditioner inverse to a vector will involve linear solves with $\boldsymbol{J}_{\boldsymbol{u}}$, $\boldsymbol{J}_{\boldsymbol{u}}^T$, and $\boldsymbol{B}_{\boldsymbol{yy}}$; however, it is included in our study as a benchmark for comparison, representing a best-case scenario in terms of iterative solver iterations.

P-Multigrid coarse scale corrections

In the context of the DG method, several studies have utilized P-multigrid techniques. These techniques are employed either as stand-alone methods to iteratively solve the linear system $\boldsymbol{As} = \boldsymbol{b}$ (Fidkowski et al., 2005) or as preconditioners for iterative solvers like GMRES (Persson and Peraire, 2008). The term *P-multigrid* refers to a multi-level approach combined with a smoother $\tilde{\boldsymbol{A}}$ where the high-order linear system (for instance, when $P > 2$) and the current iterate $\boldsymbol{s}$ are projected onto spaces of lower polynomial order. On the fine levels, the solution is smoothed using an operation of the form $\boldsymbol{s} \leftarrow \boldsymbol{s} + \tilde{\boldsymbol{A}}^{-1}(\boldsymbol{b} - \boldsymbol{As})$. On the coarsest level (typically with $P = 0$ or $P = 1$), the linear system is solved exactly.

Following Persson and Peraire (2008), we employ a two-level P-multigrid strategy for the linearized HOIST system $\boldsymbol{As} = \boldsymbol{b}$. In this approach, we first restrict the state variables $(\boldsymbol{u}, \boldsymbol{y})$ to the coarse scale. Specifically, $\boldsymbol{u}$ will be restricted to a piecewise constant solution ($P = 0$), and $\boldsymbol{y}$ is constrained to a mesh with straight-sided elements ($P_c = 1$). Upon returning to the finer level through prolongation, a smoothing operation $\tilde{\boldsymbol{A}}$ is applied; for this, one of the preconditioners outlined in Section 6.2.3 is utilized. This entire process is interpreted as an operator $\tilde{\boldsymbol{A}}_{p0}^{-1}$ approximating $\boldsymbol{A}^{-1}$ and is employed as a preconditioner for a Krylov solver.

The prolongation process involves utilizing a linear operator $\boldsymbol{P}$ represented as

$$\boldsymbol{P} = \begin{pmatrix} \boldsymbol{P}_{\boldsymbol{u}} & 0 & 0 \\ 0 & \boldsymbol{P}_{\boldsymbol{y}} & 0 \\ 0 & 0 & \boldsymbol{P}_{\boldsymbol{u}} \end{pmatrix} \tag{6.43}$$

which transfers a solution from the coarse level $\tilde{\boldsymbol{s}}^{(0)}$ to the fine level $\tilde{\boldsymbol{s}}$ via $\tilde{\boldsymbol{s}} = \boldsymbol{P}\tilde{\boldsymbol{s}}^{(0)}$. Here $\boldsymbol{P}_{\boldsymbol{u}}$ represents the prolongation operator for both the DG coefficients $\boldsymbol{u}$ and the Lagrange multiplier $\boldsymbol{\lambda}$. Detailed information about its construction can be found in the work of Fidkowski et al. (2005). Additionally, the prolongation $\boldsymbol{P}_{\boldsymbol{y}}$ for the mesh $\boldsymbol{y}$ involves inserting high-order nodes into the linear elements, as illustrated in Figure 6.5. Similarly, a linear restriction operator $\boldsymbol{Q}$ is applied, defined as

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{P}_{\boldsymbol{u}}^T & 0 & 0 \\ 0 & \boldsymbol{Q}_{\boldsymbol{y}} & 0 \\ 0 & 0 & \boldsymbol{P}_{\boldsymbol{u}}^T \end{pmatrix}. \tag{6.44}$$

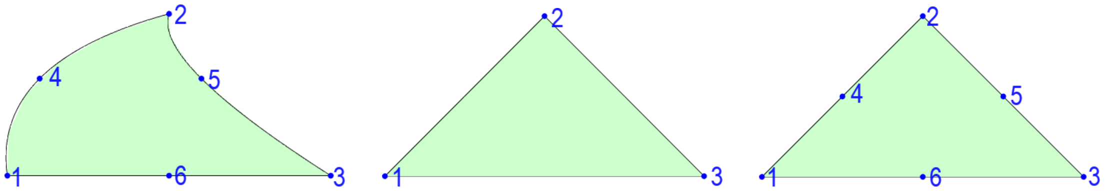


Figure 6.5: Example of mesh restriction/prolongation for a second order mesh ($P_c = 2$) with one element (*left*). The original element is restricted to $P_c = 1$ (*middle*) removing the high order nodes $4, 5, 6$. Prolongation (*right*) is performed by inserting high order nodes interpolating the low order element.

This operator projects a fine level solution $\tilde{\boldsymbol{s}}$ to the coarse level $\tilde{\boldsymbol{s}}^{(0)}$ via $\tilde{\boldsymbol{s}}^{(0)} = \boldsymbol{Q}\tilde{\boldsymbol{s}}$. Here $\boldsymbol{P}_{\boldsymbol{u}}^T$ is used for the restriction of both the DG coefficients $\boldsymbol{u}$ and the Lagrange multiplier $\boldsymbol{\lambda}$. Furthermore, for the mesh variables $\boldsymbol{y}$, a distinct restriction operator $\boldsymbol{Q}_{\boldsymbol{y}}$ is employed, which effectively functions as a selection operator, eliminating all high-order nodes ($P_c > 1$). For a single element, this process is depicted in Figure 6.5.

Algorithm 3 Two level P-multigrid

```
Input: KKT matrix $\boldsymbol{A}$, right-hand-side $\boldsymbol{b}$, precomputed prolongation operator $\boldsymbol{P}$, restriction
  operator $\boldsymbol{Q}$, smoother $\tilde{\boldsymbol{A}}$, and coarse matrix $\boldsymbol{A}^{(0)} = \boldsymbol{QAP}$
Output: approximate solution $\tilde{\boldsymbol{s}}$ to $\boldsymbol{As} = \boldsymbol{b}$
  Restrict right-hand side: $\boldsymbol{b}^{(0)} = \boldsymbol{Qb}$
  Solve coarse problem: $\boldsymbol{A}^{(0)}\tilde{\boldsymbol{s}}^{(0)} = \boldsymbol{b}^{(0)}$
  Prolongate solution to fine level: $\tilde{\boldsymbol{s}} = \boldsymbol{P}\tilde{\boldsymbol{s}}^{(0)}$
  Apply smoother: $\tilde{\boldsymbol{s}} = \tilde{\boldsymbol{s}} + \tilde{\boldsymbol{A}}^{-1}(\boldsymbol{b} - \boldsymbol{A}\tilde{\boldsymbol{s}})$
```

The entire algorithm (Algorithm 3) is described as follows: given the coarse matrix $\boldsymbol{A}^{(0)} := \boldsymbol{QAP}$, written as

$$\boldsymbol{A}^{(0)} := \boldsymbol{QAP} = \begin{pmatrix} \boldsymbol{P}_{\boldsymbol{u}}^T \boldsymbol{B}_{\boldsymbol{uu}} \boldsymbol{P}_{\boldsymbol{u}} & \boldsymbol{P}_{\boldsymbol{u}}^T \boldsymbol{B}_{\boldsymbol{uy}} \boldsymbol{P}_{\boldsymbol{y}} & \boldsymbol{P}_{\boldsymbol{u}}^T \boldsymbol{J}_{\boldsymbol{u}}^T \boldsymbol{P}_{\boldsymbol{u}} \\ \boldsymbol{Q}_{\boldsymbol{y}} \boldsymbol{B}_{\boldsymbol{uy}}^T \boldsymbol{P}_{\boldsymbol{u}} & \boldsymbol{Q}_{\boldsymbol{y}} \boldsymbol{B}_{\boldsymbol{yy}} \boldsymbol{P}_{\boldsymbol{y}} & \boldsymbol{Q}_{\boldsymbol{y}} \boldsymbol{J}_{\boldsymbol{y}}^T \boldsymbol{P}_{\boldsymbol{u}} \\ \boldsymbol{P}_{\boldsymbol{u}}^T \boldsymbol{J}_{\boldsymbol{u}} \boldsymbol{P}_{\boldsymbol{u}} & \boldsymbol{P}_{\boldsymbol{u}}^T \boldsymbol{J}_{\boldsymbol{y}} \boldsymbol{P}_{\boldsymbol{y}} & 0 \end{pmatrix}, \tag{6.45}$$

the right-hand-side is restricted to the coarse level: $\boldsymbol{b}^{(0)} = \boldsymbol{Qb}$. Then, the coarse problem is solved: $\boldsymbol{A}^{(0)}\tilde{\boldsymbol{s}}^{(0)} = \boldsymbol{b}^{(0)}$ using a direct sparse solve. Subsequently, the solution is prolonged back to the fine level as $\tilde{\boldsymbol{s}} = \boldsymbol{P}\tilde{\boldsymbol{s}}^{(0)}$ and an iterative smoothing process is applied as $\tilde{\boldsymbol{s}} = \tilde{\boldsymbol{s}} + \tilde{\boldsymbol{A}}^{-1}(\boldsymbol{b} - \boldsymbol{A}\tilde{\boldsymbol{s}})$.

We close this section by summarizing all eight preconditioners that will be studied in Section 6.3 in Table 6.2. As we did not observe a significant benefit in preliminary studies, we do not combine P-multigrid with the preconditioners where $\tilde{\boldsymbol{B}}_{\boldsymbol{yy}} = \mathrm{ilu}(\boldsymbol{B}_{\boldsymbol{yy}})$.

6.3 Numerical experiments

In this section, we present a series of numerical experiments designed to evaluate the performance of the introduced preconditioners. First, we define the metrics employed to measure their effectiveness (Section 6.3.1) and describe two shock-dominated flow benchmarks (Euler equations) that will be used to study the preconditioners (Section 6.3.2). Second, we present

Table 6.2: Summary of all HOIST preconditioners studied.

Preconditioner	$\tilde{\boldsymbol{B}}_{yy} \approx \boldsymbol{B}_{yy}$	$\tilde{\boldsymbol{J}}_{u} \approx \boldsymbol{J}_{u}$	P-multigrid
$\tilde{\boldsymbol{A}}_0$	$\boldsymbol{B}_{yy}$	$\boldsymbol{J}_{u}$	no
$\tilde{\boldsymbol{A}}_{\mathrm{BJ}}$	$\mathrm{diag}(\boldsymbol{B}_{yy})$	$\tilde{\boldsymbol{J}}_{\mathrm{BJ}}$	no
$\tilde{\boldsymbol{A}}_{\mathrm{BILU}}$	$\mathrm{diag}(\boldsymbol{B}_{yy})$	$\tilde{\boldsymbol{J}}_{\mathrm{BILU}}$	no
$\tilde{\boldsymbol{A}}_{\mathrm{BJ}\,+\,\mathrm{ilu}(\boldsymbol{B}_{yy})}$	$\mathrm{ilu}(\boldsymbol{B}_{yy})$	$\tilde{\boldsymbol{J}}_{\mathrm{BJ}}$	no
$\tilde{\boldsymbol{A}}_{\mathrm{BILU}\,+\,\mathrm{ilu}(\boldsymbol{B}_{yy})}$	$\mathrm{ilu}(\boldsymbol{B}_{yy})$	$\tilde{\boldsymbol{J}}_{\mathrm{BILU}}$	no
$\tilde{\boldsymbol{A}}_{\mathrm{0p0}}$	$\boldsymbol{B}_{yy}$	$\boldsymbol{J}_{u}$	yes
$\tilde{\boldsymbol{A}}_{\mathrm{BJp0}}$	$\mathrm{diag}(\boldsymbol{B}_{yy})$	$\tilde{\boldsymbol{J}}_{\mathrm{BJ}}$	yes
$\tilde{\boldsymbol{A}}_{\mathrm{BILUp0}}$	$\mathrm{diag}(\boldsymbol{B}_{yy})$	$\tilde{\boldsymbol{J}}_{\mathrm{BILU}}$	yes

and analyze the results from numerical experiments, focusing on various key HOIST parameters (Sections 6.3.5-6.3.8). We solely consider the generalized minimum residual (GMRES) Krylov solver in all studies because the preconditioned system does not have special structure that would allow us to use a more specialized solver. Finally, we summarize and conclude the results (Section 6.3.9).

6.3.1 Testing metrics

We assess the performance of the preconditioners based on the number of GMRES iterations required to achieve a convergence criterion. In practical applications, this involves monitoring the relative residual norm of the preconditioned system and stopping at the first iteration where

$$\frac{\|\tilde{\boldsymbol{A}}^{-1}\boldsymbol{A}\boldsymbol{s} - \tilde{\boldsymbol{A}}^{-1}\boldsymbol{b}\|}{\|\tilde{\boldsymbol{A}}^{-1}\boldsymbol{b}\|} < \mathrm{tol}, \tag{6.46}$$

where $\mathrm{tol} > 0$ is a specified tolerance. It is important to note that this convergence criterion is preconditioner-dependent. To ensure a fair comparison, we opt for a convergence criterion based on the exact solution $\boldsymbol{s}_{\mathrm{ex}}$ satisfying $\boldsymbol{A}\boldsymbol{s}_{\mathrm{ex}} = \boldsymbol{b}$:

$$\frac{\|\boldsymbol{s}_{\mathrm{ex}} - \boldsymbol{s}\|}{\|\boldsymbol{s}_{\mathrm{ex}}\|} < \mathrm{tol}, \tag{6.47}$$

with $\mathrm{tol} = 10^{-3}$. We also set the maximal GMRES iterations to be 1000.

The parameter space influencing the effectiveness of preconditioners for the HOIST method is vast and multifaceted. It encompasses choices related to the equations, specific problem formulations, the number of elements $|\mathcal{K}_h|$ utilized, the polynomial degrees P and P_c, the state $\boldsymbol{z}_k$ around which the system is linearized, and finally, the selection of γ and κ, which significantly affect the condition number of the system. Studying all these dimensions collectively is infeasible. Consequently, we will conduct separate investigations to gauge the relative impact of each of these parameters.

Table 6.3: Legend for plots comparing (# GMRES Iterations) for each preconditioner

$\tilde{\boldsymbol{A}}_0$	$\tilde{\boldsymbol{A}}_{0\text{p}0}$	$\tilde{\boldsymbol{A}}_{\text{BILU}}$	$\tilde{\boldsymbol{A}}_{\text{BILU}p0}$	$\tilde{\boldsymbol{A}}_{\text{BJ}}$	$\tilde{\boldsymbol{A}}_{\text{BJ}p0}$	$\tilde{\boldsymbol{A}}_{\text{BILU + ilu}(\boldsymbol{B_{yy}})}$	$\tilde{\boldsymbol{A}}_{\text{BJ + ilu}(\boldsymbol{B_{yy}})}$
(···○···)	(–·●–·)	(–·□·–)	(–··■··–)	(–·○–·)	(—●—)	(·····▲·····)	(– –△– –)

6.3.2 Description of examined cases

In this work, we focus exclusively on experiments related to the steady, inviscid two-dimensional Euler equations (Section 3.4.3). Specifically, we consider two problems with unique solution features: supersonic flow around a cylinder and supersonic flow over a diamond-shaped obstacle in a tunnel.

Supersonic flow over a cylinder

In our first problem, we explore a supersonic flow (Mach 2) over a cylinder to demonstrate the preconditioners performance for problems with curved shocks.

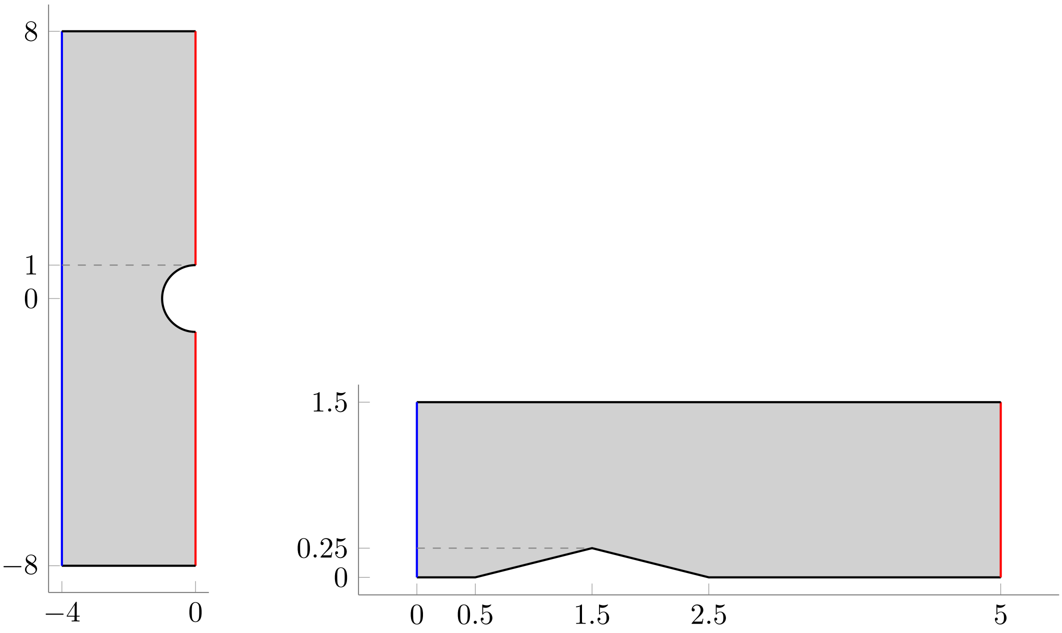


Figure 6.6: Geometry and boundary conditions for the `cylinder` (*left*) and `diamond` (*right*) test cases. Boundary conditions: slip walls (——), Mach 2 supersonic inflow (——), and supersonic outflow (——).

The domain (Figure 6.6, problem: `cylinder`) is discretized using a coarse unstructured triangular mesh with 90 elements and throughout most of our investigations, we utilize a third-order approximation for the flow variables and the geometry ($P = P_c = 2$), allowing the HOIST method to iterate to $k = 100$. A first-order finite volume solution is used for initialization of the method and it converges to a mesh that tracks the shock. Figure 6.7 displays the density for selected iterations $k = 1, 50, 100$ obtained for this configuration. For the upcoming studies, the corresponding linear systems derived from the states $\boldsymbol{z}_1, \boldsymbol{z}_{50}, \boldsymbol{z}_{100}$ are utilized to evaluate the performance of the preconditioners.

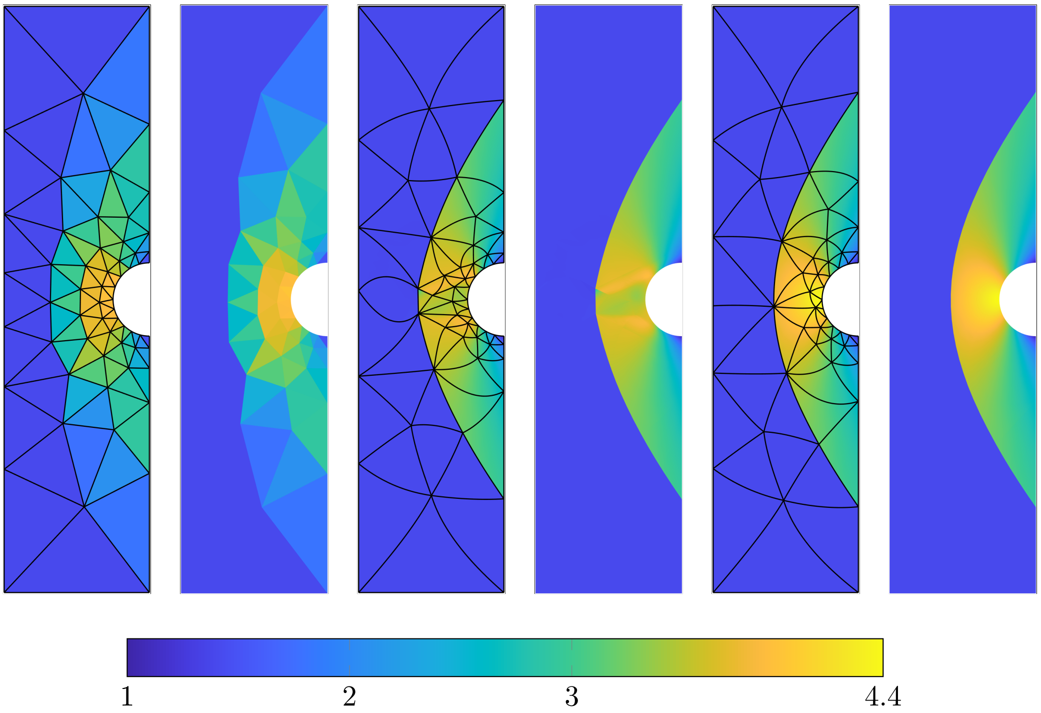


Figure 6.7: Selected $P = P_c = 2$ HOIST iterations $k \in \{1, 50, 100\}$ (*left-to-right*) for the `cylinder` test case (density) shown with and without mesh edges.

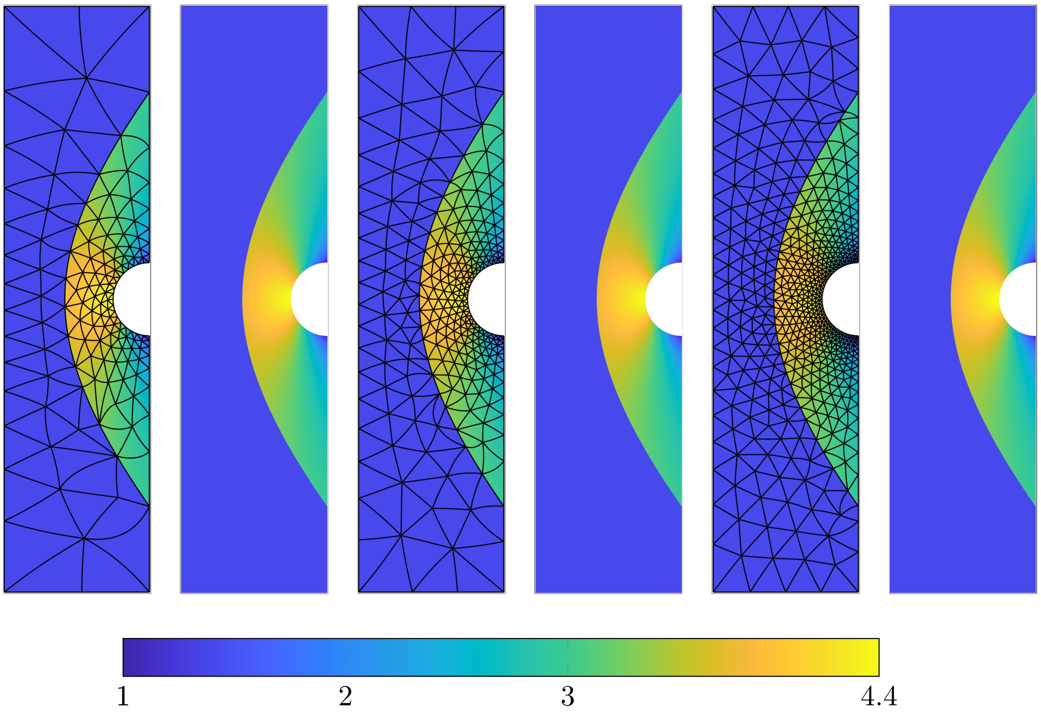


Figure 6.8: Selected $P = P_c = 2$ HOIST iterations $k = 100$ and different refinement levels $n_{\text{ref}} = 1, 2, 3$ (*left-to-right*) for the `cylinder` test case (density) shown with and without mesh edges.

Additionally, we investigate the impact of mesh refinement ($|\mathcal{K}_h|$) and the approximation orders (P and P_c) on the performance of the preconditioners. We apply the HOIST method to the bow shock problem, incorporating three additional refinement levels. Figure 6.8 illustrates the density of converged solutions ($k = 100$) and corresponding meshes resulting from this refinement process. To study the impact of polynomial degree, we first compute the HOIST solution on a grid with solution degree $P = 4$ and mesh degree $P_c = 4$. To obtain comparable lower order solutions, we restrict the $(P, P_c) = (4, 4)$ solution to degrees $(P, P_c) = (3, 3)$, $(P, P_c) = (2, 2)$, $(P, P_c) = (1, 1)$, and $(P, P_c) = (0, 1)$ (Figure 6.9). We opt for this approach over computing a HOIST solution at a given polynomial degree to avoid situations where the HOIST iterations do not converge to a tracked configuration due to insufficient resolution.

For all scenarios considered, the HOIST solver parameters (Huang and Zahr, 2022) are set as follows

- Adaptive regularization $(\gamma_0, \gamma_{\min}, \tau, \sigma_1, \sigma_2) = (10^{-2}, 10^{-2}, 10^{-1}, 10^{-2}, 10^{-1})$
- Adaptation of κ $(\kappa_0, \kappa_{\min}, \upsilon, \xi) = (1, 0, 1, 0.8)$
- Mesh operations $(c_1, c_2, c_3, c_4, c_4') = (0.025, 10^{-10}, 5, 0.025, 10^{-2})$
- Reinitialization procedure $(c_5, c_6, c_7, c_8) = (0.5, 10^{-2}, 0.5, 10^{-2})$.

Note, that for brevity, these parameters have not been explicitly introduced in this manuscript and their notation may overlap with notations used in other chapters. A complete description of all parameters and the overall algorithm can be found in the work of Huang and Zahr (2022).

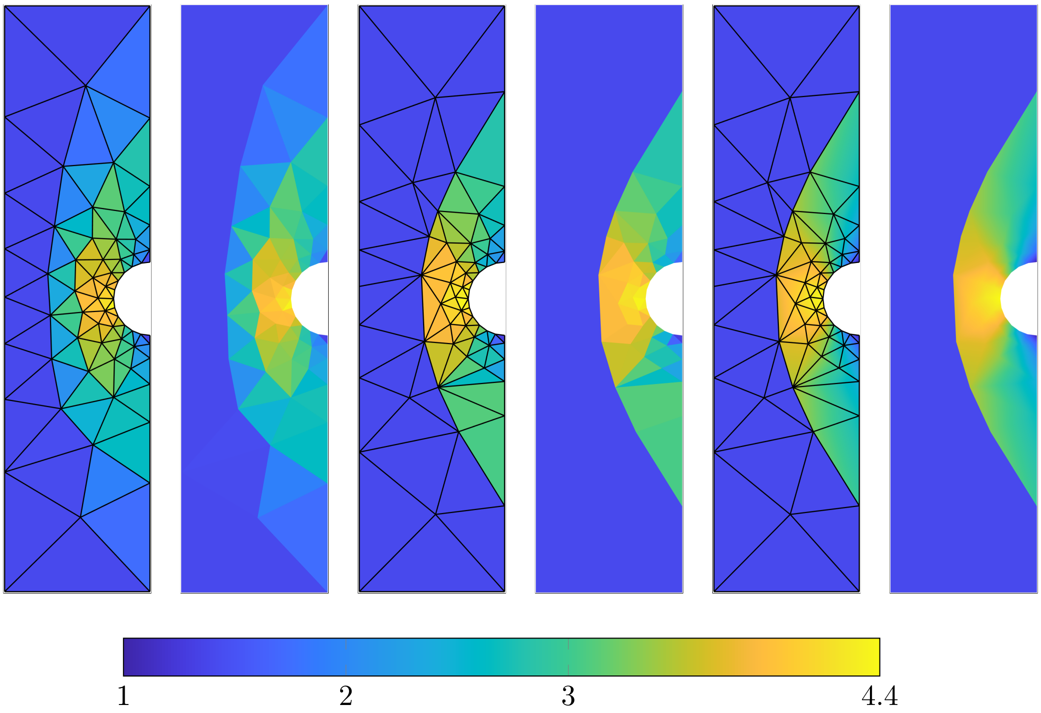


Figure 6.9: HOIST iterations ($P = P_c = 4$) projected to $P = 0$, $P_c = 1$ for $k = 1$ (*left*) and $k = 100$ (*middle*), $P = P_c = 1$ for $k = 100$ (*right*) for the `cylinder` test case (density) shown with and without mesh edges.

Supersonic flow over a two-dimensional diamond in a tunnel

Next, we study supersonic flow (Mach 2) passing over a two-dimensional diamond-shaped object within a tunnel (Figure 6.6). This test case, denoted as `diamond`, presents complex features such as reflecting, intersecting, and curved shocks. To discretize the domain, we generate an unstructured triangular mesh of 220 elements using DistMesh (Persson and Strang, 2004). Similar to previous cases, a third-order approximation is employed for both the geometry and flow variables ($P = P_c = 2$). Initializing the HOIST method with this unstructured mesh and the corresponding first-order finite volume solution, the method converges to a shock-aligned mesh that accurately represents all shocks and their intersections. Density plots of iterations at specific points in the optimization process ($k = 1, 150, 300$) are highlighted in Figure 6.10. The corresponding states $\boldsymbol{z}_1, \boldsymbol{z}_{150}, \boldsymbol{z}_{300}$ will be utilized for the subsequent analysis.

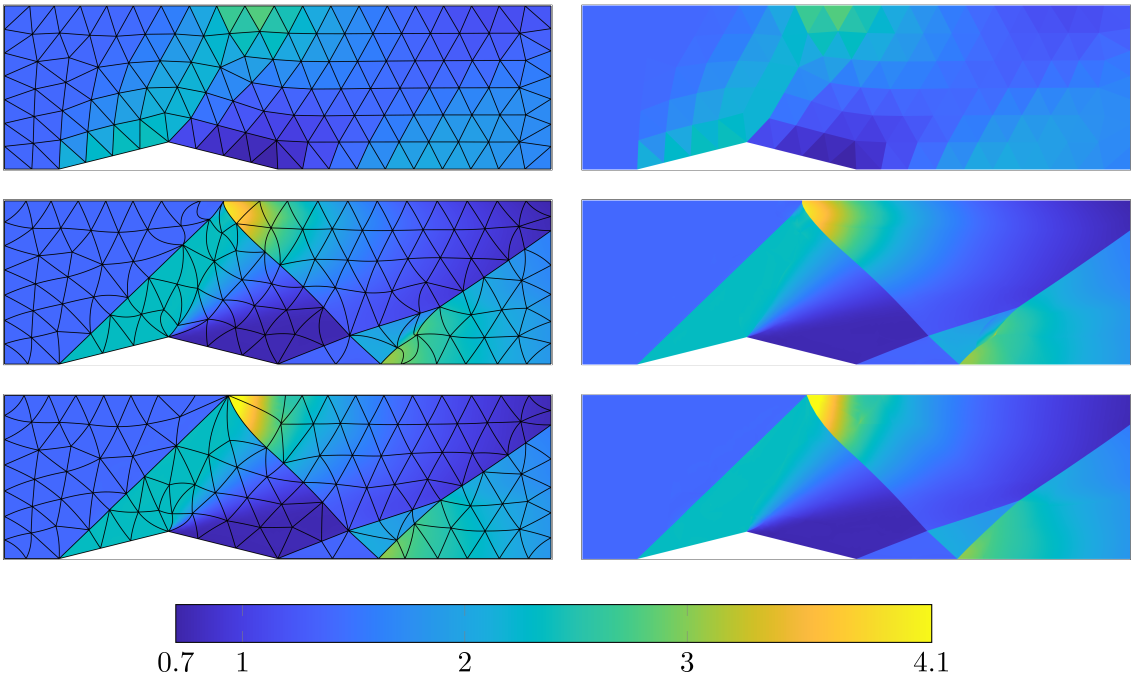


Figure 6.10: Selected $P = P_c = 2$ HOIST iterations $k \in \{1, 150, 300\}$ (*top-to-bottom*) for the `diamond` test case (density) with and without mesh edges.

For all scenarios considered, the HOIST solver parameters (Huang and Zahr, 2022) are set as follows

- Adaptive regularization parameters $(\gamma_0, \gamma_{\text{min}}, \tau, \sigma_1, \sigma_2) = (10^{-2}, 10^{-2}, 10^{-1}, 10^{-2}, 10^{-1})$
- Mesh quality adaptation parameters $(\kappa_0, \kappa_{\text{min}}, \upsilon, \xi) = (1, 0, 1, 0.8)$
- Mesh operation parameters $(c_1, c_2, c_3, c_4, c_4') = (0.25, 10^{-10}, 4, 0.3, 10^{-2})$
- Reinitialization parameters $(c_5, c_6, c_7, c_8) = (0.5, 10^{-2}, 0.5, 10^{-2})$.

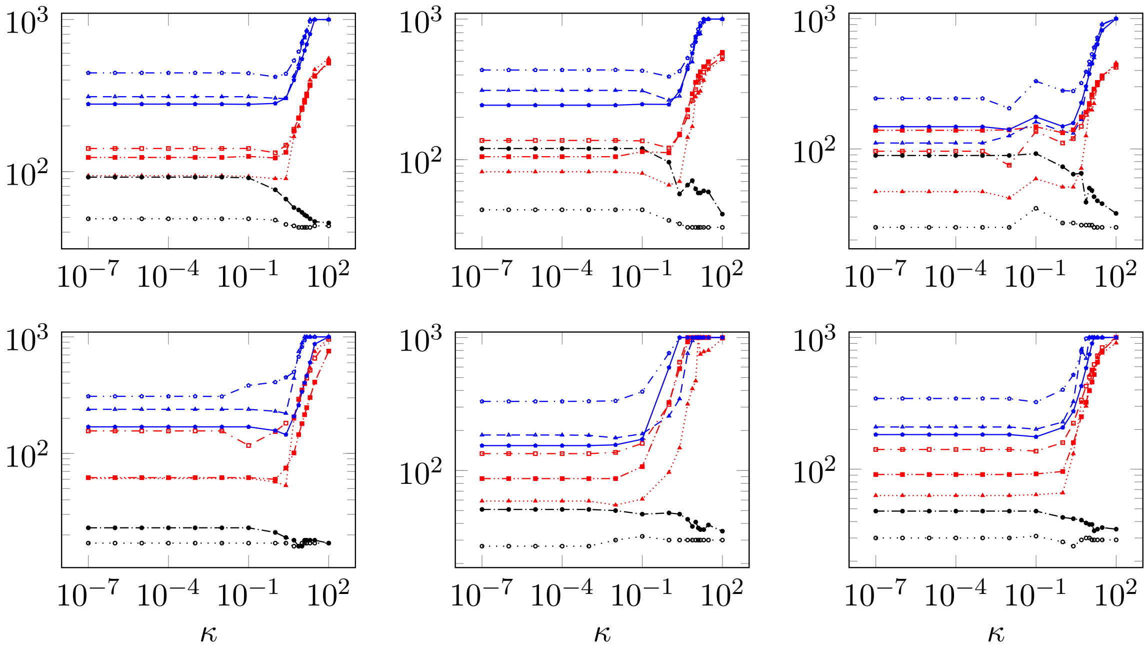


Figure 6.11: (# GMRES iterations) vs. mesh quality parameter κ for both test cases (*top*: `cylinder`, *bottom*: `diamond`) (legend in Table 6.3) for polynomial degree $P = P_c = 2$, regularization parameters $\gamma = 0.1$, and states z_k (`cylinder`: $k = 1, 50, 100$, `diamond`: $k = 1, 150, 300$) (*left-to-right*).

6.3.3 Influence of mesh quality parameter κ

In our first experiment, we aim to measure the impact of the choice of κ by considering both test cases (`cylinder` and `diamond` with $P = P_c = 2$) with the linear system formed from specific states $\boldsymbol{z}_k$ corresponding to different HOIST iterations ($k = 1, 50, 100$ for `cylinder` and $k = 1, 150, 300$ for `diamond`) with a single regularization parameter of $\gamma = 0.1$. For these six configuration, we compute the HOIST matrix for $\kappa = \{10^{-10}, 10^{-9}, \ldots, 10^2\}$ and measure the needed GMRES iterations for each preconditioner tested (Table 6.2) (results in Figure 6.11).

Analyzing the results shown in Figure 6.11, we make the following observations. Most preconditioners demonstrate significant deterioration after a critical κ, usually in the range $\kappa \in [0.1, 1]$. As κ increases beyond this range, the GMRES iterations of all preconditioners except $\tilde{\boldsymbol{A}}_0$ and $\tilde{\boldsymbol{A}}_{0\text{p}0}$ quickly increase, which suggests the approximations to $\boldsymbol{B}_{\boldsymbol{yy}}$ (the only block depending on κ) deteriorates as κ rises beyond the critical value. Fortunately, such large κ values rarely occur in the method, mitigating this sensitivity issue. Intriguingly, when $\kappa < 0.1$, no sensitivity was observed for any preconditioner so we fix $\kappa = 10^{-7}$ for the remaining studies in this work. Furthermore, we observe that as κ rises beyond 0.1, the benefit of P-multigrid diminishes.

For all values of κ, the expensive, best-case scenario $\tilde{\boldsymbol{A}}_0$ preconditioner outperforms all others across all states considered for both problems. It is also interesting to note that P-multigrid actually degrades the performance of the $\tilde{A}_0$, in some cases making it worse than preconditioners that use approximate inverses. For the $\tilde{\boldsymbol{A}}_{\text{BILU}}$ preconditioners, the addition of P-multigrid and especially the inclusion of $\text{ilu}(\boldsymbol{B}_{\boldsymbol{yy}})$ as an approximation to $\boldsymbol{B}_{\boldsymbol{yy}}$ significantly enhances its performance (in some cases, reducing the GMRES iterations by a factor of two or more). The $\tilde{\boldsymbol{A}}_{\text{BJ}}$ preconditioners also benefit from both P-multigrid and the inclusion of $\text{ilu}(\boldsymbol{B}_{\boldsymbol{yy}})$ as an approximation to $\boldsymbol{B_{yy}}$; however, in this case, P-multigrid provides the greater reduction in GMRES iterations. Finally, as expected, the $\tilde{\boldsymbol{A}}_{\mathrm{BILU}}$ outperform the $\tilde{\boldsymbol{A}}_{\mathrm{BJ}}$ preconditioners across test cases and states.

6.3.4 Influence of state z_k

In our second experiment, we investigate the dependence of the preconditioner performance on the linearization state $\boldsymbol{z}_k$. We fix the mesh quality parameter $\kappa = 10^{-7}$ and build six test cases from the two problems (`cylinder` and `diamond` with $P = P_c = 2$) and three choices for the regularization parameter $\gamma \in \{10^{-3}, 10^{-2}, 10^{-1}\}$. For each test case and preconditioner (Table 6.2), we record the number of GMRES iteration required to reach the convergence criteria (6.47) at every 5th HOIST iteration, i.e., $k \in \{1, 5, 10, 15, ..., 150\}$ for `cylinder` and $k \in \{1, 5, 10, 15, ..., 300\}$ for `diamond`(Figure 6.12).

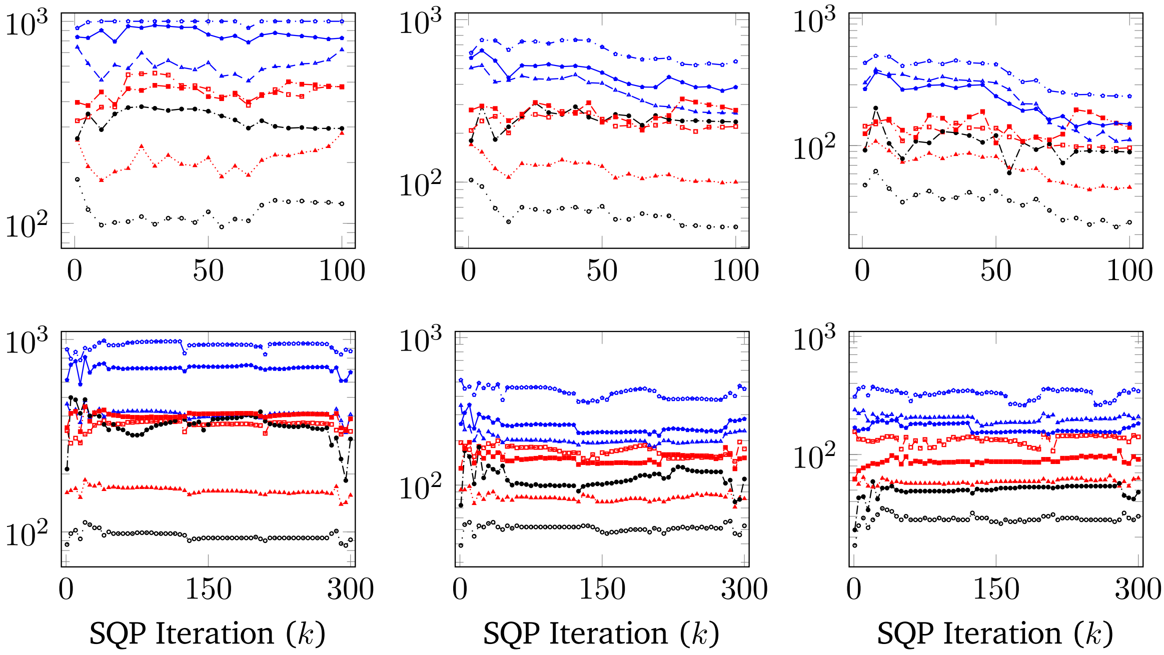


Figure 6.12: (# GMRES iterations) vs. state $\boldsymbol{z}_k$ for both test cases (*top*: `cylinder`, *bottom*: `diamond`) (legend in Table 6.3) for polynomial degree $P = P_c = 2$, mesh quality parameter $\kappa = 10^{-7}$, and different regularization parameters $\gamma = 10^{-3}, 10^{-2}, 10^{-1}$ (*left-to-right*).

Analyzing the results shown in Figure 6.12, we make the following observations. First, the linearization state has a modest impact on the GMRES iterations. In the `diamond` case, the iterations remain nearly constant, with minor fluctuations occurring due to abrupt state changes (e.g., solution reinitialization and element collapses). The `cylinder` case exhibits a more pronounced state dependency, particularly for larger γ, where the iteration count tends to decrease as the final state is approached. Once again, the $\tilde{A}_0$ demonstrates superior performance across all scenarios and $\tilde{\boldsymbol{A}}_{\mathrm{BILU}\,+\,\mathrm{ilu}(\boldsymbol{B_{yy}})}$ is the best practical preconditioner (i.e., not involving the expensive $\boldsymbol{J_u}$, $\boldsymbol{J}_{\boldsymbol{u}}^T$, and $\boldsymbol{B_{yy}}$ inverses). Again, the P-multigrid counterpart of $\tilde{\boldsymbol{A}}_0$,

$\tilde{\boldsymbol{A}}_{\text{0p0}}$, performs noticeably worse, in many cases requiring more iterations that $\tilde{\boldsymbol{A}}_{\text{BILU + ilu}(\boldsymbol{B_{yy}})}$ and often demonstrating similar performance to $\tilde{\boldsymbol{A}}_{\text{BILU}}$ and $\tilde{\boldsymbol{A}}_{\text{BILUp0}}$, despite the use of exact inverses. Unlike the previous study, there is no clear conclusion regarding $\tilde{\boldsymbol{A}}_{\text{BILU}}$ and $\tilde{\boldsymbol{A}}_{\text{BILUp0}}$. Finally, the addition of P-multigrid and the inclusion of ilu$(\boldsymbol{B_{yy}})$ as an approximation to $\boldsymbol{B_{yy}}$ enhance the performance of the $\tilde{\boldsymbol{A}}_{\text{BJ}}$ preconditioner with $\tilde{\boldsymbol{A}}_{\text{BJ + ilu}(\boldsymbol{B_{yy}})}$ holding a clear advantage for smaller values of γ.

6.3.5 Influence of regularization parameter γ

In this experiment, we study the influence of the Hessian regularization parameter γ on the performance of the preconditioners considered (Table 6.2). We build six test cases from the two problems (`cylinder` and `diamond` with $P = P_c = 2$) and three states ($k \in \{1, 50, 100\}$ for `cylinder` and $k \in \{1, 150, 300\}$ for `diamond`). Furthermore, we fix the mesh quality parameter at $\kappa = 10^{-7}$ and vary the regularization parameter $\gamma \in \{10^{-10}, 10^{-9}, \ldots, 10^{1}\}$. The resulting GMRES iterations needed to reach the convergence criteria (6.47) for each of these cases are shown in Figure 6.13.

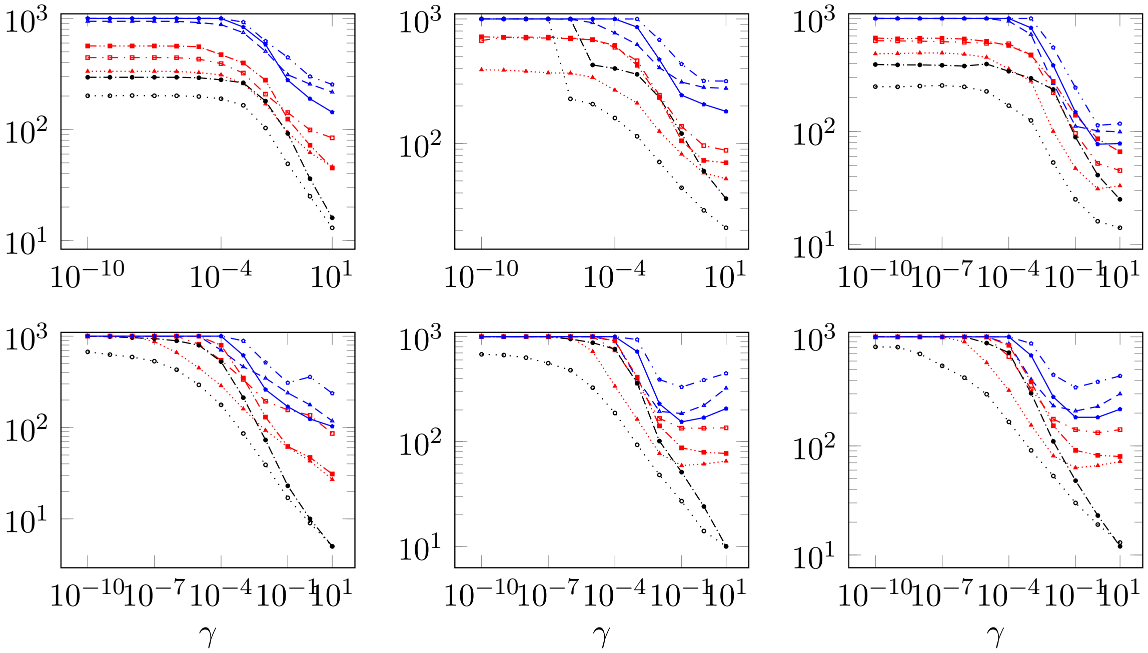


Figure 6.13: (# GMRES iterations) vs. regularization parameter γ for both test cases (*top*: `cylinder`, *bottom*: `diamond`) (legend in Table 6.3) for polynomial degree $P = P_c = 2$, mesh quality parameter $\kappa = 10^{-7}$, and different states $\boldsymbol{z}_k$ (`cylinder`: $k = 1, 50, 100$, `diamond`: $k = 1, 150, 300$) (*left-to-right*).

Analyzing the results shown in Figure 6.13, we make the following observations. Decreasing the parameter γ reduces the regularization applied to the matrix $\boldsymbol{B_{yy}}$, leaving the ill-conditioned (or singular) Gauss-Newton Hessian in the limit where $\gamma = 0$. As expected, this leads to a noticeable rise in the number of GMRES iterations, particularly evident in the case of $\tilde{\boldsymbol{A}}_0$ and $\tilde{\boldsymbol{A}}_{\text{0p0}}$. However, for the other preconditioners, we observe a relative indifference to variations for $\gamma \in [10^{-1}, 10^{1}]$, particularly in later SQP iterations ($k > 100$). This suggests that in this range, the loss in accuracy incurred by the approximations of $\boldsymbol{J_u}$ and $\boldsymbol{B_{yy}}$ dominates ill-conditioning effects.

The results suggest the presence of a problem-dependent threshold value for γ (`cylinder`: 10^{-5}, `diamond`: 10^{-9}). Below this threshold, the number of iterations ceases to increase significantly. This phenomenon is especially prominent in the case of `cylinder`, whereas for `diamond`, most preconditioners did not converge reaching the maximum number of iterations below $\gamma = 10^{-6}$. Additionally, these findings imply the possibility of establishing a lower limit for the minimum regularization parameter γ_{min} that should be set in the HOIST method. The results obtained for `diamond` suggest that γ_{min} should not be less than 10^{-4}, as the iteration counts become impractical beyond this threshold. Considering the observed increase in iteration numbers with higher polynomial degrees and finer meshes (discussed in upcoming subsections), setting a more conservative lower bound, for instance, $\gamma_{\text{min}} = 10^{-2}$, is advisable.

For the BJ-based preconditioners ($\tilde{\boldsymbol{A}}_{\text{BJ}}$, $\tilde{\boldsymbol{A}}_{\text{BJp0}}$, $\tilde{\boldsymbol{A}}_{\text{BJ + ilu}(\boldsymbol{B_{yy}})}$), the trends observed earlier remain evident: for large regularization parameters $\gamma \in [10^{-1}, 10^{1}]$, $\tilde{\boldsymbol{A}}_{\text{BJp0}}$ outperforms $\tilde{\boldsymbol{A}}_{\text{BJ + ilu}(\boldsymbol{B_{yy}})}$ while the opposite is true for $\gamma \leqslant 10^{-2}$. Both of these preconditioners perform favorly compared to $\tilde{\boldsymbol{A}}_{\text{BJ}}$. The scenario is slightly different for the BILU-based preconditioners ($\tilde{\boldsymbol{A}}_{\text{BILU}}$, $\tilde{\boldsymbol{A}}_{\text{BILUp0}}$, $\tilde{\boldsymbol{A}}_{\text{BILU + ilu}(\boldsymbol{B_{yy}})}$). In this case, $\tilde{\boldsymbol{A}}_{\text{BILU + ilu}(\boldsymbol{B_{yy}})}$ outperforms both $\tilde{\boldsymbol{A}}_{\text{BILU}}$ and $\tilde{\boldsymbol{A}}_{\text{BILUp0}}$ across all cases, with the performance gap between $\tilde{\boldsymbol{A}}_{\text{BILUp0}}$ and $\tilde{\boldsymbol{A}}_{\text{BILU + ilu}(\boldsymbol{B_{yy}})}$ widening for $\gamma \leqslant 10^{-2}$. The utilization of P-multigrid seems to add value only for $\gamma > 10^{-2}$, as $\tilde{\boldsymbol{A}}_{\text{BILU}}$ often exhibits similar or even better iteration counts than $\tilde{\boldsymbol{A}}_{\text{BILUp0}}$.

6.3.6 Influence of number of polynomial degrees (P, P_c)

In this experiment, we study the effect of the polynomial degree (P, P_c) on the GMRES iterations. We test each of our proposed preconditioners (Table 6.2) against six cases built from three states $\boldsymbol{z}_k$ for $k \in \{1, 50, 100\}$, two regularization parameters $\gamma \in \{10^{-3}, 10^{-1}\}$, and a fixed mesh quality parameter $\kappa = 10^{-7}$ for the `cylinder` problem. A $P = P_c = 4$ HOIST simulation is used to compute the initial states ($\boldsymbol{z}_k$ for $k = 1, 50, 100$), which are subsequently restricted to polynomial degrees $(P, P_c) \in \{(0,1), (1,1), (2,2), (3,3), (4,4)\}$. As discussed in Section 6.3.2, this approach is taken to yield a well-defined, systematic study and avoid HOIST convergence issues that can arise when the grid is sufficiently underresolved. The measured GMRES iterations required to achieve the convergence criteria (6.47) are depicted in Figure 6.14.

Analyzing the results shown in Figure 6.14, we make the following observations. First, increasing the polynomial degree on a fixed mesh results in a direct escalation of GMRES iterations for all preconditioners with more pronounced growth rate for the smaller regularization parameters $\gamma = 10^{-3}$. The P-multigrid versions of the BJ ($\tilde{\boldsymbol{A}}_{\text{BJp0}}$) and BILU ($\tilde{\boldsymbol{A}}_{\text{BILUp0}}$) preconditioners are sensitive to the polynomial degree as their iteration count approaches that of the original BJ ($\tilde{\boldsymbol{A}}_{\text{BJ}}$) and BILU ($\tilde{\boldsymbol{A}}_{\text{BILU}}$) preconditioner as the polynomial degree increases. Both the original and P-multigrid version of the BJ and BILU are outperformed by inclusion of $\text{ilu}(\boldsymbol{B_{yy}})$ as an approximation to $\boldsymbol{B_{yy}}$, where $\tilde{\boldsymbol{A}}_{\text{BJ + ilu}(\boldsymbol{B_{yy}})}$ is the most effective BJ preconditioner and $\tilde{\boldsymbol{A}}_{\text{BILU + ilu}(\boldsymbol{B_{yy}})}$ is the most effective BILU preconditioner. Furthermore, the $\tilde{\boldsymbol{A}}_{\text{BILU + ilu}(\boldsymbol{B_{yy}})}$ preconditioner is the most effective practical preconditioner, only being outperformed by the

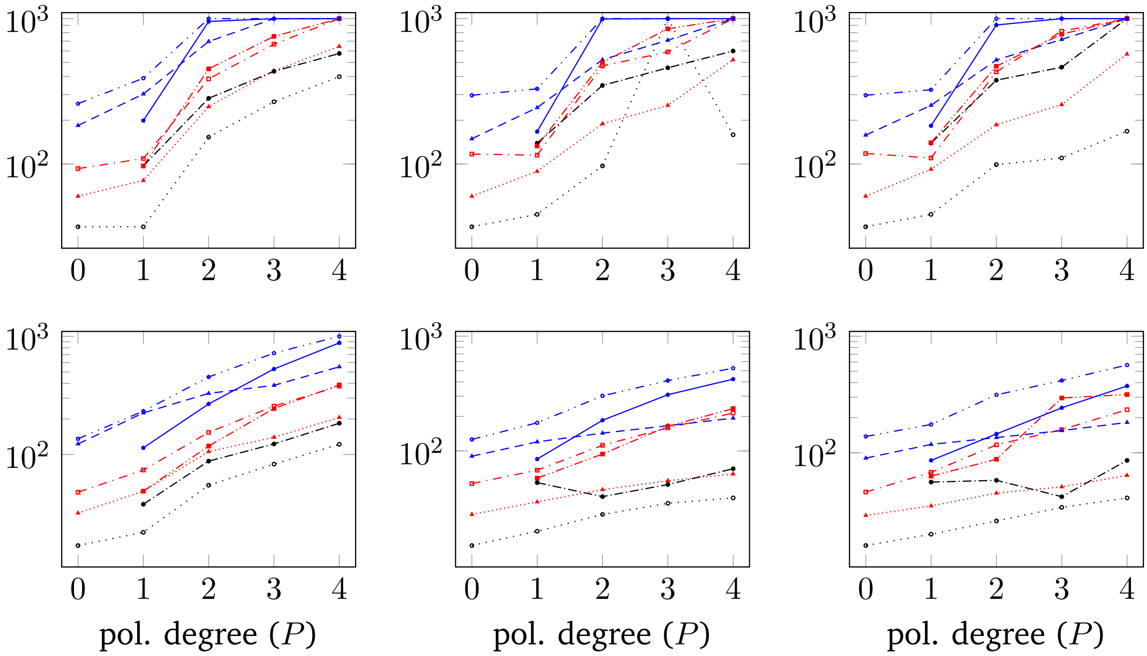


Figure 6.14: (# GMRES iterations) vs. polynomial degrees (P, P_c) for different regularization parameters $\gamma = 10^{-3}$ (*top*) and $\gamma = 10^{-1}$ (*bottom*) (legend in Table 6.3), mesh quality parameter $\kappa = 10^{-7}$, and different states $\boldsymbol{z}_k$, $(k = 1, 50, 100)$ for the `cylinder` problem. For $P = 0$, the coarse-scale updates from the P-multigrid preconditioners solve the problem directly, which only requires one GMRES iteration. These results are omitted for clarity.

best-case scenario $\tilde{\boldsymbol{A}}_0$ (and, in some cases, its P-multigrid variant). The $\tilde{\boldsymbol{A}}_{\text{BILU + ilu}(\boldsymbol{B}_{\boldsymbol{yy}})}$ preconditioner also exhibits the slowest iteration growth with polynomial degree, particularly for the larger regularization parameter $\gamma = 10^{-1}$.

6.3.7 Influence of number of mesh elements $|\mathcal{K}_h|$

In this experiment, we study the dependency of the GMRES iterations on the number of mesh elements. We test each of our proposed preconditioners (Table 6.2) against six cases built from three states $\boldsymbol{z}_k$ for $k \in \{1, 50, 100\}$, two regularization parameters $\gamma \in \{10^{-3}, 10^{-1}\}$, and a fixed mesh quality parameter $\kappa = 10^{-7}$ for the `cylinder` problem. For each of these cases, we consider four refinement levels (Figure 6.8) at fixed polynomial degree $P = P_c = 2$ with element count $|\mathcal{K}_h| \in \{70, 130, 260, 1000\}$. The measured GMRES iterations required to achieve the convergence criteria (6.47) are depicted in Figure 6.15.

Analyzing the results shown in Figure 6.15, we make the following observations. The exact preconditioner $\tilde{\boldsymbol{A}}_0$ demonstrates remarkable insensitivity to the number of elements. Its multigrid counterpart, $\tilde{\boldsymbol{A}}_{0\text{p}0}$, while less effective, exhibits a similar stable trend for the $\gamma = 10^{-1}$ case. In some cases, the GMRES iteration count slightly decreases as the number of elements rises. For the BJ preconditioners, $\tilde{\boldsymbol{A}}_{\text{BJ + ilu}(\boldsymbol{B}_{\boldsymbol{yy}})}$ is most effective for the smaller $\gamma = 10^{-3}$ (the other BJ variants often reach the maximum iterations without convergence), whereas $\tilde{\boldsymbol{A}}_{\text{BJp}0}$ is the most effective BJ preconditioner for $\gamma = 10^{-1}$ (although usually only slightly outperforms $\tilde{\boldsymbol{A}}_{\text{BJ + ilu}(\boldsymbol{B}_{\boldsymbol{yy}})}$). Similarly, for the smaller $\gamma = 10^{-3}$, the $\tilde{\boldsymbol{A}}_{\text{BILU + ilu}(\boldsymbol{B}_{\boldsymbol{yy}})}$ preconditioner is clearly

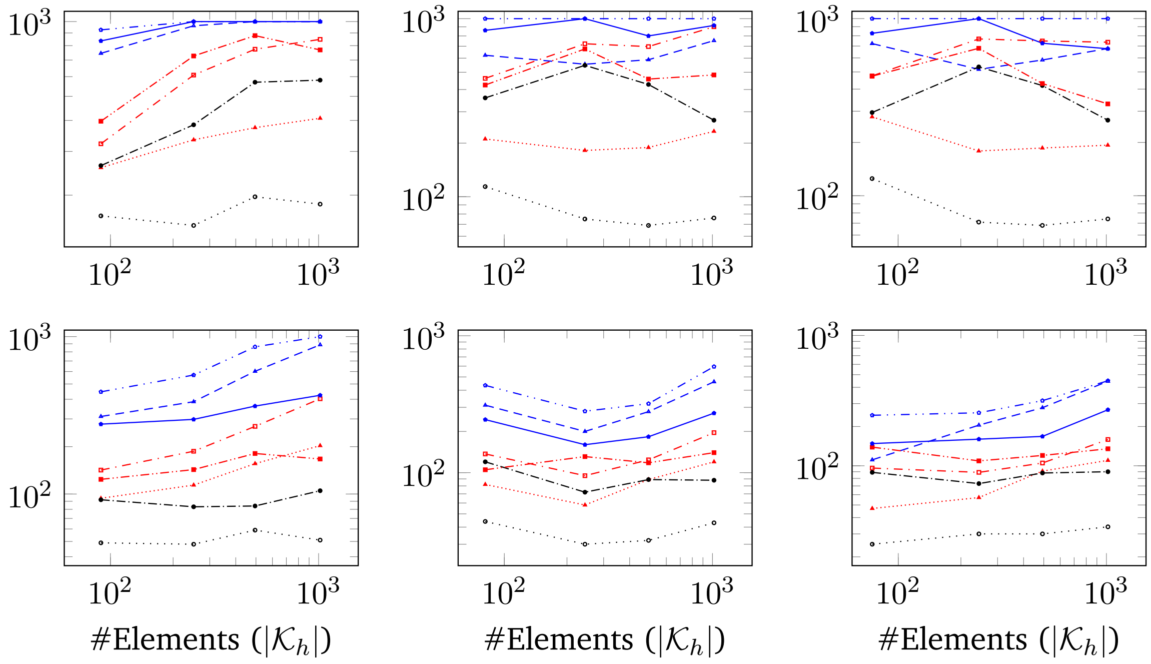


Figure 6.15: (# GMRES iterations) vs. (# Elements $|\mathcal{K}_h|$) for different regularization parameters $\gamma = 10^{-3}$ (*top*) and $\gamma = 10^{-1}$ (*bottom*) (legend in Table 6.3), polynomial degree $P = P_c = 2$, mesh quality parameter $\kappa = 10^{-7}$, and different states $\boldsymbol{z}_k$ (`cylinder`: $k = 1, 50, 100$, `diamond`: $k = 1, 150, 300$) (*left-to-right*).

superior to the other BILU variants and exhibits the slowest growth as the element count rises. The $\tilde{\boldsymbol{A}}_{\mathrm{BILU}\,+\,\mathrm{ilu}(\boldsymbol{B}_{\boldsymbol{yy}})}$ is usually the best BILU preconditioner for the larger $\gamma = 10^{-1}$, although the difference between the three BILU preconditioners is less dramatic for this scenario.

6.3.8 Comparison of preconditioners keeping original κ_k, γ_k

In our final experiment, we investigate GMRES iterations across the entire optimization history for both problems (`cylinder` and `diamond` with $P = P_c = 2$). For this experiment, we use the adaptive mesh quality κ_k and regularization parameters γ_k presented in the work by Huang and Zahr (2022), with adaptation parameters in Section 6.3.2. The measured GMRES iterations required to achieve the convergence criteria (6.47) for each state $\boldsymbol{z}_k$ ($k \in \{1, 2, \ldots, 100\}$ for `cylinder` and $k \in \{1, 2, \ldots, 300\}$ for `diamond`) encountered during the HOIST iterations are depicted in Figure 6.16. The evolution of γ_k and κ_k are also shown in this figure.

Analyzing the results shown in Figure 6.16, we observe the iteration count closely correlates to the γ value, as expected from Section 6.3.5, in that the GMRES iterations rise as γ decreases. However, extreme values of γ are not encountered during the adaptation, which avoids excessive GMRES iteration counts. Abrupt changes in GMRES iterations are associated with abrupt alterations in γ (e.g., in the `cylinder` case around $k = 50$, where γ is nearly equal to its initial value), which occur after elements are collapsed. Generally, larger γ values tend to benefit the P-multigrid preconditioners the most, granting them an advantage over their counterparts (though $\tilde{\boldsymbol{A}}_0$ is an exception due to the overall poor performance of $\tilde{\boldsymbol{A}}_{0\mathrm{p}0}$). For the BJ preconditioners, $\tilde{\boldsymbol{A}}_{\mathrm{BJp}0}$ consistently outperforms $\tilde{\boldsymbol{A}}_{\mathrm{BJ}}$ and is on par with $\tilde{\boldsymbol{A}}_{\mathrm{BJ}\,+\,\mathrm{ilu}(\boldsymbol{B}_{\boldsymbol{yy}})}$ for the `diamond` cases.

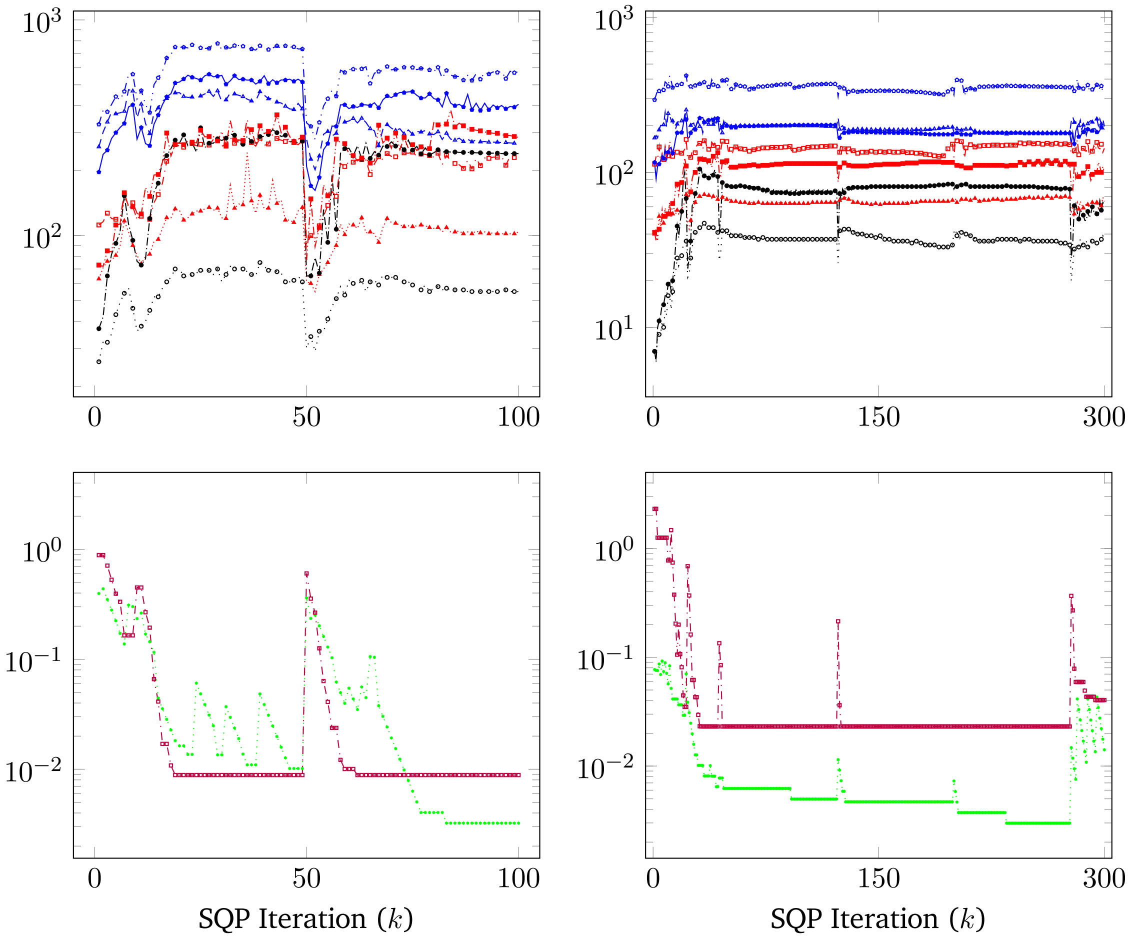


Figure 6.16: *Top*: (# GMRES iterations) vs. states $\boldsymbol{z}_k$ using adaptive mesh/regularization parameters κ_k, γ_k (legend in Table 6.3) for both test cases with $P = P_c = 2$ (*left*: `cylinder`, *right*: `diamond`). *Bottom*: History of the adaptive mesh quality parameter κ_k (···•···) and regularization parameter γ_k (-·▪·-).

For the `cylinder` problem, $\tilde{\boldsymbol{A}}_{\text{BILU + ilu}(\boldsymbol{B_{yy}})}$ performs better in the low γ regime. Among the BILU preconditioners, $\tilde{\boldsymbol{A}}_{\text{BILU + ilu}(\boldsymbol{B_{yy}})}$ consistently performs the best across all k, only matching $\tilde{\boldsymbol{A}}_{\text{BILUp0}}$ for high γ values. As expected from the previous sections, $\tilde{\boldsymbol{A}}_{\text{BILU + ilu}(\boldsymbol{B_{yy}})}$ is the most effective practical preconditioner as it is only consistently outperformed by the best-case (but impractical) $\tilde{\boldsymbol{A}}_0$, making it our preferred preconditioner.

6.3.9 Conclusion

We rigorously evaluated all preconditioners on two compressible inviscid flow problems by measuring the number of GMRES iterations needed to reach a set relative error norm. Specifically, we investigated two steady two-dimensional (2D) problems: a flow over a cylinder featuring a single bow shock and a flow over a diamond shaped obstacle in a tunnel featuring multiple straight-sided shocks, shock reflections and shock-shock interactions. The preconditioners were then tested at different stages of the optimization method, varying each of its key parameters.

Our findings showed that the regularization parameter γ significantly influences the GMRES iteration count, with the P-multigrid scheme being beneficial primarily under high regularization conditions. The iteration count was notably affected by the polynomial degree of the solution and the mesh in scenarios of low regularization. In contrast, the number of mesh elements and the mesh quality parameter κ had less impact on the GMRES iterations, with the latter being negligible for commonly used parameter-ranges. Our results particularly highlighted the effective performance of BILU-based preconditioners in various scenarios. The best and most reliable BILU variant used an ILU0 approximation to $\boldsymbol{B_{yy}}$. We also determined that the two-level P-multigrid scheme did not offer enough benefits to warrant its computational cost.

7 Conclusion

This dissertation has introduced significant advancements in the field of implicit shock tracking (IST) methods for compressible flows with shocks, through the development of a novel implicit extended discontinuous Galerkin (XDG) shock tracking method and the development of specialized preconditioners for IST linearized optimality systems. This concluding chapter summarizes these contributions and outlines promising future research directions that build on the findings presented.

This chapter is structured as follows: in Section 7.1, we provide a comprehensive summary of the key contributions and the impact of this work for implicit XDG shock tracking (XDG-IST), followed by a discussion of potential extensions and an exploration of future avenues of research that address current limitations and open questions. Section 7.2, serves the same purpose for the work on preconditioners for IST. Lastly, we synthesize the contributions for both topics in Section 7.3.

7.1 Implicit XDG shock tracking

We start summarizing the contributions for XDG-IST methods (Section 7.1.1) and outline promising future research directions that build on the findings presented (Section 7.1.2).

7.1.1 Summary of contributions

In Chapter 4, we introduced the novel XDG-IST method. It supports a level set φ_b for geometry immersion (Geisenhofer et al., 2019), and an additional shock level set φ_s, the latter being fitted to discontinuous solution features by an optimization algorithm. Within cut-cells, the DG approximation space is enriched with XDG basis functions, allowing for an accurate representation of solution discontinuities along the interfaces which are implicitly defined by the level sets. To obtain high-order shock-aligned XDG solutions, the presented shock tracking method solves a constrained optimization problem using a quasi-Newton sequential quadratic programming (SQP) solver. Solution extrapolation in newborn cut-cells, adaptive regularization and globalization through line search subroutines are used to improve the stability of the method. Additionally, the method supports different discretization approaches for the shock level set function, including discontinuous Galerkin (DG), continuous Galerkin (CG), global, and spline-based representations. We also presented robustness measures to enhance the convergence of the solver: cell agglomeration, solution re-initialization, and the P-continuation strategy.

The novel method was implemented into the bounded support spectral solver (BoSSS) framework (Kummer et al., 2024), an XDG-based open-source software for the simulation of fluid dynamics, and we have equipped the electronic version of this dissertation with hyperlinks, connecting the formal presentation with corresponding lines of code, fostering transparency, accessibility and re-usability. All studies presented can be reproduced by running corresponding Jupyter notebooks, with a single mouse-click.

Contrary to mesh-based IST, the novel XDG method circumvents complex mesh-operations, which is particularly promising for transient simulations with moving shocks and three-dimensional (3D) simulations. Additionally, thanks to the immersed boundaries employed, the XDG shock tracking method comes with the benefit of circumventing the generation of boundary fitted meshes.

In Chapter 5, a series of test cases were developed and we showed numerical results for the XDG-IST method across four different systems of conservation laws: the one-dimensional (1D) space-time Burgers equation, the 1D space-time advection equation, the 1D space-time Euler equations, and the steady two-dimensional (2D) Euler equations. Our results showcased the method's ability to accurately track both straight-sided and curved discontinuities, even when initiated with non-sub-cell accurate initial guesses. In the context of compressible flows with shocks, we successfully applied the XDG-IST method to supersonic flow over an inclined plane, over a blunt body and to 1D shock-acoustic-wave interaction problems.

Throughout our research, we evaluated various configurations of the XDG-IST method to enhance its effectiveness and efficiency. Specifically, we tested three distinct level set discretizations and we investigated the impact of different objective functions on the method. The results of our studies indicate that the XDG-IST method performs best when utilizing spline-based level sets as well as the objective function based on the full enriched residual. Further analysis involved comparing various numerical fluxes employed at the shock interface edges for the steady 2D Euler equations, showing that only Godunov's flux and Roe's flux are stable and consistent enough to be used there as numerical fluxes, and that Roe's flux is a viable alternative to the more expensive Godunov's flux.

Also, two convergence studies were conducted using the Mach 4 bow shock test case and a 1D shock-acoustic-wave interaction problem. The results showcased high-order convergence properties of the XDG-IST method, sub-optimal for the Mach 4 bow shock and super-optimal for the 1D shock-acoustic-wave interaction problem.

Additionally, for the same test cases, we compared the accuracy of the XDG shock tracking method against a traditional DG-immersed boundary method (IBM) implemented within the BoSSS framework, which incorporates a shock capturing strategy based on artificial viscosity. The presented results indicated the superiority of the XDG-IST method over the DG-IBM method, both in terms of accuracy and convergence properties for the considered examples. Remarkably, for the 1D shock-acoustic-wave interaction problem, only the XDG-IST method showed high-order convergence. For the DG-IBM method the same low-order convergence rates were observed for all polynomial degrees ($P = 1, 2, 3$).

Lastly, a study measuring the computational performance of the XDG-IST method was conducted, showing that the biggest driver in terms of computational costs is the assembly of the residual Jacobian with respect to the level set degrees of freedom (DOFs).

7.1.2 Outlook

While our examples were relatively straightforward, featuring single shocks without complex patterns, our framework holds promise for handling more difficult discontinuity scenarios in PDEs. However, to enhance the method's versatility and facilitate its adaptation to a broad spectrum of high-order XDG shock tracking applications, including its extension to three dimensions, substantial further inquiry is essential.

Shock level set All components of the XDG-IST method can be directly extended to 3D contexts except for the current preferred level set representation. The spline-based level set, optimized for computational efficiency and C^1-continuity, is presently limited to shocks that can be described solely as 1D height functions and to applications in two dimensions. Consequently, investigating alternative level set representations with greater flexibility while carefully managing associated computational costs for Jacobians emerges as a research priority.

Potential avenues of exploration include strategies such as representing shocks using polygonal meshes in conjunction with explicit quadrature rules tailored for cut-cells. Additionally, investigating cell-local continuous level sets featuring displacement fields holds significant promise. Such approaches offer the potential to replace iterative re-computation procedures with analytical transformations, thereby enhancing the method's effectiveness for XDG shock tracking in 3D scenarios.

Robustness Another critical aspect for improvement is enhancing the robustness of the XDG-IST method. In our observations for the Mach 4 bow shock scenario (discussed in Section 5.1.3), we noted instances where the method failed due to slight variations in the initial guess or method parameters. This highlights the necessity for further robustness measures. One potential enhancement could involve refining objective functions by incorporating a component that assesses the quality of the shock interface, similar to mesh-based IST methods.

Additionally, adopting a Mach-continuation strategy as proposed by Huang et al. (2023) could be beneficial for handling compressible flows with challenging high Mach numbers. This strategy involves starting with a solution computed at a low Mach number, e.g., $\mathrm{Ma} = 1.5$, and progressively increasing it until the desired Mach number is attained. Furthermore, utilizing level sets with even smoother interfaces, such as those based on Bernstein polynomials, may improve robustness. Finally, addressing the sensitivity of the method to parameters like the agglomeration parameter, background mesh selection, and initial guess quality is crucial for future investigations.

Space-time simulations In this dissertation, we computed solutions to three 1D unsteady conservation laws using a space-time formulation, solving for the entire time interval. While effective for simpler problems, we encountered difficulties in computing solutions for the 1D Euler equations at $\mathrm{Ma} > 3$, particularly in the context of the well-known 1D Shu-Osher problem (Shu and Osher, 1988). In this problem, a shock propagates through a pressure wave, leading to amplification and steepening of the pressure wave due to shock-pressure-wave interactions.

Other IST methods utilize a slab-based strategy, dividing the time interval of interest and sequentially computing the solution, with the previously converged time-step serving as the initial boundary for the next one (Corrigan et al., 2019b; Naudet and Zahr, 2024). We believe that adopting such a slab-based strategy, potentially combined with a Mach-continuation strategy as mentioned earlier, could enhance the stability of the shock tracking method for space-time problems.

Method of lines for transient simulations In future research, it would be valuable to explore an XDG shock tracking method employing a method of lines strategy for time integration. This approach avoids the computational burden of four-dimensional computations necessary for 3D space-time problems. Additionally, mesh-based IST methods, such as the one discussed by Shi et al. (2022), necessitate cumbersome re-meshing for the method of lines strategy, as only certain parts of the mesh move with the shock. This can result in inadequately resolved regions upstream and excessively resolved regions downstream of the shock, particularly without artificial mesh adaptation.

In contrast, level-set-based XDG shock tracking, utilizing a fixed grid and featuring a flexible shock interface capable of moving freely throughout the domain, may offer advantages by eliminating the need for re-meshing entirely. A hypothetical unsteady XDG shock tracking method could be devised as a predictor-corrector scheme, where an explicitly computed level set time-step initially serves as a predictive first guess, later refined by an optimizer. Exploring such an approach could enhance the efficiency and accuracy of XDG shock tracking methods for unsteady problems.

7.2 Preconditioners for implicit shock tracking

In this section, the contributions to the development of novel preconditioners for IST are summarized in Section 7.2.1, followed by an outlook for future avenues of research in Section 7.2.2.

7.2.1 Summary of contributions

In Chapter 6, we introduced matrix-based constrained preconditioners for high-order IST methods and conducted extensive performance tests across different optimization solver parameters. We primarily examined the mesh-based high-order implicit shock tracking (HOIST) method (Zahr et al., 2020), utilizing the enriched DG residual as the objective function. However, these preconditioners are also suitable for other IST approaches.

We developed a series of approximate block anti-triangular preconditioners by analyzing the block structure and sparsity of the IST linear system. These preconditioners, which include common DG techniques like block Jacobi and block incomplete LU factorization (BILU), incorporate minimal discarded fill reordering and circumvent the explicit formation of memory-intensive sub-matrices $\boldsymbol{B}_{\boldsymbol{uu}}$ and $\boldsymbol{B}_{\boldsymbol{yu}}$, ultimately enhancing efficiency. Additionally, we introduced a two-level P-multigrid scheme compatible with any of these preconditioners.

We conducted a rigorous evaluation of our preconditioners on two compressible inviscid 2D flow problems, assessing the number of GMRES iterations required to achieve a specified relative error norm. The preconditioners were tested at different stages of the HOIST method, with variations in key parameters. Our results indicated that only the regularization parameter γ and the polynomial degree of both the solution and the mesh significantly affected the GMRES iteration count. Notably, BILU-based preconditioners demonstrated effective performance across different scenarios. The most effective and reliable BILU variant employed an ILU0 approximation to $\boldsymbol{B}_{\boldsymbol{yy}}$. Additionally, we found that the two-level P-multigrid scheme did not provide sufficient benefits to justify its computational cost.

7.2.2 Outlook

Several interesting research opportunities exist for developing efficient preconditioners for IST.

Performance in massively parallel 3D settings It is imperative to investigate how the proposed preconditioners perform in massively parallel computing environments and when applied to 3D settings. Future studies should evaluate both the number of iterations required to achieve convergence and the actual computational time, i.e. both CPU and wall time. Our most reliable preconditioner was a BILU variant which employed a global ILU0 approximation to $\boldsymbol{B}_{\boldsymbol{yy}}$. In massively parallel settings, the ILU0 approximation will need to be replaced by a parallel variant. For this variant additional fill-in is discarded if it would result from communication between distinct processes. This adjustment, converging to a point Jacobi approximation in the limit where each process deals with one element, will certainly impact the preconditioner's performance.

Multigrid approach In our investigations, we found that the multigrid approach did not provide sufficient advantages to justify its computational expense. Interestingly, when using the best-case preconditioner, which does not involve approximations for $\boldsymbol{J}_{\boldsymbol{u}}$ and $\boldsymbol{B}_{\boldsymbol{yy}}$, adding multigrid increased iteration numbers, contrary to its expected behavior. However, it improved iteration numbers for other tested preconditioners. It's worth delving deeper into this discrepancy to potentially refine the multigrid approach. One avenue for exploration leads to the low-order spaces utilized at the coarse scale and the associated projections within the restriction operators. By experimenting with different spaces or introducing L^2-projections, we may enhance outcomes.

Effectiveness of multigrid in viscous problems Secondly, the benefits of using the P-multigrid approach might become more pronounced for other problems, in particular as viscosity is introduced, which was observed by Persson and Peraire (2008) for DG methods. In their study, Persson and Peraire (2008) examined BILU-based preconditioners with and without P-multigrid for the advection diffusion equation using different amounts of diffusion. When no diffusion was prescribed, P-multigrid added no value, but for higher diffusion amounts it significantly enhanced performance. Hence, evaluating the proposed preconditioners for viscous problems in the context of IST marks an interesting avenue for further research.

Application to other implicit shock tracking methods Testing the preconditioners for other IST methods, such as the XDG method introduced in this work, could also yield valuable insights. A key aspect to consider for the XDG-IST method, is the variation in size and structure of the sub-matrix $\boldsymbol{B}_{\varphi\varphi}$ resulting from the discretization of the level set. If, for instance, a DG level set is employed, the matrix will exhibit a block structure typical of DG methods, suggesting that DG-specific preconditioners might be effective. Conversely, a global representation of the level set results in a smaller, denser matrix, where simpler preconditioning approaches like ILU0 could be effective, even in large-scale applications.

7.3 Final remarks

In conclusion, we have introduced significant advancements in the field of implicit shock tracking, a high-order technique currently under investigation for the accurate and robust simulation of compressible flows featuring shocks. First, we have enhanced IST methods with the innovative XDG-IST method, which integrates the IST framework with a cut-cell technique, thus avoiding complex mesh manipulations. Although this method is in its early stages and still faces some technical challenges, it shows promise as a potential IST method of choice for a subset of compressible flow problems with dynamic shocks and fewer discontinuities.

Second, we have developed the first family of preconditioners for IST, a necessary building-block for enabling the application of IST methods to large-scale, real-world problems. Therefore, we have provided essential components that will allow IST methods to become preferred methodologies for simulating high-speed flows. In the future, IST methods could significantly support aerospace engineers in the development and testing of new technologies, offering improved accuracy and efficiency compared to current standards. This aligns with broader goals of achieving more sustainable and convenient aviation technologies, reducing noise and pollutant emissions, and facilitating progress towards supersonic civil aircrafts.

Bibliography

Anderson, J. D. (2003). *Modern compressible flow: With historical perspective*. 3rd ed. McGraw-Hill series in aeronautical and aerospace engineering. tex.lccn: QA911 .A6 2003. Boston: McGraw-Hill. ISBN: 978-0-07-242443-0.

Assonitis, A., M. Ciallella, R. Paciorri, M. Ricchiuto, and A. Bonfiglioli (2022). “A new shock-fitting technique for 2-D structured grids”. en. In: *AIAA SCITECH 2022 Forum*. San Diego, CA & Virtual: American Institute of Aeronautics and Astronautics. ISBN: 978-1-62410-631-6. DOI: 10.2514/6.2022-2008.

Assonitis, A., M. Ciallella, M. Ricchiuto, and L. Cirrottola (2023). “Numerical simulations of shock interactions on 3D structured grids using a shock-fitting approach”. en. In: *AIAA SCITECH 2023 Forum*. National Harbor, MD & Online: American Institute of Aeronautics and Astronautics. ISBN: 978-1-62410-699-6. DOI: 10.2514/6.2023-2135.

Baines, M. J., S. J. Leary, and M. E. Hubbard (2002). “Multidimensional Least Squares Fluctuation Distribution Schemes with Adaptive Mesh Movement for Steady Hyperbolic Equations”. en. In: *SIAM Journal on Scientific Computing* 23.5, pp. 1485–1502. ISSN: 1064-8275, 1095-7197. DOI: 10.1137/S1064827500370202.

Barter, G. E. and D. L. Darmofal (2010). “Shock capturing with PDE-based artificial viscosity for DGFEM: Part I. Formulation”. en. In: *Journal of Computational Physics* 229.5, pp. 1810–1827. ISSN: 00219991. DOI: 10.1016/j.jcp.2009.11.010.

Bassi, F. and S. Rebay (1997). “A High-Order Accurate Discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier–Stokes Equations”. In: *Journal of Computational Physics* 131.2, pp. 267–279. ISSN: 0021-9991. DOI: https://doi.org/10.1006/jcph.1996.5572.

Bastian, P. and C. Engwer (2009). “An unfitted finite element method using discontinuous Galerkin”. en. In: *International Journal for Numerical Methods in Engineering* 79.12, pp. 1557–1576. ISSN: 0029-5981, 1097-0207. DOI: 10.1002/nme.2631.

Beck, A. D., J. Zeifang, A. Schwarz, and D. G. Flad (2020). “A neural network based shock detection and localization approach for discontinuous Galerkin methods”. In: *Journal of Computational Physics* 423, p. 109824. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2020.109824.

Beck, L. and F. Kummer (2023). *High-Order Numerical Integration on Domains Bounded by Intersecting Level Sets*. en. arXiv:2308.10698 [cs, math].

Benson, L. (2013). *Quieting the boom: The shaped sonic boom demonstrator and the quest for quiet supersonic flight*. Nasa sp. tex.lccn: 2013004829. National Aeronautics and Space Administration, Aeronautics Research Mission Directorate. ISBN: 978-1-62683-004-2.

Benzi, M., G. H. Golub, and J. Liesen (2005). “Numerical solution of saddle point problems”. en. In: *Acta Numerica* 14, pp. 1–137. ISSN: 0962-4929, 1474-0508. DOI: 10.1017/S0962492904000212.

Biros, G. and O. Ghattas (2000). “Parallel Preconditioners for KKT Systems Arising in Optimal Control of Viscous Incompressible Flows”. en. In: *Parallel Computational Fluid Dynamics 1999*. Elsevier, pp. 131–138. ISBN: 978-0-444-82851-4. DOI: 10.1016/B978-044482851-4.50017-7.

Blokhintsev, D. (1945). “Sound receiver in motion”. In: *Dokl. Akad. Nauk SSSR*. Vol. 47. 1, pp. 22–25.

Bonfiglioli, A., R. Paciorri, and L. Campoli (2016). “Unsteady shock-fitting for unstructured grids: Unsteady shock-fitting for unstructured grids”. en. In: *International Journal for Numerical Methods in Fluids* 81.4, pp. 245–261. ISSN: 02712091. DOI: 10.1002/fld.4183.

Burgers, J. M. (1946). *On the transmission of sound waves through a shock wave*. North-Holland Publishing Company.

Caussignac, P. and R. Touzan (1990). “Solution of three-dimensional boundary layer equations by a discontinuous finite element method, part I: Numerical analysis of a linear model problem”. en. In: *Computer Methods in Applied Mechanics and Engineering* 78.3, pp. 249–271. ISSN: 00457825. DOI: 10.1016/0045-7825(90)90001-3.

Chavent, G. and B. Cockburn (1989). “The local projection P^0-P^1-discontinuous-Galerkin finite element method for scalar conservation laws”. In: *ESAIM: Mathematical Modelling and Numerical Analysis* 23.4, pp. 565–592. ISSN: 0764-583X, 1290-3841. DOI: 10.1051/m2an/1989230405651.

Ching, E. J., A. D. Kercher, and A. Corrigan (2024). “The moving discontinuous Galerkin method with interface condition enforcement for the simulation of hypersonic, viscous flows”. In: *Computer Methods in Applied Mechanics and Engineering* 427, p. 117045. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2024.117045.

Ching, E. J., Y. Lv, P. Gnoffo, M. Barnhardt, and M. Ihme (2019). “Shock capturing for discontinuous Galerkin methods with application to predicting heat transfer in hypersonic flows”. en. In: *Journal of Computational Physics* 376, pp. 54–75. ISSN: 00219991. DOI: 10.1016/j.jcp.2018.09.016.

Cockburn, B. (1998). “An introduction to the Discontinuous Galerkin method for convection-dominated problems”. en. In: *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations*. Ed. by A. Quarteroni. Vol. 1697. Series Title: Lecture Notes in Mathematics. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 150–268. ISBN: 978-3-540-64977-9 978-3-540-49804-9. DOI: 10.1007/BFb0096353.

Cockburn, B., S.-Y. Lin, and C.-W. Shu (1989). “TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: One-dimensional systems”. en. In: *Journal of Computational Physics* 84.1, pp. 90–113. ISSN: 00219991. DOI: 10.1016/0021-9991(89)90183-6.

Cockburn, B. and C.-W. Shu (1991). “The Runge-Kutta local projection P^1-discontinuous-Galerkin finite element method for scalar conservation laws”. In: *ESAIM: Mathematical Modelling and Numerical Analysis* 25.3, pp. 337–361. ISSN: 0764-583X, 1290-3841. DOI: 10.1051/m2an/1991250303371.

Coleman, T. F. and A. Verma (n.d.). “A Preconditioned Conjugate Gradient Approach to Linear Equality Constrained Minimization”. en. In: ().

Corrigan, A., A. D. Kercher, and D. A. Kessler (2019a). “A moving discontinuous Galerkin finite element method for flows with interfaces”. en. In: *International Journal for Numerical Methods in Fluids* 89.9, pp. 362–406. ISSN: 0271-2091, 1097-0363. DOI: 10.1002/fld.4697.

Corrigan, A. T., A. Kercher, and D. A. Kessler (2019b). “The Moving Discontinuous Galerkin Method with Interface Condition Enforcement for Unsteady Three-Dimensional Flows”. In:

AIAA Scitech 2019 Forum. San Diego, California: American Institute of Aeronautics and Astronautics. ISBN: 978-1-62410-578-4. DOI: 10.2514/6.2019-0642.

Corrigan, A. T., A. D. Kercher, D. A. Kessler, and D. A. Wood-Thomas (2019c). "Convergence of the Moving Discontinuous Galerkin Method with Interface Condition Enforcement in the Presence of an Attached Curved Shock". en. In: *AIAA Aviation 2019 Forum*. Dallas, Texas: American Institute of Aeronautics and Astronautics. ISBN: 978-1-62410-589-0. DOI: 10.2514/6.2019-3207.

Council, N. R., D. on Engineering, P. Sciences, Aeronautics, S. E. Board, and C. on Breakthrough Technology for Commercial Supersonic Aircraft (2002). *Commercial supersonic technology: The way ahead*. Compass series. tex.lccn: 2002284699. National Academies Press. ISBN: 978-0-309-08277-8.

Dennis, J. E. and R. B. Schnabel (1996). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. en. Society for Industrial and Applied Mathematics. ISBN: 978-0-89871-364-0 978-1-61197-120-0. DOI: 10.1137/1.9781611971200.

Di Pietro, D. A. and A. Ern (2012). *Mathematical Aspects of Discontinuous Galerkin Methods*. en. Vol. 69. Mathématiques et Applications. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-642-22979-4 978-3-642-22980-0. DOI: 10.1007/978-3-642-22980-0.

Dollar, H. S. (2007). "Constraint-Style Preconditioners for Regularized Saddle Point Problems". en. In: *SIAM Journal on Matrix Analysis and Applications* 29.2, pp. 672–684. ISSN: 0895-4798, 1095-7162. DOI: 10.1137/050626168.

Feldhusen-Hoffmann, A., V. Statnikov, M. Klaas, and W. Schröder (2018). "Investigation of shock–acoustic-wave interaction in transonic flow". en. In: *Experiments in Fluids* 59.1, p. 15. ISSN: 0723-4864, 1432-1114. DOI: 10.1007/s00348-017-2466-z.

Fidkowski, K. J., T. A. Oliver, J. Lu, and D. L. Darmofal (2005). "p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations". en. In: *Journal of Computational Physics* 207.1, pp. 92–113. ISSN: 00219991. DOI: 10.1016/j.jcp.2005.01.005.

Geisenhofer, M. (2021). "From Shock-Capturing to High-Order Shock-Fitting Using an Unfitted Discontinuous Galerkin Method". en. PhD thesis. Darmstadt: Technische Universität Darmstadt, xxix, 166 Seiten. DOI: https://doi.org/10.26083/tuprints-00017526.

Geisenhofer, M., F. Kummer, and B. Müller (2019). "A discontinuous Galerkin immersed boundary solver for compressible flows: Adaptive local time stepping for artificial viscosity–based shock-capturing on cut cells". en. In: *International Journal for Numerical Methods in Fluids* 91.9, pp. 448–472. ISSN: 0271-2091, 1097-0363. DOI: 10.1002/fld.4761.

Geisenhofer, M., F. Kummer, and M. Oberlack (2020). "An Extended Discontinuous Galerkin Method for High-Order Shock-Fitting". In: Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.2012.08860.

Godunov, S. K. and I. Bohachevsky (1959). "Finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics". In: *Matematičeskij sbornik* 47(89).3, pp. 271–306.

Gould, N. I. M., M. E. Hribar, and J. Nocedal (2001). "On the Solution of Equality Constrained Quadratic Programming Problems Arising in Optimization". en. In: *SIAM Journal on Scientific Computing* 23.4, pp. 1376–1395. ISSN: 1064-8275, 1095-7197. DOI: 10.1137/S1064827598345667.

Grube, N. E. (2020). "Shock Wave–Turbulence Interactions". en. PhD thesis. Princeton University.

Harten, A., B. Engquist, S. Osher, and S. R. Chakravarthy (1997). “Uniformly High Order Accurate Essentially Non-oscillatory Schemes, III”. en. In: *Journal of Computational Physics* 131.1, pp. 3–47. ISSN: 00219991. DOI: 10.1006/jcph.1996.5632.

Harten, A. and J. M. Hyman (1983). “Self adjusting grid methods for one-dimensional hyperbolic conservation laws”. en. In: *Journal of Computational Physics* 50.2, pp. 235–269. ISSN: 00219991. DOI: 10.1016/0021-9991(83)90066-9.

Harten, A., P. D. Lax, and B. v. Leer (1983). “On upstream differencing and godunov-type schemes for hyperbolic conservation laws”. In: *SIAM Review* 25.1, pp. 35–61. DOI: 10.1137/1025002.

Hartmann, R. and P. Houston (2008). “An optimal order interior penalty discontinuous Galerkin discretization of the compressible Navier–Stokes equations”. In: *Journal of Computational Physics* 227.22, pp. 9670–9685. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2008.07.015.

Heimann, F., C. Engwer, O. Ippisch, and P. Bastian (2013). “An unfitted interior penalty discontinuous Galerkin method for incompressible Navier–Stokes two-phase flow”. en. In: *International Journal for Numerical Methods in Fluids* 71.3, pp. 269–293. ISSN: 0271-2091, 1097-0363. DOI: 10.1002/fld.3653.

Henneaux, D., P. Schrooyen, B. Ricardo Barros Dias, A. Turchi, P. Chatelain, and T. Magin (2020). “Extended Discontinuous Galerkin Method for Solving Gas-Liquid Compressible Flows with Phase Transition”. en. In: *AIAA AVIATION 2020 FORUM*. VIRTUAL EVENT: American Institute of Aeronautics and Astronautics. ISBN: 978-1-62410-598-2. DOI: 10.2514/6.2020-2971.

Hesthaven, J. S. and T. Warburton (2008). *Nodal Discontinuous Galerkin Methods*. en. Ed. by J. E. Marsden, L. Sirovich, and S. S. Antman. Vol. 54. Texts in Applied Mathematics. New York, NY: Springer New York. ISBN: 978-0-387-72065-4 978-0-387-72067-8. DOI: 10.1007/978-0-387-72067-8.

Huang, T., C. J. Naudet, and M. J. Zahr (2023). *High-order implicit shock tracking boundary conditions for flows with parametrized shocks*. DOI: https://doi.org/10.1016/j.jcp.2023.112517.

Huang, T. and M. J. Zahr (2022). “A robust, high-order implicit shock tracking method for simulation of complex, high-speed flows”. en. In: *Journal of Computational Physics* 454, p. 110981. ISSN: 00219991. DOI: 10.1016/j.jcp.2022.110981.

Hugoniot, H. (1887). “On the Propagation of Motion in Bodies and in Perfect Bodies in Particular”. In: *IJ l’Ecole Polytech* 57, pp. 3–97.

Johnson, C. and J. Pitkäranta (1986). “An analysis of the discontinuous Galerkin method for a scalar hyperbolic equation”. en. In: *Mathematics of Computation* 46.173, pp. 1–26. ISSN: 0025-5718, 1088-6842. DOI: 10.1090/S0025-5718-1986-0815828-4.

Karniadakis, G. and S. Sherwin (2005). *Spectral/hp element methods for computational fluid dynamics*. Oxford University Press. ISBN: 978-0-19-852869-2. DOI: 10.1093/acprof:oso/9780198528692.001.0001.

Keller, C., N. I. M. Gould, and A. J. Wathen (2000). “Constraint Preconditioning for Indefinite Linear Systems”. en. In: *SIAM Journal on Matrix Analysis and Applications* 21.4, pp. 1300–1317. ISSN: 0895-4798, 1095-7162. DOI: 10.1137/S0895479899351805.

Kercher, A. D. and A. Corrigan (2021). “A least-squares formulation of the Moving Discontinuous Galerkin Finite Element Method with Interface Condition Enforcement”. en. In: *Computers & Mathematics with Applications* 95, pp. 143–171. ISSN: 08981221. DOI: 10.1016/j.camwa.2020.09.012.

Kercher, A. D., A. Corrigan, and D. A. Kessler (2021). “The moving discontinuous Galerkin finite element method with interface condition enforcement for compressible viscous flows”. en. In: *International Journal for Numerical Methods in Fluids* 93.5, pp. 1490–1519. ISSN: 0271-2091, 1097-0363. DOI: 10.1002/fld.4939.

Klapdor, E. V. (2011). “Simulation of Combustor-Turbine Interaction in a Jet Engine”. en. PhD thesis. Darmstadt: Technische Universität Darmstadt.

Knupp, P. M. (2001). “Algebraic Mesh Quality Metrics”. In: *SIAM Journal on Scientific Computing* 23.1, pp. 193–218. ISSN: 1064-8275. DOI: 10.1137/S1064827500371499.

Krais, N., A. Beck, T. Bolemann, H. Frank, D. Flad, G. Gassner, F. Hindenlang, M. Hoffmann, T. Kuhn, M. Sonntag, and C.-D. Munz (2021). “FLEXI: A high order discontinuous Galerkin framework for hyperbolic–parabolic conservation laws”. en. In: *Computers & Mathematics with Applications* 81, pp. 186–219. ISSN: 08981221. DOI: 10.1016/j.camwa.2020.05.004.

Kral, L. (1998). “Recent experience with different turbulence models applied to the calculation of flow over aircraft components”. en. In: *Progress in Aerospace Sciences* 34.7-8, pp. 481–541. ISSN: 03760421. DOI: 10.1016/S0376-0421(98)00009-8.

Krämer-Eis, S. (2017). *A high-order discontinuous Galerkin method for unsteady compressible flows with immersed boundaries*. eng. Göttingen: Cuvillier Verlag. ISBN: 978-3-7369-9635-9.

Kroll, N. (2010). “The ADIGMA Project”. In: *ADIGMA - A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications*. Ed. by E. H. Hirschel, W. Schröder, K. Fujii, W. Haase, B. Leer, M. A. Leschziner, M. Pandolfi, J. Periaux, A. Rizzi, B. Roux, Y. I. Shokin, N. Kroll, H. Bieler, H. Deconinck, V. Couaillier, H. Ven, and K. Sørensen. Vol. 113. Series Title: Notes on Numerical Fluid Mechanics and Multidisciplinary Design. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–9. ISBN: 978-3-642-03706-1 978-3-642-03707-8. DOI: 10.1007/978-3-642-03707-8_1.

Kroll, N., C. Hirsch, F. Bassi, C. Johnston, K. Hillewaert, and Editors (2015). *IDIHOM: Industrialization of high-order methods - a top down approach. Results of a collaborative research project funded by the european union, 2010-2014*. Vol. 128. ISBN: 978-3-319-12885-6. DOI: 10.1007/978-3-319-12886-3.

Kuhn, H. W. and A. W. Tucker (1951). “Nonlinear programming”. In: *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, 1950*. Berkeley and Los Angeles: University of California Press, pp. 481–492.

Kummer, F. (2017). “Extended discontinuous Galerkin methods for two-phase flows: the spatial discretization”. en. In: *International Journal for Numerical Methods in Engineering* 109.2, pp. 259–289. ISSN: 00295981. DOI: 10.1002/nme.5288.

Kummer, F., B. Müller, B. Klein, N. Emamy, R. Mousavi, D. Klingenberg, D. Benjamin, M. Geisenhofer, M. Smuda, D. Dierkes, A. Kikker, D. Krause, T. Utz, C. Kallendorf, S. Krämer-Eis, J. Vandergrift, J. Gutiérrez-Jorquera, and M. Toprak (2024). *BoSSS*. Version v250506. DOI: 10.5281/zenodo.11118910.

Kummer, F., B. Müller, and T. Utz (2018). “Time integration for extended discontinuous Galerkin methods with moving domains: Time integration for XDG methods with moving domains”. In: *International Journal for Numerical Methods in Engineering* 113.5, pp. 767–788. ISSN: 00295981. DOI: 10.1002/nme.5634.

Kummer, F., J. Weber, and M. Smuda (2021). “BoSSS: A package for multigrid extended discontinuous Galerkin methods”. en. In: *Computers & Mathematics with Applications* 81, pp. 237–257. ISSN: 08981221. DOI: 10.1016/j.camwa.2020.05.001.

Landau, L. D. and E. M. Lifshitz (2013). *Fluid mechanics: Landau and lifshitz: course of theoretical physics, volume 6*. Vol. 6. Elsevier.

Lasaint, P. and P. Raviart (1974). "On a Finite Element Method for Solving the Neutron Transport Equation". en. In: *Mathematical Aspects of Finite Elements in Partial Differential Equations*. Elsevier, pp. 89–123. ISBN: 978-0-12-208350-1. DOI: 10.1016/B978-0-12-208350-1.50008-X.

LeVeque, R. J. (1992). *Numerical methods for conservation laws*. en. 2nd ed. Lectures in mathematics ETH Zürich. Basel ; Boston: Birkhäuser Verlag. ISBN: 978-3-7643-2723-1 978-0-8176-2723-2.

Lukšan, L. and J. Vlček (2001). "Numerical experience with iterative methods for equality constrained nonlinear programming problems". en. In: *Optimization Methods and Software* 16.1-4, pp. 257–287. ISSN: 1055-6788, 1029-4937. DOI: 10.1080/10556780108805838.

Majda, A. (1984). *Compressible fluid flow and systems of conservation laws in several space variables*. eng. Applied mathematical sciences 53. New York Berlin Heidelberg: Springer. ISBN: 978-1-4612-1116-7 978-0-387-96037-1 978-3-540-96037-9.

Mengaldo, G., D. De Grazia, J. Peiro, A. Farrington, F. Witherden, P. Vincent, and S. Sherwin (2014). "A guide to the implementation of boundary conditions in compact high-order methods for compressible aerodynamics". In: AIAA AVIATION 2014 -7th AIAA Theoretical Fluid Mechanics Conference. DOI: 10.2514/6.2014-2923.

Menon, E. S. and P. S. Menon (2013). *Gas pipeline hydraulics*. eng. Bloomington, IN: Trafford Publ. ISBN: 978-1-4669-7670-2 978-1-4669-7671-9.

Moretti, G. and M. Valorani (1988). "Detection and fitting of two-dimensional shocks". In: Louvain-la-Neuve, Belgium: Friedr. Vieweg und Sohn, pp. 239–246.

Mös, N., J. Dolbow, and T. Belytschko (1999). "A finite element method for crack growth without remeshing". en. In: *International Journal for Numerical Methods in Engineering* 46.1, pp. 131–150. ISSN: 0029-5981, 1097-0207. DOI: 10.1002/(SICI)1097-0207(19990910)46:1<131::AID-NME726>3.0.CO;2-J.

Mossier, P., A. Beck, and C.-D. Munz (2022). "A p-Adaptive Discontinuous Galerkin Method with hp-Shock Capturing". en. In: *Journal of Scientific Computing* 91.1, p. 4. ISSN: 0885-7474, 1573-7691. DOI: 10.1007/s10915-022-01770-6.

Müller, B., S. Krämer-Eis, F. Kummer, and M. Oberlack (2017). "A high-order discontinuous Galerkin method for compressible flows with immersed boundaries". en. In: *International Journal for Numerical Methods in Engineering* 110.1, pp. 3–30. ISSN: 00295981. DOI: 10.1002/nme.5343.

Müller, B. (2014). "Methods for higher order numerical simulations of complex inviscid fluids with immersed boundaries". Dissertation. Darmstadt: TU Darmstadt.

Murman, S. M. (2017). "CI1 - Inviscid bow shock upstream of a blunt body in supersonic flow". en. In: Available at https://how5.cenaero.be/content/ci1-inviscid-bow-shock.

Nasuti, F. and M. Onofri (1996). "Analysis of unsteady supersonic viscous flows by a shock-fitting technique". In: *AIAA Journal* 34.7, pp. 1428–1434. ISSN: 0001-1452, 1533-385X. DOI: 10.2514/3.13249.

Naudet, C. J. and M. J. Zahr (2024). *A space-time high-order implicit shock tracking method for shock-dominated unsteady flows*. DOI: https://doi.org/10.1016/j.jcp.2024.112792.

Nocedal, J. and S. J. Wright (2006). *Numerical optimization*. en. 2nd ed. Springer series in operations research. OCLC: ocm68629100. New York: Springer. ISBN: 978-0-387-30303-1.

Paciorri, R. and A. Bonfiglioli (2009). "A shock-fitting technique for 2D unstructured grids". In: *Computers & Fluids* 38.3, pp. 715–726. ISSN: 00457930. DOI: 10.1016/j.compfluid.2008.07.007.

Pawlowski, R. P., J. P. Simonis, H. F. Walker, and J. N. Shadid (2008). “Inexact Newton Dogleg Methods”. en. In: *SIAM Journal on Numerical Analysis* 46.4, pp. 2112–2132. ISSN: 0036-1429, 1095-7170. DOI: 10.1137/050632166.

Persson, P.-O. and J. Peraire (2008). “Newton-GMRES Preconditioning for Discontinuous Galerkin Discretizations of the Navier–Stokes Equations”. en. In: *SIAM Journal on Scientific Computing* 30.6, pp. 2709–2733. ISSN: 1064-8275, 1095-7197. DOI: 10.1137/070692108.

Persson, P.-O. and J. Peraire (2006). “Sub-Cell Shock Capturing for Discontinuous Galerkin Methods”. en. In: *44th AIAA Aerospace Sciences Meeting and Exhibit*. Reno, Nevada: American Institute of Aeronautics and Astronautics. ISBN: 978-1-62410-039-0. DOI: 10.2514/6.2006-112.

Persson, P.-O. and G. Strang (2004). “A Simple Mesh Generator in MATLAB”. In: *SIAM Review* 46.2, pp. 329–345. ISSN: 0036-1445. DOI: 10.1137/S0036144503429121.

Prenter, F. de, C. Verhoosel, H. van Brummelen, M. Larson, and S. Badia (2022). *Stability and conditioning of immersed finite element methods: Analysis and remedies*. arXiv: 2208.08538 [cs, math] Number: arXiv:2208.08538.

Rankine, W. J. M. (1870). “On the Thermodynamic Theory of Waves of Finite Longitudinal Disturbance”. In: *Philosophical Transactions of the Royal Society of London* 160, pp. 277–288. ISSN: 02610523.

Raymer, D. (2012). *Aircraft Design: A Conceptual Approach, Fifth Edition*. en. Washington, DC: American Institute of Aeronautics and Astronautics, Inc. ISBN: 978-1-60086-911-2. DOI: 10.2514/4.869112.

Reed, W. H. and T. R. Hill (1973). *Triangular mesh methods for the neutron transport equation*. LA-UR-73-749.

Rieckmann, M., M. Smuda, P. Stephan, and F. Kummer (2024). “The extended Discontinuous Galerkin method for two-phase flows with evaporation”. In: *Journal of Computational Physics* 499, p. 112716. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2023.112716.

Riemann, B. (1860). “Über die Fortpflanzung ebener Luftwellen von endlicher Schwingungsweite”. In: *Abhandlungen der Königlichen Gesellschaft der Wissenschaften in Göttingen* 8, pp. 43–66.

Roca, X., A. Gargallo-Peiró, and J. Sarrate (2012). “Defining Quality Measures for High-Order Planar Triangles and Curved Mesh Generation”. en. In: *Proceedings of the 20th International Meshing Roundtable*. Ed. by W. R. Quadros. Berlin, Heidelberg: Springer, pp. 365–383. ISBN: 9783642247347. DOI: 10.1007/978-3-642-24734-7_20.

Roe, P. (1981). “Approximate Riemann solvers, parameter vectors, and difference schemes”. In: *Journal of Computational Physics* 43.2, pp. 357–372. ISSN: 0021-9991. DOI: https://doi.org/10.1016/0021-9991(81)90128-5.

Romick, C. M. and T. D. Aslam (2017). “High-order shock-fitted detonation propagation in high explosives”. In: *Journal of Computational Physics* 332, pp. 210–235. ISSN: 00219991. DOI: 10.1016/j.jcp.2016.11.049.

Salas, M. D. (2010). *A shock-fitting primer*. Chapman & Hall/CRC applied mathematics and nonlinear science series. tex.lccn: QA911 .S25 2010. Boca Raton: CRC Press. ISBN: 978-1-4398-0758-3.

Saye, R. I. (2015). “High-Order Quadrature Methods for Implicitly Defined Surfaces and Volumes in Hyperrectangles”. en. In: *SIAM Journal on Scientific Computing* 37.2, A993–A1019. ISSN: 1064-8275, 1095-7197. DOI: 10.1137/140966290.

Saye, R. I. (2022). “High-order quadrature on multi-component domains implicitly defined by multivariate polynomials”. en. In: *Journal of Computational Physics* 448, p. 110720. ISSN: 00219991. DOI: 10.1016/j.jcp.2021.110720.

Schlaich, J., R. Bergermann, W. Schiel, and G. Weinrebe (2005). “Design of Commercial Solar Updraft Tower Systems—Utilization of Solar Induced Convective Flows for Power Generation”. en. In: *Journal of Solar Energy Engineering* 127.1, pp. 117–124. ISSN: 0199-6231, 1528-8986. DOI: 10.1115/1.1823493.

Shi, A., P.-O. Persson, and M. Zahr (2022). “Implicit shock tracking for unsteady flows by the method of lines”. en. In: *Journal of Computational Physics* 454, p. 110906. ISSN: 00219991. DOI: 10.1016/j.jcp.2021.110906.

Shu, C.-W. and S. Osher (1988). “Efficient implementation of essentially non-oscillatory shock-capturing schemes”. en. In: *Journal of Computational Physics* 77.2, pp. 439–471. ISSN: 00219991. DOI: 10.1016/0021-9991(88)90177-5.

Smuda, M. and F. Kummer (2020). “On a marching level-set method for extended discontinuous Galerkin methods for incompressible two-phase flows”. en. In: *arXiv:2010.08417 [cs, math]*. arXiv: 2010.08417.

Toro, E. F. (2009). *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. en. 3rd ed. OCLC: ocn401321914. Dordrecht ; New York: Springer. ISBN: 978-3-540-25202-3 978-3-540-49834-6.

Toro, E. F., M. Spruce, and W. Speares (1994). “Restoration of the contact surface in the HLL-Riemann solver”. en. In: *Shock Waves* 4.1, pp. 25–34. ISSN: 0938-1287, 1432-2153. DOI: 10.1007/BF01414629.

Trépanier, J.-Y., M. Paraschivoiu, M. Reggio, and R. Camarero (1996). “A Conservative Shock Fitting Method on Unstructured Grids”. en. In: *Journal of Computational Physics* 126.2, pp. 421–433. ISSN: 00219991. DOI: 10.1006/jcph.1996.0147.

Vandergrift, J. and F. Kummer (2024). “An extended discontinuous Galerkin shock tracking method”. In: *International Journal for Numerical Methods in Fluids* 96.8, pp. 1384–1414. DOI: https://doi.org/10.1002/fld.5293. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.5293.

Vandergrift, J. and M. J. Zahr (2024). *Preconditioned iterative solvers for constrained high-order implicit shock tracking methods*. DOI: https://doi.org/10.1016/j.jcp.2024.113234.

Wagner, C., T. Hüttl, and P. Sagaut (2007). *Large-Eddy Simulation for Acoustics*. Cambridge Aerospace Series. Cambridge University Press.

Wang, Z., K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. Huynh, N. Kroll, G. May, P.-O. Persson, B. van Leer, and M. Visbal (2013). “High-order CFD methods: current status and perspective”. en. In: *International Journal for Numerical Methods in Fluids* 72.8, pp. 811–845. ISSN: 02712091. DOI: 10.1002/fld.3767.

Wen, T. and M. J. Zahr (2023). “A globally convergent method to accelerate large-scale optimization using on-the-fly model hyperreduction: Application to shape optimization”. en. In: *Journal of Computational Physics* 484, p. 112082. ISSN: 00219991. DOI: 10.1016/j.jcp.2023.112082.

Zahr, M., A. Shi, and P.-O. Persson (2020). “Implicit shock tracking using an optimization-based high-order discontinuous Galerkin method”. en. In: *Journal of Computational Physics* 410, p. 109385. ISSN: 00219991. DOI: 10.1016/j.jcp.2020.109385.

Zahr, M. J. and P.-O. Persson (2018). “An optimization-based approach for high-order accurate discretization of conservation laws with discontinuous solutions”. en. In: *Journal* *of Computational Physics* 365. arXiv: 1712.03445, pp. 105–134. ISSN: 00219991. DOI: 10.1016/j.jcp.2018.03.029.

Zahr, M. J. and P.-O. Persson (2020). “An r-Adaptive, High-Order Discontinuous Galerkin Method for Flows with Attached Shocks”. en. In: *AIAA Scitech 2020 Forum*. Orlando, FL: American Institute of Aeronautics and Astronautics. ISBN: 978-1-62410-595-1. DOI: 10.2514/6.2020-0537.

Zahr, M. J. and J. M. Powers (2021). “High-Order Resolution of Multidimensional Compressible Reactive Flow Using Implicit Shock Tracking”. en. In: *AIAA Journal* 59.1, pp. 150–164. ISSN: 0001-1452, 1533-385X. DOI: 10.2514/1.J059655.