Efficient Solvers for Space-Time Discontinuous Galerkin Spectral Element Methods

Versbach, Lea Miko

2022

Document Version:
Publisher's PDF, also known as Version of record

Link to publication

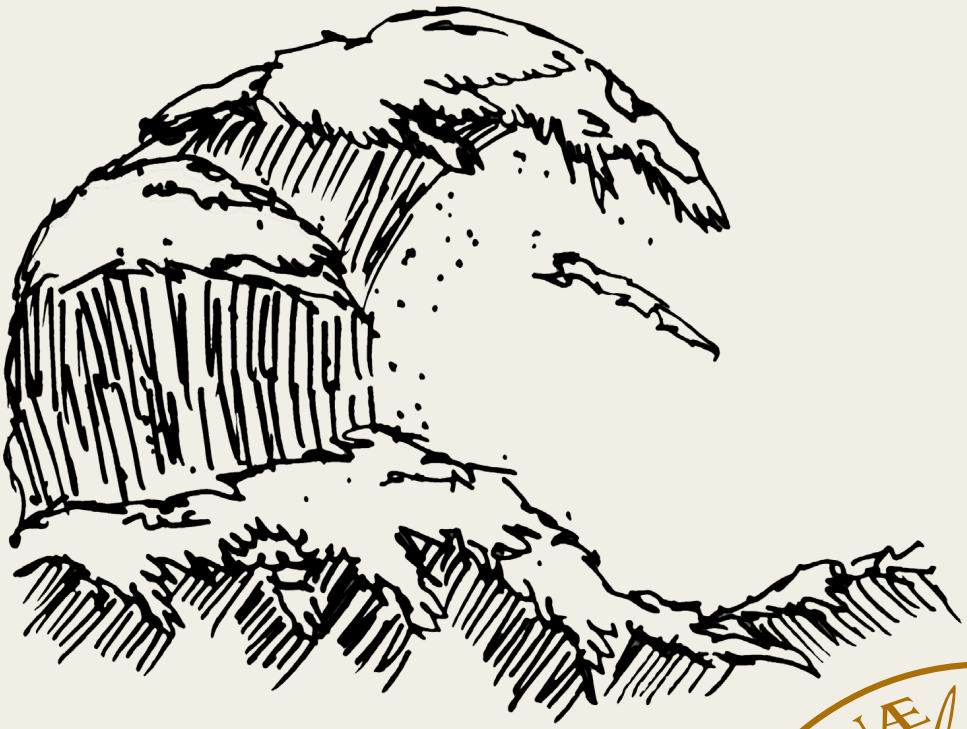
Citation for published version (APA):
Versbach, L. M. (2022). *Efficient Solvers for Space-Time Discontinuous Galerkin Spectral Element Methods*. Lund University.

Total number of authors:
1

– CENTRUM SCIENTIARUM MATHEMATICARUM –

Efficient Solvers for Space-Time Discontinuous Galerkin Spectral Element Methods

LEA MIKO VERSBACH



Lund University
Faculty of Science
Centre for Mathematical Sciences
Numerical Analysis

Efficient Solvers for Space-Time Discontinuous Galerkin Spectral Element Methods

Efficient Solvers for Space-Time Discontinuous Galerkin Spectral Element Methods

by Lea Miko Versbach

LUND UNIVERSITY

Thesis for the degree of Doctor of Philosophy in Numerical Analysis
Thesis advisors: Prof. Dr. Philipp Birken and Prof. Dr. Gregor J. Gassner
Faculty opponent: Prof. Dr. Matthias Bolten, Bergische Universität Wuppertal, Germany

To be presented, with the permission of the Faculty of Science at Lund University, for public criticism in the lecture hall MH:Hörmandersalen at the Centre for Mathematical Sciences, Sölvegatan 18A in Lund, on Monday, the 28th of February 2022 at 15:00.

Organization
LUND UNIVERSITY

Centre for Mathematical Sciences
Box 118
SE–221 00 LUND
Sweden

Document name
Doctoral Dissertation

Date of presentation
2022-02-28

Sponsoring organization
Swedish Research Council, grant number 2015-04133

Author(s)
Lea Miko Versbach

Title and subtitle
Efficient Solvers for Space-Time Discontinuous Galerkin Spectral Element Methods

Abstract

In this thesis we study efficient solvers for space-time discontinuous Galerkin spectral element methods (DG-SEM). These discretizations result in fully implicit schemes of variable order in both spatial and temporal directions. The popularity of space-time DG methods has increased in recent years and entropy stable space-time DG-SEM have been constructed for conservation laws, making them interesting for these applications.

The size of the nonlinear system resulting from differential equations discretized with space-time DG-SEM is dependent on the order of the method, and the corresponding Jacobian is of block form with dense blocks. Thus, the problem arises to efficiently solve these huge nonlinear systems with regards to CPU time as well as memory consumption. The lack of good solvers for three-dimensional DG applications has been identified as one of the major obstacles before high order methods can be adapted for industrial applications.

It has been proven that DG-SEM in time and Lobatto IIIC Runge-Kutta methods are equivalent, in that both methods lead to the same discrete solution. This allows to implement space-time DG-SEM in two ways: Either as a full space-time system or by decoupling the temporal elements and using implicit time-stepping with Lobatto IIIC methods. We compare theoretical properties and discuss practical aspects of the respective implementations.

When considering the full space-time system, multigrid can be used as solver. We analyze this solver with the local Fourier analysis, which gives more insight into the efficiency of the space-time multigrid method.

The other option is to decouple the temporal elements and use implicit Runge-Kutta time-stepping methods. We suggest to use Jacobian-free Newton-Krylov (JFNK) solvers since they are advantageous memory-wise. An efficient preconditioner for the Krylov sub-solver is needed to improve the convergence speed. However, we want to avoid constructing or storing the Jacobian, otherwise the favorable memory consumption of the JFNK approach would be obsolete. We present a preconditioner based on an auxiliary first order finite volume replacement operator. Based on the replacement operator we construct an agglomeration multigrid preconditioner with efficient smoothers using pseudo time integrators. Then only the Jacobian of the replacement operator needs to be constructed and the DG method is still Jacobian-free. Numerical experiments for hyperbolic test problems as the advection, advection-diffusion and Euler equations in several dimensions demonstrate the potential of the new approach.

Key words
Discontinuous Galerkin Method, Spectral Element Method, Finite Volume Method, Implicit Schemes, Preconditioner, Multigrid, Space-Time, Lobatto IIIC Method, Local Fourier Analysis

Classification system and/or index terms (if any)

Supplementary bibliographical information

Language
English

ISSN and key title
1404-0034 Doctoral theses in mathematical sciences

ISBN
978-91-8039-154-2 (print)
978-91-8039-153-5 (pdf)

Recipient's notes

Number of pages
227

Price

Security classification

Signature

Date January 20, 2022

Efficient Solvers for Space-Time Discontinuous Galerkin Spectral Element Methods

by Lea Miko Versbach

LUND
UNIVERSITY

Cover illustration front: Welle (© Matthias Versbach)

Funding information: The thesis work was financially supported by the Swedish Research Council, grant number 2015-04133.

Numerical Analysis
Centre for Mathematical Sciences
Lund University
Box 118
SE-221 00 Lund
Sweden

www.maths.lu.se

Doctoral Thesis in Mathematical Sciences 2022:1
ISSN: 1404-0034

ISBN: 978-91-8039-154-2 (print)
ISBN: 978-91-8039-153-5 (pdf)
LUNFNA-1010-2022

Printed in Sweden by Media-Tryck, Lund University, Lund 2022

Media-Tryck is a Nordic Swan Ecolabel
certified provider of printed material.
Read more about our environmental
work at www.mediatryck.lu.se

MADE IN SWEDEN

It always seems impossible until it's done.
Nelson Mandela

Abstract

In this thesis we study efficient solvers for space-time discontinuous Galerkin spectral element methods (DG-SEM). These discretizations result in fully implicit schemes of variable order in both spatial and temporal directions. The popularity of space-time DG methods has increased in recent years and entropy stable space-time DG-SEM have been constructed for conservation laws, making them interesting for these applications.

The size of the nonlinear system resulting from differential equations discretized with space-time DG-SEM is dependent on the order of the method, and the corresponding Jacobian is of block form with dense blocks. Thus, the problem arises to efficiently solve these huge nonlinear systems with regards to CPU time as well as memory consumption. The lack of good solvers for three-dimensional DG applications has been identified as one of the major obstacles before high order methods can be adapted for industrial applications.

It has been proven that DG-SEM in time and Lobatto IIIC Runge-Kutta methods are equivalent, in that both methods lead to the same discrete solution. This allows to implement space-time DG-SEM in two ways: Either as a full space-time system or by decoupling the temporal elements and using implicit time-stepping with Lobatto IIIC methods. We compare theoretical properties and discuss practical aspects of the respective implementations.

When considering the full space-time system, multigrid can be used as solver. We analyze this solver with the local Fourier analysis, which gives more insight into the efficiency of the space-time multigrid method.

The other option is to decouple the temporal elements and use implicit Runge-Kutta time-stepping methods. We suggest to use Jacobian-free Newton-Krylov (JFNK) solvers since they are advantageous memory-wise. An efficient preconditioner for the Krylov sub-solver is needed to improve the convergence speed. However, we want to avoid constructing or storing the Jacobian, otherwise the favorable memory consumption of

the JFNK approach would be obsolete. We present a preconditioner based on an auxiliary first order finite volume replacement operator. Based on the replacement operator we construct an agglomeration multigrid preconditioner with efficient smoothers using pseudo time integrators. Then only the Jacobian of the replacement operator needs to be constructed and the DG method is still Jacobian-free. Numerical experiments for hyperbolic test problems as the advection, advection-diffusion and Euler equations in several dimensions demonstrate the potential of the new approach.

Popular Science Summary

Weather forecasting has been of interest for millennia and will be of importance in the future with increasing extreme weather conditions. Modern forecasting techniques were developed in the 20th century and have been continuously improved since then. Weather forecast is one important example of an application in *computational fluid dynamics (CFD)*, where problems involving fluid flow are analyzed and solved. Other examples are the aerodynamic design of airplanes and cars, fire development in tunnels and blood flow simulations. CFD is based on *scientific computing*, where complex problems are studied and solved with the help of advanced computing techniques.

Even though mathematical models of fluid systems have been developed almost 200 years ago and CFD calculations have been steadily improved over the last 100 years, accurate fluid simulations are still a challenging research topic. Scientists are continuously increasing the accuracy of CFD simulations. The difference in the scale of magnitude in real world applications is a big challenge: A weather forecast for a region is done on the scale of kilometers, while windstorms can consist of wind shears and eddies with sizes ranging in diameter from centimeters to hundreds of kilometers. The goal of this thesis is to contribute to the improvement of CFD simulations by constructing efficient solvers for CDF problems.

To express CFD problems mathematically, *partial differential equations (PDEs)* are very often used. With the help of PDEs we can describe the change of a quantity in a domain over time, e.g. the temperature in a city over a day. These PDEs are often very complex and need to be solved using computers and *numerical methods*, which are approximation techniques for solving mathematical problems. It is an ongoing research topic in computational sciences to develop efficient numerical methods adapted to state of the art computing devices, which have increased computing power and require methods where many calculations are carried out in parallel. This is especially of interest when large problems can be divided into smaller ones which can be solved simultaneously.

The first step to solve PDEs with numerical methods is to choose a *discretization method*. These are based on dividing the problem domain into pieces which are small enough to accurately approximate important quantities of the problem as e.g. eddies. Different discretization methods exist, having specific advantages and varying accuracy. Methods with high accuracy are called *high order methods* and are used in this thesis. These are of advantage for problems with e.g. turbulence. It is an open problem to construct efficient solvers to be applicable to large real world problems.

Discretizing PDEs results in a system of linear equations which need to be solved with an efficient *iterative method*. In this context, we are interested in efficiency w.r.t. the number of iterations and the computing time while at the same time giving an accurate solution. To achieve a precise weather forecast, a lot of measurements need to be taken into account in the calculations. More measurements and in consequence more data increase the accuracy of the mathematical model, but also the number of unknowns in the equation system. It is common to have more than hundreds of million unknowns in typical CFD applications. Thus, very accurate calculations can take several weeks to run. This is of course not a reasonable option for weather forecasts and emphasizes the need for fast solvers for these equation systems.

The solution to this problem are *preconditioners*, which allow to transform the equation system to be solved into one more suitable for the iterative method used. The construction of good preconditioners is not easy since they depend both on the PDE to be solved and the discretization method applied. One main focus of this thesis is the construction of good preconditioners. The idea presented here is to use a combination of low order numerical methods which are simpler to construct, but work well as preconditioners given several other computational and numerical constraints we restrict ourselves to.

The other main topic of this thesis are so-called *space-time* problems. The traditional way to solve time dependent PDEs numerically is the following: for one specific time point, e.g. each second, the solution is calculated before everything is repeated for the next time point. The idea of space-time methods is to consider all time steps simultaneously in the calculations while still following the causality principle that a solution later in time is depending on a solution earlier in time. The main difference to the traditional methods is that the equation system becomes even larger. The interest for space-time methods has increased with the changing development in computer architecture. State of the art processors do not become faster any longer, instead the number of processors increases. This motivates to parallelize calculations even more. We compare different ways to implement space-time methods and present a method to analyze space-time solvers using established frameworks.

Scientific Publications

This thesis is based on the following publications, referred to by their Roman numerals. All papers are reproduced with the permission of the respective publisher.

I **Finite volume based multigrid preconditioners for discontinuous Galerkin methods**

Lea M. Versbach, Philipp Birken, Gregor J. Gassner
Proceedings in Applied Mathematics and Mechanics (PAMM), Vol 18(1) (2018)

II **Subcell finite volume multigrid preconditioning for high-order discontinuous Galerkin methods**

Philipp Birken, Gregor J. Gassner, **Lea M. Versbach**
International Journal of Computational Fluid Dynamics, 33:9, 353-361 (2019)

III **An Finite Volume Based Multigrid Preconditioner for DG-SEM for Convection-Diffusion**

Johannes Kasimir, **Lea M. Versbach**, Philipp Birken, Gregor J. Gassner, Robert Klöfkorn
WCCM-ECCOMAS2020 (2021)

IV **Theoretical and Practical Aspects of Space-Time DG-SEM Implementations**

Lea M. Versbach, Viktor Linders, Robert Klöfkorn, Philipp Birken
E-print: arXiv: 2201.05800 [math.NA] (2022)

v **Local Fourier Analysis of a Space-Time Multigrid Method for DG-SEM for the Linear Advection Equation**

Lea M. Versbach, Philipp Birken, Viktor Linders, Gregor J. Gassner
submitted to Electronic Transactions on Numerical Analysis (ETNA), E-print: arXiv: 2112.03115 [math.NA] (2021)

Author Contributions

The author's contributions to each paper is listed below.

Paper I: Finite volume based multigrid preconditioners for discontinuous Galerkin methods

I did the implementation, produced the numerical results and did all of the writing, with input from the co-authors.

Paper II: Subcell finite volume multigrid preconditioning for high-order discontinuous Galerkin methods

I contributed to the development of the presented method. I did all of the implementation and produced the numerical results. I did all of the writing, with input from the co-authors.

Paper III: An Finite Volume Based Multigrid Preconditioner for DG-SEM for Convection-Diffusion

I contributed to the development of the presented method, chose and generated the final numerical results. The implementation was done by J. Kasimir with the help of R. Klöfkorn and my supervision. I did most of the writing, with help from the co-authors.

Paper IV: Theoretical and Practical Aspects of Space-Time DG-SEM Implementations

I contributed to the idea of the article. I did the implementation with the help of R. Klöfkorn and extended an existing `Assimulo` code. All authors contributed to the writing of the article.

Paper V: Local Fourier Analysis of a Space-Time Multigrid Method for DG-SEM for the Linear Advection Equation

I came up with the idea of the article. I performed the analysis, proofs and numerical experiments. All coding was done by me building upon codes from Paper III. I did all of the writing, with input from the co-authors.

Acknowledgements

This dissertation thesis would not have been possible without the help of others and I want to thank all those that supported me along the way.

First of all, thanks to my main supervisor Philipp Birken for your support and guidance. Thanks for many discussions and for being a never ending source of correction suggestions, I have learned a lot in the last years. I also want to thank my co-supervisor Gregor Gassner for the good collaboration, for always being very helpful and positive.

Next I want to thank the Numerical Analysis group, both the current members as well as the ones who have left the group during my years at Lund University. I want especially to mention Alexandros Sopasakis, Carmen Arévalo, Gustaf Söderlind, Monika Eisenmann and Tony Stillfjord for good collaborations during my teaching duties. Without the other PhD students in our group life would not have been the same. Thank you for a nice time Azahar Monge, Fatemeh Mohammadi, Johannes Kasimir, Niklas Kotarsky and Peter Meisrimel. I would like to offer my special thanks to Claus Führer, who supported me when I decided to do a PhD and always had motivating words and advices during all the ups and downs. I also wish to acknowledge the help provided by Viktor Linders, thanks for many discussions. Robert Klöfkorn helped me with many computer issues and deserves a big thank you for that. This thesis would not exist in this form without your help. Thanks for giving me feedback on parts of this thesis: Johannes Kasimir, Pelle Pettersson, Stéphane Gaudreault, Viktor Linders and Vincent Magnoux. And thanks Gabrielle Flood for for helping me with some LaTeX issues.

Next, I want to thank my fellow PhD students and colleagues at the Centre for Mathematical Sciences for a lot of nice lunch and fika breaks. These were always funny and we discussed and learned a lot of nerdy things. Thanks to the group women in mathematical sciences for organizing interesting events. And of course I want to thank all my friends outside the department for distracting me from work-related stuff.

Being a PhD student in Sweden is a privilege since students actually have rights. It was

a pleasure to continue fighting for our rights as a secretary in the Science Doctoral Student Council (NDR) as well as a representative in the faculty board. Some of my NDR colleagues have become friends and it was always a pleasure working with you!

Finally, I want to thank my family for supporting me all the way and listening when I needed to talk, even though you might not always understand all the details. Without you I would not be where I am today. Danke! Special thanks to my father for putting so much effort into the cover of this thesis.

Last but not least my biggest thanks to Niclas. You have been on my side throughout this journey. You have suffered with me in the stressful times, always reminded me to take breaks and reassured me a million times that everything will be fine in the end. Tack, jag tror inte att jag hade klarat det här utan dig!

Funding

The work in this thesis is made possible due to the support of the Swedish Research Council, grant number 2015-04133.

Abbreviations

AMG	Algebraic Multigrid Methods
AUSM	Advection Upstream Splitting Method
BiCGSTAB	Biconjugate Gradient Stabilized
CFD	Computational Fluid Dynamics
CFL	Courant–Friedrichs–Lewy
CG	Conjugate Gradient
CPU	Central Processing Unit
DFR	Direct Flux Reconstruction
DG	Discontinuous Galerkin
DG-SEM	Discontinuous Galerkin Spectral Element Method
DOF	Degree of Freedom
DUNE	Distributed and Unified Numerics Environment
ENO	Essentially Non-Oscillatory
FAS	Full Approximation Scheme
FE	Finite Element
FGMRES	Flexible Generalized Minimal Residual Method
FR	Flux Reconstruction
FV	Finite Volume
GEF	GEM en Éléments Finis
GEM	Global Environmental Multiscale Model
GMRES	Generalized Minimal Residual Method
IBP	Integration by Parts
IVP	Initial Value Problem
JFNK	Jacobian-Free Newton-Krylov
LES	Large Eddy Simulation
LFA	Local Fourier Analysis
LGL	Legendre-Gauss-Lobatto
LHS	Left Hand Side
LoDG	Lobatto IIIC Discontinuous Galerkin
MG	Multigrid
MOL	Method of Lines
MUSCL	Monotonic Upstream-Centered Scheme for Conservation Laws
NSR	Nonsymmetric Restriction Aggregation
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation

RAM	Random-Access Memory
RANS	Reynolds-Averaged Navier–Stokes Equations
RHS	Right Hand Side
RK	Runge-Kutta
SAT	Simultaneous Approximation Term
SBP	Summation by Parts
SBP-SAT	Summation by Parts Simultaneous Approximation Term
SD	Spectral Difference
SEM	Spectral Element Method
SGS	Symmetric Gauss-Seidel
SOR	Successive Over-Relaxation
ST	Space-Time
STDG	Space-Time Discontinuous Galerkin
UFL	Unified Form Language
WENO	Weighted Essentially Non-Oscillatory

Contents

Chapter 1

Introduction

Computational fluid dynamics (CFD) simulations have become central in the design of for example next generation aerodynamic jet engines, air frames and wind turbines, simulation of tunnel fires and atmospheric boundary layers and many more. These simulations are based on numerical models for turbulent and wall bounded flows with complex geometries called large eddy simulation (LES). The resulting complex multi-scale problems may have on the order of 100 million unknowns, thus a fast low memory parallel solver is needed. Mathematically, CFD problems are modeled using the Navier-Stokes equations or simplified versions of them.

Discontinuous Galerkin (DG) methods provide numerical discretizations of differential equations based on element-wise polynomial approximations. Since it has been shown that low-order schemes can contribute dramatically to the dissipation of eddies [82] and high resolution is necessary for LES, high order methods are of interest. High order DG methods are constructed by increasing the order of the element-wise polynomial approximation. Moreover, DG methods allow for discontinuous element interfaces, which are connected using numerical fluxes [64, 80]. This is of advantage when simulating problems which contain shocks. The DG discretization results in local computations on the elements which are very dense, and the degrees of freedom (DOFs) are coupled across neighboring faces. Due to their block structure, DG methods are very well suited for domain-decomposition based parallelization [61, 137].

A specific DG variant is the discontinuous Galerkin spectral element method (DG-SEM), e.g. [81]. It is based on a Lagrange type nodal polynomial basis with Legendre-Gauss-Lobatto (LGL) nodes, which are collocated with the discrete integration of the weak form of the differential equation. The resulting DG-SEM operators satisfy the summation by

parts (SBP) property [47], which is the discrete analogue to integration by parts (IBP) and key to construct discretely entropy stable and kinetic energy preserving methods [143]. This ensures that the numerical scheme obeys the second law of thermodynamics since it provides a bound on the mathematical entropy at any time for given initial and boundary conditions.

The DG-SEM applied to the spatial direction results in a big system of stiff ordinary differential equations (ODEs) which are solved using numerical time integration methods. These can be separated into explicit and implicit schemes, where the latter ones incorporate unknown data and therefore require solving a (non)linear equation system. Explicit ODE solvers do not depend on unknown data, but require a condition on the time step to guarantee stability of the numerical solver. This condition becomes very restrictive for stiff problems, for example for the compressible Navier-Stokes equations for wall bounded and low Mach number flows. To avoid stability restrictions on the time step, it is therefore of advantage to consider implicit time integrators. For these methods, the time step only needs to be chosen error based. This comes at the cost of an increased computational effort when solving the resulting large nonlinear equation systems.

Two important techniques to construct implicit time integrators are considered in this thesis. Firstly, the method of lines (MOL) ansatz, which is based on applying implicit Runge-Kutta time-stepping methods to the ODE resulting from the spatial discretization. Secondly, a fully discrete ansatz which is the result of simultaneously discretizing space and time, resulting in a space-time DG method.

Space-time methods have gained increased attention in recent years, mostly due to the possibility to parallelize the temporal direction as well [45]. With state of the art computer architectures reaching a clock speed limit, and a trend towards more rather than faster processors, parallelization becomes even more important. Moreover, entropy stable space-time DG-SEMs for hyperbolic conservation laws have been constructed [44].

Efficiency of DG discretizations with implicit time-stepping methods can only be achieved when the resulting large nonlinear equation systems are solved cheaply w.r.t. memory consumption and central processing unit (CPU) time. Solvers for linear and nonlinear equation systems are severely lacking for three-dimensional DG applications and are one of four major obstacles that need to be solved before industry might adopt high order methods [138]. However, it is an ongoing research topic how to implement and solve these fully implicit methods efficiently. Possible solvers are full-approximation schemes (FAS), multigrid (MG) and preconditioned Jacobian-free Newton-Krylov (JFNK) methods [78].

The JFNK technology is interesting in the sense of memory minimization. JFNK meth-

ods are numerical methods to solve non-linear problems with Newton's method using Krylov subspace solvers for the linear subproblem. DG systems have a dense block structure with blocks coupled via the faces. The number of unknowns per element increases dramatically with increasing polynomial degree and dimension, leading to large dense Jacobian blocks [12, 15]. For a d-dimensional problem, the block size of a finite volume (FV) method is $d + 2$, whereas for a DG-SEM with p-th degree polynomials it is $(d + 2)(p + 1)^d$. For degree two in three dimensions this already is 135. If a preconditioner requires the storage of the Jacobian of the DG system, the favorable memory consumption of the JFNK approach is obsolete and the method is not fully Jacobian-free any longer.

Thus, efficient Jacobian-free preconditioners need to be constructed in order to have competitive JFNK solvers for DG discretizations. One option to construct preconditioners for the Krylov subproblem are multigrid methods [14]. These are a class of iterative methods originally designed to solve equation systems arising from discretized differential equations. Multigrid methods are based on a hierarchy of grids for the discretization and so-called smoothers that damp the high frequency error parts in a few iterations [124]. Multigrid methods are standard in many academical and industrial codes due to their textbook multigrid efficiency. This means only a few iterations are needed to solve a problem. Even though textbook multigrid efficiency is generally not attained in fluid problems with DG discretizations, multigrid methods provide efficient solvers for these. This motivates to use well-designed MG methods as preconditioners.

1 State of the Art

Space-time DG methods have been considered for hyperbolic problems in [33, 44], for advection-diffusion problems in [76, 116], for the Euler equations of gas dynamic in [128, 130] and for nonlinear wave equations in [129]. Multigrid solvers have been analyzed in the DG context with block smoothers for convection-diffusion problems in [50, 76, 134] and for elliptic problems in [56, 57]. Space-time MG methods have been studied mostly for parabolic problems, see for instance [36, 43, 46]. MG solvers for different problems have been investigated in [46, 56, 62, 76, 105]. Analysis of space-time MG algorithms for DG discretizations of advection dominated flows has been quite limited but can be found for the advection-diffusion equation or linearized versions of the compressible Euler equations in [127, 125] and for generalized diffusion problems in [43].

Classical block Jacobi preconditioners have been studied for inviscid and viscous incompressible flows governed by the Euler and Navier–Stokes equations in [8], for direct numerical simulations/implicit LES of turbulent flows in [42], and approximate tensor-

products of exact block Jacobi preconditioners for the advection and Euler and Navier–Stokes equations have been studied in [99]. Two-level preconditioners with incomplete LU smoothers have been considered for the compressible Navier-Stokes equations in [101, 102] and nonlinear preconditioners based on dual time-stepping for viscous flow around cylinder and NACA0012 airfoil have been studied in [14].

JFNK solvers for implicit DG-SEM have been studied for parallel time adaptive higher order solvers with block Jacobi and reduction of the off-block order symmetric Gauss-Seidel preconditioners for the three-dimensional time dependent Navier–Stokes equations in [15], for MG preconditioners for the Reynolds-averaged Navier–Stokes equations equations in [16] and for block Jacobi preconditioners for the compressible Navier-Stokes equations in [135]. Algebraic multigrid preconditioners for Newton-Krylov solvers of the fully implicit variational multiscale finite element resistive magnetohydrodynamics formulation have been studied in [88].

Preconditioned space-time DG methods have been studied by several authors. For instance tensor-product preconditioners for space-time DG-SEM matrix-free Newton-Krylov for the compressible Navier-Stokes equations in [32], lower-upper symmetric Gauss–Seidel preconditioned generalized minimal residual method (GMRES) solvers for implicit space-time DG solver for the compressible Navier-Stokes equations in [144] and MG preconditioning with block incomplete LU(0)-preconditioned GMRES smoothers for a space-time DG-SEM solver in [41].

2 Goals of the Thesis

In this thesis we discuss efficient solvers for space-time DG-SEM. The two main topics are multigrid based preconditioners for JFNK solvers as well as space-time DG-SEM and the resulting challenges w.r.t. implementation and solvers.

It has been shown that the temporal DG-SEM is equivalent to fully implicit Lobatto IIIC Runge-Kutta methods in the sense that they lead to the same discrete solution. This opens two possibilities to implement space-time DG-SEM: either as a $d+1$-dimensional problem or as a spatial DG-SEM combined with Lobatto IIIC Runge-Kutta time-stepping. However, in practice there are several differences between these two approaches, including the terminology used to describe them, the interaction with nonlinear solvers and the structure of the resulting software. We compare the advantages and disadvantages of these implementations using the Distributed and Unified Numerics Environment (DUNE) [30]. It is an open research question how to solve the resulting systems efficiently.

One possibility are space-time multigrid methods. In order to gain more insight, we

perform a local Fourier analysis (LFA), an important tool to analyze the quality of MG methods [19]. With the help of the LFA we can predict convergence rates of the MG solver. The results show that multigrid is a good solver for the space-time DG-SEM as well as a promising basis to construct space-time preconditioners.

We also present a novel idea for the construction of preconditioners for JFNK solvers, while retaining the low memory use. A visualization of our concept can be seen in Figure 1.1. The starting point is a partial differential equation (PDE) which we discretize with an implicit DG-SEM. We then solve the resulting algebraic system with a JFNK method. In order to improve the convergence speed of the GMRES sub-solver, we construct a preconditioner using MG. The core idea is to base this on a lower order FV replacement operator to avoid constructing the Jacobian of the DG discretization. Moreover, this replacement operator allows to use available knowledge about fast multigrid methods for FV discretizations on block structured meshes.

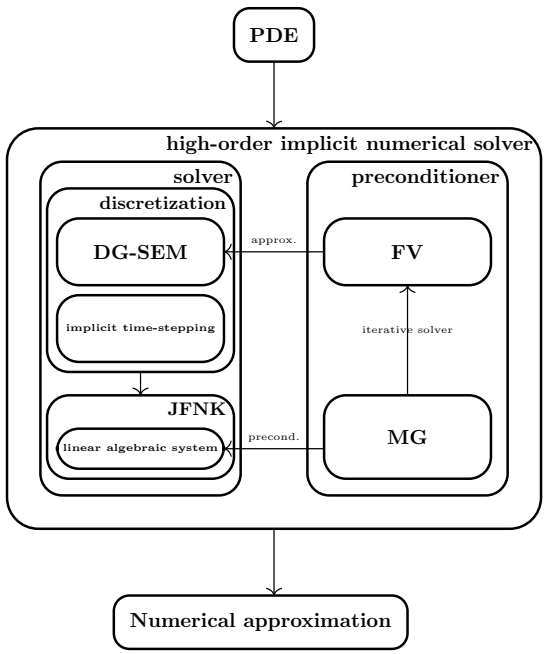


Figure 1.1: Work flow to construct a multigrid based preconditioner using a finite volume replacement operator to solve implicit DG-SEM.

The fundamental question is how to construct a preconditioner for the JFNK solver without constructing the Jacobian of the DG-SEM discretization. Our idea is to make use of a simplified replacement operator for the DG operator. One could for instance

choose a different polynomial order in the element to generate a replacement operator as presented in [15, 38, 102]. However, we want to keep the number of degrees of freedom in the replacement operator the same. This can be achieved by introducing a subcell grid in each DG element. On the subcell-element grid, the simplest replacement operator is a first order FV discretization. This choice is motivated by the equivalence of DG-SEM and specific high order FV discretizations [40]. In the resulting approximate Jacobian, we only have $(d+2)(p+1)(2d+1)$ entries [12]. As smoother we use state of the art low memory pseudo time-stepping smoothers from [16] which have been shown to work very efficient for this multigrid setup.

This dissertation thesis is a continuation of the Licentiate thesis [136], where the idea for the preconditioner has been introduced and the local Fourier analysis has been presented.

3 Organization of the Thesis

In Chapter 2 we present the DG-SEM combined with implicit time-stepping methods and in Chapter 3 the space-time DG-SEM. We give an overview over Newton-Krylov solvers for the resulting nonlinear equation systems in Chapter 4. In Chapter 5 we discuss multigrid methods for linear systems and our idea to construct efficient multigrid preconditioners for implicit DG solvers using FV replacement operators.

In Chapter 6 numerical examples are presented to show the efficiency of the suggested preconditioner. More results can be found in Paper I, II and III. Space-time DG-SEM is the topic of Paper IV and Paper V, which are also summarized in Chapter 6. A comparison of theoretical and practical aspects of two different approaches for the formulation and implementation of space-time DG-SEM is discussed in Paper IV and a space-time multigrid solver is analyzed with the help of the local Fourier analysis in Paper V.

Conclusions can be found in Chapter 7.

All publications are presented at the end of the thesis.

Chapter 2

Discontinuous Galerkin Methods

We are interested in numerically approximating solutions of mathematical models based on conservation laws, where the amount of a measurable quantity, as for example mass, momentum or energy, is only changed by transport over the boundaries of the system or by internal processes. Important models in fluid mechanics are the linear advection equation, the Euler equations, the shallow water equations and the Navier-Stokes equations.

A conservation law is mathematically expressed as

$$\begin{aligned} &\mathbf{u}_t + \nabla \cdot \mathbf{f}(\mathbf{u}, \nabla \mathbf{u}) = \mathbf{g}(\mathbf{u}), \ (\mathbf{x}, t) \in \Omega \times (0, T], \\ &\mathbf{u}(\mathbf{x}, t) = \mathbf{b}(t), && \mathbf{x} \in \partial\Omega, \ t \in (0, T], \\ &\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}), && \mathbf{x} \in \Omega, \end{aligned} \tag{2.1}$$

with space dimension d, typically $d = 1, 2, 3$, spatial variables $\mathbf{x} = (x_1, \ldots, x_d)$, domain $\Omega \subset \mathbb{R}^d$, physical flux function $\mathbf{f}$, source term $\mathbf{g}$, boundary condition $\mathbf{b}$ and initial condition $\mathbf{u}_0$. In this chapter we present discretization techniques for problems of the form (2.1).

Discontinuous Galerkin (DG) methods were introduced in 1973 by Reed and Hill to solve the hyperbolic neutron transport equation [104]. DG schemes can be interpreted as a combination of finite element (FE) and finite volume (FV) methods since they are derived from the weak form of PDEs with basis functions on elements similar to the FE ansatz with elements connected at their discontinuous boundaries by numerical fluxes as for the FV method. While standard FE methods assume continuity on the interfaces between two neighboring elements, DG methods allow for discontinuities at the interfaces. This is of advantage when solving hyperbolic conservation laws, as their solutions

might have jumps and discontinuities, and assuming global continuity on the considered physical domain would cause difficulties when constructing a numerical solution. Therefore, discontinuous Galerkin methods are more suitable for these problems. As in the FV method, elements are coupled by defining numerical fluxes which mimic physical properties of the underlying PDE by taking two adjacent states and a normal vector.

One of the main advantages of DG methods is that the order of accuracy can be quite easily improved by increasing the number of nodes within each element. DG methods are an interesting alternative to low-order FV methods which are mostly used in today's computational fluid dynamics (CFD) industrial codes [84], since higher order discretizations reduce dispersion and dissipation errors. Nowadays, a huge variety of different DG methods exist and a standard has only been partly established. Moreover, there are still a number of serious issues with DG methods that need to be solved before they are feasible for industrial applications [138], one of them being the lack of fast, efficient solvers for three-dimensional problems. An overview over many research contributions on DG methods applied to fluid dynamics can be found in [140]. Good introductions to the DG methods considered in this thesis can be found in [59] and [80] as well as the references therein.

In this chapter we discuss a method of lines approach, where we first discretize the problem in space to transform it into a system of ordinary differential equations (ODEs), which can then be discretized in time. We give a short overview over FV methods and numerical fluxes and present the choices that lead to a discontinuous Galerkin spectral element method (DG-SEM). Next, we discuss properties of DG methods, as stability, summation by parts, entropy boundaries and convergence order. We finish this chapter by presenting time integration schemes to solve the ordinary differential equation resulting from the spatial discretization.

1 Space Discretization

This section serves as an introduction to spatial discretizations based on FV and DG methods for conservation laws in two dimensions.

1.1 Finite Volume Discretization

The easiest DG discretization is of first order with constant element averages, i.e. polynomial approximations of degree $p_x = 0$ on each element, usually known as finite volume methods. These methods are conservative and easy to implement. In this section we only

give a brief introduction to FV methods, which we use as basis for constructing multigrid preconditioners in Chapter 5. For a more detailed presentation and discussion of these methods we refer to [86, 87].

The starting point for an FV discretization is the integral form of the conservation law (2.1). The computational domain $\Omega \subset \mathbb{R}^2$, which we assume to be rectangular for simplicity, is divided into $N^x \times N^y$ non-overlapping elements

$$\Omega_{n,m} = [x_{n-1/2}, x_{n+1/2}] \times [y_{m-1/2}, y_{m+1/2}], \quad n = 1, \ldots, N^x,\ m = 1, \ldots, N^y,$$

with nodes located at the center (x_n, y_m) of the respective element, see Figure 2.1. These elements are usually called volumes in the FV context. Then on each volume the integral form of (2.1) reads

$$\int_{\Omega_{n,m}} \mathbf{u}_t \, \mathrm{d}\Omega_{n,m} + \int_{\Omega_{n,m}} \nabla \cdot \mathbf{f}(\mathbf{u}, \nabla \mathbf{u}) \, \mathrm{d}\Omega_{n,m} = \mathbf{0}.$$

The divergence theorem yields

$$\int_{\Omega_{n,m}} \mathbf{u}_t \, \mathrm{d}\Omega_{n,m} + \sum_{edges} \int_{\partial\Omega_{n,m}} \mathbf{f}(\mathbf{u}, \nabla \mathbf{u}) \cdot \mathbf{n} \, \mathrm{d}\mathbf{s} = \mathbf{0}.$$

The idea of FV methods is to approximate mean values for the control volumes,

$$\mathbf{u}_{nm}(t) := \frac{1}{|\Omega_{n,m}|} \int_{\Omega_{n,m}} \mathbf{u}(\mathbf{x}, t) \, \mathrm{d}\Omega_{n,m}, \quad n = 1, \ldots, N^x,\ m = 1, \ldots, N^y,$$

on each volume $\Omega_{n,m}$ with width $|\Omega_{n,m}|$. Assuming that the volumes do not change with time, we obtain the element-wise evolution equation for the mean value

$$\dot{\mathbf{u}}_{nm}(t) + \frac{1}{|\Omega_{n,m}|} \sum_{edges} \int_{\partial\Omega_{n,m}} \mathbf{f}(\mathbf{u}, \nabla \mathbf{u}) \cdot \mathbf{n} \, \mathrm{d}\mathbf{s} = \mathbf{0},$$

for $n = 1, \ldots, N^x,\ m = 1, \ldots, N^y$. The approximation with volume averages leads to discontinuities at the volume interfaces. This results in fluxes which are not defined on the volume interfaces. Thus, numerical flux functions $\mathbf{f}^{\mathrm{N}}$ are introduced which take states from both side of the edges and approximate the exact flux based on these states. We discuss these in more details in the next section.

The FV method as presented is based on piecewise constant approximations and can thus be at most of first order. However, for many practical applications more accurate methods are necessary. One possibility to increase the order of convergence is to use a higher order

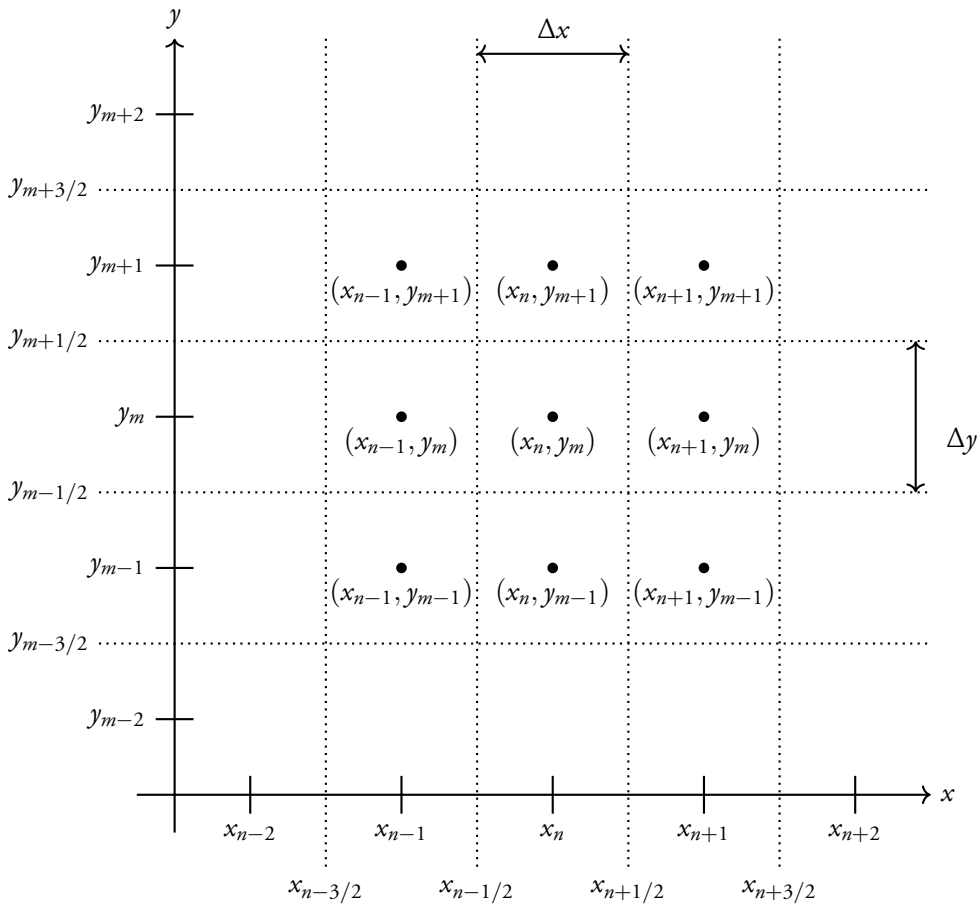


Figure 2.1: Finite volume grid in two dimensions.

DG scheme. Another option is linear reconstruction, where a piecewise constant approximation is used to reconstruct a piecewise linear approximation that has the same integral mean. This is standard in many industrial codes. The MUSCL (Monotonic Upstream-Centered Scheme for Conservation Laws) scheme [132] is based on this reconstruction technique. Other higher-order FV schemes are ENO (Essentially Non-Oscillatory) [55] and WENO (Weighted Essentially Non-Oscillatory) [91]. We refer to [6] for an overview over these schemes. High-order FV schemes are often used in practice, but require large stencils which lead to poor parallelization since both the cell coupling and the amount of information exchanges increases.

1.2 Numerical Fluxes

To couple the element-wise problems, the boundary terms need to be connected with their neighboring elements. This is done with the help of numerical flux functions $\mathbf{f}^{\mathrm{N}}$. Let us denote the solution value on the edge inside the element by $\mathbf{u}^-$ and the value outside by $\mathbf{u}^+$. The latter one is either given by the numerical solution from the neighboring element or the boundary condition from the physical problem. Moreover, the numerical flux depends on the normal vector $\mathbf{n}$, pointing from the interior value to the exterior value and thus reading $\mathbf{f}^{\mathrm{N}}(\mathbf{u}^-, \mathbf{u}^+; \mathbf{n})$. To define a reasonable numerical flux, the physical properties of the underlying PDE need to be taken into account.

A numerical flux needs to be *consistent*, that is Lipschitz continuous in the first two arguments and

$$\mathbf{f}^{\mathrm{N}}(\mathbf{u}, \mathbf{u}; \mathbf{n}) = \mathbf{f}(\mathbf{u}),$$

as well as *conservative*

$$\mathbf{f}^{\mathrm{N}}(\mathbf{u}^-, \mathbf{u}^+; \mathbf{n}) = \mathbf{f}^{\mathrm{N}}(\mathbf{u}^+, \mathbf{u}^-; -\mathbf{n}).$$

The consistency condition implies that the numerical flux has to reproduce the physical flux, and the finer the discretization, the better the approximation.

We present in the following numerical fluxes of relevance for this thesis, i.e. for hyperbolic problems. For other problems, the fluxes need to be split up into a sum of convective and diffusive fluxes. We refer to [87] for the discussion of other numerical fluxes for the most common problems.

Upwind Flux

The simplest flux is the *upwind flux*. This flux is suitable for advection problems since it takes the transportation direction into account. For a one-dimensional advection problem $\mathbf{f}(u) = au$ it is defined as

$$\mathbf{f}^{\mathrm{N}}_{\mathrm{up}}(u^-, u^+; \mathbf{n}) = \frac{au^- + au^+}{2} + \frac{|a|}{2}(\mathbf{n}u^- - \mathbf{n}u^+). \tag{2.2}$$

Moreover, the upwind flux is the natural choice for temporal discretizations from a physical point of view, since time is only moving forward.

Rusanov Flux

The *Rusanov flux*, also known as *local Lax-Friedrich flux*, is defined as

$$\mathbf{f}^{\mathrm{N}}_{\mathrm{LLF}}(\mathbf{u}^-, \mathbf{u}^+; \mathbf{n}) = \frac{\mathbf{f}(\mathbf{u}^-) + \mathbf{f}(\mathbf{u}^+)}{2} - \frac{\lambda_{\max}}{2}(\mathbf{u}^+ - \mathbf{u}^-),$$

with $\lambda_{\max} = \max\limits_{\mathbf{u}^-, \mathbf{u}^+} |\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{u})|$ the local maximum of the directional flux Jacobian, the maximum wave speed at the interface.

1.3 Boundary Conditions

On the domain boundaries, the numerical fluxes have to be constructed carefully to take care of boundary conditions of the PDE. Several options exist to handle boundary conditions. Either a solution value on the boundary can be defined or a flux function can be prescribed. Typical boundary conditions for conservation laws are fixed wall, inflow and outflow or periodic boundary conditions. We refer to [13] for a more detailed discussion.

FV Discretization of the Linear Advection Equation

To illustrate the FV method we consider the simplest conservation law, the one-dimensional linear advection equation

$$\begin{aligned} u_t + a u_x &= 0, \ (x, t) \in \Omega \times (0, T], \\ u(x, 0) &= u_0, \ u(x_L, t) = u(x_R, t), \end{aligned}$$

with $a > 0$. Since information travels from left to right with speed a, is is most natural to use an upwind flux $f^{\mathrm{N}}_{n-1/2} = u_{n-1}$. Then an equidistant FV discretization with mesh width Δx results in an evolution equation for the cell average u_n in volume n:

$$\dot{u}_n + \frac{a}{\Delta x}(u_n - u_{n-1}) = 0, \quad n = 1, \ldots, N^x.$$

With the vector $\underline{\mathbf{u}} = (u_1, \ldots, u_{N^x})^\top$ and the matrix

$$\underline{\mathbf{B}} = \begin{pmatrix} 1 & & & & -1 \\ -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{pmatrix} \in \mathbb{R}^{N^x \times N^x},$$

we obtain the system of ODEs

$$\underline{\mathbf{u}}_t + \frac{a}{\Delta x}\underline{\mathbf{B}\mathbf{u}} = \mathbf{0}. \tag{2.3}$$

1.4 Discontinuous Galerkin Method

In this section we present the DG discretization in space. For notational simplicity we derive the DG discretization for the two-dimensional case, i.e. $d = 2$ in (2.1). This can be easily extended to the three-dimensional case using a tensor product ansatz.

The first step is to divide the computational domain $\Omega \subset \mathbb{R}^2$ into $N^x \times N^y$ non-overlapping quadrilaterals $\Omega_{n,m}$ such that

$$\Omega = \bigcup_{n,m=1}^{N^x,N^y} \Omega_{n,m},$$

with $\Omega_{n,m} = [x_n, x_{n+1}] \times [y_m, y_{m+1}]$, i.e. a Cartesian grid. To derive the weak form of the problem, we multiply (2.1) on each element by a smooth test function $\psi(\mathbf{x})$ from some test space V and integrate over the spatial domain:

$$\int_{\Omega_{n,m}} \mathbf{u}_t \psi \,\mathrm{d}\Omega_{n,m} + \int_{\Omega_{n,m}} \nabla \cdot \mathbf{f} \psi \,\mathrm{d}\Omega_{n,m} = \mathbf{0} \quad \forall \psi \in V.$$

Integration by parts yields

$$\int_{\Omega_{n,m}} \mathbf{u}_t \psi \,\mathrm{d}\Omega_{n,m} + \int_{\partial\Omega_{n,m}} \mathbf{f} \cdot \mathbf{n} \psi \,\mathrm{d}\mathbf{s} - \int_{\Omega_{n,m}} \mathbf{f} \cdot \nabla \psi \,\mathrm{d}\Omega_{n,m} = \mathbf{0} \quad \forall \psi \in V. \tag{2.4}$$

In the next step, the solution $\mathbf{u}$ is approximated by a polynomial in each element. To obtain a specific DG method, a grid, a polynomial basis, a quadrature rule for the integrals in the scalar products, in particular the nodes and weights, and the approximation of the physical flux function $\mathbf{f}(\mathbf{u}, \nabla\mathbf{u})$ have to be chosen. There exist two different types of polynomials basis: modal and nodal bases. A *nodal basis* is defined by a number of nodes, which are used to define Lagrange polynomials. A *modal basis* is defined by functions only, e.g. monomials. Typically, a modal basis is hierarchical, therefore the term hierarchical basis is used by some authors.

In this thesis we focus on the DG-SEM on quadrilateral, respectively hexahedral cells in up to three dimensions. It is possible to have hanging nodes and curved boundaries. The DG-SEM has a Lagrangian basis function evaluated in the Legendre-Gauss-Lobatto (LGL) nodes, the Gaussian quadrature is based on the LGL nodes and weights as well, and an element-wise polynomial approximation is used for the flux function.

1.5 Mapping to Reference Elements

In order to simplify the implementation of DG methods, it is common to restrict the possible shapes of the elements. This allows to define the basis a priori on reference elements and to precompute as many terms as possible. The basis for a specific element is then obtained by a transformation from the reference element. We demonstrate this for a curved quadrangle as described in [73, 80].

Let (x, y) denote the coordinates in the physical space $\Omega_{n,m}$ and (ξ, η) the coordinates in the computational space $\hat{\Omega} := [-1, 1]^2$. Consider a mapping $\mathbf{X} : \hat{\Omega} \to \Omega_{n,m}$ such that

$$(x, y)^\top = \mathbf{X}(\xi, \eta).$$

The four corners of $\hat{\Omega}$ are denoted by $\mathcal{X}_i$ and the element curves are represented by polynomials Γ_i, $i = 1, 2, 3, 4$, see Figure 2.2. Using these polynomials we can construct a mapping between curves 1 and 3, denoted by $\mathbf{X}^{1,3}$, as well as between curves 2 and 4, denoted by $\mathbf{X}^{2,4}$:

$$\begin{aligned}
\mathbf{X}^{1,3}(\xi, \eta) &:= \frac{1-\eta}{2}\Gamma_1(\xi) + \frac{1+\eta}{2}\Gamma_3(\xi), \\
\mathbf{X}^{2,4}(\xi, \eta) &:= \frac{1-\xi}{2}\Gamma_4(\eta) + \frac{1+\xi}{2}\Gamma_2(\eta).
\end{aligned}$$

Note that simply adding $\mathbf{X}^{1,3}$ and $\mathbf{X}^{2,4}$ does not result in the mapping $\mathbf{X}$ since the corners do not match. This is solved by a correction term as derived in [80]:

$$\begin{aligned}
\mathbf{X}^{\text{corr}}(\xi, \eta) :=& \frac{1}{4}(\mathcal{X}_1(1-\xi)(1-\eta) + \mathcal{X}_2(1+\xi)(1-\eta) \\
&+ \mathcal{X}_3(1+\xi)(1+\eta) + \mathcal{X}_4(1-\xi)(1+\eta)).
\end{aligned}$$

Then the transformation

$$\mathbf{X} = \mathbf{X}^{1,3} + \mathbf{X}^{2,4} - \mathbf{X}^{\text{corr}},$$

defines the isoparametric mapping between the physical and the computational coordinates:

$$(x, y)^\top = \mathbf{X}(\xi, \eta). \tag{2.5}$$

An overview over similar mappings for different reference elements can be found in [73].

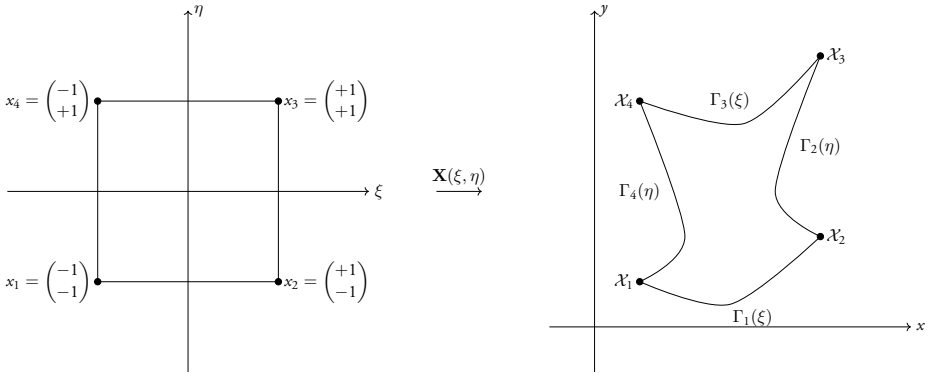


Figure 2.2: Mapping between the reference square $\hat{\Omega}$ and an arbitrary quadrilateral.

1.6 Transformation of Equations Under Mappings

The mapping (2.5) between the computational and the physical space transforms the equations themselves, as a result of the chain rule. In the following we describe a differential geometry approach to examine the transformation of equations under such mappings, see [80]. Since this approach extends well to three dimensions, let us introduce the coordinates $\mathbf{x} = (x, y, z) = (x_1, x_2, x_3)$ in the physical space and $\boldsymbol{\xi} = (\xi, \eta, \zeta) = (\xi_1, \xi_2, \xi_3)$ in the computational space. The mapping from the computational space to the physical space reads

$$\mathbf{x} = \mathbf{X}(\boldsymbol{\xi}).$$

Next, we introduce the covariant basis $\mathbf{a}_i$, $i = 1, 2, 3$, which varies along a coordinate line:

$$\mathbf{a}_i = \frac{\partial \mathbf{x}}{\partial \xi_i}.$$

Then the divergence w.r.t. the mapping is given by

$$\nabla \cdot \tilde{\mathbf{f}} = \frac{1}{\mathbf{J}} \sum_{i=1}^{3} \frac{\partial}{\partial \xi_i} \big(\mathbf{f} \cdot (\mathbf{a_j} \times \mathbf{a_k}) \big), \tag{2.6}$$

with $\tilde{\mathbf{f}}_i = \frac{1}{\mathbf{J}} (\mathbf{a_j} \times \mathbf{a_k}) \cdot \mathbf{f}(i, j, k)$ and $\mathbf{J} = \mathbf{a}_i \cdot (\mathbf{a}_j \times \mathbf{a}_k)$, (i, j, k) cyclic. The derivation of (2.6) can be found in [80]. The two-dimensional case follows with $\mathbf{a}_3 = (0, 0, 1)^\top$.

This gives the element-wise counterpart of equation (2.1) in the d-dimensional computational space $\hat{\Omega} \subset \mathbb{R}^d$:

$$\mathbf{J}\mathbf{u}_t + \nabla \cdot \tilde{\mathbf{f}}(\mathbf{u}, \nabla \mathbf{u}) = \mathbf{0}, \quad (\boldsymbol{\xi}, t) \in \hat{\Omega} \times (0, T].$$

In two dimensions, this reduces to

$$\mathbf{J}\mathbf{u}_t + \tilde{\mathbf{f}}^1_\xi(\mathbf{u}, \nabla\mathbf{u}) + \tilde{\mathbf{f}}^2_\eta(\mathbf{u}, \nabla\mathbf{u}) = \mathbf{0}, \quad (\xi, \eta, t) \in \hat{\Omega} \times (0, T],$$

with the Jacobian of the transformation $\mathbf{J} = x_\xi y_\eta - x_\eta y_\xi$ and the contravariant fluxes

$$\begin{aligned}\tilde{\mathbf{f}}^1(\mathbf{u}, \nabla\mathbf{u}) &= y_\eta \mathbf{f}^1(\mathbf{u}, \nabla\mathbf{u}) - x_\eta \mathbf{f}^2(\mathbf{u}, \nabla\mathbf{u}),\\ \tilde{\mathbf{f}}^1(\mathbf{u}, \nabla\mathbf{u}) &= -y_\xi \mathbf{f}^1(\mathbf{u}, \nabla\mathbf{u}) + x_\xi \mathbf{f}^2(\mathbf{u}, \nabla\mathbf{u}).\end{aligned}$$

In order to simplify the notation we drop in the following the tilde but work on the reference element $\hat{\Omega}$ unless stated otherwise.

1.7 Discontinuous Galerkin Spectral Element Method

From of the family of DG methods we focus on the DG-SEM in this thesis. The DG-SEM is based on element-wise approximations by nodal polynomials of degree p_x in the x-direction and degree p_y in the y-direction of the solution $\mathbf{u}(\xi, \eta, t)$

$$\mathbf{u}(\xi, \eta, t) \simeq \mathbf{u}^{\mathrm{P}}(\xi, \eta, t) = \sum_{j=1}^{N_x} \sum_{k=1}^{N_y} u_{jk}(t)\varphi_{jk}(\xi, \eta), \tag{2.7}$$

and the flux function $\mathbf{f}(\mathbf{u}, \nabla\mathbf{u})$

$$\mathbf{f}(\mathbf{u}, \nabla\mathbf{u}, \xi, \eta, t) \simeq \mathbf{f}^{\mathrm{P}}(\mathbf{u}, \nabla\mathbf{u}, \xi, \eta, t) = \sum_{j=1}^{N_x} \sum_{k=1}^{N_y} f_{jk}(t)\varphi_{jk}(\xi, \eta), \tag{2.8}$$

with time dependent coefficients $u_{jk}(t)$ and $f_{jk}(t) = \mathbf{f}(u_{jk}(t), \nabla u_{jk}(t))$, and a set of basis functions $\{\varphi_{jk}\}_{j,k=1}^{N_x,N_y}$ with $p_x = N_x - 1$, $p_y = N_y - 1$.

The global solution $\mathbf{u}(\xi, \eta)$ is then approximated by a piecewise polynomial

$$\mathbf{u}(\xi, \eta, t) \simeq \mathbf{u}^{\mathrm{num}}(\xi, \eta, t) = \bigoplus_{n,m=1}^{N^x,N^y} \mathbf{u}^{\mathrm{P}}|_{\Omega_{n,m}}(\xi, \eta, t).$$

We only assume continuity of the polynomial approximation on the elements, but not on the interfaces, i.e. we allow for discontinuous numerical approximations at element boundaries. A one-dimensional example for such a numerical solution can be seen in Figure 2.3.

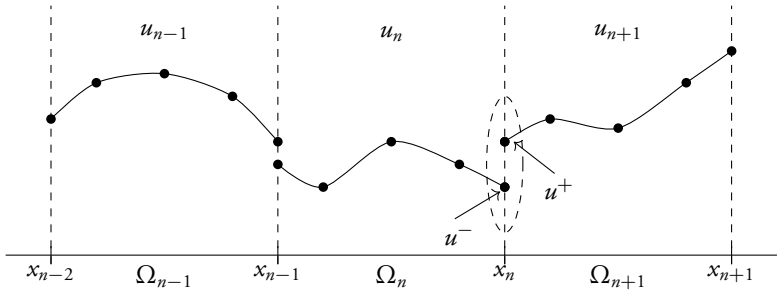


Figure 2.3: DG approximations are assumed to be continuous on the elements, but not on the interfaces.

As for FV methods, a numerical flux $\mathbf{f}^{\mathrm{N}}(\mathbf{u}^-, \mathbf{u}^+; \mathbf{n})$ is introduced at the discontinuous element boundaries, which is a function of the interface values of the neighboring elements, see Figure 2.3. Different choices of numerical fluxes were discussed before.

The test functions ψ are also chosen as polynomials in the reference element with the same basis,

$$\psi(\xi, \eta) = \sum_{j=1}^{N_x} \sum_{k=1}^{N_y} \psi_{jk}(t) \varphi_{jk}(\xi, \eta), \tag{2.9}$$

with arbitrary coefficients $\psi_{jk}(t), j = 1, \ldots, N_x$, $k = 1, \ldots, N_y$.

Mapping (2.4) onto the reference element and inserting the numerical approximations (2.7), (2.8) and the arbitrary test polynomials (2.9), yields the weak form

$$\begin{aligned} &\sum_{j=1}^{N_x} \sum_{k=1}^{N_y} J_{jk} \dot{u}_{jk}(t) \int_{\hat{\Omega}} \varphi_{jk} \varphi_{lm} \, \mathrm{d}\hat{\Omega} + \sum_{j=1}^{N_x} \sum_{k=1}^{N_y} f_{jk}(t) \int_{\partial\hat{\Omega}} \varphi_{jk} \cdot \mathbf{n} \varphi_{lm} \, \mathrm{d}\mathbf{s} \\ &- \sum_{j=1}^{N_x} \sum_{k=1}^{N_y} f_{jk}(t) \int_{\hat{\Omega}} \varphi_{jk} \cdot \nabla \varphi_{lm} \, \mathrm{d}\hat{\Omega} = 0, \end{aligned} \tag{2.10}$$

for each $l = 1, \ldots, N_x$, $m = 1, \ldots, N_y$ and $J_{jk} = \mathbf{J}(\xi_j, \eta_k)$.

The DG-SEM is based on approximating integrals with LGL quadrature on the reference element using a nodal Lagrange basis based on these nodes [60]. This yields a polynomial space of dimension $(p_x + 1)(p_y + 1)$. The LGL nodes and weights can be defined using Legendre polynomials.

Definition 2.1 (Legendre-Gauss-Lobatto nodes and weights [80]). The *Legendre polyno-*

mials are recursively defined as

$$L_{n+1}(x) = \frac{2n+1}{n+1} x L_n(x) - \frac{n}{n+1} L_{n-1}(x), \quad x \in [-1, 1],$$

with $L_0(x) = 1$, $L_1(x) = x$. Then the *Legendre-Gauss-Lobatto nodes* ξ_i include the endpoints of the reference interval, ± 1, and the interior nodes are the roots of the polynomial

$$q(x) = L_{n+1}(x) - L_{n-1}(x).$$

The *Legendre-Gauss-Lobatto weights* for the corresponding quadrature rule are given by

$$\omega_i = \frac{2}{n(n+1)(L_n(\xi_i))^2}.$$

The LGL nodes and weights for $\{2, 3, 4, 5\}$ points can be seen in Table 2.1.

Table 2.1: First Legendre-Gauss-Lobatto nodes and weights.

number of points	nodes	weights
2	± 1	1
3	± 1, 0	$\frac{1}{3}$, $\frac{4}{3}$
4	± 1, $\pm\sqrt{\frac{1}{5}}$	$\frac{1}{6}$, $\frac{5}{6}$
5	± 1, 0 ,$\pm\sqrt{\frac{3}{7}}$	$\frac{1}{10}$, $\frac{32}{45}$, $\frac{49}{90}$

Lagrange polynomials based on the LGL nodes are used as basis functions.

Definition 2.2 (Lagrange polynomial)**.** For a given set of points $\{x_1, \ldots, x_N\}$ the j-th *Lagrange polynomial* of degree $N-1$ is defined by

$$\ell_j(x) := \prod_{i=1, i \neq j}^{N} \frac{x - x_i}{x_j - x_i}, \quad j = 1, \ldots, N.$$

Lagrange polynomials satisfy the so-called *cardinal property*

$$\ell_j(x_i) = \delta_{ji} := \begin{cases} 1, & i = j, \\ 0, & i \neq j. \end{cases}$$

Lagrange polynomials of first, second and third order based on LGL nodes can be seen in Figure 2.4.

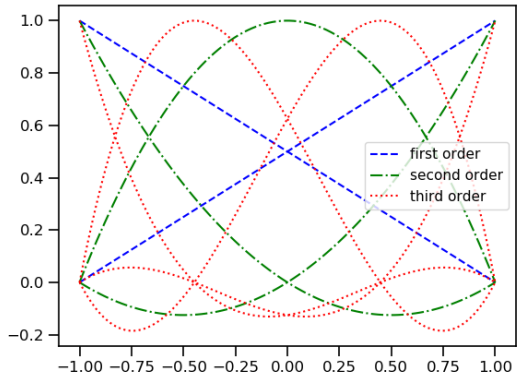


Figure 2.4: Lagrange polynomials based on LGL nodes on the reference interval $[-1, 1]$.

The basis polynomials in (2.7) are then defined with a tensor product ansatz

$$\varphi_{jk}(\xi, \eta) := \ell_j^{\xi}(\xi)\ell_k^{\eta}(\eta). \tag{2.11}$$

For the DG-SEM, interpolation and quadrature are collocated. This implies that the same nodes and weights as for the Lagrangian basis are used for the Gaussian quadrature of the integral:

$$\int_{\hat{\Omega}} \mathbf{u}(\xi, \eta)\, \mathrm{d}\hat{\Omega} \approx \sum_{j=1}^{N_x} \sum_{k=1}^{N_y} \omega_j^{\xi} \omega_k^{\eta} \mathbf{u}(\xi_j, \eta_k). \tag{2.12}$$

Gaussian quadrature with N LGL nodes is exact for polynomials of degree $2N - 1$ or less. Exact integration increases the cost of the integral approximation, thus we accept the error in (2.12).

Applying Gaussian quadrature to the first term in (2.10) gives

$$\begin{aligned} &\sum_{j=1}^{N_x} \sum_{k=1}^{N_y} J_{jk} \dot{u}_{jk}(t) \int_{\hat{\Omega}} \varphi_{jk} \varphi_{lm}\, \mathrm{d}\hat{\Omega} \\ &\approx \sum_{j=1}^{N_x} \sum_{k=1}^{N_y} J_{jk} \dot{u}_{jk}(t) \left(\sum_{i=1}^{N_x} \sum_{s=1}^{N_y} \omega_i^{\xi} \omega_s^{\eta} \varphi_{jk}(\xi_i, \eta_s) \varphi_{lm}(\xi_i, \eta_s) \right) = J_{lm} \omega_l^{\xi} \omega_m^{\eta} \dot{u}_{lm}(t), \end{aligned} \tag{2.13}$$

for $l = 1, \ldots, N_x, m = 1, \ldots, N_y$ due to the cardinal property of the basis functions. Defining a vector of coefficients $\bar{\mathbf{u}}(t) \in \mathbb{R}^{N_x N_y}$ and a block diagonal mass matrix

$$\mathbf{M} = \mathbf{M}^{\xi} \otimes \mathbf{M}^{\eta} \in \mathbb{R}^{N_x N_y \times N_x N_y},$$

with the (lumped) mass matrices

$$\mathbf{M}^{\xi} = \operatorname{diag}\big([\omega_1^{\xi}, \dots, \omega_{N_x}^{\xi}]\big) \in \mathbb{R}^{N_x \times N_x}, \tag{2.14}$$

$$\mathbf{M}^{\eta} = \operatorname{diag}\big([\omega_1^{\eta}, \dots, \omega_{N_y}^{\eta}]\big) \in \mathbb{R}^{N_y \times N_y}, \tag{2.15}$$

(2.13) can be written more compactly as $\mathbf{J}\mathbf{M}\bar{\mathbf{u}}_t$ on the reference element $\hat{\Omega}$.

Applying Gaussian quadrature to the third term in (2.10) yields

$$\begin{aligned}
&\sum_{j=1}^{N_x}\sum_{k=1}^{N_y} f_{jk}(t) \int_{\hat{\Omega}} \varphi_{jk} \cdot \nabla \varphi_{lm} \,\mathrm{d}\hat{\Omega} \\
&\approx \sum_{j=1}^{N_x}\sum_{k=1}^{N_y} f_{jk}(t) \left(\sum_{i=1}^{N_x}\sum_{s=1}^{N_y} \omega_i^{\xi}\omega_s^{\eta} \varphi_{jk}(\xi_i, \eta_s) \cdot \nabla \varphi_{lm}(\xi_i, \eta_s) \right) \\
&= \sum_{j=1}^{N_x}\sum_{k=1}^{N_y} \omega_j^{\xi}\omega_k^{\eta} f_{jk}(t) \cdot \nabla \varphi_{lm}(\xi_j, \eta_k) \\
&= \sum_{j=1}^{N_x} \omega_j^{\xi}\omega_m^{\eta} f_{jm}^1(t) \frac{\partial \ell_l^{\xi}(\xi_j)}{\partial \xi} + \sum_{k=1}^{N_y} \omega_l^{\xi}\omega_k^{\eta} f_{lk}^2(t) \frac{\partial \ell_k^{\eta}(\eta_m)}{\partial \eta},
\end{aligned} \tag{2.16}$$

for $l = 1, \dots, N_x$, $m = 1, \dots, N_y$. Let $\bar{\mathbf{f}}^1, \bar{\mathbf{f}}^2 \in \mathbb{R}^{N_x N_y}$ be the vectors of evaluations at the quadrature nodes for the first and the second component and define the mass matrices as before. Let $\mathbf{D}^{\xi} \in \mathbb{R}^{N_x N_y \times N_x N_y}$ be a block diagonal matrix with constant blocks of size $\mathbb{R}^{N_x \times N_x}$

$$(\mathbf{D}^{\xi})_{jl} = \frac{\partial \ell_l^{\xi}(\xi_j)}{\partial \xi}, \quad l, j = 1, \dots, N_x, \tag{2.17}$$

and $\mathbf{D}^{\eta} \in \mathbb{R}^{N_x N_y \times N_x N_y}$ be a block diagonal matrix with constant blocks of size $\mathbb{R}^{N_y \times N_y}$

$$(\mathbf{D}^{\eta})_{mk} = \frac{\partial \ell_k^{\eta}(\eta_m)}{\partial \eta}, \quad k, m = 1, \dots, N_y. \tag{2.18}$$

Then (2.16) can be written compactly as $(\mathbf{D}^{\xi\top}\mathbf{M}^{\xi}) \otimes \mathbf{M}^{\eta}\bar{\mathbf{f}}^1 + \mathbf{M}^{\xi} \otimes (\mathbf{D}^{\eta\top}\mathbf{M}^{\eta})\bar{\mathbf{f}}^2$.

The boundary integral in (2.10) is the sum of the four integrals on the edges of the refer-

ence cell. It can be approximated by

$$
\begin{aligned}
&\sum_{j=1}^{N_x}\sum_{k=1}^{N_y} f_{jk}(t)\int_{\partial\hat{\Omega}} \varphi_{jk}\cdot\mathbf{n}\varphi_{lm}\,\mathrm{d}\mathbf{s}\\
&\approx \omega_m^\eta\big(\mathbf{f}^{\mathrm{P}}(u,(1,\eta_m),t)\big)_1\ell_l^\xi(1)-\omega_m^\eta\big(\mathbf{f}^{\mathrm{P}}(u,(-1,\eta_m),t)\big)_1\ell_l^\xi(-1)\\
&+\omega_l^\xi\big(\mathbf{f}^{\mathrm{P}}(u,(\xi_l,1),t)\big)_2\ell_m^\eta(1)-\omega_l^\xi\big(\mathbf{f}^{\mathrm{P}}(u,(\xi_l,-1),t)\big)_2\ell_m^\eta(-1),
\end{aligned}
\tag{2.19}
$$

for $l = 1,\dots,N_x, m = 1,\dots,N_y$. Let $\hat{\mathbf{f}}^1,\hat{\mathbf{f}}^2 \in \mathbb{R}^{N_xN_y}$ be the vectors of function evaluations at the quadrature nodes on the surface for the first and the second component and

$$
\begin{aligned}
(\mathbf{S}^\xi)_{li} &= -\delta_{1i}\ell_l^\xi(-1)+\delta_{N_xi}\ell_l^\xi(1),\\
(\mathbf{S}^\eta)_{jm} &= -\delta_{j1}\ell_m^\eta(-1)+\delta_{jN_y}\ell_m^\eta(1),
\end{aligned}
$$

then (2.19) can be written compactly as $\mathbf{S}^\xi\otimes\mathbf{M}^\eta\hat{\mathbf{f}}^1+\mathbf{M}^\xi\otimes\mathbf{S}^\eta\hat{\mathbf{f}}^2$.

Replacing the boundary fluxes $\hat{\mathbf{f}}^1,\hat{\mathbf{f}}^2$ by numerical fluxes $\hat{\mathbf{f}}^{1,\mathrm{N}},\hat{\mathbf{f}}^{2,\mathrm{N}}\in\mathbb{R}^{N_xN_y}$, we obtain on each reference element a system of ordinary differential equations

$$
\begin{aligned}
\mathbf{J}\mathbf{M}\bar{\mathbf{u}}_t &+ \big(\mathbf{S}^\xi\otimes\mathbf{M}^\eta\mathbf{f}^{1,\mathrm{N}}+\mathbf{M}^\xi\otimes\mathbf{S}^\eta\mathbf{f}^{2,\mathrm{N}}\big)\\
&-\big((\mathbf{D}^{\xi^\top}\mathbf{M}^\xi)\otimes\mathbf{M}^\eta\bar{\mathbf{f}}^1+\mathbf{M}^\xi\otimes(\mathbf{D}^{\eta^\top}\mathbf{M}^\eta)\bar{\mathbf{f}}^2\big)=\mathbf{0},
\end{aligned}
\tag{2.20}
$$

or equivalently

$$
\begin{aligned}
\bar{\mathbf{u}}_t &+ \frac{1}{\mathbf{J}}\mathbf{M}^{-1}\big((\mathbf{S}^\xi\otimes\mathbf{M}^\eta\mathbf{f}^{1,\mathrm{N}}+\mathbf{M}^\xi\otimes\mathbf{S}^\eta\mathbf{f}^{2,\mathrm{N}}\big)\\
&-\big((\mathbf{D}^{\xi^\top}\mathbf{M}^\xi)\otimes\mathbf{M}^\eta\bar{\mathbf{f}}^1+\mathbf{M}^\xi\otimes(\mathbf{D}^{\eta^\top}\mathbf{M}^\eta)\bar{\mathbf{f}}^2)\big)=\mathbf{0}.
\end{aligned}
\tag{2.21}
$$

Collecting the equations for all elements in one big system $\underline{\mathbf{G}}$, which is of block form with blocks corresponding to the degrees of freedom in each element, and a vector of unknowns $\underline{\mathbf{u}}$, equation (2.21) can be rewritten as an initial value problem (IVP)

$$
\underline{\mathbf{u}}_t-\underline{\mathbf{G}}(\underline{\mathbf{u}})=\underline{\mathbf{0}},
\tag{2.22}
$$

with initial condition $\underline{\mathbf{u}}(t_0)=\underline{\mathbf{u}}^0$ and a possibly nonlinear function $\underline{\mathbf{G}}$.

2 Properties

In the following we summarize the most important properties of semi-discrete DG schemes for the context of this thesis. A good overview over the state of the art of the theoretical results is given in [59, 73, 80] as well as the references therein.

2.1 Stability

It is still an ongoing task to construct conservative and stable DG methods for nonlinear problems. For semi-discrete approximations it is not clear that the solution is bounded in the L_2 norm by an amount proportional to the initial energy. Thus, the approximate solution might blow up over finite time [80]. Non-conservative methods can result in unphysical solutions, while conservative schemes guarantee that conserved quantities of the problem only change depending on the fluid flow into or out of the domain. This is necessary for the numerical scheme to yield a weak solution of the conservation law. However, it is not clear which of the non-unique weak solutions of the conservative scheme is physically relevant. It is therefore important to carefully chose a solution. This can be done by constructing conservative schemes that numerically obey the second law of thermodynamics and considering entropy.

In the last decades, entropy stable schemes have been developed. Tadmor was the first one to develop conservative and entropy stable schemes for low order FV methods [119]. Based on this, entropy-conservative and entropy-stable fluxes have been constructed [98, 120, 121]. High-order, conservative and entropy stable schemes for linear conservation laws have been developed in [23] based on the summation by parts (SBP) simultaneous approximation term (SAT) framework, which we discuss in more details below. SBP is the discrete counterpart of integration by parts. Pioneering work from 1974 for the development of SBP can be found in [83]. With the SBP property, the discrete analysis can be done in a one-to-one fashion to the continuous analysis. This allows to construct stable numerical solvers for conservation laws.

It has been proven that DG-SEM methods with LGL nodes satisfy the discrete SBP property and the boundary and interface conditions can be weakly imposed by the SAT [47]. This allowed to extend conservative and entropy stable schemes for low order methods developed by Tadmor to high order schemes with SBP operators using flux difference schemes [39] and split forms [47].

In the following we give an introduction to the concepts of SBP-SAT schemes as well as the entropy condition.

Summation by Parts Simultaneous Approximation Term

When applying standard DG methods to systems of conservation laws it is not clear that discrete solutions remain bounded in time by the initial and boundary conditions. For instance, L_2 stability can not be guaranteed. The SBP-SAT strategy is a successful way to identify good boundary conditions [117]. The idea is to derive boundary conditions

using the energy method by requiring stability in the L_2 norm for linearized problems, and implement them using the SAT term.

Definition 2.3 (SBP operator [83]). For a set of N nodes $\{\xi_i\}_{i=1}^N$ on the reference space $\hat{\Omega} = [-1, 1]$ and with $\boldsymbol{\xi}^k := [\xi_i^k, \dots, \xi_N^k]^\top$, $k > 0$, the matrix $\mathbf{D}^{\text{SBP}} \in \mathbb{R}^{N \times N}$ is an *SBP operator* of degree p approximating $\frac{\partial}{\partial \xi}$ on the nodal distribution $\boldsymbol{\xi}$ if

$$\begin{aligned}
&\mathbf{D}^{\text{SBP}} \boldsymbol{\xi}^k = k \boldsymbol{\xi}^{k-1}, \quad k = 0, \dots, p, \\
&\mathbf{D}^{\text{SBP}} = \mathbf{M}^{\text{SBP}^{-1}} \mathbf{Q}, \\
&\mathbf{M}^{\text{SBP}} \text{ is symmetric and positive definite,} \\
&\mathbf{Q} + \mathbf{Q}^\top = \mathbf{B} = \operatorname{diag}\big([-1, 0, \dots, 0, 1]\big).
\end{aligned}$$

For a hyperbolic conservation law

$$\begin{aligned}
&\mathbf{u}_t + \nabla \cdot \mathbf{f}(\mathbf{u}) = \mathbf{0}, \ (\mathbf{x}, t) \in \Omega \times (0, T], \\
&\mathbf{u}(\mathbf{x}, t) = \mathbf{b}(t), \qquad \mathbf{x} \in \delta\Omega, \ t \in (0, T],
\end{aligned}$$

the SBP-SAT scheme reads, assuming the problem is linear with constant coefficients,

$$\mathbf{u}_t + \mathbf{D}^{\text{SBP}} \mathbf{f} = \mathbf{SAT},$$

with $\mathbf{D}^{\text{SBP}}$ an operator approximating the spatial derivative of the physical flux $\mathbf{f}$ and the SAT on the right hand side defining a penalty term to weakly incorporate the boundary condition,

$$\mathbf{SAT} = \sigma(\mathbf{u} - \mathbf{b}),$$

with a penalty parameter σ and the boundary condition $\mathbf{b}$.

In [47] it has been shown that the DG-SEM mass matrices (2.14), (2.15) and the derivative matrices (2.17), (2.18) satisfy Definition 2.3 with $\mathbf{Q} = \mathbf{MD}$ and thus define an SBP operator for the upwind and local Lax Friedrichs flux.

2.2 Entropy Condition

Weak solutions are often not unique and the correct weak solution needs to be identified. We are only interested in solutions that describe the physical behavior of the problem properly, i.e. the second law of thermodynamics should be obeyed. This means that the total entropy of a physical system should not decrease. Since the mathematical entropy is not built into the system of ODEs, an entropy condition needs to be added.

Let us consider a one-dimensional system of hyperbolic conservation laws

$$\mathbf{u}_t + \mathbf{f}(\mathbf{u})_x = \mathbf{0}. \tag{2.23}$$

A scalar function $s = s(\mathbf{u})$ is called an *entropy function* if it is strongly convex and the corresponding entropy variables $\mathbf{v} := \frac{\partial s}{\partial \mathbf{u}}$ provide a one-to-one mapping between the conservative and the entropy space [120]. Contracting (2.23) with $\mathbf{v}^\top$ from the left yields

$$\mathbf{v}^\top \big(\mathbf{u}_t + \mathbf{f}(\mathbf{u})_x\big) = \mathbf{0},$$

and by the definition of the entropy variables we get

$$\mathbf{v}^\top \mathbf{u}_t = s_t.$$

An entropy flux f^{ent} can be found by

$$\mathbf{v}^\top \frac{\partial \mathbf{f}}{\partial x} = \left(\frac{\partial s}{\partial \mathbf{u}}\right)^\top \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial x} = \left(\frac{\partial f^{\text{ent}}}{\partial \mathbf{u}}\right)^\top \frac{\partial \mathbf{u}}{\partial x} = \frac{\partial f^{\text{ent}}}{\partial x}, \tag{2.24}$$

see [119, 120] for more details. This gives an entropy/entropy-flux pair (s, f^{ent}) as well as an entropy conservation law:

$$s_t + f_x^{\text{ent}} = 0. \tag{2.25}$$

However, (2.25) is only valid for smooth solutions but not for discontinuous ones, e.g. when shocks develop in finite time. From a physical point of view, entropy can only increase. Thus, (2.25) needs to be modified for discontinuous solutions and we get an entropy inequality

$$s_t + f_x^{\text{ent}} \leq 0.$$

An important part in the construction of entropy stable schemes is the chain rule (2.24), which is difficult to mimic in the discrete case. On the continuous level it holds that

$$\mathbf{v}_x^\top \mathbf{f} = \mathbf{v}_x - f_x^{\text{ent}}.$$

Tadmor considered in [120] a first order FV discretization

$$(\mathbf{u}_n)_t + \frac{1}{\Delta x}\big(\mathbf{f}_{k+1/2}^{n,\text{N}} - \mathbf{f}_{k-1/2}^{n,\text{N}}\big) = 0,$$

to recover the chain rule (2.24) by carefully designing the numerical entropy fluxes $\mathbf{f}^{\text{N}}$.

Fisher and Carpenter's pioneering work [39] extended this to high-order numerical methods. They showed that if the discrete derivative matrix is an SBP operator, these low order entropy fluxes can be extended to high order methods. Based on Tadmor's FV framework, high order entropy constant numerical fluxes can be constructed by satisfying the Tadmor shuffle conditions [121]

$$\begin{aligned} [\![\mathbf{v}]\!]^\top s &= [\![\varphi]\!], \\ [\![\mathbf{v}]\!]^\top f^{\text{ent}} &= [\![\psi]\!], \end{aligned}$$

with the jump operator $[\![\cdot]\!]$ and the entropy flux potentials

$$\begin{aligned} \varphi &= \mathbf{v}^\top \mathbf{u} - s, \\ \psi &= \mathbf{v}^\top \mathbf{f} - f^{\text{ent}}. \end{aligned}$$

The pair (φ, ψ) is called *entropy/entropy flux potential.*

For the construction of entropy stable DG-SEM schemes for typical problems in fluid dynamics we refer to [17, 25, 26, 48, 142]. Entropy stable space-time DG-SEM schemes have been constructed in [44], with an upwind flux in time and an entropy stable flux in space.

2.3 Order of Convergence

Theoretical and practical error estimates for DG methods have been presented in the literature. For element-wise polynomial approximations of degree p is was proven for hyperbolic problems that an optimal convergence rate of $p + \frac{1}{2}$ can be measured in the L_2 norm on general grids [70, 145]. On Cartesian grids, an optimal convergence rate of $p + 1$ can be proven [59]. The order of convergence might also depend on the choice of numerical fluxes, and sometimes on whether p is odd or even.

However, numerical experiments give convergence rates of $p + 1$ even for computational meshes having no particular uniformity [107]. Even super-convergence rates of greater or equal to $2p + 1$ can be achieved for special problems [4], but these convergence rates depend on the chosen time-stepping method.

3 Time Integration

Spatial discretization with DG-SEM results in a system of ordinary differential equations, see (2.3) and (2.22). The ODE can be rewritten as an IVP

$$\underline{\mathbf{u}}_t(t) - \underline{\mathbf{G}}(\underline{\mathbf{u}}) = \underline{\mathbf{0}},\ \underline{\mathbf{u}}(t_0) = \underline{\mathbf{u}}^0, \tag{2.26}$$

with a possibly nonlinear function $\underline{\mathbf{G}}$. Disretizing first in space and then solving the resulting ODE is called methods of lines (MOL).

An overview over numerical methods to solve problems of the form (2.26) can be found in classical textbooks. Most of them are based on approximating $\underline{\mathbf{u}}^n \approx \underline{\mathbf{u}}(t_n)$ in discrete time points t_n. Numerical time integration schemes are called *implicit* if they incorporate unknown data $\underline{\mathbf{u}}^{n+1}$, otherwise they are called *explicit*.

We consider here *Runge-Kutta* (RK) time-stepping methods. An s-stage Runge-Kutta method applied to the IVP (2.26) reads

$$\begin{aligned} \underline{\mathbf{u}}(t_{n+1}) \approx \underline{\mathbf{u}}^{n+1} &= \underline{\mathbf{u}}^n + \Delta t_n \sum_{i=1}^{s} b_i\, \underline{\mathbf{G}}(\underline{\mathbf{u}}(t_n + c_i \Delta t_n)), \\ \text{with}\quad \underline{\mathbf{u}}(t_n + c_i \Delta t_n) &= \underline{\mathbf{u}}^n + \Delta t_n \sum_{j=1}^{s} a_{ij}\, \underline{\mathbf{G}}(\underline{\mathbf{u}}(t_n + c_j \Delta t_n)). \end{aligned} \tag{2.27}$$

Different choices of the parameters a_{ij}, b_i and c_i lead to different RK methods. These parameters can be expressed in a so-called *Butcher tableau*:

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & & & & \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array} = \begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b} \end{array}$$

The corresponding RK method is explicit if $\mathbf{A}$ is lower triangular and implicit otherwise.

The most famous time-stepping schemes are the explicit and implicit Euler methods. Applied to (2.22) they read

$$\underline{\mathbf{u}}^{n+1} - \underline{\mathbf{u}}^n - \Delta t\, \underline{\mathbf{G}}(\underline{\mathbf{u}}^n) = \underline{\mathbf{0}},\ \textit{(explicit Euler)}, \tag{2.28}$$

and

$$\underline{\mathbf{u}}^{n+1} - \underline{\mathbf{u}}^n - \Delta t\, \underline{\mathbf{G}}(\underline{\mathbf{u}}^{n+1}) = \underline{\mathbf{0}},\ \textit{(implicit Euler)}. \tag{2.29}$$

Implicit schemes require solving a (non)linear equation system in every step. Explicit schemes are easier to apply in each time step, but restrictions on the step size need to be imposed to guarantee stability. These restrictions depend on the order of the temporal method as well as on the order of spatial discretization.

For hyperbolic problems, the *Courant-Friedrichs-Levy (CFL) condition* is sufficient to achieve stability. For a FV method with explicit Euler time integration, we obtain the constraint

$$\Delta t < CFL_{\max} \frac{\Delta x}{\max\limits_{k,|\mathbf{n}|=1} \lambda_k(\underline{\mathbf{u}}, \mathbf{n})}, \tag{2.30}$$

with a dimensionless maximal CFL number and the eigenvalues λ_k of the Jacobian of the inviscid flux [86]. For the explicit Euler method it holds $CFL_{\max} = 1$. Equation (2.30) is often used to calculate time steps by defining $CFL < CFL_{\max}$ and to compute Δt via (2.30) .

In this thesis we are interested in implicit methods since we do not want any stability constraint on the time-stepping width. In the next chapter we present a fully discrete numerical discretization, a space-time DG-SEM, which also results in implicit schemes. Methods to efficiently solve (non)linear systems of equations of the form (2.29) are discussed in Chapters 4 and 5.

Chapter 3

Space-Time Discontinuous Galerkin Methods

In the previous chapter we have discussed a semi-discrete method, where the partial differential equation is first discretized in space, leaving the time variable continuous. This results in a system of ordinary differential equations, and a numerical time-stepping method is applied to solve the initial value problem. We presented the DG-SEM in space and gave an overview over the concept of implicit Runge-Kutta time-stepping methods.

Today, state of the art computer architectures have reached a clock speed limit and the trend is towards more rather than faster processors. Moreover, the sequential nature of traditional time-stepping methods imposes limitations on the parallel performance. Pioneering work about parallel time integration methods was published by Nievergelt in 1964 [97]. The philosophy of space-time methods is to treat time as an additional dimension [96]. This has several advantages, i.e. moving boundaries can be treated more easily [126] and parallelization in time is possible [45], but also challenges, since the temporal direction is special. It needs to follow a causality principle, with a solution later in time only affected and determined by a solution earlier in time, never the other way around.

In this chapter we present a fully-discrete numerical scheme by applying the DG-SEM to the spatial and temporal direction simultaneously. This results in a space-time method, where (almost) no distinction is made between the spatial and temporal variables. Thus, the advantages of discontinuous Galerkin methods, as h/p-adaptivity and excellent performance on parallel computers, can be exploited in the temporal direction as well. We focus on a space-time DG-SEM developed in [44], which provides an entropy stable numerical scheme in a fully-discrete space-time domain.

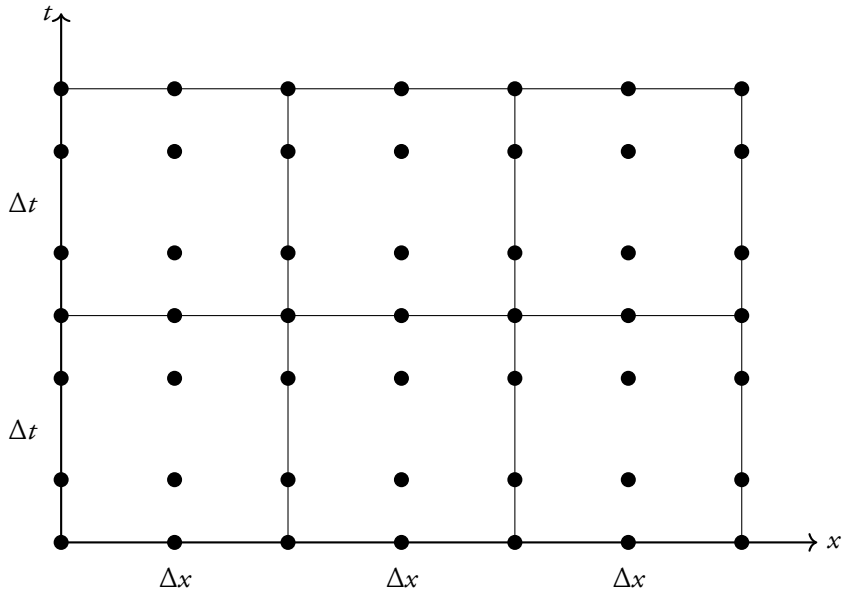


Figure 3.1: Space-time grid in one spatial dimension with three LGL nodes in the spatial and four LGL nodes in the temporal direction in each space-time element.

We consider again the conservation law

$$\begin{aligned} &\mathbf{u}_t + \nabla \cdot \mathbf{f}(\mathbf{u}, \nabla \mathbf{u}) = \mathbf{g}(\mathbf{u}), \ (\mathbf{x}, t) \in \Omega \times (0, T], \\ &\mathbf{u}(\mathbf{x}, t) = \mathbf{b}(t), \qquad\qquad \mathbf{x} \in \partial\Omega, \ t \in (0, T], \\ &\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}), \qquad\qquad \mathbf{x} \in \Omega, \end{aligned} \tag{3.1}$$

here in one space dimension $\Omega \subset \mathbb{R}$, and refer to [44] for an extension to the multi-dimensional case using a tensor product ansatz. In the following we treat time as an additional dimension and therefore start by discretizing the two-dimensional space-time domain $\Omega \times [0, T]$ into non-overlapping elements. For simplicity we assume that all elements have width Δx and height Δt. Let N^t denote the number of elements in time and N^x the number of elements in space. Then we have space-time elements

$$\Omega_{n,m} := [x_n, x_{n+1}] \times [t_m, t_{m+1}], \quad n = 1, \ldots, N^x, \ m = 1, \ldots, N^t.$$

An example for such a space-time grid can be seen in Figure 3.1.

1 Space-Time Discretization

The space-time DG-SEM can be derived as a two-dimensional problem following the steps in Chapter 2 by interpreting time as the second dimension. However, the DG-SEM in the temporal direction is special and we discuss several properties connected to it in this chapter. Thus, we present the derivation of the discretization here in more detail, writing out the tensor product ansatz.

Similar to the semi-discrete case we first multiply the conservation law (3.1) by a test function $\psi := \psi(x, t)$ and integrate in space and time over each element $\Omega_{n,m}$:

$$\int_{\Omega_{n,m}} \big(\mathbf{u}_t + \mathbf{f}(\mathbf{u}, \mathbf{u}_x)_x\big)\psi \,\mathrm{d}\Omega_{n,m} = \mathbf{0}, \quad n = 1, \dots, N^x,\ m = 1, \dots, N^t.$$

We map all physical elements to the reference element $\hat{\Omega} = [-1, 1]^2$ using the linear maps

$$\begin{aligned} \tau(t) &= 2\frac{t - t_m}{\Delta t} - 1, \quad \Delta t = t_{m+1} - t_m, \\ \xi(x) &= 2\frac{x - x_n}{\Delta x} - 1, \quad \Delta x = x_{n+1} - x_n. \end{aligned}$$

This gives the conservation law (3.1) on the computational space

$$\frac{2}{\Delta t}\tilde{\mathbf{u}}_\tau + \frac{2}{\Delta x}\tilde{\mathbf{f}}(\tilde{\mathbf{u}}, \tilde{\mathbf{u}}_\xi)_\xi = \mathbf{0}, \quad (\xi, \tau) \in \hat{\Omega} \text{ with } \hat{\Omega}.$$

Again, we skip the tilde in the following to simplify the notation.

The change of variables results in

$$\frac{2}{\Delta t}\int_{-1}^{1}\int_{-1}^{1} \mathbf{u}_\tau \psi \,\mathrm{d}\xi \,\mathrm{d}\tau + \frac{2}{\Delta x}\int_{-1}^{1}\int_{-1}^{1} \mathbf{f}(\mathbf{u}, \mathbf{u}_\xi)_\xi \psi \,\mathrm{d}\xi \,\mathrm{d}\tau = \mathbf{0}.$$

Integration by parts in both directions yields the weak form

$$\begin{aligned} &\frac{2}{\Delta t}\int_{-1}^{1}\left(\big(\mathbf{u}\psi\big)|_{-1}^{+1} - \int_{-1}^{1} \mathbf{u}\psi_\tau \,\mathrm{d}\tau\right) \mathrm{d}\xi \\ &+\frac{2}{\Delta x}\int_{-1}^{1}\left(\big(\mathbf{f}(\mathbf{u}, \mathbf{u}_\xi)\psi\big)|_{-1}^{+1} - \int_{-1}^{1} \mathbf{f}(\mathbf{u}, \mathbf{u}_\xi)\psi_\xi \,\mathrm{d}\xi\right) \mathrm{d}\tau = \mathbf{0}. \end{aligned} \tag{3.2}$$

Next, we approximate the solution $\mathbf{u}$ and the physical flux $\mathbf{f}$ on each space-time element $[-1,1]^2$ by polynomials of degree p_ξ in space and p_τ in time:

$$\mathbf{u}(\xi,\tau) \simeq \mathbf{u}^{\mathrm{P}}(\xi,\tau) = \sum_{j=1}^{N_\xi}\sum_{k=1}^{N_\tau} u_{jk}\ell_j^\xi(\xi)\ell_k^\tau(\tau), \tag{3.3}$$

$$\mathbf{f}(\mathbf{u},\mathbf{u}_\xi,\xi,\tau) \simeq \mathbf{f}^{\mathrm{P}}(\xi,\tau) = \sum_{j=1}^{N_\xi}\sum_{k=1}^{N_\tau} f_{jk}\ell_j^\xi(\xi)\ell_k^\tau(\tau), \tag{3.4}$$

with ℓ_k^τ Lagrange polynomials of degree $p_\tau = N_\tau - 1$ and ℓ_j^ξ Lagrange polynomials of degree $p_\xi = N_\xi - 1$ based on the LGL nodes $\{\tau_k\}_{k=1}^{N_\tau}$ and $\{\xi_j\}_{j=1}^{N_\xi}$. The test functions ψ are polynomials with the same Lagrange basis,

$$\psi(\xi,\tau) \simeq \boldsymbol{\psi}^{\mathrm{P}}(\xi,\tau) = \sum_{j=1}^{N_\xi}\sum_{k=1}^{N_\tau} \psi_{jk}\ell_j^\xi(\xi)\ell_k^\tau(\tau). \tag{3.5}$$

In contrast to the spatial DG method discussed in Chapter 2, where the coefficients where time-dependent, the coefficients in the space-time ansatz are constant on each space-time element.

Inserting the approximations (3.3), (3.4) and (3.5) in (3.2) we get

$$\begin{aligned}&\frac{2}{\Delta t}\int_{-1}^{1}\left(\left(\mathbf{u}^{\mathrm{P}}\ell_k^\tau\right)|_{-1}^{+1} - \int_{-1}^{1}\mathbf{u}^{\mathrm{P}}(\ell_k^\tau)_\tau\,\mathrm{d}\tau\right)\ell_j^\xi\,\mathrm{d}\xi\\ &+\frac{2}{\Delta x}\int_{-1}^{1}\left(\left(\mathbf{f}^{\mathrm{P}}(\mathbf{u}^{\mathrm{P}},\nabla\mathbf{u}^{\mathrm{P}})\ell_j^\xi\right)|_{-1}^{+1} - \int_{-1}^{1}\mathbf{f}^{\mathrm{P}}(\mathbf{u}^{\mathrm{P}},\mathbf{u}_\xi^{\mathrm{P}})(\ell_j^\xi)_\xi\,\mathrm{d}\xi\right)\ell_k^\tau\,\mathrm{d}\tau = 0,\end{aligned} \tag{3.6}$$

for $j = 1,\dots,N_\xi$ and $k = 1,\dots,N_\tau$.

To connect the space-time elements, we introduce numerical surface fluxes in the solution $\mathbf{u}^{\mathrm{N}}$ and in the physical flux $\mathbf{f}^{\mathrm{N}}$. Special attention has to be given to the temporal flux. The temporal direction needs to follow a causality principle: the solution later in time is only depending by the solution earlier in time. This can be described by the upwind flux (2.2). It has been show that this flux together with a spatial numerical surface flux based on split-form results in an entropy stable space-time DG-SEM scheme [44].

Replacing the surface fluxes in (3.6) by numerical fluxes and approximating all integrals with Gaussian quadrature based on the same LGL nodes as the Lagrange polynomials

yields the implicit weak space-time discretization

$$\begin{aligned}
&\frac{2\omega_j^\xi}{\Delta t}\Big(\big(\mathbf{u}^{\mathrm{N}}\big)_{N_\tau j}\delta_{kN_\xi} - \big(\mathbf{u}^{\mathrm{N}}\big)_{1j}\delta_{k1} - \sum_{l=1}^{N_\tau}\omega_l^\tau \ell_{k_\tau}^\tau(\tau_l)u_{lj}\Big)\\
&+\frac{2\omega_k^\tau}{\Delta x}\Big(\big(\mathbf{f}^{\mathrm{N}}\big)_{kN_\xi}\delta_{N_\tau j} - \big(\mathbf{f}^{\mathrm{N}}\big)_{k1}\delta_{1j} - \sum_{m=1}^{N_\xi}\omega_m^\xi \ell_{j_\xi}^\xi(\xi_m)f_{km}\Big) = 0,
\end{aligned} \tag{3.7}$$

for $j = 1, \dots, N_\xi,\ k = 1, \dots, N_\tau$.

Defining the temporal matrices

$$\begin{aligned}
\mathbf{M}^\tau &= \mathrm{diag}\big([\omega_1^\tau, \dots, \omega_{N_\tau}^\tau]\big) \in \mathbb{R}^{N_\tau \times N_\tau},\\
(\mathbf{D}^\tau)_{ij} &= \dot{\ell}_j^\tau(\tau_i) \in \mathbb{R}^{N_\tau \times N_\tau},\\
\mathbf{S}^\tau &= \mathrm{diag}\big([-1, 0, \dots, 0, 1]\big) \in \mathbb{R}^{N_\tau \times N_\tau},
\end{aligned} \tag{3.8}$$

and the spatial matrices

$$\begin{aligned}
\mathbf{M}^\xi &= \mathrm{diag}\big([\omega_1,^\xi \dots, \omega_{N_\xi}^\xi]\big) \in \mathbb{R}^{N_\xi \times N_\xi},\\
(\mathbf{D}^\xi)_{ij} &= \dot{\ell}_j^\xi(\xi_i) \in \mathbb{R}^{N_\xi \times N_\xi},\\
\mathbf{S}^\xi &= \mathrm{diag}\big([-1, 0, \dots, 0, 1]\big) \in \mathbb{R}^{N_\xi \times N_\xi},
\end{aligned} \tag{3.9}$$

we can write (3.7) in compact form on each reference element using the same notation as in the previous chapter:

$$\begin{aligned}
&\frac{2}{\Delta t}\Big(\mathbf{M}^\xi \otimes \mathbf{S}^\tau \mathbf{u}^{\mathrm{N}} - \mathbf{M}^\xi \otimes \big((\mathbf{D}^\tau)^\top \mathbf{M}^\tau\big)\mathbf{u}^{\mathrm{P}}\Big)\\
&+\frac{2}{\Delta x}\Big(\mathbf{S}^\xi \otimes \mathbf{M}^\tau \mathbf{f}^{\mathrm{N}} - \big(\mathbf{D}^{\xi\top}\mathbf{M}^\xi\big) \otimes \mathbf{M}^\tau \mathbf{f}^{\mathrm{P}}\Big) = \mathbf{0}.
\end{aligned} \tag{3.10}$$

2 Properties

In this section we present some properties of space-time DG-SEM with focus on the temporal discretization. We first discuss the equivalence of temporal DG-SEM to certain implicit Runge-Kutta schemes, followed by some stability results.

2.1 DG-SEM and Lobatto IIIC

DG-SEM applied to the temporal component can be connected to Lobatto IIIC Runge-Kutta schemes [18, 85].

Let us consider the initial value problem

$$\begin{aligned} \mathbf{u}_t &= \mathbf{f}(\mathbf{u}, t), \;\; t \in (0, T], \\ \mathbf{u}(0) &= \mathbf{u}^0. \end{aligned} \tag{3.11}$$

DG-SEM in Time

Discretizing the ODE (3.11) using a DG-SEM mapped to the reference interval $[-1, 1]$ leads to a scheme that can be expressed in matrix form as

$$\mathbf{B}\mathbf{u}^{\mathrm{N}} - \mathbf{D}^\top \mathbf{M}\mathbf{u} = \frac{\Delta t}{2}\mathbf{M}\mathbf{f}(\mathbf{u}). \tag{3.12}$$

The numerical solution in the current time step is obtained by solving (3.12) and choosing $\mathbf{u}^{\mathrm{N}}$ as an upwind numerical flux.

Lobatto IIIC Schemes

Given the ODE (3.11), an s-stage Lobatto Runge-Kutta method reads

$$\begin{aligned} \mathbf{u}(t_{n+1}) \approx \mathbf{u}^{n+1} &= \mathbf{u}^n + \Delta t \sum_{i=1}^{s} b_i \, \mathbf{f}(\mathbf{u}_i^n, t_n + c_i \Delta t), \\ \mathbf{u}_i^n &= \mathbf{u}^n + \Delta t \sum_{j=1}^{s} a_{ij} \, \mathbf{f}(\mathbf{u}_j^n, t_n + c_j \Delta t), \quad i = 1, \ldots, s. \end{aligned} \tag{3.13}$$

Lobatto methods can be specified via the coefficients a_{ij}, b_i and c_j by the so-called *simplifying assumptions* [69]:

$$B(p): \sum_{j=1}^{s} b_j c_j^{k+1} = \frac{1}{k}, \quad k = 1, \ldots, p, \tag{3.14}$$

$$C(q): \sum_{j=1}^{s} a_{ij} c_j^{k-1} = \frac{c_j^k}{k}, \quad i = 1, \ldots, s, \; k = 1, \ldots, q, \tag{3.15}$$

$$D(r): \sum_{i=1}^{s} b_i c_i^{k-1} a_{ij} = \frac{b_j}{k}(1 - c_j^k), \quad j = 1, \ldots, s, \; k = 1, \ldots, r. \tag{3.16}$$

Lobatto IIIC schemes of stage s are given for $B(2s-s)$, $C(s-1)$ and $D(s-1)$. Moreover, $a_{i1} = b_i$ for $i = 1, \ldots, s$, [5, 27, 34].

The Butcher tableaus for s-stage Lobatto IIIC methods for $s \in \{2, 3, 4\}$ are given below.

$$\begin{array}{c|cc} 0 & \frac{1}{2} & -\frac{1}{2} \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \qquad \begin{array}{c|ccc} 0 & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{6} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

$$\begin{array}{c|cccc} 0 & \frac{1}{12} & -\frac{\sqrt{5}}{12} & \frac{\sqrt{5}}{12} & -\frac{1}{12} \\ \frac{1}{2}-\frac{\sqrt{5}}{10} & \frac{1}{12} & \frac{1}{4} & \frac{10-7\sqrt{5}}{60} & \frac{\sqrt{5}}{60} \\ \frac{1}{2}+\frac{\sqrt{5}}{10} & \frac{1}{12} & \frac{10+7\sqrt{5}}{60} & \frac{1}{4} & -\frac{\sqrt{5}}{60} \\ 1 & \frac{1}{12} & \frac{5}{12} & \frac{5}{12} & \frac{1}{12} \\ \hline & \frac{1}{12} & \frac{5}{12} & \frac{5}{12} & \frac{1}{12} \end{array}$$

Equivalence

In [18] it has been shown that an SBP method defined on the reference interval $[-1, 1]$ is equivalent to a certain implicit Runge-Kutta method, in the sense that they lead to the same discrete solution. DG-SEM is known to have the SBP property [47]. With the next theorem, the algebraic equivalence of DG-SEM in time to Lobatto IIIC schemes follows:

Theorem 3.1 ([85]). The DG-SEM scheme (3.12) is algebraically equivalent to the following N_τ-stage implicit Runge-Kutta method:

$$\begin{aligned} \mathbf{u}^{n+1} &= \mathbf{u}^n + \Delta t \sum_{i=1}^{N_t} b_i \, \mathbf{f}^P(\mathbf{u}_i^n, t_n + c_i \Delta t), \\ \mathbf{u}_i^n &= \mathbf{u}^n + \Delta t \sum_{j=1}^{N_t} a_{ij} \, \mathbf{f}^P(\mathbf{u}_j^n, t_n + c_j \Delta t), \quad i = 1, \ldots, N_\tau, \end{aligned} \tag{3.17}$$

with

$$a_{i1} = b_1, \quad i = 1, \dots, N_\tau,$$
$$a_{ij} = \int_0^{c_i} \ell_j(t)\,\mathrm{d}t - b_1 \ell_j(c_1), \quad i = 1, \dots, N_\tau,\ j = 2, \dots, N_\tau,$$

and the Lagrange polynomials

$$\ell_i(t) := \prod_{j=2, j\neq i}^{N_t} \frac{t - c_j}{c_i - \tau_j}, \quad i = 2, \dots, N_\tau,$$

with $\ell_2 \equiv 1$ if $N_t = 2$ and the LGL nodes c_i and weights b_i mapped to $[0, 1]$.

It can be shown that a_{ij}, b_i and c_j satisfy the simplifying assumptions for Lobatto IIIC methods. Thus, it follows that the N_τ-stage implicit Runge-Kutta method is actually a Lobatto IIIC method.

2.2 Conversion Between DG Operators and Butcher Tableau

It might be convenient to rewrite the DG-SEM in matrix form (3.12) into a Lobatto IIIC Butcher tableau or reversely. In the following we denote the elements of the Butcher tableau by $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{c}$ and the DG-SEM operators as in (3.8).

Mapping the LGL nodes to the interval $[0, 1]$ gives the nodes $\mathbf{b}$ in the Butcher tableau, see Table 3.1. The LGL weights multiplied by a factor $\frac{1}{2}$ give the weights $\mathbf{c}$, see Table 3.2.

Table 3.1: LGL nodes on $[-1, 1]$ and $[0, 1]$.

N	2	3	4
LGL nodes on $[-1, 1]$	$-1, 1$	$-1, 0, 1$	$-1, -\frac{1}{\sqrt{5}}, \frac{1}{\sqrt{5}}, 1$
LGL nodes mapped to $[0, 1]$	$0, 1$	$0, \frac{1}{2}, 1$	$0, \frac{1}{2} - \frac{\sqrt{5}}{10}, \frac{1}{2} + \frac{\sqrt{5}}{10}, 1$

In [18, 89] a formula is presented to calculate a Butcher tableau given an SBP operator. Since the DG-SEM discretization yields an SBP-SAT operator [47], this can be applied, yielding:

$$\mathbf{A} = \frac{1}{2}\left(\mathbf{D} + \mathbf{M}^{-1}\mathbf{e}_1\mathbf{e}_1^\top\right)^{-1},$$
$$\mathbf{b} = \frac{1}{2}\mathbf{M}\mathbf{1},$$
$$\mathbf{c} = \frac{\boldsymbol{\tau} + \mathbf{1}}{2},$$

Table 3.2: LGL weights on $[-1, 1]$ and $[0, 1]$.

N	2	3	4
LGL weights on $[-1, 1]$	$1, 1$	$\frac{1}{3}, \frac{4}{3}, \frac{1}{3}$	$\frac{1}{6}, \frac{5}{6}, \frac{5}{6}, \frac{1}{6}$
LGL weights mapped to $[0, 1]$	$\frac{1}{2}, \frac{1}{2}$	$\frac{1}{6}, \frac{2}{3}, \frac{1}{6}$	$\frac{1}{12}, \frac{5}{12}, \frac{5}{12}, \frac{1}{12}$

with $\boldsymbol{\tau} = [\tau_1, \ldots, \tau_{N_t}]^\top$ the vector of LGL nodes and $\mathbf{e}_1 = (1, 0, \ldots, 0)^\top \in \mathbb{R}^{N_t}$.

Reversely, given a Lobatto IIIC Butcher tableau the corresponding DG-SEM operators on the reference interval $[-1, 1]$ can be calculated as

$$\begin{aligned} \mathbf{M} &= 2\mathrm{diag}\big([\mathbf{b}]\big), \\ \mathbf{D} &= \frac{1}{2}\mathbf{A}^{-1} - \mathbf{M}^{-1}\mathbf{e}_1\mathbf{e}_1^\top. \end{aligned}$$

2.3 Stability

Lobatto IIIC methods, and hence DG-SEM in time, satisfy several desirable stability properties that make them suitable for stiff problems. The stability function of Lobatto IIIC methods is a Padé approximant of the exponential function.

Theorem 3.2 (Padé approximant of e^z, [54]). The (k, m)-Padé approximant

$$r_{km}(z) = \frac{p_{km}(z)}{q_{km}(z)}$$

to the exponential function e^z is given by

$$\begin{aligned} p_{km}(z) &= 1 + \sum_{j=1}^{k} \frac{(k+m-j)!\,k!}{(k+m)!\,(k-j)!} \cdot \frac{z^j}{j!}, \\ q_{km}(z) &= 1 + \sum_{j=1}^{m} \frac{(k+m-j)!\,m!}{(k+m)!\,(m-j)!} \cdot \frac{(-z)^j}{j!}. \end{aligned}$$

For linear problems, the convergence rate of s-stage Lobatto IIIC methods may be as high as $2s - 2$ if stiff components are sufficiently well resolved:

Theorem 3.3 ([54, 69]). For $s \in \mathbb{N}$, the s-stage Lobatto IIIC scheme is of order $2s - 2$ and its stability function $R(z)$ is given by the $(s-2, s)$-Padé approximation of the exponential function e^z.

From Theorems 3.1 and 3.3 it directly follows that the stability function of the temporal DG-SEM discretization is given by a Padé approximation of the exponential function.

Chapter 4

Solving Equation Systems

As described in Chapters 2 and 3, discontinuous Galerkin discretizations with implicit time-stepping methods, either with a method of lines or a space-time ansatz, result in (non)linear equation systems. These have block structure, with block sizes depending on the number of unknowns in each DG element. For high order DG schemes the block sizes grow with increasing order, resulting in large and dense blocks. It is an ongoing research task to find competitive solvers for implicit DG schemes, especially for the three-dimensional case [137]. Explicit time-stepping schemes are limited by stability constraints, which get more restrictive for higher order spatial discretizations. For thin elements, which occur e.g. in boundary layers of turbulent flows, this stability restriction results in extremely small time steps. Explicit methods, on the other hand, are easy to implement and do not require solving an equation system in each iteration. For implicit time-stepping methods larger time steps can be chosen, which reduces the total number of temporal iterations needed to approximate the solution. But each iteration requires the solution of (non)linear equation systems of the form (2.26). This results in a more complex implementation, especially in the case of high-order three-dimensional problems. To be competitive with explicit schemes, these systems have to be solved efficiently. Hence, storage and computational time need to be considered when choosing a suitable solver.

In particular for three-dimensional problems it is too expensive to use direct solvers and efficient iterative methods need to be constructed. Several authors have worked on the construction of such solvers for high order spatial DG discretizations with implicit time-stepping methods, for instance for the two-dimensional Euler equations [139], for the two-dimensional compressible Navier-Stokes equations [7], for the three-dimensional unsteady Navier-Stokes equations [15], for the three-dimensional incompressible Navier-

Stokes equations [42] as well as for the two- and three-dimensional compressible Navier-Stokes equations [99, 101, 102]. The aim of these articles is to find efficient techniques to solve the nonlinear equations systems, mostly with the help of a multigrid approach or by constructing efficient (standard LU or block Jacobi) preconditioners for the Krylov solver. High order space-time DG discretizations are considered for the two-dimensional incompressible Navier–Stokes equations [106], for the two-dimensional advection-diffusion equations [125, 127] and for the three-dimensional incompressible Navier-Stokes equations [122].

In this chapter we present methods to solve equation systems arising from discretized partial differential equations. We focus on Newton-Krylov methods since easy to implement matrix-free variants exist and the sparsity of the corresponding Jacobian can be exploited. We start by presenting Newton's method and inexact Newton methods. Then we discuss Krylov subspace methods for the solution of the linear systems within Newton's method with focus on GMRES and its convergence properties. We finish this chapter by presenting Jacobian-free Newton-Krylov methods.

1 Newton's Method

In this section we give a short overview of the parts of the theory relevant to the needed methodology in this thesis and refer to [31] and [75] for more details.

Newton's method is one of the most classical iterative methods in numerical analysis to solve nonlinear equation systems. The idea is to linearize the corresponding root finding problem by replacing the nonlinear problem with a sequence of linear problems. Newton's method is locally convergent, provided the initial guess is close enough to the unknown zero. However, it is not predictable what happens outside the region of convergence.

Given a root finding problem

$$\underline{\mathbf{F}}(\underline{\mathbf{u}}) = \underline{\mathbf{0}}, \tag{4.1}$$

with a differentiable nonlinear function $\underline{\mathbf{F}}$, one Newton iteration reads

$$\begin{aligned} \text{solve} \quad \left.\frac{\partial \underline{\mathbf{F}}(\underline{\mathbf{u}})}{\partial \underline{\mathbf{u}}}\right|_{\underline{\mathbf{u}}^{(k)}} \Delta\underline{\mathbf{u}} &= -\underline{\mathbf{F}}(\underline{\mathbf{u}}^{(k)}), \\ \underline{\mathbf{u}}^{(k+1)} &= \underline{\mathbf{u}}^{(k)} + \Delta\underline{\mathbf{u}}, \quad k = 0, 1, 2, \ldots \end{aligned} \tag{4.2}$$

with an initial guess $\mathbf{u}^{(0)}$. (4.2) is terminated if

$$\|\underline{\mathbf{F}}(\underline{\mathbf{u}}^{(k+1)})\| \leq TOL \cdot \|\underline{\mathbf{F}}(\underline{\mathbf{u}}^{(0)})\|, \tag{4.3}$$

for a chosen tolerance TOL.

The so-called standard assumptions are introduced in [75] to prove convergence:

Definition 4.1 (Standard Assumptions, [75]). We make the following *standard assumptions* on $\underline{\mathbf{F}}$:

1. Equation (4.1) has a solution $\underline{\mathbf{u}}^*$.
2. $\underline{\mathbf{F}}' : \Omega \mapsto \mathbb{R}^{N\times N}$ is Lipschitz continuous with Lipschitz constant γ.
3. $\underline{\mathbf{F}}'(\underline{\mathbf{u}}^*)$ is nonsingular.

Let $\mathcal{B}(r)$ denote the ball of radius r about $\mathbf{u}^*$, i.e. $\mathcal{B}(r) = \{\mathbf{u} | \|\mathbf{u} - \mathbf{u}^*\| < r\}$. Then the following theorem shows convergence of the Newton iteration.

Theorem 4.2 ([75]). Let the standard assumptions hold. Then there exists a $\delta > 0$ such that if $\underline{\mathbf{u}}^{(0)} \in \mathcal{B}(\delta)$ the Newton iteration (4.2) converges quadratically to $\underline{\mathbf{u}}^*$.

An illustration of Newton's method in one dimension can be seen in Figure 4.1.

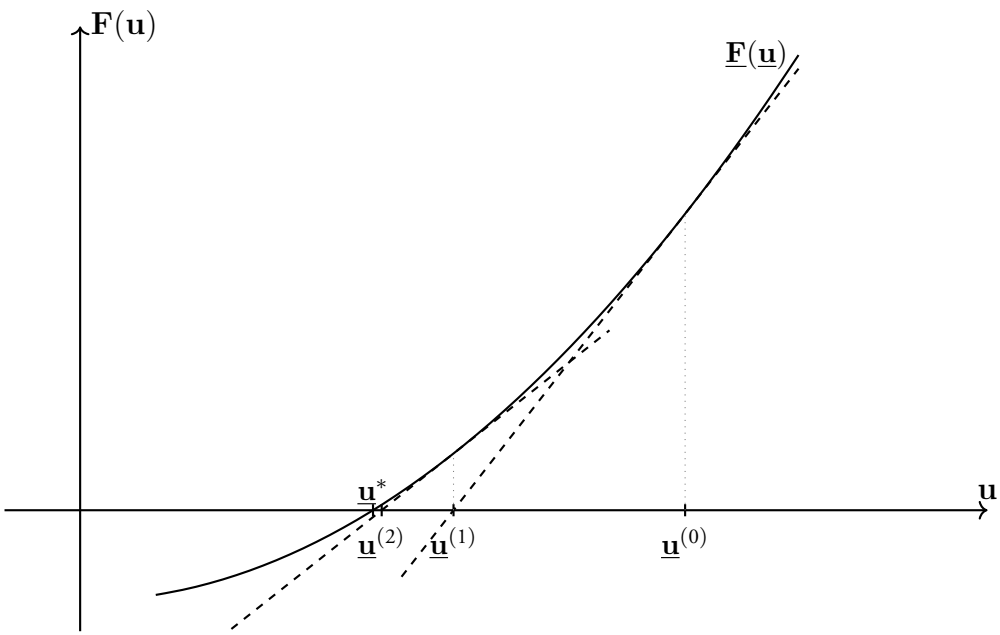


Figure 4.1: Illustration of Newton's method in one dimension.

1.1 Inexact Newton Methods

Newton's method as presented in (4.2)-(4.3) has two drawbacks, which are especially impractical for large systems, i.e. systems resulting from high order DG discretizations:

1. The Jacobian has to be computed in each iteration.
2. The linear system has to be solved exactly in each iteration.

We discuss a solution for the first problem in the last section of this chapter. For the second problem, we can use inexact Newton methods. The linear system is solved using an iterative scheme, for example with some Krylov subspace methods, which are the subjects of the next section. Any iterative method can be terminated prematurely, based on the residual of the linear equation system. An approximate solution $\widetilde{\Delta\underline{\mathbf{u}}}$ to the Newton update $\Delta\underline{\mathbf{u}}$ in step k is accepted if the relative residual is below a certain tolerance for a forcing term $\eta_k \in \mathbb{R}$. The inexact Newton method reads

$$\begin{aligned} \left\| \left.\frac{\partial \underline{\mathbf{F}}(\underline{\mathbf{u}})}{\partial \underline{\mathbf{u}}}\right|_{\underline{\mathbf{u}}^{(k)}} \widetilde{\Delta\underline{\mathbf{u}}} + \underline{\mathbf{F}}(\underline{\mathbf{u}}^{(k)}) \right\| &\leq \eta_k \|\underline{\mathbf{F}}(\underline{\mathbf{u}}^{(k)})\|, \\ \underline{\mathbf{u}}^{(k+1)} &= \underline{\mathbf{u}}^{(k)} + \widetilde{\Delta\underline{\mathbf{u}}}, \quad k = 0, 1, 2, \ldots. \end{aligned} \tag{4.4}$$

When choosing the forcing terms $\eta_k \in \mathbb{R}$ one has to keep in mind that it is not necessary to solve the first few linear systems very accurately. While far away from the solution it is not important to find the optimal search direction. To search in the generally correct direction is sufficient.

Eisenstat and Walker [35] introduced a method to choose forcing terms which fulfill the following theorem:

Theorem 4.3 ([75]). Let the standard assumptions hold. Then there exists a $\delta > 0$ such that if $\underline{\mathbf{u}}^{(0)}$ is in a δ-neighborhood of $\underline{\mathbf{u}}^*$ and $\{\eta_k\} \subset [0, \eta]$ with $\eta < 1$, then the inexact Newton method (4.4) converges linearly. Moreover, if $\eta_k \to 0$, convergence is superlinear and if $\eta_k \leq K_\eta \|\underline{\mathbf{F}}(\underline{\mathbf{u}}^{(k)})\|^p$ for some $K_\eta > 0$ and $p \in [0, 1]$, then the convergence is superlinear with order $p + 1$.

Eisenstat and Walker suggested to choose the sequence

$$\eta_k^A = \gamma \frac{\|\underline{\mathbf{F}}(\underline{\mathbf{u}}^{(k)})\|^2}{\|\underline{\mathbf{F}}(\underline{\mathbf{u}}^{(k-1)})\|^2}, \quad \gamma \in (0, 1],$$

and showed that if this sequence is bounded away from one uniformly, it has the convergence behavior required by Theorem 4.3 and the inexact Newton method converges quadratically. Thus, one can choose $\eta_0 = \eta_{max}$ for some $\eta_{max} < 1$ and

$$\eta_k^B = \min(\eta_{max}, \eta_k^A), \quad k > 0.$$

To avoid unexpected decrease in η_k they recommended to refine the definition to

$$\eta_k^C = \begin{cases} \eta_{max}, & k = 0, \\ \eta_k^B, & k > 0,\ \gamma\eta_{k-1}^2 \leq 0.1, \\ \min\big(\eta_{max}, \max(\eta_k^A, \gamma\eta_{k-1}^2)\big), & k > 0,\ \gamma\eta_{k-1}^2 > 0.1. \end{cases}$$

In order to not oversolve the final stages, Eistenstat and Walker suggested to finally compute

$$\eta_k = \min\left(\eta_{max}, \max\left(\eta_k^C, 0.5\frac{TOL \cdot \|\underline{\mathbf{F}}(\underline{\mathbf{u}}^{(0)})\|}{\underline{\mathbf{F}}(\underline{\mathbf{u}}^{(k)})}\right)\right),$$

for the tolerance TOL at which the original Newton iteration would terminate, see (4.3).

2 Krylov Subspace Methods

Krylov subspace methods are one of the most important iterative techniques available for solving large non-symmetric linear systems such as the one in (4.2) [112, 114]. They approximate the solution $\mathbf{x} \in \mathbb{R}^m$ of a linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{4.5}$$

for a nonsingular matrix $\mathbf{A} \in \mathbb{R}^{m\times m}$ and vectors $\mathbf{x}, \mathbf{b} \in \mathbb{R}^m$ using projections onto Krylov subspaces.

Definition 4.4 (Krylov subspace, [112]). The kth *Krylov subspace* is defined as

$$\mathcal{K}_k = \operatorname{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \ldots, \mathbf{A}^{k-1}\mathbf{r}_0\} = \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0),$$

with $\mathbf{r}_0 = \mathbf{A}\mathbf{x}_0 - \mathbf{b}$ for an initial guess $\mathbf{x}_0$.

Thus, approximations obtained from Krylov subspace methods are of the form

$$\mathbf{A}^{-1}\mathbf{b} \approx \mathbf{x}_k = \mathbf{x}_0 + q_{k-1}(\mathbf{A})\mathbf{r}_0, \tag{4.6}$$

with q_{k-1} a polynomial of degree $k-1$.

A Krylov subspace method is based on computing an orthonormal basis of the space $\mathbf{x}_0 + \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$ and calculating the next iterate using projection. This process needs $\mathbf{A}$ only for matrix-vector multiplications, which can be exploited if $\mathbf{A}$ is sparse.

Different Krylov methods to approximate solutions of linear systems (4.5) are suitable for different properties of the system matrix $\mathbf{A}$ and are based on different constraints to build the approximations (4.6). Some methods put the focus on an optimal method while having a bigger demand on storage and CPU time. Others are based on a short recurrence and have constant storage and CPU time per iteration. Moreover, there exist well-known methods based on the normal equations $\mathbf{A}^\top \mathbf{A}\mathbf{x} = \mathbf{A}^\top \mathbf{b}$.

Some of the most popular Krylov subspace methods are the conjugate gradient (CG) method by Hestenes and Stiefel [58] for symmetric positive definite matrices, the generalized minimum residual (GMRES) method by Saad and Schultz [114] and the biconjugate gradient stabilized (BiCGSTAB) method by Van der Vorst [131] for nonsymmetric matrices. We refer to [94] for an overview over the current state of the art of Krylov subspace methods.

2.1 GMRES

The generalized minimal residual method (GMRES) proposed by Saad and Schultz 1986 in [114] is based on computing the optimal iterate w.r.t. the 2-norm of the residual. GMRES is an iterative method to find numerical solutions of nonsymmetric linear equation systems. In every iteration, one matrix-vector product and a few scalar products have to be computed. Thus, the method is very efficient if only a few iterations are needed to approximate the solution.

In the ith GMRES iteration the minimization problem

$$\min_{\mathbf{x} \in \mathbf{x}_0 + \mathcal{K}_i} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$$

has to be solved. An orthonormal basis $\{\mathbf{v}_1, \ldots, \mathbf{v}_i\}$ of the Krylov subspace can be constructed using Arnoldi's method [3], where (modified) Gram-Schmidt is used to construct the basis. The advantage of Arnoldi's method is that it can be implemented efficiently using a Hessenberg matrix. A matrix $\mathbf{V}_i \in \mathbb{R}^{i \times m}$ is constructed with the orthonormal basis as columns. Then the upper Hessenberg matrix with an additional last row with one entry $\tilde{\mathbf{H}}_i = \mathbf{V}_{i+1}^\top \mathbf{A}\mathbf{V}_i \in \mathbb{R}^{i+1 \times i}$ is transformed to an upper triangular matrix, for example using Givens rotations.

In Algorithm 4.1 a GMRES pseudocode from [112] can be seen.

```
GMRES(A, b, x):
Choose x_0 and calculate r_0 = A x_0 − b
if r_0 == 0 then
    END
else
    v_1 = r_0 / ||r_0||
    for j = 1, 2, ..., m do
        w_j = A v_j
        for i = 1, ..., j do
            h_ij = w_j^T v_i
            w_j = w_j − h_ij v_i
        end for
        h_{j+1,j} = ||w_j||_2
        if h_{j+1,j} == 0 then
            Set m = j and BREAK
        else
            v_{j+1} = w_j / h_{j+1,j}
        end if
    end for
    V_m := [v_1, ..., v_m], H̃_m = {h_ij}_{i=1,...,m+1}^{j=1,...,m}
    Compute y_m = argmin_y ||||r_0||_2 − H̃_m y|| and x_m = x_0 + V_m y_m
end if
```

Algorithm 4.1: Pseudocode GMRES algorithm [112].

Convergence Properties

The following convergence results are known for GMRES, depending on the properties of $\mathbf{A} \in \mathbb{R}^{m \times m}$. We refer to [75] for the proofs of the next five theorems.

Definition 4.5 ([75]). The *set of* nth *degree residual polynomials* is defined as

$$\mathcal{P}_n = \{p | p \text{ is a polynomial of degree } n \text{ and } p(0) = 1\}.$$

In order to gain insight into the efficiency of the GMRES algorithm, the following residual estimate is useful for diagonalizable $\mathbf{A}$.

Theorem 4.6. Let $\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1} \in \mathbb{R}^{m \times m}$ be a nonsingular diagonalizable matrix. Let $\mathbf{x}_n$ be the nth GMRES iterate. Then for all polynomials $p_n \in \mathcal{P}_n$ it holds that

$$\frac{\|\mathbf{r}_n\|_2}{\|\mathbf{r}_0\|_2} \leq \kappa_2(\mathbf{V}) \max_{z \in \sigma(\mathbf{A})} |p_n(z)|.$$

Moreover, we can estimate the number of GMRES iterations needed to find a solution depending on the properties of $\mathbf{A}$.

Theorem 4.7. Let $\mathbf{A} \in \mathbb{R}^{m \times m}$ be a nonsingular diagonalizable matrix and assume that it has n distinct eigenvalues. Then GMRES will terminate in at most n iterations.

Theorem 4.8. Let $\mathbf{A} \in \mathbb{R}^{m \times m}$ be a nonsingular normal matrix. Let $\mathbf{b}$ be a linear combination of n eigenvectors u_i of $\mathbf{A}$,

$$\mathbf{b} = \sum_{l=1}^{n} \gamma_l u_{i_l}.$$

Then the GMRES iteration with $\mathbf{x}_0 = \mathbf{0}$ will terminate in at most n iterations.

For more general matrices $\mathbf{A}$, the following result holds:

Theorem 4.9. Let $\mathbf{A} \in \mathbb{R}^{m \times m}$ be nonsingular and let $\mathbf{x}_n$ be the nth GMRES iterate. Then for all polynomials $p_n \in \mathcal{P}_n$ it holds that

$$\|\mathbf{r}_n\|_2 = \min_{p \in \mathcal{P}_n} \|p(\mathbf{A})\mathbf{r}_0\|.$$

The estimate of the number of GRMES iterations for general matrices is less exact than the ones before.

Theorem 4.10. Let $\mathbf{A} \in \mathbb{R}^{m \times m}$ be nonsingular. Then the GMRES algorithm will find the solution within m iterations.

From Theorem 4.6 it follows that GMRES converges fast if the eigenvalues are clustered. In this case, the polynomial with zeros at the eigenvalues can be chosen. For nonnormal matrices, Theorem 4.9 states that the distribution of the eigenvalues is more useful to make statements about convergence. However, for a strongly nonnormal matrix, this can fail. A matrix with an arbitrary eigenvalue distribution can be constructed such that any nonincreasing convergence curve is possible for GMRES [51], i.e. that the residual is constant until it drops to zero in the very last step.

2.2 Preconditioning

From Theorems 4.6 and 4.9 it follows that the convergence speed of Krylov subspace methods strongly depends on the system matrix $\mathbf{A}$. Krylov subspace methods applied to

operators arising from discretized PDEs usually converge slowly since the spectrum of a discrete operator will not be clustered for any reasonable discretization. The same holds for stiff problems. Consequently, preconditioners need to be applied to get an efficient Krylov subspace method.

Preconditioning is the technique of transforming the original linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

into a linear system with the same solution, but easier to solve with an iterative method [113].

Let $\mathbf{P}_L \in \mathbb{R}^{m\times m}$ and $\mathbf{P}_R \in \mathbb{R}^{m\times m}$ be nonsingular matrices. Then

$$\mathbf{A}\mathbf{x} = \mathbf{b} \Leftrightarrow \mathbf{P}_L\mathbf{A}\mathbf{P}_R\mathbf{z} = \mathbf{P}_L\mathbf{b},\ \mathbf{P}_R\mathbf{z} = \mathbf{x}. \tag{4.7}$$

$\mathbf{P}_L$ is called *left preconditioner* and $\mathbf{P}_R$ *right preconditioner* and we refer to *right preconditioning* if $\mathbf{P}_L = \mathbf{I}$ and to *left preconditioning* if $\mathbf{P}_R = \mathbf{I}$. These precondition techniques differ in their influence on the iterative method w.r.t. computational effect and implementation.

Preconditioned GMRES

We summarize here the influence of both preconditioning techniques on the GMRES algorithm, and refer to [113] for more details.

In the case of left preconditioning, the residual $\mathbf{r}_0^P = \mathbf{P}\mathbf{A}\mathbf{x} - \mathbf{P}\mathbf{b} = \mathbf{P}\mathbf{r}_0$ changes. This influences the termination criteria of the method. The corresponding Krylov subspace is given by

$$\mathcal{K}_k(\mathbf{P}\mathbf{A}, \mathbf{r}_0^P) = \text{span}\{\mathbf{r}_0^P, \mathbf{P}\mathbf{A}\mathbf{r}_0^P, \ldots, (\mathbf{P}\mathbf{A})^{k-1}\mathbf{r}_0^P\}.$$

On the other hand, using right preconditioning leaves the residual $\mathbf{r}_0^P = \mathbf{A}\mathbf{P}\mathbf{z}_0 - \mathbf{b} = \mathbf{A}\mathbf{x}_0 - \mathbf{b}$ unchanged. The corresponding Krylov space is given by

$$\mathcal{K}_k(\mathbf{A}\mathbf{P}, \mathbf{r}_0) = \text{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{P}\mathbf{r}_0, \ldots, (\mathbf{A}\mathbf{P})^{k-1}\mathbf{r}_0\}.$$

In consequence, $\mathbf{A}\mathbf{x}$ has to be replaced by $\mathbf{A}\mathbf{P}\mathbf{x}$ in the GMRES Algorithm 4.1. This implies that the preconditioner is applied once in the beginning and once in the end, see the pseudocode from [112] in Algorithm 4.2. The only changes to Algorithm 4.1 are the application of the right preconditioner $\mathbf{P}$ in lines 8 and 21. Another advantage of right preconditioning is that the termination criterion does not need to be adapted since the right hand side is not changed. Therefore, we always refer to right preconditioning when mentioning preconditioning GMRES in the remainder of this thesis.

```
GMRES(A, b, x, P):
Choose x_0 and calculate r_0 = A x_0 − b
if r_0 == 0 then
    END
else
    v_1 = r_0 / ||r_0||
    for j = 1, 2, ..., m do
        w_j = A P v_j
        for i = 1, ..., j do
            h_ij = w_j^T v_i
            w_j = w_j − h_ij v_i
        end for
        h_{j+1,j} = ||w_j||_2
        if h_{j+1,j} == 0 then
            Set m = j and BREAK
        else
            v_{j+1} = w_j / h_{j+1,j}
        end if
    end for
    V_m := [v_1, ..., v_m], H̃_m = {h_ij}_{i=1,...,m+1}^{j=1,...,m}
    Compute y_m = argmin_y ||||r_0||_2 − H̃_m y|| and x_m = x_0 + P V_m y_m
end if
```

Algorithm 4.2: Pseudocode GMRES algorithm with preconditioner from [112].

3 Jacobian-free Newton-GMRES

As mentioned before, Newton's method (4.2)-(4.3) has two drawbacks:

1. The Jacobian has to be computed in each iteration.
2. The linear system has to be solved exactly in each iteration.

For the latter problem we have already presented methods to terminate the solver prematurely using GMRES and Eistenstat and Walker's termination criteria [35].

Instead of computing the Jacobian, approximations can be used. An extensive overview of Jacobian-free Newton-Krylov (JFNK) methods is presented in [78]. The idea is to replace the matrix-vector product in (4.2) by a difference quotient

$$\frac{\partial \underline{\mathbf{F}}(\underline{\mathbf{u}}^{(k)})}{\partial \underline{\mathbf{u}}} \Delta \underline{\mathbf{u}} \approx \frac{\underline{\mathbf{F}}(\underline{\mathbf{u}}^{(k)} + \varepsilon \Delta \underline{\mathbf{u}}) - \underline{\mathbf{F}}(\underline{\mathbf{u}}^{(k)})}{\varepsilon}, \tag{4.8}$$

to get a Jacobian-free Newton-GMRES. The parameter ε has to be chosen carefully. The approximation improves for very small ε, but cancellation errors occur. A simple choice for a parameter that is moderately small but avoids cancellation is

$$\varepsilon = \frac{\sqrt{eps}}{\|\underline{\mathbf{q}}\|_2}, \tag{4.9}$$

with the machine accuracy *eps* [103].

GMRES performs better than other iterative methods in the Jacobian-free context since the vectors in matrix-vector multiplications are normalized. The preconditioned matrix-vector product is given by

$$\frac{\partial \underline{\mathbf{F}}(\underline{\mathbf{u}}^{(k)})}{\partial \underline{\mathbf{u}}} \mathbf{P} \Delta \underline{\mathbf{u}} \approx \frac{\underline{\mathbf{F}}(\underline{\mathbf{u}}^{(k)} + \varepsilon \mathbf{P} \Delta \underline{\mathbf{u}}) - \underline{\mathbf{F}}(\underline{\mathbf{u}}^{(k)})}{\varepsilon}. \tag{4.10}$$

One drawback of Jacobian-free solvers is the difficulty to construct a preconditioner for the GMRES solver without constructing the Jacobian. In Chapter 5 we discuss the construction of multigrid preconditioners using a replacement operator in order to avoid calculating the Jacobian of the original system $\underline{\mathbf{F}}$.

Chapter 5

Multigrid Methods

Multigrid (MG) methods are a class of iterative methods specifically designed to solve equation systems arising from discretized differential equations. Pioneering work dates back to 1964, where a multigrid algorithm for the Poisson equation on a square was formulated [37]. MG methods are linearly convergent and it has been demonstrated for a large class of partial differential equations, amongst others for the Navier-Stokes equations, that so-called textbook multigrid efficiency can be achieved. This means that the convergence rate is independent of the mesh size and the solution can be approximated in only a few iterations. Multigrid solvers are a standard tool in many codes to solve differential equations, even though the theory is established only for elliptic problems in most cases. Good introductions to the basic principles of multigrid methods can be found in [22, 52, 124, 141].

For many differential operators with periodic boundary conditions it has been noticed that the eigenvectors of the discretized problem are discrete evaluations of the eigenfunctions of the analytic problem, which happen to be periodic. The fundamental idea of multigrid methods is to divide the error of the current iteration into low and high frequency parts. A so-called smoother damps the high frequency parts in a few iterations. The low frequency parts can be approximated on a coarser grid, where the transformed problem is taken care of in a space with fewer unknowns using the same smoother. This leads to a recursive method on multiple grids. Applying this procedure iteratively results in a multigrid method.

In this chapter we first explain the concept of basic iterative methods before we describe in more details the elements of geometric multigrid methods for linear problems. We then summarize the basic methodology of the local Fourier analysis, a technique to analyze

multigrid methods and finish this chapter by discussing multigrid preconditioners.

1 Stationary Iterative Methods

As we have seen in previous chapters, solving differential equations numerically results in linear equation systems

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{5.1}$$

with $\mathbf{A} \in \mathbb{R}^{m \times m}$ and $\mathbf{x}, \mathbf{b} \in \mathbb{R}^m$. Classical stationary iterative schemes are fixed-point methods to solve (5.1). We refer to [113] for a good overview over the history of iterative methods from Gauss to modern techniques.

Assume that we have an approximation $\tilde{\mathbf{x}}$ of the solution $\mathbf{x}$. The algebraic error

$$\mathbf{e} := \tilde{\mathbf{x}} - \mathbf{x} \tag{5.2}$$

is not necessarily accessible since $\mathbf{x}$ is unknown. On the other hand, the residual

$$\mathbf{r} := \mathbf{A}\tilde{\mathbf{x}} - \mathbf{b} \tag{5.3}$$

is easy to compute. These two quantities are connected via the residual equation

$$\mathbf{A}\mathbf{e} = \mathbf{r}. \tag{5.4}$$

We can solve (5.4) for the unknown algebraic error $\mathbf{e}$ and with the use of (5.2) and (5.3) we are able to calculate the solution $\mathbf{x}$:

$$\mathbf{x} = (\mathbf{I} - \mathbf{A}^{-1}\mathbf{A})\tilde{\mathbf{x}} + \mathbf{A}^{-1}\mathbf{b}. \tag{5.5}$$

However, (5.5) uses the exact inverse $\mathbf{A}^{-1}$ of the system matrix $\mathbf{A}$. Thus, we could just solve (5.1) directly. The idea is to replace $\mathbf{A}^{-1}$ by some non-singular approximation $\mathbf{N}^{-1}$ to get

$$\mathbf{x} \approx (\mathbf{I} - \mathbf{N}^{-1}\mathbf{A})\tilde{\mathbf{x}} + \mathbf{N}^{-1}\mathbf{b}.$$

Doing this approximation iteratively yields in the kth step a linear fixed point iteration

$$\mathbf{x}^{(k)} = \mathbf{M}\mathbf{x}^{(k-1)} + \mathbf{N}^{-1}\mathbf{b}, \tag{5.6}$$

with $\mathbf{M} := \mathbf{I} - \mathbf{N}^{-1}\mathbf{A}$, i.e. $\mathbf{A} = \mathbf{N} - \mathbf{N}\mathbf{M}$, for an initial guess $\mathbf{x}^{(0)}$. Moreover, $\mathbf{M}$ as well as $\mathbf{N}^{-1}$ are independent of the iteration count k. $\mathbf{N}\mathbf{x} = \mathbf{N}\mathbf{M}\mathbf{x} + \mathbf{b}$ is consistent with $\mathbf{A}\mathbf{x} = \mathbf{b}$ and if $\mathbf{x}$ is a fixed point of (5.6), it solves the original problem (5.1).

Several choices of $\mathbf{N}$ correspond to well-known iterative methods, see for instance [141]. We want to mention for example Richardson, Gauss-Seidel, Jacobi and successive over-relaxation (SOR), which are given by

$$\begin{aligned}
\mathbf{N}_R &= \mathbf{I}, \\
\mathbf{N}_{GS} &= \mathbf{D}, \\
\mathbf{N}_{JA} &= \mathbf{L} + \mathbf{D}, \\
\mathbf{N}_{SOR} &= \frac{1}{\omega(2-\omega)}(\mathbf{D}+\mathbf{U})^{-1}\mathbf{D}(\mathbf{D}+\mathbf{L}),
\end{aligned} \tag{5.7}$$

with $\mathbf{L}$ the strict lower left, $\mathbf{D}$ the diagonal and $\mathbf{U}$ the strict upper right part of $\mathbf{A}$ and a relaxation factor ω [112, 124, 141].

The convergence of iterative methods (5.6) depends on the spectral radius of the system matrix $\mathbf{M}$:

Theorem 5.1 ([141]). A linear iterative scheme (5.6) converges for any $\mathbf{b} \in \mathbb{R}^m$ to the solution of (5.1) if and only if $\rho(\mathbf{M}) < 1$. The spectral radius $\rho(\mathbf{M})$ is called the *asymptotic convergence factor*.

The asymptotic convergence factor can be interpreted as the the worst factor by which the error is reduced in each iteration. It is called asymptotic since it predicts the worst-case error reduction over many iterations, but might in general not predict the behavior of the error for a single iteration [22].

From Theorem 5.1 it is clear that small values of $\rho(\mathbf{M})$ result in a fast convergence rate. However, for the methods presented so far, we usually get $\rho(\mathbf{M}) \approx 1$ for a fine discretization, which results in slow convergence, i.e. deteriorating convergence after a few iterations [22]. This behavior is caused by the fact that high frequency error modes are damped efficiently in the first iterations, but not low frequency error components [52]. However, these components can be mapped onto a coarser grid, where their frequency changes and they can be damped efficiently. This method is called *multigrid* due to the multilevel grid hierarchy the problem is considered on, and is presented in the next section.

2 Geometric Multigrid

The pioneering multigrid method was designed 1964 in [37] for the Poisson equation,

$$\begin{aligned}
&-u_{xx} = f(x), \quad x \in (0,1),\, f \in \mathcal{C}((0,1),\mathbb{R}), \\
&u(0) = u(1) = 0.
\end{aligned} \tag{5.8}$$

Later it was generalized to other problems [141]. These methods are referred to as *geometric multigrid* methods, since they are based on discretizations and meshing the computational domain. An important extension of geometric methods are *algebraic multigrid* (AMG) methods, which are designed for problems where no partial differential equation or geometrical problem background is used to construct the multilevel hierarchy. We consider in the following only geometric multigrid methods and call them simply multigrid methods. For more information about AMG we refer to [115, 124].

Discretizing (5.8) with a second order finite difference discretization with $m+1$ discretization points and fixed mesh width $\Delta x = \frac{1}{m+1}$ results in the system

$$\mathbf{A}\mathbf{u} = \mathbf{b},$$

with

$$\mathbf{A} = \frac{1}{\Delta x^2}\begin{pmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{m-1\times m-1},$$

$$\mathbf{b} = \begin{pmatrix} f(x_1) \\ \vdots \\ f(x_{m-1}) \end{pmatrix} \in \mathbb{R}^{m-1}.$$

The eigenvalues of $\mathbf{A}$ are given by

$$\lambda_k = \frac{4}{\Delta x^2}\sin^2\left(\frac{\theta_k}{2}\right), \quad \theta_k = \frac{k\pi}{m},\ k = 1, \dots, m-1,$$

and the corresponding eigenvectors by

$$\boldsymbol{\omega}_k = \big(\sin(\theta_k), \sin(2\theta_k), \dots, \sin((m-1)\theta_k)\big)^\top, \quad k = 1, \dots, m-1. \tag{5.9}$$

For $\Delta x \to 0$ they are the discrete evaluations of the eigenfunctions

$$\phi(x) = \sin(k\pi x), \quad k \in \mathbb{N},$$

of $-u_{xx}$, which are sine functions of increasing oscillation satisfying the boundary conditions. The eigenvectors can be divided into low frequency, also called smooth, and high frequency, also called oscillatory, vectors. This can be seen in Figure 5.1 for $m = 10$, where the upper five plots show low frequencies and the lower four plots high frequencies.

In general, we can classify the frequency of the eigenvectors (5.9) in the following way:

$$\omega_k \text{ is a mode of } \begin{cases} \text{low frequency, if } k = 1, \dots, \frac{m}{2}, \\ \text{high frequency, if } k = \frac{m+1}{2}, \dots, m-1. \end{cases} \tag{5.10}$$

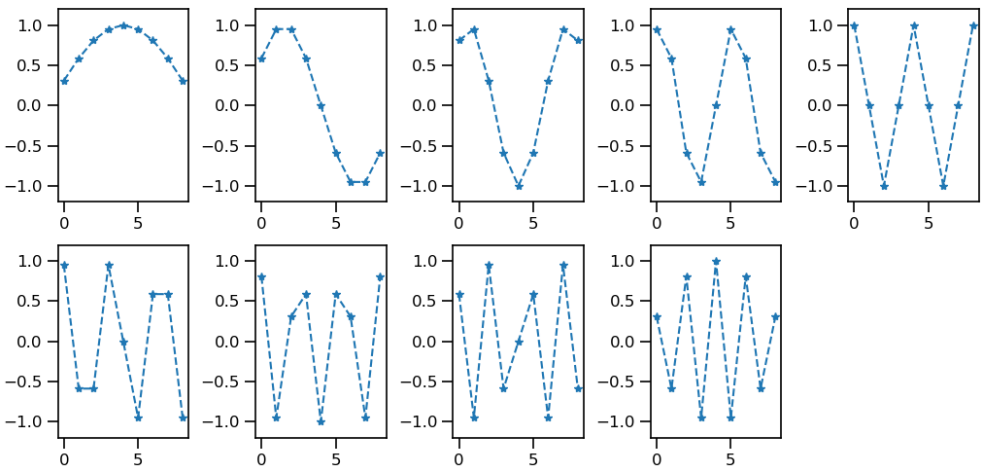


Figure 5.1: Eigenvectors of the one-dimensional discretized Poisson equation for $m = 10$ on the fine grid: low frequencies top, high frequencies bottom.

For a coarse grid defined by dropping every other grid point, only the smooth eigenvectors can be represented without loosing information about some frequencies, see Figure 5.2. In fact, the high frequency vectors are not distinguishable from the low frequency ones on the coarse grid. In other words, only low frequency modes are visible on the coarse grid. Moreover, some of the low frequencies are classified as high frequencies on the coarser grid since this grid only consists of half of the nodes.

This observation leads to the simple idea behind multigrid algorithms: Classical iterative methods as (5.7) damp high frequency error modes in a few iterations, while working poorly on low frequency modes [52]. The latter ones can without loss of information be represented on a coarser grid, where some of them appear as high frequencies and can be damped using the same iterative method as on the fine grid. Due to their property to smoothen high frequency error modes, these iterative methods are called *smoothers* in this context.

Multigrid methods usually consist of the following steps: On the fine level, the high frequency error parts are smoothed. On the coarse level, the residual equation (5.4) is solved for the error. Then the fine level solution is corrected by the prolongated error from the coarse level. This can be done iteratively on a hierarchy of grids Ω_ℓ, where

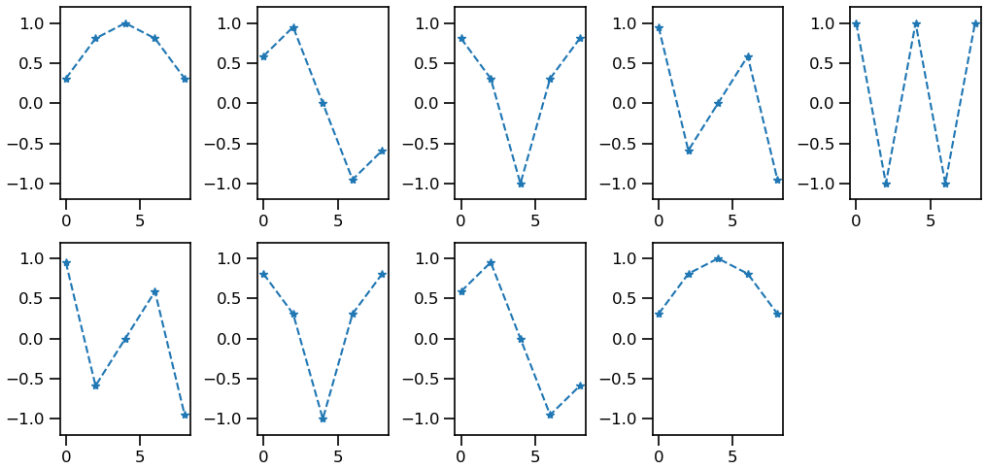


Figure 5.2: Eigenvectors of the one-dimensional discretized Poisson equation for $m = 10$ displayed on every other grid point.

each grid is denoted by its level ℓ, with a smaller index corresponding to a coarser grid. A restriction operator $\mathbf{R}_{\ell-1}^{\ell}$ is used to transfer a grid function from level ℓ to the next coarser level $\ell - 1$ and a prolongation operator $\mathbf{P}_{\ell}^{\ell-1}$ for the reverse operation.

There exist several options on the coarsest grid: either the problem is solved using an iterative method, or it is solved directly. Another option is to not solve the coarse grid problem but to apply the smoother, possibly several times.

Multigrid methods are tailored to the differential equation to be solved. When considering a different PDE, the steps described above have to be repeated: The eigenfunctions of the continuous operator and their discrete counterparts have to be calculated. Next, the coarse space has to be defined and it has to be determined which eigenvectors can be represented on this space. With this information at hand, a smoother can be chosen to take care of the high frequency error components. Thus, the classification of high and low frequency error components depends on both the equation to be solved and the numerical discretization method applied to the PDE. Moreover, all multigrid components, i.e. smoother, restriction and prolongation, need to be chosen accordingly. For space-time discretizations, a space-time multigrid method can be applied, which includes coarsening in the temporal direction.

2.1 Elements of Multigrid

Algorithm 5.1 shows the pseudo code for one multigrid iteration to approximate the solution of a linear system $\mathbf{Ax} = \mathbf{b}$. Here, we denote the smoothing operators of the form

(5.6) on grid level ℓ by $\mathbf{M}_{S,\ell}$ and $\mathbf{N}_{S,\ell}$ and the system matrix and right hand side by $\mathbf{A}_\ell$ and $\mathbf{b}_\ell$. These operators are discussed in more details in the following subsection. Algorithm 5.1 gives rise to an iterative method of the form

$$\mathbf{x}_\ell^{k+1} = \mathbf{M}_{MG}\mathbf{x}_\ell^k + \mathbf{N}_{MG}^{-1}\mathbf{b}_\ell. \tag{5.11}$$

```
MG(x_ℓ, b_ℓ, ℓ):
x_ℓ = M_{S,ℓ} x_ℓ + N_{S,ℓ}^{-1} b_ℓ (pre-smoothing)
if ℓ > 0 then
    r_{ℓ-1} = R_{ℓ-1}^{ℓ} (A_ℓ x_ℓ − b_ℓ) (restriction)
    v_{ℓ-1} = 0
    for j = 1, ..., γ do
        v_{ℓ-1} = MG(v_{ℓ-1}, r_{ℓ-1}, ℓ − 1) (coarse-grid iteration)
    end for
    x_ℓ = x_ℓ − P_ℓ^{ℓ-1} v_{ℓ-1} (fine-grid correction)
    x_ℓ = M_{S,ℓ} x_ℓ + N_{S,ℓ}^{-1} b_ℓ (post-smoothing)
end if
```

Algorithm 5.1: Pseudocode multigrid algorithm.

In the case of an ℓ_{max}-level multigrid cycle with $\gamma = 1$ and presmoothing on the coarsest level we can define the system matrix recursively

$$\begin{aligned} &\mathbf{M}_{0,MG} = 0, \text{ and for } \ell = 1, \dots, \ell_{max}: \\ &\mathbf{M}_{\ell,MG} = \mathbf{M}_{S,\ell}\big(\mathbf{I} - \mathbf{P}_\ell^{\ell-1}\mathbf{N}_{\ell-1,MG}^{-1}\mathbf{R}_{\ell-1}^{\ell}\mathbf{A}_\ell\big)\mathbf{M}_{S,\ell}^{-1}, \end{aligned} \tag{5.12}$$

and

$$\begin{aligned} &\mathbf{N}_{0,MG} = \mathbf{N}_{S,0}^{-1}, \text{ and for } \ell = 1, \dots, \ell_{max}: \\ &\mathbf{N}_{\ell,MG}^{-1} = \mathbf{M}_{S,\ell}\big(\mathbf{N}_{S,\ell}^{-1} - \mathbf{P}_\ell^{\ell-1}\mathbf{N}_{\ell-1,MG}^{-1}\mathbf{R}_{\ell-1}^{\ell}\mathbf{A}_\ell\mathbf{N}_{S,\ell}^{-1} + \mathbf{P}_\ell^{\ell-1}\mathbf{N}_{\ell-1,MG}^{-1}\mathbf{R}_{\ell-1}^{\ell}\big) + \mathbf{N}_{S,\ell}^{-1}. \end{aligned} \tag{5.13}$$

In Figure 5.3 a 4-grid iteration with $\gamma = 1$ can be seen. Due to its shape, it is called a *V-cycle*. Figure 5.4 shows a 4-grid iteration with $\gamma = 2$, referred to as *W-cycle*.

A multigrid method consists of several components: $\mathbf{A}_\ell, \mathbf{R}_{\ell-1}^{\ell}, \mathbf{P}_\ell^{\ell-1}, \mathbf{M}_{S,\ell}, \mathbf{N}_{S,\ell}^{-1}$. The first choice is the coarsening strategy. The simplest and frequently used strategy is standard coarsening, where the mesh size is doubled in each direction. In the case of several dimensions, semi-coarsening refers to doubling the mesh size in some directions only. There exist further coarsening strategies which we do not use in this thesis and thus refer to [22, 124] for more details.

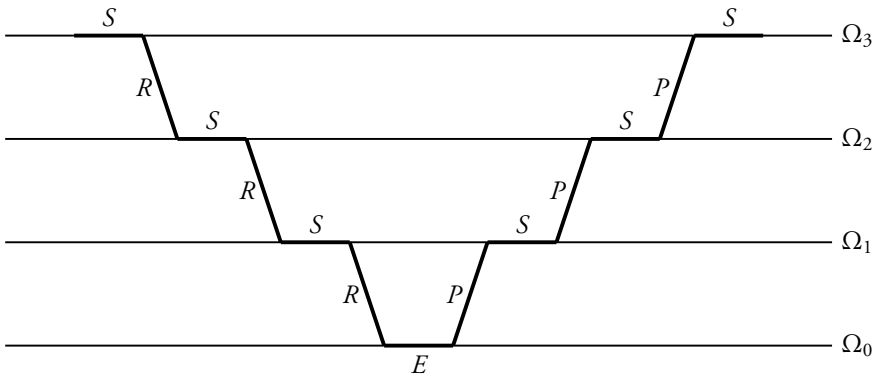


Figure 5.3: V-Multigrid cycle, $\gamma = 1$.

Next, the coarse grid operator $\mathbf{A}_{\ell-1}$ has to be constructed. One possibility is to discretize the problem on each grid level. Another option is to use a so-called *Galerkin approximation* [124, 141]

$$\mathbf{A}_{\ell-1} = \mathbf{R}_{\ell-1}^{\ell} \mathbf{A}_{\ell} \mathbf{P}_{\ell}^{\ell-1}.$$

The choice of transfer operators $\mathbf{R}_{\ell-1}^{\ell}$ and $\mathbf{P}_{\ell}^{\ell-1}$ is connected to the coarsening strategy. For standard coarsening, typical restriction operators are given by injection, full weighting and half weighting. Typical prolongation operators are linear or bilinear interpolation and projection [124, 141].

2.2 Agglomeration Multigrid

In the case of conservation and balance laws, restriction and prolongation operators should be conservative. The corresponding MG method is called *agglomeration multigrid* since the coarse grid is obtained by agglomerating a number of neighboring cells. The fine grid values are summed up, weighted by the volumes of the respective cells and divided by the total volume. This corresponds to an injection operator. For the one-dimensional case, restriction and prolongation are visualized in Figure 5.5 for a non-equidistant grid.

Restriction and prolongation parameters are defined as

$$x_k^{\ell-1} = \frac{\Delta x_i \, x_i^{\ell} + \Delta x_j \, x_j^{\ell}}{\Delta x_i + \Delta x_j},$$
$$x_i^{\ell} = x_k^{\ell-1}, \; x_j^{\ell} = x_k^{\ell-1}.$$

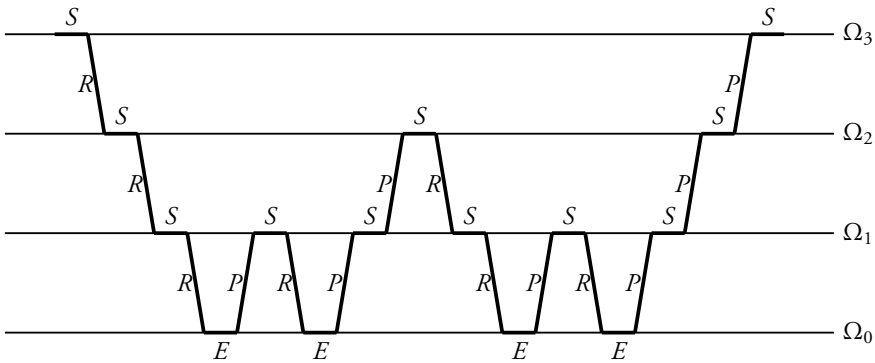


Figure 5.4: W-Multigrid cycle, $\gamma = 2$.

This corresponds to the grid transfer operators

$$\mathbf{R}_{\ell-1}^{\ell} = \begin{pmatrix} \frac{\Delta x_1}{\Delta x_1+\Delta x_1} & \frac{\Delta x_2}{\Delta x_1+\Delta x_1} & & & & \\ & & \frac{\Delta x_3}{\Delta x_3+\Delta x_4} & \frac{\Delta x_4}{\Delta x_3+\Delta x_4} & & \\ & & & \ddots & \ddots & \\ & & & & \frac{\Delta x_{N-1}}{\Delta x_{N-1}+\Delta x_N} & \frac{\Delta x_N}{\Delta x_{N-1}+\Delta x_N} \end{pmatrix},$$

$$\mathbf{P}_{\ell}^{\ell-1} = \begin{pmatrix} 1 & & & \\ 1 & & & \\ & 1 & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \\ & & & 1 \end{pmatrix}.$$

For an equidistant grid in one dimension, the restriction and prolongation operator simplify to

$$\mathbf{R}_{\ell-1}^{\ell} = \frac{1}{2}\begin{pmatrix} 1 & 1 & & & \\ & & 1 & 1 & \\ & & & \ddots & \ddots \\ & & & & 1 \quad 1 \end{pmatrix}, \ \mathbf{P}_{\ell}^{\ell-1} = 2\left(\mathbf{R}_{\ell-1}^{\ell}\right)^{\top}.$$

The corresponding multi-dimensional restriction and prolongation operators can be constructed using a tensor product.

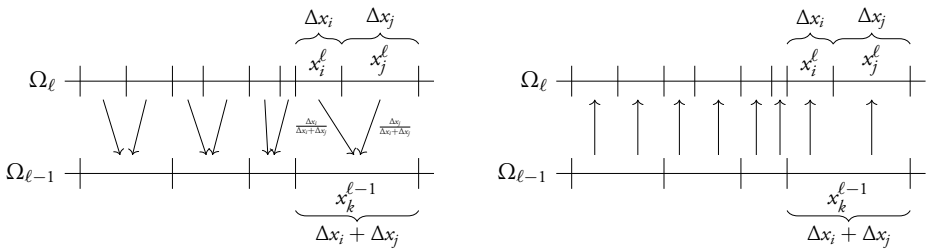


Figure 5.5: Restriction based on agglomeration (left) and prolongation based on injection (right) for a nonequidistant one-dimensional grid.

2.3 Smoothers

Another key aspect of efficient multigrid algorithms is the choice of good smoothers. Classical smoothers are given in (5.7), e.g. Jacobi or Gauss-Seidel. They can also be considered in block form, where the diagonal matrix is replaced by a block-diagonal one with blocks corresponding to some discretization properties. These classical smoothers do not perform well for all problems, e.g. for convection dominated flows [95]. Successful smothers for these problems were constructed using explicit Runge-Kutta schemes, explicit additive Runge-Kutta methods, point implicit smoothers, line implicit smoothers, symmetric Gauss-Seidel and additive W methods [9, 16, 66, 72, 76, 133].

Pseudo Time Iterations

In order to understand the efficiency of smoothers based on time integrating schemes, we first discuss how these schemes can be interpreted as iterative methods in the MG context. They are based on a dual time-stepping approach developed in [65].

Let us consider the linear equation

$$\mathbf{A}_\ell \mathbf{x}_\ell = \mathbf{b}_\ell, \tag{5.14}$$

on multigrid level ℓ. The idea is to add a pseudo time dependence on t^* to the variable $\mathbf{x}_\ell$. With a pseudo time derivative for an initial guess $\mathbf{x}_\ell^0$ we get an initial value problem

$$\begin{aligned} &\frac{\partial \mathbf{x}_\ell}{\partial t^*} + \mathbf{A}_\ell \mathbf{x}_\ell(t^*) - \mathbf{b}_\ell = \mathbf{0}, \\ &\mathbf{x}_\ell(t_0^*) = \mathbf{x}_\ell^0. \end{aligned} \tag{5.15}$$

Any convergent numerical time integration method can now be interpreted as an iterative

method to compute the solution of (5.14) as long as the IVP has a steady state with

$$\mathbf{x}_\ell^* = \lim_{t^*\to\infty} \mathbf{x}_\ell(t^*). \tag{5.16}$$

The matrix $\mathbf{A}_\ell$ is based on a numerical discretization in space and a time-stepping method, e.g. $\mathbf{A}_\ell = \mathbf{I}_\ell + \frac{a\Delta t}{\Delta x}\mathbf{B}_\ell$, when discretizing the linear advection equation with FV and implicit Euler, see (2.3). It can be guaranteed that (5.16) has a steady state when constructing $\frac{a}{\Delta x}\mathbf{B}_\ell$ such the numerical scheme is stable.

An s-stage Runge-Kutta scheme for (5.15) reads

$$\begin{aligned} \mathbf{x}_\ell^{(0)} &= \mathbf{x}_\ell^0, \\ \mathbf{x}_\ell^{(j)} &= \mathbf{x}_\ell^{(0)} - \alpha_j \Delta t^* (\mathbf{A}_\ell \mathbf{x}_\ell^{(j-1)} - \mathbf{b}_\ell), \quad j = 1, \dots, s, \\ \mathbf{x}_\ell^{n+1} &= \mathbf{x}_\ell^{(s)}, \end{aligned} \tag{5.17}$$

with pseudo time step width Δt^* and problem dependent parameters α_j.

Efficient MG schemes to solve the one-dimensional steady Euler equations around an airfoil were developed in [68], with smoothers based on the symmetric Gauss-Seidel method. Unfortunately, these methods do not perform well for the Navier-Stokes equations on high aspect grids. For these equations, additive W smoothers were shown to be efficient [16]. They are a special choice of Rosenbrock methods where the Jacobian is replaced by an approximation. The scheme is given by

$$\begin{aligned} \mathbf{x}_\ell^{(0)} &= \mathbf{x}_\ell^0, \\ \mathbf{x}_\ell^{(j)} &= \mathbf{x}_\ell^{(0)} - \alpha_j \Delta t^* \mathbf{W}^{-1} (\mathbf{A}_\ell \mathbf{x}_\ell^{(j-1)} - \mathbf{b}_\ell), \quad j = 1, \dots, s, \\ \mathbf{x}_\ell^{n+1} &= \mathbf{x}_\ell^{(s)}, \end{aligned} \tag{5.18}$$

with $\mathbf{W} \approx \mathbf{I} + \eta \Delta t^* \mathbf{A}_\ell$, pseudo time step width Δt^* and problem dependent parameters α_j and η.

The specific approximation to define $\mathbf{W}$ is based on a symmetric Gauss-Seidel approach, as suggested in [118] and modified in [67]. The first step is to approximate the Jacobian using a different first order discretization, based on a splitting

$$\mathbf{A} = \mathbf{A}^+ + \mathbf{A}^-$$

of the flux Jacobian $\mathbf{A}$. This is evaluated in the average of the values on both sides of the interface. The split Jacobians $\mathbf{A}^+$ and $\mathbf{A}^-$ correspond to positive and negative eigenvalues and can be written in terms of the matrix of right eigenvectors $\mathbf{Q}$ as

$$\mathbf{A}^+ = \mathbf{Q}\Lambda^+\mathbf{Q}^{-1}, \ \mathbf{A}^- = \mathbf{Q}\Lambda^-\mathbf{Q}^{-1},$$

where $\Lambda^{\pm}$ are diagonal matrices containing only the positive and negative eigenvalues, respectively. The values λ in Λ are then bounded away from zero using a parabolic function which takes care of the modulus of the eigenvalue λ, if it is smaller or equal to a fraction ad of the speed of sound a with free parameter $d \in [0, 1]$:

$$|\lambda| = \frac{1}{2}\left(ad + \frac{|\lambda|^2}{ad}\right), \quad |\lambda| \leq ad.$$

With this, the upwind discretization of the split Jacobian in element i is given by

$$\mathbf{x}_{i_{t^*}} + \mathbf{x}_i + \frac{\Delta t}{\Delta x_i}\big((\mathbf{A}^+_{i,i+1}\mathbf{x}_i + \mathbf{A}^-_{i,i+1}\mathbf{x}_{i+1}) - (\mathbf{A}^+_{i-1,i}\mathbf{x}_{i-1} + \mathbf{A}^-_{i-1,i}\mathbf{x}_i)\big) = 0,$$

with the flux Jacobian between elements evaluated at the center of the common edge.

The corresponding approximation of the Jacobian is then used to construct a preconditioner, i.e. the choice of the $\mathbf{W}$ matrix in (5.18). In [16] a block SGS preconditioner

$$\mathbf{W}^{-1} = (\mathbf{D} + \mathbf{L})^{-1}\mathbf{D}(\mathbf{D} + \mathbf{U})^{-1},$$

is considered with block matrices $\mathbf{L}$, $\mathbf{D}$ and $\mathbf{U}$ with 3×3 blocks.

For $\mathbf{L} + \mathbf{D} + \mathbf{U} = \mathbf{I} + \eta\Delta t^*\mathbf{J}$ one obtains

$$\begin{aligned}
\mathbf{L}_{ij} &= -\frac{\eta\Delta t\Delta t_i^*}{\Delta x_i}\mathbf{A}^+_{ij}, \\
\mathbf{U}_{ij} &= \frac{\eta\Delta t\Delta t_i^*}{\Delta x_i}\mathbf{A}^-_{ij}, \\
\mathbf{D}_{ii} &= \mathbf{I} + \eta\Delta t^*\mathbf{I} + \frac{\eta\Delta t\Delta t_i^*}{\Delta x_i}(\mathbf{A}^+_{i,i+1} - \mathbf{A}^-_{i-1,i}).
\end{aligned}$$

Applying this preconditioner requires in the one-dimensional case solving 3×3 systems coming from the diagonal. This can be done directly without much computational effort.

To improve the efficiency of these smoothers, an optimization process can be applied. Either the parameters in the smoother itself are optimized in order to damp high frequency error components more efficiently or the spectral radius of the MG iteration matrix is minimized. Both methods are discussed in [11].

3 Properties

In this section we present a formal tool to analyze multigrid methods and explain how multigrid can be used to construct preconditioners.

3.1 Local Fourier Analysis

Iterative methods converge if and only if $\rho(\mathbf{M}) < 1$ for the iteration matrix $\mathbf{M}$, see Theorem 5.1. Due to the size of the iteration matrix it can be quite difficult to calculate the eigenvalues for multigrid methods, thus analytical tools have been established to approximate them. Brandt introduced in [19] the so-called *local Fourier analysis* (LFA), also referred to as *local mode analysis* or *discrete Fourier analysis* (DFA) [53]. This is a tool to analyze the asymptotic smoothing and convergence behavior of multigrid methods. The technique is called local since it is based on the operator in the interior of its domain, where it is assumed to be represented by a constant discretization stencil. The problem has to be considered on regular infinite grids, therefore boundary conditions are neglected or assumed to be periodic. Nonlinear problems need to to be linearized in a point before applying the LFA. Under these reasonable assumptions the LFA gives an approximation of multigrid convergence factors by connecting the discrete and the frequency space via a Fourier transform.

For operators based on discretized PDEs, the Fourier transform yields so-called symbols, which are of much smaller size than the operators in the discrete space. With the help of this method, smoothing properties of relaxation methods and convergence properties of two-grid methods can be approximated more easily by assessing the spectral radius of the Fourier symbols [124].

As discussed before, multigrid methods have originally been developed for elliptic problems. The LFA was also constructed for these [53, 92], and extended to other problems in [20] and by many others, see e.g. the references in [21]. For many applications the LFA is a good prediction, i.e. theoretical and experimentally measured convergence factors coincide [124]. But there are problems for which the application of the boundary conditions are crucial to the observed performance and neglecting these in the analysis will decrease the accuracy of the analysis. This is for example the case for convection-dominated and parabolic problems [20, 43]. However, since the LFA is an asymptotic theory, the theoretical results will eventually coincide with the numerical ones when considering a very fine grid.

The fundamental quantities of the LFA are Fourier modes and frequencies:

Definition 5.2 (Fourier modes and frequencies [141]). The function

$$\boldsymbol{\varphi}(\theta_k) := [\varphi_1(\theta_k), \dots, \varphi_N(\theta_k)]^\top,$$

with $\varphi_j(\theta_k) := e^{\mathrm{i}j\theta_k}, j = 1, \dots, N \in \mathbb{N}$, is called a *Fourier mode* with *frequency*

$$\theta_k \in \Theta := \left\{ \frac{2k\pi}{N} : k = 1 - \frac{N}{2}, \dots, \frac{N}{2} \right\} \subset (-\pi, \pi].$$

The frequencies Θ can be separated into high and low frequencies

$$\Theta^{low} := \Theta \cap \left(-\frac{\pi}{2}, \frac{\pi}{2}\right], \tag{5.19}$$

$$\Theta^{high} := \Theta \cap \left(\left(-\pi, -\frac{\pi}{2}\right] \cup \left(\frac{\pi}{2}, \pi\right]\right]. \tag{5.20}$$

We recall that the notation of low and high frequencies is of importance when analyzing smoothing properties.

Theorem 5.3 (One-dimensional discrete Fourier transform [141]). Let $\mathbf{u} \in \mathbb{R}^N$ for $N \in \mathbb{N}$. Then the vector $\mathbf{u}$ can be represented as

$$\mathbf{u} = \sum_{k=1-N/2}^{N/2} \hat{\mathbf{u}}_k \boldsymbol{\psi}(\theta_k),$$

with Fourier modes $\boldsymbol{\psi}(\theta_k) = \boldsymbol{\varphi}(\theta_k) \in \mathbb{R}^N$ and frequencies $\theta_k \in \Theta$ from Definition 5.2, and

$$\hat{\mathbf{u}}_k := \frac{1}{N} \sum_{j=0}^{N-1} \mathbf{u}_j \varphi_j(-\theta_k).$$

The discrete Fourier transform can be extended to the multi-dimensional case:

Theorem 5.4 (d-dimensional discrete Fourier transform [141]). Let $\underline{\mathbf{u}} \in \mathbb{R}^{N_1 \cdots N_d}$ for $N_1, \ldots, N_d \in \mathbb{N}$. Then each subvector $\mathbf{u}_{j_1,\ldots,j_d}$ of $\underline{\mathbf{u}}$ can be represented as

$$\mathbf{u}_{j_1,\ldots,j_d} = \sum_{k_1=1-N_1/2}^{N_1/2} \cdots \sum_{k_d=1-N_d/2}^{N_d/2} \hat{\mathbf{u}}_{k_1,\ldots,k_d} \boldsymbol{\psi}_{j_1,\ldots,j_d}(\theta_{k_1}, \ldots, \theta_{k_d}),$$

with $\boldsymbol{\psi}_{j_1,\ldots,j_d}(\theta_{k_1}, \ldots, \theta_{k_d}) = \varphi_{j_1}(\theta_{k_1}) \cdots \varphi_{j_d}(\theta_{k_d})$ and

$$\hat{\mathbf{u}}_{k_1,\ldots,k_d} := \frac{1}{N_1 \cdots N_d} \sum_{j=0}^{N_1-1} \cdots \sum_{j=0}^{N_d-1} \mathbf{u}_{j_1,\ldots,j_d} \varphi_{j_1}(-\theta_{k_1}) \cdots \varphi_{j_d}(-\theta_{k_d}).$$

For the proofs of the previous two theorems we refer to [141].

One key property of the LFA is the shifting equality. In the following Lemma we consider the case $d = 2$.

Lemma 5.5 (Shifting equality [141]). Let $\theta_x \in \Theta_x$, $\theta_y \in \Theta_y$. Then the shifting equalities read

$$\begin{aligned}\boldsymbol{\psi}_{i-1,j}(\theta_x,\theta_y) &= e^{-\mathrm{i}\theta_y}\boldsymbol{\psi}_{i,j}(\theta_x,\theta_y), \quad i = 2,\dots,N_x,\\ \boldsymbol{\psi}_{i,j-1}(\theta_x,\theta_y) &= e^{-\mathrm{i}\theta_x}\boldsymbol{\psi}_{i,j}(\theta_x,\theta_y), \quad j = 2,\dots,N_y.\end{aligned}$$

Applying the shifting equality to the discrete Fourier transform of a discretization operator results in a block diagonalized operator which is called *Fourier symbol*. Instead of computing the spectral radius of a discrete operator, the spectral radius of the smaller Fourier symbol can be calculated by taking the maximum of the spectral radius over all phase angles θ_i between $-\pi$ and π.

Usually, two-grid schemes are considered when applying the LFA. In order to calculate two-grid convergence factors for a multigrid method using the LFA, the Fourier symbols of the multigrid operator (5.12) need to be determined. This is done by calculating the Fourier symbols of the respective operator, namely the smoother, the restriction and prolongation as well as the discretization operator on the fine and coarse grid. The smoothing factor can be calculated in a similar way. The result of the analysis is then used as an approximation of the asymptotic convergence and smoothing factors of the original operator.

4 Linear Multigrid Preconditioners

We recall that linear MG methods are iterative methods of the form

$$\mathbf{x}^{k+1} = \mathbf{M}_{MG}\mathbf{x}^k + \mathbf{N}_{MG}^{-1}\mathbf{b}, \tag{5.21}$$

where the matrices are defined recursively

$$\begin{aligned}&\mathbf{M}_{0,MG} = 0, \text{ and for } \ell = 1,\dots,\ell_{max}:\\ &\mathbf{M}_{\ell,MG} = \mathbf{M}_{S,\ell}\big(\mathbf{I} - \mathbf{P}_\ell^{\ell-1}\mathbf{N}_{\ell-1,MG}^{-1}\mathbf{R}_{\ell-1}^{\ell}\mathbf{A}_\ell\big)\mathbf{M}_{S,\ell}^{-1},\end{aligned}$$

and

$$\begin{aligned}&\mathbf{N}_{0,MG} = \mathbf{N}_{S,0}^{-1}, \text{ and for } \ell = 1,\dots,\ell_{max}:\\ &\mathbf{N}_{\ell,MG}^{-1} = \mathbf{M}_{S,\ell}\big(\mathbf{N}_{S,\ell}^{-1} - \mathbf{P}_\ell^{\ell-1}\mathbf{N}_{\ell-1,MG}^{-1}\mathbf{R}_{\ell-1}^{\ell}\mathbf{A}_\ell\mathbf{N}_{S,\ell}^{-1} + \mathbf{P}_\ell^{\ell-1}\mathbf{N}_{\ell-1,MG}^{-1}\mathbf{R}_{\ell-1}^{\ell}\big) + \mathbf{N}_{S,\ell}^{-1}.\end{aligned}$$

If $\mathbf{N}_{MG}^{-1} = \mathbf{A}^{-1}$, we would only need one iteration to solve (5.21). The better a MG method is designed, the better is the approximation $\mathbf{A}^{-1} \approx \mathbf{N}_{MG}^{-1}$. Thus, multigrid

methods can not only be used as iterative solvers but also to construct preconditioners for an external iterative solver. We refer to [19] and [53] for more details about multigrid preconditioners. In the following we present our idea to construct such preconditioners for JFNK solvers for DG-SEM based on a FV replacement operator.

4.1 Finite Volume Based Multigrid Preconditioners

A general overview over different preconditioning options for JFNK solvers is given in [78]. Multigrid preconditioners can be applied to the finite difference approximation as described in (4.10). The preconditioner can be implemented Jacobian free as well, this depends on the choice of the smoother. Unfortunately, more advanced smoothers as the $W3$ method cannot be implemented fully Jacobian free. This is still an open problem in the construction of these smoothers. Besides that, multigrid is an interesting option to construct efficient low storage preconditioners. For JFNK solvers these preconditioners have been used for the Navier-Stokes equations [14, 79, 77, 100], for the Fokker–Planck transport equation [24], for flow simulations [71] and for multi material equilibrium radiation diffusion problems [108]. In the context of DG methods this has been studied for instance in [93, 102].

Our core idea for constructing a preconditioner is to replace the Jacobian of the DG discretization by the Jacobian of a first order FV replacement operator, i.e. a DG discretization of order zero. This choice is motivated by the equivalence of the DG-SEM and high order FV discretizations [40]. It has been shown that efficient preconditioners for JFNK can be based on simpler operators than the Jacobian of the system [78]. Moreover, this replacement operator allows to use available knowledge about fast multigrid methods for FV discretizations on block structured meshes.

An overview over the work flow in the construction process of the preconditioned solver can be seen in Figure 5.6. First, the PDE is discretized using a DG-SEM, yielding in an initial value problem

$$\dot{\underline{\mathbf{u}}} = \underline{\mathbf{G}}(\underline{\mathbf{u}}),\ \underline{\mathbf{u}}_0 = \underline{\mathbf{u}}(\underline{\mathbf{0}}). \tag{5.22}$$

Applying an implicit time-stepping method to (5.22) results in a nonlinear system

$$\underline{\mathbf{F}}(\underline{\mathbf{u}}^{n+1}) = \underline{\mathbf{0}}. \tag{5.23}$$

With implicit Euler time-stepping we have for instance $\underline{\mathbf{F}}(\underline{\mathbf{u}}^{n+1}) = \underline{\mathbf{u}}^{n+1} - \underline{\mathbf{u}}^{n} - \Delta t\,\underline{\mathbf{G}}(\underline{\mathbf{u}}^{n+1})$, see Chapters 2 and 3 for more details.

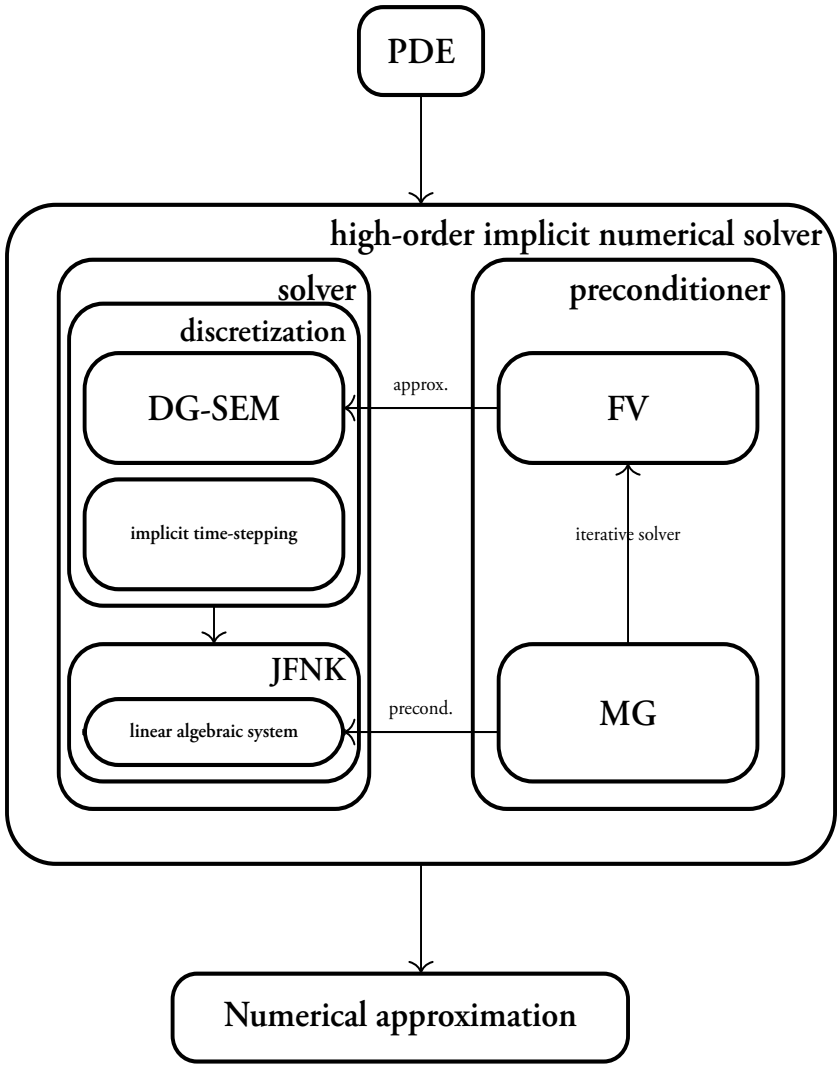


Figure 5.6: Work flow to construct a MG based preconditioner using a FV replacement operator to solve implicit DG-SEM discretizations.

A Jacobian-free Newton method

$$\begin{aligned} \frac{\underline{\mathbf{F}}(\underline{\mathbf{u}}^{(k)} + \varepsilon \Delta \underline{\mathbf{u}}) - \underline{\mathbf{F}}(\underline{\mathbf{u}}^{(k)})}{\varepsilon} &= -\underline{\mathbf{F}}(\underline{\mathbf{u}}^{(k)}), \\ \underline{\mathbf{u}}^{(k+1)} &= \underline{\mathbf{u}}^{(k)} + \Delta \underline{\mathbf{u}}, \quad k = 0, 1, 2, \ldots \end{aligned} \tag{5.24}$$

is used to solve the nonlinear system (5.23), as discussed in Chapter 4.

The linear system in (5.24) is solved using a preconditioned GMRES method. In order to avoid constructing the Jacobian of the DG discretization in (5.22), we construct a first order FV replacement discretization:

$$\underline{\dot{\mathbf{u}}} = \underline{\tilde{\mathbf{G}}}(\underline{\mathbf{u}}),\ \underline{\mathbf{u}}_0 = \underline{\mathbf{u}}(\underline{\mathbf{0}}).$$

To this end, we introduce subcells in each DG element, such that the number of DOFs is retained, i.e. the number of subcells corresponds to the number of nodes in each element. Moreover, the DOFs of both discretizations need to be transferred between the corresponding grids.

For the FV discretization, the Jacobian can be computed more easily and a right preconditioner for the GMRES sub-solver can be constructed using multigrid methods with smoothers based on the dual time-stepping ansatz.

The construction of such preconditioners raises several difficulties:

1. How to transfer the problem between the DG and the FV space?
2. How should the multigrid method be constructed?
3. How can the suggested preconditioner be improved?

We consider several strategies for the transfer between the DG and the FV space. It is not obvious how to assign the DOFs when comparing the FV and DG discretizations. The simplest and most intuitive option is an *ad-hoc assignment*. The idea is to assign DOFs in the DG-SEM and FV discretization based on their location in the grid, after a possible permutation, since these should correspond to similar values. Examples for two-dimensional DG-SEM grids with inlaid equidistant FV subgrids with 4 respective 8 LGL nodes can be seen in Figure 5.7. As it can be noticed, this only gives a reasonable assignment if each FV element contains exactly one LGL node, which is more likely the case of lower order DG discretizations. For higher order discretizations, DOFs are assigned that do not have good approximations to their actual value in the respective space.

The advantage of the ad-hoc assignment is its computational simplicity. But it is difficult to motivate this ansatz for higher order DG approximations when it is not trivial to assign an LGL node to a specific volume. In order to avoid this, the FV subgrid can be adapted to be non-equidistant, such that each volume contains exactly one quadrature node, which is located in the center of the FV element. However, this option increases the work in the multigrid method.

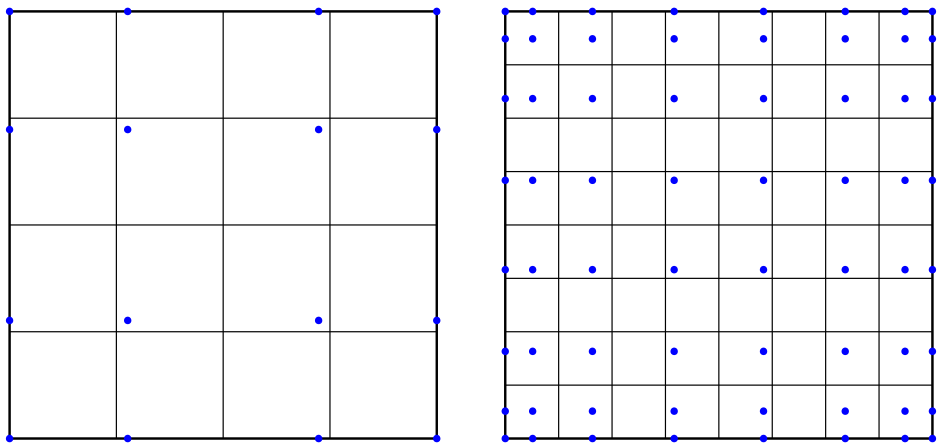


Figure 5.7: Two DG-SEM grid reference elements with LGL nodes (blue nodes), and an inlaid finite volume subgrid (black lines).

Since errors of DG methods are measured in the L_2 norm, another possibility is to construct transfer functions using L_2 projections based on either the FV or the DG-SEM basis functions.

Definition 5.6 (L_2 projection)**.** Let V be a subspace of $L^2(\Omega)$ and let S be a finite dimensional subspace of V. The L_2 *projection* of a function $f \in V$ onto S is defined by the unique $Pf \in V$ that satisfies

$$(f - Pf, v)_{L_2} = 0 \quad \forall s \in S.$$

Given a basis $\{\varphi_i\}_{i=1}^N$ of S, this corresponds to

$$\sum_{j=1}^{N} \alpha_j (\varphi_i, \varphi_j)_{L_2} = (f, \varphi_i)_{L_2}, \quad i = 1 \ldots, N.$$

By solving the linear system $\mathbf{Ax} = \mathbf{b}$ with $a_{ij} = (\varphi_i, \varphi_j)_{L_2}$, $x_j = \alpha_j$ and $b_i = (f, \varphi_i)_{L_2}$, the L_2 projection can be written

$$Pf(x) = \sum_{j=1}^{N} \alpha_j \varphi_j(x).$$

The two function spaces we consider correspond to the FV space and the DG-SEM space. Let us from now on denote the basis functions in the FV space by $\{\boldsymbol{\varphi}_i^{FV}\}_{i=1}^N$ and the basis

functions in the DG space by $\{\boldsymbol{\varphi}_i^{DG}\}_{i=1}^N$. To simplify the notation, we assume that the d-dimensional basis functions are stored in a long vector of length $N = \prod_{i=1}^d N_{x_i}$ with N_{x_i} the length in dimension i.

We define the basis functions of the FV space using the d-dimensional indicator function

$$\chi_{e_i^{FV}}(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in e_i^{FV}, \\ 0, & \mathbf{x} \notin e_i^{FV}, \end{cases}$$

with volumes e_i^{FV}, $i = 1, \ldots, N$, defined via the subgrid on the reference element $[-1, 1]^d$. To get an orthonormal basis, we need to normalize the basis functions. Since the volume of each element on the subgrid on the reference element is $\frac{2}{N}$, the value of the constant FV basis function is $\frac{N}{2}$ in one dimension.

The DG-SEM basis is defined by the Lagrange functions evaluated at the LGL nodes. In the d-dimensional case, the basis functions are given by

$$\boldsymbol{\varphi}_{k_1,\ldots,k_d}(\mathbf{x}) := \prod_{i=1}^d \varphi_{k_i}(x^i),$$

based on the LGL nodes $\{x_j^{DG}\}_{j=1}^{N_{x_i}}$, $i = 1, \ldots, d$, in all d directions and $\mathbf{x} = [x^1, \ldots, x^d]$.

Projection Onto the FV Space

Since the FV basis functions are orthonormal, we can define a projection matrix via the inner L_2 product of the DG-SEM Lagrange basis and the FV basis:

$$P_{FV}\boldsymbol{\varphi}_i^{DG}(\mathbf{x}) = \sum_{j=1}^{N_j} \frac{(\boldsymbol{\varphi}_i^{DG}, \boldsymbol{\varphi}_j^{FV})_{L_2}}{(\boldsymbol{\varphi}_j^{FV}, \boldsymbol{\varphi}_j^{FV})_{L_2}} \boldsymbol{\varphi}_j^{FV}(\mathbf{x}), \quad i = 1, \ldots, N_i.$$

This can be used to construct a projection matrix $\mathbf{P}_{FV}$ for the DOFs on each reference element $\hat{\Omega}$ with

$$(\mathbf{P}_{FV})_{ij} = \frac{(\boldsymbol{\varphi}_i^{DG}, \boldsymbol{\varphi}_j^{FV})_{L_2}}{(\boldsymbol{\varphi}_j^{FV}, \boldsymbol{\varphi}_j^{FV})_{L_2}}.$$

The transfer matrix for all DOFs is then defined as a block diagonal matrix with blocks $\mathbf{P}_{FV}$. To transfer the DOFs back from the FV space to the DG space, either the reconstruction $\mathbf{P}_{FV}^{-1}$ can be used or an L_2 projection onto the DG space.

Projection Onto the DG Space

In order to construct the L_2 projection onto the DG space, we first need to orthogonalize the LGL basis polynomials $\{\boldsymbol{\varphi}_i^{DG}\}_{i=1}^N$. This can be done using the Gram-Schmidt process, see for instance [123], which can be expressed with a triangular matrix $\mathbf{L}$ s.t. $\mathbf{L}\boldsymbol{\varphi}^{DG} = \tilde{\boldsymbol{\varphi}}^{DG}$.

We define a projection matrix via the inner L_2 product of the orthogonal DG-SEM Lagrange basis and the FV basis functions:

$$\tilde{P}_{DG}\boldsymbol{\varphi}_i^{FV}(\mathbf{x}) = \sum_{j=1}^{N_j} \frac{(\boldsymbol{\varphi}_i^{FV}, \tilde{\boldsymbol{\varphi}}_j^{DG})_{L_2}}{(\tilde{\boldsymbol{\varphi}}_j^{DG}, \tilde{\boldsymbol{\varphi}}_j^{DG})_{L_2}} \tilde{\boldsymbol{\varphi}}_j^{DG}(\mathbf{x}), \quad i = 1, \dots, N_i.$$

Then the projection matrix $\tilde{\mathbf{P}}_{DG}$ for the DOFs on each reference element is given by

$$(\tilde{\mathbf{P}}_{DG})_{ij} = \frac{(\boldsymbol{\varphi}_i^{FV}, \tilde{\boldsymbol{\varphi}}_j^{DG})_{L_2}}{(\tilde{\boldsymbol{\varphi}}_j^{DG}, \tilde{\boldsymbol{\varphi}}_j^{DG})_{L_2}}.$$

Again, the transfer matrix for all DOFs is the block diagonal matrix with blocks $\tilde{\mathbf{P}}_{DG}$.

We note that

$$\tilde{\mathbf{P}}_{DG} = (\boldsymbol{\varphi}^{FV}, \tilde{\boldsymbol{\varphi}}^{DG})_{L_2} = (\boldsymbol{\varphi}^{FV}, \mathbf{L}\boldsymbol{\varphi}^{DG})_{L_2} = \mathbf{L}(\boldsymbol{\varphi}^{FV}, \boldsymbol{\varphi}^{DG})_{L_2} = \mathbf{L}\mathbf{P}_{FV}^\top.$$

The coefficients $\tilde{\mathbf{u}} = [\tilde{u}_1, \dots, \tilde{u}_N]^\top$ in the orthogonal polynomial basis $\{\tilde{\boldsymbol{\varphi}}^{DG}\}$ can be mapped to coefficients $\mathbf{u} = [u_1, \dots, u_N]^\top$ in the original LGL basis $\{\boldsymbol{\varphi}^{DG}\}$ using $\mathbf{L}^\top$, since

$$\tilde{\mathbf{u}}^\top \tilde{\boldsymbol{\varphi}}^{DG} = \tilde{\mathbf{u}}^\top \mathbf{L}\mathbf{L}^{-1}\tilde{\boldsymbol{\varphi}}^{DG} = (\mathbf{L}^\top \tilde{\mathbf{u}})^T \boldsymbol{\varphi}^{DG} = \mathbf{u}^\top \boldsymbol{\varphi}^{DG}.$$

Thus, the L_2 projection matrix mapping coefficients from the FV to the DG space is given by

$$\mathbf{P}_{DG} = \mathbf{L}^\top \tilde{\mathbf{P}}_{DG} = \mathbf{L}^\top \mathbf{L}\mathbf{P}_{FV}.$$

It holds

$$(\mathbf{L}^\top \mathbf{L})^{-1} = \mathbf{L}^{-1}(\tilde{\boldsymbol{\varphi}}^{DG}, \tilde{\boldsymbol{\varphi}}^{DG})_{L_2}(\mathbf{L}^\top)^{-1} = (\boldsymbol{\varphi}^{DG}, \boldsymbol{\varphi}^{DG})_{L_2} = \mathbf{M},$$

with the mass matrix $\mathbf{M}$ defined as $(\mathbf{M})_{ij} = (\boldsymbol{\varphi}_i^{DG}, \boldsymbol{\varphi}_j^{DG})_{L_2}$. Hence, the projection matrix onto the DG space can be calculated as

$$\mathbf{P}_{DG} = \mathbf{M}^{-1}\mathbf{P}_{FV}.$$

As before, two options exist to transfer the DOFs back from the FV space to the DG space: either the reconstruction $\mathbf{P}_{DG}^{-1}$ can be used or an L_2 projection onto the FV space.

In Chapter 6 we present numerical results for this FV based MG preconditioner with different transfer strategies.

Chapter 6

Numerical Results

In this chapter we present numerical results for the two main topics of this thesis: the construction of efficient multigrid preconditioners for Jacobian-free Newton-Krylov solvers for high-order DG discretizations and the implementation and analysis of space-time DG-SEM. More detailed discussions of the results can be found in the publications at the end of this thesis.

1 Finite Volume Based Multigrid Preconditioners

In the following we give a summary of the numerical results for the suggested preconditioner presented in Chapter 5. Moreover, we present new numerical experiments from an ongoing collaboration which are not published yet.

1.1 One-Dimensional Advection and Euler Equations

In Paper I the idea of the suggested FV based MG preconditioner is tested for the one-dimensional linear advection equation

$$u_t + au_x = 0.$$

Pseudo time-stepping Runge-Kutta smoothers of stage 2, 3 and 4 are used for the multigrid preconditioner. The numerical results show a relatively large dependence on the choice of the smoother, which motivates to use more advanced smoothers.

This is done in Paper II, where we tested a $W3$ smoother with a symmetric Gauss-Seidel approximation of the Jacobian, see Chapter 5. Numerical tests are performed for the one-dimensional compressible Euler equations for perfect gas

$$\frac{\partial}{\partial t}\begin{pmatrix}\rho\\ m\\ \rho E\end{pmatrix}+\frac{\partial}{\partial x}\begin{pmatrix}m\\ mv+p\\ Hm\end{pmatrix}=\begin{pmatrix}0\\ 0\\ 0\end{pmatrix},$$

with density ρ, momentum $m=\rho v$, pressure p, energy E and enthalpy $H=E+\frac{p}{\rho}$.

The results for a subsonic test case

$$(\rho_0, v_0, p_0) = (1+\sin(2\pi x/10), 1, 28)$$

with Mach number 0.16 are shown in Figure 6.1. The FV reference preconditioner gives fast initial convergence, which is crucial to get fast termination within an inexact Newton's method. There is almost no loss in performance when increasing the order of the DG method. The MG preconditioners approximate the reference preconditioner very well, especially in the first 20 GMRES iterations. We refer to Section 6 in Paper II for more details on the numerical experiments and results.

1.2 Two-Dimensional Advection-Diffusion Equations

Paper III is based on the Master's thesis [74]. The preconditioner is applied to the advection dominated linear advection diffusion equation in two dimensions

$$\mathbf{u}_t + \mathbf{b}\cdot\nabla\mathbf{u} - \varepsilon\Delta\mathbf{u} = \mathbf{g}(\mathbf{u}), \quad (\mathbf{x}, t)\in\Omega\times[0,T], \tag{6.1}$$

with advection speed $\mathbf{b}$ and diffusion constant $\varepsilon\in\mathbb{R}^+$.

The preconditioner is implemented in the Distributed and Unified Numerics Environment (DUNE) [10, 29]. This is a free and open source software for grid-based numerical solvers for PDEs providing FV and DG discretizations. DUNE has Python bindings for central components, which allow to use the software without touching any of the C++ code [30, 28]. This is of advantage for rapid prototyping of new methods, which can be done in Python by expressing the weak form of the PDE symbolically using the domain specific unified form language (UFL) [1].

We discretize problem (6.1) with DG-SEM in space and implicit Euler time-stepping. An upwind flux is used for the convective numerical fluxes and an interior penalty Galerkin flux for the diffusive part. As before, we construct a FV based MG preconditioner for the GMRES solver, with 3-stage Runge-Kutta pseudo time-stepping smoothers.

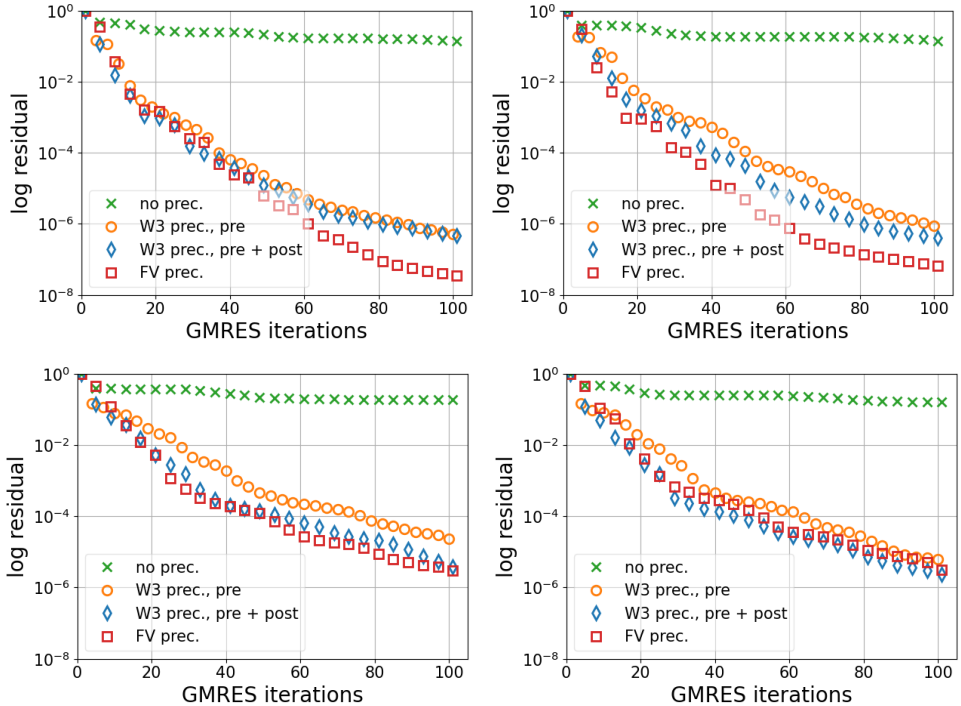


Figure 6.1: Convergence history for GMRES for one Newton iteration, subsonic case, DG order 4 (top) and 8 (bottom), 240 DOFs (left) and 480 DOFs (right).

We tested the preconditioners for a pulse-like initial guess

$$\mathbf{u}_y|_{\Omega_{SN}} = \mathbf{0}, \ \mathbf{u}|_{\partial\Omega_{WE}} = \mathbf{0},$$
$$\mathbf{u}(\mathbf{x}, 0) = \exp(-10\|\mathbf{x}\|^2),$$

with $(\mathbf{x}, t) \in \Omega \times (0, T]$, $\Omega = [-1, 1]^2$, $\mathbf{b} = \frac{1}{2}(\sqrt{3}, 1)$ and $\varepsilon = 1.e^{-4}$.

The focus of Paper III is the use of different transfer options discussed in Chapter 5. New transfer functions between the FV and the DG-SEM grid are introduced and tested. A comparison of their influence on the reference preconditioner can be seen in Figure 6.2. The L_2 projections yield a better mapping of the DOFs than the ad hoc assignment, with the projection onto the FV grid slightly outperforming the projection onto the DG-SEM grid.

We refer to Section 4 in Paper III for more details on the numerical experiments and results of the MG preconditioner for the different transfer strategies.

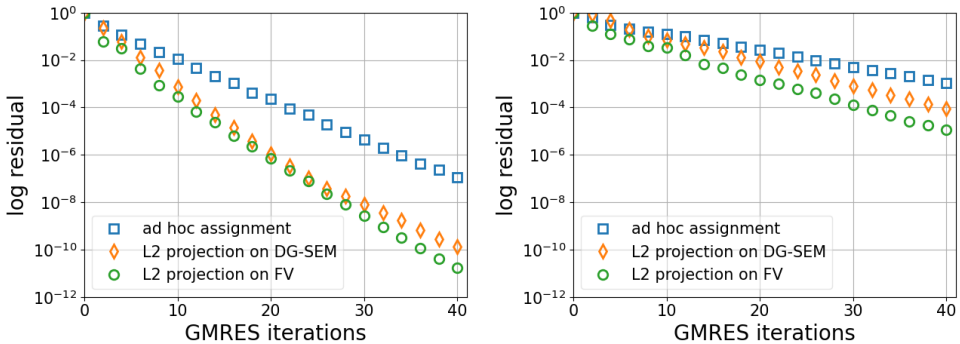


Figure 6.2: GMRES convergence rate for the reference FV preconditioner for different transfer strategies, 16384 DOFs for DG order 4 (left) and DG order 8 (right).

1.3 Two-Dimensional Euler Equations

The following numerical experiments are the result of an ongoing collaboration with Stéphane Gaudreault and Vincent Magnoux at Environment and Climate Change Canada and part of their project GEF (French acronym for GEM en Éléments Finis). The Global Environmental Multiscale Model (GEM) is an integrated forecasting and data assimilation system.

We consider the two-dimensional Euler equations with gravity in the atmosphere

$$\frac{\partial}{\partial t}\begin{pmatrix}\rho\\ \rho u\\ \rho w\\ \rho\theta\end{pmatrix}+\frac{\partial}{\partial x}\begin{pmatrix}\rho u\\ \rho u^2+p\\ \rho uw\\ \rho\theta u\end{pmatrix}+\frac{\partial}{\partial z}\begin{pmatrix}\rho w\\ \rho uw\\ \rho w^2+p\\ \rho\theta w\end{pmatrix}=\begin{pmatrix}0\\ 0\\ -\rho g\\ 0\end{pmatrix}, \tag{6.2}$$

with density ρ, velocities u, w in x- and z-direction, pressure p, potential temperature θ and the gravitational constant g.

The system (6.2) is discretized in space using the direct flux reconstruction (DFR) scheme, which is equivalent to the weak form of the nodal discontinuous Galerkin method based on Gauss-Legendre nodes in one dimension and tensor product bases for multidimensional elements [63, 110]. We refer to [49] for more details about the discretization. The advection upstream splitting method (AUSM) developed in [90] is used as numerical flux function on a square grid.

The spatial discretization results in an ODE

$$\dot{\underline{\mathbf{u}}} = \underline{\mathbf{G}}(\underline{\mathbf{u}}),$$

and is solved using the second order integrator RAT2, which for the nth timestep reads

$$\left(\mathbf{I} - \frac{\Delta t}{2} \left.\frac{\partial \underline{\mathbf{G}}}{\partial \underline{\mathbf{u}}}\right|_{\underline{\mathbf{u}}^n}\right) \underline{\mathbf{x}}^n = \underline{\mathbf{G}}(\underline{\mathbf{u}}^n), \qquad (6.3)$$
$$\underline{\mathbf{u}}^{n+1} = \underline{\mathbf{u}}^n + \Delta t \underline{\mathbf{x}}^n.$$

The Jacobian is replaced by its complex step matrix-vector product approximation

$$\left.\frac{\partial \underline{\mathbf{G}}}{\partial \underline{\mathbf{u}}}\right|_{\underline{\mathbf{u}}^n} \underline{\mathbf{v}} \approx \mathrm{Im}\frac{\underline{\mathbf{G}}(\underline{\mathbf{u}}^n + \mathrm{i}\varepsilon \underline{\mathbf{v}})}{\varepsilon},$$

with $\varepsilon \approx 10^{-8}$.

Finally, the nonlinear system in (6.3) is solved using flexible GMRES (FGMRES) [111].

We construct a preconditioner for FGMRES based on a FV replacement operator as described before. The mapping of the DOFs between the DG and FV grids is based on an evaluation of the DG Lagrange polynomials at the position of the FV points. An agglomeration multigrid preconditioner for this operator is constructed using 3-stage Runge-Kutta smoothers with one pre- and one post-smoothing step on each grid level. Moreover, we test a p-MG preconditioner: Instead of using a FV replacement operator, a p-MG coarsening is applied on the DG operator.

As initial numerical test we studied a rising cold bubble, which models the evolution of a bubble in a constant potential temperature environment. The initial conditions are similar to [109] and result in a bubble with Gaussian profile. On a rectangle computational domain Ω with zero flux boundary conditions we have a distribution of the potential temperature within the bubble given by

$$\theta = \begin{cases} \theta_0 + A_1 \exp\left(-\frac{r_1^2}{s_1^2}\right), & r_1 \leq a_2, \\ \theta_0 + A_1 \exp\left(-\frac{r_1^2}{s_1^2}\right) + A_2 \exp\left(-\frac{(r_2 - a_2)^2}{s_2^2}\right), & r_2 > a_2, \end{cases}$$

with $\theta_0 = 30\,^\circ\mathrm{C}$, $A_1 = -0.15\,^\circ\mathrm{C}$, $s_1 = 50\,\mathrm{m}$, $r_1^2 = (x-500)^2 + (z-640)^2$, $A_2 = 0.5\,^\circ\mathrm{C}$, $a_2 = 150\,\mathrm{m}$, $s_2 = 50\,\mathrm{m}$, $r_2^2 = (x-500)^2 + (z-300)^2$, reference pressure $p_0 = 10^5\,\mathrm{Pa}$, gravity constant $g = 9.80616\,\frac{\mathrm{m}}{\mathrm{s}^2}$, specific gas constant for dry air $R_{specific} = 287.05\,\mathrm{J}\cdot\mathrm{K}^{-1}\cdot\mathrm{kg}^{-1}$ and the specific heat constant at constant pressure $c_p = 1005.46\,\mathrm{J}\cdot\mathrm{K}^{-1}\cdot\mathrm{kg}^{-1}$ and constant volume $c_v = c_p - R_{specific}$. Then we get

$$\rho = \frac{p_0}{R\theta_0}\pi(z)^{\frac{c_v}{R}},$$

with the exner pressure function $\pi(z) = 1 - \frac{g}{c_p \theta_0} z$.

The results are produced on an 8-core AMD processor with 16GB of RAM. The plots for FGMRES iterations and work over time to reach a residual of 10^{-5} can be seen in Figure 6.3 for DG order 2 and in Figure 6.4 for DG order 4. In the legend, *p-MG* denotes the p-multigrid preconditioner, *RK3* denotes the FV multigrid preconditioner, *FV* denotes the FV reference preconditioner and the number corresponds to the number of the FV elements respectively the order of the DG approximation on the coarsest level.

First of all it can be noted that all tested preconditioners beat the unpreconditioned case w.r.t. FGMRES iterations. The reference preconditioner gives very promising convergence results. The p-multigrid and the FV multigrid preconditioners give similar results w.r.t. FGRMES iterations, but there is up to a factor of 1000 iterations difference between the reference and the multigrid preconditioners. This difference increases with increasing DG order. It is of advantage to apply more coarsening steps in both multigrid methods to get a better convergence rate which mimics the one of the reference preconditioner.

As expected, the reference preconditioner is very expensive w.r.t. run time. The same holds for the multigrid preconditioners, all of them being more expensive than the unpreconditioned case except for the p-multigrid preconditioner with several coarsening steps for DG order 4. However, when increasing the DG order we notice that some preconditioners, i.e. those with more coarsening steps, mimic the work of the unpreconditioned case in the beginning before getting slightly more expensive. Hence we are not fully competitive yet w.r.t. CPU time.

We expect that optimizing the multigrid preconditioners such that they mimic the FGMRES convergence rate of the reference preconditioner better should result in a decreasing runtime and thus give a well-working preconditioner. As in the numerical tests presented in the previous sections, the choice of the smoothers is of importance for the quality of the preconditioner. We are therefore currently extending the preconditioner with the *W3* smoother.

2 Space-Time DG-SEM

The other main topic of this thesis are space-time DG-SEM. We studied both implementation aspects of space-time discretizations as well as solvers for the resulting space-time system.

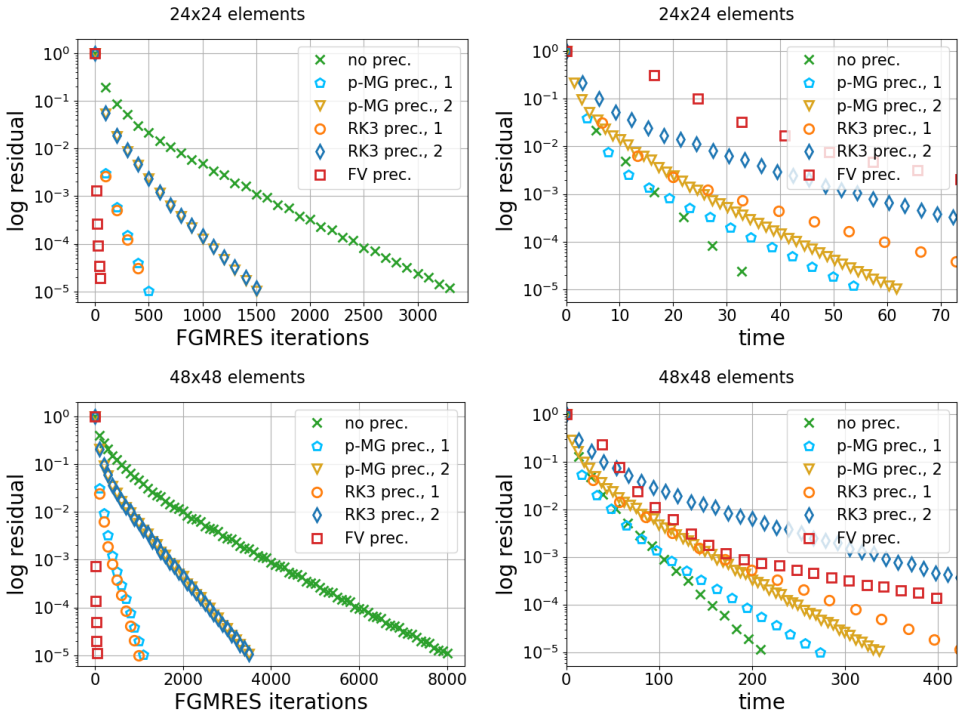


Figure 6.3: Convergence history for the rising bubble test, DG order 2: residual evolution (left) and work in clock time (right).

2.1 Comparison of Space-Time DG-SEM Implementations

In Paper IV we discuss theoretical and practical aspects of space-time DG-SEM implementations. The DG-SEM in time is equivalent to Lobatto IIIC methods, in the sense that they lead to the same discrete solution, see the discussion in Chapter 3 as well as Section 5 in the article. This gives different options to implement a space-time DG-SEM: Either the DG-SEM in space is combined with Lobatto IIIC time-stepping, which we refer to as *LoDG*, or time is considered as an additional dimension and a $d+1$ dimensional space-time DG-SEM problem is considered, which we refer to as *STDG*. For the latter one we use the Python based front-end for `DUNE-FEM` [29]. The Lobatto IIIC method has been implemented in `Assimulo` [2], a Python package that can be readily used together with `DUNE-FEM` for the spatial discretization. However, there are differences in certain key aspects in the respective implementation and interaction with approximate nonlinear solvers, which we highlight and discuss in the article.

Both LoDG and STDG result in nonlinear systems to be solved on each space-time element, see Chapter 2 and 3. This can be done with Newton's method. For the nonlinear

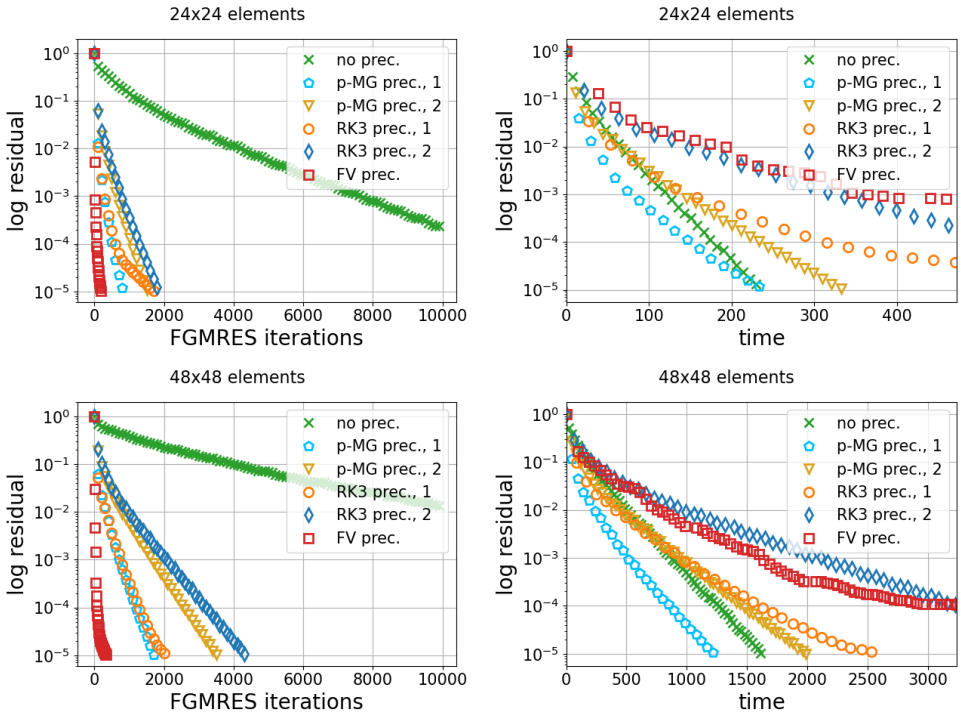


Figure 6.4: Convergence history for the rising bubble test, DG order 4: residual evolution (left) and work in clock time (right).

LoDG system, the Jacobian is given by

$$\underline{\mathbf{I}} - \Delta t(\mathbf{A} \otimes \mathbf{I}_\xi)\mathcal{J}(\underline{\mathbf{F}}),$$

where $\mathcal{J}(\underline{\mathbf{F}})$ contains the Jacobian of the spatial discretization and $\mathbf{A}$ is defined by the Butcher tableau. The Lobatto solver we implemented, as well as many other codes for implicit Runge-Kutta methods, are based on the ideas presented in [54] and instead use the mathematically equivalent Jacobian

$$(\Delta t\mathbf{A})^{-1} \otimes \mathbf{I}_\xi - \mathcal{J}(\underline{\mathbf{F}}).$$

For the nonlinear STDG system the Jacobian is given by

$$\left(\mathbf{D}_\tau^\top \mathbf{M}_\tau - \mathbf{e}_{N_\tau}\mathbf{e}_{N_\tau}^\top\right) \otimes \mathbf{I}_\xi + \frac{\Delta t}{2}(\mathbf{M}_\tau \otimes \mathbf{I}_\xi)\mathcal{J}(\underline{\mathbf{F}}).$$

In Figure 6.5 the sparsity patterns of the Jacobians for the one-dimensional linear advection equation discretized using a single element in space and time for order 1, 2 and 3 are

presented. While the LoDG formulation results in a dense Jacobian, the STDG and the modified LoDG Jacobians have the same number on nonzero elements and a reordering of the unknowns results in the same sparsity pattern. This shows that the efficiency of solvers and preconditioners for the respective discretizations must be deduced through careful testing and depends on the underlying implementation.

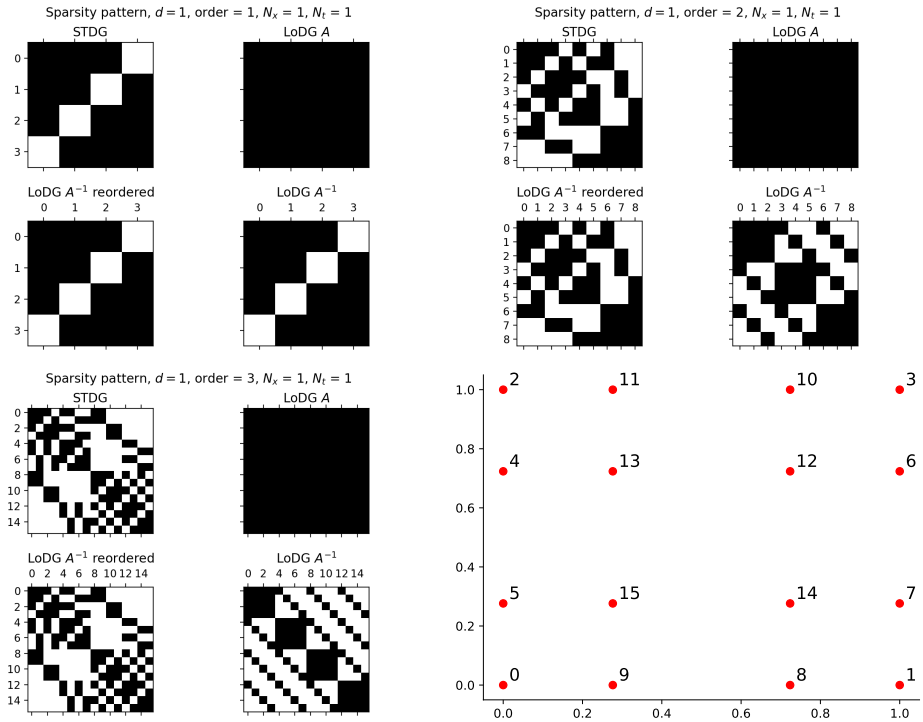


Figure 6.5: Sparsity patterns Jacobian of the advection problem for STDG and LoDG. Node order for one space-time element ($p = 3$) for STDG as generated by `DUNE-FEM` (bottom right).

We refer to Section 6 in the article for more details and a discussion about practical aspects of the respective space-time DG-SEM implementations.

To compare the two implementations, we considered the Euler equations of gas dynamics

$$\frac{\partial}{\partial t}\begin{pmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ \varepsilon\end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix}\rho u\\ \rho u^2 + P\\ \rho uv\\ \rho uw\\ (\varepsilon + P)u\end{pmatrix} + \frac{\partial}{\partial y}\begin{pmatrix}\rho v\\ \rho vu\\ \rho v^2 + P\\ \rho vw\\ (\varepsilon + P)v\end{pmatrix} + \frac{\partial}{\partial z}\begin{pmatrix}\rho w\\ \rho wu\\ \rho wv\\ \rho w^2 + P\\ (\varepsilon + P)w\end{pmatrix} = \begin{pmatrix}0\\0\\0\\0\\0\end{pmatrix}, \tag{6.4}$$

with ρ the density of the fluid, u, v, w the velocities in x-, y- and z-direction and ε the internal energy.

We studied the two- and three-dimensional Euler equations with periodic boundary conditions and vortex initial condition

$$\rho = \left(1 - S^2(\gamma - 1)M^2 \frac{\exp(f)}{(8\pi^2)}\right)^{\frac{1}{\gamma-1}},$$
$$v_1 = 1 - S\mathbf{x}_1 \frac{\exp\left(\frac{f}{2}\right)}{2\pi}, \quad v_2 = S\mathbf{x}_0 \frac{\exp\left(\frac{f}{2}\right)}{2\pi}, \quad v_3 = S\mathbf{x}_2 \frac{\exp\left(\frac{f}{2}\right)}{2\pi},$$
$$\varepsilon = \frac{P}{\gamma - 1} + 0.5\frac{v_1^2 + v_2^2 + v_3^2}{\rho}, \; P = \frac{\rho^\gamma}{\gamma M^2},$$

with vortex strength $S = 5$, Mach number $M = 0.5$, $\gamma = 1.4$ and $f = 1 - x^2 - y^2 - z^2$.

The numerical solutions obtained by the STDG and the LoDG implementation for the two-dimensional Euler test case can be seen in Figure 6.6. A uniform mesh on the space-time domain $[-10, 10]^2 \times (0, 2.5]$ is used in space with $\Delta x = \Delta y = 0.04$ and time steps of uniform size $\Delta t = 0.01$ throughout the simulation. The problem is under-resolved for order 1, leading to a smeared solution. As the DG order in space-time is increased, this phenomenon is reduced. Some differences can be noted in the numerical results for the two implementations. This is likely caused by the fact that two different solvers are used for the nonlinear systems arising from the discretizations. Due to the differences between these solvers we can in general not expect identical numerical solutions despite the mathematical equivalence of the two algorithms.

To show the potential of our STDG code we present a three-dimensional Euler test case. We modify the problem slightly and consider the space-time domain $[-5, 5]^3 \times (0, 2]$, with the initial vortex placed slightly to the left of the center. A uniform mesh is used in space with $\Delta x = \Delta y = \Delta z = 1$ and time steps of uniform size $\Delta t = 0.5$ are used throughout the simulation. The initial condition and the final time element of the density can be seen in Figure 6.7. These results show the potential of the STDG code even for four-dimensional problems in space-time. To the best of our knowledge, this is the first four-dimensional DG-SEM implementation available publicly.

For convergence tests and further numerical results we refer to Section 7 in the article.

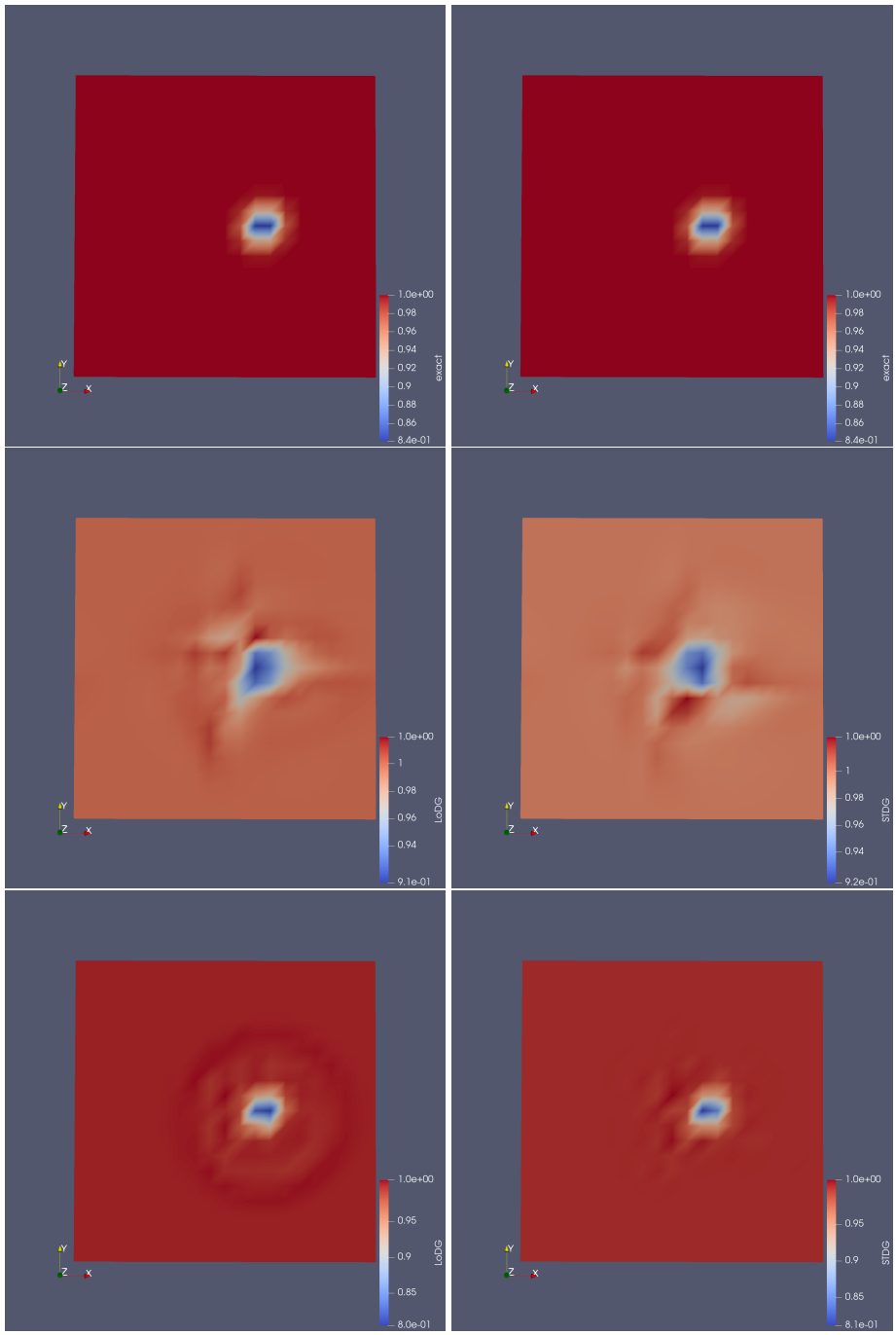


Figure 6.6: Solution of ρ for a vortex problem subject to the two-dimensional Euler equations, exact solution (top), DG order 1 (middle) and DG order 2 (bottom), LoDG solution (left), STDG solution (right).

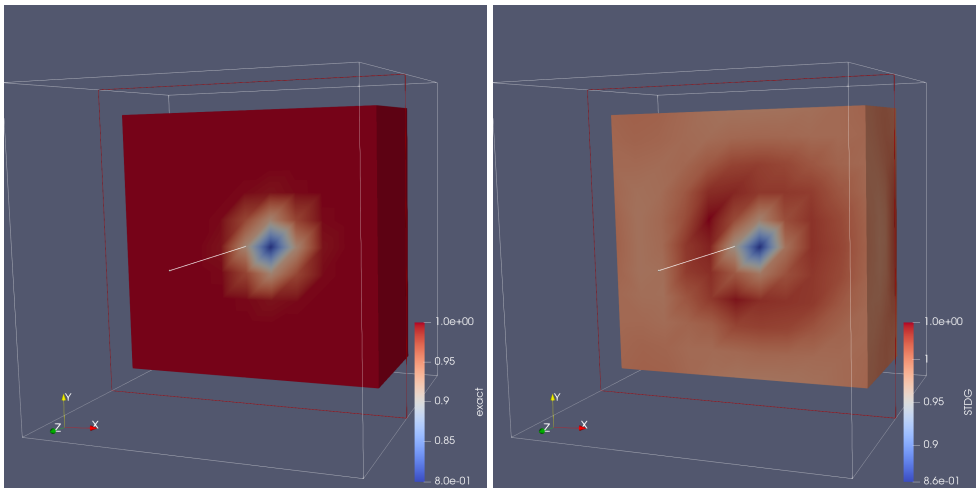


Figure 6.7: Density ρ for a vortex problem subject to the 3D Euler equations using STDG, exact solution (left), STDG solution (right).

2.2 Space-Time Local Fourier Analysis

Finally, in Paper V a local Fourier analysis of a space-time multigrid solver is performed. The linear advection equation

$$u_t + au_x = 0$$

is discretized with a space-time DG method using a FV discretization in space and a DG-SEM in time. The resulting linear system is of block form with bocks corresponding to the space-time discretization and solved with a space-time multigrid method. A weighted block Jacobi smoother is used and two different coarsening strategies are applied: coarsening in space-time as well as coarsening in the temporal direction only.

We performed a smoothing as well as a two-grid analysis and refer to the article for a detailed discussion of the analysis. For large CFL numbers we found an asymptotic smoothing factor of around $\frac{1}{\sqrt{2}}$ for sufficiently high CFL numbers μ.

Since it was not possible to find analytical expressions for the complex eigenvalues of the two-grid operators, we calculated them numerically for N_x volumes in space and N temporal elements. The results are shown in Figure 6.8, giving asymptotic convergence factors of about 0.5 for first order temporal DG-SEM, which can be improved to asymptotic convergence factors of about 0.375 by increasing the DG-SEM in time to second order. We refer to Section 6 in the article for more details.

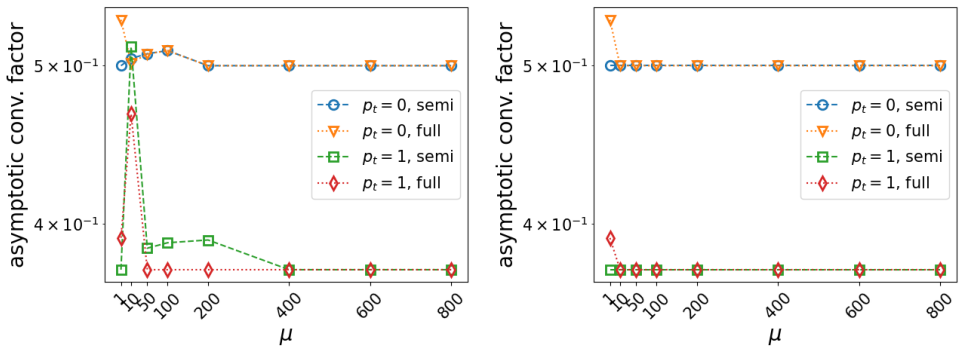


Figure 6.8: Results of the LFA for the test problem: $N_x = 2^5$, $N = 2^3$ (left), $N_x = 2^{10}$, $N = 2^3$ (right).

We also compared the results of the analysis to numerical experiments using the two-grid method as solver. The assumption of periodic boundary conditions in space and time needed for the LFA cannot be used for the numerical tests since this will result in singular systems and iteration matrices. We therefore considered advection problems with nonperiodic boundary conditions discretized with space-time DG-SEM in one and two dimensions.

The asymptotic convergence results for the space-time multigrid solver can be seen in Figure 6.9, giving rates of approximately 0.25 for $p_t = p_x = 0$ and approximately 0.3 for $p_t = p_x = 1$ in the one-dimensional case and approximately 0.25 in the two-dimensional case. We refer to Section 7 in the article for a more detailed discussed of the results.

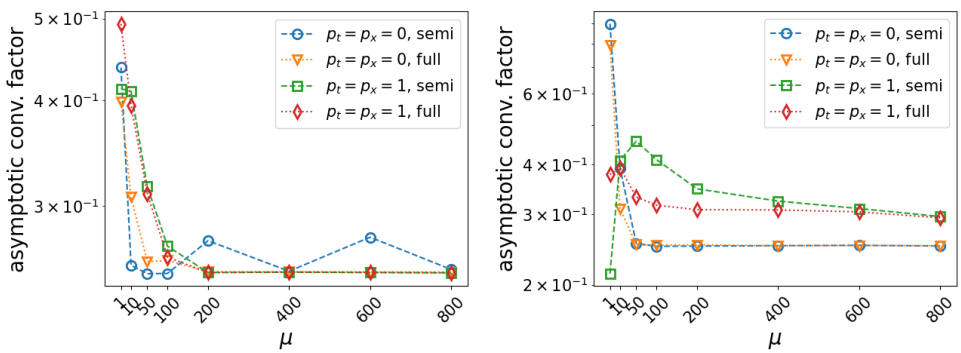


Figure 6.9: Numerical convergence results: one spatial dimension with $N = 2^3$, $N_x = 2^{10}$ (left), two spatial dimensions with $N = 2^3$, $N_x = 2^5$ (right).

Chapter 7

Conclusions and Outlook

1 Summary and Conclusions

Multigrid Preconditioner

In this thesis we have presented a method to construct preconditioners for implicit DG-SEM based on a multigrid method applied to a FV replacement operator. It is an ongoing research topic to find efficient solvers for the nonlinear systems resulting from implicit high order discretizations. Since the number of unknowns increases with increasing polynomial degree and dimension, leading to large dense Jacobian blocks, Jacobian-free solvers are needed. This increases the difficulty to construct good preconditioners.

We use a Jacobian-free Newton-Krylov solver, which is favorable with regards to memory consumption. To improve the convergence speed of the GMRES sub-solver, a preconditioner is needed. Thus the problem arises how to construct a good and efficient preconditioner without using the DG-SEM Jacobian. We suggested to make use of a simple first order FV replacement operator. The choice of this replacement operator is motivated by the equivalence of DG-SEM and a high order FV discretization. Based on this, an agglomeration multigrid preconditioner can be constructed for the GMRES sub-solver. This ansatz allows to avoid constructing the Jacobian of the original DG discretization while keeping the number of DOFs in the replacement operator constant.

Numerical experiments showed the efficiency of the suggested preconditioner, both for linear and nonlinear hyperbolic test problems. For nonlinear problems and problems of higher dimension, the choice of the smoother in the MG preconditioner becomes

more relevant. Since classical smoothers as Jacobi or Gauss-Seidel do not perform well, we constructed smoothers based on pseudo time iterations. However, even with these smoothers more advanced ones need to be used for more complex problems to obtain a good preconditioner.

Moreover, the efficiency of the suggested preconditioner is influenced by the mapping of the DOFs between the FV and the DG-SEM grid. This has to be done carefully, with an L_2 projection improving the performance of the preconditioner compared to a simple adhoc assignment, where the DOFs are assigned based on location.

Space-Time DG-SEM

In recent years, interest in space-time DG methods has increased, mostly due to the possibility of parallelization in the temporal direction. Space-time discretizations are based on discretizing space and time simultaneously, resulting in implicit schemes.

An algebraic equivalence between DG-SEM in time and Lobatto IIIC Runge-Kutta methods has been proven, yielding two approaches for the formulation and implementation of space-time DG-SEM: Either time is treated as an additional coordinate direction and the DG-SEM is applied to the entire problem, or the method of lines ansatz is used with DG-SEM in space and the fully implicit Lobatto IIIC method in time. We compared theoretical properties of the resulting space-time DG-SEM as well as practical aspects as for instance algorithmic and implementation details. Our open source implementations are based on `DUNE-FEM` and `Assimulo` with the aim to provide a user-friendly baseline for further testing.

Depending on the existing code as well as the application in mind, each implementation has its advantages and disadvantages. The advantage of the Lobatto ansatz are its flexibility and the reuse of existing simulation workflows, but this approach might cause several challenges as interfaces, parallelization and more depending on the codes. The space-time DG-SEM code is relatively easy to implement given an existing Galerkin code. Moreover, this approach allows to reuse existing code and provides full control over the implementation. However, it might not be easy to extend existing codes to the $d+1$-dimensional case w.r.t. mesh generation, solvers and visualization for $d = 3$.

One possibility to solve the systems resulting from space-time DG-SEM are multigrid methods. We performed a local Fourier analysis to gain insight into these solvers. The LFA becomes challenging for the advection test problem discretized with DG methods since the resulting Fourier symbols are complex. However, we could find promising asymptotic convergence factors for large CFL numbers. The numerical tests confirmed

the theoretical convergence results, showing that multigrid is a good solver for the space-time DG-SEM as well as a promising basis to construct space-time preconditioners.

2 Future work

Multigrid Preconditioner

We are currently working on extending the suggested preconditioner to three-dimensional problems and in parallel. The challenge here is to parallelize specific pseudo time iteration based smoothers as for instance the $W3$ smoother. Moreover, it is work in progress to add our preconditioner to the open-source software DUNE such that it becomes one of the default preconditioning options for users.

One potential topic for future research could be to further focus on the construction of new numerical methods by optimizing the preconditioner w.r.t. the smoother, the replacement operator or the mapping between the different discretization grids.

Space-Time DG-SEM

With our work on space-time DG-SEM we have laid the background for further development. One of the main topics for future work are efficient solvers for space-time DG-SEM, and of course extending our preconditioner to the space-time setting. The option of coarsening in some directions only could be investigated with regards to the efficiency of the multigrid preconditioner. Again, it would be necessary to parallelize the solver.

References

[1] M. S. Alnæs, A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells. Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Transactions on Mathematical Software (TOMS)*, 40(2):1–37, 2014. doi: 10.1145/2566630.

[2] C. Andersson, C. Führer, and J. Åkesson. Assimulo: A unified framework for ODE solvers. *Mathematics and Computers in Simulation*, 116(0):26 – 43, 2015. doi: 10.1016/j.matcom.2015.04.007.

[3] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9(1):17–29, 1951. doi: 10.1090/qam/42792.

[4] H. Atkins and B. Helenbrook. Super-Convergence of Discontinuous Galerkin Method Applied to the Navier-Stokes Equations. In *19th AIAA Computational Fluid Dynamics*, page 3787. American Institute of Aeronautics and Astronautics, 2009. doi: 10.2514/6.2009-3787.

[5] O. Axelsson. A note on a class of strongly A-stable methods. *BIT Numerical Mathematics*, 12(1):1–4, 1972. doi: 10.1007/bf01932668.

[6] T. Barth and M. Ohlberger. Finite Volume Methods: Foundation and Analysis. *Encyclopedia of computational mechanics*, 1(15):1–57, 2004. doi: 10.1002/9781119176817.ecm2010.

[7] F. Bassi and S. Rebay. GMRES Discontinuous Galerkin Solution of the Compressible Navier-Stokes Equations. In *Discontinuous Galerkin Methods*, pages 197–208. Springer, Berlin, Heidelberg, 2000. doi: 10.1007/978-3-642-59721-3_14.

[8] F. Bassi, A. Crivellini, D. A. Di Pietro, and S. Rebay. An implicit high-order discontinuous Galerkin method for steady and unsteady incompressible flows. *Computers & Fluids*, 36(10):1529–1546, 2007. doi: 10.1016/j.compfluid.2007.03.012.

[9] F. Bassi, A. Ghidoni, and S. Rebay. Optimal Runge-Kutta smoothers for the p-multigrid discontinuous Galerkin solution of the 1D Euler equations. *Journal of Computational Physics*, 230(11):4153–4175, 2011. doi: 10.1016/j.jcp.2010.04.030.

[10] P. Bastian, M. Blatt, A. Dedner, N.-A. Dreier, C. Engwer, R. Fritze, C. Gräser, C. Grüninger, D. Kempf, R. Klöfkorn, et al. The Dune framework: Basic concepts and recent developments. *Computers & Mathematics with Applications*, 81:75–112, 2021. doi: 10.1016/j.camwa.2020.06.007.

[11] P. Birken. Optimizing Runge-Kutta smoothers for unsteady flow problems. *Electronic Transactions on Numerical Analysis*, 39(1):298–312, 2012.

[12] P. Birken. *Numerical Methods for the Unsteady Compressible Navier-Stokes Equations*. Habilitation thesis, Universität Kassel, 2012.

[13] P. Birken. *Numerical Methods for Unsteady Compressible Flow Problems*. CRC Press, Boca Raton, London, New York, 2021. doi: 10.1201/9781003025214.

[14] P. Birken and A. Jameson. On nonlinear preconditioners in Newton–Krylov methods for unsteady flows. *International Journal for Numerical Methods in Fluids*, 62 (5):565–573, 2010. doi: 10.1002/fld.2030.

[15] P. Birken, G. Gassner, M. Haas, and C.-D. Munz. Preconditioning for modal discontinuous Galerkin methods for unsteady 3D Navier-Stokes equations. *Journal of Computational Physics*, 240:20–35, 2013. doi: 10.1016/j.jcp.2013.01.004.

[16] P. Birken, J. Bull, and A. Jameson. Preconditioned Smoothers for the Full Approximation Scheme for the RANS Equations. *Journal of Scientific Computing*, 78(2): 995–1022, 2019. doi: 10.1007/s10915-018-0792-9.

[17] M. Bohm, A. R. Winters, G. J. Gassner, D. Derigs, F. Hindenlang, and J. Saur. An entropy stable nodal discontinuous Galerkin method for the resistive MHD equations. Part I: Theory and Numerical Verification. *Journal of Computational Physics*, 422:108076, 2020. doi: 10.1016/J.JCP.2018.06.027.

[18] P. D. Boom and D. W. Zingg. High-Order Implicit Time-Marching Methods Based on Generalized Summation-by-Parts Operators. *SIAM Journal on Scientific Computing*, 37:A2682–A2709, 2015. doi: 10.1137/15M1014917.

[19] A. Brandt. Multi-Level Adaptive Solutions to Boundary-Value Problems. *Mathematics of Computation*, 31(138):333–390, 1977. doi: 10.2307/2006422.

[20] A. Brandt. Multigrid solvers for non-elliptic and singular-perturbation steady-state problems. *The Weizmann Institute of Science, Rehovot, Israel*, page 37, 1981.

[21] A. Brandt. Guide to multigrid development. In *Multigrid Methods*, pages 220–312. Springer, Berlin, Heidelberg, 1982. doi: 10.1007/BFb0069930.

[22] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. Society for Industrial and Applied Mathematics, Philadelphia, 2000. doi: 10.1137/1.9780898719505.

[23] M. H. Carpenter, D. Gottlieb, and S. Abarbanel. The Stability of Numerical Boundary Treatments for Compact High-Order Finite-Difference Schemes. *Journal of Computational Physics*, 108(2):272–295, 1991. doi: 10.1006/jcph.1993.1182.

[24] L. Chacón, D. Barnes, D. Knoll, and G. Miley. An Implicit Energy-Conservative 2D Fokker–Planck Algorithm: II. Jacobian-Free Newton–Krylov Solver. *Journal of Computational Physics*, 157(2):654–682, 2000. doi: 10.1006/jcph.1999.6395.

[25] J. Chan. On discretely entropy conservative and entropy stable discontinuous Galerkin methods. *Journal of Computational Physics*, 362:346–374, 2018. doi: 10.1016/j.jcp.2018.02.033.

[26] T. Chen and C.-W. Shu. Entropy stable high order discontinuous Galerkin methods with suitable quadrature rules for hyperbolic conservation laws. *Journal of Computational Physics*, 345:427–461, 2017. doi: 10.1016/j.jcp.2017.05.025.

[27] F. Chipman. A-stable Runge-Kutta processes. *BIT Numerical Mathematics*, 11(4): 384–388, 1971. doi: 10.1007/BF01939406.

[28] A. Dedner and R. Klöfkorn. Extendible and Efficient Python Framework for Solving Evolution Equations with Stabilized Discontinuous Galerkin Methods. *Communications on Applied Mathematics and Computation*, pages 1–40, 2021. doi: 10.1007/s42967-021-00134-5.

[29] A. Dedner, R. Klöfkorn, M. Nolte, and M. Ohlberger. A generic interface for parallel and adaptive scientific computing: Abstraction principles and the Dune-Fem module. *Computing*, 90:165–196, 2010. doi: 10.1007/s00607-010-0110-3.

[30] A. Dedner, R. Klöfkorn, and M. Nolte. Python Bindings for the DUNE-FEM Module. *Zenodo (March 2020)*, 2020. doi: 10.5281/zenodo.3706994.

[31] P. Deuflhard. *Newton Methods for Nonlinear Problems*. Springer-Verlag Berlin Heidelberg, 2011. doi: 10.1007/978-3-642-23899-4.

[32] L. T. Diosady and S. M. Murman. Tensor-product preconditioners for higher-order space–time discontinuous Galerkin methods. *Journal of Computational Physics*, 330: 296–318, 2017. doi: 10.1016/j.jcp.2016.11.022.

[33] W. Dörfler, S. Findeisen, and C. Wieners. Space-Time Discontinuous Galerkin Discretizations for Linear First-Order Hyperbolic Evolution Systems. *Computational Methods in Applied Mathematics*, 16(3):409–428, 2016. doi: 10.1515/cmam-2016-0015.

[34] B. L. Ehle. *On Padé approximations to the exponential function and A-stable methods for the numerical solution of initial value problems*. Dissertation thesis, University of Waterloo, Ontario, 1969.

[35] S. C. Eisenstat and H. F. Walker. Choosing the Forcing Terms in an Inexact Newton Method. *SIAM Journal on Scientific Computing*, 17(1):16–32, 1996. doi: 10.1137/0917003.

[36] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, J. B. Schroder, and S. Vandewalle. Multigrid methods with space–time concurrency. *Computing and Visualization in Science*, 18(4-5):123–143, 2017. doi: 10.1007/s00791-017-0283-9.

[37] R. P. Fedorenko. The speed of convergence of one iterative process. *USSR Computational Mathematics and Mathematical Physics*, 4(3):227–235, 1964. doi: 10.1016/0041-5553(64)90253-8.

[38] K. J. Fidkowski, T. A. Oliver, J. Lu, and D. L. Darmofal. p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations. *Journal of Computational Physics*, 207(1):92–113, 2005. doi: 10.1016/j.jcp.2005.01.005.

[39] T. C. Fisher and M. H. Carpenter. High-order entropy stable finite difference schemes for nonlinear conservation laws: Finite domains. *Journal of Computational Physics*, 252:518–557, 2013. doi: 10.1016/j.jcp.2013.06.014.

[40] T. C. Fisher, M. H. Carpenter, J. Nordström, N. K. Yamaleev, and C. Swanson. Discretely Conservative Finite-Difference Formulations for Nonlinear Conservation Laws in Split Form: Theory and Boundary Conditions. *Journal of Computational Physics*, 234:353–375, 2013. doi: 10.1016/j.jcp.2012.09.026.

[41] M. Franciolini and S. M. Murman. Multigrid preconditioning for a space-time spectral-element discontinuous-Galerkin solver. In *AIAA Scitech 2020 Forum*, page 1314, 2020. doi: 10.2514/6.2020-1314.

[42] M. Franciolini, A. Crivellini, and A. Nigro. On the efficiency of a matrix-free linearly implicit time integration strategy for high-order Discontinuous Galerkin solutions of incompressible turbulent flows. *Computers & Fluids*, 159:276–294, 2017. doi: 10.1016/J.COMPFLUID.2017.10.008.

[43] S. Friedhoff, S. MacLachlan, and C. Borgers. Local Fourier analysis of space-time relaxation and multigrid schemes. *SIAM Journal on Scientific Computing*, 35(5): S250–S276, 2013. doi: 10.1137/120881361.

[44] L. Friedrich, G. Schnücke, A. R. Winters, D. C. R. Fernández, G. J. Gassner, and M. H. Carpenter. Entropy Stable Space–Time Discontinuous Galerkin Schemes with Summation-by-Parts Property for Hyperbolic Conservation Laws. *Journal of Scientific Computing*, 80(1):175–222, 2019. doi: 10.1007/s10915-019-00933-2.

[45] M. J. Gander. 50 Years of Time Parallel Time Integration. In T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, editors, *Multiple Shooting and Time Domain Decomposition Methods*, pages 69–113, Cham, 2015. Springer. doi: 10.1007/978-3-319-23321-5_3.

[46] M. J. Gander and M. Neumüller. Analysis of a New Space-Time Parallel Multigrid Algorithm for Parabolic Problems. *SIAM Journal on Scientific Computing*, 38(4): A2173–A2208, 2016. doi: 10.1137/15M1046605.

[47] G. J. Gassner. A Skew-Symmetric Discontinuous Galerkin Spectral Element Discretization and Its Relation to SBP-SAT Finite Difference Methods. *SIAM Journal on Scientific Computing*, 35(3):A1233–A1253, 2013. doi: 10.1137/120890144.

[48] G. J. Gassner, A. R. Winters, and D. A. Kopriva. Split form nodal discontinuous Galerkin schemes with summation-by-parts property for the compressible Euler equations. *Journal of Computational Physics*, 327:39–66, 2016. doi: 10.1016/j.jcp.2016.09.013.

[49] S. Gaudreault, M. Charron, V. Dallerit, and M. Tokman. High-order numerical solutions to the shallow-water equations on the rotated cubed-sphere grid. *Journal of Computational Physics*, 449:110792, 2022. doi: 10.1016/j.jcp.2021.110792.

[50] J. Gopalakrishnan and G. Kanschat. A multilevel discontinuous Galerkin method. *Numerische Mathematik*, 95(3):527–550, 2003. doi: 10.1007/s002110200392.

[51] A. Greenbaum, V. Pták, and Z. Strakoš. Any Nonincreasing Convergence Curve is Possible for GMRES. *SIAM Journal on Matrix Analysis and Applications*, 17(3): 465–469, 1996. doi: 10.1137/S0895479894275030.

[52] W. Hackbusch. Parabolic multigrid methods. In R. Glowinski and J.-L. Lions, editors, *Computing Methods in Applied Sciences and Engineering IV*, pages 189–197. Elsevier Science Publisher B.V., North-Holland, Amsterdam, 1984. doi: 10.5555/4673.4714.

[53] W. Hackbusch. *Multi-Grid Methods and Applications*, volume 4. Springer, Berlin, Heidelberg, 2013. doi: 10.1007/978-3-662-02427-0.

[54] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II*, volume 14 of *Springer Series in Computational Mathematics*. Springer, Berlin, Heidelberg, 2010. doi: 10.1007/978-3-642-05221-7.

[55] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy. Uniformly High Order Accurate Essentially Non-oscillatory Schemes, III. In *Upwind and high-resolution schemes*, pages 218–290. Springer, Berlin, Heidelberg, 1987. doi: 10.1016/0021-9991(87)90031-3.

[56] P. W. Hemker, W. Hoffmann, and M. Van Raalte. Two-level Fourier Analysis of a Multigrid Approach for Discontinuous Galerkin Discretization. *SIAM Journal on Scientific Computing*, 25(3):1018–1041, 2003. doi: 10.1137/S1064827502405100.

[57] P. W. Hemker, W. Hoffmann, and M. Van Raalte. Fourier two-level analysis for discontinuous Galerkin discretization with linear elements. *Numerical linear algebra with applications*, 11(5-6):473–491, 2004. doi: 10.1002/nla.356.

[58] M. R. Hestenes and E. Stiefel. Methods of Conjugate Gradients for Solving Linear Systems. *Journal of research of the National Bureau of Standards*, 49:409–435, 1952. doi: 10.6028/jres.049.044.

[59] J. S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. Springer, New York, NY, 2008. doi: 10.1007/978-0-387-72067-8.

[60] J. S. Hesthaven, S. Gottlieb, and D. Gottlieb. *Spectral Methods for Time-Dependent Problems*, volume 21. Cambridge University Press, Cambridge, 2007. doi: 10.1017/CBO9780511618352.

[61] F. Hindenlang, G. J. Gassner, C. Altmann, A. Beck, M. Staudenmaier, and C.-D. Munz. Explicit discontinuous Galerkin methods for unsteady problems. *Computers & Fluids*, 61:86–93, 2012. doi: 10.1016/j.compfluid.2012.03.006.

[62] G. Horton and S. Vandewalle. A Space-Time Multigrid Method for Parabolic Partial Differential Equations. *SIAM Journal on Scientific Computing*, 16:848–864, 1995. doi: 10.1137/0916050.

[63] H. Huynh. Discontinuous Galerkin via Interpolation: The Direct Flux Reconstruction Method. *Journal of Scientific Computing*, 82(3):1–32, 2020. doi: 10.1007/s10915-020-01175-3.

[64] H. T. Huynh. A Flux Reconstruction Approach to High-Order Schemes Including Discontinuous Galerkin Methods. In *18th AIAA Computational Fluid Dynamics Conference*, pages 4079–4120. American Institute of Aeronautics and Astronautics, 2007. doi: 10.2514/6.2007-4079.

[65] A. Jameson. Time Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings. In *10th Computational Fluid Dynamics Conference*, pages 1596–1609. American Institute of Aeronautics and Astronautics, 1991. doi: 10.2514/6.1991-1596.

[66] A. Jameson. Aerodynamics. *Encyclopedia of Computational Mechanics*, 2004. doi: 10.1002/0470091355.ecm062.

[67] A. Jameson. Evaluation of Fully Implicit Runge Kutta Schemes for Unsteady Flow Calculations. *Journal of Scientific Computing*, 73(2-3):819–852, 2017. doi: 10.1007/s10915-017-0476-x.

[68] A. Jameson and D. Caughey. How Many Steps are Required to Solve the Euler Equations of Steady, Compressible Flow: In Search of a Fast Solution Algorithm. In *15th AIAA Computational Fluid Dynamics Conference*, pages 2673–2684. American Institute of Aeronautics and Astronautics, 2001. doi: 10.2514/6.2001-2673.

[69] L. O. Jay. Lobatto methods. In B. Engquist, editor, *Encyclopedia of Applied and Computational Mathematics*, pages 817–826. Springer, Berlin, Heidelberg, 2015. doi: 10.1007/978-3-540-70529-1_123.

[70] C. Johnson and J. Pitkäranta. An Analysis of the Discontinuous Galerkin Method for a Scalar Hyperbolic Equation. *Mathematics of Computation*, 46:1–26, 1986. doi: 10.2307/2008211.

[71] J. E. Jones and C. S. Woodward. Newton–Krylov-multigrid solvers for large-scale, highly heterogeneous, variably saturated flow problems. *Advances in Water Resources*, 24(7):763–774, 2001. doi: 10.1016/S0309-1708(00)00075-0.

[72] R. Kannan and Z. J. Wang. A Study of Viscous Flux Formulations for a p-Multigrid Spectral Volume Navier Stokes Solver. *Journal of Scientific Computing*, 41(2):165–199, 2009. doi: 10.1007/s10915-009-9269-1.

[73] G. Karniadakis and S. Sherwin. *Spectral/hp Element Methods for Computational Fluid Dynamics*. Oxford University Press, Oxford, 2013. doi: 10.1093/acprof:oso/9780198528692.001.0001.

[74] J. Kasimir. Subgrid finite volume preconditioner for Discontinuous Galerkin implemented in the DUNE framework. Master's thesis, Lund University, Sweden, 2021.

[75] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. Society for Industrial and Applied Mathematics, Philadelphia, 1995. doi: 10.1137/1.9781611970944.

[76] C. M. Klaij, M. H. van Raalte, H. van der Ven, and J. J. van der Vegt. h-Multigrid for space-time discontinuous Galerkin discretizations of the compressible Navier-Stokes equations. *Journal of Computational Physics*, 227(2):1024–1045, 2007. doi: 10.1016/j.jcp.2007.08.034.

[77] D. Knoll and V. Mousseau. On Newton–Krylov Multigrid Methods for the Incompressible Navier–Stokes Equations. *Journal of Computational Physics*, 163(1): 262–267, 2000. doi: 10.1006/jcph.2000.6561.

[78] D. A. Knoll and D. E. Keyes. Jacobian-Free Newton-Krylov Methods: A Survey of Approaches and Applications. *Journal of Computational Physics*, 193(2):357–397, 2004. doi: 10.1016/j.jcp.2003.08.010.

[79] D. A. Knoll and W. J. Rider. A Multigrid Preconditioned Newton–Krylov Method. *SIAM Journal on Scientific Computing*, 21(2):691–710, 1999. doi: 10.1137/S1064827598332709.

[80] D. A. Kopriva. *Implementing Spectral Methods for Partial Differential Equations*. Springer, Dordrecht, 2009. doi: 10.1007/978-90-481-2261-5.

[81] D. A. Kopriva, S. L. Woodruff, and M. Y. Hussaini. Computation of electromagnetic scattering with a non-conforming discontinuous spectral element method. *International Journal for Numerical Methods in Engineering*, 53(1):105–122, 2002. doi: 10.1002/nme.394.

[82] A. Kravchenko and P. Moin. On the Effect of Numerical Errors in Large Eddy Simulations of Turbulent Flows. *Journal of Computational Physics*, 131(2):310–322, 1997. doi: 10.1006/jcph.1996.5597.

[83] H.-O. Kreiss and G. Scherer. Finite Element and Finite Difference Methods for Hyperbolic Partial Differential Equations. In C. de Boor, editor, *Mathematical Aspects of Finite Elements in Partial Differential Equations*, pages 195–212. Academic Press, London, 1974. doi: 10.1016/B978-0-12-208350-1.50012-1.

[84] M. Kronbichler. The Discontinuous Galerkin Method: Derivation and Properties. In *Efficient High-Order Discretizations for Computational Fluid Dynamics*, pages 1–55. Springer, Cham, 2021. doi: 10.1007/978-3-030-60610-7_1.

[85] P. Lasaint and P. Raviart. On a Finite Element Method for Solving the Neutron Transport Equation. In C. de Boor, editor, *Mathematical Aspects of Finite Elements in Partial Differential Equations*, pages 89–123. Academic Press, London, 1974. doi: 10.1016/B978-0-12-208350-1.50008-X.

[86] R. J. LeVeque and R. J. Leveque. *Numerical Methods for Conservation Laws*, volume 132. Birkhäuser, Basel, 1992. doi: 10.1007/978-3-0348-5116-9.

[87] R. J. LeVeque et al. *Finite Volume Methods for Hyperbolic Problems*, volume 31. Cambridge University Press, Cambridge, 2002. doi: 10.1017/CBO9780511791253.

[88] P. T. Lin, J. N. Shadid, and P. H. Tsuji. On the performance of Krylov smoothing for fully coupled AMG preconditioners for VMS resistive MHD. *International Journal for Numerical Methods in Engineering*, 120(12):1297–1309, 2019. doi: 10.1002/nme.6178.

[89] V. Linders, J. Nordström, and S. H. Frankel. Properties of Runge-Kutta-Summation-By-Parts methods. *Journal of Computational Physics*, 419:109684, 2020. doi: 10.1016/j.jcp.2020.109684.

[90] M.-S. Liou and C. J. Steffen Jr. A New Flux Splitting Scheme. *Journal of Computational physics*, 107(1):23–39, 1993. doi: 0.1006/jcph.1993.1122.

[91] X.-D. Liu, S. Osher, and T. Chan. Weighted Essentially Non-oscillatory Schemes. *Journal of computational physics*, 115(1):200–212, 1994. doi: 10.1006/jcph.1994.1187.

[92] J. Mandel, S. McCormick, and R. Bank. *Variational Multigrid Theory*, pages 131–177. Society for Industrial and Applied Mathematics, Philadelphia, 1987. doi: 10.1137/1.9781611971057.ch5.

[93] G. May, F. Iacono, and A. Jameson. A hybrid multilevel method for high-order discretization of the Euler equations on unstructured meshes. *Journal of Computational Physics*, 229(10):3938–3956, 2010. doi: 10.1016/j.jcp.2010.01.036.

[94] G. A. Meurant and J. D. Tebbens. *Krylov Methods for Nonsymmetric Linear Systems: From Theory to Computations*, volume 57. Springer, Cham, 2020. doi: 10.1007/978-3-030-55251-0.

[95] W. A. Mulder. A New Multigrid Approach to Convection Problems. In R. G. V. Douglas L. Dwoyer, M. Yousuff Hussaini, editor, *11th International Conference on Numerical Methods in Fluid Dynamics*, pages 429–433, Berlin, Heidelberg, 1989. Springer. doi: 10.1016/0021-9991(89)90121-6.

[96] M. Neumüller. *Space-Time Methods: Fast Solvers and Applications*, volume 20 of *Monograph Series TU Graz: Computation in Engineering and Science*. TU Graz, Graz, 2013.

[97] J. Nievergelt. Parallel methods for integrating ordinary differential equations. *Communications of the ACM*, 7(12):731–733, 1964. doi: 10.1145/355588.365137.

[98] S. Osher and E. Tadmor. On the Convergence of Difference Approximations to Scalar Conservation Laws. *Mathematics of Computation*, 50(181):19–51, 1988. doi: 10.2307/2007913.

[99] W. Pazner and P.-O. Persson. Approximate tensor-product preconditioners for very high order discontinuous Galerkin methods. *Journal of Computational Physics*, 354: 344–369, 2018. doi: 10.1016/j.jcp.2017.10.030.

[100] M. Pernice and M. D. Tocci. A Multigrid-Preconditioned Newton–Krylov Method for the Incompressible Navier–Stokes Equations. *SIAM Journal on Scientific Computing*, 23(2):398–418, 2001. doi: 10.1137/S1064827500372250.

[101] P.-O. Persson and J. Peraire. An Efficient Low Memory Implicit DG Algorithm for Time Dependent Problems. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, pages 113–123. American Institute of Aeronautics and Astronautics, 2006. doi: 10.2514/6.2006-113.

[102] P.-O. Persson and J. Peraire. Newton-GMRES Preconditioning for Discontinuous Galerkin Discretizations of the Navier–Stokes Equations. *SIAM Journal on Scientific Computing*, 30(6):2709–2733, 2008. doi: 10.1137/070692108.

[103] N. Qin, D. Ludlow, and S. Shaw. A matrix-free preconditioned Newton/GMRES method for unsteady Navier-Stokes solutions. *International Journal for Numical Methods in Fluids*, 33:223–248, 2000. doi: 10.1002/(SICI)1097-0363(20000530)33:2<223::AID-FLD10>3.0.CO;2-V.

[104] W. H. Reed and T. Hill. Triangular mesh methods for the neutron transport equation. *Los Alamos Report LA-UR-73-479*, pages 1–23, 1973.

[105] S. Rhebergen, J. J. van der Vegt, and H. van der Ven. Multigrid Optimization for Space-Time Discontinuous Galerkin Discretizations of Advection Dominated

Flows. In *ADIGMA-A European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications*, pages 257–269. Springer, Heidelberg, Berlin, 2010. doi: 10.1007/978-3-642-03707-8_18.

[106] S. Rhebergen, B. Cockburn, and J. J. Van Der Vegt. A space–time discontinuous Galerkin method for the incompressible Navier–Stokes equations. *Journal of computational physics*, 233:339–358, 2013. doi: 10.1016/j.jcp.2012.08.052.

[107] G. Richter. On the Order of Convergence of the Discontinuous Galerkin Method for Hyperbolic Equations. *Mathematics of computation*, 77(264):1871–1885, 2008. doi: 10.1090/S0025-5718-08-02126-1.

[108] W. J. Rider, D. A. Knoll, and G. L. Olson. A Multigrid Newton–Krylov Method for Multimaterial Equilibrium Radiation Diffusion. *Journal of Computational Physics*, 152(1):164–191, 1999. doi: 10.1006/jcph.1999.6240.

[109] A. Robert. Bubble Convection Experiments with a Semi-implicit Formulation of the Euler Equations. *Journal of the Atmospheric Sciences*, 50(13):1865–1873, 1993. doi: 10.1175/1520-0469(1993)050<1865:BCEWAS>2.0.CO;2.

[110] J. Romero, K. Asthana, and A. Jameson. A Simplified Formulation of the Flux Reconstruction Method. *Journal of Scientific Computing*, 67(1):351–374, 2016. doi: 10.1007/s10915-015-0085-5.

[111] Y. Saad. A Flexible Inner-Outer Preconditioned GMRES Algorithm. *SIAM Journal on Scientific Computing*, 14(2):461–469, 1993. doi: 10.1137/0914028.

[112] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, second edition, 2003. doi: 10.1137/1.9780898718003.

[113] Y. Saad. Iterative methods for linear systems of equations: A brief historical journey. In S. Brenner, I. Shparlinski, C.-W. Shu, and D. Szyld, editors, *75 Years of Mathematics of Computation*, pages 197–216. American Mathematical Society, Providence, Rhode Island, 2020. doi: 10.1090/conm/754/15141.

[114] Y. Saad and M. Schultz. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, 7:856–869, 1986. doi: 10.1137/0907058.

[115] Y. Shapira. *Matrix-Based Multigrid: Theory and Applications*, volume 2. Springer, Boston, MA, 2008. doi: 10.1007/978-0-387-49765-5.

[116] J. J. Sudirham, J. J. W. van der Vegt, and R. M. J. van Damme. Space-time discontinuous Galerkin method for advection-diffusion problems on time-dependent domains. *Applied Numerical Mathematics*, 56(12):1491–1518, 2006. doi: 10.1016/j.apnum.2005.11.003.

[117] M. Svärd and J. Nordström. Review of summation-by-parts schemes for initial–boundary-value problems. *Journal of Computational Physics*, 268:17–38, 2014. doi: 10.1016/j.jcp.2014.02.031.

[118] R. C. Swanson, E. Turkel, and C.-C. Rossow. Convergence acceleration of Runge-Kutta schemes for solving the Navier-Stokes equations. *Journal of Computational Physics*, 224(1):365–388, 2007. doi: 10.1016/j.jcp.2007.02.028.

[119] E. Tadmor. Skew-selfadjoint form for systems of conservation laws. *Journal of Mathematical Analysis and Applications*, 103(2):428–442, 1984. doi: 10.1016/0022-247X(84)90139-2.

[120] E. Tadmor. The Numerical Viscosity of Entropy Stable Schemes for Systems of Conservation Laws. I. *Mathematics of Computation*, 49(179):91–103, 1987. doi: 10.2307/2008251.

[121] E. Tadmor. Entropy stability theory for difference approximations of nonlinear conservation laws and related time-dependent problems. *Acta Numerica*, 12:451–512, 2003. doi: 10.1017/S0962492902000156.

[122] M. Tavelli and M. Dumbser. A staggered space–time discontinuous Galerkin method for the three-dimensional incompressible Navier–Stokes equations on unstructured tetrahedral meshes. *Journal of Computational Physics*, 319:294–323, 2016. doi: 10.1016/j.jcp.2016.05.009.

[123] L. N. Trefethen and D. Bau III. *Numerical Linear Algebra*, volume 50. Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[124] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, London, 2001. doi: 10.2307/4148428.

[125] J. van der Vegt and S. Rhebergen. HP-Multigrid as Smoother algorithm for higher order discontinuous Galerkin discretizations of advection dominated flows. Part II: Optimization of the Runge–Kutta smoother. *Journal of Computational Physics*, 231:7564–7583, 2012. doi: 10.1016/j.jcp.2012.05.037.

[126] J. J. van der Vegt. *Space-time discontinuous Galerkin finite element methods*, pages 1–37. Von Karman Institute for Fluid Dynamics, Brussels, 2006.

[127] J. J. van der Vegt and S. Rhebergen. hp-Multigrid as Smoother algorithm for higher order discontinuous Galerkin discretizations of advection dominated flows: Part I. Multilevel analysis. *Journal of computational physics*, 231(22):7537–7563, 2012. doi: 10.1016/j.jcp.2012.05.038.

[128] J. J. van der Vegt and H. van der Ven. Space–Time Discontinuous Galerkin Finite Element Method with Dynamic Grid Motion for Inviscid Compressible Flows: I. General Formulation. *Journal of Computational Physics*, 182(2):546–585, 2002. doi: 10.1006/jcph.2002.7185.

[129] J. J. van der Vegt and Y. Xu. Space–time discontinuous Galerkin method for nonlinear water waves. *Journal of computational physics*, 224(1):17–39, 2007. doi: 10.1016/j.jcp.2006.11.031.

[130] H. Van der Ven and J. J. van der Vegt. Space–time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows: II. Efficient flux quadrature. *Computer methods in applied mechanics and engineering*, 191(41-42):4747–4780, 2002. doi: 10.1016/S0045-7825(02)00403-6.

[131] H. Van der Vorst. Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, 13:631–644, 1992. doi: 10.1137/0913035.

[132] B. Van Leer. Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. *Journal of computational Physics*, 32(1):101–136, 1979. doi: 10.1016/0021-9991(79)90145-1.

[133] B. Van Leer, W.-T. Lee, P. L. Roe, K. G. Powell, and C.-H. Tai. Design of optimally smoothing multistage schemes for the Euler equations. *Communications in Applied Numerical Methods*, 8(10):761–769, 1992. doi: 10.1002/cnm.1630081006.

[134] M. Van Raalte and P. W. Hemker. Two-level multigrid analysis for the convection–diffusion equation discretized by a discontinuous Galerkin method. *Numerical Linear Algebra with Applications*, 12(5-6):563–584, 2005. doi: 10.1002/nla.441.

[135] S. Vangelatos. *On the Efficiency of Implicit Discontinuous Galerkin Spectral Element Methods for the Unsteady Compressible Navier-Stokes Equations*. Dissertation thesis, University of Stuttgart, 2020.

[136] L. M. Versbach. *Multigrid Preconditioners for the Discontinuous Galerkin Spectral Element Method: Construction and Analysis*. Licentiate thesis, Lund University, Sweden, 2020.

[137] P. Vincent, F. Witherden, B. Vermeire, J. S. Park, and A. Iyer. Towards Green Aviation with Python at Petascale. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11, Piscataway, New Jersey, 2016. IEEE Press. doi: 10.1109/SC.2016.1.

[138] P. E. Vincent and A. Jameson. Facilitating the Adoption of Unstructured High-Order Methods Amongst a Wider Community of Fluid Dynamicists. *Mathematical Modelling of Natural Phenomena*, 6(3):97–140, 2011. doi: 10.1051/mmnp/20116305.

[139] L. Wang and D. J. Mavriplis. Implicit solution of the unsteady Euler equations for high-order accurate discontinuous Galerkin discretizations. *Journal of Computational Physics*, 225(2):1994–2015, 2007. doi: 10.1016/j.jcp.2007.03.002.

[140] Z. J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. T. Huynh, et al. High-order CFD methods: current status and perspective. *International Journal for Numerical Methods in Fluids*, 72(8):811–845, 2013. doi: 10.1002/fld.3767.

[141] P. Wesseling. *An Introduction to Multigrid Methods*. R.T. Edwards, Portland, 2004.

[142] N. Wintermeyer, A. R. Winters, G. J. Gassner, and D. A. Kopriva. An Entropy Stable Nodal Discontinuous Galerkin Method for the Two Dimensional Shallow Water Equations on Unstructured Curvilinear Meshes with Discontinuous Bathymetry. *Journal of Computational Physics*, 340:200–242, 2017. doi: 10.1016/j.jcp.2017.03.036.

[143] A. R. Winters, D. Derigs, G. J. Gassner, and S. Walch. A Uniquely Defined Entropy Stable Matrix Dissipation Operator for High Mach Number Ideal MHD and Compressible Euler Simulations. *Journal of Computational Physics*, 332:274–289, 2017. doi: 10.1016/j.jcp.2016.12.006.

[144] Y. Xia, X. Liu, H. Luo, and R. Nourgaliev. A third-order implicit discontinuous Galerkin method based on a Hermite WENO reconstruction for time-accurate solution of the compressible Navier–Stokes equations. *International Journal for Numerical Methods in Fluids*, 79(8):416–435, 2015. doi: 10.1002/fld.4057.

[145] Q. Zhang and C.-W. Shu. Error Estimates to Smooth Solutions of Runge–Kutta Discontinuous Galerkin Method for Symmetrizable Systems of Conservation Laws. *SIAM Journal on Numerical Analysis*, 44(4):1703–1720, 2006. doi: 10.1137/040620382.

Scientific Publications

Paper 1

Received: 21 August 2018 | Accepted: 24 August 2018

DOI: 10.1002/pamm.201800203

Finite volume based multigrid preconditioners for discontinuous Galerkin methods

Lea Miko Versbach[1,*], **Philipp Birken**[1,**], and **Gregor J. Gassner**[2,***]

[1] Centre for the mathematical sciences, Numerical Analysis, Lund University, Lund, Sweden
[2] Mathematisches Institut, Universität zu Köln, Weyertal 86-90, 50931 Köln, Germany

Our aim is to construct efficient preconditioners for high order discontinuous Galerkin (DG) methods. We consider the DG spectral element method with Gauss-Lobatto-Legendre nodes (DGSEM-GL) for the 1D linear advection equation. It has been shown in [4] that DGSEM-GL has the summation-by-parts (SBP) property and an equivalent finite volume (FV) discretization is presented in [3]. Thus we present a multigrid (MG) preconditioner based on a simplified FV discretization.

1 Introduction

An efficient implicit DG variant is DGSEM, where the interpolation of the flux is collocated with the numerical quadrature used for the inner products, [4]. Key to an efficient algorithm is a fast solver with low memory footprint. Our aim is to constuct a matrix-free preconditioner using approximations to the FV discretization. We solve the 1D advection equation with DGSEM using a right preconditioner based on an agglomeration multigrid method. We show first results for 2-, 3- and 4-stage Runge-Kutta (RK) smoothers with optimized parameters from [1].

2 Discontinuous Galerkin Spectral Element Method

To introduce the DGSEM method, we consider the one-dimensional linear scalar advection equation

$$u_t + au_x = 0 \tag{1}$$

with $a > 0$, periodic boundary and suitable initial conditions. We construct a grid with M elements and introduce a nodal polynomial approximation of $N+1$ degrees of freedom $\{u_j\}_{j=0}^N$ located at the element grid nodes $x_0^n, \dots, x_N^n$

$$u_n(x,t) \approx u_h(x,t) = \sum_{j=0}^{N} u_j(t)\ell_j(x), \tag{2}$$

in each element. We choose GL grid nodes and Lagrange basis functions $\ell_j(x)$ of degree N. Multiplying with a test function $\ell \in \{\ell_j\}_{i=0}^N$ and integrating over the element transformed to $[-1,1]$ gives the weak form of (1). Inserting the numerical approximation (2) and integration by parts yields

$$\int_{-1}^{1} \dot{u}_h \ell_j dx + [u^* \ell_j]|_{-1}^{1} - \int_{-1}^{1} u_h \ell_j' dx = 0,\ j = 0, \dots, N \tag{3}$$

with $\dot{u}_h$ the time derivative of (2), $(.)'$ the spatial derivative w.r.t x and u^* the numerical upwind flux between elements. Integration by numerical quadrature with GL nodes and Lagrange basis yields a matrix vector formulation on each element:

$$\frac{\Delta x_n}{2} M\dot{u} - D^T M u = Bu^*, \tag{4}$$

$D_{ki} = \ell_i(x_k), i, k = 0, \dots, N$ and $B = diag([-1, 0, \dots, 0, 1])$.

3 Finite Volume Based Agglomeration Multigrid Preconditioner

We construct a preconditioner for (2) based on an MG method applied to a simple FV discretization of (1). The grid points of the FV cells coincide with the DGSEM elements plus additional N equidistant points inside of each element. An equidistant FV discretization for (1) with mesh width $\Delta\tilde{x}$ reads

$$\dot{u}_i + \frac{a}{\Delta\tilde{x}}(u_i - u_{i-1}) = 0, \tag{5}$$

* Corresponding author: e-mail lea_miko.versbach@math.lu.se, phone +46 46 222 6811
** e-mail philipp.birken@na.lu.se
*** e-mail ggassner@math.uni-koeln.de

for the cell average u_i on cell $i = 1, \ldots, K$. Implicit Euler time stepping yields a linear system to be solved in each time step:

$$u^{n+1} - u^n + \frac{a\Delta t}{\Delta \tilde{x}} C u^{n+1} = 0 \Leftrightarrow A u^{n+1} = u^n, \tag{6}$$

for $A = I + \frac{\nu}{\Delta \tilde{x}} C \in \mathcal{R}^{K \times K}$ and $\nu = a\Delta t$. The two-grid preconditioner for a smoother $x^{k+1} = M_S x^k + N_S^{-1} b$ reads

$$(I - P_1^0 A_0^{-1} R_0^1 A_1) N_S^{-1} + P_1^0 A_0^{-1} R_0^1 A_1. \tag{7}$$

We consider s-stage RK schemes with initial condition $u_0 = u^n$ as smoothers:

$$u_j = u^n + \alpha_j \Delta t^* (u^n - A u_{j-1}),\ j = 1, \ldots, s-1,$$
$$u_{n+1} = u^n + \Delta t^* (u^n - A u_{s-1}).$$

On each grid level l, the explicit pseudo time step is defined by $\Delta t_l^* = c\Delta x_l / \nu$ for mesh width Δx_l. We use optimized s-stage RK parameters from [1].

4 Numerical Results

We test the preconditioner on (1) with $a = 2$ on the interval $[0, 2]$ with periodic boundary conditions and initial condition $\sin(\pi x)$. We keep the number of unknowns constant 300 and perform one time step with $CFL = 1$. In figure 1 we see

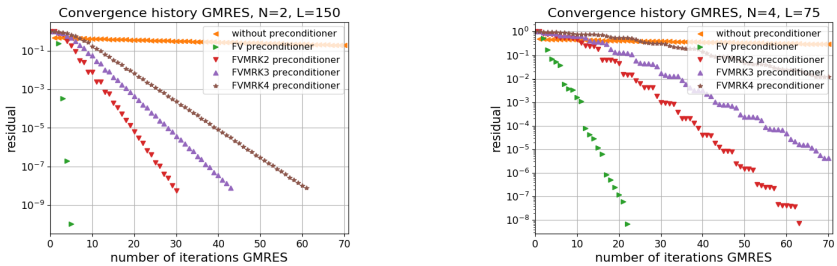


Fig. 1: Performance of preconditioned DGSEM for different optimized smoothers

the effect of the preconditioner. As expected, using the FV method itself results in the best preconditioner. FV MG based preconditioners perform also well, with a visible influence of the smoother. In this setting, the optimized RK2 smoother gives the best results. While unpreconditioned GMRES does not terminate in 70 iterations, using the FV based MG preconditioner with RK2 smoother GMRES terminates in less that 70 iterations. It is moreover noticeable that the preconditioner performs better for smaller N, namely in the left figure. This might be caused by the optimized smoothing parameters, which are constructed for the case $N = 1$.

5 Conclusions and Outlook

We suggested an FV based MG preconditioner for DGSEM to solve the advection equation in 1D. Using explicit 2-, 3- and 4-stage RK smoothers we have achieved an improved performance compared to the unpreconditioned case. The relatively large influence of the smoother choice suggests to consider more advanced smoothers, such as W-smoothers [2], and to find new optimal parameters for higher order DGSEM discretizations.

References

[1] P. Birken, Optimizing Runge-Kutta smoothers for unsteady flow problems, Electronic Transactions on Numerical Analysis **39**, pp. 298-312 (2012).
[2] P. Birken, J. Bull, A. Jameson, Preconditioned smoothers for the full approximation scheme for the RANS equations, J. Sci. Comput. preprint (2016).
[3] T. C. Fisher, M. H. Carpenter, J, Nordström, N. K. Yamaleev, C. Swanson, Discretely conservative finite-difference formulations for nonlinear conservation laws in split form: Theory and boundary conditions, Journal of Computational Physics **234**, pp. 353-375 (2013).
[4] G. J. Gassner, A skew-symmetric discontinuous Galerking spectral element discretization and its relation to SBP-SAT finite difference methods, J. Sci. Comput. **35**, pp. A1233-A1253 (2012).
[5] D. A. Kopriva, Implementing Spectral Element Methods for Partial Differential Equations, Scientific Computing, Springer, Dordrecht (2009).

Paper II

INTERNATIONAL JOURNAL OF COMPUTATIONAL FLUID DYNAMICS
2019, VOL. 33, NO. 9, 353–361
https://doi.org/10.1080/10618562.2019.1667983

Subcell finite volume multigrid preconditioning for high-order discontinuous Galerkin methods

Philipp Birken[a], Gregor J. Gassner[b] and Lea M. Versbach[a]

[a]Numerical Analysis, Centre for the Mathematical Sciences, Lund University, Lund, Sweden; [b]Department for Mathematics and Computer Science, Center for Data and Simulation Science, University of Cologne, Köln, Germany

ABSTRACT
We suggest a new multigrid preconditioning strategy for use in Jacobian-free Newton–Krylov (JFNK) methods for the solution of algebraic equation systems arising from implicit Discontinuous Galerkin (DG) discretisations. To define the new preconditioner, use is made of an auxiliary first-order finite volume discretisation that refines the original DG mesh, but can still be implemented algebraically. As smoother, we consider the pseudo-time iteration W3 with a symmetric Gauss–Seidel-type approximation of the Jacobian. As a proof of concept numerical tests are presented for the one-dimensional Euler equations, demonstrating the potential of the new approach.

ARTICLE HISTORY
Received 14 December 2018
Accepted 7 August 2019

KEYWORDS
Discontinuous Galerkin; multigrid preconditioner; W smoother; Euler equations

1. Introduction

The goal of our research is the construction of efficient Jacobian-free preconditioners for high order Discontinuous Galerkin (DG) discretisations with implicit time integration. One of our main interests is three-dimensional unsteady compressible flow. High-order DG methods (and related methods such as Flux Reconstruction (FR) discretisations) offer great potential for Large Eddy Simulation (LES) of turbulent flows with geometries, such as jet engines. The idea of DG (or FR) is to approximate the solution element-wise using a polynomial, which is allowed to be discontinuous across element interfaces, see Kopriva (2009) and Huynh (2007). Communication and coupling of degrees of freedom (DOF) is only across faces, whereas the element-local computations are very dense. As a result, DG methods are very well suited for domain-decomposition-based parallelisation (see, e.g. Hindenlang et al. 2012; Vincent et al. 2016). The specific variant we consider is the *DG Spectral Element Method* (DG-SEM), e.g. Kopriva, Woodruff, and Hussaini (2002). We use a Lagrange-type (nodal) basis with Gauss–Lobatto (GL) quadrature nodes with the collocation of the discrete integration. These choices yield DG operators that satisfy the summation-by-parts (SBP) property (see Gassner 2013), which is the discrete analogue to integration-by-parts. SBP is key to construct methods that are discretely entropy stable and/or kinetic energy preserving.

DG discretisation in space results in a big system of ODEs. Due to geometry features and thin boundary layers that occur in challenging compressible turbulent flow applications as the design of jet engines, aeroplanes and wind turbines, the resulting large system of ODEs is stiff. Implicit time integrators can overcome the deficiency of explicit time integrators with restrictive CFL conditions. However, efficiency can only be restored when the arising large non-linear systems are solved efficiently in terms of CPU time, but also regarding memory consumption. Vincent and Jameson mention that solvers for linear and non-linear equation systems are severely lacking for 3D DG applications as one of four major obstacles that need to be solved if high-order methods are to be widely adopted by, e.g. industry, Vincent and Jameson (2011). Candidates for solvers are *FAS-Multigrid* (full-approximation multigrid scheme) and *preconditioned Jacobian-Free Newton–Krylov methods* (JFNK) (Knoll and Keyes 2004) where multigrid can be used as a preconditioner (see Birken and Jameson 2010). The JFNK technology is in general interesting, as the memory use is minimised. Although DG systems have a weakly coupled block structure, the blocks themselves can be large. In particular, the problem for high-order

CONTACT Lea M. Versbach lea@maths.lth.se, lea_miko.versbach@math.lu.se

DG methods is that the number of unknowns per element increases dramatically with increasing polynomial degree and dimension, leading to large dense Jacobian blocks (see Birken et al. 2013; Birken 2012). For a finite volume method, the block size is $2+d$ with dimension d, whereas for a DG-SEM with pth degree polynomials, it is $(d+2)(p+1)^d$. For degree 2 in 3D, this is already 135. The favourable memory consumption of the JFNK approach is obsolete if the preconditioner requires the storage of (parts of) the DG system Jacobian.

Hence in this article, we present a novel idea for the construction of a well-performing preconditioner for the JFNK approach, while retaining the low memory use, i.e. a Jacobian-free preconditioner. The main ingredient consists in the construction of a simplified replacement operator. A motivation for this is a previously proved equivalence between a DG-SEM discretisation and a high order FV discretisation, see Fisher et al. (2013). One could, for instance, choose a different polynomial order in the element to generate a replacement operator as in Fidkowski et al. (2005) and Birken et al. (2013). However, we aim to retain the number of DOFs in our replacement operator by introducing subcells in each element, namely $p+1$ in each spatial direction. On this subcell-element grid, the simplest replacement operator is a first-order finite volume (FV) discretisation. In some sense, we reinterpret the nodal values as input for an FV method. This gives a semi-structured–unstructured approximation, where the elements are unstructured, but inside the element the subcells are structured (Versbach, Birken, and Gassner 2018). We extend the idea of this paper to the Euler equations. In the resulting approximate Jacobian, we now only have $(d+2)(p+1)(2d+1)$ entries (Birken 2012). Furthermore, it allows to use the available knowledge about fast multigrid (MG) methods for FV discretisations on (block-)structured meshes. As a smoother for our FV discretisation, we use a state of the art low memory W3 smoother from Birken, Bull, and Jameson (2018).

A related approach was proposed in Allaneau, Li, and Jameson (2012) for spectral difference (SD) methods, where the replacement operator is also an FV discretisation on a potentially fine grid. However, there the FV grid is not embedded in the high-order grid, but overlayed. Thus, it is necessary to interpolate the solution (and the residual) in-between different grids (with different topologies), which needs interpolation and reconstruction operators similar to Chimera techniques. In contrast, we want to harness in particular the semi-unstructured–structured mesh-topology: our FV discretisation basically lives on the same DOFs as the nodal high-order DG method.

In the remainder of the paper, we first describe our prototype problem, the one-dimensional compressible Euler equations. We then present the DG-SEM and the FV subcell discretisations as well as the multigrid solver. In the last part of the paper, we show numerical experiments to validate the approach and draw our conclusions.

2. One-dimensional Euler equations

As a prototype problem for our novel idea, we consider the one-dimensional compressible Euler equations for a perfect gas

$$\begin{pmatrix} \rho \\ m \\ \rho E \end{pmatrix}_t + \begin{pmatrix} m \\ mv + p \\ Hm \end{pmatrix}_x = 0, \tag{1}$$

with appropriate initial and boundary conditions. Here ρ is the density, $m = \rho v$ the momentum, p the pressure, E the energy and $H = E + p/\rho$ the enthalpy. Defining $U = (\rho, m, \rho E)^{\mathrm{T}}$ and $f(U) = (m, mv + p, Hm)^{\mathrm{T}}$, we can write the Euler equations in vector form:

$$U_t + f(U)_x = 0. \tag{2}$$

3. Spatial discretisation

3.1. DG-SEM

For the spatial discretisation, we introduce a grid with K elements e^k of width Δx_k, $k = 1, \ldots, K$. Each element is transformed to the reference element $[-1, 1]$ by a linear mapping with Jacobian $J_k := \Delta x_k/2$. The solution is approximated by a nodal polynomial in reference space with degree p in each element e^k

$$U(\xi, t)|_{e^k} \approx U^k(\xi, t) = \sum_{j=1}^{p+1} U_j^k(t)\psi_j(\xi), \tag{3}$$

where the interpolation nodes are the GL nodes $\{\xi_j\}_{j=1}^{p+1}$. We use the element mapping to transform the Euler equations into reference space

$$J_k U_t + f(U)_\xi = 0, \tag{4}$$

and insert our ansatz (3). Next, we integrate over the reference element, use integration-by-parts for the flux term and replace the fluxes at the element interfaces with so-called numerical flux functions f^* to arrive at

$$\int_{-1}^{1} J_k U_t^k \psi(\xi)\,\mathrm{d}\xi + [f^*\psi(\xi)]_{-1}^{1} - \int_{-1}^{1} f(U^k)\psi_\xi(\xi)\,\mathrm{d}\xi = 0. \tag{5}$$

As a numerical flux function, we choose the Rusanov flux (or local Lax–Friedrich flux)

$$f^*(U^-, U^+) = \frac{f(U^-) + f(U^+)}{2} - \frac{\lambda_{\max}}{2}\left(U^+ - U^-\right), \tag{6}$$

where U^-, U^+ are the values left and right at an element interface and $\lambda_{\max}$ is an estimate of the maximum wave speed at the interface.

As stated above, the main idea of the DG-SEM is collocation. We use collocation for our discrete integration, i.e. we replace the integrals in (5) with GL quadrature rules at the same location as our interpolation, which can be interpreted as a collocation of the non-linear fluxes $f(U)$. With this choice, the DG-SEM operators simplify a lot: we get the diagonal mass matrix $M_{ij} = \delta_{ij}\omega_i, i, j = 1, \ldots, p+1$, the diagonal boundary matrix $\boldsymbol{B} = diag(-1, 0, \ldots, 0, 1)$ and the derivative matrix $D_{ij} := (\psi_j)_\xi(\xi_i)$, $i, j = 1, \ldots, p+1$. Replacing the integrals in (5) and using the definitions of the DG-SEM operators, we arrive at the DG-SEM method in the matrix–vector formulation for one cell e^k

$$\dot{\boldsymbol{U}}^k = -\frac{1}{J_k}(\boldsymbol{M}^{-1}\boldsymbol{B}\boldsymbol{f}^* - \boldsymbol{M}^{-1}\boldsymbol{D}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{f}). \tag{7}$$

The elemental residuals are coupled through the numerical flux f^* with DOFs from other elements connected via the interfaces. Collecting the equations for all elements in one big system with unknown $\underline{\boldsymbol{u}}$ and applying implicit Euler in time gives

$$\underline{\boldsymbol{u}}^{n+1} - \underline{\boldsymbol{u}}^n - \Delta t\underline{\boldsymbol{G}}(\underline{\boldsymbol{u}}^{n+1}) =: \underline{\boldsymbol{F}}(\underline{\boldsymbol{u}}^{n+1}) = \underline{\boldsymbol{0}}, \tag{8}$$

where $\underline{\boldsymbol{G}}(\underline{\boldsymbol{u}}^{n+1})$ collects the DG-SEM residuals, i.e. the right-hand side of (7).

3.2. Finite volume d iscretisation

Based on the DG-SEM discretisation, we define an FV discretisation on a subcell mesh with $p+1$ cells

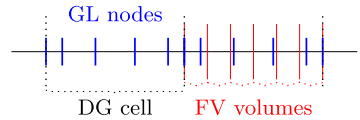


Figure 1. Two DG cells with 6 GL nodes each. An equidistant FV mesh is shown in the right cell.

(see Figure 1). The discretisation on this semi-structured–unstructured grid is used to define the preconditioner and to construct the multigrid method. An FV method is based on approximating the cell averages in a subcell i in an element e^k

$$\frac{1}{\Delta x_i}\int_{x_{i-1/2}}^{x_{i+1/2}} U(x, t^n)\,\mathrm{d}x \approx U_i^{k,n} \tag{9}$$

at each time level t^n. For a subcell in the element e^k with subcell size Δx_i, the subcell FV discretisation reads

$$\dot{U}_i^k + \frac{1}{\Delta x_i}(f_{i+1/2}^* - f_{i-1/2}^*) = 0, \quad i = 1, \ldots, p+1;\ k = 1, \ldots, K. \tag{10}$$

The numerical flux function $f_{i+1/2}^*$ is again Rusanov flux (6), with the values left and right being not the polynomial values, but the subcell average values instead.

4. Preconditioned Jacobian-free Newton-GMRES

The resulting system of nonlinear equations (8) is solved using Newton's method, written for the equation $\underline{\boldsymbol{F}}(\underline{\boldsymbol{u}}) = \underline{\boldsymbol{0}}$:

$$\begin{aligned} &\text{solve}\ \frac{\partial \underline{\boldsymbol{F}}(\underline{\boldsymbol{u}})}{\partial \underline{\boldsymbol{u}}}\Big|_{\underline{\boldsymbol{u}}^{(k)}} \underline{\boldsymbol{s}} = -\underline{\boldsymbol{F}}(\underline{\boldsymbol{u}}^{(k)}), \\ &\underline{\boldsymbol{u}}^{(k+1)} = \underline{\boldsymbol{u}}^{(k)} + \underline{\boldsymbol{s}}, \quad k = 0, 1, \ldots \end{aligned} \tag{11}$$

for a given initial guess $\underline{\boldsymbol{u}}^{(0)}$. The linear system in (11) is solved using right preconditioned GMRES with a relative tolerance.

In order to not compute the Jacobian in each iteration (11), we replace the matrix–vector products appearing in GMRES by a difference quotient:

$$\frac{\partial \underline{\boldsymbol{F}}(\underline{\boldsymbol{u}})}{\partial \underline{\boldsymbol{u}}}\underline{\boldsymbol{q}} \approx \frac{\underline{\boldsymbol{F}}(\underline{\boldsymbol{u}} + \epsilon\underline{\boldsymbol{q}}) - \underline{\boldsymbol{F}}(\underline{\boldsymbol{u}})}{\epsilon}, \quad \text{with}\ \epsilon = \frac{1\mathrm{e}{-7}}{\|\underline{\boldsymbol{q}}\|}. \tag{12}$$

5. Finite volume multigrid preconditioner

The basis for the preconditioner is an agglomeration multigrid method on the finite volume grid. The coarse grid problems are given by applying the FV discretisation on the coarse grid. To restrict fine grid values, they are summed, weighted by the volumes of the respective cells and divided by the total volume. For an equidistant grid in one dimension, the corresponding restriction operator is given by

$$\underline{\boldsymbol{R}}_{l-1}^{l} = \frac{1}{2}\begin{pmatrix} 1 & 1 & & & & \\ & & 1 & 1 & & \\ & & & \ddots & \ddots & \\ & & & & 1 & 1 \end{pmatrix}. \tag{13}$$

As prolongation we use the injection, where the value in the coarse cell is taken for all corresponding fine cells:

$$\underline{\boldsymbol{P}}_{l}^{l-1} = 2\underline{\boldsymbol{R}}_{l-1}^{l^{\mathrm{T}}}. \tag{14}$$

A generic smoother is an iterative method for the solution of $\underline{\boldsymbol{A}}_l\underline{\boldsymbol{s}}_l = \underline{\boldsymbol{b}}_l$ and is given by $\underline{\boldsymbol{s}}_l^{k+1} = \underline{\boldsymbol{M}}_{S,l}\underline{\boldsymbol{s}}_l^{k} + \underline{\boldsymbol{N}}_{S,l}^{-1}\underline{\boldsymbol{b}}_l$. The index l specifies the multigrid level, while S denotes that this iterative method represents the smoother. It is possible to construct an MG preconditioner based on a V- or a W-cycle, as well as several consecutive cycles. The number of pre- and postsmoothing steps is also flexible. We now write down a V-cycle multigrid algorithm with one step of pre- and postsmoothing, which corresponds to $\gamma = 1$ in the following pseudo-code:

`function MG(`$\underline{\boldsymbol{s}}_l, \underline{\boldsymbol{b}}_l, l$`) :`

- $\underline{\boldsymbol{s}}_l = \underline{\boldsymbol{M}}_{S,l}\underline{\boldsymbol{s}}_l + \underline{\boldsymbol{N}}_{S,l}^{-1}\underline{\boldsymbol{b}}_l$ (presmoothing)

if($l > 0$)

- $\underline{\boldsymbol{r}}_{l-1} = \underline{\boldsymbol{R}}_{l-1}^{l}(\underline{\boldsymbol{A}}_l\underline{\boldsymbol{s}}_l - \underline{\boldsymbol{b}}_l)$ (restriction)
- $\underline{\boldsymbol{v}}_{l-1} = 0$
  - for $j = 1, \ldots, \gamma$: $\underline{\boldsymbol{v}}_{l-1} = MG(\underline{\boldsymbol{v}}_{l-1}, \underline{\boldsymbol{r}}_{l-1}, l-1)$
- $\underline{\boldsymbol{s}}_l = \underline{\boldsymbol{s}}_l - \underline{\boldsymbol{P}}_{l}^{l-1}\underline{\boldsymbol{v}}_{l-1}$ (fine grid correction)
- $\underline{\boldsymbol{s}}_l = \underline{\boldsymbol{M}}_{S,l}\underline{\boldsymbol{s}}_l + \underline{\boldsymbol{N}}_{S,l}^{-1}\underline{\boldsymbol{b}}_l$ (postsmoothing)

This gives rise to an iterative method of the form $\underline{\boldsymbol{s}}_l^{k+1} = \underline{\boldsymbol{M}}_{MG}\underline{\boldsymbol{s}}_l^{k} + \underline{\boldsymbol{N}}_{MG}^{-1}\underline{\boldsymbol{b}}_l$. In the case of an $l_{\max}$-level multigrid cycle with presmoothing on the coarsest level, the preconditioner $\underline{\boldsymbol{N}}_{\ell,MG}^{-1} \approx \underline{\boldsymbol{A}}^{-1}$ is defined recursively by

$$\begin{aligned} \underline{\boldsymbol{N}}_{0,MG}^{-1} &= \underline{\boldsymbol{N}}_{S,0}^{-1}, \quad \text{and for } l = 1, \ldots, l_{\max}: \\ \underline{\boldsymbol{N}}_{l,MG}^{-1} &= \underline{\boldsymbol{M}}_{S,l}(\underline{\boldsymbol{N}}_{S,l}^{-1} - \underline{\boldsymbol{P}}_{l}^{l-1}\underline{\boldsymbol{N}}_{l-1,MG}^{-1}\underline{\boldsymbol{R}}_{l-1}^{l}\underline{\boldsymbol{A}}_l\underline{\boldsymbol{N}}_{S,l}^{-1} \\ &\quad + \underline{\boldsymbol{P}}_{l}^{l-1}\underline{\boldsymbol{N}}_{l-1,MG}^{-1}\underline{\boldsymbol{R}}_{l-1}^{l}) + \underline{\boldsymbol{N}}_{S,l}^{-1}. \end{aligned} \tag{15}$$

5.1. Smoothers: pseudo-time iterations

As smoothers for the multigrid preconditioner, we consider W schemes (see Birken, Bull, and Jameson 2018). A pseudo-time derivative is added to Equation (11) to yield

$$\frac{\partial \underline{\boldsymbol{s}}}{\partial t^*} + \underline{\boldsymbol{A}}\underline{\boldsymbol{s}} - \underline{\boldsymbol{b}} = \underline{\boldsymbol{0}}. \tag{16}$$

A W smoother is given by

$$\begin{aligned} \underline{\boldsymbol{s}}_0 &= \underline{\boldsymbol{s}}_n, \\ \underline{\boldsymbol{s}}_j &= \underline{\boldsymbol{s}}_n - \alpha_j \Delta t^* \underline{\boldsymbol{W}}^{-1}(\underline{\boldsymbol{A}}\underline{\boldsymbol{s}}_{j-1} - \underline{\boldsymbol{b}}), \quad j = 1, \ldots, s, \\ \underline{\boldsymbol{s}}_{n+1} &= \underline{\boldsymbol{s}}_s, \end{aligned} \tag{17}$$

where $\underline{\boldsymbol{W}} \approx \underline{\boldsymbol{I}} + \eta\Delta t^*\underline{\boldsymbol{A}}$. The free parameters are η, usually taken from the range $[0.25, 1.5]$, as well as $\alpha_j \in [0, 1]$ and a local Δt^*, given by a pseudo-CFL number c^* depending on the maximal eigenvalue of the Jacobian $\lambda_{\max}$:

$$\Delta t^* = c^* \frac{\Delta x_i}{|\lambda_{\max}|}. \tag{18}$$

Smoothers of form (17) can be written as a linear iterative scheme

$$\underline{\boldsymbol{s}}_{n+1} = \underline{\boldsymbol{M}}_S\underline{\boldsymbol{s}}_n + \underline{\boldsymbol{N}}_S^{-1}\underline{\boldsymbol{b}}, \tag{19}$$

where for a 3-stage W smoother, we obtain

$$\begin{aligned} \underline{\boldsymbol{N}}_S^{-1} &= \alpha_3\Delta t^*\underline{\boldsymbol{W}}^{-1} - \alpha_3\alpha_2\Delta t^{*2}\underline{\boldsymbol{W}}^{-1}\underline{\boldsymbol{A}}\,\underline{\boldsymbol{W}}^{-1} \\ &\quad + \alpha_3\alpha_2\alpha_1\Delta t^{*3}(\underline{\boldsymbol{W}}^{-1}\underline{\boldsymbol{A}})^2\underline{\boldsymbol{W}}^{-1}. \end{aligned} \tag{20}$$

The approximation of $\underline{\boldsymbol{W}}^{-1}$ will be explained in the next section. The coefficients are given by $[\alpha_1, \alpha_2, \alpha_3] = [0.1481, 2/5, 1]$.

5.1.1. SGS preconditioner

The specific approximation to define $\underline{\boldsymbol{W}}$ is based on a symmetric Gauss–Seidel (SGS) approach. Such a method was originally suggested in Swanson, Turkel, and Rossow (2007) and further developed by several

authors. We follow a recent version from Birken, Bull, and Jameson (2018).

The first step is to approximate the Jacobian by using a different first-order discretisation of the linearised Euler equation. It is based on a splitting $\mathbf{A} = \mathbf{A}^+ + \mathbf{A}^-$ of the flux Jacobian. This is evaluated in the average of the values on both sides of the interface. The split Jacobians correspond to positive and negative eigenvalues and can be written in terms of the matrix of right eigenvectors $\mathbf{Q}$ as

$$\mathbf{A}^+ = \mathbf{Q}\Lambda^+\mathbf{Q}^{-1}, \quad \mathbf{A}^- = \mathbf{Q}\Lambda^-\mathbf{Q}^{-1},$$

where $\Lambda^\pm$ are diagonal matrices containing the positive and negative eigenvalues, respectively. These are then bounded away from zero using a parabolic function which takes care when the modulus of the eigenvalue λ is smaller or equal to a fraction ad of the speed of sound a with free parameter $d \in [0, 1]$:

$$|\lambda| = \frac{1}{2}\left(ad + \frac{|\lambda|^2}{ad}\right), \quad |\lambda| \leq ad. \tag{21}$$

With this, an upwind discretisation of the split Jacobian is given in cell i by

$$\begin{aligned}\mathbf{u}_{i_{t^*}} + \mathbf{u}_i + \frac{\Delta t}{\Delta x_i}((\mathbf{A}_{ii}^+\mathbf{u}_i + \mathbf{A}_{i,i+1}^-\mathbf{u}_{i+1}) \\ - (\mathbf{A}_{i-1,i}^+\mathbf{u}_{i-1} + \mathbf{A}_{ii}^-\mathbf{u}_i)) = 0.\end{aligned} \tag{22}$$

The corresponding approximation of the Jacobian is then used to construct a preconditioner. Specifically, we consider the block SGS preconditioner

$$\underline{\boldsymbol{W}}^{-1} = (\underline{\boldsymbol{D}} + \underline{\boldsymbol{L}})^{-1}\underline{\boldsymbol{D}}(\underline{\boldsymbol{D}} + \underline{\boldsymbol{U}})^{-1}, \tag{23}$$

where $\underline{\boldsymbol{L}}$, $\underline{\boldsymbol{D}}$ and $\underline{\boldsymbol{U}}$ are block matrices with 3×3 blocks.

We have $\underline{\boldsymbol{L}} + \underline{\boldsymbol{D}} + \underline{\boldsymbol{U}} = \mathbf{I} + \eta\Delta t^*\mathbf{J}$ and obtain

$$\mathbf{L}_{i-1,i} = -\frac{\eta\Delta t\Delta t_i^*}{\Delta x_i}\mathbf{A}_{i-1,i}^+, \quad \mathbf{U}_{i,i+1} = \frac{\eta\Delta t\Delta t_i^*}{\Delta x_i}\mathbf{A}_{i,i+1}^-, \tag{24}$$

$$\underline{\boldsymbol{D}}_{ii} = \mathbf{I} + \eta\Delta t^*\mathbf{I} + \frac{\eta\Delta t\Delta t_i^*}{\Delta x_i}(\mathbf{A}_{ii}^+ - \mathbf{A}_{ii}^-).$$

Applying this preconditioner requires solving 3×3 systems coming from the diagonal, which can be done directly. A fast implementation is obtained by transforming first to a certain set of symmetrising variables (see Swanson, Turkel, and Rossow 2007).

6. Numerical results

We consider the one-dimensional Euler equations on the interval [0, 10] and study one step of implicit Euler where we look at the convergence rate of GMRES with maximal 100 iterations in the first Newton step. All results are produced in Python. Measuring the CPU time will not give a great insight into performance in the one-dimensional case and is therefore not discussed in the following, but is important to consider in higher dimensions.

We equip the Euler equations with periodic boundary conditions and consider two different initial conditions: A subsonic case $(\rho_0, v_0, p_0) = (1 + 0.1\sin(2\pi x/10), 1, 28)$ with Mach number 0.16 and a transonic case $(\rho_0, v_0, p_0) = (1 + 0.1\sin(2\pi x/10), 1, 1)$ with Mach number 0.85. We choose a CFL number of 100 and test the discussed MG preconditioner with W3 smoother for a V cycle, see (15). A simple block Jacobi preconditioner with blocks corresponding to the 3×3 systems does not improve the convergence rate of the GMRES cycle compared to no preconditioner. This motivates to consider more sophisticated peconditioners for the given problem. We also note that applying the W3 presmoothing step several times gives almost no convergence improvement while being very expensive in terms of computational costs. The same holds for using a W cycle or two consecutive V cycles, as well as for the method of nonsymmetric Restriction Aggregation (NSR) from Sala and Tuminaro (2008).

We test the framework for 4th- and 8th-order DG methods with 240 and 480 DOFs, respectively. In order to have a reference for efficiency, we construct a preconditioner based on the Jacobian of FV discretisation (12). The new multigrid preconditioner approximates the inverse of the FV Jacobian and thus cannot be expected to behave superior to the reference preconditioner. The reference preconditioner is applied by using GMRES with tolerance 1e-14 and maximal 300 iterations. This is very expensive in terms of computations and only suggested to compare how well the proposed MG preconditioner approximates the inverse of the FV Jacobian.

6.1. Subsonic case

In the subsonic case, the pseudo-CFL number for the W3 smoother is $c^* = 10$, $\eta = 1.4$ and $d = 0.1$. We consider two different MG preconditioners: one with only one presmoothing step on each level and one with one

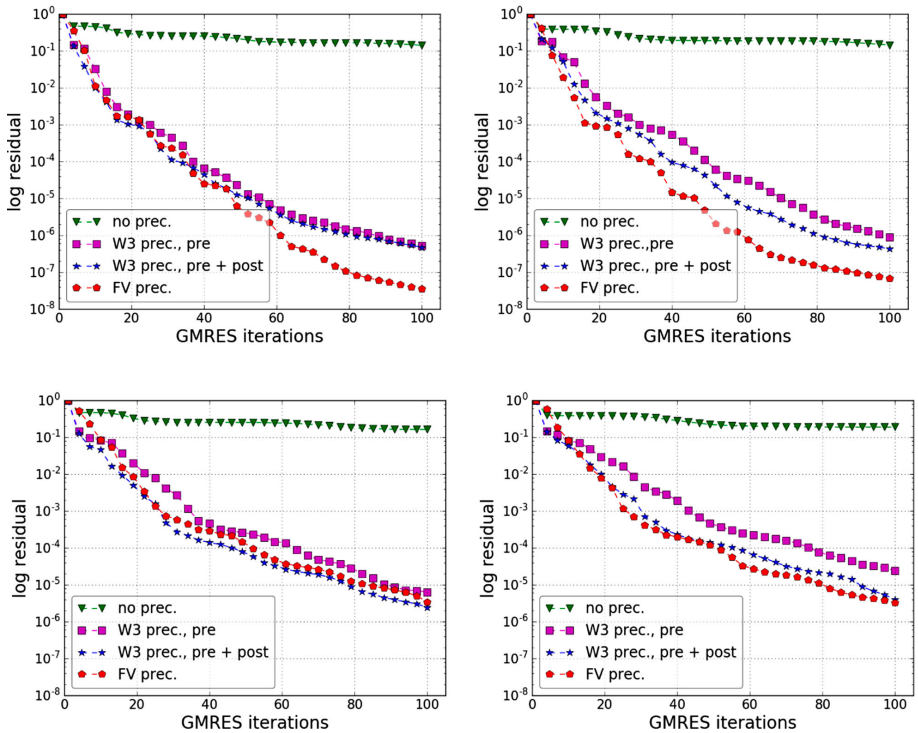


Figure 2. Convergence history for GMRES for one Newton iteration, subsonic case, DG order 4 (top) and 8 (bottom), 240 DOFs (left) and 480 DOFs (right).

pre- and one postsmoothing step on each level. The convergence results are shown in Figure 2.

We see that the reference preconditioner (FV) gives very good results for all four test cases, but works better for 4th order. In particular, there is fast initial convergence, which is crucial to get fast termination within an inexact Newton's method. The suggested MG preconditioners yield a very good approximation to the reference preconditioner, especially within the first 20 GMRES iterations. The MG preconditioner with pre- and postsmoothing gives the best results, outperforming the MG preconditioner equipped with only presmoothing slightly. This holds for DG solvers of 4th and 8th order as well as for DOFs 240 and 480. Increasing the DOFs does not have a visible impact on the convergence rate of the reference preconditioner for both 4th and 8th order while we can notice small differences in the behaviour of the MG preconditioners. The residual after 100 GMRES iterations differs slightly more for the two MG preconditioners when increasing the DOFs for both 4th and 8th order. Since the FV replacement operator is of the first order, the question arises how it behaves for the increased order of the DG method. When going from 4th to 8th order for 240 DOFs, there are two orders of magnitude decrease of the reference preconditioner, whereas the decrease for the W3 preconditioners is very small and they behave very similarly. For 480 DOFs, there is a 1.5 order of magnitude decrease of the reference preconditioner and the MG preconditioners behave similarly. It is noticeable that the MG preconditioners mimic the FV preconditioner better when increasing the order of the DG discretisation. We expect a performance improvement of the smoother for optimised c^*, η and d.

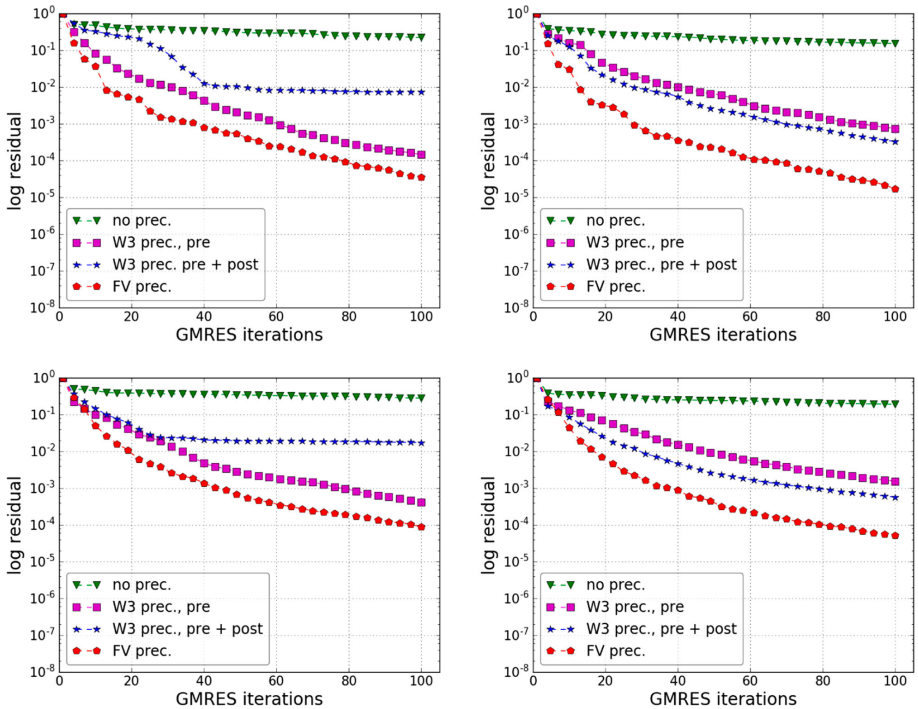


Figure 3. Convergence history for GMRES for one Newton iteration, transonic case, DG order 4 (top) and 8 (bottom), 240 DOFs (left) and 480 DOFs (right).

6.2. Transonic case

In the transonic case, the pseudo-CFL number for the W3 smoother is $c^* = 2$, $\eta = 0.7$ and $d = 0.1$. We consider the same two different MG preconditioners as in the transonic case. The convergence results are shown in Figure 3.

Both reference FV and MG preconditioners perform worse than in the subsonic case. Such a loss of performance has been reported for a p-multigrid method in Premasuthan et al. (2009). In the transonic case, the reference preconditioner works very similar for 4th and 8th order and different DOFs. We notice only a slight decrease in performance when increasing the order as well as when increasing the DOF. For 240 DOFs, the first MG preconditioner mimics the performance of the reference preconditioner with approximately 0.5 order of magnitude degradation and works slightly better for the 4th order. The second MG preconditioner does not work well for 240 DOFs and the 4th and 8th order, respectively. When increasing the DOFs, the MG preconditioner with pre- and postsmoothing slightly outperforms the one with presmoothing only. In the case of 480 DOFs, the second MG preconditioner works around 1 order of magnitude worse than the reference preconditioner, giving for both 4th and 8th order methods a result approximately 0.5 order of magnitude degradation compared to 240 DOFs. Again the order affects the performance only slightly for different DOFs.

7. Conclusions

We presented a new multigrid preconditioning strategy for use in JFNK methods for the solution of equation systems arising from DG discretisations. The core idea is to make use of an auxiliary first-order FV

discretisation that refines the original DG mesh, but can still be implemented algebraically. As smoother, we consider W3. Numerical results show the potential of the approach as a proof of concept for the one-dimensional Euler equations. A simple block Jacobi preconditioner does not improve the convergence rate compared to no preconditioner at all, which justifies the necessity of using this more sophisticated preconditioner. The convergence results of the proposed preconditioner are promising, being close to a quasi-exact preconditioner in the subsonic case. Our results indicate that the performance of the preconditioner is only weakly influenced by the order of the DG discretisation. A possible extension of the preconditioner to multiple spatial dimensions could be based on a tensor product strategy, which is also the natural approach for the extension of DG-SEM to multiple spatial dimensions. It should be noted that W3 smoothers are designed for high aspect ratio grids and have been shown to achieve even better performance on those compared to equidistant meshes (Birken, Bull, and Jameson 2018). This is of special interest for Navier–Stokes equations with a boundary layer.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

Philipp Birken and Lea Versbach gratefully acknowledge the support of the Swedish Research Council – Vetenskapsrådet [grant number 2015-04133]. Gregor Gassner has been supported by the European Research Council (ERC) under the EUs 8th Framework Program Horizon 2020 with the research project Extreme [ERC grant agreement no. 714487].

References

Allaneau, Yves, Lala Y. Li, and Antony Jameson. 2012. "Convergence Acceleration of High Order Numerical Simulations using a Hybrid Spectral Difference – Finite Volume Multigrid Method." In *7th International Conference on Computational Fluid Dynamics (ICCFD7)*, Big Island, Hawaii, July 9–13.

Birken, Philipp. 2012. "Numerical Methods for the Unsteady Compressible Navier-Stokes Equations." Habilitation Thesis, University of Kassel.

Birken, Philipp, Jonathan Bull, and Antony Jameson. 2018. "Preconditioned Smoothers for the Full Approximation Scheme for the RANS Equations." *Journal of Scientific Computing* 78 (2): 995–1022.

Birken, Philipp, Gregor Gassner, Mark Haas, and Claus-Dieter Munz. 2013. "Preconditioning for Modal Discontinuous Galerkin Methods for Unsteady 3D Navier-Stokes Equations." *Journal of Computational Physics* 240: 20–35.

Birken, Philipp, and Antony Jameson. 2010. "On Nonlinear Preconditioners in Newton-Krylov-Methods for Unsteady Flows." *International Journal of Numerical Methods in Fluids* 62: 565–573.

Fidkowski, Krzysztof J., Todd A. Oliver, James Lu, and David L. Darmofal. 2005. "p-Multigrid Solution of High-order Discontinuous Galerkin Discretizations of the Compressible Navier-Stokes Equations." *Journal of Computational Physics* 207: 92–113.

Fisher, Travis C., Mark H. Carpenter, Jan Nordström, Nail K. Yamaleev, and Charles Swanson. 2013. "Discretely Conservative Finite-difference Formulations for Nonlinear Conservation Laws in Split Form: Theory and Boundary Conditions." *Journal of Computational Physics* 234: 353–375.

Gassner, Gregor J. 2013. "A Skew-symmetric Discontinuous Galerking Spectral Element Discretization and Its Relation to SBP-SAT Finite Difference Methods." *Journal of Scientific Computing* 35: A1233–A1253.

Hindenlang, Florian, Gregor J. Gassner, Christoph Altmann, Andrea Beck, Mark Staudenmaier, and Claus-Dieter Munz. 2012. "Explicit Discontinuous Galerkin Methods for Unsteady Problems." *Computers Fluids* 61 (May): 86–93.

Huynh, H. T. 2007. "A Flux Reconstruction Approach to High-Order Schemes Including Discontinuous Galerkin Methods." In *AIAA*, 2007–4079. Miami, FL.

Knoll, Dana A., and David E. Keyes. 2004. "Jacobian-free Newton-Krylov Methods: A Survey of Approaches and Applications." *Journal of Computational Physics* 193: 357–397.

Kopriva, David A. 2009. *Implementing Spectral Element Methods for Partial Differential Equations*. Dordrecht: Springer Scientific Computing.

Kopriva, David A., S. L. Woodruff, and M. Y Hussaini. 2002. "Computation of Electromagnetic Scattering with a Non-conforming Discontinuous Spectral Element Method." *International Journal for Numerical Methods in Engineering* 53: 105–122.

Premasuthan, Sachin, Chunlei Liang, Antony Jameson, and Z. J. Wang. 2009. "A P-Multigrid Spectral Difference Method for Viscous Compressible Flow Using 2D Quadrilateral Meshes." AIAA Paper 2009-950. Orlando, FL.

Sala, Marzio, and Raymond S. Tuminaro. 2008. "A New Petrov-Galerkin Smoothed Aggregation Preconditioner for Nonsymmetric Linear Systems." *SIAM Journal on Scientific Computing* 31 (1): 143–166.

Swanson, R. Charles, Eli Turkel, and Cord-Christian Rossow. 2007. "Convergence Acceleration of Runge-Kutta Schemes for Solving the Navier-Stokes Equations." *Journal of Computational Physics* 224 (1): 365–388.

Versbach, Lea M., Philipp Birken, and Gregor J. Gassner. 2018. "Finite Volume Based Multigrid Preconditioners for Discontinuous Galerkin Methods." PAMM, Vol. 18. Munich.

Vincent, Peter E., and Antony Jameson. 2011. "Facilitating the Adoption of Unstructured High-Order Methods Amongst a Wider Community of Fluid Dynamicists." *Mathematical Modelling of Natural Phenomena* 6 (3): 97–140.

Vincent, Peter E., Freddie Witherden, Brian Vermeire, Jin S. Park, and Arvind Iyer. 2016. "Towards Green Aviation with Python at Petascale." SC16: International Conference for High Performance Computing, Networking, Storage and Analysis. Article No. 1. Salt Lake City, UT.

Paper III

14th World Congress on Computational Mechanics (WCCM)
ECCOMAS Congress 2020)
Virtual Congress: 11-15 January 2021
F. Chinesta, R. Abgrall, O. Allix and M. Kaliske (Eds)

AN FINITE VOLUME BASED MULTIGRID PRECONDITIONER FOR DG-SEM FOR CONVECTION-DIFFUSION

Johannes KASIMIR[*,1], **Lea M. VERSBACH**[*,2], **Philipp BIRKEN**[*,3], **Gregor J. GASSNER**[4] **AND Robert KLÖFKORN**[*,5]

[*] Lund University
Centre for Mathematical Sciences, Box 118, 221 00 Lund, Sweden

[1] johannes.kasimir.418@student.lu.se

[2] lea_miko.versbach@math.lu.se, http://www.maths.lu.se/staff/lea-miko-versbach/

[3] philipp.birken@math.lu.se, http://www.maths.lu.se/staff/philipp-birken/

[4] University of Cologne
Division of Mathematics/Center for Data and Simulation Science, Weyertal 86 90,50931 Köln, Germany, ggassner@math.uni-koeln.de, https://www.mi.uni-koeln.de/NumSim/gregor-gassner/

[5] robertk@math.lu.se, http://www.maths.lu.se/staff/robert-kloefkorn/

Key words: Discontinuous Galerkin, Finite Volume, Multigrid Preconditioners, Compressible Flow

Abstract. The goal of our research is the construction of efficient Jacobian-free preconditioners for high order Discontinuous Galerkin (DG) discretizations with implicit time integration. We are motivated by three-dimensional unsteady compressible flow applications, which often result in large stiff systems. Implicit time integrators overcome the impact upon restrictive CFL conditions on explicit ones but leave the problem to solve huge nonlinear systems. In this paper we consider a multigrid preconditioning strategy for Jacobian-free Newton-Krylov (JFNK) methods for the solution of algebraic equation systems arising from implicit Discontinuous Galerkin (DG) discretizations. The preconditioner is defined by an auxiliary first order Finite Volume (FV) discretization that refines the original DG mesh, but can still be implemented algebraically. Different options exist to define the grid transfer between DG and FV. We suggest an ad hoc assignment of the unknowns as well as L_2 projections. We present new numerical results for the two-dimensional convection-diffusion equation in combination with the different transfer options, which demonstrate the quality and efficiency of the suggested preconditioner with regards to convergence speed up and CPU time. The suggested L_2 projection from this paper result in the best convergence speed up.

1 INTRODUCTION

In our research we are motivated by three-dimensional unsteady compressible flow applications. Our goal is the construction of efficient Jacobian-free preconditioners for high order Discontinuous Galerkin (DG) discretizations with implicit time integration. High order methods such as DG or Flux Reconstruction discretizations offer great potential for Large Eddy Simulation (LES) of turbulent flows with complex geometries, e.g. jet engines. The idea of these discretizations is an element-wise polynomial

approximation, with possible discontinuities across element interfaces [1, 2]. The degrees of freedom are coupled across faces and inside each element, resulting in dense element-wise computation. Thus, these methods are very well suited for high performance computing [3, 4]. We consider a specific DG variant, the DG Spectral Element Method (DG-SEM), see e.g. [5]. This discretization is based on a Lagrange type (nodal) basis with Legendre-Gauss-Lobatto (LGL) quadrature nodes, with collocation of the discrete integration. Moreover, the same Lagrange basis is used for ansatz and test space. These choices yield DG operators that satisfy the summation-by-parts (SBP) property [6], which is the discrete analogue to integration-by-parts. This property is key to construct methods that are discretely entropy stable and/or kinetic energy preserving.

Discontinuous Galerkin discretization in space results in a big system of ODEs. In a lot of challenging compressible turbulent flow applications as e.g. the design of jet engines, airplanes and windturbines, these systems are stiff due to geometry features and thin boundary layers. It is therefore of advantage to use implicit time integrators to overcome the deficiency of explicit ones with restrictive CFL conditions. However, this causes challenges in terms of CPU time and memory consumption. In [7], Vincent and Jameson mention that solvers for linear and nonlinear equation systems are severely lacking for 3D DG applications, and are therefore one of four major obstacles that need to be solved if high order methods are to be widely adopted by e.g. industry. Full-Approximation multigrid (FAS-Multigrid) schemes and preconditioned Jacobian-Free Newton-Krylov methods (JFNK) combined with multigrid preconditioners are candidates for such efficients solvers [8, 9]. We focus on the latter one and the construction of efficient preconditioners. DG systems have a block structure with weakly coupled blocks at the faces. The blocks can be of very large size depending on the order of the DG method: the number of unknowns per element increases dramatically with increasing polynomial degree and dimension, leading to large dense Jacobian blocks [10, 11]. While for Euler or Navier-Stokes equations the block size of a Finite Volume method is $2+d$ with dimension d, it is $(d+2)\,(p+1)^d$ for DG-SEM with p-th degree polynomials on quadrilateral grids. For degree 2 polynomials in 3D this results in dense blocks of size 135×135. Thus, the favorable memory consumption of the JFNK approach is obsolete if the preconditioner requires the storage of the DG system Jacobian.

In this article we extend the idea for the construction of a well-performing low storage preconditioner for the JFNK approach, i.e. a Jacobian-free preconditioner as suggested in [12, 13], to the two-dimensional case and consider a convection-diffusion problem. This extension is straightforward using a tensor product ansatz for the DG-SEM discretization. The main ingredient is the construction of a simplified replacement operator. While it is possible to choose a different polynomial order in the element to generate a replacement operator as in [14, 10], we aim to retain the number of DOFs in our replacement operator by introducing subcells in each element, namely $p+1$ in each spatial direction. A related approach was proposed in [15] for spectral difference (SD) methods. The simplest replacement operator is a first order Finite Volume (FV) discretization defined on this subcell grid. To do so, the nodal DOFs from the DG method need to be reinterpreted as unknowns for the FV method. In extension to previous work, we consider different transfer operators between the DG-SEM and FV discretization based on L_2 projections.

We measure the quality and efficiency of the suggested preconditioners by considering the convergence rate of the Krylov subsolver as well as the CPU time. Moreover, the FV discretization itself can be interpreted as a reference preconditioner. This is motivated by the fact that a well constructed multigrid method should approximate the FV method as well as possible. All results presented in this paper are produced using the Python bindings in the Distributed and Unified Numerics Environment (DUNE) [16].

This is a modular toolbox for solving partial differential equations with grid-based methods, ensuring efficiency in scientific computations and supporting high-performance computing applications. Numerical methods can be developed and tested in Python, before they are transferred to C++ to improve performance efficiency.

In the remainder of this paper, we first describe the here considered problem, the two-dimensional convection diffusion-equation. We then present the DG-SEM discretization and the FV subcell discretization. In section 3 we describe how to construct an agglomeration multigrid preconditioner and present the different transfer operators between DG and FV. In the last part of the paper, we show numerical experiments to validate the approach and draw our conclusions.

2 PROBLEM DESCRIPTION AND DISCRETIZATION

In this paper we consider the two-dimensional linear convection-diffusion equation

$$\begin{aligned} \mathbf{u}_t + \mathbf{b}\cdot\nabla\mathbf{u} - \varepsilon\Delta\mathbf{u} &= \mathbf{g}(\mathbf{u}), \\ \mathbf{u}_y|_{\Omega_{SN}} = \mathbf{0},\ \mathbf{u}|_{\partial\Omega_{WE}} &= \mathbf{0}, \\ \mathbf{u}(\mathbf{X},0) &= \exp(-10\|\mathbf{x}\|^2), \end{aligned} \tag{1}$$

with $(\mathbf{X},t)\in\Omega\times[0,T]$, $\Omega=[-1,1]^2$, $\mathbf{b}=\frac{1}{2}(\sqrt{3},1)$ and $\varepsilon\in\mathbb{R}^+$.

2.1 Discontinuous Galerkin Spectral Element Method

We discretize the spatial components of (1) using a discontinuous Galerkin method and rewrite problem (1) as

$$\mathbf{u}_t + \nabla\cdot\mathbf{f}(\mathbf{u},\nabla\mathbf{u}) = \mathbf{0},\ (\mathbf{X},t)\in\Omega\times[0,T] \text{ with } \Omega\subset\mathbb{R}^d. \tag{2}$$

To derive the discretization we consider the weak form of the conservation law

$$\int_\Omega \mathbf{u}_t\psi\, d\Omega + \int_\Omega \nabla\cdot\mathbf{f}\psi\, d\Omega = \mathbf{0}, \tag{3}$$

using test functions ψ from some test space. Integration by parts yields

$$\int_\Omega \mathbf{u}_t\psi\, d\Omega + \int_{\partial\Omega}\mathbf{f}\cdot\mathbf{n}\psi\, d\mathbf{s} - \int_\Omega \mathbf{f}\cdot\nabla\psi\, d\Omega = \mathbf{0}. \tag{4}$$

In the next step, the domain Ω is subdivided into cells Ω_i. In each cell the solution $\mathbf{u}$ and the flux function $\mathbf{f}(\mathbf{u},\nabla\mathbf{u})$ are approximated by a polynomial. In the following we assume that $\Omega\subset\mathbb{R}^2$, which simplifies the notation. Moreover, we map all cells Ω_i onto the unit cell $\hat{\Omega} := [-1,1]^2$ as described in [1, 17]. Then the numerical approximation on the unit cell is given by

$$\mathbf{u}(\mathbf{x},t)|_{\hat{\Omega}} \approx \quad \mathbf{u}^P(\mathbf{x},t) = \textstyle\sum_{j=1}^{N_x}\sum_{k=1}^{N_y}\mathbf{u}_{jk}(t)\varphi_{jk}(\mathbf{x}). \tag{5}$$

The choice of the basis functions gives rise to different DG methods. We consider here the DG spectral element method (DG-SEM), where the same Lagrange basis is used for ansatz and test space. Moreover, the flux function is approximated by

$$\mathbf{f}(\mathbf{u},\nabla\mathbf{u})|_{\hat{\Omega}} \approx \mathbf{f}^P(\mathbf{u},\mathbf{x},t) = \sum_{j=1}^{N_x}\sum_{k=1}^{N_y}\mathbf{f}_{jk}(t)\varphi_{jk}(\mathbf{x}), \tag{6}$$

with polynomial basis functions $\varphi_{jk}(\mathbf{x}) = \varphi_j(x)\varphi_k(y)$ of degree p and $\mathbf{f}_{jk}(t) = \mathbf{f}(\mathbf{u}_{jk}(t), \nabla\mathbf{u}_{jk}(t))$. The global solution is then approximated by a piecewise polynomial. We only have continuity inside the elements, but not on the interfaces, i.e. we allow for discontinuous polynomial approximations. The test functions are also chosen to be polynomials of degree up to p. Then (4) becomes on the reference cell

$$J\int_{\hat{\Omega}} \mathbf{u}_t^P \varphi_{jk}\, d\hat{\Omega} + \int_{\partial\hat{\Omega}} \mathbf{f}^P \cdot \mathbf{n}\varphi_{jk}\, d\mathbf{s} - \int_{\hat{\Omega}} \mathbf{f}^P \cdot \nabla\varphi_{jk}\, d\hat{\Omega} = \mathbf{0},\ j = 1,\dots,N_x, k = 1,\dots,N_y, \tag{7}$$

with $J = \frac{\Delta x \Delta y}{4}$. As polynomial basis for the DG-SEM method we choose a Lagrange basis with a nodal basis based on Legendre-Gauss-Lobatto (LGL) nodes, which are defined on the unit cell $\hat{\Omega}$. Next we apply a Gaussian quadrature to approximate the integrals. For the DG-SEM method, interpolation and quadrature are collocated. Thus polynomials up to degree $2p-1$ can be approximated exactly.

This gives for the mass integral

$$J\int_{\hat{\Omega}} \mathbf{u}_t^P \varphi_{jk}\, d\hat{\Omega} = J\sum_{l=1}^{N_x}\sum_{m=1}^{N_y} \dot{\mathbf{u}}_{lm}(t)\int_{\hat{\Omega}} \varphi_{lm}\varphi_{jk}\, d\hat{\Omega} \approx J\dot{\mathbf{u}}_{jk}(t)\omega_j\omega_k,\ j = 1,\dots,N_x, k = 1,\dots,N_y. \tag{8}$$

Defining a vector of coefficients $\tilde{\mathbf{u}}$ and a diagonal mass matrix $\mathbf{M}$ with diagonal elements given by the quadrature weights, equation (8) can be written more compactly as $J\mathbf{M}\dot{\tilde{\mathbf{u}}}$.

Applying Gaussian quadrature to the volume integral yields

$$\begin{aligned}\int_{\hat{\Omega}} \mathbf{f}^P \cdot \nabla\varphi_{jk}\, d\hat{\Omega} &= \int_{\hat{\Omega}} \sum_{l=1}^{N_x}\sum_{m=1}^{N_y} \mathbf{f}_{lm}(t)\varphi_{lm} \cdot \nabla\varphi_{jk}\, d\hat{\Omega} \\ &\approx \sum_{l=1}^{N_x} \omega_l\omega_k(\mathbf{f}_{lk})_1\varphi_j'(x_l) + \sum_{m=1}^{N_y} \omega_j\omega_m(\mathbf{f}_{jm})_2\varphi_k'(x_m),\ j = 1,\dots,N_x, k = 1,\dots,N_y.\end{aligned} \tag{9}$$

With $\tilde{\mathbf{f}}$ the vector of evaluations at the quadrature nodes and $\mathbf{S}$ such that

$$\begin{aligned}(\mathbf{S}_{jk})_1 &= \omega_l\omega_k\varphi_j'(\mathbf{x}_l),\ j = 1,\dots,N_x, k = 1,\dots,N_y, \\ (\mathbf{S}_{jk})_2 &= \omega_j\omega_m\varphi_k'(\mathbf{x}_m),\ j = 1,\dots,N_x, k = 1,\dots,N_y,\end{aligned} \tag{10}$$

equation (9) can be written compactly as $\sum_{i=1}^{2} \mathbf{S}_i(\tilde{\mathbf{f}})_i$.

The boundary integral is the sum of the four integrals on the edges of the reference cell. Then the integral can be calculated as

$$\begin{aligned}\int_{\partial\Omega_i} \mathbf{f}^P \cdot \mathbf{n}\varphi_{jk}\, d\mathbf{s} &\approx \omega_k(\mathbf{f}^P(\mathbf{u},(1,y_k),t))_1\varphi_j(1) - \omega_k(\mathbf{f}^P(\mathbf{u},(-1,y_k),t))_1\varphi_j(-1) \\ &+ \omega_j(\mathbf{f}^P(\mathbf{u},(x_j,1),t))_2\varphi_k(1) - \omega_j(\mathbf{f}^P(\mathbf{u},(x_j,-1),t))_2\varphi_k(-1), \\ &\quad j = 1,\dots,N_x, k = 1,\dots,N_y.\end{aligned} \tag{11}$$

We define the vector of function evaluations at the quadrature nodes on the surface as $\hat{\mathbf{f}}$ and $\mathbf{M}_{jk}^s = \omega_j\varphi_k(\mathbf{x}_{face})$, then equation (11) can be written compactly as $\sum_{i=1}^{4}(\mathbf{M}_{jk}^s)_i(\hat{\mathbf{f}})_i$. Next, the boundary terms need to be coupled with the neighboring cells and since we allow for discontinuities at the cell boundaries for numerical solution, numerical flux functions are needed. We approximate

$$\mathbf{f}^P(\mathbf{x},t) \approx \mathbf{f}_c^N(\mathbf{u}^P(\mathbf{x},t), \hat{\mathbf{u}}^P(\mathbf{x},t); \mathbf{n}) + \mathbf{f}_d^N(\mathbf{u}^P(\mathbf{x},t), \nabla\mathbf{u}^P(\mathbf{x},t), \hat{\mathbf{u}}^P(\mathbf{x},t), \nabla\hat{\mathbf{u}}^P(\mathbf{x},t); \mathbf{n}), \tag{12}$$

where we write the numerical flux as a sum of the convective and the diffusive numerical fluxes, $\hat{\mathbf{u}}^P$ is the polynomial in the neighboring cell and $\mathbf{n}$ the normal.

In this paper we use the upwind flux for the convective numerical fluxes and an interior penalty Galerkin flux in the diffusive part, see [18]:

$$\mathbf{f}_{IP}^N(\mathbf{u}^P(\mathbf{x},t), \nabla\mathbf{u}^P(\mathbf{x},t); \hat{\mathbf{u}}^P(\mathbf{x},t), \nabla\hat{\mathbf{u}}^P(\mathbf{x},t); \mathbf{n}) := \{\{\varepsilon\nabla\mathbf{u}^P\}\} - \frac{\eta}{h}\varepsilon[\![\mathbf{u}^P]\!], \tag{13}$$

with the classic average $\{\{\cdot\}\}$ and jump $[\![\cdot]\!]$, local mesh width h and a stability coefficient η which we set

$$\eta = \begin{cases} 10 \times \text{order}^2, & \text{if order} > 0, \\ 1, & \text{else.} \end{cases} \tag{14}$$

On the reference cell we obtain a system of ordinary differential equations:

$$J\mathbf{M}\tilde{\mathbf{u}}_t + \sum_{j=1}^{4} \mathbf{M}_j^S(\mathbf{f}^N)_j - \sum_{i=1}^{2} \mathbf{S}_i(\tilde{\mathbf{f}})_j = \mathbf{0}. \tag{15}$$

Collecting the equations for all elements in one big system with vector of unknowns $\underline{\mathbf{u}}$ and applying implicit Euler time stepping gives

$$\begin{aligned} &\underline{\mathbf{u}}^{n+1} - \underline{\mathbf{u}}^n - \Delta t\underline{\mathbf{G}}\underline{\mathbf{u}}^{n+1} = \mathbf{0}, \\ &\Leftrightarrow (\underline{\mathbf{I}} - \Delta t\underline{\mathbf{G}})\underline{\mathbf{u}}^{n+1} = \underline{\mathbf{u}}^n, \end{aligned} \tag{16}$$

where $\underline{\mathbf{G}}$ is a matrix since we have a linear problem. This linear system of equations can be solved using a preconditioned Jacobian-free GMRES method. Matrix-vector products with the Jacobian are replaced by a finite difference approximation to get a Jacobian-free variant.

In the next section we describe how to construct a preconditioner for this linear system based on a multigrid (MG) method for a Finite Volume replacement operator.

3 MULTIGRID BASED PRECONDITIONER

In order to construct a preconditioner for the linear system (16), we replace the DG-SEM problem by a lower order FV problem. This ansatz should not be confused with a p-multigrid method. It is motivated by the equivalence of the DG-SEM discretization to specific FV discretizations [19]. Instead of constructing the equivalent FV operator, we construct a first order FV replacement operator. This simplifies the construction of the Jacobian and allows us to use the available knowledge about fast multigrid methods for FV discretizations. The question arises how to transfer the degrees of freedom (DOFs) back and forth in these two discretizations inside each DG element. We present different transfer strategies later in this section and explain first the construction of FV agglomeration multigrid preconditioners.

In the multigrid method considered in this paper, the coarse grid problems are given by the FV discretization on the corresponding grid. The grid transfer is based on agglomeration: to restrict fine grid values, they are summed, weighted by the volumes of the respective cells and divided by the total volume. This is denoted by the restriction operator $\mathbf{R}_{l-1}^l$, where l indicates the grid level s.t. $l = 0$ is the coarsest level. The corresponding prolongation operator $\mathbf{P}_l^{l-1}$ is defined via injection, where the value in the coarse cell is taken for all corresponding fine cells.

A generic smoother is an iterative method for the solution of a linear system $\mathbf{A}_l\mathbf{x}_l = \mathbf{b}_l$ given by

$$\mathbf{x}_l^{k+1} = \mathbf{M}_{S,l}\mathbf{x}_l^k + \mathbf{N}_{S,l}^{-1}\mathbf{b}_l, \tag{17}$$

with an iteration matrix $\mathbf{M}_{S,l}$ and a nonsingular matrix $\mathbf{N}_{S,l}^{-1}$. As before, the index l specifies the multigrid level and S denotes that this iterative method represents a smoother. It is possible to construct an MG preconditioner based on a V- or a W-cycle, as well as several consecutive cycles. The number of pre- and postsmoothing steps is also flexible. A multigrid algorithm to solve the linear problem $\mathbf{Ax} = \mathbf{b}$ is presented in the following pseudo-code:

`function MG(`$\mathbf{x}_l, \mathbf{b}_l, l$`):`
if($l = 0$) solve $\mathbf{A}_l\mathbf{x}_l = \mathbf{b}_l$
else:
- for $pre = 1, \ldots, N_{pre}$: $\mathbf{x}_l = \mathbf{M}_{S,l}\mathbf{x}_l + \mathbf{N}_{S,l}^{-1}\mathbf{b}_l$ (presmoothing)

if($l > 0$)
- $\mathbf{r}_{l-1} = \mathbf{R}_{l-1}^{l}(\mathbf{A}_l\mathbf{x}_l - \mathbf{b}_l)$ (restriction)
- $\mathbf{v}_{l-1} = 0$
  - for $j = 1, \ldots, \gamma$: $\mathbf{v}_{l-1} = MG(\mathbf{v}_{l-1}, \mathbf{r}_{l-1}, l-1)$
- $\mathbf{s}_l = \mathbf{s}_l - \mathbf{P}_l^{l-1}\mathbf{v}_{l-1}$ (fine grid correction)
- for $post = 1, \ldots, N_{post}$: $\mathbf{s}_l = \mathbf{M}_{S,l}\mathbf{x}_l + \mathbf{N}_{S,l}^{-1}\mathbf{b}_l$ (postsmoothing)

This gives rise to an iterative method of the form

$$\mathbf{x}_l^{k+1} = \mathbf{M}_{MG}\mathbf{x}_l^k + \mathbf{N}_{MG}^{-1}\mathbf{b}_l. \tag{18}$$

The multigrid preconditioner is then given by $\mathbf{N}_{MG}^{-1}$.

3.1 Smoothers: Pseudo time stepping methods

As smoothers for the multigrid preconditioner we consider Runge-Kutta schemes. Adding a pseudo time derivative to an initial value problem $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x}(0) = \mathbf{x}^0$ yields

$$\frac{\partial \mathbf{x}}{\partial t^*} + \mathbf{Ax} - \mathbf{b} = \mathbf{0}. \tag{19}$$

Then any time-stepping method becomes a smoother. In this paper we consider low storage explicit s-stage Runge-Kutta schemes of the form

$$\begin{aligned} \mathbf{x}_j &= \mathbf{x}^n + \Delta t^* \alpha_j c(\mathbf{b} - \mathbf{Ax}^n),\ j = 1, \ldots, s \\ \mathbf{x}^{n+1} &= \mathbf{x}^n + \Delta t^* c^*(\mathbf{b} - \mathbf{Ax}_s), \end{aligned} \tag{20}$$

with $\Delta t^* = \frac{c^*}{CFL_{adv} + CFL_{diff}}$, $CFL_{adv} =:= \frac{|\mathbf{b}|\Delta t*}{\Delta \mathbf{x}}$, $CFL_{diff} := \frac{2\varepsilon\Delta t*}{\Delta \mathbf{x}^2}$ and parameters α_j and c^*.

3.2 Transfer between DG and FV discretizations

In order to construct an FV replacement operator, the DG-SEM DOFs need to be reinterpreted as FV DOFs. In the following we discuss three different options to transfer the vector of unknowns between these two discretizations inside each DG element. The vector of unknowns in the DG-SEM discretization in one element is given by $\tilde{\mathbf{u}}$ and we define the corresponding vector in the FV discretization by $\hat{\mathbf{u}}$, where the number of volumes corresponds to the number of LGL nodes.

The first transfer option is referred to as *ad hoc assignment* of LGL nodes and FV cells in this paper and corresponds to the grid transfer presented in [13]. The idea is to assign the values of $\tilde{\mathbf{u}}$ to the values

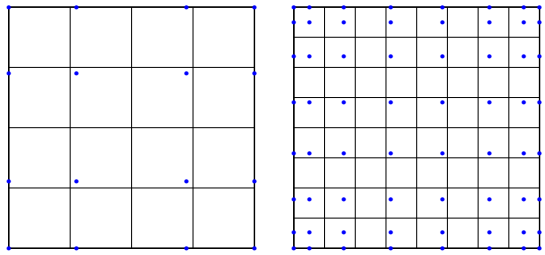


Figure 1: Two DG-SEM grid reference elements with LGL nodes (blue nodes), and an inlaid finite volume subgrid (black lines)

of $\hat{\mathbf{u}}$ based on the location of the DOF, i.e. each DG-SEM DOF is assigned to the closest FV DOF. A permutation matrix might be needed to take care of possible differences in the ordering of the DOFs. The distribution of 4 respective 8 LGL nodes and an overlaid equidistant FV grid is visualized in Figure 1. If every subgrid volume contains exactly one LGL node, a sufficient ad hoc assignment of DOFs between the two discretizations is provided, while for high order DG-SEM discretizations the FV subgrid may have volumes which do not contain any LGL node. Thus, some FV DOFs are assigned wrong values for their location. This could be improved by modifying the FV subgrid such that each volume contains exactly one LGL node.

Since the natural norm for DG methods is the L_2 norm, another idea for the space transfer is to consider an L_2 projection. To define it, either the basis functions of the FV space or the DG-SEM space can be used.

The basis functions in the FV space are already orthonormalized, and we define a projection matrix via $\mathbf{P}_{FV}\tilde{\mathbf{u}} = \hat{\mathbf{u}}$, defined as the inner L_2 product of the DG-SEM Lagrange basis and the FV basis functions. To transfer the DOFs from the FV grid to the DG grid, the reconstruction $\mathbf{P}_{FV}^{-1}$ can be used. This is not an L_2 projection, but gives good results in the numerical tests presented in the next section.

Another option is to define the L_2 projection matrix based on the DG-SEM space. To this end, the LGL basis has to be orthonormalized. Then the projection $\mathbf{P}_{DG}\hat{\mathbf{u}} = \tilde{\mathbf{u}}$ can be defined as $\mathbf{P}_{DG} := \mathbf{M}^{-1}\mathbf{P}_{FV}^T$ with the DG mass matrix $\mathbf{M}$ and the reverse transfer as $\mathbf{P}_{DG}^{-1}$.

4 NUMERICAL RESULTS

We test the efficiency of the presented preconditioner for the two-dimensional convection diffusion equation with $\mathbf{b} = \frac{1}{2}(\sqrt{3}, 1)$ and $\varepsilon = 1.e-4$. We show 40 iterations of the GMRES convergence rate using different FV based multigrid preconditioners. The number of multigrid levels is chosen such that the coarsest grid in the FV space corresponds to one DG cell. On the coarsest grid, the system is solved exactly using the sparse python solver `spsolve` which is based on an LU decomposition. For smoothing we use a 3-stage Runge-Kutta scheme with parameters $\alpha = [0.145, 0.395]$ and $c^* = 1.1$ [20]. We discretized the convection diffusion problem with a 4th and 8th order DG discretization and consider a $2^7 \times 2^7$ grid on the finest level. The time stepping width is set to be $\Delta t = 0.75$, which gives a hyperbolic CFL-like ratio of $\Delta t/\Delta x = 48$.

As a first numerical test we examine how accurate the replacement operator approximates the original

problem and how efficient the different transfer operators are. The convergence results for the reference preconditioner based on the replacement first order FV operator can be seen in Figure 2, on the left side for a 4th order DG method, in the following referred to as DG4, and on the right side for a 8th order DG method referred to as DG8. In both cases, L_2 projection gives better results than the ad hoc assignment. Since the L_2 norm is used for DG discretizations, these are the results we expected. L_2 projection on the FV space gives slightly better results, with a difference in residual of approximately $1.e-1$ after 40 GMRES iterations both for DG4 and DG8. Increasing the order of the DG method gives results with a more similar convergence rate for all transfer options while the overall quality of the FV reference preconditioner decreases from between $1e-7$ and $1e-11$ to between $1e-3$ and $1e-5$. Since the reference operator is of first order, it had to be expected that the reference preconditioner works better for lower DG discretizations.

Next, we test the suggested multigrid preconditioners. A good multigrid method should approximate the reference operator as good as possible. We construct a 3-level preconditioner based on the FV replacement operator and test different combinations of pre- and postsmoothing for one V-cycle for DG4 and a 4-level preconditioner for DG8. In the legend of the plots, the first two numbers specify the pre- and postsmoothing steps on the finest grid and the last two numbers the pre- and postsmoothing steps on the coarser grids. On the coarsest grid we solve the system directly without any smoothing steps. As before, on the left side we show the results for the 4th and on the right side for the 8th order DG discretization.

In Figure 3 the convergence results can be seen using the ad hoc transfer between the DG and FV space. We notice that the suggested multigrid preconditioners give convergence rates which mimic the FV reference preconditioner to some extent and the preconditioner works better than an RK3 preconditioner. Nevertheless, the overall convergence improvement is not very good when increasing the DG order, for the reference preconditioner we decrease the relative residual by $1e-7$ in 40 iterations for a 4th order method compared to decreasing the relative residual by $1e-3$ in 40 iterations for a 8th order method. Moreover, the efficiency of the multigrid preconditioner is highly influenced by the number of smoothing steps on the finer grid levels.

In Figure 4 we consider interpolation by an L_2 projection on the FV grid. This transfer gives overall better results than the previous one, for the reference preconditioner we get a decrease the relative residual by $1e-11$ in 40 iterations for a 4th order method compared to decreasing the in relative residual by $1e-5$ for a 8th order method. With the suggested multigrid preconditioners we can decrease the relative residual between $1e-5$ and $1e-9$ for the 4th order DG method and between $1e-1$ and $1e-4$ for the 8th order DG method. Again it is of benefit to apply the smoother several times on the finest grid in order to get a convergence rate close to the one of the reference preconditioner. Moreover, postsmoothing on the finest grid improves the convergence rate. In comparison to the ad hoc transfer, the influence of the smoother on the multigrid preconditioner is more visible in this case and results in convergence rates with a difference of approximately one order of magnitude. The RK3 preconditioner does not perform well.

The results using an L_2 projection on the DG grid can be seen in Figure 5. The overall convergence improvement is about one order of magnitude worse than for the other L_2 projection both for the 4th and 8th order method. But we notice that the reference preconditioner is better approximated by the suggested multigrid preconditioners and the difference between the tested smoothing strategies in the multigrid preconditioners decreases. For the 4th order method, the relative residual can be decreased between $1e-7$ and $1e-9$ and for the 8th order method between $1e-3$ and $1e-3$. Again, presmoothing

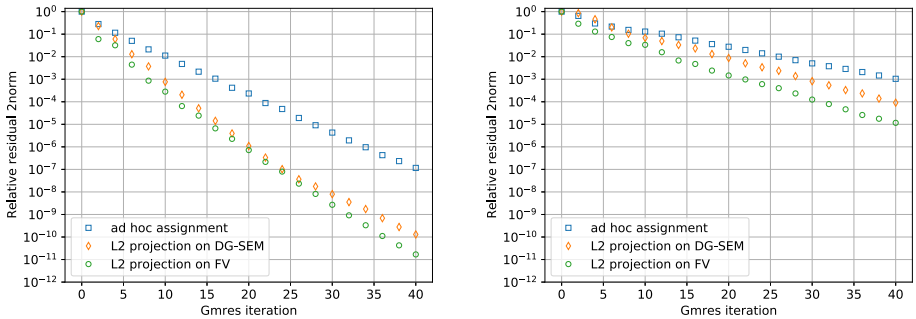


Figure 2: GMRES convergence rate for the reference FV preconditioner for different transfer strategies, 16384 DOFs for DG4 (left) and DG8 (right)

on the finest grid results in the best preconditioner. As before, the RK3 preconditioner is beaten by the multigrid preconditioners.

4.1 CPU time

Since the efficiency of the preconditioners is also influenced by their computational effort, we consider the corresponding CPU times. The results can be seen in Tables 1 and 2. As expected, applying the smoother on different levels has a direct impact on the CPU time. The multigrid based preconditioners are of very similar cost for different pre- and postsmoothing strategies, with more smoothing on the finest grid being more expensive. While the convergence rates are highly influenced by the choice of the transfer functions, this has not a huge impact on the measured CPU time. The same holds for increasing the order of the DG method. Setting the CPU times in relation to the convergence plots discussed before shows that the multigrid based preconditioner with L_2 projection on DG-SEM is most efficient.

Table 1: CPU time in seconds for 40 GMRES iterations, DG4 and 16384 DOFs

	ad hoc assignment	L2 on DG-SEM	L2 on FV
no prec.	6.39e-1	5.74e-1	6.69e-1
4× smoother	6.66e0	6.99e0	6.72e0
10,11	1.08e1	1.10e1	1.12e1
01, 11	1.13e1	1.11e1	1.09e1
22, 11	3.10e1	2.78e1	2.72e1
exact FV	4.90e0	4.82e0	5.06e0

5 CONCLUSIONS

In this paper we extended a multigrid preconditioning strategy for use in Jacobian-free Newton-Krylov methods for the solution of equation systems arising from implicit DG discretizations as presented in [13] to the two-dimensional case. The core idea is to make use of an auxiliary first order

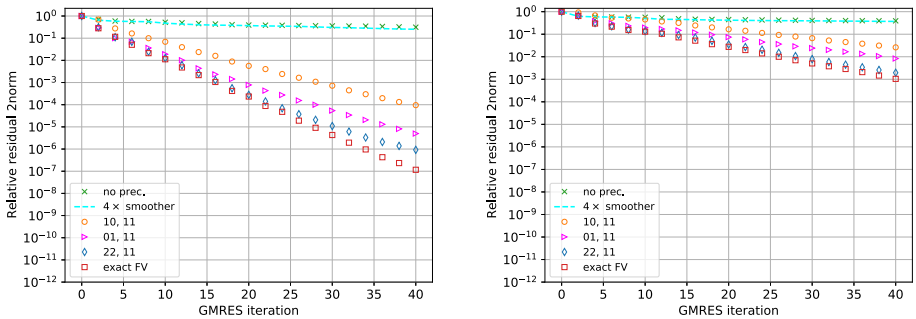


Figure 3: GMRES convergence rate for transfer via ad hoc assignment, 16384 DOFs for DG4 (left) and DG8 (right)

Table 2: CPU time in seconds for 40 GMRES iterations, DG8 and 16384 DOFs

	ad hoc assignment	L2 on DG-SEM	L2 on FV
no prec.	5.11e-1	5.47e-1	5.78e-1
4× smoother	6.22e0	6.82e0	7.17e0
10,11	1.48e1	1.52e1	1.53e1
01, 11	1.49e1	1.48e1	1.62e1
22, 11	3.27e1	3.17e1	3.32e1
exact FV	5.14e0	5.51e0	5.10e0

FV discretization that refines the original DG mesh, but can still be implemented algebraically. As smoother, we considered a 3-stage Runge-Kutta method. Numerical results for the two-dimensional convection-diffusion equation were presented. The convergence results of the proposed preconditioner are promising, being close to a quasi-exact preconditioner. Our results indicate that the performance of the preconditioner is highly influenced by the choice of grid transfer between DG and FV as well as on the order of the DG method. An L_2 projection improves the convergence rate compared to a simple ad hoc assignment. The projection can be precomputed for a given DG order, either based on the DG space or the FV space. For the presented problem, the L_2 projection on the FV grid is slightly more efficient than an L_2 projection on the DG-SEM grid. With both strategies we can decrease the GMRES residual from not even $1.e-1$ without preconditioner to approximately $1.e-5$-$1.e-11$ for a 4th order DG method and approximately $1.e-1$-$1.e-5$ for a 8th order DG method with the suggested multigrid preconditioners for the here presented L_2 grid transfer operators. Furthermore, our numerical experiments show the importance of the smoothing strategy in the multigrid method, both for the convergence rate and the CPU time. In order to improve the convergence results without increasing the CPU time drastically, more efficient and advanced smoothers can be used, for example a W3 smoother as presented in [13].

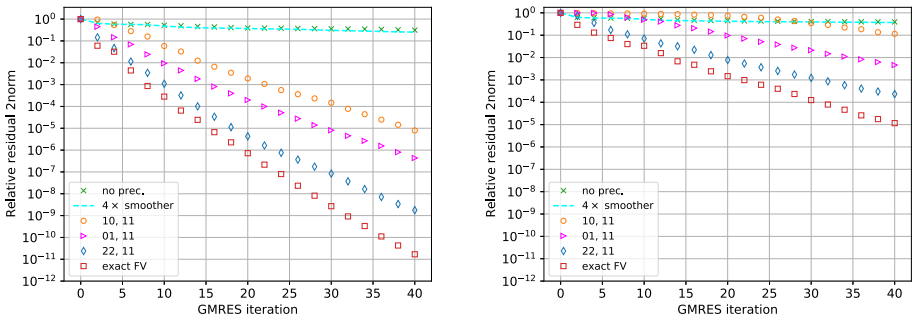


Figure 4: GMRES convergence rate for transfer via L2 projection on FV grid, 16384 DOFs for DG4 (left) and DG8 (right)

Funding

Philipp Birken and Lea Versbach gratefully acknowledge the support of the Swedish Research Council, grant number 2015-04133. Gregor Gassner has been supported by the European Research Council (ERC) under the EUs 8ths Framework Program Horizon 2020 with the research project Extreme, ERC grant agreement no. 714487.

REFERENCES

[1] D. A. Kopriva. *Implementing Spectral Methods for Partial Differential Equations*. Springer, 2009.

[2] H. T. Huynh. A flux reconstruction approach to high-order schemes including discontinuous Galerkin methods. In *18th AIAA Computational Fluid Dynamics Conference*, page 4079, 2007.

[3] F. Hindenlang, G. J. Gassner, C. Altmann, A. Beck, M. Staudenmaier, and C.-D. Munz. Explicit discontinuous Galerkin methods for unsteady problems. *Comp. & Fluids*, 61:86–93, 2012.

[4] P. Vincent, F. Witherden, B. Vermeire, J. S. Park, and A. Iyer. Towards green aviation with python at petascale. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11. IEEE, 2016.

[5] D. A. Kopriva, S. L. Woodruff, and M. Y. Hussaini. Computation of electromagnetic scattering with a non-conforming discontinuous spectral element method. *Int. J. Numer. Methods Eng.*, 53(1):105–122, 2002.

[6] G. J. Gassner. A skew-symmetric discontinuous Galerkin spectral element discretization and its relation to SBP-SAT finite difference methods. *SIAM J. Sci. Comput.*, 35(3):A1233–A1253, 2013.

[7] P. E. Vincent and A. Jameson. Facilitating the adoption of unstructured high-order methods amongst a wider community of fluid dynamicists. *Math. Model. Nat. Phenom.*, 6(3):97–140, 2011.

[8] D. A. Knoll and D. E. Keyes. Jacobian-Free Newton-Krylov Methods: A Survey of Approaches and Applications. *J. Comput. Phys.*, 193(2):357397, 2004.

[9] P. Birken and A. Jameson. On nonlinear preconditioners in Newton-Krylov methods for unsteady flows. *Int. J. Numer. Methods Fluids*, 62(5):565–573, 2010.

[10] P. Birken, G. J. Gassner, M. Haas, and C.-D. Munz. Preconditioning for modal discontinuous

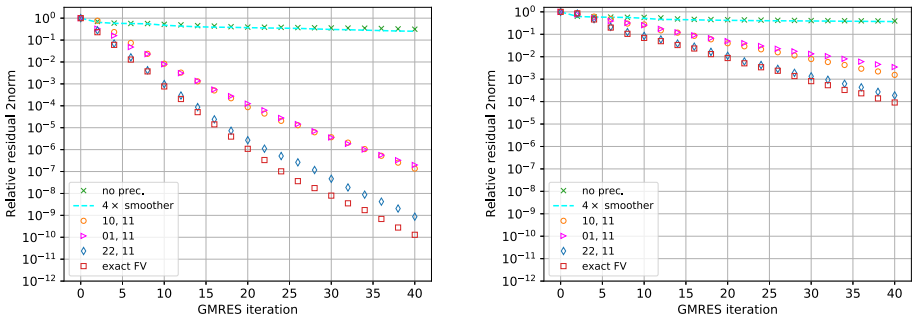


Figure 5: GMRES convergence rate for transfer via L2 projection on DG-SEM grid, 16384 DOF for DG4 (left) and DG8 (right)

Galerkin methods for unsteady 3D Navier-Stokes equations. *J. Comp. Phys.*, 240:20–35, 2013.

[11] P. Birken. *Numerical methods for the unsteady compressible Navier-Stokes equations, Habilitation Thesis*. Habilitation thesis, Universität Kassel, 2012.

[12] L. M. Versbach, P. Birken, and G. J. Gassner. Finite volume based multigrid preconditioners for discontinuous Galerkin methods. *PAMM*, 18(1):e201800203, 2018.

[13] P. Birken, G. J. Gassner, and L. M. Versbach. Subcell finite volume multigrid preconditioning for high-order discontinuous Galerkin methods. *Int. J. Comput. Fluid Dyn.*, pages 1–9, 2019.

[14] K. J. Fidkowski, J. Oliver, T. A .and Lu, and D. L. Darmofal. p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations. *J. Comput. Phys.*, 207(1):92–113, 2005.

[15] Y. Allaneau, L. Y. Li, and A. Jameson. Convergence Acceleration of High Order Numerical Simulations using a Hybrid Spectral Difference-Finite Volume Multigrid Method. In *7th International Conference on Computational Fluid Dynamics (ICCFD7)*, pages 9–13, 2012.

[16] P. Bastian, M. Blatt, M. Dedner, N.-A. Dreier, R. Engwer, Ch. Fritze, C. Gräser, Ch. Grüninger, D. Kempf, R. Klöfkorn, M. Ohlberger, and O. Sander. The Dune framework: Basic concepts and recent developments. *CAMWA*, 2020.

[17] G. Karniadakis and S. Sherwin. *Spectral/hp element methods for computational fluid dynamics*. Oxford University Press, 2013.

[18] S. Brdar, A. Dedner, and R. Klöfkorn. Compact and stable Discontinuous Galerkin methods for convection-diffusion problems. *SIAM J. Sci. Comput.*, 34(1):A263–A282, 2012.

[19] T. C. Fisher and M. H. Carpenter. High-order entropy stable finite difference schemes for nonlinear conservation laws: Finite domains. *J. Comput. Phys.*, 252:518–557, 2013.

[20] P. Birken. Optimizing Runge-Kutta smoothers for unsteady flow problems. *Electron. Trans. Numer. Anal.*, 39(1):298–312, 2012.

Paper IV

Theoretical and Practical Aspects of Space-Time DG-SEM Implementations

Lea M. Versbach[1*], Viktor Linders[1], Robert Klöfkorn[1], Philipp Birken[1]

[1]Centre for Mathematical Sciences, Numerical Analysis, Lund University, Lund, Sweden

*lea_miko.versbach@math.lu.se

Abstract

In this paper we discuss two approaches for the formulation and implementation of space-time discontinuous Galerkin spectral element methods (DG-SEM). In one, time is treated as an additional coordinate direction and a Galerkin procedure is applied to the entire problem. In the other, the method of lines is used with DG-SEM in space and the fully implicit Runge-Kutta method Lobatto IIIC in time. The two approaches are mathematically equivalent in the sense that they lead to the same discrete solution. However, in practice they differ in several important respects, including the terminology used to describe them, the structure of the resulting software, and the interaction with nonlinear solvers. Challenges and merits of the two approaches are discussed with the goal of providing the practitioner with sufficient consideration to choose which path to follow. Additionally, implementations of the two methods are provided as a starting point for further development. Numerical experiments validate the theoretical accuracy of these codes and demonstrate their utility, even for 4D problems.

1 Introduction

Typically, partial differential equations are numerically treated with a method of lines ansatz; the spatial directions are discretized first, leaving the time variable continuous. The resulting system of ordinary differential equations is then solved using a numerical method for initial value problems.

An alternative ansatz is to treat the time dimension simply as another coordinate direction, and discretize the whole space-time problem simultaneously, resulting in a fully discrete numerical scheme [55]. This approach has several advantages: Moving boundaries can be treated more easily [70] and parallelization in time is made possible [28]. However, it also imposes new challenges since the temporal direction is special and needs to follow a causality principle: The solution at a given time is affected and determined only by the solution at earlier times, never the other way around. An overview of space-time computations in practical engineering applications during the last 25 years can be found in [69].

In this paper we consider the discontinuous Galerkin spectral element method (DG-SEM); see e.g. [6] for an overview and [44] for a detailed exposition. These methods have been very successful for spatial discretizations as they are unstructured, of high order and are very suitable for high performance computing [47]. Further, DG-SEM fits the so-called Summation-By-Parts Simultaneous-Approximation-Term (SBP-SAT) framework [14, 30], implying L_2 stability for linear problems. Further, in the last decade, work within this framework has resulted in the development of entropy stable (i.e. nonlinearly stable) discretizations of arbitrarily high order [24, 23].

Our motivation to consider DG-SEM in a space-time formulation is twofold: Firstly, with a specific choice of numerical fluxes, entropy stability can be extended to incorporate the temporal discretization for hyperbolic conservation laws [26], thereby resulting in a nonlinearly stable, fully discrete scheme. Secondly, the formulation naturally allows for perfectly scaling parallelization in time, with a speedup equal to the number of discretization points within a time element. There are other approaches for parallelization in time that allow for much larger speedups, but need an initial factor of additional processors before giving any speedup at all [56].

There is a strong connection between DG discretizations in time and fully implicit Runge-Kutta (RK) methods: DG-SEM in time using an upwind numerical flux is equivalent to the Lobatto IIIC family of RK methods, in the sense that the two methods give yield the same numerical solution [8, 61]. This observation lends itself to two very different strategies for implementing DG-SEM in space and time. We can either use the method of lines with DG-SEM in space and Lobatto IIIC in time, or we can use space-time DG-SEM.

While mathematically equivalent, there are important differences between these two approaches:

- The theories of DG and RK methods have been developed largely independently. Hence, there is a language barrier between these communities, where different terminology is used to describe equivalent mathematical concepts.

- The two approaches lead to different algebraic systems of linear or nonlinear equation. If solved exactly, these systems have the same solutions. However, in practice these solutions must be approximated, typically using iterative solvers. The interplay between iterative methods and the algebraic systems will in general be different, thus the two methods yield unequal numerical solutions.

- Implementing the two approaches lead to very different software structure, in particular if we wish to reuse pre-existing software. This implies that various numerical tools and techniques may be more readily accessible in one implementation than the other, depending on whether the DG or the RK approach is chosen.

In this paper, we discuss these differences in detail so that practitioners can make an educated choice about which path to follow. Further, we present a code base for the two approaches that may be used as a basis for further development of the methods. In particular, we make use of the open source softwares Dune, the Distributed and Unified Numerics Environment, which is a modular toolbox for solving partial differential equations (PDEs) with grid-based methods [18]. We also make use of Assimulo [2], a solver package for initial value problems.

This paper is organized as follows: Following a brief literature review below, our target equation and choice of software is introduced in Section 2. In Section 3 we introduce the

method of lines approach using DG-SEM with Lobatto IIIC for time stepping. In Section 4 the space-time DG-SEM is described. Throughout, code snippets are included to illustrate the details of the implementations. Theoretical aspects of the two approaches are included in Section 5. In particular, we demonstrate the mathematical equivalence of DG-SEM in time and Lobatto IIIC methods, and relate the terminology employed by the DG and RK communities. Practical aspects of the respective implementations are the subject of Section 6. Here we compare algorithmic and implementation specific requirements and merits of the two approaches. In Section 7 we present numerical experiments that validate some of our discussion points before we finish the article with some concluding remarks in Section 8. The Appendix A contains instructions on how to install the code discussed in this paper.

1.1 Further reading

For the practitioner who wishes to delve deeper into various aspects of the topics discussed in this paper, we here suggest a few places to start for further reading.

The book [44] provides much background material on spatial DG-SEM as well as a guide to its implementation. An overview of entropy stable DG-SEM is given in [31] and full mathematical detail is provided in [13]. The theory builds upon the SBP-SAT framework, reviews of which are found in [22, 68].

For a broad background on implicit Runge-Kutta methods, the book [35] is a good starting point. An overview of the properties of DG-SEM and other SBP-SAT methods for time integration viewed from the Runge-Kutta perspective is given in [51]. An evaluation of fully implicit RK methods for use in computational fluid dynamics is given in [38], including discussions of how to solve the nonlinear algebraic systems.

Several nonlinear solvers with application to RK and DG methods have recently been presented in the literature. For solver options based on Newton-type methods, see e.g. [60, 21] and the references therein. Solvers utilizing multigrid techniques are introduced and analyzed in [29, 71, 25].

The implementation of space-time methods with a focus on the challenge of 4D problems has been studied in [27], and the generation of different 4D space-time meshes have been presented in [4, 12].

2 Governing Equations and Simulation Software

We consider a general class of time dependent nonlinear advection-diffusion-reaction problems

$$\partial_t u = \mathcal{L}(u) \quad := \quad -\nabla \cdot \big(F_c(u) - F_v(u, \nabla u)\big) + S(u) \quad \text{in } \Omega \times (0, T) \tag{1}$$

for a vector valued function $u: \Omega \times (0, T) \to \mathbb{R}^r$ with $r \in \mathbb{N}^+$ components. Here, $\Omega \subset \mathbb{R}^d$, $d = 1, 2, 3$. Suitable initial and boundary conditions are assumed to be available. F_c and F_v describe the convective and viscous fluxes respectively, and S is a source term. We allow for the possibility that any of the coefficients in the partial differential equation (PDE) (1) depend explicitly on the spatial variable x and on time t, but to simplify the presentation we suppress this dependency in our notation. Further, any of the terms are allowed to be zero.

For the discretization of (1) we consider two approaches: The first is a method of lines approach, in which the spatial differential operator is discretized using a DG-SEM approximation, yielding a system of ordinary differential equations (ODEs). This system is then solved using a time stepping scheme. In particular, we consider the Lobatto IIIC family of implicit Runge-Kutta methods.

The second approach is to apply the DG-SEM methodology to the entire equation (1), thereby obtaining a fully implicit DG space-time discretization.

In the following we will include code snippets to clarify the overall structure of the mathematical formulations at hand and to illustrate how the two approaches can be implemented in an existing code base. We utilize Dune [3], which is a free and open source software framework for the grid-based numerical solution of PDEs. Dune provides one of the most flexible and comprehensive grid interfaces available, allowing n-dimensional grids, which we will use in this paper. Additionally, standard state-of-the-art features such as parallelization, grid adaptivity and load balancing, and moving grids are supported. From the variety of Dune modules available we will make use of the Python based front-end for Dune-Fem [19] and Dune-Fem-DG [17], which is able to handle weak forms of PDEs described in the Unified Form Language (UFL) [1]. As shown in the next section, the description of weak forms with UFL is straight forward and easy to use. Internally, PDEs described in UFL are translated into C++ code just-in-time, to ensure that the resulting simulation code is performant. For a more detailed description we refer to [19, 17] and the tutorial[1].

The implementation of the Lobatto IIIC method (see [49]) has been done in Assimulo [2], which is also a Python package that can be readily used together with Dune-Fem.

Comments on how to install Dune-Fem-DG and Assimulo are found in Appendix A.

3 Method of Lines DG-SEM

In this section we describe the method of lines (MOL) approach to discretizing (1). A generic DG method is first presented, followed by the specifications needed to obtain the DG-SEM. Finally, the Lobatto IIIC time stepping method is specified.

3.1 DG-SEM in Space

Given a tessellation $\mathcal{T}_h$ of the computational domain Ω into elements E with $\bigcup_{E\in\mathcal{T}_h} E = \Omega$, consider the piecewise polynomial space

$$V_h = \{\boldsymbol{v} \in L^2(\Omega, \mathbb{R}^r) : \boldsymbol{v}|_E \in [\mathcal{P}_p(E)]^r,\ E \in \mathcal{T}_h\},\quad p \in \mathbb{N}, \tag{2}$$

where $\mathcal{P}_p(E)$ is the space of polynomials whose degree do not exceed p. We let Γ_i denote the set of intersections between all pairs of elements in $\mathcal{T}_h$ and accordingly Γ the set of all intersections including the boundary of Ω. In Dune, the following commands generate the tessellation $\mathcal{T}_h$ and the space V_h:

[1] https://dune-project.org/sphinx/content/sphinx/dune-fem/

```
d = 2 # 1,2,3
from dune.grid import cartesianDomain, structuredGrid as leafGrid
# create grid that tessellates [0,1]^d with 10 elements in each coordinate direction
T_h = leafGrid(cartesianDomain([0]*d, [1]*d, [10]*d))

from dune.fem.space import dglagrangelobatto
p = 3 # polynomial degree
# create DG space with Lagrange basis and Gauss-Lobatto interpolation points
V_h = dglagrangelobatto( T_h, order=p )
```

We seek an approximate solution $\boldsymbol{u}_h \in V_h$ by discretizing the spatial operator $\mathcal{L}(u)$ in (1). To this end we define for all test functions $\boldsymbol{\psi} \in V_h$,

$$\langle \boldsymbol{\psi}, \mathcal{L}_h(\boldsymbol{u}_h) \rangle := \langle \boldsymbol{\psi}, K_h(\boldsymbol{u}_h) \rangle + \langle \boldsymbol{\psi}, I_h(\boldsymbol{u}_h) \rangle. \tag{3}$$

Here, the element integrals are given by

$$\langle \boldsymbol{\psi}, K_h(\boldsymbol{u}_h) \rangle \quad := \quad \sum_{E \in \mathcal{T}_h} \int_E \Big(\big(F_c(\boldsymbol{u}_h) - F_v(\boldsymbol{u}_h, \nabla \boldsymbol{u}_h) \big) : \nabla \boldsymbol{\psi} + S(\boldsymbol{u}_h) \cdot \boldsymbol{\psi} \Big) \, dx, \tag{4}$$

where : denotes the inner product of two second order tensors. In the code this looks as follows:

```
from ufl import TrialFunction, TestFunction, inner, grad, dx
# trial and test function
u   = TrialFunction(V_h)
psi = TestFunction(V_h)
# element integral from equation (4)
K_h = inner(F_c(u) - F_v(u)*grad(u)), grad(psi)) * dx \  # fluxes
    + inner(S(u), psi) * dx                              # source term
```

The surface integrals are given by

$$\begin{aligned} \langle \boldsymbol{\psi}, I_h(\boldsymbol{u}_h) \rangle \quad := \quad & \sum_{e \in \Gamma_i} \int_e \Big(\big\{\!\big\{ F_v(\boldsymbol{u}_h, [\![\boldsymbol{u}_h]\!]_e)^T : \nabla \boldsymbol{\psi} \big\}\!\big\}_e + \big\{\!\big\{ F_v(\boldsymbol{u}_h, \nabla \boldsymbol{u}_h) \big\}\!\big\}_e : [\![\boldsymbol{\psi}]\!]_e \Big) \, dS \\ & - \sum_{e \in \Gamma} \int_e \Big(H_c(\boldsymbol{u}_h) - H_v(\boldsymbol{u}_h, \nabla \boldsymbol{u}_h) \Big) : [\![\boldsymbol{\psi}]\!]_e \, dS. \end{aligned} \tag{5}$$

The corresponding code reads:

```
from ufl import FacetNormal, FacetArea, CellVolume, avg, jump, dS, ds
# normal and mesh width
n = FacetNormal(V_h)
h_e = avg( CellVolume(V_h) ) / FacetArea(V_h)
# penalty parameter for Symmetric Interior Penalty scheme
from dune.ufl import Constant
eta = Constant( 10*V_h.order**2 if V_h.order > 0 else 1, "penalty" )
# surface integral from equation (5)
I_h = inner(jump(H_c(u), jump(psi)) * dS \  # interior skeleton for convective part
    + H_cb(u)*psi*ds \                      # domain boundary for convective part
    - inner(jump(F_v(u),n),avg(grad(psi)))   * dS \   # symmetry term
    - inner(avg(F_v(u)*grad(u)),jump(psi,n)) * dS \   # consistency term
    + eta/h_e*inner(jump(u, avg(F_v(u))*n),jump(psi,n)) * dS  # penalty term
```

This formulation arises from considering the weak form of the problem: Replace u by $\boldsymbol{u}_h$ in (1), multiply by the test function $\boldsymbol{\psi}$ and integrate the spatial terms by parts. Here,

H_c and H_v are suitable numerical fluxes, imposed at the element interface e. Further, $\{\!\{u\}\!\}_e$ and $[\![u]\!]_e$ denote the average and jump of u over e,

$$\{\!\{u\}\!\}_e := \frac{1}{2}(u_E + u_K) \quad \text{and} \quad [\![u]\!]_e := \boldsymbol{n}_e \cdot (u_E - u_K) \tag{6}$$

where E and K are neighboring elements over intersection e and $\boldsymbol{n}_e$ is outward pointing from element E. Note, for readability $\boldsymbol{n}_e$ is simply called $\boldsymbol{n}$ in the code.

To obtain the DG-SEM we follow [46, 45]. First, we restrict our focus to cuboid meshes and map each $E \in \mathcal{T}_h$ to a reference element using an affine mapping. In the DUNE implementation, the reference element is $[0,1]^d$. This is due to a generic construction of reference element of different shapes in arbitrary dimensions in DUNE; see [20] for details.

In each spatial dimension, a set of $p+1$ Legendre-Gauss-Lobatto (LGL) nodes are introduced and a corresponding set of Lagrange basis polynomials are defined. The discrete solution $\boldsymbol{u}_h(t) \in V_h$ takes the form

$$\boldsymbol{u}_h(t,x) = \sum_i u_i(t)\boldsymbol{\psi}_i(x),$$

where the sum is taken over all tensor product LGL nodes in d dimensions and $\boldsymbol{\psi}_i(x)$ is constructed as the product of Lagrange basis polynomials along each dimension. In practice, this is achieved through the command

```
# create discrete function given a discrete space
u_h = V_h.function(name="u_h")
```

The convective and viscous fluxes are approximated using the interpolation

$$\boldsymbol{F}_h(t,x) \approx \sum_{i=1} F(u_i(t))\boldsymbol{\psi}_i(x),$$

where F is either F_c or F_v.

Finally, the element and surface integrals in (4) and (5) are approximated using Gauss-Lobatto quadrature rules. The collocation of the quadrature with the LGL nodes results in a diagonal positive definite local mass matrix. The choice of a cuboid mesh and a tensor product formulation of the basis functions ensures that the global mass matrix remains diagonal positive definite and is consequently trivially invertible.

The convective numerical flux H_c can be any appropriate numerical flux known for standard finite volume methods. We use the local Lax-Friedrichs (Rusanov) flux function

$${H_c}^{LLF}(\boldsymbol{u}_h)|_e := \{\!\{F_c(\boldsymbol{u}_h)\}\!\}_e + \frac{\lambda_e}{2}[\![\boldsymbol{u}_h]\!]_e \tag{7}$$

where λ_e is an estimate of the maximum wave speed on the interface e. Other options are implemented in DUNE-FEM-DG (cf. [16, 17]) as well.

A wide range of diffusion fluxes H_v can be found in the literature (cf. [9] and references therein), however, of those only the fluxes from the Interior Penalty family can currently be described in UFL due to the missing description and implementation in UFL of lifting terms needed for the other fluxes. For the Interior Penalty method the flux is chosen to be $H_v(u,\nabla u) = \{\!\{\nabla u\}\!\}_e - \frac{\eta}{h_e}\{\!\{F_v(u,\nabla u)\}\!\}_e[\![u]\!]_e$ with η being the penalty parameter.

3.2 Temporal Discretization

After spatial discretization, we obtain a system of ODEs for the coefficient functions $\boldsymbol{u}(t) = (u_1(t), u_2(t), \dots)^\top$, which reads

$$\boldsymbol{u}'(t) = \boldsymbol{F}(t, \boldsymbol{u}(t)), \quad t \in (0, T], \quad \boldsymbol{u}(0) = \boldsymbol{u}_0. \tag{8}$$

Here, $\boldsymbol{F}(t, \boldsymbol{u}(t)) = \boldsymbol{M}^{-1}\mathcal{L}_h(\boldsymbol{u}_h(t))$, where $\mathcal{L}_h$ is defined in (3) and $\boldsymbol{M}$ is the (diagonal) global mass matrix of the DG-SEM discretization. The initial data $\boldsymbol{u}_0$ for (8) is given by the projection of u_0 onto V_h.

Any Runge-Kutta method can in principle be used to solve (8). Explicit methods are easy to implement but suffer from severe time step restrictions for stiff systems.

Consider instead an implicit RK method with Butcher tableau

$$\begin{array}{c|c} \boldsymbol{c} & \boldsymbol{A} \\ \hline & \boldsymbol{b}^\top \end{array}$$

The *stage equations* of the RK method take the form

$$\underline{\boldsymbol{u}} = \boldsymbol{1} \otimes \boldsymbol{u}^n + \Delta t_n (\boldsymbol{A} \otimes \boldsymbol{I}_\xi) \underline{\boldsymbol{F}}, \tag{9}$$

where the vector $\underline{\boldsymbol{u}}^\top = (\boldsymbol{u}^1, \dots, \boldsymbol{u}^{N_\tau})$ contains the N_τ intermediate solution stages and $\underline{\boldsymbol{F}}^\top = (\boldsymbol{F}(t_n + \Delta t_n c_1, \boldsymbol{u}^1), \dots, \boldsymbol{F}(t_n + \Delta t_n c_{N_\tau}, \boldsymbol{u}^{N_\tau}))^\top$. Here, $\boldsymbol{u}^n$ denotes the RK solution in the previous time step. The new solution is given by

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \Delta t_n (\boldsymbol{b}^\top \otimes \boldsymbol{I}_\xi) \underline{\boldsymbol{F}}. \tag{10}$$

Herein we consider a particular family of implicit RK methods, namely Lobatto IIIC [39, 49]. These methods are A-, L- and B-stable and are thus suitable for stiff and nonlinear problems. The order of the N_τ-stage Lobatto IIIC method is $2(N_\tau - 1)$ and the order of the individual stages is $N_\tau - 1$. However, this choice of method is also motivated by its equivalence to a space-time DG-SEM formulation, which is described in the next section. The Butcher tableaus for the 2-, 3- and 4-stage Lobatto IIIC methods are found in Appendix B.

The following code is an example how to use the Lobatto IIIC solvers in Assimulo:

```
# import solver form assimulo
import assimulo.ode as aode
import assimulo.solvers as aso
# import Lobatto IIIC solvers
from Lobatto_IIIC_2s import Lobatto2ODE
from Lobatto_IIIC_3s import Lobatto3ODE
from Lobatto_IIIC_4s import Lobatto4ODE

# set up explicit problem, user-defined rhs
prob = aode.Explicit_Problem(rhs, y0, t0)
# user-defined Jacobian
prob.jac = jacobian
# choose solver
solver = Lobatto2ODE(prob)
# run solver until endTime
t, y = solver.simulate(endTime)
```

4 Space-Time DG-SEM

We now consider DG-SEM applied to (1) with the time variable t treated simply as an additional dimension. The result is space-time DG-SEM.

Defining the gradient $\underline{\nabla} := \left(\nabla, \frac{\partial}{\partial t}\right)$ and the new convective and viscous fluxes

$$\underline{F}_c = \begin{bmatrix} F_c & u \end{bmatrix}, \quad \underline{F}_v = \begin{bmatrix} F_v & 0 \end{bmatrix},$$

we can rewrite (1) as a $d+1$-dimensional problem over the space-time domain $\underline{\Omega} := \Omega \times (0,T) \subset \mathbb{R}^{d+1}$ as

$$\underline{\nabla} \cdot \left(\underline{F}_c(u) - \underline{F}_v(u, \nabla u)\right) = S(u) \quad \text{in } \underline{\Omega}. \tag{11}$$

Given a tessellation $\underline{\mathcal{T}_h}$ of $\underline{\Omega}$ we introduce the piecewise polynomial space

$$\underline{V_h} = \{\underline{\boldsymbol{v}} \in L^2(\underline{\Omega}, \mathbb{R}^r) : \underline{\boldsymbol{v}}|_E \in [\mathcal{P}_p(E)]^r,\ E \in \underline{\mathcal{T}_h}\}, \quad p \in \mathbb{N}. \tag{12}$$

Then the space-time DG-SEM discretization of (11) follows analogously to (4) and (5):

$$\langle \underline{\boldsymbol{\psi}}, \underline{\mathcal{L}}_h(\underline{\boldsymbol{u}}_h)\rangle := \langle \underline{\boldsymbol{\psi}}, \underline{K}_h(\underline{\boldsymbol{u}}_h)\rangle + \langle \underline{\boldsymbol{\psi}}, \underline{I}_h(\underline{\boldsymbol{u}}_h)\rangle, \tag{13}$$

with the element integrals

$$\langle \underline{\boldsymbol{\psi}}, \underline{K}_h(\underline{\boldsymbol{u}}_h)\rangle \quad := \quad \sum_{E \in \underline{\mathcal{T}_h}} \int_E \left(\left(\underline{F}_c(\underline{\boldsymbol{u}}_h) - \underline{F}_v(\underline{\boldsymbol{u}}_h, \underline{\nabla}\underline{\boldsymbol{u}}_h)\right) : \nabla \underline{\boldsymbol{\psi}} + S(\underline{\boldsymbol{u}}_h) \cdot \underline{\boldsymbol{\psi}} \right) dx, \tag{14}$$

and the surface integrals

$$\begin{aligned} \langle \underline{\boldsymbol{\psi}}, \underline{I}_h(\boldsymbol{u}_h)\rangle \quad := \quad & \sum_{e \in \underline{\Gamma}_i} \int_e \left(\left\{\!\!\left\{ \underline{F}_v(\underline{\boldsymbol{u}}_h, [\![\underline{\boldsymbol{u}}_h]\!]_e)^\top : \nabla \underline{\boldsymbol{\psi}} \right\}\!\!\right\}_e + \left\{\!\!\left\{ \underline{F}_v(\underline{\boldsymbol{u}}_h, \underline{\nabla}\underline{\boldsymbol{u}}_h) \right\}\!\!\right\}_e : [\![\underline{\boldsymbol{\psi}}]\!]_e \right) d\underline{S} \\ & - \sum_{e \in \underline{\Gamma}} \int_e \left(\underline{H}_c(\boldsymbol{u}_h) - \underline{H}_v(\underline{\boldsymbol{u}}_h, \underline{\nabla}\underline{\boldsymbol{u}}_h) \right) : [\![\underline{\boldsymbol{\psi}}]\!]_e \, dS. \end{aligned} \tag{15}$$

Here, $\underline{\Gamma}_i$ and $\underline{\Gamma}$ have analogous meanings to their spatial counterparts Γ_i and Γ. The numerical fluxes are given by

$$\underline{H}_c = \begin{bmatrix} H_c & u^* \end{bmatrix}, \quad \underline{H}_v = \begin{bmatrix} H_v & 0 \end{bmatrix},$$

where u^* is a simple upwind flux in time.

In our considered framework, (13) can be implemented quite nicely by increasing the dimension and applying the above discussed modifications[2].

```python
d = 2 # 1,2,3 is the spatial dimension
from dune.grid import cartesianDomain, structuredGrid as leafGrid
t_end, timeSteps = 1.0, 10
dt = t_end / timeSteps
# create grid that tessellates [0,1]^d × [0,Δt] with 10 elements in spatial directions and 1 element in time
T_h = leafGrid(cartesianDomain([0]*d + [0], [1]*d + [dt], [10]*d + [1])) # create a space-time grid
p = 3 # polynomial degree
# create DG space with Lagrange basis and Gauss-Lobatto interpolation points
V_h = dglagrangelobatto( T_h, order=p )
```

[2]Note that for the 4D version ($3d + time$) a UFL patch (see Appendix C) was added to introduce the 4D reference elements to UFL code.

```python

def appendTime( F, u ):
  return ufl.as_tensor([ *[[F[k,i] if i<d else u[k] for i in range(d+1)] for k in range(len(u))] ])

def F_c( u ):
  from molspacediscr import F_c # import F_c used in MOL discretization
  F_spc = F_c(u) # compute spatial fluxes
  # append time derivative as last column
  return appendTime( F_spc, u )

def F_v( u ):
  from molspacediscr import F_v # import F_v used in MOL discretization
  F_spc = F_v(u) # compute spatial fluxes
  # append column of zeros since there is no diffusion in time
  return appendTime( F_spc, [0.]*len(u) )

# trial and test function
u   = TrialFunction(V_h)
psi = TestFunction(V_h)
# element integral from equation (14)
K_h = inner(F_c(u) - F_v(u)*grad(u)), grad(psi)) * dx \  # fluxes
    + inner(S(u), psi) * dx                              # source term

# normal and mesh width
n    = FacetNormal(V_h)
h_e  = avg( CellVolume(V_h) ) / FacetArea(V_h)
# penalty parameter for Symmetric Interior Penalty scheme
eta  = Constant( 10*V_h.order**2 if V_h.order > 0 else 1, "penalty" )
# surface integral from equation (15)
I_h = inner(jump(H_c(u), jump(psi)) * dS \  # interior skeleton for convective part
    + H_cb(u)*psi*ds \                      # domain boundary for convective part
    - inner(jump(F_v(u),n),avg(grad(psi)))   * dS \   # symmetry term
    - inner(avg(F_v(u)*grad(u)),jump(psi,n)) * dS \   # consistency term
    + eta/h_e*inner(jump(u, avg(F_v(u))*n),jump(psi,n)) * dS  # penalty term
```

Remark 4.1. It is of practical interest to generalize the space $\underline{V_h}$ so that the time dimension may be discretized by polynomials of a different order than the spatial dimensions. We will henceforth refer to the number of temporal nodes in each element as N_τ so that the polynomial degree in time is $N_\tau - 1$. This notation contrasts standard DG terminology, where nodes are typically indexed from 0 to p. Additionally, note that this is the same notation used for the number of stages of the Lobatto IIIC method in Section 3. Stages are typically indexed from 1 to s. However, to minimize the use of notation and to make the connection between the two viewpoints clearer, we write N_τ to count the degrees of freedom within a time element, whether this pertains to the DG or RK interpretation.

After space-time discretization, the discrete solution $\underline{\boldsymbol{u}}_h \in \underline{V_h}$ takes the form $\underline{\boldsymbol{u}}_h(t,x) = \sum_{i,n} u_i^n \boldsymbol{\psi}_i(x)\psi_n(t)$. Here, the sum is taken over all tensor product LGL nodes in $d+1$ dimensions. The vector of coefficients is now given by

$$\underline{\boldsymbol{u}} = (\boldsymbol{u}^1, \ldots, \boldsymbol{u}^{N_\tau})^\top, \tag{16}$$

where $\boldsymbol{u}^i$ contains all the spatial unknowns in the ith time element.

The space-time discretization (13) can alternatively be derived by starting from (8) and discretizing in time with DG-SEM. Multiplying (8) by a test function $\psi(t)$ and integrating over the nth time element results in

$$\int_{t_n}^{t_{n+1}} \boldsymbol{u}_t \psi \mathrm{d}t = \int_{t_n}^{t_{n+1}} \boldsymbol{F}(t, \boldsymbol{u}(t))\psi \mathrm{d}t.$$

We transform this equation to the reference element $[-1,1]$ using the mapping $t = t_n + \frac{\Delta t_n}{2}(1+\tau)$, where $\Delta t_n = t_{n+1} - t_n$. After integration by parts the resulting equation reads

$$[\boldsymbol{u}\psi]_{-1}^{1} - \int_{-1}^{1} \boldsymbol{u}\psi_\tau \mathrm{d}\tau = \frac{2}{\Delta t_n}\int_{-1}^{1} \boldsymbol{F}(\tau, \boldsymbol{u}(\tau))\psi \mathrm{d}\tau.$$

We now follow the steps of DG-SEM, i.e. approximating $\boldsymbol{u}$ and $\boldsymbol{F}$ by interpolants

$$\boldsymbol{u} \approx \sum_{j=1}^{N_\tau} \boldsymbol{u}^j \psi_j(\tau),$$
$$\boldsymbol{F} \approx \sum_{j=1}^{N_\tau} \boldsymbol{F}^j \psi_j(\tau),$$

and the integrals by Gauss-Lobatto quadrature with nodes τ_j and weights ω_j. Using the cardinal property of the Lagrange basis polynomials ψ, the resulting DG-SEM discretization becomes

$$\delta_{iN_\tau}\boldsymbol{u}^* - \delta_{i1}\boldsymbol{u}^* - \sum_{j=1}^{N_\tau} \omega_j \boldsymbol{u}^j \left.\frac{\mathrm{d}\psi_i}{\mathrm{d}\tau}\right|_{\tau_j} = \frac{2}{\Delta t_n}\omega_i \boldsymbol{F}^i, \quad i = 1, \ldots, N_\tau. \tag{17}$$

Here, we have replaced the boundary terms with numerical fluxes $\boldsymbol{u}^*$. With DG-SEM in time, the numerical flux $\underline{\boldsymbol{u}}^*$ is always chosen as the upwind flux

$$\underline{\boldsymbol{u}}^* = (\boldsymbol{u}^n, \boldsymbol{0}, \ldots, \boldsymbol{0}, \boldsymbol{u}^{N_\tau})^\top, \tag{18}$$

where $\boldsymbol{u}^n$ is the numerical solution from the previous time element. This choice leads to an entropy stable numerical scheme if the spatial terms are handled appropriately [26]. It also has the advantage of decoupling the temporal elements. Thus, (19) can be solved as a stand-alone nonlinear system on the nth time element.

Defining the boundary, mass and differentiation matrices

$$\boldsymbol{B}_\tau = \mathrm{diag}([-1, 0, \ldots, 0, 1]) \in \mathbb{R}^{N_\tau \times N_\tau},$$
$$\boldsymbol{M}_\tau = \mathrm{diag}([\omega_1, \ldots, \omega_{N_\tau}]) \in \mathbb{R}^{N_\tau \times N_\tau},$$
$$(\boldsymbol{D}_\tau)_{ji} = \left.\frac{\mathrm{d}\psi_i}{\mathrm{d}\tau}\right|_{\tau_j} \in \mathbb{R}^{N_\tau \times N_\tau},$$

we can write (17) in matrix form on each reference element as

$$(\boldsymbol{B}_\tau \otimes \boldsymbol{I}_\xi)\,\underline{\boldsymbol{u}}^* - (\boldsymbol{D}_\tau^\top \boldsymbol{M}_\tau \otimes \boldsymbol{I}_\xi)\,\underline{\boldsymbol{u}} = \frac{\Delta t_n}{2}(\boldsymbol{M}_\tau \otimes \boldsymbol{I}_\xi)\underline{\boldsymbol{F}}(\underline{\boldsymbol{u}}), \tag{19}$$

where $\boldsymbol{M}_\tau$ is the local temporal mass matrix and $\boldsymbol{M}_\tau \boldsymbol{D}_\tau$ defines the corresponding stiffness matrix. Here, $\underline{\boldsymbol{F}}^\top(\underline{\boldsymbol{u}}) = (\boldsymbol{F}^\top(t_n + \frac{\Delta t_n}{2}(1+\tau_1), \boldsymbol{u}^1), \ldots, \boldsymbol{F}^\top(t_n + \frac{\Delta t_n}{2}(1+\tau_{N_\tau}), \boldsymbol{u}^{N_\tau}))$, where $\underline{\boldsymbol{u}}$ is given by (16) and τ_k is the kth LGL node; see [30] for details. The operation $\otimes$ denotes the Kronecker product and $\boldsymbol{I}_\xi$ is the identity matrix whose dimension is given by the number of spatial nodes.

We finish this section by remarking that while (13) describes the global space-time DG-SEM discretization, the alternative formulation (19) pertains to a single time element.

5 Theoretical Aspects of Space-Time DG-SEM

In this section we discuss important properties of the space-time DG-SEM, in particular the equivalence of the temporal discretization and the Lobatto IIIC family of Runge-Kutta methods. To make the connection between DG-SEM and Runge-Kutta methods clear in the following sections, we consider the solution at the final point in the time element, i.e.

$$\boldsymbol{u}^{N_\tau} \equiv (\boldsymbol{e}_{N_\tau}^\top \otimes \boldsymbol{I}_\xi)\underline{\boldsymbol{u}}, \tag{20}$$

where $\boldsymbol{e}_{N_\tau}^\top = (0, \dots, 0, 1) \in \mathbb{R}^{N_\tau}$. Following [8], we set out to show that $\boldsymbol{u}^{N_\tau} = \boldsymbol{u}^{n+1}$, where $\boldsymbol{u}^{n+1}$ is the numerical solution arising from the Lobatto IIIC method in (10). To achieve this, we assume that this equality holds in the previous (i.e. in the $(n-1)$st) time element and show that it then also holds in the current (i.e. in the nth) element. In the sequel we will also make use of the vector $\boldsymbol{e}_1^\top = (1, 0, \dots, 0) \in \mathbb{R}^{N_\tau}$.

The DG-SEM discretization (19) constitutes a so called *Summation-By-Parts* (SBP) method [30], meaning that the following conditions are satisfied:

$$\boldsymbol{M}_\tau = \boldsymbol{M}_\tau^\top > \boldsymbol{0}, \quad \boldsymbol{M}_\tau \boldsymbol{D}_\tau + (\boldsymbol{M}_\tau \boldsymbol{D}_\tau)^\top = \boldsymbol{B}_\tau. \tag{21}$$

The SBP property (21) is at the heart of the connection of DG-SEM in time to implicit Runge-Kutta methods, which is detailed in the following section.

5.1 DG-SEM and Lobatto IIIC

SBP methods were historically developed to be used as spatial discretizations [48, 66]. For an overview of these techniques, see [68, 22]. In recent years, their use as time stepping schemes has been explored [58] and connections to implicit Runge-Kutta methods have been discovered [8]. Here we summarize the steps showing that (19) can be reformulated as an implicit RK method applied to the system of ODEs (8).

We begin by using the SBP property (21) in the second term of (19) and then multiplying by $(\boldsymbol{M}_\tau^{-1} \otimes \boldsymbol{I}_\xi)$ to obtain the so called *strong form*,

$$(\boldsymbol{D}_\tau \otimes \boldsymbol{I}_\xi)\underline{\boldsymbol{u}} = (\boldsymbol{M}_\tau^{-1}\boldsymbol{B}_\tau \otimes \boldsymbol{I}_\xi)(\underline{\boldsymbol{u}} - \underline{\boldsymbol{u}}^*) + \frac{\Delta t_n}{2}\underline{\boldsymbol{F}}. \tag{22}$$

Note that we may write $\boldsymbol{B}_\tau = \boldsymbol{e}_{N_\tau}\boldsymbol{e}_{N_\tau}^\top - \boldsymbol{e}_1\boldsymbol{e}_1^\top$ and that $(\boldsymbol{e}_{N_\tau}^\top \otimes \boldsymbol{I}_\xi)(\underline{\boldsymbol{u}} - \underline{\boldsymbol{u}}^*) = (\boldsymbol{u}^{N_\tau} - \boldsymbol{u}^{N_\tau}) = \boldsymbol{0}$. Using (18), the second term in (22) can therefore be expressed as

$$(\boldsymbol{M}_\tau^{-1}\boldsymbol{B}_\tau \otimes \boldsymbol{I}_\xi)(\underline{\boldsymbol{u}} - \underline{\boldsymbol{u}}^*) = -(\boldsymbol{M}_\tau^{-1} \otimes \boldsymbol{I}_\xi)[(\boldsymbol{e}_1\boldsymbol{e}_1^\top \otimes \boldsymbol{I}_\xi)\underline{\boldsymbol{u}} - (\boldsymbol{e}_1 \otimes \boldsymbol{u}^n)].$$

Grouping together terms that multiply the solution $\underline{\boldsymbol{u}}$, we may rewrite (22) as

$$((\boldsymbol{D}_\tau + \boldsymbol{M}_\tau^{-1}\boldsymbol{e}_1\boldsymbol{e}_1^\top) \otimes \boldsymbol{I}_\xi)\underline{\boldsymbol{u}} = (\boldsymbol{M}_\tau^{-1}\boldsymbol{e}_1 \otimes \boldsymbol{u}^n) + \frac{\Delta t_n}{2}\underline{\boldsymbol{F}}. \tag{23}$$

Next, we multiply (23) by $((\boldsymbol{D}_\tau + \boldsymbol{M}_\tau^{-1}\boldsymbol{e}_1\boldsymbol{e}_1^\top) \otimes \boldsymbol{I}_\xi)^{-1}$. Upon doing this, first note that

$$(\boldsymbol{D}_\tau + \boldsymbol{M}_\tau^{-1}\boldsymbol{e}_1\boldsymbol{e}_1^\top)^{-1}\boldsymbol{M}_\tau^{-1}\boldsymbol{e}_1 = \boldsymbol{1} := (1, \dots, 1)^\top \in \mathbb{R}^{N_\tau},$$

which follows from observing that $(\boldsymbol{D}_\tau + \boldsymbol{M}_\tau^{-1}\boldsymbol{e}_1\boldsymbol{e}_1^\top)\mathbf{1} = \boldsymbol{M}_\tau^{-1}\boldsymbol{e}_1$ since $\boldsymbol{D}_\tau\mathbf{1} = \mathbf{0}$ by consistency. Thus, the following system arises:

$$\begin{aligned}\underline{\boldsymbol{u}} &= \mathbf{1}\otimes\boldsymbol{u}^n + \Delta t_n\frac{1}{2}((\boldsymbol{D}_\tau + \boldsymbol{M}_\tau^{-1}\boldsymbol{e}_1\boldsymbol{e}_1^\top)\otimes\boldsymbol{I}_\xi)^{-1}\underline{\boldsymbol{F}}\\ &= \mathbf{1}\otimes\boldsymbol{u}^n + \Delta t_n\frac{1}{2}((\boldsymbol{D}_\tau + \boldsymbol{M}_\tau^{-1}\boldsymbol{e}_1\boldsymbol{e}_1^\top)^{-1}\otimes\boldsymbol{I}_\xi)\underline{\boldsymbol{F}}.\end{aligned} \tag{24}$$

The equation system (24) should be compared with the stage equations (9) that arose from the MOL discretization using implicit RK. We see that the temporal DG-SEM discretization defines an RK method with coefficient matrix $\boldsymbol{A} = \frac{1}{2}(\boldsymbol{D}_\tau + \boldsymbol{M}_\tau^{-1}\boldsymbol{e}_1\boldsymbol{e}_1^\top)^{-1}$ and nodes $\boldsymbol{c} = (\mathbf{1}+\boldsymbol{\tau})/2$, where $\boldsymbol{\tau} = (\tau_1,\dots,\tau_{N_\tau})^\top$ is the vector of LGL nodes. Further, the vector $\underline{\boldsymbol{u}}$, which in the DG-SEM context contains the interpolation coefficients u_i^n, has adopted the role of the stages of the RK method.

To complete the transition from DG-SEM to RK, we compute the numerical solution at the final time node, $\boldsymbol{u}^{N_\tau} = (\boldsymbol{e}_{N_\tau}^\top\otimes\boldsymbol{I}_\xi)\underline{\boldsymbol{u}}$. To this end we observe that the SBP property (21) gives the relation

$$\mathbf{1}^\top\boldsymbol{M}_\tau(\boldsymbol{D}_\tau + \boldsymbol{M}_\tau^{-1}\boldsymbol{e}_1\boldsymbol{e}_1^\top) = \mathbf{1}^\top(\boldsymbol{e}_{N_\tau}\boldsymbol{e}_{N_\tau}^\top - \boldsymbol{D}_\tau^\top\boldsymbol{M}_\tau) = \boldsymbol{e}_{N_\tau}^\top,$$

so that

$$\boldsymbol{e}_{N_\tau}^\top(\boldsymbol{D}_\tau + \boldsymbol{M}_\tau^{-1}\boldsymbol{e}_1\boldsymbol{e}_1^\top)^{-1} = \mathbf{1}^\top\boldsymbol{M}_\tau.$$

Consequently, multiplying (24) by $(\boldsymbol{e}_{N_\tau}^\top\otimes\boldsymbol{I}_\xi)\underline{\boldsymbol{u}}$ yields

$$\boldsymbol{u}^{N_\tau} = \boldsymbol{u}^n + \Delta t_n\frac{1}{2}(\mathbf{1}^\top\boldsymbol{M}_\tau\otimes\boldsymbol{I}_\xi)\underline{\boldsymbol{F}}. \tag{25}$$

Comparing (25) with the solution (10) of the implicit RK method, we see that the vector $\boldsymbol{b}$ in the Butcher tableau is related to the DG-SEM discretization by $\boldsymbol{b}^\top = \mathbf{1}^\top\boldsymbol{M}_\tau/2$, and that the RK solution is simply the N_τth component of the DG solution $\underline{\boldsymbol{u}}$.

To summarize, a DG-SEM time discretization is equivalent to an implicit RK method whose Butcher tableau is defined in terms of the DG method as

$$\boldsymbol{A} = \frac{1}{2}(\boldsymbol{D}_\tau + \boldsymbol{M}_\tau^{-1}\boldsymbol{e}_1\boldsymbol{e}_1^\top)^{-1},\quad \boldsymbol{b} = \frac{1}{2}\boldsymbol{M}_\tau\mathbf{1},\quad \boldsymbol{c} = \frac{\mathbf{1}+\boldsymbol{\tau}}{2}. \tag{26}$$

The two methods yield two different nonlinear systems; for DG-SEM and RK they are respectively given by

$$(\boldsymbol{B}_\tau\otimes\boldsymbol{I}_\xi)\,\underline{\boldsymbol{u}}^* - (\boldsymbol{D}_\tau^\top\boldsymbol{M}_\tau\otimes\boldsymbol{I}_\xi)\,\underline{\boldsymbol{u}} = \frac{\Delta t_n}{2}(\boldsymbol{M}_\tau\otimes\boldsymbol{I}_\xi)\underline{\boldsymbol{F}}, \tag{27a}$$

$$\underline{\boldsymbol{u}} = \mathbf{1}\otimes\boldsymbol{u}^n + \Delta t_n(\boldsymbol{A}\otimes\boldsymbol{I}_\xi)\underline{\boldsymbol{F}}. \tag{27b}$$

These systems have the same solution $\underline{\boldsymbol{u}}$ since we can transition from (27a) to (27b) in a series of algebraic steps. More precisely, the connection is made by rewriting (27a) in strong form, then multiplying by $(\boldsymbol{D}_\tau + \boldsymbol{M}_\tau^{-1}\boldsymbol{e}_1\boldsymbol{e}_1^\top\otimes\boldsymbol{I}_\xi)^{-1}$.

Note that the latter step demands that $\boldsymbol{D}_\tau + \boldsymbol{M}_\tau^{-1}\boldsymbol{e}_1\boldsymbol{e}_1^\top$ is invertible. This is the case if and only if $\boldsymbol{D}_\tau$ is *null-space consistent*, i.e. if $\ker(\boldsymbol{D}_\tau) = \operatorname{span}(\mathbf{1})$ [51]. This is known to hold for all $N_\tau > 1$ [63, 50].

Finally, the Butcher tableau formed from (26) coincides with that of the Lobatto IIIC family of implicit Runge-Kutta methods. This follows from the use of LGL nodes

and quadrature weights, together with a set of accuracy conditions satisfied by the two formulations [61]. We will detail these in the next section. The derivation above therefore shows that DG-SEM in time and the Lobatto IIIC methods are mathematically equivalent, and that we in fact have $\boldsymbol{u}^{N_\tau} = \boldsymbol{u}^{n+1}$. The coefficients for the DG-SEM matrices with $N_\tau \in \{2, 3, 4\}$ are listed in Appendix D.

5.2 Comparison of terminology

While DG-SEM in time and Lobatto IIIC are algebraically equivalent methods, they have been developed in different research communities and disparities have consequently arisen in terms of the terminology used to describe these methods. This pertains in particular to the notions of order and stability.

Beginning with RK methods, we take as our starting point the system of ODEs (8). The (classical) notion of order is defined as follows:

Definition 5.1. A Runge-Kutta method is of *order* p if the estimate

$$\|\boldsymbol{u}^{n+1} - \boldsymbol{u}(t_{n+1})\| \leq K\Delta t^{p+1}$$

holds with some constant K independent of Δt, whenever problem (8) is sufficiently smooth.

In Definition 5.1, we may take $\Delta t = \max_n \Delta t_n$. The norm can be any vector norm.

The classical order of RK methods is determined by certain order conditions. To make the connection with DG-SEM as clear as possible, we present here a set of simplified conditions that are sufficient for the method to be of order p [10]:

Theorem 5.2. Suppose that an implicit Runge-Kutta method satisfies the conditions

$B(p_B)$**:** $\boldsymbol{b}^\top \boldsymbol{c}^{j-1} = \frac{1}{j}, \quad j = 1, \ldots, p_B,$

$C(p_C)$**:** $\boldsymbol{A}\boldsymbol{c}^{j-1} = \frac{\boldsymbol{c}^j}{j}, \quad j = 1, \ldots, p_C,$

$D(p_D)$**:** $\boldsymbol{A}^\top \mathrm{diag}(\boldsymbol{b})\boldsymbol{c}^{j-1} = \frac{1}{j}\mathrm{diag}(\boldsymbol{b})(\boldsymbol{1} - \boldsymbol{c}^j), \quad j = 1, \ldots, p_D,$

where $p_B \leq 2(p_C + 1)$ and $p_B \leq p_C + p_D + 1$. Then the method is of order $p = p_B$.

The conditions $C(p_C)$ play a particularly important role in the context of stiff problems and have its own moniker:

Definition 5.3. A Runge-Kutta method that satisfies the order conditions $C(p_C)$ is said to have *stage order* p_C.

The stage order of the RK method describes the accuracy with which the intermediate stages are approximated. We will delve into the meanings of the conditions B, C and D shortly. However, first we summarize the various order concepts for Lobatto IIIC; see [33, Chapter IV.5].

Theorem 5.4. The Lobatto IIIC method with N_τ stages satisfies $B(2N_\tau - 2)$, $C(N_\tau - 1)$ and $D(N_\tau - 1)$. Consequently it has stage order $N_\tau - 1$ and is of order $2N_\tau - 2$.

We momentarily leave the RK viewpoint and focus on DG methods. DG-SEM was developed for spatial discretizations of time-dependent PDEs. Thus, errors are measured in an L_2 norm over a spatial domain. This norm of a discrete solution $\boldsymbol{u}_h$ can be computed via the quadrature rule exactly:

$$\|\boldsymbol{u}_h\|_{L_2(\Omega)} = \sum_i \left(\boldsymbol{u}_{h_i}^\top \boldsymbol{M}_{\xi_i} \boldsymbol{u}_{h_i}\right)^{\frac{1}{2}}. \tag{28}$$

The sum is taken over all elements and $\boldsymbol{M}_{\xi_i}$ is the local spatial mass matrix on element i. Assuming vanishingly small errors from the time discretization, the order of convergence measured in this norm is typically $p+1$ or $p+\frac{1}{2}$, depending on the nature of problem (1), the choice of numerical fluxes, and sometimes on whether p is odd or even [36, 73].

Conversely, when using DG-SEM as a time integration method, one works in the space $L_2([0,T])$ with a corresponding discrete norm. With an upwind flux in time, for sufficiently smooth and nonstiff problems, the order of convergence in this norm is N_τ [52]. This order is much smaller than the one of the Lobatto IIIC method, which requires some discussion.

There are several other order concepts in the DG literature. Here we follow [8] and relate these to the corresponding concepts in the RK framework.

- The *order of the operator* is the highest degree q for which $\boldsymbol{D}_\tau \boldsymbol{\tau}^q = q\boldsymbol{\tau}^{q-1}$. The exponentiation should be interpreted elementwise, and we take $\boldsymbol{\tau}^0 = \mathbf{1}$ as a definition. For DG-SEM we have $q = N_\tau - 1$.

  Multiplying $C(p_C)$ by $\boldsymbol{A}^{-1}$ as given in (26) and utilizing the fact that the first element in $\boldsymbol{c}$ is zero, we see that the RK order condition $C(p_C)$ actually describes precisely the order of the operator $\boldsymbol{D}_\tau$. A transformation of the reference element to $[0,1]$ is necessary in this step. In other words, the order of the operator is a concept identical to the stage order of the corresponding Lobatto IIIC method.

- The *order of the norm/quadrature/mass matrix* is the highest degree m such that $(m+1)\mathbf{1}^\top \boldsymbol{M}_\tau \boldsymbol{\tau}^m = 1-(-1)^{m+1}$, i.e. for which $\boldsymbol{M}_\tau$ exactly integrates polynomials. For DG-SEM, $\mathbf{1}^\top \boldsymbol{M}_\tau$ is a row vector with the N_τ weights of the Gauss-Lobatto quadrature rule and we consequently have $m = 2(N_\tau - 1)$.

  Using (26) we note that the condition $B(p_B)$ simply describes the order of the quadrature, although applied to $\boldsymbol{c}$ rather than $\boldsymbol{\tau}$. Again, this amounts to a transformation from $\tau \in [-1,1]$ to $[0,1]$.

- Pertinently, it turns out that the order of accuracy of the final component $\boldsymbol{u}^{N_\tau} \equiv \boldsymbol{u}^{n+1}$ is $2(N_\tau - 1)$ [52], at least for smooth nonstiff problems. This *superconvergence* can be proven using the theory of dual consistent SBP methods [37]. Here it suffices to say that it is a consequence of the order of the quadrature and choosing the upwind numerical flux (18).

  The superconvergence result pertaining to DG-SEM corresponds to the classical order of Lobatto IIIC as introduced in Definition 5.1. Note that this is a consequence of considering the pointwise error in time rather than $\|\cdot\|_{L_2([0,T])}$.

To the best of our knowledge, conditions $D(p_D)$ have no clear interpretation in the language of DG. Nevertheless, using the SBP property (21) and the diagonality of $\boldsymbol{M}_\tau$ it

is shown in [8] that $C(p_C)$ is satisfied with $p_C = N_\tau - 1$, which is consistent with Theorem 5.4.

The convergence thery for RK methods rely on certain regularity properties of the problem being solved. In particular, they assume that the right-hand side of the system of ODEs (8) satisfies a one-sided Lipschitz condition,

$$\langle \boldsymbol{u} - \boldsymbol{v}, \boldsymbol{F}(t, \boldsymbol{u}) - \boldsymbol{F}(t, \boldsymbol{v}) \rangle \leq \beta \|\boldsymbol{u} - \boldsymbol{v}\|^2, \tag{29}$$

where $\langle \cdot, \cdot \rangle$ denotes some inner product and $\| \cdot \|$ the corresponding norm. If $\beta \leq 0$, the problem is *contractive*. DG-SEM in time, and hence Lobatto IIIC, is stable for contractive problems, i.e. they are B-stable methods. Convergence for contractive problems is correspondingly known as B-convergence. B-convergence can be shown for Lobatto IIIC if $\beta < 0$, but in general not if $\beta = 0$ if $N_\tau > 2$; see [64] for details.

Regularity of the type (29) is not standard in the literature on spatial discretization using high order DG methods. Rather, estimates of the form $\langle \boldsymbol{u}, \boldsymbol{F}(t, \boldsymbol{u}) \rangle \leq 0$ are common. Such discretizations are referred to as *semi-bounded*, or in Runge-Kutta parlance, as *monotonic*. They arise particularly for discretizations of linear, homogeneous hyperbolic or parabolic problems, but also e.g. for the velocity components of the incompressible Navier-Stokes equations [57].

If $\boldsymbol{F}(t, \boldsymbol{0}) = \boldsymbol{0}$, then semi-boundedness is a special case of contractivity and results on B-stability and B-convergence apply. However, for e.g. the equations of compressible flow, semi-boundedness must typically be replaced by *entropy stability*, i.e. regularity of the form $\langle \eta'(\boldsymbol{u}), \boldsymbol{F}(t, \boldsymbol{u}) \rangle \leq 0$. Here, η is some convex function of $\boldsymbol{u}$ referred to as an entropy [24, 23]. A convergence theory for implicit RK methods applied to entropy stable problems is desirable but currently appears to be missing from the literature.

6 Practical Aspects of Space-Time DG-SEM

In this section we discuss two archetypal implementations: On the one hand, the method of lines approach with Lobatto IIIC as discussed in Section 3 and on the other hand the space-time DG approach, as discussed in Section 4. We will refer to these as LoDG and STDG, respectively.

It is of course possible to produce a code that uses elements from both the LoDG and STDG formulation and thereby falls somewhere in between these approaches. However, here we adopt the point of view of a user who seeks to use an already available code base rather than producing a brand new solver.

Even though STDG and LoDG are mathematically equivalent methods, their respective implementations differ in several key aspects, each with particular requirements and accompanying merits. Here we will outline several such differences, and the choices a user will inevitable face when deciding on which implementation to select.

Several multi-dimensional DG-SEM solvers exist, such as Nektar++ [41], Fluxo, Flexi and the latest iteration Trixi [62] and others. In particular, this approach is popular for weather and climate prediction and has been used e.g. in NUMA [53] and HOMAM [54]. Thus, in the following discussion we assume that the user has access to a multi-dimensional DG solver for spatial discretization.

Our work here is based on the solver from the Dune-Fem framework, hence the challenges outlined below are flavoured by this choice. Depending on the software framework at hand, a user may find that one approach is easier to implement than the other.

6.1 STDG

As described in Section 4, the defining feature of STDG is the treatment of the d-dimensional time-dependent problem as a $d+1$-dimensional stationary problem.

Requirements: The problem description in the code must be extended to a $d+1$-dimensional stationary PDE, which requires the software to be able to handle such problems. In particular, this includes the assembly of mass and stiffness matrices as well as having access to appropriate solvers for the resulting nonlinear system.

This new stationary PDE requires the use of different numerical fluxes in space and time; the temporal direction follows a causality principle, enforced by the upwind flux, which may not be the best choice for the spatial directions.

It is desirable to be able to choose different orders of accuracy in space and time, which then needs to be made possible in the DG code. In DUNE this is implemented for certain DG spaces [32] but not yet available for the Lagrange basis used in this work.

The numerical solution should be accessible at specific time steps in order for the user to effectively visualize intermediate results and the final solution. For $d = 3$, this includes extracting 3-dimensional slices from 4-dimensional data sets. Related to this issue is the problem of 4-dimensional mesh generation. An example of how to handle this for the STDG ansatz in DUNE-FEM is found in Section 4.

Merits: If the requirements above are met, then existing software can be reused to solve the problem, which implies full control over the code. Moreover, only one code is needed. This code closely follows the mathematical derivation of the space-time method and may therefore be more intuitive than alternatives.

Due to the relatively simple adaption of an existing code for spatial problems of dimension $d < 3$, the STDG approach is fast for preliminary testing. An existing DG code is most likely optimized for computational resources and might even allow for parallelization in time by solving for several time steps at once.

In summary, this technique allows re-usability and full control over the code.

6.2 LoDG

The defining feature of LoDG is the method of lines approach outlined in Section 3. In this DG-SEM solver, each time step is solved individually.

Requirements: The (spatial) DG-SEM code needs to be coupled with an ODE solver with an implementation of a Lobatto IIIC method, most likely coming from another code. Difficulties may arise from the particular requirements of the two codes, such as interfaces for time and space adaptivity, parallelization etc.

The ODE solver might require input in a specific format not native to the DG code. Further, an efficient solution procedure may require information from the DG-SEM solver not directly available, such as the Jacobian of the spatial discretization.

An example of a Lobatto IIIC solver implemented in ASSIMULO [2] is found at the end of Section 3.

Merits: No adaption of the DG code with respect to the PDE or its dimension is necessary. There are no additional difficulies arising in the treatment of 3-dimensional problems, and the resulting solution can be visualized in a straightforward way.

Most existing ODE solvers are optimized and equipped with several options, for instance adaptive time stepping. Intermediate results are easy to access and the order of accuracy in space and time can be chosen independently.

In summary, this technique provides flexibility and allows reuse of existing simulation workflows.

6.3 Algorithmic Aspects

In the following we suppose that we have overcome the most important challenges of the two approaches presented in the previous subsections. Thus we now have access to

(a) a code that generates a d-dimensional spatial DG-SEM discretization by following the steps in Section 3, and a code for time marching using Lobatto IIIC (LoDG),

(b) a code that generates a $d+1$-dimensional space-time DG-SEM discretization by following the steps in Section 4 (STDG).

In each time step, the LoDG code (approximately) solves (27b) while the STDG code solves (27a), or equivalently solves $\underline{\mathcal{L}}_h(\underline{\boldsymbol{u}}_h) = 0$ from (13). Depending on the nature of the problem (1), solving these systems accurately and efficiently will require a variety of algorithmic capabilities, some of which are likely to be more readily available in one code than the other. A selection of such capabilities is discussed in the following.

Accessing time steps and stages: Accessing the numerical solution at a particular time is straightforward in most ODE solvers using Runge-Kutta methods. The times of interest are predefined and the solution is computed either by aligning the step sizes with the target times or through accurate interpolation.

While it is in principle possible to implement such techniques with STDG, it is unlikely to be available in a pre-existing DG code. Further, the code will return the numerical solution at all points in one (or several) time elements simultaneously. In fact, it is not obvious that the STDG code will be able to return $\boldsymbol{u}^{n+1}$ in a simple way since this requires the extraction of a specific subset of coefficients from the numerical solution vector $\underline{\boldsymbol{u}}$. Yet, this may be necessary e.g. for visualization, to use adaptive time stepping, or in case the solution needs to be filtered or otherwise modified between time steps. The solution can in principle be constructed using $\boldsymbol{u}^{n+1} = (\boldsymbol{e}_{N_\tau}^\top \otimes \boldsymbol{I}_\xi)\underline{\boldsymbol{u}}$ with $\boldsymbol{e}_{N_\tau}^\top = (0, \ldots, 0, 1)$, as was done in Section 5. However, this assumes that the ordering of the unknowns in $\underline{\boldsymbol{u}}$ is identical to the one used in that analysis. If not, $(\boldsymbol{e}_{N_\tau}^\top \otimes \boldsymbol{I}_\xi)$ must be suitably permuted into some matrix $\boldsymbol{E}_{N_\tau}$ before application. Finding the appropriate permutation matrix may be a nontrivial task, in particular in 4D.

On the other hand, with STDG we have access too all the intermediate time stages by default, something that may be challenging with LoDG. This may be useful to compute L_2 errors of the numerical solution and has the additional advantage of allowing visualization of the solution away from the main time steps.

Adaptive time-stepping: The availability of adaptive time-stepping for Runge-Kutta methods is standard in modern software. Adaptivity generally requires a way of estimating the numerical error in the next time step. This information is used to adapt the time step to fit a predefined tolerance. Embedding techniques use a vector $\hat{\boldsymbol{b}}$ to compute a second numerical solution $\hat{\boldsymbol{u}}^{n+1}$ from (10) whose accuracy is one order lower than that of $\boldsymbol{u}^{n+1}$. The difference $\boldsymbol{u}^{n+1} - \hat{\boldsymbol{u}}^{n+1}$ can be used to estimate the local error without the need to solve the nonlinear system (27b) more than once. A detailed strategy for estimating the error and choosing the time step based on the embedding technique is available in [35, Chapter IV.8] for the Radau IIA method, but can be easily adapted to Lobatto IIIC [49]. This type of adaptivity will almost certainly be available in an implementation of LoDG.

An STDG code that follows the steps outlined in Section 4 will not automatically generate an embedded method. However, if the matrix $\boldsymbol{E}_{N_\tau}$ can be found that extracts $\boldsymbol{u}^{n+1}$, then it is also possible to construct a matrix $\hat{\boldsymbol{E}}_{N_\tau}$ that extracts $\hat{\boldsymbol{u}}^{n+1}$ such that an embedding technique can be used. However, depending on the space-time code, the user and the intended application, this procedure may be more invasive than desirable.

Alternatively, the numerical error may be estimated using Richardson extrapolation [34, Chapter II.4]. This procedure requires solving the nonlinear system (27a) three times; once with a step size $2\Delta t$ and twice with a step size Δt. The difference between the two solutions yields an error estimate. However, due to its expense, this approach hardly seems feasible for a 4D problem.

Adaptive Mesh Refinement (AMR): It seems to be challenging to effectively implement adaptive time stepping with STDG, at least when we sequentially solve for single time steps. However, since STDG uses a 4D mesh it is straightforward to set up a system that accounts for multiple temporal elements at once, which is not possible with LoDG. This introduces the possibility of using AMR in time in addition to space; see [40, 15] and the references therein. Like with Richardson extrapolation, using AMR forces us to solve the nonlinear system multiple times. Additionally, the system now consists of multiple coupled time steps. However, the additional cost may be offset by two factors: Firstly, we expect that the number of degrees of freedom necessary to achieve a given accuracy is significantly reduced by the AMR. Secondly, parallelism can be employed in the temporal direction.

Space-time AMR is not likely to be simple to set up with commercially available software. However, if the initial hurdles can be circumvented, then it is in principle possible to use completely unstructured space-time grids with h/p-refinement. The technique requires a generator for unstructured cuboid meshes in 4D (tesseracts) [12] and a way of estimating the numerical error in the final time. Such tools have been developed for 4D simplex meshes in [72, 12], but appear to be missing for other mesh types.

Shock capturing and limiting: The DG spatial discretizations used with Runge-Kutta time stepping are stable when applied to linear problems such as linear hyperbolic systems. However, for nonlinear problems spurious oscillations occur near strong shocks or steep gradients. In this case the DG method requires some extra stabilization unless a first order scheme ($p = 0$) is used that produces a monotonic structure in the shock region. For higher order schemes many approaches have been suggested to make this property available without introducing an excessive amount of numerical viscosity, which is a characteristic feature of first order schemes. Several approaches exist, including slope limiters, artificial

diffusion (viscosity) techniques, and even a posteriori techniques and order reduction methods. A comprehensive literature list is presented in [65].

In Dune-Fem-DG [17], both limiter based approaches and artificial diffusion are available to stabilize a DG scheme. The slope limiter based approach implemented in Dune-Fem-DG is coupled with a troubled cell indicator which makes the overall scheme highly non-linear and therefore not suitable for implicit methods, since the selection of troubled cells could change between linear iterations and lead to divergence of the linear solver. On the other hand, artificial diffusion approaches require a discretization of a diffusion term. This may result in severe time step restrictions and is thus a more suitable approach for fully implicit DG discretizations, in particular since stabilization diffusion coefficients only need to be re-computed every time step. A standard approach is available in Dune-Fem-DG. Since both LoDG and STDG are fully implicit schemes, it seems more suitable to apply artificial diffusion techniques for problems where strong shocks occur.

Nonlinear solvers and preconditioning: The solutions to the nonlinear systems (27a) and (27b) that appear in STDG and LoDG must be approximated in some way. It is natural to consider iterative methods for large systems. There have been a multitude of suggestions for how to design such methods; early solvers for implicit Runge-Kutta methods based on modified Newton iterations were introduced in [11] and [5]. A more optimized algorithm is described in [35], and many later developments use this as a starting point. These can be considered black box solvers in the sense that they do not utilize information about the spatial terms in the solution process.

Methods designed specifically for spatial DG discretizations and implicit Runge-Kutta methods are found in e.g. [59, 60]. Likewise, nonlinear sovlers designed for space-time DG and FEM discretizations have been developed [43, 67].

Unless the user is willing to make the (possibly considerable) effort to develop and/or implement a nonlinear solver specifically designed for LoDG or STDG, the natural recourse is to use a black box solver. Efficiency gains can possibly be made by introducing a preconditioner designed for DG discretizations; see e.g. [7, 60, 42] for recent developments. However, attention must be payed to the fact that the systems (27a) and (27b) have different algebraic properties and therefore likely will respond differently to preconditioners and nonlinear solvers.

For the nonlinear LoDG system (27b), the Jacobian is given by

$$\underline{\boldsymbol{I}} - \Delta t_0 (\boldsymbol{A} \otimes \boldsymbol{I}_\xi) \mathcal{J}(\underline{\boldsymbol{F}}), \tag{30}$$

where $\mathcal{J}(\underline{\boldsymbol{F}})$ contains the Jacobian of the spatial discretization. The solver in [35], and many recent developments that build upon it, instead use the mathematically equivalent

$$(\Delta t_0 \boldsymbol{A})^{-1} \otimes \boldsymbol{I}_\xi - \mathcal{J}(\underline{\boldsymbol{F}}). \tag{31}$$

For the nonlinear STDG system (27a), the Jacobian is given by

$$\left(\boldsymbol{D}_\tau^\top \boldsymbol{M}_\tau - \boldsymbol{e}_{N_\tau} \boldsymbol{e}_{N_\tau}^\top\right) \otimes \boldsymbol{I}_\xi + \frac{\Delta t_0}{2} (\boldsymbol{M}_\tau \otimes \boldsymbol{I}_\xi) \mathcal{J}(\underline{\boldsymbol{F}}). \tag{32}$$

Note from (26) that (31) arises by multiplying (30) by $(\boldsymbol{D}_\tau + \boldsymbol{M}_\tau^{-1} \boldsymbol{e}_1 \boldsymbol{e}_1^\top) \otimes \boldsymbol{I}_\xi$. This formulation is therefore very closely related to the STDG Jacobian (32). In fact, they

only differ by an application of the SBP property (21) and a multiplication by the temporal mass matrix.

Consider the 1D linear advection equation $u_t + u_x = 0$ discretized using a single element in space and time. With STDG, the discretization is generated using DUNE. With LoDG, the spatial terms are generated with DUNE whereas the temporal terms are set up manually as in Section 5. Figure 1 shows the sparsity patterns of the Jacobians using order 1,2 and 3 in space and time. In each figure quadruplet, the Jacobian (32) of STDG is shown in the top left and the Jacobian (30) of LoDG in the top right. In the bottom right, the alternative formulation (31) is shown.

The first thing to note is that the LoDG formulation leads to a dense discretization whereas STDG is sparse. The formulation using $\boldsymbol{A}^{-1}$ is also sparse. It has the same number of nonzero elements as STDG, although their distribution is different. The explanation for this lies in the ordering of the unknowns. With LoDG, the node order is lexicographic in the temporal direction. However, the space-time element generated by DUNE is as shown in the bottom right of Figure 1, here with $N_\tau = 4$. This ordering is the result of a generic construction of the reference elements, which is based on a recursion over the spatial dimension d starting at the 0-dimensional reference element, i.e. a point. This recursion is also generating a natural ordering for the basis functions, starting with the basis functions located at points in an element and recursively down to the basis functions located inside the element. A detailed description of this construction is found in [20]. With a suitable permutation of the unknowns, the sparsity pattern of LoDG using $\boldsymbol{A}^{-1}$ coincides with STDG as seen in the bottom left of each figure quadruplet.

These observations suggest that the LoDG system (27b) may be more expensive to work with than the STDG system (27a), and that the $\boldsymbol{A}^{-1}$ formulation (31) may be a better choice. However, the interaction of particular preconditioners and solvers with these systems may also depend on the node ordering in ways that must be deduced through careful testing.

7 Experiments

In this section we perform a series of numerical tests to verify that the implementations of LoDG and STDG behave as expected. This entails a validation that the two codes give similar numerical solutions, and that the convergence rates of the temporal parts of the discretizations are as outlined in Section 5.

We start with solving the linear test equation to validate the convergence rates of the two solvers. A two-dimensional advection-diffusion test case follows, with the purpose of highlighting slight differences in the numerical solutions and particular challenges with respect to visualizing the solutions. Finally, the two and three dimensional Euler equations of gas dynamics are solved to demonstrate that both codes are capable of handling nonlinear space-time dynamics in multiple dimensions.

As mentioned previously, the spatial parts of both LoDG and STDG are generated using DUNE-FEM. The temporal part of LoDG is implemented in ASSIMULO whereas DUNE-FEM is used for the entire space-time discretization in STDG.

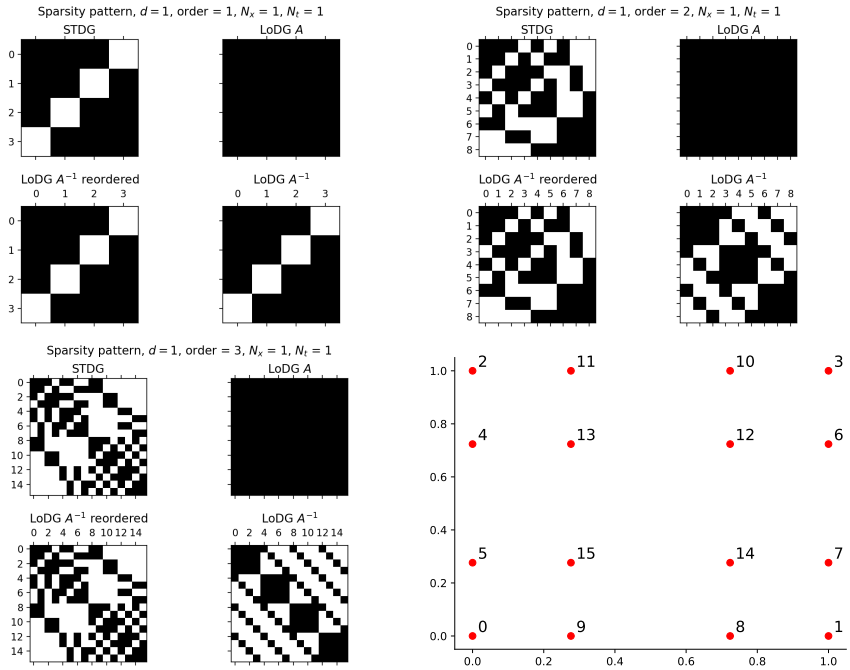


Figure 1: Sparsity patterns Jacobian of the advection problem for STDG and LoDG. Node order for one space-time element ($p = 3$) for STDG as generated by DUNE-FEM (bottom right).

7.1 Validation of Convergence Rates

To verify that the temporal discretizations converge as expected we perform a simple test on the linear test equation,

$$\begin{aligned} u_t &= -u, \quad t \in (0,1], \\ u(0) &= 4. \end{aligned} \tag{33}$$

For both methods, Python's sparse linear solver is used to solve the algebraic systems arising from the discretizations. The experimental order of convergence (EOC) of the pointwise error $|\boldsymbol{u}^{n+1} - u(1)|$ is shown for LoDG and STDG in Table 1. Here, N denotes the number of time steps/time elements and $N_\tau \in \{2,3,4\}$. Recall from Section 5 that the order of LoDG, and correspondingly the superconvergence of STDG, is $2(N_\tau - 1)$. This is indeed what we observe in Table 1. With $N_\tau = 3$, the errors are approaching machine precision when $N = 2^9$, hence a drop in the convergence rate is seen in Table 1b. The STDG appears to be more sensitive in this respect than LoDG. The same thing happens when $N_\tau = 4$ and $N = 2^5$, as seen in Table 1c.

We now repeat the experiment but measure the EOC via the L_2 norm $\|\cdot\|_{L_2[0,1]}$. This type of error measurement is straightforward to perform with the STDG code. However, the LoDG implementation does not by default save the intermediate RK stages necessary to perform the computation of the L_2 error. We expect the EOC to be given by N_τ.

Table 1: EOC of pointwise error for LoDG and STDG applied to the test equation (33).

(a) $N_\tau = 2$

N	Lobatto	DG-SEM
2^4	1.93	1.93
2^5	1.97	1.97
2^6	1.98	1.98
2^7	1.99	1.99
2^8	1.99	1.99
2^9	1.99	1.99

(b) $N_\tau = 3$

N	Lobatto	DG-SEM
2^4	3.96	3.96
2^5	3.98	3.98
2^6	3.99	3.99
2^7	3.99	4.01
2^8	3.99	4.28
2^9	4.05	0.47

(c) $N_\tau = 4$

N	Lobatto	DG-SEM
2^4	5.98	5.96
2^5	6.64	4.22

Table 2 shows that this indeed is observed. For $N_\tau = 4$ with $N = 2^8$ time elements the convergence rate drops due to very small errors, as seen in Table 2c. Again, STDG appears to be more sensitive to this phenomenon than LoDG.

Table 2: EOC of L_2 error for LoDG and STDG applied to the test equation (33).

(a) $N_\tau = 2$

N	Lobatto	DG-SEM
2^4	1.96	1.96
2^5	1.98	1.98
2^6	1.99	1.99
2^7	2.0	2.0
2^8	2.0	2.0
2^9	2.0	2.0

(b) $N_\tau = 3$

N	Lobatto	DG-SEM
2^4	2.98	2.98
2^5	2.99	2.99
2^6	2.99	2.99
2^7	3.0	3.0
2^8	3.0	3.0
2^9	3.0	3.0

(c) $N_\tau = 4$

N	Lobatto	DG-SEM
2^4	3.99	3.99
2^5	4.0	4.0
2^6	4.0	4.0
2^7	4.0	4.0
2^8	4.0	3.6

7.2 Advection-Diffusion

The next test case is the linear advection-diffusion problem in two dimensions;

$$\begin{aligned} \partial_t \boldsymbol{u} + \mathbf{b} \cdot \nabla \boldsymbol{u} - \varepsilon \Delta \boldsymbol{u} &= 0 \qquad \text{in } (\Omega \times (0,T]),\ \Omega \subset \mathbb{R}^2, \\ \boldsymbol{u}(0) &= \boldsymbol{u}_0 \qquad \text{in } \Omega. \end{aligned} \tag{34}$$

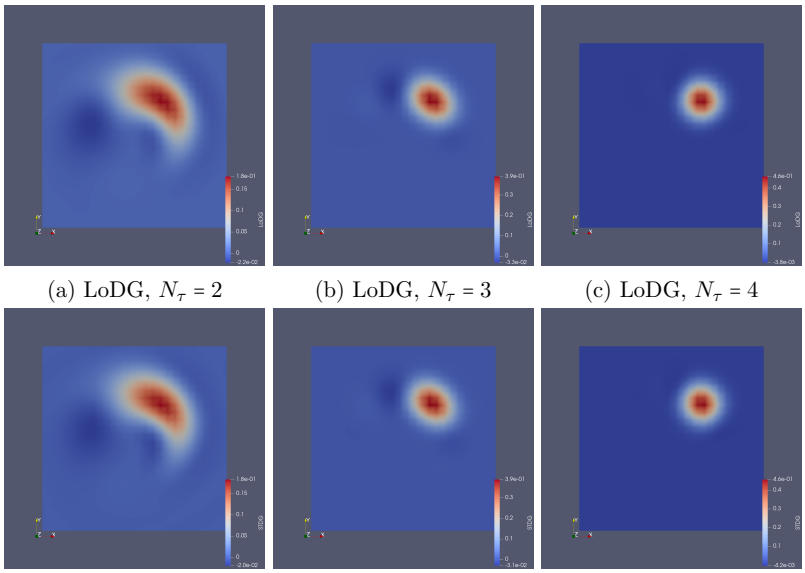
We test both implementations for the rotating pulse problem with analytic solution

$$\begin{aligned} \boldsymbol{u}(t,\mathbf{x}) &= \frac{0.004}{0.004 + 4\varepsilon t} \exp\left(-\frac{x_q^2 + y_q^2}{0.004 + 4\varepsilon t} \right), \\ x_q &= x_0 \cos(4t) + y_0 \sin(4t) + 0.25, \\ y_q &= -x_0 \sin(4t) + y_0 \cos(4t). \end{aligned}$$

Here, $x_0 = x - 0.5$, $y_0 = y - 0.5$, $\mathbf{b} = [-4y_0, 4x_0]$, $\varepsilon = 0.001$, and $(t,\mathbf{x}) \in [0,1] \times [0,1]^2$. The initial condition is given by $\boldsymbol{u}(0,\mathbf{x})$ and we apply periodic boundary conditions in space. The linear systems arising from the discretizations are solved using built-in routines in

Dune and Assimulo. Thus, despite the mathematical equivalence of LoDG and STDG, we do not expect the two codes to yield identical solutions.

The numerical solutions obtained by the two codes with $N_\tau \in \{2, 3, 4\}$ are shown in Figure 2. Here, a uniform mesh is used in space with $\Delta x = \Delta y = 0.04$. Time steps of uniform size $\Delta t = 0.1$ are used throughout the simulation. With $N_\tau = 2$ the problem is significantly under-resolved, leading to a smeared solution. As N_τ is increased, this phenomenon is reduced. To the eye, the numerical solutions using the two codes are barely distinguishable.



(a) LoDG, $N_\tau = 2$ (b) LoDG, $N_\tau = 3$ (c) LoDG, $N_\tau = 4$

(d) STDG, $N_\tau = 2$ (e) STDG, $N_\tau = 3$ (f) STDG, $N_\tau = 4$

Figure 2: Numerical solution of a rotating pulse subject to the advection-diffusion equation (34) using LoDG (top) and STDG (bottom).

To get a more detailed comparison of the numerical results obtained by the two implementations we compare their spatial L_2 error in the final time point, $t = 1$. This time we vary the space-time grid with $\Delta x = \Delta y = \Delta t = 1/N$. The errors and the EOC are shown in Table 3. Notice that the L_2 errors are very similar, although not identical, testifying to the influence of the different solvers of the algebraic equations. Note also that the behaviour of the EOC is less clear than it was for the linear test equation. In this experiment we have refined space and time simultaneously, and therefore do not have a theoretical convergence result to rely on. The results indicate a convergence rate higher than N_τ, although not quite as high as $2(N_\tau - 1)$.

Finally, we highlight a feature of the STDG code that may be of use in certain situations. Since this code returns all points in a given time element (or equivalently, all intermediate RK stages in each time step), these can be visualized using a 3D plotting software, thereby obtaining a space-time visualization of the solution. These stages are usually discarded by ODE solvers for efficiency reasons. The visualization is done for a

Table 3: Errors and EOC for the advection-diffusion problem (34).

	$N_\tau = 2$		$N_\tau = 3$		$N_\tau = 4$	
N	LoDG	STDG	LoDG	STDG	LoDG	STDG
2^2	8.94E-2	7.28E-2	4.45E-2	4.37E-2	2.68E-2	2.69E-2
2^3	4.66E-2	4.46E-2	2.42E-2	2.41E-2	6.05E-3	6.04E-3
2^4	3.49E-2	3.39E-2	5.36E-3	5.38E-3	4.92E-4	4.93E-4
2^5	1.86E-2	1.84E-2	5.85E-4	5.94E-4	1.06E-5	9.88E-6

(a) Error

	$N_\tau = 2$		$N_\tau = 3$		$N_\tau = 4$	
N	LoDG	STDG	LoDG	STDG	LoDG	STDG
2^3	0.9	0.7	0.9	0.9	2.1	2.2
2^4	0.4	0.4	2.2	2.2	3.6	3.6
2^5	0.9	0.9	3.2	3.2	5.5	5.6

(b) EOC

single time step in Figure 3. Here, the exact solution over the whole space-time domain is also shown for reference.

The same technique is of course considerably more challenging to use in a 4D setting. Nevertheless, this feature may be useful for prototyping and testing code in simpler contexts.

7.3 Euler Equations

A prime example for evolution equations are the *Euler equations of gas dynamics*. They are derived from the conservation of mass, momentum, and energy of a compressible inviscid fluid. In Eulerian coordinates they have the form:

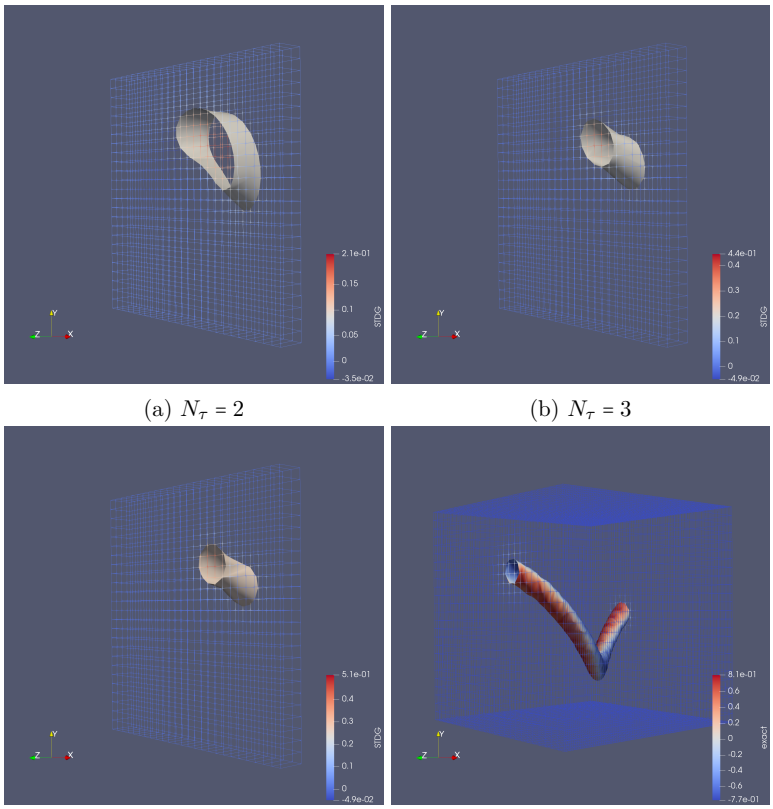
$$\begin{aligned} \partial_t \boldsymbol{u} + \nabla \cdot F_c(\boldsymbol{u}) &= 0 \qquad \text{in } (\Omega \times (0,T]),\ \Omega \subset \mathbb{R}^d, \quad d \in \{1,2,3\}, \\ \boldsymbol{u}(0) &= \boldsymbol{u}_0 \qquad \text{in } \Omega, \end{aligned} \tag{35}$$

where the vector of the conservative variables has the form

$$\boldsymbol{u} = \begin{pmatrix} \rho \\ \rho \boldsymbol{v} \\ \varepsilon \end{pmatrix}, \quad \rho \boldsymbol{v} = (\rho v_1, \dots, \rho v_d)^T, \quad \varepsilon = \rho \mathcal{E}, \tag{36}$$

augmented with suitable boundary conditions (which are discussed in detail in [6]). Here, ρ denotes the density of the fluid, $\boldsymbol{v}$ the velocity, ε the internal energy, and $\mathcal{E}$ the total energy. The convective flux function $F_c(\boldsymbol{u}) := (\boldsymbol{f}_1(\boldsymbol{u}), \dots, \boldsymbol{f}_d(\boldsymbol{u}))$ has for $i = 1, ..., d$ the form

$$\boldsymbol{f}_i(\boldsymbol{u}) := \begin{pmatrix} \boldsymbol{u}_{i+1} \\ \boldsymbol{u}_{i+1}\boldsymbol{u}_2/\boldsymbol{u}_1 + \delta_{i,1} P(\boldsymbol{u}) \\ \vdots \\ \boldsymbol{u}_{i+1}\boldsymbol{u}_{d+1}/\boldsymbol{u}_1 + \delta_{i,d} P(\boldsymbol{u}) \\ (\boldsymbol{u}_{d+2} + P(\boldsymbol{u}))\, \boldsymbol{u}_{i+1}/\boldsymbol{u}_1 \end{pmatrix},$$



(a) $N_\tau = 2$ (b) $N_\tau = 3$

(c) $N_\tau = 4$ (d) Exact solution in space-time

Figure 3: 3D space-time visualization of the rotating pulse produced with the STDG code.

where $\delta_{i,j}$ is the Kronecker delta. For example, choosing $d = 3$ and directly using $\boldsymbol{u}$ from (36) we obtain the three flux functions

$$\boldsymbol{f}_1(\boldsymbol{u}) = \begin{pmatrix} \rho v_1 \\ \rho v_1^2 + P \\ \rho v_1 v_2 \\ \rho v_1 v_3 \\ (\varepsilon + P) v_1 \end{pmatrix}, \quad \boldsymbol{f}_2(\boldsymbol{u}) = \begin{pmatrix} \rho v_2 \\ \rho v_2 v_1 \\ \rho v_2^2 + P \\ \rho v_2 v_3 \\ (\varepsilon + P) v_2 \end{pmatrix}, \quad \boldsymbol{f}_3(\boldsymbol{u}) = \begin{pmatrix} \rho v_3 \\ \rho v_3 v_1 \\ \rho v_3 v_2 \\ \rho v_3^2 + P \\ (\varepsilon + P) v_3 \end{pmatrix}.$$

We consider the two- and three-dimensional Euler equations with periodic boundary

conditions and vortex initial condition

$$
\begin{aligned}
\rho &= \left(1 - S^2(\gamma-1)M^2\frac{\exp(f)}{(8\pi^2)}\right)^{\frac{1}{\gamma-1}},\\
v_1 &= 1 - S\mathbf{x}_1\frac{\exp\left(\frac{f}{2}\right)}{2\pi},\\
v_2 &= S\mathbf{x}_0\frac{\exp\left(\frac{f}{2}\right)}{2\pi},\\
v_3 &= S\mathbf{x}_2\frac{\exp\left(\frac{f}{2}\right)}{2\pi},\\
\varepsilon &= \frac{P}{\gamma-1} + 0.5\frac{v_1^2+v_2^2+v_3^2}{\rho},\ P = \frac{\rho^\gamma}{\gamma M^2},
\end{aligned}
$$

with vortex strength $S = 5$, Mach number $M = 0.5$, $\gamma = 1.4$ and $f = 1 - \mathbf{x}_0^2 - \mathbf{x}_1^2 - \mathbf{x}_2^2$.

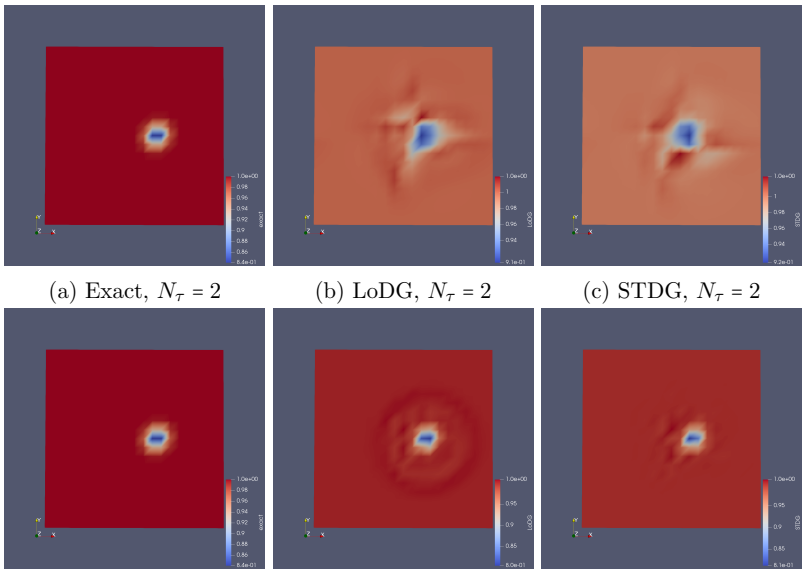
The numerical solutions obtained by the two codes with $N_\tau \in \{2, 3\}$ are shown in Figure 4. Here, a uniform mesh on the space-time domain $[-10, 10]^2 \times (0, 2.5]$ is used in space with $\Delta x = \Delta y = 0.04$. Time steps of uniform size $\Delta t = 0.01$ are used throughout the simulation. Again, the problem is significantly under-resolved with $N_\tau = 2$, leading to a smeared solution. As N_τ is increased, this phenomenon is reduced. Some differences can be seen in the numerical results for the two implementations. This is likely caused by the fact that two different solvers, inherent to Assimulo and Dune respectively, are used for the nonlinear systems arising from the discretizations. Due to the differences between these solvers we can in general not expect identical numerical solutions despite the mathematical equivalence of the two algorithms.

To show the potential of our STDG code we present a 3D Euler test case with $N_\tau = 3$. We modify the problem slightly and consider the space-time domain $[-5, 5]^3 \times (0, 2]$ and place the initial vortex slightly to the left of the centre. A uniform mesh is used in space with $\Delta x = \Delta y = \Delta z = 1$ and time steps of uniform size $\Delta t = 0.5$ are used throughout the simulation. The initial condition and the final time element of the density can be seen in Figure 5. These results show the potential of the Dune code even for 4D problems. Recently, 4D problems have been taken into consideration [27], but to the best of our knowledge this is the first 4D DG-SEM implementation available publicly.

8 Conclusions

In this paper we have presented a comparison of the theoretical and practical aspects of two different space-time DG-SEM implementations. DG-SEM in time using an upwind numerical flux is equivalent to the Lobatto IIIC family of Runge-Kutta methods in the sense that the two methods yield the same numerical solution when solved exactly. Thus two strategies for implementing DG-SEM in space-time exist: Either the method of lines with DG-SEM in space and Lobatto IIIC in time, which we refer to as LoDG, or a space-time DG-SEM, which we refer to as STDG.

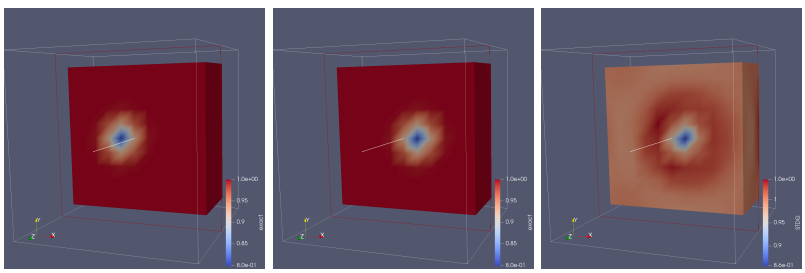
Despite the mathematical equivalence, there are important differences between the approaches. They are described by different terminology, each originating in its respective community. We have related these terminologies of the DG and RK methods. Moreover, the approaches lead to different algebraic systems of linear or nonlinear equations. When



(a) Exact, $N_\tau = 2$ (b) LoDG, $N_\tau = 2$ (c) STDG, $N_\tau = 2$

(d) Exact, $N_\tau = 3$ (e) LoDG, $N_\tau = 3$ (f) STDG, $N_\tau = 3$

Figure 4: Numerical solution of ρ for a vortex problem subject to the 2D Euler equations (35) using LoDG and STDG.



(a) Initial condition (b) Exact solution (c) STDG solution, $N_\tau = 3$

Figure 5: Numerical solution of ρ for a vortex problem subject to the 3D Euler equations (35) using STDG.

approximating their solution numerically, different challenges and possibilities arise with respect to algorithmic aspects.

We have compared the two implementations using an STDG code in Dune and an LoDG code with a spatial DG-SEM in Dune combined with Lobatto IIIC time stepping in Assimulo. The two approaches lead to very different software structure. An overview of the algorithmic capabilities have been given, some of which are likely to be more readily available, depending on which approach is chosen.

The choice of an appropriate space-time DG-SEM implementation depends on the needs of the user as well as on the codes available. For prototype testing of space-time

simulations of dimension $d \leq 2$ the STDG is a good alternative that allows to reuse optimized code. For larger simulations the LoDG ansatz is preferable since it provides more flexibility.

We are currently extending the STDG code in DUNE such that the DG orders in space and time can be chosen independently. Moreover, the code needs to be parallelized. We are also interested in considering problems with shocks, adaptive time stepping as well as unstructured grids in space and to combine the spatial DG-SEM with other implicit Runge-Kutta time stepping methods to yield a space-time DG implementation.

Appendix A

There exist different ways to install DUNE and ASSIMULO. Here, we only describe the simplest and most straight forward way to install both, DUNE and ASSIMULO, which is to use a conda environment. Then the installation is done in the following way:

```
conda create -n duneproject # create a new conda environment

conda activate duneproject # activate the conda environment

conda install -c conda-forge assimulo # install assimulo

pip install -U dune-fem-dg # install dune using pip, no conda package yet

conda install -c conda-forge scipy # install scipy
```

The space-time DG-SEM code is available online at https://gitlab.maths.lth.se/dune/spacetimelobattocode.

Appendix B

The Butcher tableaus for the N_τ-stage Lobatto IIIC methods with $N_\tau = 2, 3, 4$ can be seen in Table 4.

Appendix C

This patch adds 4D simplex and cuboid reference elements to UFL needed for the 3D+t simulations. This patch is currently implemented in DUNE-FEM and will be discussed with the UFL community.

```
# 4d patching of reference elements
def _patchufl4d():
    from ufl.sobolevspace import H1
    from ufl.finiteelement.elementlist import ufl_elements, any_cell, register_element
    from ufl.cell import num_cell_entities, cellname2facetname,
    from ufl.cell import _simplex_dim2cellname, _hypercube_dim2cellname

    # check if this has been added before
    if not 'pentatope' in ufl.cell.num_cell_entities:
        # 4d-simplex
        ufl.cell.num_cell_entities["pentatope"] = (5, 10, 10, 5, 1)
        # 4d-cube
        ufl.cell.num_cell_entities["tesseract"]   = (16, 32, 24, 8, 1)
```

$$
\begin{array}{c|cc}
0 & \frac{1}{2} & -\frac{1}{2} \\
1 & \frac{1}{2} & \frac{1}{2} \\
\hline
 & \frac{1}{2} & \frac{1}{2}
\end{array}
$$

(a) $N_\tau = 2$

$$
\begin{array}{c|ccc}
0 & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} \\
\frac{1}{2} & \frac{1}{6} & \frac{5}{12} & -\frac{1}{12} \\
1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\
\hline
 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6}
\end{array}
$$

(b) $N_\tau = 3$

$$
\begin{array}{c|cccc}
0 & \frac{1}{12} & -\frac{\sqrt{5}}{12} & \frac{\sqrt{5}}{12} & -\frac{1}{12} \\
\frac{1}{2} - \frac{\sqrt{5}}{10} & \frac{1}{12} & \frac{1}{4} & \frac{10-7\sqrt{5}}{60} & \frac{\sqrt{5}}{60} \\
\frac{1}{2} + \frac{\sqrt{5}}{10} & \frac{1}{12} & \frac{10+7\sqrt{5}}{60} & \frac{1}{4} & -\frac{\sqrt{5}}{60} \\
1 & \frac{1}{12} & \frac{5}{12} & \frac{5}{12} & \frac{1}{12} \\
\hline
 & \frac{1}{12} & \frac{5}{12} & \frac{5}{12} & \frac{1}{12}
\end{array}
$$

(c) $N_\tau = 4$

Table 4: Butcher Tableaus for Lobatto IIIC methods

```python

        # recompute cell name to dimension mapping
        ufl.cell.cellname2dim = dict((k, len(v) - 1) for k, v \
          in ufl.cell.num_cell_entities.items())

        ufl.cell.cellname2facetname["pentatope"] = "tetrahedron"
        ufl.cell.cellname2facetname["tesseract"]   = "hexahedron"

        ufl.cell._simplex_dim2cellname[4]   = "pentatope"
        ufl.cell._hypercube_dim2cellname[4] = "tesseract"

        # add types to element lists
        ufl.finiteelement.elementlist.simplices =\
            ufl.finiteelement.elementlist.simplices + ("pentatope",)
        ufl.finiteelement.elementlist.cubes = \
            ufl.finiteelement.elementlist.cubes + ("tesseract",)
        ufl.finiteelement.elementlist.any_cell =\
                ufl.finiteelement.elementlist.any_cell + ("pentatope", "tesseract", )

        # register Lagrange again with new element type list
        ufl_elements.pop("Lagrange")
        ufl_elements.pop("CG")
        register_element("Lagrange", "CG", 0, H1, "identity", (1, None), \
          ufl.finiteelement.elementlist.any_cell)
```

```python
# selecting a cell based on the dimension of the domain and or grid
def cell(dimDomainOrGrid):
    if isinstance(dimDomainOrGrid,ufl.Cell):
        return dimDomainOrGrid
    try:
        dimWorld = int(dimDomainOrGrid.dimWorld)
        dimDomain = int(dimDomainOrGrid.dimGrid)
    except:
        dimDomain = dimDomainOrGrid
        if isinstance(dimDomain, tuple):
            if len(dimDomain) != 2:
                raise Exception('dimDomain tuple must contain exactly two elements.')
            dimWorld = int(dimDomain[1])
            dimDomain = dimDomain[0]
        else:
```

```
            dimWorld = int(dimDomain)
    if dimDomain == 1:
        return ufl.Cell("interval", dimWorld)
    elif dimDomain == 2:
        return ufl.Cell("triangle", dimWorld)
    elif dimDomain == 3:
        return ufl.Cell("tetrahedron", dimWorld)
    elif dimDomain == 4:
        # add 4d cell types to ufl data structures
        _patchufl4d()
        return ufl.Cell("pentatope", dimWorld)
    else:
        raise NotImplementedError('UFL cell not implemented for dimension '\
           + str(dimDomain) + '.')
```

Appendix D

For $N_\tau = 2$ we get

$$\boldsymbol{B}_\tau = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \ \boldsymbol{M}_\tau = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \ \boldsymbol{D}_\tau = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix}. \tag{37}$$

For $N_\tau = 3$ we get

$$\boldsymbol{B}_\tau = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \ \mathbf{M}_\tau = \begin{pmatrix} \frac{1}{3} & 0 & 0 \\ 0 & \frac{4}{3} & 0 \\ 0 & 0 & \frac{1}{3} \end{pmatrix}, \ \boldsymbol{D}_\tau = \begin{pmatrix} -\frac{3}{2} & 2 & -\frac{1}{2} \\ -\frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & -2 & \frac{3}{2} \end{pmatrix}. \tag{38}$$

For $N_\tau = 4$ we get

$$\boldsymbol{B}_\tau = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \ \boldsymbol{M}_\tau = \begin{pmatrix} \frac{1}{6} & 0 & 0 & 0 \\ 0 & \frac{5}{6} & 0 & 0 \\ 0 & 0 & \frac{5}{6} & 0 \\ 0 & 0 & 0 & \frac{1}{6} \end{pmatrix}, \ \boldsymbol{D}_\tau = \begin{pmatrix} -3 & \frac{5+5\sqrt{5}}{4} & \frac{5-5\sqrt{5}}{4} & \frac{1}{2} \\ \frac{-1-\sqrt{5}}{4} & 0 & \frac{-\sqrt{5}}{2} & \frac{1-\sqrt{5}}{4} \\ \frac{-1+\sqrt{5}}{4} & \frac{-\sqrt{5}}{2} & 0 & \frac{1+\sqrt{5}}{4} \\ -\frac{1}{2} & \frac{5\sqrt{5}-5}{4} & \frac{-5-5\sqrt{5}}{4} & 3 \end{pmatrix}. \tag{39}$$

References

[1] M. S. Alnæs, A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells. Unified Form Language: A Domain-Specific Language for Weak Formulations of Partial Differential Equations. *ACM Trans. Math. Softw.*, 40(2), 2014.

[2] C. Andersson, C. Führer, and J. Åkesson. Assimulo: A unified framework for ODE solvers. *Math. Comput. Simulat.*, 116(0):26 – 43, 2015.

[3] P. Bastian, M. Blatt, M. Dedner, N.-A. Dreier, R. Engwer, Ch. Fritze, C. Gräser, C. Grüninger, D. Kempf, R. Klöfkorn, M. Ohlberger, and O. Sander. The Dune framework: Basic concepts and recent developments. *Comput. Math. Appl.*, 2020.

[4] M. Behr. Simplex space–time meshes in finite element simulations. *Int. J. Numer. Meth. Fl.*, 57(9):1421–1434, 2008.

[5] T. A. Bickart. An efficient solution process for implicit Runge–Kutta methods. *SIAM J. Numer. Anal.*, 14(6):1022–1027, 1977.

[6] P. Birken. *Numerical Methods for Unsteady Compressible Flow Problems*. CRC Press, Boca Raton, London, New York, 2021.

[7] P. Birken, G. J. Gassner, and L. M. Versbach. Subcell finite volume multigrid preconditioning for high-order discontinuous Galerkin methods. *Int. J. Comput. Fluid. D.*, pages 1–9, 2019.

[8] P. D. Boom and D. W. Zingg. High-Order Implicit Time-Marching Methods Based on Generalized Summation-by-Parts Operators. *SIAM J. Sci. Comput.*, 37:A2682–A2709, 2015.

[9] S. Brdar, A. Dedner, and R. Klöfkorn. Compact and stable Discontinuous Galerkin methods for convection-diffusion problems. *SIAM J. Sci. Comput.*, 34(1):263–282, 2012.

[10] J. C. Butcher. Implicit Runge-Kutta Processes. *Math. Comput.*, 18(85):50–64, 1964.

[11] J. C. Butcher. On the implementation of implicit Runge-Kutta methods. *BIT*, 16(3):237–240, 1976.

[12] P. C. Caplan, R. Haimes, D. L. Darmofal, and M. C. Galbraith. Four-dimensional anisotropic mesh adaptation. *Comput. Aided Design*, 129:102915, 2020.

[13] M. H. Carpenter, T. C. Fisher, E. J. Nielsen, and S. H. Frankel. Entropy stable spectral collocation schemes for the Navier–Stokes equations: Discontinuous interfaces. *SIAM J. Sci. Comput.*, 36(5):B835–B867, 2014.

[14] M. H. Carpenter and D. Gottlieb. Spectral methods on arbitrary grids. *J. Comput. Phys.*, 129(1):74–86, 1996.

[15] Z. Chen, H. Steeb, and S. Diebels. A space-time discontinuous Galerkin method applied to single-phase flow in porous media. *Computat. Geosci.*, 12(4):525–539, 2008.

[16] A. Dedner, S. Girke, R. Klöfkorn, and T. Malkmus. The DUNE-FEM-DG module. *Archive of Numerical Software*, 5(1), 2017.

[17] A. Dedner and R. Klöfkorn. Extendible and Efficient Python Framework for Solving Evolution Equations with Stabilized Discontinuous Galerkin Method. *Comm. App. Math. Comp. Sci.*, 2021.

[18] A. Dedner, R. Klöfkorn, and M. Nolte. Python bindings for the dune-fem module. *Zenodo (Mar 2020)*, 2020.

[19] A. Dedner, R. Klöfkorn, M. Nolte, and M. Ohlberger. A Generic Interface for Parallel and Adaptive Scientific Computing: Abstraction Principles and the DUNE-FEM Module. *Computing*, 90(3–4):165–196, 2010.

[20] A. Dedner and M. Nolte. Construction of Local Finite Element Spaces Using the Generic Reference Elements. In A. Dedner, B. Flemisch, and R. Klöfkorn, editors, *Advances in DUNE*, pages 3–16, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[21] L. T. Diosady and S. M. Murman. Tensor-product preconditioners for higher-order space–time discontinuous Galerkin methods. *J. Comput. Phys.*, 330:296–318, 2017.

[22] D. C. D. R. Fernández, J. E. Hicken, and D. W. Zingg. Review of summation-by-parts operators with simultaneous approximation terms for the numerical solution of partial differential equations. *Comput. Fluids*, 95:171–196, 2014.

[23] T. C. Fisher and M. H. Carpenter. High-order entropy stable finite difference schemes for nonlinear conservation laws: Finite domains. *J. Comput. Phys.*, 252:518–557, 2013.

[24] T. C. Fisher, M. H. Carpenter, J. Nordström, N. K. Yamaleev, and C. Swanson. Discretely conservative finite-difference formulations for nonlinear conservation laws in split form: Theory and boundary conditions. *J. Comput. Phys.*, 234:353–375, 2013.

[25] M. Franciolini and S. M. Murman. Multigrid preconditioning for a space-time spectral-element discontinuous-Galerkin solver. In *AIAA Scitech 2020 Forum*, page 1314, 2020.

[26] L. Friedrich, G. Schnücke, A. R. Winters, D. C. R. Fernández, G. J. Gassner, and M. H. Carpenter. Entropy Stable Space–Time Discontinuous Galerkin Schemes with Summation-by-Parts Property for Hyperbolic Conservation Laws. *J. Sci. Comput.*, 80(1):175–222, 2019.

[27] C. V. Frontin, G. S. Walters, F. D. Witherden, C. W. Lee, D. M. Williams, and D. L. Darmofal. Foundations of space-time finite element methods: Polytopes, interpolation, and integration. *Appl. Numer. Math.*, 166:92–113, 2021.

[28] M. J. Gander. 50 Years of Time Parallel Time Integration. In T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, editors, *Multiple Shooting and Time Domain Decomposition Methods*, pages 69–113, Cham, 2015. Springer.

[29] M. J. Gander and M. Neumüller. Analysis of a new space-time parallel multigrid algorithm for parabolic problems. *SIAM J. Sci. Comput.*, 38(4):A2173–A2208, 2016.

[30] G. J. Gassner. A skew-symmetric discontinuous Galerkin spectral element discretization and its relation to SBP-SAT finite difference methods. *SIAM J. Sci. Comput.*, 35(3):A1233–A1253, 2013.

[31] G. J. Gassner and A. R. Winters. A novel robust strategy for discontinuous Galerkin methods in computational fluid mechanics: Why? When? What? Where? *Front. Phys.*, page 612, 2021.

[32] C. Gersbacher. *Higher-order discontinuous finite element methods and dynamic model adaptation for hyperbolic systems of conservation laws.* Phd thesis, University of Freiburg, 2017.

[33] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, volume 31. Springer, Berlin, Heidelberg, 2006.

[34] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I.* Springer, Berlin, Heidelberg, 2009.

[35] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II*, volume 14 of *Springer Series in Computational Mathematics.* Springer, Berlin, Heidelberg, 2010.

[36] J. S. Hesthaven and T. Warburton. *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications.* Springer-Verlag New York, 2008.

[37] J. E. Hicken and D. W. Zingg. Superconvergent functional estimates from summation-by-parts finite-difference discretizations. *SIAM J. Sci. Comput.*, 33(2):893–922, 2011.

[38] A. Jameson. Evaluation of fully implicit Runge Kutta schemes for unsteady flow calculations. *J. Sci. Comput.*, 73(2-3):819–852, 2017.

[39] L. O. Jay. Lobatto methods. In B. Engquist, editor, *Encyclopedia of Applied and Computational Mathematics*, pages 817–826. Springer, Berlin, Heidelberg, 2015.

[40] S. Jayasinghe, D. L. Darmofal, N. K. Burgess, M. C. Galbraith, and S. R. Allmaras. A space-time adaptive method for reservoir flows: formulation and one-dimensional application. *Computat. Geosci.*, 22(1):107–123, 2018.

[41] G. Karniadakis and S. Sherwin. *Spectral/hp Element Methods for Computational Fluid Dynamics.* Oxford University Press, Oxford, 2013.

[42] J. Kasimir, L. M. Versbach, P. Birken, G. J. Gassner, and R. Klöfkorn. An Finite Volume Based Multigrid Preconditioner for DG-SEM for Convection-Diffusion. In *14th WCCM-ECCOMAS Congress 2020*, volume 600, 2021.

[43] C. M. Klaij, M. H. van Raalte, H. van der Ven, and J. J. van der Vegt. h-Multigrid for space-time discontinuous Galerkin discretizations of the compressible Navier-Stokes equations. *J. Comput. Phys.*, 227(2):1024–1045, 2007.

[44] D. A. Kopriva. *Implementing Spectral Methods for Partial Differential Equations.* Springer, Dordrecht, 2009.

[45] D. A. Kopriva and G. Gassner. On the quadrature and weak form choices in collocation type discontinuous galerkin spectral element methods. *J. Sci. Comput.*, 44:136–155, 2010.

[46] D. A. Kopriva, S. L. Woodruff, and M. Y. Hussaini. Computation of electromagnetic scattering with a non-conforming discontinuous spectral element method. *Int. J. Numer. Methods. Eng.*, 53(1):105–122, 2002.

[47] N. Krais, A. Beck, T. Bolemann, H. Frank, D. Flad, G. Gassner, F. Hindenlang, M. Hoffmann, T. Kuhn, M. Sonntag, et al. Flexi: A high order discontinuous galerkin framework for hyperbolic–parabolic conservation laws. *Comput. Math. Appl.*, 81:186–219, 2021.

[48] H.-O. Kreiss and G. Scherer. Finite Element and Finite Difference Methods for Hyperbolic Partial Differential Equations. In C. de Boor, editor, *Mathematical Aspects of Finite Elements in Partial Differential Equations*, pages 195–212. Academic Press, London, 1974.

[49] E. Lehsten. *Implementation of 3 stage Lobatto IIIC into Assimulo package*. Bachelor thesis, Lund University.

[50] V. Linders. On an eigenvalue property of Summation-By-Parts operators. *arXiv preprint arXiv:2201.01193*, 2022.

[51] V. Linders, J. Nordström, and S. H. Frankel. Properties of Runge-Kutta-Summation-By-Parts methods. *J. Comput. Phys.*, 419:109684, 2020.

[52] T. Lundquist and J. Nordström. The SBP–SAT technique for initial value problems. *J. Comput. Phys.*, 270:86–104, 2014.

[53] A. Müller, M. A. Kopera, S. Marras, L. C. Wilcox, T. Isaac, and F. X. Giraldo. Strong scaling for numerical weather prediction at petascale with the atmospheric model NUMA. *Int. J. High. Perform. Comput. Appl.*, 33(2):411–426, 2019.

[54] R. Nair, L. Bao, M. Toy, and R. Klöfkorn. A High-Order Multiscale Global Atmospheric Model. *AIAA AVIATION Forum*, 2016.

[55] M. Neumüller. *Space-Time Methods*, volume 20 of *Monograph Series TU Graz: Computation in Engineering and Science*. TU Graz, 2013.

[56] J. Nievergelt. Parallel methods for integrating ordinary differential equations. *Commun. ACM*, 7(12):731–733, 1964.

[57] J. Nordström and C. La Cognata. Energy stable boundary conditions for the nonlinear incompressible Navier–Stokes equations. *Math. Comput.*, 88(316):665–690, 2019.

[58] J. Nordström and T. Lundquist. Summation-by-parts in time. *J. Comput. Phys.*, 251:487–499, 2013.

[59] W. Pazner and P.-O. Persson. Stage-parallel fully implicit Runge–Kutta solvers for discontinuous Galerkin fluid simulations. *J. Comput. Phys.*, 335:700–717, 2017.

[60] W. Pazner and P.-O. Persson. Approximate tensor-product preconditioners for very high order discontinuous Galerkin methods. *J. Comput. Phys.*, 354:344–369, 2018.

[61] H. Ranocha. Some notes on summation by parts time integration methods. *Results Appl. Math.*, 1:100004, 2019.

[62] H. Ranocha, M. Schlottke-Lakemper, A. R. Winters, E. Faulhaber, J. Chan, and G. Gassner. Adaptive numerical simulations with Trixi.jl: A case study of Julia for scientific computing, 08 2021.

[63] A. A. Ruggiu and J. Nordström. On pseudo-spectral time discretizations in summation-by-parts form. *J. Comput. Phys.*, 360:192–201, 2018.

[64] J. Schneid. B-convergence of Lobatto IIIC formulas. *Numer. Math.*, 51(2):229–235, 1987.

[65] C.-W. Shu. High order WENO and DG methods for time-dependent convection-dominated PDEs: A brief survey of several recent developments. *J. Comput. Phys.*, 316:598 – 613, 2016.

[66] B. Strand. Summation by parts for finite difference approximations for d/dx. *J. Comput. Phys.*, 110(1):47–67, 1994.

[67] J. J. Sudirham, J. J. W. van der Vegt, and R. M. J. van Damme. Space-time discontinuous Galerkin method for advection-diffusion problems on time-dependent domains. *Appl. Numer. Math.*, 56(12):1491–1518, 2006.

[68] M. Svärd and J. Nordström. Review of summation-by-parts schemes for initial–boundary-value problems. *J. Comput. Phys.*, 268:17–38, 2014.

[69] T. E. Tezduyar and K. Takizawa. Space–time computations in practical engineering applications: A summary of the 25-year history. *Comput. Mech.*, 63(4):747–753, 2019.

[70] J. J. van der Vegt. *Space-time discontinuous Galerkin finite element methods*, pages 1–37. Von Karman Institute for Fluid Dynamics, Brussels, 2006.

[71] L. M. Versbach, P. Birken, V. Linders, and G. Gassner. Local Fourier Analysis of a Space-Time Multigrid Method for DG-SEM for the Linear Advection Equation. *arXiv preprint arXiv:2112.03115*, 2021.

[72] M. Yano and D. L. Darmofal. An optimization-based framework for anisotropic simplex mesh adaptation. *J. Comput. Phys.*, 231(22):7626–7649, 2012.

[73] Q. Zhang and C.-W. Shu. Error estimates to smooth solutions of runge–kutta discontinuous galerkin method for symmetrizable systems of conservation laws. *SIAM J. Numer. Anal.*, 44(4):1703–1720, 2006.

Paper v

Local Fourier Analysis of a Space-Time Multigrid Method for DG-SEM for the Linear Advection Equation

Lea M. Versbach[1*], Philipp Birken[1], Viktor Linders[1], Gregor Gassner[2]

[1]Centre for Mathematical Sciences, Numerical Analysis, Lund University, Lund, Sweden

[2]Center for Data and Simulation Sciences; Department of Mathematics and Computer Science, Weyertal 86-90, 50931 Köln, Germany

*lea_miko.versbach@math.lu.se

Abstract

In this paper we present a local Fourier analysis of a space-time multigrid solver for a hyperbolic test problem. The space-time discretization is based on arbitrarily high order discontinuous Galerkin spectral element methods in time and a first order finite volume method in space. We apply a block Jacobi smoother and consider coarsening in space-time, as well as temporal coarsening only. Asymptotic convergence factors for the smoother and the two-grid method for both coarsening strategies are presented. For high CFL numbers, the convergence factors for both strategies are 0.5 for first order, and 0.375 for second order accurate temporal approximations. Numerical experiments in one and two spatial dimensions for space-time DG-SEM discretizations of varying order give even better convergence rates of around 0.3 and 0.25 for sufficiently high CFL numbers.

Keywords: Local Fourier Analysis, Space-Time, Multigrid, Discontinuous Galerkin Spectral Element Method, Linear Advection Equation
Mathematics Subject Classification 2020: 65M55,65M22,65M60,65T99

1 Introduction

Space-time discontinuous Galerkin (DG) discretizations have received increased attention in recent years. One reason is that they allow for high order implicit discretizations and parallelization in time [12]. Moreover, new space-time DG spectral element methods have been constructed [11]. These are provably entropy stable for hyperbolic conservation laws which is of great interest in that community. Several authors have studied space-time DG methods for different equations, for instance hyperbolic problems in [8, 11], advection-diffusion problems in [23, 25], the Euler equations of gas dynamics in [30, 32] and nonlinear wave equations in [31].

The philosophy of space-time methods is to treat time as just an additional dimension [24]. This has several advantages, i.e. moving boundaries can be treated more easily [27] and parallelization in time is possible [12]. However, the technique also has challenges, since the temporal direction has a special role. Time always needs to follow a causality principle: a solution later in time is only determined by a solution earlier in time, never the other way around.

Several time parallel numerical methods exist, and can be divided into four groups [12]: Methods based on multiple shooting, methods based on domain decomposition and waveform relaxation, space-time multigrid methods and direct time parallel methods. In this article we focus on space-time multigrid methods, which often scale linearly with the number of unknowns [26].

An analysis tool for multigrid methods is the Local Fourier Analysis (LFA), introduced in [4]. It can be used to study smoothers and two-grid algorithms. The technique is based on assuming periodic boundary conditions and transforming the given problem into the frequency domain using a discrete Fourier transform. Thus, the LFA can be used as a predictor for asymptotic convergence rates when considering problems with non-periodic boundary conditions [10]. The smoothing and asymptotic convergence are both related to the eigenvalues of the operators for the smoother and the two-grid algorithm. These operators are very large for space-time discretizations, since the effective dimension becomes $d + 1$ for a d-dimensional problem. It is therefore not feasible to calculate the eigenvalues. When performing an LFA, the operators are of block diagonal form in the Fourier space, which reduces the problem to an analysis of so-called Fourier symbols. These are of much smaller size and make calculations feasible. Multigrid solvers have been analyzed in the DG context with block smoothers for convection-diffusion problems in [14, 23, 33] and for elliptic problems in [19, 20]. Space-time MG methods have been analyzed mostly for parabolic problems [9, 10, 13]. Analysis of space-time MG algorithms for DG discretizations of advection dominated flows has been quite limited but can be found for the advection-diffusion equation or linearized versions of the compressible Euler equations [29, 28] and for generalized diffusion problems [10].

In this article we use the LFA to analyze a space-time multigrid solver for a hyperbolic model problem. The analysis is similar to [13], where the authors considered a one-dimensional heat equation discretized with a finite element method in space and DG in time. Instead we study the one-dimensional linear advection equation discretized with a first order finite volume (FV) method in space and a discontinuous Galerkin spectral element method (DG-SEM) in time. This results in a fully discrete space-time DG discretization.

Due to the choice of the test problem as well as the spatial discretization we get complex Fourier symbols, making the analysis more difficult. However, for large CFL numbers we are able to determine asymptotic smoothing factors analytically. We use a block Jacobi smoother and compare two different coarsening strategies: coarsening in both temporal and spatial directions as well as coarsening in the temporal direction only. The two-grid convergence factors need to be calculated numerically based on the complex Fourier symbols. We compare the results of the analysis to numerical convergence rates for multi-dimensional advection problems with more general boundary conditions. These experiments are produced using the the Distributed and Unified Numerics Environment (DUNE), an open source modular toolbox for solving partial differential equations with grid-based methods as DG, finite element and finite differences [1, 5, 6, 7].

In section 2 we describe the model problem and the DG space-time discretization. Moreover, we introduce the multigrid solver and its components. In section 3 we discuss some preliminaries: the equivalence of temporal DG-SEM discretizations with upwind flux and Lobatto IIIC Runge-Kutta methods, stability results, as well as the basic tools needed for the LFA. In section 4 we derive the Fourier symbols for all elements of the multigrid iteration: discretization operator, smoother, restriction and prolongation. In section 5 we analyze the smoothing properties for a block Jacobi smoother to find an optimal damping parameter. The two-grid asymptotic convergence factors are calculated in section 6. Numerical results for the advection equation in one and two dimensions are presented in section 7 and serve as a comparison to the theoretical results obtained from the LFA. Conclusions are drawn in section 8.

2 Problem Description

The goal of this paper is to analyze a space-time multigrid solver for a discretized hyperbolic model problem. We consider the one-dimensional linear advection equation

$$u_t + au_x = 0, \ (x,t) \in [L,R] \times [0,T] =: \mathbf{\Omega} \subset \mathbb{R}^2 \tag{1}$$

with $a > 0$. For the analysis, we need periodic boundary conditions in space and time.

2.1 Discretization

For the analysis we discretize (1) with a space-time DG method with a FV method in space (which corresponds to one a DG method of order $p_x = 0$ with one Legendre-Gauss node, i.e., the midpoint) and variable temporal polynomial degree p_t. For the temporal direction, we use a DG-SEM. This is a specific DG discretization based on the following choices: The solution and the physical flux function are approximated element-wise by nodal polynomials using a Lagrangian basis based on Legendre-Gauss-Lobatto (LGL) nodes. Moreover, integrals are approximated by Gaussian quadrature, which is collocated with the nodes of the polynomial approximation [21].

To discretize equation (1), we start by dividing $\mathbf{\Omega}$ into space-time elements $[x_n, x_{n+1}] \times [t_m, t_{m+1}]$, $n = 1, \dots, N_x$, $m = 1, \dots, N$. A weak form on each space-time element $[x_n, x_{n+1}] \times [t_m, t_{m+1}]$ is

$$\int_{x_n}^{x_{n+1}} \int_{t_m}^{t_{m+1}} (u_t + au_x)\psi \mathrm{d}x\mathrm{d}t = 0,$$

for test functions ψ in a test space $C^1(\mathbf{\Omega})$.

It is of advantage for the DG-SEM discretization to map the temporal elements to the reference element $[-1,1]$ using the linear map $\tau(t) = 2\frac{t-t_m}{\Delta t}$ with $\Delta t = t_{m+1} - t_m$. Then we get the modified weak form

$$\frac{2}{\Delta t} \int_{x_n}^{x_{n+1}} \int_{-1}^{+1} u_\tau \psi \mathrm{d}x\mathrm{d}\tau + \int_{x_n}^{x_{n+1}} \int_{-1}^{1} au_x \psi \mathrm{d}x\mathrm{d}\tau = 0.$$

Integration by parts in both direction yields

$$\frac{2}{\Delta t} \int_{x_n}^{x_{n+1}} (u\psi|_{-1}^{1} - \int_{-1}^{+1} u\psi_\tau \mathrm{d}\tau)\mathrm{d}x + a \int_{-1}^{1} (u\psi|_{x_n}^{x_{n+1}} - \int_{x_n}^{x_{n+1}} u\psi_\xi \mathrm{d}x)\mathrm{d}\tau = 0.$$

We now approximate u on each space-time element $[x_n, x_{n+1}] \times [-1, 1]$ by polynomials of variable degree p_t in time and constant degree $p_x = 0$ in space:

$$u(x, \tau) = \sum_{i=1}^{N_t} u_{ij} \ell_i(\tau),$$

with $N_t = p_t + 1$ and $j = 1, \ldots, N_x$ volumes in space. For the DG-SEM discretization, the basis functions ℓ_i are Lagrange polynomials of degree p_t based on the Legendre-Gauss-Lobatto (LGL) nodes $\{\tau_j\}_{i=1}^{N_t}$ in $[-1, 1]$. We choose ψ from the same space, thus

$$\psi(x, \tau) = \sum_{i=1}^{N_t} \psi_{ij} \ell_i(\tau), \; j = 1, \ldots, N_x.$$

Inserting the polynomial approximations gives

$$\frac{2}{\Delta t} \int_{x_n}^{x_{n+1}} (u\psi|_{-1}^{1} - \int_{-1}^{+1} u\psi_\tau \mathrm{d}\tau) \mathrm{d}x + a \int_{-1}^{1} (u(x_{n+1}, \tau)\psi(x_{n+1}, \tau) - u(x_n, \tau))\psi(x_n, \tau) \mathrm{d}\tau = 0.$$

Next, we approximate the integrals in time with Gaussian quadrature using the same LGL nodes $\{\tau_i\}_{i=1}^{N_\tau}$ and weights $\{\omega_i\}_{i=1}^{N_t}$ as for the basis functions

$$\int_{-1}^{1} f(\tau) \mathrm{d}\tau \approx \sum_{i=1}^{N_t} \omega_i f(\tau_i),$$

and consider mean values in the spatial direction. This yields

$$\frac{2}{\Delta t}(u^*_{N_t j} \delta_{iN_x} - u^*_{1j} \delta_{i1} - \sum_{l=1}^{N_t} \omega_l \ell_i'(\tau_l) u_{lj}) + \frac{a\omega_i}{\Delta x}(u^*_{i,j} - u^*_{i,j-1}) = 0, \; i = 1, \ldots, N_t, j = 2, \ldots, N_x,$$

where we have replaced the boundary terms by numerical flux functions u^*. Here, we choose the upwind flux in the spatial and the temporal direction.

We can combine the temporal and spatial discretizations with a tensor product ansatz. Let us denote the spatial discretization operators by the index ξ and the temporal discretization operators by τ. With the temporal DG-SEM operators

$$\mathbf{M}_\tau = \frac{\Delta t}{2} \begin{pmatrix} \omega_1 & & \\ & \ddots & \\ & & \omega_{N_t} \end{pmatrix} \in \mathbb{R}^{N_t \times N_t}, \; \mathbf{C}_\tau = \begin{pmatrix} 0 & & 1 \\ & \ddots & \\ & & 0 \end{pmatrix} \in \mathbb{R}^{N_t \times N_t}, \; \mathbf{E}_{N_t} = \begin{pmatrix} 0 & & & \\ & \ddots & & \\ & & 0 & \\ & & & 1 \end{pmatrix} \in \mathbb{R}^{N_t \times N_t},$$

$$\mathbf{K}_\tau = \mathbf{E}_{N_t} - \mathbf{D}_\tau^T \mathbf{M}_\tau \in \mathbb{R}^{N_t \times N_t}, \; (\mathbf{D}_\tau)_{ij} = \frac{2}{\Delta t} \ell_j'(\tau_i), \; i, j = 1 \ldots, N_t, \tag{2}$$

and the spatial operator

$$\mathbf{K}_\xi = \frac{a}{\Delta x} \begin{pmatrix} 1 & & & -1 \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{pmatrix} \in \mathbb{R}^{N_x \times N_x}, \; \mathbf{I}_\xi \in \mathbb{R}^{N_x \times N_x}, \tag{3}$$

the following linear space-time system has to be solved on each so-called space-time slab n:

$$(\mathbf{I}_\xi \otimes \mathbf{K}_\tau + \mathbf{K}_\xi \otimes \mathbf{M}_\tau)\mathbf{u}^{n+1} = (\mathbf{I}_\xi \otimes \mathbf{C}_\tau)\mathbf{u}^n, \tag{4}$$

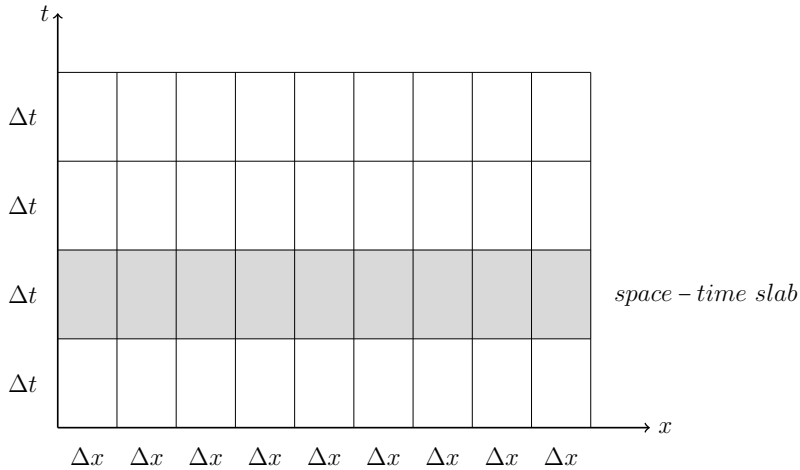


Figure 1: Equidistant space-time grid in one spatial dimension.

Here, the spatial matrices correspond to the whole domain in space while the temporal matrices correspond to one DG element in time.

An example for an equidistant space-time grid can be seen in Figure 1. Then all N_x spatial elements and one temporal element n represent one space-time slab $n = 1, \dots, N$, as highlighted gray in the figure.

Let us write the vector of unknowns for the space-time problem as $\underline{\mathbf{u}} = [\mathbf{u}^1, \dots, \mathbf{u}^N]^T \in \mathbb{R}^{NN_xN_t}$. The components of $\mathbf{u}^n \in \mathbb{R}^{N_xN_t}$ are given by $\mathbf{u}^n_{j,k} \in \mathbb{R}$, where n denotes the space-time slab, j is the index for the unknowns in space and k the index for the unknowns in one time element, i.e. $\mathbf{u}^n_j \in \mathbb{R}^{N_t}$. This index notation is used throughout this paper for vectors in the space $\mathbb{R}^{NN_xN_t}$.

The full space-time system for N space-time slabs on $[0, T]$ can then be written in block form

$$\underline{\mathbf{L}}_{\tau,\xi}\underline{\mathbf{u}} := \begin{pmatrix} \mathbf{A}_{\tau,\xi} & & & \mathbf{B}_{\tau,\xi} \\ \mathbf{B}_{\tau,\xi} & \mathbf{A}_{\tau,\xi} & & \\ & \ddots & \ddots & \\ & & \mathbf{B}_{\tau,\xi} & \mathbf{A}_{\tau,\xi} \end{pmatrix} \begin{pmatrix} \mathbf{u}^1 \\ \mathbf{u}^2 \\ \vdots \\ \mathbf{u}^N \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix} =: \underline{\mathbf{b}} \tag{5}$$

with

$$\mathbf{A}_{\tau,\xi} := \mathbf{I}_\xi \otimes \mathbf{K}_\tau + \mathbf{K}_\xi \otimes \mathbf{M}_\tau \in \mathbb{R}^{N_xN_t \times N_xN_t}, \tag{6}$$

$$\mathbf{B}_{\tau,\xi} := -\mathbf{I}_\xi \otimes \mathbf{C}_\tau \in \mathbb{R}^{N_xN_t \times N_xN_t}. \tag{7}$$

and the space-time operator $\underline{\mathbf{L}}_{\tau,\xi} \in \mathbb{R}^{NN_tN_x \times NN_tN_x}$ and space-time vectors $\underline{\mathbf{u}}, \underline{\mathbf{b}} \in \mathbb{R}^{NN_xN_t}$.

2.2 Multigrid Solver

In the next step, the linear system (5) has to be solved. One method is to apply a forward substitution w.r.t. the time blocks. This involves inverting the diagonal blocks in each

time step and results in a sequential process. Instead, we use a space-time multigrid method to solve (5). Multigrid algorithms consist of three main components: a smoother, intergrid transfer operators, i.e. prolongation and restriction, and a coarse grid solver. It is of advantage to study the intergrid transfer operators in space and time separately due to the special causality principle in the temporal direction. We consider a geometric multigrid method in space and time. We choose block Jacobi smoothers with blocks corresponding to one space-time slab since it has been shown in the case of space-time multigrid methods that pointwise or line-smoothers, when not chosen carefully, can result in divergent methods [16, 24, 26]. The choice of these blocks follows from the block form of the full space-time system (5).

Let $\mathbf{\Omega}_\ell \subset \mathbb{R}^2$ be the grid on level $\ell = 0, \dots, M$ with $\ell = 0$ the coarsest and $\ell = M$ the finest level. We denote the number of time slabs on multigrid level ℓ by $N_{\ell_t} \in \mathbb{N}$ and the number of spatial elements by $N_{\ell_x} \in \mathbb{N}$. In consequence, on each space-time grid level ℓ the system matrix $\underline{\mathbf{L}}_{\tau_\ell,\xi_\ell}$ is defined by (3) with N_{ℓ_x} volumes and the space-time system (5) with N_{ℓ_t} time steps.

For the temporal component, restriction and prolongation matrices are defined using $\mathcal{L}_2$ projections,

$$\begin{aligned} \mathbf{R}_{\ell_t-1}^{\ell_t} &:= \begin{pmatrix} \mathbf{R}_1 & \mathbf{R}_2 & & & & \\ & & \mathbf{R}_1 & \mathbf{R}_2 & & \\ & & & \ddots & \ddots & \\ & & & & \mathbf{R}_1 & \mathbf{R}_2 \end{pmatrix} \in \mathbb{R}^{N_t N_{\ell_t-1} \times N_t N_{\ell_t}}, \\ \mathbf{P}_{\ell_t}^{\ell_t-1} &:= (\mathbf{R}_{\ell_t-1}^{\ell_t})^T \in \mathbb{R}^{N_t N_{\ell_t} \times N_t N_{\ell_t-1}}, \end{aligned} \tag{8}$$

with local prolongation matrices $\mathbf{R}_1^T := \mathbf{M}_{\tau_\ell}^{-1}\widetilde{\mathbf{M}}_{\tau_\ell}^1$ and $\mathbf{R}_2^T := \mathbf{M}_{\tau_\ell}^{-1}\widetilde{\mathbf{M}}_{\tau_\ell}^2$, see [13], [23]. For basis functions $\{\ell_k\}_{k=1}^{N_t} \subset \mathbb{P}^{p_t}(0,\tau_\ell)$ on the fine grid and $\{\tilde{\ell}_k\}_{k=1}^{N_t} \subset \mathbb{P}^{p_t}(0,2\tau_\ell)$ on the coarse grid, the local projection matrices from the coarse to the fine grid are defined by

$$\widetilde{M}_{\tau_\ell}^1(k,l) := \int_0^{\tau_\ell} \tilde{\ell}_l(t)\ell_k(t)dt, \quad \widetilde{M}_{\tau_\ell}^2(k,l) := \int_{\tau_\ell}^{2\tau_\ell} \tilde{\ell}_l(t)\ell_k(t-\tau)dt, \; k,l = 1,\dots,N_t. \tag{9}$$

Restriction and prolongation matrices in space are given by agglomeration

$$\begin{aligned} \mathbf{R}_{\ell_x-1}^{\ell_x} &:= \frac{1}{2}\begin{pmatrix} 1 & 1 & & & & \\ & & 1 & 1 & & \\ & & & \ddots & \ddots & \\ & & & & 1 & 1 \end{pmatrix} \in \mathbb{R}^{N_{\ell_x-1} \times N_{\ell_x}}, \\ \mathbf{P}_{\ell_x}^{\ell_x-1} &:= (2\mathbf{R}_{\ell_x-1}^{\ell_x})^T \in \mathbb{R}^{N_{\ell_x} \times N_{\ell_x-1}}. \end{aligned} \tag{10}$$

We study two different coarsening strategies: coarsening in space and time, referred to as full-coarsening and denoted by index f, and coarsening in the temporal direction only, which we refer to as semi-coarsening with index s. Since we are especially interested in the efficiency of the multigrid method in time, we study a semi-coarsening strategy in this direction.

For the space-time system, restriction and prolongation operators can then be defined with a tensor product

$$(\underline{\mathbf{R}}_{\ell-1}^{\ell})^s := \mathbf{I}_{N_{\ell_x}} \otimes \mathbf{R}_{\ell_t-1}^{\ell_t}, \; (\underline{\mathbf{R}}_{\ell-1}^{\ell})^f := \mathbf{R}_{\ell_x-1}^{\ell_x} \otimes \mathbf{R}_{\ell_t-1}^{\ell_t}, \tag{11}$$

$$(\underline{\mathbf{P}}_{\ell}^{\ell-1})^s := \mathbf{I}_{N_{\ell_x}} \otimes \mathbf{P}_{\ell_t}^{\ell_t-1}, \; (\underline{\mathbf{P}}_{\ell}^{\ell-1})^f := \mathbf{P}_{\ell_x}^{\ell_x-1} \otimes \mathbf{P}_{\ell_t}^{\ell_t-1}. \tag{12}$$

As smoother we choose a damped block Jacobi method

$$\underline{\mathbf{u}}^{(k+1)} = \omega_t(\underline{\mathbf{D}}_{\tau_\ell,\xi_\ell})^{-1}\underline{\mathbf{b}} + (\underline{\mathbf{I}}_{\tau_\ell,\xi_\ell} - \omega_t(\underline{\mathbf{D}}_{\tau_\ell,\xi_\ell})^{-1}\underline{\mathbf{L}}_{\tau_\ell,\xi_\ell})\underline{\mathbf{u}}^{(k)}, \tag{13}$$

with damping factor ω_t and block diagonal matrix

$$\underline{\mathbf{D}}_{\tau_\ell,\xi_\ell} := \operatorname{diag}([\mathbf{A}_{\tau_\ell,\xi_\ell}, \dots, \mathbf{A}_{\tau_\ell,\xi_\ell}]). \tag{14}$$

Here, the blocks correspond to a space-time slab on the given grid level due to the block form of the full space-time system (5). The block Jacobi iteration matrix reads

$$\underline{\mathbf{S}}_{\tau_\ell,\xi_\ell} := \underline{\mathbf{I}} - \omega_t(\underline{\mathbf{D}}_{\tau_\ell,\xi_\ell})^{-1}\underline{\mathbf{L}}_{\tau_\ell,\xi_\ell}. \tag{15}$$

With this, the iteration matrices for a two-grid V-cycle with ν_1 pre- and ν_2 post-smoothing steps on the fine grid are given by

$$\underline{\mathbf{M}}^s_{\tau_\ell,\xi_\ell} := \underline{\mathbf{S}}^{\nu_2}_{\tau_\ell,\xi_\ell}\left[\underline{\mathbf{I}} - (\underline{\mathbf{P}}^{\ell-1}_\ell)^s(\underline{\mathbf{L}}_{2\tau_\ell,\xi_\ell})^{-1}(\underline{\mathbf{R}}^\ell_{\ell-1})^s\underline{\mathbf{L}}_{\tau_\ell,\xi_\ell}\right]\underline{\mathbf{S}}^{\nu_1}_{\tau_\ell,\xi_\ell}, \tag{16}$$

$$\underline{\mathbf{M}}^f_{\tau_\ell,\xi_\ell} := \underline{\mathbf{S}}^{\nu_2}_{\tau_\ell,\xi_\ell}\left[\underline{\mathbf{I}} - (\underline{\mathbf{P}}^{\ell-1}_\ell)^f(\underline{\mathbf{L}}_{2\tau_\ell,2\xi_\ell})^{-1}(\underline{\mathbf{R}}^\ell_{\ell-1})^f\underline{\mathbf{L}}_{\tau_\ell,\xi_\ell}\right]\underline{\mathbf{S}}^{\nu_1}_{\tau_\ell,\xi_\ell}, \tag{17}$$

for semi-coarsening and full-coarsening respectively. Here, it is assumed that the systems are solved exactly on the coarse grid.

3 Preliminaries

In this section we discuss some preliminaries which are needed for the local Fourier analysis presented in the following sections.

3.1 DG-SEM and Lobatto IIIC Methods

We start with the temporal DG-SEM discretization (2). Using an upwind flux, which is a flux to fulfill the temporal causality principle, the DG-SEM discretization in time for the linear test equation

$$u_t + \lambda u = 0,\ t \in [0,T],\ u(0) = u_0,\ \lambda \geq 0, \tag{18}$$

reads

$$(\mathbf{K}_\tau + \lambda\mathbf{M}_\tau)\mathbf{u}^{n+1} = \mathbf{C}_\tau\mathbf{u}^n. \tag{19}$$

One can show that the scheme is equivalent to a specific Runge-Kutta time-stepping method:

Theorem 3.1 ([3, 17])**.** The DG-SEM (19) with $p_t + 1$ Legendre-Gauss-Lobatto nodes is equivalent to the $(p_t + 1)$-stage Runge-Kutta scheme Lobatto IIIC.

Here, equivalence is referred to the solution of the unknowns in the end of each element assuming the resulting systems are solved exactly.

With the help of Theorem 3.1 some stability results for the temporal discretization can be drawn.

Theorem 3.2 ([18, 22])**.** For $s \in \mathbb{N}$ the s-stage Lobatto IIIC scheme is of order $2s-1$ and its stability function $R(z)$ is given by the $(s-2, s)$-Padé approximation of the exponential function e^z. The method is L-stable and furthermore algebraically stable, thus B-stable and A-stable.

Corollary 3.3. The stability function $R(z)$ of the DG-SEM (19) with $p_t+1 \in \mathbb{N}$ Legendre-Gauss-Lobatto nodes, $p_t \geq 1$, is given by the (p_t-1, p_t+1)-Padé approximation to the exponential function e^z.

Proof. By Theorem 3.1 DG-SEM is equivalent to the Lobatto IIIC method. Thus, both methods have the same stability function $R(z)$. By Theorem 3.2 the stability function is given by the (p_t-1, p_t+1)-Padé approximation to the exponential function e^z. □

The Padé approximant for the exponential function can be calculated directly.

Theorem 3.4 ([18])**.** The (k, m)-*Padé approximant*

$$r_{km}(z) = \frac{p_{km}(z)}{q_{km}(z)}$$

of the exponential function e^z is given by

$$\begin{aligned} p_{km}(z) &= 1 + \sum_{j=1}^{k} \frac{(k+m-j)!\,k!}{(k+m)!\,(k-j)!} \cdot \frac{z^j}{j!}, \\ q_{km}(z) &= 1 + \sum_{j=1}^{m} \frac{(k+m-j)!\,m!}{(k+m)!\,(m-j)!} \cdot \frac{(-z)^j}{j!}. \end{aligned}$$

With the help of the stability function R, the eigenvalues of the discretization matrix (19) can be calculated.

Lemma 3.5 ([13])**.** For $\lambda \in \mathbb{C}$ the spectrum of the matrix $(\mathbf{K}_\tau + \lambda \mathbf{M}_\tau)^{-1}\mathbf{C}_\tau \in \mathbb{C}^{N_t \times N_t}$ is given by

$$\sigma((\mathbf{K}_\tau + \lambda \mathbf{M}_\tau)^{-1}\mathbf{C}_\tau) = \{0, R(-\lambda\tau)\}$$

where $R(z)$ is the stability function of the DG time stepping scheme, see Corollary 3.3.

These results are used for the smoothing analysis in section 5.

3.2 Definitions and Notation for the Local Fourier Analysis

In this section we present the basic tools needed to perform a local Fourier analysis for the multigrid solver as presented in section 2.2. For a more detailed description of this technique we refer to [2, 15].

First we define the Fourier modes and frequencies.

Definition 3.6 ([34])**.** The function

$$\boldsymbol{\varphi}(\theta_k) := [\varphi_1(\theta_k), \ldots, \varphi_N(\theta_k)]^T, \ \varphi_j(\theta_k) := e^{\mathrm{i}j\theta_k}, \ j = 1, \ldots, N, \ N \in \mathbb{N},$$

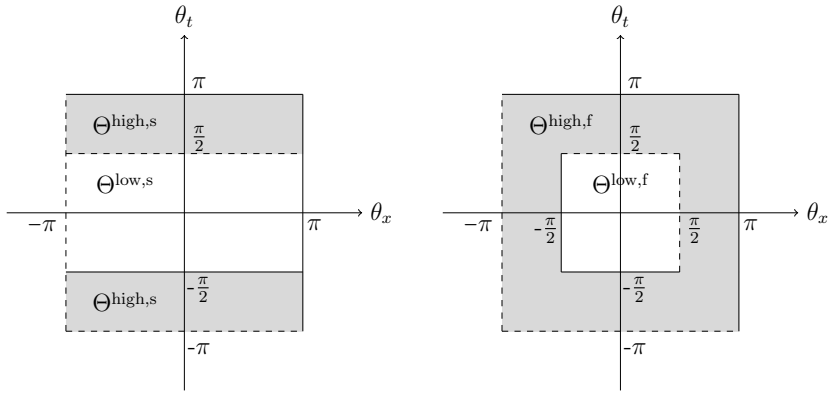


Figure 2: Low and high frequencies for semi coarsening (left) and full coarsening (right)

is called *Fourier mode with frequencies*

$$\theta_k \in \Theta := \left\{ \frac{2k\pi}{N} : k = 1 - \frac{N}{2}, \dots, \frac{N}{2} \right\} \subset (-\pi, \pi].$$

The frequencies can be separated into *low and high frequencies*

$$\Theta^{low} := \Theta \cap \left(-\frac{\pi}{2}, \frac{\pi}{2} \right], \ \Theta^{high} := \Theta \cap \left(\left(-\pi, -\frac{\pi}{2} \right] \cup \left(\frac{\pi}{2}, \pi \right] \right].$$

In this paper we consider frequencies on a two-dimensional space-time domain. Given the set of space-time frequencies

$$\Theta_{\ell_x,\ell_t} := \{ (\theta_x, \theta_t) : \theta_x \in \Theta_{\ell_x}, \ \theta_t \in \Theta_{\ell_t} \} \subset (-\pi, \pi]^2,$$

low and high frequencies are defined as

$$\Theta^{high,s}_{\ell_x,\ell_t} := \Theta_{\ell_x,\ell_t} \setminus \Theta^{low,s}_{\ell_x,\ell_t} \quad \text{for} \quad \Theta^{low,s}_{\ell_x,\ell_t} := \Theta_{\ell_x,\ell_t} \cap (-\pi, \pi] \times \left(-\frac{\pi}{2}, \frac{\pi}{2} \right], \tag{20}$$

$$\Theta^{high,f}_{\ell_x,\ell_t} := \Theta_{\ell_x,\ell_t} \setminus \Theta^{low,f}_{\ell_x,\ell_t} \quad \text{for} \quad \Theta^{low,f}_{\ell_x,\ell_t} := \Theta_{\ell_x,\ell_t} \cap \left(-\frac{\pi}{2}, \frac{\pi}{2} \right]^2, \tag{21}$$

for semi-coarsening in time and full space-time coarsening respectively. In Figure 2 the ranges for the frequencies in the space-time domain are visualized for both coarsening strategies.

With this, the discrete Fourier transform reads:

Theorem 3.7 (Discrete Fourier transform [34])**.** Let $\underline{\mathbf{u}} \in \mathbb{R}^{N_t N_{\ell_x} N_{\ell_t}}$ for $N_t, N_{\ell_x}, N_{\ell_t} \in \mathbb{N}$, and assume that N_{ℓ_x} and N_{ℓ_t} are even. The vector $\underline{\mathbf{u}}$ can be represented as

$$\underline{\mathbf{u}} = \sum_{\theta_x \in \Theta_{\ell_x}} \sum_{\theta_t \in \Theta_{\ell_t}} \underline{\boldsymbol{\psi}}(\theta_x, \theta_t),$$

where $\underline{\boldsymbol{\psi}}(\theta_x, \theta_t) \in \mathbb{C}^{N_t N_{\ell_x} N_{\ell_t}}$ consists of the vectors

$$\boldsymbol{\psi}^n_j(\theta_x, \theta_t) := \mathbf{U}(\theta_x, \theta_t) \boldsymbol{\Phi}^n_j(\theta_x, \theta_t) \in \mathbb{C}^{N_t}, \ n = 1, \dots, N_{\ell_t}, \ j = 1, \dots, N_{\ell_x},$$

and the vector $\boldsymbol{\Phi}^n_j(\theta_x, \theta_t) \in \mathbb{C}^{N_t}$ has elements

$$\Phi^n_{j,l}(\theta_x, \theta_t) := \varphi_n(\theta_t) \varphi_j(\theta_x), \ l = 1, \dots, N_t.$$

Moreover, we define the coefficient matrix as

$$\mathbf{U}(\theta_x,\theta_t) := \operatorname{diag}(\hat{u}_1,\dots,\hat{u}_{N_t}) \in \mathbb{C}^{N_t\times N_t},$$

with coefficients

$$\hat{u}_l := \frac{1}{N_{\ell_x}}\frac{1}{N_{\ell_t}}\sum_{j=1}^{N_{\ell_x}}\sum_{n=1}^{N_{\ell_t}} u^n_{j,l}\varphi_j(-\theta_x)\varphi_n(-\theta_t),\ l=1,\dots,N_t.$$

Then, the linear space of Fourier modes can be defined.

Definition 3.8. Consider the frequencies $\theta_x \in \Theta_{\ell_x}$ and $\theta_t \in \Theta_{\ell_t}$ and the vector $\mathbf{\Phi}^n_j(\theta_x,\theta_t)$ as in Theorem 3.7. Then the *linear space of Fourier modes* with frequencies (θ_x,θ_t) is defined as

$$\begin{aligned}\Psi_{\ell_x,\ell_t}(\theta_x,\theta_t) &:= \operatorname{span}\{\underline{\mathbf{\Phi}}(\theta_x,\theta_t)\}\\ &:= \{\underline{\boldsymbol{\psi}}(\theta_x,\theta_t) \in \mathbb{C}^{N_t\cdot N_{\ell_x}\cdot N_{\ell_t}} : \boldsymbol{\psi}^n_j(\theta_x,\theta_t) := \mathbf{U}\mathbf{\Phi}^n_j(\theta_x,\theta_l),\\ &\qquad \text{for } n=1,\dots,N_{\ell_t},\ j=1,\dots,N_{\ell_x} \text{ and } \mathbf{U}\in\mathbb{C}^{N_t\times N_t}\}.\end{aligned}$$

With the result from the next theorem it suffices to consider low frequencies.

Theorem 3.9 ([26])**.** Let $\underline{\mathbf{u}} = [\mathbf{u}^1,\dots,\mathbf{u}^{N_{\ell_t}}]^T \in \mathbb{R}^{N_tN_{\ell_x}N_{\ell_t}}$ and assume that N_{ℓ_x} and N_{ℓ_t} are even numbers. Then the vector $\underline{\mathbf{u}}$ can be written as

$$\underline{\mathbf{u}} = \sum_{(\theta_x,\theta_t)\in\Theta^{low,f}_{\ell_x,\ell_t}} \left(\underline{\boldsymbol{\psi}}(\theta_x,\theta_t) + \underline{\boldsymbol{\psi}}(\gamma(\theta_x),\theta_t) + \underline{\boldsymbol{\psi}}(\theta_x,\gamma(\theta_t)) + \underline{\boldsymbol{\psi}}(\gamma(\theta_x),\gamma(\theta_t))\right),$$

with the shifting operator

$$\gamma(\theta) := \begin{cases}\theta+\pi, & \theta<0,\\ \theta-\pi, & \theta\geq 0,\end{cases}$$

and $\underline{\boldsymbol{\psi}}(\theta_x,\theta_t) \in \mathbb{C}^{N_tN_{\ell_x}N_{\ell_t}}$ as in Lemma 3.7.

Since $\underline{\boldsymbol{\psi}}(\theta_x,\theta_t)$ consists of the vectors $\boldsymbol{\psi}^n_j(\theta_x,\theta_t) = \mathbf{U}\mathbf{\Phi}^n_j(\theta_x,\theta_t)$, which itself build the vector $\underline{\mathbf{\Phi}}(\theta_x,\theta_t)$, the previous theorem implies that $\underline{\mathbf{u}} = [\mathbf{u}^1,\dots,\mathbf{u}^{N_{\ell_t}}]^T$ can be written as a linear combination of the low frequency vectors

$$\{\underline{\mathbf{\Phi}}(\theta_x,\theta_t),\underline{\mathbf{\Phi}}(\gamma(\theta_x),\theta_t),\underline{\mathbf{\Phi}}(\theta_x,\gamma(\theta_t)),\underline{\mathbf{\Phi}}(\gamma(\theta_x),\gamma(\theta_t))\}.$$

Thus, four fine grid modes get aliased to one coarse grid mode. In the following it suffices therefore to only consider low frequencies and use the shifting operator $\gamma : \Theta^{low}_\ell \to \Theta^{high}_\ell$. We can therefore define a new Fourier space, based on low frequencies only:

Definition 3.10. For $N_t, N_{\ell_x}, N_{\ell_t}$ consider the vector $\underline{\boldsymbol{\psi}}(\theta_x,\theta_t) \in \mathbb{C}^{N_tN_{\ell_x}N_{\ell_t}}$ for $(\theta_x,\theta_t) \in \Theta^{low,f}_{\ell_x,\ell_t}$ as in Lemma 3.7. The *linear space of low frequency harmonics* is defined as

$$\begin{aligned}\mathcal{E}_{\ell_x,\ell_t}(\theta_x,\theta_t) &:= \operatorname{span}\{\underline{\mathbf{\Phi}}(\theta_x,\theta_t),\underline{\mathbf{\Phi}}(\gamma(\theta_x),\theta_t),\underline{\mathbf{\Phi}}(\theta_x,\gamma(\theta_t)),\underline{\mathbf{\Phi}}(\gamma(\theta_x),\gamma(\theta_t))\}\\ &= \{\underline{\boldsymbol{\psi}}(\theta_x,\theta_t) \in \mathbb{C}^{N_t\cdot N_{\ell_x}\cdot N_{\ell_t}} : \boldsymbol{\psi}^n_j(\theta_x,\theta_t) = \mathbf{U}_1\mathbf{\Phi}^n_j(\theta_x,\theta_t)\\ &\quad + \mathbf{U}_2\mathbf{\Phi}^n_j(\gamma(\theta_x),\theta_t) + \mathbf{U}_3\mathbf{\Phi}^n_j(\theta_x,\gamma(\theta_t)) + \mathbf{U}_4\mathbf{\Phi}^n_j(\gamma(\theta_x),\gamma(\theta_t)),\\ &\quad n=1,\dots,N_{\ell_t},\ j=1,\dots,N_{\ell_x} \text{ and } \mathbf{U}_1,\mathbf{U}_2,\mathbf{U}_3,\mathbf{U}_4\in\mathbb{C}^{N_t\times N_t}\}.\end{aligned}$$

Moreover, we can define the Fourier space for the semi-coarsening strategy.

Definition 3.11 (Fourier space, semi-coarsening)**.** For $N_t, N_{\ell_x}, N_{\ell_{t-1}} \in \mathbb{N}$ and $(\theta_x, \theta_t) \in \Theta^{low,f}_{\ell_x,\ell_t}$ consider $\underline{\mathbf{\Phi}}(\theta_x, \theta_t) \in \mathbb{C}^{N_t N_{\ell_x} N_{\ell_{t-1}}}$ as in Lemma 3.7. We define the linear space with frequencies $(\theta_x, 2\theta_t)$ as

$$\begin{aligned}\Psi_{\ell_x,\ell_t-1}(\theta_x, 2\theta_t) &:= \operatorname{span}\{\underline{\mathbf{\Phi}}^{\ell_x,\ell_t-1}(\theta_x, 2\theta_t), \underline{\mathbf{\Phi}}^{\ell_x,\ell_t-1}(\gamma(\theta_x), 2\theta_t)\}\\ &= \{\underline{\boldsymbol{\psi}}^{\ell_x,\ell_t-1}(\theta_x, 2\theta_t) \in \mathbb{C}^{N_t,N_{\ell_x},N_{\ell_{t-1}}} :\\ &\quad \boldsymbol{\psi}_j^{n,\ell_x,\ell_t-1}(\theta_x, 2\theta_t) = \mathbf{U}_1 \mathbf{\Phi}_j^{n,\ell_x,\ell_t-1}(\theta_x, 2\theta_t)\\ &\quad + \mathbf{U}_2 \mathbf{\Phi}_j^{n,\ell_x,\ell_t-1}(\gamma(\theta_x), 2\theta_t) \text{ for } n = 1, \dots, N_{\ell_t} - 1,\\ &\quad j = 1, \dots, N_{\ell_x},\ \mathbf{U}_1, \mathbf{U}_2 \in \mathbb{C}^{N_t \times N_t}\}.\end{aligned}$$

One key property of the LFA is the shifting equality, which is used extensively when deriving the Fourier symbols of the operators in the next section.

Lemma 3.12 ([24])**.** Let $\theta_x \in \Theta_{\ell_x}$, $\theta_t \in \Theta_{\ell_t}$ and $\underline{\boldsymbol{\psi}}(\theta_x, \theta_t) \in \Psi_{\ell_x,\ell_t}(\theta_x, \theta_t)$. Then the following *shifting equalities* hold:

$$\begin{aligned}\boldsymbol{\psi}_j^{n-1}(\theta_x, \theta_t) &= e^{-\mathrm{i}\theta_t} \boldsymbol{\psi}_j^n(\theta_x, \theta_t),\ n = 2, \dots, N_{\ell_t}\\ \boldsymbol{\psi}_{j-1}^n(\theta_x, \theta_t) &= e^{-\mathrm{i}\theta_x} \boldsymbol{\psi}_j^n(\theta_x, \theta_t),\ j = 2, \dots, N_{\ell_x}.\end{aligned}$$

4 Fourier Symbols

The first step of the local Fourier analysis is to derive the Fourier symbols of all operators in the MG iteration (17) and (16), i.e. of the system matrix, smoother, restriction and prolongation. These symbols are also referred to as formal eigenvalues [26] since they are derived by multiplying the operators by the vector $\underline{\boldsymbol{\psi}}(\theta_x, \theta_t)$ from Theorem 3.7.

We start with the Fourier symbol of the system matrix (5).

Lemma 4.1 (Fourier symbol of $\underline{\mathbf{L}}_{\tau_\ell,\xi_\ell}$)**.** For $\theta_x \in \Theta_{\ell_x}$ and $\theta_t \in \Theta_{\ell_t}$ we consider the vector $\underline{\boldsymbol{\psi}}(\theta_x, \theta_t) \in \Psi_{\ell_x,\ell_t}(\theta_x, \theta_t)$. For

$$\boldsymbol{\mathcal{L}}_{\tau_\ell,\xi_\ell}(\theta_x, \theta_t) := -e^{-\mathrm{i}\theta_t} \mathbf{C}_{\tau_\ell} + \mathbf{K}_{\tau_\ell} + \frac{a}{\Delta x}(-e^{-\mathrm{i}\theta_x} + 1)\mathbf{M}_{\tau_\ell} \in \mathbb{C}^{N_t \times N_t}$$

it holds that

$$(\underline{\mathbf{L}}_{\tau_\ell,\xi_\ell} \underline{\boldsymbol{\psi}}(\theta_x, \theta_t))_j^n = \boldsymbol{\mathcal{L}}_{\tau_\ell,\xi_\ell}(\theta_x, \theta_t) \boldsymbol{\psi}_j^n(\theta_x, \theta_t),$$

for $n = 2, \dots, N_{\ell_t},\ j = 2, \dots, N_{\ell_x} - 1$ and we call $\boldsymbol{\mathcal{L}}_{\tau_\ell,\xi_\ell}(\theta_x, \theta_t) \in \mathbb{C}^{N_t \times N_t}$ the Fourier symbol of $\underline{\mathbf{L}}_{\tau_\ell,\xi_\ell} \in \mathbb{C}^{N_{\ell_t} N_t N_{\ell_x} \times N_{\ell_t} N_t N_{\ell_x}}$.

Proof. With Lemma 3.12 we get for $\underline{\boldsymbol{\psi}}(\theta_x, \theta_t) \in \Psi_{\ell_x,\ell_t}(\theta_x, \theta_t)$

$$\begin{aligned}(\underline{\mathbf{L}}_{\tau_\ell,\xi_\ell} \underline{\boldsymbol{\psi}}(\theta_x, \theta_t))^n &= \mathbf{B}_{\tau_\ell,\xi_\ell} \boldsymbol{\psi}^{n-1}(\theta_x, \theta_t) + \mathbf{A}_{\tau_\ell,\xi_\ell} \boldsymbol{\psi}^n(\theta_x, \theta_t)\\ &= (e^{-\mathrm{i}\theta_t} \mathbf{B}_{\tau_\ell,\xi_\ell} + \mathbf{A}_{\tau_\ell,\xi_\ell}) \boldsymbol{\psi}^n(\theta_x, \theta_t),\ n = 2, \dots, N_{\ell_t}.\end{aligned}$$

Thus, we have to study the product of $\mathbf{A}_{\tau_\ell,\xi_\ell} = \mathbf{I}_{\xi_\ell} \otimes \mathbf{K}_{\tau_\ell} + \mathbf{K}_{\xi_\ell} \otimes \mathbf{M}_{\tau_\ell}$ and $\mathbf{B}_{\tau_\ell,\xi_\ell} = -\mathbf{I}_{\xi_\ell} \otimes \mathbf{C}_{\tau_\ell}$ with the vector $\boldsymbol{\psi}^n(\theta_x,\theta_t)$:

$$\begin{aligned}
(\mathbf{A}_{\tau_\ell,\xi_\ell}\boldsymbol{\psi}^n(\theta_x,\theta_t))_{j,l} &= \sum_{i=1}^{N_{\ell_x}}\sum_{k=1}^{N_t} I_{\xi_\ell}(j,i)K_{\tau_\ell}(l,k)\psi^n_{i,k}(\theta_x,\theta_t) \\
&\quad + \sum_{i=1}^{N_{h_\ell}}\sum_{k=1}^{N_t} K_{\xi_\ell}(j,i)M_{\tau_\ell}(l,k)\psi^n_{i,k}(\theta_x,\theta_t) \\
&= (\mathbf{K}_{\tau_\ell}\boldsymbol{\psi}^n_j(\theta_x,\theta_t))_l + \frac{a}{\Delta x}(-e^{-\mathrm{i}\theta_x}+1)(\mathbf{M}_{\tau_\ell}\boldsymbol{\psi}^n_j(\theta_x,\theta_t))_l \\
&= (\mathbf{K}_{\tau_\ell} + \frac{a}{\Delta x}(-e^{-\mathrm{i}\theta_x}+1)\mathbf{M}_{\tau_\ell})\boldsymbol{\psi}^n_j(\theta_x,\theta_t))_l,
\end{aligned}$$

and

$$\begin{aligned}
(\mathbf{B}_{\tau_\ell,\xi_\ell}\boldsymbol{\psi}^n(\theta_x,\theta_t))_{j,l} &= -\sum_{i=1}^{N_{\ell_x}}\sum_{k=1}^{N_t} I_{\xi_\ell}(j,i)C_{\tau_\ell}(l,k)\psi^n_{i,k}(\theta_x,\theta_t) \\
&= -\sum_{k=1}^{N_t} C_{\tau_\ell}(l,k)\psi^n_{j,k}(\theta_x,\theta_t) = -(\mathbf{C}_{\tau_\ell}\boldsymbol{\psi}^n_j(\theta_x,\theta_t))_l,
\end{aligned}$$

for $j = 2,\dots,N_{\ell_x}-1$ and $l = 1,\dots,N_t$. Then

$$(\underline{\mathbf{L}}_{\tau_\ell,\xi_\ell}\underline{\boldsymbol{\psi}}(\theta_x,\theta_t))^n_j = (-e^{-\mathrm{i}\theta_t}\mathbf{C}_{\tau_\ell} + \mathbf{K}_{\tau_\ell} + \frac{a}{\Delta x}(-e^{-\mathrm{i}\theta_x}+1)\mathbf{M}_{\tau_\ell})\boldsymbol{\psi}^n_j(\theta_x,\theta_t),$$

and thus

$$\boldsymbol{\mathcal{L}}_{\tau_\ell,\xi_\ell} = -e^{-\mathrm{i}\theta_t}\mathbf{C}_{\tau_\ell} + \mathbf{K}_{\tau_\ell} + \frac{a}{\Delta x}(-e^{-\mathrm{i}\theta_x}+1)\mathbf{M}_{\tau_\ell} \in \mathbb{C}^{N_t\times N_t}.$$

□

With this result we can derive the symbol of the the block Jacobi smother (13).

Lemma 4.2 (Fourier symbol of $\underline{\mathbf{S}}_{\tau_\ell,\xi_\ell}$)**.** For $\theta_x \in \Theta_{\ell_x}$ and $\theta_t \in \Theta_{\ell_t}$ we consider the vector $\underline{\boldsymbol{\psi}}(\theta_x,\theta_t) \in \Psi_{\ell_x,\ell_t}(\theta_x,\theta_t)$. For

$$\boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t) := (1-\omega_t)\mathbf{I}_{N_t} + \omega_t e^{-\mathrm{i}\theta_t}(\mathbf{K}_{\tau_\ell} + \frac{a}{\Delta x}(-e^{-\mathrm{i}\theta_x}+1)\mathbf{M}_{\tau_\ell})^{-1}\mathbf{C}_{\tau_\ell} \in \mathbb{C}^{N_t\times N_t}$$

it holds that

$$(\underline{\mathbf{S}}_{\tau_\ell,\xi_\ell}\underline{\boldsymbol{\psi}}(\theta_x,\theta_t))^n_j = \boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t)\boldsymbol{\psi}^n_j(\theta_x,\theta_t)$$

for $n = 1,\dots,N_{\ell_t}$, $j = 1,\dots,N_{\ell_x}$ and we call $\boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t) \in \mathbb{C}^{N_t\times N_t}$ the Fourier symbol of $\underline{\mathbf{S}}_{\tau_\ell,\xi_\ell} \in \mathbb{C}^{N_{\ell_t}N_tN_{\ell_x}\times N_{\ell_t}N_tN_{\ell_x}}$.

Proof. Let $\underline{\boldsymbol{\psi}}(\theta_x,\theta_t) \in \Psi_{\ell_x,\ell_t}(\theta_x,\theta_t)$. For fixed $n = 1,\dots,N_{\ell_t}$ and $j = 1,\dots,N_{\ell_x}$ it holds

$$\begin{aligned}
(\underline{\mathbf{S}}_{\tau_\ell,\xi_\ell}\underline{\boldsymbol{\psi}}(\theta_x,\theta_t))^n_j &= ((\underline{\mathbf{I}}_{N_tN_{\ell_x}N_{\ell_t}} - \omega_t(\underline{\mathbf{D}}_{\tau_\ell,\xi_\ell})^{-1}\underline{\mathbf{L}}_{\tau_\ell,\xi_\ell})\underline{\boldsymbol{\psi}}(\theta_x,\theta_t))^n_j \\
&= (\mathbf{I}_{N_t} - \omega_t(\hat{\mathbf{A}}_{\tau_\ell,\xi_\ell}(\theta_x))^{-1}\boldsymbol{\mathcal{L}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t))\boldsymbol{\psi}^n_j(\theta_x,\theta_t) \\
&= \boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t)\boldsymbol{\psi}^n_j(\theta_x,\theta_t),
\end{aligned}$$

with $\hat{\mathbf{A}}_{\tau_\ell,\xi_\ell}(\theta_x) := \mathbf{K}_{\tau_\ell} + \frac{a}{\Delta x}(-e^{-\mathrm{i}\theta_x}+1)\mathbf{M}_{\tau_\ell}$ derived as in the previous proof. Moreover,

$$\begin{aligned}(\hat{\mathbf{A}}_{\tau_\ell,\xi_\ell}(\theta_x))^{-1}\boldsymbol{\mathcal{L}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t) &= (\mathbf{K}_{\tau_\ell} + \frac{a}{\Delta x}(-e^{-\mathrm{i}\theta_x}+1)\mathbf{M}_{\tau_\ell})^{-1} \\ &\quad (-e^{-\mathrm{i}\theta_t}\mathbf{C}_{\tau_\ell} + \mathbf{K}_{\tau_\ell} + \frac{a}{\Delta x}(-e^{-\mathrm{i}\theta_x}+1)\mathbf{M}_{\tau_\ell}) \\ &= \mathbf{I}_{N_t} - e^{-\mathrm{i}\theta_t}(\mathbf{K}_{\tau_\ell} + \frac{a}{\Delta x}(-e^{-\mathrm{i}\theta_x}+1)\mathbf{M}_{\tau_\ell})^{-1}\mathbf{C}_{\tau_\ell}.\end{aligned}$$

Thus,

$$\begin{aligned}\boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t) &= \mathbf{I}_{N_t} - \omega_t(\mathbf{I}_{N_t} - e^{-\mathrm{i}\theta_t}(\mathbf{K}_{\tau_\ell} + \frac{a}{\Delta x}(-e^{-\mathrm{i}\theta_x}+1)\mathbf{M}_{\tau_\ell})^{-1}\mathbf{C}_{\tau_\ell}) \\ &= (1-\omega_t)\mathbf{I}_{N_t} + \omega_t e^{-\mathrm{i}\theta_t}(\mathbf{K}_{\tau_\ell} + \frac{a}{\Delta x}(-e^{-\mathrm{i}\theta_x}+1)\mathbf{M}_{\tau_\ell})^{-1}\mathbf{C}_{\tau_\ell} \in \mathbb{C}^{N_t\times N_t}.\end{aligned}$$

□

With Theorems 3.7 and 3.9 and Lemma 4.1 we get for the system matrix $\underline{\mathbf{L}}_{\tau_\ell,\xi_\ell}$ and $(\theta_x,\theta_t) \in \Theta^{low,f}_{\ell_x,\ell_t}$ the following mapping property:

$$\begin{aligned}&\underline{\mathbf{L}}_{\tau_\ell,\xi_\ell} : \mathcal{E}_{\ell_x,\ell_t}(\theta_x,\theta_t) \to \mathcal{E}_{\ell_x,\ell_t}(\theta_x,\theta_t), \\ &\begin{pmatrix}\mathbf{U}_1\\ \mathbf{U}_2\\ \mathbf{U}_3\\ \mathbf{U}_4\end{pmatrix} \to \begin{pmatrix}\boldsymbol{\mathcal{L}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t)\mathbf{U}_1\\ \boldsymbol{\mathcal{L}}_{\tau_\ell,\xi_\ell}(\gamma(\theta_x),\theta_t)\mathbf{U}_2\\ \boldsymbol{\mathcal{L}}_{\tau_\ell,\xi_\ell}(\theta_x,\gamma(\theta_t))\mathbf{U}_3\\ \boldsymbol{\mathcal{L}}_{\tau_\ell,\xi_\ell}(\gamma(\theta_x),\gamma(\theta_t))\mathbf{U}_4\end{pmatrix} =: \widetilde{\boldsymbol{\mathcal{L}}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t)\begin{pmatrix}\mathbf{U}_1\\ \mathbf{U}_2\\ \mathbf{U}_3\\ \mathbf{U}_4\end{pmatrix},\end{aligned} \tag{22}$$

with a block diagonal matrix $\widetilde{\boldsymbol{\mathcal{L}}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t) \in \mathbb{C}^{4N_t\times 4N_t}$, $\boldsymbol{\mathcal{L}}_{\tau_\ell,\xi_\ell} \in \mathbb{C}^{N_t\times N_t}$ as defined in Lemma 4.1 and the space of low frequencies $\mathcal{E}_{\ell_x,\ell_t}$ as defined in 3.10. Accordingly, we obtain with Lemma 4.2 for the smoother $\underline{\mathbf{S}}_{\tau_\ell,\xi_\ell}$ and $(\theta_x,\theta_t) \in \Theta^{low,f}_{\ell_x,\ell_t}$

$$\begin{aligned}&\underline{\mathbf{S}}_{\tau_\ell,\xi_\ell} : \mathcal{E}_{\ell_x,\ell_t}(\theta_x,\theta_t) \to \mathcal{E}_{\ell_x,\ell_t}(\theta_x,\theta_t), \\ &\begin{pmatrix}\mathbf{U}_1\\ \mathbf{U}_2\\ \mathbf{U}_3\\ \mathbf{U}_4\end{pmatrix} \to \begin{pmatrix}\boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t)\mathbf{U}_1\\ \boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\gamma(\theta_x),\theta_t)\mathbf{U}_2\\ \boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\theta_x,\gamma(\theta_t))\mathbf{U}_3\\ \boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\gamma(\theta_x),\gamma(\theta_t))\mathbf{U}_4\end{pmatrix} =: \widetilde{\boldsymbol{\mathcal{S}}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t)\begin{pmatrix}\mathbf{U}_1\\ \mathbf{U}_2\\ \mathbf{U}_3\\ \mathbf{U}_4\end{pmatrix},\end{aligned} \tag{23}$$

with a block diagonal matrix $\widetilde{\boldsymbol{\mathcal{S}}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t) \in \mathbb{C}^{4N_t\times 4N_t}$ and $\boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell} \in \mathbb{C}^{N_t\times N_t}$ as defined in Lemma 4.2.

Next, we derive the Fourier symbols of the restriction and prolongation operators.

Lemma 4.3 (Fourier symbols of spatial prolongation and restriction)**.** Consider the spatial restriction and prolongation operators $\mathbf{R}^{\ell_x}_{\ell_x-1} \in \mathbb{C}^{N_{\ell_x-1}\times N_{\ell_x}}$ and $\mathbf{P}^{\ell_x-1}_{\ell_x} \in \mathbb{C}^{N_{\ell_x}\times N_{\ell_x-1}}$ defined in (10). Let $\boldsymbol{\varphi}^{\ell_x}(\theta_x) \in \mathbb{C}^{N_{\ell_x}}$ be a fine Fourier mode and $\boldsymbol{\varphi}^{\ell_x-1}(2\theta_x) \in \mathbb{C}^{N_{\ell_x-1}}$ a coarse Fourier mode for $\theta_x \in \Theta^{low}_{\ell_x}$. Then for $\mathcal{R}^{\ell_x}_{\ell_x-1}(\theta_x) := \frac{1}{2}(e^{-\mathrm{i}\theta_x}+1)$ it holds

$$(\mathbf{R}^{\ell_x}_{\ell_x-1}\boldsymbol{\varphi}^{\ell_x}(\theta_x))_j = \mathcal{R}^{\ell_x}_{\ell_x-1}(\theta_x)\varphi^{\ell_x-1}_j(2\theta_x),\ j = 1,\dots,N_{\ell_x-1},$$

and we call $\mathcal{R}^{\ell_x}_{\ell_x-1}(\theta_x) \in \mathbb{C}$ the Fourier symbol of the restriction operator in space.
For $\mathcal{P}^{\ell_x-1}_{\ell_x}(\theta_x) := \frac{1}{2}(e^{\mathrm{i}\theta_x}+1)$ it holds

$$(\mathbf{P}^{\ell_x-1}_{\ell_x}\boldsymbol{\varphi}^{\ell_x-1}(2\theta_x))_i = \mathcal{P}^{\ell_x-1}_{\ell_x}(\theta_x)\varphi^{\ell_x}_i(\theta_x) + \mathcal{P}^{\ell_x-1}_{\ell_x}(\gamma(\theta_x))\varphi^{\ell_x}_i(\gamma(\theta_x)),\ i = 1,\dots,N_{\ell_x}$$

and we call $\mathcal{P}^{\ell_x-1}_{\ell_x}(\theta_x) \in \mathbb{C}$ the Fourier symbol of the prolongation operator in space.

Proof. For the restriction operator we get

$$\begin{aligned}(\mathbf{R}^{\ell_x}_{\ell_x-1}\boldsymbol{\varphi}^{\ell_x}(\theta_x))_j &= \frac{1}{2}(\varphi^{\ell_x}_{2j-1}(\theta_x)+\varphi^{\ell_x}_{2j}(\theta_x)) = \frac{1}{2}(e^{-\mathrm{i}\theta_x}+1)\varphi^{\ell_x}_{2j}(\theta_x)\\ &= \frac{1}{2}(e^{-\mathrm{i}\theta_x}+1)\varphi^{\ell_x-1}_{j}(2\theta_x) = \mathcal{R}^{\ell_x}_{\ell_x-1}(\theta_x)\varphi^{\ell_x-1}_{j}(2\theta_x),\ j=1,\dots,N_{\ell_x-1},\end{aligned}$$

using the shifting Lemma 3.12 and $\varphi^{\ell_x}_{2j}(\theta_x) = \varphi^{\ell_x-1}_j(2\theta_x)$.
For the prolongation operator it holds

$$(\mathbf{P}^{\ell_x-1}_{\ell_x}\boldsymbol{\varphi}^{\ell_x-1}(2\theta_x))_{2j-1} = \varphi^{\ell_x-1}_j(2\theta_x) = \varphi^{\ell_x}_{2j}(\theta_x) = e^{\mathrm{i}\theta_x}\varphi^{\ell_x}_{2j-1}(\theta_x),$$

and

$$(\mathbf{P}^{\ell_x-1}_{\ell_x}\boldsymbol{\varphi}^{\ell_x-1}(2\theta_x))_{2j} = \varphi^{\ell_x-1}_j(2\theta_x) = \varphi^{\ell_x}_{2j}(\theta_x),$$

for $j=1,\dots,N_{\ell_x-1}$, with the same arguments as before. Then

$$(\mathbf{P}^{\ell_x-1}_{\ell_x}\boldsymbol{\varphi}^{\ell_x-1}(2\theta_x))_j = \begin{cases} e^{\mathrm{i}\theta_x}\varphi^{\ell_x}_j(\theta_x), & j \text{ odd},\\ \varphi^{\ell_x}_j(\theta_x), & j \text{ even},\end{cases}$$

for $j=1,\dots,N_{\ell_x}$. Moreover,

$$\varphi^{\ell_x}_j(\gamma(\theta_x)) = e^{\mathrm{i}j\gamma(\theta_x)} = \begin{cases} e^{\mathrm{i}j\pi}e^{\mathrm{i}j\theta_x}, & \theta_x<0,\\ e^{-\mathrm{i}j\pi}e^{\mathrm{i}j\theta_x}, & \theta_x\geq 0,\end{cases} = \begin{cases} -\varphi^{\ell_x}_j(\theta_x), & j \text{ odd},\\ \varphi^{\ell_x}_j(\theta_x), & j \text{ even},\end{cases}$$

and

$$\mathcal{P}^{\ell_x-1}_{\ell_x}(\gamma(\theta_x)) = \frac{1}{2}(e^{\mathrm{i}\gamma(\theta_x)}+1) = \frac{1}{2}(-e^{\mathrm{i}\theta_x}+1),$$

for $j=1,\dots,N_{\ell_x}$. This implies

$$\begin{aligned}&\mathcal{P}^{\ell_x-1}_{\ell_x}(\theta_x)\varphi^{\ell_x}_j(\theta_x) + \mathcal{P}^{\ell_x-1}_{\ell_x}(\gamma(\theta_x))\varphi^{\ell_x}_j(\gamma(\theta_x))\\ &= \frac{1}{2}(e^{\mathrm{i}\theta_x}+1)\varphi^{\ell_x}_j(\theta_x) + \frac{1}{2}(-e^{\mathrm{i}\theta_x}+1)\begin{cases} -\varphi^{\ell_x}_j(\theta_x), & j \text{ odd},\\ \varphi^{\ell_x}_j(\theta_x), & j \text{ even}\end{cases}\\ &= \begin{cases} e^{\mathrm{i}\theta_x}\varphi^{\ell_x}_j(\theta_x), & j \text{ odd},\\ \varphi^{\ell_x}_j(\theta_x), & j \text{ even},\end{cases} = (\mathbf{P}^{\ell_x-1}_{\ell_x}\boldsymbol{\varphi}^{\ell_x-1}(2\theta_x))_j,\end{aligned}$$

for $j=1,\dots,N_{\ell_x}$. □

The following five Lemmata from [13] give us the Fourier symbols of the restriction and prolongation operators for the different coarsening strategies.

Lemma 4.4 (Fourier symbols for temporal prolongation and restriction)**.** Consider temporal restriction and prolongation operators $\mathbf{R}^{\ell_t}_{\ell_t-1} \in \mathbb{C}^{N_tN_{\ell_t-1}\times N_tN_{\ell_t}}$ and $\mathbf{P}^{\ell_t-1}_{\ell_t} \in \mathbb{C}^{N_tN_{\ell_t}\times N_tN_{\ell_t-1}}$ as defined in (8). Let $\boldsymbol{\Phi}^{\ell_t}(\theta_t) \in \mathbb{C}^{N_tN_{\ell_t}}$ be a fine Fourier mode and $\boldsymbol{\Phi}^{\ell_t-1}(\theta_t) \in \mathbb{C}^{N_tN_{\ell_t-1}}$ a coarse Fourier mode for $\theta_t \in \Theta^{low}_{\ell_t}$ with elements

$$\begin{aligned}\Phi^{n,\ell_t}_l(\theta_t) &:= \varphi_n(\theta_t), \quad l=1,\dots,N_t,\ n=1,\dots,N_{\ell_t},\\ \Phi^{n,\ell_t-1}_l(\theta_t) &:= \varphi_n(\theta_t), \quad l=1,\dots,N_t,\ n=1,\dots,N_{\ell_t-1}.\end{aligned}$$

Then for $\boldsymbol{\mathcal{R}}_{\ell_t-1}^{\ell_t}(\theta_t) := e^{-\mathrm{i}\theta_t}\mathbf{R}_1 + \mathbf{R}_2 \in \mathbb{C}^{N_t\times N_t}$, with $\mathbf{R}_1$ and $\mathbf{R}_2$ defined in (8), it holds

$$(\mathbf{R}_{\ell_t-1}^{\ell_t}\boldsymbol{\Phi}^{\ell_t}(\theta_t))^n = \boldsymbol{\mathcal{R}}_{\ell_t-1}^{\ell_t}(\theta_t)\boldsymbol{\Phi}^{n,\ell_t-1}(2\theta_t),\ n = 1,\dots,N_{\ell_t-1},$$

and we call $\boldsymbol{\mathcal{R}}_{\ell_t-1}^{\ell_t}(\theta_t) \in \mathbb{C}^{N_t\times N_t}$ the Fourier symbol for the restriction operator in time. Moreover, for $\boldsymbol{\mathcal{P}}_{\ell_t}^{\ell_t-1}(\theta_t) := \frac{1}{2}(e^{\mathrm{i}\theta_t}\mathbf{R}_1^T + \mathbf{R}_2^T) \in \mathbb{C}^{N_t\times N_t}$ it holds

$$(\mathbf{P}_{\ell_t}^{\ell_t-1}\boldsymbol{\Phi}^{\ell_t-1}(2\theta_t))^n = \boldsymbol{\mathcal{P}}_{\ell_t}^{\ell_t-1}(\theta_t)\boldsymbol{\Phi}^{n,\ell_t}(\theta_t) + \boldsymbol{\mathcal{P}}_{\ell_t}^{\ell_t-1}(\gamma(\theta_t))\boldsymbol{\Phi}^{n,\ell_t}(\gamma(\theta_t)),\ n = 1,\dots,N_{\ell_t},$$

and we call $\boldsymbol{\mathcal{P}}_{\ell_t}^{\ell_t-1}(\theta_t) \in \mathbb{C}^{N_t\times N_t}$ the Fourier symbol for the prolongation in time.

With these results we can get the mapping properties for the semi-restriction and semi-prolongation operators.

Lemma 4.5 (Fourier symbol for restriction, semi-coarsening)**.** The following mapping property holds for the restriction operator $(\underline{\mathbf{R}}_{\ell-1}^{\ell})^s$:

$$\begin{aligned}(\underline{\mathbf{R}}_{\ell-1}^{\ell})^s : \mathcal{E}_{\ell_x,\ell_t}(\theta_x,\theta_t) &\to \Psi_{\ell_x,\ell_t-1}(\theta_x,2\theta_t),\\ \begin{pmatrix}\mathbf{U}_1\\ \mathbf{U}_2\\ \mathbf{U}_3\\ \mathbf{U}_4\end{pmatrix} &\mapsto (\widetilde{\boldsymbol{\mathcal{R}}}_{\ell-1}^{\ell})^s(\theta_t)\begin{pmatrix}\mathbf{U}_1\\ \mathbf{U}_2\\ \mathbf{U}_3\\ \mathbf{U}_4\end{pmatrix},\end{aligned}$$

with

$$(\widetilde{\boldsymbol{\mathcal{R}}}_{\ell-1}^{\ell})^s(\theta_t) := \begin{pmatrix}\boldsymbol{\mathcal{R}}_{\ell_t-1}^{\ell_t}(\theta_t) & 0 & \boldsymbol{\mathcal{R}}_{\ell_t-1}^{\ell_t}(\gamma(\theta_t)) & 0\\ 0 & \boldsymbol{\mathcal{R}}_{\ell_t-1}^{\ell_t}(\theta_t) & 0 & \boldsymbol{\mathcal{R}}_{\ell_t-1}^{\ell_t}(\gamma(\theta_t))\end{pmatrix} \in \mathbb{C}^{2N_t\times 4N_t}$$

and the Fourier symbol $\boldsymbol{\mathcal{R}}_{\ell_t-1}^{\ell_t}(\theta_t) \in \mathbb{C}^{N_t\times N_t}$ as defined in Lemma 4.4.

Lemma 4.6 (Fourier symbol of prolongation, semi-coarsening)**.** The following mapping property holds for the prolongation operator $(\underline{\mathbf{P}}_{\ell}^{\ell-1})^s$:

$$\begin{aligned}(\underline{\mathbf{P}}_{\ell}^{\ell-1})^s : \Psi_{\ell_x,\ell_t-1}(\theta_x,2\theta_t) &\to \mathcal{E}_{\ell_x,\ell_t}(\theta_x,\theta_t),\\ \begin{pmatrix}\mathbf{U}_1\\ \mathbf{U}_2\end{pmatrix} &\mapsto (\widetilde{\boldsymbol{\mathcal{P}}}_{\ell}^{\ell-1})^s(\theta_t)\begin{pmatrix}\mathbf{U}_1\\ \mathbf{U}_2\end{pmatrix},\end{aligned}$$

with

$$(\widetilde{\boldsymbol{\mathcal{P}}}_{\ell}^{\ell-1})^s(\theta_t) := \begin{pmatrix}\boldsymbol{\mathcal{P}}_{\ell_t}^{\ell_t-1}(\theta_t) & 0\\ 0 & \boldsymbol{\mathcal{P}}_{\ell_t}^{\ell_t-1}(\theta_t)\\ \boldsymbol{\mathcal{P}}_{\ell_t}^{\ell_t-1}(\gamma(\theta_t)) & 0\\ 0 & \boldsymbol{\mathcal{P}}_{\ell_t}^{\ell_t-1}(\gamma(\theta_t))\end{pmatrix} \in \mathbb{C}^{4N_t\times 2N_t}$$

and the Fourier symbol $\boldsymbol{\mathcal{P}}_{\ell_t}^{\ell_t-1}(\theta_t) \in \mathbb{C}^{N_t\times N_t}$ as defined in Lemma 4.4.

Analogously to the semi-coarsening case we can get the mapping properties for the full-restriction and full-prolongation operators.

Lemma 4.7 (Fourier symbol of restriction, full-coarsening)**.** The following mapping property holds for the restriction operator $(\underline{\mathbf{R}}^{\ell}_{\ell-1})^f$:

$$(\underline{\mathbf{R}}^{\ell}_{\ell-1})^f : \mathcal{E}_{\ell_x,\ell_t}(\theta_x,\theta_t) \to \Psi_{\ell_x-1,\ell_t-1}(2\theta_x,2\theta_t),$$
$$\begin{pmatrix}\mathbf{U}_1\\ \mathbf{U}_2\\ \mathbf{U}_3\\ \mathbf{U}_4\end{pmatrix} \mapsto (\widetilde{\boldsymbol{\mathcal{R}}}^{\ell}_{\ell-1})^f(\theta_x,\theta_t)\begin{pmatrix}\mathbf{U}_1\\ \mathbf{U}_2\\ \mathbf{U}_3\\ \mathbf{U}_4\end{pmatrix},$$

with

$$(\widetilde{\boldsymbol{\mathcal{R}}}^{\ell}_{\ell-1})^f(\theta_x,\theta_t) :=$$
$$\begin{pmatrix}\hat{\boldsymbol{\mathcal{R}}}^{\ell}_{\ell-1}(\theta_x,\theta_t) & \hat{\boldsymbol{\mathcal{R}}}^{\ell}_{\ell-1}(\gamma(\theta_x),\theta_t) & \hat{\boldsymbol{\mathcal{R}}}^{\ell}_{\ell_t}(\theta_x,\gamma(\theta_t)) & \hat{\boldsymbol{\mathcal{R}}}^{\ell}_{\ell-1}(\gamma(\theta_x),\gamma(\theta_t))\end{pmatrix} \in \mathbb{C}^{N_t\times 4N_t},$$

and the Fourier symbol

$$\hat{\boldsymbol{\mathcal{R}}}^{\ell}_{\ell-1}(\theta_x,\theta_t) := \mathcal{R}^{\ell_x}_{\ell_x-1}(\theta_x)\boldsymbol{\mathcal{R}}^{\ell_t}_{\ell_t-1}(\theta_t) \in \mathbb{C}^{N_t\times N_t},$$

with $\mathcal{R}^{\ell_x}_{\ell_x-1}(\theta_x) \in \mathbb{C}$ from Lemma 4.3 and $\boldsymbol{\mathcal{R}}^{\ell_t}_{\ell_t-1}(\theta_t) \in \mathbb{C}^{N_t\times N_t}$ from Lemma 4.4.

Lemma 4.8 (Fourier symbol of prolongation, full-coarsening)**.** The following mapping property holds for the prolongation operator $(\underline{\mathbf{P}}^{\ell-1}_{\ell})^f$:

$$(\underline{\mathbf{P}}^{\ell-1}_{\ell})^f : \Psi_{\ell_x-1,\ell_t-1}(2\theta_x,2\theta_t) \to \mathcal{E}_{\ell_x,\ell_t}(\theta_x,\theta_t),$$
$$\mathbf{U} \mapsto (\widetilde{\boldsymbol{\mathcal{P}}}^{\ell-1}_{\ell})^f(\theta_t,\theta_x)\mathbf{U},$$

with

$$(\widetilde{\boldsymbol{\mathcal{P}}}^{\ell-1}_{\ell})^f(\theta_t,\theta_x) := \begin{pmatrix}\hat{\boldsymbol{\mathcal{P}}}^{\ell-1}_{\ell}(\theta_x,\theta_t)\\ \hat{\boldsymbol{\mathcal{P}}}^{\ell-1}_{\ell}(\gamma(\theta_x),\theta_t)\\ \hat{\boldsymbol{\mathcal{P}}}^{\ell-1}_{\ell}(\theta_x,\gamma(\theta_t))\\ \hat{\boldsymbol{\mathcal{P}}}^{\ell-1}_{\ell}(\gamma(\theta_x),\gamma(\theta_t))\end{pmatrix} \in \mathbb{C}^{4N_t\times N_t},$$

and the Fourier symbol

$$\hat{\boldsymbol{\mathcal{P}}}^{\ell-1}_{\ell}(\theta_x,\theta_t) := \mathcal{P}^{\ell_x-1}_{\ell_x}(\theta_x)\boldsymbol{\mathcal{P}}^{\ell_t-1}_{\ell_t}(\theta_t) \in \mathbb{C}^{N_t\times N_t},$$

with $\mathcal{P}^{\ell_x-1}_{\ell_x} \in \mathbb{C}$ from Lemma 4.3 and $\boldsymbol{\mathcal{P}}^{\ell_t-1}_{\ell_t} \in \mathbb{C}^{N_t\times N_t}$ from Lemma 4.4.

With Lemma 4.1 we obtain the mapping property for coarse grid correction when semi-coarsening in time is applied:

$$(\underline{\mathbf{L}}_{2\tau_\ell,\xi_\ell})^{-1} : \Psi_{\ell_x,\ell_t-1}(\theta_x,2\theta_t) \to \Psi_{\ell_x,\ell_t-1}(\theta_x,2\theta_t),$$
$$\begin{pmatrix}\mathbf{U}_1\\ \mathbf{U}_2\end{pmatrix} \mapsto (\hat{\boldsymbol{\mathcal{L}}}^{s}_{2\tau_\ell,\xi_\ell}(\theta_x,2\theta_t))^{-1}\begin{pmatrix}\mathbf{U}_1\\ \mathbf{U}_2\end{pmatrix}, \tag{24}$$

with

$$(\hat{\boldsymbol{\mathcal{L}}}^{s}_{2\tau_\ell,\xi_\ell}(\theta_x,2\theta_t))^{-1} := \begin{pmatrix}(\boldsymbol{\mathcal{L}}_{2\tau_\ell,\xi_\ell}(\theta_x,2\theta_t))^{-1} & 0\\ 0 & (\boldsymbol{\mathcal{L}}_{2\tau_\ell,\xi_\ell}(\gamma(\theta_x),2\theta_t))^{-1}\end{pmatrix} \in \mathbb{C}^{2N_t\times 2N_t}.$$

A complication arises for frequencies (θ_x, θ_t) such that $\boldsymbol{\mathcal{L}}_{2\tau_\ell,\xi_\ell}(\theta_x, 2\theta_t) = 0$. For some more discussion of the reasons for this formal complication we refer to [26]. In order to make sure that $\hat{\boldsymbol{\mathcal{L}}}^s$ exists, we exclude the set of frequencies

$$\Lambda_s := \left\{(\theta_x, \theta_t) \in (-\pi, \pi] \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right] : \boldsymbol{\mathcal{L}}_{\tau_\ell,\xi_\ell}(\theta_x, \theta_t) = 0 \text{ or } \boldsymbol{\mathcal{L}}_{2\tau_\ell,\xi_\ell}(\theta_x, 2\theta_t) = 0\right\}. \tag{25}$$

For the full-coarsening case we obtain the mapping property

$$\begin{aligned}(\underline{\mathbf{L}}_{2\tau_\ell,2\xi_\ell})^{-1} : \Psi_{\ell_x-1,\ell_t-1}(2\theta_x, 2\theta_t) &\to \Psi_{\ell_x-1,\ell_t-1}(2\theta_x, 2\theta_t),\\ \mathbf{U} &\mapsto (\hat{\boldsymbol{\mathcal{L}}}^f_{2\tau_\ell,2\xi_\ell}(2\theta_x, 2\theta_t))^{-1}\mathbf{U},\end{aligned} \tag{26}$$

with

$$(\hat{\boldsymbol{\mathcal{L}}}^f_{2\tau_\ell,2\xi_\ell}(2\theta_x, 2\theta_t))^{-1} := (\boldsymbol{\mathcal{L}}_{2\tau_\ell,2\xi_\ell}(2\theta_x, 2\theta_t))^{-1} \in \mathbb{C}^{N_t\times N_t}. \tag{27}$$

As before, a complication arises for frequencies (θ_x, θ_t) such that $\boldsymbol{\mathcal{L}}_{2\tau_\ell,2\xi_\ell}(2\theta_x, 2\theta_t) = 0$. In order to make sure that $\hat{\boldsymbol{\mathcal{L}}}^f$ exists, we exclude the set of frequencies

$$\Lambda_f := \left\{(\theta_x, \theta_t) \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right]^2 : \boldsymbol{\mathcal{L}}_{\tau_\ell,\xi_\ell}(\theta_x, \theta_t) = 0 \text{ or } \boldsymbol{\mathcal{L}}_{2\tau_\ell,2\xi_\ell}(2\theta_x, 2\theta_t) = 0\right\}.$$

We are now able to get the Fourier symbol of the two-grid operators and calculate the asymptotic convergence factors.

Theorem 4.9 (Fourier symbol of two-grid operator, semi-coarsening)**.** For $(\theta_x, \theta_t) \in \Theta^{low,f}_{\ell_x,\ell_t}$ the following mapping property holds for the two-grid operator $\underline{\mathbf{M}}^s_{\tau_\ell,\xi_\ell}$ in (17) with semi-coarsening in time:

$$\begin{aligned}\underline{\mathbf{M}}^s_{\tau_\ell,\xi_\ell} : \mathcal{E}_{\ell_x,\ell_t}(\theta_x, \theta_t) &\to \mathcal{E}_{\ell_x,\ell_t}(\theta_x, \theta_t),\\ \begin{pmatrix}\mathbf{U}_1\\ \mathbf{U}_2\\ \mathbf{U}_3\\ \mathbf{U}_4\end{pmatrix} &\mapsto \boldsymbol{\mathcal{M}}^s(\theta_x, \theta_t)\begin{pmatrix}\mathbf{U}_1\\ \mathbf{U}_2\\ \mathbf{U}_3\\ \mathbf{U}_4\end{pmatrix},\end{aligned}$$

with

$$\begin{aligned}\boldsymbol{\mathcal{M}}^s(\theta_x, \theta_t) := \widetilde{\boldsymbol{\mathcal{S}}}^{\nu_2}_{\tau_\ell,\xi_\ell}(\theta_x, \theta_t)(\mathbf{I}_{4N_t} - (\widetilde{\boldsymbol{\mathcal{P}}}^{\ell-1}_\ell)^s(\theta_t)(\hat{\boldsymbol{\mathcal{L}}}^s_{2\tau_\ell,\xi_\ell}(\theta_x, 2\theta_t))^{-1}\\ (\widetilde{\boldsymbol{\mathcal{R}}}^\ell_{\ell-1})^s(\theta_t)\widetilde{\boldsymbol{\mathcal{L}}}_{\tau_\ell,\xi_\ell}(\theta_x, \theta_t))\widetilde{\boldsymbol{\mathcal{S}}}^{\nu_1}_{\tau_\ell,\xi_\ell}(\theta_x, \theta_t) \in \mathbb{C}^{4N_t\times 4N_t}.\end{aligned} \tag{28}$$

Proof. The two-grid operator for semi-coarsening is given by

$$\underline{\mathbf{M}}^s_{\tau_\ell,\xi_\ell} = \underline{\mathbf{S}}^{\nu_2}_{\tau_\ell,\xi_\ell}(\underline{\mathbf{I}} - (\underline{\mathbf{P}}^{\ell-1}_\ell)_s(\underline{\mathbf{L}}_{2\tau_\ell,\xi_\ell})^{-1}(\underline{\mathbf{R}}^\ell_{\ell-1})_s\underline{\mathbf{L}}_{\tau_\ell,\xi_\ell})\underline{\mathbf{S}}^{\nu_1}_{\tau_\ell,\xi_\ell}.$$

By previous results we obtain

$$\begin{aligned}\underline{\mathbf{M}}^s_{\tau_\ell,\xi_\ell} :&\mathcal{E}_{\ell_x,\ell_t} \xrightarrow{(23)} \mathcal{E}_{\ell_x,\ell_t} \xrightarrow{(22)} \mathcal{E}_{\ell_x,\ell_t} \xrightarrow{4.5} \Psi_{\ell_x,\ell_t-1}(\theta_x, 2\theta_t)\\ &\xrightarrow{(24)} \Psi_{\ell_x,\ell_t-1}(\theta_x, 2\theta_t) \xrightarrow{4.6} \mathcal{E}_{\ell_x,\ell_t} \xrightarrow{(23)} \mathcal{E}_{\ell_x,\ell_t},\end{aligned}$$

with the mapping

$$\begin{pmatrix}\mathbf{U}_1\\\mathbf{U}_2\\\mathbf{U}_3\\\mathbf{U}_4\end{pmatrix} \mapsto \widetilde{\boldsymbol{\mathcal{S}}}^{\nu_1}_{\tau_\ell,\xi_\ell}\begin{pmatrix}\mathbf{U}_1\\\mathbf{U}_2\\\mathbf{U}_3\\\mathbf{U}_4\end{pmatrix} \mapsto \widetilde{\boldsymbol{\mathcal{L}}}_{\tau_\ell,\xi_\ell}\widetilde{\boldsymbol{\mathcal{S}}}^{\nu_1}_{\tau_\ell,\xi_\ell}\begin{pmatrix}\mathbf{U}_1\\\mathbf{U}_2\\\mathbf{U}_3\\\mathbf{U}_4\end{pmatrix} \mapsto (\widetilde{\boldsymbol{\mathcal{R}}}^{\ell}_{\ell-1})^s\widetilde{\boldsymbol{\mathcal{L}}}_{\tau_\ell,\xi_\ell}\widetilde{\boldsymbol{\mathcal{S}}}^{\nu_1}_{\tau_\ell,\xi_\ell}\begin{pmatrix}\mathbf{U}_1\\\mathbf{U}_2\\\mathbf{U}_3\\\mathbf{U}_4\end{pmatrix}$$

$$\mapsto (\hat{\boldsymbol{\mathcal{L}}}^{s}_{2\tau_\ell,\xi_\ell})^{-1}(\widetilde{\boldsymbol{\mathcal{R}}}^{\ell}_{\ell-1})^s\widetilde{\boldsymbol{\mathcal{L}}}_{\tau_\ell,\xi_\ell}\widetilde{\boldsymbol{\mathcal{S}}}^{\nu_1}_{\tau_\ell,\xi_\ell}\begin{pmatrix}\mathbf{U}_1\\\mathbf{U}_2\\\mathbf{U}_3\\\mathbf{U}_4\end{pmatrix}$$

$$\mapsto (\widetilde{\boldsymbol{\mathcal{P}}}^{\ell-1}_{\ell})^s(\hat{\boldsymbol{\mathcal{L}}}^{s}_{2\tau_\ell,\xi_\ell})^{-1}(\widetilde{\boldsymbol{\mathcal{R}}}^{\ell}_{\ell-1})^s\widetilde{\boldsymbol{\mathcal{L}}}_{\tau_\ell,\xi_\ell}\widetilde{\boldsymbol{\mathcal{S}}}^{\nu_1}_{\tau_\ell,\xi_\ell}\begin{pmatrix}\mathbf{U}_1\\\mathbf{U}_2\\\mathbf{U}_3\\\mathbf{U}_4\end{pmatrix}$$

$$\mapsto (\mathbf{I}_{4N_t} - (\widetilde{\boldsymbol{\mathcal{P}}}^{\ell-1}_{\ell})^s(\hat{\boldsymbol{\mathcal{L}}}^{s}_{2\tau_\ell,\xi_\ell})^{-1}(\widetilde{\boldsymbol{\mathcal{R}}}^{\ell}_{\ell-1})^s\widetilde{\boldsymbol{\mathcal{L}}}_{\tau_\ell,\xi_\ell})\widetilde{\boldsymbol{\mathcal{S}}}^{\nu_1}_{\tau_\ell,\xi_\ell}\begin{pmatrix}\mathbf{U}_1\\\mathbf{U}_2\\\mathbf{U}_3\\\mathbf{U}_4\end{pmatrix}$$

$$\mapsto \widetilde{\boldsymbol{\mathcal{S}}}^{\nu_2}_{\tau_\ell,\xi_\ell}(\mathbf{I}_{4N_t} - (\widetilde{\boldsymbol{\mathcal{P}}}^{\ell-1}_{\ell})^s(\hat{\boldsymbol{\mathcal{L}}}^{s}_{2\tau_\ell,\xi_\ell})^{-1}(\widetilde{\boldsymbol{\mathcal{R}}}^{\ell}_{\ell-1})^s\widetilde{\boldsymbol{\mathcal{L}}}_{\tau_\ell,\xi_\ell})\widetilde{\boldsymbol{\mathcal{S}}}^{\nu_1}_{\tau_\ell,\xi_\ell}\begin{pmatrix}\mathbf{U}_1\\\mathbf{U}_2\\\mathbf{U}_3\\\mathbf{U}_4\end{pmatrix}.$$

□

Theorem 4.10 (Fourier symbol of two-grid operator, full-coarsening)**.** For $(\theta_x,\theta_t) \in \Theta^{low,f}_{\ell_x,\ell_t}$ the following mapping property holds for the two-grid operator $\underline{\mathbf{M}}^f_{\tau_\ell,\xi_\ell}$ in (16) with space-time coarsening:

$$\underline{\mathbf{M}}^f_{\tau_\ell,\xi_\ell} : \mathcal{E}_{\ell_x,\ell_t}(\theta_x,\theta_t) \to \mathcal{E}_{\ell_x,\ell_t}(\theta_x,\theta_t),$$
$$\begin{pmatrix}\mathbf{U}_1\\\mathbf{U}_2\\\mathbf{U}_3\\\mathbf{U}_4\end{pmatrix} \mapsto \boldsymbol{\mathcal{M}}^f(\theta_x,\theta_t)\begin{pmatrix}\mathbf{U}_1\\\mathbf{U}_2\\\mathbf{U}_3\\\mathbf{U}_4\end{pmatrix},$$

with

$$\begin{aligned}\boldsymbol{\mathcal{M}}^f(\theta_x,\theta_t) := \widetilde{\boldsymbol{\mathcal{S}}}^{\nu_2}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t)(\mathbf{I}_{4N_t} - (\widetilde{\boldsymbol{\mathcal{P}}}^{\ell-1}_{\ell})^f(\theta_x,\theta_t)(\hat{\boldsymbol{\mathcal{L}}}^{f}_{2\tau_\ell,2\xi_\ell}(2\theta_x,2\theta_t))^{-1}\\ (\widetilde{\boldsymbol{\mathcal{R}}}^{\ell}_{\ell-1})^f(\theta_x,\theta_t)\widetilde{\boldsymbol{\mathcal{L}}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t))\widetilde{\boldsymbol{\mathcal{S}}}^{\nu_1}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t) \in \mathbb{C}^{4N_t\times 4N_t}.\end{aligned} \tag{29}$$

Proof. The proof follows analogous to the previous one. □

5 Smoothing Analysis

We now have all tools at hand to analyze the elements of the multigrid iteration. We start with the smoother. The asymptotic smoothing factor of the damped block Jacobi method (15) can be measured by computing the spectral radius of its symbol $\boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t)$, which is of much smaller size and thus makes the calculations feasible.

Definition 5.1 ([34])**.** We define the *asymptotic smoothing factors* for semi- and full-coarsening as

$$\varrho(\boldsymbol{\mathcal{S}}^s) := \max\{\varrho(\boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t)) : (\theta_x,\theta_t) \in \Theta^{high,s}_{\ell_x,\ell_t}\},$$
$$\varrho(\boldsymbol{\mathcal{S}}^f) := \max\{\varrho(\boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t)) : (\theta_x,\theta_t) \in \Theta^{high,f}_{\ell_x,\ell_t}\},$$

with $\boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t)$ the Fourier symbol of the smoother and the set of frequencies defined in (20) and (21).

Lemma 5.2. The spectral radius of the Fourier symbol of the smoother $\boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t)$ is given by

$$\rho(\boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t)) = \max\{|1-\omega_t|, S(\omega_t,\theta_x,\theta_t)\}$$

with

$$S(\omega_t,\theta_x,\theta_t) := |1-\omega_t + e^{-\mathrm{i}\theta_t}\omega_t R(-\mu\beta(\theta_x))|, \tag{30}$$

R the stability function of the DG-SEM time stepping scheme, $\beta(\theta_x) := 1-e^{-\mathrm{i}\theta_x}$ and CFL number $\mu := \frac{a\Delta\tau_\ell}{\Delta x_\ell}$.

Proof. The eigenvalues of $\boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t)$ are given by

$$\sigma(\boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t)) = 1-\omega_t + e^{-\mathrm{i}\theta_t}\omega_t\sigma\left(\left(\mathbf{K}_{\tau_\ell} + \frac{a}{\Delta x}(-e^{-\mathrm{i}\theta_x}+1)\mathbf{M}_{\tau_\ell}\right)^{-1}\mathbf{C}_{\tau_\ell}\right).$$

With Lemma 3.5 we can compute the spectrum as

$$\sigma(\boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t)) = \{1-\omega_t, 1-\omega_t + e^{-\mathrm{i}\theta_t}\omega_t R(-\mu\beta(\theta_x))\}.$$

Therefore it follows that

$$\rho(\boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\theta_x,\theta_t)) = \max\{|1-\omega_t|, |1-\omega_t + e^{-\mathrm{i}\theta_k}\omega_t R(-\mu\beta(\theta_x))|\}.$$

□

5.1 Optimal Damping Parameter

The goal is to find a smoother with optimal smoothing behavior, i.e. to find a damping parameter ω_t for the block Jacobi smoother such that the high frequencies frequencies are smoothed as efficiently as possible. We analyze the Fourier symbol $\boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}$ to find the optimal damping parameter $\omega_t \in (0,1]$ in (15). In order to do so, the frequencies which are damped less efficiently need to be determined. These are also called worst case frequencies:

Definition 5.3. The *worst case frequencies* for the Fourier symbol $\boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}$ of the smoother are defined as those high frequencies that are damped least efficiently.

This can be done by calculating the maximum of $|1-\omega_t|$ and

$$(\theta_x^*(\omega_t,\mu),\theta_t^*(\omega_t,\mu)) := \underset{(\theta_x,\theta_t)\in\Theta^{high}}{\arg\sup}\ S(\omega_t,\theta_x,\theta_t), \tag{31}$$

with the function S defined in Lemma 5.2, see equation (30).

Straightforward calculations give

$$\begin{aligned} S(\omega_t,\theta_x,\theta_t)^2 &= |1-\omega_t+e^{-\mathrm{i}\theta_t}\omega_t R(-\mu\beta(\theta_x))|^2 \\ &= (1-\omega_t)^2+2\omega_t(1-\omega_t)(\cos(\theta_t)\mathrm{Re}(R(-\mu\beta(\theta_x)))+\sin(\theta_t)\mathrm{Im}(R(-\mu\beta(\theta_x)))) \\ &\quad+\omega_t^2|R(-\mu\beta(\theta_x))|^2. \end{aligned}$$

When considering implicit time integration, large CFL numbers $\mu \gg 0$ are of interest. Then

$$\cos(\theta_t)\mathrm{Re}(R(-\mu\beta(\theta_x)))+\sin(\theta_t)\mathrm{Im}(R(-\mu\beta(\theta_x))) \xrightarrow[\mu\to\infty]{} 0,$$

since $\cos(\theta_t),\sin(\theta_t)\in[-1,1]$, and the method is L-stable, see Corollary 3.3, thus $R(-z)\to 0$ for $z\to\infty$ and $\mathrm{Re}(-\mu\beta(\theta_x))\le 0$ for $\theta_x\in[-\pi,\pi]$.

To find the worst case frequencies in space for large CFL numbers μ we thus need to maximize $|R(-\mu\beta(\theta_x))|^2$. From L-stability it follows that $|R(-\mu\beta(\theta_x))|^2\le 1$ and moreover we have by definition of the Padé approximant that $R(0)=1$. Thus we have found the worst case frequency in space:

$$\theta_x^* = \underset{\theta_x\in(-\pi,\pi]}{\arg\sup}\ S(\omega_t,\theta_x,\theta_t) = 0.$$

Evaluating (30) at $\theta_x=\theta_x^*=0$ gives

$$\begin{aligned} S(\omega_t,0,\theta_t) &= |1-\omega_t+e^{-\mathrm{i}\theta_t}\omega_t R(-\mu\beta(\theta_x))| \\ &= \sqrt{(1-\omega_t)^2+2\omega_t(1-\omega_t)\cos(\theta_t)+\omega_t^2}. \end{aligned}$$

With this, it follows that the worst case frequencies in time are given by

$$\begin{aligned} \theta_t^* &= \underset{\theta_t\in[\pi/2,\pi]}{\arg\sup}\ S(\omega_t,\theta_x^*,\theta_t) = \frac{\pi}{2}, \\ \theta_t^* &= \underset{\theta_t\in[-\pi,-\pi/2]}{\arg\sup}\ S(\omega_t,\theta_x^*,\theta_t) = -\frac{\pi}{2}. \end{aligned}$$

Thus, we have found worst case frequencies $(\theta_x^*,\theta_t^*)\in\Theta_{\ell_x,\ell_t}^{high,s}$ for the semi-coarsening strategy as well as $(\theta_x^*,\theta_t^*)\in\Theta_{\ell_x,\ell_t}^{high,f}$ for the full-coarsening strategy. The optimal damping parameter can then be calculated by

$$\omega_t^* = \underset{\omega_t\in(0,1]}{\arg\inf}\ S(\omega_t,\theta_x^*,\theta_t^*) = 0.5.$$

5.2 Asymptotic Smoothing Factor

With the optimal damping parameter $\omega_t^*=0.5$ and the worst case frequencies (θ_x^*,θ_t^*) at hand we can calculate the asymptotic smoothing factor from Definition 5.1.

In the case of full space-time coarsening we get for μ large enough

$$\varrho(\boldsymbol{\mathcal{S}}) = \rho(\boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\theta_x^*,\theta_t^*)) = \max\left\{|0.5|, S(0.5,0,\frac{\pi}{2})\right\} = \max\{0.5,\sqrt{0.5}\} = \frac{1}{\sqrt{2}},$$

and for semi-coarsening in time

$$\varrho(\boldsymbol{\mathcal{S}}) = \rho(\boldsymbol{\mathcal{S}}_{\tau_\ell,\xi_\ell}(\theta_x^*,\theta_t^*)) = \max\left\{|0.5|, S(0.5,0,-\frac{\pi}{2})\right\} = \max\{0.5,\sqrt{0.5}\} = \frac{1}{\sqrt{2}}.$$

6 Two-Grid Analysis

In this section we analyze the two-grid iteration for full- and semi-coarsening by studying the corresponding iteration matrices $\underline{\mathbf{M}}^f_{\tau_\ell,\xi_\ell}$ and $\underline{\mathbf{M}}^s_{\tau_\ell,\xi_\ell}$, see equations (16) and (17). With Theorems 4.9 and 4.10 we can analyze the asymptotic convergence behavior of the two-grid cycle by computing the maximal spectral radius of the Fourier symbols $\boldsymbol{\mathcal{M}}^s_\mu(\theta_x,\theta_t)$ and $\boldsymbol{\mathcal{M}}^f_\mu(\theta_x,\theta_t)$ for $(\theta_x,\theta_t) \in \Theta^{low,f}_{\ell_x,\ell_t}$, see (29) and (28).

Definition 6.1. We define the *asymptotic two-grid convergence factors* as

$$\varrho(\boldsymbol{\mathcal{M}}^s) := \max\{\varrho(\boldsymbol{\mathcal{M}}^s(\theta_x,\theta_t)) : (\theta_x,\theta_t) \in \Theta^{low,f}_{\ell_x,\ell_t} \setminus \Lambda_s\},$$
$$\varrho(\boldsymbol{\mathcal{M}}^f) := \max\{\varrho(\boldsymbol{\mathcal{M}}^f(\theta_x,\theta_t)) : (\theta_x,\theta_t) \in \Theta^{low,f}_{\ell_x,\ell_t} \setminus \Lambda_f\},$$

with $\boldsymbol{\mathcal{M}}^s(\theta_x,\theta_t)$ and $\boldsymbol{\mathcal{M}}^f(\theta_x,\theta_t)$ the symbols of the two-grid iteration matrices and Λ_s defined in (25) and Λ_f defined in (27).

To derive $\varrho(\boldsymbol{\mathcal{M}}^s)$ and $\varrho(\boldsymbol{\mathcal{M}}^f)$ for a given CFL number $\mu \in \mathbb{R}_+$ and a given polynomial degree $p_t \in \mathbb{N}$, it is necessary to compute the eigenvalues of

$$\boldsymbol{\mathcal{M}}^s(\theta_x,\theta_t) \in \mathbb{C}^{4N_t\times 4N_t} \quad \text{and} \quad \boldsymbol{\mathcal{M}}^f(\theta_x,\theta_t) \in \mathbb{C}^{4N_t\times 4N_t},$$

with $N_t = p_t + 1$ for all low frequencies $(\theta_x,\theta_t) \in \Theta^{low,s}_{\ell_x,\ell_t}$ respectively $(\theta_x,\theta_t) \in \Theta^{low,f}_{\ell_x,\ell_t}$.

It is difficult to find analytical expressions for the eigenvalues of the two-grid operators $\boldsymbol{\mathcal{M}}^f(\theta_x,\theta_t)$ and $\boldsymbol{\mathcal{M}}^s(\theta_x,\theta_t)$ since they are the product of several Fourier symbols which itself are complex functions. We therefore compute the eigenvalues numerically for all frequencies $(\theta_x,\theta_t) \in \Theta^{low,f}_{\ell_x,\ell_t}$ and $(\theta_x,\theta_t) \in \Theta^{low,s}_{\ell_x,\ell_t}$. We consider a space-time discretization with N_x volumes in space and N space-time slabs on the domain $[0,1]\times[0,T]$, where we adapt T via the CFL number $\mu = \frac{\Delta t}{\Delta x} \in [1,800]$.

The results for the LFA can be seen in Figure 3 for $N_x = 2^5$, $N = 2^3$ to the left and for $N_x = 2^{10}$, $N = 2^3$ to the right, $\mu \in [1,10,50,100,200,400,600,800]$ for both coarsening strategies, referred to as semi and full, and $p_t = 0$ and $p_t = 1$, respectively.

They show that for high CFL numbers the multigrid solver has excellent asymptotic convergence rates about 0.5 for $p_t = 0$ which can be improved to 0.375 by increasing the polynomial degree in time to $p_t = 1$. Moreover, these convergence rates are independent of the coarsening strategy.

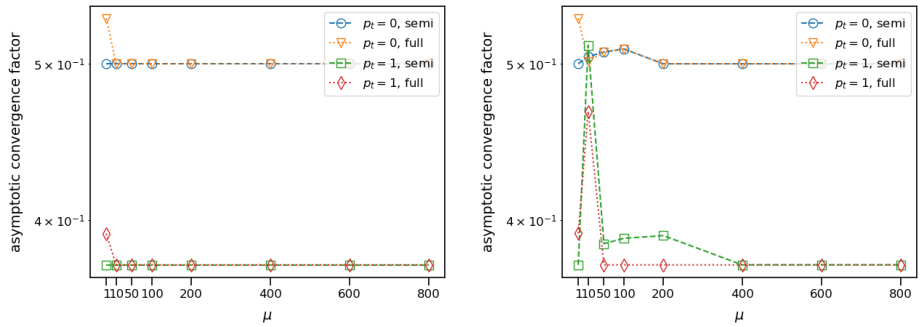


Figure 3: Results of the LFA for the test problem: left for $N_x = 2^5$, $N = 2^3$, right for $N_x = 2^{10}$, $N = 2^3$.

7 Numerical Examples

We now solve the system (5) using this two-grid method. Periodic boundary conditions in space and time, needed to perform the LFA, cannot be used for the numerical tests since this results in singular iteration matrices. We therefore adjust test case (1) and consider the following problems in one respectively two spatial dimensions:

$$u_t + a u_x = 0, \;\; a = 1, \;\; (x,t) \in (0,1] \times (0,T], \tag{32}$$

with solution $u(x,t) = \sin(\pi(x-t))$, and

$$\mathbf{u}_t + \mathbf{a} \cdot \mathbf{u}_x = \mathbf{0}, \;\; \mathbf{a} = (1,1], \;\; (\mathbf{x},t) \in (0,1]^2 \times [0,T], \tag{33}$$

with solution $\mathbf{u}(\mathbf{x},t) = \sin(\pi(x_1-t))\sin(\pi(x_2-t))$. Moreover, we consider a full space-time DG-SEM, i.e. a DG-SEM in space and time. As before, the time interval is determined via $T = N\mu\Delta x$.

All numerical tests in this section are performed using the Python interface of DUNE [6] on an Intel Xeon E5-2650 v3 processor (Haswell) on the LUNARC Aurora cluster at Lund University.

We calculate the asymptotic convergence rate from 60 multigrid iterations by

$$\max_{i=1,\dots,59} \frac{\|\mathbf{r}^{i+1}\|_2}{\|\mathbf{r}^i\|_2}, \;\; \mathbf{r}^i = \underline{\mathbf{L}}_{\tau,\xi}\underline{\mathbf{u}}^i - \underline{\mathbf{b}}.$$

The results for one spatial dimension can be seen in Figure 4 to the left, with $N = 2^3$ and $N_x = 2^{10}$. The convergence rates converge for both coarsening strategies to approximately 0.25 for $p_t = p_x = 0$ and to approximately 0.3 for $p_t = p_x = 1$. The CFL number to achieve these convergence rates increases when increasing the order of the polynomial approximation. Moreover, we get slightly higher convergence rates for small CFL numbers when using the semi-coarsening strategy.

Increasing the number of spatial dimensions we measure the numerical convergence rates for $N = 2^3$ and $N_x = N_y = 2^5$. The results can be seen in Figure 4 to the right. Here, the convergence rates for both coarsening strategies and different DG orders are

very similar, converging to approximately 0.25. However, we notice some oscillations for the semi-coarsening ansatz with $p_t = p_x = 1$. This might vanish when increasing the CFL number. While the numerical convergence rates are similar to the one-dimensional case for $p_t = p_x = 0$, they improve slightly for $p_x = p_t = 1$ when increasing the number of spatial dimensions.

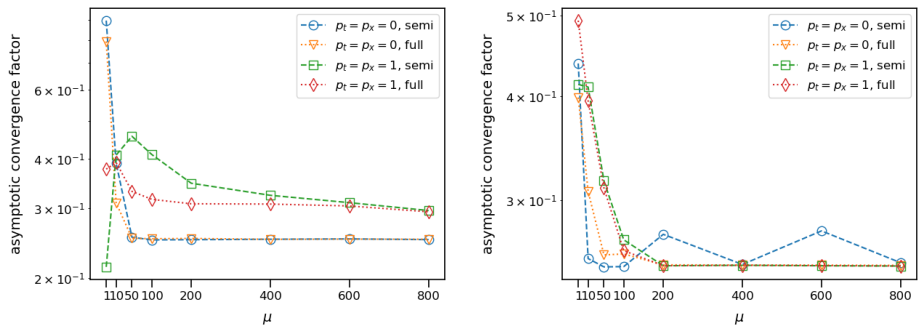


Figure 4: Numerical convergence results: left for one spatial dimension, right for two spatial dimensions.

We now fix $CFL = 600$ and vary the number of spatial elements to study the grid independence of the multigrid solver. The results can be seen in Figure 5. For $p_t = p_x = 1$ we can conclude a grid independence, while the convergence rate increases slightly when increasing the order of the DG approximation.

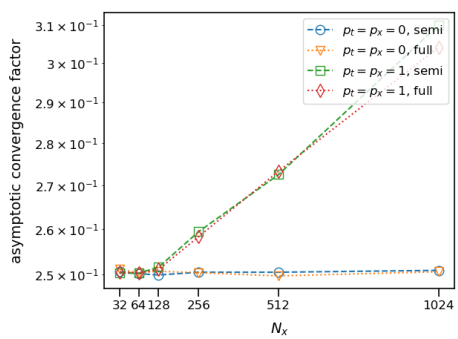


Figure 5: Numerical convergence results two spatial dimension, $\mu = 600$, $N = 2^5$

8 Conclusions

In this article we have applied the LFA to a space-time multigrid solver for the advection equation discretized with a space-time DG method. With the help of the analysis we calculated asymptotic convergence factors for the smoother and the two-grid method. The

resulting Fourier symbols are complex since the spatial FV discretization with upwind flux results in a non-symmetric operator. For large CFL numbers we could analytically find promising asymptotic smoothing factors converging to $\frac{1}{\sqrt{2}}$ with increasing CFL number for both coarsening strategies independent of the temporal DG-SEM order. As for the smoother, it was difficult to find analytical expressions for the two-grid asymptotic convergence rates since they are based on the product of several complex Fourier symbols. We therefore calculated these numerically. The LFA gave excellent asymptotic convergence rates converging to 0.5 for $p_t = 0$ and decreasing to 0.375 for $p_t = 1$ for higher CFL numbers after some oscillations for small CFL numbers. The influence of the coarsening strategies on the convergence rates is minimal, with semi-coarsening in time resulting in slightly better asymptotic convergence rates for smaller CFL numbers.

For the numerical tests we considered non-periodic advection problems in one and two spatial dimensions with a space-time DG-SEM approximations and executed the numerical experiments in DUNE. We obtained asymptotic convergence rates of approximately 0.25 for $p_t = p_x = 0$ and 0.3 for $p_t = p_x = 1$ and high CFL numbers, independent of the coarsening strategy in the one-dimensional case. For two dimensions, asymptotic convergence rates of approximately 0.25 were measured for high CFL numbers, independent of the DG-SEM order and the coarsening strategy

The tests showed that the theoretical asymptotic convergence rates from the LFA were slightly larger than the convergence rates obtained in the numerical experiments. This can be explained by the different boundary conditions and more dimensions considered for in numerical experiments. Moreover, the coarsening strategy does not influence the results very much and simple block Jacobi smoothers can be used to get smoothing factors of $\frac{1}{\sqrt{2}}$.

However, solving the resulting space-time system at once results in large systems and it is thus advisable to either parallelize the solver or use a block multigrid solver for each space-time block.

Acknowledgements

Gregor Gassner has been supported by the European Research Council (ERC) under the European Union's Eights Framework Program Horizon 2020 with the research project Extreme, ERC grant agreement no. 714487.

References

[1] P. Bastian, M. Blatt, A. Dedner, N.-A. Dreier, C. Engwer, R. Fritze, C. Gräser, C. Grüninger, D. Kempf, R. Klöfkorn, et al. The Dune framework: Basic concepts and recent developments. *Comput. Math. Appl.*, 81:75–112, 2021.

[2] P. Birken. *Numerical Methods for Unsteady Compressible Flow Problems.* CRC Press, 2021.

[3] P. D. Boom and D. W. Zingg. High-Order Implicit Time-Marching Methods Based on Generalized Summation-by-Parts Operators. *SIAM J. Sci. Comput.*, 37:A2682–A2709, 2015.

[4] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Math. Comp.*, 31(138):333–390, 1977.

[5] A. Dedner and R. Klöfkorn. Extendible and Efficient Python Framework for Solving Evolution Equations with Stabilized Discontinuous Galerkin Methods. *Commun. Appl. Math. Comput.*, pages 1–40, 2021.

[6] A. Dedner, R. Klöfkorn, and M. Nolte. Python bindings for the dune-fem module. *Zenodo (March 2020)*, 2020.

[7] A. Dedner, R. Klöfkorn, M. Nolte, and M. Ohlberger. A generic interface for parallel and adaptive discretization schemes: abstraction principles and the dune-fem module. *Computing*, 90:165–196, 2010.

[8] W. Dörfler, S. Findeisen, and C. Wieners. Space-time discontinuous Galerkin discretizations for linear first-order hyperbolic evolution systems. *Comput. Methods Appl. Math.*, 16(3):409–428, 2016.

[9] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, J. B. Schroder, and S. Vandewalle. Multigrid methods with space–time concurrency. *Comput. Vis. Sci.*, 18(4-5):123–143, 2017.

[10] S. Friedhoff, S. MacLachlan, and C. Borgers. Local Fourier analysis of space-time relaxation and multigrid schemes. *SIAM J. Sci. Comput.*, 35(5):S250–S276, 2013.

[11] L. Friedrich, G. Schnücke, A. R. Winters, D. C. R. Fernández, G. J. Gassner, and M. H. Carpenter. Entropy Stable Space–Time Discontinuous Galerkin Schemes with Summation-by-Parts Property for Hyperbolic Conservation Laws. *J. Sci. Comput.*, 80(1):175–222, 2019.

[12] M. J. Gander. 50 Years of Time Parallel Time Integration. In T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, editors, *Multiple Shooting and Time Domain Decomposition Methods*, pages 69–113, Cham, 2015. Springer International Publishing.

[13] M. J. Gander and M. Neumüller. Analysis of a new space-time parallel multigrid algorithm for parabolic problems. *SIAM J. Sci. Comput.*, 38(4):A2173–A2208, 2016.

[14] J. Gopalakrishnan and G. Kanschat. A multilevel discontinuous Galerkin method. *Numer. Math.*, 95(3):527–550, 2003.

[15] B. Gustafsson. *High order difference methods for time dependent PDE*, volume 38. Springer Science & Business Media, 2007.

[16] W. Hackbusch. Parabolic multigrid methods. In R. Glowinski and J.-L. Lions, editors, *Computing Methods in Applied Sciences and Engineering IV*, pages 189–197. Elsevier Science Publisher B.V., Noth-Holland, 1984.

[17] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media, 2006.

[18] E. Hairer and G. Wanner. *Solving ordinary differential equations II*, volume 14 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2010.

[19] P. W. Hemker, W. Hoffmann, and M. Van Raalte. Two-level Fourier analysis of a multigrid approach for discontinuous Galerkin discretization. *SIAM J. Sci. Comput.*, 25(3):1018–1041, 2003.

[20] P. W. Hemker, W. Hoffmann, and M. Van Raalte. Fourier two-level analysis for discontinuous Galerkin discretization with linear elements. *Numer. Linear Algebra Appl.*, 11(5-6):473–491, 2004.

[21] J. S. Hesthaven, S. Gottlieb, and D. Gottlieb. *Spectral methods for time-dependent problems*, volume 21. Cambridge University Press, 2007.

[22] L. O. Jay. Lobatto methods. In B. Engquist, editor, *Encyclopedia of Applied and Computational Mathematics*, pages 817–826. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[23] C. M. Klaij, M. H. van Raalte, H. van der Ven, and J. J. van der Vegt. h-Multigrid for space-time discontinuous Galerkin discretizations of the compressible Navier-Stokes equations. *J. Comput. Phys.*, 227(2):1024–1045, 2007.

[24] M. Neumüller. *Space-Time Methods*, volume 20 of *Monograph Series TU Graz: Computation in Engineering and Science*. TU Graz, 2013.

[25] J. J. Sudirham, J. J. W. van der Vegt, and R. M. J. van Damme. Space-time discontinuous Galerkin method for advection-diffusion problems on time-dependent domains. *Appl. Numer. Math.*, 56(12):1491–1518, 2006.

[26] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid*. Elsevier Ldt., 2001.

[27] J. Van der Vegt. *Space-time discontinuous Galerkin finite element methods*, pages 1–37. VKI Lecture Series. Von Karman Institute for Fluid Dynamics, 2006. Conference date: 14-11-2005 Through 18-11-2005.

[28] J. van der Vegt and S. Rhebergen. hp-Multigrid as Smoother algorithm for higher order discontinuous Galerkin discretizations of advection dominated flows. Part II: Optimization of the Runge–Kutta smoother. *J. Comput. Phys.*, 231:7564–7583, 2012.

[29] J. J. van der Vegt and S. Rhebergen. hp-multigrid as smoother algorithm for higher order discontinuous Galerkin discretizations of advection dominated flows: Part I. Multilevel analysis. *J. Comput. Phys.*, 231(22):7537–7563, 2012.

[30] J. J. van der Vegt and H. van der Ven. Space-time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows. I. General formulation. *J. Comput. Phys.*, 182(2):546–585, 2002.

[31] J. J. van der Vegt and Y. Xu. Space–time discontinuous Galerkin method for nonlinear water waves. *J. Comput. Phys.*, 224(1):17–39, 2007.

[32] H. Van der Ven and J. J. van der Vegt. Space–time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows: II. Efficient flux quadrature. *Comput. Methods Appl. Mech. Engrg.*, 191(41-42):4747–4780, 2002.

[33] M. Van Raalte and P. W. Hemker. Two-level multigrid analysis for the convection–diffusion equation discretized by a discontinuous Galerkin method. *Numer. Linear Algebra Appl.*, 12(5-6):563–584, 2005.

[34] P. Wesseling. *An Introduction to Multigrid Methods*. An Introduction to Multigrid Methods. R.T. Edwards, 2004.

Printed by Media-Tryck, Lund 2022 NORDIC SWAN ECOLABEL 3041 0903

LUND UNIVERSITY

Doctoral Theses in Mathematical Sciences 2022:1
ISBN 978-91-8039-154-2
ISSN 1404-0034
LUNFNA-1010-2022