The Dissertation Committee for Sriramkrishnan Muralikrishnan certifies that this is the approved version of the following dissertation:

Fast and Scalable solvers for High-Order Hybridized Discontinuous Galerkin Methods with applications to Fluid Dynamics and Magnetohydrodynamics

Committee:

Tan Bui-Thanh, Supervisor

Leszek F. Demkowicz

Omar Ghattas

Laxminarayan L. Raja

John N. Shadid

François L. Waelbroeck

Mary F. Wheeler

Fast and Scalable solvers for High-Order Hybridized Discontinuous Galerkin Methods with applications to Fluid Dynamics and Magnetohydrodynamics

by

Sriramkrishnan Muralikrishnan

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2019

Dedicated to my mom Valli,

my late great grandmother Rukmani,

and my late grandfather Gopalakrishnan Thiruvankatam.

Acknowledgments

This thesis would not be possible without the following people and I would like to thank them now. The first person I would like to thank the most is my advisor Prof. Tan Bui-Thanh. He brought me to UT Austin and gave me this wonderful PhD opportunity for which I am always grateful. He always encouraged me whenever I came up with new ideas and results and it gave me the boost to work even harder. On the other hand, he also pointed out my flaws directly and this helped me to correct them. I think this is the most important and helpful part, as usually you don't get that many people around you who can do this. His passion for research is incomparable and I always remember the times when we worked for hours together on proofs, papers and ideas. I don't know how many advisors would reply 06:00 AM on a Sunday morning on an idea I sent the previous night. He also used to call me many times from his car while driving or while waiting in airport because some idea popped up and this passion inspired me so much. Though at some times it was hard to work with him especially because of his very high standards for everything, I don't think I would have been happier with any other advisor either and in that respect I am very fortunate. I learnt so many things from him apart from research like physical fitness and hope I could follow them in my career.

I would like to thank my other committee members Prof. Wheeler, Prof. Demkowicz, Prof. Ghattas, Prof. Raja, Prof. Waelbroeck and Dr. Shadid for being a part of it. I am very fortunate to have so many of these world class experts in my thesis committee and if I am not wrong this is one of the largest PhD committees I have known so far in UT. I want to thank Prof. Demkowicz for his countless

recommendation letters which helped me to get my post-doc position. He is very kind and also the best math professor I ever had. I am very fortunate that I took his classes and was able to interact with him closely. I cannot thank him enough for all the help and I am always grateful for that.

I can only say that I am incredibly lucky to know Prof. Wheeler. She is amazing and have shown extreme care, affection, encouragement and support throughout my PhD. She is one of the legends in FEM and her humility and humbleness always surprises me. She took special care in my research and every time I talk to her I feel rejuvenated. Her life is an inspiration to everyone including me. Prof. Ghattas' class is arguably one of the best classes I have attended in my PhD (even though technically I audited it). His passion for teaching and research is what I want to follow in my academic career. I want to thank Dr. Shadid for the MHD part in my thesis. His knowledge in solvers for MHD especially AMG and plasma physics was very helpful to me and I am very glad that I got a chance to collaborate with him. I want to thank Prof. Raja for his kindness and support throughout my PhD and Prof. Waelbroeck for the fruitful discussions we had regarding fusion energy and plasma sciences.

I want to thank my collaborators Dr. Minh-Binh Tran and Dr. Tim Wildey for the projects we worked together and it was a great experience. Thanks to Dr. Wildey for all the recommendation letters which helped me to get my post-doc. I would like to thank Sue for helping with all the travels, conferences and also for taking special care towards our graduation. I want to thank Dr. Hari Sundar for sharing his homg library, on top of which I implemented many of our algorithms. I also want to thank TACC for processing all our requests so quickly and giving us one of the world's fastest supercomputers.

It is time to thank my colleagues and friends. I want to thank Shinhoo Kang

for traveling with me throughout this PhD. He is like a brother to me and we have seen so much happiness and sadness together. I can only thank God for having him also join in my group at the same time I joined and now I am very happy that we are graduating together. He has always been a huge pillar of support to me and I cannot thank him enough for that. I would like to thank Stephen Shannon for all the interesting discussions we had, for sharing his code on MHD so that I could implement my algorithms on top of it.

Among my friends, I would like to thank Premkumar for countless discussions we had on almost everything. He made me felt that I am back in my home town as we connect on so many things and he is also a wonderful human being. Prem and my other dear friends Ashish, Mahesh, Janaki and Vivek made my Austin life so memorable and beautiful in the past couple of years. I always remember the countless dinner sessions we had, games we played and other activities we did together and all those memories will always remain fresh inside me. I want to thank them for all of this and help me get a social life apart from my PhD. I also would like to thank my friends Young Joon, Jason and Daiju for all the trips we went and the lunches. My other friends in Austin/PhD time Harpreet, Sadhika, Gurpreet, Sundeep, Rituparna, Vinod special thanks to each one of you.

I would like to thank my first year roommates Memo and Sahil. Memo has helped me in so many ways and especially when I knew no one in Austin he even picked me up in his car the first day I arrived. He is a kind-hearted man and I wish him well for his married life. I would like to thank Vishwas for all the help. He is also like another brother to me and he make sure that I am doing ok all the times. I cannot thank him enough for all the help during my job search among other things.

I would like to thank my friends in India especially my best friend Manickam.

I am very fortunate to have him as my friend and after I came to US whenever we talk, for some reason, I felt more connected probably because we are somehow in the same place in our lives and our thoughts were in the same wavelength. I remember all the long talks we had and I am looking forward to meeting him again in India.

Last but not the least, I would like to thank my mom, dad, sister, brother-in-law, my two nephews and the rest of my family for the support they have given me all along. Thanks Amma (mom) for listening to all my complaints patiently every day and telling me repeatedly that I will get through everything. Many days when I am stressed her sense of humor and jokes only relieve me. Saranya, I know we didn't talk much as both of us were busy in our own lives but thank you so much for making all my India visits so memorable and once I return to India I hope we will spend more time together. Appa (dad), unlike Amma we also didn't talk much over phone but I always felt you were there for me in the times I needed the most and that's what matters the most. I thank God every day for giving me such an amazing family and feel very fortunate for that. Thatha (Grandfather) I miss you very much and cannot believe that you are not there to see me graduating, but I understand this is life and somehow you prepared me for this long time back itself. I still believe you are there in my core thoughts of everything and you have taught me a way of living which I will never forget. This thesis is for you and mom the two special people in my life. This list is already so long and I hope I have thanked most of the people if not everyone, If I have left anyone here I sincerely apologize for that and thanks to them and God.

This work was partially supported by DOE grants DE-SC0010518, DE-SC0011118, DE-SC0018147 and NSF Grant DMS-1620352. I am grateful for the support.

Fast and Scalable solvers for High-Order Hybridized Discontinuous Galerkin Methods with applications to Fluid Dynamics and Magnetohydrodynamics

by

Sriramkrishnan Muralikrishnan, Ph.D.
The University of Texas at Austin, 2019

Supervisor: Tan Bui-Thanh

The hybridized discontinuous Galerkin methods (HDG) introduced a decade ago is a promising candidate for high-order spatial discretization combined with implicit/implicit-explicit time stepping. Roughly speaking, HDG methods combines the advantages of both discontinuous Galerkin (DG) methods and hybridized methods. In particular, it enjoys the benefits of equal order spaces, upwinding and ability to handle large gradients of DG methods as well as the smaller globally coupled linear system, adaptivity, and multinumeric capabilities of hybridized methods. However, the main bottleneck in HDG methods, limiting its use to small to moderate sized problems, is the lack of scalable linear solvers. In this thesis we develop fast and scalable solvers for HDG methods consisting of domain decomposition, multigrid and multilevel solvers/preconditioners with an ultimate focus on simulating large scale problems in fluid dynamics and magnetohydrodynamics (MHD).

First, we propose a domain decomposition based solver namely iterative HDG for partial differential equations (PDEs). It is a fixed point iterative scheme, with

each iteration consisting only of element-by-element and face-by-face embarrassingly parallel solves. Using energy analysis we prove the convergence of the schemes for scalar and system of hyperbolic PDEs and verify the results numerically.

We then propose a novel geometric multigrid approach for HDG methods based on fine scale Dirichlet-to-Neumann maps. The algorithm combines the robustness of algebraic multigrid methods due to operator dependent intergrid transfer operators and at the same time has fixed coarse grid construction costs due to its geometric nature. For diffusion dominated PDEs such as the Poisson and the Stokes equations the algorithm gives almost perfect $hp-$scalability.

Next, we propose a multilevel algorithm by combining the concepts of nested dissection, a fill-in reducing ordering strategy, variational structure and high-order properties of HDG, and domain decomposition. Thanks to its root in direct solver strategy the performance of the solver is almost independent of the nature of the PDEs and mostly depends on the smoothness of the solution. We demonstrate this numerically with several prototypical PDEs.

Finally, we propose a block preconditioning strategy for HDG applied to incompressible visco-resistive MHD. We use a least squares commutator approximation for the inverse of the Schur complement and algebraic multigrid or the multilevel preconditioner for the approximate inverse of the nodal block. With several 2D and 3D transient examples we demonstrate the robustness and parallel scalability of the block preconditioner.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Computational science and engineering has revolutionized the developments in almost all fields of science and engineering [131]. With the availability of extreme scale architectures we now can simulate extremely complex multiphysics applications such as magnetohydrodynamics (MHD). These applications usually have disparate spatial and temporal scales which necessitate the use of high-order methods in both spatial and temporal discretizations to capture the scales accurately. Moreover, the presence of fast waves in many of these applications necessitates the use of fully implicit or implicit-explicit time stepping to avoid otherwise overly restrictive small stepsizes with the explicit methods. High-order methods also make the most efficient use of the extreme scale architectures because of their high computation to communication ratio.

Hybridized discontinuous Galerkin (HDG) methods introduced a decade ago [38] together with high-order implicit time integrators [87] is one of the promising combinations. HDG methods have now been developed for a wide range of PDEs including, but not limited to, Poisson-type equation [38, 39], Stokes equation [36, 113, 126], Euler and Navier–Stokes equations [115, 106, 127], wave equations [117, 116, 76, 93]. In [25, 26, 27], an upwind HDG framework was proposed that provides a unified and a systematic construction of HDG methods for a large class of PDEs.

Roughly speaking, HDG methods combine the advantages of hybrid(ized) methods [130, 71, 23] and discontinuous Galerkin (DG) discretizations [95, 154, 5, 7].

In particular, its inherent characteristics from DG include: (i) arbitrary high-order with compact stencil; (ii) ability to handle complex geometries; (iii) local conservation; and (iv) upwinding for hyperbolic systems. On the other hand, it also possesses the advantages of hybrid(ized) methods, namely, (i) having smaller and sparser linear system for steady state problems or time-dependent problems with implicit time integrators; (ii) *hp*-adaptivity-ready using the trace space; (iii) facilitating multinumerics with different hybrid(ized) methods in different parts of the domain; and (iv) when applicable, providing superconvergence by local post-processing [6, 38].

As any implicit method is as good as the preconditioner/solver available to solve the resulting linear(ized) system, the main challenge facing HDG methods is, the construction of scalable solvers/preconditioners. The linear systems resulting from HDG methods are different from volume based discretizations such as continuous Galerkin (CG), stabilized finite element methods (FEM) and DG methods in the sense that, it contains only the trace, or skeletal unknowns, which live on the skeleton of the mesh. This presents challenges in the construction of solvers/preconditioners as they are usually developed in the context of volume based discretizations. Moreover, the high-order aspect of the method makes the linear systems highly non-diagonally dominant even for simple equations like Poisson [91]. Our main objective in this thesis is to develop fast, scalable linear solvers/preconditioners for linear systems arising from HDG discretizations for a wide variety of problems appearing in fluid dynamics and MHD. We next review some existing efforts that are relevant to the developments in this thesis.

1.1 Review on solvers/preconditioners for HDG trace system

The first geometric multigrid solver for HDG methods was introduced in [41]. It is based on the multigrid introduced for hybridized mixed methods [74]. The main idea is to transfer the residual from HDG skeletal space to linear continuous Galerkin FEM space, and then carry out the standard geometric or algebraic multigrid algorithm. Even though the algorithm works well for Poisson equation as demonstrated in [41] there are few issues which needs to be addressed.

First, it is a non-inherited algorithm in the sense that the coarse scale operators do not inherit all the properties of the fine scale ones. For example, in case of subsurface flows through porous media one of the important properties which needs to be satisfied is local conservation [155]. HDG methods satisfy this property whereas the CG methods do not have local conservation (but global conservation). The violation of local conservation leads to non-physical results for CG methods as reported in [155] and the references therein. Thus in the multigrid algorithm for HDG, transferring of information from skeletal space to CG space may affect the local conservation property of the HDG method especially when the tolerance of the iterative solver is not strict.

Secondly, we can form the coarse grid operators for multigrid methods either by re-discretizing the equation in the coarse mesh or by means of Galerkin coarse grid correction [145]. In [91], the authors observed that using the Galerkin coarse grid operator for the multigrid algorithm in [41] performs much better (in terms of number of iterations) compared to constructing coarse grid operators by re-discretization in [41]. However, Galerkin coarse grid operators have limitations in terms of memory, especially for high orders and large scale problems. Thus with this algorithm one has to compromise one of the aspects whereas in the classical multigrid algorithm for CG

methods, since the coarse-grid operators formed through both approaches coincide, there is no difference and the performance is optimal.

Finally, we know the CG space is not suitable for convection-dominated problems or for vector equations like Stokes where we need inf-sup stable FEM spaces. Hence, it is not clear how to extend the multigrid algorithm [41] for these kind of problems without affecting the stability of coarse grid problems.

In [32], the multigrid algorithm in [41] with few modifications was pursued for the simulation of high frequency Helmholtz equation discretized by HDG. The main difference, however, is the authors followed the approach in [55] and used GMRES as smoother for coarse grids whereas simple smoothers like Jacobi or Gauss-Seidel are used for the fine grids. The reason is, GMRES being a strong smoother is able to handle strong indefiniteness of the Helmholtz problem on coarse grids where the simple smoothers fail. The coarse grid level in which the smoothing is changed from Gauss-Seidel to GMRES is chosen based on the wavenumber, meshsize and solution order. The performance of the algorithm was found to be mesh-independent, but mild dependence with wave-number was observed.

One level Schwarz type domain decomposition algorithms in the context of HDG and hybridizable interior penalty DG schemes have been studied for elliptic equation [60, 61, 58] and Maxwell's equations [96, 78]. In [61, 58], the authors derived optimized interface coefficients for accelerating the convergence of Schwarz methods in the context of Poisson equation discretized by hybridizable interior penalty DG schemes whereas similar efforts have been carried out in [78] for HDG discretization of Maxwell's equations. The number of iterations for these schemes still increases with the increase in sub-domains due to lack of coarse solvers.

A balancing domain decomposition by constraints (BDDC) algorithm for HDG

was introduced in [49] and studied for Euler and Navier–Stokes equations. BDDC algorithms are capable of delivering iteration counts independent of mesh refinements, if the subdomain size and meshsize are chosen appropriately. For a range of CFD problems the algorithm was applied and parallel scaling performance up to 30 processors was shown.

More recently a block preconditioner for the HDG discretization of Stokes system was presented in [128]. The authors eliminated only the volume velocity unknowns from the full HDG system with volume and trace unknowns and hence the final linear system contains the volume pressure unknowns in addition to the velocity and pressure trace unknowns. Scalability with respect to mesh refinements was demonstrated for low order solutions.

1.2 Objective and Contributions

The objective of this work is to develop fast, scalable and robust solvers for high-order HDG discretizations applied to a wide variety of problems in fluid dynamics and MHD. We consider both steady state problems and time dependent problems discretized with implicit time integrators. Availability of these solvers is critical for applicability of high-order HDG schemes for large scale complex multiphysics applications. In the construction of solvers/preconditioners we make efforts to utilize the variational structure of HDG methods or more generally hybridized methods as much as possible. High-order and parallel scalability are two of the most important factors we take into consideration in the development. The contributions of this thesis are now briefly summarized:

- **A domain decomposition solver namely iterative HDG (iHDG) for PDEs**. It is a fixed point iterative scheme, with each iteration consisting only

of element-by-element and face-by-face embarrassingly parallel solves. The algorithm is provably convergent and we show the convergence of these schemes for the transport, the linearized shallow water, the convection-diffusion and the Poisson equations. The scheme involves only local matrices, has excellent strong scaling and can also be used as a smoother in multilevel/multigrid schemes.

- **A novel geometric multigrid approach for HDG methods based on fine scale Dirichlet-to-Neumann (DtN) maps**. The algorithm combines the robustness of algebraic multigrid methods due to operator dependent inter-grid transfer operators and at the same time has fixed coarse grid construction costs due to its geometric nature. It also avoids explicit upscaling of parameters as it only uses fine scale DtN maps. It is applicable to both structured and unstructured meshes and does not require them to be nested. For scalar and vector diffusion problems the algorithm gives almost perfect $hp-$scalability even for flow through porous media with highly heterogeneous and discontinuous coefficients.

- **A multilevel preconditioner for generic hyperbolic system of PDEs**. The algorithm combines the concepts of nested dissection, a fill-in reducing ordering strategy, variational structure and high-order properties of HDG methods and domain decomposition concepts. The performance of the multilevel preconditioner appears to be independent of the nature of the PDE and mostly depends on the smoothness of the solution.

- **A block preconditioning strategy for the HDG trace systems applied to the incompressible resistive MHD**. The preconditioner uses a least squares commutator approximation for the inverse of the Schur complement and algebraic multigrid with GMRES smoother or the multilevel preconditioner

for the approximate inverse of the nodal block. For several 2D and 3D transient examples from MHD, including, but not limited to the island coalescence problem at high Lundquist numbers the preconditioner is robust and scalable.

1.3 Outline

In this thesis we first start with proposing simple fixed point iterative schemes for HDG in chapter 2. These come under the category of one level domain decomposition methods. With rigorous energy analysis we prove the convergence of these schemes for the transport, the linearized shallow water and the convection-diffusion equations. We present several 2D and 3D numerical results verifying the theoretical estimates derived. In chapter 3 we improve the scheme introduced in chapter 2 for diffusion dominated equations and system of hyperbolic equations. Again with theoretical and supporting numerical examples we study the performance of the scheme and also compare it with the scheme in chapter 2. We move on to introduce a geometric multigrid approach in chapter 4 designed specifically for trace systems resulting from hybridized methods. We prove the stability of the intergrid transfer operators and also show that the coarse grid operator formed through Galerkin coarse grid correction in the algorithm is also a Dirichlet-to-Neumann map on that level. With several numerical examples from scalar and vector equations we show the $hp-$scalability of these schemes. In chapter 5 we introduce a multilevel scheme for HDG combining the concepts of nested dissection, a fill-in reducing ordering strategy, high-order and variational properties of the HDG scheme and domain decomposition ideas. We show that the iterative multilevel algorithm can also be interpreted as a multigrid scheme with specific intergrid transfer and smoothing operators. We derive the theoretical complexity of these schemes and compare the performance with several numerical results. In chapter 6 we develop a block preconditioning strategy for the trace systems

resulting from HDG discretizations of the incompressible resistive MHD. With several 2D and 3D transient examples from MHD the robustness and parallel scalability of the preconditioning strategy is demonstrated. Finally, in chapter 7 we summarize our work in this thesis and also provide possible directions for future research.

Chapter 2

iHDG: An Iterative HDG Solver for Partial Differential Equations

2.1 Domain decomposition applied to HDG

In this chapter[1], we blend the HDG method and the Schwarz idea [122] to create an one level domain decomposition iterative solver for HDG discretizations namely iterative HDG (iHDG). One of the main features of the proposed approach is that it is provably convergent. From a linear algebra point of view, the method can be understood as a block Gauss–Seidel iterative solver for the augmented HDG system with volume and trace unknowns. Usually the HDG system is not realized from this point of view and the linear system is assembled for trace unknowns only. But for iHDG, since we never form any global matrices it allows us to create an efficient solver which completely depends on independent element-by-element calculations. Traditional Gauss–Seidel schemes for convection-diffusion problems or pure advection problems require the unknowns to be ordered in the flow direction for convergence. Several ordering schemes for these kinds of problems have been developed and studied in [11, 77, 88, 151]. In the context of discontinuous Galerkin methods robust Gauss–Seidel smoothers are developed in [86] and again these smoothers depend on the ordering of the unknowns. For a complex velocity field (e.g. hyperbolic systems) it is, however, not trivial to obtain a mesh and an ordering which coincide with the flow

[1]The contents of this chapter are largely based on the published manuscript [108]. The contributions of the author in the article ranged from numerical implementation of the algorithm, participation in the theoretical analysis and writing the manuscript.

direction. Moreover the point or the block Gauss–Seidel scheme (for the trace system alone) requires a lot of communication between processors for calculations within an iteration. These aspects affect the scalability of these schemes to a large extent and in general are not favorable for parallelization [8, 57].

Unlike traditional Gauss–Seidel methods, which are purely algebraic, the iHDG approach is built upon, and hence exploits, the HDG discretization. Of importance is the upwind flux, or more specifically the upwind stabilization, that automatically determines the flow directions. Consequently *the convergence of iHDG is independent of the ordering of the unknowns*. Another crucial property inherited from HDG is that each iteration consists of only independent element-by-element local solves to compute the volume unknowns. Thanks to the compact stencil of HDG, this is overlapped by a single communication of the trace of the volume unknowns restricted on faces shared between the neighboring processors. The communication requirement is thus similar to that of block-Jacobi methods (for the volume system alone). The iHDG approach is designed with these properties to suit the current and future computing systems with massive concurrencies. We rigorously show that our proposed methods are convergent with explicit contraction constants using an energy approach. Furthermore the convergence rate is independent of the solution order for hyperbolic PDEs. The theoretical findings will be verified on various 2D and 3D numerical results for steady and time-dependent problems.

This chapter is organized as follows. Section 2.2 introduces most of the common notations used throughout this thesis, additional specific notations will be introduced in the subsequent chapters when and where it is required. In section 2.3 we introduce the iHDG algorithm for an abstract system of PDEs discretized by the upwind HDG discretization [25]. The convergence of the iHDG algorithm for the scalar and for the system of hyperbolic PDEs is proved in section 2.4 using an energy

approach. In section 2.5 the convection-diffusion PDE is considered in the first order form and the conditions for the convergence of the iHDG algorithm are stated and proved. Section 2.6 presents various steady and time-dependent examples, in both two and three spatial dimensions, to support the theoretical findings.

2.2 Notation

In this section we first briefly review the upwind HDG framework for a general system of linear PDEs. To begin, let us consider the following system

$$\sum_{k=1}^{d} \partial_k \mathbf{F}_k(\mathbf{u}) + \mathbf{C}\mathbf{u} := \sum_{k=1}^{d} \partial_k (\mathbf{A}_k \mathbf{u}) + \mathbf{C}\mathbf{u} = \mathbf{f} \quad \text{in } \Omega, \tag{2.1}$$

where d is the spatial dimension (which, for clarity of the exposition, is assumed to be $d = 3$ whenever a particular value of the dimension is of concern, but the result is also valid for $d = \{1, 2\}$), $\mathbf{F}_k$ is the kth component of the flux vector (or tensor) $\mathbf{F}$, $\mathbf{u}$ is the unknown solution with values in $\mathbb{R}^m$, and $\mathbf{f}$ is the forcing term. The matrices $\mathbf{A}_k$ and $\mathbf{C}$ are assumed to be continuous[2] across Ω. The notation ∂_k stands for the kth partial derivative and by the subscript k we denote the kth component of a vector/tensor. We will discretize (2.1) using the HDG framework. To that end, let us introduce notation and conventions used in this thesis.

Let us partition $\Omega \in \mathbb{R}^d$, an open and bounded domain, into N_T nonoverlapping elements $T_j, j = 1, \ldots, N_T$ with Lipschitz boundaries such that $\Omega_h := \cup_{j=1}^{N_T} T_j$ and $\overline{\Omega} = \overline{\Omega}_h$. Here, h is defined as $h := \max_{j \in \{1,\ldots,N_T\}} diam(T_j)$. We denote the skeleton of the mesh by $\mathcal{E}_h := \cup_{j=1}^{N_T} \partial T_j$, the set of all (uniquely defined) interfaces e between elements. We conventionally identify $\mathbf{n}^-$ as the outward normal vector on the boundary ∂T of element T (also denoted as T^-) and $\mathbf{n}^+ = -\mathbf{n}^-$ as the outward normal vector of

[2]This assumption is not a limitation but for the simplicity of the exposition.

the boundary of a neighboring element (also denoted as T^+). Furthermore, we use $\mathbf{n}$ to denote either $\mathbf{n}^-$ or $\mathbf{n}^+$ in an expression that is valid for both cases, and this convention is also used for other quantities (restricted) on a face $e \in \mathcal{E}_h$. For the sake of convenience, we denote by $\mathcal{E}_h^\partial$ the set of all boundary faces on $\partial\Omega$, by $\mathcal{E}_h^o := \mathcal{E}_h \setminus \mathcal{E}_h^\partial$ the set of all interior faces, and $\partial\Omega_h := \{\partial T : T \in \Omega_h\}$.

For simplicity in writing we define $(\cdot,\cdot)_T$ as the L^2-inner product on a domain $T \in \mathbb{R}^d$ and $\langle\cdot,\cdot\rangle_T$ as the L^2-inner product on a domain T if $T \in \mathbb{R}^{d-1}$. We shall use $\|\cdot\|_T := \|\cdot\|_{L^2(T)}$ as the induced norm for both cases and the particular value of T in a context will indicate which inner product the norm is coming from. We also denote the ε-weighted norm of a function u as $\|u\|_{\varepsilon,T} := \|\sqrt{\varepsilon}u\|_T$ for any positive ε. We shall use boldface lowercase letters for vector-valued functions and in that case the inner product is defined as $(\mathbf{u},\mathbf{v})_T := \sum_{i=1}^m (\mathbf{u}_i,\mathbf{v}_i)_T$, and similarly $\langle\mathbf{u},\mathbf{v}\rangle_T := \sum_{i=1}^m \langle\mathbf{u}_i,\mathbf{v}_i\rangle_T$, where m is the number of components $(\mathbf{u}_i, i = 1,\ldots,m)$ of $\mathbf{u}$. Moreover, we define $(\mathbf{u},\mathbf{v})_\Omega := \sum_{T\in\Omega_h} (\mathbf{u},\mathbf{v})_T$ and $\langle\mathbf{u},\mathbf{v}\rangle_{\mathcal{E}_h} := \sum_{e\in\mathcal{E}_h} \langle\mathbf{u},\mathbf{v}\rangle_e$ whose induced (weighted) norms are clear, and hence their definitions are omitted. We employ boldface uppercase letters, e.g. $\mathbf{L}$, to denote matrices and tensors. We conventionally use $\mathbf{u}$ ($\mathbf{v}$ and $\hat{\mathbf{u}}$) for the numerical solution and $\mathbf{u}^e$ for the exact solution. We use the terms "skeletal unknowns" and "trace unknowns" interchangeably and they both refer to the unknowns on the mesh skeleton.

We define $\mathcal{P}^p(T)$ as the space of polynomials of degree at most p on a domain T. Next, we introduce two discontinuous piecewise polynomial spaces

$$\mathbf{V}_h(\Omega_h) := \left\{\mathbf{v} \in \left[L^2(\Omega_h)\right]^m : \mathbf{v}|_T \in \left[\mathcal{P}^p(T)\right]^m, \forall T \in \Omega_h\right\},$$
$$\mathbf{\Lambda}_h(\mathcal{E}_h) := \left\{\boldsymbol{\lambda} \in \left[L^2(\mathcal{E}_h)\right]^m : \boldsymbol{\lambda}|_e \in \left[\mathcal{P}^p(e)\right]^m, \forall e \in \mathcal{E}_h\right\},$$

and similar spaces for $\mathbf{V}_h(T)$ and $\mathbf{\Lambda}_h(e)$ by replacing Ω_h with T and $\mathcal{E}_h$ with e. For

scalar-valued functions, we denote the corresponding spaces as

$$V_h(\Omega_h) := \left\{ v \in L^2(\Omega_h) : v|_T \in \mathcal{P}^p(T), \forall T \in \Omega_h \right\},$$
$$\Lambda_h(\mathcal{E}_h) := \left\{ \lambda \in L^2(\mathcal{E}_h) : \lambda|_e \in \mathcal{P}^p(e), \forall e \in \mathcal{E}_h \right\}.$$

Following [25], we introduce an upwind HDG discretization for (2.1) as follows: for each element T, the DG local unknown $\mathbf{u}$ and the extra "trace" unknown $\hat{\mathbf{u}}$ need to satisfy

$$-(\mathbf{F}(\mathbf{u}), \nabla \mathbf{v})_T + \left\langle \hat{\mathbf{F}}(\mathbf{u}, \hat{\mathbf{u}}) \cdot \mathbf{n}, \mathbf{v} \right\rangle_{\partial T} + (\mathbf{C}\mathbf{u}, \mathbf{v})_T = (\mathbf{f}, \mathbf{v})_T \quad \forall \mathbf{v} \in \mathbf{V}_h(T), \tag{2.2a}$$
$$\left\langle [\![\hat{\mathbf{F}}(\mathbf{u}, \hat{\mathbf{u}}) \cdot \mathbf{n}]\!], \boldsymbol{\mu} \right\rangle_e = \mathbf{0} \quad \forall e \in \mathcal{E}_h^o, \quad \forall \boldsymbol{\mu} \in \mathbf{\Lambda}_h(e), \tag{2.2b}$$

where we have defined the "jump" operator for any quantity $(\cdot)$ as $[\![(\cdot)]\!] := (\cdot)^- + (\cdot)^+$. We also define the "average" operator $\{\!\{(\cdot)\}\!\}$ via $2\{\!\{(\cdot)\}\!\} := [\![(\cdot)]\!]$. The upwind HDG flux as introduced in [25] is defined by

$$\hat{\mathbf{F}} \cdot \mathbf{n} = \mathbf{F}(\mathbf{u}) \cdot \mathbf{n} + |\mathbf{A}|(\mathbf{u} - \hat{\mathbf{u}}), \tag{2.3}$$

with[3] the matrix $\mathbf{A} := \sum_{k=1}^{d} \mathbf{A}_k \mathbf{n}_k = \mathbf{R}\mathbf{S}\mathbf{R}^{-1}$, and $|\mathbf{A}| := \mathbf{R}|\mathbf{S}|\mathbf{R}^{-1}$. Here $\mathbf{n}_k$ is the kth component of the outward normal vector $\mathbf{n}$ and $|\mathbf{S}|$ represents a matrix obtained by taking the absolute value of the main diagonal of the matrix $\mathbf{S}$.

2.3 The idea of iHDG

The key idea behind the iHDG approach is the following. The approximation of the HDG solution at the $(k+1)$th iteration is governed by the local equation (2.2a) as

$$-\left(\mathbf{F}\left(\mathbf{u}^{k+1}\right), \nabla \mathbf{v}\right)_T + \left\langle \mathbf{F}\left(\mathbf{u}^{k+1}\right) \cdot \mathbf{n} + |\mathbf{A}|\mathbf{u}^{k+1} - |\mathbf{A}|\hat{\mathbf{u}}^k, \mathbf{v} \right\rangle_{\partial T} + (\mathbf{C}\mathbf{u}, \mathbf{v})_T = (\mathbf{f}, \mathbf{v})_T, \tag{2.4a}$$

[3] We assume that $\mathbf{A}$ admits an eigendecomposition, and this is valid for a large class of PDEs of Friedrichs type, for example.

where the weighted trace $|\mathbf{A}|\,\hat{\mathbf{u}}^k$ (not the trace itself) is computed using information at the kth iteration via the conservation condition (2.2b), i.e.,

$$\left\langle |\mathbf{A}|\,\hat{\mathbf{u}}^k, \boldsymbol{\mu} \right\rangle_{\partial T} = \left\langle \left\{\!\!\left\{ |\mathbf{A}|\,\mathbf{u}^k \right\}\!\!\right\} + \left\{\!\!\left\{ \mathbf{F}\left(\mathbf{u}^k\right) \cdot \mathbf{n} \right\}\!\!\right\}, \boldsymbol{\mu} \right\rangle_{\partial T}. \tag{2.4b}$$

Algorithm 1 summarizes the iHDG approach.

Algorithm 1 The iHDG approach.

Ensure: Given initial guess $\mathbf{u}^0$, compute the weighted trace $|\mathbf{A}|\,\hat{\mathbf{u}}^0$ using (2.4b).
1: **while** not converged **do**
2: Solve the local equation (2.4a) for $\mathbf{u}^{k+1}$ using the weighted trace $|\mathbf{A}|\,\hat{\mathbf{u}}^k$.
3: Compute $|\mathbf{A}|\,\hat{\mathbf{u}}^{k+1}$ using (2.4b).
4: Check convergence. If yes, **exit**; otherwise **set** $k = k + 1$ and **continue**.
5: **end while**

The appealing feature of iHDG, Algorithm 1, is that each iteration requires only *independent local solve* (2.4a) *element-by-element, completely independent of each other*. The method exploits the structure of HDG in which each local solve is well defined as long as the trace $\hat{\mathbf{u}}^k$ is given. Furthermore, the global solve via the conservation condition (2.2b) is not needed. Instead, we compute the weighted trace $|\mathbf{A}|\,\hat{\mathbf{u}}^k$ *face-by-face (on the mesh skeleton) in parallel, completely independent of each other*. The iHDG approach is therefore well suited for parallel computing systems. It can be viewed as a fixed-point iterative solver by alternating the computation of the local solver (2.2a) and conservation condition (2.2b). It can be also understood as a block Gauss–Seidel approach for the linear system with volume and weighted trace unknowns. However, unlike matrix-based iterative schemes [75, 134], the proposed iHDG method arises from the structure of HDG methods. As such its convergence does not depend upon the ordering of unknowns as the stabilization (i.e., the weighting matrix $|\mathbf{A}|$) automatically takes care of the direction. For that reason, we call it *iterative HDG discretization* (iHDG). Unlike the original HDG discretization, it promotes fine-grained parallelism in the conservation constraints. What remains is

to show that iHDG converges as the iteration k increases, and this is the focus of Sections 2.4–2.5.

2.4 iHDG methods for hyperbolic PDEs

In this section, we present iHDG methods for the scalar and the system of hyperbolic PDEs. For clarity of the exposition, we consider the transport equation and a linearized shallow water system, and the extension of the proposed approach to other hyperbolic PDEs is straightforward. To begin, let us consider the transport equation

$$\boldsymbol{\beta} \cdot \nabla u^e = f \quad \text{in } \Omega, \tag{2.5a}$$

$$u^e = g \quad \text{on } \partial\Omega^-, \tag{2.5b}$$

where $\partial\Omega^-$ is the inflow part of the boundary $\partial\Omega$, and again u^e denotes the exact solution. *Note that $\boldsymbol{\beta}$ is assumed to be a continuous function across the mesh skeleton.* An upwind HDG discretization [25] for (2.5) consists of the local equation for each element T

$$-\left(u, \nabla \cdot (\boldsymbol{\beta} v)\right)_T + \left\langle \boldsymbol{\beta} \cdot \mathbf{n} u + |\boldsymbol{\beta} \cdot \mathbf{n}| \left(u - \hat{u}\right), v \right\rangle_{\partial T} = \left(f, v\right)_T \quad \forall v \in V_h\left(T\right), \tag{2.6}$$

and conservation conditions on all edges e in the mesh skeleton $\mathcal{E}_h$:

$$\left\langle [\![\boldsymbol{\beta} \cdot \mathbf{n} u + |\boldsymbol{\beta} \cdot \mathbf{n}| \left(u - \hat{u}\right)]\!], \mu \right\rangle_e = 0 \quad \forall \mu \in \Lambda_h\left(e\right). \tag{2.7}$$

Solving (2.7) for $|\boldsymbol{\beta} \cdot \mathbf{n}| \, \hat{u}$ we get

$$|\boldsymbol{\beta} \cdot \mathbf{n}| \, \hat{u} = \{\!\{ \boldsymbol{\beta} \cdot \mathbf{n} u \}\!\} + |\boldsymbol{\beta} \cdot \mathbf{n}| \, \{\!\{ u \}\!\}. \tag{2.8}$$

Applying the iHDG algorithm, Algorithm 1, to the upwind HDG method (2.6)–(2.7) we obtain the approximate solution u^{k+1} at the $(k+1)$th iteration restricted on

each element T via the following independent local solve: for all $v \in V_h(T)$,

$$-\left(u^{k+1}, \nabla\cdot(\boldsymbol{\beta} v)\right)_T + \left\langle \boldsymbol{\beta}\cdot\mathbf{n}u^{k+1} + |\boldsymbol{\beta}\cdot\mathbf{n}|\left(u^{k+1}-\hat{u}^k\right), v\right\rangle_{\partial T} = (f,v)_T, \tag{2.9}$$

where the trace weighted trace $|\boldsymbol{\beta}\cdot\mathbf{n}|\,\hat{u}^k$ is computed using information from the previous iteration as

$$|\boldsymbol{\beta}\cdot\mathbf{n}|\,\hat{u}^k = \left\{\!\!\left\{\boldsymbol{\beta}\cdot\mathbf{n}u^k\right\}\!\!\right\} + |\boldsymbol{\beta}\cdot\mathbf{n}|\left\{\!\!\left\{u^k\right\}\!\!\right\}. \tag{2.10}$$

Next we study the convergence of the iHDG method (2.9)–(2.10). Since (2.5) is linear, it is sufficient to show that iHDG converges for the homogeneous equation with zero forcing f and zero boundary condition g. Let us define ∂T^{out} as the outflow part of ∂T, i.e. $\boldsymbol{\beta}\cdot\mathbf{n} \geq 0$ on ∂T^{out}, and ∂T^{in} as the inflow part of ∂T, i.e. $\boldsymbol{\beta}\cdot\mathbf{n} < 0$ on ∂T^{in}.

Theorem 1. *Assume $-\nabla\cdot\boldsymbol{\beta} \geq \alpha > 0$, i.e., (2.5) is well-posed. The above iHDG iterations for the homogeneous transport equation (2.5) converge exponentially with respect to the number of iterations k. In particular, there exist $J \leq N_T$ such that*

$$\sum_{T\in\Omega_h} \left\|u^k\right\|^2_{\frac{-\nabla\cdot\boldsymbol{\beta}}{2},T} + \left\|u^k\right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|,\partial T^{out}} \leq \frac{c(k)}{2^k}\left\|u^0\right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|,\mathcal{E}_h}, \tag{2.11}$$

where $c(k)$ is a polynomial in k of order at most J and is independent of h and p.

Remark 1. *Note that the factor $\varrho(k) = \frac{k^J}{2^{k/2}}$, i.e., the largest possible term in $c(k)$, is a bounded function, which implies $\frac{c(k)}{2^{\frac{k}{2}}}$ is also bounded by a constant C_J depending only on J. As a consequence,*

$$\frac{c(k)}{2^k} \leq \frac{C_J}{2^{k/2}} \overset{k\to\infty}{\longrightarrow} 0.$$

Proof. Taking $v = u^{k+1}$ in (2.9) and applying the homogeneous forcing condition yields

$$-\left(u^{k+1}, \nabla\cdot\left(\boldsymbol{\beta}u^{k+1}\right)\right)_T + \left\langle \boldsymbol{\beta}\cdot\mathbf{n}u^{k+1} + |\boldsymbol{\beta}\cdot\mathbf{n}|(u^{k+1}-\hat{u}^k), u^{k+1}\right\rangle_{\partial T} = 0. \tag{2.12}$$

Since

$$\left(u^{k+1}, \nabla \cdot \left(\boldsymbol{\beta} u^{k+1}\right)\right)_T = \left(u^{k+1}, \nabla \cdot \boldsymbol{\beta} u^{k+1}\right)_T + \left(u^{k+1}, \boldsymbol{\beta} \cdot \nabla u^{k+1}\right)_T,$$

integrating by parts the second term on the right-hand side, we get

$$\begin{aligned}\left(u^{k+1}, \nabla \cdot \left(\boldsymbol{\beta} u^{k+1}\right)\right)_T &= \left(u^{k+1}, \nabla \cdot \boldsymbol{\beta} u^{k+1}\right)_T - \left(u^{k+1}, \nabla \cdot \left(\boldsymbol{\beta} u^{k+1}\right)\right)_T \\ &\quad + \left\langle \boldsymbol{\beta} \cdot \mathbf{n} u^{k+1}, u^{k+1} \right\rangle_{\partial T}.\end{aligned}$$

Rearranging the terms, we obtain

$$\left(u^{k+1}, \nabla \cdot \left(\boldsymbol{\beta} u^{k+1}\right)\right)_T = \left(u^{k+1}, \frac{\nabla \cdot \boldsymbol{\beta}}{2} u^{k+1}\right)_T + \frac{1}{2} \left\langle \boldsymbol{\beta} \cdot \mathbf{n} u^{k+1}, u^{k+1} \right\rangle_{\partial T}. \tag{2.13}$$

Using (2.13) we can rewrite (2.12) as

$$\left\|u^{k+1}\right\|^2_{\frac{-\nabla \cdot \boldsymbol{\beta}}{2}, T} + \left\langle \left(|\boldsymbol{\beta} \cdot \mathbf{n}| + \frac{1}{2} \boldsymbol{\beta} \cdot \mathbf{n} \right) u^{k+1}, u^{k+1} \right\rangle_{\partial T} = \left\langle |\boldsymbol{\beta} \cdot \mathbf{n}| \hat{u}^k, u^{k+1} \right\rangle_{\partial T}. \tag{2.14}$$

On the other hand, (2.10) is equivalent to

$$|\boldsymbol{\beta} \cdot \mathbf{n}| \hat{u}^k = \begin{cases} |\boldsymbol{\beta} \cdot \mathbf{n}| u^k & \text{on } \partial T^{\text{out}}, \\ |\boldsymbol{\beta} \cdot \mathbf{n}| u^k_{\text{ext}}, & \text{on } \partial T^{\text{in}}, \end{cases} \tag{2.15}$$

where u^k_{ext} is either the physical boundary condition or the solution of the neighboring element that shares the same inflow boundary ∂T^{in}.

Rewriting (2.14) in terms of ∂T^{in} and ∂T^{out}, we obtain

$$\begin{aligned}&\left\|u^{k+1}\right\|^2_{\frac{-\nabla \cdot \boldsymbol{\beta}}{2}, T} + \frac{3}{2} \left\langle |\boldsymbol{\beta} \cdot \mathbf{n}| u^{k+1}, u^{k+1} \right\rangle_{\partial T^{\text{out}}} + \frac{1}{2} \left\langle |\boldsymbol{\beta} \cdot \mathbf{n}| u^{k+1}, u^{k+1} \right\rangle_{\partial T^{\text{in}}} \\ = \;& \left\langle |\boldsymbol{\beta} \cdot \mathbf{n}| u^k, u^{k+1} \right\rangle_{\partial T^{\text{out}}} + \left\langle |\boldsymbol{\beta} \cdot \mathbf{n}| u^k_{\text{ext}}, u^{k+1} \right\rangle_{\partial T^{\text{in}}}.\end{aligned}$$

By the Cauchy-Schwarz inequality we have

$$\begin{aligned}&\left\|u^{k+1}\right\|^2_{\frac{-\nabla \cdot \boldsymbol{\beta}}{2}, T} + \frac{3}{2} \left\langle |\boldsymbol{\beta} \cdot \mathbf{n}| u^{k+1}, u^{k+1} \right\rangle_{\partial T^{\text{out}}} + \frac{1}{2} \left\langle |\boldsymbol{\beta} \cdot \mathbf{n}| u^{k+1}, u^{k+1} \right\rangle_{\partial T^{\text{in}}} \\ \leq \;& \frac{1}{2} \left\langle |\boldsymbol{\beta} \cdot \mathbf{n}| u^k, u^k \right\rangle_{\partial T^{\text{out}}} + \frac{1}{2} \left\langle |\boldsymbol{\beta} \cdot \mathbf{n}| u^{k+1}, u^{k+1} \right\rangle_{\partial T^{\text{out}}} \\ &+ \frac{1}{2} \left\langle |\boldsymbol{\beta} \cdot \mathbf{n}| u^k_{\text{ext}}, u^k_{\text{ext}} \right\rangle_{\partial T^{\text{in}}} + \frac{1}{2} \left\langle |\boldsymbol{\beta} \cdot \mathbf{n}| u^{k+1}, u^{k+1} \right\rangle_{\partial T^{\text{in}}},\end{aligned}$$

which implies

$$\left\|u^{k+1}\right\|^2_{\frac{-\nabla\cdot\boldsymbol{\beta}}{2},T} + \left\|u^{k+1}\right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|,\partial T^{\text{out}}} \leq \frac{1}{2}\left\{\left\|u^k\right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|,\partial T^{\text{out}}} + \left\|u^k_{\text{ext}}\right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|,\partial T^{\text{in}}}\right\}. \quad (2.16)$$

Consider the set $\mathfrak{T}^1$ of all elements T such that ∂T^{in} is a subset of the physical inflow boundary $\partial\Omega^{\text{in}}$ on which we have $u^k_{\text{ext}} = 0$ for all $k \in \mathbb{N}$. We obtain from (2.16) that

$$\left\|u^{k+1}\right\|^2_{\frac{-\nabla\cdot\boldsymbol{\beta}}{2},T} + \left\|u^{k+1}\right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|,\partial T^{\text{out}}} \leq \frac{1}{2}\left\|u^k\right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|,\partial T^{\text{out}}}, \quad (2.17)$$

which implies

$$\left\|u^{k+1}\right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|,\partial T^{\text{out}}} \leq \frac{1}{2}\left\|u^k\right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|,\partial T^{\text{out}}} \leq \cdots \leq \frac{1}{2^{k+1}}\left\|u^0\right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|,\partial T^{\text{out}}}. \quad (2.18)$$

From (2.17) and (2.18) we also have

$$\left\|u^{k+1}\right\|^2_{\frac{-\nabla\cdot\boldsymbol{\beta}}{2},T} \leq \frac{1}{2^{k+1}}\left\|u^0\right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|,\partial T^{\text{out}}}. \quad (2.19)$$

Next, let us define $\Omega^1_h := \Omega_h$ and

$$\Omega^2_h := \Omega^1_h \backslash \mathfrak{T}^1.$$

Consider the set $\mathfrak{T}^2$ of all T in Ω^2_h such that ∂T^{in} is either (possibly partially) a subset of the physical inflow boundary $\partial\Omega^{\text{in}}$ or (possibly partially) a subset of the outflow boundary of elements in $\mathfrak{T}^1$. This implies, on $\partial T^{\text{in}} \in \mathfrak{T}^2$, u^k_{ext} either is zero for all $k \in \mathbb{N} \setminus \{1\}$ or satisfies the bound

$$\left\|u^k_{\text{ext}}\right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|,\partial T^{\text{in}}} \leq \frac{1}{2^k}\left\|u^0_{\text{ext}}\right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|,\partial T^{\text{in}}}. \quad (2.20)$$

Combining (2.16) and (2.20), we obtain

$$\left\|u^{k+1}\right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|,\partial T^{\text{out}}} \leq \frac{1}{2^{k+1}}\left\{\left\|u^0\right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|,\partial T^{\text{out}}} + (k+1)\left\|u^0_{\text{ext}}\right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|,\partial T^{\text{in}}}\right\}, \quad (2.21)$$

which, together with (2.16), leads to

$$\left\|u^{k+1}\right\|^2_{\frac{-\nabla\cdot\boldsymbol{\beta}}{2},T} \leq \frac{1}{2^{k+1}}\left\{\left\|u^0\right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|,\partial T^{\text{out}}} + (k+1)\left\|u^0_{\text{ext}}\right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|,\partial T^{\text{in}}}\right\}. \tag{2.22}$$

Now defining Ω_h^i and $\mathcal{T}^i$ recursively and repeating the above arguments concludes the proof. □

We can see that the contraction constant in this case is 1/2; in our numerical experiments we found the spectral radius of the iteration matrix to be exactly 1/2, which confirms the theoretical result. We are in a position to discuss the convergence of the kth iterative solution to the exact solution u^e. For a sufficiently smooth exact solution, e.g., $u^e|_T \in H^s(T), s > 3/2$, we assume the following standard convergence result of DG (HDG) methods for the transport equation: let $\sigma = \min\{p+1, s\}$; we have

$$\|u - u^e\|^2_{\Omega_h} \leq C\frac{h^{2\sigma-1}}{p^{2s-1}}\|u^e\|^2_{H^s(\Omega_h)}, \tag{2.23}$$

and we refer the readers to, for example, [83, 25] for a proof.

Corollary 1. *Suppose the exact solution u^e is sufficiently smooth, i.e., $u^e|_T \in H^s(T), s > 3/2$; then there exists a constant C independent of k, h, and p such that*

$$\left\|u^k - u^e\right\|^2_{\Omega_h} \leq C\left(\frac{c(k)}{2^k}\left\|u^0\right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|,\mathcal{E}_h} + \frac{h^{2\sigma-1}}{p^{2s-1}}\|u^e\|^2_{H^s(\Omega_h)}\right),$$

where $c(k)$ is a polynomial in k of order at most N_T and is independent of h and p.

Proof. The result is a direct consequence of the result from Theorem 1, the HDG (DG) convergence result (2.23), and the triangle inequality. □

Remark 2. *For the time-dependent transport equation, we discretize the spatial operator using HDG and time using the backward Euler method (or the Crank–Nicolson*

method or a high-order method if desired). The iHDG approach in this case is almost identical to the one for the steady state equation except that we now have an additional L^2-term $\left(u^{k+1}, v\right)_T/\Delta t$ in the local equation (2.9)*. This improves the convergence of iHDG. Indeed, the convergence analysis is almost identical except we now have* $\left\|u^{k+1}\right\|^2_{-\nabla\cdot\boldsymbol{\beta}/2+1/\Delta t,T}$ *instead of* $\left\|u^{k+1}\right\|^2_{-\nabla\cdot\boldsymbol{\beta}/2,T}$ *in* (2.16)*.*

Now let us consider the flux given in Nguyen, Peraire, and Cockburn (NPC flux) [112] and analyze the convergence of the iHDG scheme. The stabilization τ of the NPC flux is given by

$$\tau = |\boldsymbol{\beta}\cdot\mathbf{n}|\frac{1+sgn(\boldsymbol{\beta}\cdot\mathbf{n})}{2} - \boldsymbol{\beta}\cdot\mathbf{n}, \tag{2.24}$$

and the trace $\hat{u}^k$ at the kth iteration is computed as

$$\hat{u}^k = \frac{\{\!\{\tau u^k\}\!\} + \{\!\{\boldsymbol{\beta}\cdot\mathbf{n}u^k\}\!\}}{\{\!\{\tau\}\!\}}.$$

In this case we apply the iHDG algorithm, Algorithm 1, without the weighting $|\mathbf{A}|$ in front of the trace $\hat{u}^k$ as the stabilization comes from τ.

Theorem 2. *Assume* $-\nabla\cdot\boldsymbol{\beta} \geq \alpha > 0$*, i.e.,* (2.5) *is well-posed. There exists* $J \leq N_T$ *such that the iHDG algorithm with the NPC flux for the homogeneous transport equation converges in* J *iterations.*[4]

This theorem shows that for the scalar hyperbolic equation (2.5), iHDG with the NPC flux converges in a finite number of iterations, which is faster than the upwind HDG flux. The reason is that the NPC flux mimics the matching of wave propagation from the inflow to the outflow. However, designing such a scheme for a system of hyperbolic equations, such as the linearized shallow water system, does not

[4]The proof is similar to the proof of Theorem 1 and hence is omitted here.

seem to be tractable due to the interaction of more than one wave. In this sense the upwind HDG flux is more robust, since it is applicable for other systems of hyperbolic PDEs as well, as we now show.

We next consider the following system of linear hyperbolic PDEs arising from the oceanic linearized shallow water system [70]:

$$\frac{\partial}{\partial t}\begin{pmatrix}\phi^e\\ \Phi u^e\\ \Phi v^e\end{pmatrix}+\frac{\partial}{\partial x}\begin{pmatrix}\Phi u^e\\ \Phi\phi^e\\ 0\end{pmatrix}+\frac{\partial}{\partial y}\begin{pmatrix}\Phi v^e\\ 0\\ \Phi\phi^e\end{pmatrix}=\begin{pmatrix}0\\ f\Phi v^e-\gamma\Phi u^e+\frac{\tau_x}{\rho}\\ -f\Phi u^e-\gamma\Phi v^e+\frac{\tau_y}{\rho}\end{pmatrix}, \quad (2.25)$$

where $\phi = gH$ is the geopotential height with g and H being the gravitational constant and the perturbation of the free surface height, $\Phi > 0$ is a constant mean flow geopotential height, $\boldsymbol{\vartheta} := (u, v)$ is the perturbed velocity, $\gamma \geq 0$ is the bottom friction, $\boldsymbol{\tau} := (\tau_x, \tau_y)$ is the wind stress, and ρ is the density of the water. Here, $f = f_0 + \beta\,(y - y_m)$ is the Coriolis parameter, where f_0, β, and y_m are given constants.

Again, for simplicity of the exposition and the analysis, let us employ the backward Euler discretization for temporal derivatives and HDG [27] for spatial ones. Since the unknowns of interest are those at the $(m+1)$th time step, we can suppress the time index for clarity of the exposition. Furthermore, since the system (2.25) is linear, an argument similar to that above shows that it is sufficient to consider homogeneous system with zero initial condition, boundary condition, and forcing. Also here we consider the case of $\boldsymbol{\tau} = 0$. Applying the iHDG algorithm, Algorithm

1, to the homogeneous system gives

$$\left(\frac{\phi^{k+1}}{\Delta t},\varphi_1\right)_T-\left(\Phi\boldsymbol{\vartheta}^{k+1},\nabla\varphi_1\right)_T+\left\langle\Phi\boldsymbol{\vartheta}^{k+1}\cdot\mathbf{n}+\sqrt{\Phi}\left(\phi^{k+1}-\hat{\phi}^k\right),\varphi_1\right\rangle_{\partial T}=0, \tag{2.26a}$$

$$\left(\frac{\Phi u^{k+1}}{\Delta t},\varphi_2\right)_T-\left(\Phi\phi^{k+1},\frac{\partial\varphi_2}{\partial x}\right)_T+\left\langle\Phi\hat{\phi}^k\mathbf{n}_1,\varphi_2\right\rangle_{\partial T}=\left(f\Phi v^{k+1}-\gamma\Phi u^{k+1},\varphi_2\right)_T, \tag{2.26b}$$

$$\left(\frac{\Phi v^{k+1}}{\Delta t},\varphi_3\right)_T-\left(\Phi\phi^{k+1},\frac{\partial\varphi_3}{\partial y}\right)_T+\left\langle\Phi\hat{\phi}^k\mathbf{n}_2,\varphi_3\right\rangle_{\partial T}=\left(-f\Phi u^{k+1}-\gamma\Phi v^{k+1},\varphi_3\right)_T, \tag{2.26c}$$

where φ_1,φ_2, and φ_3 are the test functions, and

$$\hat{\phi}^k=\left\{\!\!\left\{\phi^k\right\}\!\!\right\}+\sqrt{\Phi}\left\{\!\!\left\{\boldsymbol{\vartheta}^k\cdot\mathbf{n}\right\}\!\!\right\}.$$

Our goal is to show that $\left(\phi^{k+1},\Phi\boldsymbol{\vartheta}^{k+1}\right)$ converges to zero. To that end, let us define

$$\mathcal{C}:=\frac{\mathcal{A}}{\mathcal{B}},\quad \mathcal{A}:=\max\left\{\frac{\Phi+\sqrt{\Phi}}{2},\frac{1+\sqrt{\Phi}}{2}\right\}, \tag{2.27}$$

and

$$\mathcal{B}:=\min\left\{\left(\frac{ch}{\Delta t(p+1)(p+2)}+\frac{\sqrt{\Phi}-\Phi}{2}\right),\left(\left(\gamma+\frac{1}{\Delta t}\right)\frac{ch}{(p+1)(p+2)}+\frac{(-1-\sqrt{\Phi})}{2}\right)\right\},$$

where $0<c\leq 1$ is a constant. We also need the following norms:

$$\left\|\left(\phi^k,\boldsymbol{\vartheta}^k\right)\right\|_{\Omega_h}^2:=\left\|\phi^k\right\|_{\Omega_h}^2+\left\|\boldsymbol{\vartheta}^k\right\|_{\Phi,\Omega_h}^2,\quad \left\|\left(\phi^k,\boldsymbol{\vartheta}^k\right)\right\|_{\mathcal{E}_h}^2:=\left\|\phi^k\right\|_{\mathcal{E}_h}^2+\left\|\boldsymbol{\vartheta}^k\right\|_{\Phi,\mathcal{E}_h}^2.$$

Theorem 3. *Assume that the mesh size h, the time step Δt and the solution order p are chosen such that $\mathcal{B}>0$ and $\mathcal{C}<1$; then the approximate solution at the kth iteration $\left(\phi^k,\boldsymbol{\vartheta}^k\right)$ converges to zero; i.e.,*

$$\left\|\left(\phi^k,\boldsymbol{\vartheta}^k\right)\right\|_{\mathcal{E}_h}^2\leq\mathcal{C}^k\left\|\left(\phi^0,\boldsymbol{\vartheta}^0\right)\right\|_{\mathcal{E}_h}^2,\quad \left\|\left(\phi^k,\boldsymbol{\vartheta}^k\right)\right\|_{\Omega_h}^2\leq\Delta t\mathcal{A}\left(\mathcal{C}+1\right)\mathcal{C}^{k-1}\left\|\left(\phi^0,\boldsymbol{\vartheta}^0\right)\right\|_{\mathcal{E}_h}^2,$$

where $\mathcal{C}$ is defined in (2.27).

Proof. Choosing the test functions $\varphi_1 = \phi^{k+1}$, $\varphi_2 = u^{k+1}$, and $\varphi_3 = v^{k+1}$ in (2.26), integrating the second term in (2.26a) by parts, and then summing equations in (2.26), we obtain

$$\begin{aligned}\frac{1}{\Delta t}\left(\phi^{k+1},\phi^{k+1}\right)_T + \frac{\Phi}{\Delta t}\left(\boldsymbol{\vartheta}^{k+1},\boldsymbol{\vartheta}^{k+1}\right)_T + \sqrt{\Phi}\left\langle \phi^{k+1},\phi^{k+1}\right\rangle_{\partial T} + \gamma\Phi\left(\boldsymbol{\vartheta}^{k+1},\boldsymbol{\vartheta}^{k+1}\right)_T \\ = \sqrt{\Phi}\left\langle \hat{\phi}^k,\phi^{k+1}\right\rangle_{\partial T} - \Phi\left\langle \hat{\phi}^k,\mathbf{n}\cdot\boldsymbol{\vartheta}^{k+1}\right\rangle_{\partial T}. \end{aligned} \tag{2.28}$$

Summing (2.28) over all elements yields

$$\begin{aligned}&\sum_T \frac{1}{\Delta t}\left(\phi^{k+1},\phi^{k+1}\right)_T + \frac{\Phi}{\Delta t}\left(\boldsymbol{\vartheta}^{k+1},\boldsymbol{\vartheta}^{k+1}\right)_T + \sqrt{\Phi}\left\langle \phi^{k+1},\phi^{k+1}\right\rangle_{\partial T} + \gamma\Phi\left(\boldsymbol{\vartheta}^{k+1},\boldsymbol{\vartheta}^{k+1}\right)_T \\ &= \sum_{\partial T}\sqrt{\Phi}\left\langle \hat{\phi}^k,\phi^{k+1}\right\rangle_{\partial T} - \Phi\left\langle \hat{\phi}^k,\mathbf{n}\cdot\boldsymbol{\vartheta}^{k+1}\right\rangle_{\partial T} \\ &= \sum_{e\in\mathcal{E}_h}\left\langle 2\sqrt{\Phi}\left(\{\!\{\phi^k\}\!\} + \sqrt{\Phi}\,\{\!\{\boldsymbol{\vartheta}^k\cdot\mathbf{n}\}\!\}\right), \left(\{\!\{\phi^{k+1}\}\!\} - \sqrt{\Phi}\,\{\!\{\boldsymbol{\vartheta}^{k+1}\cdot\mathbf{n}\}\!\}\right)\right\rangle_e;\end{aligned}$$

by the Cauchy-Schwarz inequality, we could bound the right-hand side as

$$\begin{aligned}&\leq \sum_{e\in\mathcal{E}_h}\sqrt{\Phi}\left(\left\|\{\!\{\phi^k\}\!\}\right\|_e^2 + \left\|\{\!\{\phi^{k+1}\}\!\}\right\|_e^2\right) + \Phi\left(\left\|\{\!\{\phi^k\}\!\}\right\|_e^2 + \left\|\{\!\{\boldsymbol{\vartheta}^{k+1}\cdot\mathbf{n}\}\!\}\right\|_e^2\right) \\ &+ \Phi\left(\left\|\{\!\{\phi^{k+1}\}\!\}\right\|_e^2 + \left\|\{\!\{\boldsymbol{\vartheta}^{k}\cdot\mathbf{n}\}\!\}\right\|_e^2\right) + \Phi\sqrt{\Phi}\left(\left\|\{\!\{\boldsymbol{\vartheta}^{k}\cdot\mathbf{n}\}\!\}\right\|_e^2 + \left\|\{\!\{\boldsymbol{\vartheta}^{k+1}\cdot\mathbf{n}\}\!\}\right\|_e^2\right),\end{aligned}$$

with a little algebraic manipulation we have

$$\begin{aligned}&\leq \sum_{\partial T}\left[\frac{\Phi+\sqrt{\Phi}}{2}\left\langle \phi^k,\phi^k\right\rangle_{\partial T} + \frac{\Phi(1+\sqrt{\Phi})}{2}\left\langle \boldsymbol{\vartheta}^k,\boldsymbol{\vartheta}^k\right\rangle_{\partial T}\right] \\ &+ \sum_{\partial T}\left[\frac{\Phi+\sqrt{\Phi}}{2}\left\langle \phi^{k+1},\phi^{k+1}\right\rangle_{\partial T} + \frac{\Phi(1+\sqrt{\Phi})}{2}\left\langle \boldsymbol{\vartheta}^{k+1},\boldsymbol{\vartheta}^{k+1}\right\rangle_{\partial T}\right].\end{aligned} \tag{2.29}$$

An application of the inverse trace inequality [31] for tensor product elements gives

$$\left(\phi^{k+1},\phi^{k+1}\right)_T \geq \frac{2ch}{d(p+1)(p+2)}\left\langle \phi^{k+1},\phi^{k+1}\right\rangle_{\partial T}, \tag{2.30a}$$

$$\left(\boldsymbol{\vartheta}^{k+1},\boldsymbol{\vartheta}^{k+1}\right)_T \geq \frac{2ch}{d(p+1)(p+2)}\left\langle \boldsymbol{\vartheta}^{k+1},\boldsymbol{\vartheta}^{k+1}\right\rangle_{\partial T}, \tag{2.30b}$$

where d is the spatial dimension, which in this case is 2, and $0 < c \leq 1$ is a constant. For simplices we can use the trace inequalities in [152] and it will change only the constants in the proof. Inequality (2.30), together with (2.29), implies

$$\begin{aligned}
&\sum_{\partial T} \left[\left(\frac{ch}{\Delta t(p+1)(p+2)} + \frac{\sqrt{\Phi}-\Phi}{2} \right) \left\langle \phi^{k+1}, \phi^{k+1} \right\rangle_{\partial T} \right. \\
&\left. + \left(\left(\gamma + \frac{1}{\Delta t} \right) \frac{ch}{(p+1)(p+2)} + \frac{(-1-\sqrt{\Phi})}{2} \right) \left\langle \Phi \boldsymbol{\vartheta}^{k+1}, \boldsymbol{\vartheta}^{k+1} \right\rangle_{\partial T} \right] \\
\leq \ &\sum_{\partial T} \left[\frac{\Phi+\sqrt{\Phi}}{2} \left\langle \phi^{k}, \phi^{k} \right\rangle_{\partial T} + \frac{(1+\sqrt{\Phi})}{2} \left\langle \Phi \boldsymbol{\vartheta}^{k}, \boldsymbol{\vartheta}^{k} \right\rangle_{\partial T} \right], \qquad (2.31)
\end{aligned}$$

which implies

$$\left\| \left(\phi^{k+1}, \boldsymbol{\vartheta}^{k+1} \right) \right\|_{\mathcal{E}_h}^2 \leq \mathcal{C} \left\| \left(\phi^{k}, \boldsymbol{\vartheta}^{k} \right) \right\|_{\mathcal{E}_h}^2,$$

where the constant $\mathcal{C}$ is computed as in (2.27). Therefore,

$$\left\| \left(\phi^{k+1}, \boldsymbol{\vartheta}^{k+1} \right) \right\|_{\mathcal{E}_h}^2 \leq \mathcal{C}^{k+1} \left\| \left(\phi^{0}, \boldsymbol{\vartheta}^{0} \right) \right\|_{\mathcal{E}_h}^2. \qquad (2.32)$$

On the other hand, inequalities (2.29) and (2.32) imply

$$\left\| \left(\phi^{k+1}, \boldsymbol{\vartheta}^{k+1} \right) \right\|_{\Omega_h}^2 \leq \Delta t \mathcal{A} \left(\mathcal{C} + 1 \right) \mathcal{C}^{k} \left\| \left(\phi^{0}, \boldsymbol{\vartheta}^{0} \right) \right\|_{\mathcal{E}_h}^2,$$

and this ends the proof. □

Remark 3. *The above theorem implies that, in order to have a convergent algorithm, we need to have the following relation between Δt, p, and h :*

$$\Delta t = \mathcal{O}\left(\frac{h}{\Phi(p+1)(p+2)} \right).$$

Unlike the convergent result in Theorem 1 for the scalar hyperbolic equation, the finding in Theorem 3 shows that iHDG is conditionally convergent for a system of hyperbolic equations. More specifically, the convergence rate depends on the mesh size h, the time step Δt, and the solution order p. This will be confirmed by numerical results in Section 2.6.

To show the convergence of the kth iterative solution to the exact solution $(\phi^e, \Phi\boldsymbol{\vartheta}^e)$, we assume that the exact solution is smooth, i.e., $(\phi^e, \Phi\boldsymbol{\vartheta}^e)|_T \in [H^s(T)]^3, s > 3/2$. If we define

$$\mathcal{E}^e(t) := \sum_T \|\phi^e(t)\|^2_{H^s(T)} + \|\boldsymbol{\vartheta}^e(t)\|^2_{\Phi,H^s(T)},$$

results from [27] show that, for $\gamma > 0$ and $\sigma = \min\{p+1, s\}$, we have

$$\|(\phi - \phi^e, \boldsymbol{\vartheta} - \boldsymbol{\vartheta}^e)\|^2_{\Omega_h} \le C\Delta t \frac{h^{2\sigma-1}}{p^{2s-1}} \mathcal{E}^e(m\Delta t) \tag{2.33}$$

at the mth time step.

Corollary 2. *Suppose the exact solution satisfies $(\phi^e, \Phi\boldsymbol{\vartheta}^e)|_T \in [H^s(T)]^3, s > 3/2$; then there exists a constant C independent of k, h, and p such that*

$$\left\|\left(\phi^k - \phi^e, \boldsymbol{\vartheta}^k - \boldsymbol{\vartheta}^e\right)\right\|^2_{\Omega_h} \le C\Delta t \left(\mathcal{A}(\mathcal{C}+1)\mathcal{C}^{k-1}\left\|\left(\phi^0, \boldsymbol{\vartheta}^0\right)\right\|^2_{\mathcal{E}_h} + \frac{h^{2\sigma-1}}{p^{2s-1}}\mathcal{E}^e(m\Delta t)\right),$$

with $\sigma = \min\{p+1, s\}$.

2.5 iHDG methods for convection-diffusion PDEs

2.5.1 First order form

In this section we apply the iHDG algorithm, Algorithm 1, to the following prototypical convection-diffusion equation in first order form:

$$\kappa^{-1}\boldsymbol{\sigma}^e + \nabla u^e = 0 \quad \text{in } \Omega, \tag{2.34a}$$

$$\nabla \cdot \boldsymbol{\sigma}^e + \boldsymbol{\beta} \cdot \nabla u^e + \nu u^e = f \quad \text{in } \Omega. \tag{2.34b}$$

We suppose that (2.34) is well-posed, i.e.,

$$\nu - \frac{\nabla \cdot \boldsymbol{\beta}}{2} \ge \lambda > 0. \tag{2.35}$$

Moreover, we restrict ourselves to a constant diffusion coefficient κ. An upwind HDG numerical flux [25] is given by

$$\hat{\mathbf{F}} \cdot \mathbf{n} = \begin{bmatrix} \hat{u}\mathbf{n}_1 \\ \hat{u}\mathbf{n}_2 \\ \hat{u}\mathbf{n}_3 \\ \boldsymbol{\sigma} \cdot \mathbf{n} + \boldsymbol{\beta} \cdot \mathbf{n} u + \tau \left(u - \hat{u} \right) \end{bmatrix}. \tag{2.36}$$

Strongly enforcing the conservation condition yields

$$\hat{u} = \frac{1}{\tau^+ + \tau^-} \left([\![\boldsymbol{\sigma} \cdot \mathbf{n}]\!] + [\![\boldsymbol{\beta} \cdot \mathbf{n} u]\!] + [\![\tau u]\!] \right),$$

with τ being chosen as

$$\tau^{\pm} = \frac{\gamma}{2} \left(\alpha - \boldsymbol{\beta} \cdot \mathbf{n}^{\pm} \right), \tag{2.37}$$

where $\gamma = 1$ and $\alpha = \sqrt{|\boldsymbol{\beta} \cdot \mathbf{n}|^2 + 4}$ for the upwind flux in [25]. We see that $\tau^{\pm}$ in general is a function which depends upon $\boldsymbol{\beta}$ and is always positive. Similar to the previous sections, it is sufficient to consider the homogeneous problem. Applying the iHDG algorithm, Algorithm 1 with $\boldsymbol{\tau}, v$ as test functions, we have the following iterative scheme:

$$\kappa^{-1} \left(\boldsymbol{\sigma}^{k+1}, \boldsymbol{\tau} \right)_T - \left(u^{k+1}, \nabla \cdot \boldsymbol{\tau} \right)_T + \left\langle \hat{u}^k, \boldsymbol{\tau} \cdot \mathbf{n} \right\rangle_{\partial T} = 0 \tag{2.38a}$$

$$\begin{aligned} &- \left(\boldsymbol{\sigma}^{k+1}, \nabla v \right)_T - \left(u^{k+1}, \nabla \cdot (\boldsymbol{\beta} v) - \nu v \right)_T \\ &+ \left\langle \boldsymbol{\sigma}^{k+1} \cdot \mathbf{n} + \boldsymbol{\beta} \cdot \mathbf{n} u^{k+1} + \tau (u^{k+1} - \hat{u}^k), v \right\rangle_{\partial T} = 0, \end{aligned} \tag{2.38b}$$

where

$$\hat{u}^k = \frac{[\![\boldsymbol{\sigma}^k \cdot \mathbf{n}]\!] + [\![\boldsymbol{\beta} \cdot \mathbf{n} u^k]\!] + [\![\tau u^k]\!]}{\sqrt{|\boldsymbol{\beta} \cdot \mathbf{n}|^2 + 4}}.$$

For ε, $h > 0$ and $0 < c \leq 1$ given, define

$$\mathcal{C}_1 := \frac{3(\|\boldsymbol{\beta}\cdot\mathbf{n}\|^2_{L^\infty(\partial T)} + \bar{\tau}^2)(\bar{\tau}\varepsilon + 1)}{2\varepsilon},\ \mathcal{C}_2 := \frac{3(\bar{\tau}\varepsilon+1)}{2\varepsilon}, \tag{2.39}$$

$$\mathcal{C}_3 := \frac{\bar{\tau} + \|\boldsymbol{\beta}\cdot\mathbf{n}\|_{L^\infty(\partial T)}}{2},\ \mathcal{C}_4 := \frac{\varepsilon}{2}, \tag{2.40}$$

$$\mathcal{D} := \frac{\mathcal{A}}{\mathcal{B}},\ \mathcal{A} = \max\{\mathcal{C}_1, \mathcal{C}_2\},\ \mathcal{E} := \frac{\max\{\mathcal{C}_3, \mathcal{C}_4\}}{\min\{\kappa^{-1}, \lambda\}},\ \mathcal{F} := \frac{\mathcal{A}}{\min\{\kappa^{-1}, \lambda\}}, \tag{2.41}$$

$$\mathcal{B} := \min\{\frac{2ch\kappa^{-1}}{d(p+1)(p+2)} - \mathcal{C}_4, \frac{2ch\lambda}{d(p+1)(p+2)} + \tau_* - \mathcal{C}_3\}. \tag{2.42}$$

As in the previous section, we need the following norms:

$$\left\|\left(\boldsymbol{\sigma}^k, u^k\right)\right\|^2_{\Omega_h} := \left\|\boldsymbol{\sigma}^k\right\|^2_{\Omega_h} + \left\|u^k\right\|^2_{\Omega_h}, \quad \left\|\left(\boldsymbol{\sigma}^k, u^k\right)\right\|^2_{\mathcal{E}_h} := \left\|\boldsymbol{\sigma}^k\right\|^2_{\mathcal{E}_h} + \left\|u^k\right\|^2_{\mathcal{E}_h}.$$

Theorem 4. *Suppose that the mesh size h and the solution order p are chosen such that $\mathcal{B} > 0$ and $\mathcal{D} < 1$; the algorithm* (2.38a)-(2.38b) *converges in the following sense:*

$$\left\|\left(\boldsymbol{\sigma}^k, u^k\right)\right\|^2_{\mathcal{E}_h} \leq \mathcal{D}^k \left\|\left(\boldsymbol{\sigma}^0, u^0\right)\right\|^2_{\mathcal{E}_h}, \quad \left\|\left(\boldsymbol{\sigma}^k, u^k\right)\right\|^2_{\Omega_h} \leq (\mathcal{E}\mathcal{D} + \mathcal{F})\mathcal{D}^{k-1} \left\|\left(\boldsymbol{\sigma}^0, u^0\right)\right\|^2_{\mathcal{E}_h},$$

where $\mathcal{D}, \mathcal{E}$, and $\mathcal{F}$ are as defined in (2.41).

Proof. Choosing $\boldsymbol{\sigma}^{k+1}$ and u^{k+1} as test functions in (2.38a)-(2.38b), integrating the second term in (2.38a) by parts, using (2.13) for second term in (2.38b), and then summing up the resulting two equations, we get

$$\begin{aligned}&\kappa^{-1}\left(\boldsymbol{\sigma}^{k+1}, \boldsymbol{\sigma}^{k+1}\right)_T + \left(\frac{-\nabla\cdot\boldsymbol{\beta}}{2}u^{k+1}, u^{k+1}\right)_T + \nu\left(u^{k+1}, u^{k+1}\right)_T - \frac{1}{2}\left\langle\boldsymbol{\beta}\cdot\mathbf{n}u^{k+1}, u^{k+1}\right\rangle_{\partial T}\\ &+ \left\langle\hat{u}^k, \boldsymbol{\sigma}^{k+1}\cdot\mathbf{n}\right\rangle_{\partial T} + \left\langle\boldsymbol{\beta}\cdot\mathbf{n}u^{k+1} + \tau(u^{k+1} - \hat{u}^k), u^{k+1}\right\rangle_{\partial T} = 0.\end{aligned} \tag{2.43}$$

Due to the condition (2.35),

$$\begin{aligned}&\kappa^{-1}\left(\boldsymbol{\sigma}^{k+1}, \boldsymbol{\sigma}^{k+1}\right)_T + \lambda\left(u^{k+1}, u^{k+1}\right)_T + \left\langle\tau u^{k+1}, u^{k+1}\right\rangle_{\partial T}\\ &\leq \left\langle\tau\hat{u}^k, u^{k+1}\right\rangle_{\partial T} - \frac{1}{2}\left\langle\boldsymbol{\beta}\cdot\mathbf{n}u^{k+1}, u^{k+1}\right\rangle_{\partial T} - \left\langle\hat{u}^k, \boldsymbol{\sigma}^{k+1}\cdot\mathbf{n}\right\rangle_{\partial T}.\end{aligned} \tag{2.44}$$

By the Cauchy-Schwarz and Young inequalities and the fact that $|\boldsymbol{\beta}\cdot\mathbf{n}| \leq \|\boldsymbol{\beta}\cdot\mathbf{n}\|_{L^\infty(\partial\Omega_h)}$ and letting $\bar{\tau} := \|\tau\|_{L^\infty(\partial\Omega_h)}$, $\tau_* := \inf_{\partial T\in\partial\Omega_h}\tau$,

$$\begin{aligned}
&\kappa^{-1}\|\boldsymbol{\sigma}^{k+1}\|^2_{L^2(T)} + \lambda\|u^{k+1}\|^2_{L^2(T)} + \tau_*\|u^{k+1}\|^2_{L^2(\partial T)}\\
\leq\ & \frac{\bar{\tau}\varepsilon+1}{2\varepsilon}\|\hat{u}^k\|^2_{L^2(\partial T)} + \frac{\bar{\tau}+\|\boldsymbol{\beta}\cdot\mathbf{n}\|_{L^\infty(\partial T)}}{2}\|u^{k+1}\|^2_{L^2(\partial T)} + \frac{\varepsilon}{2}\|\boldsymbol{\sigma}^{k+1}\|^2_{L^2(\partial T)}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\sum_T\left[\kappa^{-1}\|\boldsymbol{\sigma}^{k+1}\|^2_{L^2(T)} + \lambda\|u^{k+1}\|^2_{L^2(T)} + \tau_*\|u^{k+1}\|^2_{L^2(\partial T)}\right] \leq \sum_{\partial T}\frac{\bar{\tau}\varepsilon+1}{2\varepsilon}\|\hat{u}^k\|^2_{L^2(\partial T)}\\
&+\frac{\bar{\tau}+\|\boldsymbol{\beta}\cdot\mathbf{n}\|_{L^\infty(\partial T)}}{2}\|u^{k+1}\|^2_{L^2(\partial T)} + \frac{\varepsilon}{2}\|\boldsymbol{\sigma}^{k+1}\|^2_{L^2(\partial T)}. \qquad (2.45)
\end{aligned}$$

By the Cauchy-Schwarz inequality

$$\|\hat{u}^k\|^2_{L^2(\partial T)} \leq \frac{3\|[\![\boldsymbol{\sigma}^k\cdot\mathbf{n}]\!]\|^2_{L^2(\partial T)} + 3\|[\![\boldsymbol{\beta}\cdot\mathbf{n}u^k]\!]\|^2_{L^2(\partial T)} + 3\|[\![\tau u^k]\!]\|^2_{L^2(\partial T)}}{4},$$

which implies

$$\sum_{\partial T}\|\hat{u}^k\|^2_{L^2(\partial T)} \leq \sum_{\partial T} 3\|\boldsymbol{\sigma}^k\|^2_{L^2(\partial T)} + 3(\|\boldsymbol{\beta}\cdot\mathbf{n}\|^2_{L^\infty(\partial T)} + \bar{\tau}^2)\|u^k\|^2_{L^2(\partial T)}. \qquad (2.46)$$

Combining (2.45) and (2.46), we get

$$\begin{aligned}
&\sum_T\left[\kappa^{-1}\|\boldsymbol{\sigma}^{k+1}\|^2_{L^2(T)} + \lambda\|u^{k+1}\|^2_{L^2(T)} + \tau_*\|u^{k+1}\|^2_{L^2(\partial T)}\right]\\
\leq\ & \sum_{\partial T}\left[\frac{3(\bar{\tau}\varepsilon+1)}{2\varepsilon}\|\boldsymbol{\sigma}^k\|^2_{L^2(\partial T)} + \frac{3(\|\boldsymbol{\beta}\cdot\mathbf{n}\|^2_{L^\infty(\partial T)}+\bar{\tau}^2)(\bar{\tau}\varepsilon+1)}{2\varepsilon}\|u^k\|^2_{L^2(\partial T)}\right.\\
&\left.+\frac{\bar{\tau}+\|\boldsymbol{\beta}\cdot\mathbf{n}\|_{L^\infty(\partial T)}}{2}\|u^{k+1}\|^2_{L^2(\partial T)} + \frac{\varepsilon}{2}\|\boldsymbol{\sigma}^{k+1}\|^2_{L^2(\partial T)}\right]. \qquad (2.47)
\end{aligned}$$

By the inverse trace inequality (2.30) we infer from (2.47) that

$$\begin{aligned}
&\sum_{\partial T}\left[(\frac{2ch\kappa^{-1}}{d(p+1)(p+2)} - \mathcal{C}_4)\|\boldsymbol{\sigma}^{k+1}\|^2_{L^2(\partial T)} + (\frac{2ch\lambda}{d(p+1)(p+2)} + \tau_* - \mathcal{C}_3)\|u^{k+1}\|^2_{L^2(\partial T)}\right]\\
&\leq \sum_{\partial T}\left[\mathcal{C}_1\|u^k\|^2_{L^2(\partial T)} + \mathcal{C}_2\|\boldsymbol{\sigma}^k\|^2_{L^2(\partial T)}\right],
\end{aligned}$$

which implies

$$\left\|\left(\boldsymbol{\sigma}^{k+1}, u^{k+1}\right)\right\|_{\mathcal{E}_h}^2 \leq \mathcal{D}\left\|\left(\boldsymbol{\sigma}^{k}, u^{k}\right)\right\|_{\mathcal{E}_h}^2,$$

where the constant $\mathcal{D}$ is computed as in (2.41). Therefore,

$$\left\|\left(\boldsymbol{\sigma}^{k+1}, u^{k+1}\right)\right\|_{\mathcal{E}_h}^2 \leq \mathcal{D}^{k+1}\left\|\left(\boldsymbol{\sigma}^{0}, u^{0}\right)\right\|_{\mathcal{E}_h}^2. \tag{2.48}$$

Inequalities (2.47) and (2.48) imply

$$\left\|\left(\boldsymbol{\sigma}^{k+1}, u^{k+1}\right)\right\|_{\Omega_h}^2 \leq (\mathcal{E}\mathcal{D}+\mathcal{F})\mathcal{D}^{k}\left\|\left(\boldsymbol{\sigma}^{0}, u^{0}\right)\right\|_{\mathcal{E}_h}^2,$$

and this concludes the proof. □

Remark 4. *For time-dependent convection-diffusion equation, we choose to discretize the spatial differential operators using HDG. For the temporal derivative, we use implicit time stepping methods, again with either the backward Euler or Crank–Nicolson method for simplicity. The iHDG approach in this case is almost identical to the one for the steady state equation except that we now have an additional L^2-term, $\left(u^{k+1}, v\right)_T/\Delta t$, in the local equation* (2.38b)*. This improves the convergence of iHDG. Indeed, the convergence analysis is almost identical except we now have $\lambda + 1/\Delta t$ in place of λ.*

2.5.2 Comment on iHDG methods for elliptic PDEs

In this section we consider elliptic PDEs with $\boldsymbol{\beta} = 0$, $\kappa = 1$ in (2.34). First, using the conditions for convergence derived in section 2.5.1, let us analyze the iHDG scheme with upwind flux (2.37). Now for elliptic PDEs the stabilization τ for upwind flux reduces to $\tau = 1$, and this violates the condition $\mathcal{B} > 0$ in Theorem 4. Therefore, for any mesh, the iHDG scheme with upwind flux will diverge, and this is also observed in our numerical experiments.

To fix the issue, we carry out an analysis similar to that in section 2.5.1, but this time without a specific τ. Taking $\varepsilon = \frac{1}{\bar{\tau}}$, the condition $\mathcal{B} > 0$ dictates the following mesh-dependent τ for the convergence of the iHDG scheme:

$$\bar{\tau} > \mathcal{O}\left(\frac{d(p+1)(p+2)}{4h}\right).$$

This result shows that the convergence of the iHDG scheme with upwind flux for elliptic PDEs requires mesh-dependent stabilization. It is also worth noting that this coincides with the form of stabilization used in hybridizable interior penalty methods [61] for elliptic PDEs, even though the schemes described in [61] and iHDG are different.

Guided by the above analysis, we take $\tau = \bar{\tau} = \tau_* = \frac{\gamma(p+1)(p+2)}{h}$, where $\gamma > \frac{d}{4}$ is a constant we still need to enforce $\mathcal{D} < 1$ for the convergence of the iHDG scheme. Generally this requires four conditions depending upon minimum and maximum of constants $\mathcal{A}, \mathcal{B}$ as in Theorem 4. However, the simple choice of $\tau = \frac{\gamma(p+1)(p+2)}{h}$ makes $\mathcal{A} = \mathcal{C}_1$, and the number of conditions is reduced to two depending on whichever term in $\mathcal{B}$ is minimum. They are given by

$$h > \mathcal{O}\left(\frac{\sqrt{23\gamma d}(p+1)(p+2)}{2\sqrt{\lambda}}\right) \quad \text{and} \quad h > \mathcal{O}\left(\frac{\sqrt{24d}(p+1)(p+2)\gamma}{\sqrt{4\gamma - d}}\right).$$

We can see that the iHDG scheme as an iterative solver is conditionally convergent for diffusion dominated PDEs: that is if the mesh is too fine, then the above conditions are violated and the scheme diverges.

2.6 Numerical results

In this section various numerical results supporting the theoretical results are provided for the 2D and 3D transport equations, the linearized shallow water equation,

and the convection-diffusion equation in different regimes. In this section we also use the notation N_{el} to denote the total number of elements which is same as N_T.

2.6.1 Steady state transport equation

The goal is to verify Theorems 1 and 2 for the transport equation (2.5) in 2D and 3D settings using the upwind HDG and the NPC fluxes.

2.6.1.1 2D steady state transport equation with discontinuous solution

We consider the case similar to the one in [25, 82] where $f = 0$ and $\boldsymbol{\beta} = (1 + sin(\pi y/2), 2)$ in (2.5). The domain is $[0, 2] \times [0, 2]$ and the inflow boundary conditions are given by

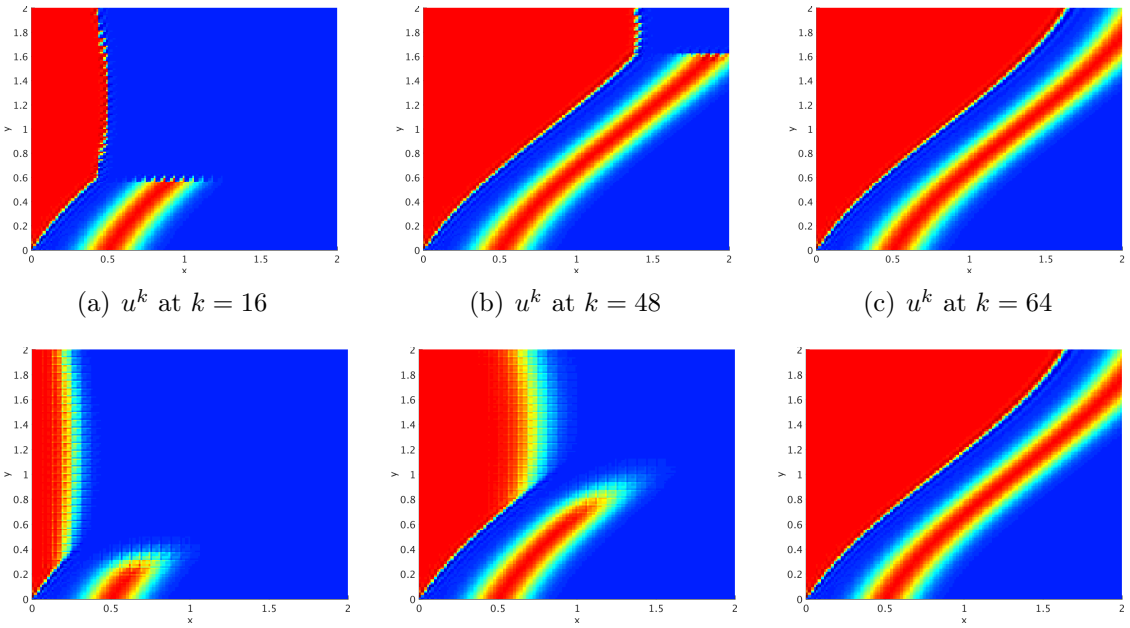
$$g = \begin{cases} 1 & x = 0, 0 \leq y \leq 2 \\ \sin^6(\pi x) & 0 < x \leq 1, y = 0 \\ 0 & 1 \leq x \leq 2, y = 0 \end{cases}.$$

To terminate the iHDG algorithm, we use the following stopping criterion:

$$\|u^k - u^{k-1}\|_{L^2} < 10^{-10}, \tag{2.49}$$

i.e., iHDG stops when there is insignificant change between two successive iterations.

The evolution of the iterative solution for the mesh with 1024 elements and solution order 4 using both upwind and NPC fluxes is shown in figure 2.1. In both cases, we observe that the iterative solution evolves from inflow to outflow as the number of iterations increases. Thanks to the built-in upwinding mechanism of the iHDG algorithm, this implicit marching is automatic; that is, we do not order the elements to march in the flow direction. As can be seen, iHDG with NPC flux converges faster (in fact, in a finite number of iterations), as predicted by Theorem 2.



(a) u^k at $k = 16$ (b) u^k at $k = 48$ (c) u^k at $k = 64$

(d) u^k at $k = 16$ (e) u^k at $k = 48$ (f) u^k at $k = 196$

Figure 2.1: Evolution of the iterative solution for the 2D transport equation using the NPC flux (top row) and the upwind HDG flux (bottom row).

N_{el}(2D)	N_{el}(3D)	p	2D solution		3D solution	
			Upwind	NPC	Upwind	NPC
16	8	3	65	9	39	7
64	64	3	91	17	49	12
256	512	3	133	33	79	23
1024	4096	3	209	65	136	47
16	8	4	65	9	35	6
64	64	4	87	17	51	12
256	512	4	129	33	83	24
1024	4096	4	196	64	143	48

Table 2.1: The number of iterations taken by the iHDG algorithm using NPC and upwind HDG fluxes for the transport equation in 2D and 3D settings.

The fourth and fifth columns of Table 2.1 show the number of iterations required to converge for both the fluxes with different meshes and solution orders 3 and

4. We observe that the number of iterations is (almost) independent of solution order[5] for both the fluxes, which is in agreement with the theoretical results in Theorems 1 and 2. *This is important for high-order methods; i.e., the solution order (and hence accuracy) can be increased while keeping the number of iHDG iterations unchanged.*

2.6.1.2 3D steady state transport equation with smooth solution

In this example we choose $\boldsymbol{\beta} = (z, x, y)$ in (2.5). Also, we take the following exact solution:

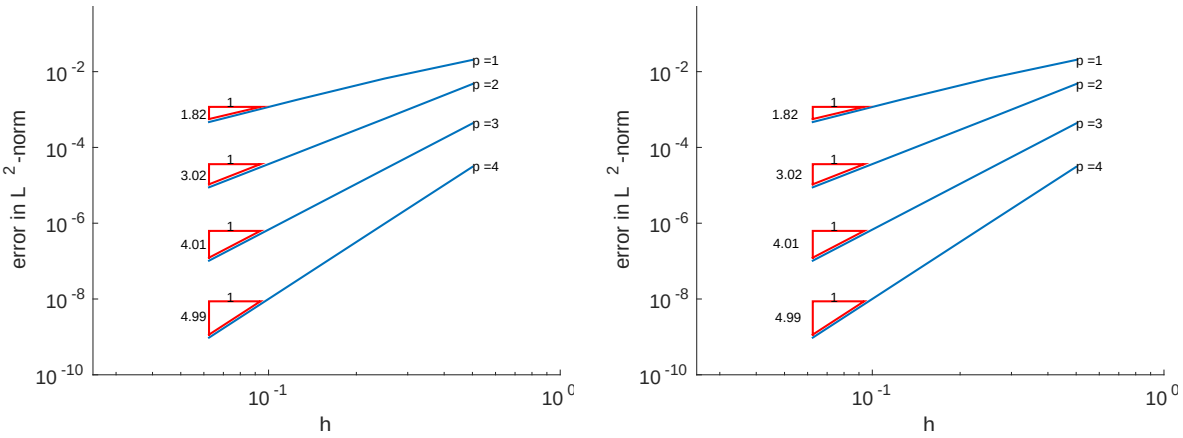
$$u^e = \frac{1}{\pi}\sin(\pi x)\cos(\pi y)\sin(\pi z).$$

The forcing is selected in such a way that it corresponds to the exact solution. Here the domain is $[0,1] \times [0,1] \times [0,1]$ with faces $x = 0$, $y = 0$, and $z = 0$ as the inflow boundaries. A structured hexahedral mesh is used for the simulations. Since we know the exact solution, we use the following stopping criterion:

$$|\|u^k - u^e\|_{L^2(\Omega)} - \|u^{k-1} - u^e\|_{L^2(\Omega)}| < 10^{-10}. \tag{2.50}$$

Figure 2.2 shows the h-convergence of the HDG discretization with the iHDG iterative solver. The convergence is optimal with rate $(p+1)$ for both fluxes. Figure 2.3 compares the convergence history of the iHDG solver in the log-linear scale. As proved in Theorem 1, the iHDG with upwind flux is exponentially convergent with respect to the number of iterations k, while the convergence is attained in a finite number of iterations for the NPC flux, as predicted in Theorem 2. Note that the stagnation region observed near the end of each curve is due to the fact that for a particular mesh size h and solution order p we can achieve only as much accuracy as prescribed by the HDG discretization error and cannot go beyond that. Numerical

[5]The results for $p = \{1, 2\}$ are not shown, as the number of iterations is very similar to that of the $p = \{3, 4\}$-cases.



(a) Upwind flux (b) NPC flux

Figure 2.2: h-convergence of the HDG method using iHDG with upwind and NPC fluxes.

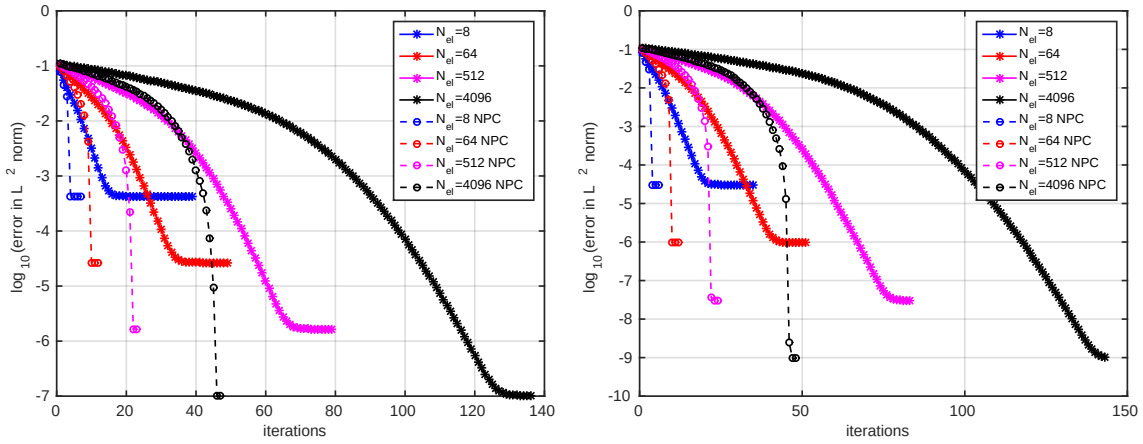
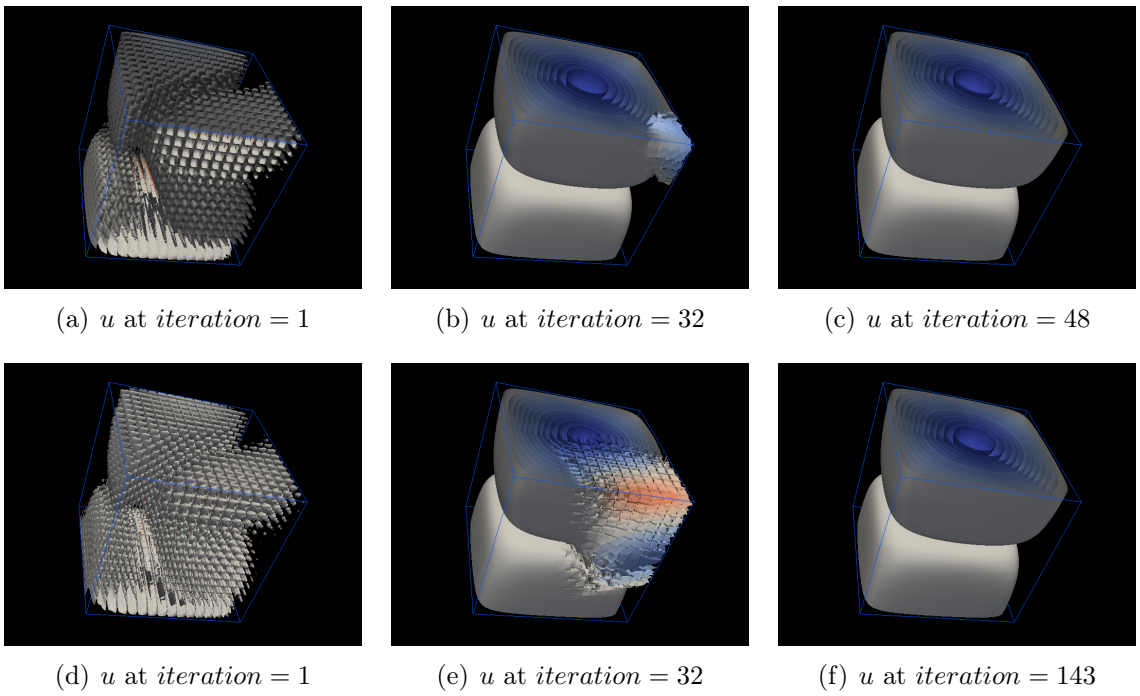


Figure 2.3: Error history in terms of the number iterations for solution order $p = 3$ (left) and $p = 4$ (right) as the mesh is refined.

results for different solution orders[6] also verify the fact that the convergence of the iHDG algorithm is independent of the solution order p. The evolution of the iHDG solution in terms of the number of iterations is shown in Figure 2.4. Again, for the scalar transport equation, iHDG automatically marches the solution from the inflow

[6]Here the results for $p = \{1, 2\}$ are omitted for brevity.



(a) u at $iteration = 1$ (b) u at $iteration = 32$ (c) u at $iteration = 48$

(d) u at $iteration = 1$ (e) u at $iteration = 32$ (f) u at $iteration = 143$

Figure 2.4: Evolution of the iHDG solution in terms of the number of iterations for the NPC flux (top row) and the upwind HDG flux (bottom row).

to the outflow. We also record in the sixth and seventh columns of Table 2.1 the number of iterations that the iHDG algorithm took for both the fluxes. As predicted by our theoretical findings, the number of iterations is independent of the solution order.

2.6.2 Linearized shallow water equations

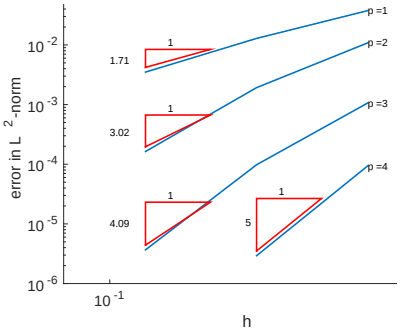
In this section we consider (2.25) with a linear standing wave, for which we set $\Phi = 1$, $f = 0$, $\gamma = 0$ (zero bottom friction), and $\boldsymbol{\tau} = 0$ (zero wind stress). The domain is $[0, 1] \times [0, 1]$, and the wall boundary condition is applied on the domain

boundary. The following exact solution [70] is taken:

$$\phi^e = \cos(\pi x)\cos(\pi y)\cos(\sqrt{2}\pi t), \tag{2.51a}$$

$$u^e = \frac{1}{\sqrt{2}}\sin(\pi x)\cos(\pi y)\sin(\sqrt{2}\pi t), \tag{2.51b}$$

$$v^e = \frac{1}{\sqrt{2}}\cos(\pi x)\sin(\pi y)\sin(\sqrt{2}\pi t). \tag{2.51c}$$



N_{el}	Solution order			
	1	2	3	4
16	12	13	9	10
64	11	12	7	9
256	9	11	7	8
1024	7	10	7	7

Figure 2.5: h-convergence of iHDG for 10^5 time steps with $\Delta t = 10^{-6}$ (left) and the number of iHDG iterations per time step with $\Delta t = \frac{h}{(p+1)(p+2)}$ (right) for the linearized shallow water equation.

The iHDG algorithm with upwind HDG flux described in Section 2.4 along with the Crank–Nicolson method for time discretization is employed in this problem. The convergence of the solution is presented in Figure 2.5. Here we have taken $\Delta t = 10^{-6}$ as the stepsize with 10^5 steps. As can be seen, the optimal convergence rate of $(p+1)$ is attained. The number of iterations required per time step in this case is constant and is always equal to 2 for all meshes and solution orders considered. The reason is due to (i) a warm-start strategy, that is, the initial guess for each time step is taken as the solution of the previous time step, and (ii) small time stepsize.

Following Remark 3, we choose $\Delta t = \frac{h}{(p+1)(p+2)}$ and report the number of iterations for different meshes and solution orders in Figure 2.5. Clearly, finer meshes and higher solution orders, require smaller time stepsizes, and hence a smaller number of iterations, for the iHDG algorithm to converge.

2.6.3 Convection-Diffusion Equation

In this section (2.34) is considered with the exact solution taken as

$$u^e = \frac{1}{\pi}\sin(\pi x)\cos(\pi y)\sin(\pi z).$$

The forcing is chosen such that it corresponds to the exact solution. The domain is the same as the one in section 2.6.1.2. The Dirichlet boundary condition based on the exact solution is applied on the boundary faces, and the stopping criterion is same as in (2.50).

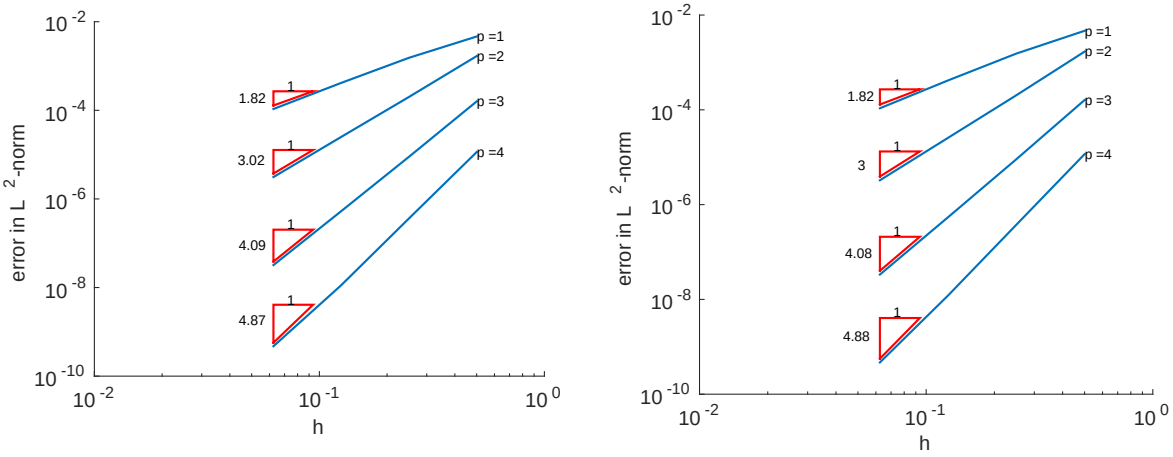
2.6.3.1 Convection dominated regime

Let us consider $10^{-3} \leq \kappa \leq 10^{-6}$, $\nu = 1$, and $\boldsymbol{\beta} = (1+z, 1+x, 1+y)$. Since the maximum velocity in this example is $\mathcal{O}(1)$, this represents a convection dominated regime. Figure 2.6 shows the optimal h-convergence of the iHDG method with the upwind HDG flux[7] for $\kappa = 10^{-3}$ and $\kappa = 10^{-6}$, respectively. The error history for solution orders $p = \{3, 4\}$ is given in Figure 2.7. As expected, for $\kappa = 10^{-6}$, the iHDG method with either upwind or NPC flux behaves similarly to the pure convection case. From Table 2.2 the convergence of the upwind iHDG approach remains the same; that is, the number of iterations is insensitive to the diffusion coefficient κ. The iHDG approach with the NPC flux improves as κ decreases. This is because the stabilization (τ) of the NPC flux contains κ, whereas the stabilization of the upwind flux does not.

2.6.3.2 Mixed (hyperbolic-elliptic) regime

In this regime, we take $\kappa = 10^{-2}$, $\nu = 1$, and $\boldsymbol{\beta} = (1 + z, 1 + x, 1 + y)$. Table 2.2 shows that both upwind and NPC fluxes fail to converge for a number of cases

[7]The convergence with the NPC flux is similar and hence not shown.



(a) $\kappa = 10^{-3}$ (b) $\kappa = 10^{-6}$

Figure 2.6: h-convergence of iHDG method with upwind HDG flux for $\kappa = \{10^{-3}, 10^{-6}\}$.

("$*$" indicates divergence) though the upwind iHDG is more robust. This is due to the violation of the necessary condition $\mathcal{B} > 0$ in Theorem 4 for finer meshes. From section 2.5.1, by choosing $\varepsilon = \mathcal{O}\left(\frac{1}{\tau}\right)$ for both upwind and NPC fluxes we can estimate the minimum mesh sizes, and they are shown in Table 2.3. Comparing the second and third rows of Table 2.3 with the first, third, and sixth columns of Table 2.2 we see that the numerical results differ from the theoretical estimates by a constant. In case of the upwind flux the constant is 2, and for the NPC flux it is 4. That is, if we multiply the second and third rows of Table 2.3 with 2 and 4, respectively, and compare it with the numerical results in Table 2.2, we can see an agreement.

h	p	Upwind flux			NPC flux		
		$\kappa = 10^{-2}$	$\kappa = 10^{-3}$	$\kappa = 10^{-6}$	$\kappa = 10^{-2}$	$\kappa = 10^{-3}$	$\kappa = 10^{-6}$
0.5	1	24	23	23	10	7	5
0.25	1	30	34	35	21	14	12
0.125	1	50	55	56	94	26	22
0.0625	1	90	94	97	*	52	46
0.5	2	26	24	25	13	8	5
0.25	2	41	42	42	22	15	12
0.125	2	66	67	67	*	26	23
0.0625	2	*	109	110	*	59	46
0.5	3	27	31	31	21	8	5
0.25	3	33	33	38	*	16	12
0.125	3	*	58	60	*	30	24
0.0625	3	*	102	106	*	59	48
0.5	4	26	27	27	73	8	5
0.25	4	50	41	43	*	16	12
0.125	4	*	71	72	*	32	24
0.0625	4	*	123	125	*	71	48

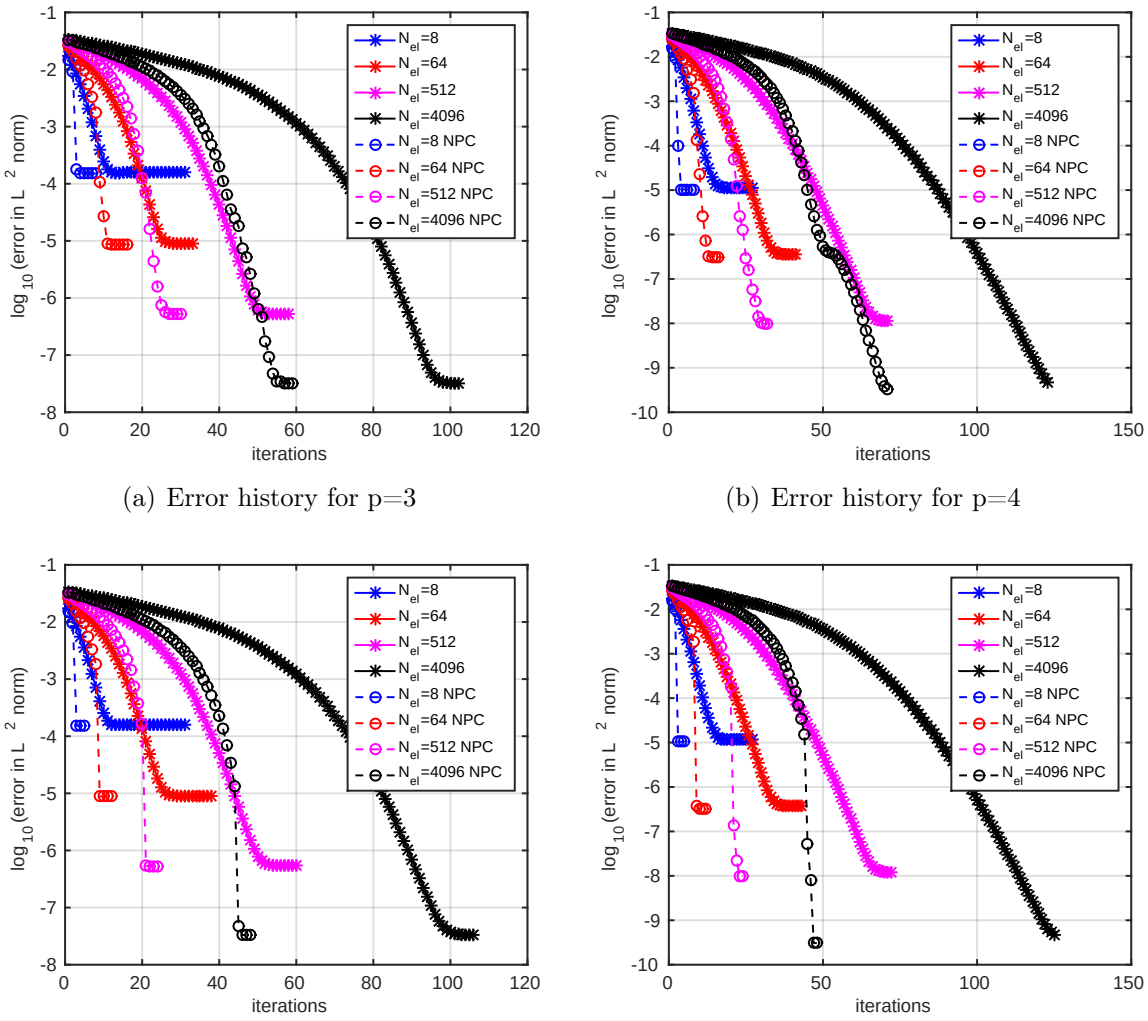
Table 2.2: The number of iHDG iterations for various κ with upwind and NPC fluxes.

Flux	$p = 1$	$p = 2$	$p = 3$	$p = 4$
Upwind	$h > 0.019$	$h > 0.0375$	$h > 0.0625$	$h > 0.094$
NPC	$h > 0.0225$	$h > 0.045$	$h > 0.075$	$h > 0.1125$

Table 2.3: Theoretical estimates on the minimum mesh size for convergence of upwind and NPC fluxes for $\kappa = 0.01$ from section 2.5.1.

2.6.3.3 Diffusion regime (elliptic equation)

As an example for the diffusion limit, we take $\boldsymbol{\beta} = 0$ and $\kappa = 1$. In order to verify the conditional convergence in section 2.5.2, we choose three different values of ν in the set $\{1, 10, 100\}$. Recall in section 2.5.2 that the upwind flux does not converge for $\kappa = 1$ and $\boldsymbol{\beta} = 0$ (pure diffusion regime). It is true for NPC flux also, due to lack of mesh-dependent stabilization. From the conditions derived in section 2.5.2 we choose $\tau = \frac{(p+1)(p+2)}{h}$. The stopping criterion is taken as in (2.50).



(a) Error history for p=3

(b) Error history for p=4

(c) Error history for p=3

(d) Error history for p=4

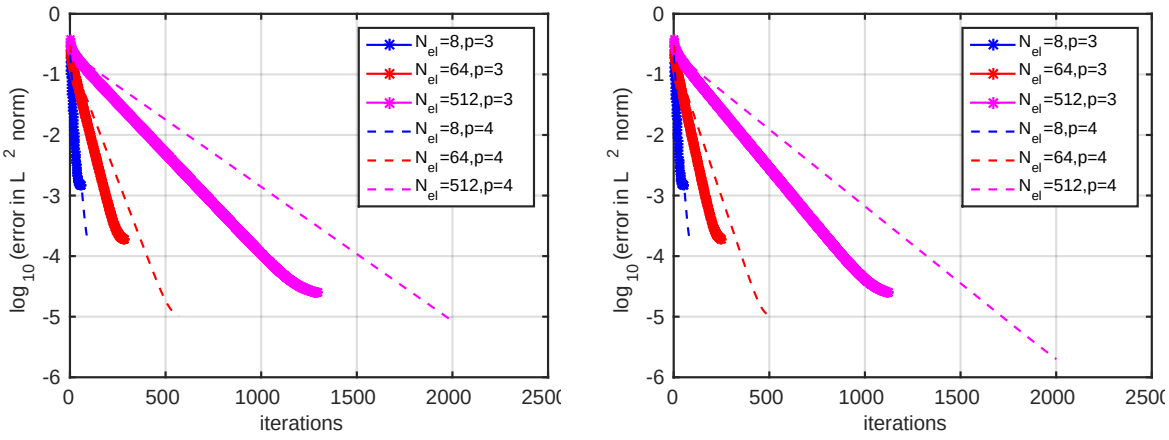
Figure 2.7: Convergence history for the iHDG method with upwind and NPC fluxes for $\kappa = 10^{-3}$ (top row) and $\kappa = 10^{-6}$ (bottom row).

In Table 2.4 we compare the number of iterations the iHDG algorithm takes to converge for $\nu = \{1, 10\}$ cases. Note that "$*$" indicates that the scheme either reaches 2000 iterations or diverges. The convergent condition in section 2.5.2 is equivalent to $h > \mathcal{O}\left(\frac{1}{\sqrt{\lambda}}\right)$, and since $\sqrt{\lambda} = \sqrt{\nu}$ we see similar convergence/divergence behavior for both $\nu = 1$ and 10 (because the lower bound for h is of the same order for these

cases). For $\nu = 100$ the lower bound for h is one order of magnitude smaller, and this allows us to obtain convergence for two additional cases: (i) $N_{el} = 512$ and $p = 4$ with 1160 iterations; and (ii) $N_{el} = 4096$ and $p = 2$ with 1450 iterations.

N_{el}	$\nu = 1$				$\nu = 10$			
	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 1$	$p = 2$	$p = 3$	$p = 4$
8	4	24	60	98	3	17	54	90
64	37	119	285	569	32	108	249	497
512	158	527	1296	*	130	429	1124	*
4096	1519	*	*	*	1178	*	*	*

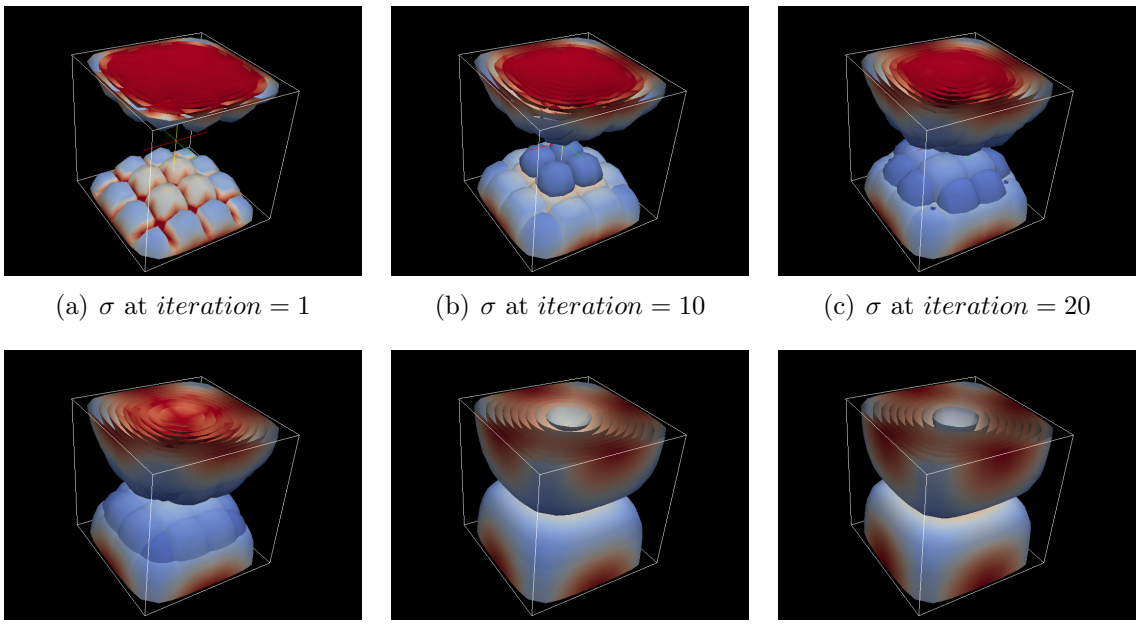
Table 2.4: The number of iHDG iterations for $\nu = 1$ and $\nu = 10$ with $\tau = \frac{(p+1)(p+2)}{h}$.



(a) $\nu = 1$ (b) $\nu = 10$

Figure 2.8: Convergence of the iHDG algorithm for different mesh size h and solution order $p = \{3, 4\}$ for 3D elliptic equation with $\nu = 1$ and $\nu = 10$.

Figure 2.8 shows the convergence history for different meshes and solution orders for $\nu = 1$ and $\nu = 10$. We notice that the convergence trend is different from the pure convection and convection-diffusion cases; that is, it is exponential in the number of iterations starting from the beginning, but the rate is less. We also show in Figure 2.9 the evolution of the magnitude of $\boldsymbol{\sigma}$ ($\sigma = |\boldsymbol{\sigma}|$) with respect to the



(a) σ at $iteration = 1$ (b) σ at $iteration = 10$ (c) σ at $iteration = 20$

(d) σ at $iteration = 30$ (e) σ at $iteration = 90$ (f) σ at $iteration = 569$

Figure 2.9: Evolution of σ with respect to the number of iterations for $\nu = 1$.

number of iterations for $N_{el} = 64$ and solution order $p = 4$. Unlike the convection (or convection-dominated) case, the convergence of the iHDG solution in this case does not have a preferable direction, as the elliptic nature of the PDEs is encoded in the iHDG algorithm via the numerical flux.

2.6.4 Time dependent convection-diffusion equation

In this section we consider the following equation:

$$\kappa^{-1}\boldsymbol{\sigma}^e + \nabla u^e = 0 \quad \text{in } \Omega, \tag{2.52a}$$

$$\frac{\partial u^e}{\partial t} + \nabla \cdot \boldsymbol{\sigma}^e + \boldsymbol{\beta} \cdot \nabla u^e = 0 \quad \text{in } \Omega. \tag{2.52b}$$

We are interested in the transport of the contaminant concentration [112, 12] with diffusivity $\kappa = 0.01$ in a 3D domain $\Omega = [0, 5] \times [-1.25, 1.25] \times [-1.25, 1.25]$ with

convective velocity field $\boldsymbol{\beta}$ given as[8]

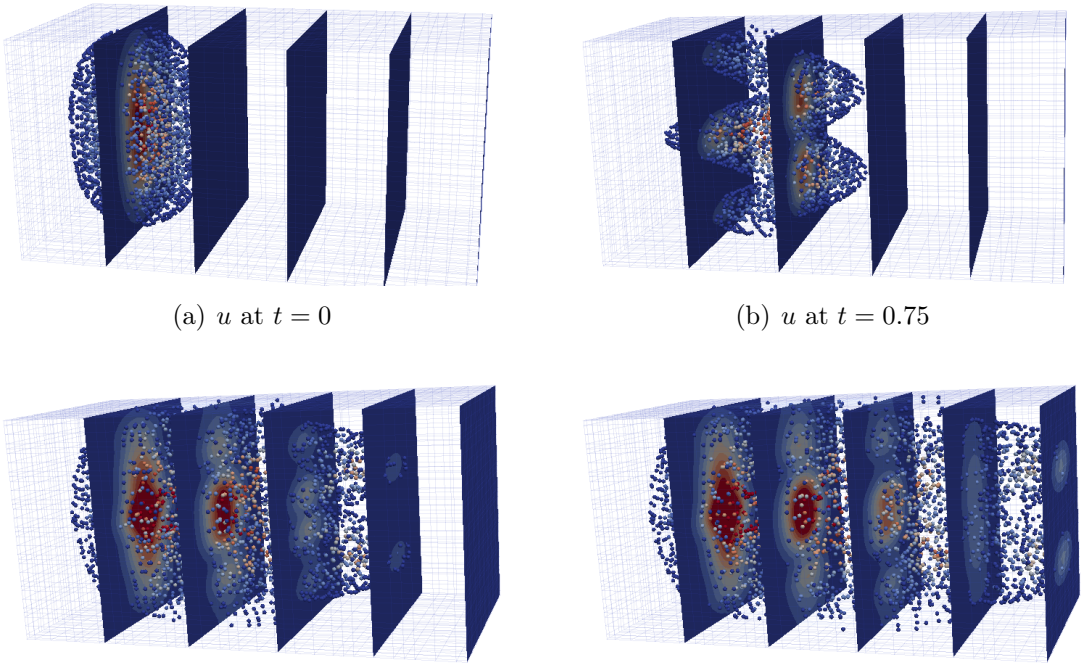
$$\boldsymbol{\beta} = \left(1 - e^{\gamma x}\cos(2\pi y), \frac{\gamma}{2\pi}e^{\gamma x}\sin(2\pi y), 0\right),$$

where $\gamma = \frac{Re}{2} - \sqrt{\frac{Re^2}{4} + 4\pi^2}$ and $Re = 100$. Starting from $t = 0$, and for every second afterward, the same distribution of contaminant concentration of the form

$$u_0 = e^{\frac{(x-1)^2+y^2+z^2}{0.5^2}} + e^{\frac{(x-1)^2+(y-0.5)^2+z^2}{0.5^2}} + e^{\frac{(x-1)^2+(y+0.5)^2+z^2}{0.5^2}}$$

is injected into the flow field. A time stepsize of $\Delta t = 0.025$ is selected and the simulation is run for 400 time steps, i.e., until $T = 10$, with the Crank–Nicolson method. Here, we use the mesh with $N_{el} = 512$ elements and solution order $p = 4$. On the left boundary, i.e., $x = 0, -1.25 \leq y \leq 1.25, -1.25 \leq z \leq 1.25$, the Dirichlet boundary condition $u = 0$ is applied, while on the remaining boundary faces, we employ the homogeneous Neumann boundary condition $\nabla u \cdot \mathbf{n} = 0$. The Peclet number for this problem is 200. This exhibits a wide range of mixed hyperbolic and parabolic regimes and hence is a good test bed for the proposed iHDG scheme. In this example, the iHDG algorithm with NPC flux does not converge even for coarse meshes and low solution orders (this can be seen in section 2.6.3). We therefore show numerical results only for the upwind HDG flux in Figure 2.10. The scheme requires approximately 46 iterations for each time step to reach the stopping criterion $\|u^k - u^{k-1}\|_{L^2} < 10^{-6}$. Since $\nabla \cdot \boldsymbol{\beta} = 0$, $\nu = 0$ and $\lambda = 1/\Delta t$ (see Remark 4), we obtain from section 2.5.1 the following estimate for the time stepsize: $\Delta t = \mathcal{O}\left(\frac{h}{(p+1)(p+2)}\right)$. Using this estimate, we compare the number of iterations the iHDG algorithm takes to converge for different meshes and solution orders in Table 2.5. We do not obtain convergence for $N_{el} = 4096$ and solution orders equal to 3 and 4: the main reason is that this problem is in the mixed hyperbolic and parabolic regime, and the above setting does not satisfy the conditions for convergence (see Section 2.6.3).

[8]Here, $\boldsymbol{\beta}$ is an extension of the 2D analytical solution of Kovasznay flow [90].



(a) u at $t = 0$ (b) u at $t = 0.75$

(c) u at $t = 2.0$ (d) u at $t = 3.0$

Figure 2.10: Solution u as a function of time using the iHDG algorithm with the upwind flux for the contaminant transport problem (2.52).

N_{el}	$p = 1$	$p = 2$	$p = 3$	$p = 4$
8	12	24	23	22
64	26	26	22	20
512	21	23	17	37
4096	17	18	*	*

Table 2.5: The number of iHDG iterations per time step for the contaminant transport problem with various solution orders and mesh sizes.

2.7 Discussion

In this chapter we have presented an iterative solver, namely iHDG, for HDG discretizations of linear PDEs. The method exploits the structure of HDG discretization and ideas from domain decomposition methods (DDMs). One of the key features of the iHDG algorithm is that it requires only local solves, i.e., element-by-element

and face-by-face, completely independent of each other, during each iteration. It can also be considered as a block Gauss–Seidel method for the augmented HDG system with volume and weighted trace unknowns. Thanks to the built-in stabilization via the weighted trace and the structure of the HDG discretization, unlike traditional Gauss–Seidel schemes, the convergence of iHDG is independent of the ordering of the unknowns. Using an energy approach, we rigorously derive the conditions under which the iHDG algorithm is convergent for the transport equation, the linearized shallow water equation, and the convection-diffusion equation. In particular, for the scalar transport equation, the algorithm is convergent for all meshes and solution orders, and the convergence rate is independent of solution order. This feature makes the iHDG solver especially suitable for high-order DG methods; that is, high-order (and hence more accurate) solutions do not require more iterations. The scheme in fact performs an implicit marching, and the solution converges in patches of elements automatically from the inflow to the outflow boundaries. For the linearized shallow water equation, we prove that the convergence is conditional on the meshsize and the solution order. Similar conditional convergence is also shown for the convection-diffusion equation in first order form. We have studied the performance of the scheme in convection dominated, mixed (hyperbolic-elliptic), and diffusion regimes and numerically verified our theoretical results.

Chapter 3

An Improved Iterative HDG Approach for Partial Differential Equations

3.1 Improvement to the iHDG approach

To reduce the cost of solving the trace system, in chapter 2 [108] we introduced the iHDG approach the idea of which is to break the coupling between $\hat{\mathbf{u}}$ and $\mathbf{u}$ in (2.2) by iteratively solving for $\mathbf{u}$ in terms of $\hat{\mathbf{u}}$ in (2.2a), and $\hat{\mathbf{u}}$ in terms of $\mathbf{u}$ in (2.2b). Let us call it as iHDG-I to distinguish it from the approach developed in this chapter[1].

A number of questions need to be addressed for the iHDG-I approach. First, with the upwind flux it theoretically takes infinite number of iterations to converge for the scalar transport equation. Second, it is conditionally convergent for the linearized shallow water system; in particular, it blows up for fine meshes and/or large time stepsizes. Furthermore, we have not been able to estimate the number of iterations as a function of time stepsize, solution order, and meshsize. Third, it is also conditionally convergent for the convection-diffusion equation, especially in the diffusion-dominated regime.

The approach constructed in this chapter, which we call iHDG-II, overcomes all the aforementioned shortcomings. In particular, it converges in a finite number of iterations for the scalar transport equation and is unconditionally convergent for both

[1]The contents of this chapter are largely based on the published manuscript [109]. The contributions of the author in the article ranged from numerical implementation of the algorithm, theoretical analysis and writing the manuscript.

the linearized shallow water system and the convection-diffusion equation. Moreover, we provide several additional findings: 1) we make a connection between iHDG and the parareal method, which reveals interesting similarities and differences between the two methods; 2) we show that iHDG can be considered as a *locally implicit* method, and hence being somewhat in between fully explicit and fully implicit approaches; 3) for both the linearized shallow water system and the convection-diffusion equation, using an asymptotic approximation, we uncover a relationship between the number of iterations and time stepsize, solution order, meshsize and the equation parameters. This allows us to choose the time stepsize such that the number of iterations is approximately independent of the solution order and the meshsize; 4) we show that iHDG-II has improved stability and convergence rates over iHDG-I; and 5) we provide both strong and weak scalings of the iHDG-II approach up to $16,384$ cores.

We now present a detailed construction of the iHDG-II approach. We define the approximate solution for the volume variables at the $(k+1)$th iteration using the local equation (2.2a) as

$$\begin{aligned} -\left(\mathbf{F}\left(\mathbf{u}^{k+1}\right), \nabla \mathbf{v}\right)_T + \left\langle \mathbf{F}\left(\mathbf{u}^{k+1}\right) \cdot \mathbf{n} + |\mathbf{A}| \left(\mathbf{u}^{k+1} - \hat{\mathbf{u}}^{k,k+1}\right), \mathbf{v} \right\rangle_{\partial T} \\ + \left(\mathbf{C}\mathbf{u}^{k+1}, \mathbf{v}\right)_T = \left(\mathbf{f}, \mathbf{v}\right)_T, \end{aligned} \tag{3.1}$$

where the weighted trace $|\mathbf{A}|\,\hat{\mathbf{u}}^{k,k+1}$ is computed from (2.2b) using volume unknown in element T at the $(k+1)$th iteration, i.e., $\left(\mathbf{u}^{k+1}\right)^-$, and volume solution of the neighbors at the (k)th iteration, i.e., $\left(\mathbf{u}^{k}\right)^+$:

$$\begin{aligned} \left\langle 2\,|\mathbf{A}|\,\hat{\mathbf{u}}^{k,k+1}, \boldsymbol{\mu} \right\rangle_{\partial T} = & \left\langle |\mathbf{A}| \left\{ \left(\mathbf{u}^{k+1}\right)^- + \left(\mathbf{u}^{k}\right)^+ \right\}, \boldsymbol{\mu} \right\rangle_{\partial T} \\ & + \left\langle \mathbf{F}\left\{ \left(\mathbf{u}^{k+1}\right)^- \right\} \cdot \mathbf{n}^- + \mathbf{F}\left\{ \left(\mathbf{u}^{k}\right)^+ \right\} \cdot \mathbf{n}^+, \boldsymbol{\mu} \right\rangle_{\partial T}. \end{aligned} \tag{3.2}$$

Algorithm 2 summarizes the iHDG-II approach. Compared to iHDG-I, iHDG-II improves the coupling between $\hat{\mathbf{u}}$ and $\mathbf{u}$ while still avoiding intra-iteration communication between elements. The trace $\hat{\mathbf{u}}$ is double-valued during the course of iterations

for iHDG-II and in the event of convergence it becomes single valued up to a specified tolerance. Another principal difference is that while the well-posedness of iHDG-I elemental local solves is inherited from the original HDG counterpart, *it has to be shown for iHDG-II*. This is due to the new way of computing the weighted trace in (3.2) that involves $\mathbf{u}^{k+1}$, and hence changing the structure of the local solves. Similar and independent work for HDG methods for elliptic/parabolic problems have appeared in [60, 61, 62]. Here, we are interested in pure hyperbolic equations/systems and convection-diffusion equations. Unlike existing matrix-based approaches, our convergence analysis is based on an energy approach that exploits the variational structure of HDG methods. Our framework is more general: indeed it recovers the contraction factor results in [60] for elliptic equations as one of the special cases.

Algorithm 2 The iHDG-II approach.

Ensure: Given initial guess $\mathbf{u}^0$, compute the weighted trace $|\mathbf{A}|\,\hat{\mathbf{u}}^{0,1}$ using (3.2).

1: **while** not converged **do**
2: Solve the local equation (3.1) for $\mathbf{u}^{k+1}$ using the weighted trace $|\mathbf{A}|\,\hat{\mathbf{u}}^{k,k+1}$.
3: Compute $|\mathbf{A}|\,\hat{\mathbf{u}}^{k+1,k+2}$ using (3.2).
4: Check convergence. If yes, **exit**, otherwise **set** $k = k + 1$ and **continue**.
5: **end while**

3.2 iHDG-II for linear hyperbolic PDEs

In this section we show that iHDG-II improves upon iHDG-I in many aspects discussed in section 2.3. The PDEs of interest are the (steady and time dependent) transport equation, and the linearized shallow water system [108].

3.2.1 Transport equation

Let us start with the (steady) transport equation (2.5), applying the iHDG-II algorithm 2 to the upwind HDG discretization [25] we obtain the approximate solution u^{k+1} at the $(k+1)$th iteration restricted on each element T via the following

independent local solve:

$$-\left(\left(u^{k+1}\right)^{-},\nabla\cdot\left(\boldsymbol{\beta}v\right)\right)_{T}+\left\langle \boldsymbol{\beta}\cdot\mathbf{n}^{-}\left(u^{k+1}\right)^{-}+\left|\boldsymbol{\beta}\cdot\mathbf{n}\right|\left\{\left(u^{k+1}\right)^{-}-\hat{u}^{k,k+1}\right\},v\right\rangle_{\partial T}$$
$$=\left(f,v\right)_{T},\quad(3.3)$$

where the weighted trace $\left|\boldsymbol{\beta}\cdot\mathbf{n}\right|\hat{u}^{k,k+1}$ is computed using information from the previous iteration and current iteration as

$$2\left|\boldsymbol{\beta}\cdot\mathbf{n}\right|\hat{u}^{k,k+1}=\left\{\boldsymbol{\beta}\cdot\mathbf{n}^{-}\left(u^{k+1}\right)^{-}+\boldsymbol{\beta}\cdot\mathbf{n}^{+}\left(u^{k}\right)^{+}\right\}$$
$$+\left|\boldsymbol{\beta}\cdot\mathbf{n}\right|\left\{\left(u^{k+1}\right)^{-}+\left(u^{k}\right)^{+}\right\}.\quad(3.4)$$

Next we study the convergence of the iHDG-II method (3.3), (3.4). Since (2.5) is linear as argued in chapter 2 it is sufficient to show that the algorithm converges to the zero solution for the homogeneous equation with zero forcing $f=0$ and zero boundary condition $g=0$. First, we will prove the well-posedness of the local solver (3.3).

Lemma 1. *Assume $-\nabla\cdot\boldsymbol{\beta}\geq\alpha>0$, i.e. (2.5) is well-posed. Then the local solver (3.3) of the iHDG-II algorithm for the transport equation is well-posed.*

Proof. Taking $v=\left(u^{k+1}\right)^{-}$ in (3.3), substituting (3.4) in (3.3) and applying homogeneous forcing condition yield

$$-\left(\left(u^{k+1}\right)^{-},\nabla\cdot\left\{\boldsymbol{\beta}\left(u^{k+1}\right)^{-}\right\}\right)_{T}+\frac{1}{2}\left\langle\left(\boldsymbol{\beta}\cdot\mathbf{n}^{-}+\left|\boldsymbol{\beta}\cdot\mathbf{n}\right|\right)\left(u^{k+1}\right)^{-},\left(u^{k+1}\right)^{-}\right\rangle_{\partial T}$$
$$=\frac{1}{2}\left\langle\left(\boldsymbol{\beta}\cdot\mathbf{n}^{+}+\left|\boldsymbol{\beta}\cdot\mathbf{n}\right|\right)\left(u^{k}\right)^{+},\left(u^{k+1}\right)^{-}\right\rangle_{\partial T}.\quad(3.5)$$

Since

$$\left(\left(u^{k+1}\right)^{-},\nabla\cdot\left\{\boldsymbol{\beta}\left(u^{k+1}\right)^{-}\right\}\right)_{T}=\left(\left(u^{k+1}\right)^{-},\nabla\cdot\boldsymbol{\beta}\left(u^{k+1}\right)^{-}\right)_{T}$$
$$+\left(\left(u^{k+1}\right)^{-},\boldsymbol{\beta}\cdot\nabla\left(u^{k+1}\right)^{-}\right)_{T},$$

integrating by parts the second term on the right hand side

$$\left(\left(u^{k+1}\right)^{-},\nabla\cdot\left\{\boldsymbol{\beta}\left(u^{k+1}\right)^{-}\right\}\right)_{T}=\left(\left(u^{k+1}\right)^{-},\nabla\cdot\boldsymbol{\beta}\left(u^{k+1}\right)^{-}\right)_{T}$$
$$-\left(\left(u^{k+1}\right)^{-},\nabla\cdot\left\{\boldsymbol{\beta}\left(u^{k+1}\right)^{-}\right\}\right)_{T}+\left\langle\boldsymbol{\beta}\cdot\mathbf{n}^{-}\left(u^{k+1}\right)^{-},\left(u^{k+1}\right)^{-}\right\rangle_{\partial T},$$

yields the following identity, after rearranging the terms

$$\left(\left(u^{k+1}\right)^{-},\nabla\cdot\left\{\boldsymbol{\beta}\left(u^{k+1}\right)^{-}\right\}\right)_{T}=\left(\left(u^{k+1}\right)^{-},\frac{\nabla\cdot\boldsymbol{\beta}}{2}\left(u^{k+1}\right)^{-}\right)_{T}$$
$$+\frac{1}{2}\left\langle\boldsymbol{\beta}\cdot\mathbf{n}^{-}\left(u^{k+1}\right)^{-},\left(u^{k+1}\right)^{-}\right\rangle_{\partial T}. \quad (3.6)$$

Using (3.6) in (3.5) we get

$$\left\|\left(u^{k+1}\right)^{-}\right\|^{2}_{\frac{-\nabla\cdot\boldsymbol{\beta}}{2},T}+\left\|\left(u^{k+1}\right)^{-}\right\|^{2}_{|\boldsymbol{\beta}\cdot\mathbf{n}|/2,\partial T}=$$
$$\frac{1}{2}\left\langle\left(\boldsymbol{\beta}\cdot\mathbf{n}^{+}+|\boldsymbol{\beta}\cdot\mathbf{n}|\right)\left(u^{k}\right)^{+},\left(u^{k+1}\right)^{-}\right\rangle_{\partial T}. \quad (3.7)$$

In equation (3.7) all the terms on the left hand side are positive. Since $\left(u^{k}\right)^{+}$ is the "forcing" for the local equation, by taking $\left(u^{k}\right)^{+}=0$ the only solution possible is $\left(u^{k+1}\right)^{-}=0$ and hence the local solver is well-posed. □

Having proved the well-posedness of the local solver we can now proceed to prove the convergence of Algorithm 2 for the transport equation.

Theorem 5. *Assume $-\nabla\cdot\boldsymbol{\beta}\geq\alpha>0$, i.e. (2.5) is well-posed. There exists $J\leq N_T$ such that the iHDG-II algorithm for the homogeneous transport equation converges to the HDG solution in J iterations.*

Proof. Using (3.7) from Lemma 1 and $\boldsymbol{\beta}\cdot\mathbf{n}^{+}>0$ on ∂T^{in}, $\boldsymbol{\beta}\cdot\mathbf{n}^{+}\leq 0$ on ∂T^{out} we can write

$$\left\|\left(u^{k+1}\right)^{-}\right\|^{2}_{\frac{-\nabla\cdot\boldsymbol{\beta}}{2},T}+\left\|\left(u^{k+1}\right)^{-}\right\|^{2}_{|\boldsymbol{\beta}\cdot\mathbf{n}|/2,\partial T}=\left\langle|\boldsymbol{\beta}\cdot\mathbf{n}|u^{k}_{\text{ext}},\left(u^{k+1}\right)^{-}\right\rangle_{\partial T^{\text{in}}}. \quad (3.8)$$

where u^k_{ext} is either the physical boundary condition or the solution of the neighboring element that shares the same inflow boundary ∂T^{in}.

Consider the set $\mathcal{T}^1$ of all elements T such that ∂T^{in} is a subset of the physical inflow boundary $\partial\Omega^{\text{in}}$ on which we have $u^k_{\text{ext}} = 0$ for all $k \in \mathbb{N}$. We obtain from (3.8) that

$$\left\| \left(u^{k+1}\right)^- \right\|^2_{\frac{-\nabla\cdot\boldsymbol{\beta}}{2},T} + \left\| \left(u^{k+1}\right)^- \right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|/2,\partial T} = 0, \tag{3.9}$$

which implies $u^1 = 0$ on $T \in \mathcal{K}^1$, i.e. our iterative solver is exact on $T \in \mathcal{T}^1$ at the first iteration.

Next, let us define $\Omega^1_h := \Omega_h$ and

$$\Omega^2_h = \Omega^1_h \backslash \mathcal{T}^1.$$

Consider the set $\mathcal{T}^2$ of all T in Ω^2_h such that ∂T^{in} is either (possibly partially) a subset of the physical inflow boundary $\partial\Omega^{\text{in}}$ or (possibly partially) a subset of the outflow boundary of elements in $\mathcal{T}^1$. This implies, on $\partial T^{\text{in}} \in \mathcal{T}^2$, $u^k_{\text{ext}} = 0$ for all $k \in \mathbb{N} \setminus \{1\}$. Thus, $\forall T \in \mathcal{T}^2$, we have

$$\left\| \left(u^{k}\right)^- \right\|^2_{\frac{-\nabla\cdot\boldsymbol{\beta}}{2},T} + \left\| \left(u^{k}\right)^- \right\|^2_{|\boldsymbol{\beta}\cdot\mathbf{n}|/2,\partial T} = 0, \quad \forall k \in \mathbb{N} \setminus \{1\}, \tag{3.10}$$

which implies $u^2 = 0$ in $T \in \mathcal{T}^2$, i.e. our iterative solver is exact on $T \in \mathcal{T}^2$ at the second iteration.

Repeating the same argument, we can construct subsets $\mathcal{T}^j \subset \Omega_h$, on which the iterative solution on $T \in \mathcal{T}^j$ is the exact HDG solution at the j-th iteration. Since the number of elements N_T is finite, there exists $J \leq N_T$ such that $\Omega_h = \cup_{j=1}^J \mathcal{T}^j$. It follows that the iHDG-II algorithm provides exact HDG solution on Ω_h after J iterations. □

Remark 5. *Compared to iHDG-I [108], which requires an infinite number of iterations to converge, iHDG-II needs finite number of iterations for convergence. The key to the improvement is the stronger coupling between* $\hat{\mathbf{u}}$ *and* $\mathbf{u}$ *by using* $\left(\mathbf{u}^{k+1}\right)^{-}$ *in* (3.2) *instead of* $\left(\mathbf{u}^{k}\right)^{-}$*. The proof of Theorem 5 also shows that iHDG-II automatically marches the flow, i.e., each iteration yields the HDG solution exactly for a group of elements. Moreover, the marching process is automatic (i.e., does not require an ordering of elements) and adapts to the velocity field* $\boldsymbol{\beta}$ *under consideration.*

3.2.2 Time-dependent transport equation

In this section we first comment on a space-time formulation of the iHDG methods and compare it with the parareal methods studied in [59] for the time-dependent scalar transport equation. Then we consider the semi-discrete version of iHDG combined with traditional time integration schemes and compare it with the fully implicit and explicit DG/HDG schemes.

3.2.2.1 Comparison of space-time iHDG and parareal methods for the scalar transport equation

Space-time finite element methods have been studied extensively for the past several years both in the context of continuous and discontinuous Galerkin methods [84, 4, 118, 89, 52] and HDG methods [125]. Parareal methods, on the other hand, were first introduced in [99] and various modifications have been proposed and studied (see [65, 56, 102, 105, 63] and references therein).

In the scope of our work, we compare our methods with the parareal scheme proposed in [59] for the scalar advection equation. Let us start with the following ordinary differential equation

$$\frac{du}{dt} = f \quad \text{in } (0,T), \quad u(0) = g, \tag{3.11}$$

for some positive constant $T > 0$.

Corollary 3. *Suppose we discretize the temporal derivative in* (3.11) *using the iHDG-II method with the upwind flux and the elements T_j are ordered such that T_j is on the left of T_{j+1}. At iteration k, $u^k\big|_{T_j}$ converges to the HDG solution $u|_{T_j}$ for $j \leq k$.*

Proof. Since (3.11) can be considered as 1D transport equation (2.5) with velocity $\boldsymbol{\beta} = 1$, the proof follows directly from Theorem 5 and induction. □

Note that the iHDG scheme can be considered as a parareal algorithm in which the fine propagator is taken to be the local solver (3.1) and the coarse propagator corresponds to the conservation condition (3.2). However, unlike existing parareal algorithms, the coarse propagator of iHDG-parareal is dependent on the fine propagator. Moreover, Corollary 3 says that after k iterations the iHDG-parareal solution converges up to element k, a feature common to the parareal algorithm studied in [59]. For time dependent hyperbolic PDEs, the space-time iHDG method again can be understood as parareal approach, and in this case, a layer of space-time elements converges after each iHDG-parareal iteration (see Remark 5). See Figure 3.1 and Table 3.1 of section 3.4 for a demonstration in 2D where either x or y is considered as "time". It should be pointed out that the specific parareal method in [59] exactly traces the characteristics, and hence may take less iterations to converge than the iHDG-parareal method, but this is only true if the forward Euler discretization in time, upwind finite difference in space, and $CFL = 1$ are used with constant advection velocity.

3.2.3 iHDG as a locally implicit method

In this section we discuss the relationship between iHDG and implicit/explicit HDG methods. For the simplicity of the exposition, we consider time-dependent

scalar transport equation given by:

$$\frac{\partial u^e}{\partial t} + \boldsymbol{\beta} \cdot \nabla u^e = f. \tag{3.12}$$

We first review the implicit/explicit HDG schemes for (3.12), and then compare them with iHDG-II. The implicit Euler HDG scheme for (3.12) reads

$$\begin{aligned}\left(\frac{u^{m+1}}{\Delta t}, v\right)_T - \left(u^{m+1}, \nabla \cdot (\boldsymbol{\beta} v)\right)_T + \left\langle \boldsymbol{\beta} \cdot \mathbf{n} u^{m+1} + |\boldsymbol{\beta} \cdot \mathbf{n}|(u^{m+1} - \hat{u}^{m+1}), v \right\rangle_{\partial T} \\ = \left(f^{m+1} + \frac{u^m}{\Delta t}, v\right)_T, \\ \left\langle [\![|\boldsymbol{\beta} \cdot \mathbf{n}| \hat{u}^{m+1}]\!], \boldsymbol{\mu} \right\rangle_{\partial T} = \left\langle [\![|\boldsymbol{\beta} \cdot \mathbf{n}| u^{m+1}]\!] + [\![\boldsymbol{\beta} \cdot \mathbf{n} u^{m+1}]\!], \boldsymbol{\mu} \right\rangle_{\partial T}. \end{aligned} \tag{3.13}$$

Here, u^{m+1} and $\hat{u}^{m+1}$ stands for the volume and the trace unknowns at the current time step, whereas u^m and $\hat{u}^m$ are the computed solutions from the previous time step. Clearly, u^{m+1} and $\hat{u}^{m+1}$ are coupled and this can be a challenge for large-scale problems.

Next let us consider an explicit HDG with forward Euler discretization in time for (3.12):

$$\begin{aligned}\left(\frac{u^{m+1}}{\Delta t}, v\right)_T = (u^m, \nabla \cdot (\boldsymbol{\beta} v))_T - \langle \boldsymbol{\beta} \cdot \mathbf{n} u^m + |\boldsymbol{\beta} \cdot \mathbf{n}|(u^m - \hat{u}^m), v \rangle_{\partial T} \\ + \left(f^m + \frac{u^m}{\Delta t}, v\right)_T, \\ \langle [\![|\boldsymbol{\beta} \cdot \mathbf{n}| \hat{u}^m]\!], \boldsymbol{\mu} \rangle_{\partial T} = \langle [\![|\boldsymbol{\beta} \cdot \mathbf{n}| u^m]\!] + [\![\boldsymbol{\beta} \cdot \mathbf{n} u^m]\!], \boldsymbol{\mu} \rangle_{\partial T}, \end{aligned}$$

which shows that we can solve for u^{m+1} element-by-element, completely independent of each other. However, since it is an explicit scheme, the CFL restriction for stability can increase the computational cost for problems involving fast time scales and/or fine meshes.

Now applying *one iteration* of the iHDG-II scheme for the implicit HDG formulation (3.13) with u^m as the initial guess yields

$$\left(\frac{u^{m+1}}{\Delta t}, v\right)_T - \left(u^{m+1}, \nabla\cdot(\boldsymbol{\beta} v)\right)_T + \left\langle \boldsymbol{\beta}\cdot\mathbf{n} u^{m+1} + |\boldsymbol{\beta}\cdot\mathbf{n}|(u^{m+1} - \hat{u}^{m,m+1}), v\right\rangle_{\partial T}$$
$$= \left(f^{m+1} + \frac{u^m}{\Delta t}, v\right)_T,$$
$$\left\langle [\![|\boldsymbol{\beta}\cdot\mathbf{n}| \hat{u}^{m,m+1}]\!], \boldsymbol{\mu}\right\rangle_{\partial T} = \left\langle |\boldsymbol{\beta}\cdot\mathbf{n}| \left\{ \left(u^{m+1}\right)^- + \left(u^m\right)^+ \right\}, \boldsymbol{\mu}\right\rangle_{\partial T}$$
$$+ \left\langle \boldsymbol{\beta}\cdot\mathbf{n}^- \left(u^{m+1}\right)^- + \boldsymbol{\beta}\cdot\mathbf{n}^+ \left(u^m\right)^+, \boldsymbol{\mu}\right\rangle_{\partial T}.$$

Compared to the explicit HDG scheme, iHDG-II requires local solves since it is *locally implicit*. As such, its CFL restriction is much less (see Figure 3.2), while still having similar parallel scalability of the explicit method.[2] Indeed, Figure 3.2 shows that the CFL restriction is only indirectly through the increase of the number of iterations; for CFL numbers between 1 and 5, the number of iterations varies mildly. Thus, as a locally implicit method, iHDG-II combines advantages of both explicit (e.g. matrix free and parallel scalability) and implicit (taking reasonably large time stepsize without facing instability) methods. Clearly, on convergence iHDG solution is, up to the stopping tolerance, the same as the fully-implicit solution.

3.2.4 iHDG-II for system of linear hyperbolic PDEs

In this section, similar to chapter 2 as an example for the system of linear hyperbolic PDEs, we consider the linearized oceanic shallow water system (2.25). We study the iHDG-II methods for this equation and compare it with the results in chapter 2.

Applying the iHDG-II algorithm, Algorithm 2, to the homogeneous system

[2] In fact, due to local solves, iHDG-II could provide more efficient communication and computation overlapping.

gives

$$\left(\frac{\phi^{k+1}}{\Delta t},\varphi_1\right)_T-\left(\Phi\boldsymbol{\vartheta}^{k+1},\nabla\varphi_1\right)_T+\left\langle\Phi\boldsymbol{\vartheta}^{k+1}\cdot\mathbf{n}+\sqrt{\Phi}\left(\phi^{k+1}-\hat{\phi}^{k,k+1}\right),\varphi_1\right\rangle_{\partial T}$$
$$=0, \tag{3.14a}$$
$$\left(\frac{\Phi u^{k+1}}{\Delta t},\varphi_2\right)_T-\left(\Phi\phi^{k+1},\frac{\partial\varphi_2}{\partial x}\right)_T+\left\langle\Phi\hat{\phi}^{k,k+1}\mathbf{n}_1,\varphi_2\right\rangle_{\partial T}$$
$$=\left(f\Phi v^{k+1}-\gamma\Phi u^{k+1},\varphi_2\right)_T, \tag{3.14b}$$
$$\left(\frac{\Phi v^{k+1}}{\Delta t},\varphi_3\right)_T-\left(\Phi\phi^{k+1},\frac{\partial\varphi_3}{\partial y}\right)_T+\left\langle\Phi\hat{\phi}^{k,k+1}\mathbf{n}_2,\varphi_3\right\rangle_{\partial T}$$
$$=\left(-f\Phi u^{k+1}-\gamma\Phi v^{k+1},\varphi_3\right)_T, \tag{3.14c}$$

where φ_1,φ_2 and φ_3 are the test functions, and

$$\hat{\phi}^{k,k+1}=\frac{1}{2}\left\{\left(\phi^{k+1}\right)^-+\left(\phi^k\right)^+\right\}+\frac{\sqrt{\Phi}}{2}\left\{\left(\boldsymbol{\vartheta}^{k+1}\right)^-\cdot\mathbf{n}^-+\left(\boldsymbol{\vartheta}^k\right)^+\cdot\mathbf{n}^+\right\}. \tag{3.15}$$

Lemma 2. *The local solver* (3.14) *of the iHDG-II algorithm for the linearized shallow water equation is well-posed.*

Proof. Since $\left\{\left(\phi^k\right)^+,\Phi\left(\boldsymbol{\vartheta}^k\right)^+\right\}$ is a "forcing" to the local solver it is sufficient to set them to $\{0,\mathbf{0}\}$ and show that the only solution possible is $\left\{\left(\phi^k\right)^-,\Phi\left(\boldsymbol{\vartheta}^k\right)^-\right\}=\{0,\mathbf{0}\}$. Choosing the test functions $\varphi_1=\phi^{k+1}$, $\varphi_2=u^{k+1}$ and $\varphi_3=v^{k+1}$ in (3.14), integrating the second term in (3.14a) by parts, and then summing equations in (3.14) altogether, we obtain

$$\frac{1}{\Delta t}\left(\phi^{k+1},\phi^{k+1}\right)_T+\frac{\Phi}{\Delta t}\left(\boldsymbol{\vartheta}^{k+1},\boldsymbol{\vartheta}^{k+1}\right)_T+\sqrt{\Phi}\left\langle\phi^{k+1},\phi^{k+1}\right\rangle_{\partial T}+\gamma\Phi\left(\boldsymbol{\vartheta}^{k+1},\boldsymbol{\vartheta}^{k+1}\right)_T$$
$$-\sqrt{\Phi}\left\langle\hat{\phi}^{k,k+1},\phi^{k+1}\right\rangle_{\partial T}+\Phi\left\langle\hat{\phi}^{k,k+1},\mathbf{n}\cdot\boldsymbol{\vartheta}^{k+1}\right\rangle_{\partial T}=0. \tag{3.16}$$

Summing (3.16) over all elements yields

$$\sum_T\frac{1}{\Delta t}\left(\phi^{k+1},\phi^{k+1}\right)_T+\frac{\Phi}{\Delta t}\left(\boldsymbol{\vartheta}^{k+1},\boldsymbol{\vartheta}^{k+1}\right)_T+\gamma\Phi\left(\boldsymbol{\vartheta}^{k+1},\boldsymbol{\vartheta}^{k+1}\right)_T$$
$$+\sqrt{\Phi}\left\langle\phi^{k+1},\phi^{k+1}\right\rangle_{\partial T}-\sqrt{\Phi}\left\langle\hat{\phi}^{k,k+1},\phi^{k+1}\right\rangle_{\partial T}+\Phi\left\langle\hat{\phi}^{k,k+1},\mathbf{n}\cdot\boldsymbol{\vartheta}^{k+1}\right\rangle_{\partial T}=0. \tag{3.17}$$

Substituting (3.15) in the above equation and canceling some terms we get,

$$\sum_T \frac{1}{\Delta t}\left\|\left(\phi^{k+1}\right)^-\right\|_T^2+\left(\gamma+\frac{1}{\Delta t}\right)\left\|\left(\boldsymbol{\vartheta}^{k+1}\right)^-\right\|_{\Phi,T}^2+\frac{\sqrt{\Phi}}{2}\left\|\left(\phi^{k+1}\right)^-\right\|_{\partial T}^2$$
$$+\frac{\sqrt{\Phi}}{2}\left\|\left(\boldsymbol{\vartheta}^{k+1}\cdot\mathbf{n}\right)^-\right\|_{\Phi,\partial T}^2=\sum_{\partial T}\frac{\sqrt{\Phi}}{2}\left\langle\left\{\left(\phi^k\right)^+ +\sqrt{\Phi}\left(\boldsymbol{\vartheta}^k\cdot\mathbf{n}\right)^+\right\},\left(\phi^{k+1}\right)^-\right\rangle_{\partial T}$$
$$-\frac{\Phi}{2}\left\langle\left\{\left(\phi^k\right)^+ +\sqrt{\Phi}\left(\boldsymbol{\vartheta}^k\cdot\mathbf{n}\right)^+\right\},\left(\boldsymbol{\vartheta}^{k+1}\cdot\mathbf{n}\right)^-\right\rangle_{\partial T}. \tag{3.18}$$

Since $\Phi>0$, all the terms on the left hand side are positive. When we set $\left\{\left(\phi^k\right)^+,\Phi\left(\boldsymbol{\vartheta}^k\right)^+\right\}=\{0,\mathbf{0}\}$, i.e. the data from neighboring elements, the only solution possible is $\left\{\left(\phi^{k+1}\right)^-,\Phi\left(\boldsymbol{\vartheta}^{k+1}\right)^-\right\}=\{0,\mathbf{0}\}$ and hence the method is well-posed. □

Next, our goal is to show that $\left(\phi^{k+1},\Phi\boldsymbol{\vartheta}^{k+1}\right)$ converges to zero. To that end, let us define

$$\mathcal{C}:=\frac{\mathcal{A}}{\mathcal{B}},\quad \mathcal{A}:=\frac{\max\{1,\Phi\}+\sqrt{\Phi}}{4\varepsilon},\quad \mathcal{G}:=\frac{\varepsilon\left(\max\{1,\Phi\}+\sqrt{\Phi}\right)}{4} \tag{3.19}$$

and

$$\mathcal{B}_1:=\left(\frac{ch}{\Delta t(p+1)(p+2)}+\frac{2\sqrt{\Phi}-(\Phi+\sqrt{\Phi})\varepsilon}{4}\right)$$
$$\mathcal{B}_2:=\left(\left(\gamma+\frac{1}{\Delta t}\right)\frac{ch}{(p+1)(p+2)}+\frac{2\sqrt{\Phi}-(1+\sqrt{\Phi})\varepsilon}{4}\right),\mathcal{B}:=\min\{\mathcal{B}_1,\mathcal{B}_2\}.$$

where $0<c\leq 1$, $\varepsilon>0$ are constants. We also need the following norms:

$$\left\|\left(\phi^k,\boldsymbol{\vartheta}^k\right)\right\|_{\Omega_h}^2:=\left\|\phi^k\right\|_{\Omega_h}^2+\left\|\boldsymbol{\vartheta}^k\right\|_{\Phi,\Omega_h}^2,$$
$$\left\|\left(\phi^k,\boldsymbol{\vartheta}^k\cdot\mathbf{n}\right)\right\|_{\mathcal{E}_h}^2:=\left\|\phi^k\right\|_{\mathcal{E}_h}^2+\left\|\boldsymbol{\vartheta}^k\cdot\mathbf{n}\right\|_{\Phi,\mathcal{E}_h}^2.$$

Theorem 6. *Assume that the meshsize h, the time step Δt and the solution order p are chosen such that $\mathcal{B}>0$ and $\mathcal{C}<1$, then the approximate solution at the kth*

iteration $\left(\phi^k, \boldsymbol{\vartheta}^k\right)$ *converges to zero in the following sense*

$$\left\|\left(\phi^k, \boldsymbol{\vartheta}^k \cdot \mathbf{n}\right)\right\|_{\mathcal{E}_h}^2 \leq \mathcal{C}^k \left\|\left(\phi^0, \boldsymbol{\vartheta}^0 \cdot \mathbf{n}\right)\right\|_{\mathcal{E}_h}^2,$$
$$\left\|\left(\phi^k, \boldsymbol{\vartheta}^k\right)\right\|_{\Omega_h}^2 \leq \Delta t \left(\mathcal{A} + \mathcal{G}\mathcal{C}\right) \mathcal{C}^{k-1} \left\|\left(\phi^0, \boldsymbol{\vartheta}^0 \cdot \mathbf{n}\right)\right\|_{\mathcal{E}_h}^2,$$

where $\mathcal{C}$, $\mathcal{A}$ *and* $\mathcal{G}$ *are defined in* (3.19).

Proof. Using Cauchy-Schwarz and Young's inequalities for the terms on the right hand side of (3.18) and simplifying we have

$$\begin{aligned}
&\sum_T \frac{1}{\Delta t} \left\|\left(\phi^{k+1}\right)^-\right\|_T^2 + \left(\gamma + \frac{1}{\Delta t}\right) \left\|\left(\boldsymbol{\vartheta}^{k+1}\right)^-\right\|_{\Phi,T}^2 + \frac{\sqrt{\Phi}}{2} \left\|\left(\phi^{k+1}\right)^-\right\|_{\partial T}^2 \\
&+ \frac{\sqrt{\Phi}}{2} \left\|\left(\boldsymbol{\vartheta}^{k+1} \cdot \mathbf{n}\right)^-\right\|_{\Phi,\partial T}^2 \leq \sum_{\partial T} \frac{\Phi + \sqrt{\Phi}}{4\varepsilon} \left\|\left(\phi^k\right)^+\right\|_{\partial T}^2 \\
&+ \frac{1 + \sqrt{\Phi}}{4\varepsilon} \left\|\left(\boldsymbol{\vartheta}^k \cdot \mathbf{n}\right)^+\right\|_{\Phi,\partial T}^2 + \frac{\varepsilon(\Phi + \sqrt{\Phi})}{4} \left\|\left(\phi^{k+1}\right)^-\right\|_{\partial T}^2 \\
&+ \frac{\varepsilon(1 + \sqrt{\Phi})}{4} \left\|\left(\boldsymbol{\vartheta}^{k+1} \cdot \mathbf{n}\right)^-\right\|_{\Phi,\partial T}^2. \qquad (3.20)
\end{aligned}$$

Inequality (2.30), together with (3.20), implies

$$\begin{aligned}
&\sum_{\partial T} \left[\left(\frac{ch}{\Delta t(p+1)(p+2)} + \frac{2\sqrt{\Phi} - (\Phi + \sqrt{\Phi})\varepsilon}{4}\right) \left\|\left(\phi^{k+1}\right)^-\right\|_{\partial T}^2 \right. \\
&\left. + \left(\left(\gamma + \frac{1}{\Delta t}\right) \frac{ch}{(p+1)(p+2)} + \frac{2\sqrt{\Phi} - (1 + \sqrt{\Phi})\varepsilon}{4}\right) \left\|\left(\boldsymbol{\vartheta}^{k+1} \cdot \mathbf{n}\right)^-\right\|_{\Phi,\partial T}^2\right] \\
\leq\ &\sum_{\partial T} \left[\frac{\Phi + \sqrt{\Phi}}{4\varepsilon} \left\|\left(\phi^k\right)^+\right\|_{\partial T}^2 + \frac{1 + \sqrt{\Phi}}{4\varepsilon} \left\|\left(\boldsymbol{\vartheta}^k \cdot \mathbf{n}\right)^+\right\|_{\Phi,\partial T}^2\right], \qquad (3.21)
\end{aligned}$$

which then implies

$$\left\|\left(\phi^{k+1}, \boldsymbol{\vartheta}^{k+1} \cdot \mathbf{n}\right)\right\|_{\mathcal{E}_h}^2 \leq \mathcal{C} \left\|\left(\phi^k, \boldsymbol{\vartheta}^k \cdot \mathbf{n}\right)\right\|_{\mathcal{E}_h}^2,$$

where the constant $\mathcal{C}$ is computed as in (3.19). Therefore

$$\left\|\left(\phi^{k+1}, \boldsymbol{\vartheta}^{k+1} \cdot \mathbf{n}\right)\right\|_{\mathcal{E}_h}^2 \leq \mathcal{C}^{k+1} \left\|\left(\phi^0, \boldsymbol{\vartheta}^0 \cdot \mathbf{n}\right)\right\|_{\mathcal{E}_h}^2. \qquad (3.22)$$

On the other hand, inequalities (3.20) and (3.22) imply

$$\left\| \left(\phi^{k+1}, \boldsymbol{\vartheta}^{k+1} \right) \right\|_{\Omega_h}^2 \leq \Delta t \left(\mathcal{A} + \mathcal{GC} \right) \mathcal{C}^k \left\| \left(\phi^0, \boldsymbol{\vartheta}^0 \cdot \mathbf{n} \right) \right\|_{\mathcal{E}_h}^2$$

and this ends the proof. □

We now derive an explicit relation between the number of iterations k, the meshsize h, the solution order p, the time step Δt and the mean flow geopotential height Φ. First, we need to find an ε which makes $\mathcal{C} < 1$. From (3.19) we obtain the following inequality for ε

$$\frac{\frac{\max\{1,\Phi\}+\sqrt{\Phi}}{4\varepsilon}}{\left(\frac{ch}{\Delta t(p+1)(p+2)} + \frac{2\sqrt{\Phi}-(\max\{1,\Phi\}+\sqrt{\Phi})\varepsilon}{4} \right)} < 1. \tag{3.23}$$

A sufficient condition for the denominator to be positive and existence of a real $\varepsilon > 0$ to the above inequality (3.23) is

$$\frac{\frac{ch}{\Delta t(p+1)(p+2)}}{\max\{1,\Phi\} + \sqrt{\Phi}} > \frac{1}{2}. \tag{3.24}$$

This allows us to find an $\varepsilon > 0$ that satisfies the inequality (3.23) for all Φ. In particular, we can pick

$$\varepsilon = \frac{\frac{2ch}{\Delta t(p+1)(p+2)} + \sqrt{\Phi}}{\max\{1,\Phi\} + \sqrt{\Phi}}. \tag{3.25}$$

Using this value of ε in definition (3.19) we get

$$\mathcal{C} = \left(\frac{\frac{\max\{1,\Phi\}+\sqrt{\Phi}}{\sqrt{\Phi}}}{1 + \frac{2ch}{\sqrt{\Phi}\Delta t(p+1)(p+2)}} \right)^2$$

and since the numerator is always greater than 1, the necessary and sufficient condition for the convergence of the algorithm is given by

$$\frac{1}{\left(1 + \frac{2ch}{\sqrt{\Phi}\Delta t(p+1)(p+2)} \right)^{2k}} \overset{k\to\infty}{\longrightarrow} 0.$$

Using binomial theorem and neglecting higher order terms we get

$$k = \mathcal{O}\left(\frac{\Delta t(p+1)(p+2)\sqrt{\Phi}}{4ch}\right). \tag{3.26}$$

Note that if we choose Δt similar to explicit method, i.e. $\Delta t = \mathcal{O}\left(\frac{h}{p^2\sqrt{\Phi}}\right)$ [143], $k = \mathcal{O}(1)$ independent of h and p. With this result in hand we are now in a better position to understand the stability of iHDG-I and iHDG-II algorithms for the linearized shallow water system. For unconditional stability of the iterative algorithms under consideration, we need $\mathcal{B} > 0$ in (3.19) independent of h, p and Δt. There are two terms in $\mathcal{B}$: $\mathcal{B}_1$ coming from ϕ and $\mathcal{B}_2$ from $\boldsymbol{\vartheta}$ or $\boldsymbol{\vartheta}\cdot\mathbf{n}$. We can write $\mathcal{B}$ in Theorem 3 for iHDG-I also as[3]

$$\mathcal{B}_1 := \left(\frac{ch}{\Delta t(p+1)(p+2)} + \frac{2\sqrt{\Phi} - (\Phi+\sqrt{\Phi})\varepsilon}{2}\right) \tag{3.27}$$

$$\mathcal{B}_2 := \left(\left(\gamma + \frac{1}{\Delta t}\right)\frac{ch}{(p+1)(p+2)} - \frac{(1+\sqrt{\Phi})\varepsilon}{2}\right), \mathcal{B} := \min\left\{\mathcal{B}_1, \mathcal{B}_2\right\}. \tag{3.28}$$

Note that for both iHDG-I and iHDG-II algorithms we have the stability in ϕ independent of h, p and Δt, since we can choose ε sufficiently small independent of h, p and Δt and make $\mathcal{B}_1 > 0$ in (3.27) and (3.19). However, from (3.28) we have to choose ε as a function of $(h, p, \Delta t)$ in order to have $\mathcal{B}_2 > 0$, and hence iHDG-I lacks the mesh independent stability in the term associated with $\boldsymbol{\vartheta}$. This explains the instability observed in iHDG-I for fine meshes and/or large time steps. Since $\mathcal{B}_2$ in (3.19) can be made positive with a sufficiently small ε, independent of h, p and Δt, iHDG-II is always stable: a significant advantage over iHDG-I.

[3]This can be obtained by using Young's inequality with ε in the proof of Theorem 3.

3.3 iHDG-II for linear convection-diffusion PDEs

3.3.1 First order form

In this section we apply the iHDG-II algorithm, Algorithm 2, to the first order form of the convection-diffusion equation (2.34). Similar to the previous sections, it is sufficient to consider the homogeneous problem. Applying the iHDG-II algorithm, Algorithm 2 with the upwind HDG flux (2.37) we have the following iterative scheme

$$\kappa^{-1}\left(\boldsymbol{\sigma}^{k+1},\boldsymbol{\tau}\right)_T-\left(u^{k+1},\nabla\cdot\boldsymbol{\tau}\right)_T+\left\langle\hat{u}^{k,k+1},\boldsymbol{\tau}\cdot\mathbf{n}\right\rangle_{\partial T}=0, \tag{3.29a}$$

$$-\left(\boldsymbol{\sigma}^{k+1},\nabla v\right)_T-\left(u^{k+1},\nabla\cdot(\boldsymbol{\beta}v)-\nu v\right)_T+$$
$$\left\langle\boldsymbol{\beta}\cdot\mathbf{n}u^{k+1}+\boldsymbol{\sigma}^{k+1}\cdot\mathbf{n}+\tau(u^{k+1}-\hat{u}^{k,k+1}),v\right\rangle_{\partial T}=0, \tag{3.29b}$$

where

$$\hat{u}^{k,k+1}=\frac{\left\{\left(\boldsymbol{\sigma}^{k+1}\cdot\mathbf{n}\right)^-+\left(\boldsymbol{\sigma}^{k}\cdot\mathbf{n}\right)^+\right\}+\left\{\boldsymbol{\beta}\cdot\mathbf{n}^-\left(u^{k+1}\right)^-+\boldsymbol{\beta}\cdot\mathbf{n}^+\left(u^{k}\right)^+\right\}}{\alpha}$$
$$+\frac{\left\{\tau^-\left(u^{k+1}\right)^-+\tau^+\left(u^{k}\right)^+\right\}}{\alpha}. \tag{3.30}$$

Lemma 3. *The local solver* (3.29) *of the iHDG-II algorithm for the convection-diffusion equation is well-posed.*

Proof. The proof is similar to the one for the shallow water equation and hence is given in the Appendix A. □

Now, we are in a position to prove the convergence of the algorithm. For ε,

$h > 0$ and $0 < c \leq 1$ given, define

$$\mathcal{C}_1 := \frac{(\|\boldsymbol{\beta}\cdot\mathbf{n}\|_{L^\infty(\partial T)} + \bar{\tau})(\bar{\tau}+1)}{2\varepsilon\alpha_*},\ \mathcal{C}_2 := \frac{(\bar{\tau}+1)}{2\varepsilon\alpha_*}, \tag{3.31}$$

$$\mathcal{C}_3 := \frac{\varepsilon\bar{\tau}(1+\bar{\tau}+\|\boldsymbol{\beta}\cdot\mathbf{n}\|_{L^\infty(\partial T)})}{2\alpha_*},\ \mathcal{C}_4 := \frac{\varepsilon(1+\bar{\tau}+\|\boldsymbol{\beta}\cdot\mathbf{n}\|_{L^\infty(\partial T)})}{2\alpha_*}, \tag{3.32}$$

$$\mathcal{D} := \frac{\mathcal{A}}{\mathcal{B}},\ \mathcal{A} = \max\{\mathcal{C}_1,\mathcal{C}_2\},\ \mathcal{E} := \frac{\max\{\mathcal{C}_3,\mathcal{C}_4\}}{\min\{\kappa^{-1},\lambda\}},\ \mathcal{F} := \frac{\mathcal{A}}{\min\{\kappa^{-1},\lambda\}}, \tag{3.33}$$

$$\mathcal{B}_1 := \frac{2ch\kappa^{-1}}{d(p+1)(p+2)} + \frac{1}{2\bar{\alpha}} - \mathcal{C}_4, \mathcal{B}_2 := \frac{2ch\lambda}{d(p+1)(p+2)} + \frac{1}{\bar{\alpha}} - \mathcal{C}_3, \tag{3.34}$$

$$\mathcal{B} := \min\{\mathcal{B}_1,\mathcal{B}_2\}, \tag{3.35}$$

where $\bar{\tau} := \|\tau\|_{L^\infty(\partial\Omega_h)}$, $\bar{\alpha} := \|\alpha\|_{L^\infty(\partial\Omega_h)}$, and $\alpha_* := \inf_{\partial T\in\partial\Omega_h}\alpha$. As in the previous section we need the following norms

$$\left\|\left(\boldsymbol{\sigma}^k,u^k\right)\right\|_{\Omega_h}^2 := \left\|\boldsymbol{\sigma}^k\right\|_{\Omega_h}^2 + \left\|u^k\right\|_{\Omega_h}^2, \quad \left\|\left(\boldsymbol{\sigma}^k\cdot\mathbf{n},u^k\right)\right\|_{\mathcal{E}_h}^2 := \left\|\boldsymbol{\sigma}^k\cdot\mathbf{n}\right\|_{\mathcal{E}_h}^2 + \left\|u^k\right\|_{\mathcal{E}_h}^2.$$

Theorem 7. *Suppose that the meshsize h and the solution order p are chosen such that $\mathcal{B} > 0$ and $\mathcal{D} < 1$, the algorithm* (3.29a)-(3.30) *converges in the following sense*

$$\left\|\left(\boldsymbol{\sigma}^k\cdot\mathbf{n},u^k\right)\right\|_{\mathcal{E}_h}^2 \leq \mathcal{D}^k\left\|\left(\boldsymbol{\sigma}^0\cdot\mathbf{n},u^0\right)\right\|_{\mathcal{E}_h}^2,$$
$$\left\|\left(\boldsymbol{\sigma}^k,u^k\right)\right\|_{\Omega_h}^2 \leq (\mathcal{E}\mathcal{D}+\mathcal{F})\mathcal{D}^{k-1}\left\|\left(\boldsymbol{\sigma}^0\cdot\mathbf{n},u^0\right)\right\|_{\mathcal{E}_h}^2,$$

where $\mathcal{D},\mathcal{E}$ and $\mathcal{F}$ are defined in (3.33).

Proof. The proof is similar to the one for the shallow water equation and hence is given in the Appendix B. □

Similar to the discussion in section 3.2.4, one can show that

$$k = \mathcal{O}\left(\frac{d(p+1)(p+2)}{8\bar{\alpha}ch\min\{\kappa^{-1},\lambda\}}\right). \tag{3.36}$$

For time-dependent convection-diffusion equation, we discretize the spatial differential operators using HDG. For the temporal derivative, we use implicit time

stepping methods, again with either backward Euler or Crank-Nicolson method for simplicity. The analysis in this case is almost identical to the one for steady state equation except that we now have an additional L^2-term $\left(u^{k+1}, v\right)_T / \Delta t$ in the local equation (3.29b). This improves the convergence of the iHDG-II method. Indeed, the convergence analysis is the same except we now have $\lambda + 1/\Delta t$ in place of λ. In particular we have the following estimation

$$k = \mathcal{O}\left(\frac{d(p+1)(p+2)}{8\bar{\alpha}ch\min\left\{\kappa^{-1}, (\lambda + 1/\Delta t)\right\}}\right).$$

Remark 6. *Similar to the shallow water system if we choose* $\Delta t = \mathcal{O}\left(\frac{h}{p^2}\right)$ *then the number of iterations is independent of* h *and* p*. This is more efficient than the iterative hybridizable IPDG method for the parabolic equation in [62], which requires* $\Delta t = \mathcal{O}(\frac{h^2}{p^4})$ *in order to achieve constant iterations. The reason is perhaps due the fact that hybridizable IPDG is posed directly on the second order form whereas HDG uses the first order form. While iHDG-I has mesh independent stability for only* u *(see Theorem 4), iHDG-II does for both* u *and* $\boldsymbol{\sigma}$*; an important improvement.*

3.4 Numerical results

In this section various numerical results verifying the theoretical results are provided for the transport equation, the shallow water equation, and the convection-diffusion equation in both two- and three-dimensions. Similar to the previous chapter here also we use the notation N_{el} to denote the total number of elements which is same as N_T.

3.4.1 Transport equation

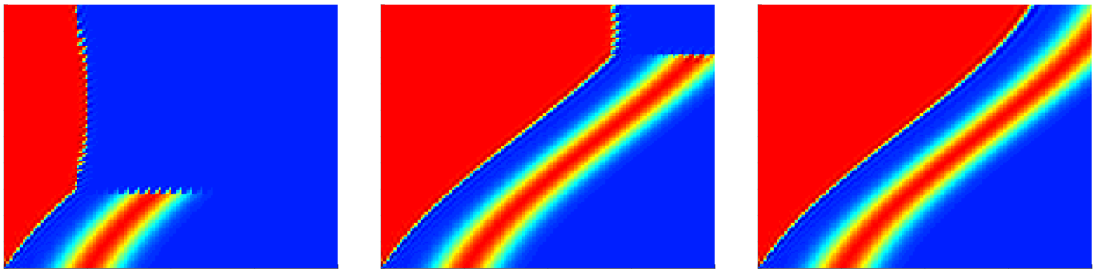
We consider the same 2D and 3D test cases in section 2.6.1 of chapter 2. The mesh consists of structured quadrilateral (2D)/hexahedral (3D) elements. Through-

out the numerical section unless otherwise stated explicitly, just as in chapter 2 we use the stopping criterion (2.50) if the exact solution is available, and (2.49) if the exact solution is not available.

From Theorem 5, the theoretical number of iterations is approximately $d \times (N_{el})^{1/d}$ (where d is the dimension). It can be seen from the fourth and fifth columns of Table 3.1 that the numerical results agree well with the theoretical prediction. We can also see that the number of iterations is independent of solution order, which is consistent with the theoretical result Theorem 5. Figure 3.1 shows the solution converging from the inflow boundary to the outflow boundary in a layer-by-layer manner. Again, the process is automatic, i.e., no prior element ordering or information about the advection velocity is required.

Now, we study the parallel performance of the iHDG algorithm. For this purpose we have implemented the iHDG algorithm on top of *mangll* [157, 29, 28] which is a high-order continuous/discontinuous finite element library that supports large scale parallel simulations using MPI. The simulations are conducted on Stampede 1 at the Texas Advanced Computing Center (TACC). Stampede 1 is a 10 petaflop supercomputer consisting of 6400 Sandy Bridge nodes. Each node consists of two 8-core Xeon E5-2680 2.7GHz processors and one 61-core Xeon Phi SE10P KNC MIC 1.1GHz coprocessor. It has 32GB main memory per node ($8 \times$ 4GB DDR3-1600 MHz) and the coprocessor has additional 8GB GDDR5 memory. The interconnect is a 56GB/s Mellanox FDR InfiniBand network in a 2-level fat-tree topology. To carry out the computations, we have used only the main processors and not the coprocessors.

Table 3.2 shows strong scaling results for the 3D transport problem. The parallel efficiency is over 90% for all the cases except for the case where we use 16,384



(a) u^k at $k = 16$ (b) u^k at $k = 48$ (c) u^k at $k = 64$

Figure 3.1: Evolution of the iterative solution for the 2D transport equation using the iHDG-II algorithm.

N_{el}(2D)	N_{el}(3D)	p	2D solution	3D solution
4x4	2x2x2	1	9	6
8x8	4x4x4	1	17	12
16x16	8x8x8	1	33	23
32x32	16x16x16	1	65	47
4x4	2x2x2	2	9	6
8x8	4x4x4	2	17	12
16x16	8x8x8	2	33	23
32x32	16x16x16	2	65	47
4x4	2x2x2	3	9	7
8x8	4x4x4	3	17	12
16x16	8x8x8	3	33	23
32x32	16x16x16	3	65	47
4x4	2x2x2	4	9	6
8x8	4x4x4	4	17	12
16x16	8x8x8	4	33	24
32x32	16x16x16	4	64	48

Table 3.1: The number of iterations taken by the iHDG-II algorithm for the transport equation in 2D and 3D settings.

cores and 16 elements per core whose efficiency is 59%. This is due to the fact that, with 16 elements per core, the communication cost dominates the computation. Table 3.3 shows the weak scaling with 1024 and 128 elements/core. Since the number of iterations increases linearly with the number of elements, we can see a similar increase

in time when we increase the number of elements, and hence cores.

Let us now consider the time dependent 3D transport equation with the following exact solution

$$u^e = e^{-5((x-0.35t)^2+(y-0.35t)^2+(z-0.35t)^2)},$$

where the velocity field is chosen to be $\boldsymbol{\beta} = (0.2, 0.2, 0.2)$. In Figure 3.2 are the numbers of iHDG iterations taken per time step to converge versus the CFL number. As can be seen, for CFL in the range $[1, 5]$ the number of iterations grows mildly. As a result, we get a much better weak scaling results in Table 3.4 in comparison to the steady state case in Table 3.3.

$N_{el} = 262,144$, $p = 4$, dof=32.768 million, Iterations=190			
#cores	time [s]	N_{el}/core	efficiency [%]
64	1758.62	4096	100.0
128	883.88	2048	99.5
256	439.94	1024	99.9
512	228.69	512	96.1
1024	113.87	256	96.5
2048	56.36	128	97.5
4096	29.26	64	91.8
16384	11.38	16	59
$N_{el} = 2,097,152$, $p = 4$, dof=262.144 million, Iterations=382			
#cores	time [s]	N_{el}/core	efficiency [%]
512	3634.89	4096	100.0
1024	1788.78	2048	101.5
2048	932.495	1024	97.3
4096	447.337	512	101.5
8192	232.019	256	97.9
16384	117.985	128	92.9

Table 3.2: Strong scaling results on TACC's Stampede system for the 3D transport problem.

1024 N_{el}/core, $p = 4$			
#cores	time [s]	time ratio	Iterations ratio
4	103.93	1	1
32	217.23	2.1	2
256	439.94	4.2	4
2048	932.49	8.9	8
128 N_{el}/core, $p = 4$			
#cores	time [s]	time ratio	Iterations ratio
4	6.52	1	1
32	13.68	2.1	2
256	27.71	4.2	4
2048	56.37	8.6	8

Table 3.3: Weak scaling results on TACC's Stampede system for the 3D transport problem.

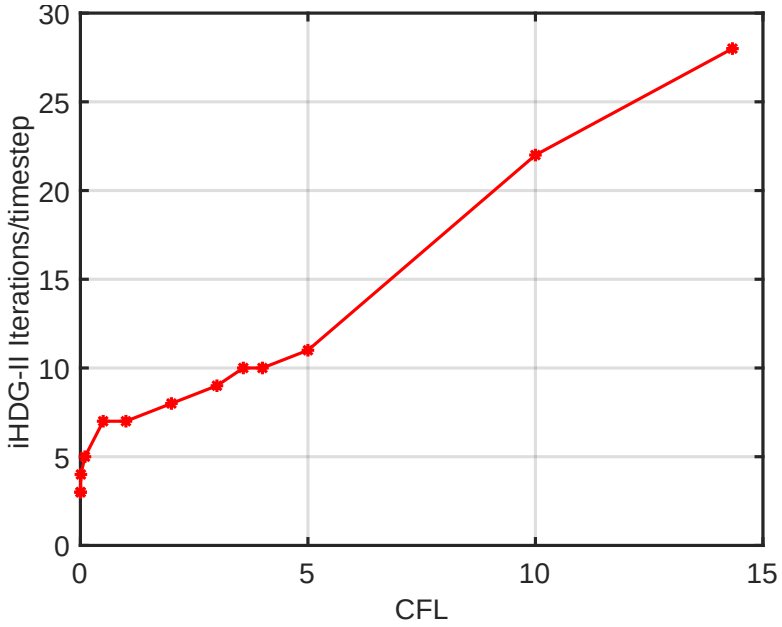


Figure 3.2: CFL versus Iterations for the 3D time dependent transport.

3.4.2 Linearized shallow water system

The goal of this section is to verify the theoretical findings in section 3.2.4. The exact solution and settings of this experiment are same as section 2.6.2 in chapter 2

128 N_{el}/core, $p = 4$, Δt=0.01, $\|\boldsymbol{\beta}\|_{max}$=0.35				
#cores	time/timestep [s]	time ratio	Iterations ratio	CFL
4	1.69	1	1	0.45
32	1.91	1.1	1.1	0.9
256	2.09	1.2	1.1	1.8
2048	2.72	1.6	1.4	3.6
16384	4.68	2.8	2.1	7.2

Table 3.4: Weak scaling results on TACC's Stampede system for the 3D time dependent transport problem.

for iHDG-I. We use Crank-Nicolson method for the temporal discretization and the iHDG-II approach for the spatial discretization. In Table 3.5 we compare the number of iterations taken by iHDG-I and iHDG-II methods for two different time steps $\Delta t = 0.1$ and $\Delta t = 0.01$. Here, "$*$" indicates divergence. As can be seen from the third and fourth columns, the iHDG-I method diverges for finer meshes and/or larger time steps. This is consistent with the findings in section 3.2.4 where the divergence is expected because of the lack of mesh independent stability in the velocity. On the contrary, iHDG-II converges for all cases.

In Table 3.5, we use a series of structured quadrilateral meshes with uniform refinements such that the ratio of successive meshsizes is 1/2. The asymptotic result (3.26), which is valid for $\frac{h}{\Delta t(p+1)(p+2)\sqrt{\Phi}} \ll 1$, predicts that the ratio of the number of iterations required by successive refined meshes is 2, and the results in Figure 3.3 confirm this prediction. The last two columns of Table 3.5 also confirms the asymptotic result (3.26) that the number of iHDG-II iterations scales linearly with the time stepsize.

Next, we study the iHDG iteration growth as the solution order p increases. The asymptotic result (3.26) predicts that $k = O(p^2)$. In Table 3.6, rows 2–4 show the ratio of the number of iterations taken for solution orders $p = \{2, 3, 4\}$ over the one

h	p	iHDG-I		iHDG-II	
		$\Delta t = 10^{-1}$	$\Delta t = 10^{-2}$	$\Delta t = 10^{-1}$	$\Delta t = 10^{-2}$
0.25	1	19	6	14	6
0.125	1	*	6	18	9
0.0625	1	*	7	32	10
0.03125	1	*	9	59	8
0.25	2	*	9	15	9
0.125	2	*	11	19	9
0.0625	2	*	13	32	11
0.03125	2	*	15	59	12
0.25	3	*	7	16	8
0.125	3	*	9	20	8
0.0625	3	*	12	31	10
0.03125	3	*	*	59	12
0.25	4	*	10	17	9
0.125	4	*	12	32	10
0.0625	4	*	*	60	9
0.03125	4	*	*	112	13

Table 3.5: Comparison of iHDG-I and iHDG-II for the linearized shallow water system.

p	Meshsize (h)				Asymptotics
	0.25	0.125	0.0625	0.03125	
2	1.07	1.06	1	1	2
3	1.14	1.11	0.97	1	3.33
4	1.21	1.78	1.87	1.9	5

Table 3.6: Growth of iterations with solution order p for the iHDG-II method for the linearized shallow water system with $\Delta t = 10^{-1}$.

for $p = 1$ for four different meshsizes as in Table 3.5. As can be seen, the theoretical estimation is conservative.

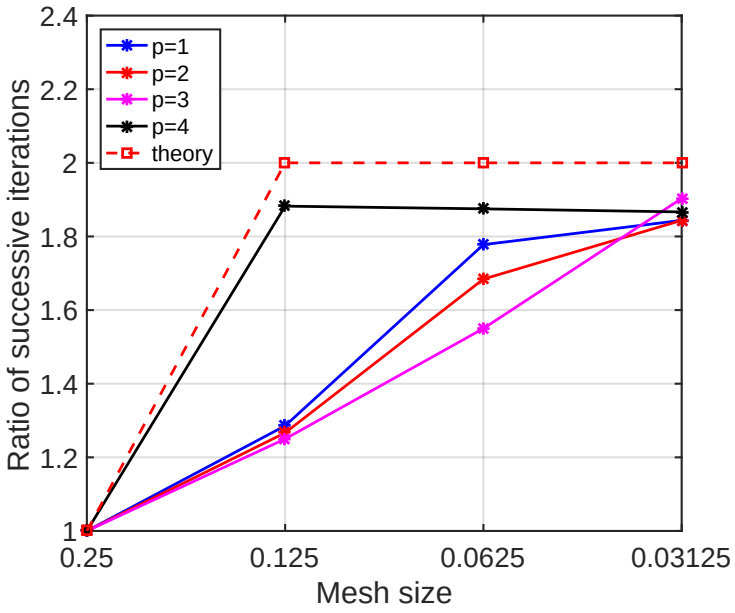


Figure 3.3: Growth of iterations with meshsize h for the iHDG-II method for the linearized shallow water system with $\Delta t = 10^{-1}$.

3.4.3 Nonlinear shallow water system

In this section, we consider the nonlinear shallow water system, given by,

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{u}) = \mathbf{f}, \tag{3.37}$$

where the forcing function $\mathbf{f}$, and the x-component and y-component of the flux $\mathbf{F}$ are given by

$$\mathbf{F}_x := \begin{pmatrix} Hu \\ Hu^2 + \frac{1}{2}gH^2 \\ Huv \end{pmatrix}, \quad \mathbf{F}_y := \begin{pmatrix} Hv \\ Huv \\ Hv^2 + \frac{1}{2}gH^2 \end{pmatrix}, \text{ and } \mathbf{f} := \begin{pmatrix} 0 \\ -gb_x \\ -gb_y \end{pmatrix},$$

while the conservative variables $\mathbf{u}$ are defined as

$$\mathbf{u} := (H, Hu, Hv)^T.$$

Here, H is the water depth, u is the depth averaged velocity component in the x-direction, v is the depth averaged velocity component in the y-direction, b is the bathymetry, and g is the gravitational constant.

For nonlinear problems both the local solver (2.2a) and the conservation constraints (2.2b) are nonlinear. In order to tackle these problems, we apply the iHDG algorithms to solve the linearized system arising from each Newton step of the HDG system (2.2). For the clarity of the exposition, let us consider one generic Newton step. To begin, we define the following residuals for time dependent versions of (2.2a) and (2.2b):

$$\mathcal{R}es = (\partial_t \mathbf{u}, \mathbf{v})_{\Omega_h} - (\mathbf{F}(\mathbf{u}), \nabla \mathbf{v})_{\Omega_h} + \left\langle \hat{\mathbf{F}}(\mathbf{u}, \hat{\mathbf{u}}) \cdot \mathbf{n}, \mathbf{v} \right\rangle_{\partial\Omega_h} + (\mathbf{C}\mathbf{u}, \mathbf{v})_{\Omega_h} - (\mathbf{f}, \mathbf{v})_{\Omega_h}, \tag{3.38a}$$

$$\mathcal{F}lx = \left\langle [\![\hat{\mathbf{F}}(\mathbf{u}, \hat{\mathbf{u}}) \cdot \mathbf{n}]\!], \boldsymbol{\mu} \right\rangle_{\mathcal{E}_h}. \tag{3.38b}$$

Here, the hybridized Lax-Friedrichs one, used in [25] is employed, i.e.,

$$\hat{\mathbf{F}}(\mathbf{u}, \hat{\mathbf{u}}) \cdot \mathbf{n} = \mathbf{F}(\mathbf{u}) \cdot \mathbf{n} + \tau \mathbf{I}(\mathbf{u} - \hat{\mathbf{u}}),$$

with $\hat{\mathbf{u}} := \left(\hat{H}, \widehat{Hu}, \widehat{Hv}\right)^T$, the stabilization $\tau = \sqrt{\hat{u}^2 + \hat{v}^2} + g\hat{H}$, and $\mathbf{I}$ the corresponding identity matrix. Note that $\hat{u}$ and $\hat{v}$ are given by $\hat{u} = \frac{\widehat{Hu}}{\hat{H}}$ and $\hat{v} = \frac{\widehat{Hv}}{\hat{H}}$.

If we define a Newton step for $\mathbf{u}$ and $\hat{\mathbf{u}}$ as $\delta\mathbf{u}$ and $\delta\hat{\mathbf{u}}$, respectively, the linear system for $\delta\mathbf{u}$ and $\delta\hat{\mathbf{u}}$ resulting from each Newton step is given by (after the temporal derivative is discretized, e.g., with backward Euler or Crank-Nicolson)

$$\begin{bmatrix} \frac{\partial \mathcal{R}es}{\partial \mathbf{u}} & \frac{\partial \mathcal{R}es}{\partial \hat{\mathbf{u}}} \\ \frac{\partial \mathcal{F}lx}{\partial \mathbf{u}} & \frac{\partial \mathcal{F}lx}{\partial \hat{\mathbf{u}}} \end{bmatrix} \begin{Bmatrix} \delta\mathbf{u} \\ \delta\hat{\mathbf{u}} \end{Bmatrix} = \begin{Bmatrix} -\mathcal{R}es \\ -\mathcal{F}lx \end{Bmatrix}, \tag{3.39}$$

Applying iHDG-II algorithm to the linear system (3.39) we get

$$\frac{\partial \mathcal{R}es}{\partial \mathbf{u}} \delta\mathbf{u}^{k+1} + \frac{\partial \mathcal{R}es}{\partial \hat{\mathbf{u}}} \delta\hat{\mathbf{u}}^{k,k+1} = -\mathcal{R}es, \tag{3.40}$$

where $\delta\hat{\mathbf{u}}^{k,k+1}$ is determined from the conservation condition as

$$[\![\frac{\partial \mathcal{F}lx}{\partial \hat{\mathbf{u}}}]\!]\delta\hat{\mathbf{u}}^{k,k+1} = -[\![\mathcal{F}lx]\!] - \left(\frac{\partial \mathcal{F}lx}{\partial \mathbf{u}}\right)^{-}\left(\delta\mathbf{u}^{k+1}\right)^{-} - \left(\frac{\partial \mathcal{F}lx}{\partial \mathbf{u}}\right)^{+}\left(\delta\mathbf{u}^{k}\right)^{+}. \tag{3.41}$$

Once convergence is obtained, the volume and trace unknown are updated as

$$\mathbf{u} \longrightarrow \mathbf{u} + \delta\mathbf{u},$$

$$\hat{\mathbf{u}} \longrightarrow \hat{\mathbf{u}} + \delta\hat{\mathbf{u}},$$

and we can proceed with the next Newton step.

For the first example, we study the convergence of the iHDG-II solver for a translating vortex solution [64] whose exact solution is given by

$$H^e = \left[1 - \frac{(\gamma-1)}{16\gamma\pi^2}\beta^2 e^{2\left(1-R^2\right)}\right]^{\frac{1}{\gamma-1}}, \qquad u^e = 1 - \beta e^{\left(1-R^2\right)}\frac{(y-y_0)}{2\pi},$$
$$v^e = \beta e^{\left(1-R^2\right)}\frac{(x-t-x_0)}{2\pi},$$

with $R^2 = (x-t-x_0)^2 + (y-y_0)^2, x_0 = 5, y_0 = 0$, and $\beta = 5$. We take $\gamma = g = 2$, and a flat bathymetry $b = b_x = b_y = 0$. The domain considered is $\Omega = [3.5, 5.5] \times [-1, 1]$ and structured quadrilateral elements are used. The exact solution is used to enforce the boundary condition. For time discretization, we again use the Crank-Nicolson method, in which the time step is $\Delta t = 10^{-4}$ and there are 100 time steps. In Figure 3.4, we compare the $h-$convergence rates obtained with the iHDG-II algorithm and the direct solver. Results from both solvers are on top of each other with the convergence rate between p and $(p + 1/2)$. In this case, each time step takes $2 - 3$ Newton iterations, each of which takes less than 10 iHDG-II iterations. The stopping criterion for iHDG-II algorithm is based on (2.49) and the stopping tolerance is taken to be 10^{-10} for both iHDG-II and Newton iterations.

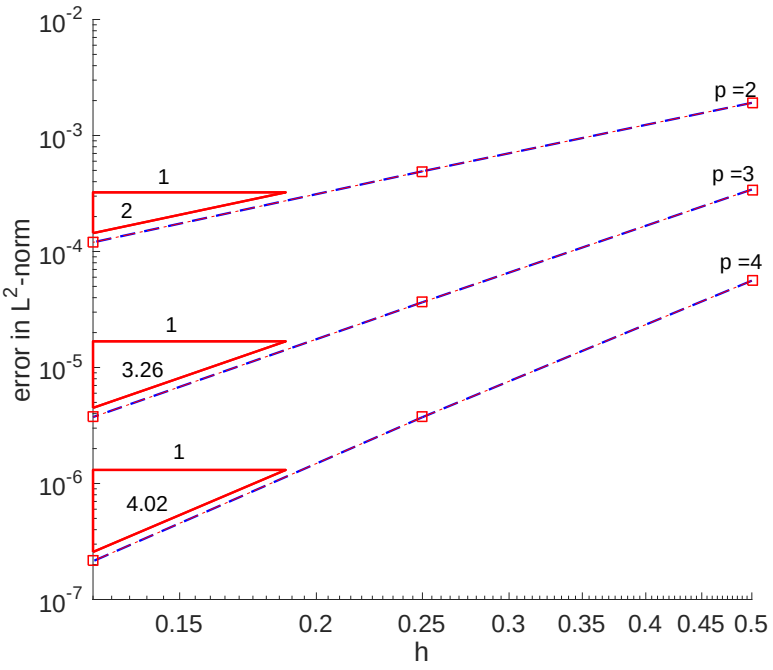


Figure 3.4: Nonlinear shallow water system with translating vortex solution: convergence rate for HDG methods with iHDG-II algorithm (blue dashed line) and direct solver (red dashed squares).

Next, we consider the water drop problem considered in [25, 136, 161]. The initial conditions are

$$H(x,y,0) := 1 + 0.1\exp\left[-100(x-0.5)^2 - 100(y-0.5)^2\right],$$

and

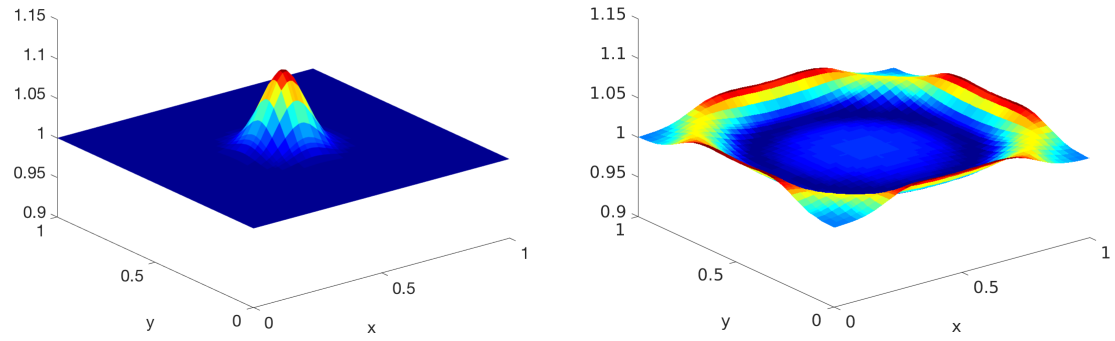
$$Hu(x,y,0) = Hv(x,y,0) = 0,$$

that is, the flow is initially at rest.

We consider the case with flat bottom i.e., $b = b_x = b_y = 0$ and the domain of interest is $\Omega = [0,1]^2$. A structured quadrilateral mesh with 64 elements ($h = 0.125$) and solution order $p = 6$ is used. Wall boundary conditions are applied to the entire boundary $\partial\Omega$. The Crank-Nicolson method with a time step of $\Delta t =$

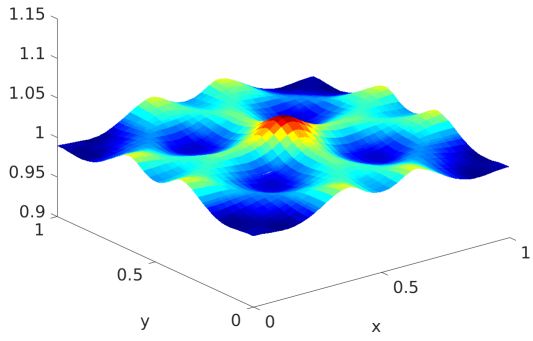
0.0005 is employed and the simulation is run for 2000 time steps. The time evolution of the water depth H and the depth averaged y-velocity v are shown in Figures 3.5 and 3.6, respectively. The u velocity evolution is same as v but rotated by 90 degrees, and hence is not shown. The numerical results are comparable to those in [25, 136, 161]. In Figure 3.7, we show the number of iHDG-II iterations taken per Newton iteration at the indicated times. The horizontal axis represents the number of Newton iterations taken from $(t - \Delta t)$ to t at the indicated times t. The markers in the vertical axis indicate the number of iHDG-II iterations taken at the corresponding Newton iteration in the horizontal axis to solve the linear system (3.39). The stopping tolerance is taken to be 10^{-10} for both iHDG-II and Newton iterations. As can be seen, approximately 16 Newton iterations are required per time step and the number of iHDG-II iterations decreases with each Newton iteration. The reason is that, after each Newton iteration, the initial guess (the solution from the previous Newton iteration) for iHDG iteration is improved and upon convergence Newton steps are smaller.

In Table 3.7 we compare the maximum number of iHDG-II iterations and Newton iterations taken per time step for different meshsizes and time steps. Similar to linear problems the number of iHDG-II iterations increases for finer meshsizes, higher solution orders, and larger time steps. The number of Newton iterations on the other hand decreases for finer meshsizes and approaches a constant. This is well-known as the number of Newton iterations depends on the nonlinearity of the problem and how well it is captured by the meshsize. Once the nonlinearity is captured by a particular meshsize and solution order, the number of iterations remains unchanged with further refinements [153, 2]. For smaller time steps the number of Newton iterations is reduced since the solutions at two consecutive time steps are close to each other.

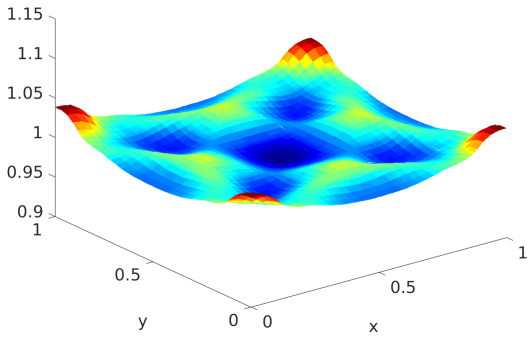


(a) H at $t = 0$

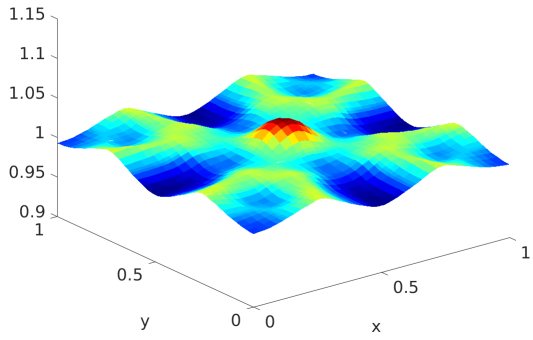
(b) H at $t = 0.15$



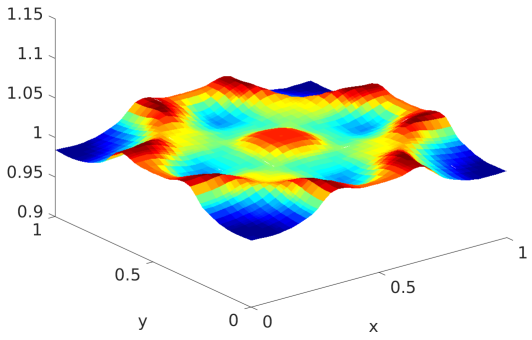
(c) H at $t = 0.3$



(d) H at $t = 0.5$



(e) H at $t = 0.7$



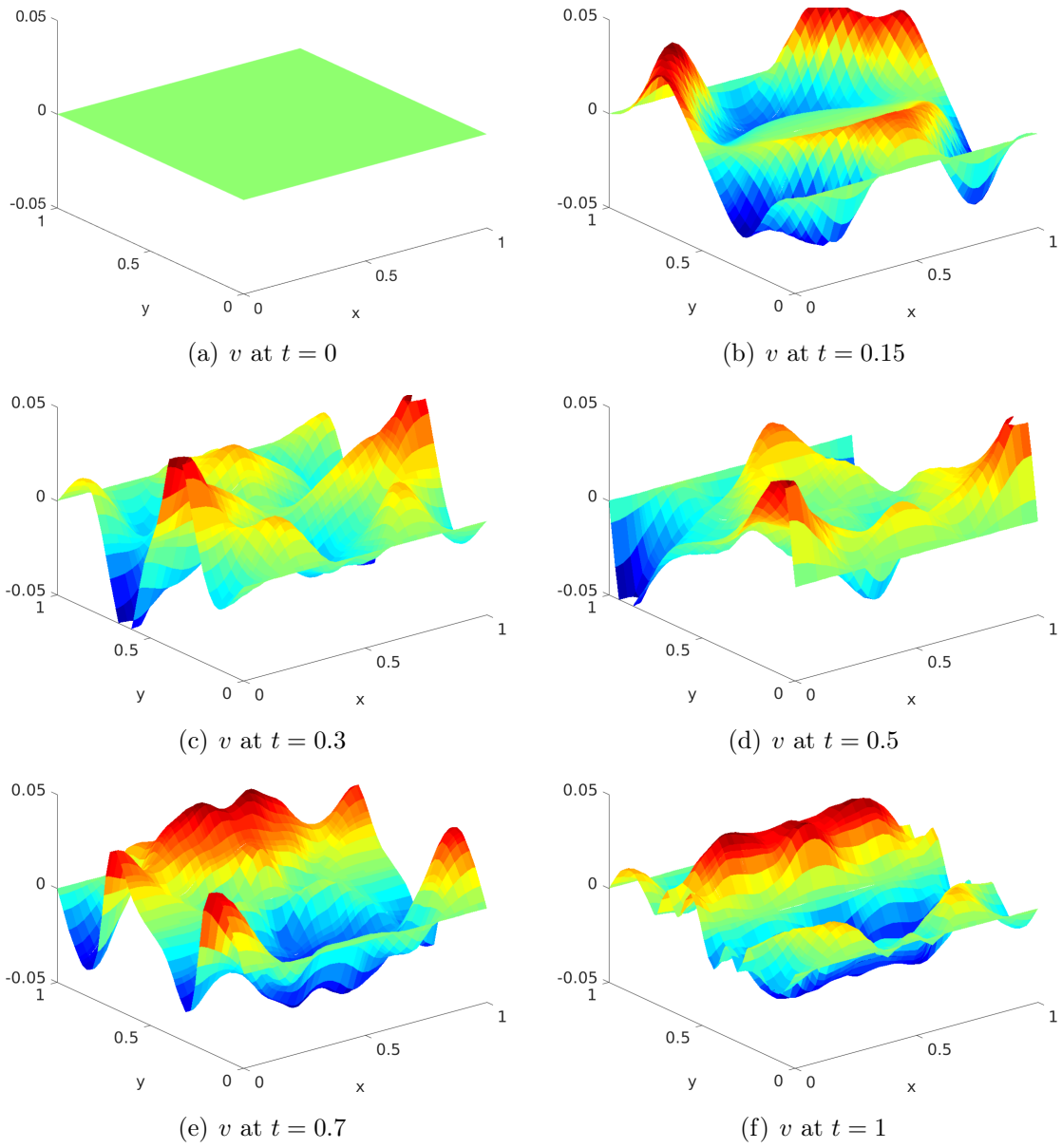
(f) H at $t = 1$

Figure 3.5: Evolution of the water depth H for the water drop test case with iHDG-II algorithm.

3.4.4 Linear convection-diffusion equation

In this section the following exact solution for equation (2.34) is considered

$$u^e = \frac{1}{\pi}\sin(\pi x)\cos(\pi y)\sin(\pi z).$$



(a) v at $t = 0$

(b) v at $t = 0.15$

(c) v at $t = 0.3$

(d) v at $t = 0.5$

(e) v at $t = 0.7$

(f) v at $t = 1$

Figure 3.6: Evolution of depth averaged y-velocity v for the water drop test case with iHDG-II algorithm.

The forcing is chosen such that it corresponds to the exact solution. The velocity field is chosen as $\boldsymbol{\beta} = (1 + z, 1 + x, 1 + y)$ and we take $\nu = 1$. The domain is $[0, 1] \times [0, 1] \times [0, 1]$. A structured hexahedral mesh is used for the simulations. The stopping criterion based on the exact solution (2.50) is used.

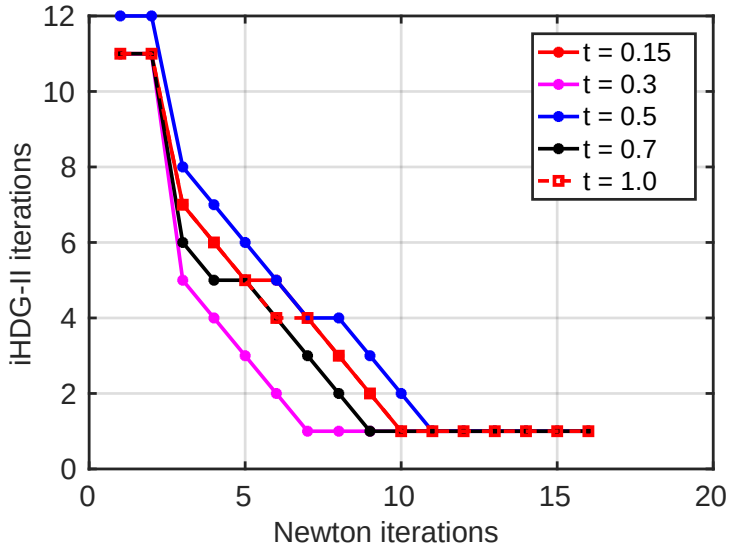


Figure 3.7: Number of iHDG-II iterations taken for each Newton iteration at different times for the water drop test case with $h = 0.125$, $p = 6$ and $\Delta t = 0.0005$.

In Table 3.8 we report the number of iterations taken by iHDG-I and iHDG-II methods for different values of the diffusion coefficient κ. Similar to the shallow water equations, due to the lack of stability in $\boldsymbol{\sigma}$, iHDG-I diverges when κ is large for fine meshes and/or high solution orders. The iHDG-II method, on the other hand, converges for all the meshes and solution orders, and the number of iterations are smaller than that of the iHDG-I method. Next, we verify the growth of iHDG-II iterations predicted by the asymptotic result (3.36).

Since $\min\{\kappa^{-1}, \lambda\} = \lambda$ for all the numerical results considered here, due to (3.36) we expect the number of iHDG-II iterations to be independent of κ and this can be verified in Table 3.8. In Figures 3.8(a), 3.8(b) and 3.8(c) the growth of iterations with respect to meshsize h for $\kappa = 10^{-2}, 10^{-3}$ and 10^{-6} are compared. In the asymptotic limit, for all the cases, the ratio of successive iterations reaches a value of around 1.7 which is close to the theoretical prediction 2. Hence the theoretical analysis predicts well the growth of iterations with respect to the meshsize h. On the other

h	p	iHDG-II		Newton	
		$\Delta t = 10^{-3}$	$\Delta t = 10^{-4}$	$\Delta t = 10^{-3}$	$\Delta t = 10^{-4}$
0.25	1	7	5	7	4
0.125	1	8	5	6	3
0.0625	1	10	5	3	3
0.03125	1	13	6	3	3
0.25	2	8	5	8	4
0.125	2	9	5	6	4
0.0625	2	12	6	6	3
0.03125	2	17	7	3	3
0.25	3	9	5	9	5
0.125	3	11	5	8	4
0.0625	3	14	6	3	3
0.03125	3	20	7	3	3
0.25	4	9	5	10	5
0.125	4	12	6	9	4
0.0625	4	18	7	5	3
0.03125	4	26	9	7	3

Table 3.7: Number of iHDG-II and Newton iterations for different meshsizes and time stepsizes for the water drop test case of nonlinear shallow water system.

hand, columns $6-8$ in Table 3.8 show that the iterations are almost independent of solution orders. This is not predicted by the theoretical results which indicates that the number of iterations scales like $\mathcal{O}(p^2)$. The reason is due to the convection dominated nature of the problem, for which we have shown that the number of iterations is independent of the solution order.

Finally, we consider the elliptic regime with $\kappa = 1$ and $\boldsymbol{\beta} = 0$. For this case we use the following stopping criterion based on the direct solver solution u_{direct}

$$\left\| u^k - u_{direct} \right\|_{L^2(\Omega)} < 10^{-10}.$$

As shown in Figure 3.8(d) and Table 3.9, our theoretical analysis predicts well the relation between the number of iterations and the meshsize and the solution order.

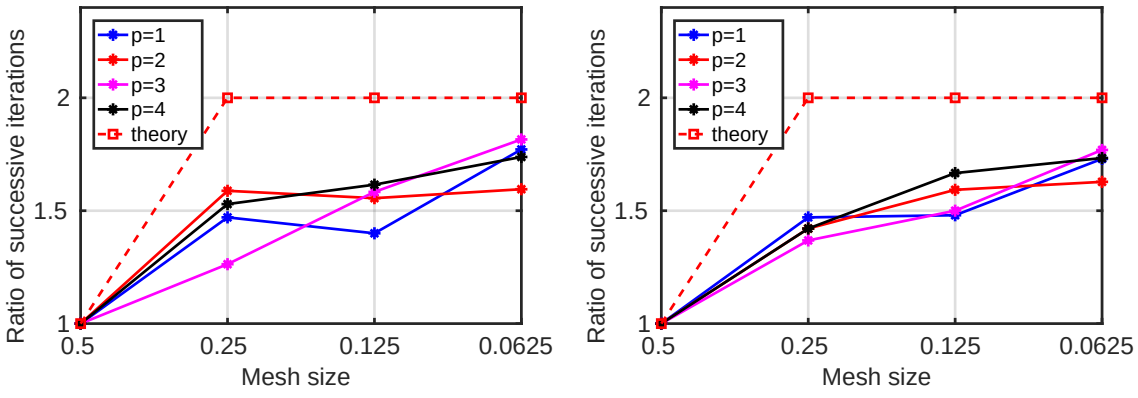
In Table 3.9 " $*$ " represents that the scheme has reached the maximum number of iterations specified i.e. 2000 and didn't converge to the specified tolerance limit of 10^{-10} yet.

h	p	iHDG-I			iHDG-II		
		$\kappa = 10^{-2}$	$\kappa = 10^{-3}$	$\kappa = 10^{-6}$	$\kappa = 10^{-2}$	$\kappa = 10^{-3}$	$\kappa = 10^{-6}$
0.5	1	24	23	23	17	17	17
0.25	1	30	34	35	25	25	26
0.125	1	50	55	56	35	37	38
0.0625	1	90	94	97	62	64	65
0.5	2	26	24	25	17	19	19
0.25	2	41	42	42	27	27	27
0.125	2	66	67	67	42	43	43
0.0625	2	*	109	110	67	70	71
0.5	3	27	31	31	19	19	19
0.25	3	33	33	38	24	26	27
0.125	3	*	58	60	38	39	41
0.0625	3	*	102	106	69	69	71
0.5	4	26	27	27	17	19	19
0.25	4	50	41	43	26	27	27
0.125	4	*	71	72	42	45	46
0.0625	4	*	123	125	73	78	79

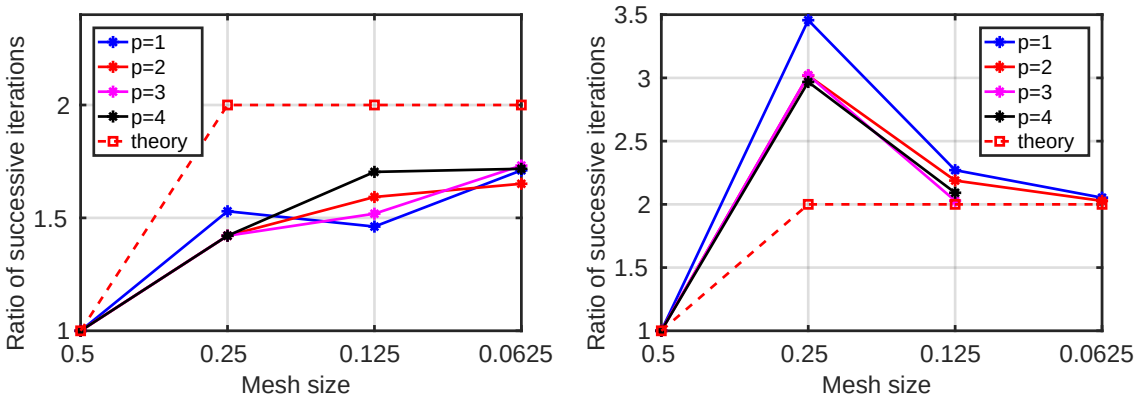
Table 3.8: Comparison of iHDG-I and iHDG-II methods for different κ.

p	Meshsize (h)				Asymptotics
	0.5	0.25	0.125	0.0625	
2	2.41	2.11	2.03	2.01	2
3	4.07	3.55	3.17	*	3.33
4	6.09	5.23	4.81	*	5

Table 3.9: Growth of iterations with p for the iHDG-II method for the elliptic equation.



(a) $\kappa = 10^{-2}$ (b) $\kappa = 10^{-3}$



(c) $\kappa = 10^{-6}$ (d) $\kappa = 1$ (elliptic)

Figure 3.8: Ratio of successive iterations as we refine the mesh for the iHDG-II method for different κ.

3.4.5 SPE10 test case

In this section we consider a benchmark problem of subsurface flow through porous media from the Tenth Society of Petroleum Comparative Solution Project (SPE10, model 2) [35]. The flow is governed by the following elliptic equation (Darcy law) written in the first-order form,

$$\begin{aligned} \boldsymbol{\sigma} &= -\kappa \nabla u \quad \text{on} \quad \Omega, \\ \nabla \cdot \boldsymbol{\sigma} &= f \quad \text{on} \quad \Omega, \end{aligned} \tag{3.42}$$

where $\boldsymbol{\sigma}$ is the Darcy velocity, u is the pressure and κ is the permeability.

We consider the permeability field corresponding to the 75th layer as shown in Figure 3.9(a). The permeability field varies by six orders of magnitude and is highly heterogeneous. It gives rise to extremely complex velocity fields. The considered domain is 1200×2200 $[ft^2]$. The mesh is chosen as 60×220 quadrilateral elements, so that the mesh skeleton aligns with the discontinuities in the permeability and the permeability field is constant within each element. We choose $f = 0$, which corresponds to the fact that there is no source or sink. For the boundary conditions, we take the pressure on the left and right faces to be 1 and 0 respectively. On the top and bottom faces, no-flux boundary condition $\boldsymbol{\sigma} \cdot \mathbf{n} = 0$ is applied. We use the upwind HDG flux (2.36) with the velocity $\boldsymbol{\beta} = 0$ for this problem. In Figures 3.9(b), and 3.10(a) we show the pressure field and the velocity field using direct solver with solution order $p = 1$.

We next consider the iHDG-II algorithm with three different initial guesses:

1. zero,

2. solution of equation (3.42) with the average of the original permeability field over the whole domain,

3. solution of equation (3.42) with the permeability field in each element given by

$$\kappa = \kappa + rand(0, 1) \times \kappa,$$

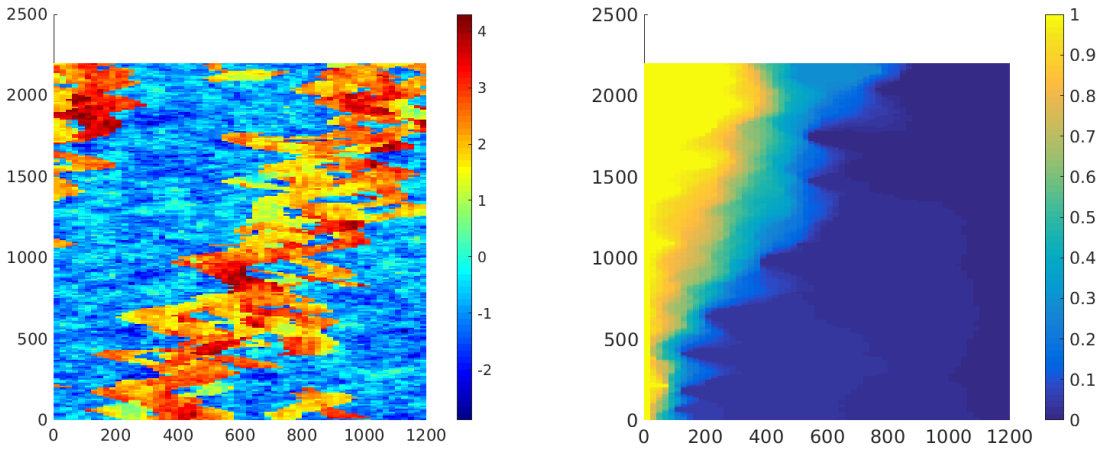
where $rand(0, 1)$ is a random number between 0 and 1, i.e., we randomly perturb the original permeability value in each element by 0%—100%.

In Figure 3.11 we show the three initial guesses for the pressure and in Figure 3.12 are the pressure fields corresponding to these initial guesses after 2000 iHDG-II iterations using solution order $p = 1$. Since the third initial guess is the closest to

the direct solution (compared to Figure 3.9(b)), the corresponding pressure field is almost the same as the direct solution, while the others are not. The conclusion is similar for the corresponding velocity fields in Figures 3.13 and 3.14. In order to have a more quantitative comparison, we plot the relative error

$$\|error\| = \frac{\sqrt{\|\boldsymbol{\sigma}^{k+1} - \boldsymbol{\sigma}^{k}\|^2 + \|u^{k+1} - u^{k}\|^2}}{\sqrt{\|\boldsymbol{\sigma}^{k+1}\|^2 + \|u^{k+1}\|^2}}$$

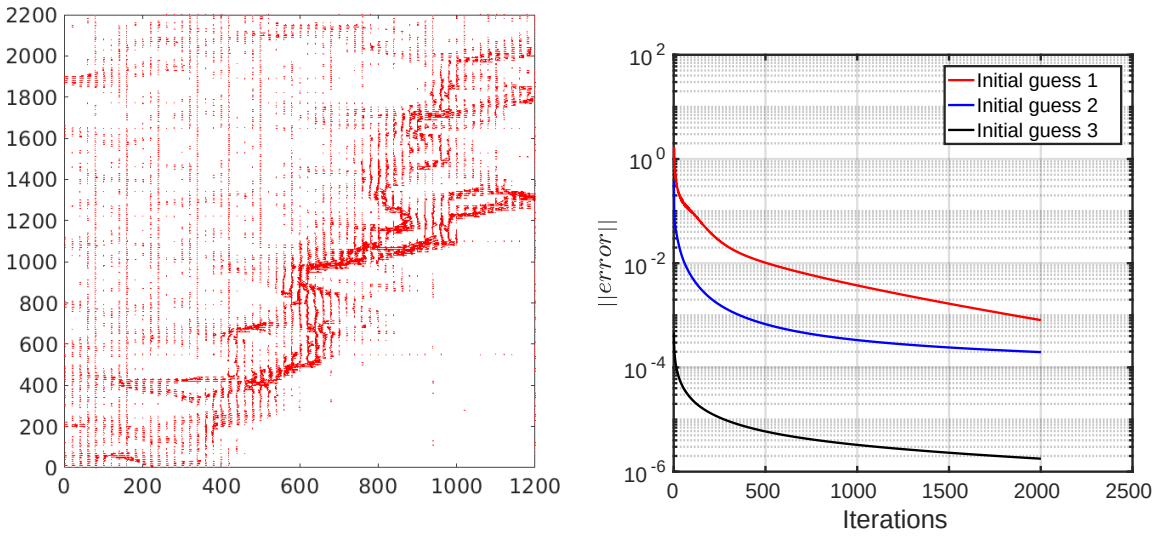
in Figure 3.10(b) as the number of iHDG iterations increases. For all of the initial guesses, there is a quick drop in the errors in the first few iterations and thereafter the errors decrease slowly. Compared to the other examples, this is the most challenging one, for which the iHDG algorithm, if used as a direct solver, could be ineffective. Nevertheless, this is a common feature of many of the fixed-point iterative methods like Jacobi and Gauss-Seidel [24, 145]. Ongoing work is to employ iHDG as a smoother for multigrid methods or as a preconditioner for GMRES iterations, and this is part of our future work.



(a) Permeability field

(b) Pressure field

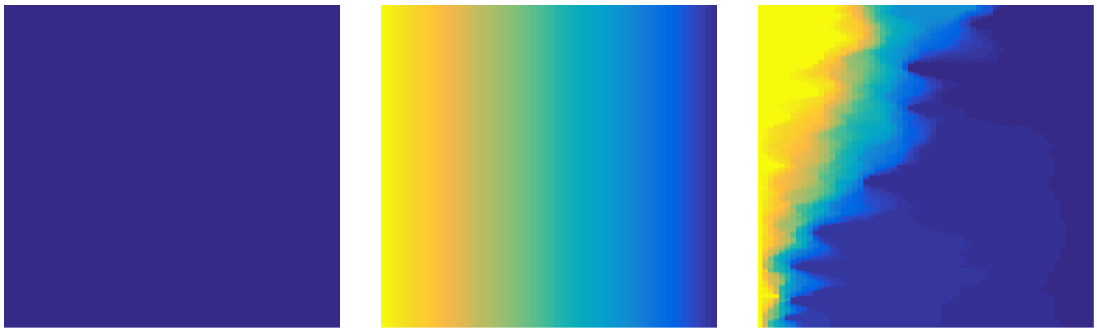
Figure 3.9: SPE10 test case: (a) permeability field in log scale, and (b) pressure field from direct solver for $p = 1$ solution.



(a) Velocity field

(b) Error history

Figure 3.10: SPE10 test case: (a) velocity field from direct solver for $p = 1$ solution, and (b) error history with respect to iHDG-II iterations for the three different initial guesses.



(a) Initial guess 1

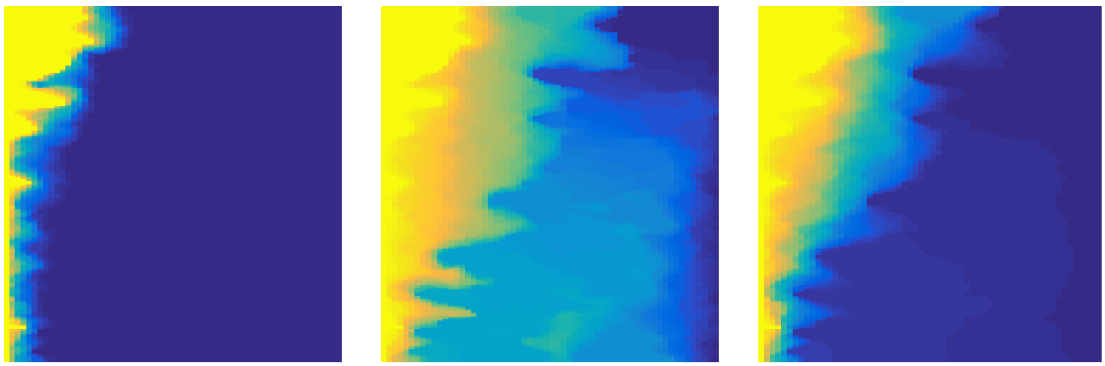
(b) Initial guess 2

(c) Initial guess 3

Figure 3.11: SPE10 test case: three different initial guesses for the pressure field.

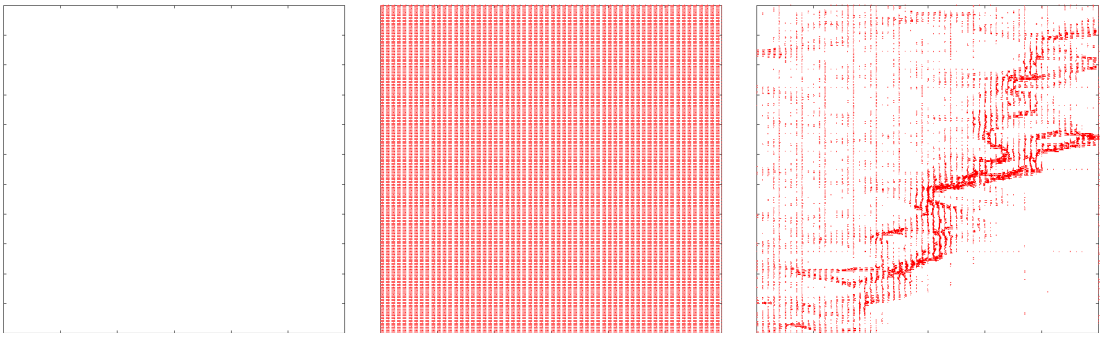
3.5 Discussion

In this chapter we have presented an iterative HDG approach which improves upon the algorithm introduced in chapter 2 in several aspects. In particular, it converges in a finite number of iterations for the scalar transport equation and is uncon-



(a) iHDG solution with initial guess 1 (b) iHDG solution with initial guess 2 (c) iHDG solution with initial guess 3

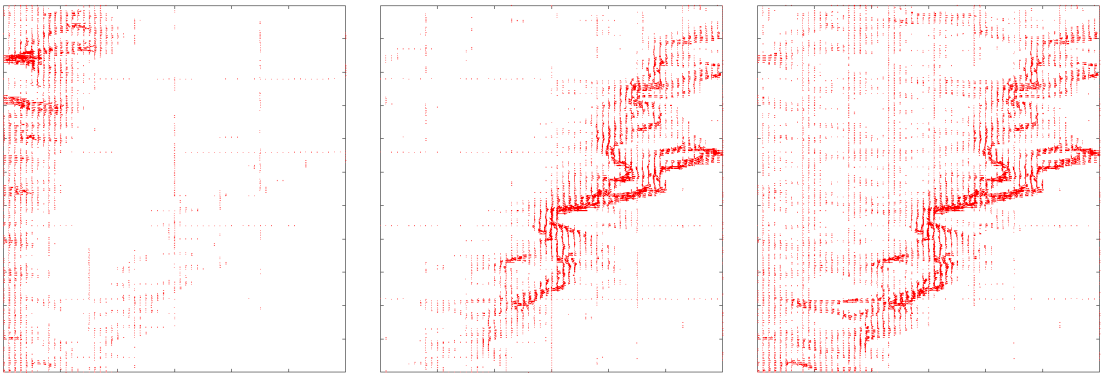
Figure 3.12: SPE10 test case: the pressure fields obtained with the iHDG-II algorithm after 2000 iterations with three different initial guesses.



(a) Initial guess 1 (b) Initial guess 2 (c) Initial guess 3

Figure 3.13: SPE10 test case: three different initial guesses for the velocity field.

ditionally convergent for both the linearized shallow water system and the convection-diffusion equation. Moreover, we provide several additional findings: 1) we make a connection between iHDG and the parareal method, which reveals interesting similarities and differences between the two methods; 2) we show that iHDG can be considered as a *locally implicit* method, and hence being somewhat in between fully explicit and fully implicit approaches; 3) for both the linearized shallow water system and the convection-diffusion equation, using an asymptotic approximation, we uncover a relationship between the number of iterations and time stepsize, solution



(a) iHDG solution with initial guess 1

(b) iHDG solution with initial guess 2

(c) iHDG solution with initial guess 3

Figure 3.14: SPE10 test case: velocity field obtained with iHDG-II algorithm after 2000 iterations with three different initial guesses.

order, meshsize and the equation parameters. This allows us to choose the time step-size such that the number of iterations is approximately independent of the solution order and the meshsize; 4) we show that iHDG-II has improved stability and convergence rates over iHDG-I; 5) we provide both strong and weak scalings of our iHDG approach up to $16,384$ cores; and 6) we show how iHDG approaches can be used as a linear solver for nonlinear problem.

We want to note here that for both iHDG-I and iHDG-II approaches the number of iterations increases with the increase in number of elements which is a characteristic of one level domain decomposition methods. In order to keep the iterations constant or reduce the growth with mesh refinements we need coarse solvers which provide global coupling to the iHDG solvers. In that sense multigrid/multilevel solvers are attractive and iHDG approaches can be used as smoothers in them.

While the usage of iHDG-I solvers as smoothers is straightforward because of the single valued nature of trace unknowns, for iHDG-II solvers it is not the case. Typically as a smoother we only perform few iterations and for iHDG-II the trace

unknowns are single valued only upon convergence up to the specified tolerance. Thus it is not clear how to directly use the iHDG-II solvers as smoothers inside multigrid/multilevel approaches for the HDG trace system. There are few options that are possible, after few iterations with iHDG-II solver we can use the conservation condition to compute the single valued trace unknowns from the volume unknowns or we can simply take average or some weighted average of the multivalued trace unknowns after few iterations. Analysis is required to study the effectiveness of iHDG approaches as smoothers and is reserved for future work.

Chapter 4

A Geometric Multigrid for HDG Trace Systems

4.1 Multigrid methods for trace systems of hybridized methods

Over the past 30 years, a tremendous amount of research has been devoted to the convergence of multigrid methods for linear systems arising from hybridized methods, both as iterative methods and as preconditioners for Krylov subspace methods. Optimal convergence with respect to the number of unknowns is usually obtained under mild elliptic regularity assumptions [18, 19, 20]. Multigrid algorithms have been developed for mortar domain decomposition methods [16, 156]. Several multigrid algorithms have been proposed for hybridized mixed finite element methods [21, 34]. Most of them are based on an equivalence between the interface operator and a nonconforming finite element method, and for these methods optimal convergence has already been established [15, 22]. Multigrid algorithms based on restricting the trace (skeletal) space to linear continuous finite element space has been proposed for hybridized mixed methods [74], hybridized discontinuous Galerkin methods [41], and weak Galerkin methods [33]. These algorithms fall under the non-inherited category, i.e., the coarse scale operators do not inherit all the properties of the fine scale ones. To the best of our knowledge, no multigrid algorithm has been developed for the case of multinumerics.

In this chapter[1] we develop a multigrid algorithm that applies to both hybridized formulations and multinumerics. Here we only show results for hybridized DG methods to match well with the contents of the other chapters and encourage the readers to refer [158] for an example on multinumerics. The algorithm applies to both structured and unstructured grids. At the heart of our approach is the energy-preserving intergrid transfer operators which are a function of only the fine scale DtN maps (to be discussed in details in section 4.3). As such they avoid any explicit upscaling of parameters, and at the same time allow for the use of multinumerics throughout the domain. The multigrid algorithm presented here thus differs from the existing approaches in that the Galerkin coarse grid operator is a discretized DtN map on every level.

This chapter is organized as follows. Section 4.2 introduces the model problem and hybridized DG methods considered. In Section 4.3, we define the necessary ingredients for our geometric multigrid algorithm, that is, the coarsening strategy, the intergrid transfer operators, the local correction operator, and the smoothing operator. These operators are then used to define the multigrid algorithm. Section 4.4 presents several numerical examples to study the robustness of the proposed algorithm for different hybridized DG methods, smoothers, and test cases.

[1]The contents of this chapter are largely based on the manuscript [158], a slightly reduced version of which is accepted for publication in the SIAM journal on scientific computing. The contributions of the author ranged from numerical implementation of the multigrid solver for HDG, participation in the theoretical analysis and writing the manuscript.

4.2 Model problem and hybridized DG methods

Consider the following second order elliptic equation

$$-\nabla \cdot (\mathbf{K}\nabla q) = f \qquad \text{in} \quad \Omega, \tag{4.1a}$$

$$q = g_D \qquad \text{on} \quad \partial\Omega. \tag{4.1b}$$

where Ω is an open, bounded, and connected subset of $\mathbb{R}^d$, with $d \in \{2, 3\}$. Here, $\mathbf{K}$ is a symmetric, bounded, and uniformly positive definite tensor, $f \in L^2(\Omega)$, and $g_D \in H^{3/2}(\partial\Omega)$.

First, we cast (4.1) into the following first-order or mixed form:

$$\mathbf{u} = -\mathbf{K}\nabla q \qquad \text{in } \Omega, \tag{4.2a}$$

$$\nabla \cdot \mathbf{u} = f \qquad \text{in } \Omega, \tag{4.2b}$$

$$q = g_D \qquad \text{on } \partial\Omega. \tag{4.2c}$$

The hybrid mixed DG method or hybridized DG (HDG) method for the discretization of (4.2) is defined as

$$\left(\mathbf{K}^{-1}\mathbf{u}, \mathbf{v}\right)_T - (q, \nabla \cdot \mathbf{v})_T + \langle \lambda, \mathbf{v} \cdot \mathbf{n} \rangle_{\partial T} = 0, \tag{4.3a}$$

$$-(\mathbf{u}, \nabla w)_T + \langle \hat{\mathbf{u}} \cdot \mathbf{n}, w \rangle_{\partial T} = (f, w)_T, \tag{4.3b}$$

$$\langle [\![\hat{\mathbf{u}} \cdot \mathbf{n}]\!], \mu \rangle_e = 0, \tag{4.3c}$$

where the numerical flux $\hat{\mathbf{u}} \cdot \mathbf{n}$ is given by

$$\hat{\mathbf{u}} \cdot \mathbf{n} = \mathbf{u} \cdot \mathbf{n} + \tau(q - \lambda). \tag{4.4}$$

For simplicity, we have ignored the fact that (4.3a), (4.3b) and (4.3c) must hold for all test functions $\mathbf{v} \in \mathbf{V}_h(T)$, $w \in W_h(T)$, and $\mu \in M_h(e)$, respectively (this

is implicitly understood throughout the chapter), where $\mathbf{V}_h$, W_h, and M_h are defined as

$$\begin{aligned}
\mathbf{V}_h(\Omega_h) &= \left\{ \mathbf{v} \in \left[L^2(\Omega_h)\right]^d : \mathbf{v}|_T \in \left[\mathcal{P}^p(T)\right]^d \forall T \in \Omega_h \right\}, \\
W_h(\Omega_h) &= \left\{ w \in L^2(\Omega_h) : w|_T \in \mathcal{P}^p(T) \forall T \in \Omega_h \right\}, \\
M_h(\mathcal{E}_h) &= \left\{ \lambda \in L^2(\mathcal{E}_h) : \lambda|_e \in \mathcal{P}^p(e) \forall e \in \mathcal{E}_h \right\},
\end{aligned}$$

and similar spaces $\mathbf{V}_h(T)$, $W_h(T)$, and $M_h(e)$ on T and e can be defined by replacing Ω_h with T and $\mathcal{E}_h$ with e, respectively.

Next, we consider the hybridized interior penalty DG (IPDG) schemes posed in the primal form. To that end, we test (4.2a) with $\mathbf{v} = \nabla w$ and then integrate by parts twice the terms on the right hand side to obtain

$$(\mathbf{u}, \nabla w)_T = -(\mathbf{K}\nabla q, \nabla w)_T + \langle (q-\lambda), \mathbf{K}\nabla w \cdot \mathbf{n} \rangle_{\partial T}.$$

Substituting this in equation (4.3b) gives

$$(\mathbf{K}\nabla q, \nabla w)_T - \langle (q-\lambda), \mathbf{K}\nabla w \cdot \mathbf{n} \rangle_{\partial T} + \langle \hat{\mathbf{u}} \cdot \mathbf{n}, w \rangle_{\partial T} = (f, w)_T. \tag{4.5}$$

For IPDG schemes, the numerical flux takes the form

$$\hat{\mathbf{u}} \cdot \mathbf{n} = -\mathbf{K}\nabla q \cdot \mathbf{n} + \tau(q-\lambda), \tag{4.6}$$

and it is required to satisfy the conservation condition (4.3c). Substituting the numerical flux (4.6) in (4.5) we get the following primal form of the hybridized IPDG scheme:

$$\begin{aligned}
(\mathbf{K}\nabla q, \nabla w)_T - \langle (q-\lambda), \mathbf{K}\nabla w \cdot \mathbf{n} \rangle_{\partial T} - \langle \mathbf{K}\nabla q \cdot \mathbf{n}, w \rangle_{\partial T} \\
+ \langle \tau(q-\lambda), w \rangle_{\partial T} = (f, w)_T.
\end{aligned} \tag{4.7}$$

This scheme is called the hybridized symmetric IPDG scheme (SIPG-H) and has been studied in [71, 38, 150]. In order to include the other hybridized IPDG schemes, we can generalize (4.7) to

$$(\mathbf{K}\nabla q, \nabla w)_T - s_f \langle (q-\lambda), \mathbf{K}\nabla w \cdot \mathbf{n}\rangle_{\partial T} - \langle \mathbf{K}\nabla q \cdot \mathbf{n}, w\rangle_{\partial T} + \langle \tau(q-\lambda), w\rangle_{\partial T} = (f, w)_T, \quad (4.8)$$

where $s_f \in \{-1, 0, 1\}$. The scheme corresponding to $s_f = -1$ is called the hybridized nonsymmetric IPDG scheme (NIPG-H) and the one with $s_f = 0$ is called the hybridized incomplete IPDG scheme (IIPG-H) [71]. It is interesting to note that in the HDG scheme, if we do not integrate by parts (4.3a) and substitute $\mathbf{u} = -\mathbf{K}\nabla q$ in (4.3b) and (4.4) we obtain the IIPG-H scheme.

The common solution procedure for all of these hybridized DG methods can now be described. First, we express the local volume unknowns $\mathbf{u}$ and/or q, element-by-element, as a function of the skeletal unknowns λ. Then, we use the conservation condition to construct a global linear system involving only the skeletal unknowns. Once they are solved for, the local volume unknowns can be recovered in an element-by-element fashion completely independent of each other. The main advantage of this Schur complement approach is that, for high-order methods, the global trace system is much smaller and sparser compared to the linear system for the volume unknowns [38, 27]. The question that needs to be addressed is how to solve the trace system efficiently. In chapters 2 and 3 we developed iterative HDG algorithms where we do not construct a global trace system and instead relied on solving local and global solvers alternatively to achieve convergence. The solvers however lacked algorithmic optimality and the iterations increases with the increase in number of elements. In the next section we develop a geometric multigrid algorithm to tackle this issue.

4.3 Geometric multigrid algorithm based on DtN maps

For all of the hybridized methods described in the previous section, the resulting linear systems for the skeletal unknowns λ, in operator form, can be written as

$$A\lambda = g. \tag{4.9}$$

The well-posedness of the trace system (4.9) for the hybridized methods discussed in the previous section has been shown in [38, 71].

The concept of a DtN map is essential to our approach, so we briefly describe it here. A map A is called a DtN map/operator if it maps Dirichlet data on a domain boundary to Neumann data. It is a particular type of Poincaré–Steklov operator, which contains a large class of operators which map one type of boundary condition to another for elliptic PDEs. In finite-dimension, the Schur complement of the linear system with volume unknowns condensed out is also known as a discrete DtN map. We refer readers to [122] for more information.

To define our multigrid algorithm we start with a sequence of partitions of the mesh Ω_h:

$$\Omega_1, \Omega_2, \ldots, \Omega_N = \Omega_h,$$

where each Ω_k contains N_{T_k} closed (not necessarily convex, except on the finest level where each N_{T_k} is in fact some element T_j in Ω_h) macro-elements consisting of unions of macro-elements from Ω_{k+1}. Associated with each partition, we define interface grids $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_N = \mathcal{E}_h$, where each $e \in \mathcal{E}_k$ is the intersection of two macro-elements in Ω_k. Each partition $\mathcal{E}_k$ is in turn associated with a skeletal (trace) space M_k (to be defined in Section 4.3.1).

Remark 7. *We note that if each Ω_k consists of simplices or parallelopipeds, then each M_k is straightforward to define. In general, however, the macro-elements need not be*

one of these standard shapes, and we need to define a parameterization of each macro-edge to define M_k. While this is certainly nontrivial, this calculation only needs to be performed one time for each mesh and can be reused for many simulation or time steps as long as the macro-elements (agglomeration) do not change. For all of the results presented in Section 4.4, the time required to calculate these parameterizations is negligible in comparison with the total simulation time.

We are now in position to introduce necessary ingredients to define our geometric multigrid algorithm.

4.3.1 Coarsening strategy

In the multigrid algorithm we first coarsen in p, followed by the coarsening in h. We explain these two coarsening procedures in detail as follows.

4.3.1.1 p-coarsening

In our p-coarsening strategy, we restrict the solution order p on the finest level N to $p = 1$ on level $N - 1$ with the same number of elements. In this case the Lagrange multiplier[2] spaces become

$$M_k = \begin{cases} \{\eta \in \mathcal{P}^p(e), \forall e \in \mathcal{E}_k\} & \text{for} \quad k = N, \\ \{\eta \in \mathcal{P}^1(e), \forall e \in \mathcal{E}_k\} & \text{for} \quad k = 1, 2, \ldots, N-1. \end{cases}$$

Let us comment on one subtle point. It would be ideal if we could use $\mathcal{P}^0$ in coarser levels because it is trivial to define piecewise constant functions along arbitrary macro-edges (two-dimensional) or macro-faces (three-dimensional), i.e., no parameterization is needed. However, multilevel algorithms using restriction and prolongation operators based on piecewise constant interpolation typically have been shown not to perform well [17, 92].

[2]Here, we refer trace or skeletal unknowns also as Lagrange multipliers.

For high-order methods, the numerical examples in Section 4.4 indicate that our strategy gives scalable results in solution order p when strong smoothers such as block-Jacobi or Gauss–Seidel are used. This is also observed in [79] for a p-multigrid approach applied to DG methods for the Poisson equation using block-Jacobi smoother. Other p-multigrid strategies, such as $p_{k-1} = p_k/2$ (p_k is the solution order on the kth level) or $p_{k-1} = p_k - 1$, can be straightforwardly incorporated within our approach. Once we interpolate to $p = 1$ we carry out the h-coarsening as described in the next section.

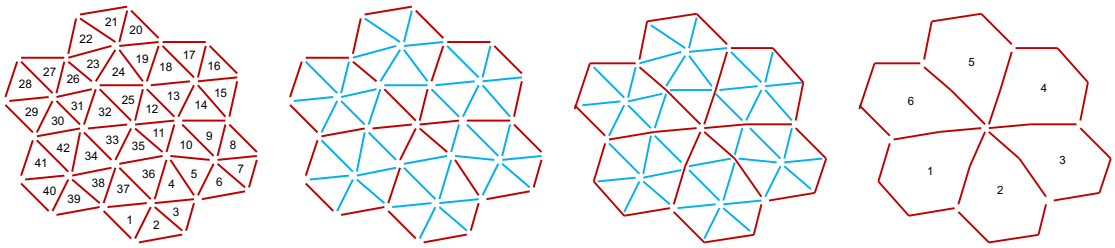


Figure 4.1: A demonstration of h-coarsening strategy. (a) Level k mesh. (b) An identification of interior $\mathcal{E}_{k,I}$ (blue) and boundary $\mathcal{E}_{k,B}$ (red) edges. (c) Coarsening of boundary edges. and (d) Level $k-1$ mesh after interior edges are statically condensed out. The numbers in (a) and (d) represent the number of (macro-) elements in levels k and $k-1$, respectively.

4.3.1.2 h-coarsening

We adopt an agglomeration-based h-coarsening, allowing intergrid transfer and coarse grid operators to be defined using the fine scale DtN maps. By taking this approach, no upscaling of parameters is required and our coarse operators are discretized DtN maps at any mesh level (see Proposition 2). In Figure 4.1, we show an h-coarsening strategy between two consecutive levels k and $k-1$. First, we identify the interior (blue) and boundary (red) edges in Figure 4.1(b). We decompose $\mathcal{E}_k = \mathcal{E}_{k,I} \oplus \mathcal{E}_{k,B}$, where $\mathcal{E}_{k,I}$ consists of edges interior to macro-elements in the

coarser partition Ω_{k-1} and $\mathcal{E}_{k,B}$ contains edges common to their boundaries. We also decompose the trace space M_k on $\mathcal{E}_k$ into two parts $M_{k,I}$ and $M_{k,B}$ corresponding to $\mathcal{E}_{k,I}$ and $\mathcal{E}_{k,B}$, respectively. Specifically, we require $M_k = M_{k,I} \oplus M_{k,B}$ such that each $\lambda_k \in M_k$ can be uniquely expressed as $\lambda_k = \lambda_{k,I} + \lambda_{k,B}$, where

$$\lambda_{k,I} = \begin{cases} \lambda_k & \text{on } M_{k,I}, \\ 0 & \text{on } M_{k,B}, \end{cases} \quad \text{and} \quad \lambda_{k,B} = \begin{cases} 0 & \text{on } M_{k,I}, \\ \lambda_k & \text{on } M_{k,B}. \end{cases}$$

Given the decomposition $M_k = M_{k,I} \oplus M_{k,B}$, the trace system (4.9) at the kth level can be written as

$$A_k \lambda_k = g_k \Leftrightarrow \begin{bmatrix} A_{k,II} & A_{k,IB} \\ A_{k,BI} & A_{k,BB} \end{bmatrix} \begin{bmatrix} \lambda_{k,I} \\ \lambda_{k,B} \end{bmatrix} = \begin{bmatrix} g_{k,I} \\ g_{k,B} \end{bmatrix}. \tag{4.10}$$

The coarser space M_{k-1} is defined such that $M_{k-1} \subset M_{k,B}$. This is done by first agglomerating the boundary edges of level k as in Figure 4.1(c), and then statically condensing out the interior edges to obtain the mesh on level $k-1$ in Figure 4.1(d). The elements in level k are numbered from 1 to 42 in Figure 4.1(a) and the macro elements in level $k-1$ are numbered from 1 to 6 in Figure 4.1(d)

Clearly, for an unstructured mesh the identification of interior and boundary edges is nontrivial and nonunique. Currently we use an ad hoc approach which is sufficient for the purpose of demonstrating our proposed algorithm. Specifically, we first select a certain number of levels N and seed points N_{T_1}. The locations of these seed points are chosen based on the geometry of the domain and the original fine mesh, such that we approximately have an equal number of fine mesh elements in each macro-element at level 1. Based on these selected seed points, we agglomerate the elements in the finest mesh ($k = N$) to form N_{T_1} macro-elements at level 1. We then divide each macro-element in level 1 into four approximately equal macro-elements to form level 2. This process is recursively repeated to create macro-elements in finer levels $k = 3, \ldots, N-1$. As an illustration we consider the mesh in Figure 4.2 (this

mesh and the corresponding coarsening strategies will be used in numerical example II in Section 4.4.1.2) and in Figures 4.3 and 4.4 we show two different coarsening strategies with $N = 7$ obtained from seven and four seed points.

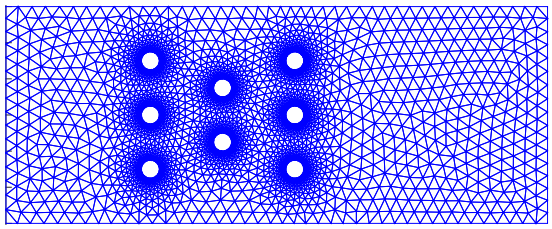
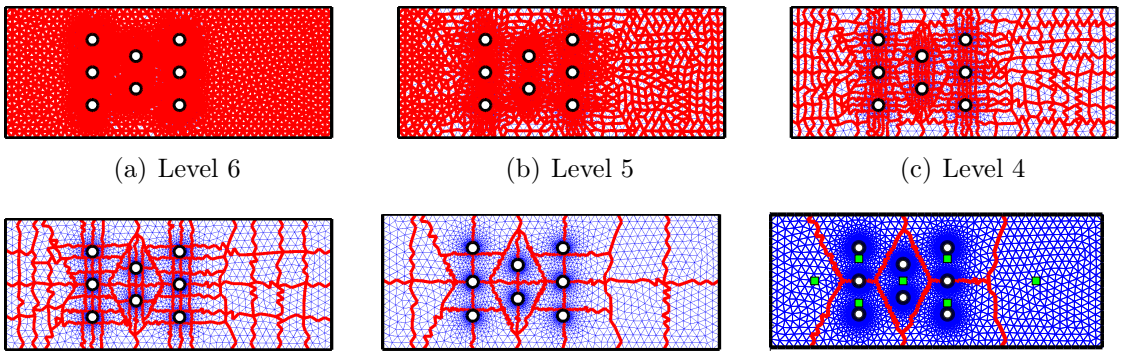


Figure 4.2: Example II: Unstructured mesh for a rectangular box with eight holes.

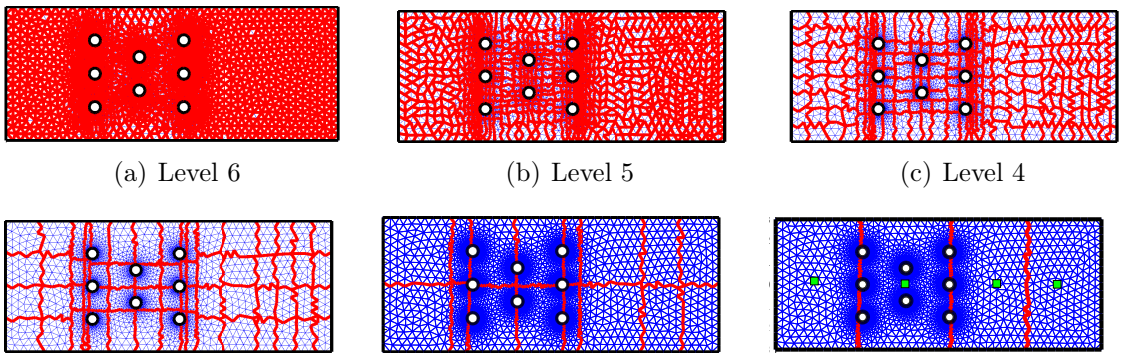


(a) Level 6 (b) Level 5 (c) Level 4

(d) Level 3 (e) Level 2 (f) Level 1

Figure 4.3: Example II. Coarsening strategy 1: the seed points are marked with green squares in level 1.

Part of our future work is to explore the coarsening strategies similar to the ones in [148, 147]. This may give better coarsened levels with the number of macro-elements reduced/increased by a constant factor between successive levels. The intergrid transfer operators necessary for moving residuals/errors between levels is described next.



(a) Level 6 (b) Level 5 (c) Level 4

(d) Level 3 (e) Level 2 (f) Level 1

Figure 4.4: Example II. Coarsening strategy 2: the seed points are marked with green squares in level 1.

4.3.2 Intergrid transfer operators

In standard nested multigrid algorithms, the intergrid transfer operators are straightforward. A popular approach is to take injection for the prolongation and its adjoint for the restriction. For a skeletal system care must be taken in the construction of these operators to ensure the convergence of the multigrid algorithm under consideration. In the following we propose "physics-based energy-preserving" operators using the fine scale DtN maps.

4.3.2.1 Prolongation

The prolongation operator, $I_k : M_{k-1} \longrightarrow M_k$, transfers the error from the grid level $k-1$ to the finer grid level k. Note that standard prolongation using injection would set the values on the interior edges to zero and thus does not work. For our multigrid algorithm it is defined as a function of the solution order p using DtN maps. In particular,

- for $p = 1$,

$$I_k := \begin{bmatrix} -A_{k,II}^{-1} A_{k,IB} J_k \\ J_k \end{bmatrix} \quad \text{for} \quad k = 2, \ldots, N, \tag{4.11}$$

- and for $p > 1$,

$$I_k := \begin{cases} J_N & \text{for} \quad k = N, \\ \begin{bmatrix} -A_{k,II}^{-1} A_{k,IB} J_k \\ J_k \end{bmatrix} & \text{for} \quad k = 2, \ldots, N-1. \end{cases} \tag{4.12}$$

Here, $J_k : M_{k-1} \longrightarrow M_k$ is the injection (interpolation). To understand the idea behind the prolongation operator, let us consider $p = 1$ in (4.11). First, through J_k, we interpolate the error from the grid level $k-1$ to obtain error on the boundary edges of the finer grid level k. Then we solve (via the first block in the definition of I_k) for the error on the interior edges as a function of the interpolated error on the boundary. For $p > 1$ the prolongation is defined in (4.12), namely, we use the same prolongation as in the case of $p = 1$ for all levels $k \leq N-1$ and interpolate error from piecewise linear polynomials at level $N-1$ to piecewise pth-order polynomials at level N.

4.3.2.2 Restriction

The restriction operator, $Q_{k-1} : M_k \longrightarrow M_{k-1}$, restricts the residual from level k to coarser level $k-1$. The popular idea is to define the restriction operator as the adjoint (with respect to the L^2-inner products $\langle \cdot, \cdot \rangle_{\mathcal{E}_k}$ and $\langle \cdot, \cdot \rangle_{\mathcal{E}_{k-1}}$ in M_k and M_{k-1}) of the prolongation operator, i.e., $Q_{k-1} = I_k^*$, and from our numerical studies (not shown here) it still works well. However, this purely algebraic procedure, though convenient, is not our desire. Here, we construct the restriction operator Q_{k-1} such that the coarse grid problem, via the Galerkin approximation, is exactly a discretized DtN problem on level $k-1$. To that end, let us define Q_{k-1} as

- for $p = 1$,

$$Q_{k-1} := \begin{bmatrix} -J_k^* A_{k,BI} A_{k,II}^{-1} & J_k^* \end{bmatrix} \quad \text{for} \quad k = 2, \ldots, N, \tag{4.13}$$

- for $p > 1$,

$$Q_{k-1} := \begin{cases} J_N^* & \text{for} \quad k = N, \\ \begin{bmatrix} -J_k^* A_{k,BI} A_{k,II}^{-1} & J_k^* \end{bmatrix} & \text{for} \quad k = 2, \ldots, N-1. \end{cases} \tag{4.14}$$

Note that if A is symmetric, then our definition of the restriction operator Q_{k-1} is indeed the adjoint of the prolongation operator I_k.

Given the prolongation and restriction operators, we obtain our coarse grid equation using the discrete Galerkin approximation [145]

$$Q_{k-1} A_k I_k \lambda_{k-1} = Q_{k-1} g_k, \tag{4.15}$$

where the coarse grid operator

$$A_{k-1} := Q_{k-1} A_k I_k, \tag{4.16}$$

for either $p = 1$ and $k \leq N$ or $p > 1$ and $k \leq N - 1$, reads

$$\begin{aligned} A_{k-1} = \begin{bmatrix} -J_k^* A_{k,BI} A_{k,II}^{-1} & J_k^* \end{bmatrix} \begin{bmatrix} A_{k,II} & A_{k,IB} \\ A_{k,BI} & A_{k,BB} \end{bmatrix} \begin{bmatrix} -A_{k,II}^{-1} A_{k,IB} J_k \\ J_k \end{bmatrix} \\ = J_k^* \left(A_{k,BB} - A_{k,BI} A_{k,II}^{-1} A_{k,IB} \right) J_k, \end{aligned} \tag{4.17}$$

and

$$A_{N-1} = J_N^* A_N J_N, \tag{4.18}$$

for $p > 1$ and $k = N$.

Energy preservation. In order for the multigrid algorithms to converge the intergrid operators should be constructed in such a way that the "energy" does not increase when transferring information between a level to a finer one [18, 74]. Note that

"energy" here need not necessarily be associated with some physical energy. Indeed, if we associate A_k with a bilinear form $a_k(.,.)$ such that $a_k(\kappa,\mu) = \langle A_k\kappa, \mu\rangle_{\mathcal{E}_k}\ \forall \kappa,\mu \in M_k$, where again $\langle .,.\rangle_{\mathcal{E}_k}$ represents the L^2-inner product on $\mathcal{E}_k$, then we call $a_k(\kappa,\kappa)$ the energy on level k associated with κ. Nonincreasing energy means

$$a_k(I_k\lambda, I_k\lambda) \le a_{k-1}(\lambda,\lambda) \quad \forall \lambda \in M_{k-1}, \quad \forall k = 2,3,\ldots,N.$$

Proposition 1 (Energy preservation)**.** *The proposed multigrid approach preserves the energy in the following sense:* $\forall k = 2,3,\ldots,N$,

$$a_k(I_k\lambda, I_k\lambda) = a_{k-1}(\lambda,\lambda) \quad \forall \lambda \in M_{k-1}. \tag{4.19}$$

Proof. We proceed first with $p = 1$. From the definition of A_k in (4.10) and the definition of the prolongation operator I_k in (4.11) we have

$$\begin{aligned}
&a_k(I_k\lambda, I_k\lambda)\\
&= \left\langle \left[\ -A_{k,II}^{-1}A_{k,IB}J_k\lambda,\ J_k\lambda\ \right], \begin{bmatrix} A_{k,II} & A_{k,IB} \\ A_{k,BI} & A_{k,BB} \end{bmatrix} \begin{bmatrix} -A_{k,II}^{-1}A_{k,IB}J_k\lambda \\ J_k\lambda \end{bmatrix} \right\rangle_{\mathcal{E}_k}\\
&= \left\langle \left(A_{k,BB} - A_{k,BI}A_{k,II}^{-1}A_{k,IB}\right) J_k\lambda, J_k\lambda \right\rangle_{\mathcal{E}_k}\\
&= \left\langle J_k^* \left(A_{k,BB} - A_{k,BI}A_{k,II}^{-1}A_{k,IB}\right) J_k\lambda, \lambda \right\rangle_{\mathcal{E}_{k-1}} = a_{k-1}(\lambda,\lambda),
\end{aligned}$$

where the last equality comes from the definition of the coarse grid operator A_{k-1} in (4.17). For $p > 1$, we need to prove (4.19) only for $k = N$, but this is straightforward since from (4.12), (4.14), and (4.18) we have

$$a_N(I_N\lambda, I_N\lambda) = \langle A_N J_N\lambda, J_N\lambda\rangle_{\mathcal{E}_k} = \langle J_N^* A_N J_N\lambda, \lambda\rangle_{\mathcal{E}_{k-1}} = a_{k-1}(\lambda,\lambda).$$

□

We now prove that the coarse grid operator (4.16) is also a discretized DtN map on every level.

Proposition 2. *At every level $k = 1, \ldots, N$, the Galerkin coarse grid operator* (4.16) *is a discretized DtN map on that level.*

Proof. The proof is a straightforward induction using the proposed coarsening strategy and the definition of the intergrid transfer operators. First, consider the case of $p = 1$. The fact that $A_N = A$ in (4.9) is a discretized DtN map on the finest level is clear by the definition of the hybridized methods. Assume at level k the operator A_k in (4.10) is a discretized DtN map. Taking $\lambda_{k,B} = J_k \lambda_{k-1}$ and condensing $\lambda_{k,I}$ out yield

$$\left(A_{k,BB} - A_{k,BI} A_{k,II}^{-1} A_{k,IB}\right) J_k \lambda_{k-1} = g_{k,B} - A_{k,BI} A_{k,II}^{-1} g_{k,I}$$

which, after restricting on the coarse space M_{k-1} using J_k^*, becomes

$$J_k^* \left(A_{k,BB} - A_{k,BI} A_{k,II}^{-1} A_{k,IB}\right) J_k \lambda_{k-1} = J_k^* \left(g_{k,B} - A_{k,BI} A_{k,II}^{-1} g_{k,I}\right),$$

which is exactly the coarse grid equation (4.15). That is, A_{k-1} is the Schur complement obtained by eliminating all the trace unknowns inside all the macro-elements on level $k-1$. By definition, the coarse grid operator A_{k-1} on level $(k-1)$ is also a discrete DtN map.

For the case of $p > 1$, it is sufficient to show that A_{N-1} is a discrete DtN map, but this is clear by: (1) taking $\lambda = J_N \lambda_{N-1}$ in (4.9), (2) restricting both sides to M_{N-1} using J_N^*, (3) recalling that A_N is a discretized DtN map, and 4) observing that the resulting equation coincides with the coarse grid equation (4.18). □

4.3.3 Smoothing

Let us denote the smoothing operator at level k by G_{k,m_k}, where m_k stands for the number of smoothing steps performed at level k. We take m_k to satisfy

$$\beta_0 m_k \leq m_{k-1} \leq \beta_1 m_k,$$

with $\beta_1 \geq \beta_0 > 1$. That is, the number of smoothing steps are increased as the mesh is coarsened. The reason for this choice is based on the theoretical analysis for nonnested multigrid methods in [20]. However, as numerically shown later in Section 4.4.1.5, even a constant number of smoothing steps at all levels, e.g., two, works well.

4.3.4 Local correction operator

To motivate the need for a local correction operator, let us consider the following decomposition of the skeletal space $M_k = \overline{M}_k \oplus \hat{M}_k$ such that

$$\lambda_k = \overline{\lambda}_k + \hat{\lambda}_k, \quad \overline{\lambda}_k \in \overline{M}_k \text{ and } \hat{\lambda}_k \in \hat{M}_k,$$

where $\overline{\lambda}_k = \overline{\lambda}_{k,I}$ is given by the local correction $T_k : M_k \longrightarrow M_k$,

$$\overline{\lambda}_k := T_k \begin{bmatrix} g_{k,I} \\ 0 \end{bmatrix} := \begin{bmatrix} A_{k,II}^{-1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} g_{k,I} \\ 0 \end{bmatrix}, \tag{4.20}$$

and the "global component" $\hat{\lambda}_k = \begin{bmatrix} \hat{\lambda}_{k,I} & \hat{\lambda}_{k,B} \end{bmatrix}^T$ by

$$\begin{bmatrix} A_{k,II} & A_{k,IB} \\ A_{k,BI} & A_{k,BB} \end{bmatrix} \begin{bmatrix} \hat{\lambda}_{k,I} \\ \hat{\lambda}_{k,B} \end{bmatrix} = \begin{bmatrix} 0 \\ g_{k,B} - A_{k,BI} A_{k,II}^{-1} g_{k,I} \end{bmatrix}.$$

Let us now consider $p = 1$. Standard two-grid cycles, for example, include: (1) smoothing iterations on the system (4.9) for the fine grid, (2) restricting the residual to the coarse grid, (3) performing the coarse grid correction, and (4) prolongating the error back to the fine grid. However, with the prolongation operator defined in (4.11), only $\hat{\lambda}_k$ can be recovered directly. In other words, the burden of capturing $\overline{\lambda}_k$ is left for the smoother and/or multigrid iterations. Indeed, in our numerical studies we found that this standard algorithm takes more multigrid iterations and still converges with Gauss–Seidel and Chebyshev accelerated Jacobi smoothers. However, it diverges with the block-Jacobi smoother at low orders as shown in Table 4.11.

This issue can be fixed using the concept of subspace correction [162, 145]. We can perform a local correction in the subspace $\overline{M}_k$, as in (4.20), either before or

after the coarse grid correction (or both if symmetry is desirable). With this local correction, we have observed that even with point Jacobi smoothers the multigrid algorithm converges. It is important to point out that the application of the local correction operator T_k is completely local to each macro-element at the $(k-1)$th level and thus can be carried out in parallel.

4.3.5 Relationship to AMG operators

The intergrid transfer operators in section 4.3.2 and the local correction operator in section 4.3.4 are closely related to the ideal interpolation and smoothing operators in the AMG literature [145]. However, the ideal operators in AMG lead to a direct solver strategy, namely, nested dissection [67], and suffer from large memory requirements. This renders the algorithm impractical for large scale simulations especially in three dimensions. In that respect, the coarsening in our operators is the key as it leads to an $\mathcal{O}(N)$ iterative algorithm and can be applied to large scale problems. In fact, this approach is pursued under the name of Schur complement multigrid methods in [149] and similar works [124, 123, 47, 46], mostly in the context of finite differences and finite volumes. Here we apply similar ideas for high-order hybridized finite element methods. Also our approach is more general in the sense that it can be applied to any unstructured mesh, whereas all the previous works deal with structured meshes. As shown in Figure 4.2, it does not require the meshes to be nested. In fact, our approach can be considered as a combination of AMG and geometric multigrid methods. As such it benefits from the robustness of AMG and the fixed coarse grid construction costs of geometric multigrid.

4.3.6 Multigrid V-cycle algorithm

We are now in position to define our multigrid algorithm. We begin with a fixed point scheme on the finest level:

$$\lambda^{i+1} = \lambda^i + B_N(r_N), \quad i = 0, \ldots.$$

where $r_N = g_N - A_N\lambda^i$ and the action of B_N on a function/vector is defined recursively using the multigrid algorithm 3.

Algorithm 3 Multigrid v-cycle algorithm

1: *Initialization*:
$e^{\{0\}} = 0$,
2: *Presmoothing*:
$e^{\{1\}} = e^{\{0\}} + G_{k,m_k}\left(r_k - A_k e^{\{0\}}\right)$,
3: *Local correction*:
$e^{\{2\}} = e^{\{1\}} + T_k\left(r_k - A_k e^{\{1\}}\right)$,
4: *Coarse grid correction*:
$e^{\{3\}} = e^{\{2\}} + I_k B_{k-1}\left(Q_{k-1}\left(r_k - A_k e^{\{2\}}\right)\right)$,
5: *Local correction*:
$e^{\{4\}} = e^{\{3\}} + T_k\left(r_k - A_k e^{\{3\}}\right)$,
6: *Postsmoothing*:
$B_k\left(r_k\right) = e^{\{5\}} = e^{\{4\}} + G_{k,m_k}\left(r_k - A_k e^{\{4\}}\right)$.

At the coarsest level M_1, we set $B_1 = A_1^{-1}$ and the inversion is computed using a direct solver. Note that both local correction steps 3 and 5 are presented for the sake of symmetry of the algorithm. In practice, we use either step 3 or step 5. For example, numerical results in Section 4.4 use only step 3.

4.4 Numerical results

In this section, we study the performance of multigrid both as a solver and as a left preconditioner. Since the global trace system of the NIPG-H and IIPG-H methods is not symmetric, we use preconditioned GMRES for all the methods to

enable a direct comparison. For all the numerical examples, (1) the stopping tolerance is taken as 10^{-9} for the normalized residual (normalized by the norm of the right-hand side of (4.9)), and (2) following [20] we choose the smoothing parameters β_0 and β_1 (see Section 4.3.3) as 2. In all the tables, " $*$ " stands for either the divergence of the GMRES/multigrid solver or 200 iterations were already taken but the normalized residual was still larger than the tolerance.

In examples 1-5 in Sections 4.4.1.1-4.4.1.5, we consider HDG and hybridized IPDG methods.

4.4.1 Elliptic equation

4.4.1.1 Example I: Poisson equation in the unit square

In this first example, we consider the Poisson equation in the unit square. We take the exact solution to be of the form $q = xye^{x^2y^3}$. Dirichlet boundary condition using the exact solution are applied directly (strongly) via the trace unknowns. We consider hybridized DG methods and study the performance of multigrid both as a solver and as a preconditioner. The number of levels in the multigrid hierarchy and the corresponding number of quadrilateral elements are shown in Table 4.1.

Levels	2	3	4	5	6	7
Elements	16	64	256	1024	4096	16384

Table 4.1: Example I: The multigrid hierarchy.

We consider the following four different smoothers and compare their performance for each of the hybridized DG methods.

- Damped point-Jacobi with the relaxation parameter $\omega = 2/3$ [142].

- Chebyshev accelerated point-Jacobi method. This smoother requires the estimates of extreme eigenvalues and we compute an approximate maximum eigenvalue Λ_{max} with $\mathcal{O}\left(10^{-2}\right)$-accuracy. Following [1], we estimate the smallest eigenvalue using $\Lambda_{max}/30$.

- Lower-upper symmetric point Gauss-Seidel method with one forward and one backward sweep during each iteration, and we name this smoother as LU-SGS for simplicity.

- Block-Jacobi smoother where one block consists of all the degrees of freedom corresponding to an edge $e \in \mathcal{E}_k$ at the kth level. Here, we do not use any damping as it does not make any difference in the number of iterations in our numerical studies.

For the stabilization parameters, we consider a mesh-independent value $\tau = 1$ and a mesh-dependent form $\tau = 1/h_{min}$ for HDG. The former corresponds to the upwind HDG proposed in [25, 26]. Following [129], we take the stabilization to be $\tau = (p+1)(p+2)/h_{min}$ for the hybridized IPDG schemes. In later examples, when the permeability $\mathbf{K}$ is spatially varying we also consider τ as a function of $\mathbf{K}$. We refer to [40, 25, 37] for the h-convergence order of HDG schemes and [129] for the IPDG schemes.

In Tables 4.2-4.5, we study the performance of multigrid as a solver and as a preconditioner for HDG with stabilization $\tau = 1/h_{min}$. We now present a few important observations from these tables. The stabilization $\tau = 1/h_{min}$ gives h-scalable results with all the smoothers, i.e., the number of required multigrid/GMRES iterations is almost unchanged when the mesh is refined. However, the actual number of iterations depends on the effectiveness of the smoother under consideration. As can

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	14	14	14	15	15	15	6	8	8	8	8	8
2	14	14	15	15	15	15	8	8	8	8	8	8
3	15	16	16	16	16	16	8	9	9	9	9	8
4	19	19	19	19	19	19	9	10	10	9	9	9
5	21	21	21	21	21	21	10	10	10	10	10	9
6	23	23	24	24	24	24	10	11	11	11	10	10
7	25	25	25	25	25	25	11	11	11	11	11	11
8	27	27	28	28	28	28	11	12	11	11	11	11
9	30	30	30	29	29	29	12	12	12	12	12	11
10	30	31	31	31	31	32	12	12	12	12	12	12

Table 4.2: Example I. HDG with stabilization $\tau = 1/h_{min}$: the number of iterations for multigrid with point-Jacobi smoother as solver and preconditioner.

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	13	14	14	14	14	14	6	8	8	8	8	8
2	9	10	10	10	10	10	6	7	7	7	7	7
3	27	29	30	31	31	31	12	12	12	12	12	12
4	31	29	29	29	28	28	12	12	12	11	11	11
5	31	32	32	32	32	32	13	13	12	12	12	12
6	34	33	32	32	32	31	13	13	12	12	12	11
7	32	32	32	32	32	32	13	12	12	12	12	12
8	33	33	33	33	33	33	13	12	12	12	12	12
9	33	33	33	33	33	33	13	13	13	12	12	12
10	33	33	33	33	33	33	13	13	13	12	12	12

Table 4.3: Example I. HDG with stabilization $\tau = 1/h_{min}$: the number of iterations for multigrid with Chebyshev accelerated point-Jacobi smoother as solver and preconditioner.

be seen in Table 4.2 with point-Jacobi smoother, the number of iterations increases with high-order solutions when multigrid is used as a solver. In contrast, the multigrid preconditioner is effective in keeping the number of GMRES iterations almost unchanged (the increase is negligible). The results in Table 4.3 shows that Chebyshev acceleration for point-Jacobi smoother behaves the same as the point-Jacobi smoother

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	5	5	5	5	5	5	4	4	4	5	5	5
2	5	5	5	5	5	5	4	4	4	4	4	4
3	5	6	6	6	6	6	5	5	5	5	5	5
4	6	6	6	6	6	6	5	5	5	5	5	5
5	7	7	7	7	7	7	5	5	6	6	6	5
6	7	7	8	8	8	8	5	6	6	6	6	6
7	8	8	8	8	8	8	6	6	6	6	6	6
8	8	8	9	9	9	9	6	6	6	6	6	6
9	9	9	9	9	9	9	7	7	7	7	7	7
10	9	9	10	10	10	10	7	7	7	7	7	7

Table 4.4: Example I. HDG with stabilization $\tau = 1/h_{min}$: the number of iterations for multigrid with LU-SGS smoother as solver and preconditioner.

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	7	7	8	8	8	8	4	5	6	6	6	6
2	6	7	8	8	9	9	4	5	6	6	6	6
3	8	9	9	9	9	9	6	6	6	6	6	6
4	9	10	10	10	10	10	6	7	7	7	7	7
5	11	12	12	12	12	12	6	8	8	8	8	7
6	12	12	13	13	13	13	7	8	8	8	8	8
7	13	14	14	14	14	15	7	8	8	8	8	8
8	14	15	15	15	15	15	8	9	9	9	9	8
9	15	16	16	16	17	17	8	9	9	9	9	9
10	16	17	17	17	17	17	8	9	9	9	9	9

Table 4.5: Example I. HDG with stabilization $\tau = 1/h_{min}$: the number of iterations for multigrid with block-Jacobi smoother as solver and preconditioner.

in terms of scalability. However, in terms of the number of iterations it takes either more (for multigrid as solver) or about the same (for multigrid preconditioned GMRES). Table 4.4 shows that LU-SGS is the most effective smoother, requiring the least number of multigrid/GMRES iterations. Moreover, the number of both multigrid and GMRES iterations are almost constant as either the mesh is refined or the solution

order increases. However, it should be pointed out that the LU-SGS smoother requires twice the amount of work compared to point-Jacobi for each iteration due to one forward and one backward sweep. Table 4.5 shows that block-Jacobi is the second best smoother in terms of the number of iterations; otherwise its scalability behavior is similar to the point-Jacobi smoother as the mesh or the solution order is refined.

	MG as solver						MG with GMRES					
p	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	6	11	21	40	76	145	4	7	11	15	21	28
2	8	15	28	53	101	193	6	8	12	17	23	31
3	10	18	32	60	113	*	7	9	13	18	24	32
4	11	19	34	62	117	*	7	10	13	18	24	32
5	13	21	36	66	124	*	8	10	14	18	24	32
6	13	22	37	68	126	*	7	10	14	18	24	32
7	15	23	39	71	131	*	8	11	14	18	24	32
8	15	24	40	72	132	*	8	11	14	18	24	31
9	16	25	42	74	136	*	8	11	14	18	24	31
10	17	26	43	75	137	*	8	11	14	18	24	31

Table 4.6: Example I. HDG with stabilization $\tau = 1$: the number of iterations for multigrid with LU-SGS smoother as solver and preconditioner.

	MG as solver						MG with GMRES					
p	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	5	8	15	27	51	97	4	6	9	12	17	23
2	5	7	9	14	24	42	4	5	7	9	12	15
3	8	9	9	9	15	25	5	6	6	7	9	11
4	9	10	10	10	10	10	6	7	7	7	8	8
5	11	11	12	12	12	12	6	7	7	8	7	7
6	12	12	13	13	13	13	7	8	8	8	8	8
7	13	14	14	14	14	14	7	8	8	8	8	8
8	14	15	15	15	15	15	8	9	9	9	9	8
9	15	16	16	16	17	17	8	9	9	9	9	9
10	16	17	17	17	17	17	8	9	9	9	9	9

Table 4.7: Example I. HDG with stabilization $\tau = 1$: the number of iterations for multigrid with block-Jacobi smoother as solver and preconditioner.

In Tables 4.6 and 4.7, we present results for the multigrid solver and GMRES with multigrid preconditioner when $\tau = 1$ is used for the HDG discretization. With the point-smoothers[3] (Table 4.6), the number of iterations for the multigrid solver scales like $\mathcal{O}(1/h)$ (since the number of iterations is nearly doubled each time h reduces by half). Even for GMRES with multigrid as preconditioner, we do not obtain h-scalability with point smoothers. On the other hand, the block-Jacobi smoother in Table 4.7 for $p \geq 4$ provides h-scalable results and with solution orders greater than 5 the results are exactly the same as those for $\tau = 1/h_{min}$. With respect to solution orders, similar to the case of $\tau = 1/h_{min}$, we do not obtain perfect p-scalability with any smoother though the growth of the number of iterations is very slow with block-Jacobi smoother.

	MG as solver						MG with GMRES					
p	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	10	12	12	12	12	12	6	8	8	8	8	8
2	8	9	9	9	9	9	6	6	6	6	6	6
3	10	10	10	11	11	11	6	7	7	7	7	7
4	11	12	12	12	13	13	7	8	8	8	8	8
5	13	14	14	14	14	14	7	8	8	8	8	8
6	15	15	16	16	16	16	9	9	9	9	9	9
7	16	17	18	18	18	18	9	9	9	9	9	9
8	18	19	19	20	20	20	10	10	10	10	10	10
9	20	21	21	21	21	22	10	11	10	10	10	10
10	21	22	23	23	23	23	11	11	11	11	11	11

Table 4.8: Example I. NIPG-H: the number of iterations for multigrid with point-Jacobi smoother as solver and preconditioner.

We present in Tables 4.8-4.10 the results for the hybridized NIPG-H scheme with $\tau = (p+1)(p+2)/h_{min}$. As can be seen, the number of iterations for both

[3]We do not show results for point-Jacobi and Chebyshev accelerated point-Jacobi smoothers as the behavior of the number of iterations is very similar to that of the LU-SGS smoother.

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	4	4	4	4	4	4	3	4	4	4	4	5
2	4	4	4	5	5	5	4	4	4	4	4	4
3	4	5	5	5	5	5	4	4	4	5	5	4
4	5	5	5	5	5	6	4	4	5	5	5	5
5	5	6	6	6	6	6	4	5	5	5	5	5
6	6	6	6	6	6	6	5	5	5	5	5	5
7	6	7	7	7	7	7	5	5	5	5	5	5
8	7	7	7	7	7	7	5	6	6	6	6	6
9	7	7	8	8	8	8	5	6	6	6	6	6
10	7	8	8	8	8	8	6	6	6	6	6	6

Table 4.9: Example I. NIPG-H: the number of iterations for multigrid with LU-SGS smoother as solver and preconditioner.

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	5	7	8	8	8	8	4	6	6	6	6	6
2	6	6	6	6	7	7	5	5	5	5	5	5
3	7	8	8	8	8	8	5	6	6	6	6	6
4	9	9	10	10	10	10	6	7	7	7	7	7
5	10	11	11	11	11	11	6	7	7	7	7	7
6	11	12	12	12	12	12	7	8	8	8	8	7
7	12	13	13	13	13	13	7	8	8	8	8	8
8	13	14	14	14	14	14	7	8	8	8	8	8
9	14	15	15	15	15	15	8	9	9	9	9	8
10	15	16	16	16	16	16	8	9	9	9	9	9

Table 4.10: Example I. NIPG-H: the number of iterations for multigrid with block-Jacobi smoother as solver and preconditioner.

multigrid as a solver and as a preconditioner is similar to those with HDG using $\tau = 1/h_{min}$. The results using IIPG-H and SIPG-H are similar and hence are omitted for brevity. For all of the IPDG schemes, using the Chebyshev acceleration increases the number of iterations and we do not include these results here.

Finally, we test the performance of multigrid without the local correction op-

erator in step 3 of Algorithm 3. The results are shown for HDG with $\tau = 1/h_{min}$ and block-Jacobi smoother in Table 4.11. As can be seen, for $p \leq 3$ multigrid as solver and also as a preconditioner fails to converge for fine meshsizes. For $p \geq 4$ we see scalable results although the number of iterations are slightly more than that in Table 4.5 with local correction. We also see an odd-even behavior, with even orders giving better iteration counts than odd ones. For $\tau = 1$, we observed similar results and hence not shown. For the hybridized IPDG schemes, with NIPG-H and IIPG-H from $p \geq 2$ onwards we observed scalable results whereas for SIPG-H for all orders we obtained scalability without local correction. The iteration counts are again slightly more than the ones obtained with local correction. With increase in order, since block-Jacobi smoother gets stronger, even without the local correction operator it is sufficient to give scalable results. However, in order to provide more robustness and since the local correction operator is inexpensive, in the following test cases we always include it in the step-3 of Algorithm 3.

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	*	*	*	*	*	*	8	18	56	*	*	*
2	8	13	21	26	27	28	6	7	9	11	11	11
3	16	60	*	*	*	*	7	11	20	39	97	*
4	10	11	12	13	13	13	7	7	8	8	8	8
5	11	12	12	12	12	12	6	8	8	8	8	7
6	12	12	13	13	13	13	7	8	8	8	8	8
7	15	20	23	25	26	26	9	10	11	11	11	11
8	14	15	15	15	15	15	8	9	9	9	9	8
9	15	16	16	16	17	17	8	9	9	9	9	9
10	18	19	19	20	20	20	9	10	10	10	10	10

Table 4.11: Example I. HDG with stabilization $\tau = 1/h_{min}$: the number of iterations for multigrid with block-Jacobi smoother as solver and preconditioner without local correction.

In summary, for HDG with $\tau = 1/h_{min}$ and multigrid preconditioned GMRES,

almost perfect *hp*-scalability has been observed for all the smoothers, while for $\tau = 1$ this scalability holds with only block smoothers together with high solution orders. All the hybridized IPDG schemes give scalable results with $\tau = (p+1)(p+2)/h_{min}$ and the number of iterations is very similar to that of HDG with $\tau = 1/h_{min}$. Of all the smoothers considered in this paper, LU-SGS seems to be the best smoother in terms of the number of iterations. However, block-Jacobi smoother, the second best, is more convenient from the implementation point of view (especially on parallel computing systems), and for that reason we show numerical results only for block-Jacobi smoother from now on. Since the Chebyshev acceleration does not seem to improve the performance of multigrid by a significant margin, we will not use it in the subsequent sections. We would like to mention that the performance of our multigrid approach in terms of iteration counts is better or at least comparable to other existing multigrids proposed in the literature [74, 41, 91, 33]. Since the time to solution depends on many other factors such as the choice of smoother, the number of smoothing steps, and the architecture of the machine we are not able to comment on that aspect at this moment. However, in future work we plan to carry out a detailed study between different multigrid algorithms for hybridized methods with respect to simulation of challenging problems.

4.4.1.2 Example II: Unstructured mesh

The goal of this section is to test the robustness of the multigrid algorithm for a highly unstructured mesh. To that end consider the mesh in Figure 4.2, which consists of 6699 simplices with an order of magnitude variation in the diameter of elements i.e. $h_{max} \approx 10 h_{min}$. Unlike structured meshes, for unstructured meshes with local clustering of elements (local refinements) it is not straightforward to select the number of levels and the "best" coarsening strategy that well balances the number

of elements (in the finer level) for each macro-element. Ideally, given a number of levels we wish to minimize the number of multigrid/GMRES iterations. Here, we compare two coarsening strategies with the same number of levels and study the performance of multigrid as a solver and as a preconditioner. In our future work, we will explore the coarsening strategies similar to the ones in [148, 147]. Figures 4.3 and 4.4 show different levels corresponding to the two coarsening strategies. The total number of levels in both the strategies is seven and the last level corresponds to the original fine mesh as shown in Figure 4.2. For solution orders $p > 1$, again, we first restrict the residual to $p = 1$ and then carry out the geometric coarsening. There are seven macro-elements in the first level for coarsening strategy 1, and four macro-elements for strategy 2.

We take the zero Dirichlet boundary condition, $\mathbf{K} = \mathbf{I}$ ($\mathbf{I}$ is the identity), and $f = 1$ for this example. We now study the performance of HDG and hybridized IPDG schemes. In Table 4.12, we compare the number of iterations taken for HDG with $\tau_1 = 1$ and $\tau_2 = 1/h_{min}$ and solution orders $p = 1, \ldots, 8$. As can be seen, for $p \in \{1, 2, 3\}$, using τ_1 takes less number of iterations, and for $p \geq 4$ both stabilization parameters require almost the same iteration counts. For coarsening strategy 2, since the coarsening happens slightly more aggressive (i.e., fewer macro-elements at any level) the iteration counts in general are higher. However, using multigrid as a preconditioner for GMRES the difference in the iteration counts for the two strategies is negligible.

In Table 4.13, we consider the NIPG-H scheme with $\tau_1 = 1/h_{min}$ and $\tau_2 = (p+1)(p+2)/h_{min}$. As can be seen, the multigrid solver diverges for many values of solution order p. The iterations for τ_1 are less than that for τ_2. We observe that the iteration counts of multigrid preconditioned GMRES with τ_1, except for solution orders $p = 2$ and $p = 3$ (which require more iterations), are similar to those for HDG.

When multigrid is used as a preconditioner, except for solution orders equal to two and three, both the coarsening strategies give similar iteration counts with τ_1. It turns out that the iteration counts for IIPG-H and SIPG-H are higher than that of HDG and NIPG-H and hence are omitted for brevity.

Within the settings of this example we conclude that *multigrid*, both *as a solver and as a preconditioner, is more effective for HDG than for hybridized IPDG schemes*. It is also relatively more robust with respect to the coarsening strategies and values of stabilization parameter when used as a preconditioner.

	MG as solver				MG with GMRES			
p	Coarsening strategy 1		Coarsening strategy 2		Coarsening strategy 1		Coarsening strategy 2	
	τ_1	τ_2	τ_1	τ_2	τ_1	τ_2	τ_1	τ_2
1	25	*	41	108	10	21	13	18
2	20	63	33	37	10	12	12	13
3	23	24	39	41	10	11	13	14
4	27	27	43	45	11	11	14	14
5	30	30	48	49	11	12	15	15
6	33	33	51	52	12	12	15	15
7	36	36	55	55	13	13	16	16
8	38	38	58	58	13	13	16	16

Table 4.12: Example II. HDG with $\tau_1 = 1$, $\tau_2 = 1/h_{min}$: the number of iterations for multigrid as solver and preconditioner for both coarsening strategies.

4.4.1.3 Example III: Smoothly varying permeability

In this example we consider a smoothly varying permeability of the form

$$\mathbf{K} = \kappa \mathbf{I},$$

where $\kappa = 1 + 0.5\sin(2\pi x)\cos(3\pi y)$. The domain considered is a circle and we take the exact solution again to be of the form $q = xye^{x^2y^3}$. The forcing and the Dirichlet boundary condition are chosen corresponding to the exact solution. The number of

	MG as solver				MG with GMRES			
p	coarsening strategy 1		coarsening strategy 2		coarsening strategy 1		coarsening strategy 2	
	τ_1	τ_2	τ_1	τ_2	τ_1	τ_2	τ_1	τ_2
1	16	*	29	*	9	133	11	112
2	*	*	*	*	136	*	*	*
3	*	*	*	*	42	42	49	31
4	22	*	37	*	10	71	13	62
5	25	67	41	95	11	16	14	19
6	28	*	46	*	11	28	14	27
7	31	71	49	99	12	17	15	21
8	34	121	53	106	12	20	15	21

Table 4.13: Example II. NIPG-H with $\tau_1 = 1/h_{min}$, $\tau_2 = (p+1)(p+2)/h_{min}$: the number of iterations for multigrid as solver and preconditioner for both coarsening strategies.

levels and the corresponding number of triangular elements in the multigrid hierarchy are shown in Table 4.14.

Levels	2	3	4	5	6	7
Elements	84	348	1500	6232	25344	102160

Table 4.14: Example III: The multigrid hierarchy.

	MG as solver						MG with GMRES					
p	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	18	20	23	21	24	27	9	10	11	11	11	12
2	13	15	18	17	19	20	9	9	10	10	11	11
3	16	18	21	20	23	24	10	10	11	11	11	12
4	18	21	23	23	25	27	10	11	12	12	12	12
5	20	24	26	25	28	29	11	12	12	12	13	13
6	22	27	28	28	31	32	11	13	13	13	13	13
7	24	30	30	30	33	34	12	13	13	13	14	14
8	26	32	32	32	35	36	12	13	14	14	14	14

Table 4.15: Example III. HDG with stabilization $\tau = 1$: the number of iterations for multigrid as solver and preconditioner.

Table 4.15 shows the number of multigrid and GMRES iterations for HDG

with $\tau = 1$. Note that perfect h-scalability with multigrid as a solver is not observed, and this is mainly due to the fact that the number of elements between successive levels is not exactly increased/decreased by a constant[4]. Again, we see a little growth in the number of iterations as p increases for the multigrid solver, while for GMRES with multigrid preconditioner we obtain almost perfect hp-scalability. We have also conducted numerical examples with $\tau = 1/h_{min}$, $\tau = \kappa/h_{min}$ and observed that the iteration counts (not shown here) are similar to those with $\tau = 1$.

Next we consider the IIPG-H scheme with $\tau = \kappa(p+1)(p+2)/h_{min}$. Table 4.16 shows that the iteration counts are comparable with HDG results in Table 4.15 for GMRES with multigrid preconditioner. The multigrid solver for IIPG-H, on the other hand, has slightly less iteration counts. An extra penalization factor of 1.5 is necessary to obtain well-posed linear systems for SIPG-H, but otherwise the corresponding results for NIPG-H and SIPG-H are similar and hence are omitted.

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	14	15	17	16	18	20	8	9	10	10	10	10
2	12	13	16	15	17	18	8	9	9	9	10	10
3	14	16	18	18	20	21	9	10	10	10	11	11
4	17	19	21	21	23	24	10	10	11	11	12	12
5	19	21	23	23	25	26	11	11	12	12	12	12
6	21	24	26	25	28	29	11	12	12	12	13	13
7	22	26	27	27	30	31	11	13	13	13	13	13
8	24	29	29	29	32	33	12	13	13	13	13	14

Table 4.16: Example III. IIPG-H with stabilization $\tau = \kappa(p+1)(p+2)/h_{min}$: the number of iterations for multigrid as solver and preconditioner.

[4]This, as argued before, is not trivial for unstructured meshes.

4.4.1.4 Example IV: Highly discontinuous permeability

In this example, we test the robustness of the multigrid solver on a highly discontinuous and spatially varying (six orders of magnitude) permeability field given by $\mathbf{K} = \kappa\mathbf{I}$ with

$$\kappa = \begin{cases} 10^6, & (x,y) \in (0,0.56) \times (0,0.56) \quad \text{or} \quad (x,y) \in (0.56,1) \times (0.56,1), \\ 1, & \text{otherwise}, \end{cases}$$

in the unit square.

We choose the zero Dirichlet boundary condition and $f = 1$. In Table 4.17, we study the performance of multigrid as a solver and as a preconditioner for GMRES using HDG with $\tau = \kappa/h_{min}$. The number of elements and the number of levels are same as in example I. Similar to the previous examples, almost perfect h- and p-scalabilities are observed with multigrid preconditioned GMRES, whereas the number of iterations increases with the solution order p for the multigrid solver. For $\tau = 1/h_{min}$ and $\tau = 1$ the multigrid solver takes too many iterations to converge for finer meshes, and for that reason we report the iteration counts only for multigrid preconditioned GMRES in Table 4.18. Clearly, the number of GMRES iterations is

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	6	8	8	9	9	9	2	5	5	5	5	5
2	6	7	9	10	10	11	3	4	5	5	5	5
3	8	9	10	10	10	10	3	5	6	6	5	5
4	10	11	11	11	11	11	4	6	6	6	6	5
5	11	13	13	13	13	13	4	6	7	7	6	6
6	13	14	14	14	14	14	4	6	7	7	7	6
7	14	15	16	16	15	15	5	7	7	7	7	7
8	15	16	17	17	17	17	5	7	8	8	7	7

Table 4.17: Example IV. HDG with stabilization $\tau = \kappa/h_{min}$: the number of iterations for multigrid as solver and preconditioner.

p	$\tau = 1/h_{min}$						$\tau = 1$					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	3	5	6	8	11	5	2	5	8	11	15	15
2	3	5	6	5	5	5	3	5	6	7	8	11
3	3	5	6	6	5	5	3	4	5	6	7	8
4	4	5	6	6	6	5	4	5	6	6	7	7
5	4	5	7	7	6	6	4	5	7	7	6	6
6	4	6	7	7	7	6	4	6	7	7	7	6
7	4	6	7	7	7	7	4	6	7	7	7	7
8	4	6	8	8	7	7	4	6	8	8	7	7

Table 4.18: Example IV. HDG with stabilizations $\tau = 1/h_{min}$ and $\tau = 1$: the number of iterations for multigrid preconditioned GMRES.

almost insensitive to the values of the stabilization parameter τ, and the results for $\tau = 1/h_{min}$ in columns $2-7$ of Table 4.18 are very similar to columns $8-13$ of Table 4.17 for $\tau = \kappa/h_{min}$. Columns $8-13$ of Table 4.18 shows that, again with $\tau = 1$, high-order solutions provide both h- and p-scalabilities and the results in these cases are similar to those with $\tau = 1/h_{min}$ (see the last four rows of Table 4.18).

Table 4.19 shows the performance of our multigrid algorithm as a solver and as a preconditioner for the SIPG-H scheme with $\tau = \kappa(p+1)(p+2)/h_{min}$. The results

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	8	9	10	11	11	12	3	6	6	6	6	6
2	6	8	9	9	10	10	4	5	5	5	5	5
3	9	10	10	10	10	11	4	6	6	6	6	5
4	10	11	12	12	12	12	4	6	7	6	6	6
5	12	13	13	13	13	13	4	6	7	7	7	6
6	13	14	15	15	15	15	5	7	7	7	7	7
7	14	16	16	16	16	16	5	7	8	8	7	7
8	15	17	17	17	17	17	5	7	8	8	7	7

Table 4.19: Example IV. SIPG-H with $\tau = \kappa(p+1)(p+2)/h_{min}$: the number of iterations for multigrid as solver and preconditioner.

for NIPG-H and IIPG-H are almost identical and are omitted. As can be observed, as a preconditioner for GMRES the multigrid algorithm is both h- and p-scalable for all the hybridized DG methods.

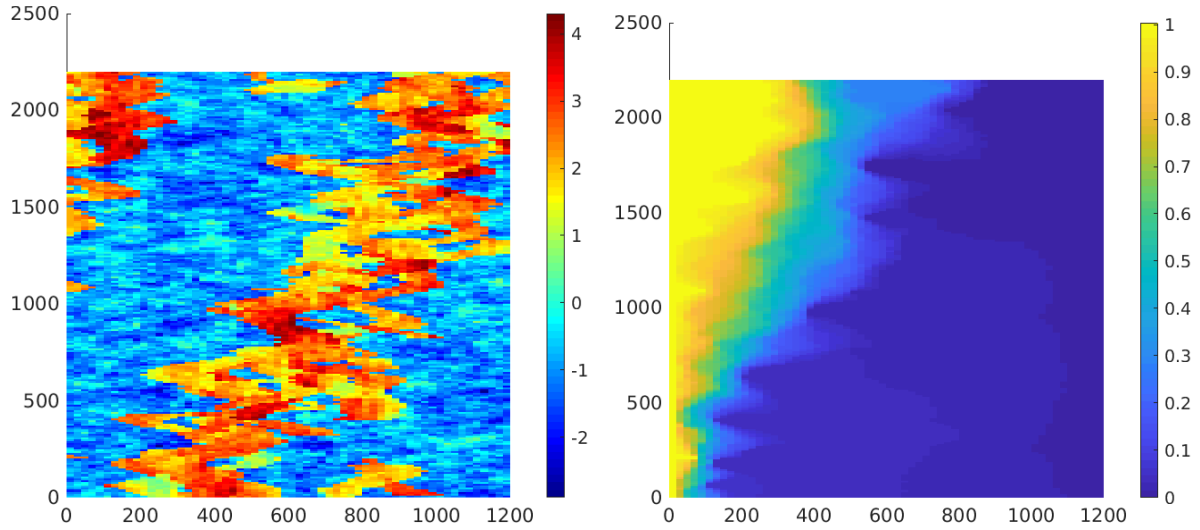
4.4.1.5 Example V: SPE10 test case

In this example, we consider the benchmark problem Model 2 from the Tenth Society of Petroleum Comparative Solution Project (SPE10) [35]. We consider the permeability field $\mathbf{K} = \kappa\mathbf{I}$, where κ corresponding to the 75th layer is shown on the left of Figure 4.5. The permeability field varies by six orders of magnitude and is highly heterogeneous which gives rise to extremely complex velocity fields. The domain is 1200×2200 $[ft^2]$. The mesh has 60×220 quadrilateral elements and the element edges align with the discontinuities in the permeability. We choose $f = 0$, which corresponds to no source or sink. For the boundary conditions we take the pressure (q) on the left and right faces to be 1 and 0, respectively. On the top and bottom faces no flux boundary condition $\mathbf{u} \cdot \mathbf{n} = 0$ is applied. From extensive numerical examples we have observed that the multigrid hierarchy with seven levels results in the least number of GMRES iterations. From numerical examples I-IV, we see that HDG method is relatively the most robust and scalable. In addition, it provides simultaneous approximations for both velocity and pressure. Thus, we consider only HDG for this example.

Since multigrid as a solver either converges very slowly or diverges for a number of cases in this example, we report results exclusively for the multigrid preconditioned GMRES. The pressure field is shown on the right of Figure 4.5 for $p = 1$. In Table 4.20 are the number of iterations for solution orders $p \in \{1, 2, 3, 4\}$ with $\tau = 1/h_{min}$ and $\tau = 1$. The second row shows that, for $p = 2$, the iterations are much larger compared to other solution orders. At the time of writing, we had not yet found

the reason for this behavior. For other solution orders, using $\tau = 1$ results in more iterations for $p = 1$ and less for $p = 3, 4$ compared to $\tau = 1/h_{min}$. Again, we like to emphasize that in this example we completely avoid upscaling of the permeability field as our multigrid algorithm is based only on the fine scale DtN maps.

The results in columns 2 and 3 of Table 4.20 correspond to geometrically increasing smoothing steps with respect to levels as explained in Section 4.3.3. In columns 4 and 5, we simply take two pre- and postsmoothing steps in all levels and, as can be seen, the iteration counts are marginally different compared to those resulted from the increasing smoothing steps. This implies that we can achieve similar accuracy with less computational cost using constant (here two) pre- and postsmoothing steps in all levels.



(a) Permeability field (b) Pressure field

Figure 4.5: Example V: Permeability (κ) in log scale (left) and pressure field (right) using solution order $p = 1$.

p	Varying smoothing		2 pre- and postsmoothing	
	$\tau = 1/h_{min}$	$\tau = 1$	$\tau = 1/h_{min}$	$\tau = 1$
1	47	78	51	80
2	81	*	72	*
3	58	40	56	48
4	48	39	49	45

Table 4.20: Example V. HDG with stabilizations $\tau = 1/h_{min}$, $\tau = 1$: the number of iterations for multigrid preconditioned GMRES with variable smoothing and constant smoothing.

4.4.2 Example VI: Convection-diffusion equation

In this example we apply the multigrid algorithm to the convection-diffusion equation (2.34) discretized with HDG. We consider a very simple constant velocity field of $\boldsymbol{\beta} = (1, 1)$ and $\nu = 0$. The exact solution is same as the one used in example I in section 4.4.1.1 and the forcing and boundary conditions are chosen based on it. The domain is the unit square discretized with quadrilateral elements and the multigrid hierarchy is given in Table 4.1. The smoother used is LU-SGS. The conclusion from this study is, for scalability in the diffusion-dominated regime with diffusion coefficient $\kappa > 0.01$ we need a mesh-dependent stabilization given by $\tau = \frac{1}{2h}\left(\sqrt{|\boldsymbol{\beta}\cdot\mathbf{n}|^2 + 4} - \boldsymbol{\beta}\cdot\mathbf{n}\right)$ in the flux (2.36), whereas in the convection dominated regime with $\kappa \leq 0.01$ pure upwind stabilization with $\tau = \frac{1}{2}\left(\sqrt{|\boldsymbol{\beta}\cdot\mathbf{n}|^2 + 4} - \boldsymbol{\beta}\cdot\mathbf{n}\right)$ is needed.

The iteration counts for representative cases in both regimes with these stabilization parameters is given in Tables 4.21 and 4.22. For more complex varying velocity fields we need robust line smoothers for the multigrid convergence and is a subject of future study.

	MG as solver						MG with GMRES					
p	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	5	5	5	5	5	5	4	4	5	5	5	5
2	5	5	5	5	5	5	4	4	4	4	4	4

Table 4.21: Example VI. $\kappa = 1$, HDG with $\tau = \frac{1}{2h}\left(\sqrt{|\boldsymbol{\beta}\cdot\mathbf{n}|^2+4} - \boldsymbol{\beta}\cdot\mathbf{n}\right)$: the number of iterations for multigrid as solver and preconditioner.

	MG as solver						MG with GMRES					
p	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	3	3	4	4	5	7	3	3	3	4	5	6
2	3	3	4	5	7	*	3	3	3	4	5	*

Table 4.22: Example VI. $\kappa = 10^{-5}$, HDG with $\tau = \frac{1}{2}\left(\sqrt{|\boldsymbol{\beta}\cdot\mathbf{n}|^2+4} - \boldsymbol{\beta}\cdot\mathbf{n}\right)$: the number of iterations for multigrid as solver and preconditioner.

4.4.3 Example VII: Stokes equations

In this section we study the performance of multigrid as solver and preconditioner for the Stokes equations discretized with HDG. To that end we consider the following first order velocity-pressure formulation of Stokes equations

$$\mathrm{Re}\mathbf{L} - \nabla\mathbf{u} = 0 \quad \text{in } \Omega, \tag{4.21a}$$

$$-\nabla\cdot\mathbf{L} + \nabla q = \mathbf{f} \quad \text{in } \Omega, \tag{4.21b}$$

$$\nabla\cdot\mathbf{u} = 0 \quad \text{in } \Omega, \tag{4.21c}$$

$$\mathbf{u} = \mathbf{u}_D \quad \text{on } \partial\Omega, \tag{4.21d}$$

$$\int_\Omega q\, d\Omega = 0. \tag{4.21e}$$

The HDG discretization of the Stokes equations (4.21) gives the following local

solver

$$\mathrm{Re}\,(\mathbf{L},\mathbf{G})_T + (\mathbf{u},\nabla\cdot\mathbf{G})_T - \langle\hat{\mathbf{u}},\mathbf{G}\mathbf{n}\rangle_{\partial T} = 0, \tag{4.22a}$$

$$-(\nabla\cdot\mathbf{L},\mathbf{v})_T + (\nabla q,\mathbf{v})_T + \langle\mathbf{S}\,(\mathbf{u}-\hat{\mathbf{u}})\,,\mathbf{v}\rangle_{\partial T} = (\mathbf{f},\mathbf{v})_T\,, \tag{4.22b}$$

$$\left(\frac{\partial q}{\partial t},r\right)_T - (\mathbf{u},\nabla r)_T + \langle\hat{\mathbf{u}}\cdot\mathbf{n},r\rangle_{\partial T} = 0, \tag{4.22c}$$

and global solver

$$\langle[\![-\mathbf{L}\cdot\mathbf{n} + q\mathbf{n} + \mathbf{S}\,(\mathbf{u}-\hat{\mathbf{u}})]\!],\boldsymbol{\mu}\rangle_{e\setminus\partial\Omega} = 0. \tag{4.23}$$

Here we have used the augmented Lagrangian approach and added a pseudo time-derivative in pressure to make the local solvers well-posed [114, 140]. The example we consider is the Kovasznay flow [90, 114] with the exact solution

$$\mathbf{u}^e = (1-\exp(cx)\cos(2\pi y)\,,\frac{c}{2\pi}\exp(cx)\sin(2\pi y)),$$
$$q^e = \frac{1}{2}\exp(2cx)\,,$$

where $c = \frac{Re}{2} - \sqrt{\frac{Re^2}{4}+4\pi^2}$ and $f = -(\mathbf{u}^e\cdot\nabla)\,\mathbf{u}^e$.

We choose $Re = 10$ and the domain $\Omega = (0,2)\times(-0.5,1.5)$ discretized with quadrilateral elements. The multigrid hierarchy is same as the one in Table 4.1. Dirichlet boundary conditions based on the exact solution are enforced through $\hat{\mathbf{u}}$. With the augmented Lagrangian approach we can eliminate both volume velocity and pressure in terms of the velocity trace unknowns using the local solver (4.22). The conservation equation (4.23) generates the global linear system which involves only the velocity trace unknowns and we apply the multigrid solver/preconditioner for solving it. The unknowns are ordered such that all the $x-$component of velocity occurs first followed by the $y-$component of velocity. The smoother used is LU-SGS. In the augmented Lagrangian approach a zero initial condition for pressure is selected and the time-derivative is discretized by backward Euler approach with a time stepsize

of Δt and iterations are carried out till convergence in pressure is reached [114]. For the Stokes equations the number of augmented Lagrangian iterations is independent of meshsize h and solution order p as proved in [114].

For this problem, we do not observe $h-$scalability with pure upwind stabilization given by $\mathbf{S} = \mathbf{I} + (\sqrt{2} - 1)\mathbf{n} \otimes \mathbf{n}$, and hence similar to scalar elliptic problems we use the stabilization $\mathbf{S} = (1/h)\mathbf{I}$. In Tables 4.23, 4.24 and 4.25 we compare the performance of multigrid as a solver and as a preconditioner to GMRES for three pseudo time stepsizes $\Delta t = 1, 8$ and 16. As can be seen for all the time stepsizes we obtain almost perfect $hp-$scalability with multigrid preconditioned GMRES. With increase in time stepsize we see an increase in number of iterations and it is much less pronounced for multigrid as a preconditioner compared to the multigrid solver. Finally, since the total number of iterations required to solve the Stokes equations is the product of multigrid/GMRES iterations and the augmented Lagrangian iterations choosing a time stepsize of $\Delta t = 8$ and multigrid preconditioned GMRES seems to be the best choice with the least number of total iterations.

4.4.4 Example VIII: Oseen equations

In this section we consider the Oseen equations obtained by the linearization of the incompressible Navier–Stokes equations around a known velocity field $\mathbf{w}$. The equations in first order velocity-pressure form is given by

$$\mathrm{Re}\mathbf{L} - \nabla\mathbf{u} = 0 \quad \text{in } \Omega, \tag{4.24a}$$

$$-\nabla \cdot \mathbf{L} + (\mathbf{w} \cdot \nabla)\,\mathbf{u} + \nabla q = \mathbf{f} \quad \text{in } \Omega, \tag{4.24b}$$

$$\nabla \cdot \mathbf{u} = 0 \quad \text{in } \Omega, \tag{4.24c}$$

$$\mathbf{u} = \mathbf{u}_D \quad \text{on } \partial\Omega, \tag{4.24d}$$

$$\int_\Omega q \, d\Omega = 0. \tag{4.24e}$$

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	13	15	17	17	18	18	7	9	9	9	9	9
2	7	8	9	10	10	10	6	6	7	7	7	7
3	8	9	10	10	11	11	6	7	7	7	7	7
4	8	10	11	11	11	11	6	7	7	7	7	7
5	10	11	12	12	12	12	7	7	8	8	8	8
6	10	12	13	13	13	13	7	8	8	8	8	8
7	12	12	13	14	14	14	7	8	8	8	8	8
8	12	13	14	14	14	14	7	8	8	8	8	8
9	13	14	15	15	15	15	8	9	9	9	9	9
10	14	14	15	16	16	16	8	9	9	9	9	9

Table 4.23: Example VII: the number of iterations for multigrid as solver and preconditioner with pseudo time stepsize $\Delta t = 1$. The number of augmented Lagrangian iterations in this case is $\approx 18 - 22$ and is independent of h, p.

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	27	31	35	36	36	36	11	13	14	14	14	13
4	17	20	21	21	21	21	10	11	11	11	11	10
6	18	21	22	23	23	23	10	11	11	11	11	11
8	19	22	24	24	24	24	11	11	12	12	11	11

Table 4.24: Example VII: the number of iterations for multigrid as solver and preconditioner with pseudo time stepsize $\Delta t = 8$. The number of augmented Lagrangian iterations in this case is $\approx 8 - 9$ and is independent of h, p.

The HDG discretization of (4.24) with corresponding local and global solvers is given by

$$\mathrm{Re}\left(\mathbf{L}, \mathbf{G}\right)_T + \left(\mathbf{u}, \nabla \cdot \mathbf{G}\right)_T - \left\langle \hat{\mathbf{u}}, \mathbf{G}\mathbf{n} \right\rangle_{\partial T} = 0, \tag{4.25}$$

$$-\left(\nabla \cdot \mathbf{L}, \mathbf{v}\right)_T + \left(\left(\mathbf{w} \cdot \nabla\right)\mathbf{u}, \mathbf{v}\right)_T + \left(\nabla q, \mathbf{v}\right)_T + \left\langle \mathbf{S}\left(\mathbf{u} - \hat{\mathbf{u}}\right), \mathbf{v} \right\rangle_{\partial T} = \left(\mathbf{f}, \mathbf{v}\right)_T, \tag{4.26}$$

$$\left(\frac{\partial q}{\partial t}, r\right)_T - \left(\mathbf{u}, \nabla r\right)_T + \left\langle \hat{\mathbf{u}} \cdot \mathbf{n}, r \right\rangle_{\partial T} = 0, \tag{4.27}$$

$$\left\langle [\![-\mathbf{L} \cdot \mathbf{n} + q\mathbf{n} + \left(\mathbf{w} \cdot \mathbf{n}\right)\mathbf{u} + \mathbf{S}\left(\mathbf{u} - \hat{\mathbf{u}}\right)]\!], \boldsymbol{\mu} \right\rangle_{e \setminus \partial\Omega} = 0, \tag{4.28}$$

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
1	39	47	51	52	53	53	13	17	17	17	17	16
4	24	27	30	30	30	30	12	13	13	13	13	13
6	27	30	33	33	33	33	13	14	14	14	14	13
8	29	32	35	35	35	35	13	14	15	14	14	14

Table 4.25: Example VII: the number of iterations for multigrid as solver and preconditioner with pseudo time stepsize $\Delta t = 16$. The number of augmented Lagrangian iterations in this case is ≈ 7 and is independent of h, p.

where just as in Stokes equations (4.22) we have added a pseudo time-derivative in pressure to make the local solver well-posed. Here, we consider two stabilizations $\mathbf{S}_{upwind}$ and $\mathbf{S}_{mesh}$ given by

$$\mathbf{S}_{upwind} = a_2\mathbf{I} + a_1\mathbf{n}\otimes\mathbf{n}, \tag{4.29a}$$

$$\mathbf{S}_{mesh} = (1/h + a_2)\mathbf{I}, \tag{4.29b}$$

$$a_1 = \frac{1}{2}\left\{\left(\sqrt{|\mathbf{w}\cdot\mathbf{n}|^2 + 8}\right) - \left(\sqrt{|\mathbf{w}\cdot\mathbf{n}|^2 + 4}\right)\right\}, \tag{4.29c}$$

$$a_2 = \frac{1}{2}\left\{\left(\sqrt{|\mathbf{w}\cdot\mathbf{n}|^2 + 4}\right) - \mathbf{w}\cdot\mathbf{n}\right\}. \tag{4.29d}$$

The example we consider is the Poiseuille flow with exact solutions given by

$$\mathbf{u}^e = \left(\frac{Ql_0^2}{2\mu}\left(1 - \frac{y^2}{l_0^2}\right), 0\right),$$
$$q^e = -Q\left(x - 5\right),$$

where Q is the prescribed pressure gradient and l_0 is the characteristic length. We take $\mathbf{f} = \left(\frac{Q}{\rho}, 0\right)$ and $\mathbf{w} = \mathbf{u}^e$, where ρ is the density. The parameters are chosen as $l_0 = \mu = \rho = 1$ and the pressure gradient Q is selected such that the Reynolds number based on the centerline velocity is $1, 10$ and 50 respectively. The domain and the multigrid hierarchy are same as the one used for the Stokes example in section

4.4.3. The smoother is LU-SGS. Boundary conditions based on the exact solution are enforced through $\hat{\mathbf{u}}$.

In Tables 4.26-4.30 we present the number of iterations for multigrid as a solver and as preconditioner (except for Re = 50 where we only present the preconditioner results as the solver takes too many iterations) for the stabilizations $\mathbf{S}_{upwind}$ and $\mathbf{S}_{mesh}$ for three Reynolds numbers Re $= 1, 10$ and 50. The conclusion is very similar to the convection-diffusion equation, i.e., for Re = 1 in the diffusion dominated regime we need mesh-dependent form of the stabilization $\mathbf{S}_{mesh}$ for scalability whereas with increase in Re the upwind stabilization performs better. For Re $>$ 50 and high orders we observed the multigrid takes either too many iterations or diverges. Hence similar to the case of convection-diffusion, here also we need robust line smoothers for convergence and scalability.

	MG as solver						MG with GMRES					
p	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
2	7	14	19	31	59	118	5	9	14	20	35	65

Table 4.26: Example VIII. Re = 1, HDG with stabilization $\mathbf{S}_{upwind}$: the number of iterations for multigrid as solver and preconditioner with pseudo time stepsize $\Delta t = 16$.

	MG as solver						MG with GMRES					
p	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
2	6	12	14	14	19	16	5	8	11	12	13	11

Table 4.27: Example VIII. Re = 1, HDG with stabilization $\mathbf{S}_{mesh}$: the number of iterations for multigrid as solver and preconditioner with pseudo time stepsize $\Delta t = 16$.

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
2	15	22	31	36	42	62	10	17	19	20	20	24

Table 4.28: Example VIII. Re = 10, HDG with stabilization $\mathbf{S}_{upwind}$: the number of iterations for multigrid as solver and preconditioner with pseudo time stepsize $\Delta t = 16$.

p	MG as solver						MG with GMRES					
	Levels						Levels					
	2	3	4	5	6	7	2	3	4	5	6	7
2	13	25	34	38	37	36	11	20	25	26	26	25

Table 4.29: Example VIII. Re = 10, HDG with stabilization $\mathbf{S}_{mesh}$: the number of iterations for multigrid as solver and preconditioner with pseudo time stepsize $\Delta t = 16$.

p	$\mathbf{S}_{upwind}$				$\mathbf{S}_{mesh}$			
	Levels				Levels			
	2	3	4	5	2	3	4	5
2	11	28	44	61	11	21	41	72

Table 4.30: Example VIII. Re = 50, HDG with stabilizations $\mathbf{S}_{upwind}$, $\mathbf{S}_{mesh}$: the number of iterations for multigrid preconditioned GMRES with pseudo time stepsize $\Delta t = 16$.

4.5 Discussion

In this chapter we have proposed a unified DtN-based geometric multigrid algorithm for hybridized high-order finite element methods. Our approach differs significantly from the previous attempts in the sense that the intergrid transfer operators are physics-based energy-preserving and the coarse grid operators are discretized DtN maps on every level. These operators completely avoid upscaling of parameters, such as permeability and source, to coarse scales. As our numerical results have indicated, they also allow for multinumerics (shown in [158]) and variable coefficients

without deteriorating the performance of the multigrid algorithm. We have presented several numerical examples with HDG methods and hybridized IPDG methods for the second-order elliptic equation and reported several observations. For HDG methods, with stabilization $\tau = 1$ or $1/h_{min}$ (depending on the example), we obtain almost perfect scalability in meshsize and solution order using multigrid preconditioned GMRES. For all the other hybridized IPDG methods, with a stabilization of the form $\tau = \mathcal{O}\left(p^2/h_{min}\right)$ similar scalability is observed. However, the multigrid algorithm applied to IPDG seems to be less robust compared to HDG with respect to stabilization parameters and coarsening strategies especially for highly unstructured meshes.

We have also tested the multigrid methods for the convection-diffusion equation, the Stokes equations and the Oseen equations discretized with HDG. The conclusion is, the multigrid preconditioned GMRES works well for Stokes equations and the convection-diffusion equation in the diffusion dominated regime giving almost perfect $hp-$scalability. For convection dominated regime in the convection-diffusion equation and for the Oseen equations we need robust line smoothers which follow the direction of convection for convergence and scalability.

Chapter 5

A Multilevel Solver for HDG Trace Systems

In chapter 4 we introduced a geometric multigrid solver for the trace system of HDG. Our numerical experiments showed that the solver gives almost perfect $hp-$scalability for the Poisson and Stokes type elliptic PDEs. However, for the convection-diffusion equation in the convection dominated regime and Oseen equations in the high Reynolds number regime the solver has difficulty in convergence. In chapter 4 we mentioned this could be rectified by means of robust line smoothers which follow the direction of convection. However, in some complex velocity fields or in some multiphysics PDEs like MHD such robust smoothers may not be available. In this chapter[1] we follow a different approach towards this issue. We create a strong coarse solver by modifying a direct solver approach namely nested dissection and combine it with simple smoothers such as block-Jacobi to create a robust solver for a variety of PDEs.

Now let us briefly discuss the main idea behind our approach. The goal is to *advance the nested dissection* [67]—a fill-in reducing direct solver strategy—to create a *scalable and robust solver utilizing the high-order and variational structure of HDG methods*. This is achieved by projecting the skeletal data at different levels to either same or high-order polynomial on a set of increasingly $h-$coarser edges/faces. Exploiting the concept of two-level domain decomposition methods we make use of our

[1]The contents of this chapter are largely based on the manuscript [110], which is under revision in the Journal of computational physics. The contributions of the author ranged from key ideas of the algorithm, numerical implementation, complexity estimates and writing the manuscript.

approach as a coarse solver together with a fine scale solver (e.g., block-Jacobi) to create a solver/preconditioner for solving the trace system iteratively. Since the coarse solver is derived from a direct solver strategy it helps in reducing the dependence of the two-level solver/preconditioner with respect to the underlying PDE. The fine scale solver, however, usually depends on the PDE being solved and hence can influence the overall performance of the two-level algorithm. Our numerical experiments show that the algorithms converge even for transport equation with discontinuous solution and elliptic equations with highly heterogeneous and discontinuous permeability. For convection-diffusion equations our multilevel preconditioning algorithms are scalable and reasonably robust for not only diffusion-dominated but also moderately convection-dominated regimes. We show that *the two-level approach can also be interpreted as a multigrid algorithm with specific intergrid transfer and smoothing operators*. Our complexity estimates show that the cost of the multilevel algorithms is somewhat in between the cost of nested dissection and standard multigrid solvers.

This chapter is organized as follows. In Section 5.1, we first recall the nested dissection approach and then explain how it can be advanced using HDG variational structure and the two-level domain decomposition approach. We also show that our approach can be interpreted as a multigrid method, and estimate the complexity of the proposed multilevel solvers. Section 5.2 presents several numerical examples to study the robustness and scalability of the proposed algorithm for the Poisson, the transport and the convection-diffusion equations as the mesh and the solution order are refined.

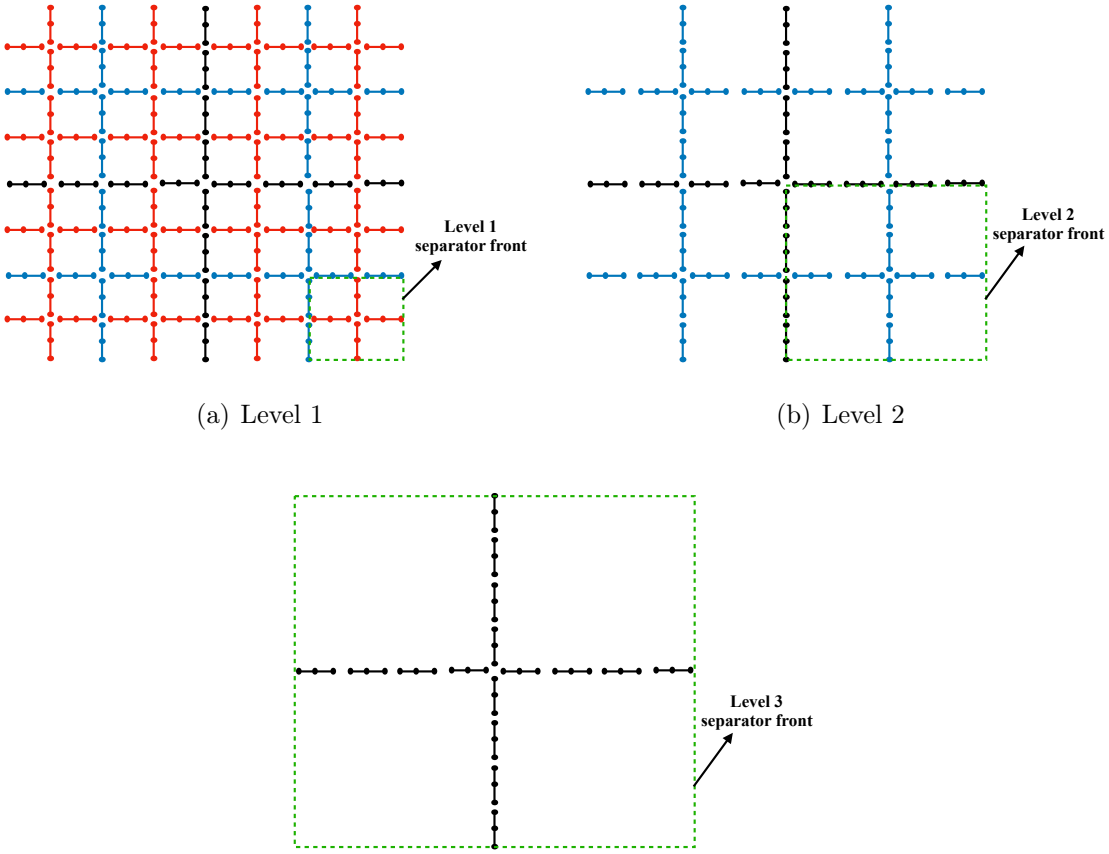
5.1 A Multilevel solver for the HDG trace system

The objective of our current work is to develop a robust multilevel solver and preconditioner for HDG discretizations for a wide variety of PDEs. The ultimate goal is to significantly reduce factorization and memory costs compared to a direct solver. Unlike cost reduction strategies for direct solvers in [103, 69, 81] which utilizes the elliptic nature of PDEs, here we exploit the high-order and variational structure of HDG methods. As a result, our method is applicable to not only elliptic but also parabolic, hyperbolic, and mixed-type PDEs. For ease of the exposition and implementation, we will focus only on structured grids. While extending the algorithm to block-structured or nested grids is fairly straightforward, applying it to a completely unstructured grid is a non-trivial task and hence left for future work.

5.1.1 Nested dissection

As nested dissection idea is utilized in the proposed multilevel algorithm, we briefly review its concept (more details can be found in [67, 68, 100, 101, 50, 45]). Nested dissection (ND) is a fill-in reducing ordering strategy introduced in 1973 [67] for efficient solution of linear systems. Consider a $p = 2$ solution on an 8×8 quadrilateral HDG skeletal mesh in Figure 5.1(a) (the boundary nodes are eliminated for clarity). In the ND algorithm, we identify a set of separators which divide the mesh into independent subdomains. For example, the black nodes in Figure 5.1(a) divide the mesh into four independent subdomains each of which can be recursively divided into four subdomains and so on. We then order the nodes such that the red ones are ordered first, followed by blue and then the black ones. This will enable a recursive Schur complement approach in a multilevel fashion and the nodes remaining in the system after elimination at each level is shown in Figure 5.1 (for three levels). A very similar concept to nested dissection is the substructuring methods used in the

structural dynamics community for the fast solution of linear systems and eigenvalue computations [13, 144].



(a) Level 1

(b) Level 2

(c) Level 3

Figure 5.1: An example of three levels in the nested dissection (ND) algorithm. The red crosses correspond to level 1 separator fronts and there are 16 fronts in Figure 5.1(a), each having 4 edges. The blue crosses correspond to level 2 separators and in Figure 5.1(b) there are four level 2 fronts, each having 8 edges. The black cross correspond to level 3 separator and in Figure 5.1(c) there is one level 3 front with 16 edges. The circles on each edge represent the nodes and there are three nodes in each edge corresponding to a solution order of $p = 2$.

There are several advantages to this algorithm. First, it can be shown that the serial complexity of factorization for an $N \times N$ matrix arising from 2D problems with

this procedure is $\mathcal{O}(N^{3/2})$, and the memory requirement is $\mathcal{O}(NlogN)$ [67]. Whereas with a naive lexicographic ordering, it is $\mathcal{O}(N^2)$ for factorization and $\mathcal{O}(N^{3/2})$ for memory [67]. Moreover, in 2D it is optimal in the sense that the lower bound of the cost for factorization using any ordering algorithm is $\mathcal{O}(N^{3/2})$ [67]. Second, all the leaf calculations at any level are independent of each other and hence are amenable to parallel implementation. However, in 3D the cost is $\mathcal{O}(N^2)$ for factorization and $\mathcal{O}(N^{4/3})$ for memory [67], and we no longer have the tremendous savings as in 2D [51]. This can be intuitively understood using Figure 5.1. The separator fronts (the crosses at each level in Figure 5.1) grow in size as the level increases. For example, the black crosses have more edges and nodes than the blue ones, which in turn has more edges and nodes than the red ones. On the last level, the size is $\mathcal{O}(N^{1/2})$ for 2D and the cost of a dense matrix factorization for the separator front matrix corresponding to the last level is $\mathcal{O}(N^{3/2})$. In 3D the size of the separator at last level is $\mathcal{O}(N^{2/3})$ and hence the factorization cost becomes $\mathcal{O}(N^2)$. Thus in order to cut down the factorization and storage cost of the ND algorithm we need to reduce the front growth.

There have been many efforts in this direction over the past decade [103, 160, 137, 69, 81]. The basic idea in these approaches is to exploit the low rank structure of the off-diagonal blocks, a characteristic of elliptic PDEs, to compress the fronts. In this way one can obtain a solver which is $\mathcal{O}(N)$ or $\mathcal{O}(NlogN)$ in both 2D and 3D [69, 81]. Unfortunately, since the compression capability is a direct consequence of the ellipticity, it is not trivially applicable for convection-dominated or pure hyperbolic PDEs. *Our goal here is to construct a multilevel algorithm that is applicable to hyperbolic PDEs and at the same time more efficient than ND.* At the heart of our approach is the exploitation of the high-order properties of HDG and the underlying variational structure.

5.1.2 Direct multilevel solvers

In our multilevel algorithm, we start with the ND ordering of the original fine mesh (red, blue, and black edges) as in Figure 5.1(a). Here, by edges we mean the original elemental edges (faces) on the fine mesh. Let us denote the fine mesh as level 0. In Figure 5.2(a), all red crosses have 4 edges, blue crosses have 8 edges and black cross has 16 edges. On these edges are the trace spaces, and thus going from level k to level $(k+1)$ the separator front grows by a factor of two. We propose to reduce the front growth by lumping the edges so that each cross at any level has only four (longer) edges as on level 1 separator fronts. We accomplish this goal by projecting the traces on original fine mesh skeletal edges into a single trace space on a single longer edge (obtained by lumping the edges). Below are the details on how we lump edges and construct the projection operators.

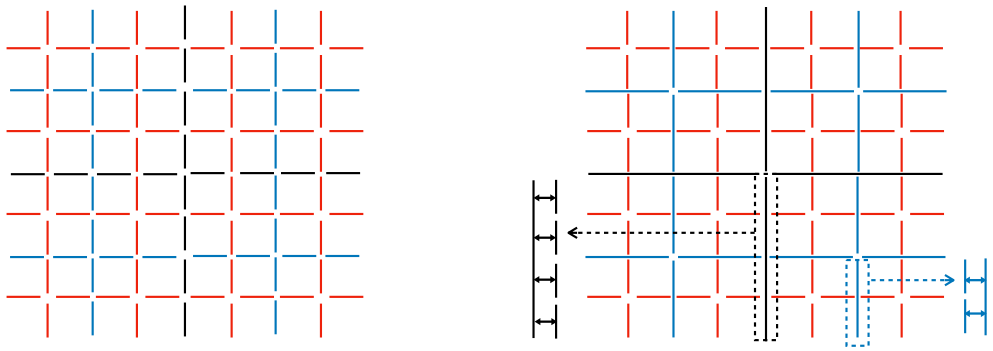
The lumping procedure is straightforward. For example, longer edges on each blue cross in Figure 5.2(b) are obtained by lumping the corresponding two blue (shorter) edges. Similarly, longer edges on each black cross in Figure 5.2(b) are obtained by lumping the corresponding four black (shorter) edges. The resulting skeleton mesh with the same number of edges on the separator fronts in all levels forms level 1 in our multilevel algorithm.

Next, we project the trace spaces on shorter edges into a single trace space on the corresponding lumped edge. The three obvious choices for the solution order of the single trace spaces: (1) lower than, (2) same as, or (3) higher than the solution order on the shorter edges. Low-order option is not sensible as we have already coarsened in h. In particular, additional coarsening in p, i.e., option (1), makes the solver even further away from being "direct". Moreover, since we already invert matrices of size $\mathcal{O}((p+1)^2)$ for separators in level 1, low-order choice will not help in reducing

the cost. For option (2), we obtain separators which are increasingly coarsened in h, and clearly when we apply the ND algorithm this approach do not yield a direct solution nor h-convergence. However, it can be used in the construction of an iterative solver/preconditioner as we shall discuss in section 5.1.3.

Option (3), i.e., high-order projection, is more interesting as we now explain. Due to exponential convergence in p for smooth solution [39], we can compensate for the coarseness in h by means of refinement in p. As a result, for sufficiently smooth solution, projecting to high-order trace spaces can provide accurate approximations to the original ND solution while almost avoiding the front growth. In our algorithm we enrich the order in the following fashion: if p is the solution order on the original fine mesh, then level 1 separators have solution of order p, $p+1$ for separators on level 2, $p+2$ for separators on level 3 and so on. For practical purposes we also (arbitrarily) limit the growth to order 10 to actually stop the front growth after 10 orders. Specifically, for a generic level k we take the solution order on the separator fronts as $p_k = \min\{p + (k-1), 10\}$. We would like to point out that this enriching strategy is natural, but by no means optimal. Optimality requires balancing accuracy and computational complexity, which in turns requires rigorous error analysis of the enrichment. This is left for future research.

To the end of the chapter, we denote option (2) as multilevel (ML) and option (3) as enriched multilevel (EML) to differentiate between them. In Figures 5.3 and 5.4 are different levels corresponding to the ML and EML approaches for solution order of $p = 2$ on the original fine mesh. Level 0 of both cases corresponds to Figure 5.1(a). Note that the number of circles on each edge is equal to the solution order plus one. For example, the solution order on each edge of Figure 5.3(c) is 2, while the enriched solution order is 4 for each edge in Figure 5.4(c).



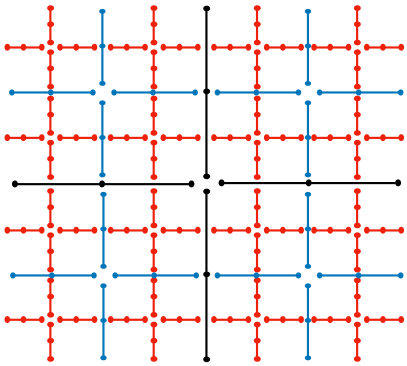
(a) Level 0 (b) Level 1

Figure 5.2: Creation of level 1 from level 0 in the multilevel algorithm: every two short blue edges in Figure 5.2(a) are projected on to the corresponding single long blue edge in Figure 5.2(b). Similarly, every four short black edges in Figure 5.2(a) are projected on to the corresponding single long black edge in Figure 5.2(b). In level 1, all the separator fronts have the same number of edges (of different lengths), which is 4. The nodes on each edge (circles in Figure 5.1) are not shown in this figure.

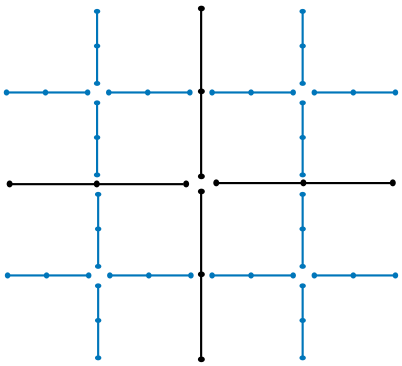
5.1.3 Combining multilevel approaches with domain decomposition methods

As discussed in Section 5.1.2, both ML and EML strategies are approximations of direct solver. A natural idea is to use them as "coarse" scale solvers in a two-level domain decomposition method [122, 141, 144]. In particular, either ML or EML approach can be used to capture the smooth components and to provide global coupling for the algorithm, whereas a fine scale solver can capture the high-frequency localized error and the small length scale details and sharp discontinuities. This combined approach can be employed in an iterative manner as a two-level solver in the domain decomposition methods.

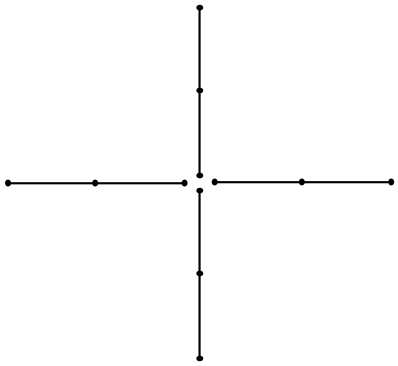
We select block-Jacobi as our fine scale solver, where each block corresponds to an edge in the original fine mesh in Figure 5.1(a). The reason for this choice is that block-Jacobi is straightforward to parallelize, insensitive to ordering direction



(a) Level 1



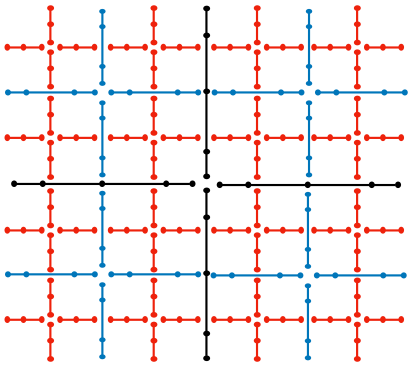
(b) Level 2



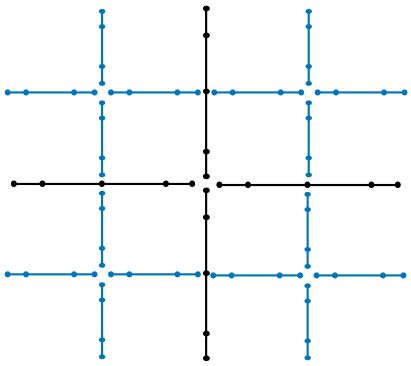
(c) Level 3

Figure 5.3: An example of different levels in the multilevel (ML) algorithm. Compared to Figure 5.1 for ND, here the separator fronts at all levels have the same number of edges, i.e., 4. Similar to Figure 5.1 the three circles on each edge represent the nodes corresponding to a solution order of $p = 2$.

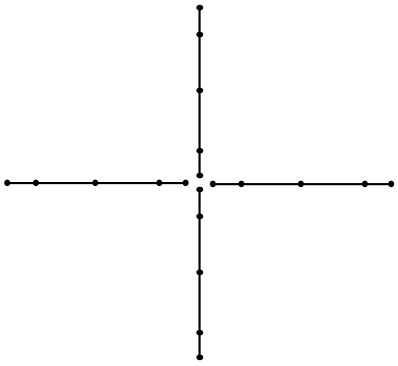
for problems with convection and also reasonably robust with respect to problem parameters. This is also supported by our previous work on geometric multigrid methods for elliptic PDEs in section 4.4.1.1, [159], where we compared few different smoothers and found block-Jacobi to be a competitive choice. We combine the fine and coarse scale solvers in a multiplicative way as this is typically more effective than additive two-level solvers especially for nonsymmetric problems [144].



(a) Level 1



(b) Level 2



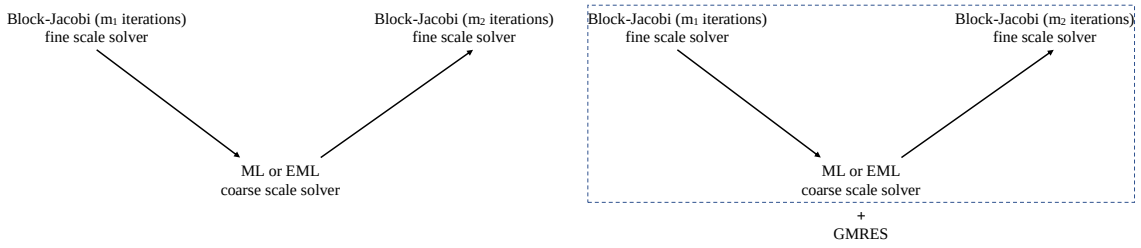
(c) Level 3

Figure 5.4: An example of different levels in the enriched multilevel (EML) algorithm. The number of edges in the separator fronts at all levels is 4. Due to polynomial enrichment, we have 3 nodes per edge, corresponding to $p = 2$, on the red crosses (level 1 separator fronts); 4 nodes per edge, corresponding to $p = 3$, on the blue crosses (level 2 separator fronts); and 5 nodes per edge, corresponding to $p = 4$, on the black cross (level 3 separator front).

We would like to point out that due to the approximate direct solver characteristic of our coarse scale solvers, regardless of the nature of the underlying PDEs, our two-level approaches are applicable. This will be verified in various numerical results in Section 5.2. Next we layout the algorithm for our iterative multilevel approach.

5.1.4 Iterative multilevel solvers/preconditioners

In Figure 5.5 we show schematic diagrams of the two-level approaches described in Section 5.1.3 combining block-Jacobi fine-scale solver and ML or EML coarse-scale solvers (coarse in the $h-$sense). Algorithm 4 describes in details every step of these iterative multilevel solvers and how to implement them. In particular, there we present the algorithm for linear problems or linear systems arising from Newton linearization or Picard linearization for nonlinear problems. Note that we perform the factorizations of both coarse- and fine-scale matrices before the start of the iteration process so that during each iteration only back solves are needed. To precondition the GMRES algorithm we simply use one v-cycle of these iterative approaches.



(a) Two-level solver (b) Two-level preconditioner

Figure 5.5: Two-level solvers and preconditioners combining block-Jacobi and ML or EML solvers.

5.1.5 Relationship between iterative multilevel approach and multigrid method

In this section we will show that the iterative multilevel approach presented in Algorithm 4 can be viewed as a multigrid approach with specific prolongation, restriction, and smoothing operators. To that end, let us consider a sequence of

Algorithm 4 An iterative multilevel approach.

1: Order the unknowns (or permute the matrix) in the nested dissection manner.
2: Construct a set of L^2 projections by visiting the edges of the original fine mesh skeleton.
3: Create the level 1 matrices for ML or EML as $A_1 = I_0^* A I_0$, where I_0 is the projection matrix from level 1 to level 0 and I_0^* is its L^2 adjoint.
4: Compute factorizations of level 1 matrices of ML or EML (by means of nested dissection), and the block-Jacobi matrices corresponding to level 0.
5: Compute the initial guess using the coarse scale solver (either ML or EML).
6: **while** not converged **do**
7: Perform m_1 iterations of the block-Jacobi method.
8: Compute the residual.
9: Perform coarse grid correction using either ML or EML.
10: Compute the residual.
11: Perform m_2 iterations of the block-Jacobi method.
12: Check convergence. If yes, **exit**, otherwise **set** $i = i + 1$, and **continue**.
13: **end while**

interface grids $\mathcal{E}_0 = \mathcal{E}_h, \mathcal{E}_1, \ldots, \mathcal{E}_N$,[2] where each $\mathcal{E}_k$ contains the set of edges which remain at level k. Here, $\mathcal{E}_0$ is the fine interface grid and $\mathcal{E}_N$ is the coarsest one. Each partition $\mathcal{E}_k$ is in turn associated with a skeletal (trace) space M_k. We decompose $\mathcal{E}_k$ as $\mathcal{E}_k = \mathcal{E}_{k,I} \oplus \mathcal{E}_{k,B}$, where $\mathcal{E}_{k,I}$ is the set of interior edges, corresponding to separator fronts at level k, and $\mathcal{E}_{k,B}$ is the set of remaining (boundary) edges. To illustrate this decomposition, let us consider Figures 5.3 and 5.4. Red edges, blue, and black lumped edges are $\mathcal{E}_{1,I}$, $\mathcal{E}_{2,I}$, and $\mathcal{E}_{3,I}$, respectively, and $\mathcal{E}_{k,B} = \mathcal{E}_k \setminus \mathcal{E}_{k,I}$ for $k = 1, 2, 3$. We also decompose the trace space M_k on $\mathcal{E}_k$ into two parts $M_{k,I}$ and $M_{k,B}$ corresponding to $\mathcal{E}_{k,I}$ and $\mathcal{E}_{k,B}$, respectively. Specifically, we require $M_k = M_{k,I} \oplus M_{k,B}$ such that each $\lambda_k \in M_k$ can be uniquely expressed as $\lambda_k = \lambda_{k,I} + \lambda_{k,B}$, where

$$\lambda_{k,I} = \begin{cases} \lambda_k & \text{on } M_{k,I}, \\ 0 & \text{on } M_{k,B}, \end{cases} \quad \text{and} \quad \lambda_{k,B} = \begin{cases} 0 & \text{on } M_{k,I}, \\ \lambda_k & \text{on } M_{k,B}. \end{cases}$$

[2]Note that the ordering here is reversed when compared to the multigrid algorithm in chapter 4.

The spaces M_k for ML algorithm is given by

$$M_k = \{\eta \in \mathcal{Q}^p(e) \forall e \in \mathcal{E}_k\} \quad \text{for} \quad k = 0, 1, 2, \ldots, N,$$

whereas for EML algorithm it is given by

$$M_k = \begin{cases} \text{for} \quad k = 0 \\ \{\eta \in \mathcal{Q}^p(e) \forall e \in \mathcal{E}_k\} \\ \text{for} \quad k = 1, 2, \ldots, N \\ \left\{\eta \in \mathcal{Q}^{\min(p+(k-1),10)}(e) \forall e \in \mathcal{E}_{k,I}\right\} \\ \left\{\eta \in \mathcal{Q}^{\min(p+k,10)}(e) \forall e \in \{\mathcal{E}_{k,B} \subset \mathcal{E}_{k+1,I}\}\right\} \\ \left\{\eta \in \mathcal{Q}^{\min(p+k+1,10)}(e) \forall e \in \{\mathcal{E}_{k,B} \subset \mathcal{E}_{k+2,I}\}\right\} \\ \vdots \\ \left\{\eta \in \mathcal{Q}^{\min(p+k+N-2,10)}(e) \forall e \in \{\mathcal{E}_{k,B} \subset \mathcal{E}_{k+N-1,I}\}\right\}. \end{cases}$$

Here, we denote by $\mathcal{Q}^p$ the space of tensor product polynomials of degree at most p in each dimension.

If the trace system at level 0 is given by

$$A\lambda = g. \tag{5.1}$$

Given the decomposition $M_k = M_{k,I} \oplus M_{k,B}$, the trace system (5.1) at the kth level can be written as

$$A_k \lambda_k = g_k \Leftrightarrow \begin{bmatrix} A_{k,II} & A_{k,IB} \\ A_{k,BI} & A_{k,BB} \end{bmatrix} \begin{bmatrix} \lambda_{k,I} \\ \lambda_{k,B} \end{bmatrix} = \begin{bmatrix} g_{k,I} \\ g_{k,B} \end{bmatrix}. \tag{5.2}$$

We next specify the prolongation, restriction and smoothing operators. Since, all of the operators except the ones between level 0 and level 1, correspond to ideal operators in [145] we explain them briefly here.

5.1.6 Prolongation operator

We define the prolongation operator $I_{k-1} : M_k \to M_{k-1}$ for our iterative algorithm as

$$I_{k-1} := \begin{cases} \Pi_0 & \text{for} \quad k = 1, \\ \begin{bmatrix} -A_{k,II}^{-1} A_{k,IB} \\ \mathfrak{I}_{BB} \end{bmatrix} & \text{for} \quad k = 2, \ldots, N. \end{cases} \tag{5.3}$$

Here, we denote by Π_0 the L^2 projection from $M_1 \to M_0$ and $\mathfrak{I}_{BB}$ the identity operator on the boundary. Clearly, apart from $k = 1$, the prolongation operator is nothing but the ideal prolongation in algebraic multigrid methods [145] as well as in the Schur complement multigrid methods [124, 149, 47, 46].

5.1.7 Restriction operator

We define the restriction operator $Q_k : M_{k-1} \to M_k$ for our iterative algorithm as

$$Q_k := \begin{cases} \Pi_0^* & \text{for} \quad k = 1, \\ \begin{bmatrix} -A_{k,BI} A_{k,II}^{-1} & \mathfrak{I}_{BB} \end{bmatrix} & \text{for} \quad k = 2, \ldots, N. \end{cases} \tag{5.4}$$

Here, Π_0^* is the L^2 adjoint of Π_0. Similar to prolongation, apart from $k = 1$, the restriction operator is the ideal restriction operator [145]. Given the restriction and prolongation operators, the Galerkin coarse grid operator is constructed as

$$A_k := Q_k A_{k-1} I_{k-1}. \tag{5.5}$$

5.1.8 Smoothing

Recall from the Algorithm 4 and Figure 5.5 that we have both pre- and postsmoothing steps using block-Jacobi at level 0. Either ML or EML algorithm implicitly provides additional smoothing. Indeed, let us consider two generic levels k and $k + 1$. At level k, given the decomposition (5.2) we can write the inverse of A_k

as [146, 124]

$$A_k^{-1} = \begin{bmatrix} A_{k,II}^{-1} & 0 \\ 0 & 0 \end{bmatrix} + I_k A_{k+1}^{-1} Q_{k+1}, \tag{5.6}$$

where $A_{k+1} = Q_{k+1} A_k I_k$ is the Galerkin coarse grid matrix (it is also the Schur complement of $A_{k,II}$ in (5.2) [145]). As can be seen, the second term on the right hand side of (5.6) is the coarse grid correction while the first term is the additive smoothing applied only to the interior nodes. Another way [124] to look at this is the following. If the coarse grid correction is $z_{k+1} = A_{k+1}^{-1} Q_{k+1} [g_{k,I} \quad g_{k,B}]^T$ then the block-Jacobi smoothing applied only on the interior nodes with initial guess as $I_k z_{k+1}$ is given by

$$\lambda_k = I_k z_{k+1} + \begin{bmatrix} A_{k,II}^{-1} & 0 \\ 0 & 0 \end{bmatrix} \left(\begin{bmatrix} g_{k,I} \\ g_{k,B} \end{bmatrix} - A_k I_k z_{k+1} \right). \tag{5.7}$$

From the definition of prolongation operator for $k > 1$ in (5.3) we see that $A_k I_k z_{k+1} = [0 \quad \circ]^T$, where "$\circ$" is a term that will subsequently be multiplied by 0, and is thus not relevant for our discussion. As a result, λ_k obtained from (5.7) is the same as A_k^{-1} given in (5.6) acting on $[g_{k,I} \quad g_{k,B}]^T$. In other words, the implicit smoothing is equivalent to block-Jacobi smoothing on the interior nodes with the initial guess as $I_k z_{k+1}$. In AMG literature [145] this is called F-smoothing, where F stands for fine nodes. To summarize, the smoothing operator at different levels is given by

$$G_k := \begin{cases} G_0 & \text{for} \quad k = 0, \\ \begin{bmatrix} A_{k,II}^{-1} & 0 \\ 0 & 0 \end{bmatrix} & \text{for} \quad k = 1, \ldots, N, \end{cases} \tag{5.8}$$

where G_0 is the block-Jacobi smoothing operator at level 0 with each block corresponding to an edge in the original fine mesh. If we denote by $m_{1,k}$ and $m_{2,k}$ the number of pre- and postsmoothing steps at level k we have

$$m_{1,k} := \begin{cases} m_1 & \text{for} \quad k = 0, \\ 0 & \text{for} \quad k = 1, \ldots, N, \end{cases} \qquad m_{2,k} := \begin{cases} m_2 & \text{for} \quad k = 0, \\ 1 & \text{for} \quad k = 1, \ldots, N. \end{cases} \tag{5.9}$$

Note that instead of postsmoothing inside the ML or EML solver we could also consider presmoothing (i.e. $m_{1,k} = 1, m_{2,k} = 0 \quad \forall k = 1, \ldots, N$) and the result remains

the same [145]. Now with these specific set of operators the iterative multilevel algorithm, Algorithm 4, is equivalent to the following multigrid v-cycle

$$\lambda^{i+1} = \lambda^i + B_0(r_0), \quad i = 0, \ldots.$$

where $r_0 = g_0 - A_0\lambda^0$. Here, the action of B_0 on a function/vector is defined recursively in the multigrid algorithm, Algorithm 5, and the initial guess λ^0 is computed from either ML or EML solver. In Algorithm 5, $k = 0, 1, \ldots, N-1$, and $G_{k,m_{1,k}}$, $G_{k,m_{2,k}}$ represent the smoother G_k with $m_{1,k}$ and $m_{2,k}$ smoothing steps respectively. At the

Algorithm 5 Iterative multilevel approach as a v-cycle multigrid algorithm

1: *Initialization*:
$e^{\{0\}} = 0$,
2: *Presmoothing*:
$e^{\{1\}} = e^{\{0\}} + G_{k,m_{1,k}}\left(r_k - A_k e^{\{0\}}\right)$,
3: *Coarse Grid Correction*:
$e^{\{2\}} = e^{\{1\}} + I_k B_{k+1}\left(Q_{k+1}\left(r_k - A_k e^{\{1\}}\right)\right)$,
4: *Postsmoothing*:
$B_k\left(r_k\right) = e^{\{3\}} = e^{\{2\}} + G_{k,m_{2,k}}\left(r_k - A_k e^{\{2\}}\right)$.

coarsest level M_N, we set $B_N = A_N^{-1}$ and the inversion is computed using a direct solver. The above multigrid approach trivially satisfies the following relation [145]

$$\langle A_k I_k \lambda, I_k \lambda \rangle_{\mathcal{E}_k} = \langle A_{k+1}\lambda, \lambda \rangle_{\mathcal{E}_{k+1}} \quad \forall \lambda \in M_{k+1}, \tag{5.10}$$

where $\langle ., . \rangle_{\mathcal{E}_k}$, $\langle ., . \rangle_{\mathcal{E}_{k+1}}$ represents the L^2-inner product on $\mathcal{E}_k$ and $\mathcal{E}_{k+1}$ respectively. This is a sufficient condition for the stability of intergrid transfer operators in a multigrid algorithm [145]. The trivialness is due to the fact that: (1) our prolongation and restriction operators are ideal ones except for $k = 1$, for which they are L^2 adjoint of each other; and (2) the coarse grid matrices are constructed by Galerkin projection (5.5).

5.1.9 Complexity estimations

In this section we estimate the serial complexity for ND, ML and EML algorithms in both 2D and 3D. For simplicity, we consider standard square or cubical domain discretized by $N_T = n^d$ quad/hex elements, where d is the dimension. The total number of levels is $N = log_2(n)$. Let p_k be the solution order on separator fronts at level k and we denote by $q_k = (p_k + 1)^{d-1}$ the number of nodes on an edge/face. For simplicity, we consider Dirichlet boundary condition and exclude the boundary edges/faces in the complexity estimates.

For the ND algorithm, we define level 0 to be the same as level 1. Following the analysis in [51], we have 4^{N-k} crosses (separator fronts) at level k and each front is of size $4\left\{\frac{n}{2^{(N+1-k)}}\right\}q_0$ and all matrices are dense. The factorization cost of the ND algorithm in 2D is then given by:

2D ND algorithm

$$Factor = \sum_{k=1}^{N} 4^{(N-k)} \left(4\left\{\frac{n}{2^{(N+1-k)}}\right\}q_0\right)^3 \tag{5.11}$$

$$= \mathcal{O}\left(16q_0^3 N_T^{3/2}\left[1 - \frac{1}{\sqrt{N_T}}\right]\right). \tag{5.12}$$

The memory requirement is given by

$$Memory = \sum_{k=1}^{N} 4^{(N-k)} \left(4\left\{\frac{n}{2^{(N+1-k)}}\right\}q_0\right)^2 \tag{5.13}$$

$$= \mathcal{O}\left(8q_0^2 N_T log_2\left(N_T\right)\right). \tag{5.14}$$

As the Schur complement matrices are dense, the cost for the back solve is same as that for memory. Similarly the estimates in 3D are given as follows:

3D ND algorithm

$$Factor = \sum_{k=1}^{N} 8^{(N-k)} \left(12 \left\{ \frac{n}{2^{(N+1-k)}} \right\}^2 q_0 \right)^3 \tag{5.15}$$

$$= \mathcal{O}\left(31 q_0^3 N_T^2 \left[1 - \frac{1}{N_T} \right] \right). \tag{5.16}$$

$$Memory = \sum_{k=1}^{N} 8^{(N-k)} \left(12 \left\{ \frac{n}{2^{(N+1-k)}} \right\}^2 q_0 \right)^2 \tag{5.17}$$

$$= \mathcal{O}\left(18 q_0^2 N_T^{4/3} \left[1 - \frac{1}{N_T^{1/3}} \right] \right). \tag{5.18}$$

Unlike [120], here we have not included the factorization and memory costs for the matrix multiplication $A_{k,BI} A_{k,II}^{-1} A_{k,IB}$. The reason is that the asymptotic complexity for ND, ML, and EML is unaffected by this additional cost. For EML in particular, the inclusion of this cost makes the analysis much more complicated because of the different solution orders involved at different levels. As shall be shown, our numerical results in section 5.2.2.1 indicate that the asymptotic estimates derived in this section are in good agreement with the numerical results.

As ML is a special case of EML with zero enrichment, it is sufficient to show the estimates for EML. In this case we still have 4^{N-k} fronts at level k and each front is of the size $4q_k$. The factorization and memory costs in 2D are then given by:

2D EML algorithm

$$Factor = \sum_{k=1}^{N} 4^{(N-k)} \left(4 q_k \right)^3 \tag{5.19}$$

$$= 64 q_0^3 \sum_{k=1}^{N} 4^{(N-k)} \alpha_k^3 \tag{5.20}$$

$$= \mathcal{O}\left(64 q_0^3 \left\{ \frac{1}{4} \left(1 + \frac{\alpha_N^3}{3} \right) N_T - \frac{\alpha_N^3}{3} \right\} \right). \tag{5.21}$$

$$Memory = \sum_{k=1}^{N} 4^{(N-k)} \left(4q_k\right)^2 \tag{5.22}$$

$$= 16q_0^2 \sum_{k=1}^{N} 4^{(N-k)} \alpha_k^2 \tag{5.23}$$

$$= \mathcal{O}\left(16q_0^2 \left\{\frac{1}{4}\left(1 + \frac{\alpha_N^2}{3}\right) N_T - \frac{\alpha_N^2}{3}\right\}\right). \tag{5.24}$$

Similarly in 3D we have:

3D EML algorithm

$$Factor = \sum_{k=1}^{N} 8^{(N-k)} \left(12q_k\right)^3 \tag{5.25}$$

$$= 1728q_0^3 \sum_{k=1}^{N} 8^{(N-k)} \alpha_k^3 \tag{5.26}$$

$$= \mathcal{O}\left(1728q_0^3 \left\{\frac{1}{8}\left(1 + \frac{\alpha_N^3}{7}\right) N_T - \frac{\alpha_N^3}{7}\right\}\right), \tag{5.27}$$

$$Memory = \sum_{k=1}^{N} 8^{(N-k)} \left(12q_k\right)^2 \tag{5.28}$$

$$= 144q_0^2 \sum_{k=1}^{N} 8^{(N-k)} \alpha_k^2 \tag{5.29}$$

$$= \mathcal{O}\left(144q_0^2 \left\{\frac{1}{8}\left(1 + \frac{\alpha_N^2}{7}\right) N_T - \frac{\alpha_N^2}{7}\right\}\right). \tag{5.30}$$

Here, $\alpha_k = \frac{q_k}{q_0}$. To enable a direct comparison with ND, we have taken $\alpha_k = \alpha_N, \quad \forall k = 1, 2, \ldots, N$. As a result, the actual cost is less than the estimated ones as $\alpha_k < \alpha_N, \quad \forall k < N$. Note that either ML or EML iterative algorithm, Algorithm 4, requires additional cost of $\mathcal{O}\left(N_T q_0^3\right)$ for factorization and of $\mathcal{O}\left(N_T q_0^2\right)$ for memory and back solves due to block-Jacobi smoothing. Since these additional costs are less than the costs for ML and EML coarse solvers, they increase the overall complexity of the algorithm by at most a constant, and hence can be omitted.

Remark 8. *From the above complexity estimates we observe that the factorization cost of our multilevel algorithms scales like* $\mathcal{O}\left(\alpha_N^3 q_0^3 N_T\right)$ *in both 2D and 3D. Compared to the cost for ND algorithms which is* $\mathcal{O}\left(q_0^3 N_T^{3/2}\right)$ *in 2D and* $\mathcal{O}\left(q_0^3 N_T^2\right)$ *in 3D, a significant gain (independent of spatial dimensions) can be achieved using our methods. Similarly, the memory cost has reduced to* $\mathcal{O}\left(\alpha_N^2 q_0^2 N_T\right)$ *independent of dimensions as opposed to* $\mathcal{O}\left(q_0^2 N_T log_2(N_T)\right)$ *in 2D and* $\mathcal{O}\left(q_0^2 N_T^{4/3}\right)$ *in 3D for the ND algorithm. Here,* $\alpha_N = 1$ *for ML whereas it is greater than one for EML. On the other hand, the memory and computational costs required by multigrid is typically* $\mathcal{O}\left(N_T q_0\right)$*. Thus the proposed multilevel algorithms are* $\mathcal{O}\left(\alpha_N^3 q_0^2\right)$ *times more expensive in computation cost and require* $\mathcal{O}\left(\alpha_N^2 q_0\right)$ *more memory compared to standard multigrid algorithms. The cost of the multilevel algorithms lie in between direct (ND) solvers and multigrid solvers.*

5.2 Numerical results

5.2.1 Two-dimensional examples

In this section we test the multilevel algorithm, Algorithm 4, on elliptic, transport, and convection-diffusion equations. To that end, we consider equations (2.34) and (2.5) discretized by the upwind HDG flux (2.36), (2.37) and (2.7). Except for the transport equation in section 5.2.3, the domain Ω is a standard unit square $[0,1]^2$ discretized with structured quadrilateral elements. Dirichlet boundary condition is enforced strongly by means of the trace unknowns on the boundary. For the transport equation, we take $\Omega = [0,2]^2$ and inflow boundary condition is enforced through trace unknowns while outflow boundary condition is employed on the remaining boundary. The number of levels in the multilevel hierarchy and the corresponding number of quadrilateral elements are shown in Table 5.1 and they are used in all numerical experiments.

Levels (N)	2	3	4	5	6	7	8	9
Elements	4^2	8^2	16^2	32^2	64^2	128^2	256^2	512^2

Table 5.1: 2D multilevel hierarchy.

We note that even though the iterative multilevel algorithm works as a solver for most of the PDEs considered, it either converges slowly or diverges for some of the difficult cases. Hence throughout the numerical section we report the iteration counts for GMRES, preconditioned by one v-cycle of the multilevel algorithm, Algorithm 4, with the number of block-Jacobi smoothing steps taken as $m_1 = m_2 = 2$.

The UMFPACK [44] library is used for the factorization and all the experiments are carried out in MATLAB in serial mode. The specifications of the machine used for the experiments is as follows. The cluster has 24 cores (2 sockets, 12 cores/socket) and 2 threads per core. The cores are Intel Xeon E5-2690 v3 with frequency 2.6 GHz and the total RAM is 256 GB.

The stopping tolerance for the residual is set to be 10^{-9} in the GMRES algorithm. The maximum number of iterations is limited to 200. In the tables of subsequent sections by "*" we mean that the algorithm has reached the maximum number of iterations.

5.2.2 Elliptic equation

5.2.2.1 Example I: Poisson equation with smooth solution

In this section we test the multilevel algorithm on the Poisson equation with a smooth exact solution given by

$$u^e = \frac{1}{\pi^2}\sin(\pi x)\cos(\pi y).$$

The forcing is chosen such that it corresponds to the exact solution, and the exact solution is used to enforce the boundary conditions. The other parameters in equation

(2.34) are taken as $\kappa = 1$, $\nu = 0$, and $\boldsymbol{\beta} = 0$.

In Table 5.2 we show the number of GMRES iterations preconditioned by one v-cycle of iterative ML and EML algorithms. First, the number of iterations for EML is much less than that for ML and for high-order ($p > 3$) solutions and fine meshes EML performs like a direct solver up to the specified tolerance. That is, the number of preconditioned GMRES iterations is 0. This is expected due to: (1) smooth exact solution, and (2) exponential convergence of high-order solutions, and thus the initial guess computed by EML is, within tolerance, same as the direct solution. As expected for ML, increasing solution order decreases the number of GMRES iterations; this is again due to the smooth exact solution and the high-order accuracy of HDG.

In Figures 5.6(b) and 5.6(a) we compare the time-to-solution of GMRES preconditioned by ML and EML algorithms to the one obtained from direct solver with nested dissection. As can be seen, the ratios (ND/ML and ND/EML) are greater than one for either large number of elements or high solution orders. That is, both ML and EML are faster than ND when accurate solution is desirable. In Figure 5.6(c), the EML and ML algorithms are compared and, apart from $p = 1$, EML is faster than ML though the speedup is not significant in this example. For high solution orders, ML behaves more like EML as the mesh is refined.

We now compare the memory usage of ML and EML against ND. Figures 5.7(a) and 5.7(b) show the ratio of memory usages (costs) of EML and ML algorithms relative to ND. We can see that regardless of meshsize and solution order, both ML and EML requires (much) less memory than ND does. In particular, ML requires almost 8 times less memory than ND at the finest mesh for any solution order. For EML, memory saving is more significant as the mesh and/or solution order are refined, and EML is six times less memory demanding than ND with sixth-order solution at

the finest mesh size.

Figure 5.7(c) compares the memory usage between EML and ML. As expected, EML always requires more memory than ML due to the enrichment. However, the maximum ratio is around 2.1 and since we limit the maximum enrichment order to 10, memory requirements at high orders for both methods are similar. This is also the reason that all the curves in Figure 5.7(c) converge to a constant value as the mesh is refined. As the maximum enrichment order can be tuned, depending on the memory availability of computing infrastructure, one can have the flexibility to adjust the EML algorithm to adapt to memory and computation demands of the problem under consideration. For example, we can perform more iterations for less memory to achieve a given accuracy.

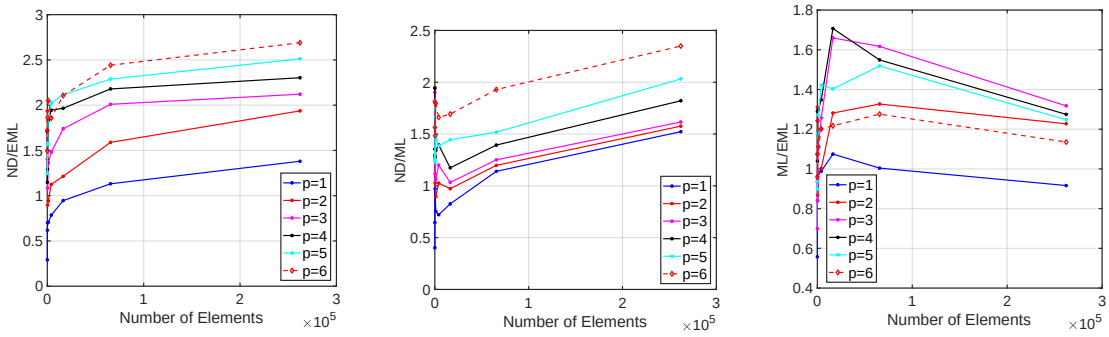
Next we verify the complexity estimates derived in section 5.1.9. Figures 5.8, 5.9 and 5.10 show that the numerical results agree well with our estimates except for the factorization cost of ND in Figure 5.8(a), which seems to indicate that the asymptotic complexity of $\mathcal{O}(N_T^{3/2})$ has not yet been reached. Since the results in Figures 5.8, 5.9 and 5.10 are independent of the PDE under consideration, in the subsequent sections we study only the iteration counts. As long as the iterations do not increase much when the mesh and the solution order are refined, we can obtain a scalable algorithm whose cost can be estimated based on the factorization and memory costs derived in section 5.1.9.

5.2.2.2 Example II: Discontinuous highly heterogeneous permeability

In this section we test the robustness of the algorithm for elliptic PDE with a highly discontinuous and heterogeneous permeability field. To that end, in equation (2.34) we take $\boldsymbol{\beta} = 0$, $\nu = 0$ and κ is chosen according to example 2 in [81] and is shown in Figure 5.11 for three different meshes. The forcing and boundary condition

N	ML with GMRES						EML with GMRES					
	p						p					
	1	2	3	4	5	6	1	2	3	4	5	6
2	3	3	3	3	2	1	2	3	3	2	2	1
3	6	6	5	4	4	2	4	4	4	3	2	1
4	10	8	8	6	5	4	7	5	3	2	0	0
5	14	11	11	8	7	4	9	5	3	0	0	0
6	21	16	16	12	10	6	12	6	2	0	0	0
7	31	24	22	17	13	8	16	5	0	0	0	0
8	44	34	32	23	19	11	19	3	0	0	0	0
9	63	49	45	33	26	16	22	0	0	0	0	0

Table 5.2: Example I: number of ML- and EML-preconditioned GMRES iterations as the mesh is refined (increasing N) and the solution order p increases.

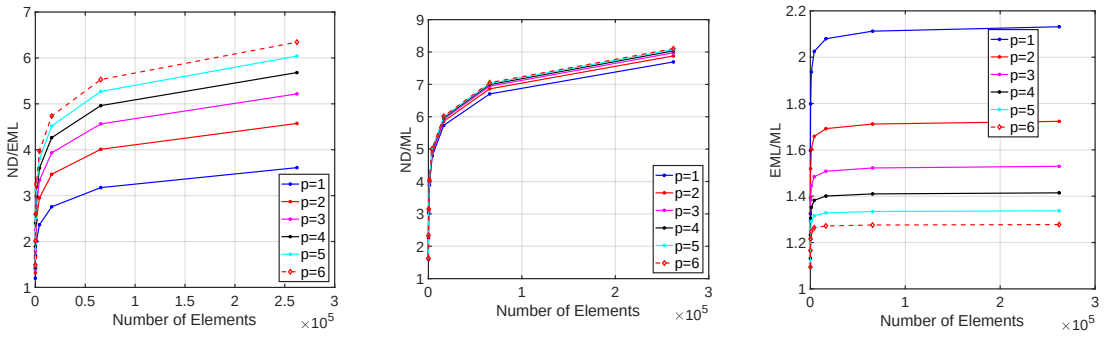


(a) EML versus ND (b) ML versus ND (c) EML versus ML

Figure 5.6: Example I: a comparison of time-to-solution for EML, ML, and ND algorithms.

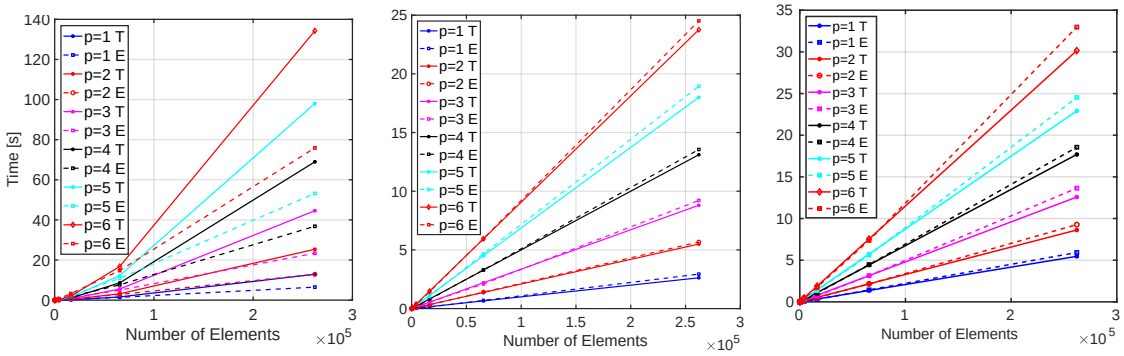
in (2.34) are chosen as $f = 1$ and $g_D = 0$. This is a difficult test case as the permeability varies by four orders of magnitude and is also highly heterogeneous as seen in Figure 5.11.

Tables 5.3 and 5.4 show the number of iterations taken by ML- and EML-preconditioned GMRES, and in Table 5.5 we compare them with those taken by GMRES preconditioned by one v-cycle of geometric multigrid presented in chapter 4, [159]. In Tables 5.3, 5.4 and 5.5 the error, $\max|\lambda_{direct} - \lambda|$, after 200 iterations is shown in the parentheses. Here, λ_{direct} and λ denote trace solution vectors obtained



(a) EML versus ND (b) ML versus ND (c) EML versus ML

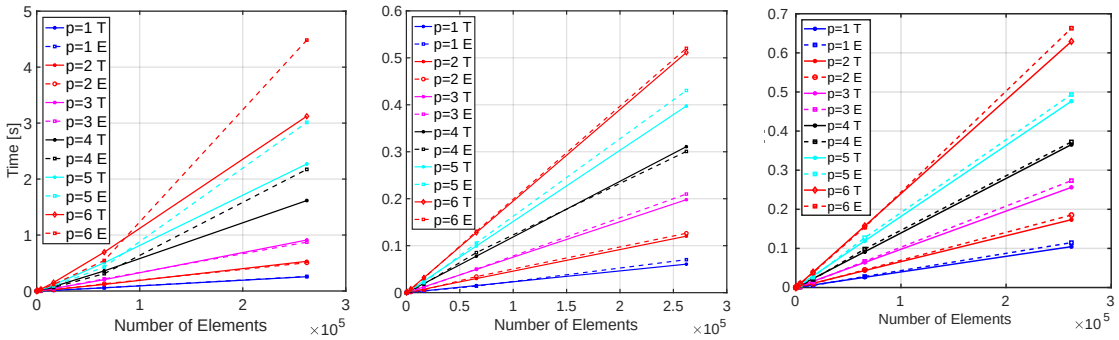
Figure 5.7: Example I: a comparison of memory requirement for EML, ML, and ND algorithms.



(a) ND (b) ML (c) EML

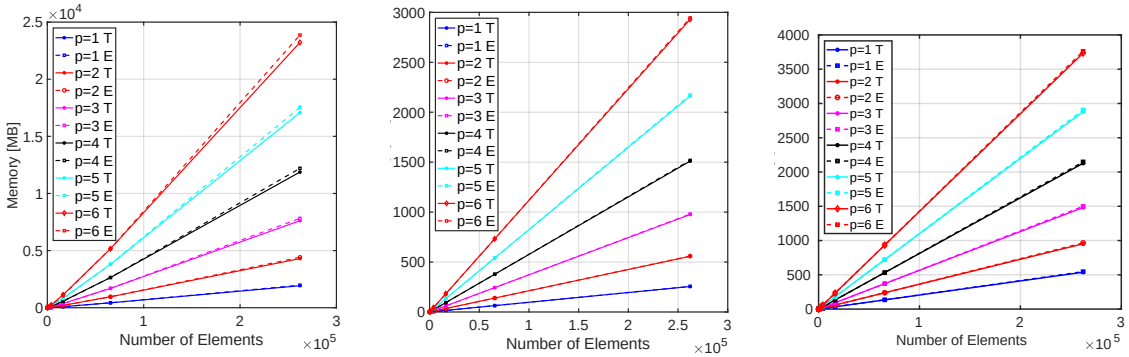
Figure 5.8: Example I: asymptotic and numerical estimates of factorization time complexity for EML, ML, and ND. Here, T stands for the theoretically estimated complexity derived in section 5.1.9 and E for numerical experiment.

by the direct solver and the corresponding iterative solver (ML, EML or geometric multigrid), respectively. The results show that the geometric multigrid yields the least number of iterations or more accurate approximation when convergence is not attained with 200 iterations. This is expected since ML and EML algorithms have smoothing only on the fine level while smoothing is performed on all levels (in addition to the local smoothing) for the geometric multigrid algorithm, and for elliptic-type PDEs performing smoothing on all levels typically provides better performance [145].



(a) ND (b) ML (c) EML

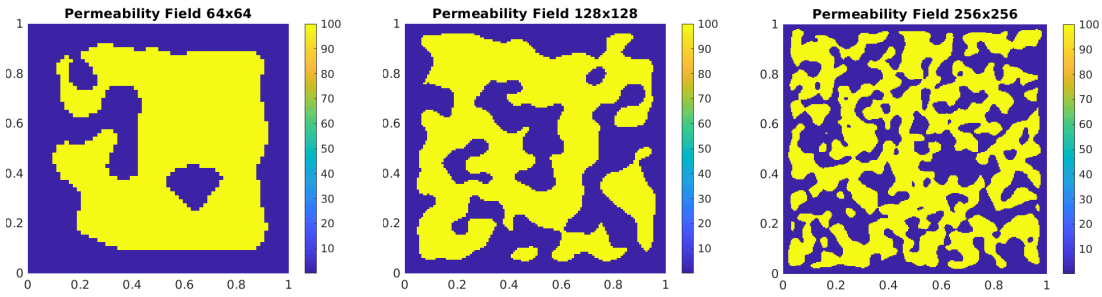
Figure 5.9: Example I: asymptotic and numerical estimates of back solve time complexity for EML, ML, and ND. Here, T stands for the theoretically estimated complexity derived in section 5.1.9 and E for numerical experiment.



(a) ND (b) ML (c) EML

Figure 5.10: Example I: asymptotic and numerical estimates of memory complexity for EML, ML, and ND. Here, T stands for the theoretically estimated complexity derived in section 5.1.9 and E for numerical experiment.

However, the proposed algorithms in this paper targets beyond elliptic PDEs, and for that reason it is not clear if smoothing on coarser levels helps reduce the iteration counts [124]. We would also like to point out that the least number of iterations in geometric multigrid does not translate directly to least overall time to solution which in turn depends on time per iteration and set-up cost for the three methods. In future work we will compare the overall time to solution for ML, EML and geometric multigrid. This challenging example clearly shows the benefits of high-order nature



(a) $N = 6$ (b) $N = 7$ (c) $N = 8$

Figure 5.11: Example II: discontinuous and heterogeneous permeability field [81] on 64^2, 128^2 and 256^2 meshes.

of HDG, that is, for all three methods as the solution order increases the solution is not only more accurate but also obtained with less number of GMRES iterations.

Between ML and EML, we can see that EML requires less number of iterations and attains more accuracy at higher levels (see, e.g., the results with $N = 8$ in Tables 5.3 and 5.4). The benefit of enrichment is clearly observed for $p = \{3, 4, 5\}$, in which EML is almost four orders of magnitude more accurate than ML (see last row and columns $4 - 6$ of Tables 5.3 and 5.4). For coarser meshes, the iteration counts of ML and EML are similar. Finally, it is interesting to notice that columns $4 - 7$ in Table 5.4, corresponding to $p = \{3, 4, 5, 6\}$, for EML (highlighted in blue) have similar iteration counts and accuracy when compared to columns $3 - 6$ in Table 5.5, corresponding to $p = \{2, 3, 4, 5\}$, for the geometric multigrid method.

Finally, solvers/preconditioners based on hierarchical interpolative decomposition [81], HSS [160] and compressible matrices [103] can provide lesser iteration counts/faster time to solution for this example as it is elliptic and have the low rank properties in the off-diagonal blocks. However, our approach is more general and applicable to hyperbolic problems such as the pure transport equation and other such equations which do not have that property. Moreover, for PDEs where the low

rank property is applicable HSS or similar compression techniques can be applied on top of our algorithm and can offer additional speedup. We will explore this avenue in our future studies.

N	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$
6	178	137	107	92	79	73
7	* (10^{-5})	* (10^{-8})	167	138	113	99
8	* (10^{-2})	* (10^{-2})	* (10^{-2})	* (10^{-4})	* (10^{-5})	* (10^{-7})

Table 5.3: Example II: number of ML-preconditioned GMRES iterations as the mesh and solution order are refined.

N	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$
6	180	132	115	98	81	72
7	* (10^{-7})	178	155	132	112	93
8	* (10^{-4})	* (10^{-5})	* (10^{-6})	* (10^{-8})	195	176

Table 5.4: Example II: number of EML-preconditioned GMRES iterations as the mesh and solution order are refined.

N	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$
6	157	114	97	85	74	69
7	* (10^{-8})	157	129	112	98	88
8	* (10^{-6})	* (10^{-7})	* (10^{-8})	190	176	170

Table 5.5: Example II: number of geometric-multigrid-preconditioned GMRES iterations as the mesh and solution order are refined.

5.2.3 Example III: Transport equation

In this section we apply ML and EML to a pure transport equation (2.5). Similar to [82, 108], we consider the velocity field $\boldsymbol{\beta} = (1 + \sin(\pi y/2), 2)$, forcing $f = 0$, and the inflow boundary conditions

$$g = \begin{cases} 1 & x = 0, 0 \leq y \leq 2 \\ \sin^6(\pi x) & 0 < x \leq 1, y = 0 \\ 0 & 1 \leq x \leq 2, y = 0 \end{cases}.$$

The solution is shown in Figure 5.12(a) and the difficulty of this test case comes from the presence of a curved discontinuity (shock) emanating from the inflow to

the outflow boundaries. In Table 5.6 we show the iteration counts for both ML- and EML-preconditioned GMRES for different solution orders and mesh levels. As can be seen, while $h-$scalability is not attained with both ML and EML, $p-$scalability is observed for both methods, i.e., the number of GMRES iterations is almost constant for all solution orders. Again EML takes less iteration counts than ML for all cases.

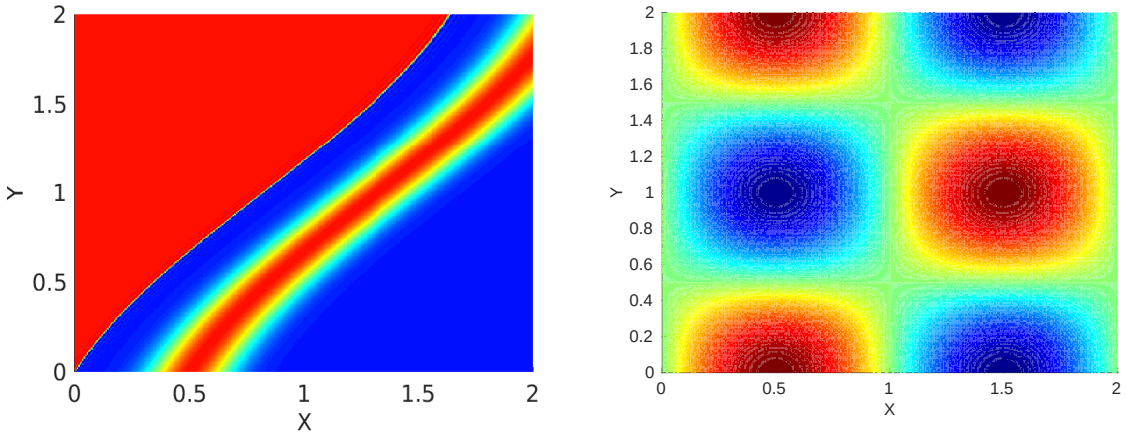
Table 5.7 shows the iteration counts for block-Jacobi preconditioned GMRES. Compared to ML and EML in Table 5.6, the iteration counts for block-Jacobi are higher, and for levels 7 and 8 block-Jacobi does not converge within the maximum number of iteration counts. This indicates though both ML and EML do not give $h-$scalable results, they provide a global coupling for the two-level algorithm and thus help in reducing the total number of iterations. Moreover, both the ML and EML algorithms are robust (with respect to convergence) even for solution with shock. It is important to point out that for pure transport problems it is in general not trivial to obtain $h-$scalable results unless some special smoother, which follows the direction of convection, is used [11, 77, 88, 151]. For this reason, the moderate growth in the iteration counts for both ML and EML algorithms is encouraging.

Next, we test the algorithms on a smooth exact solution (see Figure 5.12(b)) given by

$$u^e = \frac{1}{\pi}\sin(\pi x)\cos(\pi y).$$

All the other parameters are the same as those for the discontinuous solution considered above. Tables 5.8 and 5.9 show the number of ML-, EML-, and block-Jacobi-preconditioned GMRES iterations. Table 5.8 shows that the performance of ML- and EML-preconditioned GMRES is similar to the one observed for the elliptic equation with smooth solution in Table 5.2. Block-Jacobi preconditioned GMRES, on the other hand, is more or less independent of the smoothness of the solution as Table 5.9 for

the smooth solution is very similar to Table 5.7 for the discontinuous solution. Thus this example demonstrates that for the Poisson and the transport equations, unlike many standard iterative algorithms which depend on the nature of the PDE under consideration, the performance of ML and EML algorithms seems to depend only on the smoothness of the solution and otherwise is independent of the underlying PDE. This behavior seems to be due to their root in direct solver strategy. However, this statement cannot be claimed in general for any PDE as our numerical experiments with Helmholtz equation in high wave number regime shows dependence with wave number even for smooth solution. In our future work we will study this behavior in more detail and improve the robustness of the multilevel algorithms.



(a) Discontinuous solution

(b) Smooth solution

Figure 5.12: Example III: discontinuous and smooth solution for the transport equation on a 64×64 uniform mesh and $p = 6$ solution order.

5.2.4 Convection-diffusion equation

In this section we test the proposed algorithms for the convection-diffusion equation in both diffusion- and convection-dominated regimes. To that end, we con-

N	ML with GMRES						EML with GMRES					
	p						p					
	1	2	3	4	5	6	1	2	3	4	5	6
2	4	5	5	5	5	5	3	4	5	5	5	5
3	7	7	7	7	7	7	6	6	7	7	7	7
4	10	11	11	11	10	11	8	9	10	10	10	10
5	16	17	16	18	17	17	10	14	15	15	16	16
6	25	27	26	28	27	28	15	22	23	24	25	26
7	41	44	44	46	45	47	21	34	39	43	43	44
8	66	76	79	82	81	83	31	55	67	75	77	79

Table 5.6: Example III. Discontinuous solution: number of ML- and EML-preconditioned GMRES iterations.

N	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$
2	7	7	7	7	7	7
3	9	9	10	10	10	9
4	14	14	14	15	14	14
5	24	24	24	25	25	25
6	43	41	44	45	46	46
7	78	76	*	*	*	*
8	146	*	*	*	*	*

Table 5.7: Example III. Discontinuous solution: number of block-Jacobi preconditioned GMRES iterations.

N	ML with GMRES						EML with GMRES					
	p						p					
	1	2	3	4	5	6	1	2	3	4	5	6
2	5	4	5	4	4	3	3	4	4	4	3	3
3	6	6	6	5	5	3	5	5	5	4	3	2
4	10	9	9	8	7	5	7	7	6	4	2	1
5	15	14	13	13	10	8	8	9	7	2	1	0
6	23	22	20	20	17	13	10	11	2	1	0	0
7	36	37	37	34	30	17	12	7	1	0	0	0
8	58	63	65	62	48	21	12	2	0	0	0	0

Table 5.8: Example III. Smooth solution: number of ML- and EML-preconditioned GMRES iterations.

sider $f = 0$ in (2.34). We shall take some standard problems that are often used to test the robustness of multigrid algorithms for convection-diffusion equations.

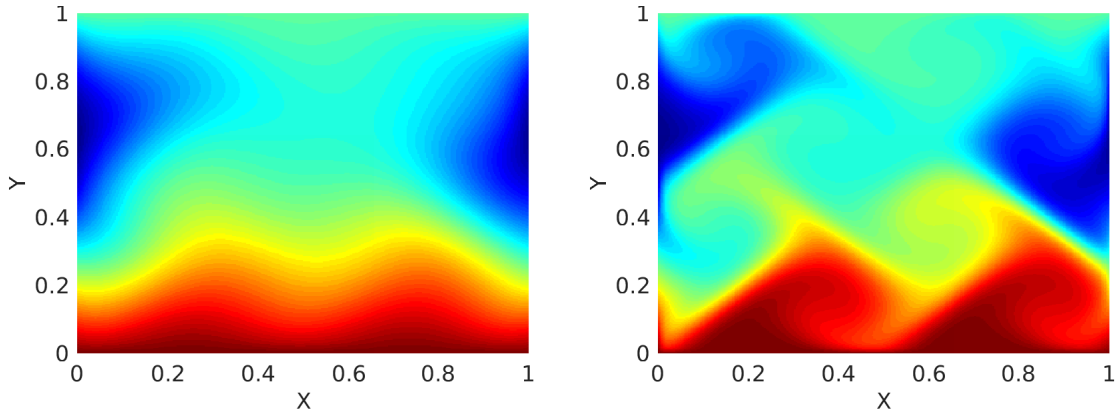
N	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$
2	6	7	7	7	7	7
3	9	9	9	9	9	9
4	14	13	14	14	14	14
5	23	22	23	24	24	24
6	41	39	43	44	45	68
7	76	74	*	*	*	*
8	142	*	*	*	*	*

Table 5.9: Example III. Smooth solution: number of block-Jacobi preconditioned GMRES iterations.

5.2.4.1 Example IV

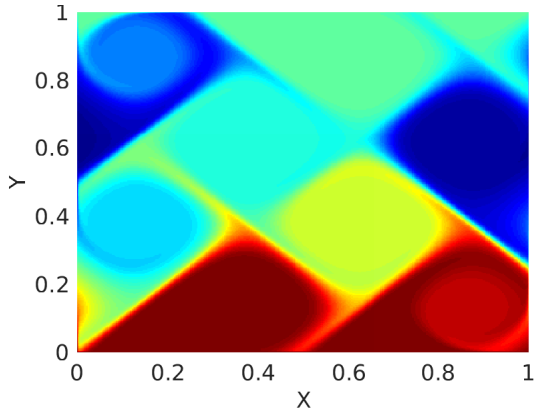
Here we consider an example similar to the one in [69]. In particular, we take $\kappa = 1, \nu = 0$, boundary condition $g_D = \cos(2y)(1-2y)$ and $\boldsymbol{\beta} = (-\alpha\cos(4\pi y), -\alpha\cos(4\pi x))$ in (2.34), where α is a parameter which determines the magnitude of convection velocity. In Figure 5.13, solutions for different values of α in the range $[10, 10^4]$ are shown. As α increases, the problem becomes more convection-dominated and shock-like structures are formed.

In Tables 5.10-5.13 are the iteration counts for ML- and EML-preconditioned GMRES with various values of α. We observe the following. In all cases, as expected, the iteration counts for EML are less than for ML. As the mesh is refined we see growth in iterations for both ML and EML, though it is less for EML than for ML. With increase in solution order the iterations remain (almost) constant, and in many cases decrease. For mildly-to-moderately convection-dominated, i.e. $\alpha \in [10, 10^3]$, both ML and EML are robust in the sense that their iteration counts negligibly vary with respect to α. For $\alpha = 10^4$, i.e., strongly convection-dominated regime, we see an increase in iteration counts for both algorithms, though the growth is much less pronounced for EML than for ML (especially with low solution orders $p = \{1, 2, 3, 4\}$).

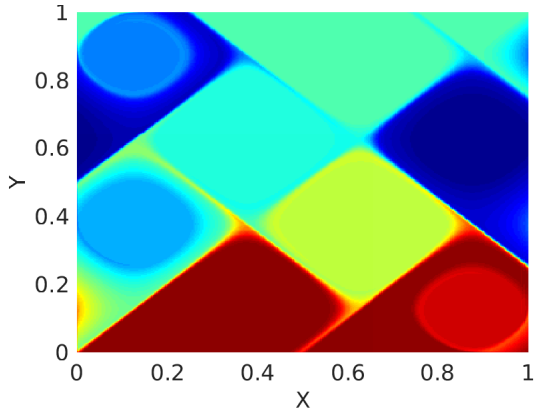


(a) $\alpha = 10$

(b) $\alpha = 10^2$



(c) $\alpha = 10^3$



(d) $\alpha = 10^4$

Figure 5.13: Example IV: solutions of the convection-diffusion equation for different values of α on a 64×64 uniform mesh and $p = 6$ solution order.

N	ML with GMRES						EML with GMRES					
	p						p					
	1	2	3	4	5	6	1	2	3	4	5	6
6	29	24	24	20	20	17	13	12	12	12	11	11
7	42	35	34	29	28	24	17	15	16	14	15	15
8	60	49	49	40	39	33	22	20	19	20	21	21

Table 5.10: Example IV. $\alpha = 10$: number of iterations for ML- and EML-preconditioned GMRES.

N	ML with GMRES						EML with GMRES					
	p						p					
	1	2	3	4	5	6	1	2	3	4	5	6
6	27	25	23	22	22	20	15	15	16	16	16	17
7	39	36	34	32	32	30	22	21	22	22	23	23
8	55	51	49	46	46	42	29	29	31	31	33	34

Table 5.11: Example IV. $\alpha = 10^2$: number of iterations for ML- and EML-preconditioned GMRES.

N	ML with GMRES						EML with GMRES					
	p						p					
	1	2	3	4	5	6	1	2	3	4	5	6
6	39	29	24	18	16	15	14	13	12	12	11	12
7	49	34	26	21	21	20	15	15	15	15	15	16
8	62	41	35	28	29	27	22	22	22	22	23	23

Table 5.12: Example IV. $\alpha = 10^3$: number of iterations for ML- and EML-preconditioned GMRES.

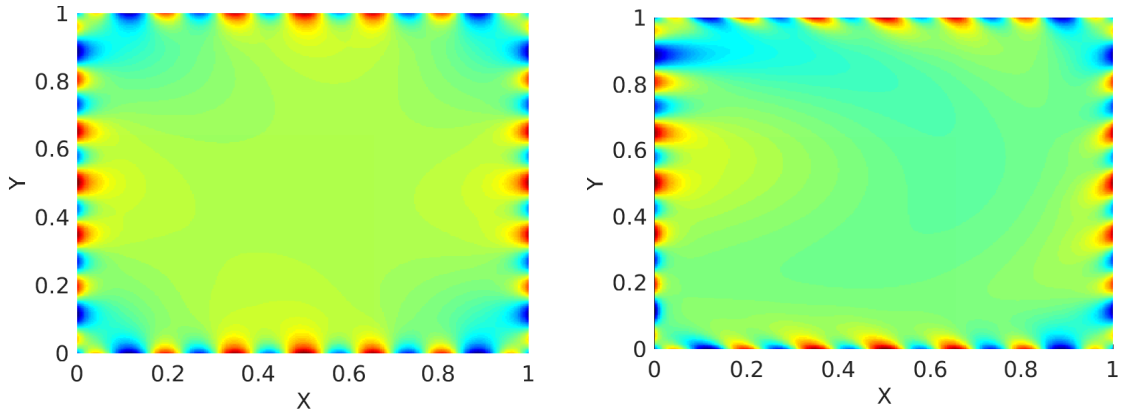
N	ML with GMRES						EML with GMRES					
	p						p					
	1	2	3	4	5	6	1	2	3	4	5	6
6	89	101	88	68	59	51	29	43	44	41	37	33
7	136	133	98	75	65	55	40	48	44	38	35	34
8	192	141	101	76	67	56	46	45	39	35	34	34

Table 5.13: Example IV. $\alpha = 10^4$: number of iterations for ML- and EML-preconditioned GMRES.

5.2.4.2 Example V

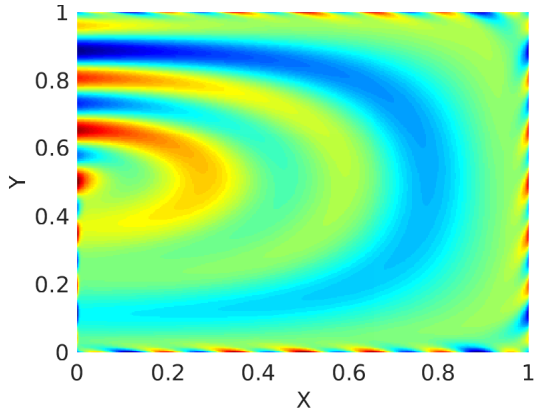
In this section a test case for multigrid method in [124] is considered. The parameters for this example are $\boldsymbol{\beta} = ((2y-1)(1-x^2), 2xy(y-1))$, boundary condition $g_D = \sin(\pi x) + \sin(13\pi x) + \sin(\pi y) + \sin(13\pi y)$ and $\nu = 0$. The solution fields for various values of κ in the range $[10^{-1}, 10^{-4}]$ are shown in Figure 5.14. As can be observed, the problem becomes more convection dominated when the diffusion coefficient κ decreases. Tables 5.14-5.17 present the iteration counts of ML- and EML-preconditioned GMRES for different values of κ. Again, the iteration counts

for EML are less than for ML. As the mesh is refined we see growth in iterations for both ML and EML, though it is less for EML than for ML. An outlier is the case of $\kappa = 10^{-4}$ in Table 5.17, where the number of EML-preconditioned GMRES iterations reduces as the mesh is refined and ML-preconditioned GMRES does not converge for $p = 2$ on mesh levels 6 and 7. At the time of writing, we had not yet found the reason for this behavior.

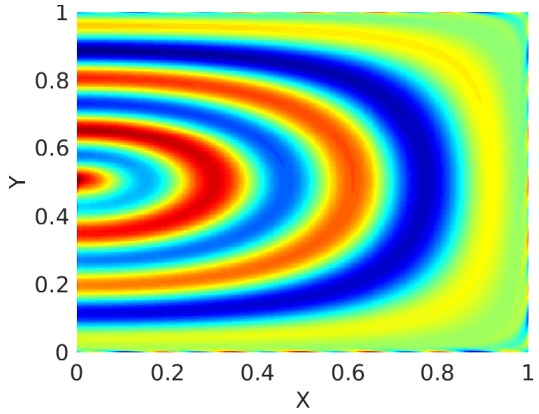


(a) $\kappa = 10^{-1}$

(b) $\kappa = 10^{-2}$



(c) $\kappa = 10^{-3}$



(d) $\kappa = 10^{-4}$

Figure 5.14: Example V: solutions of the convection-diffusion equation for different values of κ on a 64×64 uniform mesh and $p = 6$ solution order.

N	ML with GMRES						EML with GMRES					
	p						p					
	1	2	3	4	5	6	1	2	3	4	5	6
6	28	23	23	21	21	19	13	13	13	13	13	13
7	40	34	34	30	29	26	18	17	18	16	18	18
8	58	47	48	43	41	37	23	22	23	23	25	25

Table 5.14: Example V. $\kappa = 10^{-1}$: number of iterations for ML- and EML-preconditioned GMRES.

N	ML with GMRES						EML with GMRES					
	p						p					
	1	2	3	4	5	6	1	2	3	4	5	6
6	30	24	23	21	20	18	15	14	14	13	12	12
7	41	34	33	30	29	26	19	17	17	15	16	16
8	56	46	47	42	40	36	23	21	21	21	22	23

Table 5.15: Example V. $\kappa = 10^{-2}$: number of iterations for ML- and EML-preconditioned GMRES.

N	ML with GMRES						EML with GMRES					
	p						p					
	1	2	3	4	5	6	1	2	3	4	5	6
6	46	21	19	16	16	15	23	13	13	12	11	12
7	55	29	27	24	23	22	26	17	17	16	16	16
8	61	43	41	36	35	32	31	24	24	23	24	24

Table 5.16: Example V. $\kappa = 10^{-3}$: number of iterations for ML- and EML-preconditioned GMRES.

N	ML with GMRES						EML with GMRES					
	p						p					
	1	2	3	4	5	6	1	2	3	4	5	6
6	139	*	159	28	25	21	68	161	107	20	17	15
7	175	*	33	27	23	21	58	24	21	17	15	14
8	184	43	34	30	26	24	51	20	18	16	16	16

Table 5.17: Example V. $\kappa = 10^{-4}$: number of iterations for ML- and EML-preconditioned GMRES.

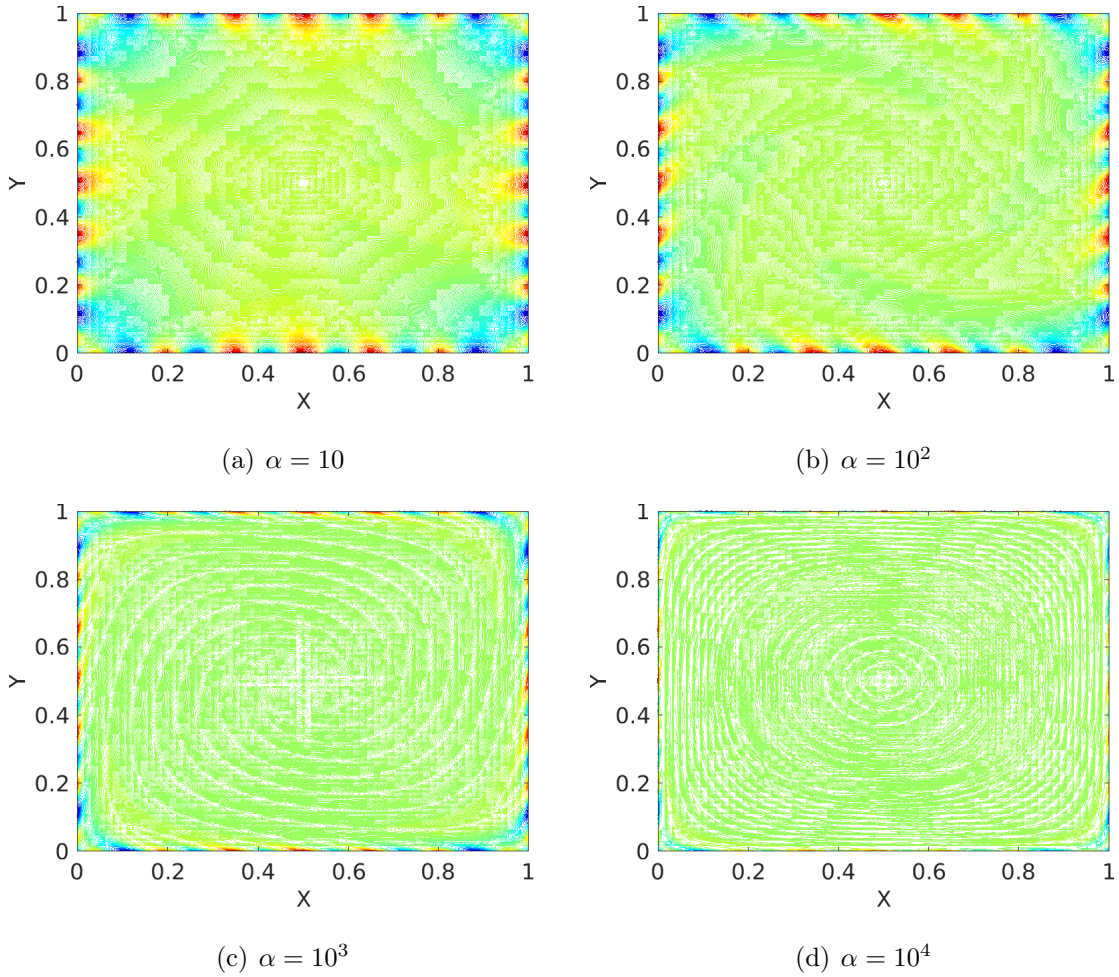
5.2.4.3 Example VI

Next we consider a test case from [9]. The parameters are $\boldsymbol{\beta} = (4\alpha x(x-1)(1-2y), -4\alpha y(y-1)(1-2x))$, boundary condition $g_D = \sin(\pi x) + \sin(13\pi x) + \sin(\pi y) + \sin(13\pi y)$, $\nu = 0$ and $\kappa = 1$. The solution fields for four values of α in $[10, 10^4]$ are shown in Figure 5.15. Similar to example IV, the problem becomes more convection-dominated as α increases. However, one difference is that the streamlines of the convection field in this case are circular [9]. This is challenging for geometric multigrid methods with Gauss–Seidel type smoothers of certain ordering if unknowns are not ordered in the flow direction. Since the block-Jacobi method, which is insensitive to direction, is used in ML and EML algorithms, we do not encounter the same challenge here. In Tables 5.18-5.21 are the iteration counts of ML- and EML-preconditioned GMRES for different values of α. As expected, all observations/conclusions made for example IV hold true for this example as well. The results for $\alpha \geq 10^3$ show that this case is, however, more challenging. Indeed, this example requires more iterations for both ML and EML, and in some cases convergence is not obtained within the 200-iteration constraint.

N	ML with GMRES						EML with GMRES					
	p						p					
	1	2	3	4	5	6	1	2	3	4	5	6
6	28	23	23	22	22	21	23	16	15	14	15	15
7	38	33	34	31	31	29	31	20	20	19	20	20
8	56	47	48	45	44	41	40	26	26	27	29	29

Table 5.18: Example VI. $\alpha = 10$: number of iterations for ML- and EML-preconditioned GMRES.

Examples IV, V and VI show that both ML and EML preconditioners behave especially well in both diffusion-dominated and moderately convection-dominated regimes. EML is more beneficial than ML in terms of robustness and iteration counts,



(a) $\alpha = 10$

(b) $\alpha = 10^2$

(c) $\alpha = 10^3$

(d) $\alpha = 10^4$

Figure 5.15: Example VI: solutions of the convection-diffusion equation for different α on a 64×64 uniform mesh and $p = 6$ solution order.

N	ML with GMRES						EML with GMRES					
	p						p					
	1	2	3	4	5	6	1	2	3	4	5	6
6	35	28	28	25	24	22	26	22	21	19	18	17
7	52	42	41	36	34	31	42	33	28	24	23	21
8	77	59	54	50	46	42	66	43	32	28	30	29

Table 5.19: Example VI. $\alpha = 10^2$: number of iterations for ML- and EML-preconditioned GMRES.

especially for low orders $p \leq 4$. The iteration counts of EML are also comparable to some of the AMG methods in [**?**] for examples V and VI

N	ML with GMRES						EML with GMRES					
	p						p					
	1	2	3	4	5	6	1	2	3	4	5	6
6	106	34	31	25	23	21	43	25	26	20	20	19
7	*	52	47	41	37	35	152	40	37	32	31	30
8	180	92	77	70	64	62	114	66	61	55	56	53

Table 5.20: Example VI. $\alpha = 10^3$: number of iterations for ML- and EML-preconditioned GMRES.

N	ML with GMRES						EML with GMRES					
	p						p					
	1	2	3	4	5	6	1	2	3	4	5	6
6	145	111	89	69	59	49	62	54	53	42	39	34
7	200	126	117	75	69	53	91	63	78	47	53	40
8	*	151	*	93	99	71	*	90	*	67	92	60

Table 5.21: Example VI. $\alpha = 10^4$: number of iterations for ML- and EML-preconditioned GMRES.

5.2.5 Three-dimensional example

In this section we test the iterative multilevel algorithm 4 on the 3D Poisson equation with a smooth exact solution given by

$$u^e = \frac{1}{\pi^2} \sin(\pi x) \cos(\pi y) \sin(\pi z).$$

The other parameters and settings of the numerical experiment are same as example I in section 5.2.2.1. The domain is a standard unit cube $\Omega = [0, 1]^3$ discretized with structured hexahedral elements. The number of elements corresponding to the number of levels in the multilevel hierarchy is shown in Table 5.22.

For the 3D calculations the multilevel approach described in section 5.1.2 is extended in a natural way. The level 1 separator has 12 faces and the separators in higher levels are projected such that all of them have same 12 faces with same (ML) or high (EML) polynomial order. In Table 5.23, the number of ML and EML preconditioned GMRES iterations are shown with respect to h and p refinements and

the conclusion is very similar to the one for the 2D problem in section 5.2.2.1. In Table 5.23, for $N = 5$ and $p = 6$ we ran into out of memory problem during coarse factorization and hence it is denoted with '-'.

In Figures 5.16-5.20, we compare ML and EML preconditioners to ND. Most of the trend is similar to that observed for the 2D problem in section 5.2.2.1 and the notable differences worth mentioning are as follows. The overall speedup of both ML and EML compared to ND is significantly better in 3D than in 2D and a maximum speedup of around 8 is observed for EML in Figure 5.16(a) and around 10 is observed for ML in Figure 5.16(b) for $p = 3$ and $N = 5$. Between ML and EML, ML is faster than EML as seen in Figure 5.16(c) even though the factor is not very high. This is different from 2D where in Figure 5.6(c) we found EML to be slightly faster than ML. ND faced out of memory problem for $N = 5$ and $p > 3$ whereas both ML and EML still work for $N = 5$ and $p = 4, 5$. Even for $p \leq 3$, the memory required by ML is 25 times less than ND at all orders and $N = 5$ as shown in Figure 5.17(b). EML requires 14 times less memory than ND at $N = 5$ and $p = 3$ as seen in Figure 5.17(a).

In terms of comparison with theoretical complexities in section 5.1.9, we see a very good agreement for the memory complexities in Figure 5.20. For the factorization and the back solve in Figures 5.18 and 5.19, we noticed the algorithms, especially ND, has not reached the asymptotic limit yet and hence there is some difference.

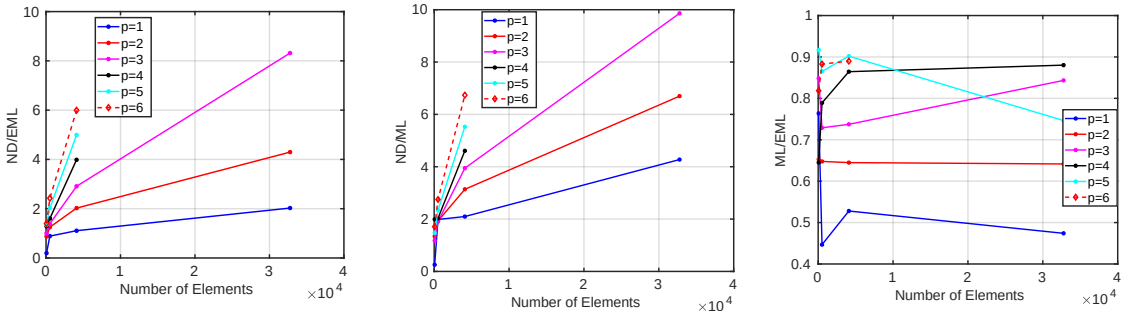
This preliminary study in 3D shows that the multilevel algorithms can offer significant speedups and memory savings compared to ND. Our future work will involve large scale parallel implementation of the multilevel algorithms for challenging problems in 3D.

Levels (N)	2	3	4	5
Elements	4^3	8^3	16^3	32^3

Table 5.22: 3D multilevel hierarchy.

N	ML with GMRES						EML with GMRES					
	p						p					
	1	2	3	4	5	6	1	2	3	4	5	6
2	6	6	7	5	4	2	4	6	5	4	2	1
3	11	9	8	6	5	3	9	7	6	3	1	0
4	16	12	10	8	6	4	12	8	5	1	0	0
5	23	17	14	11	8	-	14	8	2	0	0	-

Table 5.23: 3D Poisson equation: number of ML- and EML-preconditioned GMRES iterations as the mesh is refined (increasing N) and the solution order p increases.

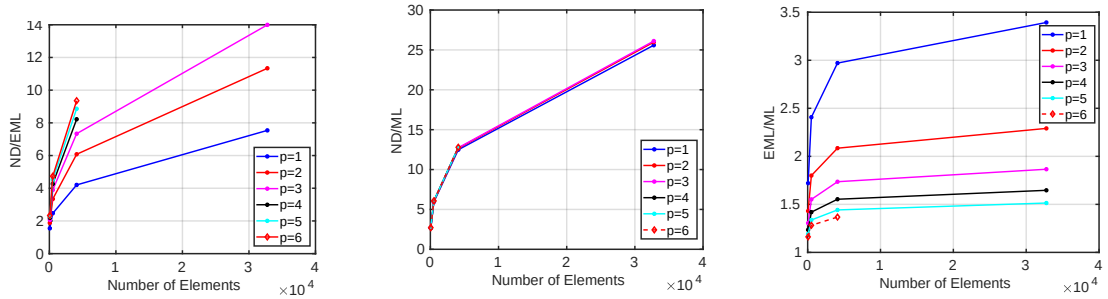


(a) EML versus ND (b) ML versus ND (c) EML versus ML

Figure 5.16: 3D Poisson equation: a comparison of time-to-solution for EML, ML, and ND algorithms.

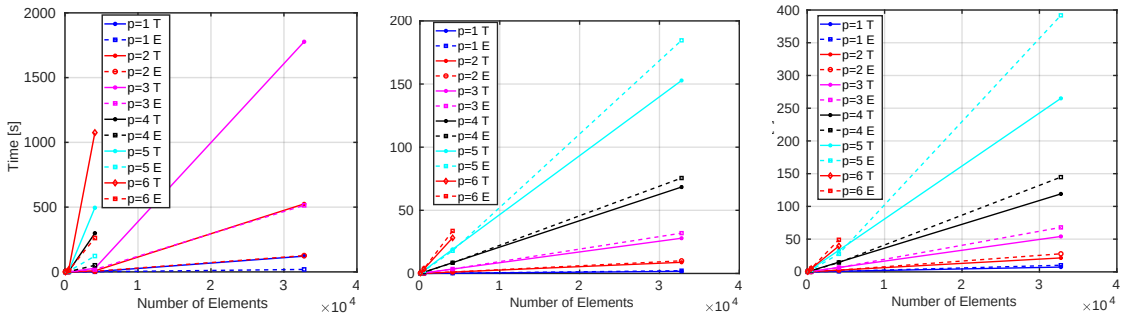
5.3 Discussion

In this chapter we have proposed a multilevel framework for HDG discretizations exploiting the concepts of nested dissection, domain decomposition, and high-order and variational structure of HDG methods. The chief idea is to create coarse solvers for domain decomposition methods by controlling the front growth of nested dissection. This is achieved by projecting the skeletal data at different levels to either same or high-order polynomial on a set of increasingly $h-$coarser edges/faces. When



(a) EML versus ND (b) ML versus ND (c) EML versus ML

Figure 5.17: 3D Poisson equation: a comparison of memory requirement for EML, ML, and ND algorithms.

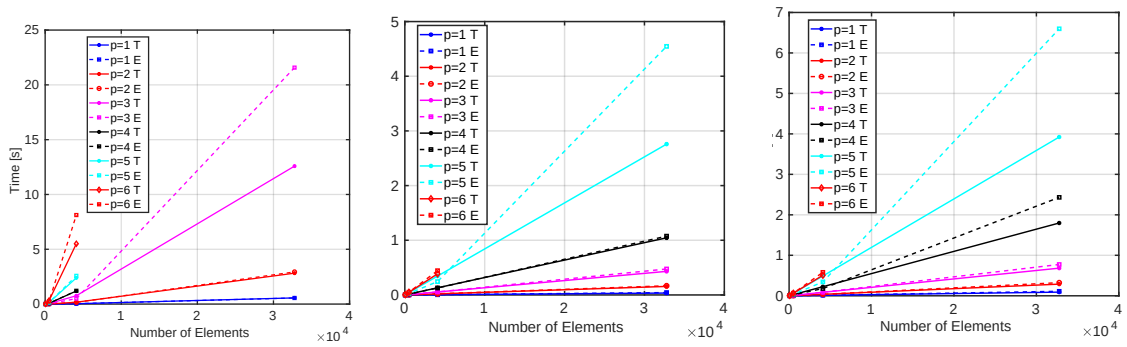


(a) ND (b) ML (c) EML

Figure 5.18: 3D Poisson equation: asymptotic and numerical estimates of factorization time complexity for EML, ML, and ND. Here, T stands for the theoretically estimated complexity derived in section 5.1.9 and E for numerical experiment.

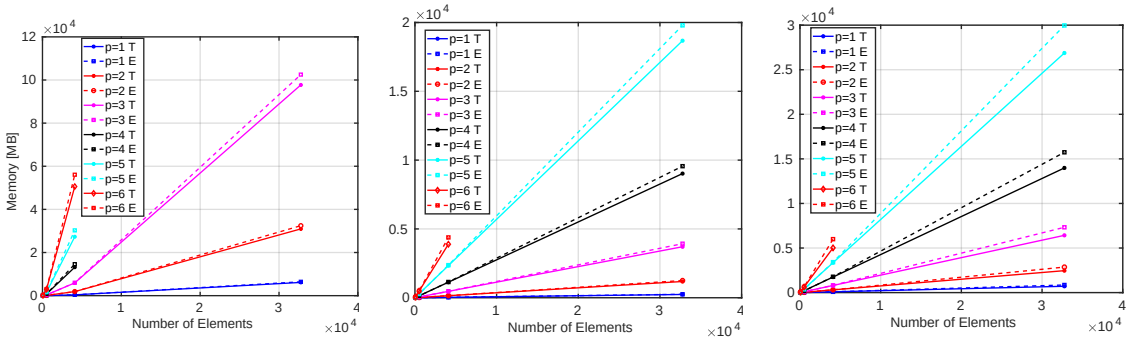
the same polynomial order is used for the projection we name the method *multilevel (ML) algorithm* and *enriched multilevel (EML) algorithm* for higher polynomial orders. The coarse solver is combined with a block-Jacobi fine scale solver to construct a two-level solver in the context of domain decomposition methods. We show that the two-level approach can also be interpreted as a multigrid algorithm with specific intergrid transfer and smoothing operators on each level. Our complexity estimates show that the cost of the multilevel algorithms is somewhat in between the cost of nested dissection and standard multigrid solvers.

We have conducted several numerical experiments with the Poisson equa-



(a) ND (b) ML (c) EML

Figure 5.19: 3D Poisson equation: asymptotic and numerical estimates of back solve time complexity for EML, ML, and ND. Here, T stands for the theoretically estimated complexity derived in section 5.1.9 and E for numerical experiment.



(a) ND (b) ML (c) EML

Figure 5.20: 3D Poisson equation: asymptotic and numerical estimates of memory complexity for EML, ML, and ND. Here, T stands for the theoretically estimated complexity derived in section 5.1.9 and E for numerical experiment.

tion, the transport equation, and the convection-diffusion equation in both diffusion- and convection-dominated regimes. The numerical experiments show that our algorithms are robust even for the transport equation with discontinuous solution and elliptic equation with highly heterogeneous and discontinuous permeability. For the convection-diffusion equation, the multilevel algorithms are scalable and reasonably robust (with respect to changes in parameters of the underlying PDE) from diffusion-dominated to moderately convection-dominated regimes. EML is more beneficial

than ML in terms of robustness and iteration counts, especially for low orders $p \leq 4$. Preliminary studies show that the multilevel algorithms can offer significant speedups and memory savings compared to the nested dissection direct solver for 3D problems.

We have demonstrated the applicability of our algorithms both as iterative solvers and as preconditioners for various prototypical PDEs in this work. One of the advantages of the algorithms is that they are designed to reduce dependence on the nature of the PDE being solved. In section 6.4.4 of the next chapter we utilize this and show an application of the multilevel approach in a block preconditioning strategy in the context of incompressible resistive MHD.

Chapter 6

A Block Preconditioner for HDG Trace Systems applied to Incompressible Resistive MHD

We have so far considered fairly simple prototypical equations and developed preconditioners and solvers for the trace systems arising from the HDG discretization. In this chapter we consider a multiphysics system involving fluid dynamics and electromagnetics and develop a block preconditioner for the HDG discretization. In particular, we consider the incompressible visco-resistive MHD equations which plays an important role in modeling low Lundquist number liquid metal flows, high Lundquist number large-guide-field fusion plasmas and low flow-Mach-number compressible flows [139]. Incompressible resistive MHD presents several challenges in terms of nonlinearity, coupled fluid and magnetic physics, incompressibility constraints in both velocity and magnetic fields to name a few.

This chapter is organized as follows. In section 6.1, we introduce the equations for the incompressible MHD system and the relevant non-dimensional parameters. Then in section 6.2, we present an HDG scheme for the discretization of the linearized MHD system and identify the block structure in it. We then proceed to introduce a block preconditioning strategy for the linear system in section 6.3. Finally, section 6.4 presents various 2D and 3D transient test cases to test the robustness and scalability of the block preconditioner.

6.1 Incompressible visco-resistive MHD system

The visco-resistive, incompressible magnetohydrodynamics equations are given by

$$\rho\frac{\partial \mathbf{u}}{\partial t} - \mu\Delta\mathbf{u} + \rho\mathbf{u}\cdot\nabla\mathbf{u} + \nabla q - \frac{1}{\mu_0}(\nabla\times\boldsymbol{b})\times\boldsymbol{b} = \mathbf{f}, \tag{6.1a}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{6.1b}$$

$$\frac{\partial \boldsymbol{b}}{\partial t} + \nabla\times(\frac{\eta}{\mu_0}\nabla\times\boldsymbol{b}) - \nabla\times(\mathbf{u}\times\boldsymbol{b}) + \nabla r = \mathbf{g}, \tag{6.1c}$$

$$\nabla\cdot\boldsymbol{b} = 0. \tag{6.1d}$$

Here, $\mathbf{u}$ is the fluid velocity field, $\boldsymbol{b}$ is the magnetic field, ρ is the fluid density, μ is the viscosity, μ_0 is the permeability of free space, η the resistivity and q is the mechanical pressure. These equations are obtained by combining the induction equation and the incompressible Navier–Stokes equations along with the addition of Lorentz force $-\mathbf{J}\times\boldsymbol{b}$ to the momentum equation, which describes the electromagnetic forces on the fluid. Here, $\mathbf{J}$ is the current which is related to the electric field $\mathbf{E}$ and magnetic field $\boldsymbol{b}$ through the generalized Ohm's law

$$\mathbf{E} + \mathbf{u}\times\boldsymbol{b} = \eta\mathbf{J}. \tag{6.2}$$

In order to arrive at this formulation we assume the displacement current to be negligible and use the Ampere's law

$$\nabla\times\boldsymbol{b} = \mu_0\mathbf{J}, \tag{6.3}$$

to eliminate $\mathbf{J}$ and write the Lorentz force in terms of magnetic field $\boldsymbol{b}$ only.

We consider ρ, μ and η to be constant and assume the magnetic source term $\mathbf{g}$ to be divergence-free. The variable r denotes the Lagrange multiplier which is used to enforce the solenoidality of magnetic field constraint. Without this variable, we would

have a over-determined system of equations. Thus it helps in enforcing the divergence-free constraint and at the same time it is just a "dummy variable" because in essence we are solving for zero. This can be seen by taking divergence of equation (6.1c) which gives $\Delta r = 0$ and together with homogeneous Dirichlet boundary conditions gives $r = 0$. More details about this approach can be found in [135, 138, 42, 140].

By choosing a length scale l_0, characteristic velocity scale u_0 and magnetic scale b_0 we non-dimensionalize (6.1) as shown in [140]. The non-dimensionalized MHD equations are given by

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla q - \frac{1}{\mathrm{Re}} \Delta \mathbf{u} - \kappa (\nabla \times \boldsymbol{b}) \times \boldsymbol{b} = \mathbf{f}, \tag{6.4a}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{6.4b}$$

$$\kappa \frac{\partial \boldsymbol{b}}{\partial t} + \frac{\kappa}{\mathrm{Rm}} \nabla \times (\nabla \times \boldsymbol{b}) - \kappa \nabla \times (\mathbf{u} \times \boldsymbol{b}) + \nabla r = \mathbf{g}, \tag{6.4c}$$

$$\nabla \cdot \boldsymbol{b} = 0. \tag{6.4d}$$

Here, $\mathrm{Re} := \frac{\rho l_0 u_0}{\mu}$ is the Reynolds number which measures the ratio of inertial forces to viscous forces, $\mathrm{Rm} := \frac{\mu_0 u_0 l_0}{\eta}$ is the magnetic Reynolds number which is the ratio of magnetic advection to magnetic diffusion, $\kappa := \frac{b_0^2}{\rho \mu_0 u_0^2}$ is the coupling parameter and is the ratio of electromagnetic forces to inertial forces. The parameters κ, Re and Rm are related by $\kappa = \frac{\mathrm{Ha}^2}{\mathrm{ReRm}}$, where $\mathrm{Ha} := \frac{b_0 l_0}{\sqrt{\mu \eta}}$ is the Hartmann number. We can also write κ as $\kappa = \frac{u_A^2}{u_0^2}$, where $u_A := \frac{b_0}{\sqrt{\rho \mu_0}}$ is the Alfvén speed. We refer the readers to [72, 107] for details on non-dimensional parameters in MHD system. Note that we have abused the notation here and all the variables in (6.4) represent the non-dimensional quantities.

We put (6.4) into first order form for discretizing with HDG and towards that end let us define the auxiliary variables $\mathbf{L}$ and $\mathbf{J}$ which represents the velocity gradient

and curl of magnetic field respectively. The first order system is given by

$$\mathrm{Re}\mathbf{L} - \nabla\mathbf{u} = \mathbf{0}, \tag{6.5a}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) + \nabla q - \nabla \cdot \mathbf{L} - \kappa(\nabla \times \boldsymbol{b}) \times \boldsymbol{b} = \mathbf{f}, \tag{6.5b}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{6.5c}$$

$$\frac{\mathrm{Rm}}{\kappa}\mathbf{J} - \nabla \times \boldsymbol{b} = \mathbf{0}, \tag{6.5d}$$

$$\kappa\frac{\partial \boldsymbol{b}}{\partial t} + \nabla r - \kappa\nabla \times (\mathbf{u} \times \boldsymbol{b}) + \nabla \times \mathbf{J} = \mathbf{g}, \tag{6.5e}$$

$$\nabla \cdot \boldsymbol{b} = 0. \tag{6.5f}$$

We refer to $\mathbf{J}$ as the current density or simply the current and it should be understood in a non-dimensional sense with the characteristic value defined by $J_0 = \frac{Rm}{\kappa}\frac{b_0}{\mu_0 l_0}$.

The MHD system (6.5) is equipped with the following set of initial conditions

$$\mathbf{u}(t=0) = \mathbf{u}_0, \qquad \boldsymbol{b}(t=0) = \boldsymbol{b}_0. \tag{6.6}$$

We also need to specify boundary conditions for the fluid components, magnetic components and the Lagrange multiplier. Since it is not important for the current discussion we will defer this till section 6.4 where we specify these details for each numerical experiment separately.

The HDG schemes for the MHD system are posed on a linearized version, and towards that we linearize (6.5) about a prescribed velocity $\mathbf{w}$ and a prescribed

magnetic field $\boldsymbol{d}$ [42]:

$$\mathrm{Re}\mathbf{L} - \nabla\mathbf{u} = \mathbf{0}, \tag{6.7a}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{w}) + \nabla q - \nabla \cdot \mathbf{L} - \kappa(\nabla \times \boldsymbol{b}) \times \boldsymbol{d} = \mathbf{f}, \tag{6.7b}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{6.7c}$$

$$\frac{\mathrm{Rm}}{\kappa}\mathbf{J} - \nabla \times \boldsymbol{b} = \mathbf{0}, \tag{6.7d}$$

$$\kappa\frac{\partial \boldsymbol{b}}{\partial t} + \nabla r - \kappa\nabla \times (\mathbf{u} \times \boldsymbol{d}) + \nabla \times \mathbf{J} = \mathbf{g}, \tag{6.7e}$$

$$\nabla \cdot \boldsymbol{b} = 0. \tag{6.7f}$$

Here $\mathbf{w}$ is assumed to reside in $H(div, \Omega)$ and be divergence-free, while $\boldsymbol{d}$ is assumed to reside in $H(div, \Omega) \cap H(curl, \Omega)$.

6.2 HDG for incompressible MHD

In this section we present the HDG scheme proposed in [140, 94].

Formulation 6.1. *Find* $(\mathbf{L}, \mathbf{u}, q, \mathbf{J}, \boldsymbol{b}, r, \hat{\mathbf{u}}, \widehat{\boldsymbol{b}}^t, \widehat{r}, \rho)$ *in* $\mathbf{G}_h \times \boldsymbol{V}_h \times W_h \times \boldsymbol{H}_h \times \boldsymbol{C}_h \times$

$S_h \times \widehat{\boldsymbol{V}}_h \times \widehat{\boldsymbol{C}}_h^t \times \widehat{S}_h \times \mathcal{P}^0(\partial T)$ *such that the local equations*

$$\mathrm{Re}\,(\mathbf{L},\mathbf{G})_T + (\mathbf{u},\nabla\cdot\mathbf{G})_T - \langle\hat{\mathbf{u}},\mathbf{G}\boldsymbol{n}\rangle_{\partial T} = 0, \tag{6.8a}$$

$$\begin{aligned}\left(\frac{\partial\mathbf{u}}{\partial t},\mathbf{v}\right)_T - (\nabla\cdot\mathbf{L},\mathbf{v})_T + (\nabla q,\mathbf{v})_T - \frac{1}{2}(\mathbf{u}\otimes\mathbf{w},\nabla\mathbf{v})_T + \frac{1}{2}(\nabla\mathbf{u},\mathbf{v}\otimes\mathbf{w})_T \\ + (\nabla\times\boldsymbol{b},\mathbf{v}\times\kappa\boldsymbol{d})_T + \left\langle\frac{1}{2}(\mathbf{w}\cdot\boldsymbol{n})\,\hat{\mathbf{u}} + \mathbf{S}_u(\mathbf{u}-\hat{\mathbf{u}}),\mathbf{v}\right\rangle_{\partial T} \\ - [1-\xi]\left\langle\boldsymbol{n}\times\left(\boldsymbol{b}^t-\widehat{\boldsymbol{b}}^t\right),\mathbf{v}\times\kappa\boldsymbol{d}\right\rangle_{\partial T} = (\mathbf{f},\mathbf{v})_T,\end{aligned} \tag{6.8b}$$

$$-(\mathbf{u},\nabla w)_T + \langle\hat{\mathbf{u}}\cdot\boldsymbol{n},w-\overline{w}\rangle_{\partial T} + \langle\overline{q}-\rho,\overline{w}\rangle_{\partial T} = 0, \tag{6.8c}$$

$$\frac{\mathrm{Rm}}{\kappa}(\mathbf{J},\mathbf{H})_T - (\boldsymbol{b},\nabla\times\mathbf{H})_T - \left\langle\boldsymbol{n}\times\widehat{\boldsymbol{b}}^t,\mathbf{H}\right\rangle_{\partial T} = 0, \tag{6.8d}$$

$$\begin{aligned}\kappa\left(\frac{\partial\boldsymbol{b}}{\partial t},\boldsymbol{c}\right)_T + (\nabla\times\mathbf{J},\boldsymbol{c})_T - (r,\nabla\cdot\boldsymbol{c})_T - (\mathbf{u}\times\kappa\boldsymbol{d},\nabla\times\boldsymbol{c})_T + \langle\widehat{r},\boldsymbol{c}\cdot\boldsymbol{n}\rangle_{\partial T} \\ + \langle([1-\xi]\,\mathbf{u}+\xi\hat{\mathbf{u}})\times\kappa\boldsymbol{d},\boldsymbol{n}\times\boldsymbol{c}\rangle_{\partial T} + \left\langle\beta_t\left(\boldsymbol{b}^t-\widehat{\boldsymbol{b}}^t\right),\boldsymbol{c}\right\rangle_{\partial T} = (\mathbf{g},\boldsymbol{c})_T,\end{aligned} \tag{6.8e}$$

$$(\nabla\cdot\boldsymbol{b},s)_T + \left\langle\frac{1}{\beta_n}(r-\widehat{r}),s\right\rangle_{\partial T} = 0, \tag{6.8f}$$

the conservation equations

$$-\left\langle-\mathbf{L}\boldsymbol{n}+q\boldsymbol{n}+\frac{1}{2}(\mathbf{w}\cdot\boldsymbol{n})\,\mathbf{u}+\mathbf{S}_u(\mathbf{u}-\hat{\mathbf{u}})+\kappa\boldsymbol{d}\times(\boldsymbol{n}\times\xi\boldsymbol{b}),\widehat{\mathbf{v}}\right\rangle_e = 0, \tag{6.8g}$$

$$-\left\langle\boldsymbol{n}\times\mathbf{J}+\beta_t\left(\boldsymbol{b}^t-\widehat{\boldsymbol{b}}^t\right)-\boldsymbol{n}\times([1-\xi]\,\mathbf{u}\times\kappa\boldsymbol{d}),\widehat{\boldsymbol{c}}^t\right\rangle_e = 0, \tag{6.8h}$$

$$-\left\langle\boldsymbol{b}\cdot\boldsymbol{n}+\frac{1}{\beta_n}(r-\widehat{r}),\widehat{s}\right\rangle_e = 0, \tag{6.8i}$$

and the additional constraint

$$\langle\hat{\mathbf{u}}\cdot\boldsymbol{n},\psi\rangle_{\partial T} = 0 \tag{6.8j}$$

hold for all $(\mathbf{G},\mathbf{v},w,\mathbf{H},\boldsymbol{c},s,\widehat{\mathbf{v}},\widehat{\boldsymbol{c}}^t,\widehat{s},\psi)$ *in* $\mathbf{G}_h\times\boldsymbol{V}_h\times W_h\times\boldsymbol{H}_h\times\boldsymbol{C}_h\times S_h\times\widehat{\boldsymbol{V}}_h\times\widehat{\boldsymbol{C}}_h^t\times\widehat{S}_h\times\mathcal{P}^0(\partial T)$. *In addition the pressure is subject to the constraint*

$$(q,1)_{\Omega_h} = 0.$$

Here, the volume spaces are defined as

$$\mathbf{G}_h := \left\{ \mathbf{G} \in \left[L^2(\Omega_h)\right]^{d\times d} \ : \ \mathbf{G}|_T \in \left[\mathcal{P}^p(T)\right]^{d\times d}, \forall T \in \Omega_h \right\}, \tag{6.9a}$$

$$\boldsymbol{V}_h := \left\{ \mathbf{v} \in \left[L^2(\Omega_h)\right]^{d} \ : \ \mathbf{v}|_T \in \left[\mathcal{P}^p(T)\right]^{d}, \forall T \in \Omega_h \right\}, \tag{6.9b}$$

$$W_h := \left\{ w \in L^2(\Omega_h) \ : \ w|_T \in \mathcal{P}^p(T), \forall T \in \Omega_h \right\}, \tag{6.9c}$$

$$\boldsymbol{H}_h := \left\{ \mathbf{H} \in \left[L^2(\Omega_h)\right]^{\tilde{d}} \ : \ \mathbf{H}|_T \in \left[\mathcal{P}^p(T)\right]^{\tilde{d}}, \forall T \in \Omega_h \right\}, \tag{6.9d}$$

$$\boldsymbol{C}_h := \left\{ \boldsymbol{c} \in \left[L^2(\Omega_h)\right]^{d} \ : \ \boldsymbol{c}|_T \in \left[\mathcal{P}^p(T)\right]^{d}, \forall T \in \Omega_h \right\}, \tag{6.9e}$$

$$S_h := \left\{ s \in L^2(\Omega_h) \ : \ s|_T \in \mathcal{P}^p(T), \forall T \in \Omega_h \right\}, \tag{6.9f}$$

where $\tilde{d}$ takes the value of one for 2D and three for 3D. We define the skeletal spaces as follows,

$$\widehat{\boldsymbol{V}}_h := \left\{ \widehat{\mathbf{v}} \in \left[L^2(\mathcal{E}_h)\right]^{d} \ : \ \widehat{\mathbf{v}}|_e \in \left[\mathcal{P}^p(e)\right]^{d}, \forall e \in \mathcal{E}_h \right\}, \tag{6.10}$$

$$\widehat{\boldsymbol{C}}_h^t := \left\{ \widehat{\boldsymbol{c}}^t \in \left[L^2(\mathcal{E}_h)\right]^{d-1} \ : \ \widehat{\boldsymbol{c}}^t|_e \in \widehat{\boldsymbol{C}}_h^t(e) \right\}, \tag{6.11}$$

$$\widehat{S}_h := \left\{ \widehat{s} \in L^2(\mathcal{E}_h) \ : \ \widehat{s}|_e \in \mathcal{P}^p(e), \forall e \in \mathcal{E}_h \right\}. \tag{6.12}$$

Here $\widehat{\boldsymbol{C}}_h^t(e)$ is a vector valued polynomial space with no normal component, defined by

$$\widehat{\boldsymbol{C}}_h^t(e) = \left\{ \sum_{i=1}^{d-1} \boldsymbol{t}^i \widehat{c}_{h,i} \ : \ \widehat{c}_{h,i} \in \mathcal{P}^p(e), \forall e \in \mathcal{E}_h \right\}, \tag{6.13}$$

where $\boldsymbol{t}^i$ are tangent vectors to e. The values $\xi = \frac{1}{2}$, $\beta_n = \beta_t = 1$ are chosen and the stabilization $\mathbf{S}_u$ is taken as

$$\mathbf{S}_u := \tau_t \mathbf{T} + \tau_n \mathbf{N}, \tag{6.14}$$

where $\mathbf{N} := \boldsymbol{n} \otimes \boldsymbol{n}$, $\mathbf{T} := -\boldsymbol{n} \times (\boldsymbol{n} \times .) = \mathbf{I} - \mathbf{N}$, $\tau_t = \frac{1}{2}\sqrt{4 + (\mathbf{w} \cdot \boldsymbol{n})^2}$ and $\tau_n = \frac{1}{2}\sqrt{8 + (\mathbf{w} \cdot \boldsymbol{n})^2}$.

The well-posedness of the local and global solvers of the scheme (6.1) and also the error analysis is shown in [140, 94]. For this scheme the volume velocity and

magnetic fields converge optimally as $\mathcal{O}(h^{p+1})$, whereas all the other volume variables converge as $\mathcal{O}(h^{p+1/2})$. The verification of this scheme for a number of prototypical MHD problems is shown in [140, 94].

Once the volume unknowns are expressed in terms of the skeletal unknowns through the local solvers, we use the conservation conditions to generate the global linear system which has the following block form [140]

$$\begin{bmatrix} A & -B^\top & E & G \\ B & 0 & 0 & 0 \\ F & 0 & C & J \\ H & 0 & K & L \end{bmatrix} \begin{bmatrix} \widehat{U} \\ \rho \\ \widehat{B}^t \\ \widehat{R} \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}. \tag{6.15}$$

We will use this block structure and develop a preconditioning strategy as shown in the next section.

6.3 A block preconditioner for the linear system

Before moving into the construction of the block preconditioner we first briefly explain the need for it in this case. If we want to precondition the linear system (6.15) using the multigrid method introduced in chapter 4 or multilevel method proposed in chapter 5 or algebraic multigrid methods (AMG), we cannot apply it directly because of the difference in nature of the trace unknowns. The unknowns $(\widehat{U}, \widehat{B}^t, \widehat{R})$ are all nodal skeletal unknowns belonging to $\mathcal{P}^p(e)$ whereas the edge average pressure ρ is an element-wise constant and is independent of the solution order p. Thus with any of the multigrid or multilevel methods, coarsening becomes an issue. This problem is also encountered in the linear systems arising from mixed finite element methods and a strategy to tackle this issue is block preconditioning. The idea is to identify and group blocks corresponding to different unknowns and use approximate block inverses for preconditioning [53, 54, 132, 43, 119]. Thus we use similar techniques to come up with a preconditioner for the linear system (6.15).

We first rewrite equation (6.15) into the saddle point form as follows

$$\begin{bmatrix} A & E & G & -B^\top \\ F & C & J & 0 \\ H & K & L & 0 \\ B & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \widehat{U} \\ \widehat{B}^t \\ \widehat{R} \\ \rho \end{bmatrix} = \begin{bmatrix} F_1 \\ F_3 \\ F_4 \\ F_2 \end{bmatrix}. \tag{6.16}$$

Denoting the 3×3 block corresponding to the unknowns $(\widehat{U}, \widehat{B}^t, \widehat{R})$ as $\mathcal{F}$ we can write the matrix as

$$\begin{bmatrix} \mathcal{F} & -B^\top \\ B & 0 \end{bmatrix}, \tag{6.17}$$

where we have abused the notation and denoted $[B \quad 0 \quad 0]$ as B. For a 2×2 block matrix such as (6.17) its block inverse (assuming $\mathcal{F}^{-1}$ exists) can be written as [146]

$$\begin{bmatrix} \mathcal{F} & -B^\top \\ B & 0 \end{bmatrix}^{-1} = \begin{bmatrix} \mathcal{F}^{-1} & \mathcal{F}^{-1}B^\top \mathcal{S}^{-1} \\ 0 & \mathcal{S}^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -B\mathcal{F}^{-1} & I \end{bmatrix}, \tag{6.18}$$

where $\mathcal{S} := B\mathcal{F}^{-1}B^\top$ is the Schur complement. Now, when we use the upper triangular matrix of the inverse (6.18) as a right preconditioner for the saddle point matrix (6.17) we get

$$\begin{bmatrix} \mathcal{F} & -B^\top \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathcal{F}^{-1} & \mathcal{F}^{-1}B^\top \mathcal{S}^{-1} \\ 0 & \mathcal{S}^{-1} \end{bmatrix} = \begin{bmatrix} I & 0 \\ B\mathcal{F}^{-1} & I \end{bmatrix}. \tag{6.19}$$

All the eigenvalues of the preconditioned matrix have the value 1, and hence with a Krylov subspace method such as GMRES at most two iterations are needed to solve the system [111, 53, 54].

However, the problem with this ideal preconditioner is that we need inverses of $\mathcal{F}$ and $\mathcal{S}$ which are expensive to compute. Hence the natural idea is to use approximations of these inverses in the construction of the preconditioner. First, let us consider the approximation for the inverse of the Schur complement matrix $\mathcal{S}$. We will follow the approach shown in [53, 54] for incompressible Navier–Stokes equations to derive an approximation for the inverse of the Schur complement.

To that extent, if we can find an approximate commutator $\tilde{\mathcal{F}}$ such that

$$B^\top \tilde{\mathcal{F}} \approx \mathcal{F} B^\top, \tag{6.20}$$

then pre-multiplying by $\mathcal{F}^{-1}$ and post-multiplying by $\tilde{\mathcal{F}}^{-1}$ (assuming both inverses exists) both sides of the above equation we get

$$\mathcal{F}^{-1} B^\top \approx B^\top \tilde{\mathcal{F}}^{-1}. \tag{6.21}$$

Using (6.21) we can write an approximate inverse for the Schur complement as

$$\mathcal{S}^{-1} = \left(B\mathcal{F}^{-1}B^\top\right)^{-1} \tag{6.22}$$

$$\approx \left(BB^\top \tilde{\mathcal{F}}^{-1}\right)^{-1} \tag{6.23}$$

$$\tilde{\mathcal{S}}^{-1} = \tilde{\mathcal{F}}\left(BB^\top\right)^{-1}. \tag{6.24}$$

Now the only thing remaining is to find the approximate commutator $\mathcal{F}$ and we can use least squares minimization for that. Solving for the normal equations corresponding to (6.20) gives [53]

$$\tilde{\mathcal{F}} = \left(BB^\top\right)^{-1} B\mathcal{F}B^\top. \tag{6.25}$$

Substituting the above equation in the approximate inverse for Schur complement (6.24) we get

$$\tilde{\mathcal{S}}^{-1} = \left(BB^\top\right)^{-1} B\mathcal{F}B^\top \left(BB^\top\right)^{-1}. \tag{6.26}$$

This is called BFBT approximation (because of the middle term sandwiched between two inverses) of the inverse Schur complement in the literature [54, 53, 132] and has been successfully used in the context of Stokes and incompressible Navier–Stokes equations discretized with finite volume, finite difference and mixed FEM methods. In those cases, $BB^\top$ corresponds to the Poisson operator and instead of taking inverses

one typically use a geometric multigrid or AMG cycle [54, 53]. Also, for approximating $\mathcal{F}^{-1}$, AMG or geometric multigrid cycles are used [54, 53]. Scaled versions of the BFBT approximation are proposed in [53, 104, 132] in order to improve the robustness.

In the context of HDG discretization of the incompressible MHD system, $BB^\top$ corresponds to a matrix of size $N_T \times N_T$ which is much smaller in size compared to $\mathcal{F}$. It is independent of the solution order p and the bandwidth is also small since it corresponds to the edge-average pressure which is piecewise constant per element. Thus as a first step we will use a parallel sparse direct solver Superludist [97] for taking inverse of this matrix in the BFBT approximation. In our future work we will replace this part with preconditioned conjugate gradient solver of lenient tolerance to improve scalability. Some of the initial studies conducted in this direction shows promise.

For approximating $\mathcal{F}^{-1}$, we use one v-cycle of AMG solver from the ML library [66] of Trilinos project [80]. Similar to [139], we order the unknowns in the nodal block such that the degrees-of-freedom within each node appear consecutively. This helps to preserve the coupling between different variables during coarsening. The ordering of unknowns within each node is $(\widehat{U}, \widehat{B}^t, \widehat{R})$, while other orderings are also possible and it can affect the performance of the AMG cycle we intend to compare them in our future work. We use non-smoothed and uncoupled aggregation with a sparse direct solver on the coarsest level as in [139]. More details about the aggregation strategy employed in AMG can be found in [139].

We also use one v-cycle of the multilevel solver presented in chapter 5 and compare the results with that of AMG in section 6.4.4. In our numerical studies we observed that the performance of the scaled BFBT approximations is very similar to

the non-scaled one (6.26) and hence we use that in all our experiments. In summary we use a right preconditioner

$$\begin{bmatrix} \tilde{\mathcal{F}}^{-1} & \tilde{\mathcal{F}}^{-1}B^{\top}\tilde{\mathcal{S}}^{-1} \\ 0 & \tilde{\mathcal{S}}^{-1} \end{bmatrix}, \tag{6.27}$$

where $\tilde{\mathcal{F}}^{-1}$ is one v-cycle of AMG or the multilevel algorithm 4 and the inverse of Schur complement is approximated by the BFBT approximation (6.26).

6.4 Numerical results

In this section we test the performance of the block preconditioner for some of the transient test cases in incompressible resistive MHD. In particular, we consider 2D and 3D versions of the island coalescence problem, hydromagnetic Kelvin-Helmholtz (HMKH) instability and hydromagnetic lid-driven cavity problems. We use quadrilateral elements in 2D and hexahedral elements in 3D. For time integration we use the backward Euler time stepping for all the 3D test cases and five stage fourth order diagonally implicit Runge-Kutta (DIRK) method of [87] for all the 2D test cases. For the nonlinear solver we employ the Picard iteration scheme with a stopping criterion based on the discrete norm of the solution update vector given in [139] with the absolute and relative tolerance values set as 10^{-6} and 10^{-4} respectively. For the stopping tolerance of the linear solver, apart from the HMKH test case, we set the value to be 10^{-6} multiplied by the norm of the right hand side of the Picard linear system. For the HMKH test case, it turns out we need a stricter tolerance of 10^{-9} (without multiplied by the norm of the right hand side) to make the Picard iterations converge.

For all the parallel results, we implemented our algorithms on top of the deal.II FEM library [10, 3]. The weak scaling studies are conducted in the Knights Landing (KNL) nodes of the Stampede2 supercomputer at the Texas Advanced Computing Center. Each node of KNL consists of 68 intel Xeon Phi 7250 1.4GHz processors with

4 threads per core. It has 96GB DDR4 RAM along with 16GB high speed MCDRAM which acts as L3 cache. We have used only pure MPI parallelism even though our deal.II code has task based parallelism using thread building blocks (TBB) in addition to MPI. The reasons for this choice are: (i) to have memory locality and avoid memory contention which may complicate the weak scaling studies (ii) the main focus of this study, which is the linear solver part, uses ML from Trilinos which does not have the threads support. In our future work we will use the latest MueLu library [121] instead of ML as it provides support for threads and GPU.

6.4.1 Magnetic reconnection - Island coalescence

Magnetic reconnection is a fundamental phenomenon by which a magnetic field changes its structure and is accompanied by conversion of magnetic field energy into plasma energy and transport [139]. Reconnection is possible only in a resistive MHD model as the ideal MHD conserves the magnetic flux and hence prevents any change in its structure. Many physical phenomena which occurs in space such as solar flares, coronal mass ejections involve magnetic reconnection and it is one of the driving factors for them to happen. It is also important in a laboratory scenario, especially in fusion experiments to understand and control plasma disruptions which can lead to loss in plasma confinement and also damages to the machine. Since fusion reactors like tokamak are typically designed to handle only certain maximum number of these eruptions, understanding and controlling these phenomena is of significant interest to the fusion and in general plasma community. More details about magnetic reconnection can be found in [14, 72, 73].

While magnetic reconnection is important for its physical significance, it is also characterized by disparate spatial and temporal scales which serves as an ideal test bed for testing the robustness of our preconditioners/solvers. In this section we

consider a specific reconnection problem which is the island coalescence studied in [139, 30]. It initially consists of two islands embedded in a Harris current sheet as shown in Figures 6.2(a), 6.5(a). A perturbation to the initial configuration and the combined magnetic field causes the center of the islands (referred as the o-points) to move towards each other and eventually coalesce to form one island. When the reconnection happens, the islands form a 'x' structure in the center of the domain as shown in Figures 6.2(d), 6.2(e), 6.6 and this is known as the x-point. In what follows we will briefly describe the settings of this problem and then evaluate the performance of our preconditioner in this case.

The domain is $[-1,1]\times[-1,1]$ in 2D and $[-1,1]\times[-1,1]\times[-1,1]$ in 3D. The boundary conditions are periodic in the x-direction on the left and right faces and also in the z-direction on the back and front faces. On the top and bottom faces in the y-direction, for the magnetic part, perfect conducting boundary conditions described by zero normal magnetic field $\boldsymbol{b}\cdot\mathbf{n}=0$ and zero tangential electric field $\mathbf{n}\times\mathbf{E}=0$ is applied. For fluid part on the top and bottom faces, mirror boundary conditions described by zero normal velocity $\mathbf{u}\cdot\mathbf{n}=0$ and zero shear stress are applied. The Lagrange multiplier r is set as zero on all the boundaries. We refer the readers to [140] for an application of these boundary conditions in the HDG setting.

The initial conditions consists of zero fluid velocity ($\mathbf{u}^0=\mathbf{0}$), and the magnetic field given by

$$\boldsymbol{b}^0=\left(\frac{\sinh(2\pi y)}{\cosh(2\pi y)+\epsilon\sinh(2\pi x)},\frac{\epsilon\sin(2\pi x)}{\cosh(2\pi y)+\epsilon\sinh(2\pi x)},0\right),$$

where in 2D the first two components of the field values are used. Here, ϵ refers to the width of the island and we choose it to be 0.2 as in [139]. In order for the initial configuration to be in equilibrium a forcing of $\mathbf{g}=\nabla\times\mathbf{J}^0$ is used where

$$\mathbf{J}^0:=\mathbf{J}(t=0)=\left(0,0,-\frac{2\pi\kappa\left(1-\epsilon^2\right)}{\mathrm{Rm}\left(\cosh(2\pi y)+\epsilon\sinh(2\pi x)\right)}\right).$$

For the momentum equation zero forcing ($\mathbf{f} = \mathbf{0}$) is selected. To set the islands into motion in a reproducible manner rather than relying on the accumulation of round-off error an initial perturbation of

$$\delta \boldsymbol{b}^0 = \sigma \left(\frac{\pi}{2} \cos(\pi x) \sin\left(\frac{\pi x}{2}\right) \cos(\pi z), -\pi \sin(\pi x) \cos\left(\frac{\pi y}{2}\right) \cos(\pi z), 0 \right),$$

is used with the value of $\sigma = 10^{-3}$ which sets the magnitude of perturbation [140]. For 2D, the first two components are used without the $z-$term. As described in [140] we choose the characteristic velocity as Alfvén speed which gives $\kappa = 1$. Also in all our numerical experiments we set the fluid Reynolds number and magnetic Reynolds number (which is Lundquist number in this case) equal to each other.

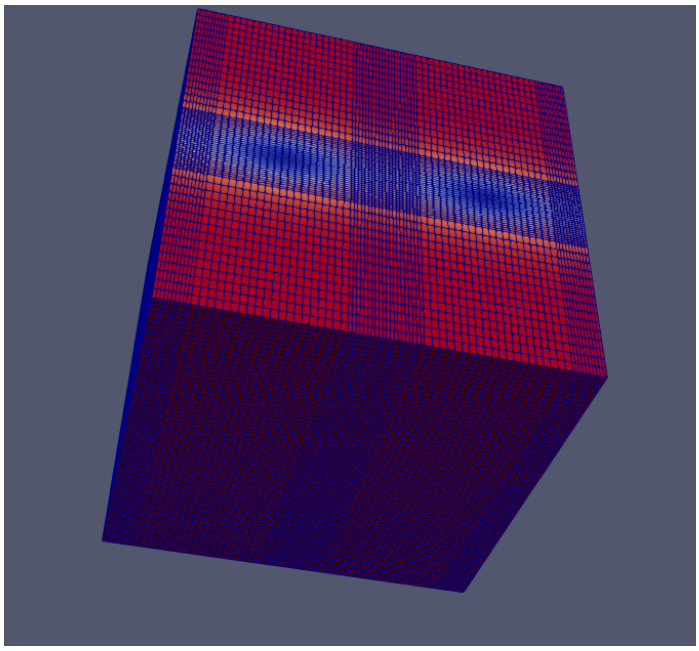


Figure 6.1: A $16 \times 12 \times 14$ clustered mesh with solution order $p = 5$ used for the simulation of the island coalescence problem at $\mathrm{Rm} = 10^3$. The mesh is colored by the $z-$component of the current (J_z) at time $t = 0.1$.

Let us choose $\mathrm{Rm} = 10^3$, and a $16 \times 12 \times 14$ clustered mesh of order $p = 5$ as shown in Figure 6.1. Figure 6.2, shows the evolution of the $z-$component of current

by taking a slice of the contour plot at $z = 0$. As can be seen the perturbation in the $z-$direction causes the current contours to bend and results in a highly kinked state as evidenced in Figures 6.2(d), 6.2(e) and 6.2(f). Our results show good agreement (visually) with the results in [30, 139].

Now we compare the performance of three preconditioners for this problem in weak scaling sense i.e., we increase the problem size proportionally with the increase in number of processors so that the number of elements per processor remains constant. We take 4^3, 8^3, 16^3 and 32^3 elements and 2, 16, 128 and 1024 processors respectively so that we have 32 elements per core. A time stepsize of 0.1 is selected and all the results are averaged over six time steps. The nonlinear Picard solver takes on average $3.2 - 4$ iterations in all the cases.

First of the preconditioners is a one level domain decomposition method with an incomplete factorization sub-domain solver with zero fill-in (ILU(0)) and overlap of one[1] (denoted as DD, ILU(0) in Figure 6.3 and let us refer this as DD with ILU(0)). The other two preconditioners are the block preconditioners given in equation (6.27) with BFBT approximation for $\tilde{\mathcal{S}}^{-1}$ and one AMG v-cycle for $\tilde{\mathcal{F}}^{-1}$. The difference between them is the smoothing inside AMG, in one of them we use the ILU(0) smoother of overlap one (denoted as BFBT+AMG, ILU(0) in Figure 6.3 and let us refer this as BFBT+AMG with ILU(0)), whereas in the other one we use the GMRES preconditioned by ILU(0) of overlap zero[2] as smoother (denoted as BFBT+AMG, GMRES in Figure 6.3 and let us refer this as BFBT+AMG with GMRES). The reason for these non-standard choice of smoothers is, classical smoothers like Jacobi, Chebyshev and Gauss–Seidel did not result in converging iterations. This is also observed in [98] for

[1]Here, overlap one means each processor will include its own set of rows and in addition it also includes the rows corresponding to its non-zero columns.

[2]Here, overlap zero means each processor will include only its own set of rows and thus no communication is needed.

linear systems arising from stabilized FEM discretization of MHD. Thus it indicates that strong smoothers are needed for AMG cycles applied to linear systems coming from incompressible resistive MHD.

We perform three pre- and three postsmoothing steps (in the case of GMRES smoother, these denote the number of inner iterations), whereas in the DD with ILU(0) preconditioner we perform three smoothing steps. The outer iterations are performed with GMRES for the DD with ILU(0) and the BFBT+AMG with ILU(0) preconditioners. For the BFBT+AMG with GMRES, we use flexible GMRES (FGMRES) [133] for the outer iterations due to the nonlinear nature of the inner iterations. In both cases similar to [139] we use non-restarted versions as it may result in degradation of convergence.

We compare the performance in terms of average iterations per Picard step and average time per Picard step in Figure 6.3. From Figure 6.3(a) for solution order $p = 4$, we can see that the iterations of DD with ILU(0) preconditioner increases much faster than the other two block preconditioners and this is due to the lack of coarse solvers. Similar results are also observed in [139] for the stabilized FEM discretizations. The average linear iterations per Picard step for the two block preconditioners are much less than the one level domain decomposition preconditioner and the growth with number of unknowns/processors (here the number of unknowns refers to the number of trace unknowns only) is also very mild. Between ILU(0) and GMRES smoothers we can see that the GMRES smoother takes slightly more iterations than the ILU(0).

In Figure 6.3(b), we compare the average time per Picard step for the three preconditioners and again the DD with ILU(0) preconditioner takes more time compared to the block preconditioners. However, the fact to notice is that the large

difference in iteration count evidenced in Figure 6.3(a) between DD with ILU(0) and BFBT+AMG with ILU(0) is not much reflected in the timing figure in 6.3(b). This is because ILU(0) with overlap one spends most of the time in the setup cost and very little time in the application of the preconditioners. Since the setup cost in both DD with ILU(0) and BFBT+AMG with ILU(0) are very similar the difference comes only from the application of the preconditioners which is high for DD with ILU(0) due to larger number of iterations. The scenario for the GMRES smoother is exactly opposite with very little time spent for the setup and most of the cost coming from the application or solving part of the preconditioner. This reflects in the timing results in Figure 6.3(b) with BFBT+AMG with GMRES smoother taking the least time (in spite of its iteration count bit higher than ILU(0) smoother) and more than two times faster than the other two preconditioners.

Another important aspect to notice is, ILU(0) smoothers with overlap one require lot of memory and this is much pronounced at high solution orders. In our numerical experiments we found that for solution orders greater than four, in the KNL nodes of Stampede2, we are able to use only 8 cores per node or less even though the system has 68 cores per node due to memory limitations. The GMRES smoother (GMRES preconditioned by ILU(0) with overlap zero) on the other hand has less memory requirements than the ILU(0) smoother with overlap one and this is the reason we test only BFBT+AMG with GMRES smoother for solution orders $p = 5, 6$ in Figure 6.4.

In Figure 6.4, the average number of iterations and time per Picard step are shown for the BFBT+AMG with GMRES smoother for solution orders $p = 5, 6$. The iteration counts are very similar to those observed for $p = 4$ in Figure 6.3(a) with a mild increase with respect to mesh refinements. The time per Picard step however, for $p = 5$ with 1024 cores and $p = 6$ with 2048 cores show a significant increase compared

to 128 and 256 cores respectively. This is a result of two things, (i) increase in iteration count; (ii) coarsening in AMG. We have not done repartitioning with the uncoupled aggregation performed in AMG and at high processor counts this resulted in fewer levels (3 or 4) in the AMG hierarchy with larger problem sizes on the coarsest level. Since, we use a serial sparse direct solver on the coarsest level the timing increased. This trend is observed in the other two numerical experiments also and especially in Figure 6.14 for the lid driven cavity problem. In that case we have almost constant iteration count which is reflected in the timing till 256 processors and after that both in 1024 and 2048 processors we see some increase in timing. In our ongoing work we are experimenting with different coarsening strategies in AMG and the initial studies have shown promise. We will report these findings as well as optimization of the other components used in the block preconditioning strategy in future.

$N_T = 32,768$, $N_{trace} = 15.2M$, p=4, $\Delta t = 0.1$, averaged over 6 time steps				
#cores	time/Picard [s]	N_T/core	Trace dof/core	efficiency [%]
128	211.6	256	118.4K	-
256	126.8	128	59.2K	83.4
512	64.2	64	29.6K	82.4
1024	36.3	32	14.8K	73
2048	19.4	16	7.4K	68
4096	16.6	8	3.7K	40
6144	14.9	5.3	2.5K	29.6

Table 6.1: 3D island coalescence problem. Strong scaling study for BFBT+AMG with GMRES smoother for solution order $p = 4$. The simulation is carried out in Skylake nodes of Stampede2 supercomputer.

In Tables 6.1 and 6.2 we study the strong scalability of BFBT+AMG with GMRES smoother for solution orders $p = 4, 6$. We see a good strong scaling performance and the efficiency improves with increase in order due to increased computation per core. For the strong scaling study we used the Skylake nodes of Stampede2 supercomputer and since it has a higher clock speed (2.1GHz) compared to the KNL nodes

$N_T = 32,768$, $N_{trace} = 29.8M$, p=6, $\Delta t = 0.1$, averaged over 6 time steps				
#cores	time/Picard [s]	N_T/core	Trace dof/core	efficiency [%]
512	290	64	58.4K	-
1024	129.5	32	29.2K	112
2048	82.6	16	14.6K	87.7
4096	46.6	8	7.3K	77.7
8192	39.4	4	3.7K	46

Table 6.2: 3D island coalescence problem. Strong scaling study for BFBT+AMG with GMRES smoother for solution order $p = 6$. The simulation is carried out in Skylake nodes of Stampede2 supercomputer.

(1.4GHz) the average time per Picard step is $4 - 5$ times lesser than that for KNL nodes. Since the BFBT+AMG with GMRES smoother takes less time compared to the ILU(0) smoother with overlap one and also less memory which is very important for high orders we only consider this preconditioner in the subsequent numerical studies. Also, since the strong scaling performance is not very much problem dependent we only study the performance in terms of weak scaling.

We now use the BFBT+AMG with GMRES smoother to simulate a challenging problem which is 2D island coalescence at high Lundquist numbers. Here as an example we consider a Lundquist number of Rm $= 10^7$. The significance of this problem is that at high Lundquist numbers the thin current sheet which forms at the center of the domain during reconnection breaks and gives rise to small structures called plasmoids. The dynamics of this problem is very different from the low Lundquist number cases where the islands monotonically approach each other and coalesce to a single island.

The settings of this experiment are as follows: While for the scaling studies we used a fixed time stepsize of $\Delta t = 0.1$, in this case we use an adaptive time step where we initially start with a stepsize of 0.05 and if the Picard iteration fails to converge in

20 iterations we cut the stepsize by half and try again. This is essential for capturing the highly nonlinear evolution of this problem and we observed a stepsize of 0.0015625 during the plasmoid formation and evolution stages. Once the time stepsize is cut by half we do not increase it later in the simulation, while increasing the stepsize may help in reducing the overall time of the simulation our focus here is to simulate the correct physics and also test the robustness of our preconditioner. The domain is discretized by 128×128 uniform elements and solution order $p = 9$. For time stepping we use the five stage fourth order DIRK scheme. The problem is run on 512 cores in the Skylake nodes of Stampede2 supercomputer.

Figure 6.5 shows the current and pressure corresponding to the initial location of islands after one time step at $t = 0.05$ and in Figure 6.6 we can see the formation of x-structure at the center of the domain at $t = 5.2$. We zoom in to the box marked at the center of the domain in Figure 6.6 to see clearly the formation and evolution of plasmoids. Figures 6.7 and 6.8 shows the current and pressure corresponding to the box region marked in Figure 6.6 at different times. From these two figures we can clearly see bubble like structures which are called plasmoids emanating from the breakdown of the thin current sheet and moving in the vertical direction. At least from Figures 6.7 and 6.8 we can see two prominent plasmoids one moving downwards and the other upwards. However, there are lot more tiny plasmoids which continuously appear and merge as time proceeds. This problem clearly shows the multiscale nature of the magnetic reconnection phenomenon in both space and time. The island widths are of $\mathcal{O}(0.1)$, whereas the size of plasmoids are of $\mathcal{O}(0.01)$. Similarly the time scale in which the island moves is almost an order of magnitude larger than the plasmoid time scales. The linear solver iterations for this problem is mostly less than 10 for most of the Picard steps with some of them taking between $15 - 20$. Thus the BFBT+AMG preconditioner with GMRES smoother seems to be fairly robust for this challenging

problem which involves multiscale physics in space and time.

6.4.2 Hydromagnetic Kelvin-Helmholtz instability

In this section we consider the 2D and 3D versions of hydromagnetic Kelvin-Helmholtz (HMKH) instability problem studied in [139, 119, 43]. The domain we consider is $[0,4]\times[-2,2]$ in 2D and $[0,4]\times[-2,2]\times[0,2]$ in 3D. The initial conditions consists of two counter flowing conducting fluids of constant velocities given by $\mathbf{u}^0(x, y\geq 0, z) = (1,0,0)$ and $\mathbf{u}^0(x, y<0, z) = (-1,0,0)$ and a Harris sheet magnetic field defined by $\boldsymbol{b}^0(x,y,z) = (B_0\tanh(y/\delta), 0, 0)$. We choose a zero forcing for both fluid and magnetic equations. Similar to the island coalescence problem, the boundary conditions are periodic in x- and z-directions. On the top and bottom faces the fluid boundary conditions are same as the island coalescence problem with zero normal velocity and zero shear stress and the magnetic field is defined by the Harris sheet in the initial condition. The Lagrange multiplier r is set as zero on all the boundaries. We select the following parameters as per [139]: $\kappa = 1$, $\mathrm{Re} = \mathrm{Rm} = 10^4$, $B_0 = 0.3333$ and $\delta = 0.1$. These values along with $\rho = \mu_0 = 1$ gives a super Alfvénic Mach number of $M_A = u/u_A = 3$ as described in [139]. If $M_A > 1$ then the magnetic field is not strong enough to suppress the instabilities and the shear layer is Kelvin-Helmholtz unstable. Thus the initial disturbances eventually grow to form vortices which roll up and merge as time proceeds.

First, we consider the 2D HMKH problem discretized in a 128×128 mesh with solution order $p = 6$. We use a time stepsize of $\Delta t = 0.001$ in the five stage fourth order DIRK method. As mentioned in the beginning of the numerical section we needed a stricter tolerance of 10^{-9} in the linear solver to make the Picard iterations converge for this problem up to a relative tolerance of 10^{-4}. The preconditioner is BFBT+AMG with GMRES smoother and the outer iterations are carried out by

FGMRES. In Figure 6.9, the evolution of vorticity with time is shown along with the magnetic vectors marked by arrows. The figure shows the roll up of vortices to form the familiar cat-eye pattern and the magnetic vectors bends and follows the fluid evolution as time proceeds. In Figure 6.10, we show the 3D HMKH problem discretized in a $20 \times 24 \times 7$ mesh clustered around the region of solution and solution order $p = 5$. An initial time stepsize of $\Delta t = 0.025$ is selected for the backward Euler time stepping and the adaptive time stepping procedure described in the previous section is employed. Here also as in 2D we see the rollup of vortices and the magnetic vectors following them.

Next we study the weak scaling performance of the BFBT+AMG with GMRES smoother for the 3D HMKH problem. For this we used a fixed time stepsize of $\Delta t = 0.01$ in the backward Euler time stepping and the results are averaged over six time steps. We used $8 \times 8 \times 10$, $16 \times 16 \times 10$, $32 \times 32 \times 10$ and $64 \times 64 \times 10$ meshes corresponding to 32, 128, 512 and 2048 processors respectively. Thus the number of elements per processor in this case is 20. The fluid CFL ranges from 0.32 to 2.56 for solution order $p = 4$ and for $p = 5$ from 1.25 to 4 corresponding to the meshsizes and time stepsize. The Alfvénic CFL which is given by $u_A \Delta t/h$ is one third of the fluid CFL in this case. The Picard solver took approximately 2 iterations in all the cases. Figure 6.11 shows the average iteration counts and time per Picard step, since the tolerance of the iterative solver in this case (10^{-9}) is stricter than for the island coalescence problem (10^{-6}) we see an increase in overall iteration counts. However, the number of iterations still lies mostly between $10 - 20$ which is moderate considering into account the tolerance of 10^{-9} and the maximum CFL numbers of 2.56 and 4 for $p = 4$ and 5 respectively. The average time per Picard step reflects the trend in the iteration count together with the decrease in scalability in the 2048 processors regime as discussed in the previous section on island coalescence.

6.4.3 Lid driven cavity

In this section we consider a hydromagnetic version of the classical lid driven cavity problem. The settings of this problem follow closely [119]. Even though we simulated the 2D version of this problem also here we present the results only for the 3D problem for brevity. The domain is $[-0.5, 0.5]^3$, with no slip boundary conditions of $\mathbf{u} = \mathbf{0}$ applied on all the walls except the top one where we apply a velocity of $\mathbf{u} = (1, 0, 0)$ which drives the flow. For the magnetic field we set the tangential component on each wall as $\boldsymbol{b} \times \mathbf{n} = (-1, 0, 0) \times \mathbf{n}$ which acts from right to left. The Lagrange multiplier r is set as zero on all the boundaries. Both initial conditions and forcings are chosen as zero. We choose the following parameters: $\kappa = 1$, $\mathrm{Re} = \mathrm{Rm} = 1000$ which corresponds to a Hartmann number of $Ha = \sqrt{\kappa \mathrm{ReRm}} = 1000$ which measures the degree of coupling between the electromagnetics and hydrodynamics.

First, we consider a 8^3 uniform mesh, solution order $p = 6$ and an initial time stepsize of $\Delta t = 0.0125$ in the adaptive time stepping with backward Euler method. The linear and nonlinear solver tolerances and the stopping criteria are same as the ones used for the island coalescence problem. Figure 6.12 shows the evolution of the streamlines together with the velocity vectors in the $x = 0$, $y = 0$ and $z = 0$ planes. Both the streamlines and the velocity vectors are colored by the $z-$component of velocity. The presence of the third dimension allows the streamlines to curl in the $z-$direction in Figures 6.12(d), 6.12(e) and 6.12(f) which gives rise to complex velocity patterns. In Figure 6.13 the corresponding magnetic field lines along with the magnetic vectors are shown. Because of the applied tangential magnetic field from right to left we can see the magnetic lines and the vectors going from right to left with some bending caused by the interaction with the fluid components.

For the weak scaling study we consider a fixed time stepsize of $\Delta t = 0.0025$

for solution orders $p = 4, 5$ whereas for $p = 6$ we consider $\Delta t = 0.001$ in the backward Euler method. The meshes we consider are 4^3, 8^3, 16^3 and 32^3 corresponding to 4, 32, 256 and 2048 processors respectively with 16 elements per core. The fluid CFL which is same as the Alfvénic CFL in this case ranges from $0.16 - 1.28$ for $p = 4$, $0.25 - 2$ for $p = 5$ and $0.14 - 1.15$ for $p = 6$. The Picard iterations in all the cases range from $3.5 - 4.7$. We also tested with $\Delta t = 0.01$ for $p = 4$ and observed that the iterations of the linear solver remains more or less same as that for $\Delta t = 0.0025$ whereas the average Picard iterations are $5.3, 6.5, 7.5$ and 9.3 corresponding to the four meshsizes. In Figure 6.14 we show the average iterations and time per Picard step for the BFBT+AMG preconditioner with GMRES smoother and the results are averaged over six time steps. Compared to the previous two experiments, in this case we have a fairly constant iteration count with mesh refinements for all the solution orders. In Figure 6.14(b) we again see some increase in time per Picard step for 2048 cores which in this case comes only from the coarsening in AMG and other components of the block preconditioner as we have a flat iteration count. We consider one more case for solution order $p = 4$ with 2, 16, 128 and 1024 cores in Figure 6.14(b), and this shows better weak scaling performance than the other case with 2048 cores for the finest meshsize.

6.4.4 BFBT+Multilevel preconditioner

In this section we will apply the multilevel solver introduced in chapter 5 for $\tilde{\mathcal{F}}^{-1}$ in the block preconditioning (6.27) and compare the performance with AMG v-cycle with GMRES smoother. The problem we consider is the 2D island coalescence problem studied in section 6.4.1.

In chapter 5 we applied the multilevel preconditioner for problems with scalar trace unknowns and now we will briefly describe how we can apply it to vector valued

trace unknowns. Similar to AMG, we order the unknowns such that on each edge all the trace unknowns (except edge average pressure) corresponding to the first node are ordered first, followed by the unknowns in the second node and so on. The ordering of unknowns within each nodal point is $(\widehat{U}, \widehat{B}^t, \widehat{R})$ i.e., velocity trace unknowns are ordered first, followed by the tangent magnetic field and then the Lagrange multiplier. A different ordering is also possible and it can affect the performance of the solver. Also, instead of ordering the unknowns within a node consecutively we can choose to order, for example, all the velocity unknowns (corresponding to all edges) first, followed by the tangent magnetic field and then the Lagrange multiplier. In fact this is the approach we followed in our geometric multigrid approach presented in chapter 4 for vector valued problems. Ordering the unknowns within each node consecutively helps to maintain the coupling between different variables during coarsening and smoothing [139]. In our future study we will compare the performance of the preconditioner/solver for different orderings and report our findings elsewhere.

We then apply one v-cycle of the iterative multilevel algorithm, Algorithm 4, for $\tilde{\mathcal{F}}^{-1}$. For the coarse-solver we use the enriched multilevel approach (EML) because of its robustness and better performance compared to the non-enriched version for hyperbolic problems. By means of several numerical experiments we observed that the number of smoothing steps in the block-Jacobi part of the multilevel algorithm, $m_1 = 0$ and $m_2 = 1$ i.e., only one postsmoothing, gives the least number of outer GMRES iterations and we use that in all the cases. We also observed that increasing the number of smoothing steps generally leads to more number of iterations in this case and postsmoothing performs better than presmoothing. Since the nodal block ($\mathcal{F}$) in the HDG discretization of MHD is a non-symmetric hyperbolic system and for the most part it is purely algebraic we cannot in general expect better performance by increasing the number of smoothing steps. The number of iterations also depends on

how the fine scale solver (block-Jacobi) interacts with the coarse-scale solver (EML) by means of capturing the overall spectrum and we do not have a clear understanding of this yet. In our future work we will investigate this by means of Fourier analysis and this can guide us to select the appropriate number of smoothing steps as well as the fine scale solver.

For the AMG v-cycle we use the same number of pre- and postsmoothing steps which is three as in the previous sections. Unlike the multilevel approach, increasing the number of smoothing steps in AMG generally leads to less number of iterations and the reason for this is, in AMG directional coarsening happens based on the matrix entries whereas in the multilevel approach the coarsening is geometric. For the time discretization we use the backward Euler time stepping with a fixed time stepsize of $\Delta t = 0.1$. The results are averaged over six time steps for all the cases except 128×128 and 256×256 meshes where we average over three time steps. We choose a Lundquist number of $\mathrm{Rm} = 10^3$. We run the AMG solver serially in this case to compare with the serial implementation of the multilevel method. So this study is mainly to assess the algorithmic scalability of EML and AMG solvers with respect to mesh refinements, solution order and Lundquist number.

Figure 6.15(a) shows the average number of iterations for both the AMG and EML solvers together with the BFBT approximation for the inverse Schur complement in the block preconditioning (6.27) as the mesh and solution order are refined. In calculating this average we have omitted the iteration counts for first Picard step in the first time step. This is because the iteration counts for both EML and AMG solvers are higher for this case than the rest of the steps as we start from a zero initial guess. Hence it is not a representative of the iteration counts taken in other steps. Both the algorithms show almost flat iteration count until 64×64 mesh with AMG taking slightly less iteration counts than EML. However, for 128×128 mesh (last marker)

and all solution orders AMG shows a sudden increase in iteration count whereas the performance of EML almost remains the same maintaining the algorithmic scalability. Thus the EML solver is more scalable than AMG in terms of mesh refinements at least in the settings of this experiment. Figure 6.15(b) shows the ratio of average time taken per Picard step for the AMG solver over EML solver and it reflects the trend observed in the iteration counts in Figure 6.15(a).

Solver	$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$	$p = 6$
BFBT+EML	1.17	1.15	1.17	1.13	1.12	1.13
ND	1.86	1.56	1.49	1.44	1.42	1.37

Table 6.3: 2D island coalescence problem. Exponent of scaling for BFBT+EML preconditioned GMRES and ND solver as the number of elements is increased.

In Figure 6.16 we compare the scaling of the average time taken per Picard step with mesh refinements for the BFBT+EML preconditioned GMRES and the nested dissection direct solver for different solution orders. As per the theoretical complexities derived in section 5.1.9, the EML solver at all orders show close to linear scaling with the number of elements whereas the ND solver shows an asymptotic scaling of $\mathcal{O}({N_T}^{3/2})$. In Table 6.3 we show the asymptotic exponent values for both BFBT+EML and ND solvers at different orders. In the BFBT+EML preconditioned GMRES the BB^T part is solved directly, the cost of which is higher than $\mathcal{O}(N_T)$ and it may be responsible for slightly more than linear complexity.

In Figure 6.17 we show the speedup of BFBT+EML preconditioned GMRES compared to ND and we can see at all solution orders, when the number of elements is greater than 10^3 (after 5 uniform refinements) the iterative solver is faster than ND. We get a maximum speedup of approximately 23 for $p = 1$ and 256×256 elements, and the ND solver ran into out of memory issues after this. In terms of memory for $p = 1$ and 256×256 mesh, the ND solver needed 98.2 GB for the L, U factors whereas

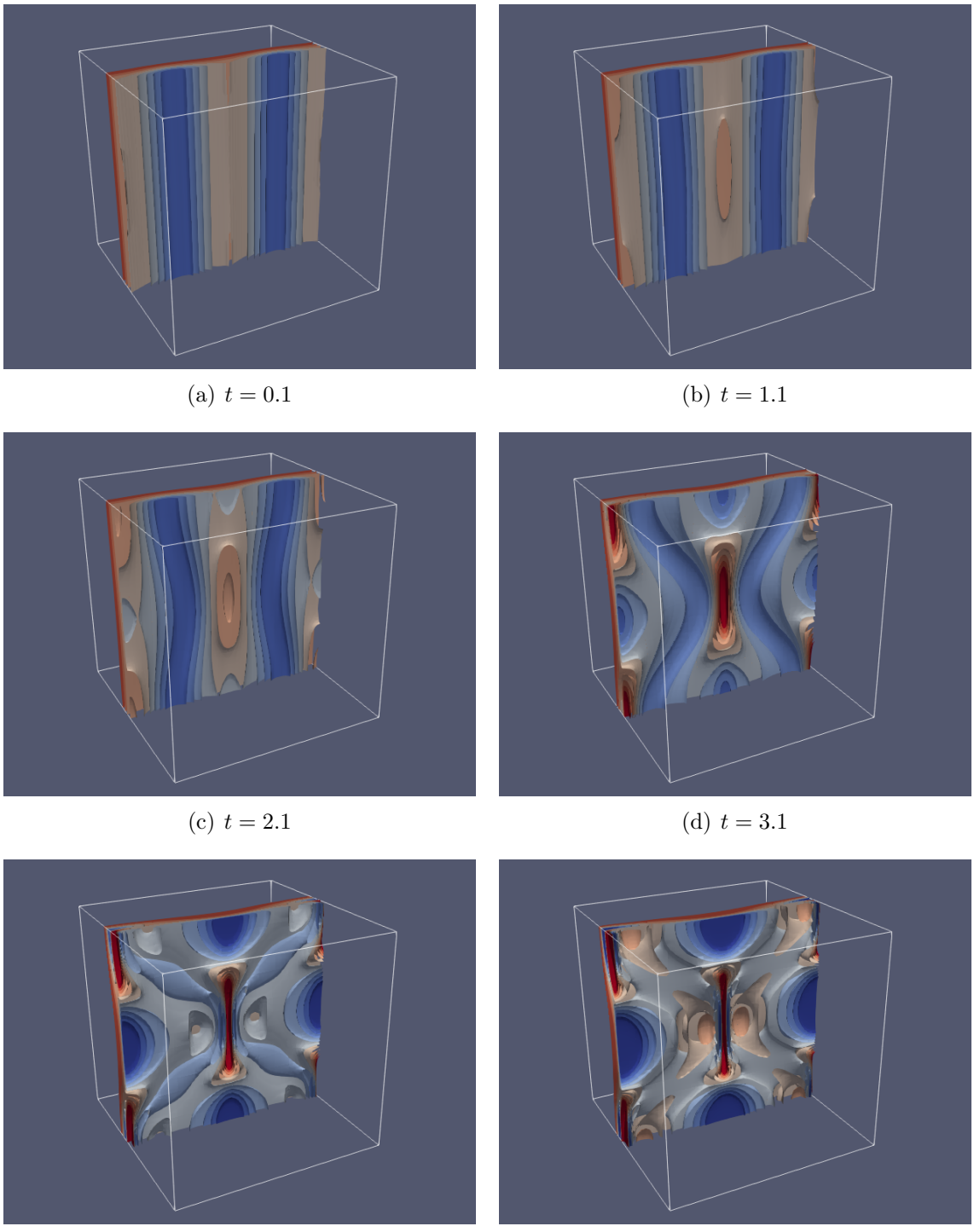
the EML solver needed only 2.4 GB which is 41 times less memory compared to the direct solver. This factor is significantly higher than the one observed for the scalar problems studied in chapter 5. Thus EML together with the block preconditioning can deliver significant speedups and memory savings compared to the ND direct solver for vector valued problems even in 2D.

Finally we test the robustness of AMG and EML preconditioners with respect to the Lundquist number for this problem. To that extent we consider 64×64, $p = 6$ mesh and choose a time stepsize of $\Delta t = 0.05$. The results are again averaged over six time steps. Figure 6.18 shows the average iteration counts and time per Picard step for Lundquist numbers in the range $[10^3, 10^6]$. We can see that the BFBT+AMG with GMRES smoother is more robust with respect to increase in Lundquist numbers than the BFBT+EML preconditioner. Nevertheless, the growth in iterations for the BFBT+EML preconditioner is still moderate and in all the cases both the preconditioners take less time than the ND direct solver. In our future work, we want to improve the robustness of the EML preconditioner by exploring more robust smoothers than block-Jacobi.

6.5 Discussion

In this chapter we introduce a block preconditioning strategy for trace systems coming from HDG discretization of the incompressible resistive MHD equations. In the block preconditioner, we use least squares commutator (BFBT) approximation for the inverse of the Schur complement and AMG v-cycle for the approximate inverse of the nodal block. For the smoother inside AMG cycle, we compare preconditioned GMRES and ILU(0) smoother of overlap one and conclude that the GMRES smoother is faster and requires less memory compared to the ILU smoother. The

outer iterations are carried out with flexible GMRES, as the inner iterations are nonlinear. We test the performance of the block preconditioner on several 2D and 3D transient test cases including, but not limited to, the island coalescence problem at high Lundquist numbers and demonstrate robustness and parallel scalability. We also show the application of multilevel preconditioner developed in chapter 5 for the approximate inverse of the nodal block and compare the performance with AMG. The multilevel preconditioner shows better algorithmic scalability compared to AMG with respect to mesh refinements. In terms of robustness with respect to Lundquist numbers AMG performs better and strong smoothers are needed in the multilevel preconditioner. The block preconditioning combined with the multilevel preconditioner is significantly faster than the nested dissection direct solver and also requires much less memory.

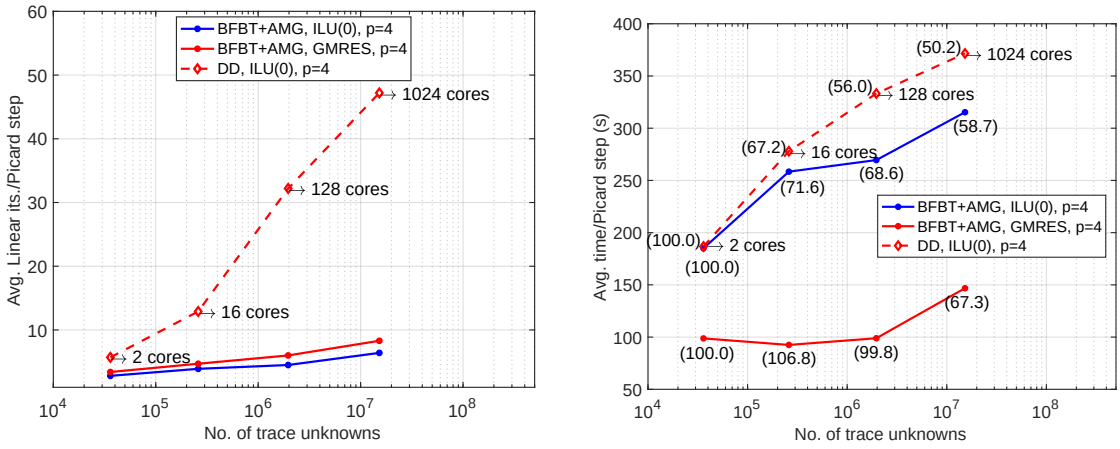


(a) $t = 0.1$ (b) $t = 1.1$

(c) $t = 2.1$ (d) $t = 3.1$

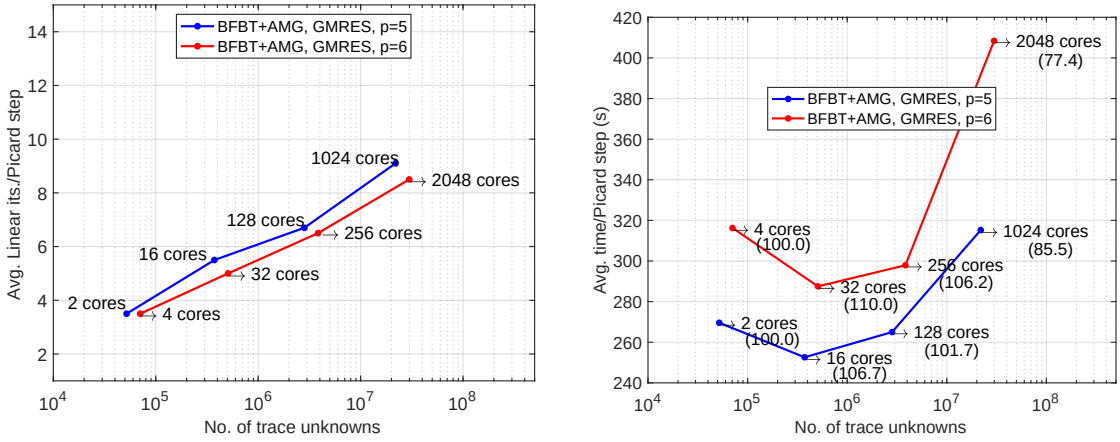
(e) $t = 3.625$ (f) $t = 3.875$

Figure 6.2: Evolution of the z-component of the current (J_z) with time. The contours of J_z show a highly kinked state after $t > 3$ due to the perturbation in the $z-$direction.



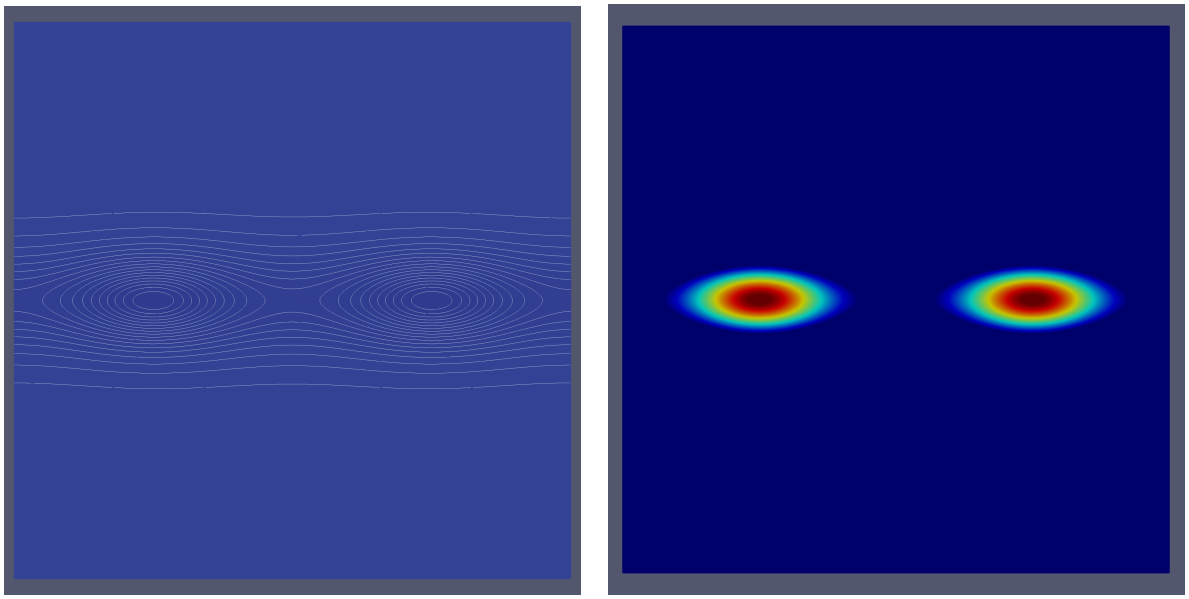
(a) (b)

Figure 6.3: 3D island coalescence problem: weak scaling study of average iterations per Picard step (left) and average time per Picard step (right) for three preconditioners and solution order $p = 4$. The markers in BFBT+AMG with ILU(0) and GMRES also represent the same number of processors as in DD with ILU(0). The values within parentheses represent weak scaling parallel efficiencies.



(a) (b)

Figure 6.4: 3D island coalescence problem. BFBT+AMG with GMRES smoother: weak scaling study of average iterations per Picard step (left) and average time per Picard step (right) for solution orders $p = 5, 6$. The values within parentheses represent weak scaling parallel efficiencies.



(a) J (b) q

Figure 6.5: 2D island coalescence problem. Current (J) and pressure (q) at time $t = 0.05$ (after one time step) showing the two islands.

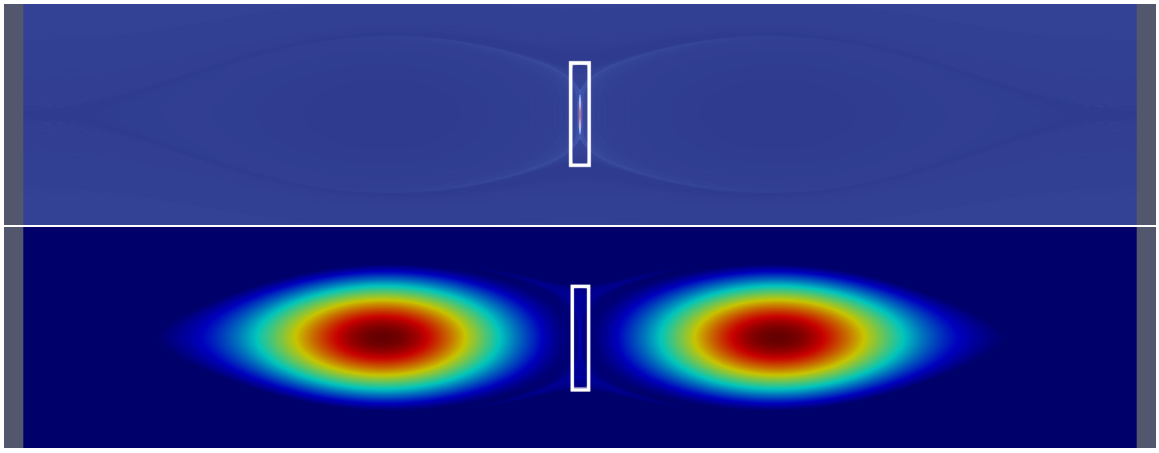
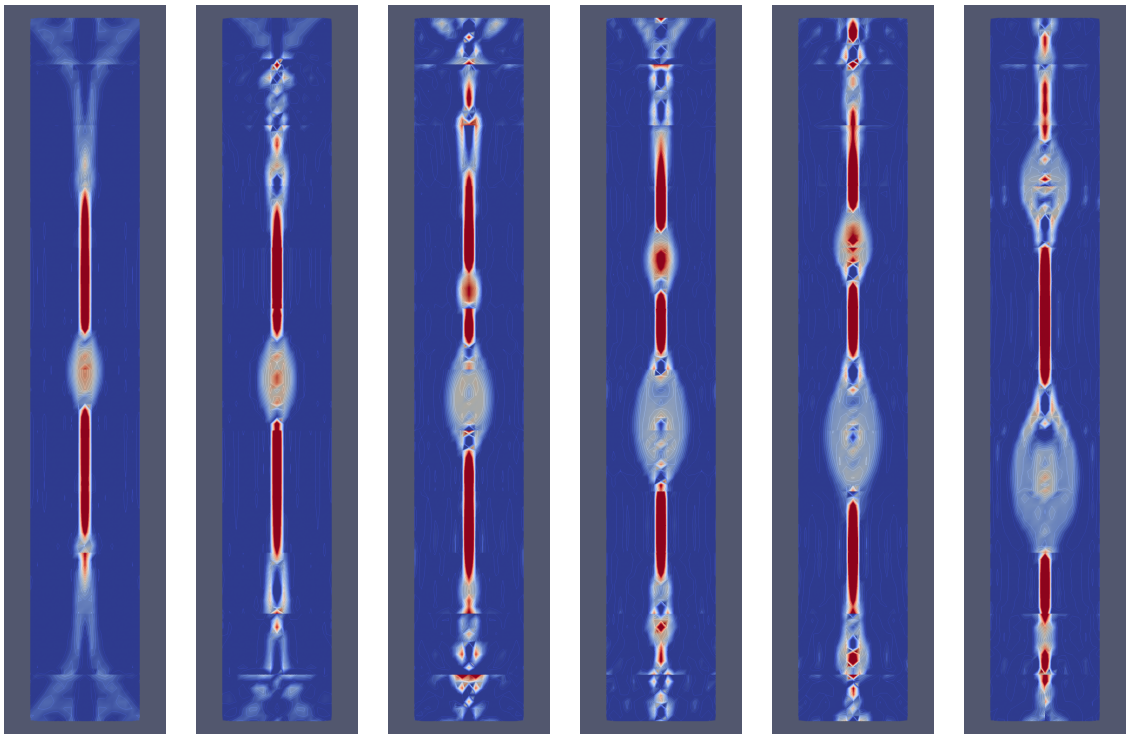
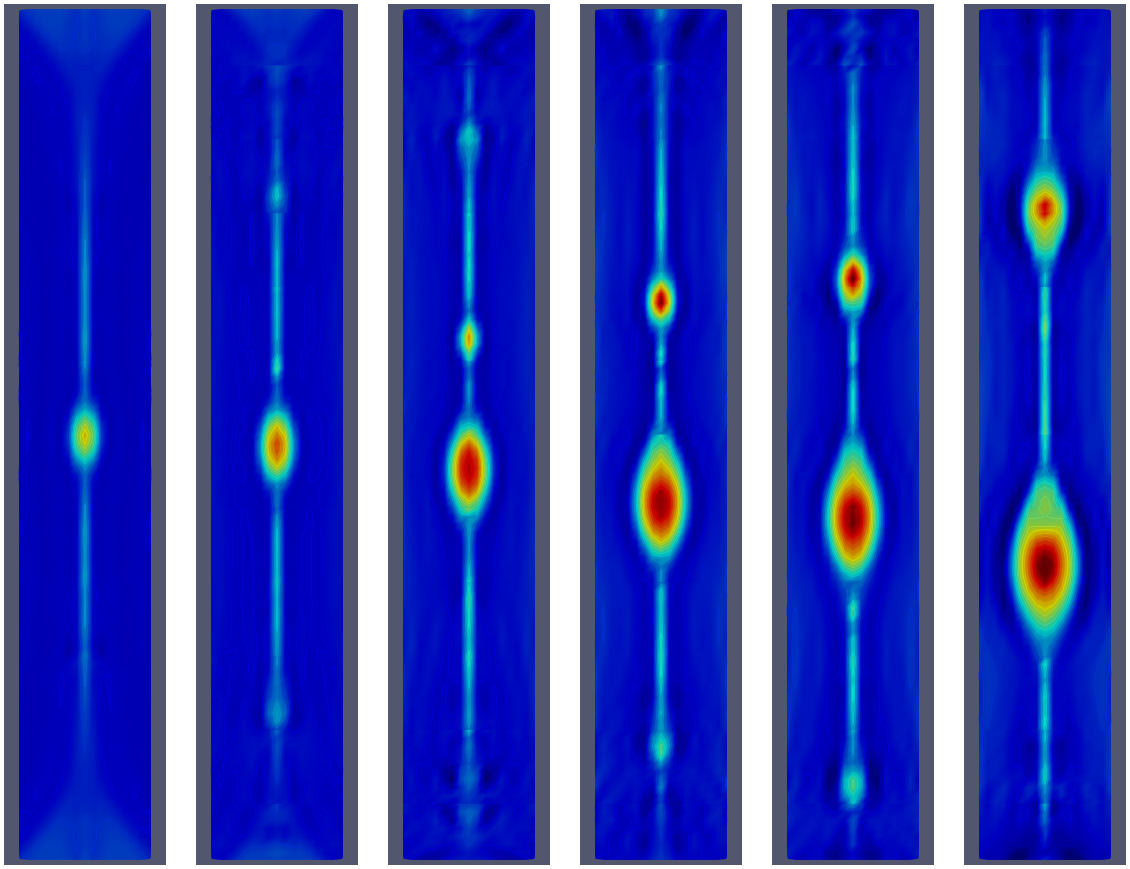


Figure 6.6: 2D island coalescence problem. Current (top) and pressure (bottom) at time $t = 5.2$ showing the coalescence of the islands. In Figures 6.7 and 6.8 we will focus on the box region marked in the central portion of the above figures.



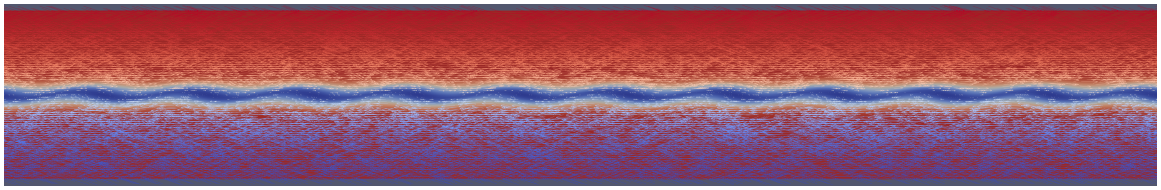
(a) $t = 5.85$ (b) $t = 5.98$ (c) $t = 6.12$ (d) $t = 6.22$ (e) $t = 6.25$ (f) $t = 6.3$

Figure 6.7: 2D island coalescence problem. Current plots at the indicated times showing the breakdown of the current sheet to form plasmoids.

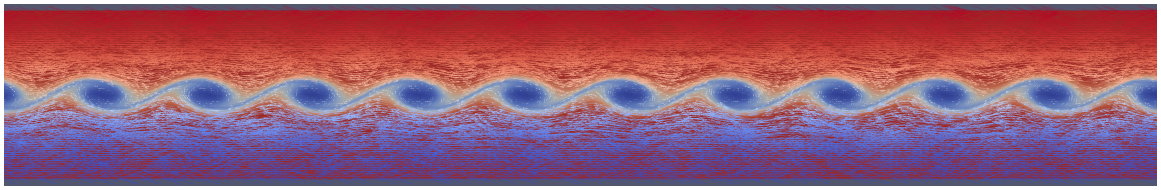


(a) $t = 5.85$ (b) $t = 5.98$ (c) $t = 6.12$ (d) $t = 6.22$ (e) $t = 6.25$ (f) $t = 6.3$

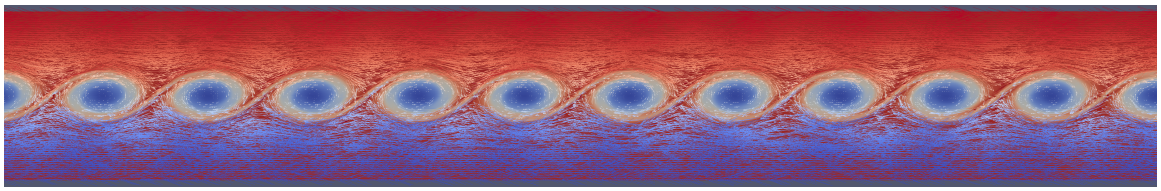
Figure 6.8: 2D island coalescence problem. Pressure plots at the indicated times showing the formation and evolution of plasmoids with time.



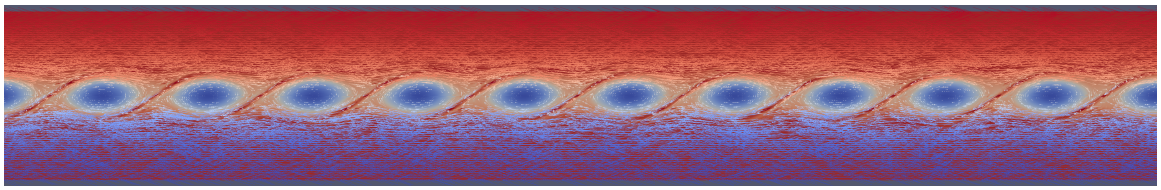
(a) $t = 1.51$



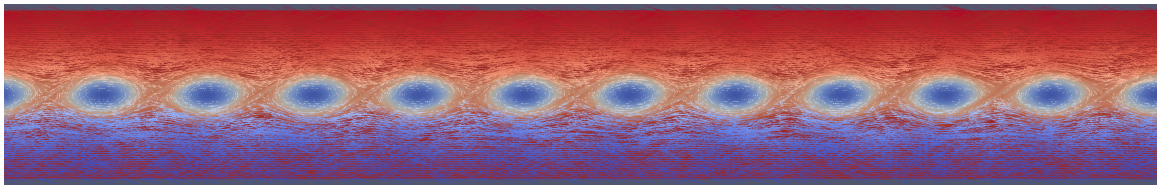
(b) $t = 1.73$



(c) $t = 1.96$



(d) $t = 2.18$



(e) $t = 2.61$

Figure 6.9: 2D HMKH problem. Vorticity plots at the indicated times along with the magnetic vectors (marked as arrows). The magnetic vectors are scaled by their magnitude and are colored by the x-component of the magnetic field (b_x). The red arrows on the top represent the positive values of b_x and blue arrows on the bottom represent the negative values.

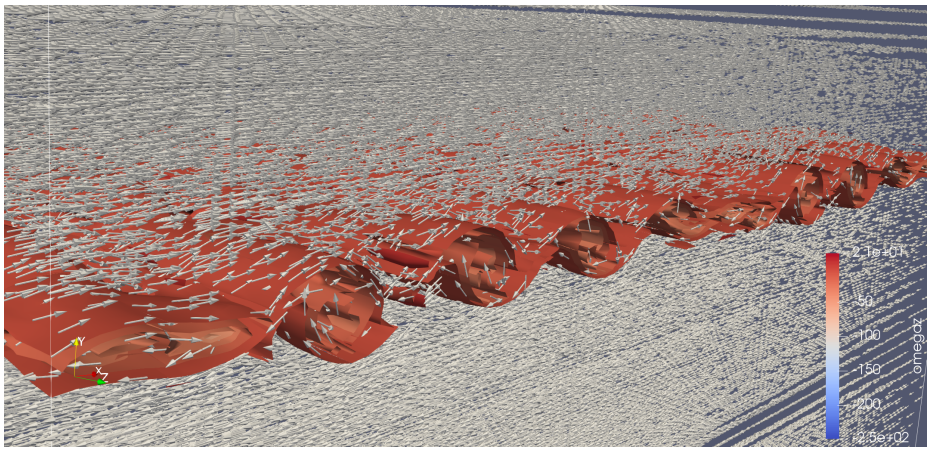
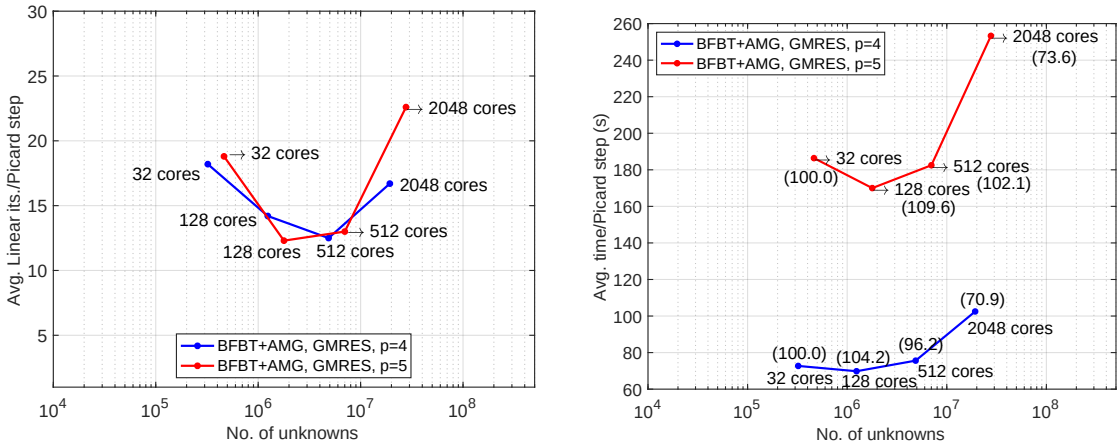
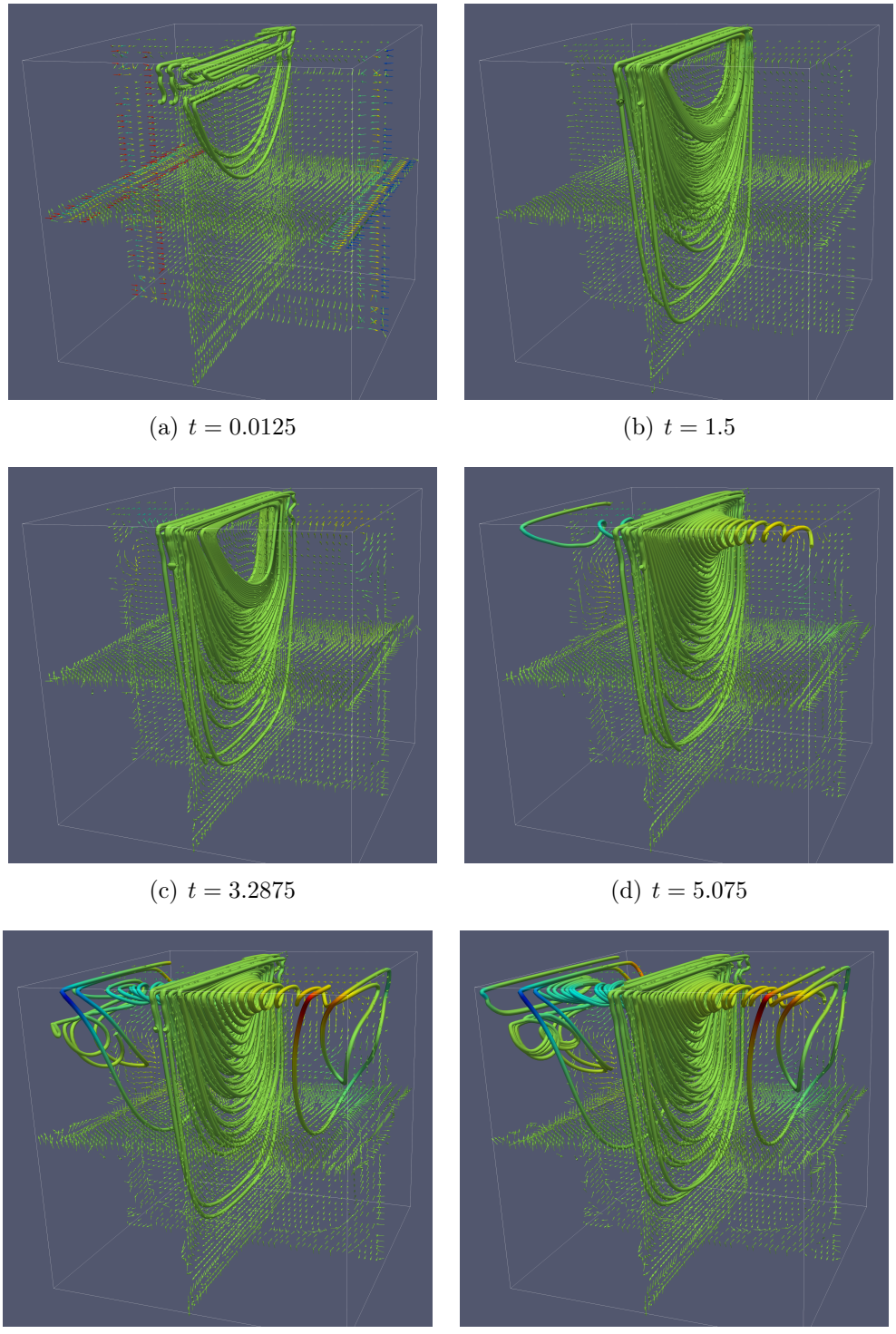


Figure 6.10: 3D HMKH problem. $Z-$component of vorticity contours along with magnetic vectors at time $t = 2.0625$. The magnetic vectors are not colored and scaled in this figure to improve visibility.



(a) (b)

Figure 6.11: 3D HMKH problem. BFBT+AMG with GMRES smoother: weak scaling study of average iterations per Picard step (left) and average time per Picard step (right) for solution orders $p = 4, 5$. The values within parentheses represent weak scaling parallel efficiencies.

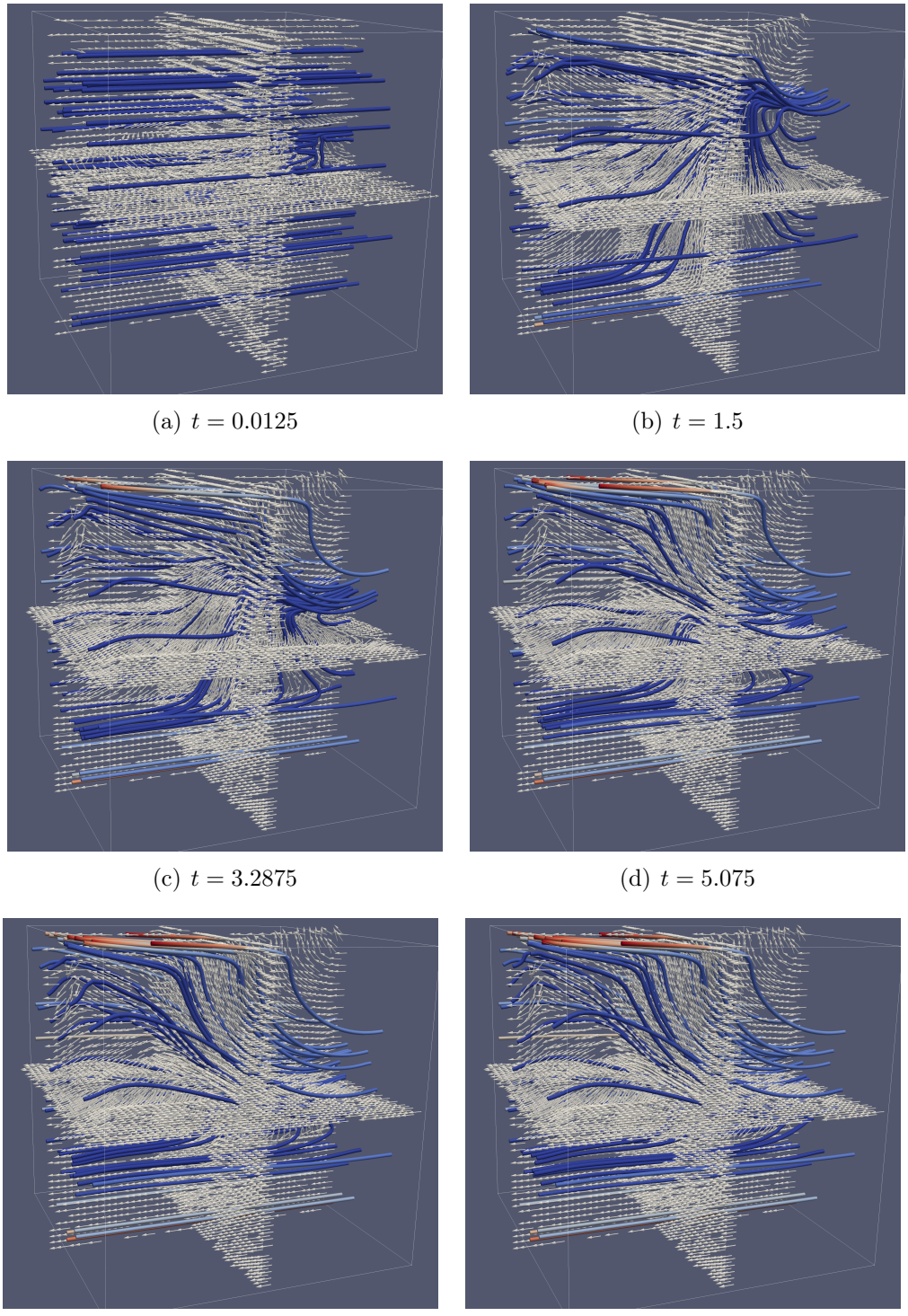


(a) $t = 0.0125$ (b) $t = 1.5$

(c) $t = 3.2875$ (d) $t = 5.075$

(e) $t = 5.3625$ (f) $t = 5.6625$

Figure 6.12: 3D lid driven cavity problem. Evolution of the streamlines and velocity vectors with time.

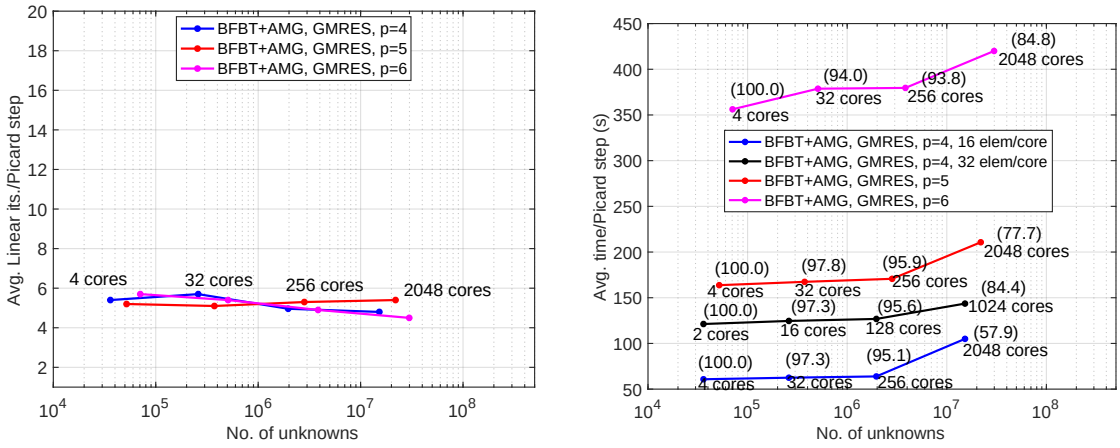


(a) $t = 0.0125$ (b) $t = 1.5$

(c) $t = 3.2875$ (d) $t = 5.075$

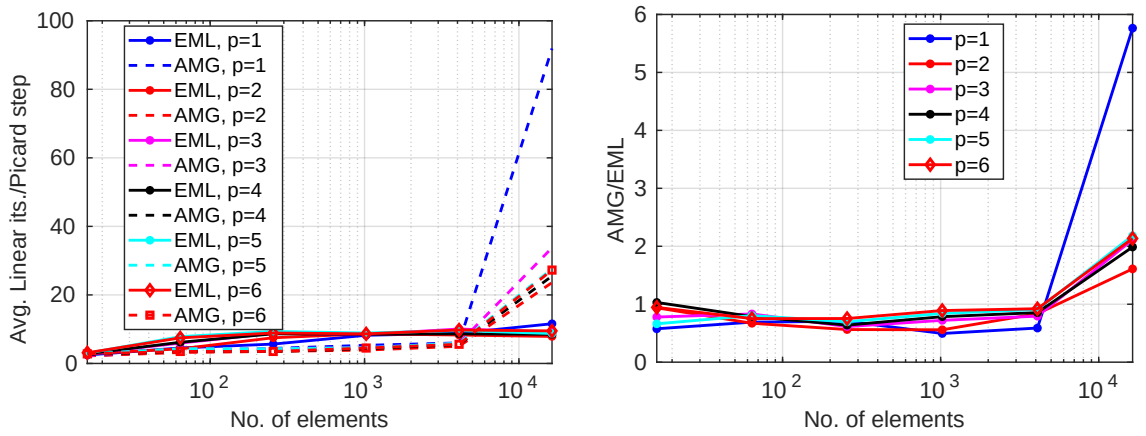
(e) $t = 5.3625$ (f) $t = 5.6625$

Figure 6.13: 3D lid driven cavity problem. Evolution of the magnetic field lines and magnetic vectors with time.



(a) (b)

Figure 6.14: 3D lid driven cavity problem. BFBT+AMG with GMRES smoother: weak scaling study of average iterations per Picard step (left) and average time per Picard step (right) for solution orders $p = 4, 5, 6$. The values within parentheses represent weak scaling parallel efficiencies.



(a) (b)

Figure 6.15: 2D island coalescence problem. Comparison of AMG and multilevel preconditioners as the mesh is refined in h and p. In both cases we use the BFBT approximation for the inverse of the Schur complement. On the left is the number of average iterations per Picard step and on the right is the ratio of average time taken per Picard step for AMG over the multilevel preconditioner.

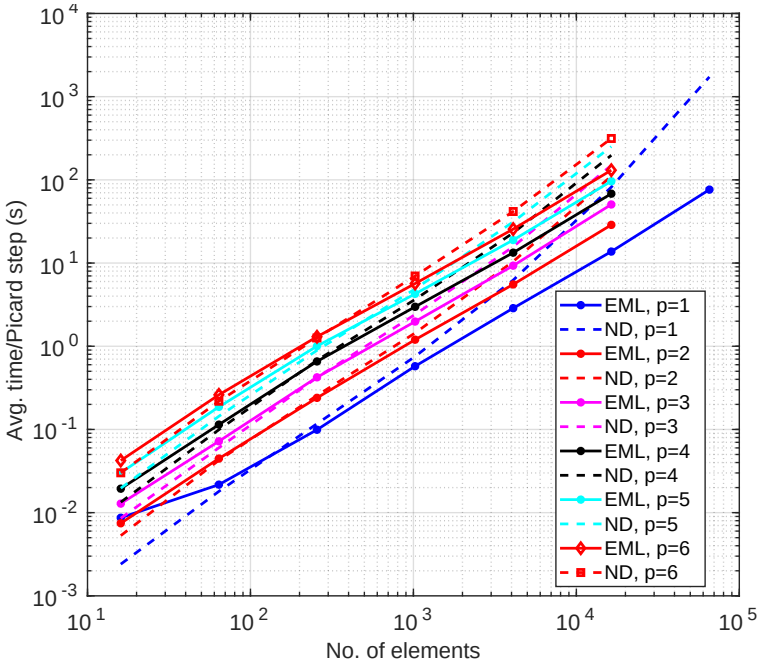
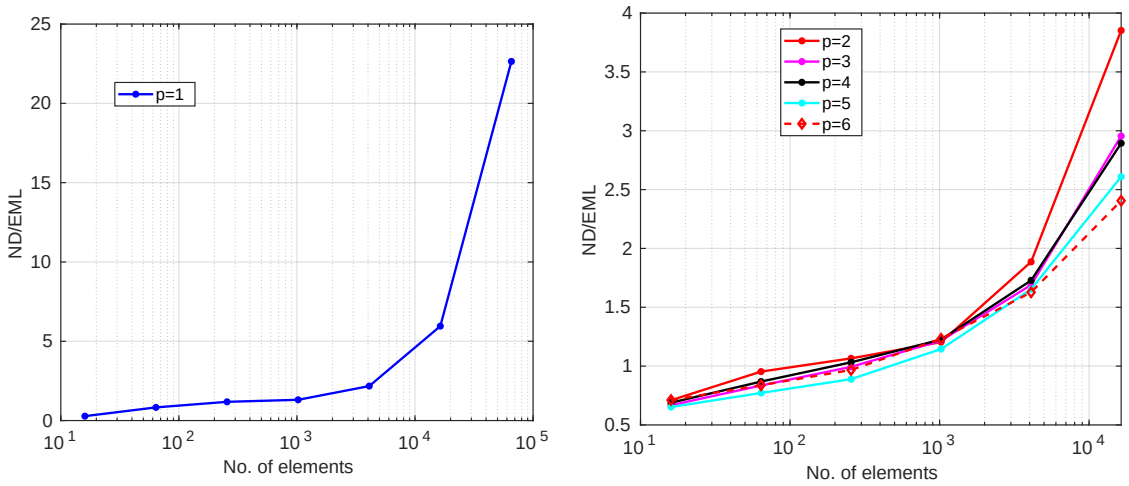
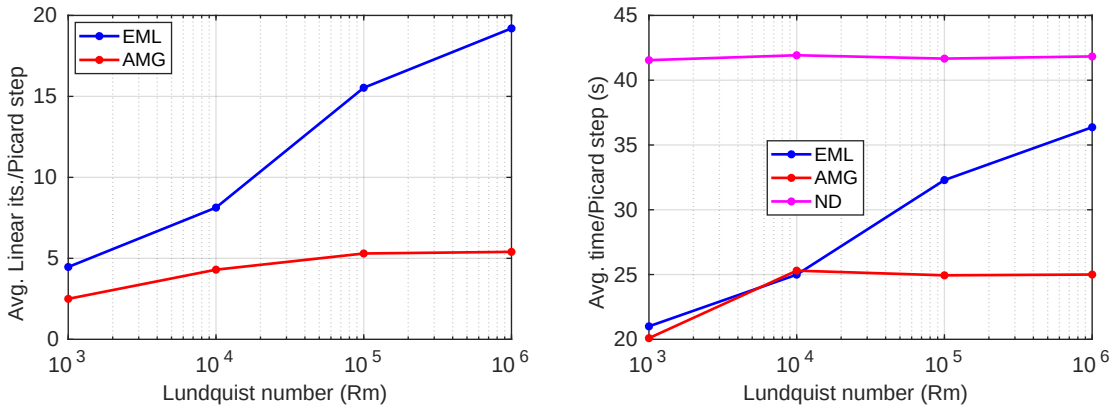


Figure 6.16: 2D island coalescence problem. Comparison of scaling of BFBT+EML preconditioned GMRES (denoted as EML) and ND with number of elements for solution orders $p = 1 - 6$. The asymptotic numerical values for the exponents are shown in Table 6.3.



(a) (b)

Figure 6.17: 2D island coalescence problem. Ratio of time taken per Picard step for ND solver over BFBT+EML preconditioned GMRES for solution order $p = 1$ (left) and for orders $p = 2 - 6$ (right). The ND solver ran into out of memory issues for orders $p > 1$ and 256^2 elements.



(a) (b)

Figure 6.18: 2D island coalescence problem. Comparison of AMG and multilevel preconditioners with increase in Lundquist number Rm. In both cases we use the BFBT approximation for the inverse of the Schur complement. On the left is the average iterations per Picard step and on the right is the average time taken per Picard step.

Chapter 7

Conclusions and Future Work

In this work we have developed fast, scalable solvers/preconditioners for high-order HDG schemes applied to a variety of problems in fluid dynamics and incompressible resistive MHD. First of the solvers developed (iHDG) is based on domain decomposition methods and involves in each iteration element-by-element and face-by-face local solves by alternating the computations between local solver and conservation condition. We analyze the solver for scalar and system of hyperbolic equations and verify the analysis by means of 2D and 3D numerical results. We also propose an improved version of the solver for diffusion dominated equations and system of hyperbolic equations. Since one level domain decomposition methods are known to lack algorithmic scalability due to lack of coarse solvers, iHDG can be used as smoothers in multilevel/multigrid methods or as preconditioners for Krylov subspace methods for large scale problems.

To have $hp-$scalability typically multigrid/multilevel solvers are needed and we propose a geometric multigrid method for HDG trace systems based on DtN maps. This multigrid method which comes under the class of Schur complement multigrid methods is different from classical geometric multigrid methods for volume based dicretizations such as continuous Galerkin, stabilized FEM, DG, etc. The main difference is the operator dependent intergrid transfer operators which provide robustness to the algorithm and also help to avoid explicit upscaling of parameters. Thus the proposed algorithm enjoys robustness similar to AMG and also has fixed

coarse grid construction costs due to the geometric nature. In this sense it is similar to the blackbox type multigrid algorithms [47, 46, 48] introduced in the context of finite differences and finite volumes. The algorithm can be applied to both structured and unstructured meshes and also does not require the meshes to be nested. With various numerical examples we conclude that the algorithm gives almost $hp-$scalable results for Poisson and Stokes type problems even for cases with highly heterogeneous and discontinuous coefficients which occur in flow through porous media. For convection dominated problems and hyperbolic systems we need robust smoothers for the convergence as well as scalability of the algorithm.

We then introduce a multilevel algorithm by blending ideas from nested dissection which is a fill-in reducing direct solver strategy, high-order and variational characteristics of HDG, and domain decomposition. Since the coarse solver in the multilevel algorithm is derived from a direct solver strategy the algorithm can be applied to general hyperbolic system of PDEs where multigrid type algorithms have difficulty in convergence. Our numerical experiments show that the performance of the algorithm mostly depends on the smoothness of the solution and is otherwise independent of the nature of the PDEs. We show that the algorithm can also be interpreted as a multigrid with specific intergrid transfer and smoothing operators on each level. We derive complexity estimates for the algorithm and show that the complexity is in between direct solvers and multigrid solvers. With the multilevel approach we can vary the solution order and hence the memory requirement depending on the architecture of the machine and thus there is flexibility. With several numerical experiments we show the robustness and scalability of the algorithm for the Poisson, the convection-diffusion and the pure transport equations.

Finally, we introduce a block preconditioning strategy for the trace systems coming from HDG discretization of the incompressible resistive MHD equations. The

trace system leads to a saddle point structure after rearrangement, which is very different from the saddle point systems arising from mixed FEM discretizations of incompressible systems. The main difference is that the system is mostly algebraic as it is generated from the static condensation of volume unknowns and the structure is much more complicated than the linear systems coming from stabilized FEM or mixed FEM discretizations. We use a least squares commutator (BFBT) approximation for the inverse of the Schur complement, and one v-cycle of AMG for the approximate inversion of the nodal block of skeletal unknowns. These strategies are previously used in the context of incompressible Navier–Stokes equations and mixed FEM or finite difference discretizations and here we show the applicability to high-order trace system generated from HDG methods. One important point to notice is we need strong smoothers for the AMG v-cycle because of the multiphysics and mostly algebraic nature of the nodal block. We show that GMRES preconditioned by ILU(0) with overlap zero is a good candidate for this compared to ILU(0) with overlap one due to its large memory requirement at high orders. With several 2D and 3D transient examples we show the robustness and parallel scalability of the block preconditioner. We also apply the multilevel algorithm for the approximate inversion of the nodal block and since the coarse solvers are strong in the multilevel approach the algorithm converges with just block-Jacobi as fine scale solver. We show that the multilevel algorithm is more scalable than AMG with respect to mesh refinements but need more strong smoothers to improve the robustness in case of high Lundquist numbers.

There are several directions in which the current work can be extended and we suggest a few of them as follows.

- **Theoretical analysis:** In this work we have shown only the convergence analysis for the iHDG solvers. For the geometric multigrid and the multilevel solvers

we show only the stability of the intergrid transfer operators but not the convergence analysis. For elliptic PDEs, with geometric multigrid we can use the standard convergence analysis used for multigrid type algorithms and can show convergence. In the case of multilevel solvers since we are mainly interested in generic hyperbolic systems the analysis is much more difficult. One possible way is to use the Fourier analysis and study the spectrum of the preconditioned matrices. This can offer lot of insights into the algorithm and mainly the interaction of fine and coarse scale solvers. This will be very useful especially in the context of MHD systems, to choose the number of smoothing steps and also in the selection of fine scale solvers so that they complement the spectrum captured by the multilevel coarse solvers. Analysis for the block preconditioner is also an interesting direction to pursue. However in this case it is much more difficult due to the complicated nature of MHD systems combined with the static condensation performed in HDG.

- **Extension of multilevel solvers:** In this work we show the multilevel solvers for structured grids generated by uniform refinements. As such the applicability is limited and we need to extend the algorithm for dealing with complicated geometries. There are two ways to address this issue. One is through graph agglomeration techniques, which can be used to find a nested dissection partitioning of an unstructured mesh. However, we need to be careful, since the approach involves high-order projections and the agglomerations generated in general do not have much smoothness in the interfaces. In these cases the interface is smoothened and this can affect the performance of the preconditioner. Another strategy is to use adaptive mesh refinement (AMR), the data structures used in AMR can be combined with the multilevel solver effectively as we describe now. Consider a mesh which is obtained through k levels of adap-

tive refinement from an initial coarse mesh, since we know how many times an edge/face is $h-$refined or coarsened, in our multilevel approach we can use it to increase/decrease the polynomial order. We can also use the indicator used for the mesh refinement or coarsening in the AMR for increasing or decreasing the polynomial order on the edges/faces. Thus the multilevel solver together with the AMR approach can provide a strategy for tackling complicated geometries efficiently.

- **Extension of block preconditioners:** We already mentioned that some of the components in the block preconditioning such as sparse direct solver Superludist for $(BB^\top)^{-1}$ and AMG for the nodal block affects the scalability of the algorithm after 512 cores. We can use preconditioned conjugate gradient with a lenient tolerance and even though it may increase the number of outer iterations it can improve the overall scalability of the algorithm for large number of elements. Some of the initial studies we performed shows promise in this direction and further detailed study is required. Similarly in the AMG v-cycle for the nodal block we can optimize the performance by experimenting with different coarsening strategies, smoothers, etc. Finally, since the multilevel solver for the nodal block shows promise for more scalability than AMG more experimentation especially in 3D is required. We also want to mention that the block preconditioning strategy mentioned is not specific to incompressible resistive MHD and can be applied to incompressible Navier–Stokes and Stokes equations. Thus this strategy can also give efficient solvers for the fluid dynamical incompressible equations and thus needs investigation.

- **Nonlinear solvers:** In this work we used mostly the Picard solver for dealing with the nonlinear system. Newton's method and Anderson acceleration to

the Picard solver can give faster and more robust nonlinear solvers and hence they can be employed. Another strategy to deal with the nonlinearity is the implicit-explicit (IMEX) time stepping, where we can use implicit HDG scheme for the linearized part and explicit DG scheme for the nonlinear part [85]. This can effectively deal with the fast waves and also just need one linear solve per stage of time discretization. It will be interesting to study how the block preconditioning performs in combination with these other nonlinear solvers.

Appendices

Appendix A

Proof of well-posedness of local solver of the iHDG-II method for the linear convection-diffusion equation

Proof. Choosing $\boldsymbol{\sigma}^{k+1}$ and u^{k+1} as test functions in (3.29a)-(3.29b), integrating the second term in (3.29a) by parts, using (3.6) for second term in (3.29b), and then summing up the resulting two equations we obtain

$$\kappa^{-1}\left(\boldsymbol{\sigma}^{k+1},\boldsymbol{\sigma}^{k+1}\right)_T+\left(\left\{\nu-\frac{\nabla\cdot\boldsymbol{\beta}}{2}\right\}u^{k+1},u^{k+1}\right)_T \\ +\left\langle\left(\frac{\boldsymbol{\beta}\cdot\mathbf{n}}{2}+\tau\right)u^{k+1},u^{k+1}\right\rangle_{\partial T}+\left\langle(\boldsymbol{\sigma}^{k+1}\cdot\mathbf{n}-\tau u^{k+1}),\hat{u}^{k,k+1}\right\rangle_{\partial T}=0. \quad \text{(A.1)}$$

Substituting (3.30) in the above equation and simplifying some terms we get

$$\sum_T\kappa^{-1}\left\|\left(\boldsymbol{\sigma}^{k+1}\right)^-\right\|_T^2+\left(\left\{\nu-\frac{\nabla\cdot\boldsymbol{\beta}}{2}\right\}\left(u^{k+1}\right)^-,\left(u^{k+1}\right)^-\right)_T \\ +\left\langle\left\{\frac{|\boldsymbol{\beta}\cdot\mathbf{n}|^2+2}{2\alpha}\right\}\left(u^{k+1}\right)^-,\left(u^{k+1}\right)^-\right\rangle_{\partial T}+\left\langle\frac{1}{\alpha}\left(\boldsymbol{\sigma}^{k+1}\cdot\mathbf{n}\right)^-,\left(\boldsymbol{\sigma}^{k+1}\cdot\mathbf{n}\right)^-\right\rangle_{\partial T} \\ +\left\langle\frac{\boldsymbol{\beta}\cdot\mathbf{n}^-}{\alpha}\left(u^{k+1}\right)^-,\left(\boldsymbol{\sigma}^{k+1}\cdot\mathbf{n}\right)^-\right\rangle_{\partial T}=\sum_{\partial T}-\left\langle\frac{1}{\alpha}\left(\boldsymbol{\sigma}^{k+1}\cdot\mathbf{n}\right)^-,\left(\boldsymbol{\sigma}^{k}\cdot\mathbf{n}\right)^+\right\rangle_{\partial T} \\ -\left\langle\left\{\frac{\boldsymbol{\beta}\cdot\mathbf{n}^++\tau^+}{\alpha}\right\}\left(\boldsymbol{\sigma}^{k+1}\cdot\mathbf{n}\right)^-,\left(u^{k}\right)^+\right\rangle_{\partial T}+\left\langle\frac{\tau^-}{\alpha}\left(u^{k+1}\right)^-,\left(\boldsymbol{\sigma}^{k}\cdot\mathbf{n}\right)^+\right\rangle_{\partial T} \\ +\left\langle\left\{\frac{\tau^-(\boldsymbol{\beta}\cdot\mathbf{n}^++\tau^+)}{\alpha}\right\}\left(u^{k+1}\right)^-,\left(u^{k}\right)^+\right\rangle_{\partial T}. \quad \text{(A.2)}$$

Using the identity

$$\left\langle\frac{\boldsymbol{\beta}\cdot\mathbf{n}}{\alpha}u^{k+1},\boldsymbol{\sigma}^{k+1}\cdot\mathbf{n}\right\rangle_{\partial T}=\left\|\frac{1}{\sqrt{2\alpha}}\left(\boldsymbol{\beta}\cdot\mathbf{n}u^{k+1}+\boldsymbol{\sigma}^{k+1}\cdot\mathbf{n}\right)\right\|_{\partial T}^2 \\ -\left\langle\frac{\boldsymbol{\beta}\cdot\mathbf{n}^2}{2\alpha}u^{k+1},u^{k+1}\right\rangle_{\partial T}-\left\langle\frac{1}{2\alpha}\boldsymbol{\sigma}^{k+1}\cdot\mathbf{n},\boldsymbol{\sigma}^{k+1}\cdot\mathbf{n}\right\rangle_{\partial T}, \quad \text{(A.3)}$$

and the coercivity condition (2.35) we can write (A.2) as

$$\sum_T \kappa^{-1} \left\| \left(\boldsymbol{\sigma}^{k+1}\right)^- \right\|_T^2 + \lambda \left\| \left(u^{k+1}\right)^- \right\|_T^2 + \left\| \left(u^{k+1}\right)^- \right\|_{1/\alpha,\partial T}^2$$
$$+ \left\| \left(\boldsymbol{\sigma}^{k+1} \cdot \mathbf{n}\right)^- \right\|_{1/2\alpha,\partial T}^2 + \left\| \frac{1}{\sqrt{2\alpha}} \left\{ \boldsymbol{\beta} \cdot \mathbf{n}^- \left(u^{k+1}\right)^- + \left(\boldsymbol{\sigma}^{k+1} \cdot \mathbf{n}\right)^- \right\} \right\|_{\partial T}^2 \leq$$
$$\sum_{\partial T} - \left\langle \frac{1}{\alpha} \left(\boldsymbol{\sigma}^{k+1} \cdot \mathbf{n}\right)^-, \left(\boldsymbol{\sigma}^k \cdot \mathbf{n}\right)^+ \right\rangle_{\partial T} - \left\langle \left\{ \frac{\boldsymbol{\beta} \cdot \mathbf{n}^+ + \tau^+}{\alpha} \right\} \left(\boldsymbol{\sigma}^{k+1} \cdot \mathbf{n}\right)^-, \left(u^k\right)^+ \right\rangle_{\partial T}$$
$$+ \left\langle \frac{\tau^-}{\alpha} \left(u^{k+1}\right)^-, \left(\boldsymbol{\sigma}^k \cdot \mathbf{n}\right)^+ \right\rangle_{\partial T} + \left\langle \left\{ \frac{\tau^-(\boldsymbol{\beta} \cdot \mathbf{n}^+ + \tau^+)}{\alpha} \right\} \left(u^{k+1}\right)^-, \left(u^k\right)^+ \right\rangle_{\partial T}. \quad \text{(A.4)}$$

Since all the terms on the left hand side are positive, when we take the "forcing" to the local solver $\left\{ \left(u^k\right)^+, \left(\boldsymbol{\sigma}^k\right)^+ \right\} = \{0, \mathbf{0}\}$, the only solution possible is $\left\{ \left(u^{k+1}\right)^-, \left(\boldsymbol{\sigma}^{k+1}\right)^- \right\} = \{0, \mathbf{0}\}$ and hence the method is well-posed. □

Appendix B

Proof of convergence of the iHDG-II method for the linear convection-diffusion equation

Proof. In equation (A.4) omitting the last term on the left hand side and using Cauchy-Schwarz and Young's inequalities for the terms on the right-hand side we get

$$\begin{aligned}
&\sum_T \kappa^{-1} \left\| \left(\boldsymbol{\sigma}^{k+1}\right)^- \right\|_T^2 + \lambda \left\| \left(u^{k+1}\right)^- \right\|_T^2 + \left\| \left(u^{k+1}\right)^- \right\|_{1/\alpha,\partial T}^2 \\
&+ \left\| \left(\boldsymbol{\sigma}^{k+1}\cdot\mathbf{n}\right)^- \right\|_{1/2\alpha,\partial T}^2 \leq \sum_{\partial T} \frac{1}{2\varepsilon} \left\langle \left\{ \frac{\tau^- + 1}{\alpha} \right\} \left(\boldsymbol{\sigma}^k\cdot\mathbf{n}\right)^+, \left(\boldsymbol{\sigma}^k\cdot\mathbf{n}\right)^+ \right\rangle_{\partial T} \\
&+ \frac{1}{2\varepsilon} \left\langle \left\{ \frac{(1+\tau^-)(\boldsymbol{\beta}\cdot\mathbf{n}^+ + \tau^+)}{\alpha} \right\} \left(u^k\right)^+, \left(u^k\right)^+ \right\rangle_{\partial T} \\
&+ \frac{\varepsilon}{2} \left\langle \left\{ \frac{1+\tau^+ + \boldsymbol{\beta}\cdot\mathbf{n}^+}{\alpha} \right\} \left(\boldsymbol{\sigma}^{k+1}\cdot\mathbf{n}\right)^-, \left(\boldsymbol{\sigma}^{k+1}\cdot\mathbf{n}\right)^- \right\rangle_{\partial T} \\
&+ \frac{\varepsilon}{2} \left\langle \left\{ \frac{\tau^-(1+\tau^+ + \boldsymbol{\beta}\cdot\mathbf{n}^+)}{\alpha} \right\} \left(u^{k+1}\right)^-, \left(u^{k+1}\right)^- \right\rangle_{\partial T}.
\end{aligned} \tag{B.1}$$

We can write the above inequality as

$$\begin{aligned}
&\sum_T \kappa^{-1} \left\| \left(\boldsymbol{\sigma}^{k+1}\right)^- \right\|_T^2 + \lambda \left\| \left(u^{k+1}\right)^- \right\|_T^2 + \frac{1}{\bar{\alpha}} \left\| \left(u^{k+1}\right)^- \right\|_{\partial T}^2 + \frac{1}{2\bar{\alpha}} \left\| \left(\boldsymbol{\sigma}^{k+1}\cdot\mathbf{n}\right)^- \right\|_{\partial T}^2 \\
&\leq \sum_{\partial T} \frac{\bar{\tau}+1}{2\varepsilon\alpha_*} \left\| \left(\boldsymbol{\sigma}^k\cdot\mathbf{n}\right)^+ \right\|_{\partial T}^2 + \frac{(1+\bar{\tau})(\|\boldsymbol{\beta}\cdot\mathbf{n}\|_{L^\infty(\partial T)} + \bar{\tau})}{2\varepsilon\alpha_*} \left\| \left(u^k\right)^+ \right\|_{\partial T}^2 \\
&+ \frac{\varepsilon(1+\bar{\tau}+\|\boldsymbol{\beta}\cdot\mathbf{n}\|_{L^\infty(\partial T)})}{2\alpha_*} \left\| \left(\boldsymbol{\sigma}^{k+1}\cdot\mathbf{n}\right)^- \right\|_{\partial T}^2 \\
&+ \frac{\varepsilon\bar{\tau}(1+\bar{\tau}+\|\boldsymbol{\beta}\cdot\mathbf{n}\|_{L^\infty(\partial T)})}{2\alpha_*} \left\| \left(u^{k+1}\right)^- \right\|_{\partial T}^2,
\end{aligned} \tag{B.2}$$

where $\bar{\tau} := \|\tau\|_{L^\infty(\partial\Omega_h)}$, $\bar{\alpha} := \|\alpha\|_{L^\infty(\partial\Omega_h)}$, and $\alpha_* := \inf_{\partial T\in\partial\Omega_h} \alpha$.

By the inverse trace inequality (2.30) we infer from (B.2) that

$$\sum_{\partial T} \mathcal{B}_1 \left\| \left(\boldsymbol{\sigma}^{k+1} \cdot \mathbf{n}\right)^- \right\|_{\partial T}^2 + \mathcal{B}_2 \left\| \left(u^{k+1}\right)^- \right\|_{\partial T}^2 \\ \leq \sum_{\partial T} \left[\mathcal{C}_1 \left\| \left(u^k\right)^+ \right\|_{\partial T}^2 + \mathcal{C}_2 \left\| \left(\boldsymbol{\sigma}^k \cdot \mathbf{n}\right)^+ \right\|_{\partial T}^2 \right],$$

which implies

$$\left\| \left(\boldsymbol{\sigma}^{k+1} \cdot \mathbf{n}, u^{k+1}\right) \right\|_{\mathcal{E}_h}^2 \leq \mathcal{D} \left\| \left(\boldsymbol{\sigma}^k \cdot \mathbf{n}, u^k\right) \right\|_{\mathcal{E}_h}^2,$$

where the constant $\mathcal{D}$ is computed as in (3.33). Therefore

$$\left\| \left(\boldsymbol{\sigma}^{k+1} \cdot \mathbf{n}, u^{k+1}\right) \right\|_{\mathcal{E}_h}^2 \leq \mathcal{D}^{k+1} \left\| \left(\boldsymbol{\sigma}^0 \cdot \mathbf{n}, u^0\right) \right\|_{\mathcal{E}_h}^2. \tag{B.3}$$

Inequalities (B.2) and (B.3) imply

$$\left\| \left(\boldsymbol{\sigma}^{k+1}, u^{k+1}\right) \right\|_{\Omega_h}^2 \leq (\mathcal{E}\mathcal{D} + \mathcal{F})\mathcal{D}^k \left\| \left(\boldsymbol{\sigma}^0 \cdot \mathbf{n}, u^0\right) \right\|_{\mathcal{E}_h}^2,$$

and this concludes the proof.

□

Bibliography

[1] M. Adams, M. Brezina, J. Hu, and R. Tuminaro. Parallel multigrid smoothing: polynomial versus Gauss–Seidel. *Journal of Computational Physics*, 188(2): 593–610, 2003.

[2] E. Allgower and K. Böhmer. Application of the mesh independence principle to mesh refinement strategies. *SIAM journal on numerical analysis*, 24(6):1335–1351, 1987.

[3] G. Alzetta, D. Arndt, W. Bangerth, V. Boddu, B. Brands, D. Davydov, R. Gassmöller, T. Heister, L. Heltai, K. Kormann, et al. The deal. ii library, version 9.0. *Journal of Numerical Mathematics*, 26(4):173–183, 2018.

[4] J. Argyris and D. Scharpf. Finite elements in time and space. *Nuclear Engineering and Design*, 10(4):456–464, 1969.

[5] D. N. Arnold. An interior penalty finite element method with discontinuous elements. *SIAM journal on numerical analysis*, 19(4):742–760, 1982.

[6] D. N. Arnold and F. Brezzi. Mixed and nonconforming finite element methods: implementation, postprocessing and error estimates. *RAIRO Modél. Math. Anal. Numér.*, 19(1):7–32, 1985. ISSN 0764-583X.

[7] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM journal on numerical analysis*, 39(5):1749–1779, 2002.

[8] H. Atkins and B. Helenbrook. Numerical evaluation of p-multigrid method for the solution of discontinuous galerkin discretizations of diffusive equations. In *17th AIAA Computational Fluid Dynamics Conference*, page 5110, 2005.

[9] M. Bader and C. Zenger. A robust and parallel multigrid method for convection diffusion equations. *Electronic Transactions on Numerical Analysis*, 15:122–131, 2003.

[10] W. Bangerth, R. Hartmann, and G. Kanschat. deal.II – a general purpose object oriented finite element library. *ACM Trans. Math. Softw.*, 33(4):24/1–24/27, 2007.

[11] R. E. Bank. A comparison of two multilevel iterative methods for nonsymmetric and indefinite elliptic finite element equations. *SIAM Journal on Numerical Analysis*, 18(4):724–743, 1981.

[12] O. Bashir, K. Willcox, O. Ghattas, B. van Bloemen Waanders, and J. Hill. Hessian-based model reduction for large-scale systems with initial condition inputs. *International Journal for Numerical Methods in Engineering*, 73:844–868, 2008.

[13] J. K. Bennighof and R. B. Lehoucq. An automated multilevel substructuring method for eigenspace computation in linear elastodynamics. *SIAM Journal on Scientific Computing*, 25(6):2084–2106, 2004.

[14] D. Biskamp. Magnetic reconnection in plasmas. *Astrophysics and Space Science*, 242(1-2):165–207, 1996.

[15] D. Braess and R. Verfurth. Multigrid methods for nonconforming finite element methods. *SIAM J. Numer. Anal.*, 27(4):pp. 979–986, 1990. ISSN 00361429.

[16] D. Braess, W. Dahmen, and C. Wieners. A multigrid algorithm for the mortar finite element method. *SIAM J. Numer. Anal.*, 37(1):48–69, 1999. ISSN 0036-1429.

[17] J. Bramble, R. Ewing, J. Pasciak, and J. Shen. The analysis of multigrid algorithms for cell centered finite difference methods. *Advances in Computational Mathematics*, 5(1):15–29, 1996.

[18] J. H. Bramble. *Multigrid methods*, volume 294. CRC Press, 1993.

[19] J. H. Bramble and J. E. Pasicak. Uniform convergence estimates for multigrid v-cycle algorithms with less than full elliptic regularity. In *Widlund, Eds, Contemporary Mathematics Series 157*, pages 17–26, 1994.

[20] J. H. Bramble, J. E. Pasciak, and J. Xu. The analysis of multigrid algorithms with nonnested spaces or noninherited quadratic forms. *Math. Comput.*, 56 (193):1–34, 1991. ISSN 00255718.

[21] S. C. Brenner. A multigrid algorithm for the lowest-order Raviart-Thomas mixed triangular finite element method. *SIAM J. Numer. Anal.*, 29(3):647–678, 1992. ISSN 0036-1429.

[22] S. C. Brenner. Convergence of nonconforming multigrid methods without full elliptic regularity. *Math. Comput.*, 68:25–53, 1999.

[23] F. Brezzi and M. Fortin. *Mixed and hybrid finite element methods*, volume 15. Springer Science & Business Media, 2012.

[24] W. L. Briggs, S. F. McCormick, et al. *A multigrid tutorial*, volume 72. Siam, 2000.

[25] T. Bui-Thanh. From Godunov to a unified hybridized discontinuous Galerkin framework for partial differential equations. *Journal of Computational Physics*, 295:114–146, 2015.

[26] T. Bui-Thanh. *From Rankine-Hugoniot Condition to a Constructive Derivation of HDG Methods*, pages 483–491. Lecture Notes in Computational Sciences and Engineering. Springer, 2015.

[27] T. Bui-Thanh. Construction and analysis of HDG methods for linearized shallow water equations. *SIAM Journal on Scientific Computing*, 38(6):A3696–A3719, 2016.

[28] C. Burstedde, O. Ghattas, M. Gurnis, E. Tan, T. Tu, G. Stadler, L. C. Wilcox, and S. Zhong. Scalable adaptive mantle convection simulation on petascale supercomputers. In *SC08: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM/IEEE, 2008. ISBN 978-1-4244-2835-9.

[29] C. Burstedde, O. Ghattas, M. Gurnis, T. Isaac, G. Stadler, T. Warburton, and L. C. Wilcox. Extreme-scale AMR. In *SC10: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM/IEEE, 2010.

[30] L. Chacón. An optimal, parallel, fully implicit newton–krylov solver for three-dimensional viscoresistive magnetohydrodynamics. *Physics of Plasmas*, 15(5): 056103, 2008.

[31] J. Chan, Z. Wang, A. Modave, J.-F. Remacle, and T. Warburton. GPU-accelerated discontinuous galerkin methods on hybrid meshes. *Journal of Computational Physics*, 318:142–168, 2016.

[32] H. Chen, P. Lu, and X. Xu. A robust multilevel method for hybridizable discontinuous Galerkin method for the Helmholtz equation. *Journal of Computational Physics*, 264:133–151, 2014.

[33] L. Chen, J. Wang, Y. Wang, and X. Ye. An auxiliary space multigrid preconditioner for the weak Galerkin method. *Computers & Mathematics with Applications*, 70(4):330–344, 2015.

[34] Z. Chen. Equivalence between and multigrid algorithms for nonconforming and mixed methods for second-order elliptic problems. *East-West J. Numer. Math.*, 4(1):1–33, 1996. ISSN 0928-0200.

[35] M. Christie, M. Blunt, et al. Tenth SPE comparative solution project: A comparison of upscaling techniques. In *SPE Reservoir Simulation Symposium*. Society of Petroleum Engineers, 2001.

[36] B. Cockburn and J. Gopalakrishnan. The derivation of hybridizable discontinuous Galerkin methods for Stokes flow. *SIAM J. Numer. Anal*, 47(2):1092–1125, 2009.

[37] B. Cockburn, B. Dong, and J. Guzmán. A superconvergent LDG-hybridizable Galerkin method for second-order elliptic problems. *Mathematics of Computation*, 77(264):1887–1916, 2008.

[38] B. Cockburn, J. Gopalakrishnan, and R. Lazarov. Unified hybridization of discontinuous Galerkin, mixed, and continuous Galerkin methods for second order elliptic problems. *SIAM J. Numer. Anal.*, 47(2):1319–1365, 2009. ISSN 0036-1429.

[39] B. Cockburn, J. Gopalakrishnan, and F.-J. Sayas. A projection-based error analysis of HDG methods. *Mathematics Of Computation*, 79(271):1351–1367, 2010.

[40] B. Cockburn, J. Gopalakrishnan, and F.-J. Sayas. A projection-based error analysis of HDG methods. *Math. Comp.*, 79(271):1351–1367, 2010. ISSN 0025-5718.

[41] B. Cockburn, O. Dubois, J. Gopalakrishnan, and S. Tan. Multigrid for an HDG method. *IMA Journal of Numerical Analysis*, 34(4):1386–1425, 2014.

[42] R. Codina and N. Hernández-Silva. Stabilized finite element approximation of the stationary magneto-hydrodynamics equations. *Computational Mechanics*, 38(4-5):344–355, 2006.

[43] E. C. Cyr, J. N. Shadid, R. S. Tuminaro, R. P. Pawlowski, and L. Chacón. A new approximate block factorization preconditioner for two-dimensional incompressible (reduced) resistive MHD. *SIAM Journal on Scientific Computing*, 35 (3):B701–B730, 2013.

[44] T. A. Davis. Algorithm 832: UMFPACK V4. 3—an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software (TOMS)*, 30 (2):196–199, 2004.

[45] T. A. Davis. *Direct methods for sparse linear systems*, volume 2. Siam, 2006.

[46] P. M. De Zeeuw. Matrix-dependent prolongations and restrictions in a blackbox multigrid solver. *Journal of computational and applied mathematics*, 33(1):1–27, 1990.

[47] J. Dendy. Black box multigrid. *Journal of Computational Physics*, 48(3):366–386, 1982.

[48] J. Dendy Jr. Black box multigrid for nonsymmetric problems. *Applied Mathematics and Computation*, 13(3-4):261–283, 1983.

[49] L. T. Diosady. *Domain decomposition preconditioners for higher-order discontinuous Galerkin discretizations*. PhD thesis, Massachusetts Institute of Technology, 2011.

[50] I. S. Duff and J. K. Reid. The multifrontal solution of indefinite sparse symmetric linear. *ACM Transactions on Mathematical Software (TOMS)*, 9(3): 302–325, 1983.

[51] V. Eijkhout. *Introduction to High Performance Scientific Computing*. Lulu. com, 2014.

[52] T. Ellis, J. Chan, and L. Demkowicz. Robust DPG methods for transient convection-diffusion. In *Building Bridges: Connections and Challenges in Modern Approaches to Numerical Partial Differential Equations*, pages 179–203. Springer, 2016.

[53] H. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro. Block preconditioners based on approximate commutators. *SIAM Journal on Scientific Computing*, 27(5):1651–1668, 2006.

[54] H. C. Elman. Preconditioning for the steady-state Navier–Stokes equations with low viscosity. *SIAM Journal on Scientific Computing*, 20(4):1299–1316, 1999.

[55] H. C. Elman, O. G. Ernst, and D. P. O'leary. A multigrid method enhanced by krylov subspace iteration for discrete Helmholtz equations. *SIAM Journal on scientific computing*, 23(4):1291–1315, 2001.

[56] C. Farhat and M. Chandesris. Time-decomposed parallel time-integrators: theory and feasibility studies for uid, structure, and fluid-structure applications. *International Journal for Numerical Methods in Engineering*, 58(9):1397–1434, 2003.

[57] K. Fidkowski. Algebraic tailoring of discontinuous galerkin p-multigrid for convection. *Computers & Fluids*, 98:164–176, 2014.

[58] M. Gander and S. Hajian. Analysis of schwarz methods for a hybridizable discontinuous Galerkin discretization: The many-subdomain case. *Mathematics of Computation*, 87(312):1635–1657, 2018.

[59] M. J. Gander. Analysis of the parareal algorithm applied to hyperbolic problems using characteristics. *Boletin de la Sociedad Espanola de Matemática Aplicada*, 42:21–35, 2008.

[60] M. J. Gander and S. Hajian. Block jacobi for discontinuous Galerkin discretizations: no ordinary Schwarz methods. In *Domain Decomposition Methods in Science and Engineering XXI*, pages 305–313. Springer, 2014.

[61] M. J. Gander and S. Hajian. Analysis of Schwarz methods for a hybridizable discontinuous Galerkin discretization. *SIAM J. Numer. Anal.*, 53(1):573–597, 2015. ISSN 0036-1429.

[62] M. J. Gander and S. Hajian. Analysis of schwarz methods for a hybridizable discontinuous galerkin discretization: the many subdomain case. *arXiv preprint arXiv:1603.04073*, 2016.

[63] M. J. Gander and S. Vandewalle. Analysis of the parareal time-parallel time-integration method. *SIAM Journal on Scientific Computing*, 29(2):556–578, 2007.

[64] R. Gandham, D. Medina, and T. Warburton. GPU accelerated discontinuous Galerkin methods for shallow water equations. *arXiv*, 1(1403.1661), 2014.

[65] I. Garrido, B. Lee, G. Fladmark, and M. Espedal. Convergent iterative schemes for time parallelization. *Mathematics of computation*, 75(255):1403–1428, 2006.

[66] M. W. Gee, C. M. Siefert, J. J. Hu, R. S. Tuminaro, and M. G. Sala. ML 5.0 smoothed aggregation user's guide. Technical report, Technical Report SAND2006-2649, Sandia National Laboratories, 2006.

[67] A. George. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10(2):345–363, 1973.

[68] A. George and J. W. Liu. An automatic nested dissection algorithm for irregular finite element problems. *SIAM Journal on Numerical Analysis*, 15(5):1053–1069, 1978.

[69] A. Gillman and P.-G. Martinsson. A direct solver with O(N) complexity for variable coefficient elliptic PDEs discretized via a high-order composite spectral collocation method. *SIAM Journal on Scientific Computing*, 36(4):A2023–A2046, 2014.

[70] F. X. Giraldo and T. Warburton. A high-order triangular discontinous Galerkin oceanic shallow water model. *International Journal For Numerical Methods In Fluids*, 56:899–925, 2008.

[71] V. Girault, S. Sun, M. F. Wheeler, and I. Yotov. Coupling discontinuous Galerkin and mixed finite element discretizations using mortar finite elements. *SIAM J. Numer. Anal.*, 46(2):949–979, 2008. ISSN 0036-1429.

[72] J. P. Goedbloed and S. Poedts. *Principles of magnetohydrodynamics: with applications to laboratory and astrophysical plasmas*. Cambridge University Press, 2004.

[73] J. P. Goedbloed, R. Keppens, and S. Poedts. *Advanced magnetohydrodynamics: with applications to laboratory and astrophysical plasmas*. Cambridge University Press, 2010.

[74] J. Gopalakrishnan and S. Tan. A convergent multigrid cycle for the hybridized mixed method. *Numer. Linear Algebra Appl.*, 16(9):689–714, 2009.

[75] A. Greenbaum. *Iterative Methods for the Solution of Linear Systems*. SIAM, Philadelphia, 1997.

[76] R. Griesmaier and P. Monk. Error analysis for a hybridizable discontinous Galerkin method for the Helmholtz equation. *J. Sci. Comput.*, 49:291–310, 2011.

[77] W. Hackbusch and T. Probst. Downwind Gauss-Seidel smoothing for convection dominated problems. *Numerical linear algebra with applications*, 4(2):85–102, 1997.

[78] Y.-X. He, L. Li, S. Lanteri, and T.-Z. Huang. Optimized schwarz algorithms for solving time-harmonic Maxwell's equations discretized by a hybridizable discontinuous Galerkin method. *Computer Physics Communications*, 200:176–181, 2016.

[79] B. T. Helenbrook and H. L. Atkins. Application of p-multigrid to discontinuous Galerkin formulations of the poisson equation. *AIAA journal*, 44(3):566–575, 2006.

[80] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, et al. An overview of the trilinos project. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):397–423, 2005.

[81] K. L. Ho and L. Ying. Hierarchical interpolative factorization for elliptic operators: differential equations. *Communications on Pure and Applied Mathematics*, 69(8):1415–1451, 2016.

[82] P. Houston, D. Schötzau, and X. Wei. A mixed DG method for linearized incompressible magnetohydrodynamics. *J. Sci. Comput.*, 40(1-3):281–314, 2009. ISSN 0885-7474.

[83] C. Johnson and J. Pitkäranta. An analysis of the discontinuous Galerkin method for a scalar hyperbolic equation. *Mathematics of Computation*, 46(173):1–26, 1986.

[84] Z. Kaczkowski. The method of finite space-time elements in dynamics of structures. *Journal of Technical Physics*, 16(1):69–84, 1975.

[85] S. Kang, F. X. Giraldo, and T. Bui-Thanh. IMEX HDG-DG: a coupled implicit hybridized discontinuous Galerkin (HDG) and explicit discontinuous Galerkin (DG) approach for shallow water systems. *arXiv preprint arXiv:1711.02751*, 2017.

[86] G. Kanschat. Robust smoothers for high-order discontinuous galerkin discretizations of advection–diffusion problems. *Journal of Computational and Applied Mathematics*, 218(1):53–60, 2008.

[87] C. A. Kennedy and M. H. Carpenter. Diagonally implicit Runge-Kutta methods for ordinary differential equations. A review. 2016.

[88] H. Kim, J. Xu, and L. Zikatanov. Uniformly convergent multigrid methods for convection–diffusion problems without any constraint on coarse grids. *Advances in Computational Mathematics*, 20(4):385–399, 2004.

[89] C. M. Klaij, J. J. van der Vegt, and H. van der Ven. Space–time discontinuous Galerkin method for the compressible Navier–Stokes equations. *Journal of Computational Physics*, 217(2):589–611, 2006.

[90] L. I. G. Kovasznay. Laminar flow behind a two-dimensional grid. *Mathematical Proceedings of the Cambridge Philosophical Society*, 44:58–62, 1 1948. ISSN 1469-8064.

[91] M. Kronbichler and W. A. Wall. A performance comparison of continuous and discontinuous Galerkin methods with fast multigrid solvers. *SIAM Journal on Scientific Computing*, 40(5):A3423–A3448, 2018.

[92] D. Y. Kwak. V-cycle multigrid for cell-centered finite differences. *SIAM Journal on Scientific Computing*, 21(2):552–564, 1999.

[93] J. J. Lee, S. Shannon, T. Bui-Thanh, and J. N. Shadid. Analysis of an HDG method for linearized incompressible resistive mhd equations. *arXiv preprint arXiv:1702.05124*, 2017.

[94] J. J. Lee, S. Shannon, T. Bui-Thanh, and J. N. Shadid. Construction and analysis of an HDG method for Incompressible Magnetohydrodynamics. *arXiv preprint arXiv:1702.05124*, 2017.

[95] P. LeSaint and P. A. Raviart. On a finite element method for solving the neutron transport equation. In C. de Boor, editor, *Mathematical Aspects of Finite Element Methods in Partial Differential Equations*, pages 89–145. Academic Press, 1974.

[96] L. Li, S. Lanteri, and R. Perrussel. A hybridizable discontinuous Galerkin method combined to a Schwarz algorithm for the solution of 3d time-harmonic Maxwell's equation. *Journal of Computational Physics*, 256:563–581, 2014.

[97] X. S. Li and J. W. Demmel. Superlu_dist: A scalable distributed-memory sparse direct solver for unsymmetric linear systems. *ACM Transactions on Mathematical Software (TOMS)*, 29(2):110–140, 2003.

[98] P. Lin, J. N. Shadid, E. G. Phillips, J. J. Hu, E. C. Cyr, R. P. Pawlowski, A. O. Prokopenko, and P. L. Tsuji. Performance of Scalable AMG-based Preconditioners for MHD and Multifluid Plasma Simulations. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States); Sandia, 2017.

[99] J.-L. Lions, Y. Maday, and G. Turinici. Résolution d'edp par un schéma en temps pararéel. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 332(7):661–668, 2001.

[100] R. J. Lipton, D. J. Rose, and R. E. Tarjan. Generalized nested dissection. *SIAM journal on numerical analysis*, 16(2):346–358, 1979.

[101] J. W. Liu. The multifrontal method for sparse matrix solution: Theory and practice. *SIAM review*, 34(1):82–109, 1992.

[102] Y. Maday and G. Turinici. A parareal in time procedure for the control of partial differential equations. *Comptes Rendus Mathematique*, 335(4):387–392, 2002.

[103] P.-G. Martinsson. A fast direct solver for a class of elliptic partial differential equations. *Journal of Scientific Computing*, 38(3):316–330, 2009.

[104] D. A. May and L. Moresi. Preconditioned iterative methods for Stokes flow problems arising in computational geodynamics. *Physics of the Earth and Planetary Interiors*, 171(1-4):33–47, 2008.

[105] M. Minion. A hybrid parareal spectral deferred corrections method. *Communications in Applied Mathematics and Computational Science*, 5(2):265–301, 2011.

[106] D. Moro, N. C. Nguyen, and J. Peraire. Navier-Stokes solution using hybridizable discontinuous Galerkin methods. *American Institute of Aeronautics and Astronautics*, 2011-3407, 2011.

[107] U. Müller and L. Bühler. *Magnetofluiddynamics in Channels and Containers*. Springer Science & Business Media, 2013.

[108] S. Muralikrishnan, M.-B. Tran, and T. Bui-Thanh. iHDG: An iterative HDG framework for partial differential equations. *SIAM Journal on Scientific Computing*, 39(5):S782–S808, 2017.

[109] S. Muralikrishnan, M.-B. Tran, and T. Bui-Thanh. An improved iterative hdg approach for partial differential equations. *Journal of Computational Physics*, 367:295–321, 2018.

[110] S. Muralikrishnan, T. Bui-Thanh, and J. N. Shadid. A Multilevel approach for trace system in HDG discretizations. *arXiv preprint arXiv:1903.11045*, 2019.

[111] M. F. Murphy, G. H. Golub, and A. J. Wathen. A note on preconditioning for indefinite linear systems. *SIAM Journal on Scientific Computing*, 21(6): 1969–1972, 2000.

[112] N. C. Nguyen, J. Peraire, and B. Cockburn. An implicit high-order hybridizable discontinuous Galerkin method for linear convection-diffusion equations. *Journal Computational Physics*, 228:3232–3254, 2009.

[113] N. C. Nguyen, J. Peraire, and B. Cockburn. A hybridizable discontinous Galerkin method for Stokes flow. *Comput Method Appl. Mech. Eng.*, 199:582–597, 2010.

[114] N. C. Nguyen, J. Peraire, and B. Cockburn. A hybridizable discontinuous galerkin method for stokes flow. *Computer Methods in Applied Mechanics and Engineering*, 199(9-12):582–597, 2010.

[115] N. C. Nguyen, J. Peraire, and B. Cockburn. An implicit high-order hybridizable discontinuous Galerkin method for the incompressible Navier-Stokes equations. *Journal Computational Physics*, 230:1147–1170, 2011.

[116] N. C. Nguyen, J. Peraire, and B. Cockburn. High-order implicit hybridizable discontinuous Galerkin method for acoustics and elastodynamics. *Journal Computational Physics*, 230:3695–3718, 2011.

[117] N. C. Nguyen, J. Peraire, and B. Cockburn. Hybridizable discontinuous Galerkin method for the time harmonic Maxwell's equations. *Journal Computational Physics*, 230:7151–7175, 2011.

[118] J. T. Oden. A general theory of finite elements. ii. applications. *International Journal for Numerical Methods in Engineering*, 1(3):247–259, 1969.

[119] E. G. Phillips, J. N. Shadid, E. C. Cyr, H. C. Elman, and R. P. Pawlowski. Block preconditioners for stable mixed nodal and edge finite element representations of incompressible resistive MHD. *SIAM Journal on Scientific Computing*, 38 (6):B1009–B1031, 2016.

[120] J. L. Poulson. *Fast parallel solution of heterogeneous 3D time-harmonic wave equations*. PhD thesis, 2012.

[121] A. Prokopenko, J. J. Hu, T. A. Wiesner, C. M. Siefert, and R. S. Tuminaro. Muelu user's guide 1.0. *Techical Report SAND2014-18874, Sandia National Laboratories*, 2014.

[122] A. Quarteroni and A. Valli. Domain decomposition methods for partial differential equations numerical mathematics and scientific computation. *Quarteroni, A. Valli–New York: Oxford University Press.–1999*, 1999.

[123] A. Reusken. Multigrid with matrix-dependent transfer operators for a singular perturbation problem. *Computing*, 50(3):199–211, 1993.

[124] A. Reusken. Multigrid with matrix-dependent transfer operators for convection-diffusion problems. In *Multigrid Methods IV*, pages 269–280. Springer, 1994.

[125] S. Rhebergen and B. Cockburn. A space–time hybridizable discontinuous galerkin method for incompressible flows on deforming domains. *Journal of Computational Physics*, 231(11):4185–4204, 2012.

[126] S. Rhebergen and G. N. Wells. Analysis of a hybridized/interface stabilized finite element method for the Stokes equations. *SIAM Journal on Numerical Analysis*, 55(4):1982–2003, 2017.

[127] S. Rhebergen and G. N. Wells. A hybridizable discontinuous Galerkin method for the Navier–Stokes equations with pointwise divergence-free velocity field. *Journal of Scientific Computing*, pages 1–18, 2018.

[128] S. Rhebergen and G. N. Wells. Preconditioning of a hybridized discontinuous Galerkin finite element method for the Stokes equations. *Journal of Scientific Computing*, 77(3):1936–1952, 2018.

[129] B. Rivière, M. F. Wheeler, and V. Girault. A priori error estimates for finite element methods based on discontinuous approximation spaces for elliptic problems. *SIAM Journal on Numerical Analysis*, 39(3):902–931, 2001.

[130] J. E. Roberts and J.-M. Thomas. Mixed and hybrid methods. 1991.

[131] U. Rüde, K. Willcox, L. C. McInnes, and H. D. Sterck. Research and education in computational science and engineering. *SIAM Review*, 60(3):707–754, 2018.

[132] J. Rudi. *Global convection in Earth's mantle: advanced numerical methods and extreme-scale simulations*. PhD thesis, 2019.

[133] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific Computing*, 14(2):461–469, 1993.

[134] Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 2003.

[135] N. B. Salah, A. Soulaimani, W. G. Habashi, and M. Fortin. A conservative stabilized finite element method for the magneto-hydrodynamic equations. *International Journal for Numerical Methods in Fluids*, 29(5):535–554, 1999.

[136] O. San and K. Kara. High-order accurate spectral difference method for shallow water equations. *International Journal of Research and Reviews in Applied Sciences*, 6:41–54, 2011.

[137] P. G. Schmitz and L. Ying. A fast direct solver for elliptic problems on general meshes in 2D. *Journal of Computational Physics*, 231(4):1314–1338, 2012.

[138] D. Schötzau. Mixed finite element methods for stationary incompressible magneto–hydrodynamics. *Numerische Mathematik*, 96(4):771–800, 2004.

[139] J. N. Shadid, R. P. Pawlowski, E. C. Cyr, R. S. Tuminaro, L. Chacón, and P. D. Weber. Scalable implicit incompressible resistive MHD with stabilized FE and fully-coupled Newton–Krylov-AMG. *Computer Methods in Applied Mechanics and Engineering*, 304:1–25, 2016.

[140] S. Shannon. *Hybridized Discontinuous Galerkin Methods for Magnetohydrodynamics*. PhD thesis, 2018.

[141] B. Smith, P. Bjorstad, W. D. Gropp, and W. Gropp. *Domain decomposition: parallel multilevel methods for elliptic partial differential equations*. Cambridge university press, 2004.

[142] H. Sundar, G. Stadler, and G. Biros. Comparison of multigrid algorithms for high-order continuous finite element discretizations. *Numerical Linear Algebra with Applications*, 22(4):664–680, 2015.

[143] M. Taylor, J. Tribbia, and M. Iskandarani. The spectral element method for the shallow water equations on the sphere. *Journal of Computational Physics*, 130(1):92 – 108, 1997.

[144] A. Toselli and O. Widlund. *Domain decomposition methods-algorithms and theory*, volume 34. Springer Science & Business Media, 2006.

[145] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid*. Academic press, 2000.

[146] P. S. Vassilevski. *Multilevel block factorization preconditioners: Matrix-based analysis and algorithms for solving finite element equations*. Springer Science & Business Media, 2008.

[147] V. Venkatakrishnan and D. J. Mavriplis. Agglomeration multigrid for the Euler equations. In *Fourteenth International Conference on Numerical Methods in Fluid Dynamics*, pages 178–182. Springer, 1995.

[148] V. Venkatakrishnan, D. J. Mavriplis, and M. J. Berger. Unstructured multigrid through agglomeration. 1993.

[149] C. Wagner, W. Kinzelbach, and G. Wittum. Schur-complement multigrid. *Numerische Mathematik*, 75(4):523–545, 1997.

[150] C. Waluga. Analysis of hybrid discontinuous Galerkin methods for incompressible flow problems. *Diplomingenieur thesis*, 2012.

[151] F. Wang and J. Xu. A crosswind block iterative method for convection-dominated problems. *SIAM Journal on Scientific Computing*, 21(2):620–645, 1999.

[152] T. Warburton and J. S. Hesthaven. On the constants in *hp*-finite element trace inverse inequalities. *Comput. Methods Appl. Mech. Engrg.*, 192(25):2765–2773, 2003. ISSN 0045-7825.

[153] M. Weiser, A. Schiela, and P. Deuflhard. Asymptotic mesh independence of Newton's method revisited. *SIAM journal on numerical analysis*, 42(5):1830–1845, 2005.

[154] M. F. Wheeler. An elliptic collocation-finite element method with interior penalties. *SIAM Journal on Numerical Analysis*, 15(1):152–161, 1978.

[155] M. F. Wheeler and M. Peszyńska. Computational engineering and science methodologies for modeling and simulation of subsurface applications. *Advances in Water Resources*, 25(8-12):1147–1173, 2002.

[156] M. F. Wheeler and I. Yotov. Multigrid on the interface for mortar mixed finite element methods for elliptic problems. *Comput. Methods Appl. Mech. Engrg*, 184:287–302, 2000.

[157] L. C. Wilcox, G. Stadler, C. Burstedde, and O. Ghattas. A high-order discontinuous Galerkin method for wave propagation through coupled elastic-acoustic media. *Journal of Computational Physics*, 229(24):9373–9396, 2010.

[158] T. Wildey, S. Muralikrishnan, and T. Bui-Thanh. Unified geometric multigrid algorithm for hybridized high-order finite element methods. *arXiv preprint arXiv:1811.09909*, 2018.

[159] T. Wildey, S. Muralikrishnan, and T. Bui-Thanh. Unified geometric multigrid algorithm for hybridized high-order finite element methods. *arXiv preprint arXiv:1811.09909*, 2018.

[160] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li. Superfast multifrontal method for large structured linear systems of equations. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1382–1411, 2009.

[161] Y. Xing and X. Zhang. Positivity-preserving well-balanced discontinuous Galerkin methods for the shallow water equations on unstructured triangular meshes. *J. Sci. Comput.*, 57:19–41, 2013.

[162] J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM review*, 34(4):581–613, 1992.

Vita

Sriramkrishnan Muralikrishnan was born in Thiruvarur, India on 24 May 1991, the son of G. Muralikrishnan and M. Valli. He received the Bachelor of Engineering degree in Aerospace Engineering from the Madras Institute of Technology in May 2012. He then went to Indian Institute of Technology, Kanpur and received the Master of Technology degree in Aerospace Engineering in May 2014. In August 2014, he started his PhD program in the Department of Aerospace Engineering and Engineering Mechanics at the University of Texas at Austin.

Permanent address: 18/1 Nettivelaikara Street,
Thiruvarur-610001,
Tamilnadu, India.

This Dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.