Domain Decomposition Preconditioners for Higher-Order Discontinuous Galerkin Discretizations

by

Laslo Tibor Diosady

S.M., Massachusetts Institute of Technology (2007)
B.A.Sc., University of Toronto (2005)

Submitted to the Department of Aeronautics and Astronautics in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2011
[February 2012]

Author ....
Department of Aeronautics and Astronautics
Sept 23, 2011

Certified by ....
David L. Darmofal
Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by ....
Alan Edelman
Professor of Mathematics
Thesis Committee

Certified by ....
Jaime Peraire
Professor of Aeronautics and Astronautics
Thesis Committee

Accepted by ....
Eytan H. Modiano
Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

Domain Decomposition Preconditioners for Higher-Order Discontinuous Galerkin Discretizations

by
Laslo Tibor Diosady

Submitted to the Department of Aeronautics and Astronautics
on Sept 23, 2011, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Aerodynamic flows involve features with a wide range of spatial and temporal scales which need to be resolved in order to accurately predict desired engineering quantities. While computational fluid dynamics (CFD) has advanced considerably in the past 30 years, the desire to perform more complex, higher-fidelity simulations remains. Present day CFD simulations are limited by the lack of an efficient high-fidelity solver able to take advantage of the massively parallel architectures of modern day supercomputers. A higher-order hybridizable discontinuous Galerkin (HDG) discretization combined with an implicit solution method is proposed as a means to attain engineering accuracy at lower computational cost. Domain decomposition methods are studied for the parallel solution of the linear system arising at each iteration of the implicit scheme.

A minimum overlapping additive Schwarz (ASM) preconditioner and a Balancing Domain Decomposition by Constraints (BDDC) preconditioner are developed for the HDG discretization. An algebraic coarse space for the ASM preconditioner is developed based on the solution of local harmonic problems. The BDDC preconditioner is proven to converge at a rate independent of the number of subdomains and only weakly dependent on the solution order or the number of elements per subdomain for a second-order elliptic problem. The BDDC preconditioner is extended to the solution of convection-dominated problems using a Robin-Robin interface condition.

An inexact BDDC preconditioner is developed based on incomplete factorizations and a p-multigrid type coarse grid correction. It is shown that the incomplete factorization of the singular linear systems corresponding to local Neumann problems results in a non-singular preconditioner. The inexact BDDC preconditioner converges in a similar number of iterations as the exact BDDC method, with significantly reduced CPU time.

The domain decomposition preconditioners are extended to solve the Euler and Navier-Stokes systems of equations. An analysis is performed to determine the effect of boundary conditions on the convergence of domain decomposition methods. Optimized Robin-Robin interface conditions are derived for the BDDC preconditioner which significantly improve the performance relative to the standard Robin-Robin interface conditions. Both ASM and BDDC preconditioners are applied to solve several fundamental aerodynamic flows. Numerical results demonstrate that for high-Reynolds number flows, solved on anisotropic meshes, a coarse space is necessary in order to obtain good performance on more than 100 processors.

Thesis Supervisor: David L. Darmofal
Title: Professor of Aeronautics and Astronautics

Acknowledgments

I would like to thank all of those who helped make this work possible. First, I would like to thank my advisor Prof. David Darmofal, for giving me the opportunity to work with him. I am grateful for his guidance and encouragement throughout my studies at MIT. Second, I would like to thank my committee members Profs. Alan Edelman and Jaime Peraire for their criticism and direction throughout my PhD work. I would also like to thank my thesis readers Prof. Qiqi Wang and Dr. Venkat Venkatakrishnan for providing comments and suggestions on improving this thesis.

This work would not have been possible without the contributions of the ProjectX team past and present (Julie Andren, Garrett Barter, Krzysztof Fidkowski, Bob Haimes, Josh Krakos, Eric Liu, JM Modisette, Todd Oliver, Mike Park, Huafei Sun, Masa Yano). Particularly, I would like to acknowledge JM whose been a good friend and office mate my entire time at MIT.

I would like to thank my wife, Laura, my parents, Klara and Levente, and my brother, Andrew, for their encouragement throughout my time at MIT. Without their love and support this work would never have been completed.

Finally, I would like to acknowledge the financial support I have received throughout my graduate studied. This work was supported by funding from MIT through the Zakhartchenko Fellowship and from The Boeing Company under technical monitor Dr. Mori Mani. This research was also supported in part by the National Science Foundation through TeraGrid resources provided under grant number TG-DMS110018.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Computational fluid dynamics (CFD) has advanced considerably in the past 30 years, such that CFD tools have become invaluable in the design and analysis of modern aircraft and spacecraft. CFD has become important due to the ability to simulate a large range of flow conditions at all relevant parameters in a flow (eg. Mach number, Reynolds number, specific heat ratio, etc.) which may not be possible in wind-tunnel experiments. Through advances in both numerical algorithms and computer hardware the complexity of problems solved has increased dramatically since the first two-dimensional potential flow simulations in the mid 1970s.

While the complexity of the problems has increased significantly, the desire to perform even more complex, higher-fidelity simulations remains. A recent review of the AIAA Drag Prediction Workshops shows that insufficient resolution remains a limiting factor in achieving accurate predictions [84]. Aerodynamic flows involve features with a large range of spatial and temporal scales which need to be resolved in order to accurately predict desired engineering quantities. Higher-order methods provide a possible means of obtaining engineering required accuracy with fewer degrees of freedom. However, the requirements to have increased spatial and temporal resolution necessitates the use of more powerful computing resources.

Today's most powerful supercomputers are able to reach a peak performance of more than one petaflops. However, peak performance has been reached by a continuing trend of parallelization, with the most powerful machines now employing more than 100,000 processors. While several CFD codes have been used on large parallel systems with up to several thousand processors, the scalability of most CFD codes tops out around 512 cpus [89]. Developing CFD codes which are able to scale efficiently to tens or hundreds of thousands of processors remains a significant challenge.

1.2 Background

1.2.1 Higher-order Discontinuous Galerkin Methods

Higher-order methods have the potential for reducing the computational cost required to obtain engineering accuracy in CFD simulations. Typical industrial CFD methods employ second-order finite volume schemes for which the error scales as $E \propto h^2$, where h is the characteristic mesh size. Thus, in three dimensions, decreasing the mesh size by a factor of two leads to a four-fold reduction in solution error, but an eight-fold increase in the number of degrees of freedom. Higher-order numerical methods are those which achieve an error convergence $E \propto h^r$, $r > 2$, provided the solution is sufficiently smooth. Thus, if a low error tolerance is required, higher-order methods allow for the error tolerance to be met with fewer degrees of freedom than typical second-order methods.

In this work a higher-order discontinuous Galerkin (DG) discretization is used [9, 11–14, 16, 37, 42, 56, 87, 97, 98]. DG methods are attractive since the piecewise discontinuous representation of the solution provides a natural means of achieving higher-order accuracy on arbitrary unstructured meshes. Even at high order, DG methods maintain a nearest-neighbour stencil as element-wise coupling is introduced only through the flux across element boundaries. In particular, this work uses a new DG variant known as the hybridizable discontinuous Galerkin (HDG) method [39–41, 96, 100]. In HDG methods, both the state variables and gradients are approximated on each element. As a result, HDG methods converge at optimal order (p+1) in both the state variables and gradients. Additionally, the state and gradient degrees of freedom may be locally eliminated to give a system of equations where the only globally-coupled degrees of freedom are associated with the trace values of the state variables on the element boundaries. Thus, for higher-order solutions HDG methods have much a smaller number of globally-coupled degrees of freedom compared to traditional

DG methods [41, 96].

1.2.2 Scalable Implicit CFD

A key use of massively parallel computers is to perform large-scale simulations in similar amount of time as typical industrial simulations on desktop machines. In addition to being able to partition the computational work among large numbers of processors, "optimal" algorithms are required for which the work scales linearly with the number of degrees of freedom. Two definitions of parallel scaling are commonly used: "strong scaling" and "weak scaling". Strong scaling, discussed in reference to Amdahl's Law [2], refers in general to parallel performance for fixed problem size, while weak scaling, discussed in reference to Gustafson's Law [61], refers to parallel performance in terms of fixed problem size per processor. While the parallel performance of a particular CFD code depends upon an efficient implementation, the performance is limited by the scalability of the underlying algorithm. Thus, this work focuses primarily on the algorithmic aspects to ensure scalability. In the context of high-fidelity CFD simulations, weak scaling is more important than strong scaling, as weak scaling relates closely to the ability of an algorithm to be optimal. Thus, unless otherwise stated the term "scalable" is used to imply "weakly scalable". An iterative solution algorithm is said to be scalable if the rate of convergence is independent of the number of subdomains into which the mesh has been partitioned, for a fixed number of elements on each subdomain. Thus, for a fixed number of elements on each subdomain, a scalable algorithm may be viewed as being optimal on a macro scale. A scalable algorithm is truly optimal if the rate convergence is also independent of the number of elements on each subdomain.

DG discretizations have long been associated with the use of explicit Runge-Kutta time integration schemes, due to the simplicity of implementation, relatively small memory requirements, and small stencils [42]. While explicit methods are relatively simple to implement, the largest allowable time step is limited by the CFL condition, hence the number of iterations required for a particular simulation depends upon the mesh size. Thus, while explicit methods have the potential for very good strong scaling, these methods are not optimal. Implicit methods, on the other hand, do not have such a time step restriction. As a result, implicit methods have become the method of choice when the time step required for numerical stability is well below that required to resolve the unsteady features of the flow. Implicit schemes have also become widely used for the solution of steady-state problems obtained through pseudo-transient continuation [66], where time-stepping enables reliable convergence for nonlinear problems [3, 10, 26, 59, 60, 69, 83, 95, 122]. While most portions of an implicit code, such as residual and Jacobian evaluations, are trivially parallelized, implicit methods require at each iteration the solution of a globally coupled system of equations. Thus, implicit algorithms are optimal only if the globally coupled system may be solved in an optimal manner.

For aerodynamic problems, the most successful solution techniques have been nonlinear multigrid methods [3, 57, 82, 90, 93, 94] and preconditioned Newton-Krylov methods [59, 60, 69, 95, 103, 122]. Mavriplis showed that using a multigrid method as a preconditioner to a Newton-Krylov approach results in significantly faster convergence in terms of CPU time than a full nonlinear multigrid scheme [88]. Thus, in this work Newton-Krylov methods are considered, where the nonlinear system is solved using an approximate Newton method, while the linear system at each Newton iteration is solved using a preconditioned Krylov subspace method. In this context, multigrid methods may be viewed as one possible choice for the preconditioner. Thus, the development of an optimal solution method hinges on the ability to develop scalable preconditioners.

1.2.3 Domain Decomposition Theory

The desire to perform large scale simulations has led to an increased interest in domain decomposition methods for the solution of large algebraic systems arising from the discretization of partial differential equations (PDEs). The term domain decomposition in the engineering community has often been used simply to refer to the partitioning of data across a parallel machine. However, data parallelism alone is insufficient to ensure good parallel performance. In particular, the performance of a domain decomposition preconditioner for the solution of large linear systems is strongly coupled to the discretization and the underlying PDE.

While high-fidelity simulations of aerodynamic flows involve solutions of the nonlinear compressible Euler and Navier-Stokes equations, performance of the algorithms developed for the systems resulting from the discretization of these equations are often analyzed in reference to simple scalar linear model equations for which the mathematical analysis is possible.

Early aerodynamic simulations involved potential flow calculations. Thus, the Poisson equation has often been used as a model problem. In particular, the elliptic nature of the Poisson equation is appropriate for the analysis of acoustic modes in low speed, incompressible flows. Convective modes, on the other hand are hyperbolic and thus a convection equation is a more appropriate model for the analysis of these modes. A singularly perturbed convection-diffusion equation is often used as a model problem for high Reynolds number compressible flows, where convective behaviour is dominant in most regions of the flow, while elliptic behaviour is dominant in the boundary layer region. Since much of the grid resolution is introduced in the boundary layer region, it is important to understand the elliptic behaviour present in these regions.

For elliptic PDEs, the Green's function extends throughout the entire domain decaying with increasing distance from the source. This implies that a residual at any point in the domain affects the solution everywhere else. In an unpreconditioned Krylov method, the application of the Jacobian matrix to a residual vector at each Krylov iteration exchanges information only to the extent of the numerical stencil. Thus, the number of iterations for an error to be transmitted across a domain of unit diameter is $O(\frac{1}{h})$, where h is the characteristic element size. In general, the convergence rate for symmetric problems is bounded by the condition number of the preconditioned system. An efficient preconditioner attempts to cluster the eigenvalues of the preconditioned system to ensure rapid convergence of the Krylov method. In particular, an efficient preconditioner for elliptic problems requires a means of controlling the lowest frequency error modes which extend throughout the domain.

Domain decomposition methods precondition the system of equations resulting from the discretization of a PDE by repeatedly solving the PDE in subdomains of the original domain. If at each iteration information is exchanged only to neighbouring subdomains, the number of iterations for an error to be transmitted across a domain of unit diameter is $O(\frac{1}{H})$, where H is the characteristic subdomain size. Thus the condition number of the preconditioned system, and hence the number of iterations required to converge the linear system, will depend on the number of subdomains. In order to ensure that the condition number of the preconditioned system is independent of H scalable preconditioners include a coarse space able to control the low frequency error modes which span the entire domain [118].

While elliptic problems are characterized by Green's functions that extend throughout the entire domain, convection-dominated problems have a hyperbolic behaviour where the errors propagate along characteristics in the flow. Thus, for convection-dominated problems, the resulting discretization is strongly coupled along the characteristics with little dissipation of errors across characteristics. Control of these errors is often accomplished by preconditioners that maintain strong coupling and often can be interpreted as increasing the propagation of errors out of the domain in the purely hyperbolic case.

For convection-diffusion problems, domain decomposition methods with a coarse space have been shown to be scalable, provided the diameter of the subdomains are sufficiently small [27, 29, 31]. Namely, if the Peclet number, defined using the subdomain length scale, H, is sufficiently small, then the behaviour matches the symmetric, diffusion-dominated limit. In the convection-dominated limit, the errors are propagated along characteristics in the domain. Thus, the number of iterations required to converge is proportional to the number of subdomains through which a characteristic must cross before exiting a domain.

In the case of unsteady convection-diffusion problems, solved using implicit time integration, analysis of domain decomposition methods shows that a coarse space may not be necessary to guarantee scalability if the time step is sufficiently small relative the size of the subdomains [28, 30]. This behaviour may be interpreted using physical intuition. Namely, for small time steps the evolution of the flow is mostly local, thus a coarse space is not required for the global control of error modes. From a linear algebra standpoint, the presence of the large temporal term leads to a diagonally dominant system, which tend to be easier to solve using iterative methods.

1.2.4 Large Scale CFD simulations

As aerodynamic flows involve both elliptic and hyperbolic features, the most successful serial algorithms have combined effective solvers for elliptic and hyperbolic problems. For example multigrid methods have been used in combination with tri-diagonal line solvers by Mavriplis and Pirzadeh [90] and Fidkowski et al. [57]. The success of these algorithms may be attributed to the ability of line solvers to control error modes in strongly coupled directions (either along characteristics or in regions of high anisotropy), while low frequency errors are corrected through the multigrid process. An alternative approach which appears to be very successful for higher-order discretizations is a two-level method using an ILU(0) preconditioner with a minimum discarded fill ordering combined with a coarse grid correction presented by Persson and Peraire [103].

The development of efficient parallel preconditioners for aerodynamic flows builds upon successful algorithms in the serial context. While multigrid methods have been employed for large-scale parallel simulations, Mavriplis notes that special care must be taken in forming the nested coarse grid problems to ensure good performance [90, 94]. Domain decomposition preconditioners presented in this thesis may be viewed as two-level preconditioners, where local solvers are employed on each subdomain, while specially constructed coarse spaces are used to ensure the control of low frequency (global) modes throughout the domain.

The most widely used domain decomposition methods for CFD applications are additive Schwarz methods [25, 32, 33, 35, 59, 60, 102, 122]. For cell-centered finite-volume, or higher-order discontinuous Galerkin discretizations, where degrees of freedom are naturally associated with element interiors, a non-overlapping additive Schwarz method corresponds to a subdomain-wise block-Jacobi preconditioner [45, 60, 102]. An overlapping additive Schwarz method may be developed by extending the size of the subdomain problems by layers of elements. Gropp et al. showed that adding a very small overlap corresponding only to a few layers of elements results in a significant improvement in the number of iterations required to converge a finite volume discretization of inviscid compressible flows [60]. However, as increasing the amount of overlap lead to increased communication costs, the lowest CPU times were achieved using an overlap region of just two layers of elements. A variant presented by Cai et al, known as the restricted additive Schwarz method, updates only locally owned degrees of freedom, eliminating communication during the solution update [35]. Numerical results have shown that this method actually requires fewer iterations to converge than the basic additive Schwarz preconditioner for both scalar convection-diffusion [35] and compressible Euler problems [33].

The use of domain decomposition methods for large scale applications involves additional considerations in order to achieve good performance. Large scale CFD applications may be both memory and CPU limited, making the exact solution of the local problems using LU factorization intractable. Thus, the local solvers are replaced with an iteration of an efficient serial preconditioner, such as an ILU factorization or a multigrid cycle. The performance of the Schwarz method will, in general, depend upon the strength of the local solver. For example, Venkatakrishnan showed significant improvement using block-ILU(0) as opposed to block-Jacobi for the local solvers for an additive Schwarz method with zero overlap [122]. ILU factorizations have been particularly popular as local solvers for additive Schwarz methods with or without a coarse correction [25, 32, 33, 59, 60, 102, 122]. Cai, Farhat and Sarkis also employed a preconditioned GMRES method to solve the local problem on each subdomain [25, 32]. In particular, this allowed for different number of iterations to be used in each subdomain ensuring that each local problem was solved with sufficient accuracy.

For practical aerodynamic flows, the question remains whether a coarse space is necessary for a scalable preconditioner. For the solution of steady compressible Euler equations, Venkatakrishnan used a coarse space developed using an algebraic multigrid-type scheme [122]. In numerical simulations with up to 128 processors, Venkatakrishnan showed that the presence of the coarse grid gives some improvement in the performance of the preconditioner in terms of number of iterations, though this did not necessarily translate into faster solution time. Gropp et al. did not employ a coarse space, and showed only modest increase in the number of linear iterations for strong scaling results from 32 to 128 processors [60]. In particular, Anderson, Gropp, and collaborators have performed large scale inviscid CFD simulations using over 3000 processors without employing a coarse space [4, 59, 60]. For unsteady simulations for the compressible Navier-Stokes equations, Cai, Farhat, and Sarkis found only a small increase in the number of iterations for strong scaling results up to 512 subdomains without the presence of a coarse space [25, 33]. Similarly, Persson showed good strong scaling performance up to 512 processors for the unsteady Navier-Stokes equations using a subdomain wise block-Jacobi preconditioner without a coarse space [102]. This observation is consistent with the theoretical result for the time-dependent convection-diffusion problems, where a coarse space is not necessary if the time step is sufficiently small.

As the time step is allowed to increase, Persson showed that the performance of the preconditioner without a coarse space degrades significantly [102]. For steady state problems solved with little or no pseudo-temporal contributions, Diosady showed very poor strong scaling results using a similar preconditioner, particularly for viscous problems [45]. A partitioning strategy weighted by the strength of the coupling between elements was introduced in order to improve the parallel scaling of this preconditioner [45]. A similar strategy was also employed by Persson [102]. However, the resulting partitions had larger surface area to volume ratios resulting in more communication per computation. While such a technique improves parallel performance on a moderate number of processors, the use of a coarse space may be essential for obtaining a scalable method for steady state viscous flow problems on massively parallel systems.

Inexact Schur complement methods have been used as an alternative means of obtaining a global correction for CFD simulations [10, 64]. Schur complement methods reduce the size of the global system to a system corresponding only to the degrees of freedom on (or near) the interfaces between subdomains. Inexact Schur complement methods solve an approximate Schur complement problem in order to precondition the global system. The solution of the approximate Schur complement problem acts as a global coarse grid correction. For example, Barth et al. developed a global preconditioner based on an approximate Schur complement for the solution of the conforming finite element discretization of the Euler equations, where approximate Schur complements were computed using ILU preconditioned GMRES [10]. Similarly, Hicken and Zingg developed a preconditioner for a finite-difference discretization of the compressible Euler equation by computing an approximate Schur complement using an ILU factorization. Both Barth et al. and Hicken and Zingg used a preconditioned GMRES method to iteratively solve the inexact Schur complement system leading to non-stationary preconditioners which were applied to the flexible variant of GMRES [109]. While much smaller than the entire global system, the Schur complement system is still globally coupled and an effective preconditioner for the Schur complement problem requires a coarse space in order to effectively control the low frequency error modes.

1.2.5 Balancing Domain Decomposition by Constraints

In order to solve the Schur complement problem, this work considers a class of domain decomposition preconditioners known as Neumann-Neumann methods which were originally developed to solve elliptic problems [21, 44, 116]. While all of the methods discussed have defined local problems based on blocks of the fully assembled discrete system, Neumann-Neumann methods exploit the finite element residual assembly to define the local problem. While the original Neumann-Neumann methods lacked a coarse space, Mandel introduced a coarse space leading to the Balancing Domain Decomposition (BDD) method [73, 74].

The BDD method was later shown to be closely related to the Finite Element Tearing and Interconnecting (FETI) method [54, 58]. FETI methods are among the most widely used and well tested methods for structural mechanics problems. For example Bhardwaj et al. used FETI methods to solve structural mechanics problems on up to 1000 processors [18]. The most advanced of the FETI and Neumann-Neumann class of methods are the dual-primal FETI (FETI-DP) [53, 80] and the Balancing Domain Decomposition by Constraints (BDDC) method [47, 75]. Like FETI and BDD, FETI-DP and BDDC methods are closely related and have essentially the same eigenvalue spectra [71, 76]. The analysis of these preconditioners have been extended to the case where inexact local solvers are used in [48, 67, 72]. Additionally, several authors have presented multi-level versions of the BDDC method [78, 79, 120]. An adaptive method for adding primal degrees of freedom to ensure rapid convergence has also been presented[77]. Practical implementations of the FETI-DP method has been used to solve structural mechanics problems on up to 3000 processors [19, 104].

While originally developed for elliptic problems, Achdou et al. modified the Neumann-Neumann interface condition to Robin-Robin interface conditions for a convection-diffusion problem, ensuring that the local bilinear forms were coercive [1]. A Fourier analysis, on a partitioning of the domain into strips normal to the stream-wise direction, showed that in the convective limit, the resulting algorithm converges in a number of iterations equal to half the number of subdomains in the stream-wise direction. The Robin-Robin interface conditions have been used along with a FETI method to solve linear convection-diffusion problems by Toselli [117]. Similarly, Tu and Li used the Robin-Robin interface condition to extend the BDDC method to convection-diffusion problems [121].

Neumann-Neumann and FETI methods have in general not been used for large scale CFD simulations, however recent work is beginning to make these methods available to the systems of equations for compressible flows. Dolean and collaborators have extended the Robin-Robin interface condition to the isentropic Euler equations using a Smith factorization [50, 51]. Yano and Darmofal also provided a generalization of the Robin-Robin interface condition to the Euler equations based on entropy symmetrization theory [123, 124]. However, detailed analysis of the performance of this algorithm has yet to be performed.

1.3 Thesis Overview

1.3.1 Objectives

The objective of this work is to develop a scalable high-fidelity solver for higher-order discretizations of convection-dominated flows. In order to meet this objective, this work presents a solution strategy based on an implicit Newton-Krylov method using domain decomposition preconditioners.

1.3.2 Thesis Contributions

The contributions of this thesis are the following:

- An additive Schwarz preconditioner with a coarse space based on harmonic extensions is developed for higher-order hybridizable discontinuous Galerkin discretizations. The coarse space correction is shown to improve the convergence relative to a single level additive Schwarz preconditioner for convection dominated flows solved on anisotropic meshes.

- A Balancing Domain Decomposition by Constraints (BDDC) preconditioner is developed for higher-order hybridizable discontinuous Galerkin (HDG) discretizations. The BDDC method is proven to converge at a rate independent of the number of subdomains and only weakly dependent on the solution order and the number of elements per subdomain, for a second-order elliptic problem. The BDDC preconditioner is extended to the solution of convection-dominated problems using a Robin-Robin interface condition. Numerical results show that the BDDC preconditioner reduces the number of local linear solves required to converge relative the additive Schwarz preconditioner with or without coarse space for convection-dominated problems solved on anisotropic meshes.

- An inexact BDDC preconditioner is developed based on incomplete factorizations and a p-multigrid type coarse grid correction. It is shown that the incomplete factorization of the singular linear systems corresponding to local Neumann problems results in a non-singular preconditioner. Numerical results show that the inexact BDDC preconditioner converges in a similar number of iterations as the exact BDDC method, with significantly reduced CPU time.

- An analysis is performed to assess the effect of far-field boundary conditions on the convergence of domain decomposition methods for the compressible Euler equations. It is shown that, even in the one-dimensional case, applying reflecting boundary conditions leads to exponential convergence (as opposed to convergence in a finite number of iterations).

- The BDDC preconditioner is extended to the solution of the compressible Euler equations using optimized interface conditions. Numerical results show that the optimized interface conditions significantly improve the performance of the BDDC preconditioner relative to the standard Robin-Robin interface condition.

- A restricted overlapping Schwarz preconditioner is developed for a higher-order discontinuous Galerkin (DG) discretization of the compressible Euler and Navier-Stokes equations. It is shown that using an overlap of consisting of only a single layer of elements significantly improves relative to a non-overlapping preconditioner.

- A BDDC preconditioner is developed for a large class of higher-order discontinuous Galerkin discretizations. The BDDC method is proven to converge at a rate independent of the number of subdomains and only weakly dependent on the solution order or the number of elements per subdomain, for a second-order elliptic problem.

The thesis is organized as follows. Chapter 2 presents the HDG discretization. Chapter 3 presents the domain decomposition and develops the additive Schwarz preconditioner. Chapter 4 introduces the BDDC preconditioner, while Chapter 5 presents an inexact variant of the BDDC preconditioner. Chapter 6 provides an analysis of the performance of the domain decomposition preconditioners for the Euler and Navier-Stokes systems of equations. Chapter 7 presents numerical results for several fundamental aerodynamic flows. Finally, Chapter 8 gives conclusions and recommendations for future work.

Chapter 2

Discretization and Governing Equations

2.1 HDG Discretization

The hybridizable discontinuous Galerkin discretization is presented for a general system of conservation equations. Let $u(x,t) : \mathbb{R}^d \times \mathbb{R}^+ \rightarrow \mathbb{R}^m$ be the vector of m-state variables in d-dimensions. A general conservation law in a physical domain, $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, is given by:

$$u_{k,t} + \left(F_{ik}(u) + G_{ik}(u, u_{,x})\right)_{,x_i} = f_k(u, u_{,x}, x), \tag{2.1}$$

where $k \in \{1, \ldots, m\}$ is the component index of the governing equations, $i \in \{1, \ldots, d\}$ is the spatial index, $(\cdot)_{,t}$ is the temporal derivative, and $(\cdot)_{,x_i}$ is the spatial derivative with respect to x_i. $F(u) : \mathbb{R}^m \rightarrow \mathbb{R}^{m \times d}$ and $G(u, u_{,x}) : \mathbb{R}^m \times \mathbb{R}^{m \times d} \rightarrow \mathbb{R}^{m \times d}$ are the viscous and inviscid fluxes, respectively. (2.1) may be rewritten as the following first order system of equations:

$$\begin{aligned} q_{ik} - u_{k,x_i} &= 0 &(2.2)\\ u_{k,t} + \left(F_{ik}(u) + G_{ik}(u, q)\right)_{,x_i} &= f_k(u, q, x). &(2.3) \end{aligned}$$

The HDG discretization is obtained from a weak form of (2.2)-(2.3). Let the tessellation $\mathcal{T}_h$ be a partition of Ω into triangles or quadrilaterals (if $d = 2$) or tetrahedra or hexahedra (if $d = 3$). Define $\mathcal{E}$ to be the union of edges (if $d = 2$) or faces (if $d = 3$) of elements, κ. Additionally, define $\mathcal{E}^{\text{Int}} \subset \mathcal{E}$ and $\mathcal{E}^{\partial} \subset \mathcal{E}$ to be the set of interior and boundary edges, respectively. Any edge $e \in \mathcal{E}^{\text{Int}}$ is shared by two adjacent elements κ^+ and κ^- with corresponding outward pointing normal vectors n^+ and n^-.

Let $\mathcal{P}^p(D)$ denote the space of polynomials of order at most p on D. Given the tessellation $\mathcal{T}_h$, define the following finite element spaces:

$$\mathbf{V}_h^p := \{v \in \left(\mathbf{L}^2(\Omega)\right)^{m\times d} : v|_\kappa \in (\mathcal{P}^p(\kappa))^{m\times d} \quad \forall \kappa \in \mathcal{T}\} \tag{2.4}$$

$$W_h^p := \{w \in \left(\mathbf{L}^2(\Omega)\right)^{m} : w|_\kappa \in (\mathcal{P}^p(\kappa))^{m} \quad \forall \kappa \in \mathcal{T}\} \tag{2.5}$$

$$M_h^p := \{\mu \in \left(\mathbf{L}^2(\mathcal{E})\right)^{m} : \mu|_e \in (\mathcal{P}^p(e))^{m} \quad \forall e \in \mathcal{E}\}. \tag{2.6}$$

Consider the steady state solution to (2.2)-(2.3). The weak form is given by: Find $(q, u) \in \mathbf{V}_h^p \times W_h^p$ such that for all $\kappa \in \mathcal{T}_h$,

$$(q_{ik}, v_{ik})_\kappa + (u_k, v_{ik,x_i})_\kappa - \langle \hat{u}_k, v_{ik} n_i \rangle_{\partial\kappa} = 0 \qquad \forall v \in \mathbf{V}_h^p \tag{2.7}$$

$$-(F_{ik}(u) + G_{ik}(u,q), w_{k,x_i})_\kappa + \left\langle (\widehat{F}_{ik} + \widehat{G}_{ik}) n_i, w_k \right\rangle_{\partial\kappa} = (f_k, w_k)_\kappa \quad \forall w \in W_h^p, \tag{2.8}$$

where $(w, v)_\kappa := \int_\kappa wv$ and $\langle w, v\rangle_{\partial\kappa} := \int_{\partial\kappa} wv$. In the case of unsteady problems, or steady problems driven to steady-state using pseudo-transient time continuation, (2.8) also includes temporal terms appropriate to the time stepping scheme used. In (2.8), $(\widehat{F}_{ik} + \widehat{G}_{ik})n_i$ is the numerical flux which approximates $(F_{ik}(u) + G_{ik}(u,q))n_i$ on $\partial\kappa$, while $\hat{u} \in M_h^p$ approximates the trace of u on $\mathcal{E}$. The numerical flux takes on the form:

$$(\widehat{F}_{ik} + \widehat{G}_{ik})n_i = (F_{ik}(\hat{u}) + G_{ik}(\hat{u}, q))n_i + S_{kl}(\hat{u}, n)(u_l - \hat{u}_l) \qquad \text{on } \partial\kappa, \tag{2.9}$$

where $S(\hat{u}, n)$ is a local stabilization matrix. The local stabilization matrix has the form:

$$S(\hat{u}, n) = |A_n| + S_{\text{Diff}}, \tag{2.10}$$

where $A_n = \frac{\partial F_i(\hat{u}) n_i}{\partial \hat{u}}$, while S_{Diff} is a diagonal matrix with diagonal values given by the coefficients of the diffusive terms [39].

Elements in the HDG discretization are coupled by ensuring that the numerical flux is continuous across faces in $\mathcal{E}^{\mathrm{Int}}$. Specifically, the HDG discretization enforces that the jump in the numerical flux, $(\widehat{F}_{ik} + \widehat{G}_{ik})^+ n_i^+ + (\widehat{F}_{ik} + \widehat{G}_{ik})^- n_i^-$, between elements κ^+ and κ^- is orthogonal to all functions in $M_h^p(\mathcal{E}^{\mathrm{Int}})$. Summing over all element gives the HDG discretization: Find $(q, u, \hat{u}) \in \mathbf{V}_h^p \times W_h^p \times M_h^p$ such that

$$\begin{aligned}
(q_{ik}, v_{ik})_{\mathcal{T}} + (u_k, v_{ik,x_i})_{\mathcal{T}} - \langle \hat{u}_k, v_{ik} n_i \rangle_{\partial\mathcal{T}} &= 0 \qquad \forall v \in \mathbf{V}_h^p &(2.11)\\
-(F_{ik}(u) + G_{ik}(u,q), w_{k,x_i})_{\mathcal{T}} + \left\langle (\widehat{F}_{ik} + \widehat{G}_{ik}) n_i, w_k \right\rangle_{\partial\mathcal{T}} &= (f_k, w_k)_{\mathcal{T}} \quad \forall w \in W_h^p &(2.12)\\
-\left\langle (\widehat{F}_{ik} + \widehat{G}_{ik}) n_i, \mu_k \right\rangle_{\partial\mathcal{T}\backslash\partial\Omega} &= 0 \qquad \forall \mu \in M_{h,i}^p &(2.13)\\
\left\langle \widehat{B}_k, \mu_k \right\rangle_{\partial\Omega} &= 0 \qquad \forall \mu \in M_{h,\partial}^p, &(2.14)
\end{aligned}$$

where $(\cdot,\cdot)_{\mathcal{T}} := \sum_{\kappa\in\mathcal{T}} (\cdot,\cdot)_\kappa$, $\langle\cdot,\cdot\rangle_{\partial\mathcal{T}} := \sum_{\kappa\in\mathcal{T}} \langle\cdot,\cdot\rangle_{\partial\kappa}$ and $\langle\cdot,\cdot\rangle_{\partial\mathcal{T}\backslash\partial\Omega} := \sum_{\kappa\in\mathcal{T}} \langle\cdot,\cdot\rangle_{\partial\kappa\backslash\partial\Omega}$, while $M_{h,i}^p$ and $M_{h,\partial}^p$ are subsets of M_h^p corresponding to edge functions interior to Ω and on $\partial\Omega$, respectively. $\hat{B}$ is a boundary term used to enforce the desired boundary conditions on $\partial\Omega$. The specific form of $\hat{B}$ for different types of boundary conditions is discussed in Section 2.3. Specifying a basis for $\mathbf{V}_h^p$, W_h^p and M_h^p, (2.11) - (2.13) leads to a non-linear system of equations for the coefficients of the basis functions. Applying Newton's method results in a discrete linear system at each Newton iteration of the form:

$$\begin{bmatrix} A_{qq} & A_{qu} & A_{q\lambda} & A_{q\lambda_b} \\ A_{uq} & A_{uu} & A_{u\lambda} & A_{u\lambda_b} \\ A_{\lambda q} & A_{\lambda u} & A_{\lambda\lambda} & A_{\lambda\lambda_b} \\ A_{\lambda_b q} & A_{\lambda_b u} & A_{\lambda_b\lambda_b} & A_{\lambda_b\lambda_b} \end{bmatrix} \begin{bmatrix} \Delta q \\ \Delta u \\ \lambda \\ \lambda_b \end{bmatrix} = \begin{bmatrix} b_q \\ b_u \\ b_\lambda \\ b_{\lambda_b} \end{bmatrix}, \qquad (2.15)$$

where Δq, Δu, λ, λ_b denote the vector of updates for the coefficients of basis functions corresponding to q, u and $\hat{u}$ on $e \in \mathcal{E}^{\mathrm{Int}}$ and $e \in \mathcal{E}^{\partial}$ respectively. The system (2.15) has an element-wise block diagonal form such that Δq, Δu and λ_b in (2.15) may be eliminated element by element to get the following system:

$$\hat{A}\lambda = b, \qquad (2.16)$$

where

$$\hat{A} = A_{\lambda\lambda} - \begin{bmatrix} A_{\lambda q} & A_{\lambda u} & A_{\lambda\lambda_b} \end{bmatrix} \begin{bmatrix} A_{qq} & A_{qu} & A_{q\lambda_b} \\ A_{uq} & A_{uu} & A_{u\lambda_b} \\ A_{\lambda_b q} & A_{\lambda_b u} & A_{\lambda_b\lambda_b} \end{bmatrix}^{-1} \begin{bmatrix} A_{q\lambda} \\ A_{u\lambda} \\ A_{\lambda_b\lambda} \end{bmatrix} \tag{2.17}$$

$$b = b_{\lambda} - \begin{bmatrix} A_{\lambda q} & A_{\lambda u} & A_{\lambda\lambda_b} \end{bmatrix} \begin{bmatrix} A_{qq} & A_{qu} & A_{q\lambda_b} \\ A_{uq} & A_{uu} & A_{u\lambda_b} \\ A_{\lambda_b q} & A_{\lambda_b u} & A_{\lambda_b\lambda_b} \end{bmatrix}^{-1} \begin{bmatrix} b_q \\ b_u \\ b_{\lambda_b} \end{bmatrix}. \tag{2.18}$$

The superscript in $\hat{A}$ is used to denote the globally assembled finite element matrix consistent with the notation in Chapter 4. Alternatively, $\lambda = \Delta\hat{u}$ may be viewed as the solution of linearized form:

$$a(\lambda, \mu) = b(\mu) \qquad \forall \mu \in M^p_{h,i}, \tag{2.19}$$

where $a(\lambda, \mu)$ and $b(\mu)$ are given by:

$$a(\lambda, \mu) = \sum_{\kappa \in \mathcal{T}} a_\kappa(\lambda, \mu) = \sum_{\kappa \in \mathcal{T}} - \left\langle (\Delta\widehat{F}^{\lambda,0}_{ik} + \Delta\widehat{G}^{\lambda,0}_{ik}) n_i, \mu \right\rangle_{\partial\kappa \backslash \partial\Omega} \tag{2.20}$$

$$b(\mu) = \sum_{\kappa \in \mathcal{T}} b_\kappa(\mu) = \sum_{\kappa \in \mathcal{T}} \left\langle (\Delta\widehat{F}^{0,R}_{ik} + \Delta\widehat{G}^{0,R}_{ik}) n_i, \mu \right\rangle_{\partial\kappa \backslash \partial\Omega}. \tag{2.21}$$

Here, $(\Delta\widehat{F}^{\lambda,0}_{ik} + \Delta\widehat{G}^{\lambda,0}_{ik}) n_i$ is the change in numerical flux obtained by solving the linearized problem about $(\bar{q}, \bar{u}, \bar{\hat{u}})$ corresponding to (2.7) - (2.8) with $\Delta\hat{u} = \lambda$ and zero right-hand side. Similarly, $(\Delta\widehat{F}^{0,R}_{ik} + \Delta\widehat{G}^{0,R}_{ik}) n_i$ is the change in numerical flux obtained by solving the linearized problem with $\Delta\hat{u} = 0$ and right-hand sides corresponding to the residual of (2.7) - (2.8) at the linearization point.

2.2 Compressible Navier-Stokes Equations

The compressible Euler and Navier-Stokes equations are systems of non-linear conservation equations of the form (2.1). In particular, the Euler equations are obtained from the Navier-Stokes equations by setting the viscous flux vector to zero and omitting (2.11) for the gradient. The conservative state vector is given by $u = [\rho, \rho v_i, \rho E]^T$, where ρ is the density,

v_i the velocity in direction i, and E the total internal energy. The inviscid flux vector is given by:

$$F_i(u) = \begin{bmatrix} \rho v_i \\ \rho v_i v_j + \delta_{ij} p \\ \rho v_i H \end{bmatrix}, \tag{2.22}$$

where p is the static pressure, $H = E + \frac{p}{\rho}$ is the total enthalpy, and δ_{ij} is the Kronecker delta:

$$\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & \text{otherwise.} \end{cases} \tag{2.23}$$

The pressure is given by:

$$p = (\gamma - 1)\rho\left(E - \frac{1}{2}\rho v_i v_i\right), \tag{2.24}$$

where γ is the specific heat ratio ($\gamma = 1.4$ for air). The viscous flux vector is given by:

$$G_i(u, u_{,x}) = -\begin{bmatrix} 0 \\ \tau_{ij} \\ v_j \tau_{ij} + \kappa_T T_{,x_i} \end{bmatrix}, \tag{2.25}$$

where $\boldsymbol{\tau}$ is the shear stress tensor, κ_T is the thermal conductivity, $T = p/\rho R$ is the temperature, and R is the gas constant. The shear stress tensor, $\boldsymbol{\tau}$, is given by:

$$\tau_{ij} = \mu\left(v_{i,x_j} + v_{j,x_i}\right) - \delta_{ij}\lambda v_{k,x_k}, \tag{2.26}$$

where μ is the dynamic viscosity and $\lambda = \frac{2}{3}\mu$ is the bulk viscosity coefficient. The dynamic viscosity is given by Sutherland's Law:

$$\mu = \mu_{ref}\left(\frac{T}{T_{ref}}\right)^{\frac{3}{2}} \frac{T_{ref} + T_s}{T + T_s}. \tag{2.27}$$

The thermal conductivity is a function of the dynamic viscosity:

$$\kappa_T = \frac{\gamma \mu R}{(\gamma - 1)Pr}, \tag{2.28}$$

where Pr is the Prandtl number. For air, $T_{ref} = 288°$ K, $T_s = 110.4°$ K and $Pr = 0.71$.

2.3 Boundary Conditions

This section gives the exact form of the boundary term $\hat{B}$ used to define various boundary conditions for the compressible Euler and Navier-Stokes equations.

Flow Tangency BC

A slip boundary condition is enforced on a symmetry plane or at a solid surface in inviscid flow. The boundary condition may be written as:

$$\hat{B} = \begin{bmatrix} \rho - \hat{\rho} \\ \rho v_i - (\rho v_k n_k) n_i - \widehat{\rho v_i} \\ \rho E - \widehat{\rho E} \end{bmatrix} \tag{2.29}$$

where $\rho v_k n_k$ is the normal component of the velocity. Thus setting $\hat{B} = 0$ weakly enforces $\hat{\rho} = \rho$, $\widehat{\rho E} = \rho E$, and $\widehat{\rho v_i}$ has zero component in the normal direction (since $\rho v_i - (\rho v_k n_k) n_i$ is the interior momentum with the normal component removed).

Adiabatic No-slip BC

At a solid surface for viscous flow an adiabatic no slip boundary condition is set by enforcing zero velocity and heat flux. Namely, the boundary condition is written as:

$$\hat{B} = \begin{bmatrix} (\hat{F}_{i1} + \hat{G}_{i1}) \\ \widehat{\rho v_i} \\ (\hat{F}_{im} + \hat{G}_{im}) \end{bmatrix} \tag{2.30}$$

The first condition enforce zero mass flux through the boundary, the second condition ensures zero momentum at the surface, while the third condition ensures the heat flux into the domain is zero.

Isothermal No-slip BC

In order to set an isothermal no slip boundary condition at a solid surface for viscous flow, the boundary condition is similar to the adiabatic no slip condition except the last component of the boundary condition is set to $T_w - T(\hat{u})$. T_w is the desired surface temperature and $T(\hat{u})$ is the temperature computed as a function of $\hat{u}$.

Characteristic Inflow/Outflow BCs

At an inflow/outflow boundary, a free-stream condition, u_∞, is specified and the boundary condition is given by:

$$\hat{B} \;=\; \left[(\hat{F}_i + \hat{G}_i)n_i\right] - \left[F_i(\hat{u})n_i + A_n^-(u_\infty - u) + (G_i n_i)_\infty\right]. \tag{2.31}$$

where $A_n = \partial(F_i(\hat{u})n_i)/\partial(\hat{u})$, with $A_n^\pm = (A_n \pm |A_n|)/2$. Here $(G_i n_i)_\infty$ is a specified normal viscous flux at the boundary, usually assumed to be zero. This choice of $\hat{B}$ ensures that the mathematically correct number of boundary conditions are specified at inflow and outflow boundary conditions. Namely, for two-dimensional subsonic inviscid flow three inflow and one outflow boundary conditions are set, while for viscous flow four inflow and three outflow conditions are set [62]. In the case of inviscid flow this boundary condition simplifies to the expression given by Peraire et al. [100]:

$$\hat{B} \;=\; A_n^+(u - \hat{u}) - A_n^-(u_\infty - \hat{u}) \tag{2.32}$$

Other Inflow/Outflow BCs

It is often convenient to specify inflow and outflow BCs without complete knowledge of the boundary state. For example total pressure, total temperature and angle of attack may be specified at an inflow boundary or a desired static pressure may be specified at an outflow boundary. For these types of boundary conditions, a boundary state u_∞ is reconstructed as a function of the interior state u and the desired boundary condition. The boundary flux $\hat{B}$ is then set in the same manner as the characteristic the BCs given above.

2.4 Solution Method

While the focus of this work is primarily to obtain steady-state solutions, pseudo-transient continuation is used to improve the transient behaviour of the solver. Specifically, a local time-stepping procedure is employed whereby the time step for each element Δt_κ is set using a global CFL number:

$$\Delta t_\kappa \quad = \quad \text{CFL}\frac{h_{\min}}{\lambda_{\max}} \tag{2.33}$$

where $\lambda_{\max}$ is the largest characteristic speed in element κ, while $h_{\min}$ is the minimum altitude of κ. At each pseudo time-step a single Newton iteration is used to update $(q, u, \hat{u})$, while the global CFL number is incrementally increased as the solution approaches its steady state value.

Each Newton iteration requires the solution of a linear system of the form (2.15). As noted in Section 2.1, the HDG discretization allows for the degrees of freedom corresponding to $\Delta \boldsymbol{q}$ and Δu to be locally eliminated on each element. Thus, storage of the entire linear system is not necessary. In particular, the solution updates Δq^κ and Δu^κ for an element κ are given by:

$$\begin{aligned}
\begin{bmatrix} \Delta q^\kappa \\ \Delta u^\kappa \end{bmatrix} &= \begin{bmatrix} A^\kappa_{qq} & A^\kappa_{qu} \\ A^\kappa_{uq} & A^\kappa_{uu} \end{bmatrix}^{-1} \begin{bmatrix} b^\kappa_q \\ b^\kappa_u \end{bmatrix} - \begin{bmatrix} A^\kappa_{qq} & A^\kappa_{qu} \\ A^\kappa_{uq} & A^\kappa_{uu} \end{bmatrix}^{-1} \begin{bmatrix} A^\kappa_{q\lambda} \\ A^\kappa_{u\lambda} \end{bmatrix} \lambda^\kappa \\
&= \begin{bmatrix} \Delta q^\kappa_{\text{local}} \\ \Delta u^\kappa_{\text{local}} \end{bmatrix} + \begin{bmatrix} \Delta q^\kappa_{\text{global}} \\ \Delta u^\kappa_{\text{global}} \end{bmatrix} \lambda^\kappa
\end{aligned} \tag{2.34}$$

Thus, for each element only the local update vectors $\Delta q^\kappa_{\text{local}}$ and $\Delta u^\kappa_{\text{local}}$ and global update matrices $\Delta q^\kappa_{\text{global}}$ and $\Delta u^\kappa_{\text{global}}$ along with the globally coupled reduced system (2.16) need to be stored. For a linear problem $\Delta q^\kappa_{\text{local}}$ and $\Delta u^\kappa_{\text{local}}$ will be zero after the first iteration, even if the global reduced system is not solved exactly. Thus $\Delta q^\kappa_{\text{local}}$ and $\Delta u^\kappa_{\text{local}}$ provide a measure of the nonlinearity of the system and may be used as an indicator to drive an inner nonlinear iteration on an element-wise level.

The contribution to the global reduced system and residual due to element κ are given by:

$$A^{\kappa} = A^{\kappa}_{\lambda\lambda} - \begin{bmatrix} A^{\kappa}_{\lambda q} & A^{\kappa}_{\lambda u} \end{bmatrix} \begin{bmatrix} A^{\kappa}_{qq} & A^{\kappa}_{qu} \\ A^{\kappa}_{uq} & A^{\kappa}_{uu} \end{bmatrix}^{-1} \begin{bmatrix} A^{\kappa}_{q\lambda} \\ A^{\kappa}_{u\lambda} \end{bmatrix} \tag{2.35}$$

$$b^{\kappa} = b^{\kappa}_{\lambda} - \begin{bmatrix} A^{\kappa}_{\lambda q} & A^{\kappa}_{\lambda u} \end{bmatrix} \begin{bmatrix} A^{\kappa}_{qq} & A^{\kappa}_{qu} \\ A^{\kappa}_{uq} & A^{\kappa}_{uu} \end{bmatrix}^{-1} \begin{bmatrix} b^{\kappa}_{q} \\ b^{\kappa}_{u} \end{bmatrix} \tag{2.36}$$

In general, the linear system (2.16) is too large to solve directly, thus a preconditioned GMRES algorithm is used to iteratively obtain a solution. In particular, this work develops domain decomposition preconditioners for the efficient parallel solution of (2.16). Note, that the local solve and solution updates are trivially parallelized as these operations are completely independent on each element.

Chapter 3

Domain Decomposition and the Additive Schwarz Method

In this chapter an additive Schwarz method (ASM) is developed for the solution of the HDG discretization. Section 3.1 presents the domain decomposition and introduces notation. Section 3.2 presents a minimum overlapping additive Schwarz preconditioner with and without a coarse space for the parallel solution of the HDG discretization. In Section 3.3, the performance of the ASM preconditioners is assessed for several scalar PDE problems.

3.1 Domain Decomposition

Consider a partition of the domain Ω into subdomains Ω_i such that the closure of Ω, $\bar{\Omega}$, is given by $\bar{\Omega} = \cup_{i=1}^{N} \bar{\Omega}_i$. The subdomains Ω_i are disjoint regions of diameter $O(H)$, consisting of a union of elements in $\mathcal{T}$. Define $\mathcal{E}_i$ to be the union of edges and faces on Ω_i. The interface of subdomain Ω_i is defined as $\Gamma_i = \partial\Omega_i \backslash \partial\Omega$ while the collection of subdomain interfaces Γ is defined as $\Gamma = \cup_{i=1}^{N} \Gamma_i$. Figure 3-1 depicts graphically a sample grid partitioned into three non-overlapping subdomains.

Denote by $\hat{\Lambda}$ the space of finite element functions on edges in $\mathcal{E}$, while $\Lambda^{(i)}$ denotes finite element functions on edges in $\dot{\mathcal{E}}_i$. A Lagrange nodal basis is used to define $\Lambda^{(i)}$ and $\hat{\Lambda}$. Define restriction operator $R^{(i)} : \hat{\Lambda} \rightarrow \Lambda^{(i)}$ which maps global vectors on $\mathcal{E}$ to its local components on $\mathcal{E}_i$. Thus, the local solution vector $\lambda^{(i)} = R^{(i)}\lambda$ is defined by nodal values on $\mathcal{E}_i$. The

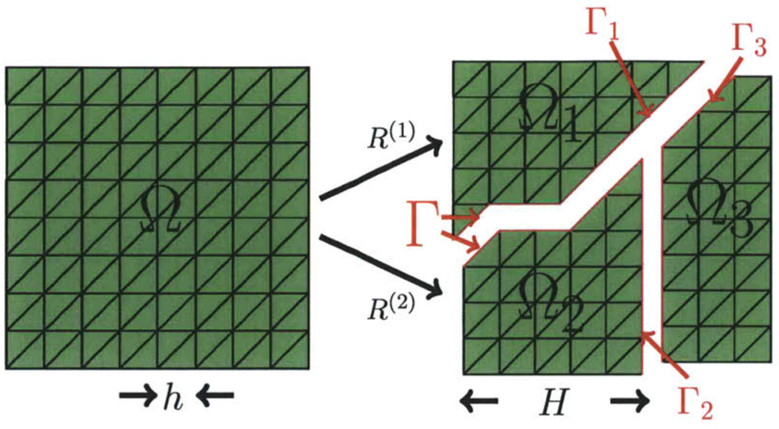


Figure 3-1: Sample grid and non-overlapping partition

local stiffness matrices, $A^{(i)}$, and load vectors, $b^{(i)}$, correspond to local forms:

$$a_i(\lambda, \mu) = \sum_{\kappa \in \Omega_i} a_\kappa(\lambda, \mu) \qquad\qquad b_i(\mu) = \sum_{\kappa \in \Omega_i} b_\kappa(\mu) \tag{3.1}$$

which may be computed independently on each subdomain. Note that the local bilinear form $a_i(\lambda, \mu)$ has natural boundary conditions on Γ_i. Thus the system, $A^{(i)}\lambda^{(i)} = b^{(i)}$, corresponds to a problem with Neumann boundary conditions on Γ_i.

The global stiffness matrix and load vector are obtained by assembling with respect to the interface degrees of freedom:

$$\hat{A} = \sum_{i=1}^{N} R^{(i)^T} A^{(i)} R^{(i)} \qquad \text{and} \qquad b = \sum_{i=1}^{N} R^{(i)^T} b^{(i)} \tag{3.2}$$

3.2 Additive Schwarz method

Consider the domain decomposition presented in Section 3.1. Denote by $\Omega_i^{'}$ the region obtained by extending Ω_i by a single element across each edge $e \in \Gamma_i$. Figure 3-2

Denote by $\hat{A}^{(i)} := R^{(i)} \hat{A} R^{(i)^T}$ the block extracted from $\hat{A}$ corresponding to edge degrees of freedom on interior to $\Omega_i^{'}$. While $A^{(i)}$ and $\hat{A}^{(i)}$ have the same size, $A^{(i)} \neq \hat{A}^{(i)}$. As noted previously $A^{(i)}$ corresponds to a finite element problem in Ω_i with Neumann boundary condition on Γ_i. On the other hand, $\hat{A}^{(i)}$ corresponds to a finite element problem on $\Omega_i^{'}$ with

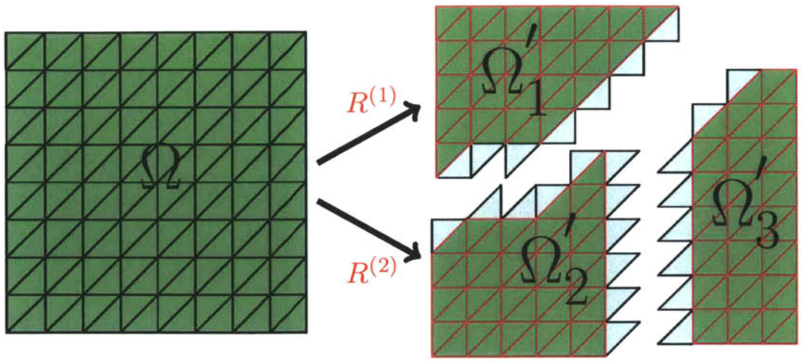


Figure 3-2: Overlapping partition

homogeneous Dirichlet boundary condition on $\partial\Omega_i'$. The additive Schwarz method involves the solution of N independent Dirichlet problems corresponding to each overlapping subdomain, Ω_i', which may be performed in parallel, by assigning a subdomain to each processor. Using the notation previously defined, the overlapping additive Schwarz preconditioner is written as:

$$M_{\text{ASM}}^{-1} = \sum_{i=1}^{N} R^{(i)^T} \hat{A}^{(i)^{-1}} R^{(i)} \tag{3.3}$$

This preconditioner is called minimum overlapping since if G_A is the adjacency graph corresponding to $\hat{A}$, then $R^{(i)}$ extracts nodes of G_A with overlap corresponding only to nodes on Γ. Figure 3-3 depicts the solution obtained by applying one iteration of the additive Schwarz method to solve a $p = 5$ discretization of the Poisson problem presented in Section 3.3 on the sample partitioned grid.

The basic form of the additive Schwarz method lacks a global correction and thus is not scalable for elliptic problems. Such a coarse space correction is necessary in order to control low frequency error modes that span multiple subdomains [118]. Additive Schwarz preconditioners with coarse spaces have been widely studied for continuous finite element discretizations of elliptic problems [29, 52], as well as for mixed finite elements [22], spectral elements [36], and discontinuous Galerkin discretizations [5–7, 55, 70]. In a typical analysis the condition number of the preconditioned system, $M_{\text{ASM}}^{-1}\hat{A}$, is bounded as $\kappa\left(M_{\text{ASM}}^{-1}\hat{A}\right) \leq C\left(1 + \frac{H}{\delta}\right)$, where H is the diameter of a subdomain Ω_i, while δ is the amount of overlap

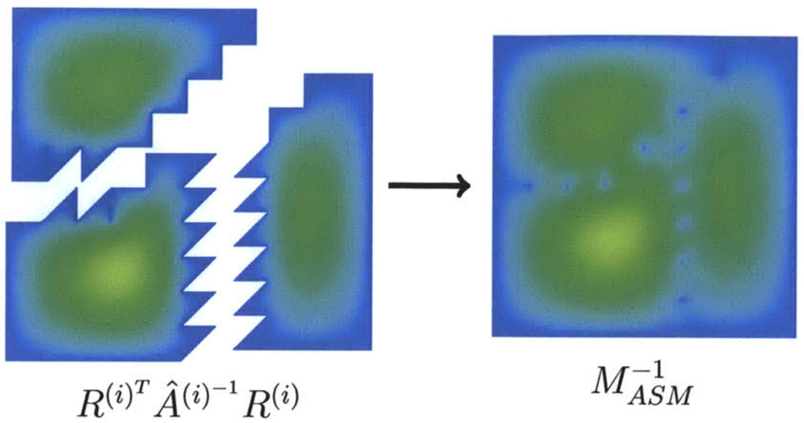


Figure 3-3: Additive Schwarz method applied to sample Poisson problem

and C is a constant independent of H or δ [29, 52]. The condition number does not depend directly upon H but only upon the factor $\frac{H}{\delta}$. If the overlap is such that $\delta \geq cH$ for some constant c, the subdomains are said to have "generous" overlap. With generous overlap, the condition number of the preconditioned system becomes independent of $\frac{1}{H}$ and $\frac{H}{h}$ and the method is both scalable and optimal. On the other hand, this work considers an additive Schwarz method with overlap, δ, proportional to the element size, h. Thus, the condition number bound has the form $\kappa \leq C\left(1+\frac{H}{h}\right)$, hence the preconditioner with coarse space is scalable, but not optimal.

In many cases, the coarse space for Schwarz methods is based on a discretization of the finite element problem on a coarser mesh [118]. In particular, it is often assumed that finite element mesh on which the solution is desired is obtained from successive refinements of a coarse mesh. This allows for very simple definition of restriction and prolongation operators. However, for problems involving complex geometries solved on unstructured meshes, an appropriate coarse problem may be much more difficult to define as the subdomains may have arbitrary shape. A possible approach involves generating a coarse problem by agglomerating elements of the fine mesh as in an agglomeration multigrid approach [81, 86, 90]. However, the particular choice of agglomeration strategy may significantly impact the performance of the solver [90]. Additionally, the communication pattern of the restriction and prolongation operators may be quite complicated [90].

In this work a coarse space problem is defined algebraically using a projection of the global system matrix. Denote by Λ_0 the coarse finite element space. The coarse space is defined by a set of coarse primal basis functions $\{\Phi_{\mathcal{E}_k}\}$ and coarse adjoint basis functions $\{\Phi^*_{\mathcal{E}_k}\}$. For now the exact form of the coarse basis functions is left undefined, except to note that each coarse basis function, $\Phi_{\mathcal{E}_k}$ or $\Phi^*_{\mathcal{E}_k}$, is associated with a subdomain interface $\mathcal{E}_k = \partial\Omega_i \cap \partial\Omega_j, i \neq j$. Define by Φ and Φ^* the matrices of coarse basis functions such that $\Phi : \Lambda_0 \longrightarrow \hat{\Lambda}$ defines prolongation operator from the coarse space to the fine space while ${\Phi^*}^T : \hat{\Lambda} \longrightarrow \Lambda_0$ defines a restriction operator to the coarse space. The coarse system, A_0, is given by the projection:

$$A_0 = {\Phi^*}^T \hat{A} \Phi \tag{3.4}$$

The additive Schwarz preconditioner with additive coarse grid correction is written as:

$$M^{-1}_{ASM_A} = \Phi A_0^{-1} {\Phi^*}^T + \sum_{i=1}^{N} {R^{(i)}}^T {\hat{A}^{(i)}}^{-1} R^{(i)} \tag{3.5}$$

A simple variant of this preconditioner is obtained by applying the coarse grid correction in a multiplicative manner [113]. Namely, this preconditioner involves two sequential steps: 1) the solution of the coarse grid problem followed by a corresponding update of the residual, 2) the solution of N independent subdomain problems. The additive Schwarz preconditioner with multiplicative coarse grid correction is written as:

$$M^{-1}_{ASM_M} = \Phi A_0^{-1} \Phi^{*T} + \sum_{i=1}^{N} {R^{(i)}}^T {\hat{A}^{(i)}}^{-1} R^{(i)} \left(I - \hat{A}\Phi A_0^{-1} \Phi^T\right) \tag{3.6}$$

It remains to define the coarse basis functions $\{\phi_{\mathcal{E}_k}\}$ and $\{\phi^*_{\mathcal{E}_k}\}$. As noted previously, each coarse basis function, $\phi_{\mathcal{E}_k}$ or $\phi^*_{\mathcal{E}_k}$, is associated with a subdomain interface $\mathcal{E}_k$. The value of the coarse basis functions on each edge e is given by:

$$\phi_{\mathcal{E}_k} = \begin{cases} \delta_{kk'} & e \in \mathcal{E}_{k'} \\ \mathcal{H}_i\left(\phi_{\mathcal{E}_k}|_{\partial\Omega_i}\right) & e \in \Omega_i, \forall i \end{cases} \qquad \phi^*_{\mathcal{E}_k} = \begin{cases} \delta_{kk'} & e \in \mathcal{E}_{k'} \\ \mathcal{H}^*_i\left(\phi_{\mathcal{E}_k}|_{\partial\Omega_i}\right) & e \in \Omega_i, \forall i \end{cases} \tag{3.7}$$

where $\delta_{kk'}$ is the Kronecker delta, while $\mathcal{H}_i$ and $\mathcal{H}^*_i$ are harmonic extension operators to the interior of Ω_i. In order to define the harmonic extension operators, the degrees of freedom

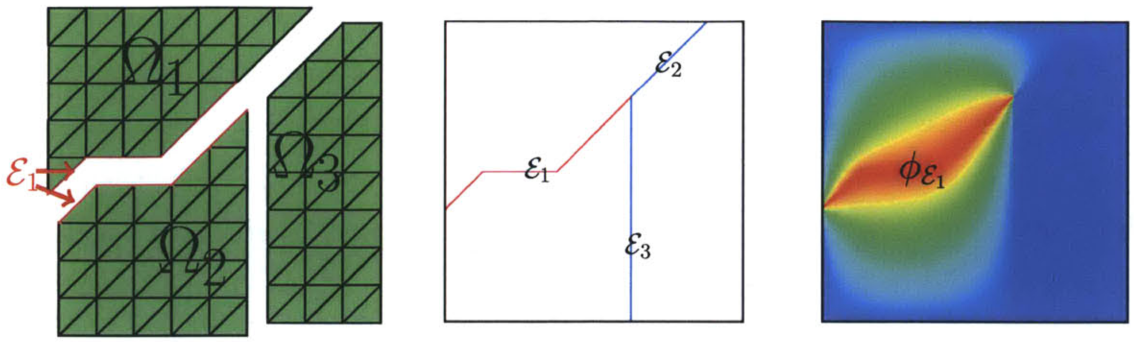


Figure 3-4: Coarse basis functions for additive Schwarz method

on each subdomain are partitioned into interior, $\lambda_I^{(i)} \in \Lambda_I^{(i)}$, and interface, $\lambda_\Gamma^{(i)} \in \Lambda_\Gamma^{(i)}$, parts. The subspace of $\Lambda^{(i)}$ consisting of functions which vanish on Γ_i is denoted by $\Lambda_I^{(i)}$, while $\Lambda_\Gamma^{(i)}$ denotes the space of edge function on Γ_i. The local solution vector, stiffness matrix and load vector are written as:

$$\lambda^{(i)} = \begin{bmatrix} \lambda_I^{(i)} \\ \lambda_\Gamma^{(i)} \end{bmatrix}, \qquad A^{(i)} = \begin{bmatrix} A_{II}^{(i)} & A_{I\Gamma}^{(i)} \\ A_{\Gamma I}^{(i)} & A_{\Gamma\Gamma}^{(i)} \end{bmatrix}, \qquad \text{and} \qquad b^{(i)} = \begin{bmatrix} b_I^{(i)} \\ b_\Gamma^{(i)} \end{bmatrix}, \tag{3.8}$$

The harmonic extension operators $\mathcal{H}_i$ and $\mathcal{H}_i^*$ which define mappings $\Lambda_\Gamma^{(i)} \to \Lambda^{(i)}$ are given by:

$$\mathcal{H}_i = \begin{bmatrix} -A_{II}^{(i)^{-1}} A_{I\Gamma}^{(i)} \\ I_\Gamma^{(i)} \end{bmatrix} \qquad \text{and} \qquad \mathcal{H}_i^* = \begin{bmatrix} -A_{II}^{(i)^{-T}} A_{\Gamma I}^{(i)^T} \\ I_\Gamma^{(i)} \end{bmatrix} \tag{3.9}$$

For elliptic problems, the harmonic extension operators are energy minimizing extensions of the interface values on Γ_i to all of Ω_i. The definition of the coarse space is purely algebraic, as the coarse basis functions is obtained from the global system matrix and its connectivity graph without relying on any geometry information of the underlying mesh. Figure 3-4 depicts a particular coarse basis function, while Figure 3-5 depicts the solution obtain by applying a single iteration of the ASM$_A$ preconditioner to the Poisson model problem.

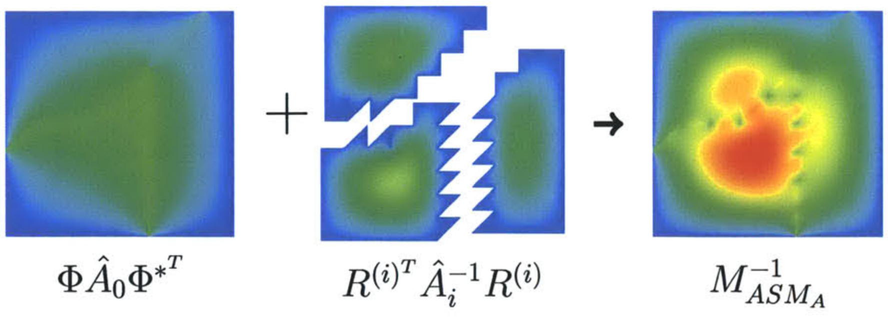


Figure 3-5: Additive Schwarz method with coarse space applied to sample Poisson problem

3.3 Numerical Results

In this section the performance of the ASM preconditioners are assessed for two linear scalar model problems in a square domain $\Omega = [0,1]^2 \in \mathbb{R}^2$. The first model problem is the Poisson problem:

$$-\Delta u = f \tag{3.10}$$

where $u(x,y)$ is the state while f is a source function so that the exact solution is given by:

$$u = \sin(\pi x)\sin(\pi y) \tag{3.11}$$

The second model problem is the advection-diffusion problem:

$$\nabla \cdot (\boldsymbol{c}u) - \mu \Delta u = f \qquad \text{in } \Omega \tag{3.12}$$

where $\boldsymbol{c} = (1,0)$ is the convective velocity, and μ is the diffusivity and f is set such that the exact solution is given by:

$$u = e^{-y/\sqrt{\mu \bar{x}}} \qquad \text{where } \bar{x} = x + 0.1 \tag{3.13}$$

Figures 3-6(a) and 3-6(b), respectively, plot the solution for the Poisson and advection-diffusion problems.

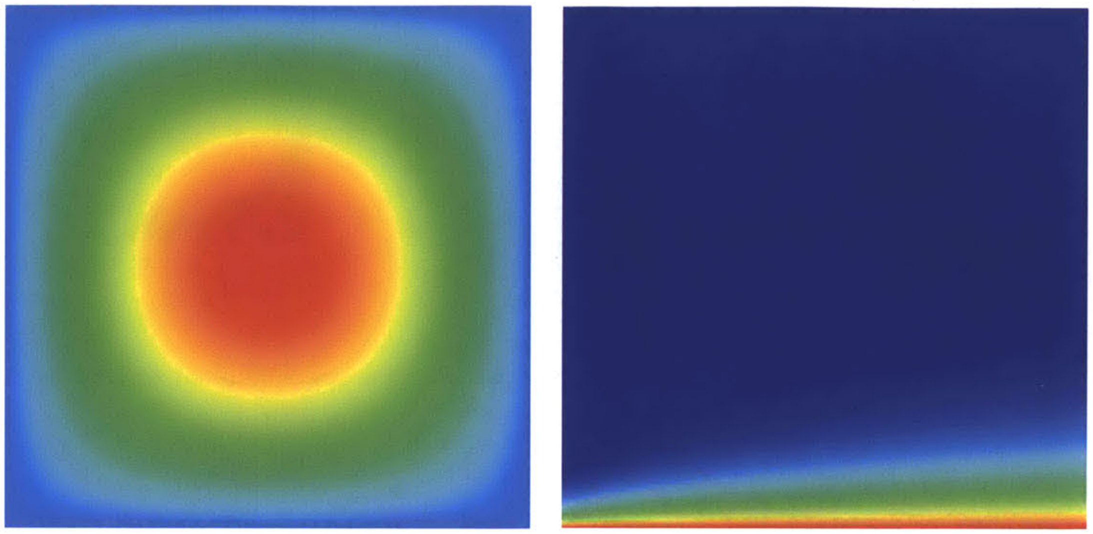
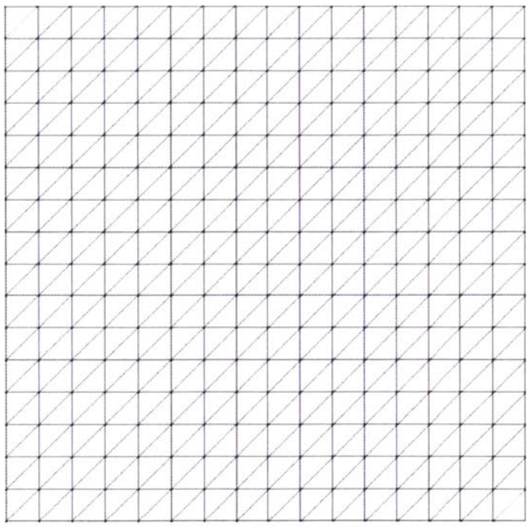

(a) Poisson (b) Advection-Diffusion ($\mu = 10^{-1}$)

Figure 3-6: Plot of solution for 2d scalar model problems

The relative performance of the different ASM algorithms is assessed in terms of the number of GMRES iterations required to converge. It is assumed that the cost of each GMRES iteration is dominated by the cost of the local solves, which are the same for all ASM variants presented.

In the first numerical experiment, the Poisson model problem is solved on an isotropic structured mesh, using a structured partition of the domain. The domain Ω is partitioned into N subdomains in a $\sqrt{N} \times \sqrt{N}$ structured pattern. Locally, each subdomain consists of n elements obtained by dividing Ω_i into squares of equal size and splitting each square into two triangular elements. Figure 3-7(a) plots the isotropic structured mesh.

The performance of the preconditioners are assessed for varying N and n for solutions with polynomial orders $p = 2$ and 5. Table 3.1 shows the number of GMRES iterations required to decrease the l_2-norm of the residual by a factor of 10^6. For a fixed number of elements per subdomain, $n = 128$, the performance of the ASM preconditioner without a coarse space degrades as the number of subdomains, N, increases. This behaviour is due to the lack of a coarse space able to control the lower frequency error modes. On the other hand for the additive (ASM$_A$) and multiplicative (ASM$_M$) preconditioners with coarse spaces, the number of iterations appears to be bounded as the number of subdomains increases. For


(a) Isotropic Mesh

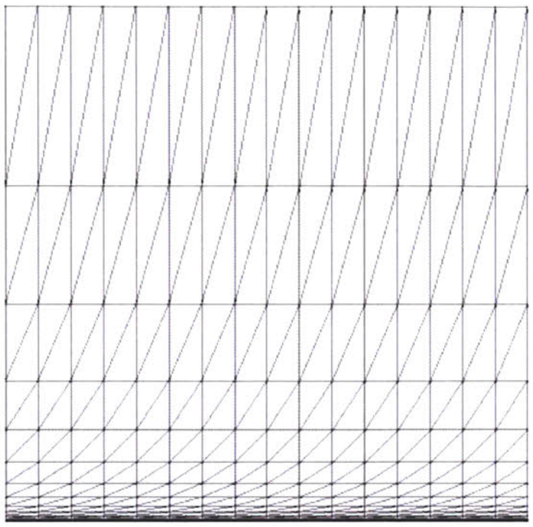

(b) Anisotropic Mesh

Figure 3-7: Plot of meshes used 2d scalar model problems

		$p=2$			$p=5$		
N	n	ASM	ASM$_A$	ASM$_M$	ASM	ASM$_A$	ASM$_M$
4	128	19	20	19	23	24	22
16	128	33	29	27	39	38	34
64	128	52	35	30	63	43	38
256	128	78	36	33	96	45	41
1024	128	107	37	35	139	46	43
64	8	30	22	16	36	26	21
64	32	40	27	22	47	32	28
64	128	52	35	30	63	43	38
64	512	66	48	42	78	60	52
64	2048	77	66	59	92	82	72

Table 3.1: Number of GMRES iterations for Poisson problem on isotropic structured mesh

a fixed number of subdomains $N = 64$, the performance of the ASM preconditioner degrades with increasing number of elements per subdomain, due to the non-optimality of this preconditioner in the case of small overlap. The ASM_M preconditioner with multiplicative coarse grid correction is, generally, slightly superior to the ASM_A preconditioner with additive coarse grid correction. However, the use of a multiplicative coarse grid correction does not alter the scaling of the preconditioner with N or n. Finally, note that the number of iterations appears only weakly dependent on the solution order for all three preconditioners.

In the second numerical experiment, the advection-diffusion model problem is solved on the same structured mesh as for the Poisson problem. The performance of the ASM preconditioners are assessed over a large range of diffusivity, μ, highlighting the difference between the diffusion- and convection-dominated limits. Tables 3.2 and 3.3 show the number of linear solves required to converge the l_2-norm of the residual by a factor of 10^6 for $\mu = 1$ and $\mu = 10^{-6}$, respectively.

In the diffusion-dominated limit, ($\mu = 1$), the behaviour of all preconditioners closely match that of the purely elliptic Poisson problem. Namely, a coarse space is necessary to ensure good scaling as the number of subdomains is increased, while the number of iterations grows with the number of elements per subdomain.

			$p = 2$			$p = 5$	
N	n	ASM	ASM_A	ASM_M	ASM	ASM_A	ASM_M
4	128	24	20	18	28	23	21
16	128	45	27	23	52	31	26
64	128	82	27	23	93	31	26
256	128	152	27	22	168	31	26
1024	128	239	27	23	261	31	26
64	8	43	19	14	49	22	16
64	32	61	21	18	69	24	20
64	128	82	27	23	93	31	26
64	512	110	34	29	122	38	33
64	2048	146	42	37	162	47	41

Table 3.2: Number of GMRES iterations for advection-diffusion boundary layer problem with $\mu = 1$ on isotropic structured mesh

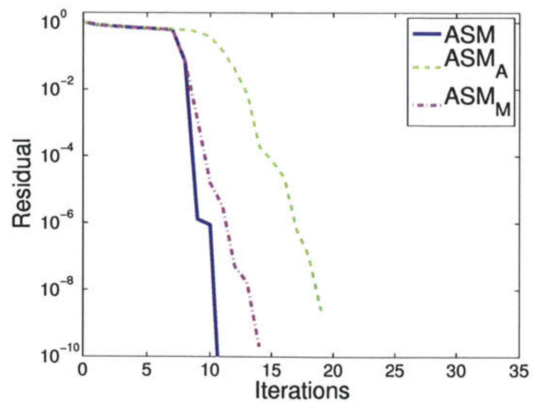
In the convection-dominated limit, $\mu = 10^{-6}$, all three ASM preconditioners converge in a small number of iterations proportional to the number of subdomains in the stream-wise direction ($\sqrt{N}$). For this test case, the boundary layer is restricted to a small region near the bottom of the domain. The elemental Peclet number, $Pe_h = \frac{ch}{\mu} \gg 1$ and there is insufficient resolution to accurately resolved the boundary layer profile. As the $Pe_h \gg 1$ the diffusion effects do not play a significant role in the convergence of the different preconditioners. The coarse space does not provides any benefit in this case as it is unable to resolve the high-frequency convective modes. In particular, a coarse space is not justified as the ASM method without coarse space converges in fewer iterations than the slightly more expensive ASM$_A$ and ASM$_M$ preconditioners.

		$p=2$			$p=5$		
N	n	ASM	ASM$_A$	ASM$_M$	ASM	ASM$_A$	ASM$_M$
4	128	3	6	5	3	6	5
16	128	5	10	7	5	11	8
64	128	10	17	12	11	18	13
256	128	19	30	20	19	29	21
1024	128	35	46	36	35	44	37
64	8	9	16	11	9	17	12
64	32	9	17	11	10	17	12
64	128	10	17	12	11	18	13
64	512	11	17	12	11	17	13
64	2048	10	17	13	11	18	14

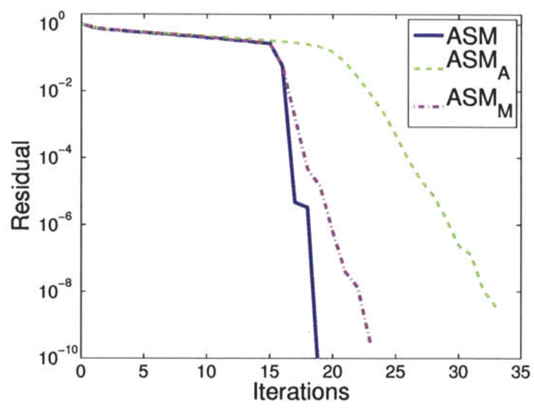
Table 3.3: Number of GMRES iterations for advection-diffusion boundary layer problem with $\mu = 10^{-6}$ on isotropic structured mesh

In order to better understand the performance of the preconditioner in the convection-dominated limit, Figure 3-8 plots the convergence history for $p = 2$, $n = 128$, and $N = 64$ or 256. The residual essentially does not decrease until iteration $\sqrt{N}$ at which point high frequency errors have essentially convected out of the domain and the residual drops rapidly.

While the results of Table 3.3 suggest that a coarse space may not be necessary for convection-dominated problems, the diffusive effects are masked by the lack of resolution in the boundary layer region. In practice, a significant portion of the mesh should be clustered near the bottom surface to ensure that the boundary layer region is fully resolved. In a third numerical experiment an anisotropic boundary layer mesh is employed, with uniform spacing in the x-direction and an exponential spacing in the y-direction. The aspect ratio of the elements at $y = 0$ is given by $AR = 1/\sqrt{Pe}$, where $Pe = |c|/\mu$ is the Peclet number.



(a) $N = 64$



(b) $N = 256$

Figure 3-8: GMRES convergence history for advection-diffusion boundary layer problem with $\mu = 10^{-6}$, $p = 2$ and $n = 128$ on isotropic structured mesh

Figure 3-7(b) gives an example of one of the anisotropic meshes. Table 3.4 shows the number of iterations required for the GMRES algorithm to converge by a factor of 10^6 for $\mu = 10^{-6}$. In this case there is sufficient resolution in the boundary layer region to resolve boundary layer profile near the bottom of the domain such that $Pe_h \approx 1$, where Pe_h is the Peclet number based on the element length scale in the cross-stream direction. Thus, the diffusive effects play a significant role in the convergence of the different preconditioners. Compared to Table 3.3, the performance of the ASM preconditioner without a coarse space is seen to degrade relative to the ASM$_A$ and ASM$_M$ preconditioners.

Table 3.5 shows the performance on both the isotropic and anisotropic meshes over a range of viscosities, for fixed N and n. Table 3.5 also gives the H_1-norm of the error in the solution using both sets of meshes. On the isotropic meshes, the relative performance of the ASM preconditioner without coarse space improves rapidly as the viscosity is reduced. However, on these meshes the boundary layer region is under-resolved. On the other hand, for the anisotropic meshes on which the boundary layer is resolved, a coarse space is important throughout the range of viscosities.

		$p=2$			$p=5$		
N	n	ASM	ASM$_A$	ASM$_M$	ASM	ASM$_A$	ASM$_M$
4	128	3	7	6	3	8	7
16	128	16	14	12	16	14	11
64	128	33	24	19	31	22	17
256	128	65	34	27	60	31	25
1024	128	126	40	37	118	37	35
64	8	18	17	13	18	16	12
64	32	25	19	15	23	18	14
64	128	33	24	19	31	22	17
64	512	46	32	25	42	29	23
64	2048	63	41	34	57	37	30

Table 3.4: Number of GMRES iterations for advection-diffusion boundary layer problem with $\mu = 10^{-6}$ on anisotropic structured mesh

		$p=2$				$p=5$			
	μ	$\frac{\|u-u^*\|_{H_1}}{\|u^*\|_{H_1}}$	ASM	ASM$_A$	ASM$_M$	$\frac{\|u-u^*\|_{H_1}}{\|u^*\|_{H_1}}$	ASM	ASM$_A$	ASM$_M$
Isotropic Mesh	1	1.2×10^{-6}	82	27	23	4.2×10^{-11}	93	31	26
Isotropic Mesh	10^{-1}	2.7×10^{-6}	80	30	25	2.5×10^{-11}	90	34	29
Isotropic Mesh	10^{-2}	2.8×10^{-5}	39	29	23	3.4×10^{-10}	43	33	27
Isotropic Mesh	10^{-3}	6.0×10^{-4}	20	22	17	5.2×10^{-8}	21	23	18
Isotropic Mesh	10^{-4}	1.2×10^{-2}	11	20	15	1.9×10^{-5}	11	21	15
Isotropic Mesh	10^{-5}	1.4×10^{-1}	11	19	13	3.4×10^{-3}	11	19	15
Isotropic Mesh	10^{-6}	3.1×10^{-1}	10	17	12	7.8×10^{-2}	11	18	13
Anisotropic Mesh	1	1.2×10^{-6}	82	27	23	4.2×10^{-11}	93	31	26
Anisotropic Mesh	10^{-1}	8.8×10^{-7}	85	31	24	2.3×10^{-11}	99	35	27
Anisotropic Mesh	10^{-2}	1.9×10^{-6}	68	30	22	1.6×10^{-11}	74	32	24
Anisotropic Mesh	10^{-3}	4.2×10^{-6}	51	29	23	1.7×10^{-11}	51	26	22
Anisotropic Mesh	10^{-4}	5.0×10^{-6}	41	28	22	2.1×10^{-11}	40	26	21
Anisotropic Mesh	10^{-5}	6.6×10^{-6}	36	26	20	2.6×10^{-11}	34	24	19
Anisotropic Mesh	10^{-6}	7.2×10^{-6}	33	24	19	3.8×10^{-11}	31	22	17

Table 3.5: Number of GMRES iterations on both isotropic and anisotropic meshes with $N = 64$, $n = 128$

Chapter 4

Balancing Domain Decomposition by Constraints

This chapter develops the BDDC preconditioner for the parallel solution of the HDG discretization. Section 4.1 derives a Schur complement problem involving degrees of freedom on the subdomain interfaces. Section 4.2 presents the BDDC preconditioner for the Schur complement problem, while Section 4.3 presents the BDDC algorithm as a preconditioner for the entire finite element problem. Section 4.4 provides a theoretical analysis of the BDDC preconditioner for second-order elliptic problems. Section 4.5 extends the BDDC preconditioner for the solution of advection-diffusion problems and presents the Robin-Robin interface condition. Finally, Section 4.6 presents numerical results using the BDDC preconditioner.

4.1 A Schur Complement Problem

In order to define the Schur complement system, the degrees of freedom on each subdomain are partitioned into interior and interface parts as in Section 3.2. The subspace of $\Lambda^{(i)}$ consisting of functions which vanish on Γ_i is denoted by $\Lambda_I^{(i)}$, while $\Lambda_\Gamma^{(i)}$ denotes the space of edge functions on Γ_i. Also define Λ_I, the space of functions which vanish on Γ, and $\hat{\Lambda}_\Gamma$ which corresponds to the space of edge functions associated with edges on Γ. Note that Λ_I is equal to the product of spaces $\Lambda_I^{(i)}$ (i.e. $\Lambda_I := \Pi_{i=1}^N \Lambda_I^{(i)}$), while in general $\hat{\Lambda}_\Gamma \subset \Lambda_\Gamma := \Pi_{i=1}^N \Lambda_\Gamma^{(i)}$.

The assembled system, (2.16), may be rewritten as:

$$\begin{bmatrix} A_{II} & A_{I\Gamma I} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} \lambda_I \\ \lambda_\Gamma \end{bmatrix} = \begin{bmatrix} b_I \\ b_\Gamma \end{bmatrix}. \tag{4.1}$$

where λ_I and λ_Γ corresponds to degrees of freedom associated with Λ_I and $\hat{\Lambda}_\Gamma$ respectively. Since the degrees of freedom associated with $\Lambda_I^{(i)}$ (and Λ_I) are local to a particular subdomain they may be locally eliminated. The local harmonic extension operators (3.9) are used to restrict (4.1) to a Schur complement system corresponding only to interface degrees of freedom. The local Schur complement matrices and load vectors are given by:

$$S_{\Gamma\Gamma}^{(i)} = \mathcal{H}_i^{*^T} A^{(i)} \mathcal{H}_i = A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)} A_{II}^{(i)^{-1}} A_{I\Gamma}^{(i)}, \tag{4.2}$$

$$g_\Gamma^{(i)} = \mathcal{H}_i^{*^T} b^{(i)} = b_\Gamma^{(i)} - A_{\Gamma I}^{(i)} A_{II}^{(i)^{-1}} b_I^{(i)}. \tag{4.3}$$

Define local operators $R_\Gamma^{(i)} : \hat{\Lambda}_\Gamma \rightarrow \Lambda_\Gamma^{(i)}$ which extract the local degrees of freedom on Γ_i from those on Γ. Additionally define a global operator $R_\Gamma : \hat{\Lambda}_\Gamma \rightarrow \Lambda_\Gamma$ which is formed by a direct assembly of $R_\Gamma^{(i)}$. The global Schur complement problem may then be written as:

$$\hat{S}_{\Gamma\Gamma} \lambda_\Gamma = g_\Gamma, \tag{4.4}$$

where

$$\hat{S}_{\Gamma\Gamma} = A_{\Gamma\Gamma} - A_{\Gamma I} A_{II}^{-1} A_{I\Gamma I} = \sum_{i=1}^{N} R_\Gamma^{(i)^T} S_{\Gamma\Gamma}^{(i)} R_\Gamma^{(i)}, \tag{4.5}$$

$$g_\Gamma = b_{\Gamma\Gamma} - A_{\Gamma I} A_{II}^{-1} b_{\Gamma I} = \sum_{i=1}^{N} R_\Gamma^{(i)^T} g_\Gamma^{(i)}. \tag{4.6}$$

Alternatively, the fully assembled Schur complement may be written as:

$$\hat{S}_{\Gamma\Gamma} = R_\Gamma^T S_{\Gamma\Gamma} R_\Gamma, \tag{4.7}$$

where $S_{\Gamma\Gamma}$ is the block diagonal matrix where each block corresponds to $S_{\Gamma\Gamma}^{(i)}$. Namely:

$$S_{\Gamma\Gamma} = \begin{bmatrix} S_{\Gamma\Gamma}^{(1)} & & \\ & \ddots & \\ & & S_{\Gamma\Gamma}^{(N)} \end{bmatrix}. \tag{4.8}$$

In practice, the Schur complement system need not be formed explicitly for a preconditioned GMRES algorithm. Instead, the action of $\hat{S}_{\Gamma\Gamma}$ on a vector $v_\Gamma \in \hat{\Lambda}_\Gamma$ is computed by solving local Dirichlet problems corresponding to $A_{II}^{(i)^{-1}}$ and summing the resulting vectors.

Finally, note that (4.4) is the discrete equivalent of a weak form: find $\lambda_\Gamma \in \hat{\Lambda}_\Gamma$ such that

$$s_\Gamma(\lambda_\Gamma, \mu_\Gamma) = g_\Gamma(\mu_\Gamma) \qquad \forall \mu_\Gamma \in \hat{\Lambda}_\Gamma, \tag{4.9}$$

where $s_\Gamma(\lambda_\Gamma, \mu_\Gamma)$ and $g_\Gamma(\mu_\Gamma)$ are given by

$$s_\Gamma(\lambda_\Gamma, \mu_\Gamma) = \sum_{i=1}^{N} s_i(\lambda_\Gamma, \mu_\Gamma) = \sum_{i=1}^{N} -\left\langle (\Delta\widehat{F}_{ik}^{\lambda_\Gamma,0} + \Delta\widehat{G}_{ik}^{\lambda_\Gamma,0})\boldsymbol{n}_i, \mu \right\rangle_{\Gamma_i}, \tag{4.10}$$

$$g_\Gamma(\mu_\Gamma) = \sum_{i=1}^{N} g_i(\mu_\Gamma) = \sum_{i=1}^{N} \left\langle (\Delta\widehat{F}_{ik}^{0} + \Delta\widehat{G}_{ik}^{0})\boldsymbol{n}_i, \mu \right\rangle_{\Gamma_i}, \tag{4.11}$$

where $(\Delta\widehat{F}_{ik}^{\lambda_\Gamma,0} + \Delta\widehat{G}_{ik}^{\lambda,0})\boldsymbol{n}_i$ is the change in numerical flux on Γ obtained by solving the linearized problem corresponding to (2.11) - (2.13) in Ω_i with zero right-hand side and boundary condition $\Delta u = \lambda_\Gamma$ on Γ_i. Similarly, $(\Delta\widehat{F}_{ik}^{0,R} + \Delta\widehat{G}_{ik}^{0,R})\boldsymbol{n}_i$ is the change in numerical flux obtained by solving the linearized problem in Ω_i with $\Delta\hat{u} = 0$ on Γ_i and right-hand sides corresponding to the residuals of (2.11) - (2.13).

4.2 The BDDC Preconditioner

In order to define the BDDC preconditioner for the Schur complement problem, (4.4), $\Lambda_\Gamma^{(i)}$ is partitioned into two orthogonal spaces $\Lambda_\Delta^{(i)}$ and $\Lambda_\Pi^{(i)}$. The dual space, $\Lambda_\Delta^{(i)}$, corresponds to the subset of function in $\Lambda_\Gamma^{(i)}$ constrained to have zero average on all subdomain interfaces,

$\mathcal{E}_k$. Namely, every function $\lambda^{(i)} \in \Lambda_\Delta^{(i)}$ satisfies:

$$\sum_{e \in \mathcal{E}_k} \int_e \lambda^{(i)} = 0 \qquad \forall \mathcal{E}_k \in \Gamma_i. \tag{4.12}$$

The primal space, $\Lambda_\Pi^{(i)}$, is spanned by basis functions $\{\psi_{\mathcal{E}_k}^{(i)}\}$ associated with each subdomain interface $\mathcal{E}_k$:

$$\sum_{e \in \mathcal{E}_j} \int_e \psi_{\mathcal{E}_k}^{(i)} = \delta_{jk}. \tag{4.13}$$

For conforming finite element methods, subdomain corner degrees of freedom are also included in the primal space. However, for the HDG discretization there are multiple degrees of freedom at subdomain corners, corresponding to the endpoints of each subdomain interface, $\mathcal{E}_k$ meeting at the corner. As each degree of freedom in $\hat{\Lambda}_\Gamma$ is associated with exactly two subdomains, and there is no need to add any corner degrees of freedom to the primal space for the HDG discretization.

Define the partially assembled space as:

$$\tilde{\Lambda}_\Gamma = \hat{\Lambda}_\Pi \oplus \left(\Pi_{i=1}^N \Lambda_\Delta^{(i)} \right), \tag{4.14}$$

where $\hat{\Lambda}_\Pi$ is the assembled global primal space, single valued on Γ, which is formed by assembling the local primal spaces, $\Lambda_\Pi^{(i)}$. The BDDC preconditioner may be viewed as solving a finite element problem on a partially assembled finite element space, $\tilde{\Lambda}_\Gamma$, in order to precondition the Schur complement problem whose solution lies in the fully assembled space $\hat{\Lambda}_\Gamma$ [71].

A key component of the BDDC preconditioner is an averaging operator which restricts functions from Λ_Γ to $\hat{\Lambda}_\Gamma$ [71]. A positive scaling factor $\delta_i^\dagger(\mathcal{E}_k)$ is defined for each interface $\mathcal{E}_k$ of subdomain Ω_i such that $\delta_i^\dagger(\mathcal{E}_k) + \delta_j^\dagger(\mathcal{E}_k) = 1$ where $\mathcal{E}_k = \partial\Omega_i \cap \partial\Omega_j$. For elliptic problems with discontinuous coefficients across subdomains, $\delta_i^\dagger(\mathcal{E}_k)$ should be weighted by the coefficients on subdomains Ω_i and Ω_j to obtain good performance [118]. However, in this work $\delta_i^\dagger(\mathcal{E}_k)$ is simply set to 1/2, unless otherwise specified. Define $D_\Gamma^{(i)} : \Lambda_\Gamma^{(i)} \rightarrow \Lambda_\Gamma^{(i)}$ as the diagonal matrix formed by setting the diagonal entries corresponding to each nodal degree of freedom on $\mathcal{E}_k$ to $\delta_i^\dagger(\mathcal{E}_k)$. Also define $D_\Gamma : \Lambda_\Gamma \to \Lambda_\Gamma$ as the diagonal matrix with diagonal blocks $D_\Gamma^{(i)}$. Finally, define $R_{D,\Gamma} : \hat{\Lambda}_\Gamma \to \Lambda_\Gamma$ as the product $R_{D,\Gamma} := D_\Gamma R_\Gamma$.

The BDDC preconditioner is given by:

$$M_{\Gamma_{\text{BDDC}}}^{-1} = R_{D,\Gamma}^T \tilde{S}_{\Gamma\Gamma}^{-1} R_{D,\Gamma}. \tag{4.15}$$

The operator $\tilde{S}_{\Gamma\Gamma}^{-1} : \tilde{\Lambda}_\Gamma \to \tilde{\Lambda}_\Gamma$ corresponds to the solution of the finite element problem on the partially assembled space. The action of $\tilde{S}_{\Gamma\Gamma}^{-1}$ may be efficiently computed as:

$$\tilde{S}_{\Gamma\Gamma}^{-1} = \Psi_\Gamma S_{\Pi\Pi}^{-1} \Psi_\Gamma^{*T} + \begin{bmatrix} \tilde{S}_{\Gamma\Gamma}^{(1)^{-1}} & & 0 \\ & \ddots & \\ 0 & & \tilde{S}_{\Gamma\Gamma}^{(N)^{-1}} \end{bmatrix}, \tag{4.16}$$

where Ψ_Γ, Ψ_Γ^*, $S_{\Pi\Pi}$ and $\tilde{S}_{\Gamma\Gamma}^{(i)^{-1}}$ are defined below. Denote by $\Psi_\Gamma^{(i)}$ and $\Psi_\Gamma^{*(i)} \in \Lambda_\Pi^{(i)}$ the local coarse basis functions defined by:

$$\begin{bmatrix} S_{\Gamma\Gamma}^{(i)} & B_\Gamma^{(i)^T} \\ B_\Gamma^{(i)} & 0 \end{bmatrix} \begin{bmatrix} \Psi_\Gamma^{(i)} \\ * \end{bmatrix} = \begin{bmatrix} 0 \\ I_\Pi^{(i)} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} S_{\Gamma\Gamma}^{(i)} & B_\Gamma^{(i)^T} \\ B_\Gamma^{(i)} & 0 \end{bmatrix}^T \begin{bmatrix} \Psi_\Gamma^{*(i)} \\ * \end{bmatrix} = \begin{bmatrix} 0 \\ I_\Pi^{(i)} \end{bmatrix}, \tag{4.17}$$

where each row of $B_\Gamma^{(i)^T}$ defines a constraint corresponding to a subdomain interface $\mathcal{E}_k \in \Gamma_i$, such that the second block row of (4.17) corresponds to satisfying the zero-mean condition, (4.12). In (4.17) and throughout this chapter, "$*$" denotes any term which is not used and, thus, not defined. Define local operators $R_\Pi^{(i)} : \hat{\Lambda}_\Pi \to \Lambda_\Pi^{(i)}$ which extract the local coarse degrees of freedom on Γ_i from those on Γ. The global coarse basis functions Ψ_Γ and Ψ_Γ^* are the matrices formed by concatenating the local coarse basis functions; namely:

$$\Psi_\Gamma = \begin{bmatrix} \Psi_\Gamma^{(1)} R_\Pi^{(1)} \\ \vdots \\ \Psi_\Gamma^{(N)} R_\Pi^{(N)} \end{bmatrix} \quad \text{and} \quad \Psi_\Gamma^* = \begin{bmatrix} \Psi_\Gamma^{*(1)} R_\Pi^{(1)} \\ \vdots \\ \Psi_\Gamma^{*(N)} R_\Pi^{(N)} \end{bmatrix}. \tag{4.18}$$

The global coarse system matrix $S_{\Pi\Pi}$ is given by:

$$S_{\Pi\Pi} = \Psi_\Gamma^{*^T} S_{\Gamma\Gamma} \Psi_\Gamma = \sum_{i=1}^N R_\Pi^{(i)^T} \Psi_\Gamma^{*(i)^T} S_{\Gamma\Gamma}^{(i)} \Psi_\Gamma^{(i)} R_\Pi^{(i)}. \tag{4.19}$$

The application of $\tilde{S}_{\Gamma\Gamma}^{(i)^{-1}}$ to a vector $v_\Gamma^{(i)} \in \Lambda_\Gamma^{(i)}$ corresponds to the solution of a constrained Neumann problem, defined by:

$$\begin{bmatrix} S_{\Gamma\Gamma}^{(i)} & B_\Gamma^{(i)^T} \\ B_\Gamma^{(i)} & 0 \end{bmatrix} \begin{bmatrix} \tilde{S}_{\Gamma\Gamma}^{(i)^{-1}} v_\Gamma^{(i)} \\ * \end{bmatrix} = \begin{bmatrix} v_\Gamma^{(i)} \\ 0 \end{bmatrix}. \tag{4.20}$$

As discussed previously, $S_{\Gamma\Gamma}^{(i)}$ is usually not formed explicitly. Instead the solution of the constrained Neumann problem is computed as

$$\begin{bmatrix} A_{II}^{(i)} & A_{I\Gamma}^{(i)} & 0 \\ A_{\Gamma I}^{(i)} & A_{\Gamma\Gamma}^{(i)} & B_\Gamma^{(i)^T} \\ 0 & B_\Gamma^{(i)} & 0 \end{bmatrix} \begin{bmatrix} * \\ \tilde{S}_{\Gamma\Gamma}^{(i)^{-1}} v_\Gamma^{(i)} \\ * \end{bmatrix} = \begin{bmatrix} 0 \\ v_\Gamma^{(i)} \\ 0 \end{bmatrix}. \tag{4.21}$$

The BDDC preconditioner may be applied efficiently in parallel as the constrained Neumann problems are solved independently on each subdomain. Thus, the only globally coupled solve corresponds to the primal correction, $\Psi^{*^T} S_{\Pi\Pi}^{-1} \Psi$, which has only a small number of degrees of freedom, on the order of the number of subdomains.

In Section 4.4 a condition number bound

$$\kappa\left(M_{\Gamma_{\text{BDDC}}}^{-1} \hat{S}_{\Gamma\Gamma}\right) \leq C(1 + \log(p^2 H/h))^2, \tag{4.22}$$

is given for the preconditioned operator $M_{\Gamma_{\text{BDDC}}}^{-1} \hat{S}_{\Gamma\Gamma}$ of a second-order elliptic problem. The constant C is independent of solution order, p, element size, h, and the subdomain size, H. Thus the condition number and hence number of iterations required to converge is independent of the number of subdomains and only weakly dependent on the solution order and the size of the subdomains.

4.3 The BDDC Preconditioner Revisited

Section 4.2 presented the BDDC preconditioner for the Schur complement problem (4.4). In this section the BDDC preconditioner is reformulated as a preconditioner for the entire finite element problem, (2.16). This formulation is necessary in order to be able to develop inexact BDDC preconditioners for which only an approximate Schur complement problem is formed using inexact local solves.

Consider the global fully-assembled system (2.16) written as in Equation (4.1). This system may be rewritten as:

$$\begin{bmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{bmatrix} = \begin{bmatrix} I_I & 0 \\ A_{\Gamma I}A_{II}^{-1} & I_\Gamma \end{bmatrix} \begin{bmatrix} A_{II} & 0 \\ 0 & \hat{S}_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} I_I & A_{II}^{-1}A_{I\Gamma} \\ 0 & I_\Gamma \end{bmatrix}, \qquad (4.23)$$

with corresponding inverse

$$\begin{bmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{bmatrix}^{-1} = \begin{bmatrix} I_I & -A_{II}^{-1}A_{I\Gamma} \\ 0 & I_\Gamma \end{bmatrix} \begin{bmatrix} A_{II}^{-1} & 0 \\ 0 & \hat{S}_{\Gamma\Gamma}^{-1} \end{bmatrix} \begin{bmatrix} I_I & 0 \\ -A_{\Gamma I}A_{II}^{-1} & I_\Gamma \end{bmatrix}. \qquad (4.24)$$

The BDDC preconditioner for the global system may be obtained by replacing $\hat{S}_{\Gamma\Gamma}^{-1}$ in (4.24) with $M_{\Gamma_{\text{BDDC}}}^{-1}$. Namely, the BDDC preconditioner for the global system (4.1) may be written as:

$$M_{\text{BDDC}}^{-1} = \begin{bmatrix} I_I & -A_{II}^{-1}A_{I\Gamma} \\ 0 & I_\Gamma \end{bmatrix} \begin{bmatrix} A_{II}^{-1} & 0 \\ 0 & M_{\Gamma_{\text{BDDC}}}^{-1} \end{bmatrix} \begin{bmatrix} I_I & 0 \\ -A_{\Gamma I}A_{II}^{-1} & I_\Gamma \end{bmatrix}. \qquad (4.25)$$

The BDDC preconditioner for the global system may be viewed as solving a finite-element problem on the partially assembled space $\tilde{\Lambda} = \tilde{\Lambda}_\Gamma \oplus \Lambda_I$ in order to precondition the finite element problem whose solution lies in the fully assembled space $\hat{\Lambda}$. It has been shown that the preconditioned operator, $M_{\text{BDDC}}^{-1}A$, has the same eigenvalue spectrum as the preconditioned Schur complement operator $M_{\Gamma_{\text{BDDC}}}^{-1}\hat{S}_{\Gamma\Gamma}$ [71]. Inserting the expression for M_{BDDC}^{-1} and performing a simple algebraic manipulation (4.25) may be rewritten as:

$$M_{\text{BDDC}}^{-1} = \tilde{\mathcal{H}}\tilde{A}^{-1}\tilde{\mathcal{H}}^{*^T}. \qquad (4.26)$$

Here $\tilde{\mathcal{H}}$ and $\tilde{\mathcal{H}}^{*^T} : \tilde{\Lambda} \rightarrow \hat{\Lambda}$ are harmonic averaging operators defined as:

$$\tilde{\mathcal{H}} = \begin{bmatrix} I_I & A_{II}^{-1}\tilde{A}_{I\Gamma}J_{D,\Gamma} \\ 0 & R_{D,\Gamma}^T \end{bmatrix} \quad \text{and} \quad \tilde{\mathcal{H}}^* = \begin{bmatrix} I_I & 0 \\ J_{D,\Gamma}^T\tilde{A}_{\Gamma I}A_{II}^{-1} & R_{D,\Gamma} \end{bmatrix}, \qquad (4.27)$$

with $\tilde{A}_{I\Gamma}$ and $\tilde{A}_{\Gamma I}$ corresponding to the unassembled stiffness matrix components (i.e $A_{I\Gamma} =$

$\tilde{A}_{I\Gamma}R_\Gamma$ and $A_{\Gamma I} = R_\Gamma^T \tilde{A}_{\Gamma I}$) and jump operator $J_{D,\Gamma} = \left(I - R_{D,\Gamma}R_\Gamma^T\right)$. The solution of the partially assembled system $\tilde{A}^{-1}$ may be written in the same form as (4.16):

$$\tilde{A}^{-1} \;=\; \Psi S_{\Pi\Pi}^{-1} \Psi^{*^T} + \operatorname{diag}\left(\{\tilde{A}^{(i)^{-1}}\}\right), \tag{4.28}$$

where Ψ, Ψ^* and $\tilde{A}^{(i)^{-1}}$ take on a similar form as Ψ_Γ, Ψ_Γ^* and $\tilde{S}_{\Gamma\Gamma}^{(i)^{-1}}$, respectively. Namely, the global primal basis functions are obtained by concatenating local primal basis functions defined by:

$$\begin{bmatrix} A^{(i)} & B^{(i)^T} \\ B^{(i)} & 0 \end{bmatrix} \begin{bmatrix} \Psi^{(i)} \\ * \end{bmatrix} = \begin{bmatrix} 0 \\ I_\Pi^{(i)} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} A^{(i)} & B^{(i)^T} \\ B^{(i)} & 0 \end{bmatrix}^T \begin{bmatrix} \Psi^{*(i)} \\ * \end{bmatrix} = \begin{bmatrix} 0 \\ I_\Pi^{(i)} \end{bmatrix}, \tag{4.29}$$

where constraints $B^{(i)}$ are obtained by expanding $B_\Gamma^{(i)}$ by zeros to all interior degrees of freedom. The application of $\tilde{A}^{(i)^{-1}}$ to a vector $v^{(i)} \in \Lambda^{(i)}$ is defined by the solution of a constrained Neumann problem:

$$\begin{bmatrix} A^{(i)} & B^{(i)^T} \\ B^{(i)} & 0 \end{bmatrix} \begin{bmatrix} \tilde{A}^{(i)^{-1}} v^{(i)} \\ * \end{bmatrix} = \begin{bmatrix} v^{(i)} \\ 0 \end{bmatrix}. \tag{4.30}$$

The primal Schur complement $S_{\Pi\Pi}$ may also be written as:

$$S_{\Pi\Pi} \;=\; \sum_{i=1}^{N} R_\Pi^{(i)^T} \Psi^{*(i)^T} A^{(i)} \Psi^{(i)} R_\Pi^{(i)}. \tag{4.31}$$

As in Section 3.2 the action of the BDDC preconditioner is depicted graphically for the sample Poisson problem presented in Section 3.3. Figure 4-1 depicts the coarse basis function associated with interface $\mathcal{E}_1$. Note that coarse basis function is in the partially assembled space, such that only the primal degree of freedom (i.e. the edge average) match between Ω_1 and Ω_2. Figure 4-2 depicts the partially assembled solve, while Figure 4-3 depicts the solution upon applying the harmonic extensions.

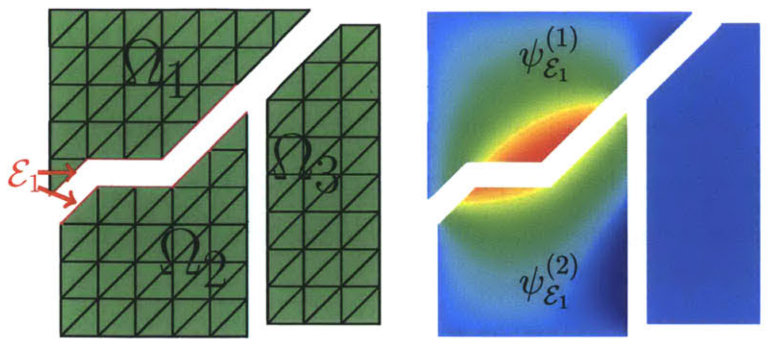


Figure 4-1: Coarse basis functions for BDDC

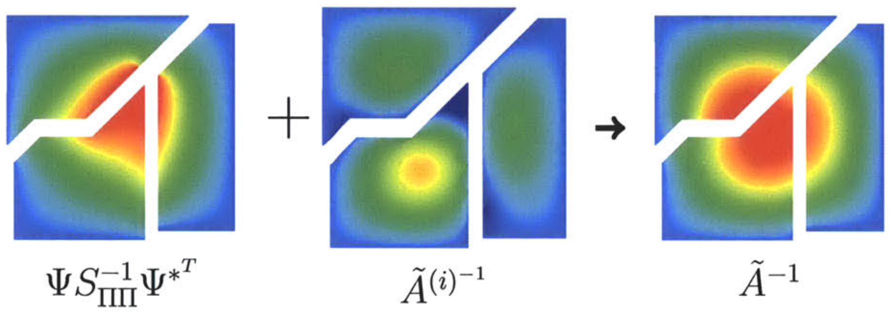


Figure 4-2: Partially Assembled solve applied to sample Poisson problem

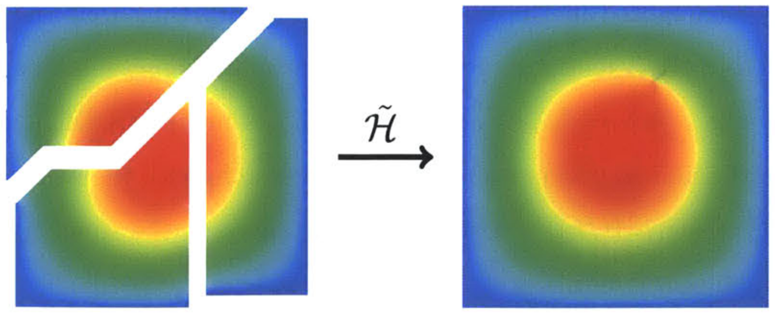


Figure 4-3: Effect of harmonic extensions for sample Poisson problem

4.4 Convergence Analysis

This section provides an analysis of the BDDC preconditioner for elliptic problems and proves a condition number bound for the preconditioned system. As a model problem, consider the following linear second-order scalar elliptic equation in a domain $\Omega \subset \mathbb{R}^2$

$$\begin{aligned} -\nabla \cdot (\mu \nabla u) &= f && \text{in } \Omega, \\ u &= 0 && \text{on } \partial\Omega, \end{aligned} \tag{4.32}$$

with positive $\mu \in L^\infty(\Omega)$ and $f \in L^2(\Omega)$. Additionally, assume that μ takes on a constant value on each subdomain Ω_i. The BDDC preconditioner presented in the previous sections is used to solve the HDG discretization of (4.32). In order to allow for large jumps in the coefficient μ across subdomains, the weighting function $\delta^\dagger$ is modified to be:

$$\delta_i^\dagger(\mathcal{E}_k) = \frac{\mu_i^\gamma}{\mu_i^\gamma + \mu_j^\gamma} \qquad \mathcal{E}_k = \Omega_i \cap \Omega_j \qquad \gamma \in [1/2, \infty]\,. \tag{4.33}$$

The HDG discretization of (4.32) results in a linear system which has the same structure and connectivity as a hybridized-mixed Raviart-Thomas (RT) or Brezzi-Douglas-Marini (BDM) finite element discretization of the same problem. The development and analysis of the BDDC preconditioner for hybridized-mixed finite element methods has already been presented in [119]. Given the similarities between hybridized versions of mixed methods and HDG methods, highlighted in [40] and [39], it is not surprising that the BDDC algorithm may be extended to HDG methods. Specifically, the analysis presented here builds upon the results of [119] which in turn builds upon [43] to show that the BDDC algorithm applied to the HDG discretization results in a preconditioned system with the usual condition number bound $\kappa \leq C(1 + \log(p^2 H/h))^2$. Using a similar analysis, the same condition number is derived for a large class DG methods in Appendix B.

A key component in the analysis of the BDDC method is to connect the HDG discretization to a continuous finite element discretization on a sub-triangulation $\tilde{\mathcal{T}}$ of $\mathcal{T}$ (See [43]). The following lemmas allow this connection to be drawn.

Lemma 4.4.1. *The local bilinear form $a_\kappa(\lambda_h, \lambda_h)$ satisfies*

$$a_\kappa(\lambda_h, \lambda_h) \;\geq\; 0, \tag{4.34}$$

with $a_\kappa(\lambda_h, \lambda_h) = 0$ iff λ_h is constant on $\partial\kappa$.

Proof. Follows from Proposition 3.3 in [40] Proposition 3.3. □

Lemma 4.4.2. *There exist constants c and C independent of p, h, H and $\mu_\kappa = \mu|_\kappa$ such that for all $\lambda_h \in M_h(\kappa)$*

$$ca_\kappa(\lambda_h, \lambda_h) \;\leq \mu_\kappa p^4 h^{n-2} \textstyle\sum_{\mathbf{x}_i, \mathbf{x}_j \in \kappa} (\lambda_h(\mathbf{x}_i) - \lambda_h(\mathbf{x}_j))^2 \leq \; Ca_\kappa(\lambda_h, \lambda_h), \tag{4.35}$$

where $\mathbf{x}_i, \mathbf{x}_j$ are the nodes defining the basis for λ_h on κ.

Proof. Lemma 4.4.2 is a direct consequence of Lemma 4.4.1 and a scaling argument. See [43] Lemma 4.3 for the equivalent proof for a mixed finite element discretization. □

Lemma 4.4.3. *For any region $\omega \in \Omega$ composed of elements κ in $\mathcal{T}$, there exist constants c and C independent of p, h, $|\omega|$ and μ such that for all $\lambda_h \in M_h(\omega)$*

$$ca_\omega(\lambda_h, \lambda_h) \;\leq \textstyle\sum_{\kappa \in \omega} \mu_\kappa p^4 h^{n-2} \sum_{\mathbf{x}_i, \mathbf{x}_j \in \kappa} (\lambda_h(\mathbf{x}_i) - \lambda_h(\mathbf{x}_j))^2 \leq \; Ca_\omega(\lambda_h, \lambda_h). \tag{4.36}$$

Proof. See [43] Theorem 4.1. □

Define $U_h(\Omega)$ to be the continuous linear finite element space defined on the triangulation $\hat{\mathcal{T}}$. Additionally define $U_h(\Omega_i)$ and $U_h(\partial\Omega_i)$, as the restriction of $U_h(\Omega)$ to Ω_i and $\partial\Omega_i$ respectively. Define $I_h^{\Omega_i}$ as the interpolant of any function into the space $U_h(\Omega)$ using the function values at the nodes $\mathbf{x}_i$. Similarly define $I_h^{\partial\Omega_i}$ which maps functions into the space $U_h(\partial\Omega)$ using nodal values at nodes $\mathbf{x}_i$ on $\partial\Omega_i$. Finally, define $\tilde{U}_h(\Omega_i)$ and $\tilde{U}_h(\partial\Omega_i)$ as the range of $I_h^{\Omega_i}$ and $I_h^{\partial\Omega_i}$ respectively.

The following lemmas which are the equivalent of those presented in [43] for mixed finite elements, allow the quadratic form given in Lemma 4.4.2 to be connected to a continuous finite element discretization.

Lemma 4.4.4. *There exists a constant $C > 0$ independent of p, h and H such that*

$$\begin{aligned} \left| I_h^{\partial\Omega_i}\phi \right|_{H^1(\Omega_i)} &\leq C\,|\phi|_{H^1(\Omega_i)} && \forall\phi \in U_h(\Omega_i), && (4.37)\\ \| I_h^{\partial\Omega_i}\phi \|_{L^2(\Omega_i)} &\leq C\| \phi \|_{L^2(\Omega_i)} && \forall\phi \in U_h(\Omega_i). && (4.38) \end{aligned}$$

Proof. See [43] Lemma 6.1. □

Lemma 4.4.5. *There exist constants $c, C > 0$ independent of p, h and H such that for any $\hat{\phi} \in \tilde{U}_h(\partial\Omega_i)$*

$$\begin{aligned} c\| \hat{\phi} \|_{H^{1/2}(\partial\Omega_i)} &\leq \inf_{\phi\in\tilde{U}_h(\Omega_i),\phi|_{\partial\Omega_i}=\hat{\phi}} \| \phi \|_{H^1(\Omega_i)} &&\leq C\| \hat{\phi} \|_{H^{1/2}(\partial\Omega_i)}, && (4.39)\\ c\left|\hat{\phi}\right|_{H^{1/2}(\partial\Omega_i)} &\leq \inf_{\phi\in\tilde{U}_h(\Omega_i),\phi|_{\partial\Omega_i}=\hat{\phi}} |\phi|_{H^1(\Omega_i)} &&\leq C\left|\hat{\phi}\right|_{H^{1/2}(\partial\Omega_i)}. && (4.40) \end{aligned}$$

Proof. See [43] Lemma 6.2. □

Lemma 4.4.6. *There exists a constant $C > 0$ independent of p, h and H such that*

$$\| I_h^{\partial\Omega_i}\hat{\phi} \|_{H^{1/2}(\partial\Omega_i)} \leq C\| \hat{\phi} \|_{H^{1/2}(\partial\Omega_i)} \qquad \forall\hat{\phi} \in U_h(\partial\Omega_i). \qquad (4.41)$$

Proof. See [43] Lemma 6.3. □

Lemma 4.4.7. *There exist constants c and C independent of p, h, H and μ_i such that for all $\lambda_\Gamma^{(i)} \in \Lambda_\Gamma^{(i)}$,*

$$c\mu_i \left| I_h^{\partial\Omega_i}\lambda_\Gamma^{(i)} \right|^2_{H^{1/2}(\partial\Omega_i)} \leq \lambda_\Gamma^{(i)T} S_{\Gamma\Gamma}^{(i)} \lambda_\Gamma^{(i)} \leq C\mu_i \left| I_h^{\partial\Omega_i}\lambda_\Gamma^{(i)} \right|^2_{H^{1/2}(\partial\Omega_i)}. \qquad (4.42)$$

Proof. See [43] Theorem 6.5. □

Using known results for continuous finite element discretizations gives the main theoretical result.

Theorem 4.4.8. *The condition number of the preconditioner operator $M_{\Gamma_{BDDC}}^{-1}\hat{S}_{\Gamma\Gamma}$ is bounded by $C(1 + \log(p^2H/h))^2$ where C is a constant independent of p, h, H and μ.*

Proof. See for example [75] Theorem 3 or [119] Theorem 6.1. □

4.5 Robin-Robin Interface Condition

The BDDC preconditioner was originally developed for the solution of symmetric positive-definite system and must be modified for the solution of non-symmetric systems. In [1], Achdou et al. replaced the Neumann-Neumann interface condition corresponding to natural boundary conditions on Γ, with a Robin-Robin interface condition for a continuous finite element discretization of an advection-diffusion problem. The Robin-Robin interface condition ensures the coercivity of the local bilinear form, $a_i(\lambda, \mu)$. Additionally, Fourier analysis, on a partitioning of the domain into strips in the cross-stream direction, showed that in the convective limit, the resulting algorithm converges in a number of iterations equal to half the number of subdomains in the stream-wise direction. The Robin-Robin interface conditions have been used along with a FETI method to solve linear convection-diffusion problems by Toselli [117]. Similarly, Tu and Li used the Robin-Robin interface condition to extend the BDDC method to convection-diffusion problems [121]. Tu and Li introduced additional primal degrees of freedom corresponding to "flux" constraints and showed that the resulting BDDC algorithm was scalable if the subdomain length scale, H, was sufficiently small relative the viscosity. Namely, the behaviour of BDDC preconditioner matches the symmetric, diffusion dominated limit if the subdomain Peclet number is sufficiently small. Yano and Darmofal generalized the Robin-Robin interface condition to the Euler equations discretized using entropy variables by deriving a local bilinear for which conserves the energy stability of the global bilinear form[123, 124].

As in the case of continuous finite elements, the local bilinear forms for the HDG discretization are not in general coercive for convection-diffusion problems. In the remainder of this section, a Robin-Robin interface condition is derived for the HDG discretization which modifies the local bilinear form to ensure its positivity. Consider the linearized problem:

$$\left(A_{ikl}u_l - K_{ijkl}u_{l,x_j}\right)_{x_i} = f_k, \tag{4.43}$$

with symmetric flux Jacobian and viscosity tensor (i.e. $A_{ikl} = A_{ilk}$ and $K_{ijkl} = K_{jilk}$).

Equations (2.7)-(2.8) are rewritten as:

$$\begin{aligned}
(q_{ik}, v_{ik})_\kappa + (u_k, v_{ik,x_i})_\kappa - \langle \hat{u}_k, v_{ik} n_i \rangle_{\partial\kappa} &= 0 \qquad \forall v \in \mathbf{V}_h^p, &(4.44)\\
-(A_{ikl}u_k - K_{ijkl}q_{jl}, w_{k,x_i})_\kappa + \left\langle (\widehat{F}_{ik} + \widehat{G}_{ik}) n_i, w_k \right\rangle_{\partial\kappa} &= (f_k, w_k)_\kappa \quad \forall w \in W_h^p, &(4.45)
\end{aligned}$$

with numerical flux:

$$\left(\widehat{F}_{ik} + \widehat{G}_{ik}\right) n_i = A_{ikl} n_i u_l - K_{ijkl} n_i q_{jl} + S_{kl}(u_l - \hat{u}_l). \tag{4.46}$$

The element wise contribution to the HDG bilinear form, (ignoring boundary contributions on $\partial\Omega$), is given by:

$$\begin{aligned}
a_\kappa(\lambda, \lambda) &= -\langle (A_{ikl}\lambda_l - K_{ijkl}q_{jl}) n_i + S_{kl}(u_l - \lambda_l), \lambda_k \rangle_{\partial\kappa} &(4.47)\\
&= -\langle A_{ikl}\lambda_l n_i, \lambda_k \rangle_{\partial\kappa} + \langle K_{ijkl} q_{jl} n_i, \lambda_k \rangle_{\partial\kappa} - \langle S_{kl}(u_l - \lambda_l), \lambda_k \rangle_{\partial\kappa}.
\end{aligned}$$

From (4.44) with $v_{jl} = K_{ijkl} q_{jl}$ (and switching i,j and k,l) gives

$$\begin{aligned}
\langle \lambda_k, K_{ijkl} q_{jl} n_i \rangle_{\partial\kappa} &= \left(u_k, (K_{ijkl} q_{jl})_{,x_i} \right)_\kappa + (q_{ik}, K_{ijkl} q_{jl})_\kappa &(4.48)\\
&= -(u_{k,x_i}, K_{ijkl} q_{jl})_\kappa + \langle u_k, K_{ijkl} q_{jl} \rangle_{\partial\kappa} + (q_{ik}, K_{ijkl} q_{jl})_\kappa.
\end{aligned}$$

From the conservation equation (4.45) with $w_k = u_k$ (and $f_k = 0$):

$$\begin{aligned}
0 &= -(A_{ikl}u_l - K_{ijkl}q_{jl}, u_{k,x_i})_\kappa + \left\langle \widehat{F_{ik} n_i}, u_k \right\rangle_{\partial\kappa} &(4.49)\\
&= -(A_{ikl}u_l, u_{k,x_i})_\kappa + (K_{ijkl}q_{jl}, u_{k,x_i})_\kappa\\
&\quad + \langle A_{ikl}\lambda_l, n_i u_k \rangle_{\partial\kappa} - \langle K_{ijkl} q_{jl}, n_i u_k \rangle_{\partial\kappa} + \langle S_{kl}(u_l - \lambda_l), u_k \rangle_{\partial\kappa}.
\end{aligned}$$

Adding (4.47) and (4.49), using the substitution in (4.48), and then integrating by parts gives:

$$\begin{aligned}
a_\kappa(\lambda, \lambda) &= (q_{ik}, K_{ijkl} q_{jl})_\kappa + \left\langle \left(S_{kl} - \frac{1}{2} A_{ikl} n_i \right) (u_l - \lambda_l), (u_k - \lambda_k) \right\rangle_{\partial\kappa}\\
&\quad -\frac{1}{2} \langle A_{ikl} n_i \lambda_l, \lambda_k \rangle_{\partial\kappa}. &(4.50)
\end{aligned}$$

While the first two terms of (4.50) are positive, $-\frac{1}{2}\langle A_{ikl}n_i\lambda_l, \lambda_k\rangle_{\partial\kappa}$, in general, is not. Consider a modified local bilinear form $\tilde{a}_\kappa(\lambda,\mu)$ which is obtained from $a_\kappa(\lambda,\mu)$ by subtracting the term which may be negative from each face $\partial\kappa\backslash\partial\Omega$:

$$\tilde{a}_\kappa(\lambda,\mu) \;:=\; a_\kappa(\lambda,\mu) + \frac{1}{2}\langle A_{ikl}n_i\lambda_l, \mu_k\rangle_{\partial\kappa}. \tag{4.51}$$

By construction $\tilde{a}_\kappa(\lambda,\lambda) \geq 0$. Note that on each edge e shared by elements $\kappa^\pm$, $\frac{1}{2}\left\langle A_{ikl}n_i^+\lambda_l, \lambda_k\right\rangle_{\partial\kappa^+} + \frac{1}{2}\left\langle A_{ikl}n_i^-\lambda_l, \lambda_k\right\rangle_{\partial\kappa^-} = 0$, since $n^- = -n^+$. Thus, the sum of the modified local bilinear forms $\tilde{a}_\kappa(\lambda,\mu)$ gives exactly the sum of the original local bilinear forms $a_\kappa(\lambda,\mu)$ namely:

$$a(\lambda,\mu) \;=\; \sum_\kappa a_\kappa(\lambda,\mu) = \sum_\kappa \tilde{a}_\kappa(\lambda,\mu). \tag{4.52}$$

A corresponding subdomain-wise bilinear form may thus be defined as:

$$\tilde{a}_i(\lambda,\mu) \;:=\; \sum_{\kappa\in\Omega_i} \tilde{a}_\kappa(\lambda,\mu), \tag{4.53}$$

where the positivity of $\tilde{a}_i(\lambda,\lambda)$ is guaranteed by construction:

$$\begin{aligned}\tilde{a}_i(\lambda,\lambda) &= \sum_{\kappa\in\Omega_i} (q_{ik}, K_{ijkl}q_{jl})_\kappa + \left\langle \left(S_{kl} - \frac{1}{2}A_{ikl}n_i\right)(u_l - \lambda_l), (u_k - \lambda_k)\right\rangle_{\partial\kappa} \\ &\geq 0.\end{aligned} \tag{4.54}$$

In practice, the element-wise forms only need to be modified on edges $e \in \Gamma$ as the subdomain interior contributions cancel. While the original subdomain-wise bilinear form $a_i(\lambda,\mu)$ corresponds to setting a Neumann boundary condition on Γ_i for an advection-diffusion problem, the modified subdomain-wise form $\tilde{a}_i(\lambda,\mu)$ corresponds to setting a Robin boundary condition on Γ_i. Thus, the interface condition corresponding to the use of the modified subdomain-wise bilinear form is called a Robin-Robin interface condition. In the case of a purely hyperbolic system of equations (eg. the Euler equations) the interface conditions are not strictly speaking Neumann or Robin. Nonetheless, throughout this thesis the algorithm using the original local bilinear form, $a_i(\lambda,\mu)$, will be referred to as the Neumann-Neumann algorithm (having Neumann interface conditions), while the algorithm using the modified bilinear form, $\tilde{a}_i(\lambda,\mu)$, will be referred to as the Robin-Robin algorithm (having Robin interface conditions).

While the Robin-Robin interface condition required to ensure the positivity of the local bilinear form is unique for a scalar PDE, other choices are possible when considering a system of equations. In the case of a system of equations, the local bilinear form may be modified to be:

$$\breve{a}_\kappa(\lambda, \mu) = -\left\langle \widehat{F_{ikl} n_i}, \mu_k \right\rangle_{\partial\kappa} + \frac{1}{2} \langle A_{ikl} n_i \lambda_l, \mu_k \rangle_{\partial\kappa} + \langle Z_{ikl} n_i \lambda_l, \mu_k \rangle_{\partial\kappa}, \tag{4.55}$$

where $Z_{ikl} n_i$ is skew-symmetric matrix (i.e $Z_{ikl} n_i = -Z_{ilk} n_i$). The addition of the skew symmetric term does not change the positivity of the local bilinear form as:

$$\begin{aligned} \breve{a}_\kappa(\lambda, \lambda) &= -\left\langle \widehat{F_{ikl} n_i}, \lambda_k \right\rangle_{\partial\kappa} + \frac{1}{2} \langle A_{ikl} n_i \lambda_l, \lambda_k \rangle_{\partial\kappa} + \langle Z_{ikl} n_i \lambda_l, \lambda_k \rangle_{\partial\kappa} \\ &= -\left\langle \widehat{F_{ikl} n_i}, \lambda_k \right\rangle_{\partial\kappa} + \frac{1}{2} \langle A_{ikl} n_i \lambda_l, \lambda_k \rangle_{\partial\kappa} + \cancel{\frac{1}{2} \langle Z_{ikl} n_i \lambda_l, \lambda_k \rangle_{\partial\kappa}} - \cancel{\frac{1}{2} \langle Z_{ilk} n_i \lambda_l, \lambda_k \rangle_{\partial\kappa}} \\ &= \tilde{a}_\kappa(\lambda, \lambda). \end{aligned} \tag{4.56}$$

In Chapter 6 the Euler system is further analyzed in order to choose a form for Z_{ikl} which improves the convergence of the BDDC algorithm.

4.6 Numerical Results

This section presents numerical results to assess the performance of the BDDC preconditioner.

In the first numerical experiment, the Poisson model problem, (3.10), is solved on the structured mesh described in Section 3.3. Table 4.1 shows the number of GMRES iterations required to decrease the l_2-norm of the residual by a factor of 10^6, varying p, n and N. As predicted by the analysis in Section 4.4 the number of iterations is bounded as the number of subdomains, N, increases. Additionally, the number of iterations grows only weakly when increasing the number of elements per subdomain, n, or the solution order p.

The second numerical experiment examines the behaviour of the BDDC preconditioner for the elliptic problem (4.32), with large jumps in the coefficient μ. The domain is partitioned in a checkerboard pattern with $\mu = 1$ in half of the subdomains and $\mu = 1000$ in the remaining subdomains. Initially the weighting function $\delta_i^\dagger$ is set such that $\delta_i^\dagger(\mathcal{E}_k) = \delta_j^\dagger(\mathcal{E}_k) =$

N	n	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$
4	128	2	1	1	1	1	1
16	128	7	8	8	9	9	9
64	128	10	12	12	14	14	16
256	128	9	12	13	14	15	16
1024	128	10	12	13	14	15	16
64	8	5	8	10	10	12	12
64	32	7	10	11	12	13	14
64	128	10	12	12	14	14	16
64	512	11	13	14	16	16	17
64	2048	13	15	16	17	18	19

Table 4.1: Number of GMRES iterations using BDDC preconditioner for Poisson problem on isotropic structured mesh

1/2 on each subdomain interface $\mathcal{E}_k = \Omega_i \cap \Omega_j$, which corresponds to setting $\gamma = 0$ in (4.33). This choice of γ does not satisfy the assumption $\gamma \in [0, \infty)$, and poor convergence of the BDDC algorithm is seen. Table 4.2 shows the corresponding iteration counts for varying p, and N with fixed $n = 128$. Next $\delta_i^\dagger$ is set as in (4.33) with $\gamma = 1$. With this choice of $\delta_i^\dagger$ the good convergence properties of the BDDC algorithm are recovered.

	N	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$
$\delta_i^\dagger(x) = \frac{1}{2}$	4	5	7	6	6	5	5
	16	18	24	26	29	32	39
	64	48	78	101	118	135	149
	256	73	112	143	166	187	237
	1024	79	124	157	209	231	251
$\delta_i^\dagger(x) = \frac{\mu_i}{\mu_i+\mu_j}$	4	2	2	1	1	1	1
	16	5	5	5	6	6	7
	64	9	10	10	11	11	11
	256	15	15	17	17	17	17
	1024	15	16	18	18	18	18

Table 4.2: Iteration count for BDDC preconditioner with $\mu = 1$ or $\mu = 1000$, $n = 128$

For the third numerical experiment, the advection-diffusion boundary layer problem, (3.12), is solved on the isotropic mesh. Tables 4.3 and 4.4 give the number of iterations required to converge for $\mu = 1$ and $\mu = 10^{-6}$, respectively. In the diffusion-dominated case, $(\mu = 1)$, the behaviour is similar to the purely elliptic case. Namely, the number of iterations depends weakly on n and p, and is bounded as N increases. In the convection-dominated case, ($\mu = 10^{-6}$), the number of iterations is approximately half the number of subdomains in the convective direction (i.e. $\sqrt{N}/2$).

N	n	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$
4	128	4	4	5	5	5	5
16	128	6	7	8	8	8	9
64	128	7	7	8	8	9	9
256	128	6	7	8	8	8	9
1024	128	7	7	8	8	8	9
64	8	4	6	7	7	8	8
64	32	5	7	7	8	8	9
64	128	7	7	8	8	9	9
64	512	7	8	8	8	9	10
64	2048	7	8	8	9	10	10

Table 4.3: Number of GMRES iterations for advection-diffusion boundary layer problem with $\mu = 1$ on isotropic structured mesh

N	n	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$
4	128	1	1	1	1	1	1
16	128	2	2	3	3	3	3
64	128	4	5	5	5	6	6
256	128	9	9	10	10	11	12
1024	128	17	17	18	18	19	19
64	8	4	4	4	5	5	5
64	32	4	4	5	5	5	5
64	128	4	5	5	5	6	6
64	512	5	5	5	6	7	7
64	2048	5	5	6	7	7	7

Table 4.4: Number of GMRES iterations for advection-diffusion boundary layer problem with $\mu = 10^{-6}$ on isotropic structured mesh

As noted in Section 3.3, the boundary layer is not resolved on the isotropic mesh and an anisotropic mesh should be used. Table 4.5 gives the number of GMRES iterations required to converge the boundary layer problem with $\mu = 10^{-6}$ on the set of anisotropic meshes introduced in Section 3.3. Table 4.5 shows only slight increase in the number of iterations relative Table 4.4. Table 4.6 shows the iteration count on both isotropic and anisotropic meshes over a wide range of μ. The BDDC preconditioner performs well over the entire range of μ for both isotropic and anisotropic meshes.

N	n	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$
4	128	1	1	1	1	1	1
16	128	4	4	4	4	4	4
64	128	7	8	8	8	8	8
256	128	11	12	12	12	12	11
1024	128	19	20	20	20	20	20
64	8	6	8	8	8	8	8
64	32	7	8	8	8	8	8
64	128	7	8	8	8	8	8
64	512	8	8	8	8	8	7
64	2048	7	7	7	7	7	7

Table 4.5: Number of GMRES iterations for advection-diffusion boundary layer problem with $\mu = 10^{-6}$ on anisotropic structured mesh

	μ	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$\frac{\|u-u^*\|_{H_1}}{\|u^*\|_{H_1}}$ $(p=5)$
Isotropic Mesh	1	7	7	8	8	9	9	4.2×10^{-11}
	10^{-1}	7	8	9	9	10	10	2.5×10^{-11}
	10^{-2}	9	10	11	11	11	12	3.4×10^{-10}
	10^{-3}	7	8	8	9	10	10	5.2×10^{-8}
	10^{-4}	6	7	7	7	7	7	1.9×10^{-5}
	10^{-5}	5	5	6	7	7	7	3.4×10^{-3}
	10^{-6}	4	5	5	5	6	6	7.8×10^{-2}
Anisotropic Mesh	1	7	7	8	8	9	9	4.2×10^{-11}
	10^{-1}	8	9	10	10	10	11	2.3×10^{-11}
	10^{-2}	8	9	10	10	11	11	1.6×10^{-11}
	10^{-3}	8	8	8	9	9	9	1.7×10^{-11}
	10^{-4}	7	8	8	8	7	7	2.1×10^{-11}
	10^{-5}	7	7	7	7	7	7	2.6×10^{-11}
	10^{-6}	7	8	8	8	8	8	3.8×10^{-11}

Table 4.6: Number of GMRES iterations on both isotropic and anisotropic meshes with $N = 64$, $n = 128$

In the final numerical experiment presented in this section, the performance of the BDDC preconditioner is compared to the ASM and ASM_A preconditioners developed in Chapter 3. The advection-diffusion boundary layer problem is solved with $\mu = 10^{-4}$ on a sequence of anisotropic unstructured meshes generated using the Bidimensional Anisotropic Mesh Generator (BAMG) [63]. An initial mesh with approximately 512 elements is generated through an adaptive process, while finer meshes are obtained by refining the desired grid metric and completely remeshing. The meshes are partitioned into N subdomains, with approximately n elements each, using ParMetis [65]. Figure 4-4 plots the mesh with 512 elements as well its partition into four subdomains.

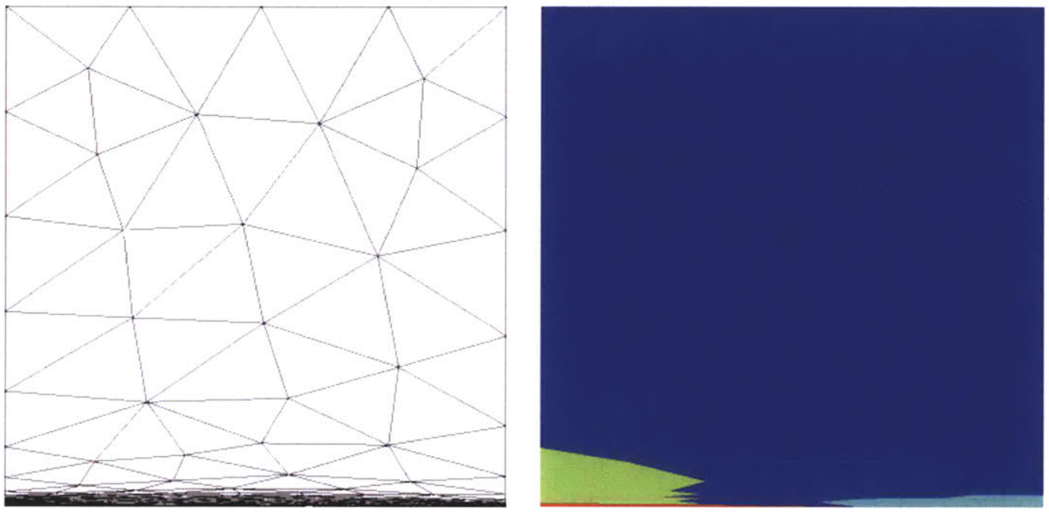

(a) Mesh (b) Partition

Figure 4-4: Unstructured mesh and partition for advection-diffusion boundary layer problem

A single application of the BDDC preconditioner involves a local constrained Neumann solve on each subdomain, and either a forward- or back-solve for the application of the harmonic extensions. As it is assumed that the dominant cost of the application of any of the preconditioners is the cost of the local solves, a single application of the BDDC preconditioner is approximately twice that of the ASM or ASM_A preconditioners. Thus the relative performance of the different preconditioning algorithms is assessed in terms of the number of local linear solves required to reduce the l_2-norm of the residual by a factor of 10^4. Figure 4-5 plots the number of local linear solves required to solve the advection-diffusion boundary layer problem for fixed $n = 128$ over a large range of N, while Figure 4-6 shows the performance of the preconditioners for fixed, $N = 64$ with increasing number of elements per subdomain. In general, the performance of the BDDC preconditioner is superior to the ASM and ASM_A preconditioners, for the range of N and n considered. The performance of ASM preconditioner without a coarse space degrades much more rapidly than the ASM_A and BDDC preconditioners with coarse spaces as the number of subdomains is increased.

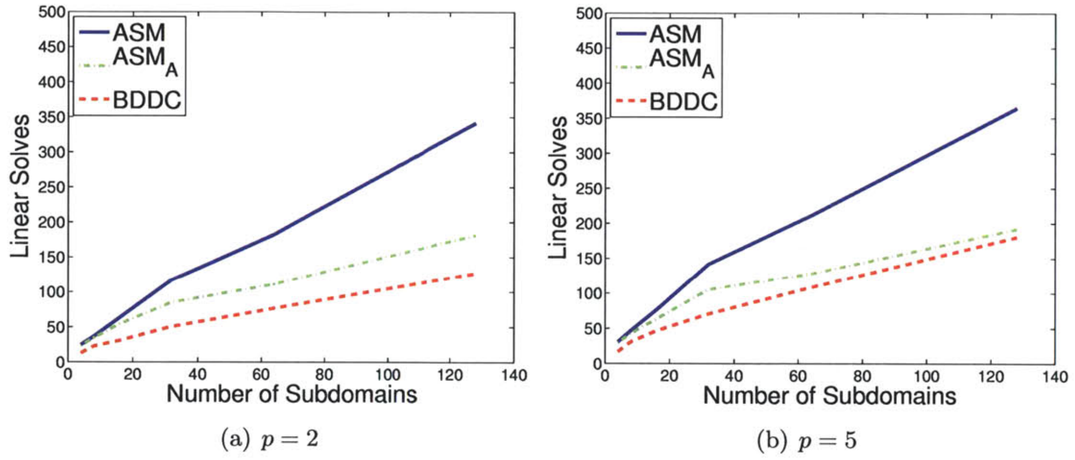
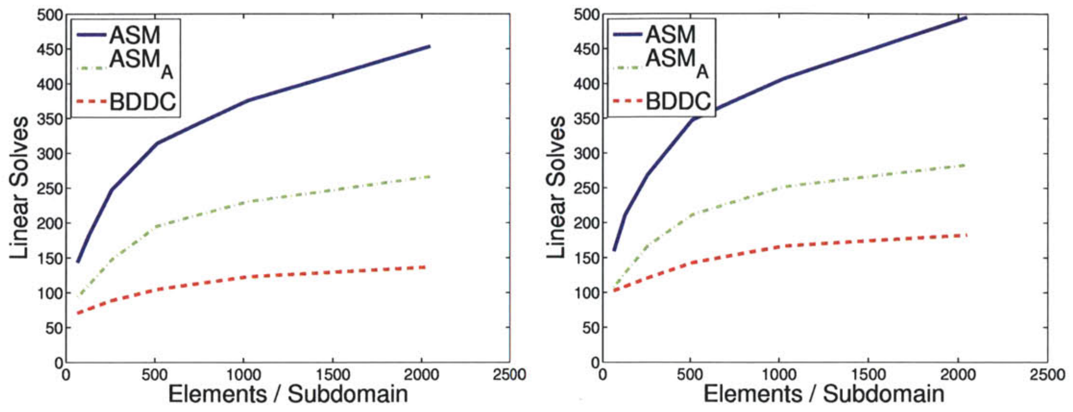


Figure 4-5: Number of local linear solves for advection-diffusion boundary layer problem with $\mu = 10^{-4}$, and $n \approx 128$ on anisotropic unstructured meshes

Unfortunately, the performance of all three preconditioners is much worse on the unstructured anisotropic meshes than that using the structured meshes. Table 4.7 gives the number of local linear solves for this problem using both structured and unstructured meshes. For example at $n = 128$ and $N = 64$ the unstructured mesh problem requires approximately 10 times as many iteration as for the structure mesh problem.



(a) $p = 2$ (b) $p = 5$

Figure 4-6: Number of local linear solves for advection-diffusion boundary layer problem with $\mu = 10^{-4}$, and $N = 64$ on anisotropic unstructured meshes

			$p=2$			$p=5$		
	N	n	ASM	ASM$_A$	BDDC	ASM	ASM$_A$	BDDC
Structured	4	128	3	6	2	3	6	2
	16	128	14	9	8	13	8	8
	64	128	25	12	8	22	9	8
	64	32	21	11	10	19	9	8
	64	128	25	12	8	22	9	8
	64	512	30	12	8	25	8	6
Unstructured	4	128	24	32	12	30	29	16
	16	128	63	54	30	76	63	46
	64	128	182	111	76	211	128	108
	64	32	113	72	52	122	83	72
	64	128	182	111	76	211	128	108
	64	512	314	195	104	347	211	142

Table 4.7: Number of local linear solves both isotropic and anisotropic meshes for scalar boundary layer problem with $\mu = 10^{-4}$

Chapter 5

Inexact BDDC

The majority of the cost of the BDDC preconditioner comes from solving local Dirichlet problems corresponding to ${A_{II}^{(i)}}^{-1}$ in the harmonic extension operators, and solving local constrained Neumann problems corresponding to ${\tilde{A}^{(i)}}^{-1}$ to invert the partially assembled finite element problem. For practical problems computing and storing exact factorizations for the solution of these problems may be prohibitively expensive. Thus, the action of these solvers may be replaced with inexact solvers. Inexact BDDC algorithms have been previously studied for continuous finite element discretizations of elliptic problems in [48, 72]. In [72] h-multigrid V-cycles are used to replace the action of the local solvers, while in [48] approximate local solvers are developed using both an inexact factorization, and an algebraic multigrid solver. In [124] the action of the local solvers are replaced with a dual-threshold incomplete factorization (ILUT) and a $p = 1$ coarse grid correction for the solution of the advection-diffusion equations.

This chapter develops an inexact variant of the BDDC preconditioner presented in Chapter 4 based on an incomplete factorization combined with a p-multigrid type coarse grid correction. Thus, the local solvers presented here most closely match those presented in [124]. A key difference between the work of [124] and that presented here involves the implementation of the BDDC algorithm which is discussed in Section 5.1. Section 5.2 presents inexact local solvers using block ILU(0) and ILUT factorizations. Section 5.3 presents the coarse grid correction. Finally, Section 5.4 presents numerical results using the inexact preconditioner.

5.1 A Note on the BDDC Implementation

In [124] the BDDC algorithm is implemented using a change of basis where the degrees of freedom spanning Λ_Γ are reparameterized such that all primal and dual variables are explicitly specified [68]. In this implementation the local degrees of freedom may be written as $\lambda^{(i)} = \begin{bmatrix} \lambda_I^{(i)} & \lambda_\Delta^{(i)} & \lambda_\Pi^{(i)} \end{bmatrix}^T$, with local system matrices:

$$A^{(i)} = \begin{bmatrix} A_{II}^{(i)} & A_{I\Delta}^{(i)} & A_{I\Pi}^{(i)} \\ A_{\Delta I}^{(i)} & A_{\Delta\Delta}^{(i)} & A_{\Delta\Pi}^{(i)} \\ A_{\Pi I}^{(i)} & A_{\Pi\Delta}^{(i)} & A_{\Pi\Pi}^{(i)} \end{bmatrix} = \begin{bmatrix} A_{rr}^{(i)} & A_{r\Pi}^{(i)} \\ A_{\Pi r}^{(i)} & A_{\Pi\Pi}^{(i)} \end{bmatrix}. \tag{5.1}$$

The local constrained solves then simplify to an inversion of the blocks $A_{rr}^{(i)}$ corresponding to interior and dual degrees of freedom. While the local systems $A^{(i)}$ may be singular, the blocks $A_{rr}^{(i)}$ are guaranteed to be nonsingular as it is assumed that sufficient number of primal degrees of freedom have been chosen such that the constrained Neumann problems are well defined. The advantage of this approach is that saddle point problems of the form (4.30) do not need to solved. In particular, when exact solvers are used, the change of basis approach ensures that zero pivots will not be encountered during the factorization, allowing the use of standard Cholesky factorization for symmetric problems [68].

Unfortunately, the change of basis to make explicit the dual and primal degrees of freedom destroys some of the nice properties of the local system matrices. In particular, for elliptic problems, discretized using standard nodal basis functions, $A^{(i)}$'s are (possibly singular) M-matrices. On the other hand when a change of basis is performed as described in [68], neither the systems (5.1), nor the blocks $A_{rr}^{(i)}$ are M-matrices. While ILU and common smoothers such as Jacobi or Gauss-Seidel are well defined for M-matrices [91], these iterative methods may perform poorly or break down for other type of matrices (even if they are symmetric positive definite). In particular, numerical experiments, not presented here, show very poor performance when using an ILU factorization to solve the local system $A_{rr}^{(i)}$. Interestingly, allowing additional fill-in does not necessarily improve the performance of the ILU solver, and in some cases leads to a degradation in performance.

In order to be able to use common iterative methods such as ILU, Jacobi, or Gauss-Seidel as smoothers, the inexact local solvers presented in the following sections are based on an implementation of the BDDC algorithm with the original variables and using constraints.

5.2 BDDC using Incomplete Factorizations

Incomplete factorizations have been extensively used as preconditioners to solve discretizations of the Euler and Navier-Stokes equations [3, 10, 20, 34, 46, 59, 69, 81, 95, 103, 105, 122]. ILU factorizations with an appropriate ordering have proven successful for convection-dominated problems, where the inexact factorization is able to capture strong coupling between elements in dominant directions in the flow. This work uses the dual-threshold incomplete factorization ILUT(τ,π), with drop tolerance, τ, and additional fill-in per row, π[110]. In particular, the block variant of the ILUT algorithm is used where the fill-in is introduced block-wise [123]. In the limit $\pi \to \infty$, $\tau = 0$ the ILUT factorization gives an exact factorization, while for $\pi = 0$, no additional fill-in is introduced such that the factorization reduces to ILU(0). In this case, the ILU(0) factorization may be performed in-place in order to save both memory and computational time [46].

5.2.1 Inexact Harmonic Extensions

The application of the discrete harmonic extensions $\mathcal{H}$ and $\mathcal{H}^*$ require computing the action of $-A_{II}^{(i)^{-1}} A_{I\Gamma}^{(i)}$ and $-A_{\Gamma I}^{(i)} A_{II}^{(i)^{-1}}$ respectively. In order to replace the action of these operators with inexact solvers based on an incomplete factorization, consider first the exact block LU factorization of the local stiffness matrices:

$$\begin{bmatrix} A_{II}^{(i)} & A_{I\Gamma}^{(i)} \\ A_{\Gamma I}^{(i)} & A_{\Gamma\Gamma}^{(i)} \end{bmatrix} = \begin{bmatrix} L_{II}^{(i)} & 0 \\ L_{\Gamma I}^{(i)} & I \end{bmatrix} \begin{bmatrix} U_{II}^{(i)} & U_{I\Gamma}^{(i)} \\ 0 & S_{\Gamma\Gamma}^{(i)} \end{bmatrix}. \tag{5.2}$$

Here, $L_{II}^{(i)} U_{II}^{(i)}$ is the exact block LU factorization of $A_{II}^{(i)}$ while $L_{\Gamma I}^{(i)} = A_{\Gamma I}^{(i)} U_{II}^{(i)^{-1}}$, $U_{I\Gamma}^{(i)} = L_{II}^{(i)^{-1}} A_{I\Gamma}^{(i)}$ and $S_{\Gamma\Gamma}^{(i)} = A_{\Gamma\Gamma}^{(i)} - L_{\Gamma I}^{(i)} U_{I\Gamma}^{(i)}$. The action of $-A_{II}^{(i)^{-1}} A_{I\Gamma}^{(i)}$ on a vector $v_\Gamma^{(i)} \in \Lambda_\Gamma^{(i)}$ may be computed by backwards substitution:

$$-A_{II}^{(i)^{-1}} A_{I\Gamma}^{(i)} v_\Gamma^{(i)} = -U_{II}^{(i)^{-1}} U_{I\Gamma}^{(i)} v_\Gamma^{(i)} \quad \text{or} \quad \begin{bmatrix} U_{II}^{(i)} & U_{I\Gamma}^{(i)} \\ 0 & I \end{bmatrix} \begin{bmatrix} -A_{II}^{(i)^{-1}} A_{I\Gamma}^{(i)} v_\Gamma^{(i)} \\ * \end{bmatrix} = \begin{bmatrix} 0 \\ v_\Gamma^{(i)} \end{bmatrix}. \tag{5.3}$$

Similarly the action of $-A_{\Gamma I}^{(i)} A_{II}^{(i)^{-1}}$ on a vector $v_I \in \Lambda_I^{(i)}$ may be computed by forwards substitution:

$$-A_{\Gamma I}^{(i)} A_{II}^{(i)^{-1}} v_I^{(i)} = -L_{\Gamma I}^{(i)} L_{II}^{(i)^{-1}} v_I^{(i)} \quad \text{or} \quad \begin{bmatrix} L_{II}^{(i)} & 0 \\ L_{\Gamma I}^{(i)} & I \end{bmatrix} \begin{bmatrix} * \\ -A_{\Gamma I}^{(i)} A_{II}^{(i)^{-1}} v_I^{(i)} \end{bmatrix} = \begin{bmatrix} v_I^{(i)} \\ 0 \end{bmatrix}. \tag{5.4}$$

Inexact harmonic extensions are developed by replacing the exact factorization in (5.2) with an incomplete factorization. The inexact harmonic extensions are computed as in (5.3) and (5.4). As fill-in is dropped in the computation of factors $\tilde{U}_{I\Gamma}^{(i)}$ and $\tilde{L}_{\Gamma I}^{(i)}$, in general

$$-\tilde{A}_{II}^{(i)^{-1}} A_{I\Gamma}^{(i)} = -\tilde{U}_{II}^{(i)^{-1}} \tilde{L}_{II}^{(i)^{-1}} A_{I\Gamma}^{(i)} \neq -\tilde{U}_{II}^{(i)^{-1}} \tilde{U}_{I\Gamma}^{(i)}, \tag{5.5}$$

$$-A_{\Gamma I}^{(i)} \tilde{A}_{II}^{(i)^{-1}} = -A_{\Gamma I}^{(i)} \tilde{U}_{II}^{(i)^{-1}} \tilde{L}_{II}^{(i)^{-1}} \neq -\tilde{L}_{\Gamma I}^{(i)} \tilde{L}_{II}^{(i)^{-1}}, \tag{5.6}$$

where $\tilde{(\cdot)}$ correspond to the incomplete factorization.

5.2.2 Inexact Partially Assembled Solve

The creation of the primal basis functions and the partially assembled solve involves, on each subdomain, inverting a system of the form:

$$\tilde{\mathbf{A}}^{(i)} = \begin{bmatrix} A^{(i)} & B^{(i)^T} \\ B^{(i)} & 0 \end{bmatrix}. \tag{5.7}$$

Solving problems with the system (5.7) is complicated by the fact that $A^{(i)}$ is potentially singular. Thus when an exact factorization of (5.7) is performed without reordering, a zero pivot will be encountered and the factorization will fail. For continuous finite element discretizations where subdomain corner degrees of freedom are included as primal variables, simply removing the rows and columns corresponding to the corner degrees of freedom ensures that the remaining block of $A^{(i)}$ is non-singular [48]. In this work, primal degrees of freedom correspond only to averages on subdomain interfaces, thus such an approach is not possible. A zero pivot may also be avoided by using a reordering where a small number of row/columns from $A^{(i)}$ are placed after the constraint equations such that the remaining block of $A^{(i)}$ is non-singular [68]. However, this approach relies on a good characterization of the null-space of $A^{(i)}$, which, while obvious for linear elasticity problems, is less clear for

Euler and Navier-Stokes systems.

In this work, inexact solvers for the partially assembled system are developed by replacing the action of the exact solvers corresponding to (5.7) with inexact solvers of the form:

$$\tilde{\mathbf{M}}^{(i)} = \begin{bmatrix} M^{(i)} & B^{(i)^T} \\ B^{(i)} & 0 \end{bmatrix}, \tag{5.8}$$

where $M^{(i)}$ is a nonsingular matrix. For now the procedure for obtaining $M^{(i)}$ has yet to be specified. However, since $M^{(i)}$ is nonsingular it admits an LU factorization $M^{(i)} = \tilde{L}^{(i)}\tilde{U}^{(i)}$. After simple algebraic manipulation the inexact local primal basis functions may be written as:

$$\tilde{\Psi} = \tilde{U}^{(i)^{-1}}\tilde{L}^{(i)^{-1}}B^{(i)^T}\left(B^{(i)}\tilde{U}^{(i)^{-1}}\tilde{L}^{(i)^{-1}}B^{(i)^T}\right)^{-1}, \tag{5.9}$$

$$\tilde{\Psi}^* = \tilde{L}^{(i)^{-T}}\tilde{U}^{(i)^{-T}}B^{(i)^T}\left(B^{(i)}\tilde{U}^{(i)^{-1}}\tilde{L}^{(i)^{-1}}B^{(i)^T}\right)^{-T}. \tag{5.10}$$

Similarly, the application of the inexact variant of (4.30) simplifies to

$$\tilde{M}^{(i)^{-1}}v^{(i)} = \tilde{U}^{(i)^{-1}}\tilde{L}^{(i)^{-1}}v^{(i)} - \tilde{\Psi}\left(B^{(i)}\tilde{U}^{(i)^{-1}}\tilde{L}^{(i)^{-1}}B^{(i)^T}\right)\tilde{\Psi}^{*^T}. \tag{5.11}$$

At this point $M^{(i)}$ has yet to be specified. In the remainder of the section, an inexact local solver is developed where $M^{(i)}$ is obtained from an inexact factorization of $A^{(i)}$. As $A^{(i)}$ is potentially singular, the exact factorization of $A^{(i)}$ may result in a zero pivot and the inverse $A^{(i)^{-1}}$ may not exist. Thus, initially it may seem inappropriate to use an inexact factorization of $A^{(i)}$ to generate $M^{(i)}$ as an exact factorization will result in the failure of the algorithm.

However, in the following it is shown that as long as some fill-in is dropped during the inexact factorization, the factorization will not fail due to a zero pivot. Additionally, the product of the resulting factors $\tilde{M}^{(i)} = \tilde{L}^{(i)}\tilde{U}^{(i)}$ is non-singular. The following lemmas extend the analysis of incomplete factorization with zero fill, ILU(0), in [24] to incomplete factorizations with arbitrary fill-in. This enables the proof of the desired result in Theorem 5.2.7.

Lemma 5.2.1. *Let $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ be an M-matrix with real entries. Let the elements* *of $B = (b_{ij})$ satisfy the relations*

$$a_{ij} \le b_{ij} \le 0 \quad \textit{for } i \ne j, \qquad \textit{and} \qquad 0 \le a_{ii} \le b_{ii}. \tag{5.12}$$

Then

1. *B is an M-matrix.*
2. *If B is a singular M-matrix, then A is a singular M-matrix.*
3. *If A is an irreducible M-matrix and if $b_{ij} = 0$ for some $i \ne j$ where $a_{ij} \ne 0$ then B is nonsingular*

Proof. See [24] Lemma 1. □

The exact LU factorization algorithm may be given recursively as:

$$\begin{aligned} A^0 &= A, \\ C^k &= A^{k-1} \\ A^k &= L^k C^k, \end{aligned} \tag{5.13}$$

where $k > 0$, and L^k is equal to the identity matrix except for the kth column which written row-wise is as follows:

$$\begin{bmatrix} 0 & 0 & \dots & 1 & -c^{k-1}_{k+1,k}/c^{k-1}_{kk} & \dots - c^{k-1}_{nk}/c^{k-1}_{kk} \end{bmatrix} \qquad \text{when } c^{k-1}_{kk} \ne 0,$$

otherwise L^k is the identity matrix. This gives the factorization $A = LU$ where $L = (L^1)^{-1}(L^2)^{-1}\dots(L^{n-1})^{-1}(L^n)^{-1}$ and $U = A^n$. The exact LU factorization algorithm fails for a singular matrix A, if $c^k_{kk} = 0$ and $c^k_{tk} \ne 0$ for $t > k$.

Let G_A denote the graph of any matrix $A = (a_{ij})$, consisting of the set of ordered pairs of integers (i, j), with $i \le n$, $j \le n$ such that $a_{ij} \ne 0$. Denote by $P_A = \{(i,j) : a_{ij} = 0, i \ne j\}$ the graph of the off-diagonal zero entries of A. Let G_M be the graph for the allowable fill in the inexact factors $\tilde{L}$ and $\tilde{U}$, while P_M is the graph of the off-diagonal zero entries in the inexact factors. The incomplete factorization algorithm is obtained from the inexact factorization by replacing (5.13) with:

$$\begin{aligned} A^0 &= A, \\ C^k &= A^{k-1} + R^k, && (5.14) \\ A^k &= L^k C^k, && (5.15) \end{aligned}$$

where

$$r_{ij}^k = \begin{cases} -a_{ij}^k & \text{if } (i,j) \in P_M, \\ 0 & \text{otherwise.} \end{cases} \tag{5.16}$$

Lemma 5.2.2. *At any step in the incomplete factorization algorithm, let A^{k-1} be an M-matrix, then C^k is also an M-matrix.*

Proof. From (5.14):

$$c_{ij}^k = \begin{cases} 0 & \text{if } (i,j) \in P_M, \\ a_{ij}^{k-1} & \text{otherwise.} \end{cases} \tag{5.17}$$

Hence

$$a_{ij}^{k-1} \leq c_{ij}^k \leq 0 \quad \text{for } i \neq j, \qquad \text{and} \qquad 0 \leq a_{ii}^{k-1} = c_{ii}^k. \tag{5.18}$$

Thus, from condition 1 of Lemma 5.2.1, C^k is an M-matrix. □

Corollary 5.2.3. *Let A^{k-1} be an irreducible singular M-matrix. If $R^k \neq 0$ then C^k is nonsingular.*

Proof. Given $(i,j) \in P_M$ such that $r_{ij}^k \neq 0$ then by definition $a_{ij}^{k-1} \neq 0$, $r_{ij}^k = -a_{ij}^{k-1}$ and $c_{ij}^k = 0$. Thus, from condition 2 of Lemma 5.2.1, C^k is a nonsingular M-matrix. □

Lemma 5.2.4. *Let C^k be an M-matrix. If the incomplete algorithm does not fail at (5.15) on step k, then A^k is an M-matrix.*

Proof. The proof follows directly from Theorem 2.1 of [91]. □

Lemma 5.2.5. *Let A be an M-matrix. If $a_{kk}^{k-1} = 0$ at step k of the incomplete factorization algorithm then A is a singular M-matrix.*

Proof. Since the incomplete factorization algorithm does not fail through steps $1, \dots, k-1$, A^{k-1} and C^k are M-matrices by Lemmas and . Without loss of generality, assume $k = 1$. For $i, j > k$, the Gaussian elimination step (5.2.2) gives:

$$a_{ij}^1 \;=\; c_{ij}^1 - c_{i1}^1 c_{1j}^1 / c_{11}^1, \qquad c_{11}^1 \neq 0. \tag{5.19}$$

Since $c_{tq}^1 \leq 0$ for $t \neq q$, $a_{ij}^1 \leq c_{ij}^1$. Since C^1 is an M-matrix all of its principle minors are non-negative, thus $c_{11}^1 c_{ii}^1 - c_{i1}^1 c_{1i}^1 \geq 0$. Hence

$$a_{ii}^1 = c_{ii}^1 - c_{i1}^1 c_{1i}^1 / c_{11}^1 \geq 0. \tag{5.20}$$

Thus, by condition 2 of Lemma 5.2.1 A^{k-1} is singular. The case for $c_{11}^1 = 0$ is trivially satisfied since L^1 is then the identity matrix. Backtracking to $A = A^0$ proves A is a singular M-matrix. □

Lemma 5.2.6. *Let A be an M-matrix, while for any set of indices, s, $A[s]$ denotes the block of a corresponding to the rows and columns s. Denote by $\langle k \rangle$ the indices $1, \dots, k$, and let $s = (s_1, \dots, s_k)$ be any subset of $\langle n \rangle$ for which $A[s]$ is singular and irreducible. If*

$$(t, p) \in P_M \qquad \forall t > s_k \text{ and } \forall p \in s, \tag{5.21}$$

then the incomplete factorization algorithm does not fail at any step.

Proof. The proof of Lemma 5.2.6 closely follows the proof of Theorem 4 in [24]. Assume that (5.15) fails at step k. Then $c_{kk}^k = 0$ and $c_{rk}^k \neq 0$ for some $r > k$. However, since the algorithm did work through $k-1$ steps, $A^{k-1}[\langle k \rangle]$ exists and has the form:

$$A^{k-1}[\langle k \rangle] \;=\; \begin{bmatrix} A^{k-1}[\langle k-1 \rangle] & * \\ 0 & a_{kk}^{k-1} \end{bmatrix}. \tag{5.22}$$

Since $a_{kk}^{k-1} = c_{kk}^k = 0$, $A^k[\langle k \rangle] = A^{k-1}[\langle k \rangle]$ are singular M-matrices.

Set $1 \leq s_1 < s_2 \dots < s_j = k$ to be the largest subset of $\langle k \rangle$ for which $A[s]$ is irreducible.

Since $s_j = k$ and $a_{kk}^{k-1} = 0$, then $A^{k-1}[s]$ is singular, and by Lemma 5.2.1 $A[s]$ is singular. Now since $c_{rk} \neq 0$, $(r,k) \notin P_M$ which contradicts (5.21). $\square$

Theorem 5.2.7. *Let A be an irreducible $n \times n$ M matrix. Let P_M satisfy the condition (5.21). Then $M = \tilde{L}\tilde{U}$ produced by the incomplete factorization algorithm is nonsingular provided $R = \sum_{k=1}^{N} R^k \neq 0$.*

Proof. Since P_M satisfies the condition (5.21), Lemma 5.2.6 ensures that the incomplete factorization algorithm will not fail. By construction $\tilde{L}$ is nonsingular, it remains to show that $\tilde{U}$ is nonsingular. Since $R \neq 0$, there exists k, for which $R^k \neq 0$. By Corollary 5.2.3 C^k is a non-singular M-matrix. Then $A^k, \ldots, A^n$ must be non-singular otherwise Lemma 5.2.2 is contradicted. Thus $\tilde{U} = A^n$ is non-singular. $\square$

Theorem 5.2.7 ensures that an inexact solver of the form (5.8) may be constructed through an incomplete factorization algorithm, provided the incomplete factorization algorithm is not exact. Since P_M may be chosen in such a manner that $G_A \cap P_M \neq \emptyset$, Theorem 5.2.7 applies equally to Jacobi and Gauss-Seidel local solvers, as these are specific instances of an inexact factorization.

5.2.3 One or Two matrix method

Both the inexact harmonic extensions and inexact constrained Neumann solves require an inexact factorization of all or part of the block matrices $A^{(i)}$. In this work two strategies are considered: a one-matrix and a two-matrix method. In the one-matrix method, the same inexact factorization is used for both the inexact harmonic extensions and the inexact constrained Neumann solve. In the two-matrix method, different factorizations are used for the harmonic extensions and constrained Neumann solves, requiring the computation and storage of two separate systems. The motivation behind considering the two matrix approach is that the performance of the incomplete factorization algorithm is strongly dependent on the ordering of the unknowns [17, 20, 45]. The ordering of unknowns is particularly important in the case of convection-dominated flows where ordering degrees of freedom along characteristic directions may significantly improve performance [45]. This work uses a reordering based on the block minimum discarded fill (MDF) algorithm presented in [103]. The MDF method orders the unknowns by choosing in a greedy manner the block which produces the least discarded fill-in. In the one-matrix approach, the MDF algorithm must be modified to ensure that the interior degrees of freedom are ordered before the interface degrees of freedom, potentially resulting in an ordering which has large discarded fill. On the other hand, the two-matrix approach allows for different orderings to be used for the harmonic extensions and the constrained Neumann solves, potentially improving the quality of the incomplete factorization.

5.3 Inexact BDDC with p-multigrid correction

While incomplete factorizations have been widely used for convection-dominated problems, incomplete factorizations do not control the low frequency error modes present in elliptic problems [91]. Multigrid methods, on the other hand, provide an efficient means of controlling low frequency error modes which extend throughout the domain. Multigrid methods have also been widely used for the solution of the Euler and Navier-Stokes equations [3, 46, 57, 81, 85, 88, 103]. This work uses a two-level method with a p-multigrid type coarse correction similar to that presented in [103]. The coarse space is obtained by projecting the higher-order solution to $p = 0$ or $p = 1$ and solving the coarse problem exactly.

5.3.1 Inexact Harmonic Extensions

As noted in the previous section the harmonic extensions involve the application of ${A_{II}^{(i)}}^{-1} A_{I\Gamma}^{(i)}$ and $A_{\Gamma I}^{(i)} {A_{II}^{(i)}}^{-1}$. The inexact application of the harmonic extensions using the coarse grid correction are given by:

$$\begin{aligned} {A_{II}^{(i)}}^{-1} A_{I\Gamma}^{(i)} &\approx {\tilde{U}_{II}^{(i)}}^{-1} \tilde{U}_{I\Gamma}^{(i)} + P_I^{(i)} {A_{0,II}^{(i)}}^{-1} {P_I^{(i)}}^T \left(A_{I\Gamma} - A_{II}^{(i)} {\tilde{U}_{II}^{(i)}}^{-1} \tilde{U}_{I\Gamma}^{(i)} \right), & (5.23) \\ A_{\Gamma I}^{(i)} {A_{II}^{(i)}}^{-1} &\approx A_{\Gamma I}^{(i)} P_I^{(i)} {A_{0,II}^{(i)}}^{-1} {P_I^{(i)}}^T + \tilde{L}_{\Gamma I}^{(i)} {\tilde{L}_{II}^{(i)}}^{-1} \left(I - A_{II}^{(i)} P_I^{(i)} {A_{0,II}^{(i)}}^{-1} {P_I^{(i)}}^T \right). & (5.24) \end{aligned}$$

where $P_I : \Lambda_I^{(i),p=0 \text{ or } 1} \rightarrow \Lambda_I^{(i)}$ projects the coarse interior degrees of freedom from $p = 0$ or $p = 1$ to the higher-order solutions, while $A_{0,II}^{(i)}$ is given by the Galerkin projection:

$$A_{0,II}^{(i)} = {P_I^{(i)}}^T A_{II}^{(i)} P_I^{(i)}. \qquad (5.25)$$

As in the case when using only the ILU factorization, the application of the harmonic extensions involves either a backwards substitution or a forward substitution with the inexact factorization.

5.3.2 Inexact Partially Assembled Solve

The inexact partially assembled solve requires the computation of constrained Neumann problems of the form (4.30). The inexact solver with coarse grid correction for the constrained Neumann problem may be written as:

$$\tilde{\mathbf{A}}^{(i)^{-1}} \approx \tilde{\mathbf{P}}^{(i)}\tilde{\mathbf{A}}_0^{(i)^{-1}}\tilde{\mathbf{P}}^{(i)^T} + \tilde{\mathbf{M}}^{(i)^{-1}}\left(I - \tilde{\mathbf{A}}\tilde{\mathbf{P}}^{(i)}\tilde{\mathbf{A}}_0^{(i)^{-1}}\tilde{\mathbf{P}}^{(i)^T}\right), \tag{5.26}$$

where

$$\tilde{\mathbf{P}}^{(i)} = \begin{bmatrix} P^{(i)} & 0 \\ 0 & I_\Pi \end{bmatrix} \quad \text{and} \quad \tilde{\mathbf{A}}_0^{(i)} = \tilde{\mathbf{P}}^{(i)^T}\tilde{\mathbf{A}}^{(i)}\tilde{\mathbf{P}}^{(i)} = \begin{bmatrix} P^{(i)^T}A^{(i)}P^{(i)} & B^{(i)^T}P^{(i)^T} \\ P^{(i)}B^{(i)} & 0 \end{bmatrix}, \tag{5.27}$$

and $P^{(i)} : \Lambda^{(i),p=0 \text{ or } 1} \rightarrow \Lambda^{(i)}$ projects all coarse interior of freedom on Ω_i from $p = 0$ or $p = 1$ to the higher-order solutions. Since the coarse grid correction exactly satisfies the constraint equation, the expression for the inexact constrained solves may be simplified, such that the inexact primal basis functions are given by:

$$\Psi^{(i)} \approx P^{(i)}\Psi_0^{(i)} - \tilde{M}^{(i)^{-1}}A^{(i)}\Psi_0^{(i)}, \tag{5.28}$$

$$\Psi^{*(i)} \approx P^{(i)}\Psi_0^{*(i)} - \tilde{M}^{(i)^{-T}}A^{(i)^T}\Psi_0^{*(i)}, \tag{5.29}$$

where

$$\begin{bmatrix} P^{(i)^T}A^{(i)}P^{(i)} & B^{(i)^T}P^{(i)^T} \\ P^{(i)}B^{(i)} & 0 \end{bmatrix}\begin{bmatrix} \Psi_0^{(i)} \\ * \end{bmatrix} = \begin{bmatrix} 0 \\ I_\Pi^{(i)} \end{bmatrix}, \tag{5.30}$$

$$\begin{bmatrix} P^{(i)^T}A^{(i)}P^{(i)} & B^{(i)^T}P^{(i)^T} \\ P^{(i)}B^{(i)} & 0 \end{bmatrix}\begin{bmatrix} \Psi_0^{*(i)} \\ * \end{bmatrix} = \begin{bmatrix} 0 \\ I_\Pi^{(i)} \end{bmatrix}. \tag{5.31}$$

Similarly,

$$\tilde{A}^{(i)^{-1}} \approx P^{(i)}\tilde{A}_0^{(i)^{-1}}P^{(i)^T} - \tilde{M}^{(i)^{-1}}A^{(i)}P^{(i)}\tilde{A}_0^{(i)^{-1}}P^{(i)^T}, \tag{5.32}$$

where $\tilde{A}_0^{(i)^{-1}}$ is defined by

$$\begin{bmatrix} P^{(i)^T}A^{(i)}P^{(i)} & B^{(i)^T}P^{(i)^T} \\ P^{(i)}B^{(i)} & 0 \end{bmatrix} \begin{bmatrix} \tilde{A}_0^{(i)^{-1}}v_0^{(i)} \\ * \end{bmatrix} = \begin{bmatrix} v_0^{(i)} \\ 0 \end{bmatrix}. \tag{5.33}$$

The approach for the inexact partially assembled solve presented in this section resembles most closely the approach taken in [48]. This approach differs from that taken in [72] or [124] where the multigrid scheme acts on the global partially assembled system. In particular, the two-level inexact BDDC method of Yano and Darmofal [124] requires two primal system solves corresponding to the inexact solve and the coarse grid solve. In the following section, numerical results are presented which show that both methods perform similarly in terms of number of iterations, however, the approach described here requires communication and computation of only a single primal solve for each application of the preconditioner.

5.4 Numerical Results

In this section, the performance of the inexact BDDC preconditioners is assessed for two 3D model problems. The first model problem is the 3D extension of the 2D Poisson problem (3.10) to the unit cube $\Omega = [0, 1]^3 \in \mathbb{R}^3$. Similarly, the second model problem is the extension of the 2D scalar advection-diffusion boundary layer problem (3.12) to the unit cube. The 3D model problems are solved on a set of unstructured tetrahedral meshes generated using TetGen [112]. The domain is partitioned into N subdomains of approximately n element using ParMetis [65]. The performance of different inexact solvers are assessed in term of the number of iterations required to converge the l_2 norm of the residual by a factor of 10^6. The simulations were performed on a Dell PowerEdge 1950 Linux cluster with 680 nodes, each with two Quad Core Intel Xeon 2.33GHz 64-bit processors and 8 GB of memory, connected with 10 Gb/sec Infiniband.

5.4.1 Inexact BDDC using Incomplete Factorizations

3D Poisson Problem

In the first set of numerical experiments the 3D Poisson problem is solved using the BDDC preconditioner with inexact solvers based on the block ILUT factorization. Table 5.1 gives the number of iterations varying N and n for both the one-matrix and two-matrix methods. A drop tolerance of 10^{-6} is used, while the allowable fill-in is varied from $0 \to \infty$. A fill-in of 0 implies the ILU(0) algorithm is used, while fill-in ∞ implies the exact BDDC algorithm is used.

(a) One-matrix Method

		$p=2$							$p=5$						
N	n	0	2	5	10	20	50	∞	0	2	5	10	20	50	∞
4	400	39	29	24	20	18	15	14	57	43	32	27	22	19	19
16	400	62	47	41	35	28	22	22	95	61	55	45	36	28	24
64	400	114	102	65	52	39	28	22	172	121	89	69	52	37	29
256	400	198	148	105	81	59	37	25	316	266	147	111	79	49	33
1024	400	314	356	165	124	88	49	27	501	486	232	172	118	76	37
64	100	63	38	28	23	20	19	19	119	87	58	45	34	26	26
64	200	87	56	41	31	24	22	22	150	110	79	61	45	30	30
64	400	114	102	65	52	39	28	22	172	121	89	69	52	37	29
64	800	137	95	73	58	43	29	25	218	279	114	90	68	38	33
64	1600	172	117	92	72	55	35	24	244	339	134	107	87	*	*

(b) Two-matrix Method

		$p=2$							$p=5$						
N	n	0	2	5	10	20	50	∞	0	2	5	10	20	50	∞
4	400	42	28	23	19	16	15	14	62	38	29	24	20	19	19
16	400	63	43	34	28	23	19	22	97	58	45	36	29	25	24
64	400	102	67	51	39	29	22	22	157	93	71	54	39	30	29
256	400	171	109	81	60	39	26	25	268	151	109	78	52	35	33
1024	400	273	171	124	88	55	29	27	435	242	178	124	73	39	37
64	100	63	38	28	23	20	19	19	100	55	40	31	27	26	26
64	200	87	56	41	31	24	22	22	135	78	57	42	33	30	30
64	400	102	67	51	39	29	22	22	157	93	71	54	39	30	29
64	800	137	95	73	58	43	29	25	210	130	99	76	56	38	33
64	1600	172	117	92	72	55	35	24	266	158	120	94	*	*	*

Table 5.1: Number of GMRES iterations for 3D Poisson problem using ILUT(τ,π) inexact solver, with $\tau = 10^{-6}$ and varying π

For fixed $n = 400$ with small allowable fill-in, π =0, 2 or 5, the inexact BDDC preconditioner performs poorly relative the exact BDDC method. The inexact BDDC algorithm is not scalable as the number of iterations grows with increasing number of subdomains. However, as the fill-in is allowed to increase, the performance of the inexact BDDC method approaches that for the exact BDDC method.

Unfortunately, the fill-in required in order to maintain the good performance of the exact BDDC preconditioner is dependent upon the number of elements per subdomain. For $N = 64$ and $n = 100$ a fill-in of $\pi = 10$ is sufficient to maintain the good performance of the exact BDDC algorithm. However, for fixed $N = 64$ and increasing number of elements per subdomain, increasing amount of fill-in is required in order to maintain good performance. In particular, for 1600 elements per subdomain, a fill-in of 50 is insufficient to maintain the good performance of the exact BDDC algorithm.

A key advantage of using an inexact solver is the reduced storage required for the factorization. For $p = 5$ and $n = 1600$ both the exact BDDC and the inexact BDDC with large fill-in fail as the memory required exceed the 1Gb/core of available memory. The failure of the BDDC algorithm due to insufficient memory is reported as * in Table 5.1. Using the two-matrix method the inexact BDDC algorithm fails for $\pi > 10$ while using the one-matrix method, a fill-in of up to $\pi = 20$ may be used without running out of memory. Comparing Tables 5.0(a) and 5.0(b) there is relatively little advantage of using the two matrix factorization approach over the single matrix approach. Thus, for this test case, using the one-matrix approach with larger fill-in is a more efficient approach than using the two-matrix factorization approach.

3D Advection-Diffusion Boundary Layer Problem

In the second numerical experiment, the 3D advection-diffusion boundary layer problem is solved on the isotropic unstructured meshes with varying μ. Tables 5.2 and 5.3 report the corresponding number of iteration for $\mu = 1$ and $\mu = 10^{-4}$ respectively. In the diffusion-dominated case, ($\mu = 1$) the behaviour is similar to that for the 3D Poisson problem. Namely, the performance of the inexact preconditioner improves with increasing allowable fill-in. However, increasing fill-in is required with increasing size of the subdomains. The advantage of the two-matrix method over the one-matrix approach is again relatively small, such that using the one matrix approach with more fill-in is a more efficient approach than

(a) One-matrix Method

		$p=2$							$p=5$						
N	n	0	2	5	10	20	50	∞	0	2	5	10	20	50	∞
4	400	35	27	22	18	15	12	12	49	37	27	22	18	15	14
16	400	56	43	36	30	24	17	15	81	53	46	37	29	21	18
64	400	107	93	58	45	34	23	17	154	105	75	57	42	29	22
256	400	182	135	96	72	51	30	19	276	230	125	94	52	33	24
1024	400	292	304	148	110	76	41	20	402	214	168	117	59	38	26
64	100	72	56	37	28	21	16	16	105	75	49	37	27	21	20
64	200	92	68	49	37	27	20	17	134	91	65	48	36	25	23
64	400	107	93	58	45	34	23	17	154	105	75	57	42	29	22
64	800	135	117	74	59	45	31	19	192	154	96	75	56	38	25
64	1600	153	130	91	72	54	35	18	215	161	114	90	65	*	*

(b) Two-matrix Method

		$p=2$							$p=5$						
N	n	0	2	5	10	20	50	∞	0	2	5	10	20	50	∞
4	400	37	25	19	16	14	12	12	53	32	24	20	16	15	14
16	400	58	39	30	24	19	15	15	83	49	38	29	23	18	18
64	400	94	61	45	34	24	17	17	138	79	58	42	30	22	22
256	400	157	97	70	50	32	20	19	231	127	91	63	38	24	24
1024	400	246	149	105	73	42	21	20	373	198	138	94	54	27	26
64	100	57	34	24	18	16	16	16	85	45	32	24	21	20	20
64	200	74	47	34	25	19	17	17	109	62	44	33	25	23	23
64	400	94	61	45	34	24	17	17	138	79	58	42	30	22	22
64	800	124	82	62	48	35	23	19	183	106	80	60	43	28	25
64	1600	156	104	80	62	45	28	18	224	132	100	76	*	*	*

Table 5.2: Number of GMRES iterations for 3D advection-diffusion boundary layer problem with $\mu = 1$ on isotropic unstructured mesh using ILUT(τ,π) inexact solver, with $\tau = 10^{-6}$ and varying π

using the two-matrix method.

(a) One-matrix Method

		$p=2$							$p=5$						
N	n	0	2	5	10	20	50	∞	0	2	5	10	20	50	∞
4	400	16	10	8	8	8	8	8	15	9	7	7	7	7	7
16	400	20	11	10	10	9	9	9	18	11	9	9	9	9	9
64	400	22	13	11	10	10	10	10	21	12	10	9	9	9	9
256	400	30	17	14	14	13	13	13	31	18	15	15	14	13	13
1024	400	42	23	19	18	17	17	17	44	24	20	19	19	18	18
64	100	19	12	11	10	10	10	10	18	11	10	10	10	10	10
64	200	20	12	11	10	10	10	10	19	12	10	9	9	9	9
64	400	22	13	11	10	10	10	10	21	12	10	9	9	9	9
64	800	25	15	12	11	11	11	11	24	14	12	11	10	10	10
64	1600	27	16	13	12	12	12	12	26	16	13	12	12	*	*

(b) Two-matrix Method

		$p=2$							$p=5$						
N	n	0	2	5	10	20	50	∞	0	2	5	10	20	50	∞
4	400	8	8	8	8	8	8	8	7	7	7	7	7	7	7
16	400	10	10	10	9	9	9	9	9	9	9	9	9	9	9
64	400	11	11	10	10	10	10	10	11	10	10	10	9	9	9
256	400	15	13	13	13	13	13	13	15	14	13	13	13	13	13
1024	400	19	18	17	17	17	17	17	23	19	19	19	19	18	18
64	100	11	10	10	10	10	10	10	10	10	10	10	10	10	10
64	200	10	10	10	10	10	10	10	10	9	9	9	9	9	9
64	400	11	11	10	10	10	10	10	11	10	10	10	9	9	9
64	800	12	11	11	11	11	11	11	11	11	10	10	10	10	10
64	1600	13	12	12	12	12	12	12	13	12	12	12	*	*	*

Table 5.3: Number of GMRES iterations for 3D advection-diffusion boundary layer problem with $\mu = 10^{-4}$ on isotropic unstructured mesh using ILUT(τ,π) inexact solver, with $\tau = 10^{-6}$ and varying π

In the convection-dominated case, ($\mu = 10^{-4}$), the number of iterations required to converge the BDDC algorithm is approximately half of the number of subdomains through which the characteristics pass. The incomplete factorization with MDF reordering is able to capture the strong coupling along characteristics. Thus much less fill-in is required compared to the diffusion dominated case in order to maintain the good performance of the exact BDDC algorithm. The ordering of the degrees of freedom in the ILUT factorization is much more important than for the diffusion-dominated case. Thus, the two matrix method performs much better than the one-matrix method. In particular, the iteration count for the two-matrix method with zero fill is within 1 or 2 iterations of the exact BDDC method. Additionally, the fill-in required to maintain good performance using the two-matrix method does not appear to grow with increasing n or N. On the other hand, significantly more fill-in is required to maintain good performance using the one-matrix method. Additionally, the amount of fill-in required increases with both N and n. Thus, for this convection-dominated case, it is more efficient to use the two-matrix approach, than using the one-matrix approach with additional fill-in.

5.4.2 Inexact BDDC with p-multigrid correction

3D Poisson Problem

In the third numerical experiment the 3D Poisson problem is solved using the inexact BDDC algorithm with coarse grid correction. Table 5.4 gives the number of iterations using both one- and two-matrix methods with $p = 0$ coarse grid correction. The use of the coarse grid correction significantly improves the performance relative to only using the ILUT local solver. As with the exact BDDC algorithm, the number of iterations required to converge the inexact BDDC algorithm grows slowly with increasing subdomain size, n, for fixed $N = 64$. Similarly, using either exact or inexact BDDC the number of iterations does not grow significantly as the number of subdomains N increase for fixed $n = 400$. The number of iterations required to converge using the inexact BDDC algorithm appears to be a constant multiple of the number of iterations for the exact BDDC algorithm. In particular, using ILU(0) the number of iterations using the inexact BDDC method is approximately twice that of using the exact BDDC method, with the performance of the inexact BDDC algorithm improving as the fill-in is allowed to increase. As was the case when using only the ILUT inexact solver, the two-matrix method does not perform better than the one-matrix method, and in some cases results in a larger number of iterations. Thus for this case the use of the two-matrix method is not warranted.

Table 5.5 gives the number of iterations using both one- and two-matrix methods with $p = 1$ coarse grid correction. The performance trends using the $p = 1$ correction is similar to that using the $p = 0$ correction. Namely, the number of iterations grows slowly with N and n with the number of iterations using the inexact BDDC appearing to be a constant multiple of the number of iterations for the exact BDDC algorithm. For $p = 2$ the $p = 1$ coarse

(a) One-matrix Method

		$p = 2$				$p = 5$			
N	n	0	2	5	∞	0	2	5	∞
4	400	22	21	19	14	36	30	26	19
16	400	30	27	25	22	46	37	36	24
64	400	37	35	30	22	65	53	44	29
256	400	45	41	35	25	79	67	51	33
1024	400	48	42	37	27	90	85	56	37
64	100	33	29	25	19	61	45	37	26
64	200	37	35	29	22	65	54	43	30
64	400	37	35	30	22	65	53	44	29
64	800	41	39	34	25	70	60	50	33
64	1600	41	46	36	24	70	61	52	*

(b) Two-matrix Method

		$p = 2$				$p = 5$			
N	n	0	2	5	∞	0	2	5	∞
4	400	26	21	18	14	42	31	26	19
16	400	32	27	24	22	51	40	34	24
64	400	37	32	28	22	62	48	41	29
256	400	43	36	32	25	73	55	47	33
1024	400	46	39	35	27	81	60	51	37
100	64	32	25	22	19	55	39	32	26
200	64	37	30	27	22	62	46	39	30
400	64	37	32	28	22	62	48	41	29
800	64	42	37	34	25	69	55	48	33
1600	64	44	38	36	24	71	58	52	*

Table 5.4: Number of GMRES iterations for 3D Poisson problem using ILUT with $p = 0$ coarse grid correction, with $\tau = 10^{-6}$ and varying π

(a) One-matrix Method

		$p = 2$				$p = 5$			
N	n	0	2	5	∞	0	2	5	∞
4	400	17	16	15	14	29	24	21	19
16	400	23	20	19	22	42	31	28	24
64	400	26	25	22	22	50	42	33	29
256	400	31	30	26	25	59	58	38	33
1024	400	33	31	28	27	65	54	42	37
64	100	24	22	20	19	47	37	30	26
64	200	27	25	23	22	51	42	34	30
64	400	26	25	22	22	50	42	33	29
64	800	30	28	26	25	55	46	37	33
64	1600	30	30	25	24	54	47	36	*

(b) Two-matrix Method

		$p = 2$				$p = 5$			
N	n	0	2	5	∞	0	2	5	∞
4	400	17	15	15	14	28	23	20	19
16	400	22	20	19	22	37	29	26	24
64	400	25	23	22	22	43	35	32	29
256	400	29	26	26	25	52	40	36	33
1024	400	31	29	28	27	57	44	40	37
64	100	22	20	19	19	38	31	28	26
64	200	25	23	22	22	44	35	32	30
64	400	25	23	22	22	43	35	32	29
64	800	28	26	25	25	48	39	36	33
64	1600	28	26	25	24	51	39	35	*

Table 5.5: Number of GMRES iterations for 3D Poisson problem using ILUT(τ,π) with $p = 1$ coarse grid correction, with $\tau = 10^{-6}$ and varying π

grid correction provides a very strong local solver, such that even using ILU(0) the number of iterations required to converge using the inexact BDDC method is within 4 iterations of the exact BDDC method. For $p = 5$, using the $p = 1$ coarse grid correction provides some improvement over the $p = 0$ coarse grid correction, though this is less significant than for $p = 2$. As with previous cases, there is a slight performance improvement using the two-matrix method as opposed to the one-matrix method, however the performance gain is not sufficient to warrant the additional memory required for the two-matrix method.

		$p=2$				$p=5$			
N	n	0	2	5	∞	0	2	5	∞
4	400	17	15	15	14	28	23	20	19
16	400	22	20	19	22	37	29	26	24
64	400	25	23	22	22	43	35	32	29
256	400	29	27	26	25	52	40	36	33
1024	400	31	29	28	27	56	44	40	37
64	100	22	20	19	19	38	30	28	26
64	200	25	23	22	22	44	35	32	30
64	400	25	23	22	22	43	35	32	29
64	800	28	26	25	25	48	39	35	33
64	1600	28	26	25	24	51	39	35	*

Table 5.6: Number of GMRES iterations for 3D Poisson problem using ILUT(τ,π) with $p = 1$ coarse grid correction applied to global partially assembled problem, with $\tau = 10^{-6}$ and varying π

Table 5.6 gives the iteration count using the two-matrix method with the coarse grid correction applied to the global problem as in [124]. Comparing with Table 5.5, both coarse grid corrections schemes give similar iteration counts. Thus, the coarse grid correction scheme presented in Section 5.3 is preferred as this scheme requires only a single global primal solve for each application of the preconditioner.

3D Advection-Diffusion Boundary Layer Problem

In the next numerical experiment the performance of the inexact BDDC method with coarse grid correction is assessed for the 3D scalar boundary layer problem. Tables 5.7 and 5.8 report, respectively, the number of iterations using the inexact BDDC with $p = 0$ and $p = 1$ coarse grid corrections with $\mu = 1$. For this diffusion-dominated case the behaviour of the inexact BDDC is similar to that for the 3D Poisson problem. Namely, even using little or no additional fill-in, good performance of the inexact BDDC preconditioner is seen over a large range of N and n. As with the 3D Poisson problem, the two-matrix approach is not advocated as both one- and two-matrix approaches perform similarly.

(a) One-matrix Method

		$p = 2$				$p = 5$			
N	n	0	2	5	∞	0	2	5	∞
4	400	16	15	14	12	24	22	19	14
16	400	22	20	18	15	31	26	24	18
64	400	26	27	21	17	41	38	29	22
256	400	29	29	24	19	46	37	33	24
1024	400	30	73	25	20	51	45	35	26
64	100	24	22	20	16	40	31	25	20
64	200	26	27	21	17	43	38	29	23
64	400	26	27	21	17	41	38	29	22
64	800	28	28	24	19	44	39	33	25
64	1600	27	33	23	18	43	39	32	*

(b) Two-matrix Method

		$p = 2$				$p = 5$			
N	n	0	2	5	∞	0	2	5	∞
4	400	19	16	14	12	29	21	18	14
16	400	23	20	18	15	35	27	23	18
64	400	26	22	20	17	41	31	27	22
256	400	29	25	22	19	46	35	29	24
1024	400	31	26	24	20	50	38	33	26
64	100	22	19	17	16	37	27	23	20
64	200	24	22	20	17	41	31	27	23
64	400	26	22	20	17	41	31	27	22
64	800	29	25	24	19	44	36	32	25
64	1600	29	25	23	18	44	36	32	*

Table 5.7: Number of GMRES iterations for 3D advection-diffusion boundary layer problem with $\mu = 1$, using ILUT(τ,π) with $p = 0$ coarse grid correction, with $\tau = 10^{-6}$ and varying π

Tables 5.9 and 5.10 give the number of iterations using the inexact BDDC with $p = 0$ and $p = 1$ coarse grid corrections respectively for the 3D advection-diffusion problem with $\mu = 10^{-4}$. In this advection-dominated case the coarse grid correction does not provide any additional benefit over the inexact BDDC using only ILUT. As with the ILUT preconditioner without p-multigrid correction, the ordering of the degrees of freedom plays a significant role in the performance of the algorithm, such that the two-matrix approach

(a) One-matrix Method

		$p = 2$				$p = 5$			
N	n	0	2	5	∞	0	2	5	∞
4	400	13	12	12	12	20	17	15	14
16	400	17	15	15	15	29	22	19	18
64	400	19	19	17	17	32	30	24	22
256	400	22	23	19	19	37	32	26	24
1024	400	23	22	20	20	40	38	28	26
64	100	18	17	16	16	31	26	22	20
64	200	20	19	17	17	33	30	24	23
64	400	19	19	17	17	32	30	24	22
64	800	22	21	19	19	37	31	26	25
64	1600	21	24	18	18	33	28	24	*

(b) Two-matrix Method

		$p = 2$				$p = 5$			
N	n	0	2	5	∞	0	2	5	∞
4	400	13	12	12	12	20	16	15	14
16	400	16	15	15	15	26	20	19	18
64	400	19	17	17	17	30	24	22	22
256	400	21	19	19	19	33	27	25	24
1024	400	22	21	20	20	36	29	27	26
64	100	17	16	16	16	27	22	21	20
64	200	19	17	17	17	30	25	23	23
64	400	19	17	17	17	30	24	22	22
64	800	21	20	19	19	33	27	25	25
64	1600	20	18	18	18	31	26	23	*

Table 5.8: Number of GMRES iterations for 3D advection-diffusion boundary layer problem with $\mu = 1$, using ILUT(τ,π) with $p = 1$ coarse grid correction, with $\tau = 10^{-6}$ and varying π

performs significantly better than the one-matrix approach.

(a) One-matrix Method

		$p=2$				$p=5$			
N	n	0	2	5	∞	0	2	5	∞
4	400	14	9	8	8	15	10	8	7
16	400	16	11	10	9	16	10	9	9
64	400	19	13	11	10	19	12	10	9
256	400	25	16	14	13	25	17	14	13
1024	400	32	22	20	17	35	23	20	18
64	100	16	11	11	10	16	11	10	10
64	200	18	12	10	10	18	12	10	9
64	400	19	13	11	10	19	12	10	9
64	800	21	14	12	11	22	13	11	10
64	1600	23	15	13	12	24	15	12	*

(b) Two-matrix Method

		$p=2$				$p=5$			
N	n	0	2	5	∞	0	2	5	∞
4	400	8	8	8	8	8	7	7	7
16	400	10	10	9	9	9	9	9	9
64	400	11	10	10	10	11	10	10	9
256	400	14	13	13	13	15	14	13	13
1024	400	18	17	17	17	22	19	19	18
64	100	11	10	10	10	10	10	10	10
64	200	10	10	10	10	10	9	9	9
64	400	11	10	10	10	11	10	10	9
64	800	11	11	11	11	12	11	11	10
64	1600	13	12	12	12	13	12	12	*

Table 5.9: Number of GMRES iterations for 3D advection-diffusion boundary layer problem with $\mu = 10^{-4}$, using ILUT(τ,π) with $p = 0$ coarse grid correction, with $\tau = 10^{-6}$ and varying π

While iteration counts may provide an estimate of how well the inexact BDDC methods perform relative the exact BDDC method, the most important metric for evaluating the performance of the inexact solver is the CPU time taken. Figures 5-1 and 5-2 plot the CPU time required to solve the 3D advection-diffusion boundary layer problem with $\mu = 1$ and $\mu = 10^{-4}$ for $p = 5$ and $n = 400$ using several different local solvers using the two-matrix method.

For $\mu = 1$ the weak scalability of the exact BDDC method is reflected in the reported CPU time as the simulation time does not grow significantly with the number of processes.

(a) One-matrix Method

		$p=2$				$p=5$			
N	n	0	2	5	∞	0	2	5	∞
4	400	11	9	9	8	13	9	8	7
16	400	13	10	10	9	15	10	9	9
64	400	15	11	11	10	18	11	10	9
256	400	17	14	14	13	21	15	13	13
1024	400	20	18	17	17	27	20	19	18
64	100	13	11	11	10	15	10	10	10
64	200	14	11	10	10	16	11	9	9
64	400	15	11	11	10	18	11	10	9
64	800	16	12	11	11	20	12	11	10
64	1600	16	13	12	12	20	13	12	*

(b) Two-matrix Method

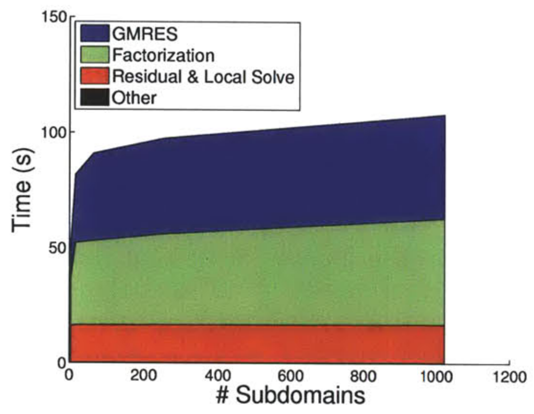
		$p=2$				$p=5$			
N	n	0	2	5	∞	0	2	5	∞
4	400	8	8	8	8	7	7	7	7
16	400	10	10	9	9	9	9	9	9
64	400	11	10	10	10	11	10	10	9
256	400	14	13	13	13	14	14	13	13
1024	400	17	17	17	17	19	18	18	18
64	100	10	10	10	10	10	10	10	10
64	200	10	10	10	10	10	9	9	9
64	400	11	10	10	10	11	10	10	9
64	800	11	11	11	11	11	10	10	10
64	1600	12	12	12	12	12	12	12	*

Table 5.10: Number of GMRES iterations for 3D advection-diffusion boundary layer problem with $\mu = 10^{-4}$, using ILUT(τ,π) with $p = 1$ coarse grid correction, with $\tau = 10^{-6}$ and varying π

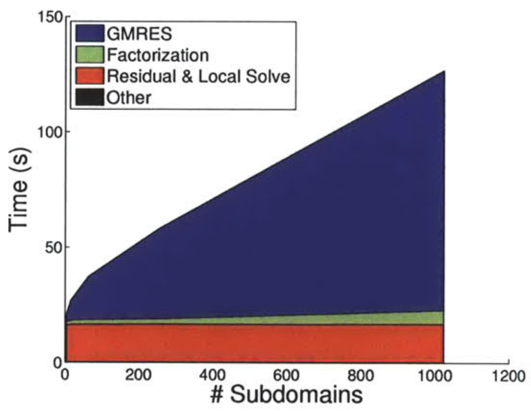
In particular, the CPU time for solving the problem on 1024 processor is only twice that for solving a 256 times smaller problem on 4 processors. However, when the exact BDDC algorithm is used, a significant portion of the run time is spent on the factorization and GMRES solve. Using the inexact BDDC algorithm with ILU(0) the cost of the factorization is significantly reduced. However, since this algorithm is not scalable, the cost of the GMRES solve grows with increasing number of subdomains, such that for large numbers of processors, the total CPU time using the inexact solver is greater than using the exact solver. However, when a $p = 0$ or $p = 1$ coarse grid correction is used, the inexact BDDC algorithm remains scalable, and the cost of both factorization and GMRES solve is lower than using the exact BDDC algorithm. In this case the $p = 1$ coarse grid correction performs slightly better than the $p = 0$ correction in terms of CPU time, as the increase in cost of the coarse grid correction is offset by fewer GMRES iterations.

For $\mu = 10^{-4}$ the coarse grid correction is not necessary to obtain good performance. Thus the inexact BDDC using only ILU(0) performs similarly to the case when a $p = 0$ or $p = 1$ coarse grid correction is applied. As with $\mu = 1$ the use of the inexact solver reduces the cost of factorization and linear solve.

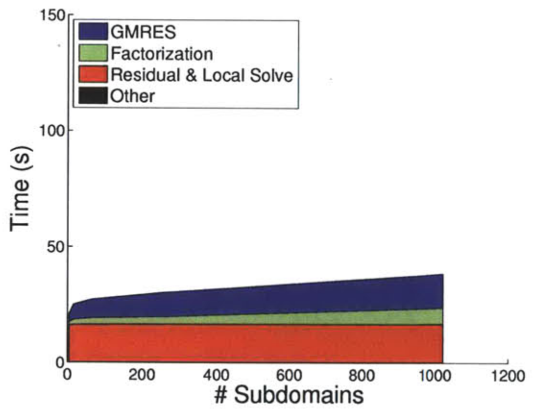
In the final numerical experiment, the performance of the inexact BDDC preconditioner is assessed for the 3D advection-diffusion problem with $\mu = 10^{-4}$ using an unstructured anisotropic mesh. The anisotropic mesh is generated from the unstructured isotropic 3D meshes used previously in this section, by using an exponential scaling of the y-coordinate such that the aspect ratio of the elements on the lower surface are approximately $\sqrt{Pe}$. Table 5.11 reports the number of iterations required to converge the l_2 residual by a factor of 10^3 using the inexact BDDC algorithm with $p = 1$ coarse grid correction. For this anisotropic unstructured mesh case, the performance of the BDDC algorithm degrades relative to the isotropic case. Additionally, more fill-in is required in order for the inexact BDDC method to maintain the performance of the exact BDDC preconditioner. In particular, for small amount of fill-in the inexact BDDC algorithm is unable to converge in 500 GMRES iterations (denoted by ** in Table 5.11). While diffusive effect are more important in this anisotropic case, than in the isotropic case, the ordering of the degrees of freedom remains important, such that the two-matrix method performs better than the one-matrix method.



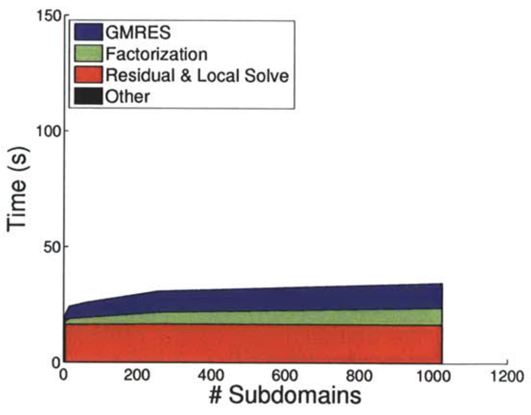
(a) Exact BDDC



(b) ILU(0) Two-Matrix

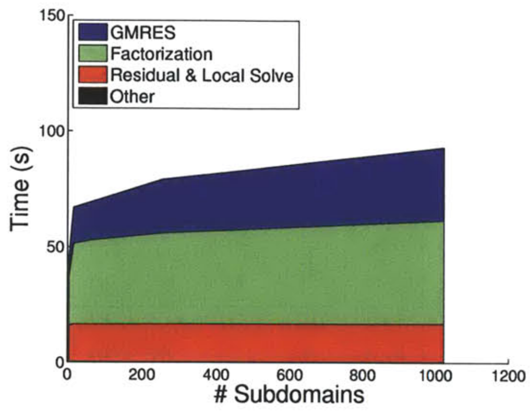


(c) ILU(0) Two-Matrix, $p = 0$ correction

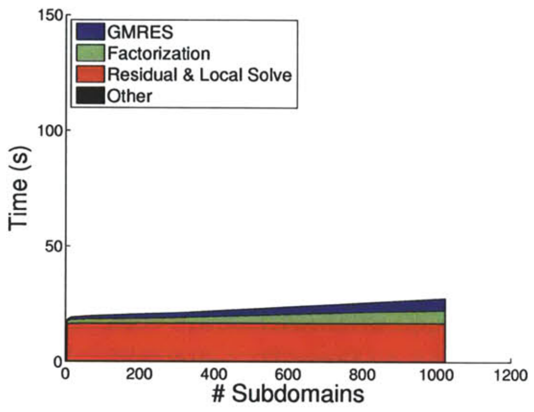


(d) ILU(0) Two-Matrix, $p = 1$ correction

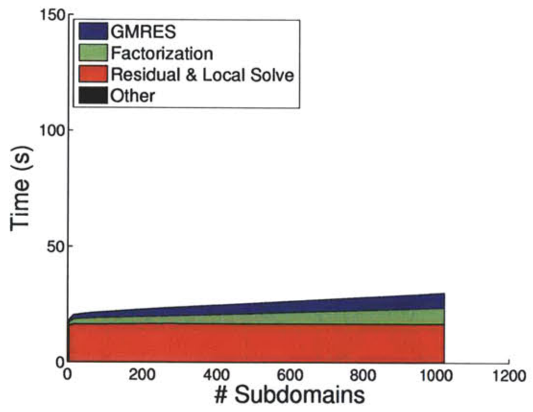
Figure 5-1: CPU time for 3D advection-diffusion boundary layer problem with $\mu = 1$, and $n = 400$ on isotropic unstructured mesh



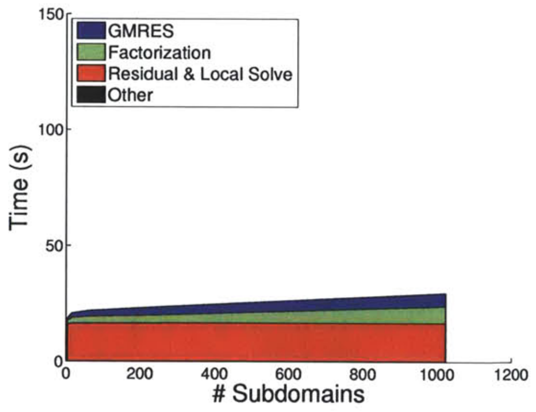
(a) Exact BDDC



(b) ILU(0)



(c) ILU(0) with $p = 0$ correction



(d) ILU(0) with $p = 1$ correction

Figure 5-2: CPU time for 3D advection-diffusion boundary layer problem with $\mu = 10^{-4}$, and $n = 400$ on isotropic unstructured mesh

(a) One-matrix Method

N	n	$p=2$						$p=5$					
		0	2	5	10	20	∞	0	2	5	10	20	∞
4	400	5	5	4	4	4	4	8	9	7	6	6	6
16	400	47	31	11	11	11	11	57	52	40	19	19	19
64	400	464	316	81	22	22	22	**	**	**	51	39	39
256	400	**	**	**	179	34	33	**	**	**	**	80	64
64	100	39	57	13	13	14	14	114	98	66	23	22	22
64	200	180	202	35	16	15	15	460	305	171	30	29	29
64	400	464	316	81	22	22	22	**	**	**	51	39	39
64	800	**	**	124	23	22	22	**	**	**	82	41	42

(b) Two-matrix Method

N	n	$p=2$						$p=5$					
		0	2	5	10	20	∞	0	2	5	10	20	∞
4	400	5	4	4	4	4	4	8	7	7	6	6	6
16	400	12	12	11	11	11	11	33	32	19	19	19	19
64	400	81	32	22	22	22	22	486	481	59	39	39	39
256	400	**	**	275	33	33	33	**	**	**	141	64	64
64	100	15	14	14	14	14	14	46	35	23	22	22	22
64	200	40	48	15	15	15	15	103	88	29	29	29	29
64	400	81	32	22	22	22	22	486	481	59	39	39	39
64	800	**	227	24	22	22	22	**	**	153	58	42	42

Table 5.11: Number of GMRES iterations for 3D advection-diffusion boundary layer problem with $\mu = 10^{-4}$, using ILUT(τ,π) with $p = 1$ coarse grid correction, with $\tau = 10^{-6}$ and varying π

Chapter 6

Euler and Navier-Stokes Systems

In this chapter the performance of the domain decomposition preconditioners are analyzed for the Euler and Navier-Stokes equations. Section 6.1 presents the linearized Euler equations and a change of variables used throughout this chapter. In Section 6.2, a one dimensional analysis is performed in order to demonstrate the effects of boundary conditions on the performance of domain decomposition preconditioners. Section 6.3 presents Fourier analysis of the two-dimensional Euler equations in order to estimate the performance of the BDDC preconditioner. In Section 6.4, an optimal Robin-Robin interface condition is derived for the Euler system. Finally, Section 6.5 presents numerical results for several model problems.

6.1 Linearized Euler Equations

Consider the semi-discrete two-dimensional linearized Euler equations using primitive variables:

$$\frac{u_k}{\Delta t} + A_{ikl} u_{l,x_i} = f_k, \tag{6.1}$$

with state vector $u = \begin{bmatrix} \delta\rho & \delta v_1 & \delta v_2 & \delta P \end{bmatrix}^T$ and

$$A_1 = \begin{bmatrix} v_1 & \rho & 0 & 0 \\ 0 & v_1 & 0 & 0 \\ 0 & 0 & v_1 & \frac{1}{\rho} \\ 0 & \rho c^2 & 0 & v_1 \end{bmatrix} \quad \text{and} \quad A_2 = \begin{bmatrix} v_2 & 0 & \rho & 0 \\ 0 & v_2 & 0 & \frac{1}{\rho} \\ 0 & 0 & v_2 & 0 \\ 0 & 0 & \rho c^2 & v_2 \end{bmatrix}, \tag{6.2}$$

where $c = \sqrt{\frac{\gamma P}{\rho}}$ is the speed of sound. The flow is assumed to be subsonic, such that $0 < |v| < c$. As opposed to working directly with primitive variables u, the linearized Euler system (6.1) is written in terms of non-dimensional characteristic variables, W, with:

$$W = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\left(\frac{\delta P}{\rho c^2} - \frac{\delta v_1}{c}\right) \\ \frac{1}{2}\left(\frac{\delta P}{\rho c^2} + \frac{\delta v_1}{c}\right) \\ \frac{1}{\sqrt{2}}\left(\frac{\delta P}{\rho c^2} - \frac{\delta \rho}{\rho}\right) \\ \frac{1}{\sqrt{2}}\frac{\delta v_2}{c} \end{bmatrix}, \tag{6.3}$$

such that, non-dimensionalized by the speed of sound, (6.1) simplifies to

$$\frac{W}{c\Delta t} + \Lambda\frac{\partial W}{\partial x} + \bar{A}_2\frac{\partial W}{\partial y} = \bar{f}, \tag{6.4}$$

with

$$\Lambda = \begin{bmatrix} M_x - 1 & 0 & 0 & 0 \\ 0 & M_x + 1 & 0 & 0 \\ 0 & 0 & M_x & 0 \\ 0 & 0 & 0 & M_x \end{bmatrix} \quad \bar{A}_2 = \begin{bmatrix} M_y & 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & M_y & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & M_y & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & M_y \end{bmatrix}, \tag{6.5}$$

and $\bar{f}$ is the source in the transformed variables. The change of variables allows the entropy equation (corresponding to w_3) to be decoupled from the other three state variables. However, while the change of variables diagonalizes A_1, a similarity transformation cannot, in general, simultaneously diagonalize both A_1 and A_2 [108]. Thus, the remaining state equations cannot be decoupled.

6.2 1D Analysis

The performance of the domain decomposition preconditioners for coupled systems of equations is significantly more complicated than in the case of a scalar equation. In the scalar case, a purely hyperbolic advection problem converges in a number of iterations equal to the number of subdomains through which the characteristics pass [106]. In [106], it is shown that in one dimension this property is preserved for coupled hyperbolic systems of equations when characteristic boundary conditions are imposed on all subdomain boundaries. This result is trivially obtained for the Euler system, as (6.4) simplifies to

$$\frac{W}{c\Delta t} + \Lambda \frac{\partial W}{\partial x} = \bar{f}. \tag{6.6}$$

Thus, the three state equations decouple into three scalar advection equations, provided coupling is not introduced through the boundary conditions. In this section, the analysis is extended to the case where boundary conditions other than characteristics are imposed, which leads to coupling between different state equations. A simple additive Schwarz algorithm without overlap is considered to solve (6.1). However, it is shown in Appendix A that, an iteration of the Robin-Robin algorithm applied to the Euler system is essentially equivalent to two iterations of the Schwarz method. Thus, the trends observed in this section apply equally to the Robin-Robin family of algorithms.

Consider a partitioning of the domain $\Omega = [0,1]$ into N subdomains $\Omega_i = [x_i, x_{i+1}]$ of size $H_i = 1/N$. The additive Schwarz algorithm updates the solution $W_k^{(i)}$ on subdomain i at iteration k by iteratively solving on each subdomain:

$$\left\{ \begin{array}{rcll} \frac{1}{\Delta t} W_k^{(i)} + \Lambda \frac{\partial W_k^{(i)}}{\partial x} & = & \bar{f} & \text{in } \Omega_i, \\ B_{Left}^{(i)} W_k^{(i)} & = & B_{Left}^{(i)} W_{k-1}^{(i-1)} & \text{at } x = x_i, \\ B_{Right}^{(i)} W_k^{(i)} & = & B_{Right}^{(i)} W_{k-1}^{(i+1)} & \text{at } x = x_{i+1}. \end{array} \right. \tag{6.7}$$

where $B_{Right}^{(i)}$ and $B_{Left}^{(i)}$ respectively enforce the outflow and inflow boundary conditions on Ω_i. At domain boundaries $x = 0$ and $x = x_{N+1}$, $B_{Left}^{(1)} W^{(0)}$ and $B_{Right}^{(N)} W^{(N+1)}$ denote the desired boundary quantities. Define

$$B^{(i)} = \begin{bmatrix} B^{(i)}_{Left} \\ B^{(i)}_{Right} \end{bmatrix} = \begin{bmatrix} B_{Left,Up} & B_{Left,Down} \\ B_{Right,Up} & B_{Right,Down} \end{bmatrix}, \qquad (6.8)$$

where subscripts *Up* and *Down* denote, respectively, upstream or downstream traveling characteristic variables. The boundary conditions are well posed if $B_{Left,Down}$ and $B_{Right,Up}$ are nonsingular. Additionally, an inflow boundary condition is said to be non-reflecting if $B_{Left,Up} = 0$. Similarly, an outflow boundary conditions is non-reflecting if $B_{Right,Down} = 0$. Table 6.1 lists the outflow and inflow boundary conditions considered.

(a) Outflow BCs

Outflow BCs	Reflecting
Upstream acoustic	no
Upstream Riemann invariant	yes
Pressure	yes

(b) Inflow BCs

Inflow BCs	Reflecting
Downstream acoustic and entropy	no
Entropy and downstream Riemann invariant	no
Entropy and total enthalpy	yes
Total pressure and total temperature	yes

Table 6.1: Boundary Conditions

Due to the linearity of the hyperbolic system (6.7), it is possible to consider only the error equation, and solve for the decay of the error $e^{(i)}_k = W^{(i)} - W^{(i)}_k$. The error function on subdomain Ω_i has the form:

$$e^{(i)}_{k,j} = \alpha^{(i)}_{k,j} e^{-\lambda_j (x - x_i)}, \qquad (6.9)$$

where $\alpha^{(i)}_{k,j}$ are coefficients for each mode, while λ_j are eigenvalues:

$$\lambda_1 = \frac{1}{(M_x - 1)c\Delta t} \qquad \lambda_2 = \frac{1}{(M_x + 1)c\Delta t} \qquad \lambda_3 = \frac{1}{M_x c\Delta t}. \qquad (6.10)$$

The additive Schwarz algorithm, (6.7), may be written as an iteration for the error coefficients

$\alpha_k = T\alpha_{k-1}$. In particular the iteration matrix has the form:

$$\begin{bmatrix} \alpha_k^{(1)} \\ \alpha_k^{(2)} \\ \vdots \\ \alpha_k^{(N-1)} \\ \alpha_k^{(N)} \end{bmatrix} = \begin{bmatrix} 0 & T_+^{(1)} & & & \mathbf{0} \\ T_-^{(2)} & 0 & T_+^{(2)} & & \\ & \ddots & \ddots & \ddots & \\ & & T_-^{(N-1)} & 0 & T_+^{(N-1)} \\ \mathbf{0} & & & T_-^{(N)} & 0 \end{bmatrix} \begin{bmatrix} \alpha_{k-1}^{(1)} \\ \alpha_{k-1}^{(2)} \\ \vdots \\ \alpha_{k-1}^{(N-1)} \\ \alpha_{k-1}^{(N)} \end{bmatrix}, \quad (6.11)$$

where $T_-^{(i)}$ and $T_+^{(i)}$ are determined by the interface conditions as:

$$T_-^{(i)} = \begin{bmatrix} B_{Right}^{(i)} E^{(i)} \\ B_{Left}^{(i)} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ B_{Left}^{(i)} \end{bmatrix} \qquad T_+^{(i)} = \begin{bmatrix} B_{Right}^{(i)} E^{(i)} \\ B_{Left}^{(i)} \end{bmatrix}^{-1} \begin{bmatrix} B_{Right}^{(i)} E^{(i-1)} \\ 0 \end{bmatrix}, \quad (6.12)$$

and $E^{(i)} = \text{diag}\left(\begin{bmatrix} e^{-\lambda_1 H_i} & e^{-\lambda_2 H_i} & e^{-\lambda_3 H_i} \end{bmatrix}\right)$. The convergence rate of the additive Schwarz methods is given by $\rho(T)$, the spectral radius of T. If $\rho(T) = 0$ then $T^k = 0$ for some $k \leq N$, and the additive Schwarz method converges in k iterations. As noted previously, using characteristic boundary conditions (which corresponds to setting all $B^{(i)}$'s to the identity matrix) results in convergence in N iterations. However, often the values of the characteristic variables are not known at the far field boundaries, and other types of boundary conditions must be imposed.

Convergence of the Schwarz algorithm cannot, in general, be guaranteed in N iterations when different types of far field boundary conditions are used. Convergence in a finite number of iterations requires that certain interface and boundary conditions are non-reflecting. In particular, if both upstream and downstream far field boundary conditions are reflecting then, (except in the special case where the reflected wave at one end of the domain does not cause a reflection at the opposite end), convergence is not achieved in a finite number of iterations. Table 6.2 summarizes the convergence of the Schwarz algorithm in term of the types of boundary and interface conditions imposed.

Generally, non-reflecting interface conditions corresponding to characteristics are set on subdomain interfaces. Table 6.3 summarizes the convergence of the Schwarz algorithm using characteristic interface conditions for the different types of boundary conditions considered in Table 6.1.

Iterations	Type of Boundary and Interface Conditions
N	1. All ICs & BCs are non-reflecting
$2N-3$	2. Inflow IC & BC are non-reflecting and outflow IC & BC are the same 3. Outflow IC & BC are non-reflecting and inflow IC & BC are the same
$2N-1$	4. Inflow IC & BC are non-reflecting 5. Outflow IC & BC are non-reflecting
$3N-4$	6. Inflow IC & BC are the same, outflow IC is non-reflecting and reflected wave at outflow boundary is not reflected by inflow IC/BC
$3N-2$	7. All ICs are non-reflecting and reflected wave at outflow boundary is not reflected by inflow BC 8. Inflow BC & outflow IC are non-reflecting and reflected wave at outflow boundary is not reflected by inflow IC
∞	Otherwise

Table 6.2: Number of iterations required for the Schwarz algorithm to convergence using different interface and boundary conditions

	Outflow BC		
Inflow BC	w_1	J^-	P
w_2, w_3	N	$2N-1$	$2N-1$
S, J^+	N	$2N-1$	$2N-1$
S, T_o	$2N-1$	$3N-2$	∞
P_o, T_o	$2N-1$	∞	∞

Table 6.3: Number of iterations required for the Schwarz algorithm to convergence using characteristic interface conditions and different boundary conditions

As noted previously, when the Schwarz algorithm does not converge in a finite number of iterations, the convergence rate is given by the spectral radius of the iteration matrix T. The convergence rate setting entropy, S, and total temperature, T_o, on the inflow and the static pressure, P, on the outflow boundary is given by:

$$\rho(T) = \left(\frac{1-M_x}{1+M_x}\right)^{\frac{1}{N-1}} e^{-\frac{2(N-2)}{N-1}\frac{1}{1-M_x^2}\frac{H}{c\Delta t}}. \tag{6.13}$$

Note, for fixed domain size, $H = \frac{1}{N}$. Thus, the convergence rate degrades exponentially as the number subdomains, N, increases. The convergence rate for other combinations of reflecting boundary conditions have similar forms.

The one-dimensional results presented in this section have an implication on the expected performance of domain decomposition methods in multiple dimensions. As perfectly non-reflecting boundary conditions are not generally available in multiple dimensions, convergence is not likely in a finite number of iterations. Additionally, degradation in the convergence rate is to be expected as the number of subdomains increases.

6.3 2D Analysis

In this section Fourier analysis is used to estimate the convergence rate of the Robin-Robin algorithm in the case of two strip domains of finite width. The theoretical convergence rate is compared with the discrete case to validate the analysis performed.

Consider the two-dimensional semi-discrete linearized Euler equation with zero forcing function:

$$\frac{W}{c\Delta t} + \begin{bmatrix} M_x-1 & 0 & 0 & 0 \\ 0 & M_x+1 & 0 & 0 \\ 0 & 0 & M_x & 0 \\ 0 & 0 & 0 & M_x \end{bmatrix} \frac{\partial W}{\partial x} + \begin{bmatrix} M_y & 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & M_y & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & M_y & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & M_y \end{bmatrix} \frac{\partial W}{\partial y} = 0. \tag{6.14}$$

Performing a Fourier transform in the y-direction gives:

$$\frac{\tilde{W}}{c\Delta t} + \begin{bmatrix} M_x - 1 & 0 & 0 & 0 \\ 0 & M_x + 1 & 0 & 0 \\ 0 & 0 & M_x & 0 \\ 0 & 0 & 0 & M_x \end{bmatrix} \frac{d\tilde{W}}{dx} + \begin{bmatrix} M_y & 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & M_y & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & M_y & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & M_y \end{bmatrix} i\xi\tilde{W} = 0, \tag{6.15}$$

where $\hat{\xi}$ is the wave number. Simplifying, (6.15) gives a system of ODEs of the form:

$$\frac{d\tilde{W}}{dx} + \tilde{A}\tilde{W} = 0. \tag{6.16}$$

Consider a partition of the domain $\Omega = [-H, H] \times \mathbb{R}$ into two subdomains $\Omega_1 = [-H, 0] \times \mathbb{R}$ and $\Omega_2 = [0, H] \times \mathbb{R}$. The Robin-Robin algorithm to solve for $\hat{W}$, the Fourier coefficients of state on the interface at $x = 0$, is given by:

Dirichlet Solve:

$$\Omega_i : \begin{cases} \frac{d\tilde{W}_i^k}{dx} + \tilde{A}\tilde{W}_i^k = 0 & \text{in } \Omega_i, \\ \Lambda_{n_i}^- \tilde{W}_i^k = \Lambda_{n_i}^- \hat{W}^k & \text{at } x = 0, \\ \Lambda_{n_i}^- \tilde{W}_i^k = 0 & \text{at } x = \pm H, \end{cases}$$

Robin Solve:

$$\Omega_i : \begin{cases} \frac{d\tilde{V}_i^k}{dx} + \tilde{A}\tilde{V}_i^k = 0 & \text{in } \Omega_i, \\ \Lambda_{n_i}^- \tilde{V}_i^k = \Lambda_{n_i}^- \hat{V}_i^k & \text{at } x = 0, \\ \Lambda_{n_i}^- \tilde{V}_i^k = 0 & \text{at } x = \pm H, \\ \frac{1}{2}\Lambda_{n_i}\hat{V}_i^k + |\Lambda_{n_i}|(\tilde{V}_i^k - \hat{V}_i^k) + Zn_i\hat{V}_i^k = -\frac{1}{2}|\Lambda_{n_i}|\left(\tilde{W}_1^k + \tilde{W}_2^k - 2\hat{W}^k\right) & \text{at } x = 0, \end{cases}$$

Update: $\hat{W}^{k+1} = \hat{W}^k + \frac{\omega}{2}\left(\hat{V}_1^k + \hat{V}_2^k\right).$ (6.17)

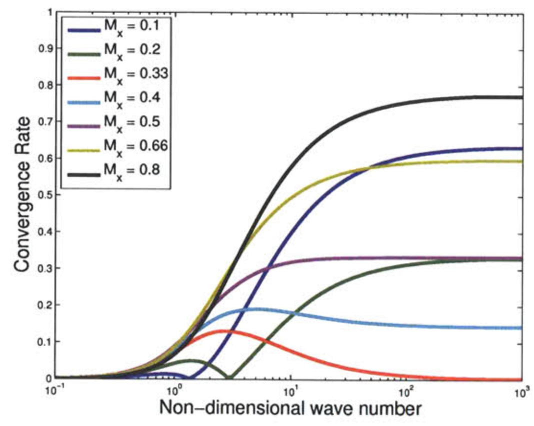
where ω is an under-relaxation parameter. In the Robin solve, Z is a skew symmetric term which may be added to the interface condition without modifying the energy stability of the Robin-Robin algorithm. In the following section an optimal choice of Z is derived which minimizes the convergence rate.

As in Section 6.2, it is possible to derive an iteration matrix for the Fourier coefficients of the error. In Appendix A the eigenvalues are computed analytically for the case of two infinite domains. However, in general it is not possible to compute analytically the eigenvalues of the iteration matrix, thus these must be computed numerically for different non-dimensional wave numbers, $\xi := H\hat{\xi}$ and subdomain CFL number, $\text{CFL}_H := \frac{c\Delta t}{H}$.

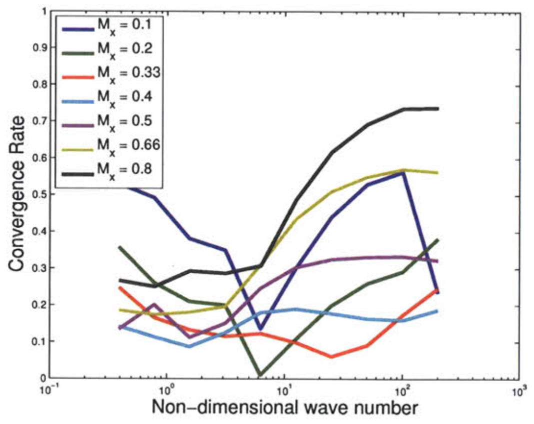
In order to assess the validity of the analysis in the discrete setting, a set of numerical experiments are performed in an effort to mimic the conditions of the analysis. The domain $\Omega = [-1, 1] \times [0, 1]$ partitioned into two square subdomains with an interface at $x = 0$. Each subdomain is discretized using a uniform structured mesh with 8192 triangular $p = 5$ elements. The boundary conditions are set such that the solution is periodic in the y-direction with wave number ξ. The discrete convergence rate is determined from the decrease in the l_2-norm of the residual over 10 Richardson iterations using the BDDC preconditioner. (As the initial error is orthogonal to constant functions, the BDDC and Robin-Robin algorithms are equivalent for this problem).

Initially, the convergence behaviour of the Robin-Robin algorithm is assessed for the case where $Z = 0$. Figure 6-1 plots both analytic and discrete convergence rates as a function of ξ for $\text{CFL}_H = 1$, 10 and 100. For small CFL_H, the convergence rate is similar to that obtained in the infinite subdomain case. This behaviour is expected as the far field boundary conditions do not significantly effect the solution at the interface. On the other hand, for $\text{CFL}_H = 10$, or $\text{CFL}_H = 100$, the boundary conditions have a significant effect on the solution at the interface for low frequency modes and markedly different behaviour is observed for $\xi < 10$. In the limit as $\xi \to \infty$ the analytical convergence rate asymptotes to the same value independent of CFL_H. However, for large CFL_H the convergence rate approaches the asymptotic value at much lower wave numbers.

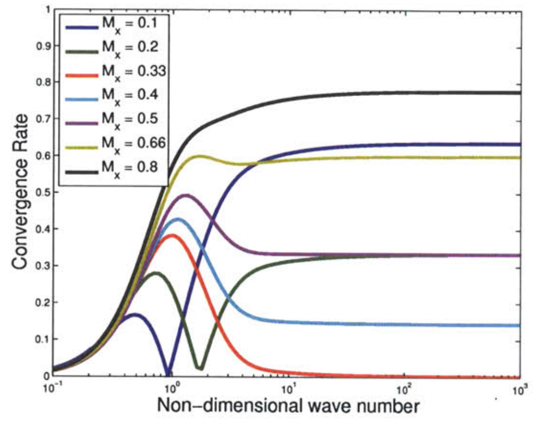
The discrete and analytical convergence rates match very well over a wide range of wave numbers. As the wave number is increased, there is insufficient resolution for the discrete solution to represent the high-frequency modes. In particular, high-frequency modes above the Nyquist frequency of ≈ 200, are aliased and the discrete convergence rate deviates from the asymptotic value. Additionally, in the discrete setting it is impossible to ensure that the initial error has components corresponding only to a single frequency. Thus, for $M_x = \frac{1}{3}$ the convergence rate for $\xi \to \infty$ is not zero but is limited by the convergence of the under-



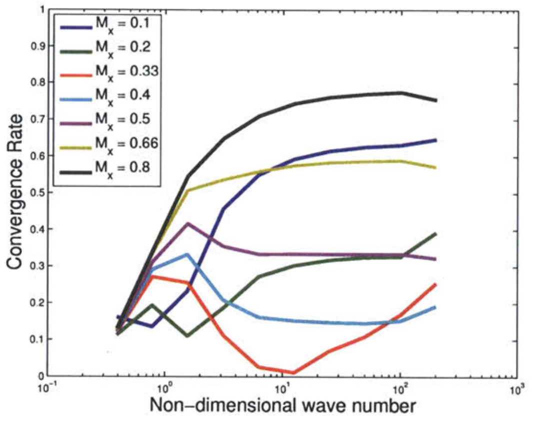
(a) Analytical, $\text{CFL}_H = 1$



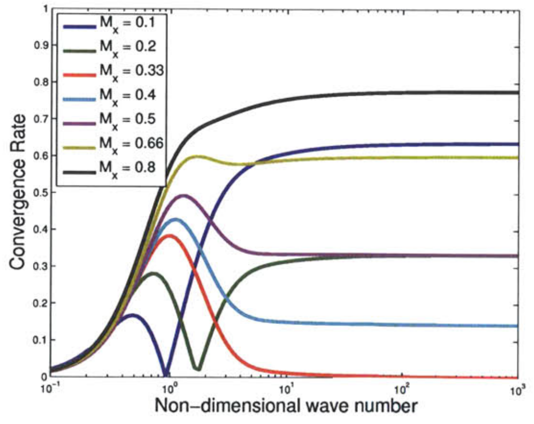
(b) Discrete, $\text{CFL}_H = 1$



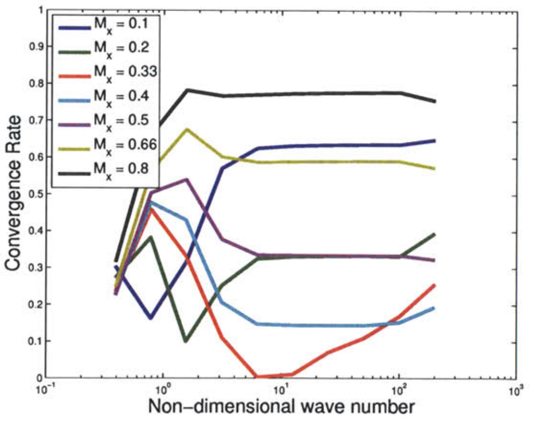
(c) Analytical, $\text{CFL}_H = 10$



(d) Discrete, $\text{CFL}_H = 10$



(e) Analytical, $\text{CFL}_H = 100$



(f) Discrete, $\text{CFL}_H = 100$

Figure 6-1: Convergence rate versus wave number, ξ using basic Robin-Robin algorithm

resolved modes.

6.4 Optimized Robin Interface Condition

In this section the Fourier analysis presented in the previous section is used determine an optimal value for the skew symmetric term Z as a function of M_x and M_y.

Prior to setting up the optimization problem, a specific form for Z is chosen. As the entropy equation decouples from the other three state equations, the interface term should not introduce coupling between these states, thus the third row and column of Z should be zero. Additionally, it is assumed that coupling should only be introduced between the two acoustic modes, allowing Z to be specified by a single parameter z:

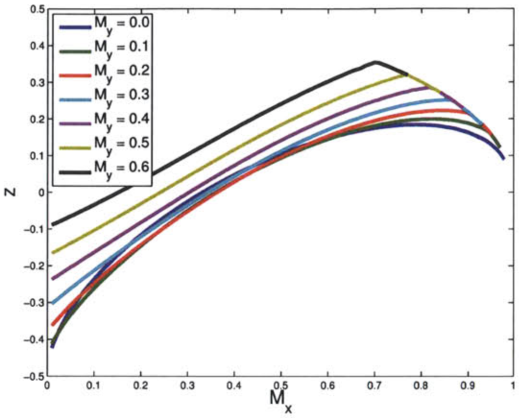
$$Z = \begin{bmatrix} 0 & z & 0 & 0 \\ -z & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{6.18}$$

The convergence rate of the Robin-Robin algorithm is limited by the maximum convergence rate over all wave numbers. For large CFL$_H$, the convergence rate for $\xi > \pi$ essentially takes on the asymptotic value corresponding to $\xi \rightarrow \infty$. Additionally, it is assumed that the primal correction in the BDDC algorithm is able to control the low frequency error modes. Thus, initially, an optimal value $z_{opt,1}$ is determined which minimizes the asymptotic convergence rate. As under-relaxation may be applied to the Robin-Robin algorithm $z_{opt,1}$, and under-relaxation parameter, $\omega_{opt,1}$ are formally defined as:

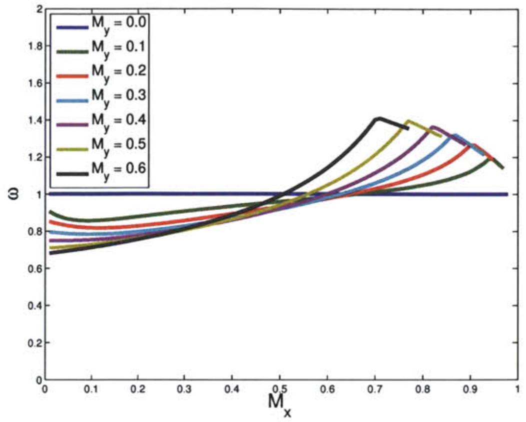
$$(z_{opt,1}, \omega_{opt,1}) = \arg\min_{z,\omega} \left[\lim_{\xi \rightarrow \infty} \rho(T(\xi)) \right]. \tag{6.19}$$

In practice the optimization problem (6.19) is solved numerically, and the formal limit $\lim_{\xi \rightarrow \infty}$ is replaced by setting $\xi = 50$.

Figure 6-2 plots the optimal values $z_{opt,1}$ and $\omega_{opt,1}$ as a function of M_x and M_y, while Figure 6-3 plots the corresponding optimal asymptotic convergence rate. Interestingly, the optimal asymptotic convergence rate is zero for all $0 < M_x < 1$ with $M_y = 0$. Moreover, the optimal under relaxation parameter is $\omega_{opt,1} = 1$. For finite M_y, even with under-relaxation, the optimal asymptotic convergence rate is finite, thus the Robin-Robin algorithm can not converge in one iteration on the two-subdomain problem.



(a) $z_{opt,1}$



(b) $\omega_{opt,1}$

Figure 6-2: Optimization of asymptotic wave number

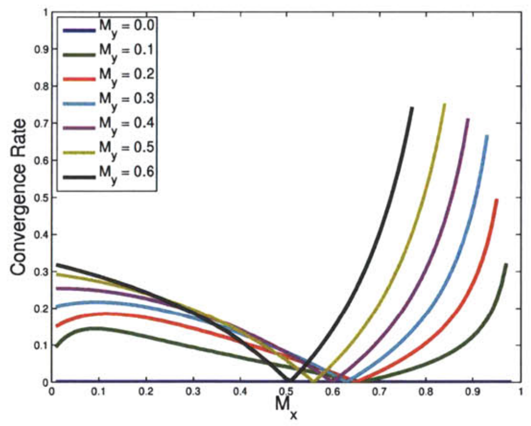
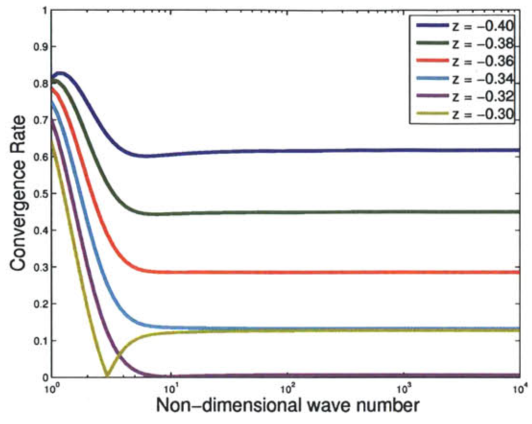
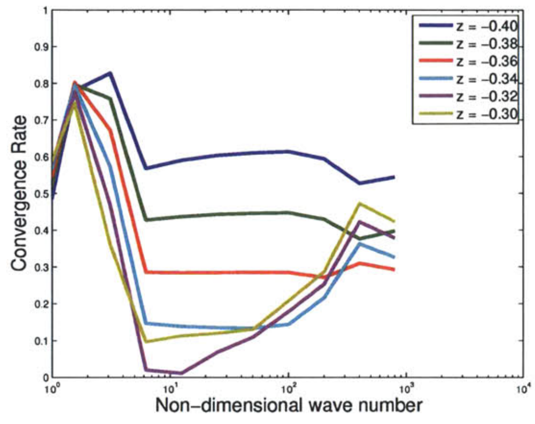


Figure 6-3: Minimum asymptotic convergence rate

Figure 6-4 plots both analytical and discrete convergence rate as a function of ξ for $M_x = 0.05$ and $M_y = 0.0$ for a range of $z = [-0.4, -0.3]$. The optimal analytic value is $z_{opt,1} = -0.32$. The analytical results show that for $z = -0.32$ a convergence rate of zero is achieved as $\xi \to \infty$. Unfortunately, in the discrete case the convergence rate at higher frequencies is dominated by the convergence rate of the under-resolved modes. Thus the optimal asymptotic convergence rate is not achieved. In the discrete setting the optimal value of z appears to be approximately -0.36. It is interesting to note that for $z \leq -0.36$ the convergence rate of the under-resolved modes is better than the analytic asymptotic convergence rate. While for $z > -0.36$ the convergence rate of the under-resolved modes is generally worse than the asymptotic value.



(a) Analytical



(b) Discrete

Figure 6-4: Convergence rate for different z, for $M = 0.05$, $\text{CFL}_H = 100$

In order to assess the behaviour of the Robin-Robin algorithm in a more general problem, a numerical experiment was performed on the two subdomain problem where the initial solution error is a Gaussian. Figure 6-5 plots the convergence rate as a function of z. A value of $z = -0.34$ gives best convergence rate, which matches well with the value of z which minimizes the discrete asymptotic convergence rate in Figure 6-4. Thus simply minimizing the asymptotic convergence rate may be sufficient to guarantee good performance of the Robin-Robin algorithm.

As the convergence rate at any flow condition is actually dominated by the maximum convergence rate over all frequencies, a second optimization problem is considered in which the optimal $z_{opt,2}$ and $\omega_{opt,2}$ are given by the solution of the following min-max problem:

$$(z_{opt,2}, z_{opt,2}) = \arg\min_{z,\omega} \left[\max_{\xi} \rho(T(\xi)) \right]. \tag{6.20}$$

In practice, is not possible to perform the maximization problem over all wave numbers, and the optimization problem is performed over 20 discrete wave numbers in the range $(10^{-2}, 10^{2})$.

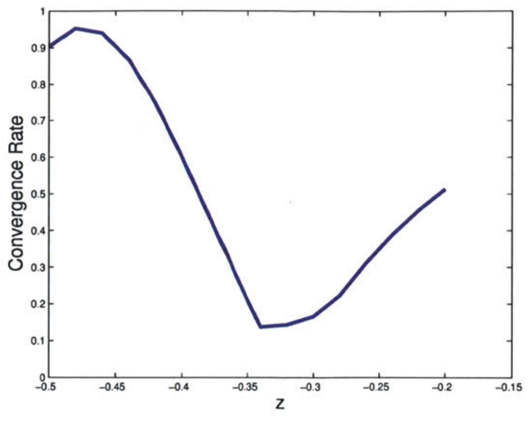
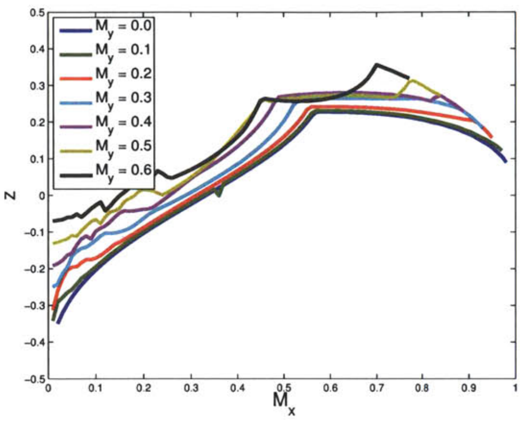


Figure 6-5: Convergence rate for different z, for $M = 0.05$, $\text{CFL}_H = 100$

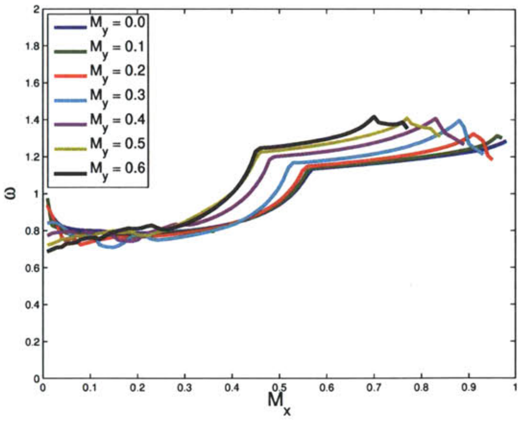
Figure 6-6 plots the optimal values $z_{opt,2}$ and $\omega_{opt,2}$ as a function of M_x and M_y. Qualitatively, $(z_{opt,2}, \omega_{opt,2})$ behave similarly to $(z_{opt,1}, \omega_{opt,1})$ for $M_x < \frac{1}{3}$. At higher Mach number, the analytical convergence rate becomes limited by the convergence rate for very small wave numbers ($\xi < 10^{-1}$), resulting in a kink in the curves for $z_{opt,2}$ and $\omega_{opt,2}$. Figure 6-7 plots the convergence rate using $z = 0$ and $z = z_{opt}$. For $z = 0$ an optimal choice of the under-relaxation parameter is determined by solving a similar max-min problem as (6.20). The optimized interface conditions lead to significantly improved performance for $M_x < \frac{1}{3}$, while the performance gains are less significant for larger Mach numbers. Particularly, unlike the basic Robin interface condition, the convergence rate using the optimized interface condition does not approach unity in the limit as $M_x \to 0$.

In practice, the BDDC algorithm is not applied as a Richardson iteration, but instead as a preconditioner to GMRES. The performance of the optimized Robin interface condition is assessed in the discrete setting by solving a simple linearized Euler model problem in the domain $\Omega = [0, 1]^2$ with homogeneous Dirichlet boundary conditions starting from a random initial condition. The domain is partitioned into four vertical strips and solved with BDDC preconditioned GMRES. An optimal value, z_{opt}, is obtained by minimizing the l_2 norm of the residual after 10 GMRES iterations, averaging over four different initial random conditions. Figure 6-8 plots the optimal value, z_{opt} as a function of M_x and M_y. The optimal z from the contour plots shows the same qualitative behaviour as $z_{opt,1}$ and $z_{opt,2}$.

Figure 6-9 plots the corresponding reduction in the residual after 10 GMRES iterations

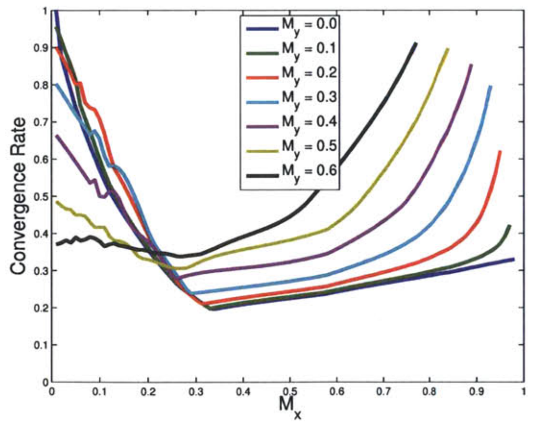


(a) $z_{opt,2}$

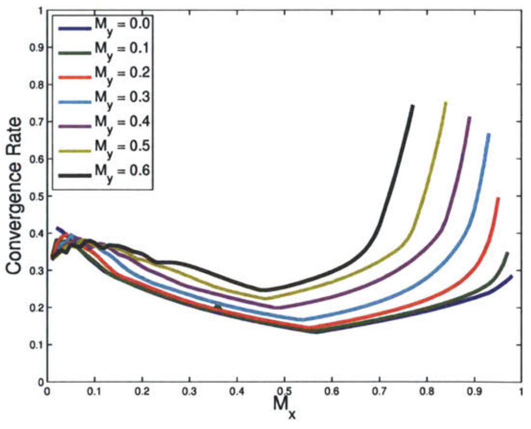


(b) $\omega_{opt,2}$

Figure 6-6: Optimization over all wave numbers



(a) $z = 0$



(b) $z = z_{opt,2}$

Figure 6-7: Convergence rate using optimized interface conditions

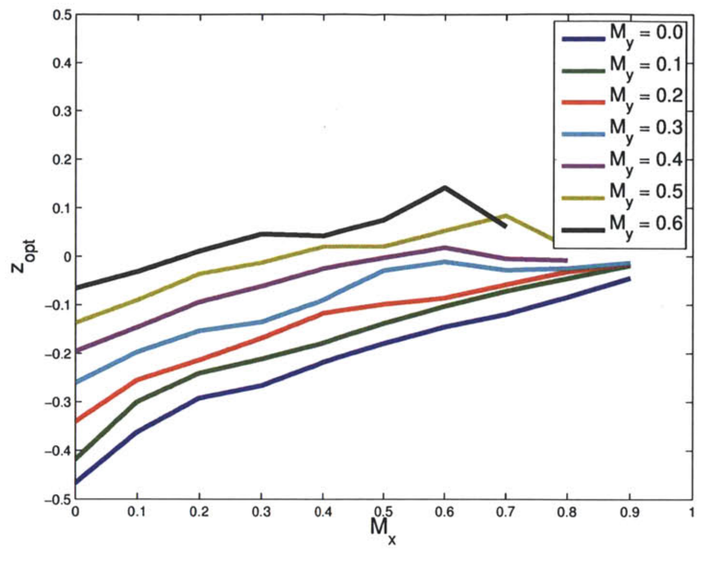
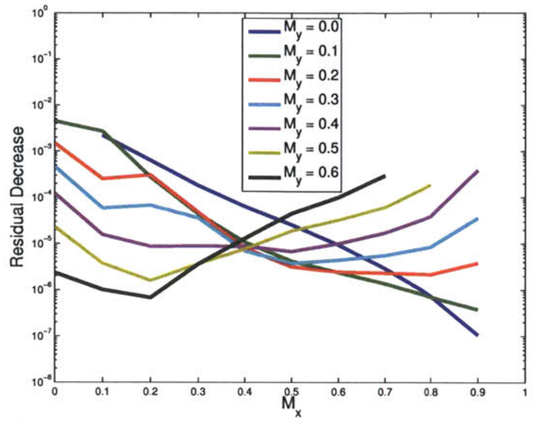


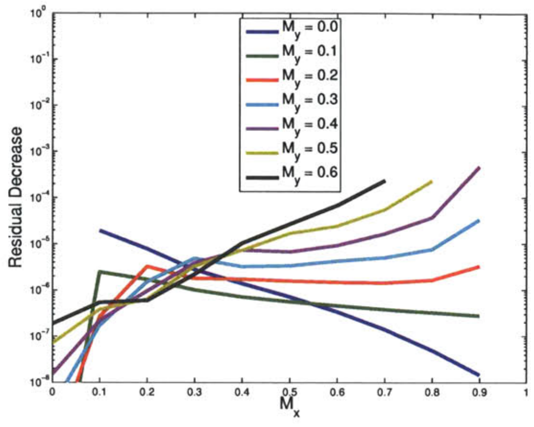
Figure 6-8: Discrete optimization of z

using $z = 0$ and the optimal z. For small M_x, using the optimized interface conditions result in a significant improvement in the convergence rate. On the other hand, for larger M_x and $M_y = 0.5$, z does not significantly affect the convergence behaviour, which matches the analytical behaviour observed in Figure 6-7.

In order to provide a better understanding of the optimized Robin-Robin interface condition, the effect of the optimized interface condition is shown graphically. Figure 6-10(a) plots W_1 of the exact solution of an Euler problem in $\Omega = [0,1]^2$ with homogeneous boundary conditions and a Gaussian source at $(0.1, 0.5)$. The domain is partitioned into 4 strips and a single iteration of the BDDC algorithm is used starting from zero initial condition. Figure 6-10(b) gives the solution after a single iteration using the basic Robin interface condition ($z = 0$), while Figure 6-10(c) gives the corresponding solution after one iteration with $z = -0.40$, which corresponds to the value of z which minimizes the residual after 10 GMRES iterations. Using the basic interface conditions, the solution is poorly represented in the all but the left-most domain. However, using the optimized conditions, the BDDC algorithm is able to propagate the disturbance two subdomains in a single iteration.

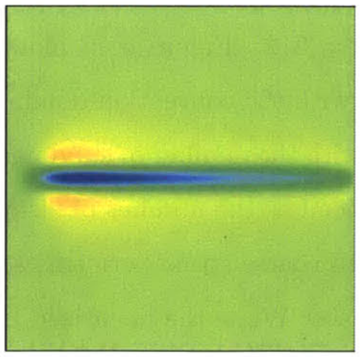


(a) $z = 0$

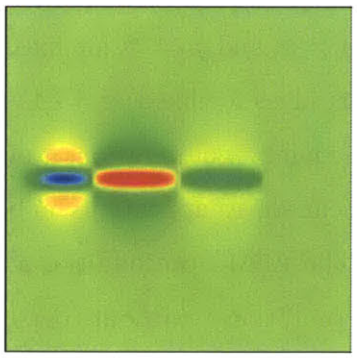


(b) $z = z_{opt}$

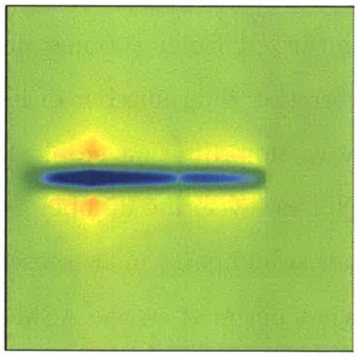
Figure 6-9: Residual reduction for linearized Euler problem after 10 GMRES iterations



(a) Exact Solution



(b) $z = 0.00$



(c) $z = z_{opt} = -0.40$

Figure 6-10: Solution after one application of the BDDC preconditioner

6.5 Numerical Results

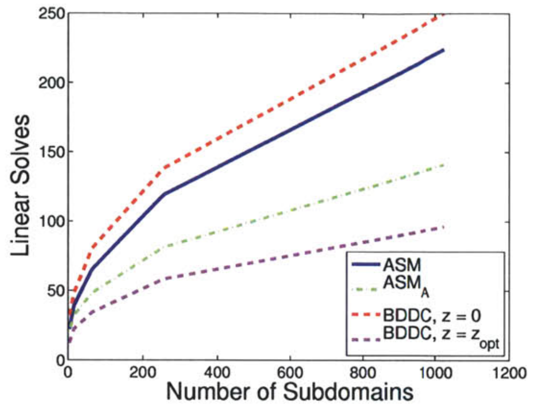
In this section numerical results are presented for two simple 2D model problems.

Linearized Euler Model Problem

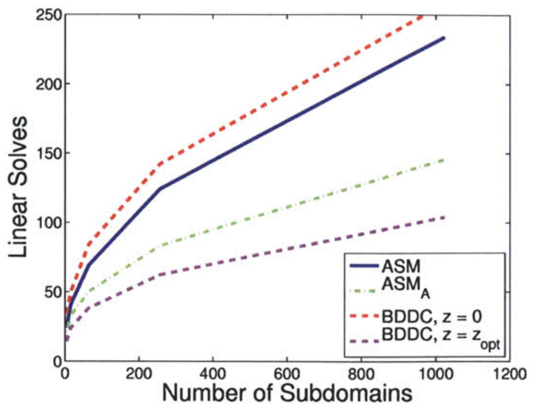
In the first model problem, the steady linearized Euler equations are solved on the unit domain $\Omega = [0,1]^2$ linearizing about a uniform flow with $M = 0.2$. A source function and boundary conditions are specified such that the exact solution is given by:

$$W = \begin{bmatrix} \sin(\pi x)\cos(\pi y) \\ \cos(\pi x)\sin(\pi y) \\ e^{-\frac{y}{\sqrt{x}}} \\ 1 - e^{-\frac{y}{\sqrt{x}}} \end{bmatrix}. \tag{6.21}$$

The linearized Euler problem is solved on the set of isotropic structured meshes presented in Chapter 3. The performance of the BDDC preconditioner with and without optimized interface conditions is compared with the ASM and ASM$_A$ preconditioners developed in Chapter 3. The relative performance of the different preconditioning algorithms is assessed in terms of the number of local linear solves required to reduce the l_2-norm of the residual by a factor of 10^3. Figure 6-11 plots the number of local linear solves required to solve the linearized Euler problem for $p = 2$ and $p = 5$ for fixed $n = 512$. Figure 6-12 plots the corresponding number of linear solves for fixed $N = 64$. As with the convection-dominated advection-diffusion problem, the number of iterations grows with the number of subdomains, N. However, the number of linear solves appears to be bounded as the number of elements per subdomain, n, increases. The ASM$_A$ preconditioner with coarse space performs somewhat better than the ASM preconditioner without coarse space. While the linearized Euler equations are hyperbolic, the coarse space is believed to provide a benefit in controlling the acoustic modes which exhibit an elliptic behaviour. The BDDC preconditioner with standard interface conditions performs poorly relative the ASM$_A$ preconditioner. However, using the optimized interface conditions results in significantly improved performance, such that the BDDC preconditioner with optimized interface conditions results in convergence in the fewest number of local linear solves.

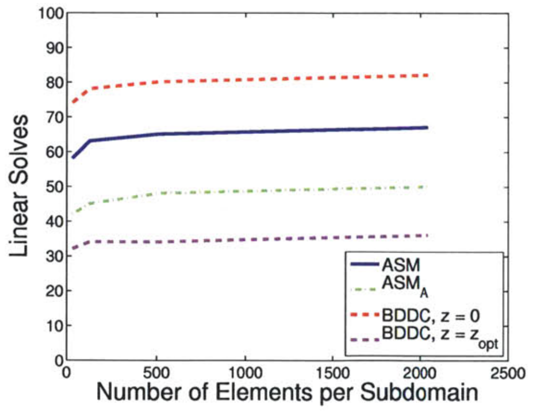


(a) $p = 2$

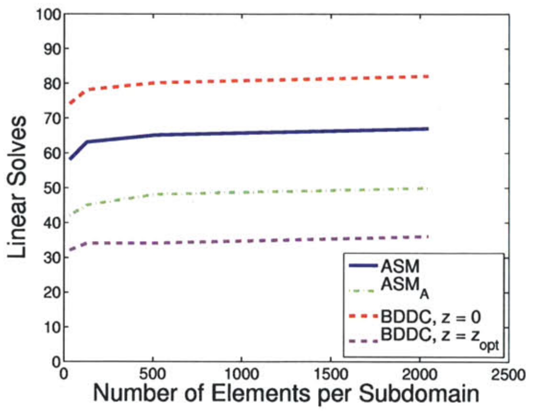


(b) $p = 5$

Figure 6-11: Number of local linear solves for linearized Euler problem with $n = 512$



(a) $p = 2$



(b) $p = 5$

Figure 6-12: Number of local linear solves for linearized Euler problem with $N = 64$

In order to provide a better understanding of the convergence behaviour of the different preconditioning algorithms, a second linearized Euler problem is solved with homogeneous boundary conditions and a Gaussian forcing function near the inflow part of the domain. The forcing function leads to a high-frequency perturbation in the solution which must be convected downstream through the domain. Figure 6-13 shows the GMRES convergence history for $n = 512$ and $N = 16$ and 64. As with the advection-dominated scalar problem, the residual does not appear to converge until the high-frequency error modes have been propagated out of the domain. However, unlike the scalar problem, the Euler system propagates information in all directions due to the acoustic modes, as opposed to only along the convective direction. Using the BDDC preconditioner with optimized interface conditions or the ASM and ASM_A preconditioners, the residual begins to drop off after $2\sqrt{N}$ local linear solves. On the other hand using the standard BDDC algorithm the residual drops off only after approximately $4\sqrt{N}$ local linear solves. While the residual begins to drop after a finite number of iterations, the linearized Euler cannot converge in a finite number of iterations. As predicted in Section 6.2 the convergence rate degrades with increasing number of subdomains.

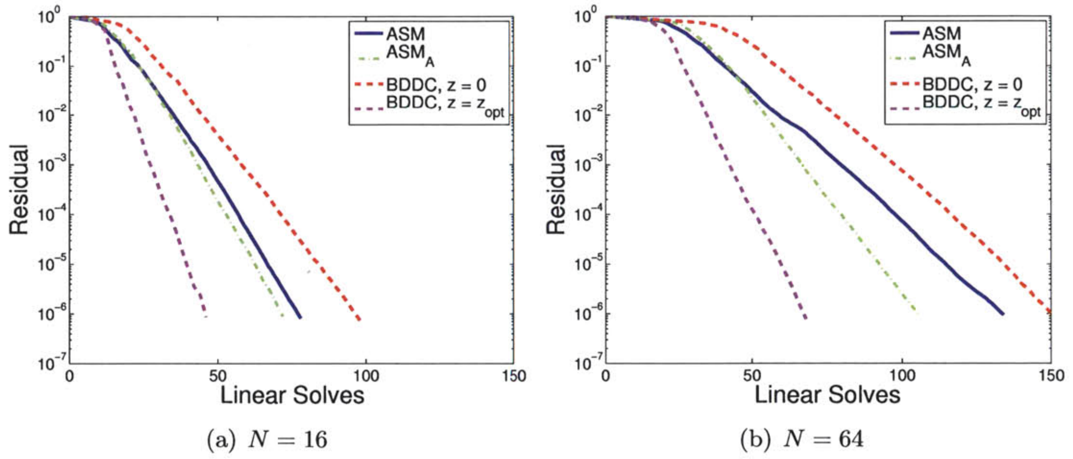


Figure 6-13: GMRES convergence plot for linearized Euler problem with $p = 2$ and $n = 512$

Linearized Navier-Stokes Model Problem

In the second model problem, the steady linearized Navier-Stokes equations are solved on the unit domain $\Omega = [0,1]^2$ linearizing about a boundary layer flow with a Mach number profile given by $M = 0.2(1 - e^{-\frac{y}{\sqrt{Re}}})$ where Re is the Reynolds number. A source function and boundary conditions are specified such that the exact solution is given by:

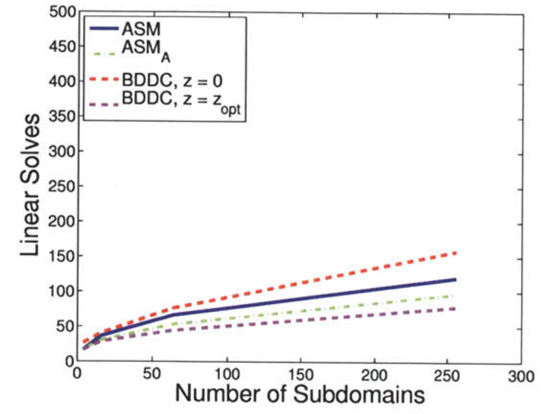
$$W = \begin{bmatrix} \sin(\pi x)\cos(\pi y) \\ \cos(\pi x)\sin(\pi y) \\ e^{-\frac{y}{\sqrt{Rex}}} \\ 1 - e^{-\frac{y}{\sqrt{Rex}}} \end{bmatrix}. \tag{6.22}$$

The linearized Navier-Stokes problem is solved on a set of anisotropic structured meshes with the aspect ratio of elements at the lower surface given by $1/\sqrt{Re}$.

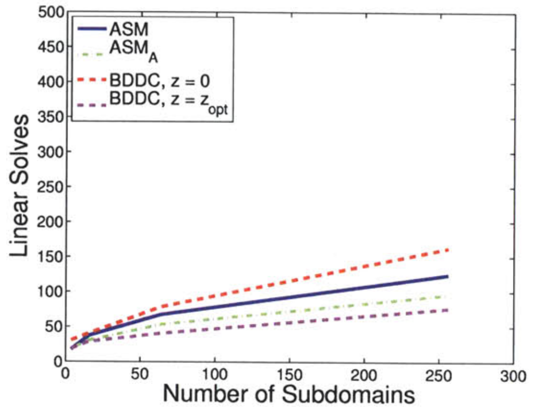
Figures 6-14 - 6-16 plot the number of local linear solves required to reduce the l_2 residual by 10^3 for the linearized Navier-Stokes problem with Reynolds number $Re = 10^6$, 10^4 and 10^2, for fixed $n = 512$ with N ranging from 4 to 1024. For the high Reynolds number case, $Re = 10^6$, the behaviour is similar to that with the linearized Euler problem. As the Reynolds number is decreased, the viscous effects become more important and the performance of the ASM preconditioner degrades relative to the ASM$_A$ and BDDC preconditioners which have a coarse space. As the Reynolds number is decreased further, the performance of the BDDC preconditioner degrades relative to the ASM$_A$ preconditioner.

Figure 6-17 shows the performance of the different preconditioners for fixed $N = 64$. In the high Reynolds number case, $Re = 10^6$ the behaviour is again similar to the linearized Euler problem. Namely, using any of the preconditioners the number of iterations does not grow significantly as the number of elements per subdomain is increased. For $Re = 10^2$, the number of linear solves required grows with increasing number of elements per subdomain. However, the performance of the BDDC preconditioner again degrades with respect to the ASM$_A$ preconditioner.

The degradation of the performance of the BDDC algorithm in the limit as the Reynolds number goes to zero may be attributed to the fact that the viscosity matrix for the Navier-Stokes system is not full rank. In the limit as the Reynolds number goes to zero, the local

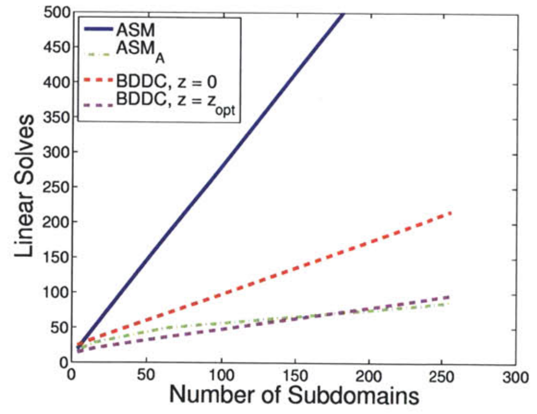


(a) $p = 2$

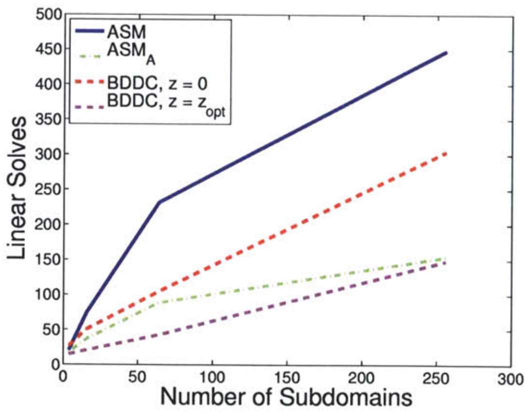


(b) $p = 5$

Figure 6-14: Number of local linear solves for linearized Navier-Stokes problem, $Re = 10^6$ with $n = 512$

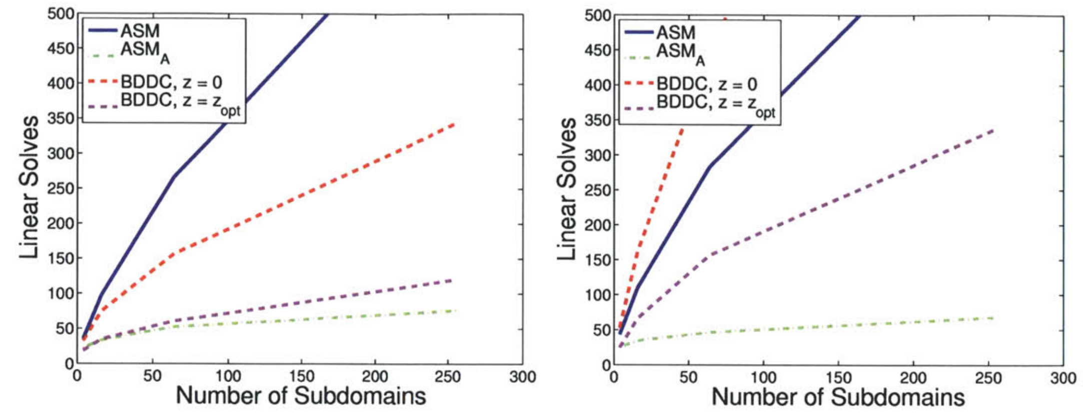


(a) $p = 2$



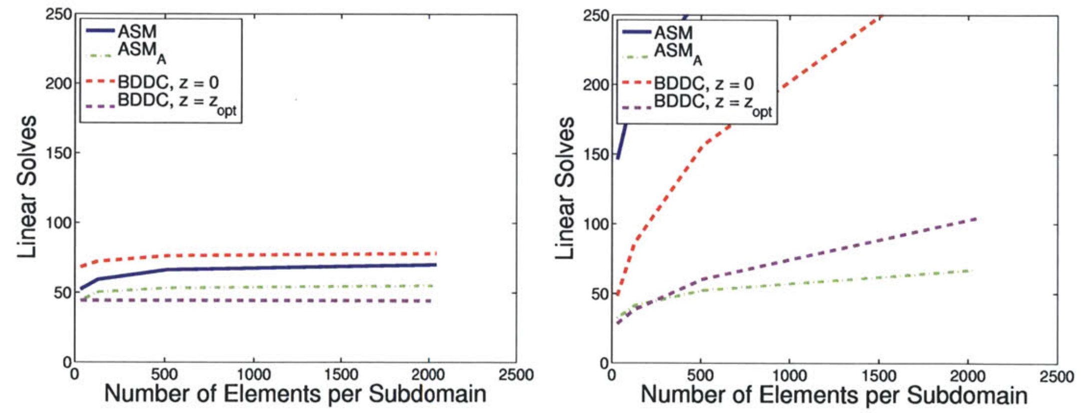
(b) $p = 5$

Figure 6-15: Number of local linear solves for linearized Navier-Stokes problem, $Re = 10^4$ with $n = 512$



(a) $p = 2$ (b) $p = 5$

Figure 6-16: Number of local linear solves for linearized Navier-Stokes problem, $Re = 10^2$ with $n = 512$



(a) $Re = 10^6$ (b) $Re = 10^2$

Figure 6-17: Number of local linear solves for linearized Navier-Stokes problem, with $p = 2$ and $N = 64$

Neumann problems correspond to the bilinear form:

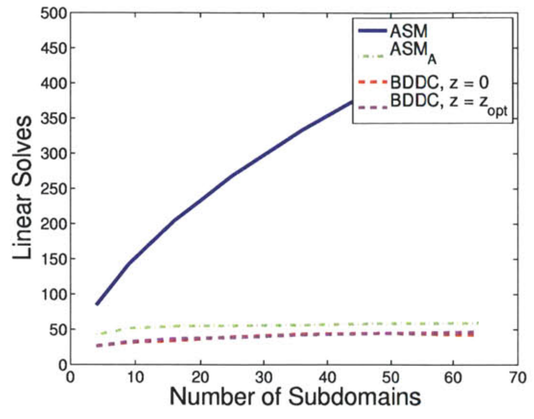
$$\begin{aligned} a_i(\lambda, \mu) &= \sum_{\kappa \in \Omega_i} a_\kappa(\lambda, \mu) \\ &= \sum_{\kappa \in \Omega_i} - \langle -K_{ijkl} q_{jl} n_i + S_{kl}(u_l - \lambda_l), \lambda_k \rangle_{\partial\kappa} \end{aligned} \tag{6.23}$$

Where, K_{ijkl} is the viscosity matrix. The energy contribution of each element simplifies to:

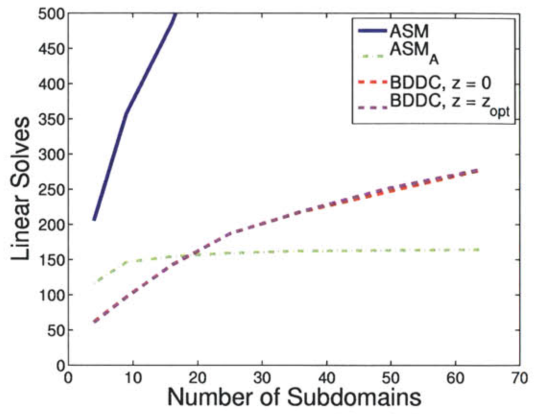
$$\begin{aligned} a_\kappa(\lambda, \lambda) &= (q_{ik}, K_{ijkl} q_{jl})_\kappa + \langle S_{kl}(u_l - \lambda_l), (u_k - \lambda_k) \rangle_{\partial\kappa} \\ &\geq 0 \end{aligned} \tag{6.24}$$

In the case where K_{ijkl} is full rank $a_\kappa(\lambda, \lambda) = 0$ implies $q_{jl} = 0$ and $u_l = \lambda_l$ is constant. However, in the case where K_{ijkl} is not full rank, $a_\kappa(\lambda, \lambda) = 0$ only implies that u_{l,x_j} is in the null space of K. Hence, constraining only interface averages on subdomain boundaries is not sufficient to ensure that the local constrained Neumann problems are well posed.

It is interesting to note that for $p = 0$ the null space of $a_\kappa(\lambda, \lambda)$ corresponds only to constant functions. Thus for $p = 0$ at very small Reynolds numbers the BDDC preconditioner behaves in the same manner as the diffusion-dominated scalar problem. Figure 6-18 and 6-19 plot the number of local linear solves required to converge the linearized Navier-Stokes problem with Re for $p = 0$ and $p = 1$, varying N and n respectively. For $p = 0$ the number of linear solves is bounded as the number of subdomains increases using either the BDDC or ASM$_A$ preconditioners, which matches the behaviour observed in the diffusion-dominated scalar case. Additionally, the number of linear solves grows only weakly when increasing the number of elements per subdomain. Similar behaviour is observed with the ASM$_A$ preconditioner for $p \geq 1$. This is in contrast to the performance of the BDDC preconditioner which degrades relative the ASM$_A$ for $p \geq 1$. Note that for this test case, the standard and optimized BDDC algorithms perform similarly, as the convergence behaviour is dictated by the viscous effects.

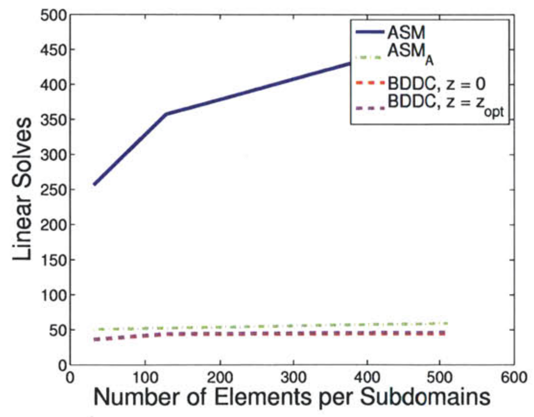


(a) $p = 0$

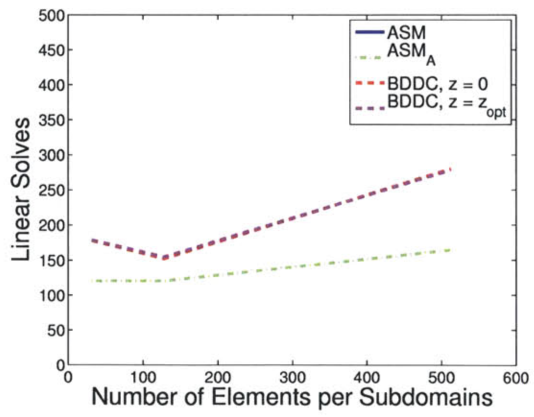


(b) $p = 1$

Figure 6-18: Number of local linear solves for linearized Navier-Stokes problem, $Re = 0.01$, $n = 512$



(a) $p = 0$



(b) $p = 1$

Figure 6-19: Number of local linear solves for linearized Navier-Stokes problem, $Re = 0.01$, $N = 64$

Chapter 7

Application to Aerodynamics Flows

This chapter presents numerical results using domain decomposition methods for the solution of several fundamental aerodynamic flows. In Section 7.1 the additive Schwarz and BDDC preconditioners are used for the solution of the HDG discretization for inviscid and laminar viscous flows. In Section 7.2 an additive Schwarz preconditioner with and without overlap is presented for the solution of the DG discretization of inviscid as well as both laminar and turbulent viscous flows.

7.1 HDG Results

This section presents weak scaling results for the HDG discretization of two dimensional aerodynamic flows. For each of test problems presented in this section an initial coarse mesh is generated through an adaptive process, while finer meshes are obtained by refining the grid metric and completely remeshing using BAMG [63]. The non-linear solution procedure is started from an initial flow interpolated from a $p = 0$ solution on the coarsest mesh. Pseudo-transient continuation is performed starting from an initial global CFL number of 1, which is increase by a factor of 5 after each successful non-linear update. The non-linear solution procedure continues until the non-linear residual has been reduced to below 10^{-10}.

At each non-linear iteration, the linear system is solved to a relative tolerance of 10^{-4} using preconditioned GMRES with a restart value of 200. The inexact local solvers use an ILU(0) factorization with MDF reordering and a $p = 0$ correction, with the BDDC preconditioner using the two matrix approach. Numerical results are presented with both

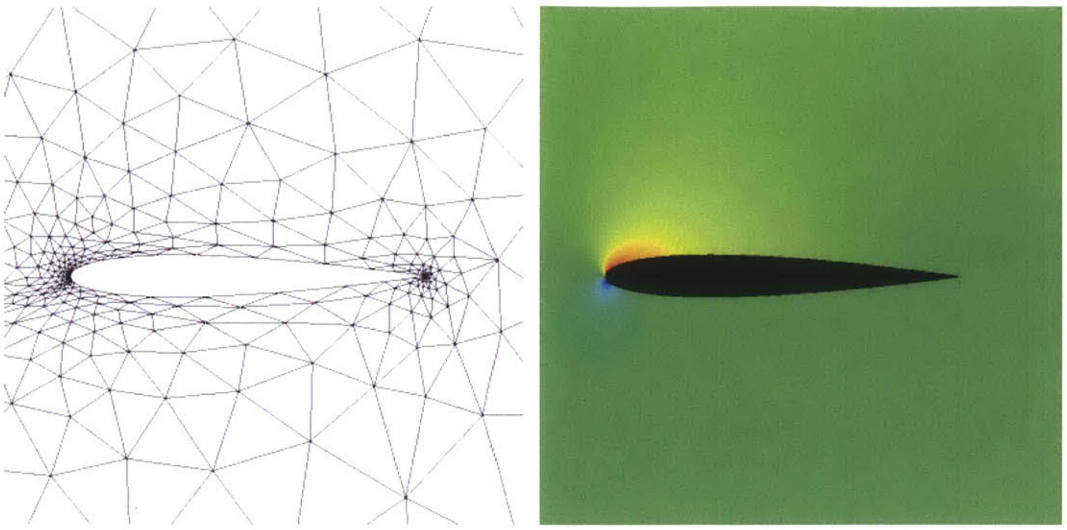

(a) Mesh (b) Mach Number

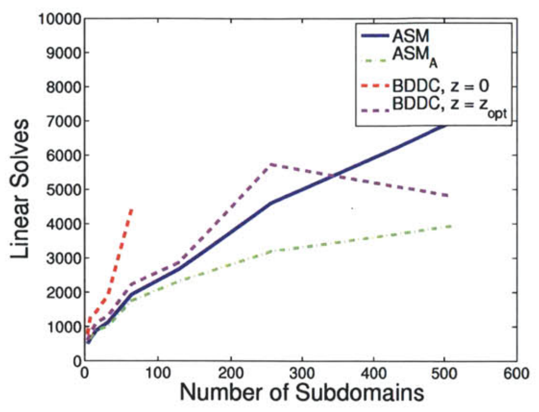
Figure 7-1: Grid and flow solution for inviscid flow over NACA0012, $M_\infty = 0.3$, $\alpha = 5.0°$, $p = 5$

non-optimized ($z = 0$) and optimized ($z = z_{opt}$) Robin-Robin interface conditions. For the optimized BDDC preconditioner, the value of z is evaluated as a function of the normal and tangential Mach numbers at each point on the interface, using a quartic polynomial fit of the discrete optimal z derived in Chapter 6. The appropriate interface condition is then obtained using a transformation from characteristic variables (in the direction normal to the interface) to the conservative variables.

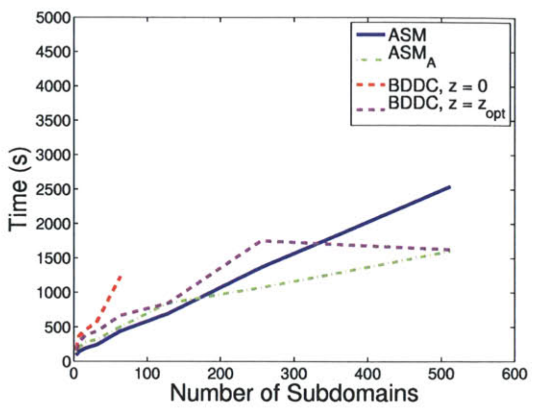
NACA0012, Inviscid flow, $M_\infty = 0.3$, $\alpha = 5.0°$

In the first test case, the inviscid flow over the NACA0012 airfoil is solved at a freestream Mach number $M_\infty = 0.3$ and angle of attack $\alpha = 5.0°$. Weak scaling results are presented for $p = 2$ and $p = 5$ solutions from 4 to 512 processors. For $p = 2$ solutions each subdomain has approximately 2000 elements, while for $p = 5$ each subdomain has approximately 500 elements. Thus, for either $p = 2$ or $p = 5$, the finest mesh problem has slightly more than 5 million degrees of freedom. Figure 7-1 gives a sample mesh with 1000 elements and plots the Mach number for the $p = 5$ solution on this mesh.

Figures 7-2 and 7-3 presents the number of local linear solves and CPU time for $p = 2$ and $p = 5$ respectively. For $p = 2$, with small numbers of processors, the ASM, ASM_A and



(a) Local Linear Solves

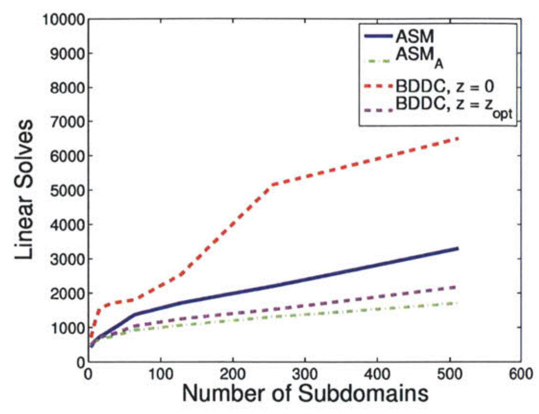


(b) CPU Time

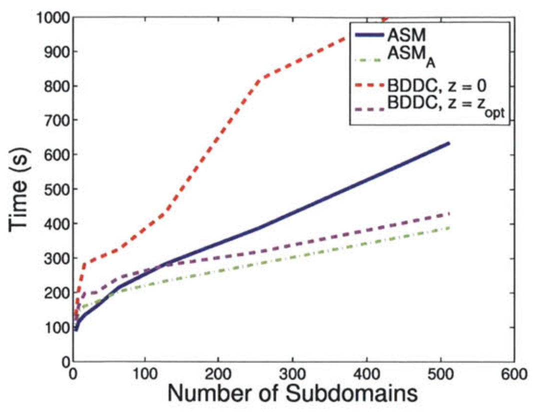
Figure 7-2: Weak scaling results for inviscid flow over NACA0012, $M_\infty = 0.3$, $\alpha = 5.0°$, $p = 2$, 2000 elements per subdomain

optimized BDDC preconditioner perform similarly in terms of the number of local linear solves. However, the ASM_A and BDDC preconditioners are slightly more expensive in terms of CPU time, thus the ASM preconditioner performs the best in terms of CPU time for less than 128 subdomains. As the number of subdomains is increased, the coarse space becomes more important. With more than 128 processors, the ASM_A preconditioner reduces both the number of local linear solves and the CPU time relative to the ASM preconditioner. At 512 processors, both ASM_A and optimized BDDC preconditioners perform similarly in terms of CPU time, with significant improvement over the ASM preconditioner. The standard BDDC preconditioner without optimized interface conditions performs significantly worse than the other three preconditioners, and in fact is unable to converge the $p = 2$ problem on more than 64 processors.

For $p = 5$, using more than 8 processors, the ASM_A and optimized BDDC preconditioners reduce the number of local linear solves required in order to converge the linear system relative to the ASM preconditioner without a coarse space. However, for relatively small number of processors the ASM preconditioner is superior to the ASM_A and BDDC preconditioners in terms of CPU time, as the preconditioners with coarse space have the additional cost of setting up the coarse space problem in the factorization step. As the number of subdomains increases, the ASM_A and optimized BDDC preconditioners also perform better than the ASM preconditioner in terms of CPU time, as the additional cost of forming



(a) Local Linear Solves

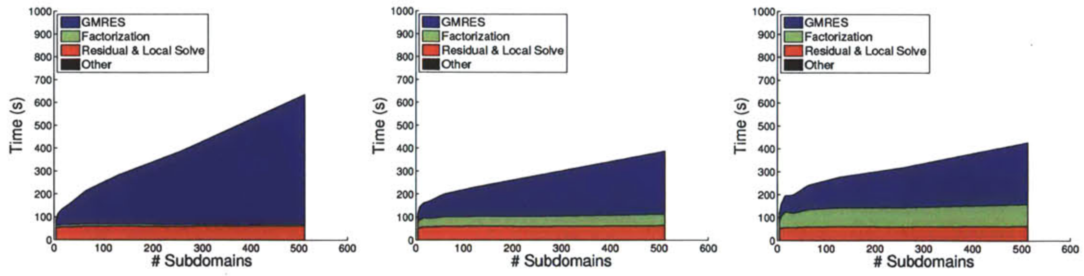


(b) CPU Time

Figure 7-3: Weak scaling results for inviscid flow over NACA0012, $M_\infty = 0.3$, $\alpha = 5.0°$, $p = 5$, 500 elements per subdomain

the coarse space is offset by reduced CPU time in the GMRES solve. Figure 7-4 plots the portion of the run time spent in the residual evaluation, factorization and GMRES parts of the solution procedure for the $p = 5$ solutions. The ASM$_A$ and BDDC preconditioners have significantly larger factorization times than the ASM preconditioner, as the factorization times for the preconditioners with coarse spaces includes the local linear solves required to form the coarse basis functions. Early in the non-linear solution procedure, when the CFL number is small, the number of local linear solves required to form the coarse basis functions may be greater than the number of local solves performed during the GMRES solve. However, as the CFL number increases, and the linear system becomes more difficult to solve, the number of linear solves required to form the coarse space is much smaller than that used in the GMRES solve.

For small numbers of processors, the larger overhead in the factorization procedure for the ASM$_A$ and BDDC preconditioners results in longer CPU time than the ASM preconditioner. However, as the number of subdomains increases, the coarse space becomes more important and the increased cost in the factorization procedure is offset by a smaller number of linear iterations. The ASM$_A$ and optimized BDDC preconditioner perform similarly, with the BDDC preconditioner performing slightly better with larger number of processors.



(a) ASM (b) ASM_A (c) BDDC, $z = z_{opt}$

Figure 7-4: Detailed timing for inviscid flow over NACA0012, $M_\infty = 0.3$, $\alpha = 5.0°$, $p = 5$, 500 elements per subdomain

NACA0005, $M_\infty = 0.2$, $\alpha = 2.0°$, $Re_c = 5000$

In the second test case, the viscous flow of over the NACA0005 airfoil is solved at a freestream Mach number of $M_\infty = 0.2$, angle of attack $\alpha = 2.0°$ and Reynolds number $Re_c = 5000$. As in the previous test case, the local inexact solver is a block-ILU(0) factorization with a $p = 0$ coarse grid correction. The two matrix approach is used for the BDDC preconditioner. Figure 7-5 plots a mesh with approximately 1000 elements and the Mach number of the $p = 5$ solution.

Weak scaling results are presented for higher-order $p = 2$ and $p = 5$ solutions. For $p = 2$ solutions, each subdomain has approximately 1000 elements, while for $p = 5$, each subdomain has approximately 250 elements, such that the finest mesh problem has slightly more than 2.5 million degrees of freedom. Figures 7-6 and 7-7 plot the number of local linear solves and CPU time for $p = 2$ and $p = 5$, respectively. For this viscous test case, the ASM_A and optimized BDDC preconditioner reduce the number of local linear solves required to converge relative to the ASM preconditioner without coarse space even at small numbers of processors. However, for small numbers of processors, the ASM preconditioner performs the best in terms of CPU time, as the additional cost of the setup time for the coarse space problem is not offset by the reduced number of linear solves during GMRES. On the other hand, as the number of subdomains increases, the coarse space becomes more important and the ASM_A and optimized BDDC preconditioners are superior for more than 128 subdomains. As with the previous test case, the standard BDDC preconditioner without optimized interface conditions performs poorly and is not competitive with the other methods.

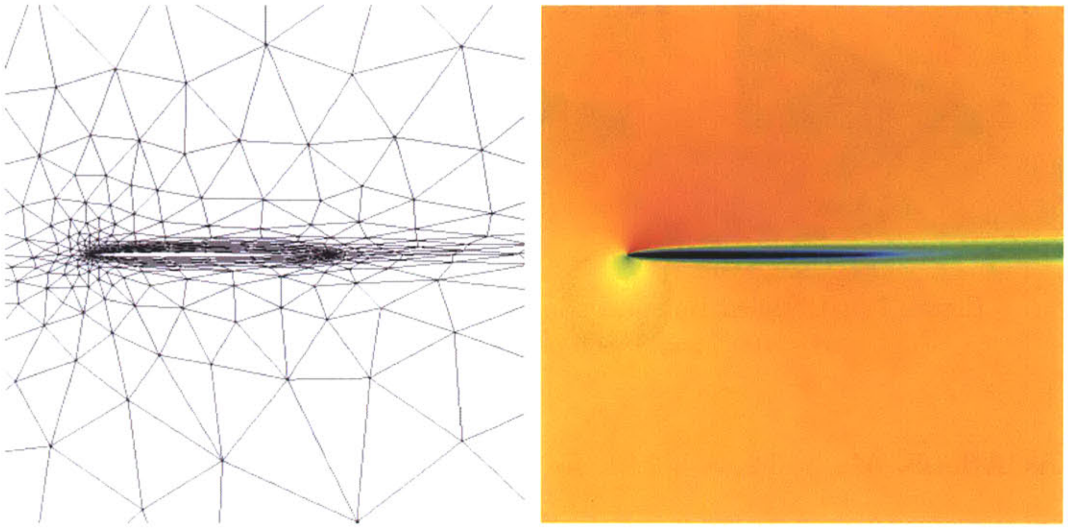
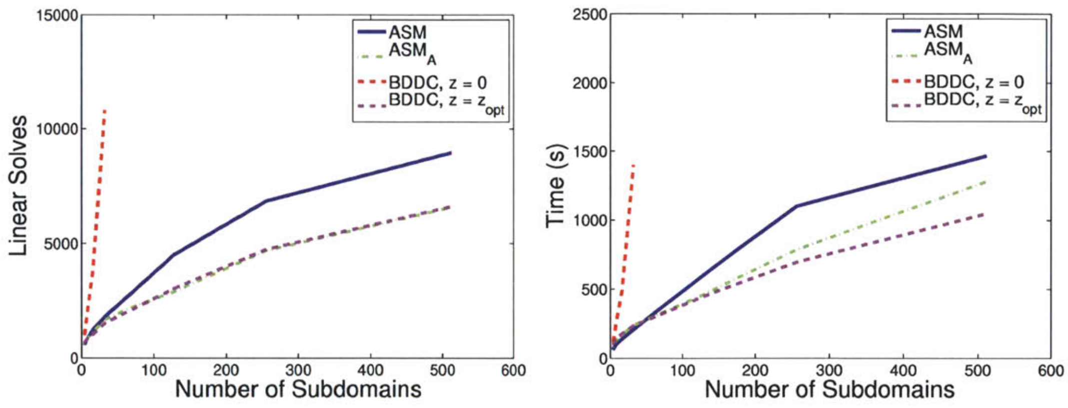

(a) Mesh (b) Mach Number

Figure 7-5: Grid and flow solution for viscous flow over NACA0005, $M_\infty = 0.2$, $\alpha = 2.0°$, $Re_c = 5000$, $p = 5$



(a) Local Linear Solves (b) CPU Time

Figure 7-6: Weak scaling results for viscous flow over NACA0005, $M_\infty = 0.2$, $\alpha = 2.0°$, $Re_c = 5000$, $p = 2$, 1000 elements per subdomain

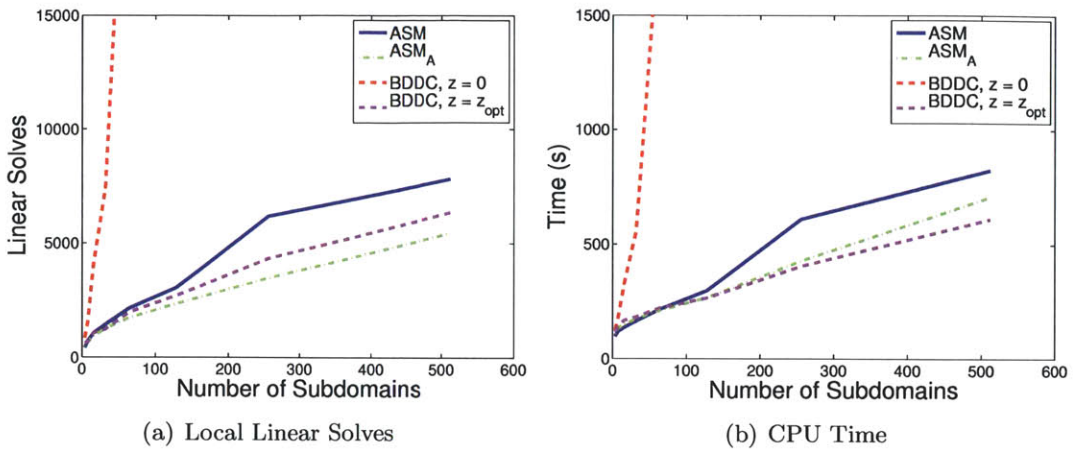


Figure 7-7: Weak scaling results for viscous flow over NACA0005, $M_\infty = 0.2$, $\alpha = 2.0°$, $Re_c = 5000$, $p = 5$, 250 elements per subdomain

For both inviscid and viscous test cases the $p = 2$ and $p = 5$ solutions require approximately the same CPU time. As higher-order methods have the potential of reducing the number of degrees of freedom required to achieve a desired error tolerance, the use of higher-order methods may significantly reduce the computational time relative to standard second-order methods.

7.2 DG Results

In this section an additive Schwarz preconditioner without overlap and a restricted additive Schwarz preconditioner with overlap are presented for the DG discretization. This section provides only a brief review of the DG discretization; a complete description may be found in the thesis of Oliver [99].

The DG discretization is obtained by multiplying the conservation law (2.1) by test functions $w \in W_h^p$ and integrating by parts. The weak form is: Find $u \in W_h^p$ such that

$$\mathcal{R}_h(u, w) \;=\; 0, \qquad \forall w \in W_h^p, \tag{7.1}$$

where

$$\mathcal{R}_h(u, w) \;=\; R_{h,I}(u, w) + R_{h,V}(u, w) + R_{h,S}(u, w), \tag{7.2}$$

and $R_{h,I}(u,w)$, $R_{h,V}(u,w)$ and $R_{h,S}(u,w)$ denote respectively the inviscid, viscous and source terms of the discretization. In this work the inviscid discretization uses the Roe flux [107], while the viscous discretization uses the second method of Bassi and Rebay [14, 15]. Source terms which are functions of the gradient of the state are evaluated using the asymptotically dual consistent formulation of Oliver [99]. The DG discretization is solved using the same pseudo-transient continuation scheme as for the HDG discretization. At each non-linear iteration a linear system of the form

$$Ax = b, \tag{7.3}$$

is solved for the solution update, $x = \Delta u$. The system A has a block structure with each block row/column corresponding to degrees of freedom on a single element. Additionally, the DG discretization maintains a nearest neighbor stencil, such that each element is coupled only to those elements with which it shares a common face.

Consider a decomposition of the domain Ω into N, non-overlapping subdomains Ω_i consisting of the union of elements in $\mathcal{T}$, such that each element κ is associated a single subdomain. Denote by $R^{(i)}$ the restriction operator which extracts degrees of freedom on Ω_i from those on all of Ω. A non-overlapping additive Schwarz preconditioner for (7.3) is given by:

$$M_{BJ}^{-1} = \sum_{i=1}^{N} R^{(i)^T} A_i^{-1} R^{(i)}, \tag{7.4}$$

where $A_i = R^{(i)} A R^{(i)^T}$. The subscript $_{BJ}$ is used to denote this preconditioner as the non-overlapping additive Schwarz preconditioner corresponds to a subdomain-wise block-Jacobi preconditioner. The local solves involving the action of A_i^{-1} correspond to the solution of a Dirichlet problem in Ω_i with fixed solution on neighboring subdomains Ω_j. The application of this preconditioner involves no communication as the local residuals and solves are performed independently on each subdomain.

In order to define an overlapping preconditioner denote by Ω_i' the region obtained by extending each subdomain Ω_i by a single element across each face on $\partial\Omega_i$. Denote by $R_\delta^{(i)}$ the restriction operator which extracts degrees of freedom on Ω_i' from those on all of Ω. The standard overlapping additive Schwarz preconditioner may be written as:

$$M_{ASM}^{-1} = \sum_{i=1}^{N} R_\delta^{(i)^T} A_{i,\delta}^{-1} R_\delta^{(i)}, \tag{7.5}$$

where $A_{i,\delta} = R_\delta^{(i)} A R_\delta^{(i)^T}$. This preconditioner involves the solution of N independent Dirichlet problems on the overlapping regions $\Omega_i^{'}$. Note that each application of this preconditioner involves two communication steps: first in the residual assembly step (i.e the action of $R_\delta^{(i)}$) and second in the solution update step (the action of $R_\delta^{(i)^T}$). In this work a restricted additive Schwarz preconditioner is used which allows communication only in the residual assembly step. In order to define the restricted additive Schwarz preconditioner a third restriction operator $R_0^{(i)}$ is defined which extends $R^{(i)}$ by zeros to the overlapping region $\Omega_i^{'}$. The restricted additive Schwarz preconditioner is given by:

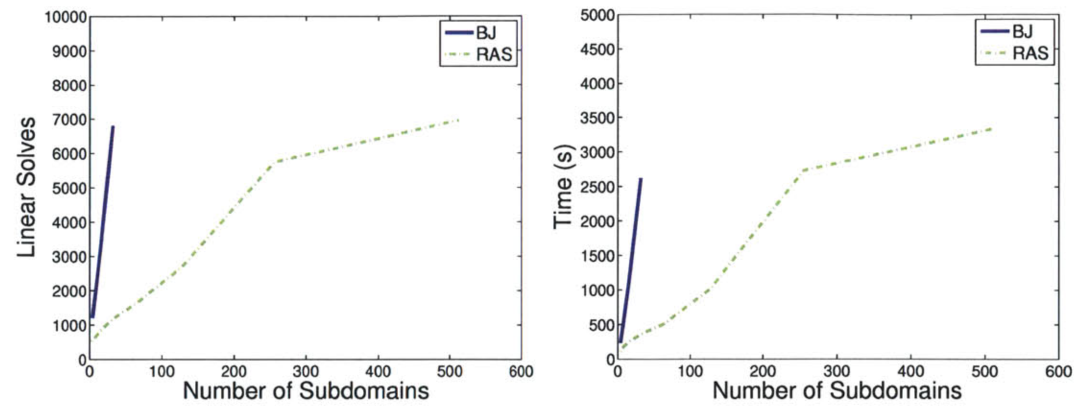
$$M_{RAS}^{-1} = \sum_{i=1}^{N} R_0^{(i)^T} A_{i,\delta}^{-1} R_\delta^{(i)}. \tag{7.6}$$

The action of this preconditioner corresponds to solving N Dirichlet problem on the overlapping regions $\Omega_i^{'}$, while only updating the local part of the solution corresponding to elements in Ω_i. The restricted additive Schwarz preconditioner has been shown to decrease both the communication cost as well as the number of iterations required to converge relative to the standard additive Schwarz preconditioner for a finite volume discretization of inviscid compressible flows [33, 35].

The remainder of this section presents weak scaling results for the DG discretization of two dimensional flows. The non-linear solution procedure used for these test cases is the same as those for the results using the HDG discretization.

NACA0012, Inviscid flow, $M_\infty = 0.3$, $\alpha = 5.0°$

In the first test case, the inviscid flow over the NACA0012 airfoil is solved at freestream Mach number $M_\infty = 0.3$ and angle of attack $\alpha = 5.0°$. Weak scaling results are presented for higher-order $p = 2$ and $p = 5$ solutions from 4 to 512 processors. For $p = 2$ solutions each subdomain has approximately 2000 elements, while for $p = 5$ each subdomain has approximately 500 elements. Figures 7-8 and 7-9 presents the number of local linear solves



(a) Local Linear Solves (b) CPU Time

Figure 7-8: Weak scaling results for inviscid flow over NACA0012, $M_\infty = 0.3$, $\alpha = 5.0°$, $p = 2$, 2000 elements per subdomain

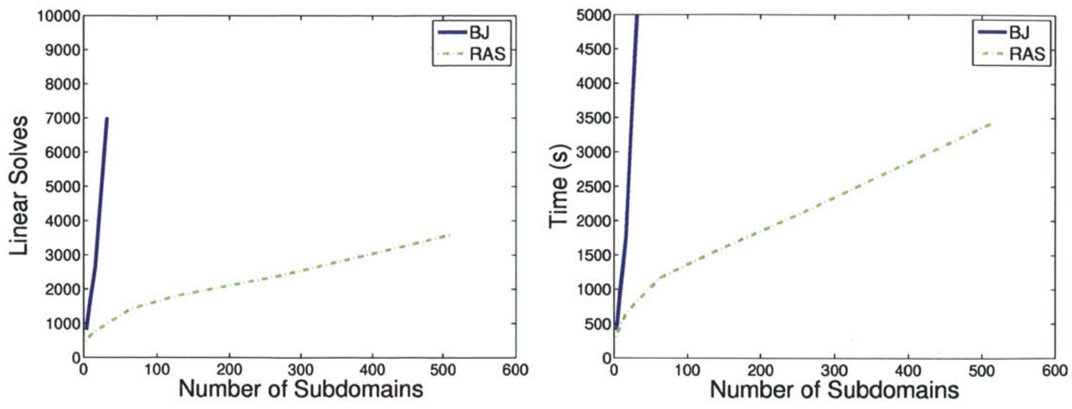
and CPU time for $p = 2$ and $p = 5$ respectively.

For both $p = 2$ and $p = 5$ the restricted additive Schwarz preconditioner performs significantly better than the subdomain-wise block-Jacobi preconditioner. Using more than 32 processors, the block-Jacobi preconditioner is unable to sufficiently solve the linear system at each Newton iteration in order to converge the non-linear problem. On the other hand, the restricted additive Schwarz preconditioner is able to converge the non-linear problem with up to 512 processors. However, both the number of iterations and the CPU time grows with increasing number of subdomains. Thus, a coarse space may be necessary when a larger number of processors is used.

NACA0005, $M_\infty = 0.2$, $\alpha = 2.0°$, $Re_c = 5000$

In the second test case, the viscous flow of over the NACA0005 airfoil is solved at a freestream Mach number of $M_\infty = 0.2$, angle of attack $\alpha = 2.0°$ and Reynolds number $Re_c = 5000$. As in the previous test case, the local inexact solver is a block-ILU(0) factorization with a $p = 0$ coarse grid correction.

Weak scaling results are presented for higher-order $p = 2$ and $p = 5$ solutions. For $p = 2$ solutions, each subdomain has approximately 1000 elements, while for $p = 5$, each subdomain has approximately 250 elements. Figures 7-10 and 7-11 plot the number of local linear solves and CPU time for $p = 2$ and $p = 5$, respectively. Once again, the restricted additive



(a) Local Linear Solves (b) CPU Time

Figure 7-9: Weak scaling results for inviscid flow over NACA0012, $M_\infty = 0.3$, $\alpha = 5.0°$, $p = 5$, 500 elements per subdomain

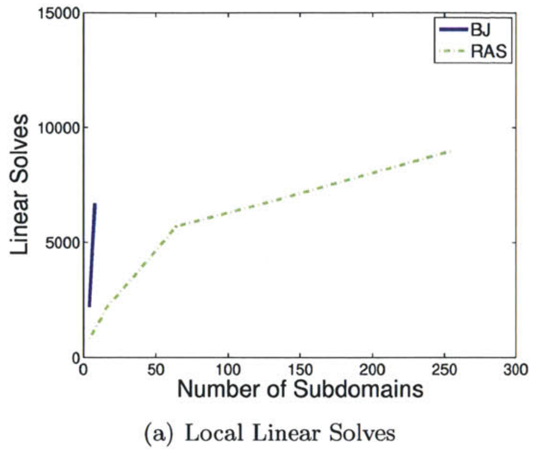
Schwarz preconditioner performs significantly better than the non-overlapping block-Jacobi preconditioner. In particular the subdomain-wise block-Jacobi is unable to converge this problem on more that 8 processors. As with the previous test case, the number of iterations using the restricted additive Schwarz preconditioner grows with increasing number of subdomains.

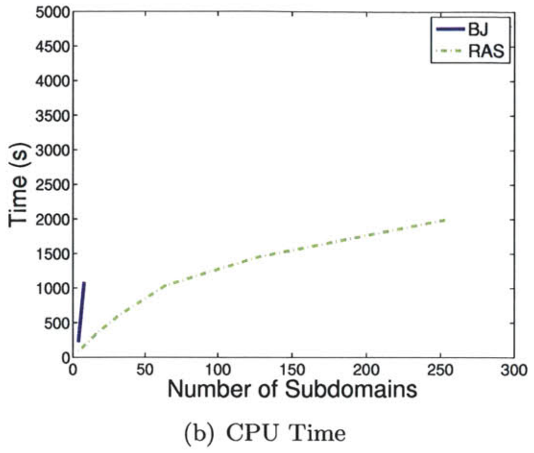
RAE2822, RANS-SA, $M_\infty = 0.3$, $\alpha = 2.31°$, $Re_c = 6.5 \times 10^6$

In the third test case, the subsonic turbulent flow is solved over the RAE2822 airfoil at a Mach number $M_\infty = 0.3$, angle of attack $\alpha = 2.31°$ and Reynolds number $Re_c = 6.5 \times 10^6$. The flow is solved using the Reynolds-Averaged Navier-Stokes equations with the Spalart-Allmaras turbulence model in the fully turbulent mode [114, 115]. Details of the DG discretization of the RANS-SA equations may be found in the thesis of Oliver [99]. Figure 7-12 plots a mesh with 1000 elements and the corresponding eddy viscosity for the $p = 2$ solution.

The introduction of the turbulence model results in a system of equations which are highly nonlinear. In order to ensure the convergence of the non-linear solution procedure, the initial global CFL number is set to 10^{-2} and is allowed to increase by a factor of 2 each successful solution update. Additionally, a line-search is introduced to ensure that the pseudo-unsteady residual decreases in each non-linear iteration [66].

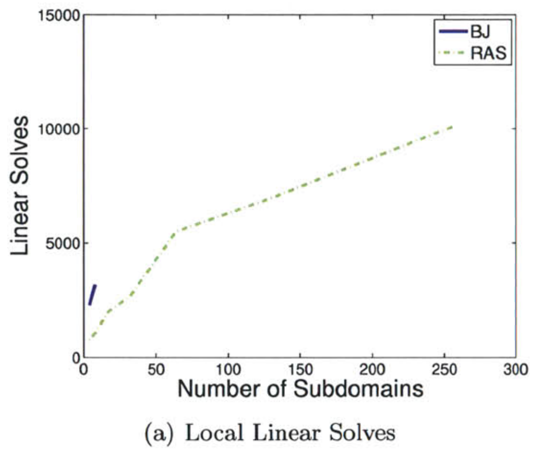
Strong scaling results are presented in order to assess the performance of the additive

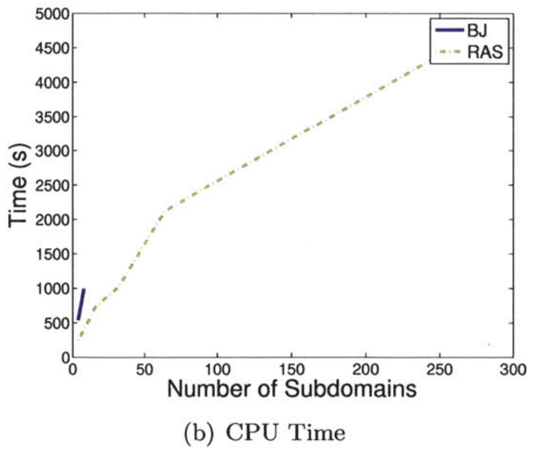




(a) Local Linear Solves (b) CPU Time

Figure 7-10: Weak scaling results for viscous flow over NACA0005, $M_\infty = 0.2$, $\alpha = 2.0°$, $Re_c = 5000$, $p = 2$, 1000 elements per subdomain





(a) Local Linear Solves (b) CPU Time

Figure 7-11: Weak scaling results for viscous flow over NACA0005, $M_\infty = 0.2$, $\alpha = 2.0°$, $Re_c = 5000$, $p = 5$, 250 elements per subdomain

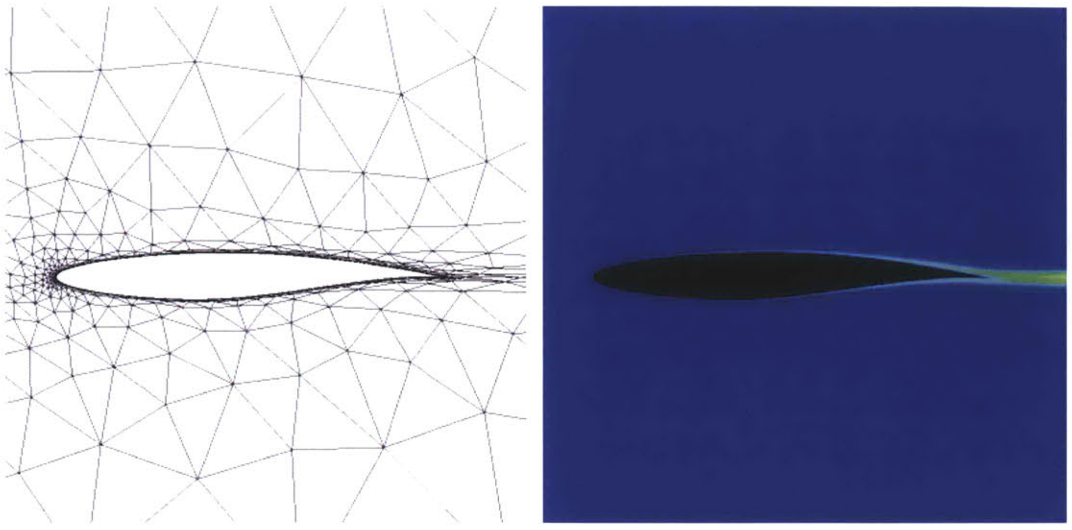

(a) Mesh (b) Eddy Viscosity

Figure 7-12: Grid and flow solution for turbulent flow over RAE2822, $M_\infty = 0.3$, $\alpha = 2.31°$, $Re_c = 6.5 \times 10^6$, $p = 2$

Schwarz preconditioners on small numbers of processors. As a test problem a single iteration of a fixed degree of freedom adjoint-based error estimation/adaptation procedure is performed [92]. A single iteration of the adaptation procedure involves a flow solution and linear adjoint solve at $p = 2$, and 10 Newton iterations at $p = 3$ followed by a $p = 3$ adjoint solve. At each iteration of the non-linear solution procedure the linear system is solved to a relative tolerance of 10^{-4}, while the linear adjoint problems are solved to a tolerance of 10^{-10}.

Figure 7-13 shows the number of linear iterations required throughout the solution procedure as well as the parallel speed-up for meshes with 1000, 4000 and 16000 elements. As with the previous test cases, the restricted additive Schwarz preconditioner performs superiorly to the subdomain-wise block Jacobi preconditioner. Both preconditioners show an increase in the number of linear iterations required to achieve convergence as the number of subdomains increases. However, the increase in the number of linear iterations is much more severe using subdomain-wise block-Jacobi preconditioner than the restricted additive Schwarz preconditioner. In particular, using the subdomain-wise block-Jacobi preconditioner the number of iterations jumps by a factor of two going from 1 to 2 processors. The increase in the number of linear iterations is reflected in the reduced parallel speed-up using the subdomain-wise block-Jacobi preconditioner such that the parallel efficiency (Speed-Up/#Processors) on the 16000 element grid is less than 50%. On the other hand, using the restricted additive Schwarz preconditioner a parallel efficiency of greater that 75% is observed using up to 30 processors.

Finally, weak scaling results are presented for the RAE2822 test case. Figure 7-14 plots the number of local linear solves and CPU time required to obtain a $p = 2$ solution with approximately 1000 elements per subdomain. As in the previous test cases, the restricted additive Schwarz preconditioner performs better than the non-overlapping block-Jacobi preconditioner. Using either preconditioner the number of iterations grows with increasing number of subdomains. However, the growth in the number of iterations is less for this test case than in the previous test cases. This is likely due to the fact that a much less aggressive pseudo-transient continuation scheme is used resulting in many more non-linear iterations, but which are easier to solve due to the large temporal component added to the linear system. It is interesting to note that both the number of iterations and CPU time decreases going from 2 to 4 processors. This reduction in CPU time is attributable to a significant reduction in non-linear iterations as the solution procedure has difficulty obtaining a steady-state solution on the coarsest mesh, due to insufficient resolution on this mesh.

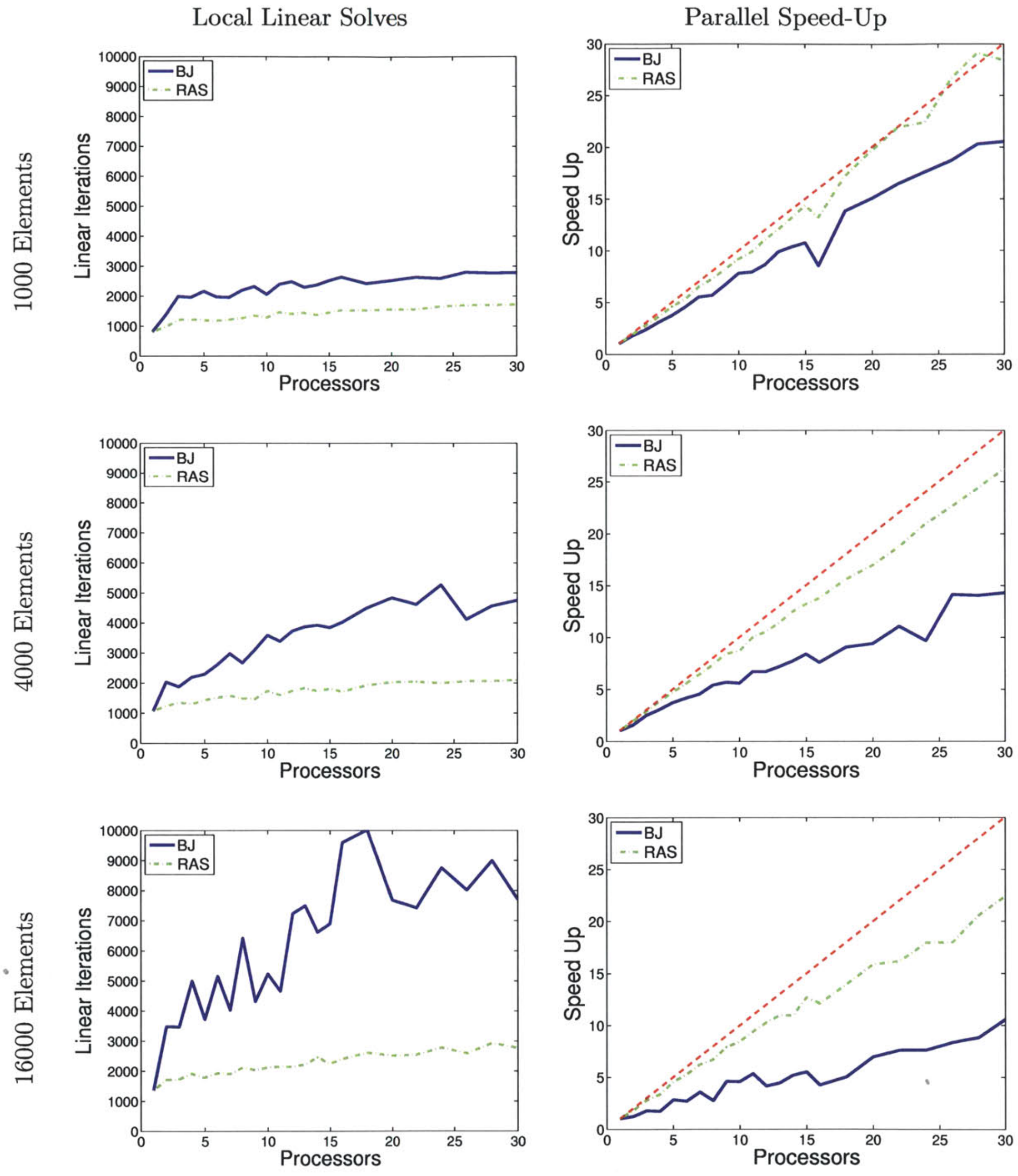
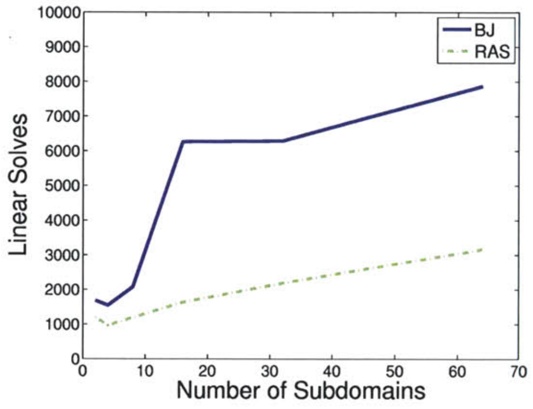
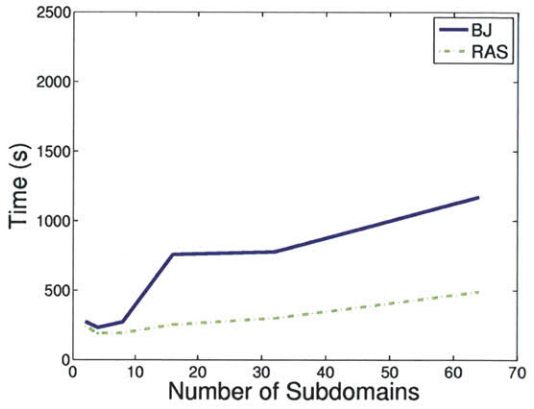


Figure 7-13: Strong scaling results for turbulent flow over RAE2822, $M_\infty = 0.3$, $\alpha = 2.31°$, $Re_c = 6.5 \times 10^6$, $p = 2$ adaptation step



(a) Local Linear Solves



(b) CPU Time

Figure 7-14: Weak scaling results for turbulent flow over RAE2822, $M_\infty = 0.3$, $\alpha = 2.31°$, $Re_c = 6.5 \times 10^6$, $p = 2$, 1000 elements per subdomain

Chapter 8

Conclusions

8.1 Summary and Conclusions

This thesis presented domain decomposition preconditioners for the solution of higher-order discretizations of compressible flows.

In Chapter 3, an additive Schwarz preconditioner was developed with coarse space based on local harmonic extensions. Using several scalar model problems it was shown that a coarse space is necessary for high Peclet number flows solved on anisotropic meshes.

In Chapter 4, a BDDC preconditioner was developed for the HDG discretization. For a second-order elliptic problem, it was proven that the condition number of the preconditioned system is independent of the number of subdomains and only weakly dependent upon the number of elements per subdomain and the solution order. Using a Robin-Robin interface condition, the BDDC preconditioner was extended to the solution of a scalar advection-diffusion problem. Numerical results show that the BDDC preconditioner performs superiorly to the additive Schwarz preconditioners with or without coarse space for the solution of high Peclet number flows on unstructured anisotropic meshes.

In Chapter 5, an inexact BDDC preconditioner was developed based on an incomplete factorization and a p-multigrid type coarse grid correction. It was shown that the incomplete factorization of possibly singular systems corresponding to local Neumann problems results in a non-singular preconditioner. Numerical results show that the inexact BDDC preconditioner with well-designed local solvers converges in similar number of iterations as the exact BDDC method, with significantly reduced CPU time.

Chapter 6 presented analysis of domain decomposition preconditioners for the linearized Euler and Navier-Stokes systems. A one-dimensional analysis showed that domain decomposition methods do not converge in a finite number of iterations for the hyperbolic Euler system if reflecting boundary conditions are imposed at domain boundaries. A Fourier analysis was presented to analyze the performance of the Robin-Robin algorithm for the Euler system. Optimized interface conditions were developed in the analytical and discrete setting by coupling the two acoustic modes. Numerical results showed the optimized interface condition resulted in significant improvements in the performance of the BDDC preconditioner. For inviscid and high-Reynolds number flows the BDDC preconditioner performed similarly to the additive Schwarz preconditioner with coarse space. It was shown that for very low Reynolds number flows, the performance of the BDDC preconditioner degrades due to the viscous matrix being rank deficient.

In Chapter 7, numerical results were presented to evaluate the performance of the additive Schwarz and BDDC preconditioners for several fundamental aerodynamic flows. The numerical results presented showed that a coarse space becomes important as the number of subdomains is increased. For small numbers of subdomains, the use of the ASM_A and optimized BDDC preconditioners is not justified, as the additional cost of the factorization is not offset by reduced time in the GMRES solve. As the number of subdomain is increased, the coarse space becomes more important and ASM_A and optimized BDDC preconditioners with coarse spaces perform better than the ASM preconditioner. In general, the ASM_A and optimized BDDC preconditioners perform similarly for most test cases. Thus the optimized BDDC preconditioner may be seen as an alternative approach to the ASM_A approach.

Chapter 7 also presented numerical results for the DG discretization using additive Schwarz preconditioners with and without overlap. The numerical results showed that an overlap consisting only of a single layer of elements results in significantly improved parallel performance over the non-overlapping case. The performance of the overlapping preconditioner degrades as the number of processors becomes large and a coarse space may be necessary for large numbers of processors:

8.2 Recommendations and Future Work

While the BDDC algorithm has shown significant benefit for scalar advection-diffusion problems solved on structured meshes, the performance for the Euler and Navier-Stokes systems on unstructured anisotropic meshes has been much less satisfactory despite the effort to optimized the interface conditions. Several issues need to be resolved before BDDC may be considered sufficiently mature to be used for practical aerodynamic simulations.

Unstructured Anisotropic Meshes

Each of the different domain decomposition methods presented performed significantly worse for problems involving unstructured anisotropic meshes as compared with structured meshes. As the performance degradation was observed even for scalar advection-diffusion problems, initial investigation into the root causes of the loss in performance due to unstructured meshes should focus on the scalar case.

Optimized Interface Conditions

The optimized interface conditions for the Euler system has been shown empirically to improve the convergence of the BDDC algorithm. However, a complete understanding of the mechanism by which the optimized interface conditions improve the transmission of information across subdomain interfaces is still required. A more complete understanding of the optimized interface conditions may allow for the simple extension of these conditions to the three-dimensional Euler equations, as well as to other hyperbolic systems.

Rank-deficient Viscous Problems

In the low Reynolds number limit the performance of the BDDC algorithm degrades relative to the ASM_A preconditioner. This degradation in performance was attributed to the rank-deficiency of the viscous tensor for the Navier-Stokes equations. Further work is required in order to develop and analyze methods for these incompletely parabolic systems.

BDDC for DG

While the BDDC algorithm has been extended to a large class of DG discretizations for second order elliptic problems, the BDDC algorithm has yet to be successfully applied to the solution of the DG discretization of convection-dominated flows. In particular, future work is required in order to extend the BDDC algorithm to the solution of DG discretizations of the scalar advection-diffusion equations.

Bibliography

[1] Achdou, Y., Tallec, P. L., Nataf, F., and Vidrascu, M. "A Domain Decomposition Preconditioner for an Advection-Diffusion Problem." *Computer Methods in Applied Mechanics and Engineering*, 184:145–170, 2000.

[2] Amdahl, G. "Validity of the single processor approach to achieving large scale computing capabilities." In *AFIPS Conference Proceedings*, volume 30, pages 483–485. AFIPS Press, Reston, Va, 1967.

[3] Anderson, W., Rausch, R., and Bonhaus, D. "Implicit multigrid algorithms for incompressible turbulent flows on unstructured grids." AIAA 1995-1740, 1995.

[4] Anderson, W. K., Gropp, W. D., Kaushik, D. K., Keyes, D. E., and Smith, B. F. "Achieving High Sustained Performance in an Unstructured Mesh CFD Application." In *Proceedings of SC99*, pages 69–80. Portland, OR, 1999.

[5] Antonietti, P. F. and Ayuso, B. "Schwarz Domain Decomposition Preconditioners for Discontinuous Galerkin Approximations of Elliptic Problems: Non-overlapping case." *Mathematical Modeling and Numerical Analysis*, 41(1):21–54, 2007.

[6] Antonietti, P. F. and Ayuso, B. "Class of Preconditioners for Discontinuous Galerkin Approximation of Elliptic Problems." In *Domain Decomposition Methods in Science and Engineering*, volume 60, pages 185–192. Springer Berlin, 2008.

[7] Antonietti, P. F. and Ayuso, B. "Two-level Schwarz Preconditioners for Super Penalty Discontinuous Galerkin Methods." In *Communications in Computational Physics*, volume 5, pages 398–412, 2009.

[8] Arnold, D. N., Brezzi, F., Cockburn, B., and Marini, L. D. "Unified analysis of discontinuous Galerkin methods for elliptical problems." *SIAM Journal on Numerical Analysis*, 39(5):1749–1779, 2002.

[9] Barter, G. and Darmofal, D. "Shock Capturing with Higher-order, PDE-Based Artificial Viscosity." AIAA 2007-3823, 2007.

[10] Barth, T. J., Chan, T. F., and Tang, W.-P. "A Parallel Non-Overlapping Domain-Decomposition Algorithm for Compressible Fluid Flow Problems on Triangulated Domains." *Contemporary Mathematics*, 218:23–41, 1998.

[11] Bassi, F., Crivellini, A., Rebay, S., and Savini, M. "Discontinuous Galerkin solution of the Reynolds averaged Navier-Stokes and k-ω turbulence model equations." *Computers & Fluids*, 34:507–540, May-June 2005.

[12] Bassi, F. and Rebay, S. "High-order accurate discontinuous finite element solution of the 2D Euler equations." *Journal of Computational Physics*, 138(2):251–285, 1997.

[13] Bassi, F. and Rebay, S. "A High-order discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations." *Journal of Computational Physics*, 131:267–279, 1997.

[14] Bassi, F. and Rebay, S. "GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations." In Cockburn, K. and Shu, editors, *Discontinuous Galerkin Methods: Theory, Computation and Applications*, pages 197–208. Springer, Berlin, 2000.

[15] Bassi, F. and Rebay, S. "Numerical evaluation of two discontinuous Galerkin methods for the compressible Navier-Stokes equations." *International Journal for Numerical Methods in Fluids*, 40:197–207, 2002.

[16] Bassi, F., Crivellini, A., Ghidoni, A., and Rebay, S. "High-order discontinuous Galerkin discretization of transonic turbulent flows." AIAA 2009-180, 2009.

[17] Benzi, M., Szyld, D. B., and v.Duin, A. "Orderings for incomplete factorization preconditioning of nonsymmetric problems." *SIAM Journal on Scientific Computing*, 20 (5):1652–1670, 1999.

[18] Bhardwaj, M., Day, D., Farhat, C., Lesoinne, M., Pierson, K., and Rixen, D. "Application of the FETI method to ASCI problems - scalability results on 1000 processors and discussion on highly heterogeneous problems." *Int. J. Numer. Meth. Engng*, 47: 513–535, 2000.

[19] Bhardwaj, M., Pierson, K., Reese, G., Walsh, T., Day, D., Alvin, K., Peery, J., Farhat, C., and Lesoinne, M. "Salinas: A scalable software for high-performance structural and solid mechanics simulations." In *Proceedings of the 2002 ACM/IEEE conference on supercomputing*, pages 35–53, Baltimore, MD, 2002.

[20] Blanco, M. and Zingg, D. W. "A Fast Solver for the Euler Equations on Unstructured Grids using a Newton-GMRES Method." AIAA 1997-0331, January 1997.

[21] Bourgat, J.-F., Glowinski, R., Tallec, P. L., and Vidrascu, M. "Variational formulation and algorithm for trace operator in domain decomposition calculations." In Chan, T., Glowinski, R., Periaux, J., and Widlund, O., editors, *Domain decomposition methods. Second international symposium on domain decomposition methods*, pages 3–16. SIAM, 1988.

[22] Brenner, S. C. "Two-level Additive Schwarz Preconditioners for Nonconforming Finite Element Methos." *Mathematics of Computation*, 65(215):897–921, 1996.

[23] Brezzi, F., Manzini, G., Marini, D., Pietra, P., and Russo, A. "Discontinuous Galerkin Approximations for Elliptic Problems." *Numerical Methods for Partial Differential Equations*, 16(4):365–378, July 2000.

[24] Buoni, J. J. "Incomplete Factorization of Singular Matrices." *SIAM Journal on Algebraic & Discrete Methods*, 7(2):193–198, 1986.

[25] Cai, X.-C., Farhat, C., and Sarkis, M. "Schwarz Methods for the Unsteady Compressible Navier-Stokes Equations on Unstructured Meshes." In *Domain Decomposition Methods in Science and Engineering.* John Wiley & Sons, 1997.

[26] Cai, X.-C., Gropp, W. D., Keyes, D. E., and Tidriri, M. D. "Newton-Krylov-Schwarz Methods in CFD." pages 17–30. Proceedings of the International Workshop on Numerical Methods for the Navier-Stokes Equations, 1995.

[27] Cai, X.-C. "An Additive Schwarz Algorithm for Nonselfadjoint Elliptic Equations." In *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 232–244, Philidelphia, 1990.

[28] Cai, X.-C. "Additive Schwarz algorithms for parabolic convection-diffusion equations." *Numerische Mathematik*, 60:41–61, 1991.

[29] Cai, X.-C. "An Optimal two-level overlapping domain decomposition method for elliptic problems in two and three dimensions." *SIAM Journal on Scientific Computing*, 14(1):239–247, 1993.

[30] Cai, X.-C. "Multiplicative Schwarz methods for parabolic problemc." *SIAM Journal on Scientific Computing*, 15(3):587–603, 1994.

[31] Cai, X.-C. "A Family Of Overlapping Schwarz Algorithms For Nonsymmetric And Indefinite Elliptic Problems." In *Domain-Based Parallelism and Problem Decomposition Methods in Computational Science and Engineering*, pages 1–19. SIAM, 1995.

[32] Cai, X.-C., Farhat, C., and Sarkis, M. "Variable degree Schwarz methods for the implicit solution of unsteady compressible Navier-Stokes equations on two-dimensional unstructured meshes." ICASE 96-48, 1996.

[33] Cai, X.-C., Farhat, C., and Sarkis, M. "A Minimum Overlap Restricted Additive Schwarz Preconditioner and Applications in 3D Flow Simulations." *Contemporary Mathematics*, 218:479–485, 1998.

[34] Cai, X.-C., Gropp, W. D., and Keyes, D. E. "A Comparison of Some Domain Decomposition and ILU Preconditioned Iterative Methods for Nonsymmetric Elliptic Problems." *Journal of Computational Physics*, 157:1765–1774, 2000.

[35] Cai, X.-C. and Sarkis, M. "Restricted Additive Schwarz Preconditioner for general sparse linear systems." *SIAM Journal on Scientific Computing*, 21(2):792–797, 1999.

[36] Casarin, M. A. "Quasi-Optimal Schwarz Methods for the Conforming Spectral Element Discretization." *SIAM Journal on Numerical Analysis*, 34(6):2482–2502, 1997.

[37] Cockburn, B., Karniadakis, G., and Shu, C. "The development of discontinuous Galerkin methods." In *Lecture Notes in Computational Science and Engineering*, volume 11. Springer, 2000.

[38] Cockburn, B. and Shu, C. W. "The local discontinuous Galerkin method for time-dependent convection-diffusion systems." *SIAM Journal on Numerical Analysis*, 35 (6):2440–2463, December 1998.

[39] Cockburn, B., Dong, B., and Guzman, J. "A Superconvergent LDG-Hybridizable Galerkin method for second-order elliptic problems." *Mathematics of Computation*, 77 (264):1887–1916, 2008.

[40] Cockburn, B., Gopalakrishnan, J., and Lazarov, R. "Unified hybridization of discontinuous Galerkin, mixed, and continuous Galerkin methods for second order elliptic problems." *SIAM Journal on Numerical Analysis*, 47(2):1319–1365, 2009.

[41] Cockburn, B., Guzman, J., and Wang, H. "Superconvergent discontinuous Galerkin methods for second-order elliptic problems." *Mathematics of Computation*, 78(265): 1–24, 2009.

[42] Cockburn, B. and Shu, C.-W. "Runge-Kutta discontinuous Galerkin methods for convection-dominated problems." *Journal of Scientific Computing*, pages 173–261, 2001.

[43] Cowsar, L. C., Mandel, J., and Wheeler, M. F. "Balancing Domain Decomposition for mixed finite elements." *Mathematics of Computation*, 64(211):989–1015, 1995.

[44] DeRoeck, Y.-H. and LeTallec, P. "Analysis and Test of a Local Domain-Decomposition Preconditioner." In *Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 112–128, Philidelphia, PA, 1991. SIAM.

[45] Diosady, L. T. "A linear multigrid preconditioner for the solution of the Navier-Stokes equations using a discontinuous Galerkin discretization." Masters thesis, Mass. Inst. of Tech., CDO, May 2007.

[46] Diosady, L. T. and Darmofal, D. L. "Preconditioning methods for discontinuous Galerkin solutions of the Navier-Stokes equations." *Journal of Computational Physics*, 228:3917–3935, 2009.

[47] Dohrmann, C. R. "A Preconditioner for Substructuring Based on Constrained Energy Minimization." *SIAM Journal on Scientific Computing*, 25(1):246–258, 2003.

[48] Dohrmann, C. R. "An approximate BDDC preconditioner." *Numerical Linear Algebra with Applications*, 14:149–168, 2007.

[49] Dolean, V., Lanteri, S., and Nataf, F. "Convergence analysis of additive Schwarz for the Euler equations." *Applied Numerical Mathematics*, 49:153 – 186, 2004.

[50] Dolean, V. and Nataf, F. "A new domain decomposition method for compressible Euler equations." *Mathematical Modeling and Numerical Analysis*, 40(4):689–704, 2006.

[51] Dolean, V., Nataf, F., and Rapin, G. "New constructions of domain decomposition methods for systems of PDEs." *Comptes Rendus Mathematique*, 340(9):693 – 696, 2005.

[52] Dryja, M. and Widlund, O. "An additive variant of the Schwarz alternating method for the case of many subregions." Tech report 339, Department of Computer Science, Courant Institute, 1987.

[53] Farhat, C., Lesoinne, M., LeTallec, P., Pierson, K., and Rixen, D. "FETI-DP: a dual-primal unified FETI method - part I: A faster alternative to the two-level FETI method." *International Journal for Numerical Methods in Engineering*, 50:1523–1544, 2001.

[54] Farhat, C., Mandel, J., and Roux, F.-X. "Optimal convergence properties of the FETI domain decomposition method." *Computer Methods in Applied Mechanics and Engineering*, 115:365–385, 1994.

[55] Feng, X. and Karakashian, O. A. "Two-level Additive Schwarz methods for a Discontinuous Galerkin Approximation of second order elliptic problems." *SIAM Journal on Numerical Analysis*, 39(4):1343–1365, 2002.

[56] Fidkowski, K. J. and Darmofal, D. L. "Development of a higher-order solver for aerodynamic applications." AIAA 2004-0436, January 2004.

[57] Fidkowski, K. J., Oliver, T. A., Lu, J., and Darmofal, D. L. "p-Multigrid solution of high-order discontiunous Galerkin discretizations of the compressible Navier-Stokes equations." *Journal of Computational Physics*, 207(1):92–113, 2005.

[58] Fragakis, Y. and Papadrakakis, M. "The mosaic of high performance domain decomposition methods for structural mechanics: formulation, interrelation and numerical efficiency of primal and dual methods." *Computer Methods in Applied Mechanics and Engineering*, 192:3799–3830, 2003.

[59] Gropp, W., Kaushik, D. K., Smith, B. F., and Keyes, D. E. "Analyzing the parallel scalability of an implicit unstructured mesh CFD code." In *HiPC '00: 7th Int. Conf. on HPC*, pages 395–404. Springer-Verlag, 2000.

[60] Gropp, W., Keyes, D., Mcinnes, L. C., and Tidriri, M. D. "Globalized Newton-Krylov-Schwarz algorithms and software for parallel implicit CFD." *International Journal of High Performance Computing Applications*, 14(2):102–136, 2000.

[61] Gustafson, J. L. "Reevaluating Amdahl's Law." *Communications of the ACM*, 31: 532–533, 1988.

[62] Gustafsson, B. and Sundstrom, A. "Incompletely Parabolic Problems in Fluid Dynamics." *SIAM Journal on Applied Mathematics*, 35(2):343–357, 1978.

[63] Hecht, F. "BAMG: Bidimensional Anisotropic Mesh Generator." 1998. `http://www-rocq1.inria.fr/gamma/cdrom/www/bamg/eng.htm`.

[64] Hicken, J. E. and Zingg, D. W. "A Parallel Newton-Krylov flow solver for the Euler equations on multi-block grids." AIAA 2007-4333, 2007.

[65] Karypis, G. "ParMETIS: Parallel Graph Partitioning and Sparse Matrix Ordering Library." 2006. `http://glaros.dtc.umn.edu/gkhome/views/metis/parmetis`.

[66] Kelley, C. T. and Keyes, D. E. "Convergence analysis of pseudo-transient continuation." *SIAM Journal on Numerical Analysis*, 35(2):508–523, 1998.

[67] Klawonn, A. and Rheinbach, O. "Inexact FETI-DP methods." *International Journal for Numerical Methods in Engineering*, 69:284–307, 2007.

[68] Klawonn, A. and Widlund, O. B. "Dual-primal FETI methods for linear elasticity." *Communications on Pure and Applied Mathematics*, 59:1523–1572, 2006.

[69] Knoll, D. A. and Keyes, D. E. "Jacobian-free Newton-Krylov methods: a survey of approaches and applications." *Journal of Computational Physics*, 193(1):357–397, 2004.

[70] Lasser, C. and Toselli, A. "An Overlapping Domain Decomposition Preconditioner for a Class of Discontinuous Galerkin Approximations of Advection-Diffusion Problems." *Mathematics of Computation*, 72:1215–1238, 2003.

[71] Li, J. and Widlund, O. B. "FETI-DP, BDDC, and Block Cholesky Methods." *International Journal for Numerical Methods in Engineering*, 66:250–271, 2006.

[72] Li, J. and Widlund, O. B. "On the use of inexact subdomain solvers for BDDC algorithms." *Computer Methods in Applied Mechanics and Engineering*, 196:1415–1428, 2007.

[73] Mandel, J. "Balancing Domain Decomposition." *Communications in Numerical Methods in Engineering*, 9:233–241, 1993.

[74] Mandel, J. and Brezina, M. "Balancing Domain decomposition for problems with large jumps in coefficients." *Mathematics of Computation*, 65(216):1387–1401, 1996.

[75] Mandel, J. and Dohrmann, C. R. "Convergence of a Balancing Domain Decomposition by constraints and energy minimization." *Numerical Linear Algebra with Applications*, 10:639–659, 2003.

[76] Mandel, J., Dohrmann, C. R., and Tezaur, R. "An Algebraic Theory for Primal and Dual Substructuring Methods by Constraints." *Applied Numerical Mathematics*, 54: 167–193, 2005.

[77] Mandel, J. and Sousedik, B. "Adaptive Selection of face coarse degrees of freedom in the BDDC and the FETI-DP iterative substructuring methods." *Computer Methods in Applied Mechanics and Engineering*, 196:1389–1399, 2007.

[78] Mandel, J. and Sousedik, B. "Multispace and Multilevel BDDC." *Computing*, 83: 55–85, 2008.

[79] Mandel, J., Sousedik, B., and Dohrmann, C. R. "On Multilevel BDDC." *Lecture Notes in Computational Science and Engineering*, 60:287–294, 2008.

[80] Mandel, J. and Tezaur, R. "On the convergence of a dual-primal substructuring method." *Numerische Mathematik*, 88:543–558, 2001.

[81] Mavriplis, D. J. "Unstructured grid techniques." *Annual Review of Fluid Mechanics*, 29:473–514, 1997.

[82] Mavriplis, D. J. "Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes." *Journal of Computational Physics*, 145:141–165, 1998.

[83] Mavriplis, D. J. "On convergence acceleration techniques for unstructured meshes." AIAA 1998-2966, 1998.

[84] Mavriplis, D. J. "Results from the 3rd Drag Prediction Workshop using the NSU3D unstructured mesh solver." AIAA 2007-256, 2007.

[85] Mavriplis, D. J. and Jameson, A. "Multigrid Solution of the Navier-Stokes Equations on Triangular Meshes." *AIAA Journal*, 28:1415–1425, 1990.

[86] Mavriplis, D. J. and Venkatakrishnan, V. "A 3D agglomeration multigrid sovler for the Reynolds-averaged Navier-Stokes equations on unstructured meshes." *International Journal for Numerical Methods in Fluids*, 23(6):527–544, 1996.

[87] Mavriplis, D., Nastase, C., Shahbazi, K., Wang, L., and Burgess, N. "Progress in high-order discontinuous Galerkin methods for aerospace applications." AIAA 2009-601, 2009.

[88] Mavriplis, D. J. "An assessment of linear versus nonlinear multigrid methods for unstructured mesh solvers." *Journal of Computational Physics*, 175(1):302–325, 2002.

[89] Mavriplis, D. J., Darmofal, D., Keyes, D., and Turner, M. "Petaflops opportunities for the NASA fundamental aeronautics program." AIAA 2007-4084, 2007.

[90] Mavriplis, D. J. and Pirzadeh, S. "Large-scale parallel unstructured mesh computations for 3D high-lift analysis." *AIAA Journal of Aircraft*, 36:987–998, 1999.

[91] Meijerink, J. and v. d.Vorst, H. "An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is a Symmetric M-Matrix." *Mathematics of Computation*, 31(137):148–162, 1977.

[92] Modisette, J. M. *An Automated Reliable Method for Two-Dimensional Reynolds-averaged Navier-Stokes Simulations.* PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, 2011.

[93] Nastase, C. R. and Mavriplis, D. J. "High-order discontinuous Galerkin methods using an *hp*-multigrid approach." *Journal of Computational Physics*, 213(1):330–357, 2006.

[94] Nastase, C. R. and Mavriplis, D. J. "A Parallel *hp*-Multigrid Solver for Three-Dimensional Discontinuous Galerkin Discretizations of the Euler Equations." AIAA 2007-0512, 2007.

[95] Nejat, A. and Ollivier-Gooch, C. "Effect of Discretization Order on Preconditioning and Convergence of a Higher-Order Unstructured Newton-Krylov Solver for Inviscid Compressible Flows." AIAA 2007-0719, January 2007.

[96] Nguyen, N., Peraire, J., and Cockburn, B. "An implicit high-order hybridizable discontinuous Galerkin method for linear convection-diffusion equations." *Journal of Computational Physics*, 228(9):3232–3254, 2009.

[97] Nguyen, N., Persson, P.-O., and Peraire, J. "RANS solutions using high order discontinuous Galerkin methods." AIAA 2007-0914, 2007.

[98] Oliver, T. and Darmofal, D. "An unsteady adaptation algorithm for discontinuous Galerkin discretizations of the RANS equations." AIAA 2007-3940, 2007.

[99] Oliver, T. A. *A Higher-Order, Adaptive, Discontinuous Galerkin Finite Element Method for the Reynolds-averaged Navier-Stokes Equations.* PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, June 2008.

[100] Peraire, J., Nguyen, N., and Cockburn, B. "A Hybridizable Discontinuous Galerkin Method for the Compressible Euler and Navier-Stokes Equations." AIAA 2010-363, 2010.

[101] Peraire, J. and Persson, P.-O. "The compact discontinuous Galerkin (CDG) method for elliptic problems." *SIAM Journal on Scientific Computing*, 30(4):1806–1824, 2008.

[102] Persson, P.-O. "Scalable Parallel Newton-Krylov Solvers for Discontinuous Galerkin discretizations." AIAA 2009-606, 2009.

[103] Persson, P.-O. and Peraire, J. "Newton-GMRES preconditioning for Discontinuous Galerkin discretizations of the Navier-Stokes equations." *SIAM Journal on Scientific Computing*, 30(6):2709–2722, 2008.

[104] Pierson, K. H., Reese, G. M., Bhardwaj, M. K., Walsh, T. F., and Day, D. M. "Experiences with FETI-DP in a production level finite element application." Sandia National Laboratories SAND2002-1371, 2002.

[105] Pueyo, A. and Zingg, D. W. "An Efficient Newton-GMRES Solver for Aerodynamic Computations." AIAA 1997-1955, June 1997.

[106] Quarteroni, A. and Valli, A. *Domain Decomposition Methods for Partial Differential Equations.* Oxford, New York, 1999.

[107] Roe, P. L. "Approximate Riemann solvers, parameter vectors, and difference schemes." *Journal of Computational Physics*, 43(2):357–372, 1981.

[108] Roe, P. and Turkel, E. "The quest for diagonalization of differential systems." In Salas, M. D., editor, *Barriers and Challenges in Computational Fluid Dynamics.* Kluwer Academic Publishers, Boston, 1998.

[109] Saad, Y. "A Flexible Inner-Outer Preconditioned GMRES Algorithm." *SIAM Journal on Scientific Computing*, 14(2):461–469, 1993.

[110] Saad, Y. "ILUT: a Dual Threshold Incomplete LU Factorization." *Numerical Linear Algebra with Applications*, 1(4):387–402, 1994.

[111] Shahbazi, K. "An explicit expression for the penalty parameter of the interior penalty method." *Journal of Computational Physics*, 205(2):401–407, 2005.

[112] Si, H. "TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator." Weierstrass Institute for Applied Analysis and Stochastics, 2005. `http://tetgen.berlios.de`.

[113] Smith, B., Bjorstad, P., and Gropp, W. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Cambridge University Press, New York, NY, 1996.

[114] Spalart, P. R. and Allmaras, S. R. "A one-equation turbulence model for aerodynamics flows." AIAA 1992-0439, January 1992.

[115] Spalart, P. R. and Allmaras, S. R. "A one-equation turbulence model for aerodynamics flows." *La Recherche Aérospatiale*, 1:5–21, 1994.

[116] Tallec, P. L., Roeck, Y. D., and Vidrascu, M. "Domain Decomposition Methods for large linearly elliptic three-dimensional problems." *Journal of Computational and Applied Mathematics*, 34:93–117, 1991.

[117] Toselli, A. "FETI Domain Decomposition Methods for Scalar Advection-Diffusion Problems." *Computer Methods in Applied Mechanics and Engineering*, 190:5759–5776, 2001.

[118] Toselli, A. and Widlund, O. *Domain Decomposition Methods Algorithm and Theory.* Springer-Verlag, 2005.

[119] Tu, X. "A BDDC algorithm for flow in porous media with a hybrid finite element discretization." *Electronic Transactions on Numerical Analysis*, 26:146–160, 2007.

[120] Tu, X. "Three-level BDDC in three dimensions." *SIAM Journal on Scientific Computing*, 29(4):1759–1780, 2007.

[121] Tu, X. and Li, J. "A Balancing Domain Decomposition Method by Constraints for Advection-Diffusion Problems." *Communications in Applied Mathematics and Computational Science*, 3(1):25–60, 2008.

[122] Venkatakrishnan, V. "Implicit Scheme and Parallel Computing in Unstructured Grid CFD." ICASE 95-28, 1995.

[123] Yano, M. "Massively Parallel Solver for the High-Order Galerkin Least-Squares Method." Masters thesis, Mass. Inst. of Tech., CDO, May 2009.

[124] Yano, M. and Darmofal, D. "BDDC Preconditioning for High-Order Galerkin Least Squares Methods using Inexact Solvers." *Computer Methods in Applied Mechanics and Engineering*, 2010.

Appendix A

2D Euler Analysis: Infinite Domain

In this section Fourier analysis is used to estimate the convergence rate of the Robin-Robin algorithm in the case of two infinite subdomains. The basic Robin-Robin algorithm is shown to be equivalent to two iterations of the classical Schwarz algorithm.

After performing a Fourier transform the linearized Euler equations, (6.15), gives a system of ODEs of the form:

$$\frac{d\tilde{W}}{dx} + \tilde{A}\tilde{W} = 0, \tag{A.1}$$

with

$$\tilde{A} = \begin{bmatrix} \frac{a}{M_x-1} & 0 & 0 & \frac{i\xi}{(M_x-1)\sqrt{2}} \\ 0 & \frac{a}{M_x+1} & 0 & \frac{i\xi}{(M_x+1)\sqrt{2}} \\ 0 & 0 & \frac{a}{M_x} & 0 \\ \frac{i\xi}{M_x\sqrt{2}} & \frac{i\xi}{M_x\sqrt{2}} & 0 & \frac{a}{M_x} \end{bmatrix}, \tag{A.2}$$

where $\xi = c\Delta t\hat{\xi}$, $b = \frac{1}{c\Delta t}$ and $a = b + i\xi M_y$. This system has an eigenvalue decomposition $\tilde{A} = X\tilde{\Lambda}X^{-1}$ with eigenvalues:

$$\mu_{1,2} = \frac{-aM_x \mp R}{1 - M_x^2}, \tag{A.3}$$

$$\mu_{3,4} = \frac{a}{M_x}, \tag{A.4}$$

and eigenvectors:

$$X = \begin{bmatrix} | & | & | & | \\ X_1 & X_2 & X_3 & X_4 \\ | & | & | & | \end{bmatrix} = \begin{bmatrix} -\frac{(R+a)(1+M_x)}{\sqrt{2}} & \frac{(R-a)(1+M_x)}{\sqrt{2}} & 0 & -\frac{i\xi M_x}{\sqrt{2}} \\ \frac{(R-a)(1-M_x)}{\sqrt{2}} & -\frac{(R+a)(1-M_x)}{\sqrt{2}} & 0 & \frac{i\xi M_x}{\sqrt{2}} \\ 0 & 0 & 1 & 0 \\ i\xi(1-M_x^2) & i\xi(1-M_x^2) & 0 & a \end{bmatrix}, \tag{A.5}$$

with $R = \sqrt{a^2 + \xi^2(1-M_x^2)}$. Thus the solution may be expressed as:

$$\tilde{W} = \sum_{j=1}^{4} e^{-\mu_j x} \alpha_j X_j. \tag{A.6}$$

Consider a partition $\mathbb{R}^2$ into two non-overlapping domains: $\Omega_1 = (-\infty, 0] \times \mathbb{R}$ and $\Omega_2 = [0, \infty) \times \mathbb{R}$. The Robin-Robin algorithm to solve for, $\hat{W}$, the state on the interface at $x = 0$, is given by:

Dirichlet Solve:

$$\Omega_i : \begin{cases} \frac{d\tilde{W}_i^k}{dx} + \tilde{A}\tilde{W}_i^k = 0 & \text{in } \Omega_i, \\ \Lambda_{n_i}^- \tilde{W}_i^k = \Lambda_{n_i}^- \hat{W}^k & \text{at } x = 0, \end{cases}$$

Robin Solve:

$$\Omega_i : \begin{cases} \frac{d\tilde{V}_i^k}{dx} + \tilde{A}\tilde{V}_i^k = 0 & \text{in } \Omega_i, \\ \Lambda_{n_i}^- \tilde{V}_i^k = \Lambda_{n_i}^- \hat{V}_i^k & \text{at } x = 0, \\ \frac{1}{2}\Lambda_{n_i}\hat{V}_i^k + |\Lambda_{n_i}|(\tilde{V}_i^k - \hat{V}_i^k) = -\frac{1}{2}|\Lambda_{n_i}|\left(\tilde{W}_1^k + \tilde{W}_2^k - 2\hat{W}^k\right) & \text{at } x = 0, \end{cases}$$

$$\textbf{Update:} \quad \hat{W}^{k+1} = \hat{W}^k + \frac{1}{2}\left(\hat{V}_1^k + \hat{V}_2^k\right). \tag{A.7}$$

Using the eigenvalue decomposition, $\hat{W}^k$ and $\hat{V}_i^k$ may be expressed as:

$$\hat{W}^k = \sum_{j=1}^{4} \hat{\alpha}_j^k X_j \quad \text{and} \quad \hat{V}_i^k = \sum_{j=1}^{4} \hat{\gamma}_{i,j}^k X_j. \tag{A.8}$$

Similarly, $\tilde{W}_i^k$ and $\tilde{V}_i^k$ and may be expressed in terms of the eigenvalue decomposition. However, to ensure the solution remains bounded as $x \to \pm\infty$, only the upstream characteristic is allowed to be non-zero in Ω_1 while only downstream characteristics are non-zero in Ω_2.

Namely,

$$\tilde{W}_1^k = \alpha_{1,1}^k e^{-\lambda_1 x} X_j, \qquad \tilde{W}_2^k = \sum_{j=2}^{4} \alpha_{2,j}^k e^{-\lambda_j x} X_j,$$

$$\tilde{V}_1^k = \gamma_{1,1}^k e^{-\lambda_1 x} X_j, \qquad \tilde{V}_2^k = \sum_{j=2}^{4} \gamma_{2,j}^k e^{-\lambda_j x} X_j.$$

Solving for the coefficients $\alpha_{i,j}$ from the Dirichlet step gives:

$$\begin{bmatrix} \alpha_{1,1}^k \\ \alpha_{2,2}^k \\ \alpha_{2,3}^k \\ \alpha_{2,4}^k \end{bmatrix} = \begin{bmatrix} 1 & -\frac{(R-a)}{(R+a)} & 0 & \frac{i\xi M_x}{(R+a)(1+M_x)} \\ -\frac{(R-a)(a+M_xR)}{(R+a)(a-M_xR)} & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{2i\xi R(1-M_x^2)(1-M_x)}{(R+a)(a-M_xR)} & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{\alpha}_1^k \\ \hat{\alpha}_2^k \\ \hat{\alpha}_3^k \\ \hat{\alpha}_4^k \end{bmatrix}. \tag{A.9}$$

Combining the Robin solve and the solution update gives an iteration matrix of the form:

$$\begin{bmatrix} \hat{\alpha}_1^{k+1} \\ \hat{\alpha}_2^{k+1} \\ \hat{\alpha}_3^{k+1} \\ \hat{\alpha}_4^{k+1} \end{bmatrix} = \begin{bmatrix} T_{11} & 0 & 0 & 0 \\ 0 & T_{22} & 0 & T_{24} \\ 0 & 0 & 0 & 0 \\ 0 & T_{42} & 0 & T_{44} \end{bmatrix} \begin{bmatrix} \hat{\alpha}_1^k \\ \hat{\alpha}_2^k \\ \hat{\alpha}_3^k \\ \hat{\alpha}_4^k \end{bmatrix}, \tag{A.10}$$

with

$$T_{11} = \frac{(R+a)(R(1-3M_x)-a(1+M_x))}{(R+a)^2(1+M_x)}, \tag{A.11}$$

$$T_{22} = \frac{(R+a)(R(1-3M_x)-a(1+M_x))}{(R+a)^2(1+M_x)}, \tag{A.12}$$

$$T_{24} = \frac{i\xi M_x(\xi^2 M_x(1+M_x)-a^2+Ra)}{(R+a)(1+M_x)(R+a)(\xi^2 M_x(1+M_x)-a^2-Ra)}, \tag{A.13}$$

$$T_{42} = \frac{2i\xi R(R-a)(1-M_x^2)}{(R+a)(\xi^2 M_x(1+M_x)-a^2-Ra)}, \tag{A.14}$$

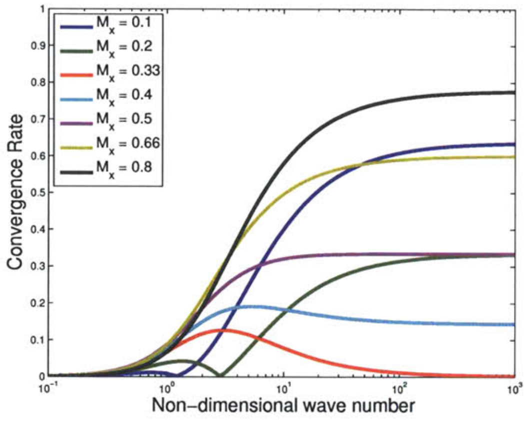
$$T_{44} = \frac{2\xi^2 M_x R(1-M_x)}{(R+a)(\xi^2 M_x(1+M_x)-a^2-Ra)}. \tag{A.15}$$

From (A.10) it is easily verified that the entropy equation (corresponding to the third row) has zero error after one iteration of the Robin-Robin algorithm. Additionally the iteration matrix T is identical to the iteration matrix corresponding to two steps of the classical

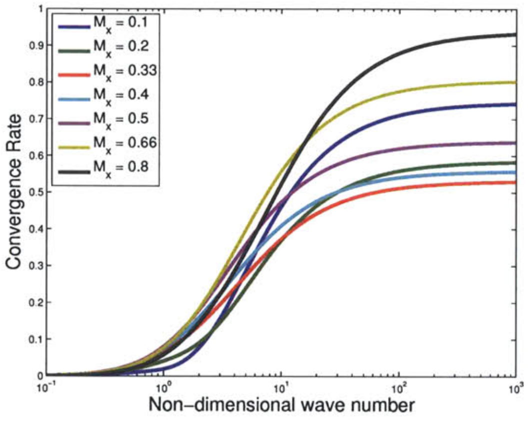
Schwarz algorithm without overlap [49]. The corresponding spectral radius of T is:

$$\rho(T) = \frac{(R+a)(R(1-3M_x) - a(1+M_x))}{(R+a)^2(1+M_x)}. \tag{A.16}$$

From [49], $\rho(T) < 1$, thus Robin-Robin algorithm converges. As the expression (A.16) is quite complex, the convergence rate is plotted versus non-dimensional wave number to provide a better understanding of the convergence rate of the Robin-Robin algorithm. Figure A-1 plots the convergence rate versus non-dimensional wave number varying M_x with $M_y = 0$ and $M_y = 0.5$. As $\xi \to \infty$, the convergence rate approaches a constant value independent of ξ. It is interesting to note that for $M_x = \frac{1}{3}$ and $M_y = 0$ the asymptotic value of the convergence rate is zero, implying that convergence of those error modes is achieved in one iteration.



(a) $M_y = 0.0$



(b) $M_y = 0.5$

Figure A-1: Analytical convergence rate versus wave number, ξ using basic Robin-Robin algorithm on two infinite domains

The convergence rate at any particular flow condition is given by the maximum convergence rate over all wave numbers. Figure A-2 plots the maximum convergence versus M_x for different values of M_y. The convergence rate degrades significantly for very small M_x, with the convergence rate approaching 1 in the limit as $M_x \to 0$. Similarly, the convergence rate approaches 1 as $M \to 1$.

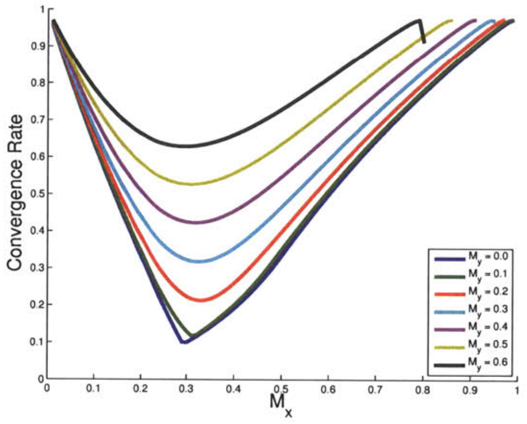


Figure A-2: Analytical convergence rate vs M_x using basic Robin-Robin algorithm on two infinite domains

Appendix B

BDDC for DG Discretizations

In this chapter the BDDC preconditioner is extended to the DG discretization of a second order elliptic problem. A key component for the development and analysis of the BDDC algorithm is a novel perspective presenting the DG discretization as the sum of element-wise "local" bilinear forms. The element-wise perspective leads naturally to the appropriate choice for the subdomain-wise local bilinear forms. Additionally, this new perspective enables a connection to be drawn between the DG discretization and a related continuous finite element discretization. By exploiting this connection, the condition number bound of $\kappa \leq C(1 + \log(p^2H/h))^2$ is proven for the BDDC preconditioned system for a large class of conservative and consistent DG methods considered in the unified analysis of Arnold et al. [8]

Section B.1 gives a classical presentation of the DG discretization. Section B.2 presents the new perspective on the DG discretization. Section B.3 presents the domain decomposition strategy and defines the constraints for the BDDC algorithm. The analysis of the BDDC algorithm in presented in Section B.4, while Section B.5 presents numerical results confirming the analysis.

B.1 DG Discretization

Consider the following second order elliptic equation in a domain $\Omega \subset \mathcal{R}^d$, $d = 2, 3$.

$$\begin{aligned} -\nabla \cdot (\mu \nabla u) &= f && \text{in } \Omega, \\ u &= 0 && \text{on } \partial\Omega, \end{aligned} \tag{B.1}$$

with positive $\mu > 0 \in L^\infty(\Omega)$, $f \in L^2(\Omega)$. Rewrite (B.1) to obtain the following first order system of equations:

$$\begin{aligned} \mu^{-1} q + \nabla u &= 0 \\ \nabla \cdot q &= f && \text{in } \Omega, \\ u &= 0 && \text{on } \partial\Omega. \end{aligned} \tag{B.2}$$

Given the triangulation $\mathcal{T}_h$ define the following finite element spaces:

$$\mathbf{V}_h^p := \{v_h \in \left(\mathbf{L}^2(\Omega)\right)^d : v_h|_\kappa \in (\mathcal{P}^p(\kappa))^d \quad \forall \kappa \in \Omega\}, \tag{B.3}$$

$$W_h^p := \{w_h \in \mathbf{L}^2(\Omega) : w_h|_\kappa \in \mathcal{P}^p(\kappa) \quad \forall \kappa \in \Omega\}. \tag{B.4}$$

Note that traces of functions $u_h \in W_h^p$ are in general double valued on each edge, $e \in \mathcal{E}^i$, with values u_h^+ and u_h^- corresponding to traces from elements κ^+ and κ^- respectively. On $e \in \mathcal{E}^\partial$, associate u_h^+ with the trace taken from the element, $\kappa^+ \in \mathcal{T}_h$, neighbouring e. Consider the following weak form of (B.2): Find $(q_h, u_h) \in \mathbf{V}_h^p \times W_h^p$ such that for all $\kappa \in \mathcal{T}_h$,

$$(\mu^{-1} q_h, v_h)_\kappa - (u_h, \nabla \cdot v_h)_\kappa + \left\langle \hat{u}_h n^+, v_h^+ \right\rangle_{\partial\kappa} = 0 \qquad \forall v_h \in (\mathcal{P}^p(\kappa))^d, \tag{B.5}$$

$$-(q_h, \nabla w_h)_\kappa + \left\langle \hat{q}_h, w_h^+ n^+ \right\rangle_{\partial\kappa} = (f, w_h)_\kappa \qquad \forall w_h \in \mathcal{P}^p(\kappa), \tag{B.6}$$

where $(\cdot,\cdot)_\kappa := \int_\kappa$ and $\langle \cdot,\cdot \rangle_{\partial\kappa} := \int_{\partial\kappa}$. Superscript $^+$ is used to explicitly denote values on $\partial\kappa$, taken from κ. For all $w_h \in W_h^p$, $\hat{w}_h = \hat{w}_h(w_h^+, w_h^-)$ is a single valued numerical trace on $e \in \mathcal{E}^i$, while $\hat{w}_h = 0$ for $e \in \mathcal{E}^\partial$. Note that $\hat{u}_h = 0$ on $e \in \mathcal{E}^\partial$, corresponds to weakly enforced homogeneous boundary conditions on $\partial\Omega$. Similarly $\hat{q} = \hat{q}(\nabla u_h^+, \nabla u_h^-, u_h^+, u_h^-)$ is a single valued numerical flux on $e \in \mathcal{E}$. Using integration by parts on (B.5) gives

$$(\mu^{-1}q_h, v_h)_\kappa + (\nabla u_h, v_h)_\kappa - \langle (u_h^+ - \hat{u}_h)n^+, v_h^+ \rangle_{\partial\kappa} = 0 \qquad \forall v_h \in (\mathcal{P}^p(\kappa))^d, \tag{B.7}$$

such that q_h may be eliminated locally on each element to give:

$$q_h = -\mu(\nabla u_h - r_\kappa((u_h^+ - \hat{u}_h)n^+)), \tag{B.8}$$

where $r_\kappa(\phi) \in (\mathcal{P}^p(\kappa))^d$ is defined by:

$$(r_\kappa(\phi), v_h)_\kappa = \langle \phi, v_h^+ \rangle_{\partial\kappa} \qquad \forall v_h \in (\mathcal{P}^p(\kappa))^d. \tag{B.9}$$

Replacing v_h with ∇w_h and substituting into (B.6) gives the primal formulation

$$(\mu\nabla u_h, \nabla w_h)_\kappa - \langle \mu(u_h^+ - \hat{u}_h)n^+, \nabla w_h^+ \rangle_{\partial\kappa} + \langle \hat{q}_h, w_h^+ n^+ \rangle_{\partial\kappa} = (f, w_h)_\kappa \qquad \forall w_h \in \mathcal{P}^p(\kappa). \tag{B.10}$$

Summing over all elements gives the complete DG discretization: Find $u_h \in W_h^p$ such that

$$a(u_h, w_h) = (f, w_h)_\Omega \qquad \forall w_h \in W_h^p. \tag{B.11}$$

The choice of the numerical trace $\hat{u}_h$ and flux $\hat{q}_h$ define the particular DG method considered. Table B.1 lists the numerical traces and fluxes for the DG methods considered. In the definition of the different DG methods, the following average and jump operators are used to define the numerical trace and flux on $e \in \mathcal{E}^i$:

$$\{u_h\} = \frac{1}{2}(u_h^+ + u_h^-) \qquad \text{and} \qquad [\![u_h]\!] = u_h^+ n^+ + u_h^- n^-. \tag{B.12}$$

Additionally, define a second set of jump operators involving the numerical trace $\hat{u}$:

$$[\![u_h]\!]^+ = u_h^+ n^+ + \hat{u}_h n^- \qquad \text{and} \qquad [\![u_h]\!]^- = \hat{u}_h n^+ + u_h^- n^-, \tag{B.13}$$

such that q_h may be expressed as:

$$q_h = -\mu(\nabla u_h - r_\kappa([\![u_h]\!]^+)). \tag{B.14}$$

Method	$\hat{u}_h$	$-\mu^{-1}\hat{q}_h$
Interior Penalty	$\{u_h\}$	$\{\nabla u_h\} - \frac{\eta_e}{2h}[\![u_h]\!]$
Bassi and Rebay [13]	$\{u_h\}$	$\{\nabla u_h\} - \eta_e \left\{r_e([\![u_h]\!]^\pm)\right\}$
Brezzi et al. [23]	$\{u_h\}$	$\{q_h\} - \eta_e \left\{r_e([\![u_h]\!]^\pm)\right\}$
LDG [38]	$\{u_h\} - \beta \cdot [\![u_h]\!]$	$\{q_h\} + \beta\,[\![q_h]\!] - \frac{\eta_e}{h}[\![u_h]\!]$
CDG [101]	$\{u_h\} - \beta \cdot [\![u_h]\!]$	$\{q_h^e\} + \beta\,[\![q_h^e]\!] - \frac{\eta_e}{h}[\![u_h]\!]$

Table B.1: Numerical fluxes for different DG methods

Note that in the definition of the different DG methods, η_e is a penalty parameter defined on each edge in $\mathcal{E}$, while $r_e(\phi) \in (\mathcal{P}^p(\kappa))^d$ is a local lifting operator defined by:

$$(r_e(\phi), v_h)_\kappa = \left\langle \phi, v_h^+ \right\rangle_e \qquad \forall v_h \in (\mathcal{P}^p(\kappa))^d. \tag{B.15}$$

Additionally q^e is given by:

$$q_h^e = -\mu(\nabla u_h - r_e([\![u]\!]^+)). \tag{B.16}$$

For the Local Discontinuous Galerkin (LDG) and Compact Discontinuous Galerkin (CDG) methods, β is a vector which is defined on each edge in $\mathcal{E}^i$ as

$$\beta = \frac{1}{2}\left(S_{\kappa^+}^{\kappa^-} n^+ + S_{\kappa^-}^{\kappa^+} n^-\right), \tag{B.17}$$

where $S_{\kappa^+}^{\kappa^-} \in \{0, 1\}$ is a switch defined on each face of element κ^+ shared with element κ^-, such that

$$S_{\kappa^+}^{\kappa^-} + S_{\kappa^-}^{\kappa^+} = 1. \tag{B.18}$$

B.2 The DG discretization from a new perspective

A key component, required for the development and analysis of the BDDC algorithm presented, is to express the global bilinear form $a(u_h, w_h)$ as the sum of element-wise contributions $a_\kappa(u_h, w_h)$ such that

$$a(u_h, w_h) \quad = \quad \sum_{\kappa \in \mathcal{T}} a_\kappa(u_h, w_h), \tag{B.19}$$

where $a_\kappa(u_h, w_h)$ is a symmetric, positive semi-definite "local bilinear form". In particular, the local bilinear form must have a compact stencil, such that $a_\kappa(u_h, w_h)$ is a function of only u_h, ∇u_h in κ, and u_h^+, ∇u_h^+ and $\hat{u}_h$ on $\partial\kappa$. In particular, note that in (B.10), which is summed over all elements to give $a(u_h, w_h)$, $\hat{q}$ depends in general upon u^+, u^-, ∇u^+ and ∇u^-. The local bilinear form is written as:

$$\begin{aligned} a_\kappa(u_h, w_h) \quad &= \quad (\mu \nabla u_h, \nabla w_h)_\kappa - \langle \mu(u_h^+ - \hat{u}_h) n^+, \nabla w_h^+ \rangle_{\partial\kappa} + \langle \hat{q}_h^+, (w_h^+ - \hat{w}_h) n^+ \rangle_{\partial\kappa} \\ &= \quad (\mu \nabla u_h, \nabla w_h)_\kappa - \langle \mu [\![u]\!]_h^+, \nabla w_h^+ \rangle_{\partial\kappa} + \langle \hat{q}_h^+, [\![w_h]\!]^+ \rangle_{\partial\kappa}, \end{aligned} \tag{B.20}$$

where $\hat{q}_h^+ = \hat{q}_h^+(\nabla u_h^+, u_h^+, \hat{u}_h)$ is a "local numerical flux". In particular, in order to recover the original global bilinear form, $\hat{q}_h^\pm$ must satisfy the following relationship on each edge, e:

$$\hat{q}_h [\![w_h]\!] \quad = \quad \hat{q}_h^+ [\![w_h]\!]^+ + \hat{q}_h^- [\![w_h]\!]^- \qquad \forall w_h \in W_h^p. \tag{B.21}$$

Table B.2 lists the numerical traces and local fluxes for the DG methods considered, while Table B.3 lists the corresponding local bilinear forms. It is simple to verify that (B.21) holds for each of the DG methods considered by using the identities:

$$[\![u_h]\!] = [\![u_h]\!]^+ + [\![u_h]\!]^-,$$

$$\begin{cases} [\![u_h]\!]^+ = [\![u_h]\!]^- = \frac{1}{2} [\![u_h]\!] & \text{if } \hat{u}_h = \{u_h\}, \\ [\![u_h]\!]^+ = [\![u_h]\!], \quad [\![u_h]\!]^- = 0 & \text{if } \hat{u}_h = \{u_h\} - \beta [\![u_h]\!] \text{ and } S_{\kappa^+}^{\kappa^-} = 1, \\ [\![u_h]\!]^+ = 0, \qquad [\![u_h]\!]^- = [\![u_h]\!] & \text{if } \hat{u}_h = \{u_h\} - \beta [\![u_h]\!] \text{ and } S_{\kappa^+}^{\kappa^-} = 0. \end{cases}$$

Consider using a nodal basis on each element κ to define W_h^p. Figure B-1 shows the degrees of freedom involve in the local bilinear form, $a_\kappa(u_h, w_h)$. For the Interior Penalty

Method	$\hat{u}_h$	$-\mu^{-1}\hat{q}_h^+$
Interior Penalty	$\{u_h\}$	$\nabla u_h^+ - \frac{\eta_e}{h} [\![u_h]\!]^+$
Bassi and Rebay [13]	$\{u_h\}$	$\nabla u_h^+ - \eta_e r_e([\![u_h]\!]^+)$
Brezzi et al. [23]	$\{u_h\}$	$q_h^+ - \eta_e r_e([\![u_h]\!]^+)$
LDG [38]	$\{u_h\} - \beta \cdot [\![u_h]\!]$	$q_h^+ - \frac{\eta_e}{h} [\![u_h]\!]^+$
CDG [101]	$\{u_h\} - \beta \cdot [\![u_h]\!]$	$q_h^{e+} - \frac{\eta_e}{h} [\![u_h]\!]^+$

Table B.2: Numerical fluxes for different DG methods

Method	$a_\kappa(u_h, w_h)$
Interior Penalty	$g + \sum_{e\in\partial\kappa} \frac{\eta_e}{h_e} \langle \mu [\![u_h]\!]^+, [\![w_h]\!]^+ \rangle_e$
Bassi and Rebay [13]	$g + \sum_{e\in\partial\kappa} \eta_e \left(\mu r_e([\![u_h]\!]^+), r_e([\![w_h]\!]^+)\right)_\kappa$
Brezzi et al. [23]	$g + \left(\mu r_\kappa([\![u_h]\!]^+), r_\kappa([\![w_h]\!]^+)\right)_\kappa + \sum_{e\in\partial\kappa} \eta_e \left(\mu r_e([\![u_h]\!]^+), r_e([\![w_h]\!]^+)\right)_\kappa$
LDG [38]	$g + \left(\mu r_\kappa([\![u_h]\!]^+), r_\kappa([\![w_h]\!]^+)\right)_\kappa + \sum_{e\in\partial\kappa} \frac{\eta_e}{h_e} \langle \mu [\![u_h]\!]^+, [\![w_h]\!]^+ \rangle_e$
CDG [101]	$g + \sum_{e\in\partial\kappa} \left(\mu r_e([\![u_h]\!]^+), r_e([\![w_h]\!]^+)\right)_\kappa + \sum_{e\in\partial\kappa} \frac{\eta_e}{h_e} \langle \mu [\![u_h]\!]^+, [\![w_h]\!]^+ \rangle_e$

Where $g = (\mu \nabla u_h, \nabla w_h)_\kappa - \langle \mu [\![u_h]\!]^+, \nabla w_h^+ \rangle_{\partial\kappa} - \langle \mu \nabla u_h, [\![w_h]\!]^+ \rangle_{\partial\kappa}$

Table B.3: Elementwise bilinear form for different DG methods

(IP) method and the methods of Bassi and Rebay, and Brezzi et al., the numerical trace $\hat{u}_h$ on an edge depends on both u_h^+ and u_h^-. Hence the local bilinear form corresponds to all nodal degrees of freedom defining u_h on κ as well as nodal values on all edges of $\partial\kappa \cap \mathcal{E}^i$ corresponding to the trace of u_h from elements neighbouring κ. On the other hand, for the LDG and CDG methods, the numerical trace $\hat{u}_h$ takes on the value of u_h^+ if $S_{\kappa^+}^{\kappa^-} = 0$ or u_h^- if $S_{\kappa^+}^{\kappa^-} = 1$. Hence the local bilinear form corresponds only to degrees of freedom defining u_h on κ and nodal values corresponding to the trace of u_h on neighbouring elements across edges of $\partial\kappa \cap \mathcal{E}^i$ for which $S_{\kappa^+}^{\kappa^-} = 1$.

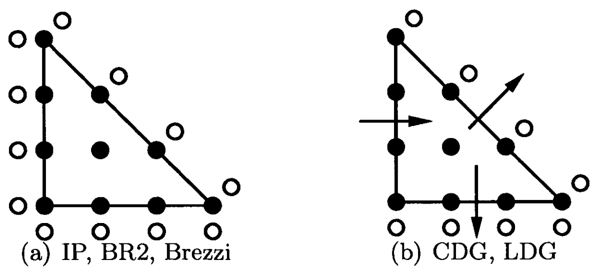

(a) IP, BR2, Brezzi (b) CDG, LDG

Figure B-1: Degrees of freedom involved in "local" bilinear form. •: Element Node, ∘: Neighbor Node, $\rightarrow$: Switch (β)

The following Lemmas characterize the local bilinear form:

Lemma B.2.1. *The element-wise bilinear form $a_\kappa(u_h, u_h)$ satisfies*

$$a_\kappa(u_h, u_h) \ \geq \ 0, \tag{B.22}$$

with $a_\kappa(u_h, u_h) = 0$ iff $u_h = \hat{u}_h = K$ for some constant K.

Proof. The following proves Lemma B.2.1 for all of the DG methods considered. The proof closely follows the proof of boundedness and stability of the different DG methods presented in Arnold et al. [8], though here only the contribution of a single element is considered.

To show $u_h = \hat{u}_h = K \Rightarrow a_\kappa(u_h, u_h) = 0$, note that $u_h = K \Rightarrow \nabla u_h = 0$. Substituting in to the bilinear form for the different DG methods considered gives the desired result. It remains to prove $a_\kappa(u_h, u_h) \geq 0$ and $a_\kappa(u_h, u_h) = 0 \Rightarrow u_h = \hat{u}_h = K$.

In order to prove the result for the interior penalty method consider the following result from Arnold et al [8]:

$$c\left(\mu r_e(w), r_e(w)\right)_\kappa \leq \frac{1}{h_e}\left\langle \mu w, w\right\rangle_e \leq C\left(\mu r_e(w), r_e(w)\right)_\kappa \qquad \forall w \in W_h^p, \tag{B.23}$$

where c and C are constants which depend only upon the minimum angle of κ and the polynomial order p. Choosing η_e sufficiently large for the interior penalty method

$$a_{\kappa,\mathrm{IP}}(u_h, u_h) \geq a_{\kappa,\mathrm{BR2}}(u_h, u_h), \tag{B.24}$$

and hence it is sufficient to show that Lemma B.2.1 holds for the method of Bassi and Rebay [13]. Specifically, η_e may be chosen for the interior penalty method as described in Shahbazi

[111]. For the method of Bassi and Rebay,

$$\begin{aligned}
a_{\kappa,\mathrm{BR2}}(u_h,u_h) &= (\mu\nabla u_h,\nabla u_h)_\kappa - 2\left\langle \mu\nabla u_h, [\![u_h]\!]^+\right\rangle_{\partial\kappa} + \sum_{e\in\partial\kappa}\eta_e\left(\mu r_e([\![u_h]\!]^+), r_e([\![u_h]\!]^+)\right)_\kappa \\
&= (\mu\nabla u_h,\nabla u_h)_\kappa - \sum_{e\in\partial\kappa} 2\left(\mu\nabla u_h, r_e([\![u_h]\!]^+)\right)_\kappa + \sum_{e\in\partial\kappa}\eta_e\left(\mu r_e([\![u_h]\!]^+), r_e([\![u_h]\!]^+)\right)_\kappa \\
&\geq \sum_{e\in\partial\kappa}\frac{1}{N_e}\left(\mu(\nabla u_h - r_e([\![u_h]\!]^+)), \nabla u_h - r_e([\![u_h]\!]^+)\right)_\kappa \\
&\qquad + \sum_{e\in\partial\kappa}(\eta_e - N_e)\left(\mu r_e([\![u_h]\!]^+), r_e([\![u_h]\!]^+)\right)_\kappa \\
&\geq 0, \qquad\qquad (\mathrm{B.25})
\end{aligned}$$

given $\eta_e > N_e$, where N_e is the number of edges of κ. In order to show $a_{\kappa,\mathrm{BR2}}(u_h,u_h)=0 \Rightarrow u_h = \hat{u}_h = K$, note that $a_{\kappa,\mathrm{BR2}}(u_h,u_h)=0$ implies

$$\sum_{e\in\partial\kappa}\frac{1}{N_e}\left(\mu(\nabla u_h - r_e([\![u_h]\!]^+)), \nabla u_h - r_e([\![u_h]\!]^+)\right)_\kappa + \sum_{e\in\partial\kappa}(\eta_e - N_e)\left(\mu r_e([\![u_h]\!]^+), r_e([\![u_h]\!]^+)\right)_\kappa = 0. \qquad (\mathrm{B.26})$$

Hence $r_e([\![u_h]\!]^+)=0$ and $\nabla u_h - r_e([\![u_h]\!]^+)=0$ which implies $\hat{u}_h = u_h^+$ on $\partial\kappa$ and $\nabla u_h = 0$ in κ.

Proof of the method of Brezzi et al. [23] follows in a similar manner. Namely:

$$\begin{aligned}
a_{\kappa,\mathrm{Br}}(\mu u_h,u_h) &= (\mu\nabla u_h,\nabla u_h)_\kappa - 2\left\langle \mu\nabla u_h, [\![u_h]\!]^+\right\rangle_{\partial\kappa} + \left(\mu r_\kappa([\![u_h]\!]^+), r_\kappa([\![u_h]\!]^+)\right)_\kappa \\
&\qquad + \sum_{e\in\partial\kappa}\eta_e\left(\mu r_e([\![u_h]\!]^+), r_e([\![u_h]\!]^+)\right)_\kappa \\
&\geq \left(\mu(\nabla u_h - r_\kappa([\![u_h]\!]^+)), \nabla u_h - r_\kappa([\![u_h]\!]^+)\right)_\kappa + \sum_{e\in\partial\kappa}\eta_e\left(\mu r_e([\![u_h]\!]^+), r_e([\![u_h]\!]^+)\right)_\kappa \\
&\geq 0, \qquad\qquad (\mathrm{B.27})
\end{aligned}$$

provided $\eta_e > 0$. In order to show $a_{\kappa,\mathrm{Br.}}(u_h,u_h)=0 \Rightarrow u_h = \hat{u}_h = K$, note that $a_{\kappa,\mathrm{Br.}}(u_h,u_h)=0$ implies

$$\left(\mu(\nabla u_h - r_\kappa([\![u_h]\!]^+)), \nabla u_h - r_\kappa([\![u_h]\!]^+)\right)_\kappa + \sum_{e\in\partial\kappa}\eta_e\left(\mu r_e([\![u_h]\!]^+), r_e([\![u_h]\!]^+)\right)_\kappa = 0 \qquad (\mathrm{B.28})$$

Hence $r_e([\![u_h]\!]^+)=0$ and $\nabla u_h - r_\kappa([\![u_h]\!]^+)=0$ which implies $\hat{u}_h = u_h^+$ on $\partial\kappa$ and $\nabla u_h = 0$

in κ.

For the LDG method

$$\begin{aligned}
a_{\kappa,\mathrm{LDG}}(u_h,u_h) &= (\mu\nabla u_h,\nabla u_h)_\kappa - 2\left\langle \mu\nabla u_h, [\![u_h]\!]^+\right\rangle_{\partial\kappa} + \left(\mu r_\kappa([\![u_h]\!]^+), r_\kappa([\![u_h]\!]^+)\right)_\kappa \\
&\quad + \sum_{e\in\partial\kappa}\frac{\eta_e}{h_e}\left\langle \mu\,[\![u_h]\!]^+, [\![u_h]\!]^+\right\rangle_e \\
&= \left(\mu(\nabla u_h - r_\kappa([\![u_h]\!]^+)), \nabla u_h - r_\kappa([\![u_h]\!]^+)\right)_\kappa + \sum_{e\in\partial\kappa}\frac{\eta_e}{h_e}\left\langle \mu\,[\![u_h]\!]^+, [\![u_h]\!]^+\right\rangle_e \\
&\geq 0. \qquad\qquad (\mathrm{B.29})
\end{aligned}$$

Setting $\eta_e > 0$ ensures $a_{\kappa,\mathrm{LDG}}(u_h,u_h) = 0 \Rightarrow u_h = \hat{u}_h = K$. Namely, $a_{\kappa,\mathrm{LDG}}(u_h,u_h) = 0$ implies

$$\left(\mu(\nabla u_h - r_\kappa([\![u_h]\!]^+)), \nabla u_h - r_\kappa([\![u_h]\!]^+)\right)_\kappa + \sum_{e\in\partial\kappa}\frac{\eta_e}{h_e}\left\langle \mu\,[\![u_h]\!]^+, [\![u_h]\!]^+\right\rangle_e = 0. \qquad (\mathrm{B.30})$$

Hence $[\![u_h]\!]^+ = 0$ and $\nabla u_h + r_\kappa([\![u_h]\!]^+) = 0$, which implies $\nabla u_h = 0$.

Finally for the CDG method, using (B.23) and noting that if η_e is chosen sufficiently large for the CDG method then

$$a_{\kappa,\mathrm{CDG}}(u_h,u_h) \geq a_{\kappa,\mathrm{BR2}}(u_h,u_h) \qquad (\mathrm{B.31})$$

Hence, proof of Lemma B.2.1 for the CDG method follows directly from the proof for the method of Bassi and Rebay. □

Assume that in each element κ, μ has a constant value $\mu = \mu_\kappa$. The following lemmas show that the bilinear form is equivalent to a quadratic form based on the value of u_h at the nodes x.

Lemma B.2.2. *There exist constants c and C independent of h and μ_κ such that for all $u_h \in W_h^p$*

$$c a_\kappa(u_h,u_h) \leq \mu_\kappa p^4 h^{n-2}\textstyle\sum_{x_i,x_j\in\kappa\cup\kappa'}(u_h(x_i)-u_h(x_j))^2 \leq C a_\kappa(u_h,u_h), \qquad (\mathrm{B.32})$$

where x_i, x_j are the nodes on κ defining the basis for u_h and nodes on $\partial\kappa'$ defining a basis for the trace u_h^- from neighbours κ' of κ. (Note that for the LDG and CDG methods, nodes

x_i, x_j include nodes defining a basis for u_h^- only on faces for which $S_{\kappa^+}^{\kappa^-} = 1$.)

Proof. Lemma B.2.2 is a direct consequence of Lemma B.2.1 and a scaling argument. See [43] Lemma 4.3 for the equivalent proof for a mixed finite element discretization. □

Lemma B.2.3. *For any region $\omega \subset \Omega$ composed of elements in $\mathcal{T}$, there exist constants c and C independent of h, $|\omega|$ and μ_κ such that for all $u_h \in W_h^p$*

$$ca_\omega(u_h, u_h) \quad \leq \textstyle\sum_{\kappa \in \omega} \mu_\kappa h^{n-2} \sum_{x_i, x_j \in \kappa \cup \kappa'} (u_h(x_i) - u_h(x_j))^2 \leq \quad Ca_\omega(u_h, u_h). \tag{B.33}$$

Proof. Lemma B.2.3 follows directly from Lemma B.2.2 and a summation over all element $\kappa \in \omega$. □

B.3 BDDC

The Schur complement problem and the BDDC algorithm for the DG discretization have the same structure as that for the HDG discretization. It remains to define the space of interior and interface degrees of freedom and the primal constraints.

Consider a partition of the domain Ω into subdomains Ω_i. Assume the following assumption holds for all subdomains.

Assumption B.3.1. *Each element κ in Ω_i with an edge e on $\partial\Omega_i \cap \partial\Omega_j$ has neighbours only in $\Omega_i \cup \Omega_j$.*

Note that while this assumption may appear limiting, in practice it is always possible to locallly split elements on corners/edges in 2D/3D respectively in order to satisfy this requirement.

Denote by $W_\Gamma^{(i)}$ the space of discrete nodal values on Γ_i which correspond to degrees of freedom shared between Ω_i and neighbouring subdomains Ω_j, while $W_I^{(i)}$ denotes the space of discrete unknowns local to a single substructure Ω_i. In particular, note that for the Interior penalty method, and the methods of Bassi and Rebay, and Brezzi et al. $W_\Gamma^{(i)}$ includes for each edge $e \in \Gamma_i$ degrees of freedom defining two sets of trace values u^+ from $\kappa^+ \in \Omega_i$ and u^- for $\kappa^- \in \Omega_j$. Thus, $W_I^{(i)}$ corresponds to nodal values strictly interior to Ω_i

or on $\partial\Omega_i \backslash \Gamma_i$. On the other hand, for the CDG and LDG methods $W_\Gamma^{(i)}$ includes for each edge $e \in \Gamma_i$ degrees of freedom defining a single trace value corresponding to either u^+ from $\kappa^+ \in \Omega_i$ if $S_{\kappa^+}^{\kappa^-} = 0$ or u^- from $\kappa^- \in \Omega_j$ if $S_{\kappa^+}^{\kappa^-} = 1$. Hence, $W_I^{(i)}$ corresponds to nodal values interior to Ω_i and on $\partial\Omega_i \backslash \Gamma_i$ as well as nodal values defining u^+ on $e \in \Gamma_i$ for which $S_{\kappa^+}^{\kappa^-} = 1$.

Similarly, define the spaces $\hat{W}_\Gamma$ and W_I which correspond to the space of discrete unknowns associated with coupled degrees of freedom on Γ and local degrees of freedom on substructures Ω_i respectively. Local operators $R_\Gamma^{(i)} : \hat{W}_\Gamma \to W_\Gamma^{(i)}$ extract the local degrees of freedom on Γ_i from those on Γ, while global operator $R_\Gamma : \hat{W}_\Gamma \to W_\Gamma$ is formed by a direct assembly of $R_\Gamma^{(i)}$.

The global system corresponding to the DG discretization may also be written in the form:

$$\begin{bmatrix} A_{II} & A_{\Gamma I}^T \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} u_I \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} b_I \\ b_\Gamma \end{bmatrix}, \tag{B.34}$$

where u_I and u_Γ corresponds to degrees of freedom associated with W_I and $\hat{W}_\Gamma$ respectively. The Schur complement problem is formed by eliminating degrees of freedom in W_I.

The BDDC preconditioner is specified by a set of constraints on the subdomain interfaces which define the primal and dual spaces, W_Π and W_Δ. For the DG discretization, the local primal spaces, $W_\Pi^{(i)}$, are spanned by basis functions $\{\psi_{\mathcal{E}_k}^{(i)}\}$ which defined by :

$$\sum_{e \in \mathcal{E}_j} \int_e \hat{\psi}_{\mathcal{E}_k}^{(i)} n_e n_{\mathcal{E}_j} = \delta_{jk}, \tag{B.35}$$

where $\hat{\psi}_{\mathcal{E}_k}^{(i)}$ denotes the numerical trace on $\mathcal{E}_k$ evaluated as a function of $\psi_{\mathcal{E}_k}^{(i)}$. The dual space, $W_\Delta^{(i)}$ corresponds to functions for which the integral in (B.35) vanishes on all subdomain interfaces.

As with HDG discretization, scaling operator $D_\Gamma^{(i)} : W_\Gamma^{(i)} \to W_\Gamma^{(i)}$ must be defined. In order to allow for large jumps in the coefficient μ across subdomains, the scaling operators are formed as the diagonal matrices with diagonal values set to the weighting function:

$$\delta_i^\dagger(\mathcal{E}_k) \;=\; \frac{\mu_i^\gamma}{\mu_i^\gamma + \mu_j^\gamma} \quad \mathcal{E}_k = \Omega_i \cap \Omega_j \qquad \gamma \in [1/2, \infty]\,. \tag{B.36}$$

Note that due to Assumption B.3.1 each nodal degree of freedom on Γ, may correspond to only two subdomains, Ω_i and Ω_j. Again $D_\Gamma : W_\Gamma \to W_\Gamma$ is defined as the diagonal matrix with diagonal blocks $D_\Gamma^{(i)}$.

Having specified all of the relevant finite element spaces and operators, the BDDC preconditioner for the DG discretization takes on the same form as that for the HDG discretization presented in Chapter 4. In the following section the same condition number bound is proven for the BDDC preconditioned DG system as for the HDG system.

B.4 Analysis

The analysis presented in this section is similar to that presented in Section 4.4 for the HDG discretization. Namely, the desired condition number bound is obtained by connecting the DG discretization to a related continuous finite element discretization. In particular, all of the results presented in this section are simply the DG equivalents of similar results presented in [119] or [43] for mixed finite elements. These are a direct result of the new perspective on the DG discretization presented in Section B.2.

In order to define the related continuous finite element discretization consider a special reparameterization of the space W_h^p on each subdomain Ω_i. Specifically, a nodal basis is employed on each element using a special set of nodal locations on each element κ. Specifically, on elements, κ, which do not touch $\partial\Omega_i$ nodal locations are chosen strictly interior to κ. On elements κ which touch $\partial\Omega_i$ nodal location are chosen on $\partial\kappa \cap \partial\Omega_i$ such that $\hat{u}|_{\partial\kappa\cap\partial\Omega_i}$ is uniquely defined by nodal values on $\partial\kappa$, while remaining nodal location are chosen interior to κ. This reparameterization is used so that each node defining the basis corresponds to a unique coordinate $\tilde{x}$, and $\hat{u}|_{\partial\Omega_i}$ is determined by nodal values on $\partial\Omega_i$. The following Lemma connects the two different parameterizations of the space W_h^p.

Lemma B.4.1. *There exist constants c and C independent of h such that for each element*

κ

$$c\sum_{x_i\in\kappa}\phi(x_i)^2 \leq \textstyle\sum_{\tilde{x}_i\in\kappa}\phi(\tilde{x}_i)^2 \leq C\sum_{x_i\in\kappa}\phi(x_j)^2 \qquad \forall\phi\in P^p(\kappa), \qquad \text{(B.37)}$$

and

$$c\sum_{x_i,x_j\in\kappa}(\phi(x_i)-\phi(x_j))^2 \leq \textstyle\sum_{\tilde{x}_i,\tilde{x}_j\in\kappa}(\phi(\tilde{x}_i)-\phi(\tilde{x}_j))^2 \leq C\sum_{x_i,x_j\in\kappa}(\phi(x_i)-\phi(x_j))^2 \qquad \forall\phi\in P^p(\kappa). \qquad \text{(B.38)}$$

Proof. Proof of Lemma B.4.1 follows directly from the fact that using either nodes x or $\tilde{x}$ we can form a Lagrange basis for $\phi \in P^p(\kappa)$, with basis function bounded as in [118] Lemma B.5. □

In order to define the subtriangulation $\hat{\mathcal{T}}$ of $\mathcal{T}$ consider each element $\kappa \in \mathcal{T}$. The subtriangulation on each element κ consists of the primary vertices used to define W_h^p, and secondary vertices corresponding to nodes on $\partial\kappa\backslash\partial\Omega_i$ required to form a quasi-uniform triangulation of κ. Note that such a subtriangulation may be obtained on the reference element $\hat{\kappa}$ then mapped to $\mathcal{T}$. As an example, Figure B-2 shows the nodes defining the reparameterization as well as the subtriangulation for a $p = 1$ triangular element.

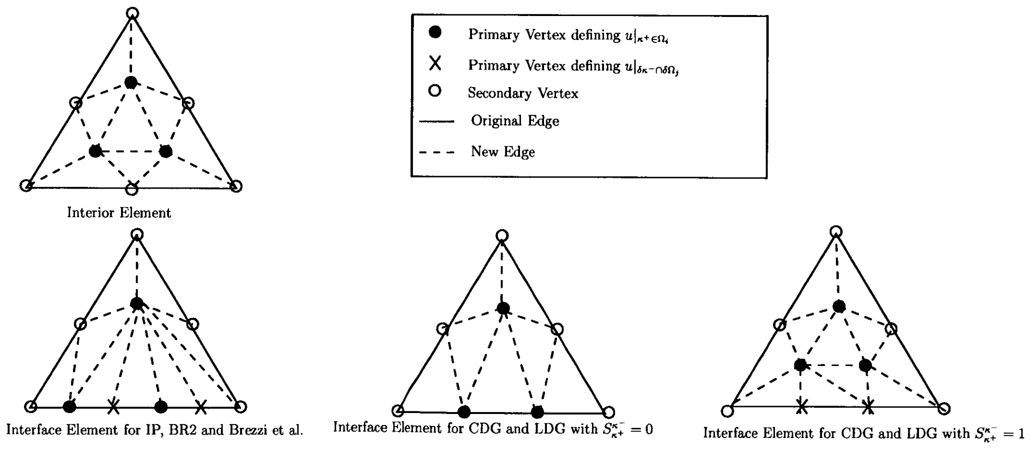

● Primary Vertex defining $u|_{\kappa^+\in\Omega_i}$
X Primary Vertex defining $u|_{\delta\kappa^-\cap\delta\Omega_j}$
O Secondary Vertex
Original Edge
New Edge
Interior Element
Interface Element for IP, BR2 and Brezzi et al.
Interface Element for CDG and LDG with $S_{\kappa^+}^{\kappa^-} = 0$
Interface Element for CDG and LDG with $S_{\kappa^+}^{\kappa^-} = 1$

Figure B-2: Examples of subtriangulations of $p = 1$ triangular elements

Define $U_h(\Omega)$ to be the continuous linear finite element space defined on the triangulation $\hat{\mathcal{T}}$. Additionally define $U_h(\Omega_i)$ and $U_h(\partial\Omega_i)$, as the restriction of $U_h(\Omega)$ to Ω_i and $\partial\Omega_i$ respectively. Now define a mapping $I_h^{\Omega_i}$ from any function ϕ defined at the primary vertices in Ω_i to $U_h(\Omega_i)$ as

$$I_h^{\Omega_i}\phi(x) = \begin{cases} \phi(x), \text{ if } x \text{ is a primary vertex;} \\ \\ \text{the average of all adjacent primary vertices on } \partial\Omega_i, \\ \text{if } x \text{ is a secondary vertex on } \partial\Omega_i; \\ \\ \text{the average of all adjacent primary vertices on } \Omega_i, \\ \text{if } x \text{ is a secondary vertex in the interior of } \Omega_i; \\ \\ \text{the linear interpolation of the vertex values,} \\ \text{if } x \text{ is not a vertex of } \mathcal{T}. \end{cases} \tag{B.39}$$

Since $(I_h^{\Omega_i}\phi)|_{\partial\Omega_i}$ is uniquely defined by $\phi|_{\partial\Omega_i}$, $I_h^{\partial\Omega_i}$ may be define as the map from a function defined on the primary vertices on $\partial\Omega_i$ to $U_h(\partial\Omega_i)$ such that $I_h^{\partial\Omega_i}\phi|_{\partial\Omega_i} = (I_h^{\Omega_i}\phi)|_{\partial\Omega_i}$. Define $\tilde{U}_h(\Omega_i) \subset U_h(\Omega_i)$ and $\tilde{U}_h(\partial\Omega_i) \subset U_h(\partial\Omega_i)$ as the range of $I_h^{\Omega_i}$ and $I_h^{\partial\Omega_i}$ respectively.

The original DG discretization is connected to the continuous finite element discretization on $\tilde{\mathcal{T}}$ by showing that both discretizations are equivalent to a quadratic form in terms of the nodal values on $\tilde{\mathcal{T}}$. The following lemmas and theorems are the equivalent of similar theorems for mixed finite element discretizations presented in [43] and [119]. These results are a direct consequence of Lemma B.2.1, which is the DG equivalent of Lemma 4.2 of [43].

Lemma B.4.2. *For Ω_i composed of elements κ in $\mathcal{T}$, there exist constants c and C independent of h, H and μ such that for all $u_h \in W_h^p$*

$$c a_i(u_h, u_h) \leq \sum_{\kappa\in\Omega_i} \mu_\kappa h^{n-2} \sum_{\tilde{x}_i,\tilde{x}_j\in\kappa\cup\kappa'} (u_h(\tilde{x}_i) - u_h(\tilde{x}_j))^2 \leq C a_i(u_h, u_h). \tag{B.40}$$

Proof. Lemma B.4.2 follows directly from Lemmas B.2.3 and B.4.1. □

Lemma B.4.3. *There exists a constant $C > 0$ independent of h and H such that*

$$\left| I_h^{\partial\Omega_i} \phi \right|_{H^1(\Omega_i)} \leq C \left| \phi \right|_{H^1(\Omega_i)} \qquad \forall \phi \in U_h(\Omega_i), \tag{B.41}$$

$$\| I_h^{\partial\Omega_i} \phi \|_{L^2(\Omega_i)} \leq C \| \phi \|_{L^2(\Omega_i)} \qquad \forall \phi \in U_h(\Omega_i). \tag{B.42}$$

Proof. See [43] Lemma 6.1. □

Lemma B.4.4. *There exist constants $c, C > 0$ independent of h and H such that for any $\hat{\phi} \in \tilde{U}_h(\partial\Omega_i)$*

$$c \| \hat{\phi} \|_{H^{1/2}(\partial\Omega_i)} \leq \inf_{\substack{\phi \in \tilde{U}_h(\Omega_i) \\ \phi|_{\partial\Omega_i} = \hat{\phi}}} \| \phi \|_{H^1(\Omega_i)} \leq C \| \hat{\phi} \|_{H^{1/2}(\partial\Omega_i)}, \tag{B.43}$$

$$c \left| \hat{\phi} \right|_{H^{1/2}(\partial\Omega_i)} \leq \inf_{\substack{\phi \in \tilde{U}_h(\Omega_i) \\ \phi|_{\partial\Omega_i} = \hat{\phi}}} \left| \phi \right|_{H^1(\Omega_i)} \leq C \left| \hat{\phi} \right|_{H^{1/2}(\partial\Omega_i)}. \tag{B.44}$$

Proof. See [43] Lemma 6.2. □

Lemma B.4.5. *There exists a constant $C > 0$ independent of h and H such that*

$$\| I_h^{\partial\Omega_i} \hat{\phi} \|_{H^{1/2}(\partial\Omega_i)} \leq C \| \hat{\phi} \|_{H^{1/2}(\partial\Omega_i)} \qquad \forall \hat{\phi} \in U_h(\partial\Omega_i). \tag{B.45}$$

Proof. See [43] Lemma 6.3. □

Lemma B.4.6. *There exist constants c and C independent of h, H and μ_i such that for all $u_\Gamma^{(i)} \in W_\Gamma^{(i)}$,*

$$c\mu_i \left| I_h^{\partial\Omega_i} u_i \right|^2_{H^{1/2}(\partial\Omega_i)} \leq \left| u_i \right|_{S_\Gamma^{(i)}} \leq C\mu_i \left| I_h^{\partial\Omega_i} u_i \right|^2_{H^{1/2}(\partial\Omega_i)}. \tag{B.46}$$

Proof. See [43] Theorem 6.5. □

Define the interface averaging operator $E_D : \tilde{W}_\Gamma \rightarrow \tilde{W}_\Gamma$ as:

$$E_D = R_\Gamma R_{D,\Gamma}^T. \tag{B.47}$$

Lemma B.4.7. *There exist constants c and C independent of h and H such that for all $u_\Gamma \in \tilde{W}_\Gamma$*

$$|E_D u_\Gamma|^2_{\tilde{S}_\Gamma} \leq C(1 + \log(p^2 H/h))^2 \, |u_\Gamma|^2_{\tilde{S}_\Gamma}. \tag{B.48}$$

Proof. The proof of Lemma B.4.7 closely follows that of [119] Lemma 5.5. Note Assumption B.3.1 is essential for this result. In particular, if Assumption B.3.1 were not valid then $(E_D u_\Gamma)_j$ the restriction of $E_D u_\Gamma$ to degrees of freedom on Ω_j would necessarily depend on degrees of freedom u_k corresponding to a subdomain Ω_k which does not neighbour Ω_j however are connected through the element κ in Ω_i which has edges on both $\partial\Omega_i \cap \partial\Omega_j$ and $\partial\Omega_i \cap \partial\Omega_k$. □

Theorem B.4.8. *The condition number of the preconditioner operator $M^{-1}_{BDDC}\hat{S}$ is bounded by $C(1 + \log(p^2 H/h))^2$ where C is a constant independent of p, h, H or μ.*

Proof. Theorem B.4.8 follows directly from Lemma B.4.7. (See for example [119] Theorem 6.1). □

B.5 Numerical Results

This section presents numerical results for the BDDC preconditioner introduced in Section B.3. For each numerical experiment the linear system resulting from the DG discretization is solved using a Preconditioned Conjugate Gradient (PCG) method. The PCG algorithm is run until the initial l_2 norm of the residual is decreased by a factor of 10^{10}. Consider a domain $\Omega \in \mathbb{R}^2$ with $\Omega = (0,1) \times (0,1)$. Ω is partitioned into $N \times N$ square subdomains Ω_i with side lengths H such that $N = \frac{1}{H}$. Each subdomain is the union of triangular elements obtained by bisecting squares of side length h, ensuring that Assumption B.3.1 is satisfied. Thus each subdomain has n_i elements, where $n_i = 2\left(\frac{H}{h}\right)^2$.

(a) $p = 1$

$\frac{H}{h}$ \ $\frac{1}{H}$	2	4	8	16	32
2	4	16	24	29	29
4	13	20	30	31	31
8	15	21	32	34	34
16	15	24	34	35	35

(b) $p = 3$

$\frac{H}{h}$ \ $\frac{1}{H}$	2	4	8	16	32
2	8	17	25	30	30
4	13	21	29	31	31
8	13	22	31	33	32
16	12	23	33	34	33

(c) $p = 5$

$\frac{H}{h}$ \ $\frac{1}{H}$	2	4	8	16	32
2	8	20	25	31	31
4	12	21	30	33	31
8	11	23	32	35	33
16	11	26	34	36	35

Table B.4: Iteration count for BDDC preconditioner using Interior Penalty Method

In the first set of numerical experiments (B.1) is solved on Ω where f is chosen such that the exact solution is given by $u = \sin(\pi x)\sin(\pi y)$. Tables B.4- B.8 show the corresponding number of PCG iteration required to converge for polynomial orders $p =$ 1, 3, and 5, using each of the DG methods considered. As predicted by the analysis the number of iterations is independent of the number of subdomains and only weakly dependent upon the number of elements per subdomain. In addition the number of iterations is only weakly dependent on the solution order p. Finally, note that the number of iterations required for the solution of the LDG and CDG discretizations is somewhat smaller than those of the other DG methods. For the LDG and CDG methods the Schur complement problem has approximately half the number of degrees of freedom as for the other DG methods, hence it is not surprising that a smaller number of iterations is required to converge.

In the second numerical experiment the behaviour of the preconditioner is examined for large jumps in the coefficient μ. The domain is partitioned in a checkerboard pattern where $\mu = 1$ on half of the subdomains and $\mu = 1000$ in the remaining subdomains. Equation (B.1) is solved with forcing function $f = 1$ using the CDG method. Initially $\delta_i^\dagger$ is set to $\delta_i^\dagger = \frac{1}{2}$, which corresponds to setting $\gamma = 0$ in (B.36). Since the choice of $\delta_i^\dagger$ does not satisfy the assumption $\gamma \in [1/2, \infty)$ poor convergence of the BDDC algorithm is seen as shown in Table B.8(a). Next $\delta_i^\dagger$ is set as in (B.36) with $\gamma = 1$. With this choice of $\delta_i^\dagger$ the good convergence

(a) $p = 1$

			$\frac{1}{H}$		
$\frac{H}{h}$	2	4	8	16	32
2	4	19	28	33	33
4	14	24	34	36	36
8	18	26	36	38	38
16	18	28	37	40	40

(b) $p = 3$

			$\frac{1}{H}$		
$\frac{H}{h}$	2	4	8	16	32
2	8	21	29	34	33
4	16	25	33	35	35
8	16	27	34	36	36
16	15	28	36	37	37

(c) $p = 5$

			$\frac{1}{H}$		
$\frac{H}{h}$	2	4	8	16	32
2	9	23	30	35	34
4	16	26	34	36	36
8	15	28	35	36	36
16	14	29	37	38	38

Table B.5: Iteration count for BDDC preconditioner using the method of Bassi and Rebay

(a) $p = 1$

			$\frac{1}{H}$		
$\frac{H}{h}$	2	4	8	16	32
2	4	16	23	25	24
4	13	19	27	28	28
8	15	20	28	30	29
16	16	22	30	32	32

(b) $p = 3$

			$\frac{1}{H}$		
$\frac{H}{h}$	2	4	8	16	32
2	7	17	25	26	26
4	14	20	28	29	28
8	15	22	29	31	30
16	14	24	31	33	33

(c) $p = 5$

			$\frac{1}{H}$		
$\frac{H}{h}$	2	4	8	16	32
2	8	18	26	27	27
4	14	22	28	29	29
8	15	23	30	32	32
16	14	25	33	34	33

Table B.6: Iteration count for BDDC preconditioner using the method of Brezzi et al.

(a) $p = 1$

			$\frac{1}{H}$		
$\frac{H}{h}$	2	4	8	16	32
2	12	18	20	20	20
4	13	20	23	23	23
8	14	23	26	26	26
16	14	25	28	29	28

(b) $p = 3$

			$\frac{1}{H}$		
$\frac{H}{h}$	2	4	8	16	32
2	11	20	22	22	22
4	12	22	25	25	25
8	12	24	28	28	27
16	12	25	29	30	30

(c) $p = 5$

			$\frac{1}{H}$		
$\frac{H}{h}$	2	4	8	16	32
2	12	21	24	24	23
4	12	23	27	28	27
8	11	25	29	30	30
16	11	26	31	32	31

Table B.7: Iteration count for BDDC preconditioner using the LDG method

(a) $p = 1$

			$\frac{1}{H}$		
$\frac{H}{h}$	2	4	8	16	32
2	12	19	20	20	19
4	12	20	23	23	22
8	13	23	25	25	25
16	13	24	28	28	27

(b) $p = 3$

			$\frac{1}{H}$		
$\frac{H}{h}$	2	4	8	16	32
2	11	20	22	22	22
4	12	21	24	25	24
8	12	23	27	27	27
16	12	25	28	29	29

(c) $p = 5$

			$\frac{1}{H}$		
$\frac{H}{h}$	2	4	8	16	32
2	11	22	25	24	24
4	12	24	27	27	26
8	12	24	29	29	29
16	11	26	31	31	31

Table B.8: Iteration count for BDDC preconditioner using the CDG method

properties of the BDDC algorithm are recovered as shown in Table B.8(b).

(a) $\delta_i^\dagger = \frac{1}{2}$

	$\frac{1}{H}$				
p	2	4	8	16	32
0	17	69	118	138	147
1	51	119	179	215	232
2	52	129	192	241	252
3	55	133	207	267	316
4	58	144	226	285	304
5	59	153	242	306	361

(b) $\delta_i^\dagger = \frac{\mu_i}{\sum_j \mu_j}$

	$\frac{1}{H}$				
p	2	4	8	16	32
0	4	6	13	15	16
1	4	7	14	18	19
2	4	7	13	17	18
3	4	7	15	18	19
4	4	7	14	19	20
5	4	7	14	19	20

Table B.9: Iteration count for BDDC preconditioner using the CDG method with $\mu = 1$ or 1000