Chapter 1

Massively Parallel Solution Techniques for Higher-order Finite-element Discretizations in CFD

Laslo T. Diosady and David L. Darmofal

Massachusetts Institute of Technology (MIT),
Aerospace Computational Design Laboratory,
77 Massachusetts Ave. 33-207, Cambridge MA, 02139,
diosady@mit.edu darmofal@mit.edu

The purpose of this paper is to present techniques to solve higher-order finite element discretizations on massively parallel architectures. Implicit schemes are considered as a means of achieving mesh independent convergence rates for both time dependent problems and steady state solutions obtained through pseudo-transient continuation. Domain decomposition preconditioners are presented for the scalable parallel solution of the linear system arising at each iteration of a Newton-Krylov approach. Basic domain decomposition methods are presented along with theoretical results for simple model problems. Practical extensions of these algorithms for simulations of the Euler and Navier-Stokes equations are reviewed in reference to the theoretical results from the model problems. Extensions of some recently developed iterative substructuring algorithms are also proposed for the Euler and Navier-Stokes equations. Numerical examples using several domain decomposition algorithms are presented for a higher-order simulation of a convection-diffusion model problem.

1. Introduction

Today's most powerful supercomputers are able to reach a peak performance of more than one petaflop/s. 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, Mavriplis notes: "The scalability of most [CFD] codes tops out around 512 cpus".[1] Developing CFD codes which are able to scale effi-

ciently to tens or hundreds of thousands of processors remains a significant challenge.

A key use of massively parallel computers is to perform large-scale simulations in similar amount of time as typical industrial simulations on commodity hardware, through the use of parallelization. Thus, "optimal" algorithms are desired, for which the work scales linearly with the number of degrees of freedom. For iterative methods, for which the work associated with each iteration scales linearly with the number of degrees of freedom, optimality implies that the method converges at a rate independent of the size of the mesh. In the context of higher-order simulations, optimality also implies that the number of iterations is independent of the solution order. As the work associated with each iteration depends upon the number of degrees of freedom, the ability to perform large-scale simulations in reasonable time additionally requires that the work associated with each iteration may be performed in parallel across a large number of processors.

Two definitions of parallel scaling are common: "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,[3] 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, we focus primarily on the algorithmic aspects to ensure scalability. In the context of high-fidelity CFD simulations, we argue 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 we will use the term "scalable" 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.

For unsteady simulations, explicit methods have been touted as being highly parallelizable, as inter-processor communication is required only in updating ghosted data from neighbouring processors, while residual evaluations are trivially parallelized. While explicit methods are relatively simple to implement, the largest allowable time step is limited by the CFL con-

dition, 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,[4] where time-stepping enables reliable convergence for nonlinear problems.[5–13] 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[5,14–18] and preconditioned Newton-Krylov methods.[6,10–13,19] 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.[20] 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, to enable the efficient solution of large linear systems.

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 PDE problems. 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 problem. 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. In particular, the elliptic nature of the Poisson equation may be seen as 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 may be 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-speed 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 felt 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 in 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.

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 present especially 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.

As aerodynamic flows involve both elliptic and hyperbolic features, the most successful algorithms have combined effective solvers for elliptic and hyperbolic problems. For example multigrid methods have been used in

combination with tri-diagonal line solvers.[15,16] 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.[19]

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,[15,18] care must be taken in forming the nested coarse grid problems to ensure good performance.[15] The domain decomposition preconditioners presented in this paper 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. In particular, successful multigrid and ILU preconditioners discussed in the serial context may be used as local solvers on each subdomain.

The purpose of this paper is twofold: first to provide the reader with an understanding of the performance of several successful solution algorithms on simple model problems; and second to discuss the extension of these algorithms to the solution of higher-order discretizations of convection-dominated flows of interest in the CFD community. In particular, we focus on describing the algorithms and give theoretical and numerical results where relevant. However, we refrain from providing proofs of the theoretical results, which may be found in the references provided. In Section 2, we present Schwarz methods in the context of the model problems, then review large-scale CFD applications of these algorithms. In Section 3, we present Schur complement techniques, while in Section 4 we discuss Neumann-Neumann methods. Finally, in Section 5, we present some numerical results discussing the algorithms presented.

2. Schwarz Methods

In this section, we present Schwarz methods, which are often referred to as overlapping methods. Schwarz methods can be traced back to 1870, when Schwarz described an iterative method for solving an elliptic PDE problem by alternatingly solving the problem in subdomains of the orig-

inal domain using the solution from a previous iterate as the boundary condition. While this classical alternating Schwarz method was not used as numerical solution technique, it forms the basis for many successful domain decomposition algorithms. We present the basic ideas for the case of two subdomains then discuss the extension to the case of many subdomains. The presentation in this section closely follows that of Smith, Bjorstad and Gropp[21] and Toselli and Widlund[22] and we refer the reader to these books for a complete presentation.

2.1. *The case of two subdomains*

Consider the Poisson problem in a domain Ω:

$$-\Delta u = f \qquad \text{in } \Omega, \tag{1}$$

$$u = 0 \qquad \text{on } \partial\Omega. \tag{2}$$

We partition the domain Ω into two overlapping subdomains $\Omega_1^{'}$ and $\Omega_2^{'}$. Given an iterate u^n, the Schwarz alternating method solves for u^{n+1} by solving successive Dirichlet problems in $\Omega_1^{'}$ and $\Omega_2^{'}$:

$$\begin{cases} -\Delta u^{n+1/2} = f & \text{in } \Omega_1^{'}, \\ u^{n+1/2} = 0 & \text{on } \partial\Omega_1^{'} \cap \partial\Omega, \\ u^{n+1/2} = u^n & \text{on } \partial\Omega_1^{'} \backslash \partial\Omega, \\ u^{n+1/2} = u^n & \text{in } \Omega_2^{'} \backslash \Omega_1^{'}, \end{cases} \tag{3}$$

$$\begin{cases} -\Delta u^{n} = f & \text{in } \Omega_2^{'}, \\ u^{n+1} = 0 & \text{on } \partial\Omega_2^{'} \cap \partial\Omega, \\ u^{n+1} = u^{n+1/2} & \text{on } \partial\Omega_2^{'} \backslash \partial\Omega, \\ u^{n+1} = u^{n+1/2} & \text{in } \Omega_1^{'} \backslash \Omega_2^{'}, \end{cases} \tag{4}$$

Consider a finite element discretization of (1)-(2). Given an appropriate bilinear form and basis, the corresponding discrete system of equations may be written as:

$$Au = f \tag{5}$$

where $u \in \mathbb{R}^n$ denotes the vector of discrete unknowns. We denote by u_1 and u_2 degrees of freedom corresponding to $\Omega_1^{'}$ and $\Omega_2^{'}$, respectively. Additionally, we denote by R_1 and R_2 the $\{0,1\}$ matrices, respectively, that extract degrees of freedom u_1 and u_2 from u (i.e. $u_i = R_i u, i \in \{1,2\}$). Using this notation, the discrete Schwarz alternating method may

be written using the following steps:

$$u^{n+1/2} = u^n + R_1^T A_1^{-1} R_1 \left(f - Au^n\right), \tag{6}$$

$$u^{n+1} = u^{n+1/2} + R_2^T A_2^{-1} R_2 \left(f - Au^{n+1/2}\right). \tag{7}$$

Here $A_1 = R_1 A R_1^T$ and $A_2 = R_2 A R_2^T$ are simply the blocks extracted from A corresponding u_1 and u_2, respectively. Eliminating $u^{n+1/2}$ we see that the Schwarz alternating method is a Richardson iteration for the preconditioned system:

$$M^{-1}Au = M^{-1}f, \tag{8}$$

with the preconditioner given by:

$$M_{MSM}^{-1} = R_1^T A_1^{-1} R_1 + R_2^T A_2^{-1} R_2 \left(I - AR_1^T A_1^{-1} R_1\right). \tag{9}$$

This preconditioner is referred to as a multiplicative Schwarz method, thus we use the subscript $_{MSM}$. In the multiplicative Schwarz method the Dirichlet problem solved in Ω_2 depends upon the intermediate solution $u^{n+1/2}$ in Ω_1 and hence this algorithm is inherently sequential. As opposed to using the intermediate solution $u^{n+1/2}$ as the boundary condition in Ω_2, the previous iterate u^n may be used as boundary conditions for both Ω_1 and Ω_2, allowing the Dirichlet problems in Ω_1 and Ω_2 to be solved independently. This method, known as an additive Schwarz method, will in general not converge through a Richardson iteration, however may be used as an effective preconditioner for a Krylov method. We write the additive Schwarz preconditioner as:

$$M_{ASM}^{-1} = R_1^T A_1^{-1} R_1 + R_2^T A_2^{-1} R_2. \tag{10}$$

The adjectives additive and multiplicative refer to the propagation of the error, $u - u^n$, in the different Schwarz algorithms. Namely, the solution of the problem restricted to a subdomain may be viewed as a projection of the error to the finite element space orthogonal to the space defined by the degrees of freedom corresponding to that particular subspace. For additive methods, each subdomain problem is solved independently and thus the error is given by the sum of the projections corresponding to each subdomain. In multiplicative methods, the subdomain problems are solved sequentially, leading the error to be reduced as the product of two projections. In this paper we will present several preconditioners, involving both additive and multiplicative components, which are sometimes referred to as hybrid Schwarz methods. In general, we will use additive to refer to operations of these preconditioner which may be performed independently,

while we use multiplicative to refer to sequential operations. We note that the convergence rate of the multiplicative Schwarz method relative the additive Schwarz method, is much like the performance of Gauss-Seidel versus Jacobi. Namely, the convergence rate of multiplicative Schwarz methods improve upon additive Schwarz methods by a constant factor.

2.2. *The case of many subdomains*

Both additive and multiplicative Schwarz methods are easily extended to the case of many subdomains. Consider a partition of the domain Ω into N nonoverlapping subdomains Ω_i, $i = 1, ..., N$. An overlapping partition of the domain is defined by extending each subdomain Ω_i by an amount δ to a region $\Omega_i^{'} \subset \Omega$. In practice, $\Omega_i^{'}$ may be defined by adding layers of elements from neighbouring subdomains to Ω_i. The additive Schwarz method involves the solution of N independent Dirichlet problems corresponding to each subdomain, which may be performed in parallel, by assigning a subdomain to each processor. Using the notation previously defined, we write the additive Schwarz preconditioner as:

$$M_{ASM}^{-1} = \sum_{i=1}^{N} R_i^T A_i^{-1} R_i, \tag{11}$$

As described in the case of two subdomains, the multiplicative Schwarz method is inherently sequential. In the case of many subdomains, parallelism is introduced using a colouring argument. Namely, each subdomain $\Omega_i^{'}$ is assigned to a "colour" corresponding to groups of subdomains which do not overlap. Subdomain problems corresponding to the same colour may be solved independently of one another. Thus, in the case of many subdomains, the multiplicative Schwarz method involves only a small number of sequential steps corresponding to each colour, as opposed to N steps corresponding to each subdomain. In order to achieve good performance, each processor should be assigned several subdomains, one corresponding to each colour. We note that each sequential step of the multiplicative Schwarz method involves a multiplication of the system matrix A in order to update the residual. However, usually only parts of the residual vector need to be updated at each iteration which may often be performed locally.

The basic forms of the additive and multiplicative Schwarz methods lack a global correction. Thus, for elliptic problems, these methods are not scalable. A coarse space capable of controlling low frequency modes can be introduced by considering a discretization of the original PDE on a coarse

triangulation $\mathcal{T}_H$.[23] In general, the fine grid $\mathcal{T}_h$ does not need to be derived from a refinement of the coarse grid $\mathcal{T}_H$, only an interpolation operator from the fine space to the coarse space needs to be defined. We denote by R_0^T the interpolation operator from $\mathcal{T}_H$ to the finite element space defined on $\mathcal{T}_h$, where R_0 may be viewed as a restriction from the original finite element space to the coarse subspace. The coarse system matrix A_0 may be obtained either from discretizing the original PDE on $\mathcal{T}_H$ or through a restriction of the form $A_0 = R_0^T A R_0$. The additive Schwarz preconditioner with coarse grid correction is thus given by:

$$M_{ASM_0}^{-1} = R_0^T A_0^{-1} R_0 + \sum_{i=1}^{N} R_i^T A_i^{-1} R_i. \tag{12}$$

A simple variant of this preconditioner may be obtained by applying the coarse grid correction in a multiplicative manner.[21] 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. Similar variants of the multiplicative Schwarz method have also be developed.[24] The presence of the coarse space enables the additive Schwarz method to be scalable for elliptic problems. Namely, the condition number of the preconditioned system is given by $\kappa\left(M_{ASM_0}^{-1} A\right) \leq C\left(1 + \frac{H}{\delta}\right)$, where H is the diameter of a subdomain Ω_i, while δ is the amount of overlap and C is a constant independent of H or h.[23,24] 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, we may consider the case where the overlap is defined by extending each nonoverlapping subdomain by a small number of element of the fine triangulation. In this case we have $\delta \geq ch$, and the condition number bound has the form $\kappa \leq C_1\left(1 + \frac{H}{h}\right)$. Thus in the case of small overlap this type of preconditioner is scalable, but not optimal.

While originally presented for the solution of self-adjoint elliptic problems,[23] the analysis of Schwarz methods has been extended to linear convection-diffusion problems by Cai and collaborators.[24–28] For linear convection-diffusion problems, Schwarz methods with generous overlap and a coarse space have been shown to be both scalable and optimal, provided the diameter of the subdomains are sufficiently small.[24–26] Namely, if the Peclet number defined using the subdomain length scale, H, is sufficiently

small, then the behaviour of the Schwarz method 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 related to the number of subdomains through which a characteristic must cross before exiting a domain. Similar behaviour is observed for other domain decomposition methods for convection-dominated problems and this remains an open area of research.

In the case of unsteady convection-diffusion problems, solved using implicit time integration, analysis of additive and multiplicative Schwarz 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.[27,28] 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.

While initially analyzed for the solution of the systems of equations arising from linear continuous finite element discretizations, overlapping Schwarz methods have been extended to mixed finite element,[29] spectral element,[30] and discontinuous Galerkin discretizations.[31–35] Schwarz methods have also been applied to finite difference,[36] and finite volume discretizations.[11] For higher-order discretization, the overlapping regions may be defined by extending nonoverlapping domains by layers of nodes corresponding to the discrete unknowns.[30,37] However, for unstructured meshes, choosing an appropriate set of nodes may be non-trivial.[38] Thus, if only moderate polynomial orders are used, the overlapping regions are typically defined by adding layers of elements.

2.3. *Large scale CFD applications*

Overlapping additive Schwarz methods are the most widely used domain decomposition methods for CFD applications. Overlapping methods may be seen as particularly well suited to cell-centered finite-volume, or higher-order discontinuous Galerkin discretizations, where degrees of freedom are naturally associated with element interiors. Thus each elemental degree of freedom is "owned" by a single processor, while overlapping regions consist of elements owned by neighbouring processors. For these type of discretizations, we may also consider the special case of zero overlap, such that the

u_i's correspond to distinct degrees of freedom. In this case the additive Schwarz method reduces to a block Jacobi preconditioner with each block A_i corresponding to a single subdomain Ω_i. Similarly, the multiplicative Schwarz method reduces to a subdomain-wise block Gauss-Seidel preconditioner for A.

For node-based finite-volume, or continuous finite-difference discretizations, a nonoverlapping partitioning of the elements results in a "minimum-overlapping" partition of nodes. In a practical implementation, a nodal degree of freedom on the interface is assigned to a unique processor, which is updated by local solves corresponding to both sides of the interface. A variant, known as the restricted additive Schwarz method, updates only locally owned degrees of freedom, eliminating communication during the solution update.[39] Numerical results have shown that this method actually requires fewer iterations to converge than the basic additive Schwarz preconditioner for both scalar convection-diffusion,[39] and compressible Euler problems.[40]

The use of domain decomposition methods for large scale applications involves additional considerations in order to achieve good performance.[10] Large scale CFD applications may be both memory and CPU limited, making the exact solution of the local problems (corresponding to A_i^{-1}) using LU factorization intractable. Thus, the local solver may 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.[6] ILU factorizations have been particularly popular as local solvers for additive Schwarz methods with and without a coarse correction.[6,10,11,40–43] Cai, Farhat and Sarkis also employed a preconditioned GMRES method to solve the local problem on each subdomain.[41,42] 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.

The ability to achieve high performance for large scale simulations also requires an appropriate balance between local computation on each processor and relatively slow communication tasks.[10] As discussed previously, the case of generous overlap ensures that the preconditioner is optimal. However, if the overlap is generous, then the number of degrees of freedom in the overlap region of a subdomain is proportional to the volume of the subdomain. On the other hand, in the case of small overlap, where the overlap is defined by extending each subdomain by a few layers of elements, the

number of degrees of freedom corresponding to the overlap region is proportional to the surface area of the subdomain. Thus if each subdomain is assigned to a single processor, the ratio of computation to communication may be much higher for the case of small overlap and thus potentially better performance may be achieved. Subdomain-wise block-Jacobi preconditioners have been used for discontinuous Galerkin discretization of the compressible Euler and Navier-Stokes equations on up to 512 processors.[43] Gropp et al. showed that adding a very small overlap results in a significant improvement in the number of iterations required to converge a finite volume discretization of inviscid compressible flows.[11] In particular, the lowest CPU times were achieved using an overlap regions of just two layers of elements.

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.[6] In numerical simulations with up to 128 processors, Venkatakrishnan shows that the presence of the coarse grid gives some improvement in the performance of the preconditioner in terms of number of iterations, though this does not necessarily translate into faster solution time. Gropp et al. do not employ a coarse space, and show only modest increase in the number of linear iterations for strong scaling results from 32 to 128 processors.[11] In particular, Anderson, Gropp, and collaborators have performed large scale inviscid CFD simulations using over 3000 processors without employing a coarse space.[10,11,44] For these simulations, the use of a coarse space may be unnecessary due to the temporal terms present as a results of the pseudo-transient continuation used to arrive at steady state solutions.[4] For unsteady simulations for the compressible Navier-Stokes equations, Cai, Farhat, and Sarkis find only a small increase in the number of iterations for strong scaling results up to 512 subdomains without the presence of a coarse space.[40,42] 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.[43] We note that 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.[43] For steady state problems solved using a p-sequencing approach with little

or no pseudo-temporal contributions, Diosady[45] showed very poor strong scaling using a similar preconditioner, particularly for viscous problems. In order to improve the parallel scaling of this preconditioner, Diosady presented a partitioning strategy weighted by the strength of the coupling between elements. A similar strategy was also employed by Persson.[43] However, the resulting partitions have larger surface area to volume ratios resulting in less computation per communication. While such a technique improves parallel performance on 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.

3. Schur Complement Methods

In this section, we present Schur complement methods, also known as nonoverlapping or iterative substructuring methods. In general these methods reduce the globally coupled system of equations to a smaller system involving only the degrees of freedom associated with the interface between subdomains. We present the basic ideas for substructuring methods for a continuous finite element discretization in the case of two subdomains, and then discuss the extensions to the case of many subdomains. The presentation in this section closely follows that of Toselli and Widlund.[22] For a full presentation we refer to the books by Toselli and Widlund,[22] Quarteroni and Valli,[46] or Smith, Bjorstad and Gropp.[21]

3.1. *An Interface Problem*

Again, we consider the Poisson problem (1)-(2) in a domain Ω. We partition the domain Ω into two nonoverlapping subdomains Ω_1 and Ω_2, with $\Gamma = \partial\Omega_1 \cap \partial\Omega_2$ the interface between the two subdomains. We may rewrite (1)-(2) as an equivalent coupled problem:

$$-\Delta u_1 = f \qquad \text{in } \Omega_1, \tag{13}$$

$$u_1 = 0 \qquad \text{on } \partial\Omega_1 \cap \partial\Omega, \tag{14}$$

$$u_1 = u_2 \qquad \text{on } \Gamma, \tag{15}$$

$$\frac{\partial u_1}{\partial \boldsymbol{n}_1} = -\frac{\partial u_2}{\partial \boldsymbol{n}_2} \qquad \text{on } \Gamma, \tag{16}$$

$$-\Delta u_2 = f \qquad \text{in } \Omega_2, \tag{17}$$

$$u_2 = 0 \qquad \text{on } \partial\Omega_2 \cap \partial\Omega, \tag{18}$$

where $\boldsymbol{n}_i$ is the outward pointing normal vector from Ω_i. The solutions, u_i, $i = 1, 2$, of the coupled problem gives the restriction of the solution, u, to each subdomain Ω_i. The transmission conditions (15) and (16) ensure that $u_\Gamma := u_1 = u_2$ and $\lambda_\Gamma := \frac{\partial u_1}{\partial \boldsymbol{n}_1} = -\frac{\partial u_2}{\partial \boldsymbol{n}_2}$ on Γ. We note that if u_Γ is known then the u_i's may be obtained by solving independent problems in each subdomain with Dirichlet boundary condition on Γ:

$$-\Delta u_i = f \qquad \text{in } \Omega_i, \tag{19}$$

$$u_i = 0 \qquad \text{on } \partial\Omega_i \cap \partial\Omega, \tag{20}$$

$$u_i = u_\Gamma \qquad \text{on } \Gamma. \tag{21}$$

Alternatively, if λ_Γ is known then the u_i's may be obtained by solving independent problems with Neumann boundary conditions on Γ:

$$-\Delta u_i = f \qquad \text{in } \Omega_i, \tag{22}$$

$$u_i = 0 \qquad \text{on } \partial\Omega_i \cap \partial\Omega, \tag{23}$$

$$\frac{\partial u_i}{\partial \boldsymbol{n}_1} = \lambda_\Gamma \qquad \text{on } \Gamma, \tag{24}$$

Schur complement algorithms are based on a discrete equivalent of the coupled problem (13)-(18). Namely, the discrete problem may be reduced to a system corresponding only to discrete unknowns u_Γ or λ_Γ, on the interface Γ. Once u_Γ or λ_Γ are known the solution interior to each subdomain may be obtained by solving discrete equivalents of the Dirichlet problem (19)-(21) or Neumann problem (22)-(24). Methods which solve for the discrete unknowns corresponding to u_Γ are known as primal substructuring methods, while dual substructuring methods are based on solving the discrete equivalent of the flux λ_Γ.

We now derive a discrete equation for the interface state u_Γ. Once again we consider the discretization of (1)-(2), which results in the discrete system (5). We denote by $u^{(1)}$ and $u^{(2)}$ degrees of freedom associated with nodes on subdomains Ω_1 and Ω_2 respectively. Additionally we use subscript $_\Gamma$ to denote degrees on freedom associated with the interface Γ, while we use subscript $_I$ to denote degrees of freedom strictly interior to a particular subdomain. The discrete system of equations (5) may be written as:

$$\begin{bmatrix} A_{II}^{(1)} & 0 & A_{I\Gamma}^{(1)} \\ 0 & A_{II}^{(2)} & A_{I\Gamma}^{(2)} \\ A_{\Gamma I}^{(1)} & A_{\Gamma I}^{(2)} & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} u_I^{(1)} \\ u_I^{(2)} \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} f_I^{(1)} \\ f_I^{(2)} \\ f_\Gamma \end{bmatrix}, \tag{25}$$

where we note that we have explicitly enforced the discrete equivalent of the first transmission condition (15), namely $u_\Gamma := u_\Gamma^{(1)} = u_\Gamma^{(2)}$. Consider

the following block factorization of the system matrix, A:

$$\begin{bmatrix} I_{II}^{(1)} & 0 & 0 \\ 0 & I_{II}^{(2)} & 0 \\ A_{\Gamma I}^{(1)} A_{II}^{(1)^{-1}} & A_{\Gamma I}^{(2)} A_{II}^{(2)^{-1}} & I_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} A_{II}^{(1)} & 0 & 0 \\ 0 & A_{II}^{(2)} & 0 \\ 0 & 0 & S \end{bmatrix} \begin{bmatrix} I_{II}^{(1)} & 0 & A_{II}^{(1)^{-1}} A_{I\Gamma}^{(1)} \\ 0 & I_{II}^{(2)} & A_{II}^{(2)^{-1}} A_{I\Gamma}^{(2)} \\ 0 & 0 & I_{\Gamma\Gamma} \end{bmatrix}. \tag{26}$$

Where S is the Schur complement given by:

$$S = A_{\Gamma\Gamma} - \sum_{i=1}^{2} A_{\Gamma I}^{(i)} A_{II}^{(i)^{-1}} A_{I\Gamma}^{(i)}. \tag{27}$$

The corresponding inverse of A may be written as, A^{-1}:

$$\begin{bmatrix} I_{II}^{(1)} & 0 & -A_{II}^{(1)^{-1}} A_{I\Gamma}^{(1)} \\ 0 & I_{II}^{(2)} & -A_{II}^{(2)^{-1}} A_{I\Gamma}^{(2)} \\ 0 & 0 & I_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} A_{II}^{(1)^{-1}} & 0 & 0 \\ 0 & A_{II}^{(2)^{-1}} & 0 \\ 0 & 0 & S^{-1} \end{bmatrix} \begin{bmatrix} I_{II}^{(1)} & 0 & 0 \\ 0 & I_{II}^{(2)} & 0 \\ -A_{\Gamma I}^{(1)} A_{II}^{(1)^{-1}} & -A_{\Gamma I}^{(2)} A_{II}^{(2)^{-1}} & I_{\Gamma\Gamma} \end{bmatrix}. \tag{28}$$

We note that the only globally coupled operation involved in computing the inverse given in (28) corresponds to solving a system with the Schur complement S. Namely, (25) may be solved using the following steps:

(1) Compute in parallel the Schur complement residual

$$g_\Gamma = f_\Gamma - \sum_{i=1}^{2} A_{\Gamma I}^{(i)} A_{II}^{(i)^{-1}} f_I^{(i)}. \tag{29}$$

(2) Solve the following global coupled Schur complement problem for u_Γ:

$$S u_\Gamma = g_\Gamma. \tag{30}$$

(3) Compute in parallel the subdomain interior degrees of freedom $u_I^{(i)}$:

$$u_I^{(i)} = A_{II}^{(i)^{-1}} \left(f_I^{(i)} + A_{I\Gamma}^{(i)} u_\Gamma \right), \qquad i = 1, 2 \tag{31}$$

We note that (31) is the discrete equivalent of the continuous Dirichlet problem (19)-(21). It remains to solve the Schur complement problem (30) for u_Γ. The Schur complement S may be too large to solve directly, thus a preconditioned Krylov method may be used to solve (30) iteratively. In the following section we discuss parallel preconditioners for the Schur complement problem (30). In particular, Schwarz methods discussed in Section 2 may also be used as preconditioners for the Schur complement, with the benefit of smaller Krylov vectors corresponding only to interface degrees of freedom.

3.2. *Classical Substructuring Methods*

In this section, we present classical substructuring methods, which are block Jacobi type preconditioners for (30) where the blocks are associated with subdomain faces, edges and vertices. The development of these type of preconditioners for symmetric elliptic problems is presented in a series of papers by Bramble, Pasciak, and Schatz.[47–50]

We consider groups of degrees of freedom on the interface Γ corresponding to the faces, edges, and vertices of subdomains. Namely, we denote by $\mathcal{F}_k$ the set of degrees of freedom interior to a subdomain face associated with exactly two subdomains Ω_i and Ω_j. Similarly, $\mathcal{E}_k$ denotes the set of degrees of freedom on a single edge between several subdomains, while $\mathcal{V}_k$ denotes the degrees of freedom associated with a single node at the cross-points between subdomains. We may consider rewriting the Schur complement matrix as:

$$S = \begin{bmatrix} S_{\mathcal{F}\mathcal{F}} & S_{\mathcal{F}\mathcal{E}} & S_{\mathcal{F}\mathcal{V}} \\ S_{\mathcal{E}\mathcal{F}} & S_{\mathcal{E}\mathcal{E}} & S_{\mathcal{E}\mathcal{V}} \\ S_{\mathcal{V}\mathcal{F}} & S_{\mathcal{V}\mathcal{E}} & S_{\mathcal{V}\mathcal{V}} \end{bmatrix}, \tag{32}$$

where $\mathcal{F}$, $\mathcal{E}$ and $\mathcal{V}$ correspond to the set of subdomain faces, edges and vertices. A simple block diagonal preconditioner for S may be given by dropping the off-diagonal blocks $S_{\mathcal{F}\mathcal{E}}$, $S_{\mathcal{F}\mathcal{V}}$, $S_{\mathcal{E}\mathcal{F}}$, $S_{\mathcal{E}\mathcal{V}}$, $S_{\mathcal{V}\mathcal{F}}$, and $S_{\mathcal{V}\mathcal{E}}$ corresponding to coupling between faces, edges, and vertices, as well as blocks in $S_{\mathcal{F}\mathcal{F}}$, $S_{\mathcal{E}\mathcal{E}}$, and $S_{\mathcal{V}\mathcal{V}}$ corresponding to the coupling between different faces, edges and vertices. We may write this block preconditioner as:

$$M^{-1} = \begin{bmatrix} \bar{S}_{\mathcal{F}\mathcal{F}}^{-1} & 0 & 0 \\ 0 & \bar{S}_{\mathcal{E}\mathcal{E}}^{-1} & 0 \\ 0 & 0 & \bar{S}_{\mathcal{V}\mathcal{V}}^{-1} \end{bmatrix}, \tag{33}$$

where $\bar{S}_{\mathcal{F}\mathcal{F}}$, $\bar{S}_{\mathcal{E}\mathcal{E}}$ and $\bar{S}_{\mathcal{V}\mathcal{V}}$ are the resulting block diagonal matrices. Several simplifications of this basic classical substructuring method exist that replace the blocks associated $\bar{S}_{\mathcal{E}\mathcal{E}}^{-1}$ and $\bar{S}_{\mathcal{V}\mathcal{V}}^{-1}$ with simple approximations, however we do not discuss these here. We note that the preconditioner (33) lacks a coarse space and hence is not scalable for elliptic problems. A coarse space may be added by considering the finite element discretization of the original problem on the coarse mesh whose elements are the subdomains. We may write this preconditioner as:

$$M^{-1} = \begin{bmatrix} \bar{S}_{\mathcal{F}\mathcal{F}}^{-1} & 0 & 0 \\ 0 & \bar{S}_{\mathcal{E}\mathcal{E}}^{-1} & 0 \\ 0 & 0 & \bar{S}_{\mathcal{V}\mathcal{V}}^{-1} \end{bmatrix} + \begin{bmatrix} \hat{R}_{\mathcal{F}}^T & \hat{R}_{\mathcal{E}}^T & I \end{bmatrix} A_H^{-1} \begin{bmatrix} \hat{R}_{\mathcal{F}} \\ \hat{R}_{\mathcal{E}} \\ I \end{bmatrix}, \tag{34}$$

where $\hat{R}_{\mathcal{F}}^T$ and $\hat{R}_{\mathcal{E}}^T$ are interpolation operators from the coarse finite element space to the faces and edges of the original finite element space. For the Poisson problem, the condition number of the preconditioned operator $M^{-1}S$ is bounded by $\kappa \leq C\left(1+\log\left(\frac{H}{h}\right)\right)^2$. We note that this algorithm is scalable, but not optimal since the condition number (and hence convergence rate) depends upon $\frac{H}{h}$. However, the condition number depends only weakly upon $\frac{H}{h}$ and we say that the method is quasi-optimal. Many of the iterative substructuring algorithms presented will have similar condition number bounds. We do not discuss the proofs of these bounds, but refer the reader to the series of papers by Bramble, Pasciak, and Schatz[47–50] or Section 4.6 of Toselli and Widlund.[22]

While originally developed for scalar elliptic problems, algorithms in the spirit of classical substructuring methods have also been applied to systems of equations arising from CFD applications. Cai et al. discussed several classical substructuring variants along with overlapping Schwarz methods for a finite-difference discretization of convection-diffusion problems.[36,51] Gropp and Keyes developed a block triangular preconditioner for the streamfunction-vorticity formulation of two-dimensional flows.[52] Their preconditioner applied to the entire discrete system of equations (5) may be written as:

$$M^{-1} = \begin{bmatrix} A_{II} & A_{I\mathcal{E}} & A_{I\mathcal{V}} \\ 0 & \bar{A}_{\mathcal{E}\mathcal{E}} & A_{\mathcal{E}\mathcal{V}} \\ 0 & 0 & A_H \end{bmatrix}^{-1} . \tag{35}$$

3.3. *Approximate Factorizations*

In the general case, the local solves corresponding to $A_{II}^{(i)^{-1}}$ in (29) and (31) may also be replaced with an approximate solver such as an ILU factorization or a multigrid cycle leading to an approximation of the Schur complement. Thus, steps corresponding to (29)-(31) may be replaced with approximate solvers to provide a preconditioner for the global problem (25). As with Schwarz methods the performance of Schur complement methods in general depend upon the choice of the approximate local solvers.

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.[9] Approximate Schur complements were formed by using an ILU preconditioned GMRES method for the solution of $A_{II}^{(i)^{-1}} A_{I\Gamma}^{(i)}$. Additional approximations were introduced to control the sparsity including element dropping and an approximate Schur complement

formed by considering a small region of elements near the interface.[9] The approximate Schur complement problem was solved using a block preconditioned GMRES method, where the blocks correspond to groups of faces and edges. The blocks which correspond to groups of edges and faces across subdomains provide a global means of correcting low frequency modes, and hence no additional coarse space was required.[9] The use of GMRES for both the local and approximate Schur complement solves means that the preconditioner was non-stationary. Thus the global problem used the flexible variant of GMRES (fGMRES).[53] Barth presented weak scaling results on up to 64 processors, which showed slight performance degradation with increasing number of processors attributed to the growth of the maximum interface size in the partitioning of the domain.[9]

In the case of higher-order finite-difference or finite-volume discretizations it is often convenient to associate each degree of freedom uniquely to a particular processor. In this situation, u_Γ, corresponds to layers of nodes/elements in the interface region, which may be split into groups u_{Γ_i} associated with a particular processor. In particular the degrees of freedom u_{Γ_i} are chosen such that A corresponding to $u_I^{(i)}$ have non-zero columns corresponding only to $u_I^{(i)}$ and u_{Γ_i}.[54] Thus (25) may be rewritten as:

$$\begin{bmatrix} A_{II}^{(1)} & 0 & A_{I\Gamma_1}^{(1)} & 0 \\ 0 & A_{II}^{(2)} & 0 & A_{I\Gamma_2}^{(2)} \\ A_{\Gamma_1 I}^{(1)} & 0 & A_{\Gamma_1\Gamma_2} & A_{\Gamma_1\Gamma_2} \\ 0 & A_{\Gamma_2 I}^{(2)} & A_{\Gamma_1\Gamma_2} & A_{\Gamma_1\Gamma_2} \end{bmatrix} \begin{bmatrix} u_I^{(1)} \\ u_I^{(2)} \\ u_{\Gamma_1} \\ u_{\Gamma_2} \end{bmatrix} = \begin{bmatrix} f_I^{(1)} \\ f_I^{(2)} \\ f_{\Gamma_1} \\ f_{\Gamma_2} \end{bmatrix}. \tag{36}$$

The corresponding Schur complement problem is given by:

$$\begin{bmatrix} S_{\Gamma_1\Gamma_2} & A_{\Gamma_1\Gamma_2} \\ A_{\Gamma_1\Gamma_2} & S_{\Gamma_1\Gamma_2} \end{bmatrix} \begin{bmatrix} u_{\Gamma_1} \\ u_{\Gamma_2} \end{bmatrix} = \begin{bmatrix} g_{\Gamma_1} \\ g_{\Gamma_2} \end{bmatrix}, \tag{37}$$

where $S_{\Gamma_i\Gamma_i} = A_{\Gamma_i\Gamma_i} - A_{\Gamma_i I}^{(i)} {A_{II}^{(i)}}^{-1} A_{I\Gamma_i}^{(i)}$. A simple block Jacobi preconditioner may then be applied to solve (37). Unfortunately, the convergence rate of this method identical to that obtained when applying a subdomain-wise block Jacobi preconditioner to the full system (36). However, if approximate factorizations are used for the local solvers and Schur complement, then an algorithm involving an inner iteration on the approximate Schur complement problem may provide a preconditioner for the full system (36). Such an approach was used by Hicken and Zingg for the solution of a finite difference discretization of the Euler equations.[54] Their algorithm involved an ILU factorization as a local solver, and solved the block-Jacobi

preconditioned approximate Schur complement problem using GMRES as a preconditioner to fGMRES. Numerical results showed good strong scaling performance on up to 48 processors.

4. Neumann-Neumann Methods

In this section we present Neumann-Neumann methods which are a class of preconditioners for the Schur complement problem (30).[55–57] While all of the methods discussed thus far have employed blocks of the fully assembled discrete system as preconditioners, Neumann-Neumann methods exploit the finite element residual assembly. Namely, the discrete system of equations (25) may be obtained by assembling contributions from each subdomain of the form:

$$A^{(i)} = \begin{bmatrix} A_{II}^{(i)} & A_{I\Gamma}^{(i)} \\ A_{\Gamma I}^{(i)} & A_{\Gamma\Gamma}^{(i)} \end{bmatrix}, \qquad f^{(i)} = \begin{bmatrix} f_I^{(i)} \\ f_\Gamma^{(i)} \end{bmatrix}, \qquad i = 1, 2 \tag{38}$$

where $A_{\Gamma\Gamma} = A_{\Gamma\Gamma}^{(1)} + A_{\Gamma\Gamma}^{(2)}$ and $f_\Gamma = f_\Gamma^{(1)} + f_\Gamma^{(2)}$. The local problems:

$$\begin{bmatrix} A_{II}^{(i)} & A_{I\Gamma}^{(i)} \\ A_{\Gamma I}^{(i)} & A_{\Gamma\Gamma}^{(i)} \end{bmatrix} \begin{bmatrix} u_I^{(i)} \\ u_\Gamma^{(i)} \end{bmatrix} = \begin{bmatrix} f_I^{(i)} \\ f_\Gamma^{(i)} + \lambda_\Gamma^{(i)} \end{bmatrix}, \qquad i = 1, 2 \tag{39}$$

correspond to a discrete equivalent of the Neumann problems (22)-(24).

The Schur complement, S, may be also be written as sum of subdomain-wise contributions $S = S^{(1)} + S^{(2)}$, where $S^{(i)} = A_{\Gamma\Gamma}^{(i)} - A_{\Gamma I}^{(i)} {A_{II}^{(i)}}^{-1} A_{I\Gamma}^{(i)}$. In the simplest form, Neumann-Neumann methods precondition $S = S^{(1)} + S^{(2)}$ with $M_{NN}^{-1} = {S^{(1)}}^{-1} + {S^{(2)}}^{-1}$. In practice, diagonal scaling matrices $D^{(i)}$ are used to average nodal values on Γ, such that the Neumann-Neumann preconditioner is given by:

$$M_{NN}^{-1} = \begin{bmatrix} D^{(1)} & D^{(2)} \end{bmatrix} \begin{bmatrix} S^{(1)} & 0 \\ 0 & S^{(2)} \end{bmatrix}^{-1} \begin{bmatrix} D^{(1)} \\ D^{(2)} \end{bmatrix}. \tag{40}$$

where the diagonal values of the scaling matrices are chosen such that at each node the $D^{(i)}$'s sum to 1. For problems with widely varying coefficients across subdomains, the choice of diagonal scaling matrices can significantly impact the performance of the preconditioner.[58]

In order to extend the Neumann-Neumann preconditioner to the case of many subdomains we introduce some additional notation which will be used throughout this section. Consider the partition of the domain Ω into N nonoverlapping subdomains Ω_i, $i = 1, ..., N$. We define $\Gamma_i = \partial\Omega_i \backslash \partial\Omega$, and $\Gamma = \cup_{i=1}^{N} \Gamma_i$. We define R_i as the $\{0, 1\}$ matrix such that $R_i u_\Gamma$ is the

restriction from u_Γ to the degrees of freedom on Γ_i. We may write the global Schur complement system as in (30) with

$$S = \sum_{i=1}^{N} R_i^T S^{(i)} R_i. \tag{41}$$

The extension of the Neumann-Neumann method to the case of many subdomains may be written using the following compact notation:

$$M_{NN}^{-1} = \sum_{i=1}^{N} R_i^T D^{(i)} {S^{(i)}}^{-1} D^{(i)} R_i. \tag{42}$$

In the basic form given in (42), the Neumann-Neumann preconditioner lacks a coarse space and hence is not scalable.[57] Additionally, if a subdomain Ω_i is strictly interior to Ω (i.e $\partial\Omega_i \cap \partial\Omega = \emptyset$), then $S^{(i)}$ is singular, since Ω_i is a "floating" subdomain upon which Neumann boundary conditions are imposed on all of $\partial\Omega_i$. In this case, ${S^{(i)}}^{-1}$ may replaced with a suitable pseudo-inverse or approximate solver, however the performance of the preconditioner will depend upon the particular choice of pseudo-inverse.[57,59,60] The Balancing Domain Decomposition (BDD) method introduced by Mandel[60] addressed the lack of scalability and the issues associated with choosing a suitable pseudo-inverse for singular subdomains, by introducing a coarse space based on the null-spaces of the local Schur complements $S^{(i)}$. The coarse correction step which is applied in a multiplicative manner is known as balancing, and is the origin of the term Balancing Domain Decomposition. The corresponding condition number of the preconditioned system is given by $\kappa = C\left(1 + \log\left(\frac{H}{h}\right)\right)^2$ for the symmetric elliptic problems, where the constant C can be shown to be independent of the coefficients of the problem.[58–60]

The BDD method is closely related to a dual substructuring method known as the Finite Element Tearing and Interconnecting (FETI) method, originally introduced by Farhat and Roux.[61] As opposed to directly enforcing the transmission condition $u_\Gamma^{(1)} = u_\Gamma^{(2)} = u_\Gamma$ by subassembling the global system as in (25), FETI methods enforce the transmission condition through the use of Lagrange multipliers. The Schur complement problem corresponding to the interface degrees of freedom may be written in the following equivalent form:

$$\begin{bmatrix} S^{(1)} & 0 & {B^{(1)}}^T \\ 0 & S^{(2)} & {B^{(2)}}^T \\ B^{(1)} & B^{(2)} & 0 \end{bmatrix} \begin{bmatrix} u_\Gamma^{(1)} \\ u_\Gamma^{(2)} \\ \lambda_\Gamma \end{bmatrix} = \begin{bmatrix} g_\Gamma^{(1)} \\ g_\Gamma^{(2)} \\ 0 \end{bmatrix}, \tag{43}$$

where λ_Γ are Lagrange multipliers which are the discrete equivalent of the flux $\frac{\partial u}{\partial \boldsymbol{n}_1}$ on Γ. We note that $B^{(1)}$ and $B^{(2)}$ are matrices with values of $\{0, 1, -1\}$ which ensure the condition $u_\Gamma^{(1)} = u_\Gamma^{(2)}$ is enforced through the last block equation of (43). In Neumann-Neumann methods, (40), the fully assembled Schur complement system, (30), is preconditioned using the upper-diagonal block of (43) obtained by dropping the rows and columns corresponding to the Lagrange multiplier λ_Γ. In FETI methods, on the other hand, the system, (43), is reduced to a system corresponding to only the Lagrange multipliers λ_Γ, which is preconditioned using the local Schur complement matrices.

In this paper we do not present FETI methods in detail but note that the FETI and BDD method are closely related, and have similar eigenvalue spectra.[62,63] 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.[64] FETI methods have also been analyzed for the case where inexact solvers are used.[65]

4.1. *BDDC and FETI-DP*

The most advanced of the FETI and Neumann-Neumann class of methods are the dual-primal FETI (FETI-DP)[66,67] and the Balancing Domain Decomposition by Constraints (BDDC) method.[68,69] Like FETI and BDD, FETI-DP and BDDC methods are closely related and have essentially the same eigenvalue spectra.[70,71]

A key component of FETI-DP and BDDC methods involves enforcing the continuity of a small number of "primal" degrees of freedom across subdomains. Strictly enforcing the continuity of the primal degrees of freedom naturally introduces a coarse space ensuring that the FETI-DP and BDDC methods are scalable. Additionally, the constraint on the continuity of the local subdomain problems ensures that the local problems are not singular. On each subdomain the degrees of freedom $u_\Gamma^{(i)}$ are partitioned into primal and dual degrees of freedom $u_\Pi^{(i)}$ and $u_\Delta^{(i)}$, where the primal degrees of freedom correspond to nodal values at subdomain corners, or averages along subdomain edges or faces. As opposed to directly enforcing the transmission condition $u_\Gamma^{(1)} = u_\Gamma^{(2)} = u_\Gamma$ by subassembling the global system a partially subassembled system is obtained by enforcing the continuity of only the primal degrees of freedom $u_\Pi^{(1)} = u_\Pi^{(2)} = u_\Pi$. The corresponding

subassembled problem may be written as:

$$\begin{bmatrix} S^{(1)}_{\Delta\Delta} & 0 & S^{(1)}_{\Delta\Pi} & B^{(1)^T}_{\Delta} \\ 0 & S^{(2)}_{\Delta\Delta} & S^{(2)}_{\Delta\Pi} & B^{(2)^T}_{\Delta} \\ S^{(1)}_{\Pi\Delta} & S^{(2)}_{\Pi\Delta} & S_{\Pi\Pi} & 0 \\ B^{(1)}_{\Delta} & B^{(2)}_{\Delta} & 0 & 0 \end{bmatrix} \begin{bmatrix} u^{(1)}_{\Delta} \\ u^{(2)}_{\Delta} \\ u_{\Pi} \\ \lambda_{\Delta} \end{bmatrix} = \begin{bmatrix} g^{(1)}_{\Delta} \\ g^{(2)}_{\Delta} \\ g_{\Pi} \\ 0 \end{bmatrix}, \tag{44}$$

where $S_{\Pi\Pi} = S^{(1)}_{\Pi\Pi} + S^{(2)}_{\Pi\Pi}$ and $f_{\Pi} = f^{(1)}_{\Pi} + f^{(2}_{\Pi}$, while $B^{(i)}_{\Delta}$ are chosen such that the last row enforces $u^{(1)}_{\Delta} = u^{(2)}_{\Delta}$.

In the FETI-DP methods, the partially assembled system, (44) is reduced to a system for the Lagrange multipliers λ_{Δ}, which is preconditioned by solving local constrained Neumann problems corresponding to $S^{(i)}_{\Delta\Delta}$. Once again, we do not describe the FETI-DP method in detail, but refer the reader to the references provided. In BDDC methods, the upper-diagonal block of partially assembled system, (44), is used to precondition the fully assembled Schur complement problem, (30), by averaging $u^{(i)}_{\Delta}$'s. We write the BDDC preconditioner as:

$$M^{-1}_{BDDC} = \begin{bmatrix} D^{(1)}_{\Delta} & D^{(2)}_{\Delta} & 0 \\ 0 & 0 & I_{\Pi} \end{bmatrix} \begin{bmatrix} S^{(1)}_{\Delta\Delta} & 0 & S^{(1)}_{\Delta\Pi} \\ 0 & S^{(2)}_{\Delta\Delta} & S^{(2)}_{\Delta\Pi} \\ S^{(1)}_{\Pi\Delta} & S^{(2)}_{\Pi\Delta} & S_{\Pi\Pi} \end{bmatrix}^{-1} \begin{bmatrix} D^{(1)}_{\Delta} & 0 \\ D^{(2)}_{\Delta} & 0 \\ 0 & I_{\Pi} \end{bmatrix}, \tag{45}$$

where $D^{(i)}_{\Delta}$ are diagonal scaling matrices corresponding to the dual degrees of freedom u_{Δ}. Prior to extending the BDDC method to the case of many subdomains, we introduce some additional notation. Let $R_{\Delta,i}$ be the $\{0,1\}$ matrix which extract degrees of freedom $u^{(i)}_{\Delta}$ from the globally assembled interface vector u_{Γ}, (i.e. $u^{(i)}_{\Delta} = R_{\Delta,i} u_{\Gamma}$). Similarly, we define R_{Π} to be the matrix such that $u_{\Pi} = R_{\Pi} u_{\Gamma}$, while $R_{\Pi,i}$ is defined such that $u^{(i)}_{\Pi} = R_{\Pi,i} u_{\Pi}$. The solution of the partially assembled system in the BDDC preconditioner may be written as the sum of independent constrained Neumann solves corresponding to $S^{(i)}_{\Delta\Delta}$ and a coarse solve involving only the primal degrees of freedom u_{Π}. Namely, we may write the BDDC preconditioner for the case of many subdomains as:

$$M^{-1}_{BDDC} = \Psi S_0^{-1} \Psi^{*^T} + \sum_{i=1}^{N} R^{(i)^T}_{\Delta} D^{(i)}_{\Delta} S^{-1}_{\Delta\Delta} D^{(i)}_{\Delta} R^{(i)}_{\Delta}, \tag{46}$$

where S_0, Ψ and Ψ^* are given by:

$$S_0 = \sum_{i=1}^{N} R_{\Pi,i}^T \left(S_{\Pi\Pi}^{(i)} - S_{\Pi\Delta}^{(i)} {S_{\Delta\Delta}^{(i)}}^{-1} S_{\Delta\Pi}^{(i)} \right) R_{\Pi,i}, \tag{47}$$

$$\Psi = R_{\Pi}^T + \sum_{i=1}^{N} R_{\Delta,i}^T D_{\Delta}^{(i)} {S_{\Delta\Delta}^{(i)}}^{-1} S_{\Delta\Pi}^{(i)} R_{\Pi,i}, \tag{48}$$

$$\Psi^* = R_{\Pi}^T + \sum_{i=1}^{N} R_{\Delta,i}^T D_{\Delta}^{(i)} {S_{\Delta\Delta}^{(i)}}^{-T} {S_{\Pi\Delta}^{(i)}}^{T} R_{\Pi,i}. \tag{49}$$

The BDDC and FETI-DP methods are amongst the most successful domain decomposition methods for second order elliptic problems and problems of structural mechanics. The analysis of these preconditioners have been extended to the case where inexact solvers are used for the local Dirichlet and constrained Neumann problems.[72–74] Additionally, several authors have presented multi-level versions of the BDDC method when the coarse problem corresponding to S_0 may be too large to solve exactly.[75–77] An adaptive method for adding primal degrees of freedom to ensure rapid convergence has also been presented.[78] Practical implementations of the FETI-DP method has been used to solve structural mechanics problem on up to 3000 processor.[79,80]

The extension of FETI and Neumann-Neumann methods, (and thus FETI-DP and BDDC) to convection-diffusion problems involves modifying the interface conditions for the local subdomain problems to ensure that these local problems are well posed in the convective limit. In particular, imposing Neumann conditions on the inflow portion of a subdomain may lead to a singular system. Achdou et al. replaced the Neumann-Neumann interface condition with a Robin-Robin interface condition,[81] ensuring that the local bilinear forms were coercive. A Fourier analysis, on a vertical strip partitioning of the domain, showed that in the convective limit, the resulting algorithm converges in a number of iterations equal to half the number of subdomains in the streamwise direction. The Robin-Robin interface conditions have been used along with a FETI method to solve linear convection-diffusion problems by Toselli.[82] Similarly, Tu and Li used the Robin-Robin interface condition to extend the BDDC method to convection-diffusion problems.[83] 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, in a manner analo-

gous to additive Schwarz methods, the behaviour of BDDC preconditioner matches the symmetric, diffusion dominated limit if the subdomain Peclet number is sufficiently small.

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.[84,85] Yano and Darmofal used a generalization of the Robin-Robin interface condition to the Euler equations based on entropy symmetrization theory.[86,87] They solved a higher-order continuous finite element discretization for two-dimensional subsonic flow using a BDDC preconditioner with up to 128 subdomains.

The success of BDDC and FETI-DP preconditioner for structural mechanics problems, and the initial results of Yano and Darmofal motivates further research into attempting to apply these types of preconditioners to large scale CFD simulations. While originally developed for linear conforming finite element methods, Neumann-Neumann type preconditioners have been extended to mixed methods,[88,89] discontinuous Galerkin discretizations[90] and higher-order spectral element methods.[91,92] We note that Neumann-Neumann type preconditioners exploit the finite element construction of the discrete system of equations, where subdomain contributions provide a discrete analog of the continuous Neumann problems (22)-(24). For finite-difference or finite-volume discretizations which do not naturally have such a finite element construction the choice for the local discrete Neumann problems and the analogy to the continuous Neumann problem is unclear. These issues need to be addressed in the context of simple model problems prior to consider using Neumann-Neumann type methods for these types of discretizations.

5. Numerical Results

In this final section we present numerical results using different preconditioning methods discussed for the solution of a higher-order hybridizable discontinuous Galerkin (HDG) discretization. The HDG discretization was recently introduced for the solution of the Poisson problem[93] , then extended to convection-diffusion equations[94] and the compressible Euler and Navier-Stokes equations.[95] The HDG discretization is a mixed method where both the state variable and its gradient are approximated separately

on each element. A unique value of the trace of the state variable is obtained by enforcing the continuity of the flux on element boundaries leading to a reduced system of equations where the only globally coupled degrees of freedom are associated with the trace values on element faces.

We solve the following convection-diffusion problem in the square domain $\Omega \in \mathbb{R}^2$ given by $\Omega = [0,1] \times [0,1]$:

$$\nabla \cdot (\boldsymbol{c}u) - \kappa \Delta u = f \qquad \text{in } \Omega \subset \mathbb{R}^2, \tag{50}$$

where $u(x,y)$ is the state, $\boldsymbol{c} = (1,0)$ is the convective velocity, and κ is the viscosity. Here f is a source function set such that the exact solution is given by:

$$u = e^{-y/\sqrt{\kappa \bar{x}}} \qquad \text{where } \bar{x} = x + 0.1. \tag{51}$$

The linear system resulting from the HDG discretization is solved using a right-preconditioned GMRES method. We examine the performance of three different parallel preconditioners: a minimum-overlap additive Schwarz preconditioner without a coarse space (ASM); a minimum-overlap additive Schwarz preconditioner with a coarse space added in a multiplicative manner (ASM_0); and a nonoverlapping BDDC preconditioner. As the globally coupled degrees of freedom of the HDG discretization correspond to the element edges the communication pattern in the application of the overlapping preconditioner is essentially the same as for a nonoverlapping method. A coarse space for the additive Schwarz preconditioner is defined using an algebraic multigrid approach where edges are agglomerated by using the graph partitioning algorithm ParMETIS,[96] resulting in an agglomeration of edges independent of the original partitioning of the domain. The number of agglomerated edges is chosen such that the resulting coarse space contains about twice the number of degrees of freedom as the corresponding coarse space for the BDDC preconditioner. For the BDDC preconditioner the coarse space is defined by choosing as primal degrees of freedom the average of the state along the interfaces between subdomains. The corresponding dual degrees of freedom have zero average on subdomain interfaces. A particular advantage of the BDDC preconditioner is the simple algebraic construction of the coarse space given the original partition of the domain. In the context of the HDG discretization, this is particularly important as multigrid type algorithms have not been studied for this type of discretization.

Numerical experiments are presented to show the performance of the three preconditioners over a large range of viscosity, κ, highlighting the

difference between the diffusion- and convection-dominated limits. While CPU time is the most appropriate metric for the comparison of different algorithms, the CPU time is closely tied to a particular implementation of the algorithm. In order to avoid these implementation dependent comparisons, the performance of the preconditioners are presented in terms of the number of iterations required for the GMRES algorithm. The relative computational cost may be estimated by taking into consideration the cost of each Krylov iteration. In particular, a single Krylov iteration involves:

- ASM: one Jacobian multiplication and one local Dirichlet solve on each subdomain
- ASM_0: two Jacobian multiplications and one local Dirichlet solve on each subdomain and a global coarse solve
- BDDC: one Jacobian multiplication, two local Dirichlet solves and one local constrained Neumann solve on each subdomain and a global coarse solve.

For the numerical experiments presented, exact solvers are used for the local and global problems. In a practical setting, approximate solvers would be employed, and thus the relative cost of the preconditioners will, in general, depend upon the choice of approximate solver. In particular, for the BDDC preconditioner, if a triangular factorization is employed as the local solver, the two Dirichlet solves may be replaced by one forward- and one back-substitution resulting in a cost equivalent to only a single local solve. As a reference for comparing the different preconditioners, we may consider using a local solver which has a computational cost which is the same as the cost of applying the Jacobian matrix, while we assume that the cost of the global solve is insignificant in comparison to the local solves. Thus the relative cost of a Krylov iteration for the ASM, ASM_0 and BDDC preconditioners is approximately 2:3:3.

In the first numerical experiment, we solve the convection-diffusion problem, (50), on a structured mesh. The domain Ω is partitioned into N square subdomains in an $\sqrt{N} \times \sqrt{N}$ structured pattern. Locally, each subdomain consists of n elements obtained by dividing Ω into squares of equal size and splitting each square into two triangular elements. We examine the performance of the preconditioners varying N and n, for higher order solution with $p = 2$ and $p = 5$. Tables 1 and 2 show the number of GMRES iterations required to converged the l_2-norm of the residual by a factor of 10^4 for $\kappa = 1$ and $\kappa = 10^{-6}$, respectively.

In the diffusion-dominated limit, ($\kappa = 1$), for a fixed number of elements

Table 1. Number of GMRES iterations for $\kappa = 1$ on isotropic structured mesh

		$p=2$			$p=5$		
N	n	ASM	ASM_0	BDDC	ASM	ASM_0	BDDC
4	128	23	21	5	28	25	5
16	128	45	31	8	55	38	10
64	128	86	41	9	103	51	11
256	128	168	47	10	201	59	11
1024	128	329	50	10	393	62	12
64	8	42	21	8	51	26	9
64	32	61	29	8	73	36	11
64	128	86	41	9	103	51	11
64	512	122	58	11	146	71	12
64	2048	172	82	11	205	100	13

per subdomain, $n = 128$, the performance of the ASM preconditioner degrades as the number of subdomains, N, increases. This behaviour is due to a lack of a coarse space able to control the lower frequency error modes. On the other hand, for the ASM_0 and BDDC preconditioners, the number of iterations appears to be bounded as the number of subdomains increases. For a fixed number of subdomains $N = 64$, the performance of the ASM_0 preconditioner degrades rapidly with increasing number of elements, due to the non-optimality of this preconditioner in the case of small overlap. On the other hand the number of iterations for the BDDC preconditioner which is quasi-optimal increases only slowly with increasing number of elements.

Table 2. Number of GMRES iterations for $\kappa = 10^{-6}$ on isotropic structured mesh

		$p=2$			$p=5$		
N	n	ASM	ASM_0	BDDC	ASM	ASM_0	BDDC
4	128	3	5	1	3	5	1
16	128	5	10	2	5	9	3
64	128	9	15	4	9	15	5
256	128	17	27	9	17	26	10
1024	128	33	48	18	33	47	19
64	8	9	12	4	9	11	4
64	32	9	14	4	9	14	5
64	128	9	15	4	9	15	5
64	512	9	15	5	9	15	6
64	2048	9	16	5	9	16	6

In the convection-dominated limit, ($\kappa = 10^{-6}$), all three preconditioners converge in a small number of iterations, proportional to the number of subdomains in the streamwise direction ($\sqrt{N}$). For this test case, the

boundary layer region is not resolved, and hence diffusive effects are relevant only on the subdomains along the bottom wall. In particular, a coarse space is not justified as the ASM method converges in fewer iterations than the more expensive ASM_0 preconditioner. Additionally, we note that in the convection-dominated limit the number of iterations to converge appears essentially independent of the number of elements per subdomain for all three preconditioners.

Table 3. Number of GMRES iterations for $\kappa = 10^{-6}$ on anisotropic structured mesh

		$p=2$			$p=5$		
N	n	ASM	ASM_0	BDDC	ASM	ASM_0	BDDC
4	128	3	5	1	3	7	1
16	128	17	15	5	18	16	5
64	128	33	25	8	34	27	8
256	128	70	41	13	71	44	14
1024	128	170	75	35	172	80	36
64	8	18	15	8	19	16	8
64	32	24	18	8	24	20	8
64	128	33	25	8	34	27	8
64	512	50	35	8	51	38	8
64	2048	76	56	8	77	58	8

While the results of Table 2 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 second 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 $Pe = |\boldsymbol{c}|/\kappa$ is the Peclet number. Table 3 shows the number of iterations required for the GMRES algorithm to converge by a factor of 10^4 for $\kappa = 10^{-6}$. As a significant portion of the mesh is in the boundary layer region, diffusive effects become more important. Compared to Table 2, the performance of the ASM preconditioner without a coarse space is seen to degrade relative to the ASM_0 and BDDC preconditioners.

In Table 4 we show the performance on both the isotropic and anisotropic meshes over a range of viscosities, for fixed N and n. 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.

Table 4. Number of GMRES iterations on both isotropic and anisotropic meshes with $N = 256$, $n = 8$

		$p = 2$			$p = 5$		
	κ	ASM	ASM_0	BDDC	ASM	ASM_0	BDDC
Isotropic Mesh	1	86	41	9	103	51	11
	10^{-1}	77	45	9	93	56	11
	10^{-2}	34	32	10	40	40	12
	10^{-4}	15	20	7	17	22	9
	10^{-4}	9	16	7	10	19	7
	10^{-5}	9	15	5	9	15	6
	10^{-6}	9	15	4	9	15	5
Anisotropic Mesh	1	86	41	9	103	51	11
	10^{-1}	90	42	11	111	54	13
	10^{-2}	71	39	11	81	47	14
	10^{-3}	53	31	9	57	34	11
	10^{-4}	43	29	8	44	31	9
	10^{-5}	37	27	9	37	29	9
	10^{-6}	33	25	8	34	27	8

In the final numerical experiment we show the performance of the three preconditioners on unstructured meshes. We solve the convection-diffusion problem with $\kappa = 10^{-3}$. A family of four anisotropic meshes with 1475, 5992, 23492, and 94313 elements were generated using the Bidimensional Anisotropic Mesh Generator (BAMG),[97] where the anisotropic metric was determined by the Hessian of the exact solution, (51). The meshes are partitioned using the ParMETIS package of Karypis,[96] into 4, 16, 64 and 256 subdomains, resulting in each subdomain having approximately 370 elements. Table 5 shows the resulting performance of the three preconditioners. Unfortunately, the performance of the preconditioners for the unstructured case is, in general, much poorer than the unstructured case. However, the importance of a coarse space is highlighted even in this test case.

Table 5. Number of GMRES iterations on unstructured anisotropic meshes with $\kappa = 10^{-3}$, $n \sim 370$

	$p = 2$			$p = 5$		
N	ASM	ASM_0	BDDC	ASM	ASM_0	BDDC
4	64	56	12	84	76	15
16	162	111	35	196	146	44
64	418	214	79	491	270	94
256	> 1000	377	158	> 1000	461	187

In summary, the numerical results presented show that the ASM_0 and BDDC preconditioners equipped with coarse spaces perform much better than the ASM preconditioner without a coarse space even in the convection dominated limit. In particular, the performance of BDDC preconditioner is only weakly dependent upon the number of elements per subdomain, and thus is expected to perform better than the ASM_0 preconditioner as the size of the subdomains in increased. Finally, we note that for the numerical test cases presented the performance of both ASM_0 and BDDC preconditioners appear to be only weakly dependent upon p. In particular, for the convection-dominated, ($\kappa = 10^{-6}$), test cases the number of iterations for $p = 2$ and $p = 5$ are essentially the same. Thus, these types of preconditioners may be suited to higher-order CFD simulations.

References

1. D. J. Mavriplis, D. Darmofal, D. Keyes, and M. Turner. AIAA 2007-4084, (2007).
2. G. Amdahl. In *AFIPS Conference Proceedings*, vol. 30, pp. 483–485. AFIPS Press, Reston, Va, (1967).
3. J. L. Gustafson, *Communications of the ACM.* **31**, 532–533, (1988).
4. C. T. Kelley and D. E. Keyes, *SIAM J. Numer. Anal.* **35**(2), 508–523, (1998).
5. W. Anderson, R. Rausch, and D. Bonhaus. AIAA 1995-1740, (1995).
6. V. Venkatakrishnan. ICASE 95-28, (1995).
7. X.-C. Cai, W. D. Gropp, D. E. Keyes, and M. D. Tidriri. pp. 17–30. Proceedings of the International Workshop on Numerical Methods for the Navier-Stokes Equations, (1995).
8. D. J. Mavriplis. AIAA 1998-2966, (1998).
9. T. J. Barth, T. F. Chan, and W.-P. Tang, *Contemp. Math.* **218**, 23–41, (1998).
10. W. Gropp, D. K. Kaushik, B. F. Smith, and D. E. Keyes. In *HiPC '00: 7th Int. Conf. on HPC*, pp. 395–404. Springer-Verlag, (2000).
11. W. Gropp, D. Keyes, L. C. Mcinnes, and M. D. Tidriri, *Int. J. High Perform. Comput. Appl.* **14**(2), 102–136, (2000).
12. D. A. Knoll and D. E. Keyes, *J. Comput. Phys.* **193**(1), 357–397, (2004).
13. A. Nejat and C. Ollivier-Gooch. AIAA 2007-0719 (Jan., 2007).
14. D. J. Mavriplis, *J. Comput. Phys.* **145**, 141–165, (1998).
15. D. J. Mavriplis and S. Pirzadeh, *AIAA J. Aircraft.* **36**, 987–998, (1999).
16. K. J. Fidkowski, T. A. Oliver, J. Lu, and D. L. Darmofal, *J. Comput. Phys.* **207**(1), 92–113, (2005).
17. C. R. Nastase and D. J. Mavriplis, *J. Comput. Phys.* **213**(1), 330–357, (2006).
18. C. R. Nastase and D. J. Mavriplis. AIAA 2007-0512, (2007).
19. P.-O. Persson and J. Peraire, *SIAM J. Sci. Comput.* **30**(6), 2709–2722, (2008).

20. D. J. Mavriplis, *J. Comput. Phys.* **175**(1), 302–325, (2002).
21. B. Smith, P. Bjorstad, and W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. (Cambridge University Press, New York, NY, 1996).
22. A. Toselli and O. Widlund, *Domain Decomposition Methods Algorithm and Theory*. (Springer-Verlag, 2005).
23. M. Dryja and O. Widlund. Tech report 339, Department of Computer Science, Courant Institute, (1987).
24. X.-C. Cai, *SIAM J. Sci. Comput.* **14**(1), 239–247, (1993).
25. X.-C. Cai. In *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pp. 232–244, Philidelphia, (1990).
26. X.-C. Cai. In *Domain-Based Parallelism and Problem Decomposition Methods in Computational Science and Engineering*, pp. 1–19. SIAM, (1995).
27. X.-C. Cai, *Numer. Math.* **60**, 41–61, (1991).
28. X.-C. Cai, *SIAM J. Sci. Comput.* **15**(3), 587–603, (1994).
29. S. C. Brenner, *Math. Comp.* **65**(215), 897–921, (1996).
30. M. A. Casarin, *SIAM J. Numer. Anal.* **34**(6), 2482–2502, (1997).
31. X. Feng and O. A. Karakashian, *SIAM J. Numer. Anal.* **39**(4), 1343–1365, (2002).
32. P. F. Antonietti and B. Ayuso. In *Domain Decomposition Methods in Science and Engineering*, vol. 60, pp. 185–192. Springer Berlin, (2008).
33. P. F. Antonietti and B. Ayuso, *Math. Model. Numer. Anal.* **41**(1), 21–54, (2007).
34. P. F. Antonietti and B. Ayuso. In *Communications in Computational Physics*, vol. 5, pp. 398–412, (2009).
35. C. Lasser and A. Toselli, *Math. Comp.* **72**, 1215–1238, (2003).
36. X.-C. Cai, W. D. Gropp, and D. E. Keyes, *J. Comput. Phys.* **157**, 1765–1774, (2000).
37. J. W. Lottes and P. F. Fischer, *J. Sci. Comput.* **24**(1), 45–78, (2005).
38. L. Olson, J. Hesthaven, and L. Wilcox. pp. 325–332. Domain Decomposition Methods in Science and Engineering XVI, (2007).
39. X.-C. Cai and M. Sarkis, *SIAM J. Sci. Comput.* **21**(2), 792–797, (1999).
40. X.-C. Cai, C. Farhat, and M. Sarkis, *Contemp. Math.* **218**, 479–485, (1998).
41. X.-C. Cai, C. Farhat, and M. Sarkis. ICASE 96-48, (1996).
42. X.-C. Cai, W. D. Gropp, D. E. Keyes, and M. D. Tidriri. In *Domain Decomposition Methods in Science and Engineering*. John Wiley & Sons, (1997).
43. P.-O. Persson. AIAA 2009-606, (2009).
44. W. K. Anderson, W. D. Gropp, D. K. Kaushik, D. E. Keyes, and B. F. Smith. In *Proceedings of SC99*, pp. 69–80. Portland, OR, (1999).
45. L. T. Diosady. 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. A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*. (Oxford, New York, 1999).
47. J. H. Bramble, J. E. Pasciak, and A. H. Schatz, *Math. Comp.* **47**(175), 103–134 (July, 1986).

48. J. H. Bramble, J. E. Pasciak, and A. H. Schatz, *Math. Comp.* **49**(179), 1–16 (July, 1987).
49. J. H. Bramble, J. E. Pasciak, and A. H. Schatz, *Math. Comp.* **51**(184), 415–430 (October, 1988).
50. J. H. Bramble, J. E. Pasciak, and A. H. Schatz, *Math. Comp.* **53**(187), 1–24 (July, 1989).
51. X.-C. Cai, W. D. Gropp, and D. E. Keyes, *Numer. Math.* **61**, 153–169, (1992).
52. W. Gropp and D. Keyes, *Internat. J. Numer. Methods Fluids.* **14**, 147–165, (1992).
53. Y. Saad, *SIAM J. Sci. Comput.* **14**(2), 461–469, (1993).
54. J. E. Hicken and D. W. Zingg. AIAA 2007-4333, (2007).
55. J.-F. Bourgat, R. Glowinski, P. L. Tallec, and M. Vidrascu. In eds. T. Chan, R. Glowinski, J. Periaux, and O. Widlund, *Domain decomposition methods. Second international symposium on domain decomposition methods*, pp. 3–16. SIAM, (1988).
56. P. L. Tallec, Y. D. Roeck, and M. Vidrascu, *J. Comput. Appl. Math.* **34**, 93–117, (1991).
57. Y.-H. DeRoeck and P. LeTallec. In *Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pp. 112–128, Philidelphia, PA, (1991). SIAM.
58. J. Mandel and M. Brezina, *Math. Comp.* **65**(216), 1387–1401, (1996).
59. M. Dryja, *Comm. Pure Appl. Math.* **48**, 121–155, (1995).
60. J. Mandel, *Comm. Numer. Methods Engrg.* **9**, 233–241, (1993).
61. C. Farhat and F.-X. Roux, *Internat. J. Numer. Methods Engrg.* **32**, 1205–1227, (1991).
62. C. Farhat, J. Mandel, and F.-X. Roux, *Comput. Methods Appl. Mech. Engrg.* **115**, 365–385, (1994).
63. Y. Fragakis and M. Papadrakakis, *Comput. Methods Appl. Mech. Engrg.* **192**, 3799–3830, (2003).
64. M. Bhardwaj, D. Day, C. Farhat, M. Lesoinne, K. Pierson, and D. Rixen, *Int. J. Numer. Meth. Engng.* **47**, 513–535, (2000).
65. A. Klawonn and O. B. Widlund, *SIAM J. Sci. Comput.* **22**(4), 1199–1219, (2000).
66. C. Farhat, M. Lesoinne, P. LeTallec, K. Pierson, and D. Rixen, *Internat. J. Numer. Methods Engrg.* **50**, 1523–1544, (2001).
67. J. Mandel and R. Tezaur, *Numer. Math.* **88**, 543–558, (2001).
68. C. R. Dohrmann, *SIAM J. Sci. Comput.* **25**(1), 246–258, (2003).
69. J. Mandel and C. R. Dohrmann, *Numer. Linear Algebra Appl.* **10**, 639–659, (2003).
70. J. Mandel, C. R. Dohrmann, and R. Tezaur, *Appl. Numer. Math.* **54**, 167–193, (2005).
71. J. Li and O. B. Widlund, *Internat. J. Numer. Methods Engrg.* **66**, 250–271, (2006).
72. C. R. Dohrmann, *Numer. Linear Algebra Appl.* **14**, 149–168, (2007).
73. J. Li and O. B. Widlund, *Comput. Methods Appl. Mech. Engrg.* **196**, 1415–1428, (2007).

74. A. Klawonn and O. Rheinbach, *Internat. J. Numer. Methods Engrg.* **69**, 284–307, (2007).
75. J. Mandel, B. Sousedik, and C. R. Dohrmann, *Lecture Notes in Computational Science and Engineering*. **60**, 287–294, (2008).
76. J. Mandel and B. Sousedik, *Computing*. **83**, 55–85, (2008).
77. X. Tu, *SIAM J. Sci. Comput.* **29**(4), 1759–1780, (2007).
78. J. Mandel and B. Sousedik, *Comput. Methods Appl. Mech. Engrg.* **196**, 1389–1399, (2007).
79. M. Bhardwaj, K. Pierson, G. Reese, T. Walsh, D. Day, K. Alvin, J. Peery, C. Farhat, and M. Lesoinne. In *Proceedings of the 2002 ACM/IEEE conference on supercomputing*, pp. 35–53, Baltimore, MD, (2002).
80. K. H. Pierson, G. M. Reese, M. K. Bhardwaj, T. F. Walsh, and D. M. Day. Sandia National Laboratories SAND2002-1371, (2002).
81. Y. Achdou, P. L. Tallec, F. Nataf, and M. Vidrascu, *Comput. Methods Appl. Mech. Engrg.* **184**, 145–170, (2000).
82. A. Toselli, *Comput. Methods Appl. Mech. Engrg.* **190**, 5759–5776, (2001).
83. X. Tu and J. Li, *Comm. Appl. Math. Comp. Sci.* **3**(1), 25–60, (2008).
84. V. Dolean, F. Nataf, and G. Rapin, *Comptes Rendus Mathematique*. **340**(9), 693 – 696, (2005).
85. V. Dolean and F. Nataf, *Math. Model. Numer. Anal.* **40**(4), 689–704, (2006).
86. M. Yano. Massively parallel solver for the high-order Galerkin least-squares method. Masters thesis, Mass. Inst. of Tech., CDO (May, 2009).
87. M. Yano and D. Darmofal, *Comput. Methods Appl. Mech. Engrg.* (2010).
88. X. Tu, *Electron. Trans. Numer. Anal.* **20**, 164–179, (2005).
89. X. Tu, *Electron. Trans. Numer. Anal.* **26**, 146–160, (2007).
90. M. Dryja, J. Galvis, and M. Sarkis, *J. Complexity*. **23**(4), 715–739, (2007).
91. A. Toselli and X. Vasseur, *IMA Journal on Numerical Analysis*. **24**, 123–156, (2004).
92. A. Klawonn, L. F. Pavarino, and O. Rheinbach, *Comput. Methods Appl. Mech. Engrg.* **198**, 511–523, (2008).
93. B. Cockburn, J. Gopalakrishnan, and R. Lazarov, *SIAM J. Numer. Anal.* **47** (2), 1319–1365, (2009).
94. N. Nguyen, J. Peraire, and B. Cockburn, *J. Comput. Phys.* **228**(9), 3232–3254, (2009).
95. J. Peraire, N. Nguyen, and B. Cockburn. AIAA 2010-363, (2010).
96. G. Karypis. Parmetis: Parallel graph partitioning and sparse matrix ordering library, (2006). http://glaros.dtc.umn.edu/gkhome/views/metis/parmetis.
97. F. Hecht. Bamg: Bidimensional anisotropic mesh generator, (1998). `http://www-rocq1.inria.fr/gamma/cdrom/www/bamg/eng.htm`.

Index