Fast Iterative Solver for Convection-Diffusion Systems with Spectral Elements

P. Aaron Lott,[1] **Howard Elman**[2]

[1] *Mathematical and Computational Sciences Division, National Institute of Standards and Technology, Gaithersburg, Maryland 20899*

[2] *Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, Maryland 20742*

Received 9 March 2009; accepted 12 July 2009
Published online 16 October 2009 in Wiley Online Library (wileyonlinelibrary.com).
DOI 10.1002/num.20518

We introduce a solver and preconditioning technique based on Domain Decomposition and the Fast Diagonalization Method that can be applied to tensor product based discretizations of the steady convection–diffusion equation. The method is based on a Robin–Robin interface preconditioner coupled to a fast diagonalization solver which is used to efficiently eliminate the interior degrees of freedom and perform subsidiary subdomain solves. Using a spectral element discretization, we first apply our technique to constant wind problems, and then propose a means for applying the technique as a preconditioner for variable wind problems. We demonstrate that iteration counts are mildly dependent on changes in mesh size and convection strength.
 Numer Methods Partial Differential Eq 27: 231–254, 2011

Keywords: convection-diffusion; domain decomposition; preconditioning; spectral element method

I. INTRODUCTION

Numerical simulation of fluid flow allows for improved prediction and design of natural and engineered systems such as those involving water, oil, or blood. The interplay between inertial and viscous forces in a fluid flow dictates the length scale where energy is transferred, thus determining the resolution required to capture flow information accurately. This resolution requirement poses computational challenges in situations where the convective nature of the flow dominates diffusive effects. In such flows, convection and diffusion occur on disparate scales, causing sharp flow features that require fine numerical grid resolution. This leads to a large system of equations which is often solved using an iterative method. Exacerbating the challenge of solving a large linear system, the discrete convection–diffusion operator is nonsymmetric and poorly conditioned.

Correspondence to: P. Aaron Lott, Mathematical and Computational Sciences Division, National Institute of Standards and Technology, Gaithersburg, Maryland 20899 (e-mail: Aaron.lott@nist.gov)
Contract grant sponsor: National Research Council Postdoctoral Fellowship
Contract grant sponsor: U.S. Department of Energy; contract grant number: DEFG0204ER25619
Contract grant sponsor: U.S. National Science Foundation; contract grant number: CCF0726017

This leads to slow convergence of iterative solvers. In total, as convection dominates the flow the discrete fluid model becomes exceedingly challenging to solve.

In recent years, the spectral element method has gained popularity as a technique for numerical simulation of fluids [1,2]. This is due in part to the method's high-order accuracy, which produces solutions with low dissipation and low dispersion with relatively few degrees of freedom. Also important is the inherent computational efficiency gained through the use of a hierarchical grid structure based on unstructured macro-elements with fine tensor-structured interiors. This structure has enabled the development of efficient multi-level solvers and preconditioners based on Fast Diagonalization and Domain Decomposition [3–6]. Application of these techniques, however, has been restricted to symmetric systems.

One way to apply such methods to nonsymmetric systems is through use of time-splitting techniques, which split the system into symmetric and nonsymmetric components. For convection–diffusion systems, the standard method for performing steady and unsteady flow simulations with spectral elements is operator integration factor splitting (OIFS) [7], which requires time integration even in steady flow simulations. Using this standard approach, convection and diffusion are treated separately; convection components are tackled explicitly using a sequence of small time steps that satisfy a CFL (Courant-Friedrichs-Lewy) condition, and diffusive components are treated implicitly with larger time steps via a backward differencing formula that couples the convection terms to the diffusion system. In this framework, the diffusive system is symmetric allowing fast solvers based on Fast Diagonalization and Domain Decomposition to be used. However, to simulate fast moving flows the discretization must be refined to capture sharp flow features accurately; this in turn leads to severe constraints on the time-step size of the semi-implicit method. Such methods can become prohibitively expensive when simulating highly convective flows over long time periods.

An alternative approach to simulating convective flows is to perform implicit time integration, or to solve the steady state system directly. Such methods, however, require fast solvers that are able to resolve the disparate convective and diffusive scales efficiently. The steady convection–diffusion equation for the 2D transport of a scalar $u(x, y)$ can be written as

$$-\epsilon \nabla^2 u + (\vec{w} \cdot \nabla) u = f \text{ in } \Omega \tag{1.1}$$

where the vector field $\vec{w} = (w_x(x, y), w_y(x, y))^T$ represents the given wind speed at each point in the domain Ω, f represents a source term, and ϵ represents the diffusivity. In addition to (1.1), we have the associated boundary conditions

$$u = u_D \text{ on } \partial\Omega_D, \qquad \nabla u_i \cdot \vec{n} = 0 \text{ on } \partial\Omega_N, \tag{1.2}$$

where $\vec{n}$ is the outward facing normal on the boundary, and subscripts D and N denote Dirichlet and Neumann boundary regions, respectively. To measure the relative contributions of convection and diffusion, Eq. (1.1) can be nondimensionalized by emphasizing the inertial terms through characteristic velocity and length scales, W and L, respectively. That is, points in Ω can be normalized by dividing by a characteristic length L, the source term is nondimensionalized by taking $f^* := fL/W^2$, and the scalar u^* and wind w^* are nondimensionalized by dividing by W. This leads to the equation in the normalized domain,

$$-\frac{1}{Pe} \nabla^{*2} u^* + (\vec{w}^* \cdot \nabla^*) u^* = f^*. \tag{1.3}$$

The quantity $Pe := \frac{WL}{\epsilon}$ is termed the Peclet number. This dimensionless number quantifies the contributions of convection and diffusion for a given flow. In diffusion-dominated flows,

$0 \leq Pe \leq O(1)$, whereas in convection-dominated flows $Pe \gg 1$. We see in (1.3) that as $Pe \to \infty$ the flow becomes dominated by convection and the diffusion term vanishes, leading to a hyperbolic system. Solutions to these nearly hyperbolic systems often exhibit sharp gradients in order to satisfy boundary conditions, thus requiring a fine numerical grid to resolve the flow accurately. We also observe that as $Pe \to 0$, (1.3) does not produce the Poisson equation, since the non-dimensionalization emphasizes the inertial terms. A similar non-dimensional form of the convection–diffusion equation can be written for slow moving flows by accentuating the viscous forces. In this form the source term is replaced with $f^* = \frac{fL}{W\epsilon}$, whereas the other quantities are normalized as above, leading to the dimensionless form

$$-\nabla^{*2}u^* + Pe(\vec{w}^* \cdot \nabla^*)u^* = f^*. \tag{1.4}$$

Now, as $Pe \to 0$ one is left with the Poisson equation to model the flow.

In this article, we introduce a new approach for simulating flows with spectral elements by developing efficient solvers for the steady convection–diffusion system (1.1). Similar ideas are explored in [8] for stabilized spectral element discretizations via overlapping domain decomposition. Our method is based on non-overlapping domain decomposition and takes advantage of Fast Diagonalization to eliminate degrees of freedom in element interiors. The efficiency of our solution method is centered on two advancements: the first is the use of an accurate high-order matrix-free discretization to construct accurate discrete solutions while minimizing memory requirements; the second is the use of fast iterative solvers, which are accelerated by domain decomposition based preconditioners that take advantage of the local tensor product structure by exploiting Fast Diagonalization. The matrix-free nature of the algorithm lends itself to parallel computation.

II. SPECTRAL ELEMENT METHOD APPLIED TO THE CONVECTION-DIFFUSION EQUATION

The spectral element method is a numerical method for discretizing differential equations that uses a finite polynomial basis to represent the solution on a set of non-overlapping subdomains. The technique is a Galerkin method derived from the method of weighed residuals, in which a weak form equation is solved. The computational domain is tessellated by subdomains Ω_e called elements, and the associated integrals are divided into a sum of integrals on individual elements which are then approximated by numerical quadrature. In one dimension, an orthogonal nodal spectral basis π^N is constructed in each element as a Lagrange interpolation polynomial based on a set of Gauss-Legendre-Lobatto (GLL) nodes, $(\Xi_{N+1} := \xi_1, \xi_2, \ldots \xi_{N+1})$. In higher dimensions, the basis is formed as a tensor product of these one-dimensional basis functions. This allows functions defined on Ω_e to be written in terms of their spectral basis on each element. Using this nodal basis representation and applying numerical quadrature results in a system of linear matrix equations that numerically represents the original integral equation on each element. Solving this system allows one to obtain the solution at the GLL interpolation points. Interelement coupling of these equations ensures continuity along elemental boundaries. These interelement couplings can be enforced either by constructing a fully coupled sparse linear system of equations, or by performing a gather-scatter operation that sums the solution along element boundaries after element-based matrix-vector products are performed. As described in [9], this gather-scatter operation, called "direct-stiffness-summation," coupled with the tensor product formulation of elemental operators, yields a matrix-free discretization, in which only local matrices associated with one-dimensional phenomena need to be stored.

The weak formulation of (1.1) is: Find $u \in X$ such that:

$$\epsilon \int_\Omega \nabla u \cdot \nabla v + \int_\Omega (\vec{w} \cdot \nabla u) v = \int_\Omega f v \qquad \forall v \in X_0, \tag{2.1}$$

where the continuous solution and test spaces are defined as

$$X := \{u \in H^1(\Omega)^2 | u = u_D \text{ on } \partial\Omega_D\} \tag{2.2}$$

$$X_0 := \{u \in H^1(\Omega)^2 | u = 0 \text{ on } \partial\Omega_D\}, \tag{2.3}$$

which are based on $L^2(\Omega)$ - the space of all square integrable functions on Ω. $H^1(\Omega)$ is the space of all functions in $L^2(\Omega)$ with first derivatives also in $L^2(\Omega)$. Existence and uniqueness of a corresponding weak solution can be demonstrated by establishing the coercivity and continuity (see [10] p. 121) of the bilinear form

$$a(u, v) := \epsilon \int_\Omega \nabla u \cdot \nabla v + \int_\Omega (\vec{w} \cdot \nabla u) v. \tag{2.4}$$

For Dirichlet problems, coercivity can be established by observing that the convection term is skew-self-adjoint when $\nabla \cdot \vec{w} = 0$ and $\vec{w} \cdot \vec{n} = 0$ along the outflow, so $a(u, u) = \epsilon \|\nabla u\|^2$ for all admissible u, and continuity is verified over X_0 using

$$|a(u, v)| \leq \epsilon \Big| \int_\Omega \nabla u \cdot \nabla v \Big| + \Big| \int_\Omega (\vec{w} \cdot \nabla u) v \Big|, \tag{2.5}$$

and bounding the terms using the Cauchy-Schwarz inequality. We refer the reader to [11] for details.

Using the spectral element method, we construct a finite-dimensional basis for X and X_0 by dividing the domain Ω into E non-overlapping sub-domains (elements) $\Omega = \cup_{e=1}^{E} \Omega_e$; each sub-domain is then discretized using tensor products of Lagrangian interpolants on degree N Legendre polynomials π^N. The corresponding discrete approximation space for the solution and test functions is defined as

$$X^N = X \cap \mathbb{P}^2_{N,E}(\Omega), \qquad X_0^N = X_0 \cap \mathbb{P}^2_{N,E}(\Omega) \tag{2.6}$$

where

$$\mathbb{P}^2_{N,E}(\Omega) = \{v(x^e(r))|_{\Omega^e} \in \mathbb{P}_N(r_1) \otimes \mathbb{P}_N(r_2),\ e = 1, \ldots, E\} \tag{2.7}$$

and $\mathbb{P}_N(r)$ is the space of Lagrangian interpolants of degree less than or equal to N defined on the Gauss-Legendre-Lobatto points on the reference domain $[-1, 1]$.

The discrete weak form is Find $u \in X^N$ such that:

$$\epsilon \int_\Omega \nabla u \cdot \nabla v + \int_\Omega (\vec{w} \cdot \nabla u) v = \int_\Omega f v \qquad \forall v \in X_0^N. \tag{2.8}$$

The solution is then written as a sum of a known function u_D, that satisfies the Dirichlet boundary conditions at the nodes corresponding to points on the Dirichlet boundary and is zero otherwise, and an unknown function u_0 that is zero on the Dirichlet boundaries, $u = u_0 + u_D$. Moving the known quantities to the right-hand side we have:

Find $u \in X_0^N$ such that:

$$\epsilon \int_\Omega \nabla u_0 \cdot \nabla v + \int_\Omega (\vec{w} \cdot \nabla u_0) v = \int_\Omega f v - \epsilon \int_\Omega \nabla u_D \cdot \nabla v - \int_\Omega (\vec{w} \cdot \nabla u_D) v \qquad \forall v \in X_0^N. \tag{2.9}$$

On each element the solution is of the form

$$u_0^e(x, y) = \sum_{i=1}^{N+1} \sum_{j=1}^{N+1} u_{ij} \pi_i(x) \pi_j(y), \tag{2.10}$$

and after obtaining u_0^e on each element, this solution can be combined with u_D to form the discrete solution satisfying (1.1) and (1.2). The coefficients u_{ij} correspond to the nodal values of u on the tensored Gauss-Legendre-Lobatto (GLL) points. After inserting u and v represented via the bases indexed by ij and $\hat{i}\hat{j}$, respectively into (2.9) we obtain the system of equations

$$\underbrace{\mathbb{M}^e \Sigma' F^e(\vec{w}^e)}_{F} u_0^e = \underbrace{\mathbb{M}^e \Sigma' (M^e f^e - F^e(\vec{w}) u_D^e)}_{b}. \tag{2.11}$$

Here, a mask matrix $\mathbb{M}^e$ sets u_0^e to zero at all Dirichlet boundary nodes. The convection–diffusion operator F^e is represented on each element with dimensions $h_x \times h_y$ through this basis as

$$\begin{aligned}
F_x^e &= \epsilon \frac{h_y}{h_x} \underbrace{\left(\int_{-1}^{1} \pi_j \pi_{\hat{j}} \right)}_{\hat{M}_{j\hat{j}}} \underbrace{\left(\int_{-1}^{1} \pi_i' \pi_{\hat{i}}' \right)}_{\hat{A}_{i\hat{i}}} + w_x^e \frac{h_y}{2} \underbrace{\left(\int_{-1}^{1} \pi_j \pi_{\hat{j}} \right)}_{\hat{M}_{j\hat{j}}} \underbrace{\left(\int_{-1}^{1} \pi_i' \pi_{\hat{i}} \right)}_{\hat{C}_{i\hat{i}}} \\
&= \underbrace{\epsilon \left(\hat{M} \otimes \frac{h_y}{h_x} \hat{A} \right)}_{\text{Diffusion in x}} + \underbrace{W_x^e \left(\hat{M} \otimes \frac{h_y}{2} \hat{C} \right)}_{\text{Convection in x}} \\
F_y^e &= \epsilon \frac{h_x}{h_y} \underbrace{\left(\int_{-1}^{1} \pi_j' \pi_{\hat{j}}' \right)}_{\hat{A}_{j\hat{j}}} \underbrace{\left(\int_{-1}^{1} \pi_i \pi_{\hat{i}} \right)}_{\hat{M}_{i\hat{i}}} + w_y^e \frac{h_x}{2} \underbrace{\left(\int_{-1}^{1} \pi_j' \pi_{\hat{j}} \right)}_{\hat{C}_{j\hat{j}}} \underbrace{\left(\int_{-1}^{1} \pi_i \pi_{\hat{i}} \right)}_{\hat{M}_{i\hat{i}}} \\
&= \underbrace{\epsilon \left(\frac{h_x}{h_y} \hat{A} \otimes \hat{M} \right)}_{\text{Diffusion in y}} + \underbrace{W_y^e \left(\frac{h_x}{2} \hat{C} \otimes \hat{M} \right)}_{\text{Convection in y}} \\
F^e(w^e) &= F_x^e + F_y^e.
\end{aligned} \tag{2.12}$$

The discrete two-dimensional diffusion operator is formed via tensor products of the one-dimensional second derivative operator $\hat{A}$ with the one-dimensional mass matrix $\hat{M}$. Similarly, the discrete convection operator is formed via a tensor product of the weak one-dimensional derivative operator $\hat{C}$ with the mass matrix $\hat{M}$, then scaled by the wind speed at each node via the $(N+1)^2 \times (N+1)^2$ diagonal matrices $W_x^e = diag(w_x(\xi_i, \xi_j))$ and $W_y^e = diag(w_y(\xi_i, \xi_j))$. The

mass matrix, $M = diag(M_e)$ is composed of local mass matrices, M^e, represented via a tensor product of one-dimensional operators, $\hat{M}$, namely:

$$M^e = \frac{h_x h_y}{4} \underbrace{\left(\int_{-1}^{1} \pi_i \pi_{\hat{i}} \right)}_{\hat{M}_{i\hat{i}}} \underbrace{\left(\int_{-1}^{1} \pi_j \pi_{\hat{j}} \right)}_{\hat{M}_{j\hat{j}}} = \frac{h_x h_y}{4} \hat{M} \otimes \hat{M}. \tag{2.13}$$

Because of the orthogonality of the basis functions π_i and π_j, the mass matrix is a diagonal matrix.

The global system matrices can be defined by assembling the elemental matrices M^e and F^e across multiple elements via a boolean connectivity matrix Q that maps global nodal values to local nodal values; that is $F(w) = \mathbb{M} Q^T F^e Q \mathbb{M}$, is the global nonsymmetric discrete convection–diffusion operator, and $M = \mathbb{M} Q^T M^e Q \mathbb{M}$ the global diagonal mass matrix. However, in practice, the local form given in (2.11) is more convenient computationally because tensor-product operations can be readily applied without a scatter operation first being applied. In this format the direct-stiffness-summation operator $\Sigma' = QQ^T$ is applied to sum all elemental contributions. Thus, (2.11) is a matrix-free formulation in the sense that no global matrix is assembled, allowing the discrete system to be written in terms of element-defined operators such that on each element several discrete operators are applied. We refer the reader to [9] for a detailed description of this formulation.

The system in (2.11) can be solved using an iterative scheme, such as preconditioned GMRES (Generalized Minimal Residual Method) [12]. In the case of constant wind, fast direct solvers in conjunction with domain decomposition with iteration for subsidiary problems are available as a more efficient solver. In the next section, we explain how to solve constant coefficient problems by this technique, and provide empirical results in Section V. Then, in Section VI we discuss how this domain decomposition scheme can be used as a preconditioner to accelerate convergence convection–diffusion problems with non-constant wind.

III. DOMAIN DECOMPOSITION VIA ITERATIVE SUBSTRUCTURING

In this section, we outline the use of a matrix-free domain decomposition method based on iterative substructuring to solve the discrete convection–diffusion problem with constant coefficients. The solution method obtains the discrete solution on a set of nonoverlapping subdomains by first solving for unknowns on inter-element interfaces, and then performing back substitution to compute the interior degrees of freedom. For large systems, the solution on elemental interfaces is obtained by an iterative method. The system that governs the elemental interfaces may be poorly conditioned, and thus requires preconditioning. We describe here a domain decomposition method that we use in conjunction with a Robin–Robin preconditioner for the interface solve. In addition, we present a generalization of the Fast Diagonalization Method for obtaining interior degrees of freedom.

A. Algorithm Overview

As discussed in the previous section, the domain is subdivided into E spectral elements. Let Γ denote the degrees of freedom contained on the union of elemental interfaces, and let the sets I^e denote the degrees of freedom belonging to the interior of the e^{th} element. The discrete system (2.11) can be written in the form

$$\begin{bmatrix} \bar{F}_{II}^1 & 0 & \dots & 0 & \bar{F}_{I\Gamma}^1 \\ 0 & \bar{F}_{II}^2 & 0 & \dots & \bar{F}_{I\Gamma}^2 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \bar{F}_{II}^E & \bar{F}_{I\Gamma}^E \\ \bar{F}_{\Gamma I}^1 & \bar{F}_{\Gamma I}^2 & \dots & \bar{F}_{I\Gamma}^E & \bar{F}_{\Gamma\Gamma} \end{bmatrix} \begin{pmatrix} u_{I^1} \\ u_{I^2} \\ \vdots \\ u_{I^E} \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} b_{I^1} - \bar{F} u_D|_{I^1} \\ b_{I^2} - \bar{F} u_D|_{I^2} \\ \vdots \\ b_{I^E} - \bar{F} u_D|_{I^E} \\ b_\Gamma - \bar{F} u_D|_\Gamma \end{pmatrix} = \begin{pmatrix} \hat{b}_{I^1} \\ \hat{b}_{I^2} \\ \vdots \\ \hat{b}_{I^E} \\ \hat{b}_\Gamma \end{pmatrix}. \tag{3.1}$$

Note that the unknowns here are ordered by block in a specific way: those associated with the interiors of an individual element are listed first, followed in the end by those lying on element interfaces. The term $\bar{F}_{\Gamma\Gamma}$ is the sum of the individual element operators $\bar{F}_{\Gamma\Gamma}^e$ defined on each element. We denote the discrete constant coefficient convection–diffusion operator by $\bar{F}$ because in Section VI we will use this operator as a preconditioner for variable wind convection–diffusion problems.

As we mentioned in the last section, Dirichlet boundary conditions are implemented outside of the system operator in (3.1), by subtracting $\bar{F} u_D$ from the right hand side vector. $\bar{F} u_D$ denotes the full convection–diffusion system matrix applied to the Dirichlet boundary vector u_D.

The goal is to solve for interface values u_Γ and then to perform back substitution and solve for u_{I^e} on each sub-domain interior. The solution algorithm is based on a formal factorization of (3.1) into the product of lower and upper block matrices. The interface variables are obtained by solving

$$\bar{F}_S u_\Gamma = g_\Gamma, \tag{3.2}$$

where

$$\bar{F}_S = \sum_{e=1}^{E} \bar{F}_S^e = \sum_{e=1}^{E} \left(\bar{F}_{\Gamma\Gamma}^e - \bar{F}_{\Gamma I}^e \bar{F}_{II}^{e\,-1} \bar{F}_{I\Gamma}^e \right) \tag{3.3}$$

represents the Schur complement of the system, and the right-hand side is defined as

$$g_\Gamma = \sum_{e=1}^{E} \left(\hat{b}_{\Gamma^e} - \bar{F}_{\Gamma I}^e \bar{F}_{II}^{e\,-1} \hat{b}_{I^e} \right). \tag{3.4}$$

The interior variables are then obtained by solving

$$\bar{F}_{II}^e u_{I^e} = \hat{b}_{I^e} - \bar{F}_{I\Gamma}^e u_\Gamma^e \tag{3.5}$$

on each element. This involves a three-step procedure:

- Perform E subdomain solves to apply the action of $\bar{F}_{II}^e$ giving g_Γ, in (3.4)
- Perform interface solve for $\bar{F}_S u_\Gamma = g_\Gamma$, as in (3.2)
- Perform E subdomain solves to apply the action of $\bar{F}_{II}^e$ yielding u_I, as in (3.5).

The Schur complement operator, $\bar{F}_S$, can be constructed element-by-element by writing $\bar{F}_S = \sum \bar{F}_S^e$. This allows for efficient tensor-product based (see Appendix (A)) computation of the elemental matrix-vector products, which can be used to apply the matrix on each element inside an iterative solver. Once u_Γ is obtained it is substituted into (3.5) to provide elemental boundary conditions for the interior solves. Note that the subdomain solves required for step 1 (to

compute g_Γ) and step 3 (to compute u_I) entail the same operations, a set of independent solves for applying the action of $\bar{F}_{II}^{e\,-1}$ on each element e. Moreover, the same operation is required to apply the elemental matrix-vector products in the iteration used for step 2. Each of these operations is performed using the element-wise Fast Diagonalization Method (FDM), which we discuss in Section C.

B. Interface Solve

To compute the interface variables in (3.2), we use preconditioned GMRES. Our choice of preconditioner is the Robin–Robin preconditioner developed in [13], which extends the popular Neumann-Neumann preconditioner used to accelerate the convergence of interface solvers corresponding to the Poisson equation, [14–16] to nonsymmetric systems. This technique uses a pseudo-inverse of the locally defined Schur complement operator, with Robin and Neumann boundary conditions applied to elemental interfaces. The preconditioning matrix is given by

$$\sum_{e=1}^{E} D^{(e)} R_e^T \left(\bar{F}_S^e\right)^{-1} R_e D^{(e)}, \tag{3.6}$$

where $\bar{F}_S^e$ from Eq. (3.3) is a modified Schur complement operator on the e^{th} element, the matrices D^e are chosen to provide an appropriate inverse scaling factor, and R_e restricts a vector to the e^{th} element.

One may gain insight into the Robin–Robin preconditioner by considering a simple two-domain case. In the case of two subdomains, the Schur complement of Eq. (3.3) is $\bar{F}_S = \bar{F}_S^1 + \bar{F}_S^2$. If the two domains are of equal size, and the wind is constant on the domain such that the matrices $\bar{F}_S^1 \approx \bar{F}_S^2$ are of similar character, then the preconditioned system is approximately a scaled identity

$$\begin{aligned}\left(\bar{F}_S^{1^{-1}} + \bar{F}_S^{2^{-1}}\right)\left(\bar{F}_S^1 + \bar{F}_S^2\right) &= 2I + \bar{F}_S^{2^{-1}}\bar{F}_S^1 + \bar{F}_S^{1^{-1}}\bar{F}_S^2 \\ &= 2I + \bar{F}_S^{2^{-1}}\bar{F}_S^1 + \left(\bar{F}_S^{2^{-1}}\bar{F}_S^1\right)^{-1}.\end{aligned} \tag{3.7}$$

As shown in [15], when applying the preconditioner, the application of the action of each component operator $\bar{F}_S^{e\,-1}$ can be performed by applying the inverse of $\bar{F}^e$ to a vector restricted to the elemental boundary, and then restricting the resulting vector to the elemental boundary. That is,

$$\bar{F}_S^{e\,-1} x_\Gamma = (0 \quad I)\bar{F}^{e\,-1}\begin{pmatrix} 0 \\ x_\Gamma \end{pmatrix}. \tag{3.8}$$

The condition number of the Robin–Robin preconditioned system is bounded by $\frac{C}{H}(1 + log(N))^2$ ([15], p. 321) where N is the order of the spectral element basis functions, H is the diameter of a typical element Ω_e, and the constant C depends on the Peclet number. As more elements are used in the discretization, the $\frac{1}{H}$ dependence dominates this bound, which can cause the number of iterations to increase significantly. Methods to eliminate the $1/H$ dependence include the two-level Balancing Robin–Robin preconditioner [15], BDDC methods (Balancing Domain Decomposition by Constraints) [17–19], and FETI-DP methods (Dual-Primal Finite Element Tearing and Interconnecting) [20, 21]. We will consider the Balancing Robin–Robin preconditioner which is given by a two-step process

$$u^{n+1/2} = \sum_{\Omega_e} D^{(e)} R_e^T \left(\bar{F}_S^e\right)^{-1} R_e D^{(e)} f,$$

$$u^{n+1} = u^{n+1/2} + R_0^T \bar{F}_0^{-1} R_0 \left(f - \bar{F}_S u^{n+1/2}\right). \tag{3.9}$$

Here, the restriction operator R_0 returns the weighted sum of the values on the boundary of a given element, and the coarse grid Schur complement matrix $\bar{F}_0$ is defined as $F_0 = R_0 \bar{F}_S R_0^T$. The first step applies the Robin–Robin preconditioner, and the second step serves as a coarse-grid correction that balances the average of the solution on each subdomain. A standard approach for the coarse grid solve involving $\bar{F}_0$ is direct methods, although this may become expensive for large number of elements [15].

Note that with this methodology, the matrices $\bar{F}^e$ correspond to discretizations of the operator with Neumann boundary conditions applied on elemental boundaries. The Neumann boundary conditions applied at elemental boundaries produces a preconditioning operator that corresponds to the bilinear form

$$a_e(u, v) = \int_{\Omega_e} (\epsilon \nabla u \cdot \nabla v + (\vec{w} \cdot \nabla u) v), \tag{3.10}$$

which can be derived from the element-based Neumann problem

$$-\epsilon \nabla^2 u + (\vec{w} \cdot \nabla) u = f \qquad \text{in } \Omega_e, \tag{3.11}$$

$$-\epsilon \frac{\partial u}{\partial n} = 0 \qquad \text{on } \Gamma_e. \tag{3.12}$$

Using $\bar{F}_S^e$ as a preconditioner corresponds to the Neumann-Neumann domain decomposition method which is commonly used in solving Poisson's equation. Because of the Neumann boundary conditions in the elemental operators, $\bar{F}_{II}^e$ contains a zero eigenvalue and a pseudo-inverse operation can be performed via Fast Diagonalization.

However, as convection becomes dominant, these Neumann element interface conditions (3.12) cause the bilinear form (3.10) to lose coercivity [14], thus rendering the Neumann-Neumann method ineffective. Achdou et al. [13] adapted the Neumann-Neumann preconditioner to non-symmetric convection–diffusion systems by choosing interface boundary conditions that reflect the movement of the flow across element boundaries. In particular, Robin boundary conditions are used instead of Neumann conditions at inflow boundaries where $\vec{w} \cdot n < 0$; Neumann boundary conditions are still imposed at outflows where $\vec{w} \cdot n > 0$. This strategy is known as a Robin–Robin preconditioner. It ensures that the symmetric part of the preconditioning operator is positive-definite (see [13, 16]), because the underlying bilinear form,

$$a_e(u, v) = \int_{\Omega_e} (\epsilon \nabla u \cdot \nabla v + (\vec{w} \cdot \nabla u) v) - \int_{\Gamma_i} (\vec{w} \cdot \vec{n}) u v, \tag{3.13}$$

derived from the element-based Robin problem

$$-\epsilon \nabla^2 u + (\vec{w} \cdot \nabla) u = f \qquad \text{in } \Omega_e, \tag{3.14}$$

$$-\epsilon \frac{\partial u}{\partial n} + (\vec{w} \cdot \vec{n}) u = 0 \qquad \text{on } \Gamma_e, \tag{3.15}$$

is coercive.

The difference between Neumann-Neumann and Robin–Robin preconditioners is the additional $(\vec{w} \cdot \vec{n})u$ term at the element interfaces; thus the two preconditioners are equivalent when $\vec{w} = 0$. This added term allows the flow to move between elements, and ensures that the preconditioned system matrix is positive definite, as described in [13] and [15]. In practice, the elemental boundary conditions are determined by considering the sign of the convection term at each element boundary. This involves modifying the one-dimensional convection–diffusion operators $\hat{F}_x$ and $\hat{F}_y$ from Eq. (2.12) by a single entry corresponding to the inflow condition

$$\begin{aligned}
\hat{F}_{x_{RR}}(1,1) &= \hat{F}_x(1,1) + w_x\hat{M}(1,1) && \text{if } w_x > 0 \\
\hat{F}_{x_{RR}}(N+1,N+1) &= \hat{F}_x(N+1,N+1) - w_x\hat{M}(N+1,N+1) && \text{if } w_x < 0 \\
\hat{F}_{y_{RR}}(1,1) &= \hat{F}_y(1,1) + w_y\hat{M}(1,1) && \text{if } w_y > 0 \\
\hat{F}_{y_{RR}}(N+1,N+1) &= \hat{F}_y(N+1,N+1) - w_y\hat{M}(N+1,N+1) && \text{if } w_y < 0.
\end{aligned}$$

In section V we provide empirical results comparing the Neumann-Neumann and Robin–Robin preconditioning schemes using our solution method. We note that in both cases the operation is performed using Fast Diagonalization, which we introduce in the following subsection.

C. Fast Diagonalization Method (FDM)

Throughout this section, we have seen that the domain decomposition strategy requires interior solves during calculations involving $\bar{F}_S$ and g_Γ as well as in solving (3.5), and computing the pseudo-inverse (3.8) in the application of the interface preconditioner. In this section, we describe an efficient technique known as the Fast Diagonalization Method (FDM) for performing these solves. The Fast Diagonalization Method was originally constructed to solve problems arising from tensor-product based finite difference discretizations of constant coefficient partial differential equations. This method only depends on the inverses of diagonal matrices, and of small matrices corresponding to one-dimensional phenomena. We exploit this in the application of our solvers and preconditioners as discussed in this section. In particular, we show how to generalize the FDM described in [22], to nonsymmetric discrete convection–diffusion systems with constant wind $\vec{w}$. We then examine the use of FDM applied to (2.11) on a single element.

The spectral element discretization enables the convection–diffusion equation to be written as sums of tensor products on each element. This form is particularly useful when performing matrix-vector products, and when solving certain elemental systems of equations. For systems in which the coefficient matrix is of order n^d and has a tensor product structure, where d represents the number of spatial dimensions and n represents the number of grid points used along each dimension on a single element, the FDM enables the solution to be computed in $O(n^{d+1})$ operations.

Consider Eq. (2.12) in the special case where $W_x^e = c_x$ and $W_y^e = c_y$ are both constant on each element. In this special case, the convection–diffusion operator on each element can be written as

$$F^e(c_x, c_y) = \hat{M} \otimes \hat{F}_x + \hat{F}_y \otimes \hat{M} =: \bar{F}^e. \tag{3.16}$$

We use the fact that $\hat{M}$ is diagonal to apply a transformation to $\bar{F}^e$ that will allow for Fast Diagonalization. That is, we can write $\bar{F}^e = M^{1/2}\tilde{F}^e M^{1/2}$ where $M = \hat{M} \otimes \hat{M}$, and

$$\begin{aligned}\tilde{F}^e &= M^{-1/2}\bar{F}^e M^{-1/2} \\ &= (\hat{M}^{-1/2}\otimes\hat{M}^{-1/2})(\hat{M}\otimes\hat{F}_x+\hat{F}_y\otimes\hat{M})(\hat{M}^{-1/2}\otimes\hat{M}^{-1/2}) \\ &= (I\otimes\hat{M}^{-1/2}\hat{F}_x\hat{M}^{-1/2})+(\hat{M}^{-1/2}\hat{F}_y\hat{M}^{-1/2}\otimes I) \\ &= (I\otimes B)+(A\otimes I). \end{aligned} \tag{3.17}$$

Assuming both A and B are diagonalizable, we have $A = S\Lambda_y S^{-1}$, $B = T\Lambda_x T^{-1}$. This gives

$$\tilde{F}^e = (S\otimes T)(\Lambda_y\otimes I + I\otimes\Lambda_x)(S^{-1}\otimes T^{-1}) \tag{3.18}$$

so that

$$\tilde{F^e}^{-1} = (S\otimes T)(\Lambda_y\otimes I + I\otimes\Lambda_x)^{-1}(S^{-1}\otimes T^{-1}). \tag{3.19}$$

That is, the transformed matrix $\tilde{F}$ can be diagonalized cheaply and the action of the inverse of $\bar{F}^e$ can also be inexpensively applied as

$$\bar{F^e}^{-1} = (\hat{M}^{-1/2}\otimes\hat{M}^{-1/2})(S\otimes T)(\Lambda_y\otimes I + I\otimes\Lambda_x)^{-1}(S^{-1}\otimes T^{-1})(\hat{M}^{-1/2}\otimes\hat{M}^{-1/2}). \tag{3.20}$$

We use this method to apply the action of $\bar{F}_{II}^{e\,-1}$ for each element in the domain decomposition method described earlier in this section. In flows where the wind coefficient $\vec{w}$ is constant, the resulting algorithm defined by (3.2)–(3.5) can be viewed as a direct solver for Eq. (2.11), although it should be noted that in practice, an iteration is needed to obtain the solution to the Schur complement system (3.2) on the union of element interfaces. We demonstrate the use of this methodology for solving constant coefficient problems in two examples in Section V. We then describe in Section VI how this method can be used to accelerate convergence of GMRES for computing solutions of more general flows. Included in this discussion is the impact and convergence properties of the inner iteration for the Schur complement system (3.2).

IV. STABILIZATION CONSIDERATIONS

We have constructed the solver in the previous section using a Galerkin discretization of the steady convection–diffusion equation. However, as the diffusion coefficient ϵ becomes small, the solution often contains boundary layers or internal layers [23, section III.1.3] that pose strict, and often prohibitive resolution requirements for small ϵ. Under-resolved grids cause spurious oscillations in the solution in regions near these sharp layers which intrude into the rest of the computational domain, spoiling the solution globally. Thus adaptive mesh refinement is required to accurately resolve flows in high Peclet number regimes. However, stabilization techniques such as streamline diffusion (SD) or Galerkin Least Squares methods [14] can be used to provide an accurate solution away from sharp layers.

These methods do not yield discrete operators with a tensor-product structure where Fast Diagonalization can be directly applied. For example, SD discretization entails finding $u \in X^N$ such that:

$$\epsilon \int_\Omega \nabla u \cdot \nabla v + \int_\Omega (\vec{w} \cdot \nabla u) v + \delta \int_\Omega (\vec{w} \cdot \nabla u)(\vec{w} \cdot \nabla v) - \delta\epsilon \sum_{e=1}^{E} \int_\Omega (\nabla^2 u)(\vec{w} \cdot \nabla v) =$$

$$\int_\Omega f v + \delta \int_\Omega f \vec{w} \cdot \nabla v \qquad \forall v \in X_0^N. \tag{4.1}$$

for some stabilization parameter δ depending on the mesh size, [24], and perhaps the norm of the wind speed [10, 25]. The two new terms

$$\delta \int_\Omega (\vec{w} \cdot \nabla u)(\vec{w} \cdot \nabla v) - \delta\epsilon \sum_{e=1}^{E} \int_\Omega (\nabla^2 u)(\vec{w} \cdot \nabla v) \tag{4.2}$$

appearing in (4.1) do not lend themselves to fast diagonalization. In particular, the (dominant) first term leads to the expression

$$\delta \left(\int_{-1}^{1} w_i^2 \pi_i' \pi_{\hat{i}}' \int_{-1}^{1} \pi_j \pi_{\hat{j}} + \underline{\int_{-1}^{1} w_i \pi_i' \pi_{\hat{i}} \int_{-1}^{1} w_j \pi_j \pi_{\hat{j}}' + \int_{-1}^{1} w_i \pi_i \pi_{\hat{i}}' \int_{-1}^{1} w_j \pi_j' \pi_{\hat{j}}} \right.$$

$$\left. + \int_{-1}^{1} \pi_i \pi_{\hat{i}} \int_{-1}^{1} w_j^2 \pi_j' \pi_{\hat{j}}' \right). \tag{4.3}$$

The underlined terms do not contain the diagonal mass matrix which is required to perform fast diagonalization. We note however that the first and last terms of (4.3) correspond to weighted diffusion in the streamwise direction.[1] These terms have tensor-product form that could be incorporated into a preconditioner for (4.1) that can be handled in the manner described in Section (IIIC).

V. SOLUTION OF PROBLEMS WITH CONSTANT WIND

In this section, we demonstrate the effectiveness of our solvers applied to two test problems with constant coefficients. We consider only the nonstabilized discretizations that resolve the flow for $Pe \leq 5000$.

A. Closed-Form Solution With Outflow Boundary Layer

We consider a problem with a closed-form solution; this problem has a zero source term and a constant wind in the vertical direction. The solution is

$$u(x, y) = x \left(\frac{1 - e^{(y-1)/\epsilon}}{1 - e^{-2/\epsilon}} \right). \tag{5.1}$$

Plots of the computed solution and its contours are displayed in Fig. 1. These were obtained using a spectral element discretization with 2 elements in each dimension ($E = 4$), and polynomial degree $N = 16$ on each element with a Peclet number, $Pe = 40$. The solution exhibits dramatic change near the outflow boundary $y = 1$; the width of this exponential boundary layer is

[1] Indeed, these are the only terms that would appear in a *finite difference* version of the SD operator.

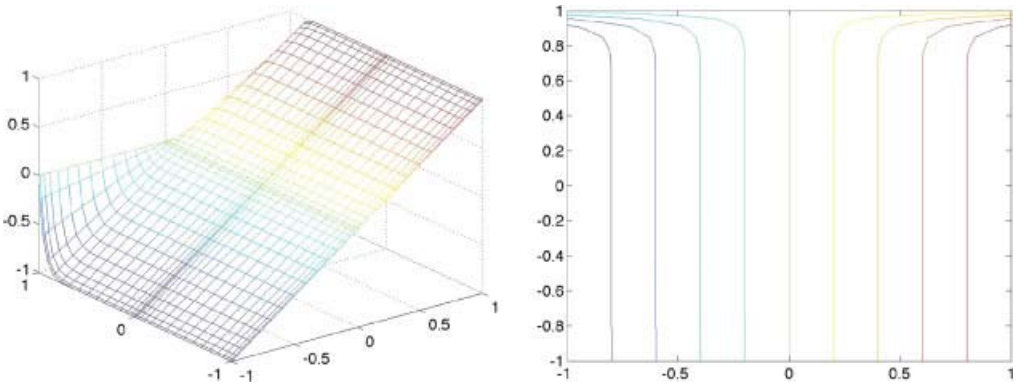

FIG. 1. Computed solution (left) and contours (right) of steady convection diffusion flow corresponding to example VA with constant wind $\vec{w} = (0, 1)$ and moderate convection $Pe = 40$, $N = 16$, and $E = 4$. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

proportional to ϵ (see [23,26]), so as the Peclet number increases the width of this boundary layer narrows.

We test the accuracy of our numerical scheme by comparing with the analytic solution and record the results in Tables I and II. In the first table, a 2×2 element grid is fixed as the polynomial degree on each element ranges from 4 to 32. We see that as the polynomial degree is doubled, the error decays exponentially, as expected from p-refinement. In the second table the polynomial degree on each element is fixed at $N = 2$ as the number of elements range from 16 to 1024. Here, we see that the solution converges algebraically as E is increased. These results are in agreement with the cubic convergence rate for quadratic elements [27].

B. Oblique Wind With Internal and Outflow Boundary Layers

In our second example, we use a flow that exhibits two layers, one characteristic boundary layer along the top wall proportional to ϵ and a second internal layer of width proportional to $\sqrt{\epsilon}$ that results from a jump discontinuity in the boundary at $(0, -1)$ (See Fig. 2 for the case $Pe = 250$, $N = 8$, and $E = 256$). Dirichlet boundary conditions are imposed along each boundary. The wind field in this test case is constant $\vec{w} = (-sin(\pi/6), cos(\pi/6))$, but unlike the previous example, here the wind is not aligned with the grid.

C. Constant Wind Results

We now focus on the iterative solution of the interface problem (3.2). We examine the influence of changes in the discretization and the Peclet number on the iterations by performing three sets

TABLE I. Exponential convergence for example VA with moderate convection ($Pe = 40$), as polynomial degree is varied on a fixed 2×2 element grid.

N	$\|u - u_N\|_2$
4	5.535×10^{-2}
8	2.505×10^{-3}
16	2.423×10^{-7}
32	7.931×10^{-13}

TABLE II. Algebraic convergence for example VA with moderate convection ($Pe = 40$), as the number of quadratic elements are varied.

E	$\|u - u_N\|_2$	$13H^3$
16	8.594×10^{-2}	2.031×10^{-1}
64	2.593×10^{-2}	2.523×10^{-2}
256	3.558×10^{-3}	3.174×10^{-3}
1024	3.610×10^{-4}	3.967×10^{-4}

of experiments for each example problem. First, we modify the number of elements in the discretization for a fixed polynomial basis and Peclet number. Next, we refine the polynomial degree while keeping the number of elements and the Peclet number fixed. Finally, we fix the number of elements and the polynomial basis, while modifying the Peclet number.

GMRES is used to perform the interface solve. In each experiment, we test four choices of preconditioner for the interface solve: no preconditioner (None), Neumann-Neumann preconditioning (N-N), Robin–Robin preconditioning (R-R), and Balancing Robin–Robin preconditioning (BR-R). The results are summarized in Tables III–V. There we record the number of iterations needed for GMRES to converge within a tolerance of 10^{-12}.

For the first experiment, we set the Peclet number to 40 and fix the polynomial degree at $N = 2$. We then increase the number of elements from $E = 16$ (as a 4×4 grid) to $E = 1024$ (32×32 grid). We record the iterations in Table III. Each method depends on the increased grid resolution. The table shows that performance of the Robin–Robin method compares well with theory and is superior to the other single-level methods. We see that the number of (R-R) iterations for both test cases are essentially of the form $\frac{C}{H}(1 + \log(N))^2$, for $C \approx 1$, where h is the diameter of a single element. As expected, the number of iterations using the the Balancing Robin–Robin method is not significantly affected by changes in E.

In our next experiment, we again set the Peclet number to 40, but now we fix the number of elements to 4 as a 2×2 element grid, and view the influence of the polynomial degree on the number of interface solve iterations. The iterations are recorded in Table IV. We note that for this test problem with moderate convection and few subdomains, (N-N), (R-R) and (BR-R) converge

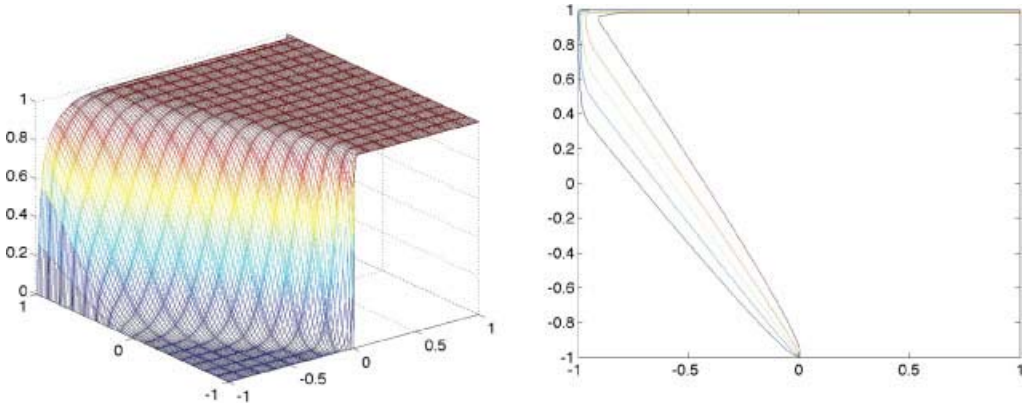

FIG. 2. Computed solution (left) and contours (right) of a convection-dominated steady convection–diffusion flow, $Pe = 250$, $N = 8$, and $E = 256$, corresponding to example VB. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

TABLE III. Convergence results for examples VA (top) and VB (bottom) with moderate convection Pe = 40, as the number of quadratic (N = 2) elements are varied.

	E	None	N-N	R-R	BR-R	$\frac{1}{H}(1+\log(N))^2$
Example VA	16	13	13	12	11	11.5
	64	49	47	25	15	22.9
	256	108	88	45	19	45.9
	1024	312	180	85	20	91.7
Example VB	16	29	33	21	11	11.5
	64	40	63	26	13	22.9
	256	69	117	46	15	45.9
	1024	132	>200	87	15	91.7

in roughly the same number of steps, and here the iterations in both test cases are approximately $\frac{C}{H}(1+\log(N))^2$, for $C \approx 1$.

Finally, we examine how changes in the Peclet number affect convergence. For this study, we compare the number of iterations needed to obtain the interface solution as the Peclet number is increased. We use a fixed grid with polynomial degree N=8, and the number of elements at $E = 1024$ (32×32 element grid). In Table V, we compare convergence results when different preconditioners are used to obtain the interface solution. The Neumann-Neumann preconditioner (N-N) is no more effective than when no preconditioning is used (None), whereas the Robin–Robin preconditioner (R-R) shows little dependence on changes in the Peclet number. Indeed, for moderate values, as the Peclet number is increased the Robin–Robin preconditioner actually improves. For higher Peclet numbers, the iteration counts with Robin–Robin preconditioning increase only mildly with the Peclet number.

We conclude that Robin–Robin preconditioner performs well for both grid-aligned and nongrid aligned flows. In cases where the Peclet number is small, this technique behaves like the Neumann-Neumann preconditioner, which is often used to precondition the discrete Poisson equation [15]. We observed that when using the Robin–Robin preconditioner, the number of iterations required for $\tilde{F}_S^{-1}$ had little dependence on the mesh size as well as the Peclet number.

As a final note for this section, we point out that for constant-wind problems this solution strategy based on iterative substructuring is significantly cheaper than naively applying GMRES directly to (2.11). The domain decomposition strategy allows one to solve highly convective systems, where in contrast without preconditioning, we have found that GMRES fails to converge. This is accomplished, in part, by eliminating interior degrees of freedom using FDM, thereby reducing memory overhead in high resolution cases by nearly an order of magnitude. For example, in the simulation above using $N = 8$ and $E = 32 \times 32$, there are roughly five times fewer

TABLE IV. Convergence results for examples VA (top) and VB (bottom) with a moderate convection $Pe = 40$, as polynomial degree is varied on a fixed 2×2 element grid.

	N	None	N-N	R-R	BR-R	$\frac{1}{H}(1+\log(N))^2$
Example VA	4	3	3	3	3	5.6
	8	7	7	7	7	9.4
	16	15	11	14	18	14.2
	32	30	16	18	19	19.9
Example VB	4	13	13	13	9	5.6
	8	25	25	18	19	9.4
	16	36	28	20	21	14.2
	32	50	29	21	23	19.9

TABLE V. Comparison of iteration counts for examples VA (top) and VB (bottom) with increasingly convection-dominated flows. N = 8, E = 1024 using 32 × 32 element grid.

	Pe	None	N-N	R-R
Example VA	125	161	165	64
	250	126	144	52
	500	107	147	46
	1000	109	164	43
	2000	135	>200	42
	5000	>200	>200	50
Example VB	125	184	186	70
	250	140	158	61
	500	107	148	52
	1000	88	166	46
	2000	96	>200	52
	5000	>200	>200	70

degrees of freedom along elemental interfaces than in the complete system. In three dimensions with large N, savings in memory would be even greater. For nonsymmetric systems solved using GMRES this is very important because one must hold all computed iterates in memory to obtain an orthogonal search direction at each iterate. In the next section, we attempt to take advantage of this inexpensive computational method as a preconditioning technique for solving nonconstant wind convection–diffusion systems.

VI. NONCONSTANT WIND SYSTEMS

When the convection coefficient $\vec{w}$ is not constant in each component, the domain decomposition solution technique described in the previous two sections does not directly apply. In this section, we present a way to use our domain decomposition solver as a preconditioner to accelerate the convergence of GMRES or Flexible GMRES (FGMRES) for solving discrete convection–diffusion systems (2.11) arising from nonconstant winds.

In the case of non-constant wind, the matrix $F(\vec{w})$ from Eq. (2.12) cannot be written in the tensor product form (3.16) where Fast Diagonalization can be applied element-wise. However, if we approximate $F(\vec{w})$ on each element in a certain way, we can construct an element-based matrix of this form to be used as a preconditioner for $F(\vec{w})$. In particular, we consider the approximation to $F(\vec{w})$ where $\vec{w}$ is approximated locally as a constant on each element. To determine this constant, we take the average of $\vec{w}$ in each component to construct a new piecewise constant wind vector approximation $\bar{w}$. Using this vector, we can construct a constant-wind approximation to $F(\vec{w})$, which we call $\bar{F}$.

To apply the preconditioner $\bar{F}$ for multiple elements, we follow the domain decomposition strategy developed in Section III. The only difference is that $\bar{F}$ is now an approximation to F based on average elemental wind speeds. Because of this, we note that it is not necessary to resolve the $\bar{F}_S$ interface solution to high precision, since these values are not likely to reflect the values of the nonconstant wind solution accurately. Instead, we seek a rough estimate of the interface values that we can obtain from a few iterations of the Schur complement solve for (3.2). That is, we apply $\bar{F}_S^{-1}$ inexactly through a few steps of GMRES. It is also important to point out that the iteration used to solve the Schur complement problem in $\bar{F}_S$ makes $\bar{F}$ a nonlinear operator. Because the preconditioning operator is non-linear, the outer GMRES iteration must be replaced with a variant such as Flexible GMRES [12], which allows for arbitrary

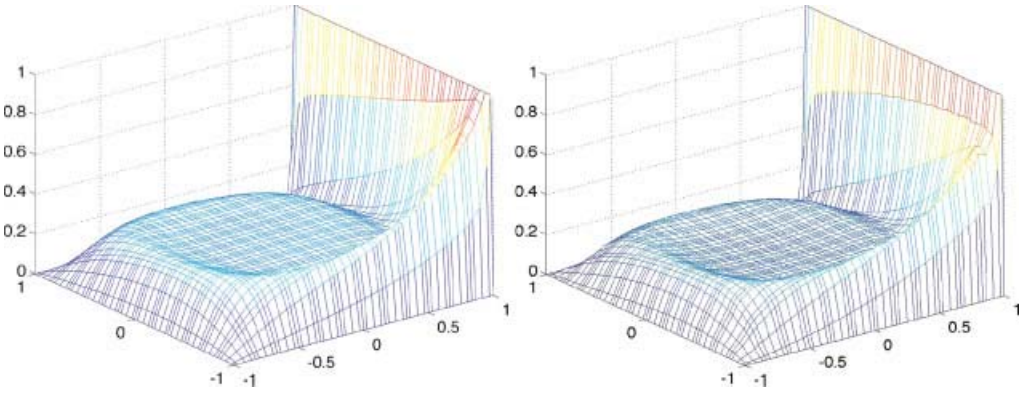

FIG. 3. Comparison of exact solution (left) and inexact solution (right) obtained by applying $\bar{F}^{-1}$ as an inexact solver for example VIA with $Pe = 400$, $N = 4$, and $E = 144$. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

changes in the preconditioner at each step of the iteration. Thus, FGMRES is used to solve the right-preconditioned system

$$[\underbrace{\mathbb{M}^e \Sigma' F^e(\vec{w})}_{F} \bar{F}^{-1}][\bar{F} u_0^e] = \underbrace{\mathbb{M}^e \Sigma' (M^e f^e - F^e(\vec{w}) u_D^e)}_{b}. \tag{6.1}$$

In summary, to solve convection–diffusion problems with nonconstant wind we use Flexible GMRES. At each step of FGMRES, a preconditioner, $\bar{F}$, based on an element-wise average wind is used. In addition, during each FGMRES iteration we allow for the possibility of using an inexact inner iteration for the interface nodes associated with (3.2). This inner iteration is coupled with FDM to obtain interior degrees of freedom on each element.

A. Double Glazing Problem

We examine our nonconstant wind solver by applying it to a flow that has a recirculating wind $\vec{w} = (2y(1 - x^2), -2x(1 - y^2))$ and discontinuities in parts of the boundaries, which lead to boundary layers. This example is known as the *double-glazing problem*, and serves as a simple model of the spread of heat in a box with a hot wall.

The left pane of Fig. 3 shows the computed solution for an example flow with $Pe = 400$, $N = 4$, and $E = 144$ using FGMRES, and the right pane shows an approximate solution obtained by replacing the coefficient matrix with the preconditioning matrix $\bar{F}$. It can be seen in the figure that in the approximate solution the constant wind approximation produces an error in regions where the convection field changes direction, such as at the corners and around the internal boundary layer.

B. Variable Wind With Curved Open Streamlines

For our second example, we use a test case from [8] with a variable flow field $\vec{w} = \frac{1}{2}((1 - x^2)(1 + y), x((1 + y)^2 - 4))$. Boundary conditions are zero except along the inflow region $(x, -1) : -1 < x < 0$, where $u = 1$. The left pane of Fig. 4 shows the computed solution for an example flow with $Pe = 125$, $N = 8$, and $E = 256$, the right pane shows a depiction of the wind field.

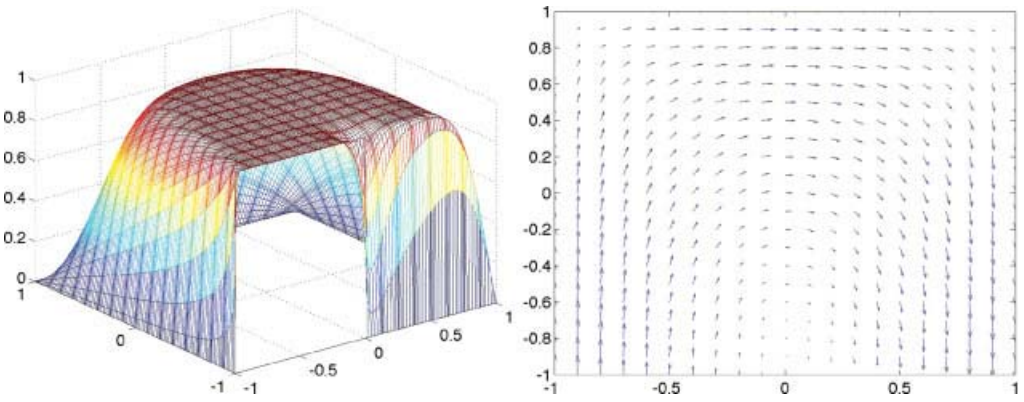

FIG. 4. Computed solution (left) and wind field (right) of flow corresponding to example VIB with $Pe = 125$, $N = 8$, and $E = 256$. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

C. Variable Wind Results

We compare the use of $\bar{F}$ as a preconditioner for FGMRES with unpreconditioned GMRES and block-Jacobi preconditioned GMRES. To construct the block Jacobi preconditioner, we use the block diagonal of $\bar{F}$. So on each element the interior nodes are obtained by solving a system using $\bar{F}^e_{II}$ via FDM, and the boundary nodes are obtained by solving a system with $\bar{F}_{BB}$ via GMRES.

Figure 5 shows the number of FGMRES iterations required to solve (6.1) to a tolerance of 10^{-12} for example VIA. The top two curves show that unpreconditioned GMRES and block Jacobi preconditioned GMRES are ineffective at solving this system. Next, we study the effect of inexactly applying F_S^{-1} using an inner iteration with unpreconditioned GMRES. The latter four curves show the results of using various approximations of $\bar{F}$ to precondition the system. These

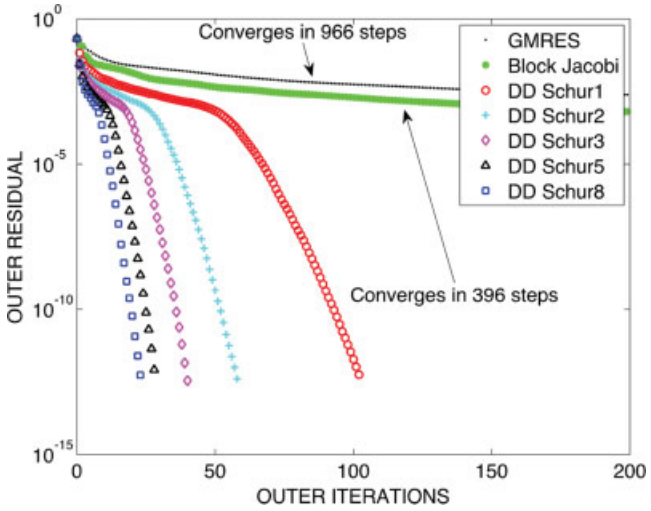

FIG. 5. Comparison of preconditioned Flexible GMRES iterations with varied inner iteration counts, Block Jacobi iterations and unpreconditioned GMRES iterations for solving Eq. (2.11) with $Pe = 400$. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

TABLE VI. Timings corresponding to the experiments in Fig. 5.

Preconditioner	Time (s)
None	53
Block Jacobi	67
DD Schur1	32
DD Schur3	20
DD Schur5	17
DD Schur8	19

curves show the influence of accuracy of the Schur complement solve $\bar{F}_S^{-1}$ on the performance of the outer FGMRES iteration. The ∘ curve shows that applying $\bar{F}_S^{-1}$ with 1 step of GMRES to obtain an approximate solution of the interface allows the outer FGMRES to converge in 102 steps. In the opposite extreme, the □ curve shows that computing an inexact interface solution u_Γ by performing 8 GMRES iterations in the inner iteration allows the outer iterations via FGMRES to converge in 23 steps. It is evident from Fig. 5 that only a few inner iterations for the preconditioner are required in order for the outer iteration to converge quickly. We provide a comparison of timings for this experiment in Table VI. Note that in this experiment, the fastest run is performed when using 5 GMRES inner iterations, even though this run takes a few more outer iterations than using 8 GMRES inner iterations.

Note that the results described in the previous paragraph come from using unpreconditioned GMRES for the Schur complement problem. We saw in the previous section that Robin–Robin preconditioning improves the performance for constant coefficient problems. However, here our conclusions are somewhat different. In Fig. 5 we observed that using a few steps of unpreconditioned GMRES for the inner iteration allowed the outer iteration to converge rapidly. The left pane of Fig. 6 shows the residual during the first 40 steps of the interface solve using GMRES without preconditioning (top) and GMRES with Robin–Robin preconditioning. We see that both solvers reduce the residual by nearly one order of magnitude in the first 10 steps before convergence slows significantly. The Robin–Robin preconditioned solver further reduces the residual around the 25th iteration, while the non-preconditioned system shows little improvement. Achdou et al. point out

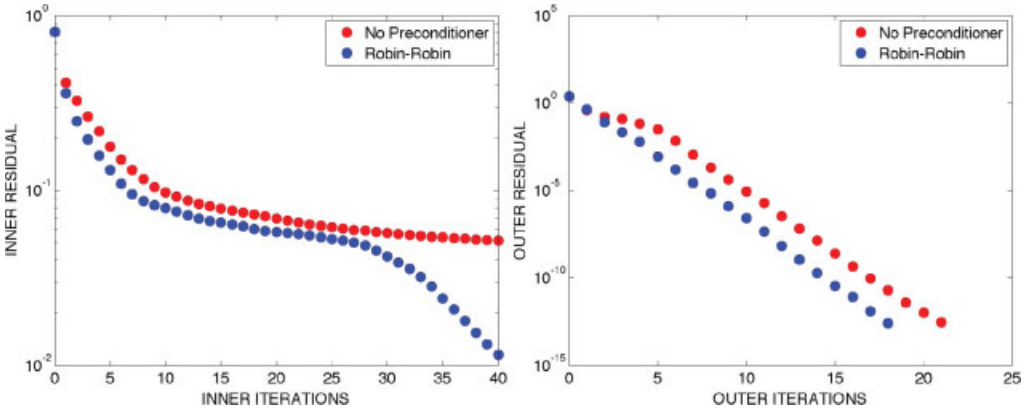

FIG. 6. Comparison of relative residuals for interface iterations obtained by GMRES without preconditioning and with Robin–Robin preconditioning (left) for example VIA. Effect on FGMRES residuals with inexact $\bar{F}^{-1}$ using no interface preconditioner and Robin–Robin (right). [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

TABLE VII. Iteration counts for examples VIA (top) and VIB (bottom) as polynomial degree is increased with $Pe = 400$ and $E = 16$.

	N	Number of FGMRES Outer Iterations	Number of Inner Iterations
Example VIA	4	40	5
	8	51	5
	16	44	13
	32	48	20
Example VIB	4	34	7
	8	35	8
	16	34	18
	32	34	20

in [12] that in convection-dominated flows, the continuous Robin–Robin preconditioned system operator is close to an idempotent (or periodic) operator of order $E/2$ where E is the number of elements. They argue that this causes GMRES to stagnate for $E/2$ steps before converging asymptotically.[2] However, the right pane of Fig. 6 shows that the residual of the outer FGMRES iteration is essentially unaffected by the difference in these two inner iteration residuals. The top curve represents the residual of the outer FGMRES iteration using unpreconditioned GMRES for the inner iteration. In comparison, the bottom curve represents the residual of the outer FGMRES iteration when the Robin–Robin preconditioner is applied to the inner GMRES iteration. Although the Robin–Robin system (bottom curve) converges slightly faster (by three iterations), it requires an extra matrix-vector product at each inner iteration, thus in this example it is roughly 40 times more expensive to use Robin–Robin preconditioning instead of using no preconditioner to obtain an inexact interface solution.

Next, we look at how the number of FGMRES iterations are influenced by mesh refinement. In Tables VII and VIII, we show the dependence of FGMRES iterations as the mesh is refined using both h-refinement and p-refinement. We use a stopping tolerance for FGMRES set at 10^{-12}, and for the $\bar{F}_S^{-1}$ interface solve, we stop the iterations when the residual is less than 10^{-1} or when a maximum of 20 steps are reached. We see that outer iterations for p-refinement (Table VII, middle column) are roughly constant, and the number of outer iterations decreases as E increases, except for $E = 1024$. A reduction in outer iterations is expected since the constant wind approximation used in the preconditioner improves as E increases. However, as seen in the right column of Tables VII and VIII, the number of inner iterations increases as the number of points on the interface increases. The anomaly for $E = 1024$ is the result of taking a maximum of 20 inner iterations without reducing the residual on the interface enough. In Table IX, we show that using a balancing Robin–Robin preconditioner for the interface solve keeps the number of inner iterations nearly constant and leads to a consistent reduction in the number of outer iterations for all E.

Finally, we consider the dependence of the Peclet number on this solution method. In Table X, we show how the number of FGMRES iterations are affected as the Peclet number is increased from 125 up to 5000. In this study, we use a grid where $N = 8$ and $E = 1024$ on a 32×32 element grid. We use the same stopping criteria as in the mesh refinement study above. We see that outer iterations (middle column) are mildly dependent on the Peclet number whereas the inner iterations (right columns) reach the maximum number of iterations (20) for each Peclet number.

[2]Elman and Chernesky [28] report a similar convergence delay when applying block Gauss-Seidel line relaxation to node orderings that do not follow the flow in 1D convection–diffusion simulations.

TABLE VIII. Iteration counts for examples VIA (top) and VIB (bottom) as the number of elements are increased with $Pe = 400$ and $N = 4$.

	E	Number of FGMRES Outer Iterations	Number of Inner Iterations
Example VIA	16	40	5
	64	25	12
	256	17	19
	1024	28	20
Example VIB	16	34	7
	64	18	8
	256	11	20
	1024	16	20

VII. SUMMARY

We have introduced two solution strategies for convection–diffusion systems. The first strategy applies to problems with constant wind coefficients. This method uses an extension to the Fast Diagonalization Method that we developed to solve convection–diffusion problems with constant wind coefficients on single domains. We coupled this result with Robin–Robin preconditioned Domain Decomposition to develop a matrix-free solution method for tensor-product based discretizations of the steady convection-diffusion equation with constant wind on each element.

This method uses iterative substructuring to resolve elemental interface values, together with Fast Diagonalization to eliminate interior degrees of freedom on each element via a direct solve. We demonstrated that this solution method has a weak dependence on Peclet number, and mild dependence on mesh refinement for both h-refinement and p-refinement. Similar conclusions have been reported in [8, 13, 29] for overlapping and nonoverlapping domain decomposition strategies. The use of FDM in our method significantly reduces the computational cost of the subsidiary interior solves throughout the calculation.

We then developed a solver for variable winds by demonstrating how the domain decomposition method we developed for constant-winds can be used as a preconditioner for general convection–diffusion systems when combined with Flexible GMRES. This variable wind solution strategy showed significant improvement over non-preconditioned GMRES and Block-Jacobi preconditioning techniques. We showed that only an inexact interface solve is needed to significantly accelerate convergence of the outer iteration. Using $\bar{F}$ as a preconditioner allowed FGMRES

TABLE IX. Iteration counts for examples VIA (top) and VIB (bottom), using Balancing Robin–Robin Preconditioner in inner iteration as the number of elements are increased with $Pe = 400$ and $N = 4$.

	E	Number of FGMRES Outer Iterations	Number of BR-R Inexact Inner Iterations
Example VIA	16	40	5
	64	25	3
	256	17	3
	1024	12	4
	4096	10	4
Example VIB	16	34	6
	64	18	4
	256	10	4
	1024	7	6
	4096	6	8

TABLE X. Iteration counts for examples VIA (top) and VIB (bottom) as the Peclet number is increased on a fixed grid with $N = 8$ and $E = 1024$.

	Pe	Number of FGMRES Outer Iterations	Number of Inner Iterations
Example VIA	125	27	20
	250	28	20
	500	30	20
	1000	32	20
	2000	37	20
	5000	48	20
Example VIB	125	19	20
	250	16	20
	500	16	20
	1000	16	20
	2000	17	20
	5000	21	20

to obtain convergence rates independent of the mesh size, and mildly dependent of changes in convection strength. Our results compare favorably with those reported in [8, 10, 13, 29].

VIII. APPENDIX

A. Tensor-Based Nodal Ordering for Domain Decomposition

In Section IIIC we explained how the tensor product basis of the spectral element method allows for efficient matrix-vector products and Fast Diagonalization. In Domain Decomposition Methods, however, it is common to use node orderings that enumerate interior degrees of freedom and then boundary degrees of freedom. In this section, we provide the one-dimensional building blocks needed to formulate the two-dimensional operators in terms of their interior and boundary couplings within a lexigraphically ordered tensor product framework. We let N be the degree of the polynomial basis for a given discretization.

We write $\hat{F}_{(N+1)\times(N+1)}$ as the full 1D convection–diffusion (or diffusion) matrix, and $\hat{M}_{(N+1)\times(N+1)}$ as the diagonal 1D mass matrix. $F_{(N+1)^2\times(N+1)^2} = \hat{F}\otimes\hat{M} + \hat{M}\otimes\hat{F}$ is the sparse 2D convection–diffusion matrix on a single element. We can decompose $\hat{F}$ and $\hat{M}$ into their interior and boundary couplings.

$$
\begin{aligned}
&\hat{F}_{ii} = \hat{F}(2:N, 2:N) \text{ Interior-Interior}\\
&\hat{F}_{ib} = \hat{F}(2:N, 1:N+1) \text{ Interior-Boundary}\\
&\hat{F}_{bi} = \hat{F}(1:N+1, 2:N) \text{ Boundary-Interior}\\
&\hat{F}_{bb} = \hat{F}(1,1) + \hat{F}(1,N) + \hat{F}(N,1) + \hat{F}(N,N) \text{ Boundary-Boundary}\\
&\hat{M}_{ii} = \hat{M}(2:N, 2:N) \text{ Interior-Interior}\\
&\hat{M}_{bb} = \hat{M}(1:1, \vec{0}, N+1:N+1) \text{ Boundary-Boundary}
\end{aligned}
$$

This decomposition allows F to be written as $F = F_{II} + F_{\Gamma\Gamma} + F_{I\Gamma} + F_{\Gamma I}$ with

$$
\begin{aligned}
F_{II} &= \hat{F}_{ii}\otimes\hat{M}_{ii} + \hat{M}_{ii}\otimes\hat{F}_{ii}\\
F_{\Gamma\Gamma} &= \hat{F}\otimes\hat{M}_{bb} + \hat{M}_{bb}\otimes\hat{F} + \hat{F}_{bb}\otimes\hat{M}_{ii} + \hat{M}_{ii}\otimes\hat{F}_{bb}\\
F_{I\Gamma} &= \hat{F}_{ib}\otimes\hat{M}_{ii} + \hat{M}_{ii}\otimes\hat{F}_{ib}\\
F_{\Gamma I} &= \hat{F}_{bi}\otimes\hat{M}_{ii} + \hat{M}_{ii}\otimes\hat{F}_{bi}.
\end{aligned}
$$

References

1. P. F. Fischer, F. Loth, S. E. Lee, S. W. Lee, D. S. Smith, and H. S. Bassiouny, Simulation of high Reynolds number vascular flows, Comput Methods Appl Mech Eng 196 (2007), 3049–3060.
2. S. J. Thomas and R. D. Loft, The NCAR spectral element climate dynamical core: semi-implicit Eulerian formulation, J Sci Comput 25 (2005), 307–322.
3. W. Couzy and M. O. Deville, A fast Schur complement method for the spectral element discretization of the incompressible Navier-Stokes equations, J Comput Phys 116 (1995), 135–142.
4. P. Fischer, An overlapping Schwarz method for spectral element solution of the incompressible Navier-Stokes equations, J Comput Phys 133 (1997), 84–101.
5. J. Lottes and P. Fischer, Hybrid multigrid/Schwarz algorithms for the spectral element method, J Sci Comput 24 (2005), 613–646.
6. H. Tufo and P. Fischer, Terascale spectral element algorithms and implementations, Supercomputing '99: Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM), ACM press, New York, 1999, p. 68.
7. Y. Maday, A. Patera, and E. M. Rønquist, An operator-integration-factor splitting method for time dependent problems: Application to incompressible fluid flow, J Sci Comput 5 (1990), 263–292.
8. L. Pavarino, Overlapping Schwarz preconditioners for spectral element discretizations of convection-diffusion problems schwarz preconditioners for spectral element discretizations of convection-diffusion problems, Int J Numer Methods Eng 53 (2002), 1005–1023.
9. M. O. Deville, P. F. Fischer, and E. H. Mund, High-order methods for incompressible fluid flows, Cambridge monographs on applied and computational mathematics, Cambridge University Press, Cambridge, 2002.
10. H. C. Elman, D. Silvester, and A. Wathen, Finite elements and fast iterative solvers with applications in incompressible fluid dynamics, Numerical Mathematics and Scientific Computation, Oxford University Press, New York, 2005.
11. S. C. Brenner and L. R. Scott, The mathematical theory of finite element methods, Springer-Verlag, New York, 1994.
12. Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, SIAM J Sci Comput 14 (1993), 461–469.
13. Y. Achdou, P. Le Tallec, F. Nataf, and M. Vidrascu, A domain decomposition preconditioner for an advection-diffusion problem, Comput Methods Appl Mech Eng 184 (2000), 145–170.
14. A. Quarteroni and A. Valli, Domain decomposition methods for partial differential equations numerical mathematics and scientific computation, Oxford University Press, 1999.
15. B. F. Smith, P. Bjørstad, and W. D. Gropp, Domain decomposition parallel multilevel methods for elliptic partial differential equations, Cambridge University Press, 1996.
16. A. Toselli and O. Widlund, Domain decomposition methods—algorithms and theory, Springer series in computational mathematics, Springer, 2005.
17. A. Klawonn, L. Pavarino, and O. Rheinbach, Spectral element FETI-DP and BDDC preconditioners with multi-element subdomains, Comput Methods Appl Mech Eng 198 (2008), 511–523.
18. J. Mandel and C. Dohrmann, Convergence of a balancing domain decomposition by constraints and energy minimization, Numer Linear Algebra Appl 10 (2003), 639–659.
19. X. Tu and J. Li, BDDC for nonsymmetric positive definite and symmetric indefinite problems, Technical Report LBNL-1316E, Lawrence Berkely National Laboratory, Available at: http://repositories.cdlib.org/lbnl/LBNL-1316E, January 2009.
20. C. Farhat, M. Lesoinne, P. Le Tallec, K. Pierson, and D. Rixen, FETI-DP: a dual–primal unified FETI method. I. A faster alternative to the two-level FETI method, Int J Numer Methods Eng 50 (2001), 1523–1544.

21. L. Pavarino, BDDC and FETI-DP preconditioners for spectral element discretizations, Comput Methods Appl Mech Eng 196 (2007), 1380–1388.

22. R. Lynch, J. Rice, and D. Thomas, Direct solution of partial difference equations by tensor product methods, Numer Math 6 (1964), 185–199.

23. H. G. Roos, M. Stynes, and L. Tobiska, Numerical methods for singularly perturbed differential equations, Springer-Verlag, Berlin, 1996.

24. F. Pasquarelli and A. Quarteroni, Effective spectral approximation of the convection-diffusion equations, Comput Methods Appl Mech Eng 116 (1994), 39–51.

25. C. Canuto and G. Puppo, Bubble stabilization of spectral Legendre methods for the advection-diffusion equation, Comput Methods Appl Mech Eng 118 (1994), 239–263.

26. W. Eckhaus, Asymptotic analysis of singular perturbations, Elsevier Science, Amsterdam, 1979.

27. D. Braess, Finite elements theory, fast solvers, and applications in solid mechanics, 2nd Ed., Cambridge University Press, 2001.

28. H. C. Elman and M. P. Chernesky, Ordering effects on relaxation methods applied to the discrete one-dimensional convection-diffusion equation, SIAM J Numer Anal 30 (1993), 1268–1290.

29. A. Toselli, FETI domain decomposition methods for scalar advection-diffusion problems, Comput Methods Appl Mech Eng 190 (2001), 5759–5776.