ABSTRACT

Title of dissertation: FAST SOLVERS FOR MODELS OF FLUID FLOW WITH SPECTRAL ELEMENTS

P. Aaron Lott, Doctor of Philosophy, 2008

Dissertation directed by: Professor Howard Elman
Department of Computer Science
Institute for Advanced Computer Studies
&
Associate Research Professor Anil Deane
Institute for Physical Sciences and Technology

We introduce a 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 and the linearized Navier-Stokes equations. The method is based on iterative substructuring where fast diagonalization is used to efficiently eliminate the interior degrees of freedom and subsidiary subdomain solves. We demonstrate the effectiveness of this preconditioner in numerical simulations using a spectral element discretization.

This work extends the use of Fast Diagonalization to steady convection-diffusion systems. We also extend the “least-squares commutator” preconditioner, originally developed for the finite element method [19], to a matrix-free spectral element framework. We show that these two advances, when used together, allow for efficient computation of steady-state solutions the the incompressible Navier-Stokes equations using high-order spectral element discretizations.

FAST SOLVERS FOR MODELS OF FLUID FLOW WITH SPECTRAL ELEMENTS

by

P. Aaron Lott

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2008

Advisory Committee:
Professor Howard Elman, Chair/Advisor
Associate Research Professor Anil Deane, Co-Advisor
Professor Ricardo Nochetto
Associate Professor Elias Balaras
Associate Professor James Baeder, Dean's Representative

Acknowledgements

I would like to thank all of those who have assisted me in this effort. I am especially indebted to Howard Elman who has provided me with exceptional guidance, instruction, and encouragement throughout this research. I greatly appreciate the effort he has spent in directing and editing this thesis as well as the financial support he has provided me. I am also very grateful to Anil Deane for introducing me to computational fluid dynamics and for providing me with funding and computational resources. I would like to thank him for connecting me with Tom Clune at NASA Goddard Space Flight Center and Paul Fischer at Argonne National Laboratory, both of whom assisted me on aspects of this research. In addition, I would like thank Ricardo Nochetto, Elias Balaras and James Bader for serving on my committee.

I would like to commend the many incredible teachers that I have had the opportunity to learn from. They have encouraged, inspired and prepared me to reach beyond what I knew to be possible. It is because of their skill, effort, and passion for research and teaching that I have pursued a career in the mathematical sciences. I owe a special thanks to my high school math teacher Eveyln Lott for mentoring me and inspiring me to study mathematics. I am also considerably grateful to Temple H. Fay for sharing his zeal for mathematics research with me throughout college, and for advising me through numerous undergraduate projects that led me to pursue this degree.

These efforts would have been much more difficult, if not impossible, without the support the many friends I have been lucky to have met while in graduate school. In particular, I would like to thank my wife, Yue Xiao, for her support. Yue has sustained me

through this research by providing me with encouragement, advice, friendship, incredible food and a wonderful life outside of school. Her efforts have given me strength and confidence to continue through the many struggles and disapointments of graduate school. I would like to thank my officemates J.T. Halbert, Chris Danforth, Dongwook Lee and Mike O'Hara for their support and many engaging conversations. I am also grateful for the many discussions I have had with Bob Shuttleworth, Diego Donzis and Alfredo Nava-Tudela involving fluid dynamics and computation.

Finally, I would like to thank my parents Paul and DeeAnn for helping me to establish a positive attitude towards learning. They have dedicated significant time, energy and money to see that I acquire a solid education by helping me with homework, talking with my teachers, and providing me with the opportunity to attend college. Their commitment to my education has motivated me to focus on school work and has given me the drive to work through many difficult problems.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Advancements in numerical simulation have led to large-scale fluid simulations that assist in characterizing flow behavior and understanding the effect of changes in physical parameters on such behavior. With improved mathematical modeling and computational methods, we are now able to attack more challenging fluid problems such as those involving strong convection in geometrically complex domains. In such flows, inertial and viscous forces occur on disparate scales causing sharp flow features. These sharp features require fine numerical grid resolution and cause the governing systems to become poorly conditioned. The coefficient matrix that arises from discretization of the convection-diffusion system is the sum of a symmetric matrix that accounts for diffusion and a non-symmetric matrix that accounts for convection, and the combined convection-diffusion system is non-symmetric. As convection dominates the flow, the large discrete non-symmetric ill-conditioned system becomes challenging to solve.

Conventional solution algorithms based on semi-implicit time stepping ([10], [42], [13], [28], [34], [43]) avoid the challenges of solving the non-symmetric convection-diffusion systems by applying intricate splitting schemes that separate the governing system into convective (inertial) and diffusive (viscous) components [34]. Convection is often handled through explicit time integration, while diffusion is treated implicitly, requiring a Poisson solve. One of the central benefits of using such methods is that they take advantage of the vast research for solving symmetric systems on large computers.

As Deville et al. explain in [12], over the past 60 years, algorithms for obtaining a three-dimensional solution to Poisson equation on a mesh with $(N+1)^3$ grid points have improved remarkably. At the time of the first digital computer in 1947, Gaussian Elimination using banded LU with an operation count $O(N^7)$ was state-of-the-art [24]. Over the next several years computational scientists took advantage of the structure of the discrete symmetric system to construct more efficient algorithms such as Successive Over-relaxation (SOR) $O(N^5)$ [60] and Modified SOR $O(N^4 log(N))$ [61], Fast Diagonalization $O(N^4)$ [32], Cyclic Reduction [7] $O(N^3 log(N))$, Multigrid $O(N^3)$ [5] and Parallel Multigrid $O(N^3 log(N)/P)$ using P processors [37].

Deville et al. also point out that these algorithmic improvements have been accompanied by (indeed, exceeded by) gains in processing speed. We refer to Figure 1.1 to illustrate gains in both algorithmic efficiencies and computer processing speed for performing a three-dimensional Poisson simulation on a 64^3 grid. The dark blue curve indicates that algorithms have contributed to nearly 6 orders of magnitude improvement, whereas the green curve shows that vector machines have produced nearly 8 orders of magnitude improvement. Gains from conventional vector hardware are not likely to continue to improve at such rates due to cost, memory and communication latency, as well as load distribution on massively parallel machines. The current and projected trend in high performance computing is in developing massively parallel architectures. The red curve shows the combined speedup of algorithms and machines on the best vector computer, and the light blue curve represents the speedup on the best parallel machine using 32 processors. Finally, the black curve depicts the speedup of the best algorithm and machine using all machine processors. This combined curve shows nearly 18 orders of magnitude in improvement from the most advanced simulation techniques available since the beginning of digital computing in 1947. That is, a simulation that would have taken centuries on the 1947 Mark I computer using Gaussian Elimination now takes only a few seconds perform using Parallel Multigrid on the IBM Blue Gene/P supercomputer constructed in 2007.

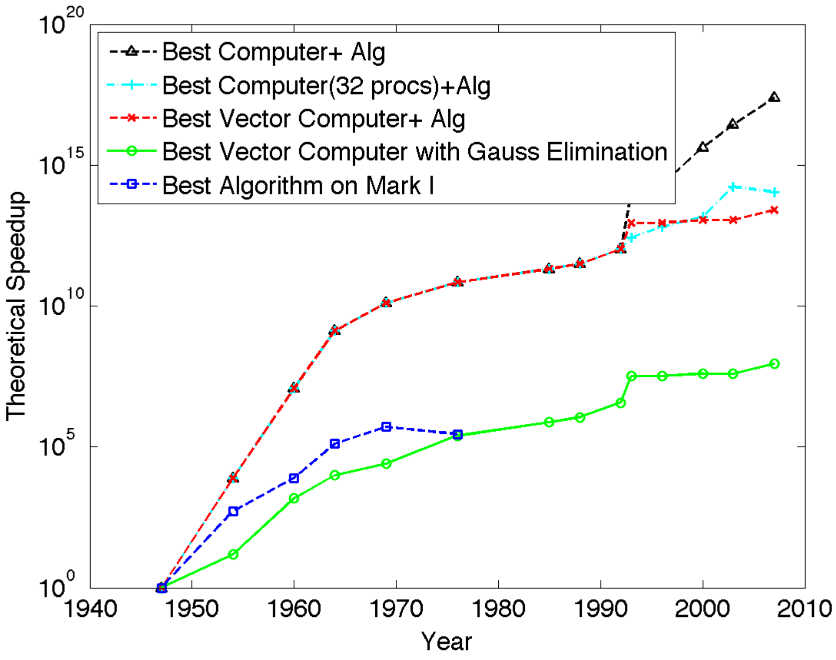

Figure 1.1: Evolution of machines, algorithms and their combination over the past 60 years shown through the speedup of solving a 3D Poisson equation. Adapted from [12] page 27.

This immense improvement in simulation speed through algorithms and hardware might give the impression that conventional solution algorithms based on solving Poisson equation are sufficient, and that there is little room to improve computational methods for simulating fluids. However, the above analysis is for a fixed problem size with a relatively small number of degrees of freedom compared to what is needed for modern science and engineering simulations. Figure 1.2 shows that there has been over 7 orders of magnitude increase in problem size in the past 30 years. Despite the exponential increase in simulation efficiency and size, relatively small gains have been made in our ability to resolve the transfer of energy in fast moving flows.

The right pane in Figure 1.2 illustrates this by comparing the size of the largest model of Taylor Microscale Reynolds number flow simulated during the past 30 years. This figure shows that three-dimensional time-dependent direct numerical simulations (DNS) that accurately resolve the length scale where viscous dissipation begins to affect flow eddies (Taylor Microscale) have succeeded in resolving flows with speeds higher only by two orders of magnitude. Even more troubling is that these state-of-the-art simulations are almost all performed in idealistic settings involving simple geometries and periodic boundary conditions [40]. In particular, the techniques used in these high-end simulations have limited applicability in engineering and scientific problems involving complicated domains with mixed boundary conditions. Thus, the ability to solve the Poisson equation efficiently using conventional methods certainly does not translate directly to a capability to study a wide range of realistic flow problems.

Moreover, direct numerical simulation of flows is not the only computational tool available for exploring the physical properties of turbulence and determining critical flow parameters that govern the behavior of fluids [8]. The study of small-amplitude perturbations to base states of a flow can be used to categorize flow behavior near steady or time-periodic flow states. This information helps quantify critical parameters that affect flow instabilities in the transition to turbulence [48], [49], [56].

To perform such analysis, numerical simulations track steady solutions in parame-

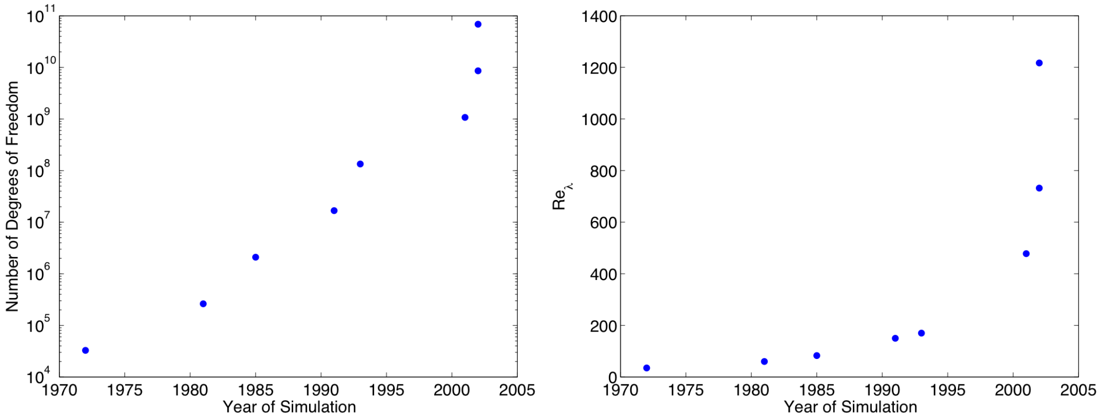

Figure 1.2: Number of degrees of freedom for a single flow field in modern turbulence simulations needed to fully resolve a flow at increasing Reynolds number (left). Largest Taylor Microscale Reynolds number flow computed via DNS versus year. Data from [40] (1972), [51] (1981), [58] (1991), [29] (1993), [25] (2001), [59] (2002).

ter space. These simulations obtain steady states by either computing them directly via nonlinear iteration or by integrating time dependent systems over long time periods until unsteady phenomena settle. Nonlinear solvers have a computational advantage over time-stepping techniques when quickly converging nonlinear iteration schemes can be applied. Moreover, in cases where time-stepping is used, fully-implicit methods are preferred over semi-implicit schemes since large time steps can be taken. Both nonlinear iteration and fully-implicit methods require solving non-symmetric convection-diffusion problems, where many techniques used by conventional solvers do not apply. Additionally, high-order numerical discretization methods have been shown to have advantages over low-order methods to accurately capture physical instabilities since low-order schemes are inherently dissipative in space and dispersive over time [22],[30],[53]. The challenge in applying these simulation techniques reduces to finding effective solvers for poorly conditioned non-symmetric linear systems that arise from high-order discretization of models of steady flows.

The structure of the discrete algebraic systems arising in such fluid models is determined by the choice of numerical discretization. This thesis will explore solution algorithms for spectral element discretizations [44]. Element-based discretizations divide the computational domain into non-overlapping sub-domains (elements), and represent the variables on each element via a polynomial basis. The accuracy of solutions can be improved by reducing element sizes (h-refinement) and also by increasing the order of the polynomial basis (p-refinement). To improve the accuracy of element-based discretizations, the spectral element method was developed by Patera [44] in 1984. This method uses a high-order Legendre polynomial basis on each element to achieve spectral accuracy while maintaining the geometric flexibility of the finite element method. These properties cause spectral element methods to be particularly well suited for modifying the discretization locally without global refinement [36]. High-order element-based discretizations, such as the spectral element method, produce large matrices having sparse block structure with dense sub-blocks. Each dense block corresponds to a single element in the discretization and can be represented using tensor products of associated one-dimensional phenomena. This element-based tensor product formulation can be coupled with gather-scatter operations to incorporate elemental interfaces. This format admits computationally efficient matrix-free solvers and preconditioners that can take advantage of the tensor product structure within elements [12].

The development of iterative solvers for non-symmetric systems that allow for preconditioners that vary at each step such as Flexible GMRES [47] and GMRESR [11] have made Krylov-Schwarz and Krylov-Schur methods the contemporary approach [26] for solving discrete steady flow systems. The number of iterations required for such methods to converge depends on various properties of the linear systems, such as the distribution of eigenvalues of the coefficient matrix or the number of sets of clustered eigenvalues. These properties can often be improved using preconditioning methods [19], [17], [14] which is the focus of this study.

In recent years, preconditioners for fluid systems have been based on Block-Jacobi and Additive-Schwarz methods [3], [20], [9]. These preconditioners have been shown to provide improved convergence for symmetric systems, but they have not been demonstrated to be effective when applied to non-symmetric systems. To incorporate the non-symmetric components, we propose a Domain Decomposition preconditioner that uses average velocity contributions on each sub-domain. We apply appropriate inter-element boundary conditions to allow information to pass between elements based on the direction of the fluid flow. We demonstrate that this class of preconditioner can be used to improve convergence of the non-symmetric systems that arise from the steady convection-diffusion equation. We also incorporate this technique into the "Least Squares Commutator" block preconditioning framework of Elman et al. [17] to solve linearized Navier-Stokes equations efficiently. The efficiency of our solution method is centered on two advancements: the use of an accurate high order matrix-free discretization to construct accurate discrete solutions while minimizing memory requirements, and the use of fast iterative solvers accelerated by domain decomposition based preconditioners to take advantage of local tensor product structures by exploiting Fast Diagonalization. These features allow the solution method to take into account memory hierarchies and efficient cache use in order to improve processor performance.

The thesis is organized as follows. In Chapter 2 we outline fundamental properties of the spectral element method that are used in the discretization of the convection-diffusion and Navier-Stokes equations. We discuss features of the discretization that are important when developing solution methods. In Chapter 3 we construct a domain decomposition solver for constant coefficient problems and then explain how this solver can be used as a preconditioner for general convection-diffusion systems. We illustrate properties of the solver and preconditioner through a number of test cases. In Chapter 4 we introduce a solution technique for solving the steady Navier-Stokes equations and then construct a block preconditioner for the linearized Navier-Stokes equations based on our convection-diffusion preconditioner and an approximation to the “Least Squares Commutator” developed by Elman et al. [19]. We discuss results of this approach via several case studies. In chapter 5 we summarize our results, draw conclusions about the method’s effectiveness, and discuss some potential directions for future study and application.

Chapter 2

Spectral Element Discretization

The spectral element method is a numerical method for solving differential equations using a finite representation of the solution. The method is based on the method of weighted residuals which we describe by considering a linear differential equation on a domain Ω, namely,

$$D(u) = 0, \tag{2.1}$$

with appropriate boundary conditions. The solution to this equation, $u(x)$, can be approximated by

$$\tilde{u}(x) = u_D(x) + \sum_{i=1}^{N} \hat{u}_i \pi_i(x) \tag{2.2}$$

where $u_D(x)$ satisfies Dirichlet boundary conditions, and $\pi_i(x)$ are known functions, referred to as trial functions and $\hat{u}_i$ are unknown weights. To determine these weights, the approximate solution is substituted into the differential equation, which produces a system

$$D(\tilde{u}) = R(\tilde{u}). \tag{2.3}$$

Restrictions are then placed on the residual $R(\tilde{u})$ in order to minimize it over Ω. This is done by setting the inner product of R and some set of test function $\{v_j\}$ equal to zero over Ω, i.e.

$$\int_{\Omega} v_j(x) R(\tilde{u}) = 0, \quad j = 1, ..., N. \tag{2.4}$$

The spectral element method is based on Galerkin method, in which the trial functions π_i and the test functions v_j are taken from the same set of functions. The set of N equations determined by (2.4) is solved for the weighting coefficients $\hat{u}$. When multiple elements are used, integrals of equation (2.4) are broken up into a sum of the integrals on each element. In particular, the test and trial functions consist of high order Legendre polynomials at Gauss-Legendre-Lobatto (GLL) or Gauss-Legendre (GL) nodes.

The resulting system of linear matrix equations consist of dense blocks that represent the coupling within local elements together with interface coupling. The use of rectangular elements allow each elemental system to be represented via tensor products of one-dimensional operators. This can be extended to higher dimensions, and general quadrilaterals as well as certain deformed domains via invertible mappings to the d-dimensional square (see [12] p. 178). Nodes that are shared between multiple elements must be coupled in this matrix system. In particular, each element has nodes along elemental boundaries which are coupled to adjacent boundary nodes, as well as interior nodes. These inter-element couplings can be enforced by either constructing a fully coupled sparse linear system of equations, or by performing a gather-scatter operation (see Program 1) that sums variables along element boundaries after element-based matrix-vector products are performed. This gather-scatter operation coupled with the tensor product formulation of elemental operators yields a matrix-free discretization in which only matrices associated with one-dimensional phenomena need to be stored.

The efficiency and accuracy of spectral methods have led them to become a primary instrument for simulating fluid flows. However, these methods can only be applied to flows that can be mapped to a simple reference domain, such as a cube or sphere with periodic boundary conditions. Since Orszag's original formulation of spectral methods for simulating flows [39], several alternative element-based methods have been proposed to extend solution methods to complex domains, notably the h/p finite element method of Babuška [2] and the spectral element method of Patera [44]. Both of these discretization methods can manage complicated domains with a variety of boundary conditions. Perhaps more importantly, these methods also allow for adaptive mesh refinement by modifying the number of elements and the polynomial basis to capture local flow behavior. Each of these discretization methods, divide the computational domain into several non-overlapping subdomains, and on each subdomain the solution is expressed through a polynomial basis. The two methods can be distinguished through the choice of basis functions [2],[30].

In the case of h/p finite elements, the solution is expressed through a *hierarchical modal* polynomial basis on each subdomain. The basis functions are written as:

$$\pi_n(\xi) = \begin{cases} (\frac{1-\xi}{2}) & n = 0 \\ (\frac{1-\xi}{2})(\frac{1+\xi}{2})P_{n-1}^{a,b} & 0 < n < N \\ (\frac{1+\xi}{2}) & n = N. \end{cases} \tag{2.5}$$

The expansion modes π_0 and π_N correspond to the basis functions for linear finite elements, and are the only modes that are non-zero on the elemental boundaries. The higher order modes are interior to the element, and are often referred to as "bubble modes". As N increases, the basis simply adds additional interior modes, thus making the basis hierarchical. The functions $P_{n-1}^{a,b}$, are orthogonal Jacobi polynomials, the indices a and b denote the type of polynomial defined as a solution to the Sturm-Liouville problem. For example $(a,b) = (0,0)$ produces Legendre polynomials, where as $(a,b) = (-1/2,-1/2)$

corresponds to Chebyshev polynomials. The choice of these interior functions and the order N determines the structure of the discrete linear systems that arise from discrete partial differential equations. Typically, the Finite Element Method is employed as a low order scheme, with polynomial basis functions on each element ranging from degree 1 to 4. Due to orthogonality with the boundary modes, the choice $(a,b) = (1,1)$, produces system matrices with minimal fill-in, thus making this the standard choice.

The Spectral Element Method was designed to combine the accuracy of spectral methods with the geometric flexibility of finite element methods [35] using a *non-hierarchical nodal* basis through Lagrange polynomials based on zeros of Gauss-Legendre polynomials. By construction, this forms an orthogonal basis containing $N+1$ polynomials, all of degree N. As the degree N increases, all the members of the basis are changed, thus making the basis non-hierarchical. The specifics of this basis are discussed in the following section.

The linear systems resulting from high-order element-based discretizations have sparse nonzero structure with dense sub-blocks and thus allow for advanced computational methods to be applied in order to improve efficiency of associated solvers. Often, implementations of the spectral element method use a “matrix-free” formulation to allow matrix-vector operations to be formulated as cache efficient matrix-matrix computations on each element. That is, the global matrix is never fully constructed, and instead all matrix operations use local, element-based computations, together with global mapping constructions to handle interelement couplings. In this chapter we introduce several mathematical and computational properties of the spectral element method that are used to construct efficient solvers.

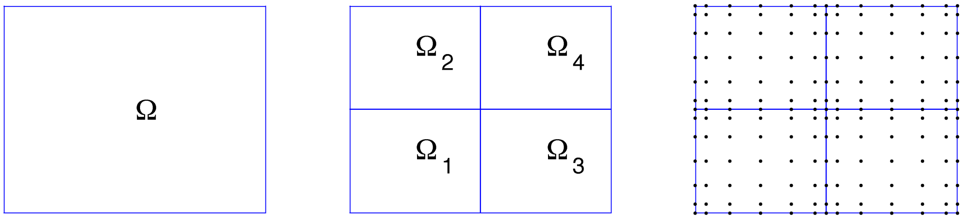

Figure 2.1: Illustration of a two dimensional spectral element discretization.

2.1 Basics

The spectral element method can be viewed as a global spectral method with multiple domains where the solution must match along adjacent domains. In Figure 2.1, we illustrate how a computational domain Ω (left) is divided into multiple elements Ω_1 through Ω_4 (middle), and with Gauss-Legendre-Lobatto interpolation nodes placed on each element (right). Spectral methods have small numerical dispersion and dissipation, compared to low-order schemes typically used in finite element and finite differencing methods. Figure 2.2 compares the amount of work (number of floating point operations) required to maintain a 10% phase error in a one-dimensional advecting wave using three different spatial discretizations. The figure shows that over long time periods, a higher order discretization method will reduce the amount of work, while lower order methods require significantly more grid points (and thus more work) to keep the wave in phase. This means that for smooth enough problems high-order discretizations lead to more accurate solutions and sharp estimates for critical parameters at less cost [30].

The computational grid of the spectral element method is the key to the method's convergence properties and its geometric flexibility. The grid is formed by dividing the domain Ω with a set of non-overlapping elements Ω_e, and placing an orthogonal nodal spectral basis π^N in each element defined on a set of Gauss-Legendre-Lobatto nodes ($\Xi_{N+1} := \xi_0, \xi_1, ...\xi_N$) which are the roots of the $N+1$ degree Legendre polynomials in equation (2.9) (see Figure 2.1). This allows functions defined on Ω_e to be written in terms of their spectral basis on each element

$$u_e^N(\xi) = \sum_{i=0}^{N} u_i(\xi_i)\pi_i^N(\xi). \tag{2.6}$$

The basis has the form

$$\pi_i^N(\xi) = \frac{-1}{N(N+1)} \frac{(1-\xi^2)L_N{}'(\xi)}{(\xi-\xi_i)L_N(\xi_i)} \qquad 0 \leq i \leq N, \tag{2.7}$$

where $L_N(\xi)$ are the N^{th} order Legendre polynomials, which can be determined via the three-term recursion relationship

$$L_0(x) = 1 \qquad L_1(x) = x, \tag{2.8}$$

$$(k+1)L_{k+1}(x) = (2k+1)xL_k(x) - kL_{k-1}(x), \qquad k \geq 1.$$

Similarly, derivatives of Legendre polynomials can be determined via the relationship

$$L_0(x) = L_1'(x) = 1, \tag{2.9}$$

$$(2k+1)L_k(x) = L_{k+1}'(x) - L_{k-1}'(x), \qquad k \geq 1.$$

We note that $\pi_i^N(\xi_i) = 1$ and $\pi_i^N(\xi_j) = 0$ when $i \neq j$ as shown in Figure 2.3. This forces the weighting coefficients u_i to be the value of u_e^N at each of the nodal points. Figure 2.3 shows an example of the one-dimensional basis functions $\pi^N(x)$ for polynomial degree six. In higher dimensions the basis is simply a tensor product of the one-dimensional basis functions. So, for example, in two dimensions a function may be expressed as

$$u_e^N(x,y) = \sum_{i=0}^{N} \sum_{j=0}^{N+1} u_{ij}\pi_i^N(x)\pi_j^N(y). \tag{2.10}$$

The two-dimensional extension of the orthogonal nodal basis shown in Figure 2.3 is displayed in Figure 2.4.

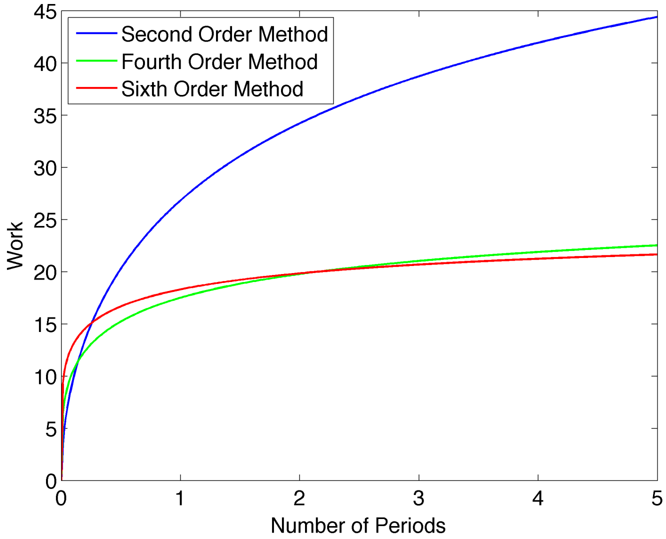

Figure 2.2: Comparison of computational work (FLOPS) needed to maintain 10% phase error in 1D advection equation [30] p. 10.

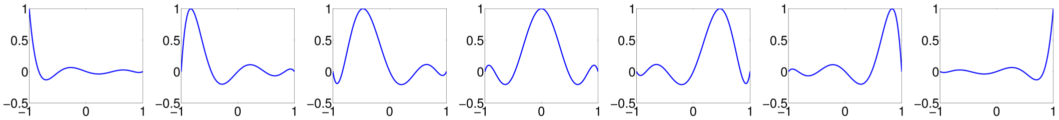

Figure 2.3: 6th Order Lagrangian nodal basis functions $\pi_0 - \pi_6$ (left to right) based on the GLL points

Spectral elements provide two mechanisms for improving spatial accuracy. The first is h-refinement, which means that the size of subdomains can be decreased, by including more subdomains to the discretization; continuing this type of refinement process leads to *algebraic convergence* (see Definition 1 below) to the solution. The second form of improving spatial accuracy is by increasing the order of the polynomial basis on each element. This is termed p-refinement. As one performs p-refinement to a discretization, the discrete solution approaches the exact solution exponentially, giving rise to what is

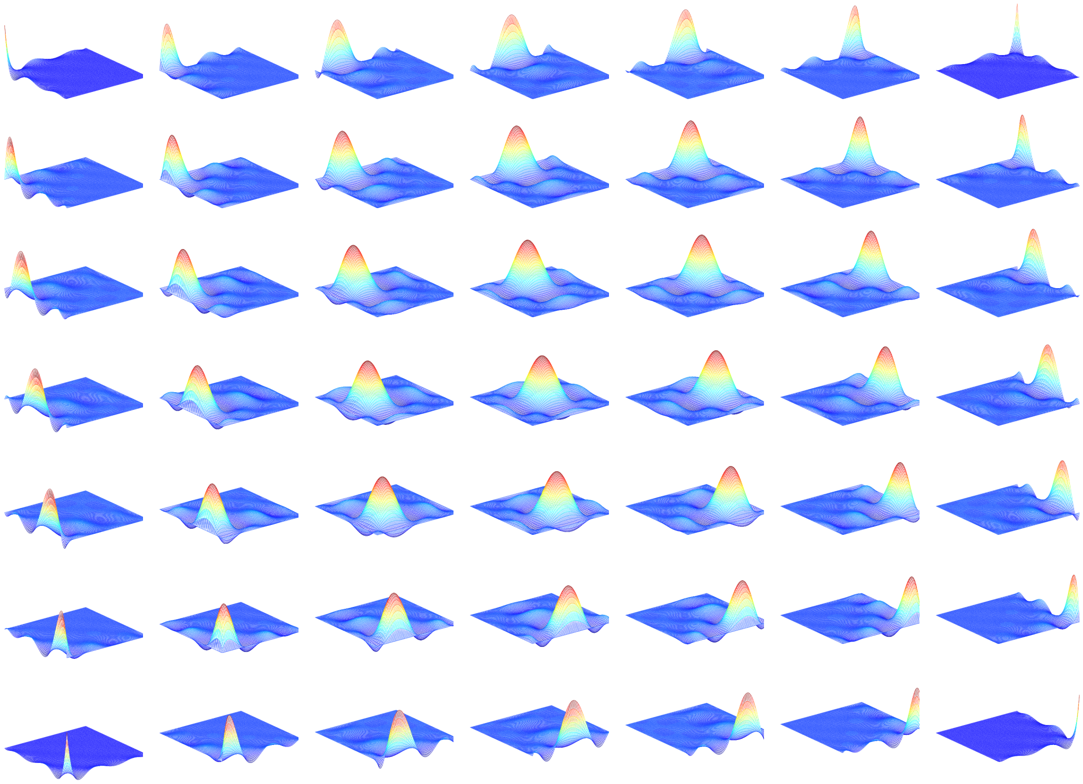

Figure 2.4: 6th Order Two Dimensional Lagrangian nodal basis functions $\pi_i \otimes \pi_j$ (ordered left to right, bottom to top) based on the Gauss-Lobatto-Legendre points.

called *exponential convergence* (see Definition 2 below). In practice, both methods have been used, [27], [36], [46]. Due to the low order (algebraic) and high order (exponential) convergence properties, spectral methods tend to require about half as many degrees of freedom in each spatial dimension to equally resolve a flow compared to low order methods [4]. In [22] Fischer et al. demonstrate that with a fixed number of degrees of freedom, roughly half of the eigen-spectrum associated with a pure advection problem is resolved by 64 8^{th} order elements, whereas only ten percent of the associated eigenvalues are represented by the same number of points with 512 linear elements.

If one considers a fixed desired accuracy, the trade off for the smaller memory footprint of higher order elements is an increase in computational cost per degree of freedom.

Definition 1 Algebraic Convergence

For fixed polynomial degree and increasing number of elements, u^N will algebraically approach u, that is, as the number of elements are doubled, the error is reduced by a constant factor $\alpha < 1$.

Definition 2 Exponential Convergence

For fixed number of elements and increasing polynomial degree, N, if the function $u \in H_0^s(\Omega) \times H^{s-1}(\Omega)$ having smoothness s , then the error in the approximation u^N will decay exponentially. That is $\|u - u^N\| \leq Ch^{\min(N,s)} N^{-s} \|u\|_{H_0^s(\Omega)}$. (see [12] p. 273).

This additional cost, however, may be mitigated by using a matrix-free discretization that invokes cache efficient element-based matrix-matrix calculations [57]. In Table 2.1, reproduced from [21], we see that matrix-matrix based calculations obtain speedup through fewer memory accesses per operation. Memory access is a major computational bottleneck since memory speed is significantly slower than processor speeds. High-order element-based calculations provide improved parallelism over low-order element-based methods by reducing global communication through a decrease in surface to volume ratios on each element. These computational efficiencies offset the added cost per degree of freedom, making spectral elements a competitive choice for discretizing the equations that govern fluid motion.

		Operations	Memory Access	Op:Memory Access
Vector-Vector	$\vec{x} = \vec{x} + c\vec{y}$	$2n$	$2n$	1
Matrix-Vector	$\vec{x} = A\vec{y}$	$2n^2$	n^2	2
Sparse Matrix-Vector	$\vec{x} = A\vec{y}$	$2mn$	mn	2
Matrix-Matrix	$C = AB$	$2n^3$	$2n^2$	n

Table 2.1: Comparison of CPU and Memory access costs for Matrix and Vector Operations. Adapted from [21], page 14.

The discrete elemental operators in two and three dimensions can be expressed as tensor products of one-dimensional operators. We derive the operators corresponding to the convection-diffusion and Navier-Stokes equations in their respective chapters. Inter-element coupling of these matrices ensures continuity on the elemental boundaries. Elemental couplings can be enforced by either assembling a coupled system of equations, or by performing a gather-scatter operation on un-assembled vectors. This gather-scatter operation, known as "direct stiffness summation", appropriately sums the weighting coefficients at element interfaces. We outline the details of this procedure in section 2.3. The tensor product representation of the elemental matrices together with direct stiffness summation allows for an efficient matrix-free formulation, in which only matrices associated with one-dimensional phenomena need to be stored. Moreover, fast diagonalization, discussed in section 3.3, enables the efficient application of the action of the inverse of certain elemental operators in higher dimensions. In the remainder of this chapter we discuss implementation strategies for our matrix-free formulation.

2.2 Tensor Products

The Spectral Element Method allows for efficient evaluation of various operations with large-dimensional matrices in terms of one-dimensional Kronecker tensor products (see Definition 3). When computing matrix-vector products one can reshape the vector as a matrix in order to exploit the property

$$(A_{n\times n} \otimes B_{n\times n})\vec{u}_{n^2\times 1} = B_{n\times n}U_{n\times n}A^T_{n\times n}, \tag{2.11}$$

where $U_{n\times n}$ is $u_{n^2\times 1}$ reshaped as a matrix. This allows a matrix-vector product to be reduced from an $O(n^4)$ calculation to an $O(n^3)$ calculation. That is, matrices of the form $C = A\otimes B$ representing an operator on a discrete two-dimensional space can be applied using matrix-matrix multiplication with one-dimensional operators A and B. We will see in the following chapters that, in fact, all the matrices arising from the discrete fluid models we will study are essentially of this form.

Definition 3 Kronecker Tensor Product

Given A of dimension $k\times l$ and B of dimension $m\times n$, the Kronecker Tensor Product $A\otimes B$ is the matrix of dimension $km\times ln$ given by

$$C := \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1l}B \\ a_{21}B & a_{22}B & \dots & a_{2l}B \\ \vdots & \vdots & & \vdots \\ a_{k1}B & a_{k2}B & \dots & a_{kl}B \end{pmatrix}. \tag{2.12}$$

In general, tensor-based matrix-vector products involving a discretization with n mesh points per spatial dimension, with d spatial dimensions require only $O(n^{d+1})$ operations. This evaluation method is routinely used inside iterative solvers to improve efficiency.

Another important property of tensor products that we use is Fast Diagonalization [32]. This procedure allows one to inexpensively apply the action of the inverse of systems defined on each element by diagonalizing corresponding one-dimensional operators. We describe this method in section 3.3 in the context of convection-diffusion operators.

2.3 Element Coupling

In the previous section we observed that when discretizing a continuous system using the spectral element method, one obtains a system of elemental equations that can be coupled either through an assembly process, or through an appropriate gather-scatter operation. In large-scale fluid applications, it is often not feasible to store the assembled system matrix in memory. In this section we introduce the direct stiffness summation process which is used to couple the matrix-free system. Direct stiffness summation, denoted as Σ', is defined in [12] as "a non-invertible local-to-local transformation that sums shared interface variables and then stores these values along the interfaces leaving interior nodes unchanged". The operation can be performed in a variety of ways; many codes use geometric information about element nodes to keep track of neighbors. We adopt a geometry-free approach proposed in [12] that uses mappings between the coupled and un-coupled degrees of freedom. These mappings are formed by representing degrees of freedom in two formats: one is the coupled (global) format and the other is the un-coupled (local) format. Figure 2.5 shows how the nodes would be indexed in each of these formats. Once the data is written in these two formats, index maps are created for each. An example of these maps for the indexing scheme presented in Figure 2.5 would be:

```
global_map(1,1:9)=(1, 2, 3, 4, 5, 6, 7, 8, 9)

global_map(2,1:9)=(3, 10, 11, 6, 12, 13, 9, 14, 15)

local_map(1,1:9)=(1, 2, 3, 4, 5, 6, 7, 8, 9)

local_map(2,1:9)=(10, 11, 12, 13, 14, 15, 16, 17, 18).
```

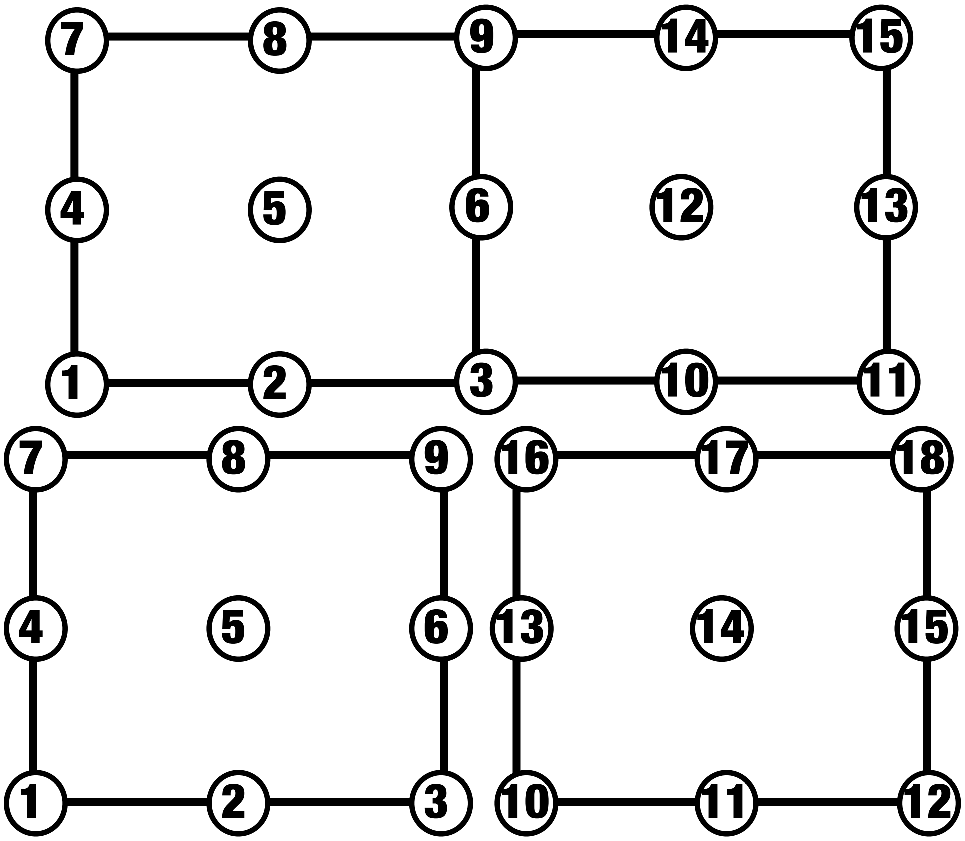

Figure 2.5: (Top) Global and (Bottom) Local ordering of the degrees of freedom.

After these maps are formed, the direct stiffness summation operation can be performed as a sequence of two loops. The first loop takes a vector stored in the local format, maps the entries to the global format and sums interface nodes for each element. The second loop takes the newly created global vector and maps the entries to the local format over each element. The end result is that the elemental interface values are all summed. Prior to the direct stiffness summation, each local vector contains only a component of the global vector along the interface. After the summation, there is a local vector whose entries contain the global value for all nodes lying on interfaces. This procedure is geometry-free, meaning that only mappings of the nodes in the two indexing schemes need to be created in order for this method to work.

Program 1 Pseudo code for Direct Stiffness Summation Σ'

```
function dss(ul)
  ug=0.0
  !gather
  do i=1,Num_El
     n=pdeg(i)+1
     n2=n*n
     ug(global_map(i,n2))=ug(global_map(i,n2))+ul(local_map(i,n2))
  end do
  !scatter
  do i=1,Num_El
     n=pdeg(i)+1
     n2=n*n
     dss(local_map(i,n2))=ug(global_map(i,n2))
  end do
end function dss
```

To provide insight to the mathematical structure of this coupling operation, Σ' can be viewed alternatively as a product of scatter Q and gather Q^T matrices (see equation (2.13)). For the node structure in Figure 2.5 the scatter operation would be represented via the matrix Q given in equation (2.13). The gather operation is simply the transpose of this matrix Q^T. The direct stiffness summation procedure can be performed as $\Sigma' := QQ^T$ to a vector in the local format. We can see in equation (2.13) how degrees of freedom expressed in a local format, corresponding to the indexing scheme at the bottom section in Figure 2.5, are mapped to values in the global format, corresponding to the indexing scheme at the top in Figure 2.5, and then back to the local format. We see here also that all interface values in the local format are stored twice. Moreover, one can see that the vector $\hat{u}_L$ formed from applying Σ' to u_L sums the interface values. For example $\hat{u}_9$ and $\hat{u}_{16}$ are the sum of u_9 and u_{16}.

Each time Σ' is applied to a vector, a weighting matrix, denoted as W, is applied to average vector values at elemental interfaces. W stores the inverse of the number of elements that contribute to a given node. This matrix is formed by performing Σ' on a unit vector in the local data format $e = 1.0$. This results in a vector y where y_i contains the number elements sharing node i. Thus the weighting matrix contains the inverse of y along the diagonal, i.e. $W_{ii} := \frac{1}{y_i}$. In the following chapters, when using Σ' we refer to the weighted direct stiffness summation operator $W\Sigma'$.

$$
\underbrace{\left(\begin{array}{c}
\hat{u}_1\\ \hat{u}_2\\ \hat{u}_3\\ \hline
\hat{u}_4\\ \hat{u}_5\\ \hat{u}_6\\ \hline
\hat{u}_7\\ \hat{u}_8\\ \hat{u}_9\\ \hline\hline
\hat{u}_{10}\\ \hat{u}_{11}\\ \hat{u}_{12}\\ \hline
\hat{u}_{13}\\ \hat{u}_{14}\\ \hat{u}_{15}\\ \hline
\hat{u}_{16}\\ \hat{u}_{17}\\ \hat{u}_{18}
\end{array}\right)}_{u_L}
=
\underbrace{
\underbrace{\left[\begin{array}{ccc|ccc|ccc||cc|cc|cc}
1&&&&&&&&&&&&&&\\
&1&&&&&&&&&&&&&\\
&&1&&&&&&&&&&&&\\ \hline
&&&1&&&&&&&&&&&\\
&&&&1&&&&&&&&&&\\
&&&&&1&&&&&&&&&\\ \hline
&&&&&&1&&&&&&&&\\
&&&&&&&1&&&&&&&\\
&&&&&&&&1&&&&&&\\ \hline\hline
&&1&&&&&&&&&&&&\\
&&&&&&&&&1&&&&&\\
&&&&&&&&&&1&&&&\\ \hline
&&&&&1&&&&&&&&&\\
&&&&&&&&&&&1&&&\\
&&&&&&&&&&&&1&&\\ \hline
&&&&&&&&1&&&&&&\\
&&&&&&&&&&&&&1&\\
&&&&&&&&&&&&&&1
\end{array}\right]}_{Q}
\underbrace{\left[\begin{array}{ccc|ccc|ccc||ccc|ccc|ccc}
1&&&&&&&&&&&&&&&&&\\
&1&&&&&&&&&&&&&&&&\\
&&1&&&&&&&1&&&&&&&&\\ \hline
&&&1&&&&&&&&&&&&&&\\
&&&&1&&&&&&&&&&&&&\\
&&&&&1&&&&&&&1&&&&&\\ \hline
&&&&&&1&&&&&&&&&&&\\
&&&&&&&1&&&&&&&&&&\\
&&&&&&&&1&&&&&&&1&&\\ \hline\hline
&&&&&&&&&&1&&&&&&&\\
&&&&&&&&&&&1&&&&&&\\ \hline
&&&&&&&&&&&&&1&&&&\\
&&&&&&&&&&&&&&1&&&\\ \hline
&&&&&&&&&&&&&&&&1&\\
&&&&&&&&&&&&&&&&&1
\end{array}\right]}_{Q^T}
}_{\Sigma'}
\underbrace{\left(\begin{array}{c}
u_1\\ u_2\\ u_3\\ \hline
u_4\\ u_5\\ u_6\\ \hline
u_7\\ u_8\\ u_9\\ \hline\hline
u_{10}\\ u_{11}\\ u_{12}\\ \hline
u_{13}\\ u_{14}\\ u_{15}\\ \hline
u_{16}\\ u_{17}\\ u_{18}
\end{array}\right)}_{u_L}
\tag{2.13}
$$

2.4 Remarks on Implementation

We conclude this chapter by providing a brief description of the computational framework we have constructed to perform our numerical experiments. We have developed a matrix-free implementation of the spectral element method within an object-oriented Fortran 90 code. The fundamental object of this code is what we call a "matrix object". This object has constructors that allow for matrix-free application of system operators that arise in fluid simulations, such as discrete Poisson, convection-diffusion, Stokes and linearized Navier-Stokes operators. Similarly, in this code, we have developed methods that apply inverse operations involved in solving these equations, such as Domain Decomposition and Fast Diagonalization which are discussed in detail in sections 3.2 and 3.3. These operations are conveniently applied to vectors using the multiplication symbol "$*$" throughout all modules of the code. The definition of a matrix object contains physical specifications about the type of matrix, such as symmetric or non-symmetric, communication maps for Σ', boundary maps for Dirichlet boundaries, and one-dimensional operators needed to apply Kronecker products. This allows for transparent application of matrices defined on a given computational grid. Our computational framework could also be extended to include interpolation operators to allow for mesh hierarchies and interfaces with coarse grid operators.

Chapter 3

Convection-Diffusion Equation

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 great theoretical, experimental and computational challenges in situations where the convective nature of the flow dominates diffusive effects. In such flows, convection and diffusion occur on disparate scales, which has motivated the development and use of splitting schemes [12]. The standard method for performing steady and unsteady flow simulations with spectral elements is operator integration factor splitting (OIFS) [34]. This typically involves time integration, even in steady flow simulations.

Using the standard approach, convection and diffusion are treated separately; convection components are tackled explicitly using a sequence of small time steps that satisfy the CFL 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. Such schemes have been successfully applied in a variety of settings ranging from the study of blood flow [22] to forecasting climate change [54]. However, to simulate fast moving flows the discretization must be refined to capture sharp flow features accurately, this in turn causes the time step of the semi-implicit methods to become acutely small. This means that 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 solve the steady state systems directly. Such methods, however, require fast steady state solvers that are able to resolve the disparate convective and diffusive scales efficiently. The steady convection-diffusion equation can be written as

$$-\varepsilon\nabla^2 u + (\vec{w}\cdot\nabla)u = f \tag{3.1}$$

where u represents velocity, the vector field $\vec{w} = (w_x(x,y), w_y(x,y))^T$ represents convection or wind speed at each point in the domain, f represents body forces acting on the fluid, and ε represents the kinematic viscosity. In addition to (3.1), we have the associated boundary conditions

$$u = u_D \text{ on } \partial\Omega_D, \qquad \nabla u_i \cdot \vec{n} = 0 \text{ on } \partial\Omega_N, \tag{3.2}$$

where $\vec{n}$ is the outward facing normal on the boundary, and subscripts D and N denote Dirichlet (inflow) and Neumann (outflow) boundary regions respectively. To measure the relative contributions of convection and diffusion, equation (3.1) can be non-dimensionalized 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 L, the forcing term is normalized by taking $f^* := fL/W^2$, and the velocity u^* and wind $w*$ are normalized by dividing by W. The velocity in the normalized domain can be described by the equation

$$-\frac{1}{Pe}\nabla^{*2}u^* + (\vec{w}^*\cdot\nabla^*)u^* = f^*. \tag{3.3}$$

The quantity $Pe := \frac{WL}{\varepsilon}$ is termed the *Peclet number*. This dimensionless number quantifies the contributions of convection and diffusion for a given flow. In diffusion dominated flows, $Pe \leq 1$, whereas in convection-dominated flows $Pe \gg 1$. We see in (4.3) that as $Pe \to \infty$ the flow becomes dominated by convection and the diffusion term goes away leading to a hyperbolic system. We observe that as $Pe \to 0$, this form of the convection-diffusion equation does not produce the Poisson equation. This is because we non-dimensionalized the equation by emphasizing the inertial terms. A similar non-dimensional form of the convection-diffusion equation can be written for slow moving flows by accentuating the viscosity. In this form the forcing term is replaced with $f^* = \frac{fL}{W\varepsilon}$, while the other quantities are normalized as above, leading to the dimensionless form

$$-\nabla^{*2}u^* + Pe(\vec{w}^* \cdot \nabla^*)u^* = f^*. \tag{3.4}$$

In this form, as $Pe \to 0$ we are left with the Poisson equation to model the flow. We use these two limiting situations to construct solvers and preconditioners that reflect the way flow information is spread throughout the domain.

In this chapter, we introduce a new approach for simulating steady flows with spectral elements by developing efficient solvers for the steady convection-diffusion system (3.1). We emphasize the case where Pe is large, although these techniques also apply to diffusion-dominated flows as well. Our method builds on ideas from iterative substructuring by exploiting 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, that are accelerated by domain decomposition based preconditioners that take advantage of the local tensor product structure by exploiting Fast Diagonalization to take into account memory hierarchies and efficient cache use to improve processor performance.

3.1 Spectral Element Method applied to the convection-diffusion equation

The spectral element method is a Galerkin method derived from the method of weighed residuals, in which a weak form equation is solved. When multiple elements are used, the associated integral is divided into a sum of integrals on individual elements. The element-based integrals are then approximated by numerical quadrature. In the original formulation of the spectral element method, Patera used interpolation points based at nodes of Chebyshev polynomials to take advantage of fast transforms [44]. However, this required analytical construction, and was found to be difficult to implement. Since then interpolation points based on roots of Legendre polynomials have been the standard choice. For the discrete convection-diffusion equation in particular, velocities are represented through a basis of high-order Legendre polynomials at Gauss-Legendre-Lobatto nodes. The resulting system of linear matrix equations numerically represents the original integral equation on each element. Rectangular elements allow each elemental system to be represented via tensor products of one-dimensional operators. Inter-element coupling of these matrix equations ensures continuity along elemental boundaries. These inter-element couplings can be enforced by either 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 Chapter 2, this gather-scatter operation coupled with the tensor product formulation of elemental operators yields a matrix-free discretization, in which only matrices associated with one-dimensional phenomena need to be stored.

Following the spectral element discretization strategy, we recast (3.1) in the weak form:

Find $u \in X$ such that:

$$\varepsilon \int_{\Omega} \nabla u \cdot \nabla v + \int_{\Omega} (\vec{w} \cdot \nabla u) v = \int_{\Omega} f v \qquad \forall v \in X_0, \tag{3.5}$$

where the velocity solution and test spaces are defined as

$$X := \left\{ \vec{u} \in H^1(\Omega)^2 | \vec{u} = \vec{u}_D \text{ on } \partial\Omega_D \right\} \tag{3.6}$$

$$X_0 := \left\{ \vec{u} \in H^1(\Omega)^2 | \vec{u} = \vec{0} \text{ on } \partial\Omega_D \right\}, \tag{3.7}$$

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 (see definition 6) and continuity (see definition 7) of the bilinear form

$$a(u,v) := \varepsilon \int_{\Omega} \nabla u \cdot \nabla v + \int_{\Omega} (\vec{w} \cdot \nabla u) v. \tag{3.8}$$

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

$$|a(u,v)| \leq \varepsilon \left| \int_{\Omega} \nabla u \cdot \nabla v \right| + \left| \int_{\Omega} (\vec{w} \cdot \nabla u) v \right|, \tag{3.9}$$

and bounding the terms using the Cauchy-Schwarz inequality. We refer the reader to [19] for details.

Next, we restrict u and v to a finite-dimensional subspace X^N to obtain the discrete weak form: Find $u \in X^N$ such that:

$$\varepsilon \int_\Omega \nabla u \cdot \nabla v + \int_\Omega (\vec{w} \cdot \nabla u) v = \int_\Omega f v \qquad \forall v \in X_0^N. \tag{3.10}$$

By splitting the solution $u = u_0 + u_D$ where $u_0 = 0$ on $\partial\Omega_D$ and u_D is a known function in X^N satisfying the Dirichlet boundary condition at nodes on the Dirichlet boundary, we can incorporate the Dirichlet boundary conditions in the right hand side, to yield:
Find $u \in X_0^N$ such that:

$$\varepsilon \int_\Omega \nabla u_0 \cdot \nabla v + \int_\Omega (\vec{w} \cdot \nabla u_0) v = \int_\Omega f_D v \qquad \forall v \in X_0^N. \tag{3.11}$$

Using the spectral element method, the basis for X^N is constructed 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 N degree Legendre polynomials π_N. The corresponding approximation space for the velocity and test functions is defined as

$$X^N = X \cap \mathbb{P}_{N,E}^2(\Omega) X_0^N = X_0 \cap \mathbb{P}_{N,E}^2(\Omega) \tag{3.12}$$

where

$$\mathbb{P}_{N,E}^2(\Omega) = \{v(x^e(r))|_{\Omega^e} \in \mathbb{P}_N(r_1) \otimes \mathbb{P}_N(r_2),\ e = 1, ..., E\} \tag{3.13}$$

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 $[-1, 1]$. Thus, on each element the solution is of the form

$$u(x,y) = \sum_{i=0}^{N} \sum_{j=0}^{N} u_{ij} \pi_{N,i}(x) \pi_{N,j}(y). \tag{3.14}$$

The coefficients u_{ij} correspond to the nodal values of u on the tensored Gauss-Legendre-Lobatto (GLL) points (defined in section 2.1). We write the solution as $u = u_0 + u_D$ where $u_0 = 0$ on $\partial\Omega_D$ and $u_D = u$ on $\partial\Omega_D$. Thus we can obtain the coefficients corresponding to u_0 by solving the linear system of equations

$$\underbrace{\mathbb{M}^e \Sigma' F^e(\vec{w}^e)}_{F} u^e = \underbrace{\mathbb{M}^e \Sigma' (M^e f^e - F^e(\vec{w}) u_D^e)}_{b}. \tag{3.15}$$

Here, a mask matrix $\mathbb{M}^e$ sets u^e to zero at all Dirichlet boundary nodes. Thus, after obtaining u_e on each element, this solution can be combined with u_D to form the discrete solution satisfying (3.1) and (3.2).

As we discussed in the previous chapter, inter-element connections are handled via direct stiffness summation Σ'. We note that this discrete system matrix is represented element-wise since Σ' is applied to sum all elemental contributions. Thus this is a matrix-free formulation in the sense that no global matrix is assembled. The discrete system is written in terms of element-defined operators such that on each element several discrete operators are applied. Namely, the discrete system involves a discrete representation of the identity in $L^2(\Omega)$, the mass matrix, to discretize the right hand side and a discrete convection-diffusion operator for the left hand side of (3.11).

The mass matrix is defined to be the block diagonal matrix $M = diag(M_e)$ where M_e is the local mass matrix on an element of size $h_x \times h_y$. M^e is represented via a tensor product of one-dimensional operators, $\hat{M}$, namely

$$M^e = \frac{h_x h_y}{4} \hat{M} \otimes \hat{M}. \tag{3.16}$$

Due to the orthogonality of the basis functions $\pi_{N,i}$ and $\pi_{N,j}$ the mass matrix is a diagonal matrix. The one-dimensional operator $\hat{M} = diag(\rho_i) \quad i = 1, ..., N+1$ where ρ_i is the GLL weight associated with the i^{th} GLL quadrature node ξ_i. The convection-diffusion operator F^e is represented on each element through this basis as

$$\begin{aligned}
F_x^e &= \underbrace{\varepsilon(\hat{M} \otimes \frac{h_y}{h_x}\hat{A})}_{\text{Diffusion in x}} + \underbrace{W_x^e(\hat{M} \otimes \frac{h_y}{2}\hat{C})}_{\text{Convection in x}} \\
F_y^e &= \underbrace{\varepsilon(\frac{h_x}{h_y}\hat{A} \otimes \hat{M})}_{\text{Diffusion in y}} + \underbrace{W_y^e(\frac{h_x}{2}\hat{C} \otimes \hat{M})}_{\text{Convection in y}} \qquad (3.17) \\
F^e(w^e) &= F_x^e + F_y^e
\end{aligned}$$

In (3.17), 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 $\hat{C} = \hat{M}\hat{D}$ 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 one-dimensional second derivative operator $\hat{A}$ is given by

$$\hat{A}_{ij} = \sum_{k=1}^{N+1} \hat{D}_{ki}\rho_k\hat{D}_{kj} \qquad i, j \in \{1, .., N+1\}^2, \qquad (3.18)$$

where $\hat{D}$ is the one-dimensional spectral differentiation operator

$$\hat{D}_{ij} = \frac{d\pi_j}{dr} |_{r=\xi_i} \qquad i, j \in \{1, .., N+1\}^2. \qquad (3.19)$$

The global system matrices can defined by combining the elemental matrices M^e and F^e across multiple elements via the connectivity matrix Q defined in section 2.3; that is $F(w) = \mathbb{M}^e Q^T F^e Q \mathbb{M}$ is the global non-symmetric convection-diffusion operator, and $M = \mathbb{M} Q^T M^e Q \mathbb{M}$ the global diagonal mass matrix.

The system in (3.15) can be solved using an iterative scheme, such as preconditioned GMRES [47]. 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. In section 3.5 we discuss how this domain decomposition scheme can be used as a preconditioner to accelerate convergence of non-constant wind convection-diffusion problems.

3.2 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 for constant coefficient convection-diffusion problems. Because we will use the system operator for constant coefficient problems later on, we denote the constant coefficient convection-diffusion operator by $\bar{F}$. The solution method obtains the discrete solution on a set of non-overlapping 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 are 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.

As discussed in the previous section, the domain is subdivided into E spectral elements, with the elemental interfaces being represented by a set Γ, and interior degrees of freedom represented by sets I^e. The resulting system of equations has the form

$$\begin{bmatrix} \bar{F}_{II}^1 & 0 & \dots & 0 & \bar{F}_{I\Gamma}^1 \\ 0 & \bar{F}_{II}^2 & 0 & \dots & \bar{F}_{I\Gamma}^2 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \bar{F}_{II}^E & \bar{F}_{I\Gamma}^E \\ \bar{F}_{\Gamma I}^1 & \bar{F}_{\Gamma I}^2 & \dots & \bar{F}_{I\Gamma}^E & \bar{F}_{\Gamma\Gamma} \end{bmatrix} \begin{pmatrix} u_{I^1} \\ u_{I^2} \\ \vdots \\ u_{I^E} \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} b_{I^1} - \bar{F}u_B|_{I^1} \\ b_{I^2} - \bar{F}u_B|_{I^2} \\ \vdots \\ b_{I^E} - \bar{F}u_B|_{I^E} \\ b_\Gamma - \bar{F}u_B|_\Gamma \end{pmatrix} = \begin{pmatrix} \hat{b}_{I^1} \\ \hat{b}_{I^2} \\ \vdots \\ \hat{b}_{I^E} \\ \hat{b}_\Gamma \end{pmatrix}. \quad (3.20)$$

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. As we mentioned in the last section, Dirichlet boundary conditions are implemented outside of the system operator in (3.20), by subtracting $\bar{F}u_B$ from the right hand side vector. $\bar{F}u_B$ denotes the full convection-diffusion system matrix applied to the Dirichlet boundary vector u_B.

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. To solve for u_Γ, the system matrix in (3.20) is formally factored as the product of lower and upper block matrices

$$\begin{bmatrix} I & 0 & \dots & 0 & 0 \\ 0 & I & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & I & 0 \\ \bar{F}_{\Gamma I}^1 \bar{F}_{II}^{1\,-1} & \bar{F}_{\Gamma I}^2 \bar{F}_{II}^{2\,-1} & \dots & \bar{F}_{\Gamma I}^E \bar{F}_{II}^{E\,-1} & I \end{bmatrix} \begin{bmatrix} \bar{F}_{II}^1 & 0 & \dots & 0 & \bar{F}_{I\Gamma}^1 \\ 0 & \bar{F}_{II}^2 & 0 & \dots & \bar{F}_{I\Gamma}^2 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \bar{F}_{II}^E & \bar{F}_{I\Gamma}^E \\ 0 & 0 & \dots & 0 & \bar{F}_S \end{bmatrix} \quad (3.21)$$

where $\bar{F}_S = \sum_{e=1}^E (\bar{F}_{\Gamma\Gamma}^e - \bar{F}_{\Gamma I}^e \bar{F}_{II}^{e\,-1} \bar{F}_{I\Gamma}^e)$ represents the Schur complement of the system.

By multiplying both sides of (3.20) with the inverse of the lower triangular matrix, one obtains the system

$$\begin{bmatrix} \bar{F}^1_{II} & 0 & \dots & 0 & \bar{F}^1_{I\Gamma} \\ 0 & \bar{F}^2_{II} & 0 & \dots & \bar{F}^2_{I\Gamma} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \bar{F}^E_{II} & \bar{F}^E_{I\Gamma} \\ 0 & 0 & \dots & 0 & \bar{F}_S \end{bmatrix} \begin{pmatrix} u_{I^1} \\ u_{I^2} \\ \vdots \\ u_{I^E} \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} \hat{b}_{I^1} \\ \hat{b}_{I^2} \\ \vdots \\ \hat{b}_{I^E} \\ g_\Gamma \end{pmatrix} \tag{3.22}$$

with $g_\Gamma = \sum_{e=1}^{E} (\hat{b}_{\Gamma^e} - \bar{F}^e_{\Gamma I} \bar{F}^{e\ -1}_{II} \hat{b}_{I^e})$. The interface variables are then obtained by solving

$$\bar{F}_S u_\Gamma = g_\Gamma, \tag{3.23}$$

and interior variables are obtained by solving

$$\bar{F}^e_{II} u_{I^e} = \hat{b}_{I^e} - \bar{F}^e_{I\Gamma} u^e_\Gamma \tag{3.24}$$

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

- Perform E subdomain solves to apply the action of $\bar{F}^e_{II}$ giving g_Γ,
- Perform interface solve for $\bar{F}_S u_\Gamma = g_\Gamma$,
- Perform E additional subdomain solves to apply the action of $\bar{F}^e_{II}$ yielding u_I.

The Schur complement operator can be constructed element-by-element by writing $\bar{F}_S = \sum \bar{F}^e_S$. This allows for efficient tensor-product based (see appendix (A-1)) 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 back into (3.22) to provide elemental boundary conditions for the interior solves. The interior variables

u_{I^e} are then obtained via element-wise fast diagonalization (FDM), which we discuss in section 3.3. We note that the action of $\bar{F}_{II}^{e\,-1}$ is also required for the matrix-vector products by the Schur complement operator, and in forming g_Γ. Each of these are also applied via FDM.

3.2.1 Interface Preconditioners

Non-overlapping domain decomposition methods are known to be effective for elliptic problems [45], [52], [55]. For such problems, the Neumann-Neumann preconditioner was developed to accelerate convergence of the Schur complement interface solve. This technique uses a pseudo-inverse of the locally defined Schur complement operator, with Neumann boundary conditions applied to elemental interfaces.

The preconditioning matrix derived using Neumann-Neumann conditions is given by

$$\sum_{e=1}^{E} D^{(e)} R_e^T \, (\bar{F}_S^e)^{-1} \, R_e D^{(e)}, \tag{3.25}$$

where $\bar{F}_S^e$ 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. The number of iterations required for the preconditioned system to converge is bounded by $\frac{C}{H}(1+log(N))^2$ ([55] p. 321) where C grows with the size of the Peclet number, N is the order of the spectral element basis functions, and H is the diameter of a typical element Ω_e. A simple two-domain case illustrates that with this choice, the preconditioned system is approximately a scaled identity when the subdomains are of similar size i.e. $\bar{F}_S^1 \approx \bar{F}_S^2$

$$(\bar{F}_S^{1\,-1} + \bar{F}_S^{2\,-1})(\bar{F}_S^1 + \bar{F}_S^2) = 2I + \bar{F}_S^{2\,-1}\bar{F}_S^1 + \bar{F}_S^{1\,-1}\bar{F}_S^2 = 2I + \bar{F}_S^{2\,-1}\bar{F}_S^1 + (\bar{F}_S^{2\,-1}\bar{F}_S^1).^{-1} \quad (3.26)$$

When applying the preconditioner, it is not necessary to form $\bar{F}_S^{(e)\,-1}$ on each element because $\bar{F}$ can be factored as

$$\bar{F}^e = \begin{pmatrix} I & 0 \\ \bar{F}_{\Gamma I}^e \bar{F}_{II}^{e\,-1} & I \end{pmatrix} \begin{pmatrix} \bar{F}_{II}^e & 0 \\ 0 & \bar{F}_S^e \end{pmatrix} \begin{pmatrix} I & \bar{F}_{II}^{e\,-1}\bar{F}_{I\Gamma}^e \\ 0 & I \end{pmatrix}. \tag{3.27}$$

This means that the inverse of $\bar{F}$ can be viewed as

$$\bar{F}^{e\,-1} = \begin{pmatrix} I & -\bar{F}_{II}^{e\,-1}\bar{F}_{I\Gamma}^e \\ 0 & I \end{pmatrix} \begin{pmatrix} \bar{F}_{II}^{e\,-1} & 0 \\ 0 & \bar{F}_S^{e\,-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -\bar{F}_{\Gamma I}^e \bar{F}_{II}^{e\,-1} & I \end{pmatrix}. \tag{3.28}$$

More importantly, we see that the action of the inverse of $\bar{F}_S^e$ can be applied to a vector by applying the inverse of $\bar{F}^e$ to the vector restricted to the elemental boundary, namely

$$\bar{F}_S^{(e)\,-1} v = (0 \quad I)\, \bar{F}_{NN}^{(e)\,-1} \begin{pmatrix} 0 \\ I \end{pmatrix} v \tag{3.29}$$

where

$$\bar{F}_{NN}^{(e)} = \begin{bmatrix} \bar{F}_{II}^e & \bar{F}_{I\Gamma}^e \\ \bar{F}_{\Gamma I}^e & \bar{F}_{\Gamma\Gamma}^e \end{bmatrix}. \tag{3.30}$$

When constructing $\bar{F}_{NN}^{(e)}$, Neumann boundary conditions are applied at elemental boundaries. This choice of boundary conditions produces a preconditioning operator corresponding to the bilinear form

$$a_e(u,v) = \int_{\Omega_e} (\varepsilon \nabla u \cdot \nabla v + (\vec{w} \cdot \nabla u)v), \tag{3.31}$$

which can be derived from the element-based Neumann problem

$$-\varepsilon\nabla^2 u + (\vec{w}\cdot\nabla)u = f \qquad \text{in } \Omega_e, \tag{3.32}$$

$$-\varepsilon\frac{\partial u}{\partial n} = 0 \text{ on } \Gamma_e. \tag{3.33}$$

However, as convection becomes dominant, these Neumann element interface conditions (3.33) become unstable [45], causing the bilinear form (3.31) to lose coercivity, and rendering the Neumann-Neumann method ineffective. Achdou et al. [1] extended the popular 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 the Neumann-Neumann preconditioner is modified using Robin boundary conditions 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. This choice of boundary conditions makes the preconditioning operator positive definite (see [1] & [55]) by defining the coercive bilinear form

$$a_e(u,v) = \int_{\Omega_e} (\varepsilon\nabla u\cdot\nabla v + (\vec{w}\cdot\nabla u)v) - \int_{\Gamma_i} \vec{w}\cdot\vec{n}uv, \tag{3.34}$$

which corresponds to the element-based Robin problem

$$-\varepsilon\nabla^2 u + (\vec{w}\cdot\nabla)u = f \qquad \text{in } \Omega_e, \tag{3.35}$$

$$-\varepsilon\frac{\partial u}{\partial n} + \vec{w}\cdot\vec{n}u = 0 \text{ on } \Gamma_e. \tag{3.36}$$

The difference between Neumann-Neumann and Robin-Robin preconditioners is the additional $\vec{w}\cdot\vec{n}u$ term at the element interfaces. This allows the flow to move between elements, and ensures the preconditioned system matrix is positive semi-definite, as described in [1] and [55]. The two preconditioners are equivalent when $\vec{w} = 0$.

The preconditioning matrix derived using these Robin conditions is defined just as for the Neumann-Neumann system,

$$\sum_{e=1}^{E} D^{(e)} R_e^T \left(\bar{F}_S^e\right)^{-1} R_e D^{(e)}. \tag{3.37}$$

We can also apply the action of the inverse of $\bar{F}_S^e$ to a vector by applying the inverse of $\bar{F}^e$ to the vector restricted to the elemental boundary, namely

$$\bar{F}_S^{(e)\,-1} v = (0 \quad I)\, \bar{F}_{RR}^{(e)\,-1} \begin{pmatrix} 0 \\ I \end{pmatrix} v \tag{3.38}$$

where the Neumann-Neumann operator F_{NN} is replaced by the Robin-Robin operator

$$\bar{F}_{RR}^{(e)} = \begin{bmatrix} \bar{F}_{II}^e & \bar{F}_{I\Gamma}^e \\ \bar{F}_{\Gamma I}^e & \bar{F}_{\Gamma\Gamma}^e \end{bmatrix} \tag{3.39}$$

which keeps the preconditioning matrix positive definite by applying Robin Boundary conditions at element inflows, and Neumann boundary conditions where the flow is orthogonal to the element boundary as well as along outflow boundaries. 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 equation (3.17) by a single entry corresponding to the inflow condition

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

In section 3.4 we provide empirical results comparing the Neumann-Neumann and Robin-Robin preconditioning schemes using our solution method. In the next section we introduce the fast diagonalization method, which we use to eliminate interior degrees of freedom.

3.3 Fast Diagonalization Method (FDM)

The Fast Diagonalization Method (FDM) was originally constructed to solve problems arising from tensor-product based finite difference discretizations of symmetric constant coefficient partial differential equations. We define the FDM as a generalization of the method described in [32], and apply it to non-symmetric convection-diffusion systems with constant wind $\vec{w}$. We then examine the use of FDM applied to (3.15) 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. The FDM allows for the solutions of systems in which the coefficient is of order n^d and has a tensor product structure in $O(n^{d+1})$ operations, where d represents the number of spatial dimensions and n represents the number of grid points used along each dimension on a single element.

Consider equation (3.17) 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 be written as

$$F^e(c_x, c_y) = \hat{M} \otimes \hat{F}_x + \hat{F}_y \otimes \hat{M} =: \bar{F}^e. \tag{3.40}$$

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 $\bar{F}^e = \tilde{M}^{1/2}\tilde{F}^e\tilde{M}^{1/2}$ where

$$\begin{aligned}
\tilde{F}^e &= \tilde{M}^{-1/2}\bar{F}^e\tilde{M}^{-1/2} \qquad (3.41)\\
&= (\hat{M}^{-1/2} \otimes \hat{M}^{-1/2})(\hat{M} \otimes \hat{F}_x + \hat{F}_y \otimes \hat{M})(\hat{M}^{-1/2} \otimes \hat{M}^{-1/2})\\
&= (I \otimes \hat{M}^{-1/2}\hat{F}_x\hat{M}^{-1/2}) + (\hat{M}^{-1/2}\hat{F}_y\hat{M}^{-1/2} \otimes I)\\
&= (I \otimes B) + (A \otimes I).
\end{aligned}$$

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

$$\tilde{F}^e = (S \otimes T)(\Lambda_y \otimes I + I \otimes \Lambda_x)(S^{-1} \otimes T^{-1}) \tag{3.42}$$

so that

$$\tilde{F}^{e\,-1} = (S \otimes T)(\Lambda_y \otimes I + I \otimes \Lambda_x)^{-1}(S^{-1} \otimes T^{-1}). \tag{3.43}$$

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

$$\bar{F}^{e\,-1} = (\hat{M}^{-1/2} \otimes \hat{M}^{-1/2})(S \otimes T)(\Lambda_y \otimes I + I \otimes \Lambda_x)^{-1}(S^{-1} \otimes T^{-1})(\hat{M}^{-1/2} \otimes \hat{M}^{-1/2}). \tag{3.44}$$

This formulation 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 the following sections.

We use this method to apply the action of $\bar{F}_{II}^{e\ -1}$ for each element in the domain decomposition method described section 3.2. In flows where the wind coefficient $\vec{w}$ is constant, $\bar{F}^{-1}$ constitutes a direct solver to equation (3.15), in practice however, an iteration is needed to obtain the solution to the Schur complement system (3.23) on the union of element interfaces. We demonstrate the use of this methodology for solving constant coefficient problems in two examples in the next section. We then describe in section 3.5 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 (3.23).

We also mention here that it is possible to extend the Fast Diagonalization Method to convection-diffusion systems with separable winds i.e. $\vec{w} = (w(x), w(y)) = (I \otimes \hat{w}_x, \hat{w}_y \otimes I)$ on each element. The diagonalization method above remains the same except the one-dimensional matrices that are diagonalized are slightly modified. Namely,

$$B = \hat{M}^{-1/2}(\varepsilon \frac{h_y}{h_x} \hat{A} + \frac{h_y}{2} \hat{w}_x \hat{C}) \hat{M}^{-1/2} \tag{3.45}$$

and

$$A = \hat{M}^{-1/2}(\varepsilon \frac{h_x}{h_y} \hat{A} + \frac{h_x}{2} \hat{w}_y \hat{C}) \hat{M}^{-1/2}. \tag{3.46}$$

This diagonalization extension may be useful in some certain flow models, and perhaps allow for more accurate local flow approximations on each element, as an extension to the constant wind approximation we develop later. This is an area we will explore in future research.

3.4 Solution of Problems with Constant Wind

In this section we demonstrate the effectiveness of our solvers applied to problems with constant coefficients.

3.4.1 Analytical solution with outflow boundary layer

We first consider a problem whose solution exhibits dramatic change near the outflow boundary $y = 1$. This problem has an analytical solution

$$u(x,y) = x\left(\frac{1-e^{(y-1)/\varepsilon}}{1-e^{-2/\varepsilon}}\right). \tag{3.47}$$

Our focus here is on the iterations for the interface solution to (3.23). In addition to these interface iterations, a solution on the element interiors is obtained via Fast Diagonalization. We show a plot of the computed solution, and contours in Figure 3.1 using a spectral element discretization with 2 elements in each dimension, and polynomial degree 16 on each element. There are roughly five times fewer interface nodes than there are interior nodes in this particular discretization. One may choose to think of this solution method as a direct method where an iterative method is used to obtain a small number of degrees of freedom at element interfaces. This means the Fast Diagonalization Method is efficiently eliminating a large portion of the degrees of freedom directly. This reduction in the size of the iterative system coupled with the low number of iterations required to obtain an interface solution significantly reduces the number of operations to obtain an accurate solution.

Tables 3.1 and 3.2 contain several items of data associated with the experiments for problem 3.4.1. In Table 3.1 we demonstrate the exponential convergence (see Def-

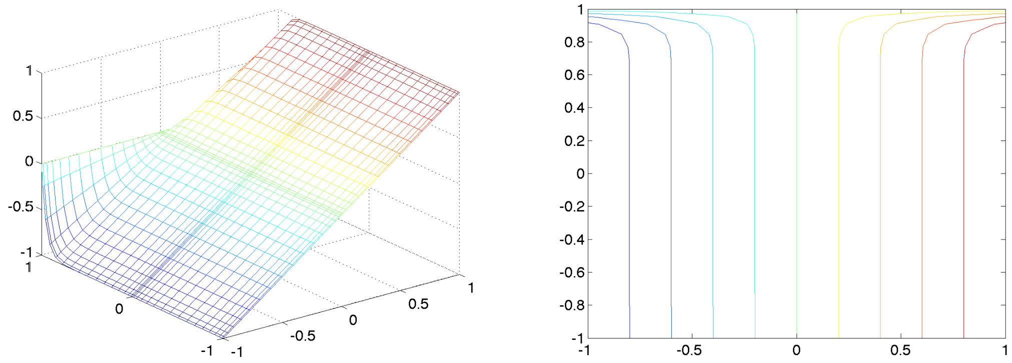

Figure 3.1: Computed solution and contours of steady convection diffusion flow corresponding to example 3.4.1 with constant wind $\vec{w} = (0,1)$ and moderate convection $Pe = 40$.

inition 2) for this example problem with moderate convection $Pe = 40$. We use a fixed number of elements E=2 × 2 as the polynomial degree is varied from N=4 through 32. In the second column of the table, we see that the solution converges exponentially as N is increased. The tables compare convergence results for the iterative solvers used to obtain the interface solution using varied preconditioning strategies. The last three columns correspond to the number of interface iterations required for GMRES without preconditioning (None), GMRES with Neumann-Neumann preconditioning (N-N), GMRES with Robin-Robin preconditioning (R-R), and the upper bound on the number of iterations for (N-N) and (R-R) preconditioned GMRES to converge without the scaling constant. The performance of each iterative method depends on the increased grid resolution, and the preconditioned methods compares well with theory. We note that for this test problem with moderate convection and few sub-domains, (N-N) and (R-R) converge in roughly the same number of steps.

In Table 3.2 we demonstrate the algebraic convergence (see Definition 2) for this

N	$\|u-u_N\|_2$	None	N-N	R-R	$\frac{1}{h}(1+log(N))^2$
4	5.535×10^{-2}	3	3	3	5.6
8	2.505×10^{-3}	7	7	7	9.4
16	2.423×10^{-7}	15	11	14	14.2
32	7.931×10^{-13}	30	16	18	19.9

Table 3.1: Convergence results for example 3.4.1 with a moderate convection $Pe = 40$, as polynomial degree is varied on a fixed 2×2 element grid.

example problem using with the same Peclet number $Pe = 40$. We fix polynomial degree at N=2, and increase the number of elements from E=4 (as a 2×2 grid) to E=1024 (32×32 grid). In the second column, we see that the solution converges algebraically as E is increased. These numbers also compare well with the expected cubic convergence rate for quadratic elements ($N = 2$). As in the previous table, we also compare convergence results for the iterative solvers used to obtain the interface solution. The last four columns correspond to the number of interface iterations required for GMRES without preconditioning (None), GMRES with Neumann-Neumann preconditioning (N-N), GMRES with Robin-Robin preconditioning (R-R), and the upper bound on the number of iterations for (N-N) (R-R) preconditioned GMRES to converge without the scaling constant. Each iterative method depends on the increased grid resolution. However, unlike the previous case with few sub-domains the Neumann-Neumann method appears to be negatively influenced by convection. The performance of the Robin-Robin method compares well with theory and beats the other two methods, especially as E becomes large.

We note that in both Tables 3.1 and 3.2, the Robin-Robin preconditioner provides marginal improvement for coarse grid simulations, whereas in the finer resolution simulations Robin-Robin provides significant savings. Next we examine how changes in the

E	$\|u-u_N\|_2$	$13h^3$	None	N-N	R-R	$\frac{1}{h}(1+log(N))^2$
16	8.594×10^{-2}	2.031×10^{-1}	13	13	12	11.5
64	2.593×10^{-2}	2.523×10^{-2}	49	47	25	22.9
256	3.558×10^{-3}	3.174×10^{-3}	108	88	45	45.9
1024	3.610×10^{-4}	3.967×10^{-4}	312	180	85	91.7

Table 3.2: Convergence results for example 3.4.1 with moderate convection Pe=40, as the number of quadratic (N=2) elements are varied.

Peclet number affects the convergence of this method.

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 $E = 256$ (16×16 element grid) to perform these experiments. Table 3.5 we compare convergence results for the iterative solvers used to obtain the interface solution. The first column lists the Peclet number and the last three columns correspond to the number of interface iterations required for GMRES without preconditioning (None), GMRES with Neumann-Neumann preconditioning (N-N), GMRES with Robin-Robin preconditioning (R-R). We see that the Neumann-Neumann preconditioner (N-N) is in-effective, taking more steps than with no preconditioner (None), while the Robin-Robin preconditioner (R-R) is virtually unaffected by changes in the Peclet number. Indeed, as the Peclet number is increased through the first several cases the Robin-Robin preconditioner actually improves. For higher Peclet numbers, these Robin-Robin iterations increase mildly with the Peclet number. It is evident from this study that the Robin-Robin preconditioner is effective for high Peclet number flows. In the next example we perform a similar analysis using a flow that is not aligned with the grid, but rather moves at an angle to see how such a flow effects the convergence of this scheme.

Pe	None	N-N	R-R
125	85	92	35
250	79	98	30
500	81	119	25
1000	103	164	27
2000	160	> 200	38
5000	> 200	> 200	74

Table 3.3: Comparison of iteration counts for example 3.4.1 with increasingly convection-dominated flows. N=8, E=256 using 16 × 16 element grid.

3.4.2 Oblique wind with internal and outflow boundary layers

In our second example we demonstrate the effectiveness as the convection becomes increasingly dominant. We use a flow that exhibits an internal boundary layer that results from a jump discontinuity in the inflow region of the boundary at $(0,-1)$ (see Figure 3.2). Dirichlet boundary conditions are imposed along each boundary as seen in Figure 3.2. Along the back boundary the velocity is set to zero causing a characteristic boundary layer proportional to ε. The internal boundary layer is proportional to $\sqrt{\varepsilon}$. The wind field in this test case is constant $\vec{w} = (-sin(\pi/6), cos(\pi/6))$, but unlike the previous example, the wind is not aligned with the grid.

In the left and right Tables in 3.4.2 we report several items of data associated with the experiments for problem 3.4.2. In the left table 3.4.2 we use a fixed number of elements E=4 as a 2 × 2 element grid and vary the degree of the polynomial basis from N=4 through 32, whereas in the right table we fix polynomial degree at N=2, and increase the number of elements from E=4 (as a 2 × 2 grid) to E=1024 (32 × 32 grid).

The tables compare convergence results for the iterative solvers used to obtain the interface solution using varied preconditioning strategies. The last three columns corre-

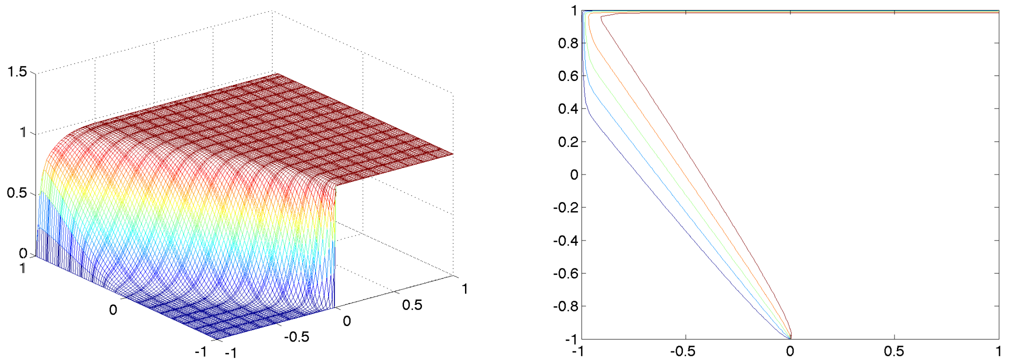

Figure 3.2: Velocity (left) and contours (right) of a convection-dominated steady convection-diffusion flow, *Pe* $= 250$, corresponding to example 3.4.2.

spond to the number of interface iterations required for GMRES without preconditioning (None), GMRES with Neumann-Neumann preconditioning (N-N), GMRES with Robin-Robin preconditioning (R-R). In the right table we see that (N-N) and (R-R) converge in roughly the same number of steps when few subdomains are used, although the Robin-Robin method is superior. Additionally we observe that as the number of elements increase in the right table, the Robin-Robin preconditioner fairs far better than no preconditioner and Neumann-Neumann preconditioning. In the next study, we consider higher Peclet number flows, and see that indeed the Robin-Robin method requires far fewer iterations than these other two methods for convective flows.

N	None	N-N	R-R
4	13	13	13
8	25	25	18
16	36	28	20
32	50	29	21

E	None	N-N	R-R
16	29	33	21
64	40	63	26
256	69	117	46
1024	132	> 200	87

Table 3.4: Convergence results for example 3.4.2 with moderate convection Pe=40, as polynomial degree is varied on a fixed 2×2 element grid (right), and as the number of quadratic (N=2) elements are varied on element grids ranging from 4×4 to 32×32.

In Table 3.5 we describe how the number of iterations of the Schur compliment system depends on the size of the Peclet number. For these calculations, we use a fixed grid with polynomial degree $N = 8$, and 256 elements (16 in each direction). We can see that the Robin-Robin (R-R) preconditioner performs significantly better than no preconditioner (None) and the Neumann-Neumann preconditioner (N-N). Comparing the iteration counts in the last row in the two tables, one can see that for highly convective flows, $Pe \propto \mathrm{O}(1000)$, flow resolution becomes an important factor in the iteration counts. In the left table, the spacing between the grid points near the characteristic boundary condition is $\mathrm{O}(10^{-2})$ where the width of the boundary layer ranges from $\mathrm{O}(10^{-3}) - \mathrm{O}(10^{-4})$. The spacing near the characteristic boundary condition for the refined grid $\mathrm{O}(10^{-3})$, and thus provides more accurate view of how the preconditioner performs for resolved flows. Indeed, the only flow that is fully resolved in this study is for $Pe = 125$ in the right table. Thus the observed dependence on the Peclet number for highly convective flows may actually be attributed to an under-resolved flow.

Pe	None	N-N	R-R
125	93	104	38
250	75	98	32
500	64	115	27
1000	69	150	30
2000	99	> 200	41
5000	> 200	> 200	94

Table 3.5: Comparison of iteration counts for example 3.4.2 with increasingly convection-dominated flows. N=8, E=256 using 16×16 element grid.

In each of these studies the number of iterations is comparable to those in the previous test case. We conclude that Robin-Robin preconditioner performs well for both grid-aligned and non-grid 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 [52]. We observed that when using the Robin-Robin preconditioner, the number of iterations required for $\bar{F}_S^{-1}$ 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 (3.15). This domain decomposition strategy allows one to solve highly convective systems where GMRES would fail 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 $E = 16 \times 16$, there are roughly eight times fewer 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 non-symmetric systems solved using GMRES this is very important since one must hold all computed iterates in memory to obtain an orthogonal search direction at each iterate.

3.5 Non-Constant Wind Systems

When the wind vector $\vec{w}$ is not constant in each component, then 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 FGMRES for solving discrete convection-diffusion systems (3.15) arising from non-constant winds.

In the case of non-constant wind, the matrix $F(\vec{w})$ from equation (3.17) cannot be written in the tensor product form (3.40) 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. In particular, we consider the approximation to $F(\vec{w})$ where the wind vector $\vec{w}$ is taken 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 $\bar{F}$.

The idea of this approximation is to incorporate the average wind component in each element in order provide a rough model for flow in each element. In essence, this approximation follows the average flow in each element. Elman and Chernesky [15] show that applying block Gauss-Seidel line relaxation to node orderings that follow the flow result in rapid convergence in one-dimensional convection-diffusion systems. Furthermore, Elman et al. proposed using multiple Gauss-Seidel "sweeps" to follow the flow in each direction in order to reduce the number of iterations for multi-dimensional convection-diffusion systems [19]. We take a similar point of view here using $\bar{F}$ to precondition the non-constant wind system, (3.15), we hope to expedite the convergence of GMRES.

In order to apply the preconditioner $\bar{F}$ for multiple elements, we follow the domain decomposition strategy developed in section 3.2. 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 non-constant 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 (3.23). 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 $\bar{F}$ makes $\bar{F}$ a non-linear operator. This non-linear preconditioning operator causes trouble with traditional GMRES methods that require the preconditioner to be the same at each step. An alternative GMRES method, called Flexible GMRES (FGMRES) was developed by Saad in [47]. This method allows for arbitrary changes in the preconditioner at each step of the iteration. Thus FGMRES can be used to solve the preconditioned system

$$\underbrace{\mathbb{M}^e\Sigma' F^e(\vec{w})}_{F}\bar{F}^{-1}\bar{F}u^e \quad = \quad \underbrace{\mathbb{M}^e\Sigma'(M^e f^e - F^e(\vec{w})u_b^e)}_{b}. \tag{3.48}$$

In summary, to solve convection-diffusion problems with non-constant wind we use Flexible GMRES (FGMRES). 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 (3.23). This inner iteration is coupled with FDM to obtain interior degrees of freedom on each element. We demonstrate the use of this solution method for non-constant wind convection-diffusion problems in the following example.

3.5.1 Double Glazing Problem

We examine our non-constant 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 (Figure 3.3).

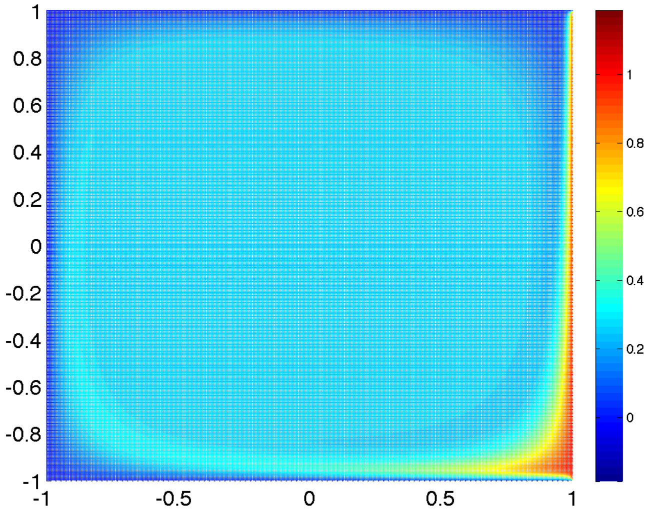

Figure 3.3: Top view of solution for example 3.5.1 with $Pe = 2000$.

The left pane of Figure 3.4 shows the computed solution for an example flow with $Pe = 400$ using FGMRES, and the right pane shows an approximate solution obtained by replacing the coefficient matrix with the preconditioning matrix $\bar{F}$. In the approximate solution we point out the constant wind approximation produces an error in regions where the convection field (shown in Figure 3.5) changes direction, such as at the corners and around the internal boundary layer.

Since our preconditioner is based on a local wind approximation, we reiterate that it may not be necessary to apply the preconditioner's interface solve exactly to obtain good results. We compare the $\bar{F}$ preconditioning method to GMRES without preconditioning and using a block Jacobi preconditioned GMRES. To construct the block Jacobi preconditioner we use the diagonal of $\bar{F}$. So on each element the interior nodes are obtained by solving a system using the diagonal of $\bar{F}^e_{II}$, and the boundary nodes are obtained by

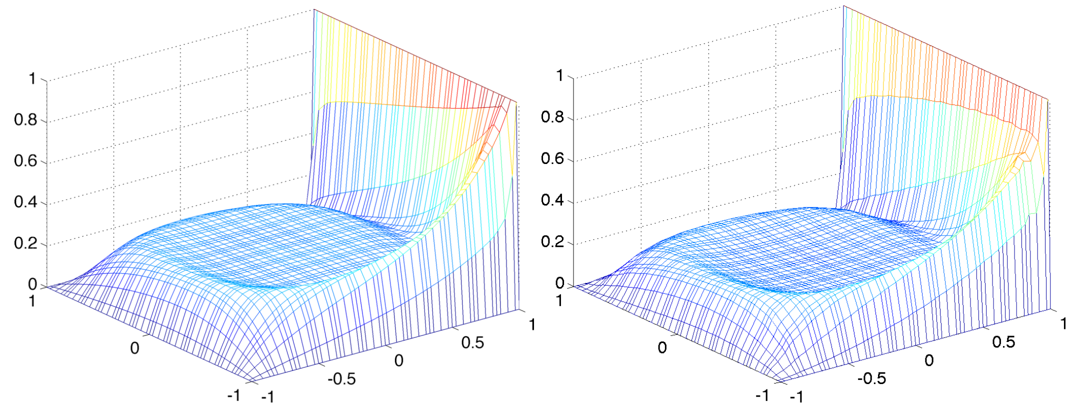

Figure 3.4: Comparison of exact solution (left) and inexact solution (right) obtained by applying $\bar{F}^{-1}$ as an inexact solver for example 3.5.1 with $Pe = 400$.

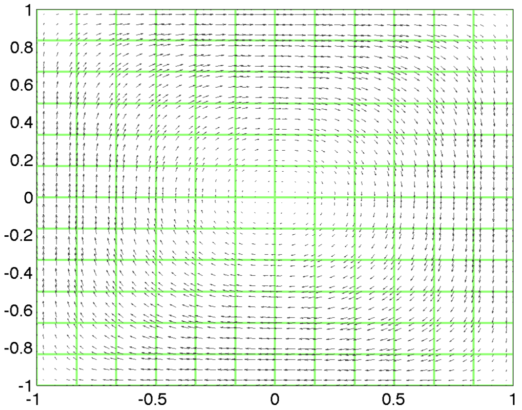

Figure 3.5: Wind field for example 3.5.1 with element overlay.

solving a system using the diagonal of $\bar{F}^{e}_{BB}$.

Figure 3.6 shows the number of FGMRES iterations required to solve (3.48) to a tolerance of 10^{-12}. The top two curves show that non-preconditioned GMRES and block Jacobi preconditioned GMRES are ineffective at solving this system as both fail to reduce the residual after 400 iterations. 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 curves show the influence of accuracy of the Schur complement solve $\bar{F}_S^{-1}$ on the performance

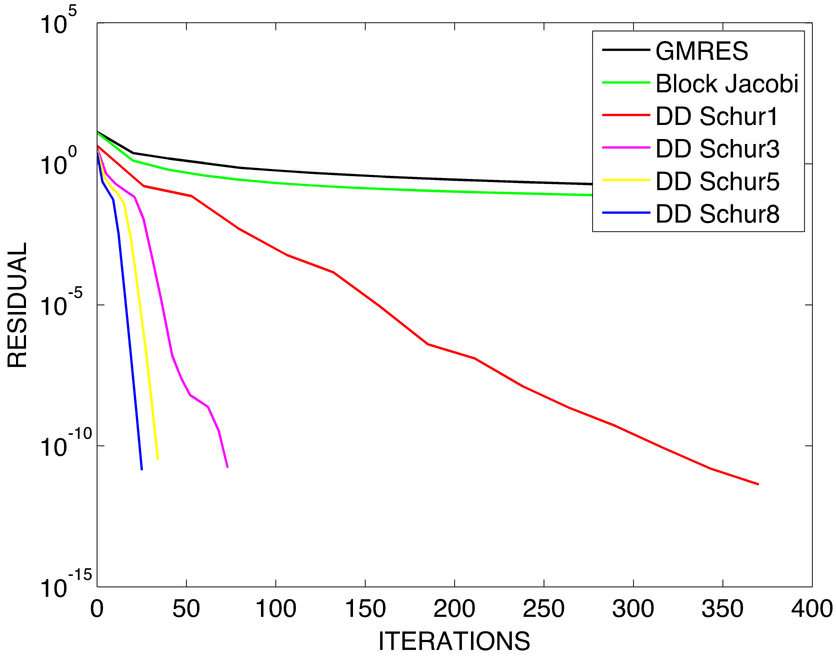

Figure 3.6: Comparison of preconditioned Flexible GMRES iterations with varied inner iteration counts, Block Jacobi iterations and non-preconditioned GMRES iterations for solving equation (3.15).

of the outer FGMRES iteration. The red curve shows us 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 370 steps. In the opposite extreme, the blue 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 22 steps. It is evident from Figure 3.6 that only a few inner iterations for the preconditioner are required in order for the outer iteration to converge quickly.

Note that the results from 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 the conclusion is somewhat different. In Figure 3.6 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 Figure 3.7 shows the residual during the first 40 steps of the interface solve using GMRES without preconditioning (red) 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 stagnating. The Robin-Robin preconditioned solver further reduces the residual around the 25^{th} iteration, while the non-preconditioned system shows no improvement. Achdou et al. point out in [1] that in convection-dominated flows, the continuous Robin-Robin preconditioned system operator is close to an idempotent (or periodic) operator of order $S/2$ where S is the number of elements. They argue that this causes GMRES to stagnate for $S/2$ steps before converging asymptotically [1]. However, the right pane of Figure 3.7 shows that the residual of the outer FGMRES iteration is essentially unaffected by the difference these two inner iteration residuals. The red curve represents the residual of the outer FGMRES iteration using unpreconditioned GMRES for the inner iteration. In comparison, the blue 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 (blue curve) converges slightly faster (by 3 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.

[1] Elman and Chernesky [15] 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.

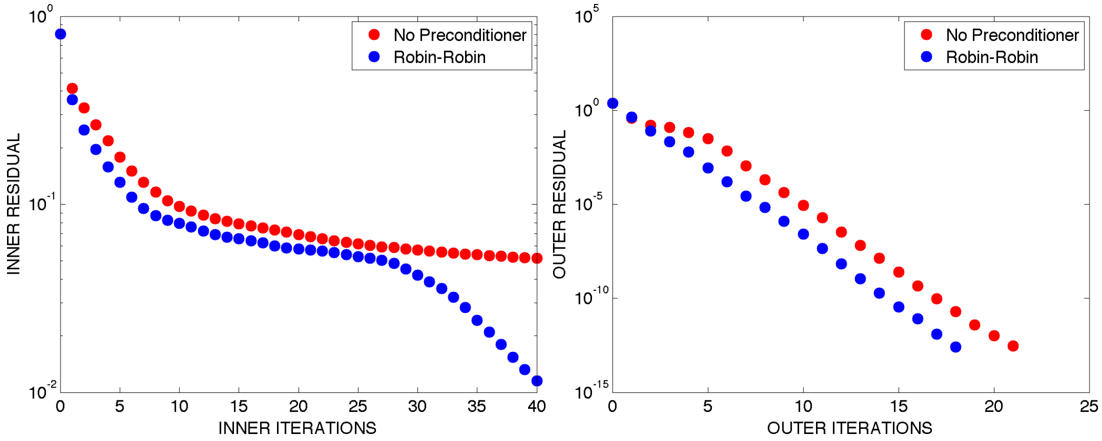

Figure 3.7: Comparison of residuals for interface iterations obtained by GMRES without preconditioning and with Robin-Robin preconditioning (left). Affect on FGMRES residuals with inexact $\bar{F}^{-1}$ using no interface preconditioner and Robin Robin (right).

Next we look at how the number of FGMRES iterations are influenced by mesh refinement. In Tables 3.6 and 3.7 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}. We see that outer iterations for both p-refinement (left table middle column) and h-refinement (right table middle column) are roughly constant, whereas the inner iterations (right columns) require show a mild dependence on the mesh size. Finally, we consider the dependence of the Peclet number on our solution method.

N	Number of FGMRES Outer Iterations	Number of Inner Iterations
4	40	5
8	51	5
16	44	13
32	48	20

Table 3.6: Iteration counts for example 3.5.1 as polynomial degree is increased with $Pe = 400$ and $E = 16$.

E	Number of FGMRES Outer Iterations	Number of Inner Iterations
16	40	5
64	25	12
256	17	19
1024	28	20

Table 3.7: Iteration counts for example 3.5.1 as the number of elements are increased with $Pe = 400$ and $N = 4$.

Pe	Number of FGMRES Outer Iterations	Number of Inner Iterations
125	14	20
250	16	16
500	19	18
1000	24	18
2000	34	16
5000	67	13

Table 3.8: Iteration counts for example 3.5.1 as the Peclet number is increased on a fixed grid with $N = 8$ and $E = 256$.

In Table 3.8 we show how the number of FGMRES iterations are affected as the Peclet is increased from 125 up to 5000. In this study we use a grid where $N = 8$ and $E = 256$ on a 16×16 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) appear independent of the Peclet number.

3.6 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 in order 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.

We then developed a solver for non-constant 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 non-constant wind solution strategy showed significant improvement over non-preconditioned GMRES and the traditional Block-Jacobi preconditioning technique. Using $\bar{F}$ as a preconditioner allowed FGMRES to obtain convergence rates independent of the mesh size, and mildly dependent of changes in convection strength.

Chapter 4

Navier-Stokes Equations

In this chapter, we build on the results obtained in Chapter 3 for the convection-diffusion equation to construct an efficient solution strategy for the steady Navier-Stokes equations. The Navier-Stokes equations describe the motion of incompressible Newtonian fluids such as water, oil or blood. The steady Navier-Stokes equations can be written as

$$\begin{aligned} -\nu\nabla^2\vec{u} + (\vec{u}\cdot\nabla)\vec{u} + \nabla p &= f \quad \text{in} \quad \Omega, \\ \nabla\cdot\vec{u} &= 0 \quad \text{in} \quad \Omega, \end{aligned} \tag{4.1}$$

where u is the velocity, p is the pressure and ν is the kinematic viscosity of the fluid. In addition to (4.1) , we impose the following boundary conditions

$$u = u_b \text{ on } \partial\Omega_D, \qquad \nabla u_i \cdot \vec{n} = 0 \text{ on } \partial\Omega_N, \tag{4.2}$$

where $\vec{n}$ is the outward facing normal on the boundary, and subscripts D and N denote Dirichlet (wall and inflow) and Neumann (outflow) boundary regions respectively. As in the convection-diffusion equation, we can measure the relative contributions of convection and diffusion by normalizing equation (4.1) by characteristic velocity and length scales, U and L respectively. Thus, points in Ω can be normalized by dividing by L, similarly, the velocity $u^* := u/U$, pressure $p^* := p/U^2$, and forces $f^* := fL/U^2$ in the normalized domain can be described by the equation

$$\begin{aligned} -\tfrac{1}{Re}\nabla^2 u^* + (\vec{u}^*\cdot\nabla)u^* + \nabla p^* &= f^* \quad \text{in} \quad \Omega^* \\ \nabla\cdot\vec{u}^* &= 0 \quad \text{in} \quad \Omega^*. \end{aligned} \tag{4.3}$$

The quantity $Re := \frac{UL}{\nu}$ is termed the *Reynolds number*. This dimensionless number quantifies the contributions of convection and diffusion for a given flow. In diffusion dominated flows, $Re \leq 1$, whereas in convection dominated flows $Re \gg 1$.

For smaller *Re*, steady flow states may exist, but as *Re* becomes larger, flows evolve to turbulent states, often undergoing transitions to time-periodic states, and fully unsteady states before eventually becoming turbulent. Determining the critical Reynolds numbers where these changes occur, and understanding how a given solution becomes unstable allows one to characterize the behavior of the flow as well as provide insight into how physical parameters affect flow states. Analytical methods such as global stability analysis [30], transient growth analysis [49], [56] and nonlinear stability analysis [8] rely on steady-state solutions to determine the behavior of flows near transitions. Thus, solvers aimed at accurately and efficiently computing steady flow states are of particular relevance to advancing our understanding of both steady and unsteady flows.

Unlike the convection-diffusion equation, the Navier-Stokes equations are nonlinear, since the convection term $(\vec{u} \cdot \nabla)$ depends on the flow velocity $\vec{u}$ instead of a prescribed wind $\vec{w}$. So, in order to solve the Navier-Stokes equations, a nonlinear solver must be used. We use the Picard nonlinear iteration scheme [41] in order to accomplish this. This technique produces a system of linearized equations that must be solved at each step. The efficiency of the nonlinear solution method depends greatly on the cost of the subsidiary linear solves. Our goal is to reduce the cost of these subsidiary linear system solves via preconditioning. To do this, we combine the preconditioner developed in the previous chapter with a block preconditioner developed by Elman et al. in [17].

The remainder of this chapter is organized as follows. In section 4.1 we use the

Spectral Element Method to derive the discrete form of the Navier-Stokes equations. There, we also construct one-dimensional operators used in applying the corresponding matrix-free system. Section 4.2 outlines the Picard and Newton iteration methods used to solve nonlinear systems of equations. We explain how each method linearizes the Navier-Stokes equations to produce the block linear systems we focus on solving. In section 4.3 we provide a brief overview of a block preconditioner developed in [17] for accelerating the convergence of Krylov subspace methods used to solve the linearized Navier-Stokes equations. We also explain how the convection-diffusion preconditioner developed in the previous chapter can be used inside this block preconditioning framework. Finally in section 4.4 we study how the number of linear iterations at each nonlinear iteration is influenced by changes in the mesh and Reynolds number. We also provide analysis that demonstrates the discrete solution obtained from this solver converges exponentially as the degree of the polynomial basis is refined on a given element discretization.

4.1 Spectral Element Method applied to the Navier-Stokes Equations

In Chapter 2 we introduced the spectral element method as a Galerkin method derived from the method of weighed residuals. Here we explain how the spectral element method is applied to the steady Navier-Stokes equations. We start by constructing the weak formulation of the steady Navier-Stokes equations

Find $u, v \in X$ and $p, q \in Y$ such that

$$\begin{aligned} \nu \int_\Omega \nabla\vec{u} : \nabla\vec{v} + \int_\Omega (\vec{u}\cdot\nabla\vec{u})\cdot\vec{v} - \int_\Omega p(\nabla\cdot\vec{v}) &= \int_\Omega f\cdot\vec{v} \quad \forall \quad \vec{v}\in X_0 \\ -\int_\Omega q(\nabla\cdot\vec{u}) &= \quad 0 \quad \forall \quad q\in Y \end{aligned} \tag{4.4}$$

where the test and trial spaces are defined as

$$\begin{aligned} X &= \{\vec{u}\in H^1(\Omega)^2 | \vec{u}=\vec{w} \quad \text{on} \quad \partial\Omega_D\}, \\ X_0 &= \{\vec{v}\in H^1(\Omega)^2 | \vec{v}=0 \quad \text{on} \quad \partial\Omega_D\}, \\ Y &= L^2(\Omega). \end{aligned}$$

$L^2(\Omega)$ is 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 $L^2(\Omega)$, $\nabla\vec{u} : \nabla\vec{v} := \nabla u_x \cdot \nabla v_x + \nabla u_y \cdot \nabla v_y$ denotes the component-wise scalar product, and the component-wise inner product $\int_\Omega \vec{u}\cdot\vec{v}$ is defined in Definition 4.

Existence and uniqueness of a corresponding weak solution can be demonstrated by establishing the coercivity (see definition 6) and continuity (see definition 7) of the bilinear form

$$a_{\vec{z}} = \nu \int_\Omega \nabla\vec{u} : \nabla\vec{v} + c(\vec{z};\vec{u},\vec{v}) \tag{4.5}$$

where the convection term is written as a trilinear form $c : X_0 \times X_0 \times X_0 \rightarrow \mathbb{R}$ defined as $c(\vec{z};\vec{u},\vec{v}) := \int_\Omega (\vec{z}\cdot\nabla\vec{u})\cdot\vec{v}$. Coercivity can be established by observing that this convection term is skew-symmetric so $c(\vec{z};\vec{u},\vec{u}) = 0$, and thus $a_{\vec{z}}(\vec{u},\vec{u}) = \nu\|\nabla\vec{u}\|^2$ for all admissible $\vec{u}$. Continuity holds if ν is not too small, however, demonstrating this result is more involved and we refer the reader to [19] and [23] for details.

Next, we restrict u, v, p and q to compatible finite dimensional velocity and pressure subspaces, X^N and Y^N. This yields the form

Definition 4 $L_2(\Omega)$ component-wise inner product

Let $L_2(\Omega)$ be the space of all square-integrable scalar-valued functions on Ω with associated scalar inner product $(\cdot,\cdot)_*$. The *component-wise inner product* of the vector fields $\vec{u} = (u_x, u_y)$, $\vec{v} = (v_x, v_y)$, denoted $(\vec{u}, \vec{v}) := \int_\Omega \vec{u} \cdot \vec{v} = (u_x, v_x)_* + (u_y, v_y)_*$ (see [19] p. 36).

Definition 5 $L_2(\Omega)$ norm

Let $L_2(\Omega)$ be the set of all square-integrable scalar-valued functions over Ω with associated component-wise inner product $(\cdot,\cdot)$. The L_2 *norm* of the vector field $\vec{u} = (u_x, u_y)$, denoted $||\vec{u}|| = (\vec{u}, \vec{u})^{1/2}$ (see [19] p. 36).

Definition 6 Coercivity

A bilinear form $a(\cdot,\cdot)$ is *coercive* with respect to the norm $||\cdot||_V$ in a space V if there is a positive constant c such that $a(u,u) \geq c||u||_V^2 \quad \forall u \in V$ (see [19] p. 121).

Definition 7 Continuity

A bilinear form $a(\cdot,\cdot)$ is *continuous* with respect to the norm $||\cdot||_W$ in a space W if there is a positive constant C such that $a(u,v) \leq c||u||_W ||v||_W \quad \forall u, v \in W$ (see [19] p. 121).

Find $u \in X^N, p \in Y^N$ such that:

$$\begin{aligned} \nu \int_\Omega \nabla \vec{u} : \nabla \vec{v} + \int_\Omega (\vec{u} \cdot \nabla \vec{u}) \cdot \vec{v} - \int_\Omega p(\nabla \cdot \vec{v}) &= \int_\Omega f \cdot \vec{v} \, \forall \, \vec{v} \in X_0^N \\ - \int_\Omega q(\nabla \cdot \vec{u}) &= 0 \, \forall \, q \in Y^N \end{aligned}$$

The bases for X^N and Y^N are constructed 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 simplest choice would be to choose the same polynomial basis for both fields. However, this approach, known as the $\mathbb{P}_N - \mathbb{P}_N$ method, does not satisfy the Inf-Sup stability condition (see Definition 8), and thus pressure solution exhibits spurious modes that must be filtered in order to obtain meaningful physical solutions. In [33] Maday et al. introduce a stable $\mathbb{P}_N - \mathbb{P}_{N-2}$ discretization by choosing the interpolant space based on a staggered mesh. Using this $\mathbb{P}_N - \mathbb{P}_{N-2}$ stable discretization, the approximation spaces for the velocity and pressure basis functions are defined as

$$\begin{aligned} X^N &= X \cap \mathbb{P}^2_{N,E}(\Omega) \\ X_0^N &= X_0 \cap \mathbb{P}^2_{N,E}(\Omega) \\ Y^N &= Y \cap \mathbb{P}^2_{N-2,E}(\Omega). \end{aligned} \tag{4.6}$$

Here, the interpolation spaces are defined as

$$\mathbb{P}^2_{N,E}(\Omega) = \{ v(x^e(r))|_{\Omega^e} \in \mathbb{P}_N(r_1) \otimes \mathbb{P}_N(r_2), \; e = 1, ..., E \}, \tag{4.8}$$

where $\mathbb{P}_N(r)$ is the space of Lagrangian interpolants of degree less than or equal to N defined on the Gauss-Legendre-Lobatto (GLL) points in the reference domain $[-1,1]$ of the velocity space. Similarly, on the pressure space, the basis $\mathbb{P}_{N-2}(r)$ is the space of Lagrangian interpolants of degree less than or equal to N defined on the Gauss-Legendre

Definition 8 Inf-sup stability condition

The spectral element spaces X^N, Y^N satisfy the *Inf-sup stability condition* if there exists a positive constant γ independent of N and E such that, for all $\vec{v} \in X_0^N$ and $q \in Y^N$,

$$\inf_{q \neq constant} \sup_{\vec{v} \neq \vec{0}} \frac{|(q, \nabla \cdot \vec{v})|_{0,\Omega}}{\|\vec{v}\|_{1,\Omega} \|\vec{q}\|} \geq \gamma. \tag{4.7}$$

where $\|\vec{v}\|_{1,\Omega} = (\int_\Omega \vec{v} \cdot \vec{v} + \nabla\vec{v} : \nabla\vec{v})^{1/2}$, and $\|q\| = \|q - \frac{1}{\Omega} \int_\Omega q\|$ is the quotient space norm (see [19] p. 224 & p. 228).

(GL) points in the reference domain $(-1, 1)$. An example of this staggered mesh is illustrated in Figure 4.1,

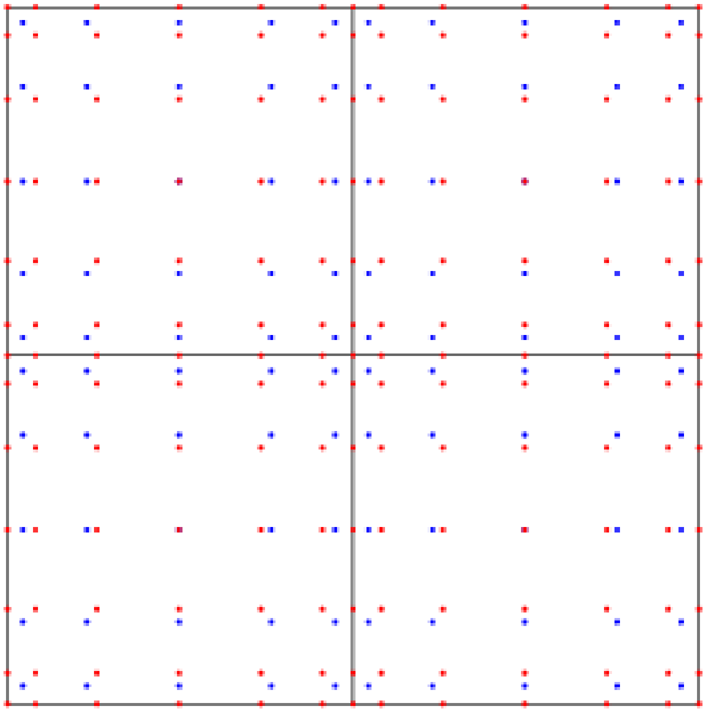

Figure 4.1: Nodes of the staggered Spectral Element Discretization for Navier-Stokes with 4 elements. Velocities are defined on 7 GLL points (red) per element and pressure is defined on 5 GL points (blue) per element.

Thus, on each element the discrete weak solution can be written as a linear combination of Legendre basis function on each element

$$\begin{aligned} u^e(x,y) &= \textstyle\sum_{i=0}^{N}\sum_{j=0}^{N} u_{ij}\pi_{N,i}(x)\pi_{N,j}(y) \\ p^e(x,y) &= \textstyle\sum_{i=1}^{N-1}\sum_{j=1}^{N-1} p_{ij}\pi_{N-2,i}(x)\pi_{N-2,j}(y). \end{aligned} \tag{4.9}$$

The coefficients u_{ij} and p_{ij} correspond to the nodal values of u and p on the tensored GLL and GL points respectively. To accommodate Dirichlet boundary conditions, we write the velocity solution as $u = u_0 + u_D$ where $u_0 = 0$ on $\partial\Omega_D$ and $u_D = u$ on $\partial\Omega_D$. Thus we can obtain the coefficients corresponding to u_0 and p by solving the nonlinear system of equations defined locally on each element

$$\begin{aligned} \underbrace{\mathbb{M}^e\Sigma' F^e(\vec{u})}_{F} u^e - \underbrace{\mathbb{M}^e\Sigma' D^{T^e}}_{D^T} p^e &= \underbrace{\mathbb{M}^e\Sigma'(M^e f^e - F^e(\vec{u})u^e_D)}_{b} \\ -\underbrace{D^e}_{D} u^e &= 0. \end{aligned} \tag{4.10}$$

As in the previous chapter, a mask matrix $\mathbb{M}^e$ sets u^e to zero at all Dirichlet boundary nodes. Thus, after obtaining u_e on each element, this solution can be combined with u_D to form the discrete solution satisfying (3.1) and (3.2).

Inter-element connections are handled via the direct stiffness summation operator Σ' defined in section 2.3. We note that this discrete system matrix is represented elementwise since Σ' is applied in order to sum elemental contributions. This again constitutes a matrix-free formulation where the discrete system can be written in terms of operators defined on each element. Namely, the discrete system involves a discrete identity, or mass matrix, on the right hand side, and the left hand block system involves a discrete convection-diffusion block, as well as discrete gradient and divergence blocks. Because we use $\mathbb{P}_N - \mathbb{P}_{N-2}$ elements, the discrete divergence and gradient operators D and D^T must also interpolate vectors between the velocity and pressure grids [20]. The elemental operators for the convection-diffusion components were developed in the previous chapter in section 3.1. To formulate the discrete divergence and gradient operators, we use the definitions of u and p in (4.9) and note that

$$\int_{\Omega^e} q(\nabla \cdot \vec{u}) = \frac{h_y}{2}\int_{-1}^{1} \pi_{N-2,k}\frac{\partial \pi_{N,i}}{\partial x} + \frac{h_x}{2}\int_{-1}^{1} \pi_{N-2,k}\frac{\partial \pi_{N,i}}{\partial y}. \tag{4.11}$$

To obtain a discrete operator we apply Gauss-Legendre quadrature and which produces the discrete form

$$D^e_{ij} = \sum_{d=1}^{2}\sum_{i,j=1}^{N} \pi_{N-2,i}(\eta_i)\pi_{N-2,j}(\eta_j)\frac{\partial \pi_N}{\partial x_d}(\eta_i,\eta_j)\sigma_i\sigma_j, \tag{4.12}$$

were σ_i denotes the Gauss-Legendre quadrature weight associated with the i^{th} Gauss-Legendre quadrature node η_i. Since the velocities and pressure are defined on different grids, the velocity derivatives must be interpolated from Gauss-Legendre points to Gauss-Legendre-Lobatto points. We define the discrete interpolated divergence operator D^e and discrete interpolated gradient operator $D^{T\,e}$ as

$$D^e = D_x{}^e + D_y{}^e = \frac{h_y}{2}\tilde{I}\otimes\tilde{D} + \frac{h_x}{2}\tilde{D}\otimes\tilde{I}, \tag{4.13}$$

$$D^{T\,e} = D_x^{T\,e} + D_y^{T\,e} = \frac{h_y}{2}\tilde{I}^T\otimes\tilde{D}^T + \frac{h_x}{2}\tilde{D}^T\otimes\tilde{I}^T, \tag{4.14}$$

where the one-dimensional weighted interpolation matrix $\tilde{I}$, with dimension as dimension $(N-1)\times(N+1)$, is constructed to map Gauss-Legendre-Lobatto nodes to Gauss-Legendre points

$$\tilde{I}_{ij} = \sigma_i \pi_{N,j}(\eta_i). \tag{4.15}$$

Similarly, the weighted one-dimensional derivative matrix $\tilde{D}$ interpolated onto the Gauss-Legendre points is defined as

$$\tilde{D}_{ij} = \sigma_i \frac{d\pi_{N,j}}{dr}\bigg|_{r=\eta_i}. \tag{4.16}$$

With this discretization strategy we obtain the discrete form of the steady Navier-Stokes equations

$$\begin{bmatrix} F(u) & -D^T \\ -D & 0 \end{bmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} Mf \\ 0 \end{pmatrix}. \tag{4.17}$$

4.2 Nonlinear Iteration Methods

The steady Navier-Stokes equations form a nonlinear system of equations. This nonlinear system can be solved using Picard iteration. Picard's method has a large domain of convergence and converges linearly [19], [41]. In this section we give an overview of the Picard iteration method, and explain how to formulate the linearized Navier-Stokes equations at each nonlinear iteration.

To determine the root of a function $g(x)$ via Picard iteration, a sequence of iterates are obtained by solving the linear system

$$Ax_{k+1} = Gx_k \tag{4.18}$$

where $g(x) = Ax - Gx$ with G being nonlinear and A being nonsingular. Each iterate x_{k+1} can be equivalently be described as an update, δx_k, to a previous iterate, x_k, by seeing that

$$x_{k+1} = A^{-1}Gx_k = x_k - A^{-1}(A-G)x_k = x_k - A^{-1}Hx_k = x_k - \delta x_k. \tag{4.19}$$

Thus at each nonlinear iteration, the update δx_k can be obtained by solving the linear system

$$A\delta x_k = Hx_k. \tag{4.20}$$

In the context of the Navier-Stokes equations, the initial guess x_0 is a vector storing an initial velocity field u_0 and an initial pressure p_0. The goal of the nonlinear iteration is to obtain the solution through a series of updates $\delta u_k, \delta p_k$ such that $u = u_k + \delta u_k, p = p_k + \delta p_k$ satisfy (4.1). Inserting u and p of this form into the steady Navier-Stokes equations produces the system of equations

$$\begin{cases} -\nabla^2(u_k+\delta u_k) + ((u_k+\delta u_k)\cdot\nabla)(u_k+\delta u_k) + \nabla(p_k+\delta p_k) = f \\ -\nabla\cdot(u_k+\delta u_k) = 0 \end{cases}. \tag{4.21}$$

The update terms are moved to the left-hand side, and the quadratic term $(\delta u_k\cdot\nabla)\delta u_k$ is omitted along with the linear term $(\delta u_k\cdot\nabla)u_k$, which gives rise to the linear system

$$\underbrace{\begin{bmatrix} -\nabla^2 + u_k\cdot\nabla & \nabla \\ -\nabla & 0 \end{bmatrix}}_{A} \underbrace{\begin{pmatrix} \delta u_k \\ \delta p_k \end{pmatrix}}_{\delta x_k} = \underbrace{\begin{pmatrix} f - (-\nabla^2 + (u_k\cdot\nabla) & -\nabla p_k) \\ \nabla & 0 \end{pmatrix}}_{H} \underbrace{\begin{pmatrix} u_k \\ p_k \end{pmatrix}}_{x_k} . \tag{4.22}$$

Applying the spectral element method, produces the discrete linear block system

$$\begin{bmatrix} F(u_k) & -D^T \\ -D & 0 \end{bmatrix} \begin{pmatrix} \delta u_k \\ \delta p_k \end{pmatrix} = \begin{pmatrix} f - (F(u_k) + D^T p_k) \\ D^T u_k \end{pmatrix}. \tag{4.23}$$

Similarly, the steady Navier-Stokes equations can be solved using Newton's method. Newton's method has a narrower domain of convergence compared to Picard's method, but, Newton's method converges quadratically, as opposed to Picard's method which converge linearly.

Newton's method for finding a root x^* of a function $g(x)$ is given by the iteration

$$x_{k+1} = x_k - \frac{g(x_k)}{g'(x_k)}. \tag{4.24}$$

Setting $\delta x_k = -\frac{g(x_k)}{g'(x_k)}$ one can view Newton iteration as a series of updates, $x_{k+1} = x_k + \delta x_k$ to an initial guess x_0. Thus, one obtains an update, δx_k, at each step by solving

$$g'(x_k)\delta x_k = -g(x_k). \tag{4.25}$$

In the context of the Navier Stokes equations, the initial guess x_0 is a vector storing an initial velocity field u_0 and an initial pressure p_0. The goal of the non-linear iteration is to obtain the solution through a series of updates $\delta u_k, \delta p_k$ such that $u = u_k + \delta u_k, p = p_k + \delta p_k$ satisfy (4.1). Inserting u and p of this form into the steady Navier-Stokes equations produces the system of equations

$$\begin{cases} -\nabla^2(u_k+\delta u_k)+((u_k+\delta u_k)\cdot\nabla)(u_k+\delta u_k)+\nabla(p_k+\delta p_k)=f \\ -\nabla\cdot(u_k+\delta u_k)=0 \end{cases}. \tag{4.26}$$

Moving the update terms to the left hand side and omitting the quadratic term $(\delta u_k\cdot\nabla)\delta u_k$, one obtains the linear system of equations in the form of the Newton update system (4.25),

$$\underbrace{\begin{bmatrix} (-\nabla^2+u_k\cdot\nabla)+((\cdot)\cdot\nabla)u_k & \nabla \\ -\nabla & 0 \end{bmatrix}}_{g'(x_k)}\underbrace{\begin{pmatrix} \delta u_k \\ \delta p_k \end{pmatrix}}_{\delta x_k}=\underbrace{\begin{pmatrix} f+\nabla^2 u_k-(u_k\cdot\nabla)u_k-\nabla p_k \\ \nabla\cdot u_k \end{pmatrix}}_{g(x_k)} \tag{4.27}$$

The corresponding discrete linear system is of the form

$$\begin{bmatrix} F(u_k)+C(\cdot)u_k & -D^T \\ -D & 0 \end{bmatrix}\begin{pmatrix} \delta u_k \\ \delta p_k \end{pmatrix}=\begin{pmatrix} f-(F(u_k)+D^T p_k) \\ Du_k \end{pmatrix}. \tag{4.28}$$

The pure convection term $C(\cdot)u_k$ is derived from the trilinear form introduced in section 4.1. The element-based tensor product form of C is

$$\begin{aligned} C_x^e &= \underbrace{W_x^e(\hat{M}\otimes\frac{h_y}{2}\hat{C})}_{\text{Convection in x}} \\ C_y^e &= \underbrace{W_y^e(\frac{h_x}{2}\hat{C}\otimes\hat{M})}_{\text{Convection in y}} \\ C^e(w^e) &= C_x^e+C_y^e \end{aligned} \tag{4.29}$$

where the convection coefficient W is taken as the iterate δu_k. The operator F is defined element-wise in equation (3.17) where the wind vector field is replaced by the previous iterate u_k.

In order to apply Newton's method one must solve for a sequence of updates to an initial guess, where each update satisfies equation (4.28). Because of the small domain of convergence of Newton's Method, the initial iterate needs to be reasonably close to the solution in order for the method to converge. In practice, this initial iterate may be obtained via a few iterations of Picard's method.

To perform the nonlinear iteration in our simulations we start by choosing an initial point $(u^0, p^0)^T = (\vec{0}, \vec{0})$. After each step of the nonlinear iteration, we determine if the nonlinear iteration has converged to the steady solution by checking to see if the k^{th} iterate $(u^{(k)}, p^{(k)})^T$ satisfies the criterion

$$\underbrace{\left\| \begin{bmatrix} Mf - (F(u_k)u_k - D^T p_k) \\ Du_k \end{bmatrix} \right\|}_{n^{(k)}} < \underbrace{10^{-5}}_{\tau} \left\| \begin{bmatrix} Mf \\ 0 \end{bmatrix} \right\|. \tag{4.30}$$

If it does, this means that the nonlinear the vector Euclidean residual norm at step k, denoted n^k has a relative error less than $\tau = 10^{-5}$, and we stop the nonlinear iteration process. If the nonlinear iteration has not converged, we formulate the linear system for step $k+1$, and solve for the next correction term $(\delta u_{k+1}, \delta p_{k+1})$. Each linear solve is performed using Flexible GMRES (FGMRES) as in example 3.5.1 of the previous chapter. This choice of linear solver accommodates the block preconditioner we apply via inner iterations. FGMRES is given an initial iterate $(\vec{0}, \vec{0})$, and we iterate until the linear residual at step m, $r^{(m)}$, satisfies

$$\|r^{(m)}\| < 10^{-1}\tau\|n^{(k)}\|. \tag{4.31}$$

In the next section we discuss a block preconditioning strategy we use to accelerate the convergence of FGMRES at each nonlinear iteration.

4.3 Block Preconditioner for Linearized Navier-Stokes

In this section, we introduce the block preconditioner design of Elman et al. outlined in [19], and follow their argument throughout this analysis. We discuss how the convection-diffusion preconditioner developed in the previous chapter can be used in this setting, and introduce the "Least-Squares Commutator" as a subsidiary component of this block preconditioning strategy.

4.3.1 Eigenvalue Structure of Block System

The number of iterations required for GMRES to converge depends on properties of the linear system operator such as its eigenvalues. Consider first a block-diagonal preconditioner

$$P = \begin{bmatrix} F & 0 \\ 0 & P_S \end{bmatrix}, \tag{4.32}$$

where P_S is an approximation to the *Schur complement* $S := DF^{-1}D^T$ of the linearized Navier-Stokes operator in equation (4.23) to be determined. We are interested in the eigenvalues of the preconditioned system

$$\begin{bmatrix} F & -D^T \\ -D & 0 \end{bmatrix} \begin{bmatrix} F & 0 \\ 0 & P_S \end{bmatrix}^{-1}, \tag{4.33}$$

or equivalently, the eigenvalues of the generalized problem

$$\begin{bmatrix} F & -D^T \\ -D & 0 \end{bmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \lambda \begin{bmatrix} F & 0 \\ 0 & P_S \end{bmatrix} \begin{pmatrix} u \\ p \end{pmatrix}. \tag{4.34}$$

We consider separately the two possibilities $\lambda = 1$ and $\lambda \neq 1$. The eigenvalue $\lambda = 1$ has algebraic and geometric multiplicity $n_u - n_p$ corresponding to the eigenvectors $(u,0)^T$ where $-Du = 0$. For the second possibility, $\lambda \neq 1$, we explore this scenario by block elimination. Starting with the first block, we solve for u:

$$\begin{aligned} Fu - D^T p &= \lambda F u, && (4.35) \\ u &= F^{-1}(\lambda F u + D^T p), && (4.36) \\ u &= \lambda u + F^{-1} D^T p, && (4.37) \\ (1-\lambda)u &= F^{-1} D^T p, && (4.38) \\ u &= -(1/\lambda - 1) F^{-1} D^T p. && (4.39) \end{aligned}$$

Eliminating u from the second block of the system gives us

$$\begin{aligned} -Du &= \lambda P_S p, && (4.40) \\ (1/(\lambda-1) D F^{-1} D^T p &= \lambda P_S p, && (4.41) \\ D F^{-1} D^T p &= \lambda(\lambda - 1) P_S p. && (4.42) \end{aligned}$$

We define the eigenvalue $\mu = \lambda(\lambda - 1)$, to satisfy the relationship

$$D F^{-1} D^T p = \mu P_S p. \tag{4.43}$$

The optimal choice for P_S is $DF^{-1}D^T$, which is the Schur Compliment of the linearized Navier-Stokes equations. In this case, $\mu = 1$ causing the preconditioned linear system 4.33 to have a total of three distinct eigenvalues

$$\lambda_1 = 1, \quad \lambda_{2,3} = \frac{1 \pm \sqrt{5}}{2}. \tag{4.44}$$

Using the Schur compliment for the P_S block would allow preconditioned GMRES to solve the linear system from Picard iteration (4.23) in three iterations [38]. This preconditioner exhibits ideal convergence, but the application of the preconditioner is prohibitively expensive since it requires computing the action of the inverse of the dense matrix $DF^{-1}D^T$, as well as F.

From the analysis of this diagonal block preconditioner we see that developing an effective preconditioner for the block linear systems arising from linearization requires a cheap approximation to the actions of $(DF^{-1}D^T)^{-1}$ and F^{-1}. We also see that these approximations should result in the preconditioned system (4.34) having eigenvalues λ tightly clustered around 1 and $\frac{1 \pm \sqrt{5}}{2}$.

To obtain a good approximation to the Schur compliment, consider the dependence of λ on μ, namely,

$$\lambda \rightarrow \frac{1 \pm \sqrt{1+4\mu}}{2}. \tag{4.45}$$

In the example above, we saw that choosing $P_s = S$ results in the optimal choice of preconditioner which led to $\mu = 1$. This observation motivates the construction of a preconditioning operator P_S that approximates the Schur compliment in order to keep μ close to 1, while at the same time allowing for inexpensive application of P_S^{-1}.

Before introducing the preconditioner P_S, we point out that in [18] Elman and Silvester show that when using a diagonal block preconditioner of the form (4.32), if the eigenvalues, μ, of the approximate Schur complement system in (4.43) are kept in a small region in the complex plane near $(1,0)$, then the eigenvalues λ of the preconditioned linear system 4.34 will be constrained inside two small groups on each side of the imaginary axis, centered along the line $R(\lambda) = 1/2$. In [16] and [18], Elman et al. consider the block preconditioner

$$P = \begin{bmatrix} F & -D^T \\ 0 & -P_S \end{bmatrix}, \tag{4.46}$$

which can be viewed as an approximation to the upper block of the LU factorization of the linear operator in 4.23, that is,

$$\begin{bmatrix} F & -D^T \\ -D & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ -DF^{-1} & I \end{bmatrix} \begin{bmatrix} F & -D^T \\ 0 & -S \end{bmatrix}. \tag{4.47}$$

Elman and Silvester [18] explain that the eigenvalues λ^* of the upper triangular block preconditioned system

$$\begin{bmatrix} F & -D^T \\ -D & 0 \end{bmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \lambda^* \begin{bmatrix} F & -D^T \\ 0 & -P_S \end{bmatrix} \begin{pmatrix} u \\ p \end{pmatrix}, \tag{4.48}$$

again correspond to $\lambda^* = 1$ and the eigenvalues of (4.43), but the eigenvalues corresponding to μ now lie in just one side of the imaginary axis, thereby reducing the number of iterations by approximately one-half. That is, using the upper triangular block system (4.46) as a preconditioner instead of the block diagonal (4.33) leads to convergence of the linear solver in roughly half the number of steps as with the block diagonal preconditioner is applied. Moreover, the extra work to apply (4.46) is minimal.

Our goal now is to develop cheap approximations to F^{-1} and $(DF^{-1}D^T)^{-1}$ that retain their spectral properties. We demonstrated in Chapter 3 that $\bar{F}^{-1}$ based on element-wise average wind approximations can be used effectively as a preconditioner for F. This suggests using $\bar{F}$ in place of F in (4.46). For the block P_S, we use the least squares commutator developed in [17] based on an approximate commutator. The details of this construction are discussed in the next subsection.

4.3.2 Least Squares Commutator Approximation

The *least squares commutator* preconditioner (LSC) was developed by Elman, Howle, Shadid, Shuttleworth and Tuminaro in [17]. They approximate the Schur complement by solving a particular least-squares problem that minimizes an algebraic commutator based on the gradient of convection-diffusion operators defined on the velocity and pressure spaces.

To begin, consider the linear convection-diffusion operator $L_v = -\nu\nabla^2 + \vec{w}\cdot\nabla$, derived as the upper left block in the linear system arising from the Picard method in the previous section. Assume that a similar operator L_p can be defined on the pressure space, that is, $L_p = (-\nu\nabla^2 + \vec{w}\cdot\nabla)_p$ and then consider the commutator of these operators with respect to the gradient,

$$\varepsilon = L_v\nabla - \nabla L_p. \tag{4.49}$$

It is clear that for smooth $\vec{w}$ in L_v and L_p, ε is small in the interior of Ω. The corresponding discrete commutator is defined as

$$\varepsilon_h = (M^{-1}F)(M^{-1}D^T) - (M^{-1}D^T)(M_p^{-1}F_p), \tag{4.50}$$

where M_p and F_p are the mass matrix and convection-diffusion matrix defined on the discrete pressure space. If ε_h is small, meaning the two discrete operators nearly commute with respect to the gradient, then an approximation to $S = DF^{-1}D^T$ can be obtained by pre-multiplying ε_h by $DF^{-1}M$ and post-multiplying ε_h by $F_p^{-1}M_p$. That is, if ε_h is small

$$DF^{-1}D^T = (DF^{-1}M)(M^{-1}F)(M^{-1}D^T) \approx DM^{-1}D^T F_p M_p. \tag{4.51}$$

The idea is to construct F_p such that ε_h is small. This is done by minimizing an associated L_2 norm of the discrete commutator in the least squares sense. That is, we want to make the quantity

$$\sup_{p \neq 0} \frac{||[\varepsilon_h]p||_M}{||p||_M}, \quad \text{where} \, ||v||_M = (Mv, v)^{1/2}, \tag{4.52}$$

small. To minimize this norm, the j^{th} column of F_p is chosen to solve the least squares problem

$$\min \left|\left|[M^{-1}FM^{-1}D^T]_j - M^{-1}D^T M_p^{-1}[F_p]_j\right|\right|_M. \tag{4.53}$$

The associated normal equations

$$M_p^{-1}DM^{-1}D^T[F_p]_j = [M_p^{-1}DM^{-1}FM^{-1}D^T]_j \tag{4.54}$$

are solved to provide the definition for F_p,

$$F_p = M_p(DM^{-1}D^T)^{-1}(DM^{-1}FM^{-1}D^T). \tag{4.55}$$

Inserting this choice of F_p into equation 4.51 and simplifying the right hand side gives us

$$\begin{aligned} DF^{-1}D^T &\approx DM^{-1}D^T F_p^{-1} M_p \\ &= DM^{-1}D^T [M_p (DM^{-1}D^T)^{-1}(DM^{-1}FM^{-1}D^T)]^{-1} M_p \\ &= DM^{-1}D^T [(DM^{-1}D^T)^{-1}(DM^{-1}FM^{-1}D^T)]^{-1} M_p^{-1} M_p \\ &= (DM^{-1}D^T)(DM^{-1}FM^{-1}D^T)^{-1}(DM^{-1}D^T). \end{aligned} \tag{4.56}$$

This leads to a computationally efficient approximation of $DF^{-1}D^T$. We use this in the block preconditioner from the previous section, thus defining

$$P_S := (DM^{-1}D^T)(DM^{-1}FM^{-1}D^T)^{-1}(DM^{-1}D^T). \tag{4.57}$$

The inverse of P_S can be applied inexpensively as

$$P_S^{-1} = (DM^{-1}D^T)^{-1}(DM^{-1}FM^{-1}D^T)(DM^{-1}D^T)^{-1}, \tag{4.58}$$

which requires two solves involving the *consistent Poisson operator* $DM^{-1}D^T$. This operator is symmetric positive definite, and the inverse can be applied via the Conjugate Gradient method.

We conclude this section by writing the final form of the block preconditioner we use to accelerate the convergence of the linearized Navier-Stokes systems at each step of nonlinear the Picard iterations, namely,

$$P = \begin{bmatrix} \bar{F} & -D^T \\ 0 & -P_S \end{bmatrix} \tag{4.59}$$

where $\bar{F}$ is defined in Chapter 3 by using the average velocity on each element, D^T is the interpolated discrete gradient defined in equation (4.10) and P_S is defined above. In the next section we demonstrate the effectiveness of this preconditioner through several test cases.

4.4 Computational Results and Analysis

In this section we use three sample flows to test our solution method. We start our analysis by considering the effect of inexactly applying the action of the inverse of the block $\bar{F}$ for the preconditioner of (4.46).

After determining a strategy for applying this block inexactly, we test this idea on three sample flow problems. The first problem is known as the lid-driven cavity; we use this problem to examine how the number of linear iterations are influenced by changes in the mesh size and the Reynolds number. We then apply our solution technique to the Kovasznay flow, which has an analytical solution that we use to demonstrate the accuracy of the solution technique. In our discussion of Kovasznay flow, we also discuss how to change tolerances inside the nonlinear and linear solvers to allow for more accurate solutions. Finally, we apply our solution method to a problem that models the flow over a step. This test case demonstrates how our method can be applied to problems with mixed boundary conditions. Throughout this section, we report the number of FGMRES iterations taken at the final nonlinear iteration as this number is characteristic of the number of linear iterations taken at each nonlinear step.

4.4.1 Lid-Driven Cavity

The lid-driven cavity is commonly used as a benchmark problem to test models of incompressible flow and solvers for the associated problems. The flow exhibits recirculations that occur on disparate scales, thus posing a significant challenge to both nonlinear solvers and subsidiary linear solvers. We consider a square computational domain $\Omega = [-1,1]^2$, where no-slip (zero-velocity) conditions are set along the sides and along the bottom. Along the top of the domain a lid moves left to right driving the flow inside the cavity. For this boundary, we impose what is called the “regularized cavity” boundary condition $u_x = 1 - x^2$ at $y = 1$. In two dimensions, steady flow states are known to exist for Reynolds number approaching 8000 [6], in three dimensional cubes, the flow becomes unsteady at Reynolds number much less than 1000 [50]. Figure 4.2 shows the streamlines and pressure of an example flow when $Re = 2000$, and Figure 4.3 shows a plot of the velocity profiles of u_x (left) and u_y (right) along the center of the cavity. In this flow, there are four regions of recirculating fluid. In the primary chamber near the center of the cavity, the fluid moves the fastest, whereas in the bottom corner regions two smaller eddies form moving much slower and counter to the primary flow. The smallest eddy occurs in the top left corner, and this eddy is present only in higher Reynolds number flows.

We study the convergence of FGMRES using the block triangular preconditioner from the previous section by first considering an “exact” version of the Least-Squares

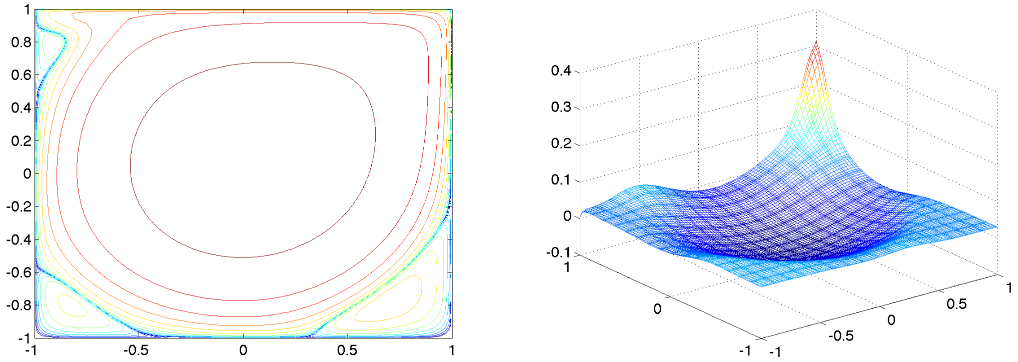

Figure 4.2: Streamline plot (left) and pressure plot (right) of Lid-Driven Cavity with $Re = 2000$.

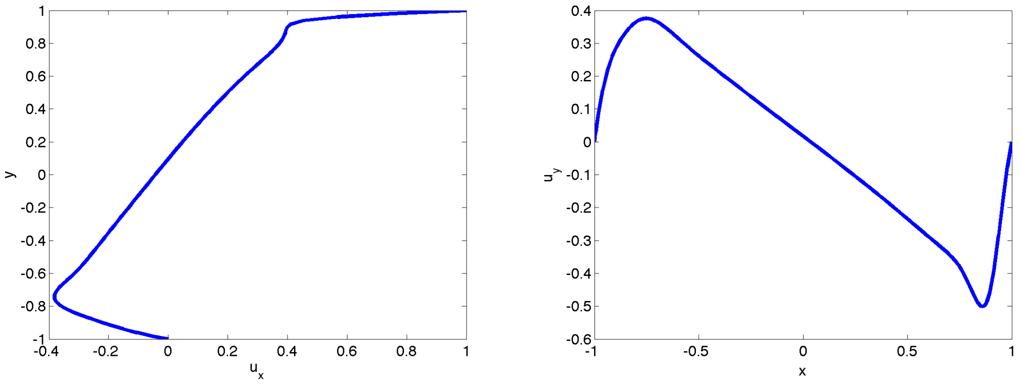

Figure 4.3: Lid-Driven Cavity velocity profiles along centerlines u_x (left) u_y (right) corresponding to $Re = 2000$.

Commutator (LSC) preconditioner

$$P_{EXACT} = \begin{bmatrix} F & -D^T \\ 0 & -P_S \end{bmatrix}. \tag{4.60}$$

We do this to elucidate two items, first, we would like to see how the "exact" LSC accelerates the convergence of FGMRES for solving the linearized Navier-Stokes equations. The second item we would like to understand is how well $\bar{F}$ preconditions the linearized convection-diffusion systems arising from Picard iteration.

Flexible GMRES is used to solve the block linear system. To apply the preconditioner, P_{EXACT}, we must apply the action of F^{-1}. This is done by applying Flexible GMRES once again, using $\bar{F}$ as the preconditioner for this subsidiary convection-diffusion system. We use a fixed mesh with $N = 4$ and $E = 64$, and compare the number of iterations required for the block linear system (4.23) to converge with three choices of Reynolds number, $Re = 10, 100, 1000$. Convergence results are shown in Table 4.1 we report the following quantities:

- Block FGMRES steps: Average number of iterations required to solve the block system (4.23) with preconditioner (4.60) to a tolerance of 10^{-6}.
- F^{-1} steps: Average number of iterations for solving systems with coefficient matrix F needed to apply the action of the inverse of (4.60) to a tolerance of 10^{6}. This solve is preconditioned by applying $\bar{F}^{-1}$ inexactly.
- $\bar{F}_S^{-1}$ steps: Fixed number of steps taken to inexactly apply $\bar{F}_S^{-1}$ in order to obtain an approximate interface solution for $\bar{F}$, which is used to precondition the upper left coefficient matrix F in (4.60).
- P_S^{-1} steps: Average number of steps required for the consistent Poisson solves within the P_S solve. There are two such solves per step, we use a stopping tolerance of 10^{-2}.

From Table 4.1 we see that the number of FGMRES iterations appear to vary only slightly as the Reynolds number is increased. This means that the "exact" preconditioner is doing a good job of preconditioning the block system. We also see that $\bar{F}$ appears to keep the number of convection-diffusion iterations relatively low. However, these iterations do grow as *Re* is increased. This means that either more elements are needed for $\bar{F}$ to be a good approximation of F, and/or the interface solution obtained from the inexact $\bar{F}_S^{-1}$ solve may not be accurate enough to provide a good approximation of F along elemental interface, especially for higher *Re*. To apply the consistent Poisson operators in the Schur complement approximation, P_S^{-1}, we use the Conjugate Gradient method with a stopping tolerance of 10^{-2}. We found that making this tolerance smaller had no effect on the number of FGMRES iterations. In [20] Fischer introduced several preconditioning techniques based on domain decomposition for the consistent Poisson operator arising from operator-splitting methods. We expect to be able to reduce the number of iterations in the consistent Poisson solves by employing these techniques.

Re	Block FGMRES steps	F^{-1} steps	$\bar{F}_S^{-1}$ steps	P_S^{-1} steps
10	24	6	10	79
100	35	12	10	97
1000	52	80	10	82

Table 4.1: Convergence results for example 4.4.1 using P_{EXACT} preconditioner for increasing Reynolds number.

Using this exact preconditioner is infeasible since it requires several nested GMRES iterations at each application. To improve the efficiency of this preconditioner, we eliminate the outer FGMRES iterations for F by replacing F directly with $\bar{F}$. Thus we consider the inexact least squares commutator block preconditioner

$$P_{INEXACT-LSC} = \begin{bmatrix} \bar{F} & -D^T \\ 0 & -P_S \end{bmatrix}. \tag{4.61}$$

Our goal is to be able to apply $\bar{F}$ inexactly and keep the number of Block FGMRES iterations low as the grid is refined and as Re is increased. As we pointed out earlier, using 10 steps to obtain an approximate interface solution resulted in inner iteration counts (middle column above) that were influenced by the Reynolds number. This motivates our next study, where we investigate the dependence of the stopping tolerance of the Schur complement problem $\bar{F}_S^{-1}$ on the number of outer FGMRES iterations.

In this study we use the "Inexact LSC" preconditioner, where the F block is replaced by $\bar{F}$. We perform two sets of simulations on the same grid where $N = 4$ and $E = 16 \times 16$. We consider a flow with $Re = 100$ and then with $Re = 1000$. For each Re, we perform four test runs by setting the tolerance for the $\bar{F}_S^{-1}$ solve to 10^{-1}, 10^{-2}, 10^{-4} and 10^{-8} respectively. We then compare how the number of FGMRES iterations are affected by the change in this interface tolerance. Figure 4.4 contains the linear system residual at each step of FGMRES during the second Picard iteration for each of these test runs. The left pane corresponds to $Re = 100$ and the right pane corresponds to $Re = 1000$. From these figures we see that the accuracy of $\bar{F}_S^{-1}$ has little bearing on the overall performance of FGMRES. This means that $\bar{F}_S^{-1}$ can be solved inexactly at each step of the block linear solve without affecting the convergence of FGMRES.

Next we show how the preconditioner from equation (4.59) affects the outer FGMRES iterations when each block is applied with a relative error of 10^{-2}. Table 4.2 contains the convergence results of this preconditioner as the number of elements and the degree of the polynomial basis are increased. In the left table we use a fixed polynomial basis of degree 4 and refine the element grid from 4×4 to 32×32. We list the number of nonlinear Picard iterations, the number of block FGMRES iterations (FGMRES column)

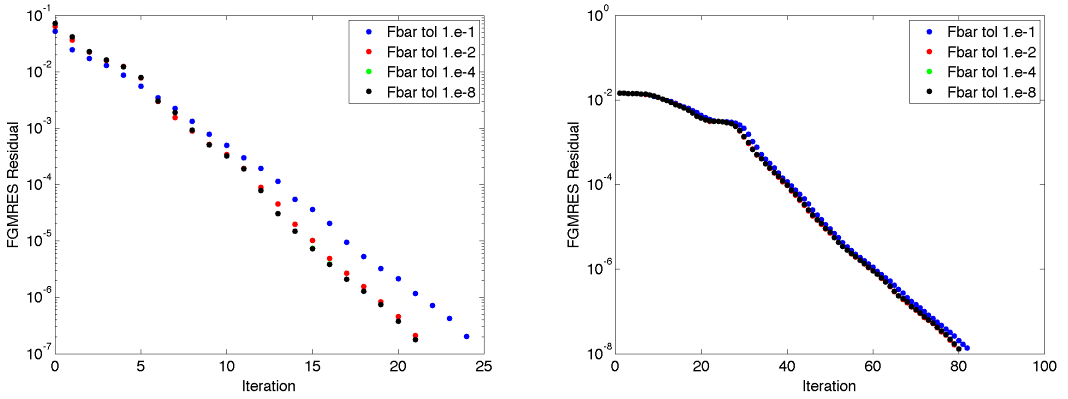

Figure 4.4: Comparison of linear residuals for $Re = 100$ (left) and $Re = 1000$ (right) using FGMRES based on solving $\bar{F}_S^{-1}$ to a tolerance of 10^{-1} (blue), 10^{-2} (red), 10^{-4} (beneath black) and 10^{-8} (black).

and the number of iterations needed to apply the subsidiary components of the block preconditioner. We observe a mild dependence on the number of elements. In the right table, we repeat this experiment with a fixed element grid with four elements in each direction as the polynomial basis is increased from 2 to 16. Here we observe the number of Picard iterations and block FGMRES steps are essentially the same as in the left table. The subsidiary convection-diffusion interface solve takes fewer steps due to the fewer subdomains, and the consistent Poisson solves appear to be affected more by changes in polynomial degree than in the number of elements.

E	Picard steps	FGMRES steps	$\bar{F}_S^{-1}$ steps	P_S^{-1} steps
16	16	20	14	8
64	12	26	32	22
256	9	37	94	45
1024	8	55	200	87

N	Picard steps	FGMRES steps	$\bar{F}_S^{-1}$ steps	P_S^{-1} steps
2	16	20	14	8
4	9	32	16	60
8	8	38	24	139
16	8	54	33	200

Table 4.2: Convergence results for example 4.4.1 with $Re = 100$ using block preconditioner with $\bar{F}_S^{-1}$ and P_S^{-1}, each applied with a relative tolerance of 10^{-2}. E is varied as $N = 4$ is fixed (left). N is varied as $E = 16$ on a fixed 4×4 element grid (right).

We now consider how increasing the Reynolds number affects the convergence of our solution method. For this study, we use a fixed grid with $N = 8$ and $E = 256$ with 16 elements in each direction and vary the Reynolds number from 10 to 5000. We see in table 4.3 that the number of FGMRES iterations are mildly dependent on the Reynolds number.

Re	Picard steps	FGMRES steps	$\bar{F}_S^{-1}$ steps	P_S^{-1} steps
10	5	30	186	200
100	7	42	141	200
500	9	57	130	200
1000	11	96	119	200
2000	15	194	145	200
5000	20	240	107	200

Table 4.3: Convergence results for example 4.4.1 using block preconditioner with $\bar{F}_S^{-1}$ and P_S^{-1}, each applied with a relative tolerance of 10^{-2} as Re is increased on a fixed grid $N = 8$ and $E = 256$.

4.4.2 Kovasznay Flow

In this section we demonstrate how our solution method can be used to perform high-accuracy simulations by modifying the tolerances of the nonlinear and linear iteration schemes. We use a sample flow with an analytical solution, and illustrate how our numerical solutions convergence exponentially to the exact solution as the degree N of the polynomial basis is increased.

To perform high accuracy simulations, the nonlinear residual tolerance τ from equation (4.30) is chosen to be one order of magnitude less than the expected L_2 error of the discrete solution. Once τ is chosen, the stopping criteria for the linear system is automatically adjusted to $10^{-1}\tau\|n^{(k)}\|$ where $\|n^{(k)}\|$ is the nonlinear residual at step k. These changes in nonlinear and linear tolerances allow the error from the linear and nonlinear iterations to be less than the discretization error, thus allowing for accurate steady solutions to be obtained at each refinement level. For example, in the test case below when $N = 8$ the discretization error is $O(10^{-9})$ so the nonlinear residual tolerance τ is set to 10^{-10}.

We use a sample flow problem with an analytical solution developed by Kovasznay in [31]. This flow exhibits behavior similar to that of a fluid moving behind a periodic array of cylinders. In contrast to Poiseuille flow where the convection term has no effect on the flow, this solution incorporates the full nonlinear effects of the convection term, and thus provides a good test for numerical convergence of our scheme. We construct a test domain $\Omega = [-.5, 1.5] \times [-.5, 1]$ and prescribe Dirichlet boundary conditions along each boundary corresponding to the analytical solution which is given as

$$\begin{aligned} u &= 1 - e^{\lambda x} cos 2\pi y, \\ v &= \tfrac{\lambda}{2\pi} e^{\lambda x} sin 2\pi y, \\ p &= \tfrac{1}{2}(1 - e^{2\lambda x}), \\ \lambda &:= \tfrac{Re}{2} - \sqrt{\tfrac{Re^2}{4} + 4\pi^2}. \end{aligned} \tag{4.62}$$

The left pane of Figure 4.5 shows the streamlines of the computed Kovasznay flow solution with $Re = 40$. The right pane shows the relative L_2 error of the computed solution as the degree of the polynomial basis is varied from $N = 4$ to $N = 12$. The blue dots represent the relative error in the horizontal velocity $\frac{\|u_x - u_x^N\|}{\|u_x\|}$, whereas the red dots show the relative error in the vertical velocity $\frac{\|u_y - u_y^N\|}{\|u_y\|}$. These values depict exponential convergence as the polynomial degree is increased.

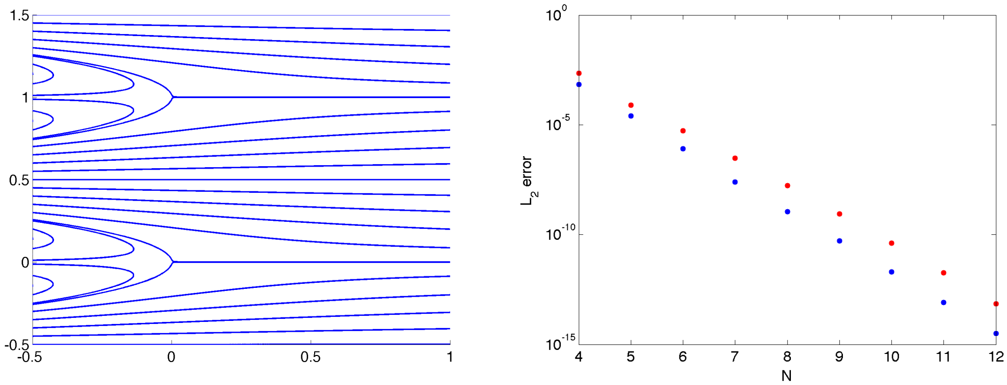

Figure 4.5: Streamline plot of Kovasznay Flow with $Re = 40$ (left). Spectral element convergence for $Re = 40$, with increasing N and fixed element grid $E = 4 \times 6$. Red denotes error in u_y blue denotes error in u_x. (right).

In Table 4.4 we show the convergence results of our solution method along with the associated relative error. This table lists the number of nonlinear Picard iterations required for the nonlinear system to converge. Since the tolerance is increased as the mesh is refined, the number of Picard iterations goes up with N. We also list the number of subsidiary linear solve iterations (FGMRES column). The convergence of FGMRES is accelerated via the block preconditioner (4.59). The interface solution for $\bar{F}_S^{-1}$ is solved to a tolerance of 10^{-1} using GMRES with Robin-Robin preconditioning. The consistent Poisson operator in P_S is applied inexactly with a tolerance of 10^{-2}. We found that using a smaller tolerance for the consistent Poisson solves had no effect on the outer FGMRES iterations. We see that the number of FGMRES iterations generally decreases as the degree of the polynomial basis is refined. These results indicate that our solution method based on nonlinear iteration and subsidiary linear solvers performs well for high-accuracy simulations.

Polynomial degree N	$\frac{\|u-u_N\|}{\|u\|}$	$\frac{\|v-v_N\|}{\|v\|}$	Picard steps	FGMRES steps	$\bar{F}_S^{-1}$ steps	P_S^{-1} steps
4	6.8×10^{-4}	2.2×10^{-3}	12	42	4	27
5	2.6×10^{-5}	8.0×10^{-5}	12	39	4	35
6	3.6×10^{-6}	2.0×10^{-5}	15	35	3	22
7	2.5×10^{-8}	3.1×10^{-7}	18	32	3	25
8	1.1×10^{-9}	1.8×10^{-8}	23	27	4	28
9	5.1×10^{-11}	8.8×10^{-10}	27	26	5	45
10	2.0×10^{-12}	4.2×10^{-11}	29	24	6	74
11	8.1×10^{-14}	1.8×10^{-12}	30	24	5	58
12	3.1×10^{-15}	7.0×10^{-14}	36	24	10	72

Table 4.4: Convergence results for example 4.4.2 using an $\bar{F}$ based LSC block preconditioner for increasing grid resolution with Re=40, E=24 on a 4×6 element grid.

4.4.3 Flow over a step

In our final flow simulation, we test our scheme on a problem with mixed boundary conditions. We construct a rectangular domain $\Omega = [-1,7] \times [-1,1]$. A step is configured along the bottom half of the left boundary by imposing a zero Dirichlet boundary condition. Along the top half of the left boundary we impose a parabolic inflow condition. More precisely, at $x = -1$, the velocity in the x-direction is defined by

$$\begin{cases} u_x &= -y(y-1) & 0 < y \leq 1 \\ u_x &= 0 & -1 \leq y \leq 0 \end{cases}. \tag{4.63}$$

Additionally, we prescribe natural outflow condition at $x = 5$,

$$\begin{cases} \nu \dfrac{\partial u_x}{\partial x} - p &= 0 \\ \dfrac{\partial u_y}{\partial x} &= 0 \end{cases}, \tag{4.64}$$

by using Neumann boundary conditions. This Neumann boundary condition automatically sets the average outflow pressure to zero.

The left pane of Figure 4.6 shows the streamlines of an example flow with $Re = 200$. We see that there is a region of recirculation behind the wall whereas the rest of the flow moves through the domain in a nearly parabolic profile. The right pane shows the corresponding pressure solution. We note that a singularity exists in the pressure solution at the point $(-1,0)$ where the flow intersects the step.

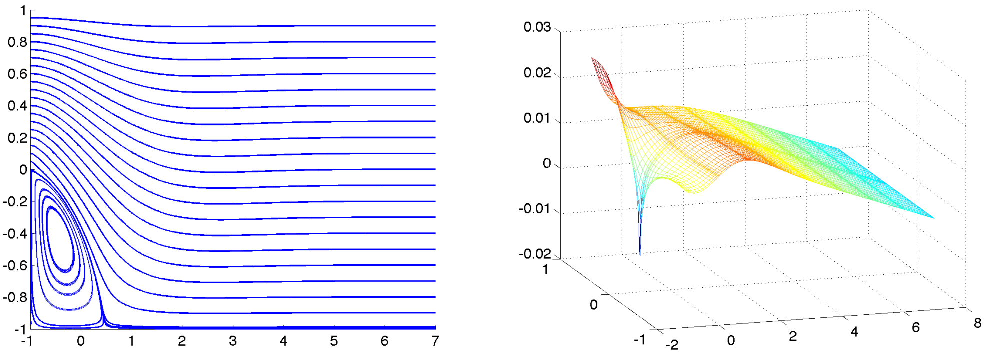

Figure 4.6: Streamline plot (left) and pressure plot (right) of flow over a step with $Re = 200$.

As the computational domain is extended in the stream-wise direction, the flow takes on a parabolic profile mimicking the flow between two plates. From this observation one can compute the expected asymptotic outflow condition using the conservation of mass. In particular, we know that in a Poiseuille flow the solution in the stream-wise direction is of the form

$$u_x = -c(y-1)(y+1), \tag{4.65}$$

so given the inflow condition (4.63), we can compute the flow rate per unit width along the inflow as

$$\int_0^1 -y(y-1)dy = \frac{1}{6}, \tag{4.66}$$

and the flow rate per unit width along the outflow in the asymptotic limit as

$$\int_{-1}^{1} -c(y-1)(y+1)dy = \frac{4}{3}c, \tag{4.67}$$

so if the channel is long enough these rates should match, in which case $c = \frac{1}{8}$. We compare the profile of the computed velocity at the outflow and the asymptotic velocity at the outflow in Figure 4.7. The left pane shows that indeed the computed solution follows a parabolic profile that nearly matches the asymptotic stream-wise velocity, the difference between the two curves is plotted in the right pane. We find that for Reynolds number greater than 200 the difference between the computed outflow stream-wise velocity and the aysmptotic stream-wise velocity is greater than 10^{-4}, while for smaller values the down-stream velocity is less affected by the step, and the residual is less than 10^{-5}.

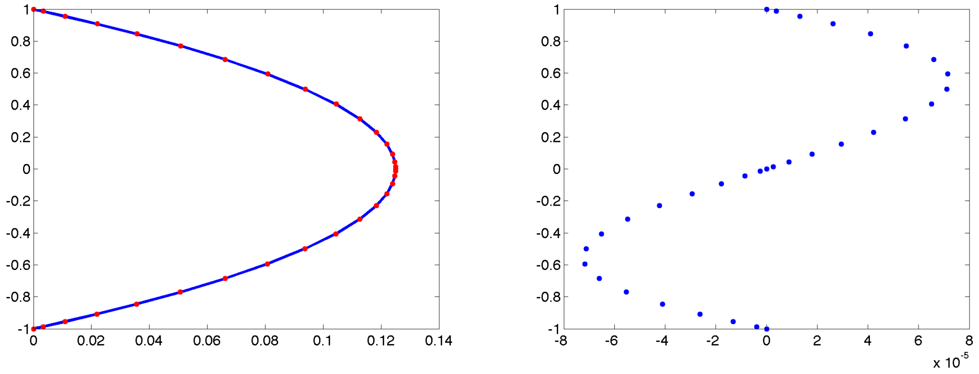

Figure 4.7: Comparison of computed outflow velocity u_x (red) and asymptotic outflow condition (blue) (left) and their difference (right), Re=200.

We examine the number of linear system iterations (4.23) required at each nonlinear Picard step as the Reynolds number varies between 10, 100 and 200. In Table 4.5 we see there is a mild dependence of the number of FGMRES iterations as the Reynolds number is increased. There is also a slight increase in FGMRES iterations as the degree of the polynomial basis is increased. In this study, we apply $\bar{F}_S^{-1}$ and P_S^{-1} to an accuracy of 10^{-6}. We use Robin-Robin preconditioned GMRES to apply $\bar{F}_S^{-1}$ and the Conjugate Gradient method to apply the inverse Poisson operator in P_S^{-1}. The iterations reported in Table 4.6 correspond to the analog of this study where the degree of the polynomial basis is fixed at $N = 4$ while the number of elements are varied. In the second table, we see that the number of FGMRES iterations are essentially unaffected by changes to the number of elements, however, the number of iterations required to apply $\bar{F}_S^{-1}$ increase with increased resolution. We also notice that there is little dependence on the Reynolds number. In both tables, we observe that the number of steps required to apply $\bar{F}_S^{-1}$ decreases as the Reynolds number is increased, we attribute this to the effectiveness of the Robin-Robin preconditioner.

Re	N	Picard steps	FGMRES steps	$\bar{F}_S^{-1}$ steps	P_S^{-1} steps
10	4	4	14	22	66
100	4	6	22	16	27
200	4	7	33	14	20
10	8	4	22	34	59
100	8	5	31	25	39
200	8	7	31	23	39
10	16	4	42	42	106
100	16	5	57	38	200
200	16	6	69	34	72

Table 4.5: Convergence results for example 4.4.3 using an $\bar{F}$ based LSC block preconditioner for increasing Reynolds number using a fixed element grid E=16 8×2 elements with N varying.

Re	E	Picard steps	FGMRES steps	$\bar{F}_S^{-1}$ steps	P_S^{-1} steps
10	16	4	14	22	66
100	16	6	22	16	27
200	16	7	33	14	20
10	64	4	16	56	47
100	64	5	22	34	24
200	64	7	28	32	28
10	256	4	28	109	124
100	256	5	31	94	48
200	256	6	34	86	38

Table 4.6: Convergence results for example 4.4.3 using an $\bar{F}$ based LSC block preconditioner for increasing Reynolds number, as the number of elements increase. N=4 is fixed while element grids vary from 8×2 to 16×4 and 32×8

4.5 Summary

We have adapted the least squares commutator method developed for Finite Element and Finite Difference Methods to a Spectral Element framework using Fast Diagonalization and inexact inner iteration for preconditioning the linearized Navier-Stokes equations. The solution technique we developed in this chapter can be categorized as a variant of the Newton-Krylov-Schur methods described in [19], [26], and [55]. Picard iteration is used to solve the nonlinear Navier-Stokes equations. At each nonlinear step we solve the discrete linearized Navier-Stokes equations. As in the previous chapter, we applied a variant of GMRES, called Flexible GMRES (FGMRES) [47] to allow for changes in the preconditioner during each linear iteration. This allowed us to precondition the linear equation using the upper triangular block preconditioner proposed in [17]. To apply the preconditioner we used the domain decomposition solver developed in the previous chapter for the subsidiary linearized convection-diffusion component, and we used the

Conjugate Gradient method for the consistent Poisson operators within the “least-squares commutator”.

We have demonstrated that our solution strategy is effective and robust for simulating a variety of fluid systems. We applied our method to accurately and efficiently solve flow problems including those involving recirculation and mixed boundary conditions. We illustrated that this solution technique can be used to provide accurate steady-state solutions that exhibit exponential convergence as the mesh is refined in N. Furthermore, we showed that when applying the convection-diffusion preconditioner inexactly within the block preconditioner, the number of FGMRES iterations become mildly dependent of the mesh and the Reynolds number.

Chapter 5

Summary and Conclusions

In this thesis we outlined the need for efficient solution methods for discrete systems that arise from high order discretization of steady state problems involving convection and diffusion. We approached this problem by developing a new set of preconditioning techniques that take advantage of flow properties and discretization structure.

We discussed how the spectral element method can be used to construct accurate solutions and efficient matrix-free solvers that can exploit cache-based matrix-matrix routines. In chapter 3 we formed a new convection-diffusion solver by adapting the Fast Diagonalization method to solve problems with constant convection coefficients. This method was used in conjunction with matrix-free domain decomposition methods to allow for efficient convection-diffusion solves using multi-element discretizations. We applied this technology to solve convection-diffusion problems with constant winds, and then extended its use by applying it as a preconditioner for non-constant coefficient convection-diffusion problems. This was accomplished by using average convection speeds on each element, and using Flexible GMRES for a solver. This method enabled us to efficiently solve non-constant coefficient problems at increased grid resolution for flows characterized by a wide range of Peclet number $Pe = 40 - 5000$.

After developing this technology for convection-diffusion systems, we introduced a method for obtaining steady-state solutions for the incompressible Navier-Stokes equations. This involved a nonlinear iteration method that required solving the linearized Navier-Stokes equations at each step. We explained how the techniques developed in chapter 3 for the convection-diffusion equation could be employed inside the "least squares commutator" block preconditioner to accelerate the convergence of the linearized Navier-Stokes equations. We pointed out that this extends the application of the least squares commutator to a matrix-free spectral element framework. We demonstrated that this method provides an efficient means of computing steady state flow solutions. We tested this method on a variety of flows including flows exhibiting recirculation, singularities and mixed boundary conditions. In our simulations we examined the influence of the mesh size and Reynolds number. We performed analysis on grids ranging from $17^2 - 129^2$ and *Re* ranging from $10 - 5000$ and found iteration counts to be mildly dependent on the mesh size and the Reynolds number.

5.1 Applications and Future Directions

To our knowledge this is the first attempt to develop efficient solvers for high-order based discretizations of non-symmetric problems arising from fluid models without applying temporal-based splitting techniques. We mentioned at several points throughout this thesis that steady flow solvers have direct application in implicit time-integration schemes as well as stability analysis techniques. We have not applied our technique to these problems in this thesis; however, we believe our method will be able to accelerate implicit time stepping methods as well as eigensolvers based on high order element based discretizations. We hope that in doing so this technique will offer an alternative to

CFL-constrained semi-implicit time integration methods commonly used in these areas.

Our solution algorithms have been tested for two-dimensional flows. However, this method holds promise for three dimensional problems as well since the computational kernel is built around fast diagonalization. Fast diagonalization scales as $\mathrm{O}(\mathrm{n}^{\mathrm{d}+1})$, where *n* is the number of grid points in a single dimension and *d* is the spatial dimension. This means that where we have a calculation $\mathrm{O}(\mathrm{n}^3)$ in two dimensions, it would be replaced with a three-dimensional calculation $\mathrm{O}(\mathrm{n}^4)$. This is significantly better than competing methods based on LU factorizations, which require calculations $\mathrm{O}(\mathrm{n}^6)$.

The matrix-free nature of our algorithms has certain advantages when implemented on parallel architectures. In essence a second level of Σ' would need to be applied to couple the solution between nodes shared on different processors. Mappings such as those described in chapter 2 can be constructed between processors to allow for a geometry-free approach [12]. Additionally the memory footprint of our method is minimal since only one-dimensional operators are stored. It is our intention to extend this method to run on parallel platforms with many processors in future studies.

Chapter

Appendix A: Spectral Element Method

A-1 Tensor-Based Nodal Ordering for Domain Decomposition

In Chapters 2 and 3 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 advection-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 advection-diffusion matrix on a single element. We can decompose $\hat{F}$ and $\hat{M}$ into their interior and boundary couplings.

$\hat{F}_{ii} = \hat{F}(2:N,2:N)$ Interior-Interior

$\hat{F}_{ib} = \hat{F}(2:N,1:N+1)$ Interior-Boundary

$\hat{F}_{bi} = \hat{F}(1:N+1,2:N)$ Boundary-Interior

$\hat{F}_{bb} = \hat{F}(1,1)+\hat{F}(1,N)+\hat{F}(N,1)+\hat{F}(N,N)$ Boundary-Boundary

$\hat{B}_{ii} = \hat{B}(2:N,2:N)$ Interior-Interior

$\hat{B}_{bb} = \hat{B}(1:1,\vec{0},N+1:N+1)$ Boundary-Boundary

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

$$F_{II} = \hat{F}_{ii} \otimes \hat{B}_{ii} + \hat{B}_{ii} \otimes \hat{F}_{ii}$$

$$F_{\Gamma\Gamma} = \hat{F} \otimes \hat{B}_{bb} + \hat{B}_{bb} \otimes \hat{F} + \hat{F}_{bb} \otimes \hat{B}_{ii} + \hat{B}_{ii} \otimes \hat{F}_{bb}$$

$$F_{I\Gamma} = \hat{F}_{ib} \otimes \hat{B}_{ii} + \hat{B}_{ii} \otimes \hat{F}_{ib}$$

$$F_{\Gamma I} = \hat{F}_{bi} \otimes \hat{B}_{ii} + \hat{B}_{ii} \otimes \hat{F}_{bi}.$$

Bibliography

[1] Y. Achdou, P. Le Tallec, F. Nataf, and M. Vidrascu. A domain decomposition preconditioner for an advection-diffusion problem. *Computational Methods in Applied Mechanics and Engineering*, 184(2):145–170, 2000.

[2] I. Babŭska and M. Suri. The P and H-P versions of the finite element method, basic principles and properties. *SIAM Review*, 36(4):578–632, December 1994.

[3] S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. PETSc Web page, 2001. http://www.mcs.anl.gov/petsc.

[4] J. P. Boyd. *Chebyshev and Fourier Spectral Methods*. Dover Publications, New York, 2000.

[5] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation*, 31(138):333–390, April 1977.

[6] C.-H. Bruneau and M. Saad. The 2D lid-driven cavity problem revisited. *Computers and Fluids*, 35(3):326–348, March 2006.

[7] B. L. Buzbee, G. H. Golub, and C. W. Nielson. On direct methods for solving Poisson's equations. *SIAM Journal on Numerical Analysis*, 7(4):627–656, December 1970.

[8] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods Evolution to Complex Geometries and Applications to Fluid Dynamics*. Scientific Computation. Springer, New York, 2007.

[9] M. A. Casarin. Quasi-optimal Schwarz methods for the conforming spectral element discretization. *SIAM Journal on Numerical Analysis*, 34(6):2482–2502, December 1997.

[10] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computation*, 22(104):745–762, October 1968.

[11] H. A. Van der Vosrst and C. Vuik. Gmresr: A family of nested GMRES methods. *Numerical Linear Algebra with Applications*, 1(4):369–386, 1994.

[12] 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.

[13] J. P. Van Doormaal and G. D. Raithby. Enhancements of the SIMPLE method for predicting incompressible fluid flows. *Numerical Heat Transfer, Part A: Applications*, 7(2):147–163, April 1984.

[14] H. C. Elman. Preconditioning for the steady-state Navier-Stokes equations with low viscosity. *SIAM Journal on Scientific Computing*, 20(4):1299–1316, 1999.

[15] H. C. Elman and M. P. Chernesky. Ordering effects on relaxation methods applied to the discrete one-dimensional convection-diffusion equation. *SIAM Journal on Numerical Analysis*, 30(5):1268–1290, October 1993.

[16] H. C. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro. A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 227(3):1790–1808, January 2008.

[17] H. C. Elman, V. E. Howle, J. Shadid, D. Silvester, and R. Tuminaro. Least squares preconditioners for stabilized discretizations of the Navier-Stokes equations. *SIAM Journal on Scientific Computing*, 30(1):290–311, 2007.

[18] H. C. Elman and D. Silvester. Fast nonsymmetric iterations and preconditioning for navier-stokes equations. *SIAM Journal on Scientific Computing*, 17(1):33–46, 1996.

[19] 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.

[20] P. Fischer. An overlapping Schwarz method for spectral element solution of the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 133:84–101, 1997.

[21] P. F. Fischer. Current trends in high-order numerical methods for scientific simulation. Computing Research Institute Seminars, Purdue University, August 2004.

[22] 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. *Computer methods in Applied Mechanics and Engineering*, 196:3049–3060, 2007.

[23] V. Girault and P.-A. Raviart. *Finite Element Methods for Navier-Stokes Equations: Theory and Algorithms*. Springer Series in Computational Mathematics. Springer-Verlag, Berlin, 1986.

[24] G. H. Golub and J. M. Ortega. *Scientific Computing: An Introduction with Parallel Computing*. Academic Press, San Diego, 1993.

[25] T. Gotoh and D. Fukayama. Pressure spectrum in homogeneous turbulence. *Physical Review Letters*, 86(17):3775–3778, 2001.

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

[27] R. D. Henderson. Dynamic refinement algorithms for spectral element methods. *Computer methods in applied mechanics and engineering*, 175(3):395–411, July 1999.

[28] R. I. Issa. Solution of implicitly discretised fluid flow equations by operator splitting. *Journal of Computational Physics*, 62:40–65, 1985.

[29] J. Jimenez, A. A. Wray, P. G. Saffman, and R. S. Rogallo. The structure of intense vorticity in isotropic turbulence. *Journal of Fluid Mechanics*, 255:65–90, 1993.

[30] G. E. Karniadakis and S. J. Sherwin. *Spectral/hp Element Methods for Computational Fluid Dynamics*. Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford, 2nd edition edition, 2005.

[31] L. I. Kovasznay. Laminar flow behind a two-dimensional grid. *Proceedings of the Cambridge Philosphical Society*, 44:58–62, 1948.

[32] R. Lynch, J. Rice, and D. Thomas. Direct solution of partial difference equations by tensor product methods. *Numerische Mathematik*, 6(1):185–199, December 1964.

[33] Y. Maday, A. Patera, and E. M. Rønquist. A well-posed optimal spectral element approximation for the Stokes problem. Technical report, ICASE, Hampton, VA, 1987.

[34] Y. Maday, A. Patera, and E.M. Rønquist. An operator-integration-factor splitting method for time dependent problems: application to incompressible fluid flow. *Journal of Scientific Computation*, 5(4):263–292, December 1990.

[35] Y. Maday and A. T. Patera. Spectral element methods for the incompressible Navier-Stokes equations. In *State-of-the-art surveys on computational mechanics*, pages 71–143, New York, 1989. American Society of Mechanical Engineers.

[36] C. Mavriplis. *Nonconforming Discretizations and a Posteriori Error Estimators for Adaptive Spectral Element Techniques*. PhD thesis, Massachusetts Institute of Technology, 1989.

[37] O. A. McBryan, P. O. Frederickson, J. Linden, K. Solchenbach, K. Stüben, C. A. Thole, and U. Trottenberg. Multigrid methods on parallel computers - a survey of recent developments. *IMPACT of Computing in Science and Engineering*, 3(1):1–75, March 1991.

[38] M. F. Murphy, G. H. Golub, and A. J. Wathen. A note on preconditioning for indefinite linear systems. *SIAM Journal on Scientific Computing*, 21(6):1969–1972, 2000.

[39] S. A. Orszag. Numerical methods for the simulation of turbulence. *Physics of Fluids*, 12:II250–II257, 1969.

[40] S. A. Orszag and G. S. Patterson. Numerical simulation of three-dimensional homogeneous isotropic turbulence. *Physical Review Letters*, 28:76–79, 1972.

[41] J. Ortega and W. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Number 30 in Classics In Applied Mathematics. Society for Industrial and Applied Mathematics, 2000.

[42] S. V. Patankar. *Numerical Heat Transfer And Fluid Flow*. Taylor and Francis, 1980.

[43] S. V. Patankar and D. B. Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15(10):1787–1806, October 1972.

[44] A. T. Patera. A spectral element method for fluid dynamics - Laminar flow in a channel expansion. *Journal of Computational Physics*, 54:468–488, June 1984.

[45] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Numerical Mathematics and Scientific Computation. Oxford University Press, 1999.

[46] D. Rosenberg, A. Fournier, P. F. Fischer, and A. Pouquet. Geophysical-astrophysical spectral-element adaptive refinement (GASpAR): object-oriented h-adaptive fluid dynamics simulation. *Journal of Computational Physics*, 215(1):59–80, June 2006.

[47] Y. Saad. A flexible inner-outer preconditioned gmres algorithm. *SIAM Journal on Scientific Computing*, 14(2):461–469, 1993.

[48] A. G. Salinger, R. B. Lehoucq, R. P. Pawlowski, and J. N. Shadid. Computational bifurcation and stability analysis of the 8:1 thermal cavity problem. *International Journal for Numerical Methods in Fluids*, 40(8):1059–1073, November 2002.

[49] P. J. Schmid. Nonmodal stability theory. *Annual Review of Fluid Mechanics*, 39:129–162, January 2007.

[50] P. N. Shankar and M. D. Deshpande. Fluid mechanics in the driven cavity. *Annual Review of Fluid Mechanics*, 32:93–136, 2000.

[51] E. D. Siggia. Numerical study of small-scale intermittency in three-dimensional turbulence. *Journal of Fluid Mechanics*, 107:375–406, 1981.

[52] B. F. Smith, P. Bjorstad, and W. D. Gropp. *Domain Decomposition Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1996.

[53] V. Theofilis, D. Barkley, and S. Sherwin. Spectral/hp element technology for global flow instability and control. *The Aeronautical Journal*, 106(1065):619–625, 2002.

[54] S. J. Thomas and R. D. Loft. The NCAR spectral element climate dynamical core: Semi-implicit Eulerian formulation. *Journal of Scientific Computing*, 25(1-2):307–322, November 2005.

[55] A. Toselli and O. Widlund. *Domain Decomposition Methods - Algorithms and Theory*. Springer Series in Computational Mathematics. Springer, 2005.

[56] L. N. Trefthen and M. Embree. *Spectra and Pseudo Spectra: The behavior of nonnormal matrices and operators*. Princeton University Press, Princeton, NJ, 2005.

[57] H. Tufo and P. Fischer. Terascale spectral element algorithms and implementations. In *Supercomputing '99: Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM)*, page 68, New York, NY, USA, 1999. ACM Press.

[58] A. Vincent and M. Meneguzzi. The spatial structure and statistical properties of homogeneous turbulence. *Journal of Fluid Mechanics*, 225:1–20, 1991.

[59] M. Yokokawa, K. Itakura, A. Uno, T. Ishihara, and Y. Kaneda. 16.4-tflops direct numerical simulation of turbulence by a fourier spectral method on the earth simulator. In *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–17. IEEE Computer Society Press, 2002.

[60] D. M. Young. Iterative methods for solving partial difference equations of elliptic type. *Transactions of the American Mathematical Society*, 76:92–111, 1954.

[61] D. M. Young. *Iterative Solution of Large Linear Systems*. Academic Press, New York, 1971.