

EFFICIENT SOLVER FOR MIXED AND CONTROL-VOLUME MIXED
FINITE ELEMENT METHODS IN THREE DIMENSIONS

by

John D. Wilson

B.S., University of Colorado at Denver, 1991

M.S., University of Colorado at Denver, 1994

A thesis submitted to the
University of Colorado at Denver
in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Applied Mathematics

2001

This thesis for the Doctor of Philosophy

degree by

John D. Wilson

has been approved

by

Thomas F. Russell

Lynn S. Bennethum

Leo Franca

Jan Mandel

Gita Alaghband

Date

Wilson, John D. (Ph.D., Applied Mathematics)

Efficient Solver for Mixed and Control-Volume Mixed Finite Element Methods
in Three Dimensions

Thesis directed by Professor Thomas F. Russell

ABSTRACT

In simulations of ground water flow in heterogeneous aquifers, accuracy of velocities can be substantially enhanced by discretizations based on mixed or control-volume mixed finite element methods. A potential side benefit is improved accuracy of transport calculations that depend on the flow field. Mixed finite element linear systems are not positive definite and cannot be solved by straightforward applications of well-known iterative solvers. We present a specialized iterative solver for mixed methods in 3-D, built around a decomposition of the discrete velocity space that leads to a symmetric positive definite system (the pressure variable is eliminated). Our discretization uses distorted hexahedral elements. Hexahedral elements are often used in practice for modeling the subsurface hydrogeology. Lowest-order Raviart-Thomas vector functions are used for the velocity space. These vector functions have continuous normal components which gives us conservation of mass.

Since the reduced system is symmetric positive definite we can use the preconditioned conjugate gradient method. An additive domain decomposition preconditioner is described and the uniform convergence of the method is established for

small overlap. Numerical results confirm the uniform convergence. Parallelization of the problem is also demonstrated. A similar decomposition of the velocity space is being investigated for the control-volume mixed discretization, hoping to obtain a positive definite system, although the system is not, in general, symmetric.

This abstract accurately represents the content of the candidate's thesis. I recommend its publication.

Signed _____
Thomas F. Russell

ACKNOWLEDGMENT

The research of the author was supported in part by National Science Foundation Grant No. DMS-9706866 and Army Research Office Grant No. 37119-GS-AAS.

CONTENTS

Figures	ix
Tables	xi
<u>Chapter</u>	
1. Introduction	1
1.1 Existing Methods	2
1.2 The New Results in this Thesis	5
1.3 Thesis Outline	6
1.4 Future Work	8
2. Governing Equations	10
2.1 Saturated Flow	10
2.1.1 Conservation of Mass	12
2.1.2 Darcy Flow	13
2.2 Flow Equation	14
3. Mixed Finite Element Method	16
3.1 Mixed Variational Formulation	16
3.2 Divergence-Free Subspace	19
3.3 Existence and Uniqueness	20
4. Discretization	25
4.1 Trilinear Hexahedral Elements	27
4.2 Approximation Spaces	34
4.2.1 Properties of the Piola Transformation	38

4.2.2	Interpolation of Velocity Functions	41
4.3	Divergence-Free Basis	45
4.3.1	Dirichlet Boundary Conditions	50
4.4	Initial Guess Vector \mathbf{v}_I^h	61
4.5	Mass and Stiffness Matrices	64
4.5.1	Mass Matrix	68
4.5.1.1	Mass Matrix with Boundary Fluxes	73
4.5.2	Stiffness Matrix	79
4.6	Numerical Integration	84
5.	Preconditioners	89
5.1	Vector Potential Space	90
5.2	Condition Number of Stiffness Matrix	95
5.3	Domain Decomposition Preconditioner	97
5.3.1	Convergence Analysis	102
5.4	Preconditioner for Bordered Matrix	106
5.4.1	Analysis of Bordered Matrix Preconditioner	110
6.	Numerical Results	115
6.1	PCG Performance	117
6.1.1	Test Case I	118
6.1.2	Test Case II	119
6.1.3	Test Case III	119
6.1.4	Test Case IV	122
6.2	Model Problem I	124
6.2.1	Parallelization of Model Problem I	128

6.3	Model Problem II	130
6.4	Error Analysis	136
<u>Appendix</u>		
A.	Subdomain Restriction and Interpolation	139
B.	Coarse-Grid Restriction and Interpolation	144
	<u>References</u>	149

FIGURES

Figure

2.1	D’Arcy’s Experiment	14
4.1	Trilinear Mapping of Reference Cube	28
4.2	Random Hexahedral Grid	28
4.3	Uniform Flow on Distorted Element	43
4.4	Relative Error of Interpolated Flux	44
4.5	Boundary Divergence-Free Basis Vector	51
4.6	Divergence-Free Pipe Function	55
4.7	Divergence-Free Corner Functions	56
4.8	Connectivity Graph of Dirichlet Boundaries	59
4.9	Spanning Tree	60
4.10	Non-Zero Structure of \mathbf{M}_0 on Non-Orthogonal Grid	73
4.11	Non-Zero Structure of \mathbf{A}_1	76
4.12	Non-Zero Structure of \mathbf{A}_2	77
4.13	Non-Zero Structure of \mathbf{M}	78
4.14	Non-Zero Structure of \mathbf{CMC}^T	79
4.15	Non-Zero Structure of \mathbf{CMC}^T with Boundary Contributions	80
4.16	Matrix Permutation	83
5.1	Partitioned of Domain into Subdomains	98
6.1	Reduction Histories for Test Case I	118
6.2	Low Conductivity Block	120

6.3	Test Case III on $4 \times 4 \times 4$ Coarse-Grid	121
6.4	Test Case III on $8 \times 8 \times 8$ Coarse-Grid	121
6.5	Reduction Histories for Test Case IV	123
6.6	Cylinder on $4 \times 4 \times 4$ Grid	126
6.7	Cylinder on $16 \times 16 \times 16$ Grid	127
6.8	Flow Around Cylinder	127
6.9	Computed Flow Lines for Tank Experiment	131
6.10	Computed Flow Lines for Tank Experiment with Sink Term	134
B.1	Coarse Grid x -Slice Function	145

TABLES

Table

4.1	Reference Coordinates and Physical Coordinates	29
5.1	Minimum and Maximum Eigenvalues	97
6.1	Preconditioner Assembly Time	116
6.2	PCG Method for Test Case I	118
6.3	PCG Method for Test Case II	119
6.4	PCG Method for Test Case III	120
6.5	PCG Method for Test Case IV	122
6.6	PCG Method for Cylinder Problem	126
6.7	Speedup on $16 \times 16 \times 16$ Grid	128
6.8	Speedup on $32 \times 32 \times 16$ Grid	128
6.9	Sand Characteristics	132
6.10	Computing P1	132
6.11	Computing P2	133
6.12	PCG Method Using P1	133
6.13	PCG Method Using P2	133
6.14	PCG Method for Model Problem II with Sink Using P1	135
6.15	PCG Method for Model Problem II with Sink Using P2	135
6.16	Extrapolated Error Estimate	138

1. Introduction

Our goal is to develop an efficient solver for numerical methods modeling three-dimensional saturated flow in a heterogeneous anisotropic porous medium. Saturated flow is modeled by a second order partial differential equation (PDE). For example, we will consider

$$-\nabla \cdot \mathbf{K} \nabla p = f \quad \text{in } \Omega \tag{1.1}$$

where the pressure, p , or hydraulic head is the primary variable to be solved for and Ω is a bounded three-dimensional domain. Approximating a solution to a PDE, such as (1.1), is typically done through a discretization either using a finite difference or finite element method. The result of the discretization is a large sparse matrix problem. It is desirable to use an iterative method for sparse matrix problems. If the matrix is symmetric positive definite, then the preconditioned conjugate gradient (PCG) method can be used. The efficiency of the PCG method depends on the condition number of the matrix. The coefficient \mathbf{K} can be highly discontinuous and this can cause problems for standard numerical methods.

The velocity, or flux, can be of primary interest in hydrogeological models. An equation for the velocity in terms of the pressure is often written as:

$$\mathbf{v} = -\mathbf{K} \nabla p. \tag{1.2}$$

This leads to a system of first order equations which can then be solved using a mixed finite element (MFE) method or a control-volume mixed finite element

(CVMFE) method. Simultaneous approximations for the velocity and the pressure can then be computed. This approach will lead to better approximations of the velocity. A sophisticated implementation of the three-dimensional solute-transport equation has recently been done in [42]. This type of implementation requires an accurate description of the flow field for mass tracking.

The discretization of either the MFE method or the CVMFE method leads to a semidefinite matrix problem. The PCG method can no longer be used because the matrix is not positive definite. Our method is to eliminate the pressure through a decomposition of the velocity space using a divergence-free subspace. When we eliminate the pressure variable from the MFE method, and discretize the problem, we end up with a symmetric positive definite matrix. If the diameters of the elements are order h , then the condition number of the matrix is of order h^{-2} . This means that the PCG method will perform poorly for small h . Therefore, a key part of our method will be to implement a domain decomposition preconditioner which will make the convergence rate of the PCG method independent of grid size.

1.1 Existing Methods

The earliest known domain decomposition method was introduced by Schwarz in 1870. The Schwarz method is an overlapping method, and thus, all overlapping methods can be referred to as Schwarz methods. The two level additive Schwarz method is due to Dryja and Widlund [24]. Glowinski and Wheeler formulated the first domain decomposition methods for MFE in [37]. Domain decomposition methods for MFE have also been analyzed in [29],[20], and [17].

Convergence analyses for Schwarz methods are often dependent on a large overlap. However, numerical results usually show that the rate of convergence for a Schwarz method is quite satisfactory even for a small overlap of size h . Dryja and Widlund in [27] were able to analyze this behavior.

The construction of a divergence-free subspace in three dimensions on a rectangular grid with pure homogeneous Neumann boundary conditions is described in [17]. A convergence analysis for the domain decomposition method for this problem is also given. This convergence analysis relies on a sufficiently large overlap of order H which is the approximate diameter of a coarse-grid element.

More recently, a decoupling of the pressure and velocity spaces using the divergence-free subspace was described by Scheichl in [54]. The method was implemented on triangle elements in two dimensions and on tetrahedra in three dimensions. A bound on the eigenvalues of the resulting stiffness matrix is analyzed giving a condition number of order h^{-2} . Various iterative solvers were tested and compared. It was clear that the PCG method using a domain decomposition preconditioner performed the best for the two-dimensional case. Results for domain decomposition in three dimensions were not given. The PCG method with additive Schwarz preconditioner was implemented using the `DOUG` package [40]. One of the important differences between the method in [17] and the method in [54] is that the divergence-free subspace is known a priori in [17]. In [54], a decoupling stage is performed as part of the assembly process, determining the divergence-free basis using a graph theoretical approach.

A mixed finite element method for second order elliptic problems is described by Raviart and Thomas in [52]. This early work sets the framework for ana-

lyzing the mixed finite element method. A discretization of the velocity space and pressure space is described in two dimensions for triangles and rectangles. The triangles and rectangles are affine mappings of a reference element. The Piola transformation is used to map the velocity from the reference element to the physical element. The use of the Piola transformation is critical. Using the properties of the Piola transformation, one can describe an interpolation operator and approximations for smooth vector valued functions onto the discrete velocity space. It also leads to satisfying the discrete inf-sup condition. This work was later extended to three dimensions by Nedelec in [51]. For three-dimensional distorted hexahedral grids the mapping of the reference cube to a hexahedral element is not affine. In this case the theory does not apply. In Thomas's thesis [57] the interpolation of smooth vector valued functions is described for two-dimensional quadrilaterals and error estimates are given. If an interpolation estimate were known for three-dimensional hexahedral elements, then it could be used along the lines of Thomas's thesis to verify the inf-sup condition and obtain error estimates. In particular, for the special case of three-dimensional elements that are products of two-dimensional quadrilaterals with a one-dimensional interval (quadrilateral prisms), Thomas's theory would apply.

A more complete mathematical theory of mixed finite element methods by Brezzi and Fortin can be found in [15]. Many different methods are discussed using model problems. The goal was to provide an analysis of the methods in order to understand their properties as much as possible. A discussion of the properties of the $H(\text{div}, \Omega)$ velocity space is given. Mixed finite element methods constantly make use of this space. A very general framework is given for existence

and uniqueness of solutions in [52] along with approximation properties.

One may also find the analysis given by Brenner and Scott in [12] to be useful for understanding existence and uniqueness of the mixed finite element method. Between [52], [15], and [12], the mathematical theory of mixed finite element methods is almost complete. Practical implementation details can be found in [44] and [46].

1.2 The New Results in this Thesis

We extend the work in [17] by constructing a divergence-free subspace in three dimensions for general hexahedral elements which are trilinear mappings of a unit cube. We also introduce Dirichlet and Neumann boundary conditions. The distorted hexahedral elements are considered so as to better model the structure of the subsurface heterogeneity. Hexahedral elements are widely used in hydrogeology therefore they have an advantage over tetrahedral elements.

We analyze the upper and lower bounds on the eigenvalues of the stiffness matrix. These bounds agree with those found in [54] for tetrahedra and are confirmed through numerical tests. A convergence analysis for the PCG method using a two level additive Schwarz method is then given for a small overlap of order h . Although this analysis is standard, we are able to confirm, through numerical results, that the convergence rate of the preconditioned system is independent of grid size and number of subdomains. We also show the effects of heterogeneity and anisotropy.

Finally, it is shown (through numerical results) that the method is highly parallelizable. This is due to being able to solve the subdomain problems independently.

Our implementation is done completely from scratch using the C language. The only exception is the use of a Cholesky factorization module downloaded from the Netlib repository [1]. The Cholesky factorization is used for solving subdomain problems and the coarse-grid problem. Very close attention is paid to structured programming techniques. Structured programming using the C language can be very much like object-oriented programming. This is due to the ability to use callback functions. Callback functions are functions which are passed as arguments into other functions. These callback functions can also be represented as data members of a structure. Therefore, C structures are really the building blocks for classes in object-oriented programming.

The implementation of our method along with the use of hexahedral elements constitutes new and original research. This research contributes to the body of scientific knowledge by providing a model, with numerical results, which can be extended and compared to existing and future implementations.

1.3 Thesis Outline

In Chapter 2 we derive the governing equations which give rise to a second order PDE. The derivation is fairly standard. However, more sophisticated models can arise from variations of these governing equations. It is important to understand the physical principles involved. Boundary conditions are also given. The second order PDE is then transformed into a system of first order PDE's using the flux variable.

The MFE method, described in Chapter 3, is used to solve the system of first order PDE's. Approximation spaces for the velocity and pressure are defined. Existence and uniqueness are well known for the MFE method using these spaces.

We discuss the existence and uniqueness proof for the MFE method as given in [12] where coercivity of the problem is shown using a divergence-free subspace. The divergence-free subspace is also a primary feature of our method.

We begin the discussion of the discretization of the MFE method in Chapter 4 by first describing the trilinear mapping of the reference cube to a hexahedral element. Next, we describe the lowest-order Raviart-Thomas basis vector functions for the velocity space. These vector functions are defined on a reference cube and then mapped to the hexahedral elements using the Piola transformation. The properties of the Piola transformation are discussed. As mentioned above, the interpolation estimate for three-dimensional non-affine elements is still an open question. We describe the theory presented in Thomas's thesis for two-dimensional quadrilaterals and note that in special cases the theory will apply to three-dimensional elements.

Continuing with chapter 4 we describe the discrete divergence-free subspace. We extend the work done in [17] to non-homogeneous boundary conditions. Once the discrete divergence-free subspaces have been defined we can derive the matrix equation which is to be solved using the PCG method. Some discussion about the non-zero structure of the stiffness matrix is also given.

In Chapter 5 we analyze the condition number of the stiffness matrix. The domain decomposition preconditioner is defined and a convergence analysis is given. If we have two Dirichlet boundaries which are disconnected, then we end up with a single dense column and row in the stiffness matrix. We describe this bordered matrix in Chapter 5 and we define a bordered matrix preconditioner. We are able to analyze the convergence properties of the preconditioned bordered

stiffness matrix.

Finally, in Chapter 6 we demonstrate the efficiency of the method in terms of iteration counts and CPU times. We also describe two model problems. The first model problem is a vertical cylinder of low permeability. We demonstrate that the cylinder problem can be parallelized with near-perfect speed up. The second model problem is a laboratory tank experiment. This model results in a bordered stiffness matrix. The numerical results of the bordered matrix problem confirm the analysis given in Chapter 5. We also extrapolate error estimates for the three-dimensional problem. At present we do not have a good three-dimensional model problem where the exact solution is known. Therefore, we use an extrapolation technique and show that the convergence of the error is of order h which agrees with Thomas's theory for lowest-order Raviart-Thomas velocities.

1.4 Future Work

The analysis for domain decomposition algorithms with small overlap is given by Dryja and Widlund in [27]. Domain decomposition methods for divergence-free subspaces have been analyzed before, for example, in [13]. However, these methods are usually defined on triangle elements in 2-D. Sharp bounds on the minimum eigenvalue of the preconditioned system for divergence-free subspaces can be shown in 2-D for triangles and quadrilaterals. A sharp bound in three dimensions on hexahedral elements may still be an open question and needs to be investigated further.

Even though we were able to show the ability to parallelize our method, we have not implemented it on a large parallel machine. It will be important to test the method on a larger computer.

Another extension to the work in this thesis would be to implement the CVMFE method using our solver. It is shown in [16] that the CVMFE method in two dimensions leads to more accurate velocities even in the presence of discontinuous anisotropic coefficients on a distorted grid. The implementation of the CVMFE method in three dimensions would be new. In the CVMFE method, the test space for the velocity has discontinuous normal components and thus, is not in $H(\text{div}, \Omega)$. We propose a divergence-free test space which is weakly divergence-free. A proposal to implement the CVMFE method using the solver described in this thesis has been submitted, and accepted, by the National Research Council. Work on this project should begin soon at the U.S. Geological Survey Hydrogeology Division.

2. Governing Equations

We will derive a second order PDE for the pressure. This particular PDE is referred to as a second order elliptic boundary value problem. The resulting flow equation is given in Section 2.2. A flux variable will be introduced in the derivation of the second order PDE. This flux variable will later be used to transform the second order PDE into a system of first order equations.

The problem we are interested in modeling is saturated subsurface flow. Saturated flow means that whatever fraction of the volume is available to a fluid is filled by that fluid. The fraction of the volume available to the fluid is called the effective porosity of the medium. We call this type of medium a porous medium.

2.1 Saturated Flow

The derivation of the second order elliptic boundary value problem is fairly standard and can be found in most PDE textbooks, for example, [39]. However, we want to emphasize the physical properties specific to hydrogeology. Units of measure and field techniques specific to hydrogeology may be found in [30].

We will begin by considering an underground volume, V , along with its boundary ∂V . The underground volume will consist of soil and rock and will have certain physical properties. The fluid, water in our case, will also have physical properties. The physical properties of the saturated flow system are given below. The dimensions are given in SI units.

- Effective Porosity - ϕ (dimensionless)

- Intrinsic Permeability - \mathbf{K}_i (m^2)
- Specific Discharge - \mathbf{v} (m/s)
- Pressure - P ($\text{kg}/(\text{ms}^2)$)
- Density of Fluid - ρ (kg/m^3)
- Viscosity - μ ($\text{kg}/(\text{ms})$)
- Source/Sink - f ($\text{kg}/(\text{m}^3\text{s})$)
- Gravitational Constant - a (m/s^2)

The effective porosity is the fraction of the porous medium which is available to hold a fluid. This is not the same as porosity which is the fraction of the porous medium which is void of material. Some of the voids may not be interconnected thus, fluid may not flow in or out of these voids. When we say saturated flow we mean that the fraction of fluid in the porous medium is equal to the effective porosity. We assume that the effective porosity does not change with time.

The intrinsic permeability is a property of the porous medium alone. It is basically a measure of the square of the mean pore diameter and thus, has the units of area. The permeability is a symmetric 3×3 matrix which is spatially dependent. We assume that it satisfies the following ellipticity condition:

$$\alpha_1 \mathbf{w}^t \mathbf{w} \leq \mathbf{w}^t \mathbf{K}_i \mathbf{w} \leq \alpha_2 \mathbf{w}^t \mathbf{w}, \quad \forall \mathbf{w} \in \mathfrak{R}^3 \quad (2.1)$$

Later, we will define a hydraulic conductivity which is measured in units of length over time.

The specific discharge has units of velocity. However, it is not a true velocity. Specific discharge is a measure of volume per time per cross-sectional area. The cross-sectional area is partially blocked by the porous medium thus, reducing the volume of fluid which passes through it. To get an average velocity or seepage velocity we would need to divide the specific discharge by the effective porosity.

The pressure is measured as force over cross-sectional area. The standard unit of measure is the Pascal (Pa). Hydraulic head is often used instead of pressure. Hydraulic head is measured in units of length and the hydraulic gradient is dimensionless.

We assume that gravity is in the vertical direction. Therefore, we can write the gravitational vector \mathbf{g} as:

$$\mathbf{g} = \begin{pmatrix} 0 \\ 0 \\ a \end{pmatrix}. \quad (2.2)$$

2.1.1 Conservation of Mass

The rate of change of the mass of the fluid in volume V is given by

$$\frac{\partial}{\partial t} \int_V \rho \phi dV = \int_V \rho_t \phi dV. \quad (2.3)$$

This rate of change in the mass must be equal to the net rate at which mass crosses the boundary of the volume minus the rate at which mass is produced from the volume plus the rate at which mass is injected into the volume. This relationship is given by

$$\int_V \rho_t \phi dV = - \int_{\partial V} \rho \mathbf{v} \cdot \mathbf{n} d\partial V + \int_V f dV. \quad (2.4)$$

We can use the divergence theorem to write the boundary integral as a volume integral giving

$$\int_V \rho_t \phi dV = - \int_V \operatorname{div} (\rho \mathbf{v}) dV + \int_V f dV. \quad (2.5)$$

2.1.2 Darcy Flow

Darcy's law is often expressed as:

$$\mathbf{v} = -\frac{\mathbf{K}_i}{\mu} (\nabla P + \rho \mathbf{g}) \quad (2.6)$$

In the mid-1800s, a French engineer, Henri D'Arcy, made the first systematic study of the movement of water through a porous medium. We can demonstrate Darcy's experiment by considering a horizontal pipe filled with sand. Water is applied under pressure at one end of the pipe at point A . The pressure can be measured through a vertical tube at that end of the pipe. Another vertical tube at the other end of the pipe can be used to measure the pressure there. The pressure is a measurement of how high the water is forced up these vertical tubes (see Figure 2.1).

Darcy found experimentally that the discharge, Q , is proportional to the gradient of the pressure, $(h_a - h_b)/L$. The discharge also depends on the cross-sectional area, A . Combined with a proportionality constant K we have:

$$Q = -KA \frac{dh}{dl} \quad (2.7)$$

We define the specific discharge as $\mathbf{v} = Q/A$.

The proportionality constant, K , is a property of the sand. In general, we will use the hydraulic conductivity tensor to represent this proportionality constant.

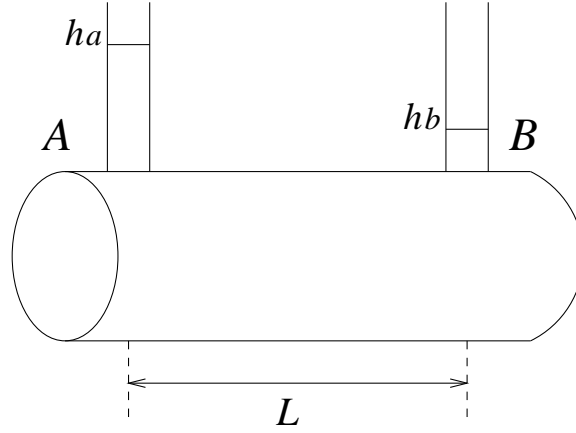


Figure 2.1: D'Arcy's Experiment

With this experiment in mind, and assuming constant density of water, we rewrite equation (2.6) as:

$$\mathbf{v} = -\mathbf{K}\nabla h \quad (2.8)$$

where the hydraulic conductivity, \mathbf{K} , is given by

$$\mathbf{K} = \mathbf{K}_i \frac{\rho a}{\mu} \quad (2.9)$$

and the hydraulic head, h , is given by

$$h = \frac{P}{\rho a} + z. \quad (2.10)$$

2.2 Flow Equation

We substitute (2.8) into (2.5) to get

$$0 = \int_{\mathcal{V}} \operatorname{div}(\rho \mathbf{K} \nabla h) dV + \int_{\mathcal{V}} f dV \quad (2.11)$$

where the left-hand side is zero because the density is constant, i.e., $\rho_t = 0$.

We now consider a three-dimensional domain Ω and its boundary $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$. Since (2.11) is defined for any arbitrary volume, $V \subset \Omega$, we drop the integrals and add boundary conditions to get

$$-\nabla \cdot \mathbf{K} \nabla h = (f/\rho) \quad \text{in } \Omega \quad (2.12)$$

$$h = q \quad \text{on } \partial\Omega_D \quad (2.13)$$

$$(\mathbf{K} \nabla h) \cdot \mathbf{n} = g \quad \text{on } \partial\Omega_N \quad (2.14)$$

where $\partial\Omega_D$ is the Dirichlet boundary and $\partial\Omega_N$ is the Neumann boundary. Equations (2.12)-(2.14) define the second order boundary value problem, or in our case, the flow equation.

We can rewrite (2.12) as a system of first order equations using the flux variable, \mathbf{v} , defined in equation (2.8). First, we rewrite (2.8) as

$$\mathbf{K}^{-1} \mathbf{v} = -\nabla h. \quad (2.15)$$

Then, we substitute (2.15) into (2.12) to get the first order equation

$$\nabla \cdot \mathbf{v} = (f/\rho) \quad \text{in } \Omega \quad (2.16)$$

Combining (2.15) with (2.16) and the boundary conditions (2.13)-(2.14) we get our system of first order equations

$$\begin{cases} \mathbf{K}^{-1} \mathbf{v} + \nabla h = 0 & \text{in } \Omega \\ \nabla \cdot \mathbf{v} = (f/\rho) & \end{cases} \quad (2.17)$$

$$h = q \quad \text{on } \partial\Omega_D \quad (2.18)$$

$$(\mathbf{K} \nabla h) \cdot \mathbf{n} = g \quad \text{on } \partial\Omega_N \quad (2.19)$$

We will use the MFE method to define a solution for this problem. From now on we will write f/ρ as just f and we will write the hydraulic head, h , as p .

3. Mixed Finite Element Method

To compute an approximate solution to the first order system of equations defined in (2.17)-(2.19) we use the MFE method. We first define a weak form of the system of equations. Existence and uniqueness is then shown for the weak form. The weak form is also sometimes called the variational form or mixed variational form. This is because the weak form can be derived from the calculus of variations [39].

3.1 Mixed Variational Formulation

The variational form will require us to define appropriate function spaces. For mixed finite element methods we use constantly the space

$$\mathbf{V} = H(\operatorname{div}, \Omega) = \{\mathbf{w} \in (L^2(\Omega))^3 : \operatorname{div} \mathbf{w} \in L^2(\Omega)\} \quad (3.1)$$

with the norm

$$\|\mathbf{w}\|_{\mathbf{V}}^2 = \|\mathbf{w}\|_{L^2(\Omega)^3}^2 + \|\operatorname{div} \mathbf{w}\|_{L^2(\Omega)}^2 \quad (3.2)$$

$$\|\mathbf{w}\|_{L^2(\Omega)^3}^2 = \int_{\Omega} \mathbf{w} \cdot \mathbf{w} \, d\Omega \quad (3.3)$$

$$\|\operatorname{div} \mathbf{w}\|_{L^2(\Omega)}^2 = \int_{\Omega} (\operatorname{div} \mathbf{w})^2 \, d\Omega. \quad (3.4)$$

It is possible to define the normal trace $\mathbf{w} \cdot \mathbf{n}$ of a function $\mathbf{w} \in \mathbf{V}$ on $\partial\Omega$, in $H^{-1/2}(\partial\Omega)$.

Lemma 3.1 For $\mathbf{w} \in \mathbf{V}$, we can define $\mathbf{w} \cdot \mathbf{n}|_{\partial\Omega} \in H^{-1/2}(\partial\Omega)$ and we have Green's formula,

$$\int_{\Omega} \operatorname{div} \mathbf{w} p \, d\Omega + \int_{\Omega} \mathbf{w} \cdot \nabla p \, d\Omega = \langle \mathbf{w} \cdot \mathbf{n}, p \rangle_{\partial\Omega}, \quad \forall p \in H^1(\Omega) \quad (3.5)$$

Proof. See Brezzi and Fortin [15, Lemma III.1.1] ■

The duality $\langle \mathbf{w} \cdot \mathbf{n}, p \rangle_{\partial\Omega}$, between $H^{-1/2}(\partial\Omega)$ and $H^{1/2}(\partial\Omega)$ is defined as

$$\langle \mathbf{w} \cdot \mathbf{n}, p \rangle_{\partial\Omega} = \int_{\partial\Omega} \mathbf{w} \cdot \mathbf{n} p \, ds. \quad (3.6)$$

We can use this definition of $\mathbf{w} \cdot \mathbf{n}|_{\partial\Omega}$ to define the following subspace of \mathbf{V} :

$$\mathbf{V}_0 = H_{0,N}(\text{div}, \Omega) = \{ \mathbf{w} \in \mathbf{V} : \langle \mathbf{w} \cdot \mathbf{n}, p \rangle_{\partial\Omega} = 0 \quad \forall p \in H_{0,D}^1(\Omega) \} \quad (3.7)$$

where

$$H_{0,D}^1(\Omega) = \{ p \in H^1(\Omega) : p|_{\partial\Omega_D} = 0 \} \quad (3.8)$$

The space (3.7) contains functions of \mathbf{V} whose normal traces vanish on $\partial\Omega_N$. It is stated in [15] that, for reasons related to pathological properties of $H^{1/2}(\partial\Omega_D)$ and $H^{-1/2}(\partial\Omega_N)$, it is necessary to use definition (3.7) and not an expression such as $\mathbf{w} \cdot \mathbf{n} = 0$ in $H^{-1/2}(\partial\Omega_N)$.

For example, we would not, at this time, define \mathbf{V}_0 to be the space given by

$$H_0(\text{div}, \Omega) = \{ \mathbf{w} \in \mathbf{V} : \mathbf{w} \cdot \mathbf{n} = 0 \text{ on } \partial\Omega_N \} \quad (3.9)$$

We will define the pressure space as:

$$\Lambda = L^2(\Omega) \quad (3.10)$$

with norm

$$\|\lambda\|_{\Lambda} = \|\lambda\|_{L^2(\Omega)}. \quad (3.11)$$

In the case of a purely homogeneous Neumann boundary condition, we have the additional constraint given by:

$$\Lambda_0 = \{ \lambda \in L^2(\Omega) : \int_{\Omega} \lambda \, d\Omega = 0 \} \quad (3.12)$$

which factors out the constant functions from $L^2(\Omega)$.

We multiply the first equation in (2.17) by a vector function $\mathbf{w} \in \mathbf{V}_0$ and integrate over Ω giving

$$\int_{\Omega} \mathbf{K}^{-1} \mathbf{v} \cdot \mathbf{w} \, d\Omega + \int_{\Omega} \nabla p \cdot \mathbf{w} \, d\Omega = 0. \quad (3.13)$$

We use integration by parts and the boundary condition, (2.18), to get

$$\int_{\Omega} \mathbf{K}^{-1} \mathbf{v} \cdot \mathbf{w} \, d\Omega - \int_{\Omega} (\nabla \cdot \mathbf{w}) p \, d\Omega = - \langle \mathbf{w} \cdot \mathbf{n}, q \rangle_{\partial\Omega} \quad (3.14)$$

Note that the Neumann boundary condition is an essential boundary condition and we specify a homogeneous Neumann boundary in the test space so that it disappears when we integrate by parts. The Dirichlet boundary is referred to as a natural boundary condition.

Next, we multiply the second equation in (2.17) by a test function in Λ (or Λ_0 for pure homogeneous Neumann boundary) and integrate over Ω to get

$$\int_{\Omega} (\nabla \cdot \mathbf{v}) \lambda \, d\Omega = \int_{\Omega} f \lambda \, d\Omega. \quad (3.15)$$

We introduce the following bilinear forms and linear functionals:

$$a(\mathbf{v}, \mathbf{w}) = \int_{\Omega} \mathbf{K}^{-1} \mathbf{v} \cdot \mathbf{w} \, d\Omega \quad (3.16)$$

$$b(\mathbf{w}, p) = \int_{\Omega} \nabla \cdot \mathbf{w} p \, d\Omega \quad (3.17)$$

$$G(\mathbf{w}) = \langle \mathbf{w} \cdot \mathbf{n}, q \rangle_{\partial\Omega} \quad (3.18)$$

$$F(\lambda) = \int_{\Omega} f \lambda \, d\Omega. \quad (3.19)$$

Now, problem (2.17) is stated in the weak form as: Find $\mathbf{v} \in \mathbf{V}_0$ and $p \in \Lambda$ such that

$$\begin{cases} a(\mathbf{v}, \mathbf{w}) - b(\mathbf{w}, p) = -G(\mathbf{w}), & \forall \mathbf{w} \in \mathbf{V}_0 \\ b(\mathbf{v}, \lambda) = F(\lambda), & \forall \lambda \in \Lambda \end{cases}. \quad (3.20)$$

If we have non-zero Neumann boundary conditions, say $\mathbf{v} \cdot \mathbf{n} = g$ on $\partial\Omega_N$, then we will need to consider $\mathbf{v} \in \mathbf{V}$ given by:

$$\mathbf{v} = \mathbf{v}_0 + \mathbf{v}_g \quad (3.21)$$

with $\mathbf{v}_0 \in \mathbf{V}_0$ and \mathbf{v}_g an arbitrary fixed element of \mathbf{V}_N where

$$\mathbf{V}_N = \{\mathbf{w} \in H(\text{div}, \Omega) : \langle \mathbf{w} \cdot \mathbf{n}, p \rangle_{\partial\Omega} = \langle g, p \rangle_{\partial\Omega}, \quad \forall p \in H_{0,D}^1(\Omega)\}. \quad (3.22)$$

We substitute this into (3.20) to get

$$\begin{cases} a(\mathbf{v}_0, \mathbf{w}) - b(\mathbf{w}, p) = -G(\mathbf{w}) - a(\mathbf{v}_g, \mathbf{w}), & \forall \mathbf{w} \in \mathbf{V}_0 \\ b(\mathbf{v}_0, \lambda) = F(\lambda) - b(\mathbf{v}_g, \lambda), & \forall \lambda \in \Lambda \end{cases}. \quad (3.23)$$

3.2 Divergence-Free Subspace

Problem (3.23) is symmetric, but indefinite. We can decompose the velocity space using a divergence-free subspace. In this way, we can eliminate the pressure from problem (3.23). The result is a symmetric positive definite form of the problem.

The divergence-free subspace is defined as:

$$\mathbf{D} = \{\mathbf{w} \in \mathbf{V}_0 : b(\mathbf{w}, \lambda) = 0 \quad \forall \lambda \in \Lambda\}. \quad (3.24)$$

We want to find a vector, $\mathbf{v}_I + \mathbf{v}_g \in \mathbf{V}$, where $\mathbf{v}_g \in \mathbf{V}_N$ and $\mathbf{v}_I \in \mathbf{V}_0$ satisfies

$$b(\mathbf{v}_I, \lambda) = F(\lambda) - b(\mathbf{v}_g, \lambda), \quad \forall \lambda \in \Lambda. \quad (3.25)$$

There are many such vectors and similarly we can easily construct one which satisfies:

$$\text{div } \mathbf{v}_I = f - \text{div } \mathbf{v}_g \quad (3.26)$$

and thus satisfies (3.25). We then write $\mathbf{v} \in \mathbf{V}$ as a vector with the correct divergence, $\mathbf{v}_I + \mathbf{v}_g$, plus a divergence-free correction as:

$$\mathbf{v} = \mathbf{v}_D + (\mathbf{v}_I + \mathbf{v}_g). \quad (3.27)$$

The divergence-free correction, $\mathbf{v}_D \in \mathbf{D}$, satisfies

$$a(\mathbf{v}_D, \mathbf{w}) = -a(\mathbf{v}_I + \mathbf{v}_g, \mathbf{w}) - G(\mathbf{w}), \quad \forall \mathbf{w} \in \mathbf{D} \quad (3.28)$$

It can be shown that $a(\cdot, \cdot)$ is a coercive, continuous bilinear form, and by the Lax-Milgram theorem, (3.28) is well posed.

With \mathbf{v}_D well-defined as the solution to (3.28), we then determine $p \in \Lambda$ such that

$$b(\mathbf{w}, p) = -a(\mathbf{v}, \mathbf{w}) - G(\mathbf{w}) \quad \forall \mathbf{w} \in \mathbf{V}_0. \quad (3.29)$$

The well-posedness of this problem will rely on a special kind of coercivity for $b(\cdot, \cdot)$.

3.3 Existence and Uniqueness

We will first show that the two bilinear forms, $a(\cdot, \cdot)$ and $b(\cdot, \cdot)$, are continuous.

Lemma 3.2 There exist positive constants, C_a and C_b , such that for all $\mathbf{u}, \mathbf{v} \in \mathbf{V}$ and for all $p \in \Lambda$ we have

$$a(\mathbf{u}, \mathbf{v}) \leq C_a \|\mathbf{u}\|_{\mathbf{V}} \|\mathbf{v}\|_{\mathbf{V}} \quad (3.30)$$

$$b(\mathbf{v}, p) \leq C_b \|\mathbf{v}\|_{\mathbf{V}} \|p\|_{\Lambda}. \quad (3.31)$$

Proof. The proof for (3.30) and (3.31) is essentially an application of the Schwarz inequality. Clearly, $\|\mathbf{u}\|_{\mathbf{V}}^2 \geq \|\mathbf{u}\|_{L^2(\Omega)^3}^2$. Therefore,

$$\|\mathbf{u}\|_{\mathbf{V}}^2 \|\mathbf{v}\|_{\mathbf{V}}^2 \geq \|\mathbf{u}\|_{L^2(\Omega)^3}^2 \|\mathbf{v}\|_{L^2(\Omega)^3}^2$$

$$\begin{aligned}
&\geq \left(\int_{\Omega} |\mathbf{u}\mathbf{v}| \, d\Omega \right)^2 \\
&\geq \alpha_0^2 a(\mathbf{u}, \mathbf{v})^2
\end{aligned} \tag{3.32}$$

The constant, α_0 , comes from condition (2.1) on the permeability tensor. As a result, the constant, C_a , in (3.30) is $\mu/(a\rho\alpha_0)$. Again, it is clear that $\|\mathbf{v}\|_{\mathbf{V}}^2 \geq \|\operatorname{div} \mathbf{v}\|_{L^2(\Omega)^3}^2$. Therefore,

$$\begin{aligned}
\|\mathbf{v}\|_{\mathbf{V}}^2 \|p\|_{\Lambda}^2 &\geq \|\operatorname{div} \mathbf{v}\|_{L^2(\Omega)^3}^2 \|p\|_{\Lambda}^2 \\
&\geq \left(\int_{\Omega} |(\operatorname{div} \mathbf{v})p| \, d\Omega \right)^2 \\
&\geq b(\mathbf{v}, p)^2
\end{aligned} \tag{3.33}$$

■

Next, we show that $a(\cdot, \cdot)$ is coercive on \mathbf{D} .

Lemma 3.3 The bilinear form, $a(\cdot, \cdot)$, is coercive on \mathbf{D} . Namely, there exists a positive constant, γ , such that for all $\mathbf{v} \in \mathbf{D}$ we have

$$\gamma \|\mathbf{v}\|_{\mathbf{V}}^2 \leq a(\mathbf{v}, \mathbf{v}). \tag{3.34}$$

Proof. Since $\operatorname{div} \mathbf{v} \in L^2(\Omega)$, we have $\operatorname{div} \mathbf{v} = \lambda \in \Lambda$. Therefore, $\|\operatorname{div} \mathbf{v}\|_{L^2(\Omega)}^2 = b(\mathbf{v}, \lambda) = 0$, and

$$\|\mathbf{v}\|_{\mathbf{V}}^2 = \|\mathbf{v}\|_{L^2(\Omega)^3}^2 \leq \alpha_1 a(\mathbf{v}, \mathbf{v}) \quad \forall \mathbf{v} \in \mathbf{D}. \tag{3.35}$$

The constant, α_1 , again comes from the condition, (2.1), on the permeability tensor and the constant, γ , in (3.34) is equal to $\mu(a\rho\alpha_1)$. ■

This proves the existence and uniqueness of a solution for problem (3.28). Now, we want to show existence and uniqueness of problem (3.29). First, we need the following:

Lemma 3.4 There exists a positive constant, β , such that for every $p \in \Lambda$ we have

$$\beta \|p\|_\Lambda \leq \sup_{\mathbf{w} \in \mathbf{V}} \frac{b(\mathbf{w}, p)}{\|\mathbf{w}\|_{\mathbf{V}}}. \quad (3.36)$$

Proof. Let $p \in \Lambda$. There exists a unique $\Phi \in H^2(\Omega)$ which satisfies

$$\begin{aligned} -\Delta \Phi &= p \\ \Phi &= 0 \text{ on } \partial\Omega \end{aligned} \quad (3.37)$$

By elliptic regularity we have

$$\|\Phi\|_{H^2(\Omega)} \leq C \|p\|_{L^2(\Omega)} \quad (3.38)$$

Since $\Phi \in H^2(\Omega)$ we have $\nabla \Phi \in \mathbf{V}$. Therefore, we let $\mathbf{w}_p = -\nabla \Phi$ and thus, $\operatorname{div} \mathbf{w}_p = p$. Furthermore,

$$\|\mathbf{w}_p\|_{\mathbf{V}} \leq C \|p\|_\Lambda \quad (3.39)$$

Now, we can finish the proof.

$$\sup_{\mathbf{w} \in \mathbf{V}} \frac{b(\mathbf{w}, p)}{\|\mathbf{w}\|_{\mathbf{V}}} \geq \frac{b(\mathbf{w}_p, p)}{\|\mathbf{w}_p\|_{\mathbf{V}}} = \frac{\|p\|_\Lambda^2}{\|\mathbf{w}_p\|_{\mathbf{V}}} \geq \frac{1}{C} \|p\|_\Lambda \quad (3.40)$$

Therefore, we let $\beta = 1/C$ and we are done. ■

The proof of Lemma 3.4 can be modified in the case when $p \in \Lambda_0$, i.e, homogeneous Neumann boundary conditions. In this case we take $\Phi \in H_0^2(\Omega)$ and $\mathbf{w}_p \in \mathbf{V}_0$. (See [12, Lemma 9.2.3].)

Let $\tilde{F}(\mathbf{w}) = -a(\mathbf{v}, \mathbf{w}) - G(\mathbf{w})$. Then, we rewrite equation (3.29) as

$$b(\mathbf{w}, p) = \tilde{F}(\mathbf{w}) \quad \forall \mathbf{w} \in \mathbf{V}_0 \quad (3.41)$$

Now, we prove existence and uniqueness of (3.41).

Theorem 3.5 Problem (3.41) has a unique solution.

Proof. Let \mathbf{D}^\perp denote the orthogonal complement of \mathbf{D} in \mathbf{V} . We need only consider $\mathbf{w} \in \mathbf{D}^\perp$ since the behavior of $b(\mathbf{w}, p)$ on \mathbf{D} is trivial. The Riesz Representation Theorem implies that there exists a linear operator, $T : \Lambda \rightarrow \mathbf{D}^\perp$, such that $b(\mathbf{w}, p) = (\mathbf{w}, Tp)$ and $\|b\|_{\mathbf{V}'} = \|Tp\|_{\mathbf{V}}$. Therefore,

$$\|Tp\|_{\mathbf{V}} = \sup_{\mathbf{w} \in \mathbf{D}^\perp} \frac{b(\mathbf{w}, p)}{\|\mathbf{w}\|_{\mathbf{V}}} \leq C_b \|p\|_{\Lambda} \quad (3.42)$$

The inequality in (3.42) is due to the continuity of $b(\cdot, \cdot)$.

We need to show that T is bijective. We show first that the image, R , of T is closed in \mathbf{D}^\perp . Suppose that $p_j \in \Lambda$ is a sequence with the property that $Tp_j \rightarrow \mathbf{w} \in \mathbf{D}^\perp$. Then $\{Tp_j\}$ is Cauchy in \mathbf{D}^\perp , and

$$\beta \|p_j - p_k\|_{\Lambda} \leq \sup_{\mathbf{w} \in \mathbf{D}^\perp} \frac{b(\mathbf{w}, p_j - p_k)}{\|\mathbf{w}\|_{\mathbf{V}}} \quad (3.43)$$

$$= \|Tp_j - Tp_k\|_{\mathbf{V}} \quad (3.44)$$

The inequality, (3.43), is from (3.36) and the equality (3.44) is because of (3.42). (By a similar argument, T is one-to-one.) Therefore, $\{p_j\}$ is Cauchy in Λ . Let $q = \lim_{j \rightarrow \infty} p_j$. By the continuity of T , $Tq = \mathbf{w}$. This shows that R is closed. Therefore, the space \mathbf{D}^\perp can be decomposed into R and its orthogonal complement R^\perp . If $\mathbf{w} \in R^\perp$, then $b(\mathbf{w}, q) = (\mathbf{w}, Tq) = 0$ for all $q \in \Lambda$ which implies $\mathbf{w} \in \mathbf{D}$.

However, $\mathbf{w} \in R^\perp$ implies $\mathbf{w} \in \mathbf{D}^\perp$ which forces $\mathbf{w} = \mathbf{0}$. Thus $\mathbf{D}^\perp = R$, and T is onto.

Using the Riesz Representation Theorem one more time gives us a unique $\mathbf{v} \in \mathbf{D}^\perp$ such that

$$\tilde{F}(\mathbf{w}) = (\mathbf{v}, \mathbf{w})_V \quad \forall \mathbf{w} \in \mathbf{D}^\perp. \quad (3.45)$$

We just let $Tp = \mathbf{v}$, which uniquely determines p by the bijectivity of T . ■

4. Discretization

In this chapter we describe the discretization of problem (3.28). We begin by defining a partition of Ω into hexahedral elements. Hexahedral elements are trilinear mappings of the reference cube.

We then define finite-dimensional velocity spaces, $\mathbf{V}_0^h \subset \mathbf{V}_0$, and $\mathbf{V}_N^h \subset \mathbf{V}_N$, and $\mathbf{V}^h \subset \mathbf{V}$, along with a finite-dimensional pressure space, $\Lambda^h \subset \Lambda$. The discrete pressure space consists of piecewise constant functions. The discrete velocity space consists of lowest-order Raviart-Thomas velocity functions.

The discrete velocity space must be treated in a special way. We define a transformation of a velocity function on the reference cube to a velocity function on the hexahedral element via the Piola transformation. Raviart and Thomas demonstrated in [52] the existence of a projection operator $\Pi_h : \mathbf{V} \rightarrow \mathbf{V}^h$ such that, for any $\mathbf{v} \in \mathbf{V}$,

$$b(\Pi_h \mathbf{v}, \lambda) = b(\mathbf{v}, \lambda), \quad \forall \lambda \in \Lambda^h, \quad (4.1)$$

and

$$\|\Pi_h \mathbf{v} - \mathbf{v}\|_{L^2(\Omega)^3} \leq Ch^s \|\mathbf{v}\|_{H^s(\Omega)^3}, \quad s = 0, 1. \quad (4.2)$$

The inf-sup condition can then be verified and error estimates given. This analysis was done for affine elements where the Jacobi matrix and the Jacobian are constant. Results for 2-D quadrilateral elements were later given in Thomas's thesis [57]. Results for non-affine 3-D elements is still an open question. If an interpolation estimate were known, it could be used along the lines of Thomas's thesis to verify the inf-sup condition and obtain error estimates. In particular,

for the special case of 3-D elements that are products of 2-D quadrilaterals with a 1-D interval (quadrilateral prisms), Thomas's theory applies.

There are numerous Piola-type transformations which satisfy the properties of the Piola transformation. The different Piola-type transformations will have certain interpolation properties which need to be considered. For example, the flux of a velocity vector representing uniform flow can be interpolated to the interior of a two-dimensional quadrilateral exactly. However, in the three-dimensional case an exact interpolation of the uniform flow, in general, can not be obtained.

The discrete velocity space can be decomposed using a discrete divergence-free velocity space, $\mathbf{D}^h \subset \mathbf{V}_0^h$. The construction of the discrete divergence-free velocity space on a rectangular grid in three dimensions is described in [17]. We will see here that this construction is also valid for the hexahedral grid. We will extend the work in [17] to hexahedral grids with non-homogeneous boundary conditions.

We now consider $\mathbf{v}_I^h \in \mathbf{V}_0^h$ and $\mathbf{v}_g^h \in \mathbf{V}_N^h$ and seek a solution to the divergence equation as before:

$$\operatorname{div} \mathbf{v}_I^h = f - \operatorname{div} \mathbf{v}_g^h. \quad (4.3)$$

Then, we seek the divergence-free correction $\mathbf{v}_D^h \in \mathbf{D}^h$ given by

$$a(\mathbf{v}_D^h, \mathbf{w}) = -a(\mathbf{v}_I^h + \mathbf{v}_g^h, \mathbf{w}) - G(\mathbf{w}), \quad \forall \mathbf{w} \in \mathbf{D}^h. \quad (4.4)$$

We write our approximate solution as:

$$\mathbf{v}^h = \mathbf{v}_D^h + (\mathbf{v}_I^h + \mathbf{v}_g^h). \quad (4.5)$$

The discretization results in a symmetric positive definite matrix problem. This matrix is defined and a right-hand side is given. The PCG method can be used to solve for \mathbf{v}_D^h .

4.1 Trilinear Hexahedral Elements

Figure 4.1 shows the image of the unit cube under the trilinear mapping and Figure 4.2 shows a random hexahedral grid in three dimensions. We partition Ω into hexahedral elements as follows:

$$\Omega = \bigcup_{i,j,k} Q_{i,j,k} \quad (4.6)$$

Each element, $Q_{i,j,k}$, is a trilinear mapping of the reference cube, \hat{Q} . We will call such a partition Ω^h . The superscript h refers to mesh size. We will usually think of h as being the maximum diameter of the elements of the decomposition.

A physical coordinate, (x, y, z) , in the hexahedral element, $Q_{i,j,k}$, is given in terms of a reference coordinate, $(\hat{x}, \hat{y}, \hat{z})$, through the trilinear mapping given by

$$\begin{aligned} x(\hat{x}, \hat{y}, \hat{z}) &= \mathbf{a}_x(\mathbf{p}_{i,j,k}^x) \cdot \mathbf{b}(\hat{x}, \hat{y}, \hat{z}) \\ y(\hat{x}, \hat{y}, \hat{z}) &= \mathbf{a}_y(\mathbf{p}_{j,k,i}^y) \cdot \mathbf{b}(\hat{x}, \hat{y}, \hat{z}) \\ z(\hat{x}, \hat{y}, \hat{z}) &= \mathbf{a}_z(\mathbf{p}_{k,i,j}^z) \cdot \mathbf{b}(\hat{x}, \hat{y}, \hat{z}) \end{aligned} \quad (4.7)$$

where

$$\mathbf{b}^T(\hat{x}, \hat{y}, \hat{z}) = (1, \hat{x}, \hat{y}, \hat{z}, \hat{x}\hat{y}, \hat{y}\hat{z}, \hat{x}\hat{z}, \hat{x}\hat{y}\hat{z}) \quad (4.8)$$

and

$$\begin{aligned} \mathbf{a}_x^T &= (a_0^x, a_1^x, a_2^x, a_3^x, a_4^x, a_5^x, a_6^x, a_7^x) \\ \mathbf{a}_y^T &= (a_0^y, a_1^y, a_2^y, a_3^y, a_4^y, a_5^y, a_6^y, a_7^y) \\ \mathbf{a}_z^T &= (a_0^z, a_1^z, a_2^z, a_3^z, a_4^z, a_5^z, a_6^z, a_7^z). \end{aligned} \quad (4.9)$$

The vector, $\mathbf{p}_{i,j,k}^x$, represents the eight x coordinates of the corners of hexahedral element $Q_{i,j,k}$. Likewise, $\mathbf{p}_{j,k,i}^y$ and $\mathbf{p}_{k,i,j}^z$ are the y and z coordinates, respectively. The vectors \mathbf{a}_x , \mathbf{a}_y and \mathbf{a}_z are uniquely determined by these points. The correspondence between the vertices of the reference cube and the vertices of $Q_{i,j,k}$ is summarized in the Table 4.1.

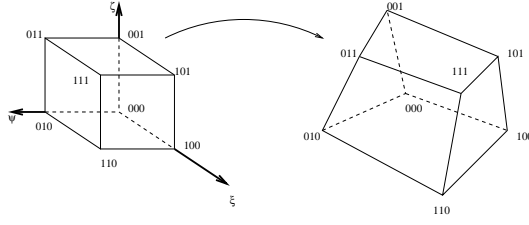


Figure 4.1: Trilinear Mapping of Reference Cube

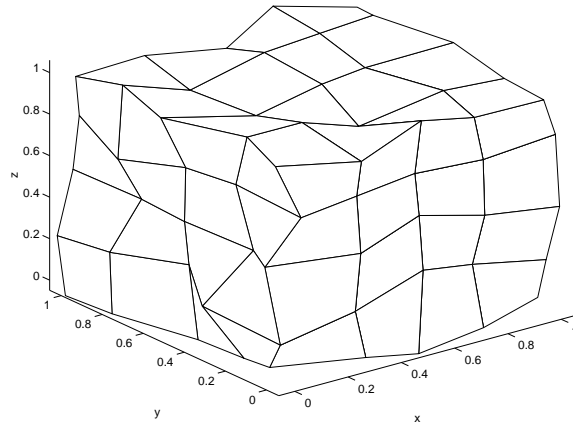


Figure 4.2: Random Hexahedral Grid

We can immediately write down eight equations

$$\begin{aligned}
 \mathbf{a}_x(\mathbf{p}_{i,j,k}^x) \cdot \mathbf{b}(0, 0, 0) &= x_{i-1/2,j-1/2,k-1/2} \\
 \mathbf{a}_x(\mathbf{p}_{i,j,k}^x) \cdot \mathbf{b}(1, 0, 0) &= x_{i+1/2,j-1/2,k-1/2} \\
 \mathbf{a}_x(\mathbf{p}_{i,j,k}^x) \cdot \mathbf{b}(0, 1, 0) &= x_{i-1/2,j+1/2,k-1/2} \\
 \mathbf{a}_x(\mathbf{p}_{i,j,k}^x) \cdot \mathbf{b}(1, 1, 0) &= x_{i+1/2,j+1/2,k-1/2} \\
 \mathbf{a}_x(\mathbf{p}_{i,j,k}^x) \cdot \mathbf{b}(0, 0, 1) &= x_{i-1/2,j-1/2,k+1/2} \\
 \mathbf{a}_x(\mathbf{p}_{i,j,k}^x) \cdot \mathbf{b}(1, 0, 1) &= x_{i+1/2,j-1/2,k+1/2} \\
 \mathbf{a}_x(\mathbf{p}_{i,j,k}^x) \cdot \mathbf{b}(0, 1, 1) &= x_{i-1/2,j+1/2,k+1/2} \\
 \mathbf{a}_x(\mathbf{p}_{i,j,k}^x) \cdot \mathbf{b}(1, 1, 1) &= x_{i+1/2,j+1/2,k+1/2}
 \end{aligned} \tag{4.10}$$

$(\hat{x}, \hat{y}, \hat{z})$	$p_{i,j,k}^x$	$p_{j,k,i}^y$	$p_{k,i,j}^z$
(0,0,0)	$x_{i-1/2,j-1/2,k-1/2}$	$y_{j-1/2,k-1/2,i-1/2}$	$z_{k-1/2,i-1/2,j-1/2}$
(1,0,0)	$x_{i+1/2,j-1/2,k-1/2}$	$y_{j-1/2,k-1/2,i+1/2}$	$z_{k-1/2,i+1/2,j-1/2}$
(0,1,0)	$x_{i-1/2,j+1/2,k-1/2}$	$y_{j+1/2,k-1/2,i-1/2}$	$z_{k-1/2,i-1/2,j+1/2}$
(1,1,0)	$x_{i+1/2,j+1/2,k-1/2}$	$y_{j+1/2,k-1/2,i+1/2}$	$z_{k-1/2,i+1/2,j+1/2}$
(0,0,1)	$x_{i-1/2,j-1/2,k+1/2}$	$y_{j-1/2,k+1/2,i-1/2}$	$z_{k+1/2,i-1/2,j-1/2}$
(1,0,1)	$x_{i+1/2,j-1/2,k+1/2}$	$y_{j-1/2,k+1/2,i+1/2}$	$z_{k+1/2,i+1/2,j-1/2}$
(0,1,1)	$x_{i-1/2,j+1/2,k+1/2}$	$y_{j+1/2,k+1/2,i-1/2}$	$z_{k+1/2,i-1/2,j+1/2}$
(1,1,1)	$x_{i+1/2,j+1/2,k+1/2}$	$y_{j+1/2,k+1/2,i+1/2}$	$z_{k+1/2,i+1/2,j+1/2}$

Table 4.1: Reference Coordinates and Physical Coordinates

and solve the system of equations for $\mathbf{p}_{i,j,k}^x$ giving

$$\mathbf{a}_x(\mathbf{p}_{i,j,k}^x) = \begin{bmatrix} a_0^x \\ a_1^x \\ a_2^x \\ a_3^x \\ a_4^x \\ a_5^x \\ a_6^x \\ a_7^x \end{bmatrix} = \begin{bmatrix} x_{i-1/2,j-1/2,k-1/2} \\ x_{i+1/2,j-1/2,k-1/2} - x_{i-1/2,j-1/2,k-1/2} \\ x_{i-1/2,j+1/2,k-1/2} - x_{i-1/2,j-1/2,k-1/2} \\ x_{i-1/2,j-1/2,k+1/2} - x_{i-1/2,j-1/2,k-1/2} \\ x_{i+1/2,j+1/2,k-1/2} - x_{i-1/2,j+1/2,k-1/2} - a_1^x \\ x_{i-1/2,j+1/2,k+1/2} - x_{i-1/2,j-1/2,k+1/2} - a_2^x \\ x_{i+1/2,j-1/2,k+1/2} - x_{i+1/2,j-1/2,k-1/2} - a_3^x \\ x_{i+1/2,j+1/2,k+1/2} - \sum_{s=0}^6 a_s^x \end{bmatrix}. \quad (4.11)$$

In the same manner we get $\mathbf{a}_y(\mathbf{p}_{j,k,i}^y)$ and $\mathbf{a}_z(\mathbf{p}_{k,i,j}^z)$ as follows:

$$\mathbf{a}_y(\mathbf{p}_{j,k,i}^y) = \begin{bmatrix} a_0^y \\ a_1^y \\ a_2^y \\ a_3^y \\ a_4^y \\ a_5^y \\ a_6^y \\ a_7^y \end{bmatrix} = \begin{bmatrix} y_{j-1/2,k-1/2,i-1/2} \\ y_{j-1/2,k-1/2,i+1/2} - y_{j-1/2,k-1/2,i-1/2} \\ y_{j+1/2,k-1/2,i-1/2} - y_{j-1/2,k-1/2,i-1/2} \\ y_{j-1/2,k+1/2,i-1/2} - y_{j-1/2,k-1/2,i-1/2} \\ y_{j+1/2,k-1/2,i+1/2} - y_{j+1/2,k-1/2,i-1/2} - a_1^y \\ y_{j+1/2,k+1/2,i-1/2} - y_{j-1/2,k+1/2,i-1/2} - a_2^y \\ y_{j-1/2,k+1/2,i+1/2} - y_{j-1/2,k-1/2,i+1/2} - a_3^y \\ y_{j+1/2,k+1/2,i+1/2} - \sum_{s=0}^6 a_s^y \end{bmatrix} \quad (4.12)$$

$$\mathbf{a}_z(\mathbf{p}_{k,i,j}^z) = \begin{bmatrix} a_0^z \\ a_1^z \\ a_2^z \\ a_3^z \\ a_4^z \\ a_5^z \\ a_6^z \\ a_7^z \end{bmatrix} = \begin{bmatrix} z_{k-1/2,i-1/2,j-1/2} \\ z_{k-1/2,i+1/2,j-1/2} - z_{k-1/2,i-1/2,j-1/2} \\ z_{k-1/2,i-1/2,j+1/2} - z_{k-1/2,i-1/2,j-1/2} \\ z_{k+1/2,i-1/2,j-1/2} - z_{k-1/2,i-1/2,j-1/2} \\ z_{k-1/2,i+1/2,j+1/2} - z_{k-1/2,i-1/2,j+1/2} - a_1^z \\ z_{k+1/2,i-1/2,j+1/2} - z_{k+1/2,i-1/2,j-1/2} - a_2^z \\ z_{k+1/2,i+1/2,j-1/2} - z_{k-1/2,i+1/2,j-1/2} - a_3^z \\ z_{k+1/2,i+1/2,j+1/2} - \sum_{s=0}^6 a_s^z \end{bmatrix}. \quad (4.13)$$

We can define a 3×8 matrix, $\mathbf{a}_{i,j,k}$, given by

$$\mathbf{a}_{i,j,k} = \begin{bmatrix} (\mathbf{a}_x(\mathbf{p}_{i,j,k}^x))^T \\ (\mathbf{a}_y(\mathbf{p}_{j,k,i}^y))^T \\ (\mathbf{a}_z(\mathbf{p}_{k,i,j}^z))^T \end{bmatrix}. \quad (4.14)$$

Once $\mathbf{a}_{i,j,k}$ has been computed for a particular element $Q_{i,j,k}$, we can then compute the physical coordinates of a point in $Q_{i,j,k}$ in terms of reference coordinates by

using

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = T_{i,j,k}(\hat{x}, \hat{y}, \hat{z}) = \mathbf{a}_{i,j,k} \mathbf{b}(\hat{x}, \hat{y}, \hat{z}). \quad (4.15)$$

We will call $T_{i,j,k}$ our trilinear mapping from \hat{Q} to $Q_{i,j,k}$. We will also assume we have an inverse mapping, $T_{i,j,k}^{-1}$ from $Q_{i,j,k}$ to \hat{Q} .

The covariant, or coordinate, basis vectors for element $Q_{i,j,k}$ are given by

$$\mathbf{X}_{i,j,k}(\hat{y}, \hat{z}) = \begin{bmatrix} X_{i,j,k}^0(\hat{y}, \hat{z}) \\ X_{i,j,k}^1(\hat{y}, \hat{z}) \\ X_{i,j,k}^2(\hat{y}, \hat{z}) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial \hat{x}}(\mathbf{a}_x(\mathbf{p}_{i,j,k}) \cdot \mathbf{b}(\hat{x}, \hat{y}, \hat{z})) \\ \frac{\partial}{\partial \hat{x}}(\mathbf{a}_y(\mathbf{p}_{j,k,i}) \cdot \mathbf{b}(\hat{x}, \hat{y}, \hat{z})) \\ \frac{\partial}{\partial \hat{x}}(\mathbf{a}_z(\mathbf{p}_{k,i,j}) \cdot \mathbf{b}(\hat{x}, \hat{y}, \hat{z})) \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \hat{x}} \\ \frac{\partial y}{\partial \hat{x}} \\ \frac{\partial z}{\partial \hat{x}} \end{bmatrix} \quad (4.16)$$

$$\mathbf{Y}_{j,k,i}(\hat{z}, \hat{x}) = \begin{bmatrix} Y_{j,k,i}^0(\hat{z}, \hat{x}) \\ Y_{j,k,i}^1(\hat{z}, \hat{x}) \\ Y_{j,k,i}^2(\hat{z}, \hat{x}) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial \hat{y}}(\mathbf{a}_x(\mathbf{p}_{i,j,k}) \cdot \mathbf{b}(\hat{x}, \hat{y}, \hat{z})) \\ \frac{\partial}{\partial \hat{y}}(\mathbf{a}_y(\mathbf{p}_{j,k,i}) \cdot \mathbf{b}(\hat{x}, \hat{y}, \hat{z})) \\ \frac{\partial}{\partial \hat{y}}(\mathbf{a}_z(\mathbf{p}_{k,i,j}) \cdot \mathbf{b}(\hat{x}, \hat{y}, \hat{z})) \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \hat{y}} \\ \frac{\partial y}{\partial \hat{y}} \\ \frac{\partial z}{\partial \hat{y}} \end{bmatrix} \quad (4.17)$$

$$\mathbf{Z}_{k,i,j}(\hat{x}, \hat{y}) = \begin{bmatrix} Z_{k,i,j}^0(\hat{x}, \hat{y}) \\ Z_{k,i,j}^1(\hat{x}, \hat{y}) \\ Z_{k,i,j}^2(\hat{x}, \hat{y}) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial \hat{z}}(\mathbf{a}_x(\mathbf{p}_{i,j,k}) \cdot \mathbf{b}(\hat{x}, \hat{y}, \hat{z})) \\ \frac{\partial}{\partial \hat{z}}(\mathbf{a}_y(\mathbf{p}_{j,k,i}) \cdot \mathbf{b}(\hat{x}, \hat{y}, \hat{z})) \\ \frac{\partial}{\partial \hat{z}}(\mathbf{a}_z(\mathbf{p}_{k,i,j}) \cdot \mathbf{b}(\hat{x}, \hat{y}, \hat{z})) \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \hat{z}} \\ \frac{\partial y}{\partial \hat{z}} \\ \frac{\partial z}{\partial \hat{z}} \end{bmatrix}. \quad (4.18)$$

We will allow ourselves to drop the dependence of the reference variables \hat{x} , \hat{y} , and \hat{z} when convenient. The unit normal vector (orthogonal to $\mathbf{Y}_{j,k,i}$ and $\mathbf{Z}_{k,i,j}$) is given by

$$\mathbf{n}_{i,j,k}^x(\hat{x}, \hat{y}, \hat{z}) = \pm \frac{\mathbf{Y}_{j,k,i} \times \mathbf{Z}_{k,i,j}}{\|\mathbf{Y}_{j,k,i} \times \mathbf{Z}_{k,i,j}\|}. \quad (4.19)$$

The \pm sign would be negative if we are talking about an outward normal at $\hat{x} = 0$ and would be positive if we are talking about an outward normal at $\hat{x} = 1$.

Likewise, we have a unit normal vector orthogonal to $\mathbf{X}_{i,j,k}$ and $\mathbf{Z}_{k,i,j}$ given by

$$\mathbf{n}_{j,k,i}^y(\hat{y}, \hat{z}, \hat{x}) = \pm \frac{\mathbf{Z}_{k,i,j} \times \mathbf{X}_{i,j,k}}{\|\mathbf{Z}_{k,i,j} \times \mathbf{X}_{i,j,k}\|}. \quad (4.20)$$

Finally, we have a unit normal vector orthogonal to $\mathbf{X}_{i,j,k}$ and $\mathbf{Y}_{j,k,i}$ given by

$$\mathbf{n}_{k,i,j}^z(\hat{z}, \hat{x}, \hat{y}) = \pm \frac{\mathbf{X}_{i,j,k} \times \mathbf{Y}_{j,k,i}}{\|\mathbf{X}_{i,j,k} \times \mathbf{Y}_{j,k,i}\|}. \quad (4.21)$$

We also define the area Jacobians as:

$$\Gamma_{i,j,k}^x(\hat{x}, \hat{y}, \hat{z}) = \|\mathbf{Y}_{j,k,i} \times \mathbf{Z}_{k,i,j}\| \quad (4.22)$$

$$\Gamma_{j,k,i}^y(\hat{y}, \hat{z}, \hat{x}) = \|\mathbf{Z}_{k,i,j} \times \mathbf{X}_{i,j,k}\| \quad (4.23)$$

$$\Gamma_{k,i,j}^z(\hat{z}, \hat{x}, \hat{y}) = \|\mathbf{X}_{i,j,k} \times \mathbf{Y}_{j,k,i}\|. \quad (4.24)$$

The Jacobi matrix is defined as:

$$B_{i,j,k} = \begin{bmatrix} X_{i,j,k}^0 & Y_{j,k,i}^0 & Z_{k,i,j}^0 \\ X_{i,j,k}^1 & Y_{j,k,i}^1 & Z_{k,i,j}^1 \\ X_{i,j,k}^2 & Y_{j,k,i}^2 & Z_{k,i,j}^2 \end{bmatrix} \quad (4.25)$$

and its determinant, the volume Jacobian $J_{i,j,k}(\hat{x}, \hat{y}, \hat{z})$, is given by

$$\begin{aligned} J_{i,j,k}(\hat{x}, \hat{y}, \hat{z}) &= \mathbf{X}_{i,j,k} \cdot (\mathbf{Y}_{j,k,i} \times \mathbf{Z}_{k,i,j}) \\ &= \mathbf{Y}_{j,k,i} \cdot (\mathbf{Z}_{k,i,j} \times \mathbf{X}_{i,j,k}) \\ &= \mathbf{Z}_{k,i,j} \cdot (\mathbf{X}_{i,j,k} \times \mathbf{Y}_{j,k,i}). \end{aligned} \quad (4.26)$$

We denote the faces with their respective unit normals in $Q_{i,j,k}$ as:

$$F_{i,j,k}^x(\hat{x}) \text{ , with normal } \mathbf{n}_{i,j,k}^x(\hat{x}, \hat{y}, \hat{z}) \quad (4.27)$$

$$F_{j,k,i}^y(\hat{y}) \text{ , with normal } \mathbf{n}_{j,k,i}^y(\hat{y}, \hat{z}, \hat{x}) \quad (4.28)$$

$$F_{k,i,j}^z(\hat{z}) \text{ , with normal } \mathbf{n}_{k,i,j}^z(\hat{z}, \hat{x}, \hat{y}). \quad (4.29)$$

The areas of the faces would then be given by

$$A_{i,j,k}^x(\hat{x}) = \int_{F_{i,j,k}^x(\hat{x})} dydz = \int_0^1 \int_0^1 \Gamma_{i,j,k}^x d\hat{y}d\hat{z} \quad (4.30)$$

$$A_{j,k,i}^y(\hat{y}) = \int_{F_{j,k,i}^y(\hat{y})} dx dz = \int_0^1 \int_0^1 \Gamma_{j,k,i}^y d\hat{x}d\hat{z} \quad (4.31)$$

$$A_{k,i,j}^z(\hat{z}) = \int_{F_{k,i,j}^z(\hat{z})} dx dy = \int_0^1 \int_0^1 \Gamma_{k,i,j}^z d\hat{x}d\hat{y}. \quad (4.32)$$

Note that throughout we have used $\{i, j, k\}$ ordering for the \mathbf{X} coordinates, $\{j, k, i\}$ ordering for the \mathbf{Y} coordinates and $\{k, i, j\}$ ordering for the \mathbf{Z} coordinates. This leads to some permutations that may be used to advantage during the implementation. For example, we define a function which computes $\mathbf{a}_{i,j,k}$ using the physical coordinates $\mathbf{p}_{i,j,k}^x$, $\mathbf{p}_{j,k,i}^y$, and $\mathbf{p}_{k,i,j}^z$ as follows:

$$\text{compute_a}(\mathbf{a}, \mathbf{p}^x, \mathbf{p}^y, \mathbf{p}^z, i, j, k). \quad (4.33)$$

Next, we define a function which takes the matrix $\mathbf{a}_{i,j,k}$ and computes $\mathbf{X}_{i,j,k}(\hat{y}, \hat{z})$ for a particular \hat{y} and \hat{z} as follows:

$$\text{compute_X}(\mathbf{X}, \mathbf{a}, \hat{y}, \hat{z}). \quad (4.34)$$

Now, suppose we call the same functions, but permute the arguments as follows:

$$\text{compute_a}(\mathbf{a}, \mathbf{p}^y, \mathbf{p}^z, \mathbf{p}^x, j, k, i) \quad (4.35)$$

$$\text{compute_X}(\mathbf{Y}, \mathbf{a}, \hat{z}, \hat{x}). \quad (4.36)$$

This will produce the vector, $\mathbf{Y}_{j,k,i}(\hat{z}, \hat{x})$, with a left cyclic shift on the components, that is, a $(Y_{i,j,k}^1, Y_{i,j,k}^2, Y_{i,j,k}^0)$ ordering.

4.2 Approximation Spaces

The pressure space, Λ^h , consists of piecewise constants. A pressure on the reference element is a constant scalar function. It, therefore, has one degree of freedom. We map this scalar function to a hexahedral element in an isoparametric way. This means that the mapping will only depend on the coordinate transformation given by the trilinear mapping. Since the scalar function is constant, there will be no dependence on the coordinates and, therefore, it will have the same constant value when mapped to a hexahedral element. The dimension of Λ^h on a $l \times m \times n$ grid is lmn . In the case of a pure homogeneous Neumann boundary condition, we will have the additional constraint specified for Λ_0^h . In this case the dimension of Λ_0^h is $lmn - 1$.

To define the finite-dimensional velocity space, $\mathbf{V}_0^h \subset \mathbf{V}_0$, we consider the lowest-order Raviart-Thomas vector function on a reference cube, \hat{Q} . We denote this space as $\mathcal{RT}_{[0]}(\hat{Q})$. We then use the Piola transformation to define the velocity space on $\mathcal{RT}_{[0]}(Q_{i,j,k})$. For $\hat{\mathbf{v}} \in \mathcal{RT}_{[0]}(\hat{Q})$, the Piola transformation is defined as:

$$\mathbf{v} = \frac{1}{J_{i,j,k}} B_{i,j,k} \hat{\mathbf{v}} \circ T_{i,j,k}^{-1} \quad (4.37)$$

and the inverse is given by

$$\hat{\mathbf{v}} = J_{i,j,k} B_{i,j,k}^{-1} \mathbf{v} \circ T_{i,j,k}. \quad (4.38)$$

Raviart-Thomas elements were first introduced in [52] for triangles and rectangles in two dimensions. This was later extended to three dimensions by Nedelec [51].

A velocity function on the reference cube has the following form:

$$\tilde{\mathbf{v}}(\hat{x}, \hat{y}, \hat{z}) = \begin{bmatrix} a + b\hat{x} \\ c + d\hat{y} \\ e + f\hat{z} \end{bmatrix}. \quad (4.39)$$

This vector function has six unknowns. Therefore, we can associate a value on each face of the reference cube for $\tilde{\mathbf{v}}$. These values will represent the fluxes across each face of the reference cube. We can define a basis vector function as a vector function with a flux of magnitude one on one of the faces and zero flux on the other faces. This will give us six basis vector functions as follows:

$$\hat{\mathbf{v}}_{-1/2,0,0} = ((1 - \hat{x}), 0, 0)^T \quad (4.40)$$

$$\hat{\mathbf{v}}_{1/2,0,0} = (\hat{x}, 0, 0)^T \quad (4.41)$$

$$\hat{\mathbf{v}}_{0,-1/2,0} = (0, 1 - \hat{y}, 0)^T \quad (4.42)$$

$$\hat{\mathbf{v}}_{0,1/2,0} = (0, \hat{y}, 0)^T \quad (4.43)$$

$$\hat{\mathbf{v}}_{0,0,-1/2} = (0, 0, 1 - \hat{z})^T \quad (4.44)$$

$$\hat{\mathbf{v}}_{0,0,1/2} = (0, 0, \hat{z})^T. \quad (4.45)$$

These basis vectors when mapped to $Q_{i,j,k}$ using the Piola transformation are given by

$$\mathbf{v}_{i,j,k;-1/2}^x = \frac{(1 - \hat{x})\mathbf{X}_{i,j,k}}{J_{i,j,k}} \quad (4.46)$$

$$\mathbf{v}_{i,j,k;1/2}^x = \frac{\hat{x}\mathbf{X}_{i,j,k}}{J_{i,j,k}} \quad (4.47)$$

$$\mathbf{v}_{j,k,i;-1/2}^y = \frac{(1 - \hat{y})\mathbf{Y}_{j,k,i}}{J_{j,k,i}} \quad (4.48)$$

$$\mathbf{v}_{j,k,i;1/2}^y = \frac{\hat{y}\mathbf{Y}_{j,k,i}}{J_{j,k,i}} \quad (4.49)$$

$$\mathbf{v}_{k,i,j;-1/2}^z = \frac{(1 - \hat{z})\mathbf{Z}_{k,i,j}}{J_{k,i,j}} \quad (4.50)$$

$$\mathbf{v}_{k,i,j;1/2}^z = \frac{\hat{z}\mathbf{Z}_{k,i,j}}{J_{k,i,j}}. \quad (4.51)$$

For now, we specify zero fluxes on the boundary of Ω . Due to the continuity of fluxes across faces, the dimension of the global velocity space on a $l \times m \times n$ grid will be $(l-1)mn + l(m-1)n + lm(n-1)$. That is, each basis vector function will have a support of two elements. The global velocity basis functions are thus defined as follows:

$$\mathbf{v}_{i+1/2,j,k}^x = \begin{cases} \mathbf{v}_{i,j,k;1/2}^x & , \text{ on } Q_{i,j,k} \\ \mathbf{v}_{i+1,j,k;-1/2}^x & , \text{ on } Q_{i+1,j,k} \end{cases} \quad (4.52)$$

$$\mathbf{v}_{j+1/2,k,i}^y = \begin{cases} \mathbf{v}_{j,k,i;1/2}^y & , \text{ on } Q_{i,j,k} \\ \mathbf{v}_{j+1,k,i;-1/2}^y & , \text{ on } Q_{i,j+1,k} \end{cases} \quad (4.53)$$

$$\mathbf{v}_{k+1/2,i,j}^z = \begin{cases} \mathbf{v}_{k,i,j;1/2}^z & , \text{ on } Q_{i,j,k} \\ \mathbf{v}_{k+1,i,j;-1/2}^z & , \text{ on } Q_{i,j,k+1}. \end{cases} \quad (4.54)$$

Our discrete velocity space, \mathbf{V}_0^h , is given by

$$\mathbf{V}_0^h = \text{span} \{ \mathbf{v}_{i+1/2,j,k}^x, \mathbf{v}_{j+1/2,k,i}^y, \mathbf{v}_{k+1/2,i,j}^z \}. \quad (4.55)$$

The volumes associated with two elements that have a face in common are given by

$$Q_{i+1/2,j,k} = Q_{i,j,k} \cup Q_{i+1,j,k} \quad (4.56)$$

$$Q_{i,j+1/2,k} = Q_{i,j,k} \cup Q_{i,j+1,k} \quad (4.57)$$

$$Q_{i,j,k+1/2} = Q_{i,j,k} \cup Q_{i,j,k+1} \quad (4.58)$$

Define the unit normals of the reference cube as follows:

$$\hat{\mathbf{n}}_{-1/2,0,0} = (-1, 0, 0)^T \quad (4.59)$$

$$\hat{\mathbf{n}}_{1/2,0,0} = (1, 0, 0)^T \quad (4.60)$$

$$\hat{\mathbf{n}}_{0,-1/2,0} = (0, -1, 0)^T \quad (4.61)$$

$$\hat{\mathbf{n}}_{0,1/2,0} = (0, 1, 0)^T \quad (4.62)$$

$$\hat{\mathbf{n}}_{0,0,-1/2} = (0, 0, -1)^T \quad (4.63)$$

$$\hat{\mathbf{n}}_{0,0,1/2} = (0, 0, 1)^T. \quad (4.64)$$

We can define an interpolation operator $\hat{\pi} : \mathbf{V}(\hat{Q}) \rightarrow \mathcal{RT}_{[0]}(\hat{Q})$. Let $\hat{\mathbf{v}} \in \mathbf{V}(\hat{Q})$.

Then, we can uniquely define $\hat{\pi}\hat{\mathbf{v}} \in \mathcal{RT}_{[0]}(\hat{Q})$ by

$$\begin{aligned} \hat{\pi}\hat{\mathbf{v}} &= f_{-1/2,0,0}\hat{\mathbf{v}}_{-1/2,0,0} + f_{1/2,0,0}\hat{\mathbf{v}}_{1/2,0,0} \\ &+ f_{0,-1/2,0}\hat{\mathbf{v}}_{0,-1/2,0} + f_{0,1/2,0}\hat{\mathbf{v}}_{0,1/2,0} \\ &+ f_{0,0,-1/2}\hat{\mathbf{v}}_{0,0,-1/2} + f_{0,0,1/2}\hat{\mathbf{v}}_{0,0,1/2} \end{aligned} \quad (4.65)$$

where

$$\begin{aligned} f_{-1/2,0,0} &= \int_0^1 \int_0^1 \hat{\mathbf{v}} \cdot \hat{\mathbf{n}}_{-1/2,0,0} d\hat{y}d\hat{z}, & f_{1/2,0,0} &= \int_0^1 \int_0^1 \hat{\mathbf{v}} \cdot \hat{\mathbf{n}}_{1/2,0,0} d\hat{y}d\hat{z} \\ f_{0,-1/2,0} &= \int_0^1 \int_0^1 \hat{\mathbf{v}} \cdot \hat{\mathbf{n}}_{0,-1/2,0} d\hat{x}d\hat{z}, & f_{0,1/2,0} &= \int_0^1 \int_0^1 \hat{\mathbf{v}} \cdot \hat{\mathbf{n}}_{0,1/2,0} d\hat{x}d\hat{z} \\ f_{0,0,-1/2} &= \int_0^1 \int_0^1 \hat{\mathbf{v}} \cdot \hat{\mathbf{n}}_{0,0,-1/2} d\hat{x}d\hat{y}, & f_{0,0,1/2} &= \int_0^1 \int_0^1 \hat{\mathbf{v}} \cdot \hat{\mathbf{n}}_{0,0,1/2} d\hat{x}d\hat{y}. \end{aligned} \quad (4.66)$$

Then, for any $\hat{\lambda} \in \mathcal{P}_0(\hat{Q})$ we have:

$$\int_{\partial\hat{Q}} (\hat{\pi}\hat{\mathbf{v}} - \hat{\mathbf{v}}) \cdot \hat{\mathbf{n}}\hat{\lambda}d\hat{s} = 0. \quad (4.67)$$

Furthermore, if we let

$$\hat{\rho}(\nabla \cdot \hat{\mathbf{v}}) = \nabla \cdot \hat{\pi}\hat{\mathbf{v}} \quad (4.68)$$

then, we have:

$$\int_{\hat{Q}} \hat{\rho}(\nabla \cdot \hat{\mathbf{v}}) \hat{\lambda} d\hat{Q} = \int_{\hat{Q}} \nabla \cdot \hat{\mathbf{v}} \hat{\lambda} d\hat{Q}. \quad (4.69)$$

In other words, $\nabla \cdot (\hat{\pi} \hat{\mathbf{v}})$ is the L^2 projection of $\nabla \cdot \hat{\mathbf{v}}$ onto the space $\mathcal{P}_0(\hat{Q})$. Note that we have used $\nabla \cdot$ to represent the divergence operator. We will use this notation constantly.

4.2.1 Properties of the Piola Transformation

The Piola transformation results in the following:

Lemma 4.1 For any $\hat{\mathbf{v}} \in \mathbf{V}(\hat{Q})$ we have

$$\int_{\hat{Q}} (\nabla \cdot \hat{\mathbf{v}}) \hat{\lambda} d\hat{x}d\hat{y}d\hat{z} = \int_{Q_{i,j,k}} (\nabla \cdot \mathbf{v}) \lambda dx dy dz, \quad \forall \hat{\lambda} \in L^2(\hat{Q}) \quad (4.70)$$

$$\int_{\partial\hat{Q}} (\hat{\mathbf{v}} \cdot \hat{\mathbf{n}}) \hat{\lambda} d\hat{s} = \int_{\partial Q_{i,j,k}} (\mathbf{v} \cdot \mathbf{n}) \lambda ds, \quad \forall \hat{\lambda} \in L^2(\partial\hat{Q}) \quad (4.71)$$

Proof. Equation (4.71) is essentially a definition of the Piola transformation. For example, let $\hat{\mathbf{v}} = (\hat{v}_1, \hat{v}_2, \hat{v}_3)^T$. Then,

$$\begin{aligned} \int_{F_{i,j,k}^x(0)} \mathbf{v} \cdot \mathbf{n}_{i,j,k}^x(0) dy dz &= \int_{F_{i,j,k}^x(0)} \hat{v}_1 \left(\frac{\mathbf{X}_{i,j,k}}{J_{i,j,k}} \right) \cdot \left(\frac{\mathbf{Y}_{i,j,k} \times \mathbf{Z}_{i,j,k}}{\|\mathbf{Y}_{i,j,k} \times \mathbf{Z}_{i,j,k}\|} \right) dy dz \\ &= \int_0^1 \int_0^1 \hat{v}_1|_{\hat{x}=0} d\hat{y}d\hat{z} \end{aligned} \quad (4.72)$$

$$= \int_{\hat{Q}} \hat{\mathbf{v}} \cdot \hat{\mathbf{n}}_{1/2,0,0} d\hat{s}. \quad (4.73)$$

We use (4.71) and Green's formula to show (4.70) as follows:

$$\int_{\hat{Q}} (\nabla \cdot \hat{\mathbf{v}}) \hat{\lambda} d\hat{x}d\hat{y}d\hat{z} = \int_{\partial\hat{Q}} (\hat{\mathbf{v}} \cdot \hat{\mathbf{n}}) \hat{\lambda} d\hat{s} - \int_{\hat{Q}} \hat{\mathbf{v}} \cdot \nabla \hat{\lambda} d\hat{x}d\hat{y}d\hat{z}$$

$$\begin{aligned}
&= \int_{\partial Q_{i,j,k}} (\mathbf{v} \cdot \mathbf{n}) \lambda \, ds - \int_{\hat{Q}} \hat{\mathbf{v}} \cdot \nabla \hat{\lambda} \, d\hat{x}d\hat{y}d\hat{z} \tag{4.74} \\
&= \int_{Q_{i,j,k}} (\nabla \cdot \mathbf{v}) \lambda \, ds + \int_{Q_{i,j,k}} \mathbf{v} \cdot \nabla \lambda \, dx dy dz - \int_{\hat{Q}} \hat{\mathbf{v}} \cdot \nabla \hat{\lambda} \, d\hat{x}d\hat{y}d\hat{z} \\
&= \int_{Q_{i,j,k}} (\nabla \cdot \mathbf{v}) \lambda \, ds.
\end{aligned}$$

To get the last equality we need to show that

$$\int_{Q_{i,j,k}} \mathbf{v} \cdot \nabla \lambda \, dx dy dz - \int_{\hat{Q}} \hat{\mathbf{v}} \cdot \nabla \hat{\lambda} \, d\hat{x}d\hat{y}d\hat{z} = 0. \tag{4.75}$$

In the case of lowest-order Raviart-Thomas spaces we can use the fact that the scalar functions, λ and $\hat{\lambda}$, are constant. However, we can also show it for higher order spaces. To show this we will look at a simple example with lowest-order \mathbf{v} and arbitrary λ ; an analogous, but more tedious, argument works for higher-order \mathbf{v} . Suppose $\hat{\mathbf{v}} = \hat{\mathbf{v}}_{1/2,0,0}$. Then,

$$\begin{aligned}
\int_{Q_{i,j,k}} \mathbf{v} \cdot \nabla \lambda \, dx dy dz &= \int_{Q_{i,j,k}} \frac{\hat{x}}{J_{i,j,k}} \mathbf{X}_{i,j,k} \cdot \nabla \lambda \, dx dy dz \\
&= \int_{Q_{i,j,k}} \frac{\hat{x}}{J_{i,j,k}} \frac{\partial \lambda}{\partial \hat{x}} \, dx dy dz \\
&= \int_{\hat{Q}} \hat{x} \frac{\partial \hat{\lambda}}{\partial \hat{x}} \, d\hat{x}d\hat{y}d\hat{z} \\
&= \int_{\hat{Q}} \hat{\mathbf{v}} \cdot \nabla \hat{\lambda} \, d\hat{x}d\hat{y}d\hat{z}.
\end{aligned}$$

Note also that (4.70) implies that

$$\nabla \cdot \mathbf{v} = \frac{1}{J_{i,j,k}} \nabla \cdot \hat{\mathbf{v}} \circ T_{i,j,k}^{-1}. \tag{4.76}$$

■

We now give the results described by Thomas in [57] for the approximation of smooth vector functions on 2-D quadrilaterals. As noted above, the analysis for non-affine 3-D elements is still an open question. However, Thomas's theory does apply for the special case of 3-D elements that are products of 2-D quadrilaterals with a 1-D interval (quadrilateral prisms).

We will use C to denote a generic positive constant independent of the mesh size h . We have for 2-D quadrilaterals the following:

Theorem 4.2 There exists an operator $\pi_{i,j,k}$ such that, for all $\mathbf{v} \in H^1(Q_{i,j,k})^2$ with $\nabla \cdot \mathbf{v} \in H^1(Q_{i,j,k})$ we have

$$\|\pi_{i,j,k}\mathbf{v} - \mathbf{v}\|_{L^2(Q_{i,j,k})^2} \leq Ch|\mathbf{v}|_{H^1(Q_{i,j,k})^2} + Ch^2|\nabla \cdot \mathbf{v}|_{H^1(Q_{i,j,k})} \quad (4.77)$$

and

$$\|\nabla \cdot (\pi_{i,j,k}\mathbf{v} - \mathbf{v})\|_{L^2(Q_{i,j,k})} \leq C\|\nabla \cdot \mathbf{v}\|_{L^2(Q_{i,j,k})} + Ch|\nabla \cdot \mathbf{v}|_{H^1(Q_{i,j,k})}. \quad (4.78)$$

Proof. The proof is given in [57, Theorem III.4.3 (pg. III-26)]. ■

A result of Theorem 4.2 is the error estimates given by

Theorem 4.3 For $p \in H^1(\Omega)$, $\mathbf{v} \in H^1(\Omega)^2$, and $\nabla \cdot \mathbf{v} \in H^1(\Omega)$ we have

$$\|p - p_h\|_{L^2(\Omega)} + \|\mathbf{v} - \mathbf{v}_h\|_{L^2(\Omega)^2} \leq Ch(|p|_{H^1(\Omega)} + |\mathbf{v}|_{H^1(\Omega)^2}) + Ch^2|\nabla \cdot \mathbf{v}|_{H^1(\Omega)} \quad (4.79)$$

and

$$\|\nabla \cdot (\mathbf{v} - \mathbf{v}_h)\|_{L^2(\Omega)} \leq C|\mathbf{v}|_{H^1(\Omega)^2} + Ch(|p|_{H^1(\Omega)} + |\nabla \cdot \mathbf{v}|_{H^1(\Omega)}). \quad (4.80)$$

Proof. The proof is given in [57, Theorem IX.3.2 (pg. IX-17)] and [57, Theorem IX.4.1 (pg. IX-19)]. ■

Note that $\nabla \cdot (\mathbf{v} - \mathbf{v}_h)$ does not necessarily go to zero as h goes to zero. For higher-order elements, $\nabla \cdot (\mathbf{v} - \mathbf{v}_h)$ does go to zero as long as $p \in H^2(\Omega)$ and $\mathbf{v}, \nabla \cdot \mathbf{v} \in H^2(\Omega)$; with $\mathcal{RT}_{[1]}$ instead of $\mathcal{RT}_{[0]}$, all the above estimates hold with one more power of h .

4.2.2 Interpolation of Velocity Functions

We now look at how the flux gets interpolated to the interior of a hexahedral element. We can define numerous Piola-type transformations in three dimensions which still have the properties of the Piola transformation discussed above. The difference between these various Piola-type mappings is how the flux is interpolated to the interior of an element. One simple modification of the Piola transformation is to include an area correction factor as follows:

$$\mathbf{v}_{i,j,k;-1/2}^x = \rho_{i,j,k;-1/2}^x \frac{(1 - \hat{x})\mathbf{X}_{i,j,k}}{J_{i,j,k}} \quad (4.81)$$

$$\mathbf{v}_{i,j,k;1/2}^x = \rho_{i,j,k;1/2}^x \frac{\hat{x}\mathbf{X}_{i,j,k}}{J_{i,j,k}} \quad (4.82)$$

$$\mathbf{v}_{j,k,i;-1/2}^y = \rho_{j,k,i;-1/2}^y \frac{(1 - \hat{y})\mathbf{Y}_{j,k,i}}{J_{j,k,i}} \quad (4.83)$$

$$\mathbf{v}_{j,k,i;1/2}^y = \rho_{j,k,i;1/2}^y \frac{\hat{y}\mathbf{Y}_{j,k,i}}{J_{j,k,i}} \quad (4.84)$$

$$\mathbf{v}_{k,i,j;1/2}^z = \rho_{k,i,j;-1/2}^z \frac{(1 - \hat{z})\mathbf{Z}_{k,i,j}}{J_{k,i,j}} \quad (4.85)$$

$$\mathbf{v}_{k,i,j;1/2}^z = \rho_{k,i,j;1/2}^z \frac{\hat{z}\mathbf{Z}_{k,i,j}}{J_{k,i,j}} \quad (4.86)$$

where the area correction factors are given by

$$\rho_{i,j,k;-1/2}^x = \frac{\|\mathbf{Y}_{j,k,i} \times \mathbf{Z}_{k,i,j}\|}{A_{i,j,k}^x(0)} \quad (4.87)$$

$$\rho_{i,j,k;1/2}^x = \frac{\|\mathbf{Y}_{j,k,i} \times \mathbf{Z}_{k,i,j}\|}{A_{i,j,k}^x(1)} \quad (4.88)$$

$$\rho_{j,k,i;-1/2}^y = \frac{\|\mathbf{Z}_{k,i,j} \times \mathbf{X}_{i,j,k}\|}{A_{j,k,i}^y(0)} \quad (4.89)$$

$$\rho_{j,k,i;1/2}^y = \frac{\|\mathbf{Z}_{k,i,j} \times \mathbf{X}_{i,j,k}\|}{A_{j,k,i}^y(1)} \quad (4.90)$$

$$\rho_{k,i,j;-1/2}^z = \frac{\|\mathbf{X}_{i,j,k} \times \mathbf{Y}_{j,k,i}\|}{A_{k,i,j}^z(0)} \quad (4.91)$$

$$\rho_{k,i,j;1/2}^z = \frac{\|\mathbf{X}_{i,j,k} \times \mathbf{Y}_{j,k,i}\|}{A_{k,i,j}^z(1)}. \quad (4.92)$$

The velocity, $\mathbf{u}_{i,j,k}$, interior to a cell, $Q_{i,j,k}$, is approximated by

$$\begin{aligned} \mathbf{u}_{i,j,k}(\hat{x}, \hat{y}, \hat{z}) &= f_{i-1/2,j,k}^x \mathbf{v}_{i,j,k;-1/2}^x(\hat{x}, \hat{y}, \hat{z}) + f_{i+1/2,j,k}^x \mathbf{v}_{i,j,k;1/2}^x(\hat{x}, \hat{y}, \hat{z}) \\ &+ f_{j-1/2,k,i}^y \mathbf{v}_{j,k,i;-1/2}^y(\hat{y}, \hat{z}, \hat{x}) + f_{j+1/2,k,i}^y \mathbf{v}_{j,k,i;1/2}^y(\hat{y}, \hat{z}, \hat{x}) \\ &+ f_{k-1/2,i,j}^z \mathbf{v}_{k,i,j;-1/2}^z(\hat{z}, \hat{x}, \hat{y}) + f_{k+1/2,i,j}^z \mathbf{v}_{k,i,j;1/2}^z(\hat{z}, \hat{x}, \hat{y}). \end{aligned} \quad (4.93)$$

The nodal values $f_{i-1/2,j,k}^x$, $f_{i+1/2,j,k}^x$, $f_{j-1/2,k,i}^y$, $f_{j+1/2,k,i}^y$, $f_{k-1/2,i,j}^z$ and $f_{k+1/2,i,j}^z$ are the fluxes on faces $F_{i,j,k}^x(0)$, $F_{i,j,k}^x(1)$, $F_{j,k,i}^y(0)$, $F_{j,k,i}^y(1)$, $F_{k,i,j}^z(0)$, and $F_{k,i,j}^z(1)$, respectively. For example,

$$\int_{F_{i,j,k}^x(0)} \mathbf{u}_{i,j,k} \cdot \mathbf{n}_{i,j,k}^x = f_{i-1/2,j,k}^x. \quad (4.94)$$

The scalar value, $\mathbf{u}_{i,j,k}(\hat{x}, \hat{y}, \hat{z}) \cdot \mathbf{n}_{i,j,k}^x(\hat{x}, \hat{y}, \hat{z})$ is given by

$$\mathbf{u}_{i,j,k}(\hat{x}, \hat{y}, \hat{z}) \cdot \mathbf{n}_{i,j,k}^x(\hat{x}, \hat{y}, \hat{z}) = \frac{f_{i-1/2,j,k}^x}{A_{i,j,k}^x(0)}(1 - \hat{x}) + \frac{f_{i+1/2,j,k}^x}{A_{i,j,k}^x(1)}(\hat{x}). \quad (4.95)$$

Equation (4.95) is a linear interpolation of the average normal velocities from the two faces at either end of the element. Suppose we have uniform flow given by $\mathbf{v} = (c, 0, 0)^T$. Furthermore, suppose $\mathbf{n}_{i,j,k}^x$ is in the direction of \mathbf{v} on element $Q_{i,j,k}$. Then, the flux across face $F_{i,j,k}^x(\hat{x})$ is $cA_{i,j,k}^x(\hat{x})$. Substituting this flux

at the two faces, $F_{i,j,k}^x(0)$ and $F_{i,j,k}^x(1)$, into equation (4.95) gives a value of c . Therefore, when we integrate over the face, $F_{i,j,k}^x(\hat{x})$, we get the correct flux. This would not have been the case with the standard Piola mapping, which would give $c(A_{i,j,k}^x(0)(1 - \hat{x}) + A_{i,j,k}^x(1)\hat{x})$; this is equal to $cA_{i,j,k}^x(\hat{x})$ if $A_{i,j,k}(\hat{x})$ varies linearly with \hat{x} as it must in 2-D, but in general this is not the same.

Now, we look at a different example. Again, we will assume a uniform flow, $\mathbf{v} = (c, 0, 0)^T$. This time, however, we will have a face where $\mathbf{n}_{i,j,k}^x(\hat{x}, \hat{y}, \hat{z})$ is not parallel to \mathbf{v} . In particular, consider the element shown in Figure 4.3.

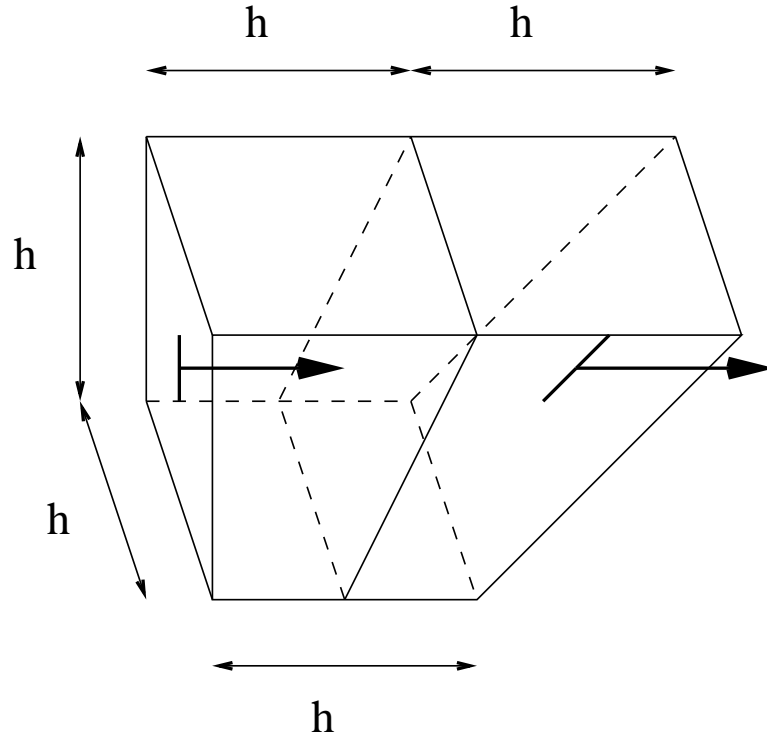


Figure 4.3: Uniform Flow on Distorted Element

We will consider the flux in only this one element, so we will drop the i, j, k subscripts. This element has a face at $\hat{x} = 0$ with area h^2 and a normal, \mathbf{n}^x , in the

direction of the flow. The face at $\hat{x} = 1$ has area $\sqrt{2}h^2$ with normal at an angle at 45 degrees with the flow. In general, the flux across the face, $F^x(\hat{x})$, will be ch^2 and the area will be $h^2\sqrt{\hat{x}^2 + 1}$. Substituting these expressions into equation (4.95) and integrating over the face, $F^x(\hat{x})$, gives the interpolated flux as:

$$\int_{F^x(\hat{x})} \mathbf{u} \cdot \mathbf{n}^x dA^x = ch^2 \left(1 + \left(\frac{\sqrt{2}}{2} - 1 \right) \hat{x} \right) \sqrt{\hat{x}^2 + 1}. \quad (4.96)$$

The relative error of this interpolation is plotted in Figure 4.4. In this case, because the flux varies linearly with \hat{x} (actually, it is constant), the standard Piola mapping gives the correct interpolated flux.

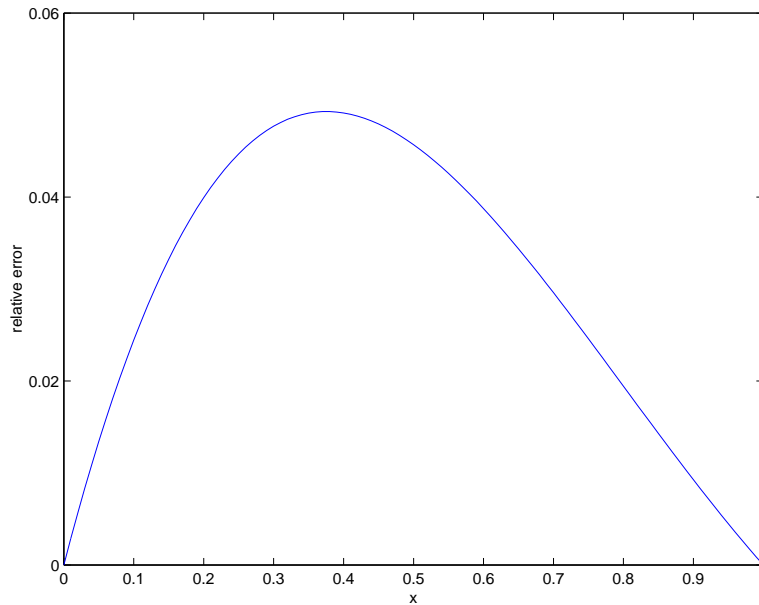


Figure 4.4: Relative Error of Interpolated Flux

For general hexahedral elements, especially when the faces are not planar, the flux will not always be interpolated exactly for uniform flow. This situation is

particular to 3-D. In two dimensions we have no problem interpolating uniform flow. This is due to the use of linear interpolation which works fine in 2-D, but not in 3-D. We perhaps need a quadratic term in the interpolation. Different velocity trial functions are being investigated by Richard Naff, [50], at the U.S. Geological Survey. It appears, at this point, that there will be no way to exactly interpolate uniform flow for general hexahedral elements. We will have to be satisfied with a close approximation on the order of h^2 when using lowest-order Raviart-Thomas velocity functions.

4.3 Divergence-Free Basis

We need a computationally convenient basis for the divergence-free velocity space. The construction of such a basis for a rectangular grid in three dimensions for homogeneous boundary conditions is described in [17]. More recently, a computationally convenient divergence-free basis is described by Scheichl in [54] for triangles in two dimensions and tetrahedra in three dimensions. Scheichl uses a spanning tree algorithm to compute the divergence-free basis. Our method is to define the divergence-free basis a priori.

We will see here that the divergence-free basis described in [17] is also valid for distorted grids using hexahedral elements. We will extend this basis to include Dirichlet boundary conditions. It became convenient to define a small part of the divergence-free basis for Dirichlet boundary conditions using the spanning tree algorithm described in [54].

A divergence-free vector function is the curl of a vector potential function. We wish to construct the divergence-free subspace such that it is contained in \mathbf{V}_0^h .

Consider the bilinear basis on the reference cube, \hat{Q} , consisting of scalar functions which are bilinear in \hat{y} and \hat{z} and constant in \hat{x} given by

$$\hat{\Phi}_{1/2,1/2}^x = \hat{y}\hat{z} \quad (4.97)$$

$$\hat{\Phi}_{-1/2,1/2}^x = (1 - \hat{y})\hat{z} \quad (4.98)$$

$$\hat{\Phi}_{-1/2,-1/2}^x = (1 - \hat{y})(1 - \hat{z}) \quad (4.99)$$

$$\hat{\Phi}_{1/2,-1/2}^x = \hat{y}(1 - \hat{z}). \quad (4.100)$$

We now define the vector potential functions on the reference cube as:

$$\hat{\Phi}_{1/2,1/2}^x = (\hat{\Phi}_{1/2,1/2}^x, 0, 0)^T \quad (4.101)$$

$$\hat{\Phi}_{-1/2,1/2}^x = (\hat{\Phi}_{-1/2,1/2}^x, 0, 0)^T \quad (4.102)$$

$$\hat{\Phi}_{-1/2,-1/2}^x = (\hat{\Phi}_{-1/2,-1/2}^x, 0, 0)^T \quad (4.103)$$

$$\hat{\Phi}_{1/2,-1/2}^x = (\hat{\Phi}_{1/2,-1/2}^x, 0, 0)^T. \quad (4.104)$$

We take the curl of each of these vector functions giving divergence-free vector functions on the reference cube as:

$$\hat{\mathbf{w}}_{1/2,1/2}^x = \mathbf{curl} \hat{\Phi}_{1/2,1/2}^x = \begin{bmatrix} 0 \\ \hat{y} \\ -\hat{z} \end{bmatrix} = \hat{\mathbf{v}}_{0,1/2,0} - \hat{\mathbf{v}}_{0,0,1/2} \quad (4.105)$$

$$\hat{\mathbf{w}}_{-1/2,1/2}^x = \mathbf{curl} \hat{\Phi}_{-1/2,1/2}^x = \begin{bmatrix} 0 \\ 1 - \hat{y} \\ \hat{z} \end{bmatrix} = \hat{\mathbf{v}}_{0,-1/2,0} + \hat{\mathbf{v}}_{0,0,1/2} \quad (4.106)$$

$$\hat{\mathbf{w}}_{-1/2,-1/2}^x = \mathbf{curl} \hat{\Phi}_{-1/2,-1/2}^x = \begin{bmatrix} 0 \\ -(1 - \hat{y}) \\ 1 - \hat{z} \end{bmatrix} = -\hat{\mathbf{v}}_{0,-1/2,0} + \hat{\mathbf{v}}_{0,0,-1/2} \quad (4.107)$$

$$\hat{\mathbf{w}}_{1/2,-1/2}^x = \mathbf{curl} \hat{\Phi}_{1/2,-1/2}^x = \begin{bmatrix} 0 \\ -\hat{y} \\ -(1-\hat{z}) \end{bmatrix} = -\hat{\mathbf{v}}_{0,1/2,0} - \hat{\mathbf{v}}_{0,0,-1/2}. \quad (4.108)$$

Now, we use a Piola-type mapping to map $\hat{\mathbf{w}}_{1/2,1/2}^x$, $\hat{\mathbf{w}}_{-1/2,1/2}^x$, $\hat{\mathbf{w}}_{-1/2,-1/2}^x$, and $\hat{\mathbf{w}}_{1/2,-1/2}^x$ to the physical elements $Q_{i,j,k}$, $Q_{i,j+1,k}$, $Q_{i,j+1,k+1}$, and $Q_{i,j,k+1}$ respectively giving

$$\mathbf{w}_{j,k,i}^x = \mathbf{v}_{j,k,i;1/2}^y - \mathbf{v}_{k,i,j;1/2}^z \text{ on } Q_{i,j,k} \quad (4.109)$$

$$\mathbf{w}_{j+1,k,i}^x = \mathbf{v}_{j+1,k,i;-1/2}^y + \mathbf{v}_{k,i,j+1;1/2}^z \text{ on } Q_{i,j+1,k} \quad (4.110)$$

$$\mathbf{w}_{j+1,k+1,i}^x = -\mathbf{v}_{j+1,k+1,i;-1/2}^y + \mathbf{v}_{k+1,i,j+1;-1/2}^z \text{ on } Q_{i,j+1,k+1} \quad (4.111)$$

$$\mathbf{w}_{j,k+1,i}^x = -\mathbf{v}_{j,k+1,i;1/2}^y + \mathbf{v}_{k+1,i,j;-1/2}^z \text{ on } Q_{i,j,k+1}. \quad (4.112)$$

Note that if we use the standard Piola mapping, then by (4.76) the \mathbf{w}^x 's will be divergence-free. If we use a different Piola-type mapping, for example (4.81)-(4.86), then the \mathbf{w}^x 's may not, in general, be strongly divergence-free. However, the Piola-type mapping still preserves the face fluxes. Therefore, the \mathbf{w}^x 's will be weakly divergence-free. From now on when we define a divergence-free vector basis function we will mean weakly divergence-free.

We use continuity across faces to define the divergence-free vector function with support on $Q_{j+1/2,k+1/2,i} = Q_{i,j,k} \cup Q_{j+1,k,i} \cup Q_{j+1,k+1,i} \cup Q_{i,j,k+1}$ as:

$$\begin{aligned} \mathbf{w}_{j+1/2,k+1/2,i}^x &= \mathbf{v}_{j+1/2,k,i}^y - \mathbf{v}_{j+1/2,k+1,i}^y \\ &\quad - \mathbf{v}_{k+1/2,i,j}^z + \mathbf{v}_{k+1/2,i,j+1}^z. \end{aligned} \quad (4.113)$$

We can think of (4.113) as an x -slice function and recognize it as such by the third index i . That is, an x -slice function has its support on four elements contained within a vertical slice of the domain for a given i . With a moments thought we

can see that the divergence-free basis vector function must have support on four elements in order to be weakly divergence-free. This is because the magnitude of the fluxes on the four faces must be equal in order for them to cancel out. The above construction also assures that a divergence-free vector function is contained in \mathbf{v}_0^h .

We define analogous divergence-free vector functions for y -slices and z -slices as follows:

$$\begin{aligned} \mathbf{w}_{k+1/2,i+1/2,j}^y &= \mathbf{v}_{k+1/2,i,j}^z - \mathbf{v}_{k+1/2,i+1,j}^z \\ &\quad - \mathbf{v}_{i+1/2,j,k}^x + \mathbf{v}_{i+1/2,j,k+1}^x \end{aligned} \quad (4.114)$$

$$\begin{aligned} \mathbf{w}_{i+1/2,j+1/2,k}^z &= \mathbf{v}_{i+1/2,j,k}^x - \mathbf{v}_{i+1/2,j+1,k}^x \\ &\quad - \mathbf{v}_{j+1/2,k,i}^y + \mathbf{v}_{j+1/2,k,i+1}^y. \end{aligned} \quad (4.115)$$

The divergence-free vector functions (4.113), (4.114), and (4.115) span the divergence-free subspace. We can write the x -slice function, $\mathbf{w}_{j+1/2,k+1/2,i}^x$, as $\mathbf{w}_{j+1/2,k+1/2,0}^x$ plus a linear combination of y -slice functions and z -slice functions as follows:

$$\begin{aligned} \mathbf{w}_{j+1/2,k+1/2,i}^x &= \mathbf{w}_{j+1/2,k+1/2,0}^x \\ &\quad + \sum_{i_s=0}^{i-1} \left(\mathbf{w}_{k+1/2,i_s+1/2,j}^y - \mathbf{w}_{k+1/2,i_s+1/2,j+1}^y \right) \\ &\quad + \sum_{i_s=0}^{i-1} \left(\mathbf{w}_{i_s+1/2,j+1/2,k}^z - \mathbf{w}_{i_s+1/2,j+1/2,k+1}^z \right). \end{aligned} \quad (4.116)$$

We can show similar linear dependencies for the y -slice and z -slice functions. These linear dependencies are given by

$$\begin{aligned} \mathbf{w}_{k+1/2,i+1/2,j}^y &= \mathbf{w}_{k+1/2,i+1/2,0}^y \\ &\quad + \sum_{j_s=0}^{j-1} \left(\mathbf{w}_{i+1/2,j_s+1/2,k}^z - \mathbf{w}_{i+1/2,j_s+1/2,k+1}^z \right) \\ &\quad + \sum_{j_s=0}^{j-1} \left(\mathbf{w}_{j_s+1/2,k+1/2,i}^x - \mathbf{w}_{j_s+1/2,k+1/2,i+1}^x \right) \end{aligned} \quad (4.117)$$

and

$$\begin{aligned}
\mathbf{w}_{i+1/2,j+1/2,k}^z &= \mathbf{w}_{i+1/2,j+1/2,0}^z \\
&+ \sum_{k_s=0}^{k-1} \left(\mathbf{w}_{j+1/2,k_s+1/2,i}^x - \mathbf{w}_{j+1/2,k_s+1/2,i+1}^x \right) \\
&+ \sum_{k_s=0}^{k-1} \left(\mathbf{w}_{k_s+1/2,i+1/2,j}^y - \mathbf{w}_{k_s+1/2,i+1/2,j+1}^y \right)
\end{aligned} \tag{4.118}$$

respectively.

Therefore, we can eliminate one of the above sets of linearly dependent functions and still span the divergence-free subspace. We choose to eliminate the linearly dependent x -slice functions to end up with the divergence-free basis given by

$$\mathbf{w}_{j+1/2,k+1/2,i}^x ; i = 0, 0 \leq j \leq m - 2, 0 \leq k \leq n - 2 \tag{4.119}$$

$$\mathbf{w}_{k+1/2,i+1/2,j}^y ; 0 \leq i \leq l - 2, 0 \leq j \leq m - 1, 0 \leq k \leq n - 2 \tag{4.120}$$

$$\mathbf{w}_{i+1/2,j+1/2,k}^z ; 0 \leq i \leq l - 2, 0 \leq j \leq m - 2, 0 \leq k \leq n - 1. \tag{4.121}$$

The pressure space, Λ_0^h , can be defined as the divergence of \mathbf{V}_0^h . This means that Λ_0^h is the range of the divergence operator and the divergence-free subspace is the kernel of this operator. Therefore, the dimension of the divergence free subspace is given by

$$\begin{aligned}
\dim \mathbf{D}^h &= \dim \mathbf{V}_0^h - \dim \Lambda_0^h \\
&= (l - 1)mn + l(m - 1)n + lm(n - 1) - (lmn - 1) \\
&= 2lmn - lm - ln - mn + 1.
\end{aligned} \tag{4.122}$$

The number of divergence-free vector functions given in (4.119)-(4.121) is equal to (4.122). This will form our divergence-free basis.

4.3.1 Dirichlet Boundary Conditions

We now consider the case when part of the boundary has Dirichlet boundary conditions. In this case we will allow our velocity space, \mathbf{V}_0^h , to have non-zero fluxes on the Dirichlet boundary. We will still require that vector functions in \mathbf{V}_0^h have zero fluxes on the Neumann boundary. We no longer have a constraint on the pressure space. In other words, we now define our pressure space, Λ^h , as being spanned by piecewise constants and allow global constant functions.

We assume that if a given side of the domain has Dirichlet boundary conditions, then all the faces on that side have a Dirichlet boundary condition. This assumption is not necessary for the method to work, but simplifies the development to be described subsequently. For example, if we impose Dirichlet boundary conditions at $x = 0$, then all the faces at $x = 0$ will have unknown fluxes. These divergence-free boundary vector functions are illustrated for a single z -slice in Figure 4.5.

For each unknown flux on the boundary, there will be a corresponding basis vector function in \mathbf{V}_0^h given by

$$\mathbf{v}_{j,k}^{x0} = \mathbf{v}_{0,j,k;-1/2}^x, \quad \text{on } Q_{0,j,k} \quad (4.123)$$

$$\mathbf{v}_{j,k}^{x1} = \mathbf{v}_{l-1,j,k;1/2}^x, \quad \text{on } Q_{l-1,j,k} \quad (4.124)$$

$$\mathbf{v}_{k,i}^{y0} = \mathbf{v}_{0,k,i;-1/2}^y, \quad \text{on } Q_{i,0,k} \quad (4.125)$$

$$\mathbf{v}_{k,i}^{y1} = \mathbf{v}_{m-1,k,i;1/2}^y, \quad \text{on } Q_{i,m-1,k} \quad (4.126)$$

$$\mathbf{v}_{i,j}^{z0} = \mathbf{v}_{0,i,j;-1/2}^z, \quad \text{on } Q_{i,j,0} \quad (4.127)$$

$$\mathbf{v}_{i,j}^{z1} = \mathbf{v}_{n-1,i,j;1/2}^z, \quad \text{on } Q_{i,j,n-1}. \quad (4.128)$$

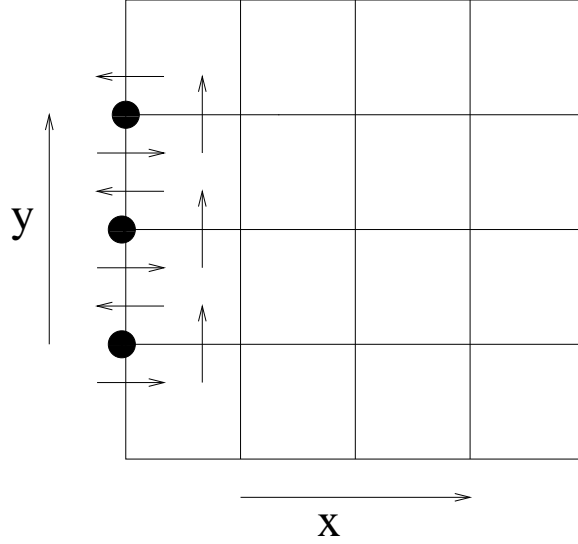


Figure 4.5: Boundary Divergence-Free Basis Vector

The divergence-free vector functions on the boundary are written in terms of vector functions from \mathbf{V}_0^h . For example, on the $x = 0$ boundary shown in Figure 4.5, we have the z -slice divergence-free vector functions given by

$$\mathbf{w}_{j+1/2,k}^{z,x_0} = \mathbf{v}_{j,k}^{x_0} - \mathbf{v}_{j+1,k}^{x_0} + \mathbf{v}_{j+1/2,k,0}^y. \quad (4.129)$$

Similarly, we have y -slice divergence-free vector functions given by

$$\mathbf{w}_{k+1/2,j}^{y,x_0} = \mathbf{v}_{j,k+1}^{x_0} + \mathbf{v}_{j,k}^{x_0} - \mathbf{v}_{k+1/2,0,j}^z. \quad (4.130)$$

Equations (4.129)-(4.130) define a divergence-free vector function for each edge on the boundary $x = 0$ and involve fluxes across faces on the boundary in addition to fluxes across internal faces. Note that divergence-free vector functions on edges not on the boundary don't involve any boundary fluxes.

The divergence-free vector functions defined in (4.129)-(4.130) span the divergence-free subspace on the boundary $x = 0$. However, we note that $\mathbf{w}_{k+1/2,j}^{y,x_0}$ can be

written in terms of $\mathbf{w}_{k+1/2,0}^{y,x0}$ plus a linear combination of other divergence-free vector functions. This relationship is given by

$$\begin{aligned} \mathbf{w}_{k+1/2,j}^{y,x0} &= \mathbf{w}_{k+1/2,0}^{y,x0} \\ &+ \sum_{s=0}^{j-1} \left(\mathbf{w}_{s+1/2,k}^{z,x0} - \mathbf{w}_{s+1/2,k+1}^{z,x0} - \mathbf{w}_{s+1/2,k+1/2,0}^x \right). \end{aligned} \quad (4.131)$$

Alternatively, we could have written $\mathbf{w}_{j+1/2,k+1}^{z,x0}$ in terms of $\mathbf{w}_{j+1/2,k}^{z,x0}$ plus a linear combination of other divergence-free vector functions using the relationship given by

$$\begin{aligned} \mathbf{w}_{j+1/2,k}^{z,x0} &= \mathbf{w}_{j+1/2,0}^{z,x0} \\ &+ \sum_{s=0}^{k-1} \left(\mathbf{w}_{s+1/2,j}^{y,x0} - \mathbf{w}_{s+1/2,j+1}^{y,x0} - \mathbf{w}_{j+1/2,s+1/2,0}^x \right). \end{aligned} \quad (4.132)$$

Therefore, the divergence-free subspace on the boundary $x = 0$ is spanned by

$$\begin{aligned} &\mathbf{w}_{j+1/2,k}^{z,x0}; 0 \leq j \leq m-2, 0 \leq k \leq n-1 \\ &\mathbf{w}_{k+1/2,j}^{y,x0}; j = 0, 0 \leq k \leq n-1 \end{aligned} \quad (4.133)$$

or

$$\begin{aligned} &\mathbf{w}_{j+1/2,k}^{z,x0}; 0 \leq j \leq m-2, k = 0 \\ &\mathbf{w}_{k+1/2,j}^{y,x0}; 0 \leq j \leq m-1, 0 \leq k \leq n-1. \end{aligned} \quad (4.134)$$

Which ever set of functions we choose, they both have dimension $mn - 1$. We no longer have the constraint on the pressure space. Therefore, the dimension of Λ^h is lmn . In general, the dimension of \mathbf{D}^h increases by the number of unknown fluxes on the boundary (because of the new degrees of freedom in \mathbf{V}_0^h), minus 1 (because of the increase of 1 in the dimension of Λ^h). Therefore, the number of linearly independent divergence-free vector functions on the boundary is $mn - 1$, which is the number of divergence-free vector functions given in (4.133), or (4.134). For our implementation we use the set of vector functions defined in (4.133) as a

divergence-free basis for the boundary $x = 0$. The divergence-free basis for all of the boundaries is summarized as:

$$\left. \begin{aligned} \mathbf{w}_{j+1/2,k}^{z,x0} &= \mathbf{v}_{j,k}^{x0} - \mathbf{v}_{j+1,k}^{x0} + \mathbf{v}_{j+1/2,k,0}^y \\ 0 \leq j \leq m-2, 0 \leq k \leq n-1 \\ \mathbf{w}_{k+1/2,0}^{y,x0} &= \mathbf{v}_{0,k+1}^{x0} - \mathbf{v}_{0,k}^{x0} - \mathbf{v}_{k+1/2,0,0}^z \\ j = 0, 0 \leq k \leq n-2 \end{aligned} \right\} \text{for } x = 0 \quad (4.135)$$

$$\left. \begin{aligned} \mathbf{w}_{j+1/2,k}^{z,x1} &= \mathbf{v}_{j,k}^{x1} - \mathbf{v}_{j+1,k}^{x1} - \mathbf{v}_{j+1/2,k,l-1}^y \\ 0 \leq j \leq m-2, 0 \leq k \leq n-1 \\ \mathbf{w}_{k+1/2,0}^{y,x1} &= \mathbf{v}_{0,k+1}^{x1} - \mathbf{v}_{0,k}^{x1} + \mathbf{v}_{k+1/2,l-1,0}^z \\ j = 0, 0 \leq k \leq n-2 \end{aligned} \right\} \text{for } x = 1 \quad (4.136)$$

$$\left. \begin{aligned} \mathbf{w}_{k+1/2,i}^{x,y0} &= \mathbf{v}_{k,i}^{y0} - \mathbf{v}_{k+1,i}^{y0} + \mathbf{v}_{k+1/2,i,0}^z \\ 0 \leq i \leq l-1, 0 \leq k \leq n-2 \\ \mathbf{w}_{i+1/2,0}^{z,y0} &= \mathbf{v}_{0,i+1}^{y0} - \mathbf{v}_{0,i}^{y0} - \mathbf{v}_{i+1/2,0,0}^x \\ 0 \leq i \leq l-2, k = 0 \end{aligned} \right\} \text{for } y = 0 \quad (4.137)$$

$$\left. \begin{aligned} \mathbf{w}_{k+1/2,i}^{x,y1} &= \mathbf{v}_{k,i}^{y1} - \mathbf{v}_{k+1,i}^{y1} - \mathbf{v}_{k+1/2,i,m-1}^z \\ 0 \leq i \leq l-1, 0 \leq k \leq n-2 \\ \mathbf{w}_{i+1/2,0}^{z,y1} &= \mathbf{v}_{0,i+1}^{y1} - \mathbf{v}_{0,i}^{y1} + \mathbf{v}_{i+1/2,m-1,0}^x \\ 0 \leq i \leq l-2, k = 0 \end{aligned} \right\} \text{for } y = 1 \quad (4.138)$$

$$\left. \begin{aligned} \mathbf{w}_{i+1/2,j}^{y,z0} &= \mathbf{v}_{i,j}^{z0} - \mathbf{v}_{i+1,j}^{z0} + \mathbf{v}_{i+1/2,j,0}^x \\ 0 \leq i \leq l-2, 0 \leq j \leq m-1 \\ \mathbf{w}_{j+1/2,0}^{x,z0} &= \mathbf{v}_{0,j+1}^{z0} - \mathbf{v}_{0,j}^{z0} - \mathbf{v}_{j+1/2,0,0}^y \\ i = 0, 0 \leq j \leq m-2 \end{aligned} \right\} \text{for } z = 0 \quad (4.139)$$

$$\left. \begin{aligned}
\mathbf{w}_{i+1/2,j}^{y,z1} &= \mathbf{v}_{i,j}^{z1} - \mathbf{v}_{i+1,j}^{z1} - \mathbf{v}_{i+1/2,j,n-1}^x \\
0 \leq i \leq l-2, 0 \leq j \leq m-1 \\
\mathbf{w}_{j+1/2,0}^{x,z1} &= \mathbf{v}_{0,j+1}^{z1} - \mathbf{v}_{0,j}^{z1} + \mathbf{v}_{j+1/2,n-1,0}^y \\
i = 0, 0 \leq j \leq m-2
\end{aligned} \right\} \text{for } z = 1. \quad (4.140)$$

Suppose we have Dirichlet boundary conditions on $x = 0$ and $x = 1$. This gives us $2mn$ unknown boundary fluxes. So, we need $2mn - 1$ divergence-free basis vector functions for the boundary. If we add up the number of divergence-free vector functions given in (4.135) and (4.136), then we have $2mn - 2$ vector functions. Therefore, we need one additional divergence-free vector function which is linearly independent from all the rest. The only such vector function is a global vector function, that is, one which connects both Dirichlet boundaries. We call this type of vector function a pipe function. We want to choose a basis which best models the physics of the equation. Imposing pressure conditions on opposite sides of the domain induces a Darcy flow across the domain. If the flow is constant in the positive x direction and zero in the other two directions, then the vector function describing this would look like the one in Figure 4.6.

In fact, if the conductivity tensor, \mathbf{K} , were diagonal and constant, then this pipe function would be $a(\cdot, \cdot)$ -orthogonal to all the other divergence-free vector functions and we would have the solution in one iteration. Therefore, we think of the ideal pipe function as an $a(\cdot, \cdot)$ -orthogonal projection on to the “locally” divergence-free subspace. Approximating the orthogonality of the pipe function plays an important role in the preconditioner. In general, for each additional Dirichlet boundary region we add, which is disconnected from the other Dirichlet boundaries, we need an additional pipe function. In our implementation, we can

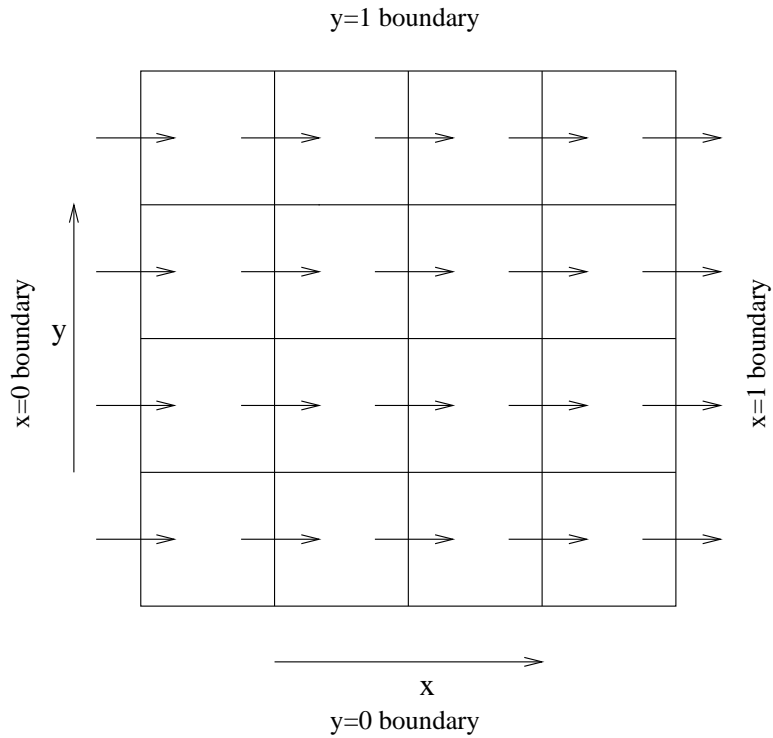


Figure 4.6: Divergence-Free Pipe Function

have at most two disconnected components of the Dirichlet boundary, requiring one pipe function. This is because if we have three sides of the domain which are Dirichlet, then they will be connected. The more general case would occur when we allow patches of Dirichlet boundaries on a given side of the domain rather than the whole side being Dirichlet.

If two adjacent sides of the domain both have Dirichlet boundary conditions, then we must define divergence-free vector functions on the edges connecting the two sides which involve fluxes from both the boundaries. These divergence-free corner functions are illustrated in Figure 4.7.

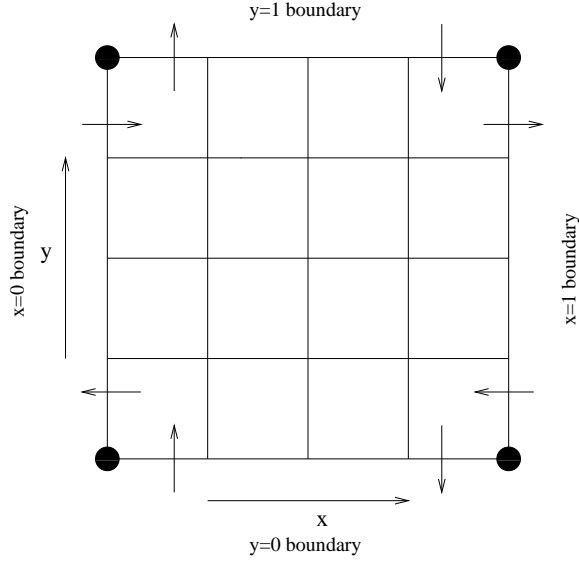


Figure 4.7: Divergence-Free Corner Functions

The divergence-free corner function on the $x = 0, y = 0$ boundary is given by

$$\begin{aligned} \mathbf{w}_k^{x0,y0} &= -\mathbf{v}_{0,k}^{x0} + \mathbf{v}_{k,0}^{y0} \\ 0 \leq k \leq n - 1. \end{aligned} \quad (4.141)$$

The vector function, $\mathbf{w}_k^{x0,y0}$, can be written as a linear combination of $\mathbf{w}_0^{x0,y0}$ plus other vector functions using the relationship

$$\mathbf{w}_k^{x0,y0} = \mathbf{w}_0^{x0,y0} - \sum_{s=0}^{k-1} \left(\mathbf{w}_{s+1/2}^{y,x0} + \mathbf{w}_{s+1/2,0}^{x,y0} \right). \quad (4.142)$$

Therefore, we only keep the vector function, $\mathbf{w}_k^{x0,y0}$, for $k = 0$. In the case of Dirichlet boundary conditions on the two sides, $x = 0$ and $y = 0$, we have $mn + ln$ unknown fluxes on the boundary. Therefore, the number of divergence-free vector functions on the boundary should be $mn + ln - 1$. If we add up the number of divergence-free functions given in (4.135) and (4.137) and add the divergence-free

corner function given in (4.141), for $k = 0$, we get $mn + ln - 1$ divergence-free vector functions. Similar results hold for the other x, y -boundaries. We also have corner functions for y, z -boundaries and z, x -boundaries. There are a total of 12 corner vector functions to consider given by

$$\mathbf{w}^{x0,y0} = -\mathbf{v}_{0,0}^{x0} + \mathbf{v}_{0,0}^{y0}, \text{ for } x = 0, y = 0 \quad (4.143)$$

$$\mathbf{w}^{x1,y0} = -\mathbf{v}_{0,0}^{x1} - \mathbf{v}_{0,0}^{y1}, \text{ for } x = 1, y = 0 \quad (4.144)$$

$$\mathbf{w}^{x0,y1} = \mathbf{v}_{0,0}^{x0} + \mathbf{v}_{0,0}^{y1}, \text{ for } x = 0, y = 1 \quad (4.145)$$

$$\mathbf{w}^{x1,y1} = \mathbf{v}_{0,0}^{x1} - \mathbf{v}_{0,0}^{y1}, \text{ for } x = 1, y = 1 \quad (4.146)$$

$$\mathbf{w}^{y0,z0} = -\mathbf{v}_{0,0}^{y0} + \mathbf{v}_{0,0}^{z0}, \text{ for } y = 0, z = 0 \quad (4.147)$$

$$\mathbf{w}^{y1,z0} = -\mathbf{v}_{0,0}^{y1} - \mathbf{v}_{0,0}^{z1}, \text{ for } y = 1, z = 0 \quad (4.148)$$

$$\mathbf{w}^{y0,z1} = \mathbf{v}_{0,0}^{y0} + \mathbf{v}_{0,0}^{z1}, \text{ for } y = 0, z = 1 \quad (4.149)$$

$$\mathbf{w}^{y1,z1} = \mathbf{v}_{0,0}^{y1} - \mathbf{v}_{0,0}^{z1}, \text{ for } y = 1, z = 1 \quad (4.150)$$

$$\mathbf{w}^{z0,x0} = -\mathbf{v}_{0,0}^{z0} + \mathbf{v}_{0,0}^{x0}, \text{ for } z = 0, x = 0 \quad (4.151)$$

$$\mathbf{w}^{z1,x0} = -\mathbf{v}_{0,0}^{z1} - \mathbf{v}_{0,0}^{x1}, \text{ for } z = 1, x = 0 \quad (4.152)$$

$$\mathbf{w}^{z0,x1} = \mathbf{v}_{0,0}^{z0} + \mathbf{v}_{0,0}^{x1}, \text{ for } z = 0, x = 1 \quad (4.153)$$

$$\mathbf{w}^{z1,x1} = \mathbf{v}_{0,0}^{z1} - \mathbf{v}_{0,0}^{x1}, \text{ for } z = 1, x = 1. \quad (4.154)$$

If the four sides, $x = 0$, $x = 1$, $y = 0$, and $y = 1$ are all Dirichlet boundaries, then we would have $2mn + 2ln$ unknown fluxes. If we add up the number of divergence-free vector functions given in (4.135)-(4.138) and (4.143)-(4.146) we end up with $2mn + 2ln$ divergence-free boundary vector functions which is one too many. It can be shown that the corner vector function, (4.146), can be written as a linear combination of other divergence-free basis vector functions. This

relationship is given by

$$\begin{aligned}
\mathbf{w}^{x1,y1} &= - \sum_{j=0}^{m-2} \sum_{i=0}^{l-2} \mathbf{w}_{i+1/2,j+1/2,0}^z \\
&\quad - \sum_{j=0}^{m-2} \left(\mathbf{w}_{j,0}^{z,x0} + \mathbf{w}_{j,0}^{z,x1} \right) \\
&\quad - \sum_{i=0}^{l-2} \left(\mathbf{w}_i^{z,y0} + \mathbf{w}_i^{z,y1} \right) \\
&\quad - \mathbf{w}^{x0,y0} - \mathbf{w}^{x1,y0} - \mathbf{w}^{x0,y1}.
\end{aligned} \tag{4.155}$$

Therefore, we eliminate (4.146) from the basis. If we have Dirichlet boundaries on all six sides of the domain, then a dimensional count would tell us that we only need five out of the twelve corner vector functions. It becomes inconvenient to determine the divergence-free basis a priori for every possible combination of Dirichlet boundaries. Knowing the correct dimension is not good enough; we need to know which corner vector functions to throw out to keep a linearly independent set. To make it computationally convenient, we use the spanning tree algorithm described in [54]. In this algorithm we define a set of vertices and a set of edges connecting the vertices. This represents a graph (see Figure 4.8). It is assumed that the graph is connected. That is, there is a series of edges, or a path, which connects any two vertices. The algorithm produces a connected graph with the same vertices, but only a subset of the edges, in such a way that there are no cycles. This is called a tree. In the context of finding a divergence-free basis we let the sides of the domain which have Dirichlet boundary conditions be represented by the vertices of the graph. Therefore, there are a maximum of six vertices in the graph. The edges of the graph correspond to the vector functions in (4.143)-(4.154). For example, the face at $x = 0$ and the face at $y = 0$ are connected by the edge corresponding to $\mathbf{w}^{x0,y0}$. The spanning tree algorithm will produce the correct number of linearly independent vector functions. With Dirichlet boundaries on all

six sides of the domain, our current implementation of the spanning tree algorithm would compute the tree shown in Figure 4.9. Based on this we would keep the vector functions (4.143), (4.144), (4.146), (4.148), and (4.150).

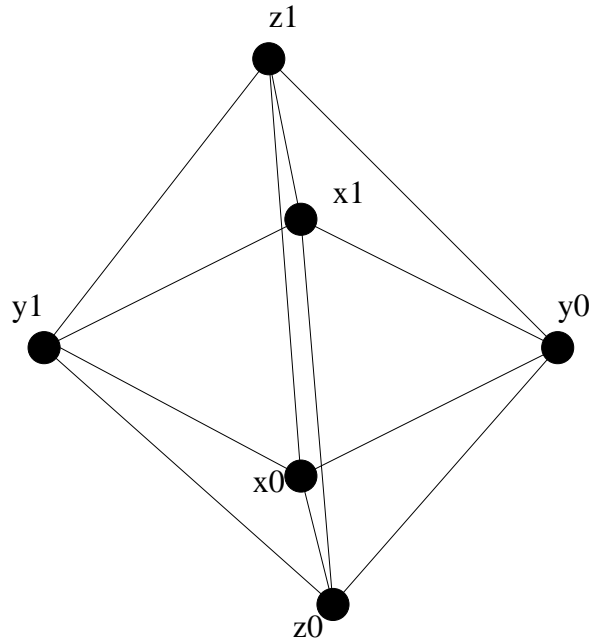


Figure 4.8: Connectivity Graph of Dirichlet Boundaries

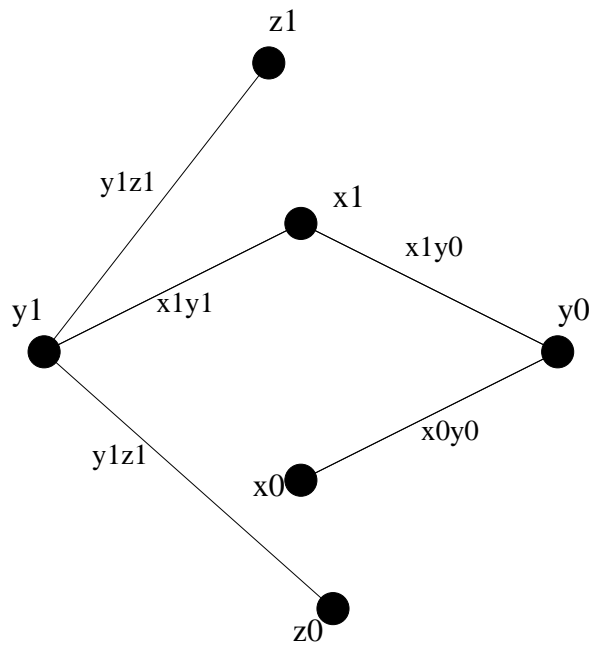


Figure 4.9: Spanning Tree

4.4 Initial Guess Vector \mathbf{v}_I^h

We define a vector, $\mathbf{f} = f_{i,j,k}$, which represents a source/sink in each element $Q_{i,j,k}$. We think of this value as being a point source/sink averaged over the volume of the element. We want to compute a vector, $\mathbf{v}_I^h \in \mathbf{V}_0^h$, which satisfies (4.3). We will do this element by element. The vector function, \mathbf{v}_I^h , in element, $Q_{i,j,k}$, is given by

$$\begin{aligned} \mathbf{v}_I^h &= u_{i-1/2,j,k}^x \mathbf{V}_{i,j,k;-1/2}^x + u_{i+1/2,j,k}^x \mathbf{V}_{i,j,k;1/2}^x \\ &\quad + u_{j-1/2,k,i}^y \mathbf{V}_{j,k,i;0,-1/2}^y + u_{j+1/2,k,i}^y \mathbf{V}_{j,k,i;1/2}^y \\ &\quad + u_{k-1/2,i,j}^z \mathbf{V}_{k,i,j;-1/2}^z + u_{k+1/2,i,j}^z \mathbf{V}_{k,i,j;1/2}^z. \end{aligned} \quad (4.156)$$

The vector function, \mathbf{v}_g^h , is given by

$$\begin{aligned} \mathbf{v}_g^h &= g_{i-1/2,j,k}^x \mathbf{V}_{i,j,k;-1/2}^x + g_{i+1/2,j,k}^x \mathbf{V}_{i,j,k;1/2}^x \\ &\quad + g_{j-1/2,k,i}^y \mathbf{V}_{j,k,i;0,-1/2}^y + g_{j+1/2,k,i}^y \mathbf{V}_{j,k,i;1/2}^y \\ &\quad + g_{k-1/2,i,j}^z \mathbf{V}_{k,i,j;-1/2}^z + g_{k+1/2,i,j}^z \mathbf{V}_{k,i,j;1/2}^z. \end{aligned} \quad (4.157)$$

Substituting (4.156) and (4.157) into (4.3) gives:

$$\begin{aligned} b(\mathbf{v}_I^h, \lambda) &= (-u_{i-1/2,j,k}^x + u_{i+1/2,j,k}^x \\ &\quad - u_{j-1/2,k,i}^y + u_{j+1/2,k,i}^y \\ &\quad - u_{k-1/2,i,j}^z + u_{k+1/2,i,j}^z) \lambda_{i,j,k} \\ &= f_{i,j,k} \lambda_{i,j,k} - (-g_{i-1/2,j,k}^x + g_{i+1/2,j,k}^x \\ &\quad - g_{j-1/2,k,i}^y + g_{j+1/2,k,i}^y \\ &\quad - g_{k-1/2,i,j}^z + g_{k+1/2,i,j}^z) \lambda_{i,j,k} \\ &= \tilde{f}_{i,j,k} \lambda_{i,j,k}. \end{aligned} \quad (4.158)$$

For now, assume $\partial\Omega_N = \partial\Omega$. It is assumed that $\tilde{\mathbf{f}} = \tilde{f}_{i,j,k}$ satisfies the compatibility condition given by

$$\sum_{k=0}^{n-1} \sum_{j=0}^{m-1} \sum_{i=0}^{l-1} \tilde{f}_{i,j,k} = 0. \quad (4.159)$$

Then, the strategy is to satisfy (4.158) element by element. We start at the $x = 0$ side and push the mass through the domain by assigning fluxes to the faces with normals in the x direction. Each time we go to the next element in the x direction, we push the mass of the source term plus the flux from the previous element as a flux into the next element. When we get to the $x = 1$ side we push the mass plus flux along that side in the y direction going from $y = 0$ to $y = 1$. Finally, when we get to the $y = 1$ side we push the mass plus flux up a column from $z = 0$ to $z = 1$. These steps are summarized as:

$$u_{1/2,j,k}^x = \tilde{f}_{0,j,k} \quad (4.160)$$

$$0 \leq j \leq m-1, 0 \leq k \leq n-1$$

$$u_{i+1/2,j,k}^x = \tilde{f}_{i,j,k} + u_{i-1/2,j,k}^x \quad (4.161)$$

$$1 \leq i \leq l-2, 0 \leq j \leq m-1, 0 \leq k \leq n-1$$

$$u_{1/2,k,l-1}^y = \tilde{f}_{l-1,0,k} + u_{l-3/2,0,k}^x \quad (4.162)$$

$$0 \leq k \leq n-1$$

$$u_{j+1/2,k,l-1}^y = \tilde{f}_{l-1,j,k} + u_{l-3/2,j,k}^x + u_{j-1/2,k,l-1}^y \quad (4.163)$$

$$1 \leq j \leq m-2, 0 \leq k \leq n-1$$

$$u_{1/2,l-1,m-1}^z = \tilde{f}_{l-1,m-1,0} + u_{l-3/2,m-1,0}^x + u_{m-3/2,0,l-1}^y \quad (4.164)$$

$$\begin{aligned} u_{k+1/2,l-1,m-1}^z &= \tilde{f}_{l-1,m-1,k} + u_{l-3/2,m-1,k}^x \\ &+ u_{m-3/2,k,l-1}^y + u_{k-1/2,l-1,m-1}^z \end{aligned} \quad (4.165)$$

$$1 \leq k \leq n - 2.$$

The remaining fluxes are assumed to be zero. The vector function, \mathbf{v}_I^h , is now given by

$$\begin{aligned} \mathbf{v}_I^h &= \sum_{i=0}^{l-2} \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} u_{i+1/2,j,k}^x \mathbf{v}_{i+1/2,j,k}^x \\ &+ \sum_{j=0}^{m-2} \sum_{k=0}^{n-1} \sum_{i=0}^{l-1} u_{j+1/2,k,i}^y \mathbf{v}_{j+1/2,k,i}^y \\ &+ \sum_{k=0}^{n-2} \sum_{i=0}^{l-1} \sum_{j=0}^{m-1} u_{k+1/2,i,j}^z \mathbf{v}_{k+1/2,i,j}^z. \end{aligned} \quad (4.166)$$

Define the residual on element $Q_{l-1,m-1,n-1}$ as:

$$\begin{aligned} &\tilde{f}_{l-1,m-1,n-1} \lambda_{l-1,m-1,n-1} - b(\mathbf{v}_I^h, \lambda_{l-1,m-1,n-1}) \\ &= (u_{l-3/2,m-1,n-1}^x + u_{m-3/2,n-1,l-1}^y \\ &+ u_{n-3/2,l-1,m-1}^z + \tilde{f}_{l-1,m-1,n-1}) \lambda_{l-1,m-1,n-1}. \end{aligned} \quad (4.167)$$

We need to show that the residual on the element $Q_{l-1,m-1,n-1}$ given by (4.167) is zero. (The residuals on the other elements are zero by construction.) Using equations (4.160)-(4.165) we can write the fluxes in (4.167) in terms of source functions $\tilde{f}_{i,j,k}$ as follows:

$$u_{l-3/2,m-1,n-1}^x = \sum_{i=0}^{l-2} \tilde{f}_{i,m-1,n-1} \quad (4.168)$$

$$\begin{aligned} u_{m-3/2,n-1,l-1}^y &= \sum_{j=0}^{m-2} (\tilde{f}_{l-1,j,n-1} + u_{l-3/2,j,n-1}^x) \\ &= \sum_{j=0}^{m-2} (\tilde{f}_{l-1,j,n-1} + \sum_{i=0}^{l-2} \tilde{f}_{i,j,n-1}) \end{aligned} \quad (4.169)$$

$$\begin{aligned} u_{n-3/2,l-1,m-1}^z &= \sum_{k=0}^{n-2} \tilde{f}_{l-1,m-1,k} + u_{l-3/2,m-1,k}^x + u_{m-3/2,k,l-1}^y \\ &= \sum_{k=0}^{n-2} (\tilde{f}_{l-1,m-1,k} \sum_{i=0}^{l-2} \tilde{f}_{i,m-1,k} \\ &+ \sum_{j=0}^{m-2} (\tilde{f}_{l-1,j,k} + \sum_{i=0}^{l-2} \tilde{f}_{i,j,k})). \end{aligned} \quad (4.170)$$

Substituting (4.168)-(4.170) into (4.167) gives:

$$\begin{aligned} & b(\tilde{\mathbf{v}}_I^h, \lambda_{l-1, m-1, n-1}) - \tilde{f}_{l-1, m-1, n-1} \lambda_{l-1, m-1, n-1} \\ &= \left(\sum_{k=0}^{n-1} \sum_{j=0}^{m-1} \sum_{i=0}^{l-1} \tilde{f}_{i, j, k} \right) \lambda_{l-1, m-1, n-1}. \end{aligned} \quad (4.171)$$

By (4.159) the residual in (4.171) is zero.

If we have Dirichlet boundaries, then (4.159) no longer holds. To insure a zero residual, we specify a flux on the Dirichlet boundary. The total flux on the Dirichlet boundary is set to be equal to the sum in (4.159). This way we get consistency and then, we use the same algorithm as before. This is summarized in the following algorithm:

Algorithm 4.4 Solving divergence equation.

1. Assemble the source/sink, f , and v_g^h .
2. Compute $\tilde{f} = f - \text{div } v_g^h$.
3. Compute the residual, $r = \sum \tilde{f}$.
4. Add the residual to the Dirichlet boundary fluxes, g_D .
5. Recompute $\tilde{f} = \tilde{f} - g_D$.
6. Compute $\tilde{\mathbf{v}}_I^h$ using (4.160)-(4.165).

4.5 Mass and Stiffness Matrices

Now that we have a discretization of the velocity space and a construction of the divergence-free subspace, we can define the system of equations which represents the discrete bilinear form.

For now, we will assume a pure homogeneous Neumann boundary condition.

We look for vectors $\mathbf{v}_I^h \in \mathbf{V}_0^h$ and $\mathbf{v}_D^h \in \mathbf{D}^h$ such that

$$b(\mathbf{v}_I^h, \lambda) = F(\lambda), \quad \forall \lambda \in \Lambda_0^h \quad (4.172)$$

$$a(\mathbf{v}_D^h, \mathbf{w}) = -a(\mathbf{v}_I^h, \mathbf{w}), \quad \forall \mathbf{w} \in \mathbf{D}^h. \quad (4.173)$$

The divergence-free vector function, \mathbf{v}_D^h , is given by

$$\begin{aligned} \mathbf{v}_D^h &= \sum_{k=0}^{n-2} \sum_{j=0}^{m-2} d_{j,k,0}^x \mathbf{w}_{j+1/2,k+1/2,0}^x \\ &+ \sum_{j=0}^{m-1} \sum_{i=0}^{l-2} \sum_{k=0}^{n-2} d_{k,i,j}^y \mathbf{w}_{k+1/2,i+1/2,j}^y \\ &+ \sum_{k=0}^{n-1} \sum_{j=0}^{m-2} \sum_{i=0}^{l-2} d_{i,j,k}^z \mathbf{w}_{i+1/2,j+1/2,k}^z. \end{aligned} \quad (4.174)$$

Define a multi-index, J , such that (4.174) can be written with a single index

as:

$$\mathbf{v}_D^h = \sum_{J=0}^{N-1} d_J \mathbf{w}_J \quad (4.175)$$

$$N = (m-1)(n-1) + (l-1)m(n-1) + (l-1)(m-1)n.$$

The vector, \mathbf{v}_I^h , which is a solution to (4.172) is given by

$$\begin{aligned} \mathbf{v}_I^h &= \sum_{k=0}^{n-1} \sum_{j=0}^{m-1} \sum_{i=0}^{l-2} u_{i,j,k}^x \mathbf{v}_{i+1/2,j,k}^x \\ &+ \sum_{i=0}^{l-1} \sum_{k=0}^{n-1} \sum_{j=0}^{m-2} u_{j,k,i}^y \mathbf{v}_{j+1/2,k,i}^y \\ &+ \sum_{j=0}^{m-1} \sum_{i=0}^{l-1} \sum_{k=0}^{n-2} u_{k,i,j}^z \mathbf{v}_{k+1/2,i,j}^z \end{aligned} \quad (4.176)$$

The multi-index, K , is used in (4.177) to write (4.176) using a single index as follows:

$$\mathbf{v}_I^h = \sum_{K=0}^{M-1} u_K \mathbf{v}_K \quad (4.177)$$

$$M = (l-1)mn + l(m-1)n + lm(n-1).$$

We note that $M > N$, i.e., \mathbf{v}_I^h belongs to the space \mathbf{V}_0^h , which is larger than the divergence-free subspace \mathbf{D}^h .

Let L be another multi-index like J . Then, for each $\mathbf{w}_L \in \mathbf{D}^h$ we substitute (4.175) and (4.177) into (4.173) to get:

$$\sum_{J=0}^{N-1} a(\mathbf{w}_J, \mathbf{w}_L) d_J = - \sum_{K=0}^{M-1} a(\mathbf{v}_K, \mathbf{w}_L) u_K. \quad (4.178)$$

We introduce two more multi-indices, P and Q , which go from 0 to $M - 1$ like the multi-index K . We then write \mathbf{w}_J and \mathbf{w}_L in terms of basis vector functions in \mathbf{V}_0^h as:

$$\mathbf{w}_J = \sum_{P=0}^{M-1} c_{J,P} \mathbf{v}_P \quad (4.179)$$

$$\mathbf{w}_L = \sum_{Q=0}^{M-1} c_{L,Q} \mathbf{v}_Q. \quad (4.180)$$

Substitution of (4.179) and (4.180) into (4.178) gives:

$$\sum_{Q=0}^{M-1} c_{L,Q} \sum_{P=0}^{M-1} a(\mathbf{v}_P, \mathbf{v}_Q) \sum_{J=0}^{N-1} c_{J,P} d_J = - \sum_{Q=0}^{M-1} c_{L,Q} \sum_{K=0}^{M-1} a(\mathbf{v}_K, \mathbf{v}_Q) u_K. \quad (4.181)$$

This gives us N equations with N unknowns. We define the $N \times M$ matrix \mathbf{C}_0 , the $M \times M$ mass matrix \mathbf{M}_0 , and the two coefficient vectors \mathbf{d} of length N and \mathbf{u} of length M as:

$$\mathbf{C}_0 = c_{L,Q} \quad (4.182)$$

$$\mathbf{M}_0 = a(\mathbf{v}_P, \mathbf{v}_Q) \quad (4.183)$$

$$\mathbf{C}_0 \mathbf{M}_0 \mathbf{C}_0^T = c_{L,Q} a(\mathbf{v}_P, \mathbf{v}_Q) c_{J,P} \quad (4.184)$$

$$\mathbf{d} = d_J \quad (4.185)$$

$$\mathbf{u} = u_K. \quad (4.186)$$

Substituting (4.182)-(4.186) into (4.181) for all L gives the matrix equation given by

$$\mathbf{C}_0 \mathbf{M}_0 \mathbf{C}_0^T \mathbf{d} = -\mathbf{C}_0 \mathbf{M}_0 \mathbf{u}. \quad (4.187)$$

The vector of coefficients, $\mathbf{v} = \mathbf{C0}^T \mathbf{d} + \mathbf{u}$, represents the fluxes of the approximate solution \mathbf{v}^h .

The $\mathbf{C0}$ matrix consists of ± 1 and is given by the relationships (4.113), (4.114), and (4.115). We don't explicitly form the matrix $\mathbf{C0M0C0}^T$. Instead we assemble only $\mathbf{M0}$ and then define operators $\mathbf{C0}$ and $\mathbf{C0}^T$ which give the action of these matrices on a vector.

When we want to include boundary fluxes into the mass matrix we need to augment it. We will call this augmented matrix \mathbf{M} which is given by

$$\mathbf{M} = \begin{bmatrix} \mathbf{M0} & \mathbf{A1} \\ \mathbf{A1}^T & \mathbf{A2} \end{bmatrix}. \quad (4.188)$$

Likewise, we will augment our $\mathbf{C0}$ operator as follows:

$$\mathbf{C} = \begin{bmatrix} \mathbf{C0} & \mathbf{0} \\ \mathbf{C1} & \mathbf{C2} \end{bmatrix}. \quad (4.189)$$

The $\mathbf{C1}$ matrix represents contributions to the divergence-free vector functions on the boundary from fluxes across interior faces, and the $\mathbf{C2}$ matrix represents contributions to the divergence-free vector functions on the boundary from fluxes across faces on the boundary.

For a Dirichlet boundary condition, we have a specified pressure, q , on the boundary which gets added to the right-hand side of equation (4.187). For example, suppose we have a Dirichlet boundary at $x = 0$. Then, we will define the vector $\mathbf{q} = q_{j,k}$, as the specified pressure on each face, $F_{0,i,j}^x(0)$, of this boundary. When we substitute this in to the functional, $G(\mathbf{w}_{j,k}^{x0})$, for each j and k we get

$$G(\mathbf{w}_{j,k}^{x0}) = \int_{F_{0,j+1,k}^x(0)} q_{j+1,k} \mathbf{v}_{j+1,k}^{x0} \cdot \mathbf{n}_{0,j+1,k}^x(0, y, z) dydz$$

$$\begin{aligned}
& - \int_{F_{0,j,k}^x(0)} q_{j,k} \mathbf{v}_{j,k}^{x0} \cdot \mathbf{n}_{0,j,k}^x(0, y, z) dydz \\
& = \int_0^1 \int_0^1 q_{j+1,k} \frac{\Gamma_{0,j+1,k}^x(0, \hat{x}, \hat{y})}{A_{0,j+1,k}^x(0)} d\hat{y}d\hat{z} \\
& - \int_0^1 \int_0^1 q_{j,k} \frac{\Gamma_{0,j,k}^x(0, \hat{x}, \hat{y})}{A_{0,j,k}^x(0)} d\hat{y}d\hat{z} \\
& = q_{j+1,k} - q_{j,k} \\
& = -\mathbf{C}2\mathbf{q}. \tag{4.190}
\end{aligned}$$

Equation (4.187) now becomes

$$\mathbf{CMC}^T \mathbf{d} = -\mathbf{CMu} + \mathbf{C}2\mathbf{q}. \tag{4.191}$$

If we have disconnected Dirichlet boundaries, then we will also have to include the pipe function. The pipe function adds a dense column and dense row of non-zeros to the matrix \mathbf{CMC}^T . We will deal with this bordered matrix when we describe the iterative solver.

4.5.1 Mass Matrix

The mass matrix, \mathbf{M} , is assembled from basis vector functions in \mathbf{V}_0^h . The \mathbf{M} matrix can best be visualized in block form. We will describe the submatrices which make up the blocks. We could define a subroutine to assemble each submatrix. However, we take advantage of the type of permutation described in Section 4.1 to minimize the number of subroutines required.

Let $\mathbf{M}0$ be the mass matrix assembled from basis vector functions in \mathbf{V}_0^h . The mass matrix, $\mathbf{M}0$, has $(l-1)mn + l(m-1)n + lm(n-1)$ rows and columns. On a rectangular grid with diagonal tensor \mathbf{K} , the $\mathbf{M}0$ matrix has the diagonal-block

form given by

$$\mathbf{M0} = \begin{bmatrix} \mathbf{M0}_{xx} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{M0}_{yy} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{M0}_{zz} \end{bmatrix}. \quad (4.192)$$

Since a basis vector function in \mathbf{V}_0^h has support on two elements we get a tridiagonal block form for the sub-matrices $\mathbf{M0}_{xx}$, $\mathbf{M0}_{yy}$, and $\mathbf{M0}_{zz}$. We define the weights which will make up the non-zero entries of $\mathbf{M0}_{xx}$ as:

$$m_{i,j,k;1/2,-1/2}^x = a(\mathbf{v}_{i,j,k;1/2}^x, \mathbf{v}_{i,j,k;-1/2}^x) \quad (4.193)$$

$$m_{i,j,k;1/2,1/2}^x = a(\mathbf{v}_{i,j,k;1/2}^x, \mathbf{v}_{i,j,k;1/2}^x) \quad (4.194)$$

$$m_{i,j,k;-1/2,-1/2}^x = a(\mathbf{v}_{i,j,k;-1/2}^x, \mathbf{v}_{i,j,k;-1/2}^x). \quad (4.195)$$

The tridiagonal entries $ml_{i,j,k}$, $md_{i,j,k}$, and $mu_{i,j,k}$ used in assembling $\mathbf{M0}_{xx}$ are given by

$$\begin{aligned} M0xx.ml_{i-1,j,k} &= m_{i,j,k;1/2,-1/2}^x \\ M0xx.md_{i,j,k} &= m_{i,j,k;1/2,1/2}^x + m_{i+1,j,k;-1/2,-1/2}^x \\ M0xx.mu_{i,j,k} &= m_{i+1,j,k;1/2,-1/2}^x. \end{aligned} \quad (4.196)$$

The weights $ml_{i-1,j,k}$, $md_{i,j,k}$, and $mu_{i,j,k}$ represent the lower diagonal, the diagonal and the upper diagonal elements of $\mathbf{M0}_{xx}$ respectively. In the MFE method, the matrix is symmetric so we need to assemble only the upper or lower diagonals along with the diagonal.

We assume that we are using vector functions defined using the area correction factors given in (4.87)-(4.92). To assemble the weights in (4.193)-(4.195) for the MFE method, we need to compute the following integrals:

$$I_{i,j,k;1/2,-1/2}^x = \int_{Q_{i,j,k}} \hat{x}(1 - \hat{x}) \frac{(\Gamma_{i,j,k}^x)^2}{J_{i,j,k}^2} (\mathbf{K}_{i,j,k}^{-1} \mathbf{X}_{i,j,k}) \cdot \mathbf{X}_{i,j,k} dx dy dz$$

$$= \int_{\hat{Q}} \hat{x}(1 - \hat{x}) \frac{(\Gamma_{i,j,k}^x)^2}{J_{i,j,k}} \left(\mathbf{K}_{i,j,k}^{-1} \mathbf{X}_{i,j,k} \right) \cdot \mathbf{X}_{i,j,k} d\hat{x}d\hat{y}d\hat{z} \quad (4.197)$$

$$\begin{aligned} I_{i,j,k;1/2,1/2}^x &= \int_{Q_{i,j,k}} \hat{x}^2 \frac{(\Gamma_{i,j,k}^x)^2}{J_{i,j,k}^2} \left(\mathbf{K}_{i,j,k}^{-1} \mathbf{X}_{i,j,k} \right) \cdot \mathbf{X}_{i,j,k} dx dy dz \\ &= \int_{\hat{Q}} \hat{x}^2 \frac{(\Gamma_{i,j,k}^x)^2}{J_{i,j,k}} \left(\mathbf{K}_{i,j,k}^{-1} \mathbf{X}_{i,j,k} \right) \cdot \mathbf{X}_{i,j,k} d\hat{x}d\hat{y}d\hat{z} \end{aligned} \quad (4.198)$$

$$\begin{aligned} I_{i,j,k;-1/2,-1/2}^x &= \int_{Q_{i,j,k}} (1 - \hat{x})^2 \frac{(\Gamma_{i,j,k}^x)^2}{J_{i,j,k}^2} \left(\mathbf{K}_{i,j,k}^{-1} \mathbf{X}_{i,j,k} \right) \cdot \mathbf{X}_{i,j,k} dx dy dz \\ &= \int_{\hat{Q}} (1 - \hat{x})^2 \frac{(\Gamma_{i,j,k}^x)^2}{J_{i,j,k}} \left(\mathbf{K}_{i,j,k}^{-1} \mathbf{X}_{i,j,k} \right) \cdot \mathbf{X}_{i,j,k} d\hat{x}d\hat{y}d\hat{z}. \end{aligned} \quad (4.199)$$

When all the faces are planar we can write analytical forms of (4.197)-(4.199). In general, we will need to numerically approximate the integrals. The weights (4.193)-(4.195) are now given by

$$m_{i,j,k;1/2,-1/2}^x = \frac{I_{i,j,k;1/2,-1/2}^x}{A_{i,j,k}^x(1)A_{i,j,k}^x(0)} \quad (4.200)$$

$$m_{i,j,k;1/2,1/2}^x = \frac{I_{i,j,k;1/2,1/2}^x}{A_{i,j,k}^x(1)^2} \quad (4.201)$$

$$m_{i,j,k;-1/2,-1/2}^x = \frac{I_{i,j,k;-1/2,-1/2}^x}{A_{i,j,k}^x(0)^2}. \quad (4.202)$$

We use permutations of the data to obtain the weights for $\mathbf{M0}_{yy}$ and $\mathbf{M0}_{zz}$.

In the presence of a non-orthogonal grid or full tensor \mathbf{K} , the matrix, $\mathbf{M0}$, will have off-diagonal blocks representing "transverse" flux contributions. In this case the block structure of $\mathbf{M0}$ is given by

$$\mathbf{M0} = \begin{bmatrix} \mathbf{M0}_{xx} & \mathbf{M0}_{xy} & \mathbf{M0}_{xz} \\ \mathbf{M0}_{yx} & \mathbf{M0}_{yy} & \mathbf{M0}_{yz} \\ \mathbf{M0}_{zx} & \mathbf{M0}_{zy} & \mathbf{M0}_{zz} \end{bmatrix}. \quad (4.203)$$

We define the weights which will make up the non-zero entries of $\mathbf{M0}_{xy}$ as:

$$m_{j,k,i;1/2,-1/2}^{xy} = a(\mathbf{v}_{i,j,k;1/2}^x, \mathbf{v}_{j,k,i;-1/2}^y) \quad (4.204)$$

$$m_{j,k,i;1/2,1/2}^{xy} = a(\mathbf{v}_{i,j,k;1/2}^x, \mathbf{v}_{j,k,i;1/2}^y) \quad (4.205)$$

$$m_{j,k,i;-1/2,-1/2}^{xy} = a(\mathbf{v}_{i,j,k;-1/2}^x, \mathbf{v}_{j,k,i;-1/2}^y) \quad (4.206)$$

$$m_{j,k,i;-1/2,1/2}^{xy} = a(\mathbf{v}_{i+1,j,k;-1/2,0,0}, \mathbf{v}_{j,k,i+1,0,1/2,0}). \quad (4.207)$$

The non-zero entries of $\mathbf{M0}_{xy}$ are now given by

$$\begin{aligned} M0xy.m1_{j-1,k,i} &= m_{j,k,i;1/2,-1/2}^{xy} \\ M0xy.m2_{j,k,i} &= m_{j,k,i;1/2,1/2}^{xy} \\ M0xy.m3_{j-1,k,i} &= m_{j,k,i+1;-1/2,-1/2}^{xy} \\ M0xy.m4_{j,k,i} &= m_{j,k,i+1;-1/2,1/2}^{xy}. \end{aligned} \quad (4.208)$$

These sub-matrices are sparse, but have a large bandwidth due to the ordering of the unknowns.

To assemble the weights in (4.204)-(4.207) for the MFE method, we need to compute the following integrals:

$$\begin{aligned} I_{j,k,i;1/2,-1/2}^{xy} &= \int_{Q_{i,j,k}} \hat{x}(1-\hat{y}) \frac{\Gamma_{i,j,k}^x \Gamma_{j,k,i}^y}{J_{i,j,k}^2} (\mathbf{K}_{i,j,k}^{-1} \mathbf{X}_{i,j,k}) \cdot \mathbf{Y}_{j,k,i} dx dy dz \\ &= \int_{\hat{Q}} \hat{x}(1-\hat{y}) \frac{\Gamma_{i,j,k}^x \Gamma_{j,k,i}^y}{J_{i,j,k}} (\mathbf{K}_{i,j,k}^{-1} \mathbf{X}_{i,j,k}) \cdot \mathbf{Y}_{j,k,i} d\hat{x} d\hat{y} d\hat{z} \end{aligned} \quad (4.209)$$

$$\begin{aligned} I_{j,k,i;1/2,1/2}^{xy} &= \int_{Q_{i,j,k}} \hat{x}\hat{y} \frac{\Gamma_{i,j,k}^x \Gamma_{j,k,i}^y}{J_{i,j,k}^2} (\mathbf{K}_{i,j,k}^{-1} \mathbf{X}_{i,j,k}) \cdot \mathbf{Y}_{j,k,i} dx dy dz \\ &= \int_{\hat{Q}} \hat{x}\hat{y} \frac{\Gamma_{i,j,k}^x \Gamma_{j,k,i}^y}{J_{i,j,k}} (\mathbf{K}_{i,j,k}^{-1} \mathbf{X}_{i,j,k}) \cdot \mathbf{Y}_{j,k,i} d\hat{x} d\hat{y} d\hat{z} \end{aligned} \quad (4.210)$$

$$I_{j,k,i;-1/2,-1/2}^{xy} = \int_{Q_{i,j,k}} (1-\hat{x})(1-\hat{y}) \frac{\Gamma_{i,j,k}^x \Gamma_{j,k,i}^y}{J_{i,j,k}^2} (\mathbf{K}_{i,j,k}^{-1} \mathbf{X}_{i,j,k}) \cdot \mathbf{Y}_{j,k,i} dx dy dz$$

$$= \int_{\hat{Q}} (1 - \hat{x})(1 - \hat{y}) \frac{\Gamma_{i,j,k}^x \Gamma_{j,k,i}^y}{J_{i,j,k}} \left(\mathbf{K}_{i,j,k}^{-1} \mathbf{X}_{i,j,k} \right) \cdot \mathbf{Y}_{j,k,i} d\hat{x}d\hat{y}d\hat{z} \quad (4.211)$$

$$\begin{aligned} I_{j,k,i;-1/2,1/2}^{xy} &= \int_{Q_{i,j,k}} (1 - \hat{x})\hat{y} \frac{\Gamma_{i,j,k}^x \Gamma_{j,k,i}^y}{J_{i+1,j,k}^2} \left(\mathbf{K}_{i,j,k}^{-1} \mathbf{X}_{i,j,k} \right) \cdot \mathbf{Y}_{j,k,i} dx dy dz \\ &= \int_{\hat{Q}} (1 - \hat{x})\hat{y} \frac{\Gamma_{i,j,k}^x \Gamma_{j,k,i}^y}{J_{i,j,k}} \left(\mathbf{K}_{i,j,k}^{-1} \mathbf{X}_{i,j,k} \right) \cdot \mathbf{Y}_{j,k,i} d\hat{x}d\hat{y}d\hat{z}. \end{aligned} \quad (4.212)$$

The weights (4.204)-(4.207) are now given by

$$m_{j,k,i;1/2,-1/2}^{xy} = \frac{I_{j,k,i;1/2,-1/2}^{xy}}{A_{i,j,k}^x(1)A_{j,k,i}^y(0)} \quad (4.213)$$

$$m_{j,k,i;1/2,1/2}^{xy} = \frac{I_{j,k,i;1/2,1/2}^{xy}}{A_{i,j,k}^x(1)A_{j,k,i}^y(1)} \quad (4.214)$$

$$m_{j,k,i;-1/2,-1/2}^{xy} = \frac{I_{j,k,i;-1/2,-1/2}^{xy}}{A_{i,j,k}^x(0)A_{j,k,i}^y(0)} \quad (4.215)$$

$$m_{j,k,i;-1/2,1/2}^{xy} = \frac{I_{j,k,i;-1/2,1/2}^{xy}}{A_{i,j,k}^x(0)A_{j,k,i}^y(1)}. \quad (4.216)$$

We can obtain the weights for the $\mathbf{M0}_{yz}$ matrix and the $\mathbf{M0}_{zx}$ matrix through permutations of the data. In the MFE method we need not assemble $\mathbf{M0}_{xz}$ which is the transpose of $\mathbf{M0}_{zx}$, or $\mathbf{M0}_{yx}$ which is the transpose of $\mathbf{M0}_{xy}$, or $\mathbf{M0}_{zy}$ which is the transpose of $\mathbf{M0}_{yz}$ since these are all self-adjoint.

The non-zero structure of the $\mathbf{M0}$ matrix, with off-diagonal blocks, is shown in Figure 4.10. On an orthogonal grid with diagonal tensor, \mathbf{K} , we would have just the tridiagonal blocks shown along the diagonal in Figure 4.10.

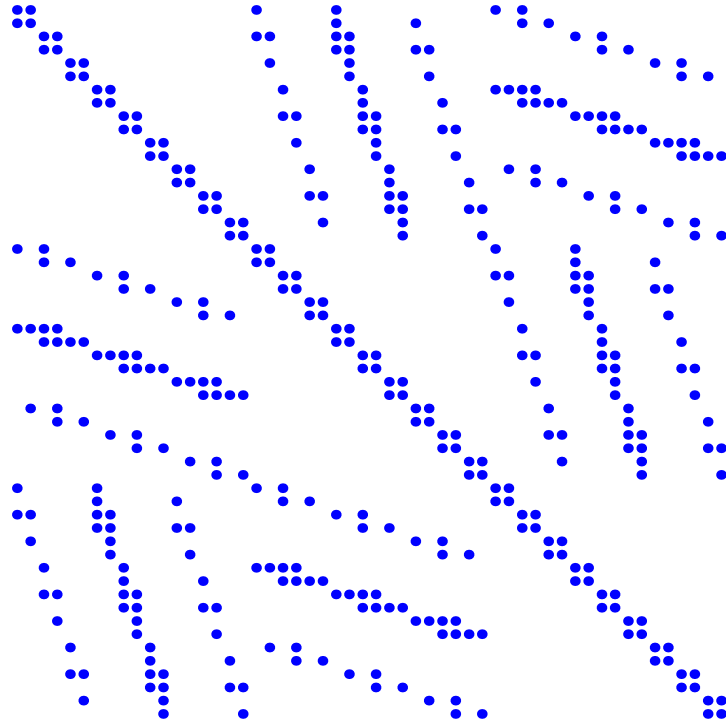


Figure 4.10: Non-Zero Structure of \mathbf{M}_0 on Non-Orthogonal Grid

4.5.1.1 Mass Matrix with Boundary Fluxes

Now we consider the case when we need to compute contributions from fluxes on the boundary. As mentioned above, this will add additional blocks to our mass matrix giving the augmented matrix, \mathbf{M} , in (4.188). The block structure of \mathbf{A}_1 is given by

$$\mathbf{A}_1 = \begin{bmatrix} \mathbf{A}_{1_{xx}} & \mathbf{A}_{1_{xy}} & \mathbf{A}_{1_{xz}} \\ \mathbf{A}_{1_{yx}} & \mathbf{A}_{1_{yy}} & \mathbf{A}_{1_{yz}} \\ \mathbf{A}_{1_{zx}} & \mathbf{A}_{1_{zy}} & \mathbf{A}_{1_{zz}} \end{bmatrix}. \quad (4.217)$$

The integrals for assembling the weights for the $\mathbf{A1}$ matrix are analogous to those computed for $\mathbf{M0}$ and we omit them here. $\mathbf{A1}_{xx}$ is assembled by computing the weights given by

$$\begin{aligned} A1xx.ml_{j,k} &= a(\mathbf{v}_{0,j,k;1/2}^x, \mathbf{v}_{j,k}^{x0}) \\ A1xx.mu_{j,k} &= a(\mathbf{v}_{l-1,j,k;-1/2}^x, \mathbf{v}_{j,k}^{x1}) \end{aligned} \quad (4.218)$$

The blocks $\mathbf{A1}_{yy}$ and $\mathbf{A1}_{zz}$ are assembled through permutations of the data.

The $\mathbf{A1}_{xy}$ represents the “transverse” flux contribution from the y -boundary to the x -fluxes. The weights for this matrix are given by

$$\begin{aligned} A1xy.m1_{k,i} &= a(\mathbf{v}_{i,0,k;1/2}^x, \mathbf{v}_{k,i}^{y0}) \\ A1xy.m2_{k,i} &= a(\mathbf{v}_{i+1,0,k;-1/2}^x, \mathbf{v}_{k,i+1}^{y0}) \\ A1xy.m3_{k,i} &= a(\mathbf{v}_{i,m-1,k;1/2}^x, \mathbf{v}_{k,i}^{y1}) \\ A1xy.m4_{k,i} &= a(\mathbf{v}_{i+1,m-1,k;-1/2}^x, \mathbf{v}_{k,i+1}^{y1}) \end{aligned} \quad (4.219)$$

The blocks $\mathbf{A1}_{yz}$ and $\mathbf{A1}_{zx}$ are assembled through permutations of the data.

The weights for the block, $\mathbf{A1}_{yx}$, are given by

$$\begin{aligned} A1yx.m1_{j,k} &= a(\mathbf{v}_{j,k,0;1/2}^y, \mathbf{v}_{j,k}^{x0}) \\ A1yx.m2_{j,k} &= a(\mathbf{v}_{j+1,k,0;-1/2}^y, \mathbf{v}_{j+1,k}^{x0}) \\ A1yx.m3_{j,k} &= a(\mathbf{v}_{j,k,l-1;1/2}^y, \mathbf{v}_{j,k}^{x1}) \\ A1yx.m4_{j,k} &= a(\mathbf{v}_{j+1,k,l-1;-1/2}^y, \mathbf{v}_{j+1,k}^{x1}) \end{aligned} \quad (4.220)$$

The blocks $\mathbf{A1}_{zy}$ and $\mathbf{A1}_{xz}$ are assembled through permutations of the data.

The block structure of the transpose of the $\mathbf{A1}$ matrix is given as:

$$\mathbf{A1}^T = \begin{bmatrix} \mathbf{A1}_{xx}^T & \mathbf{A1}_{xy}^T & \mathbf{A1}_{xz}^T \\ \mathbf{A1}_{yx}^T & \mathbf{A1}_{yy}^T & \mathbf{A1}_{yz}^T \\ \mathbf{A1}_{zx}^T & \mathbf{A1}_{zy}^T & \mathbf{A1}_{zz}^T \end{bmatrix} \quad (4.221)$$

In the MFE method, the matrix $\mathbf{A1}$ is self adjoint so we don't need to assemble the transpose. The submatrices of $\mathbf{A1}^T$ will have the same lexicographical ordering as those in the $\mathbf{A1}$ matrix. The submatrices of $\mathbf{A1}^T$ will just be aliases or references to the $\mathbf{A1}$ submatrices and have the same memory addresses.

The block structure of $\mathbf{A2}$ is given by

$$\mathbf{A2} = \begin{bmatrix} \mathbf{A2}_{xx} & \mathbf{A2}_{xy} & \mathbf{A2}_{xz} \\ \mathbf{A2}_{yx} & \mathbf{A2}_{yy} & \mathbf{A2}_{yz} \\ \mathbf{A2}_{zx} & \mathbf{A2}_{zy} & \mathbf{A2}_{zz} \end{bmatrix}. \quad (4.222)$$

The $\mathbf{A2}_{xx}$ block is assembled from the weights given by

$$\begin{aligned} A2xx0_{j,k} &= a(\mathbf{v}_{j,k}^{x0}, \mathbf{v}_{j,k}^{x0}) \\ A2xx1_{j,k} &= a(\mathbf{v}_{j,k}^{x1}, \mathbf{v}_{j,k}^{x1}) \end{aligned}. \quad (4.223)$$

The weights for $\mathbf{A2}_{yy}$ and $\mathbf{A2}_{zz}$ are assembled through permutations in the data.

The off-diagonal submatrices of $\mathbf{A2}$ are from corners where two sides of the domain are adjacent. For example, the $\mathbf{A2}_{xy}$ block represents contributions from the corners where the $x = 0$ and $x = 1$ sides are adjacent to the $y = 0$ and $y = 1$ sides. The weights for the $\mathbf{A2}_{xy}$ block are given by

$$\begin{aligned} A2xy.m1_k &= a(\mathbf{v}_{0,k}^{x0}, \mathbf{v}_{k,0}^{y0}) \\ A2xy.m2_k &= a(\mathbf{v}_{m-1,k}^{x0}, \mathbf{v}_{k,0}^{y1}) \\ A2xy.m3_k &= a(\mathbf{v}_{0,k}^{x1}, \mathbf{v}_{k,l-1}^{y0}) \\ A2xy.m4_k &= a(\mathbf{v}_{m-1,k}^{x1}, \mathbf{v}_{k,l-1}^{y1}) \end{aligned}. \quad (4.224)$$

The matrices $\mathbf{A2}_{yz}$ and $\mathbf{A2}_{zx}$ are assembled through permutations in the data.

The non-zero structures of the $\mathbf{A1}$ and $\mathbf{A2}$ matrices on a non-orthogonal grid are shown in Figures 4.11 and 4.12 respectively. The non-zero structure of the \mathbf{M} matrix on a non-orthogonal grid with boundary fluxes is shown in Figure 4.13.

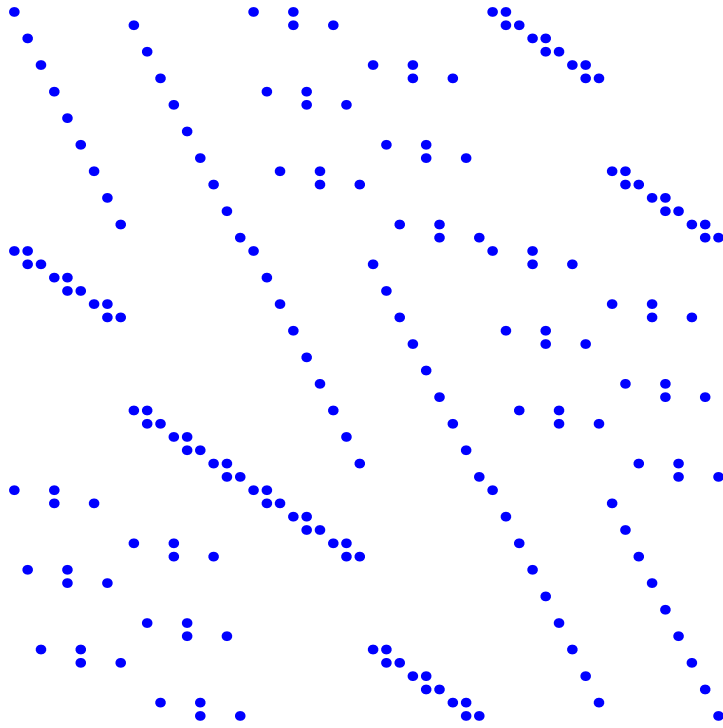


Figure 4.11: Non-Zero Structure of \mathbf{A}_1

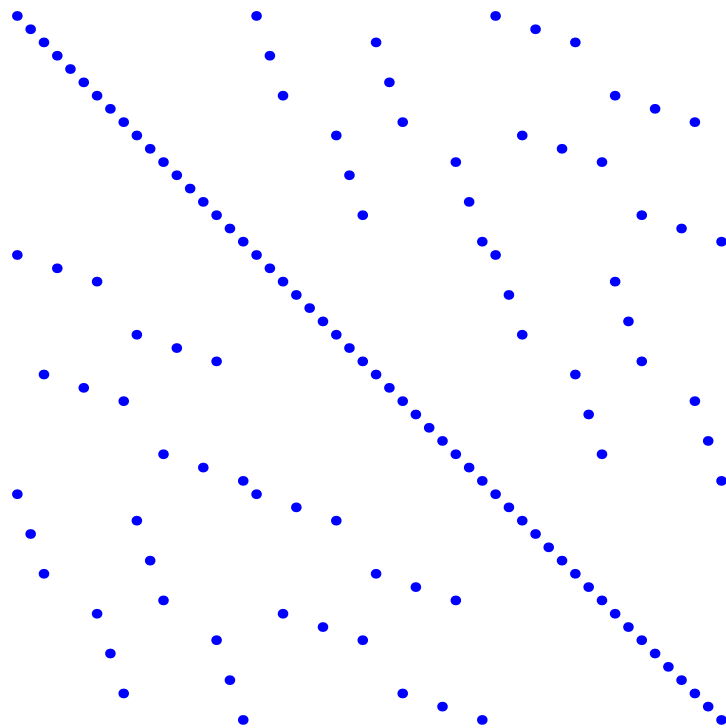


Figure 4.12: Non-Zero Structure of \mathbf{A}_2

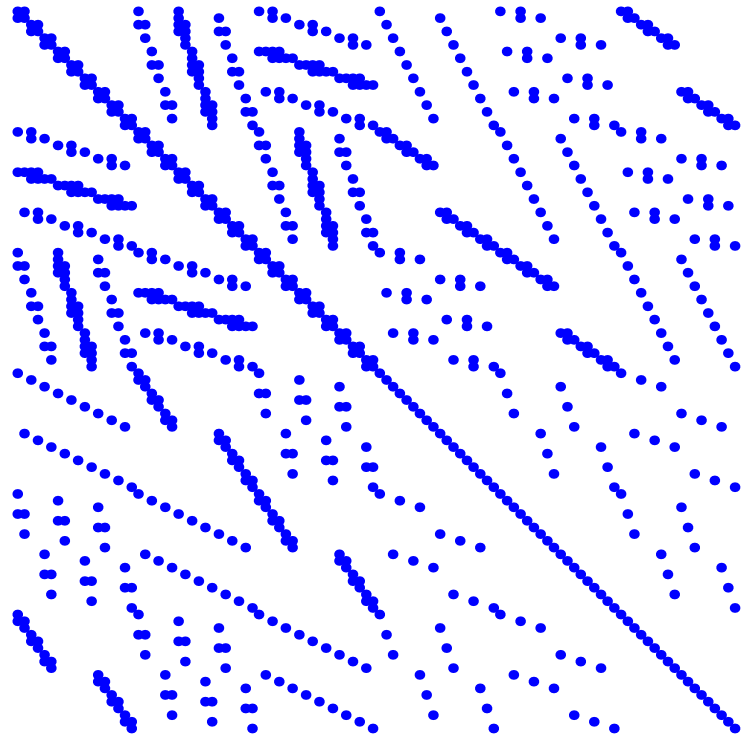


Figure 4.13: Non-Zero Structure of M

4.5.2 Stiffness Matrix

We may want to assemble the \mathbf{CMC}^T matrix explicitly so that we can do a direct solve on subdomains or the coarse grid. The non-zero structure of \mathbf{CMC}^T is shown in Figure 4.14 without boundary contributions and in Figure 4.15 with boundary contributions. We note that the bandwidth is very large.

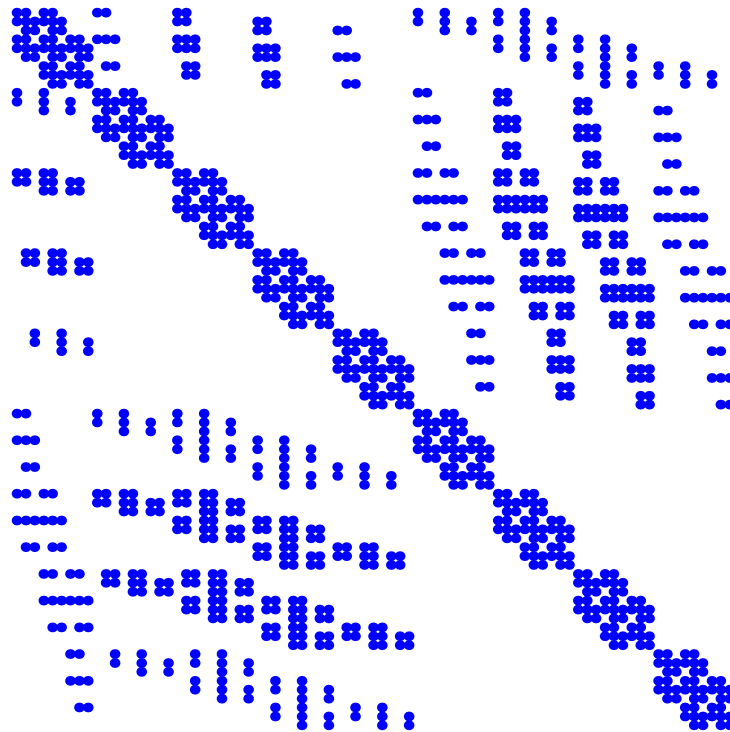


Figure 4.14: Non-Zero Structure of \mathbf{CMC}^T

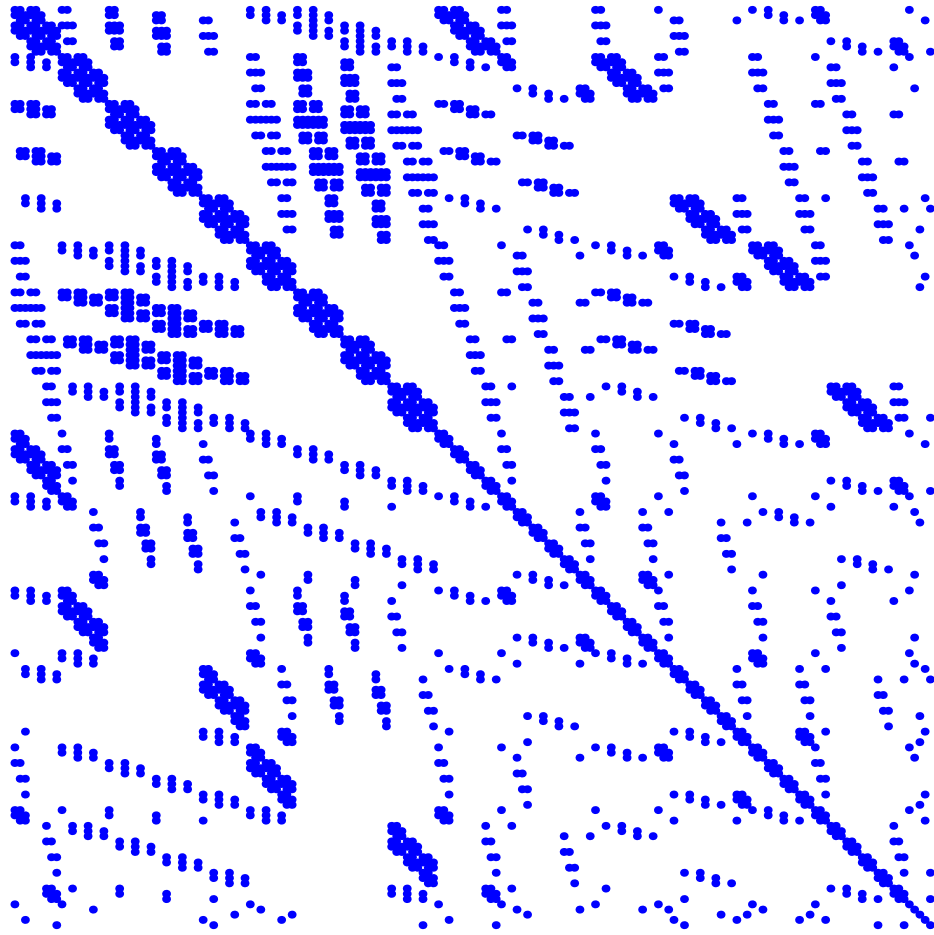


Figure 4.15: Non-Zero Structure of CMC^T with Boundary Contributions

The non-zero structure of the stiffness matrix does not depend on the conductivity tensor or the orthogonality of the grid.

Theorem 4.5 The stiffness matrix on a non-orthogonal grid with full tensor, \mathbf{K} , will not have any more non-zero entries than the stiffness matrix on an orthogonal grid with a diagonal tensor.

Proof. Suppose we have a rectangular grid and the identity matrix for the tensor \mathbf{K} . We will see that any divergence-free vector function in the basis of \mathbf{D}^h is non-orthogonal, with respect to $a(\cdot, \cdot)$, to all its neighbors (we ignore boundary functions to simplify the proof). Without loss of generality, suppose we choose $\mathbf{w}_{j+1/2, k+1/2, i}^x$ as our vector function (assume $0 < i < l - 1$). This vector function has support on four elements. All the vector functions in the basis of \mathbf{D}^h which have support overlapping the support of $\mathbf{w}_{j+1/2, k+1/2, i}^x$ are given below.

$$\begin{array}{lll}
\mathbf{w}_{j-1/2, k-1/2, i}^x & \mathbf{w}_{j+1/2, k-1/2, i}^x & \mathbf{w}_{j+3/2, k-1/2, i}^x \\
\mathbf{w}_{j-1/2, k+1/2, i}^x & \mathbf{w}_{j+1/2, k+1/2, i}^x & \mathbf{w}_{j+3/2, k+1/2, i}^x \\
\mathbf{w}_{j-1/2, k+3/2, i}^x & \mathbf{w}_{j+1/2, k+3/2, i}^x & \mathbf{w}_{j+3/2, k+3/2, i}^x \\
\mathbf{w}_{k-1/2, i-1/2, j}^y & \mathbf{w}_{k+1/2, i-1/2, j}^y & \mathbf{w}_{k+3/2, i-1/2, j}^y \\
\mathbf{w}_{k-1/2, i+1/2, j}^y & \mathbf{w}_{k+1/2, i+1/2, j}^y & \mathbf{w}_{k+3/2, i+1/2, j}^y \\
\mathbf{w}_{k-1/2, i-1/2, j+1}^y & \mathbf{w}_{k+1/2, i-1/2, j+1}^y & \mathbf{w}_{k+3/2, i-1/2, j+1}^y \\
\mathbf{w}_{k-1/2, i+1/2, j}^y & \mathbf{w}_{k+1/2, i+1/2, j}^y & \mathbf{w}_{k+3/2, i+1/2, j}^y \\
\mathbf{w}_{i-1/2, j-1/2, k}^z & \mathbf{w}_{i-1/2, j+1/2, k}^z & \mathbf{w}_{i-1/2, j+3/2, k}^z \\
\mathbf{w}_{i+1/2, j-1/2, k}^z & \mathbf{w}_{i+1/2, j+1/2, k}^z & \mathbf{w}_{i+1/2, j+3/2, k}^z
\end{array}$$

$$\begin{aligned} & \mathbf{w}_{i-1/2,j-1/2,k+1}^z, \mathbf{w}_{i-1/2,j+1/2,k+1}^z, \mathbf{w}_{i-1/2,j+3/2,k+1}^z \\ & \mathbf{w}_{i+1/2,j-1/2,k+1}^z, \mathbf{w}_{i+1/2,j+1/2,k+1}^z, \mathbf{w}_{i+1/2,j+3/2,k+1}^z \end{aligned}$$

The inner-product of $\mathbf{w}_{j+1/2,k+1/2,i}^x$ with any one of the vector functions above is non-zero. For example,

$$\begin{aligned} a(\mathbf{w}_{j-1/2,k-1/2,i}^x, \mathbf{w}_{j+1/2,k+1/2,i}^x) &= -a(\mathbf{v}_{j-1/2,k,i;0,1/2,0}, \mathbf{v}_{j-1/2,k,i;0,-1/2,0}) \\ &= -(1/6)(\mathbf{X}_{i,j,k} \cdot \mathbf{X}_{i,j,k})/(J_{i,j,k}) \\ &\neq 0 \end{aligned}$$

A similar result can be shown for each of the vector functions above either by inspection or by using the definition of a divergence-free vector function in the basis of \mathbf{D}^h . ■

Before factoring the matrix, it is best to compute a permutation which will reduce the bandwidth. For example, the Symmetric Reverse Cuthill-McKee ordering reduces the bandwidth by about half [48]. The result of the permutation is shown in Figure 4.16.



Figure 4.16: Matrix Permutation

4.6 Numerical Integration

As mentioned above, the weights for the mass matrix, \mathbf{M} , will, in general, require the use of numerical integration. The method used in our implementation is a Gaussian quadrature rule in three dimensions which is a one-dimensional Gaussian quadrature rule applied on each dimension separately. The number of integration points in each dimension is a user defined input of 1,2,3,4 or 5 points. The Gaussian quadrature formulas are found in any numerical analysis book and many finite element books (for example [44]). We use vectors, $\mathbf{x} = x_{si}$, and $\mathbf{w} = w_{si}$ to store the one-dimensional quadrature points and weights respectively.

For example, let N be the number of integration points and suppose we want to compute the integral in (4.197). Then, this would be represented as a triple sum given by

$$\begin{aligned}
 & I_{i,j,k;1/2,-1/2}^x \\
 &= c^3 \sum_{s3=0}^{N-1} w_{s3} \sum_{s2=0}^{N-1} w_{s2} \sum_{s1=0}^{N-1} \\
 & \left(w_{s1} x_{s1} (1 - x_{s1}) \frac{(\Gamma_{i,j,k;1/2}^x(x_{s1}, x_{s2}, x_{s3}))^2}{J_{i,j,k}(x_{s1}, x_{s2}, x_{s3})} \left(\mathbf{K}_{i,j,k}^{-1} \mathbf{X}_{i,j,k}(x_{s2}, x_{s3}) \right) \cdot \mathbf{X}_{i,j,k}(x_{s2}, x_{s3}) \right).
 \end{aligned} \tag{4.225}$$

The constant c is a constant of integration resulting from a change in variables in each dimension.

The 3×3 matrix, $\mathbf{K}_{i,j,k}^{-1}$, is assumed to be constant and symmetric on each element $Q_{i,j,k}$. The elements of this matrix are given by

$$\mathbf{K}_{i,j,k}^{-1} = \begin{bmatrix} K_{xx} & K_{xy} & K_{xz} \\ K_{yx} & K_{yy} & K_{yz} \\ K_{zx} & K_{zy} & K_{zz} \end{bmatrix}. \tag{4.226}$$

Since the matrix is symmetric, we only store the diagonal and 3 of the off-diagonals. The off-diagonals we choose to store and the lexicographical ordering of all these matrix elements is critical to the implementation of our method. The off-diagonals we choose to store are Kxy , Kyz , and Kzx . The matrix (4.226) is now defined as:

$$\mathbf{K}_{i,j,k}^{-1} = \begin{bmatrix} Kxx_{i,j,k} & Kxy_{j,k,i} & Kzx_{i,j,k} \\ Kxy_{j,k,i} & Kyy_{j,k,i} & Kyz_{k,i,j} \\ Kzx_{i,j,k} & Kyz_{k,i,j} & Kzz_{k,i,j} \end{bmatrix}. \quad (4.227)$$

The product, $\mathbf{K}_{i,j,k}^{-1}\mathbf{X}_{i,j,k}$, is given by

$$\mathbf{K}_{i,j,k}^{-1}\mathbf{X}_{i,j,k} = \begin{bmatrix} Kxx_{i,j,k}X_{i,j,k}^0 + Kxy_{j,k,i}X_{i,j,k}^1 + Kzx_{i,j,k}X_{i,j,k}^2 \\ Kxy_{j,k,i}X_{i,j,k}^0 + Kyy_{j,k,i}X_{i,j,k}^1 + Kyz_{k,i,j}X_{i,j,k}^2 \\ Kzx_{i,j,k}X_{i,j,k}^0 + Kyz_{k,i,j}X_{i,j,k}^1 + Kzz_{k,i,j}X_{i,j,k}^2 \end{bmatrix}. \quad (4.228)$$

We explain why we choose to define the $\mathbf{K}_{i,j,k}^{-1}$ matrix the way we do. Consider the permutations given by

$$\mathbf{K}_{i,j,k}^{-1} \rightarrow \mathbf{K}_{j,k,i}^{-1} \quad (4.229)$$

$$\mathbf{K}_{j,k,i}^{-1} = \begin{bmatrix} Kyy_{j,k,i} & Kyz_{k,i,j} & Kxy_{j,k,i} \\ Kyz_{k,i,j} & Kzz_{k,i,j} & Kzx_{i,j,i} \\ Kxy_{j,k,i} & Kzx_{i,j,k} & Kxx_{i,j,k} \end{bmatrix} \quad (4.230)$$

$$\{X^0, X^1, X^2\} \rightarrow \{Y^1, Y^2, Y^0\} \quad (4.231)$$

$$\{i, j, k\} \rightarrow \{j, k, i\}. \quad (4.232)$$

With these permutations, equation (4.228) now becomes

$$\mathbf{K}_{j,k,i}^{-1}\mathbf{Y}_{j,k,i} = \begin{bmatrix} Kyy_{j,k,i}Y_{j,k,i}^1 + Kyz_{k,i,j}Y_{j,k,i}^2 + Kxy_{j,k,i}Y_{j,k,i}^0 \\ Kyz_{k,i,j}Y_{j,k,i}^1 + Kzz_{k,i,j}Y_{j,k,i}^2 + Kzx_{i,j,k}Y_{j,k,i}^0 \\ Kxy_{j,k,i}Y_{j,k,i}^1 + Kzx_{i,j,k}Y_{j,k,i}^2 + Kxx_{i,j,k}Y_{j,k,i}^0 \end{bmatrix}. \quad (4.233)$$

Equation (4.233) gives us the vector $\mathbf{K}_{j,k,i}^{-1} \mathbf{Y}_{j,k,i}$ which has the permutation (4.231). Another, analogous, permutation gives us the permuted vector $\mathbf{K}_{k,i,j}^{-1} \mathbf{Z}_{k,i,j}$.

When we compute the covariant basis vector, $\mathbf{Y}_{j,k,i}$, we will use a permutation of the data which gives the permutation in (4.231). Therefore, the dot-product will be invariant under these permutations since the vector, $\mathbf{K}_{j,k,i}^{-1} \mathbf{Y}_{j,k,i}$, will have the same permutation as the covariant basis vector $\mathbf{Y}_{j,k,i}$. This means that the same subroutines for numerical integration in the assembly of $\mathbf{M0}_{xx}$ can be used for the assembly of $\mathbf{M0}_{yy}$ and $\mathbf{M0}_{zz}$. Our strategy of using a single subroutine for the assembly of all our tridiagonal blocks in the \mathbf{M} matrix is therefore preserved.

When assembling the $\mathbf{M0}_{xy}$ block of the mass matrix, we will need a separate subroutine for computing the product $\mathbf{K}_{i,j,k}^{-1} \mathbf{X}_{i,j,k} \cdot \mathbf{Y}_{j,k,i}$. This is because the covariant basis vectors, $\mathbf{X}_{i,j,k}$, and $\mathbf{Y}_{j,k,i}$, need to have the same ordering. However, we can permute these vectors in a consistent way when assembling $\mathbf{M0}_{yz}$ and $\mathbf{M0}_{zx}$.

For a given volume, $Q_{i,j,k}$, we compute the $3 \times N^3$ matrices $\mathbf{XV}_{i,j,k}$ and $\mathbf{XW}_{i,j,k}$ as:

$$\mathbf{XV}_{i,j,k}(s1, s2, s3) = \frac{\Gamma_{i,j,k}^x(x_{s1}, x_{s2}, x_{s3})}{J_{i,j,k}(x_{s1}, x_{s2}, x_{s3})} \mathbf{K}_{i,j,k}^{-1} \mathbf{X}_{i,j,k}(x_{s2}, x_{s3}) \quad (4.234)$$

$$\mathbf{XW}_{i,j,k}(s1, s2, s3) = \Gamma_{i,j,k}^x(x_{s1}, x_{s2}, x_{s3}) \mathbf{X}_{i,j,k}(x_{s2}, x_{s3}). \quad (4.235)$$

Now, equation (4.225) is given by

$$\begin{aligned} & a(\mathbf{v}_{i,j,k;1/2}, \mathbf{v}_{i,j,k;1/2}) \\ &= \frac{c^3}{A_{i,j,k}(0)^2} \sum_{s3=0}^{N-1} w_{s3} \sum_{s2=0}^{N-1} w_{s2} \sum_{s1=0}^{N-1} \\ & w_{s1} x_{s1}^2 \mathbf{XV}_{i,j,k}(s1, s2, s3) \cdot \mathbf{XW}_{i,j,k}(s1, s2, s3). \end{aligned} \quad (4.236)$$

The area, $A_{i,j,k}(0)$, is computed by

$$A_{i,j,k}(0) = c^2 \sum_{s_3=0}^{N-1} w_{s_3} \sum_{s_2=0}^{N-1} w_{s_2} \Gamma_{i,j,k}^x(0, x_{s_2}, x_{s_3}). \quad (4.237)$$

The reason for defining these matrices, \mathbf{XV} , and \mathbf{XW} , separately is that we may want to change the definition of the test function and thus the definition of \mathbf{XW} . It is an attempt to make the code as generic as possible so that it can be easily modified for other numerical methods such as the CVMFE method.

The computation of the other integrals needed for the weights for the tridiagonal matrix, $\mathbf{M0}_{xx}$, are given as:

$$\begin{aligned} & a(\mathbf{v}_{i+1,j,k;-1/2}, \mathbf{v}_{i+1,j,k;-1/2}) \\ &= \frac{c^3}{A_{i+1,j,k}(0)^2} \sum_{s_3=0}^{N-1} w_{s_3} \sum_{s_2=0}^{N-1} w_{s_2} \sum_{s_1=0}^{N-1} \\ & w_{s_1} (1 - x_{s_1})^2 \mathbf{XV}_{i+1,j,k}(s_1, s_2, s_3) \cdot \mathbf{XW}_{i+1,j,k}(s_1, s_2, s_3) \end{aligned} \quad (4.238)$$

$$\begin{aligned} & a(\mathbf{v}_{i+1,j,k;-1/2}, \mathbf{v}_{i+1,j,k;1/2}) \\ &= \frac{c^3}{A_{i+1,j,k}(0)A_{i+1,j,k}(1)} \sum_{s_3=0}^{N-1} w_{s_3} \sum_{s_2=0}^{N-1} w_{s_2} \sum_{s_1=0}^{N-1} \\ & w_{s_1} (1 - x_{s_1}) x_{s_1} \mathbf{XV}_{i+1,j,k}(s_1, s_2, s_3) \cdot \mathbf{XW}_{i+1,j,k}(s_1, s_2, s_3). \end{aligned} \quad (4.239)$$

To compute the integrals for the assembly of $\mathbf{M0}_{xy}$ we will need the $3 \times N^3$ matrix \mathbf{YW} given by

$$\mathbf{YW}_{i,j,k}(s_1, s_2, s_3) = \Gamma_{j,k,i}^y(x_{s_2}, x_{s_3}, x_{s_1}) \mathbf{Y}_{j,k,i}(x_{s_3}, x_{s_1}). \quad (4.240)$$

The weights for the matrix, $\mathbf{M0}_{xy}$, are given by

$$\begin{aligned} & m_{j-1,k,i} \\ &= \frac{c^3}{A_{i,j,k}(1)A_{j,k,i}(0)} \sum_{s_3=0}^{N-1} w_{s_3} \sum_{s_2=0}^{N-1} w_{s_2} (1 - x_{s_2}) \sum_{s_1=0}^{N-1} \\ & w_{s_1} x_{s_1} \mathbf{XV}_{i,j,k}(s_1, s_2, s_3) \cdot \mathbf{YW}_{i,j,k}(s_1, s_2, s_3) \end{aligned} \quad (4.241)$$

$$\begin{aligned}
& m2_{j,k,i} \\
&= \frac{c^3}{A_{i,j,k}(1)A_{j,k,i}(1)} \sum_{s3=0}^{N-1} w_{s3} \sum_{s2=0}^{N-1} w_{s2} x_{s2} \sum_{s1=0}^{N-1} \\
& w_{s1} x_{s1} \mathbf{XV}_{i,j,k}(s1, s2, s3) \cdot \mathbf{YW}_{i,j,k}(s1, s2, s3)
\end{aligned} \tag{4.242}$$

$$\begin{aligned}
& m3_{j-1,k,i} \\
&= \frac{c^3}{A_{i+1,j,k}(0)A_{j,k,i+1}(0)} \sum_{s3=0}^{N-1} w_{s3} \sum_{s2=0}^{N-1} w_{s2} (1 - x_{s2}) \sum_{s1=0}^{N-1} \\
& w_{s1} (1 - x_{s1}) \mathbf{XV}_{i+1,j,k}(s1, s2, s3) \cdot \mathbf{YW}_{i+1,j,k}(s1, s2, s3)
\end{aligned} \tag{4.243}$$

$$\begin{aligned}
& m4_{j,k,i} \\
&= \frac{c^3}{A_{i+1,j,k}(0)A_{j,k,i+1}(1)} \sum_{s3=0}^{N-1} w_{s3} \sum_{s2=0}^{N-1} w_{s2} x_{s2} \sum_{s1=0}^{N-1} \\
& w_{s1} (1 - x_{s1}) \mathbf{XV}_{i+1,j,k}(s1, s2, s3) \cdot \mathbf{YW}_{i+1,j,k}(s1, s2, s3).
\end{aligned} \tag{4.244}$$

We can compute the weights for the **A1** and **A2** matrices in an analogous fashion.

5. Preconditioners

We now discuss the iterative solver used to approximate the divergence-free solution. Our system is symmetric positive definite which means that we can use the PCG method. It is well known that the reduction of the error on each iteration is bounded by:

$$\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \tag{5.1}$$

where κ is the condition number of the preconditioned matrix. We will show that the condition number of \mathbf{CMC}^T can be bounded, at best, by a constant times h^{-2} .

In our implementation we use a two level additive Schwarz method as a preconditioner. Numerical results show that this domain decomposition method gives a convergence rate independent of grid size. We will define the method and analyze its behavior.

We also describe the bordered matrix problem which results from having disconnect Dirichlet boundaries. A preconditioner for the bordered matrix is defined and the condition number of the preconditioned bordered matrix is analyzed.

Most of the theoretical results presented in this section assume an orthogonal grid. It should be possible to extend these results to a distorted hexahedral grid although, as mentioned before, interpolation estimates are not known for non-affine distorted grids in 3-D. A Poincaré-type inequality is presented assuming homogeneous boundary conditions. This is not a problem for Neumann boundary conditions since the Neumann boundary condition is an essential boundary

condition and is specified to be zero in the test space. With Dirichlet boundary conditions we will have non-zero fluxes on the boundary. However, the divergence-free basis vector functions are defined in such a way that they will always be zero on some of the edges on the boundary. This may make it possible to prove a Poincaré-type inequality for Dirichlet boundary conditions. How to do this is not obvious. However, a similar generalization of Poincaré’s theorem is demonstrated in [36, Lemma 3.1].

5.1 Vector Potential Space

Before we analyze the condition number of the stiffness matrix, we will need to define the discrete vector potential space on Ω^h . Mathematical theory for vector potential spaces and corresponding divergence-free spaces can be found in [36], [35], and [28]. Recall that we defined the divergence-free vector space by defining a vector potential space on the reference cube, took the **curl** of the vector potential functions, and then used the Piola-type transformation.

We will now define a vector potential space, \mathbf{U}^h , by mapping the bilinear scalar hat functions to the hexahedral elements using the isoparametric method. The x -slice scalar hat functions are given as:

$$\Phi_{j+1/2,k+1/2,i}^x = \begin{cases} \hat{\Phi}_{1/2,1/2}^x \circ T_{i,j,k}^{-1}, & \text{on } Q_{i,j,k} \\ \hat{\Phi}_{-1/2,1/2}^x \circ T_{i,j+1,k}^{-1}, & \text{on } Q_{i,j+1,k} \\ \hat{\Phi}_{-1/2,-1/2}^x \circ T_{i,j+1,k+1}^{-1}, & \text{on } Q_{i,j+1,k+1} \\ \hat{\Phi}_{1/2,-1/2}^x \circ T_{i,j,k+1}^{-1}, & \text{on } Q_{i,j,k+1}. \end{cases} \quad (5.2)$$

We can do similar transformations to define y -slice and z -slice scalar hat functions. For each divergence-free basis vector function we will have exactly

one scalar hat basis function. Recall that the x -slice divergence-free vector basis functions live on only one x -slice. We now define the vector potential space as:

$$\mathbf{U}^h = \text{span} \left\{ \begin{pmatrix} \Phi_{j+1/2, k+1/2, 0}^x \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \Phi_{k+1/2, i+1/2, j}^y \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \Phi_{i+1/2, j+1/2, k}^z \end{pmatrix} \right\}. \quad (5.3)$$

If we use the standard Piola mapping to define the velocity space and the divergence-free space, then we can define \mathbf{D}^h as $\mathbf{curl} \mathbf{U}^h$. If we are using a different Piola-type mapping, then we can construct a weakly divergence-free vector space as above in a more tedious manner incorporating, for example, area correction factors. In either case we can write $\mathbf{v}^h \in \mathbf{D}^h$ as:

$$\mathbf{v}^h = \sum_{K=0}^{N-1} p_K \mathbf{curl} \Phi_K \quad (5.4)$$

where $N = \dim \mathbf{D}^h$ and $\{\Phi_K\}$ is the basis for \mathbf{U}^h . The vector function, Φ_K , is of order 1 and therefore $\mathbf{curl} \Phi_K$ is of order h^{-1} . This means that on a face which has area order h^2 the flux of $\mathbf{curl} \Phi_K$ will be of order h . Now we write \mathbf{v}^h in terms of the basis for \mathbf{D}^h as:

$$\mathbf{v}^h = \sum_{K=0}^{N-1} d_K \mathbf{w}_K. \quad (5.5)$$

The vector functions, \mathbf{w}_K , have fluxes of order 1. Therefore, we conclude that the coefficients, d_K , must be of order h times p_k . In general, we will say that $\mathbf{v}^h \in \mathbf{D}^h$ can be represented as follows:

$$\mathbf{v}^h = \sum_{K=0}^{N-1} d_K \mathbf{w}_K = \sum_{K=0}^{N-1} p_K \mathbf{curl} \Phi_K \quad (5.6)$$

and with $\mathbf{d} = \{d_K\}$ and $\mathbf{p} = \{p_K\}$ we have

$$c_0 h^2 |\mathbf{p}|^2 \leq |\mathbf{d}|^2 \leq c_1 h^2 |\mathbf{p}|^2. \quad (5.7)$$

We now state three lemmas related to the vector potential space, \mathbf{U}^h , on a quasi-uniform partition, Ω^h .

Lemma 5.1 There exists a positive constant, C independent of h , such that for all $\Phi^h \in \mathbf{U}^h$ we have

$$\|\Phi^h\|_{L^2(\Omega)^3} \leq C \|\mathbf{curl} \Phi^h\|_{L^2(\Omega)^3}. \quad (5.8)$$

Proof. Keeping in mind our choice for the basis in \mathbf{U}^h , we have

$$\Phi = (\Phi^x, \Phi^y, \Phi^z)^T \quad (5.9)$$

where, for example,

$$\Phi^x \in \text{span}\{\Phi_{j+1/2, k+1/2, 0}^x\}. \quad (5.10)$$

Let S be the x -slice where Φ^x does not vanish. Then, we have

$$\begin{aligned} \|\Phi\|_{L^2(\Omega)^3}^2 &= \|\Phi^x\|_{L^2(S)}^2 + \|\Phi^y\|_{L^2(S)}^2 + \|\Phi^z\|_{L^2(S)}^2 \\ &+ \|\Phi^y\|_{L^2(\Omega \setminus S)}^2 + \|\Phi^z\|_{L^2(\Omega \setminus S)}^2. \end{aligned} \quad (5.11)$$

Assume we have homogeneous Neumann boundary conditions. Then, we can write $\Phi^z(x, y, z)$ as:

$$\Phi^z(x, y, z) = \int_1^x \frac{\partial \Phi^z(\tau, y, z)}{\partial x} d\tau. \quad (5.12)$$

Square both sides of (5.12) and use the Cauchy-Schwarz inequality to get

$$(\Phi^z(x, y, z))^2 \leq C(\Omega) \int_1^x \left| \frac{\partial \Phi^z(\tau, y, z)}{\partial x} \right|^2 d\tau \leq C(\Omega) \int_1^x |\mathbf{curl} \Phi(\tau, y, z)|^2 d\tau \quad (5.13)$$

on $\Omega \setminus S$. Integrating over $\Omega \setminus S$ gives us

$$\|\Phi^z\|_{L^2(\Omega \setminus S)}^2 \leq C(\Omega) \|\mathbf{curl} \Phi\|_{L^2(\Omega)^3}^2 \quad (5.14)$$

where we use the fact that on $\Omega \setminus S$ we have

$$\mathbf{curl} \Phi = \left(\frac{\partial \Phi^z}{\partial y} - \frac{\partial \Phi^y}{\partial z}, -\frac{\partial \Phi^z}{\partial x}, \frac{\partial \Phi^y}{\partial x} \right)^T \quad (5.15)$$

In the exact same manner we get

$$\|\Phi^y\|_{L^2(\Omega \setminus S)}^2 \leq C(\Omega) \|\mathbf{curl} \Phi\|_{L^2(\Omega)^3}^2. \quad (5.16)$$

Next, we estimate $\|\Phi^z\|_{L^2(S)}$ in terms of $\|\mathbf{curl} \Phi\|_{L^2(\Omega)^3}$. Since Φ^z is piecewise constant in the z -direction, we will denote $\Phi_k^z(x, y)$ as the restriction of Φ^z to the k^{th} horizontal slice of Ω . Note that Φ_k^z is linear in x within S and is zero for $x = 0$.

Therefore, we have

$$\begin{aligned} \|\Phi^z\|_{L^2(S)}^2 &= \sum_{k=1}^n h \int_0^1 \int_0^h |\Phi_k^z(x, y)|^2 dx dy \\ &\leq \sum_{k=1}^n h \int_0^1 \int_0^h |\Phi_k^z(h, y)|^2 dx dy \\ &\leq h^2 \sum_{k=1}^n \int_0^1 \int_h^1 \left| \frac{\partial \Phi_k^z(\tau, y)}{\partial x} \right|^2 d\tau dy \\ &\leq h^2 C(\Omega) \|\mathbf{curl} \Phi\|_{L^2(\Omega)^3}^2. \end{aligned} \quad (5.17)$$

We estimate $\|\Phi^y\|_{L^2(S)}$ in an analogous manner.

To estimate $\|\Phi^x\|_{L^2(S)}$ we consider the identity on S ,

$$\Phi^x(y, z) = \int_0^z \left[\frac{\partial \Phi^x(y, \tau)}{\partial z} - \frac{\partial \Phi^z(x, y, \tau)}{\partial x} \right] d\tau + \int_0^z \frac{\partial \Phi^z(x, y, \tau)}{\partial x} d\tau. \quad (5.18)$$

Noting that $\frac{\partial \Phi^x(y, \tau)}{\partial z} - \frac{\partial \Phi^z(x, y, \tau)}{\partial x}$ is a component of $\mathbf{curl} \Phi$, we get

$$\|\Phi^x\|_{L^2(S)}^2 \leq C(\Omega) \left(\|\mathbf{curl} \Phi\|_{L^2(S)^3}^2 + \left\| \frac{\partial \Phi^z}{\partial x} \right\|_{L^2(S)}^2 \right). \quad (5.19)$$

We apply the inverse estimate to the last term on the right-hand side of (5.19) and use (5.17) to get the desired result and complete the proof. \blacksquare

Remark 5.1 The difficulty in proving Lemma 5.1 for Dirichlet boundary conditions is exasperated by having to use relationship (5.15). The function Φ^y will be zero at $x = 1$ except on a vertical slice of width h . However, Φ^z can be non-zero on all the edges on the boundary $x = 1$. The function Φ^z will be zero at $y = 1$ on all but one horizontal slice of width h . However, $\partial\Phi^z/\partial y$ is not an element of $\mathbf{curl} \Phi$ so the proof can not be modified in this manner.

Remark 5.2 An a priori knowledge of the construction of the divergence-free basis is used in Lemma 5.1. Scheichl demonstrates in [54] that certain choices of the divergence-free basis can lead to bounds on the \mathbf{curl} of Φ which are not independent of the mesh size.

Lemma 5.2 There exists a positive constant, C , independent of h , such that for all $\Phi^h \in \mathbf{U}^h$ we have

$$\|\mathbf{curl}\Phi^h\|_{L^2(\Omega)^3} \leq Ch^{-1}\|\Phi^h\|_{L^2(\Omega)^3}. \quad (5.20)$$

Proof. The proof follows from the standard inverse estimate. ■

Lemma 5.3 There exist positive constants, C_0 and C_1 independent of h , such that for all $\Phi^h \in \mathbf{U}^h$ we have

$$C_0h^3|\mathbf{p}|^2 \leq \|\Phi^h\|_{L^2(\Omega)^2}^2 \leq C_1h^3|\mathbf{p}|^2 \quad (5.21)$$

where

$$\Phi^h = \sum_{K=0}^{N-1} p_K \Phi_K, \quad \mathbf{p} = \{p_K\}. \quad (5.22)$$

Proof. Lemma 5.3 is just the equivalence of the L^2 norm to the Euclidean norm in three dimensions. See [46] for example. We often say that $h^3 |\mathbf{p}|^2$ is the discrete L^2 -norm. ■

5.2 Condition Number of Stiffness Matrix

We now give the bound on the condition number of \mathbf{CMC}^T .

Theorem 5.4 Let μ_0 and μ_1 be the minimum and maximum eigenvalues, respectively, of the stiffness matrix \mathbf{CMC}^T . Then, there exist positive constants, c_0 and c_1 , independent of quasi-uniform h , such that

$$c_0 h \leq \mu_0 \tag{5.23}$$

$$\mu_1 \leq c_1 h^{-1} \tag{5.24}$$

Proof. Let $\mathbf{v}^h \in \mathbf{D}^h$ be given by $\mathbf{v}^h = \mathbf{curl} \mathbf{\Phi}^h$ for some unique $\mathbf{\Phi}^h \in \mathbf{U}^h$. Furthermore, let \mathbf{d} be the vector of coefficients associated with the divergence-free vector function, \mathbf{v}^h , and let \mathbf{p} be the vector of coefficients associated with the vector potential function $\mathbf{\Phi}^h$. Then,

$$\begin{aligned}
\frac{\mathbf{d}^T \mathbf{C} \mathbf{M} \mathbf{C}^T \mathbf{d}}{|\mathbf{d}|^2} &= \frac{a(\mathbf{v}^h, \mathbf{v}^h)}{|\mathbf{d}|^2} = \frac{a(\mathbf{curl} \Phi^h, \mathbf{curl} \Phi^h)}{|\mathbf{d}|^2} \\
&\leq \gamma \frac{\|\mathbf{curl} \Phi^h\|_{L^2(\Omega)^3}^2}{|\mathbf{d}|^2} && \text{by the continuity of } a(\cdot, \cdot) \\
&\leq \gamma \frac{\|\mathbf{curl} \Phi^h\|_{L^2(\Omega)^3}^2}{c_0 h^2 |\mathbf{p}|^2} && \text{by 5.7} \\
&\leq \gamma C h^{-2} \frac{\|\Phi^h\|_{L^2(\Omega)^3}^2}{c_0 h^2 |\mathbf{p}|^2} && \text{by Lemma 5.2} \\
&\leq C h^{-1} && \text{by Lemma 5.3}
\end{aligned} \tag{5.25}$$

and

$$\begin{aligned}
\frac{\mathbf{d}^T \mathbf{C} \mathbf{M} \mathbf{C}^T \mathbf{d}}{|\mathbf{d}|^2} &= \frac{a(\mathbf{v}^h, \mathbf{v}^h)}{|\mathbf{d}|^2} = \frac{a(\mathbf{curl} \Phi^h, \mathbf{curl} \Phi^h)}{|\mathbf{d}|^2} \\
&\geq \alpha \frac{\|\mathbf{curl} \Phi^h\|_{L^2(\Omega)^3}^2}{|\mathbf{d}|} && \text{by the coercivity of } a(\cdot, \cdot) \\
&\geq \alpha \frac{\|\mathbf{curl} \Phi^h\|_{L^2(\Omega)^3}^2}{c_1 h^2 |\mathbf{p}|^2} && \text{by 5.7} \\
&\geq \alpha C \frac{\|\Phi^h\|_{L^2(\Omega)^3}^2}{c_1 h^2 |\mathbf{p}|^2} && \text{by Lemma 5.1} \\
&\geq C h && \text{by Lemma 5.3}
\end{aligned} \tag{5.26}$$

■

Numerical tests confirm Theorem 5.4. We use the symmetric power method to approximate the minimum and maximum eigenvalues for different size grids with conductivity tensor, \mathbf{K} , equal to the identity (we experiment with variable conductivities later). The power method approximates the largest eigenvalue. To

approximate the smallest eigenvalue we can either use the inverse power method or use the shifted power method. The minimum eigenvalue is difficult to approximate indicating that there are multiple eigenvalues close to the smallest eigenvalue. The inverse power method tends to underestimate the smallest eigenvalue and the shifted power method requires many digits of accuracy. The results of the power method (and the shifted power method for the smallest eigenvalues) for a uniform mesh are given in Table 5.2. Similar results can be computed for a quasi-uniform distorted mesh.

h^{-1}	μ_0	μ_1
4	3.15×10^{-1}	2.57×10^1
8	1.21×10^{-1}	6.04×10^1
16	5.49×10^{-2}	1.26×10^2
32	2.74×10^{-2}	2.55×10^2

Table 5.1: Minimum and Maximum Eigenvalues

5.3 Domain Decomposition Preconditioner

We now define the domain decomposition preconditioner. First, we partition the domain into $J = l_0 \times m_0 \times n_0$ coarse-grid elements. Then, we partition the coarse-grid elements into fine-grid non-overlapping subdomains, $\tilde{\Omega}_s$. A $l_s \times m_s \times n_s$ subdomain, Ω_s , is a non-overlapping subdomain extended by δ . The overlap, δ , is a multiple of fine-grid elements and is typically size h . This is illustrated in Figure 5.1.

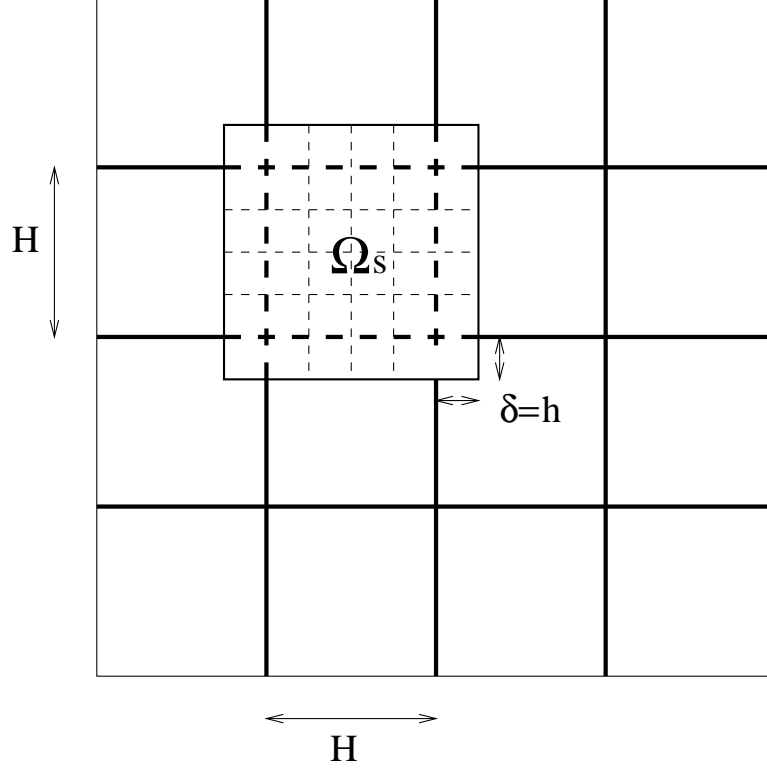


Figure 5.1: Partitioned of Domain into Subdomains

We define a divergence-free subspace, $\mathbf{D}_s \subset \mathbf{D}^h$, on each of the subdomains, with \mathbf{D}_s spanned by:

$$\mathbf{w}_{j_s+1/2, k_s+1/2, i_s}^{x_s} ; i_s = 0, 0 \leq j_s \leq m_s - 2, 0 \leq k_s \leq n_s - 2 \quad (5.27)$$

$$\mathbf{w}_{k_s+1/2, i_s+1/2, j_s}^{y_s} ; 0 \leq i_s \leq l_s - 2, 0 \leq j_s \leq m_s - 1, 0 \leq k_s \leq n_s - 2 \quad (5.28)$$

$$\mathbf{w}_{i_s+1/2, j_s+1/2, k_s}^{z_s} ; 0 \leq i_s \leq l_s - 2, 0 \leq j_s \leq m_s - 2, 0 \leq k_s \leq n_s - 1. \quad (5.29)$$

The basis for \mathbf{D}_s is defined in the same manner as the basis for \mathbf{D}^h . The dimension of \mathbf{D}_s is given by

$$N^s = (l_s - 1)(m_s - 1) + (l_s - 1)m_s(n_s - 1) + (l_s - 1)(m_s - 1)n_s. \quad (5.30)$$

We define a multi-index, $0 \leq K^s \leq N^s - 1$, and a multi-index, $0 \leq K \leq N - 1$, where $N = \dim \mathbf{D}^h$. A divergence-free vector function, $\mathbf{u}^s \in \mathbf{D}_s$, is given by:

$$\mathbf{u}^s = \sum_{K^s=0}^{N^s-1} d_{K^s}^s \mathbf{w}_{K^s} \quad (5.31)$$

We write a subdomain basis vector function, $\mathbf{w}_{K^s}^s$, in terms of the basis for \mathbf{D}^h as follows:

$$\mathbf{w}_{K^s}^s = \sum_{K=0}^{N-1} I_{K,K^s}^s \mathbf{w}_K. \quad (5.32)$$

For each subspace, \mathbf{D}_s , we define a projection operator, P_s , such that for any $\mathbf{u} \in \mathbf{D}^h$ we have $P_s \mathbf{u} \in \mathbf{D}_s$ with

$$a(P_s \mathbf{u}, \mathbf{w}^s) = a(\mathbf{u}, \mathbf{w}^s), \quad \forall \mathbf{w}^s \in \mathbf{D}_s. \quad (5.33)$$

We now derive an explicit matrix representation of the operator P_s . Let $\mathbf{u}^s = P_s \mathbf{u}$ and let $\mathbf{w}^s = \mathbf{w}_{L^s}^s$ in equation (5.33) where L^s is a multi-index like K^s . We introduce another multi-index, L , which is like the multi-index K . Now, we substitute (5.31) and (5.32) into the left-hand side of (5.33) to get

$$\begin{aligned} a(\mathbf{u}^s, \mathbf{w}_{L^s}^s) &= a\left(\sum_{K^s=0}^{N^s-1} d_{K^s}^s \mathbf{w}_{K^s}^s, \mathbf{w}_{L^s}^s\right) \\ &= a\left(\sum_{K^s=0}^{N^s-1} d_{K^s}^s \sum_{K=0}^{N-1} I_{K,K^s}^s \mathbf{w}_K, \sum_{L=0}^{N-1} I_{L,L^s}^s \mathbf{w}_L\right) \\ &= \sum_{L=0}^{N-1} I_{L,L^s}^s \sum_{K=0}^{N-1} a(\mathbf{w}_K, \mathbf{w}_L) \sum_{K^s=0}^{N^s-1} I_{K,K^s}^s d_{K^s}^s. \end{aligned} \quad (5.34)$$

Similarly, if we let \mathbf{u} be given by:

$$\mathbf{u} = \sum_{K=0}^{N-1} d_K \mathbf{w}_K \quad (5.35)$$

and substitute this along with $\mathbf{w}^s = \mathbf{w}_{L^s}^s$ into the right-hand side of (5.33) we get

$$\begin{aligned} a(\mathbf{u}, \mathbf{w}_{L^s}^s) &= a\left(\sum_{K=0}^{N-1} d_K \mathbf{w}_K, \sum_{L=0}^{N-1} I_{L,L^s}^s \mathbf{w}_L\right) \\ &= \sum_{L=0}^{N-1} I_{L,L^s}^s \sum_{K=0}^{N-1} a(\mathbf{w}_K, \mathbf{w}_L) d_K. \end{aligned} \quad (5.36)$$

Doing this for each L^s results in the matrix equation given by:

$$\mathbf{I}_h^s \mathbf{C} \mathbf{M} \mathbf{C}^T \mathbf{I}_s^h \mathbf{d}^s = \mathbf{I}_h^s \mathbf{C} \mathbf{M} \mathbf{C}^T \mathbf{d} \quad (5.37)$$

We define the subdomain stiffness matrix as:

$$\mathbf{C}_s \mathbf{M}_s \mathbf{C}_s^T = \mathbf{I}_h^s \mathbf{C} \mathbf{M} \mathbf{C}^T \mathbf{I}_s^h \quad (5.38)$$

where \mathbf{M}_s is simply the restriction of \mathbf{M} to subdomain Ω_s . Recall that the matrices \mathbf{C} and \mathbf{C}^T are not explicitly assembled. Therefore, the operator which computes $\mathbf{C}^T \mathbf{d}$ is the same one which computes $\mathbf{C}_s^T \mathbf{d}^s$. From now on we leave the subscript s off of these operators to emphasize that they are defined in the context of the type of vector they are operating on. Therefore, the coefficient vector corresponding to the projection of \mathbf{u} onto subspace \mathbf{D}_s is given by:

$$\mathbf{I}_s^h \mathbf{d}^s = \mathbf{I}_s^h (\mathbf{C} \mathbf{M}_s \mathbf{C}^T)^{-1} \mathbf{I}_h^s \mathbf{C} \mathbf{M} \mathbf{C}^T \mathbf{d} = \mathbf{P}_s \mathbf{d} \quad (5.39)$$

where \mathbf{P}_s is our projection matrix. Note that the coefficient vector corresponding to the projection of \mathbf{u} onto \mathbf{D}_s corresponds to a vector in \mathbf{D}^h . This is necessary in order to sum the projections.

Suppose we are trying to solve the equation, $\mathbf{C} \mathbf{M} \mathbf{C}^T \mathbf{d} = \mathbf{b}$, and suppose we have an approximation \mathbf{d}^k . Then, the error is $\mathbf{e}^k = \mathbf{d} - \mathbf{d}^k$ and $\mathbf{C} \mathbf{M} \mathbf{C}^T (\mathbf{d}^k + \mathbf{e}^k) = \mathbf{b}$. From this we get

$$\mathbf{C} \mathbf{M} \mathbf{C}^T \mathbf{e}^k = \mathbf{b} - \mathbf{C} \mathbf{M} \mathbf{C}^T \mathbf{d}^k = \mathbf{r}^k. \quad (5.40)$$

If we knew the error, then we would know the solution. However, computing the error is as difficult as solving the matrix equation. What we do instead is compute the projection of the error onto the subdomains. If we compute $\mathbf{P}_s \mathbf{e}^k$ we would get

$$\mathbf{P}_s \mathbf{e}^k = \mathbf{I}_s^h (\mathbf{C} \mathbf{M}_s \mathbf{C}^T)^{-1} \mathbf{I}_h^s \mathbf{r}^k. \quad (5.41)$$

We sum the matrices on the right-hand side of (5.41) to get our subdomain preconditioner given by:

$$\mathbf{S} \mathbf{D} = \sum_{s=1}^J \mathbf{B}_s \quad (5.42)$$

where

$$\mathbf{B}_s = \mathbf{I}_s^h (\mathbf{C} \mathbf{M}_s \mathbf{C}^T)^{-1} \mathbf{I}_h^s. \quad (5.43)$$

Computing the error on the subdomains is not good enough. Suppose the error, restricted to a subdomain, is a constant. Then, the projection of the error onto a subdomain with homogeneous Dirichlet boundary conditions would be zero. We can only resolve this smooth type of error on subdomains which coincide with the boundary of Ω . This situation is sometimes referred to as the kernel of a vector when restricted to a subdomain. We need an efficient way of approximating the error on the whole domain. This is done through the use of a coarse grid.

We define the basis for the coarse grid divergence-free subspace, \mathbf{D}^H , in the usual way given by:

$$\mathbf{w}_{j_0+1/2, k_0+1/2, i_0}^{xH} ; i_0 = 0, 0 \leq j_0 \leq m_0 - 2, 0 \leq k_0 \leq n_0 - 2 \quad (5.44)$$

$$\mathbf{w}_{k_0+1/2, i_0+1/2, j_0}^{yH} ; 0 \leq i_0 \leq l_0 - 2, 0 \leq j_0 \leq m_0 - 1, 0 \leq k_0 \leq n_0 - 2 \quad (5.45)$$

$$\mathbf{w}_{i_0+1/2, j_0+1/2, k_0}^{zH} ; 0 \leq i_0 \leq l_0 - 2, 0 \leq j_0 \leq m_0 - 2, 0 \leq k_0 \leq n_0 - 1 \quad (5.46)$$

The coarse grid preconditioner is given by:

$$\mathbf{B}^H = \mathbf{I}_H^h (\mathbf{C}\mathbf{M}^H\mathbf{C}^T)^{-1}\mathbf{I}_h^H \quad (5.47)$$

where \mathbf{M}^H is assembled from the basis of \mathbf{D}^H . Our two-level preconditioner is thus defined as:

$$\mathbf{B} = \mathbf{B}^H + \mathbf{S}\mathbf{D} \quad (5.48)$$

5.3.1 Convergence Analysis

We now show that the preconditioned matrix, $\mathbf{B}\mathbf{C}\mathbf{M}\mathbf{C}^T$, has a condition number bounded by a constant times $(1 + H/\delta)^2$. For this section, we will use a generic positive constant C which is independent of mesh size and the number of subdomains.

To estimate the minimum eigenvalue of $\mathbf{B}\mathbf{C}\mathbf{M}\mathbf{C}^T$ we will need the following:

Lemma 5.5 For any $\mathbf{v} \in \mathbf{D}^h$, there exists a decomposition of the form

$$\mathbf{v} = \mathbf{v}_0 + \mathbf{v}_1 + \cdots + \mathbf{v}_J \text{ with } \mathbf{v}_s \in \mathbf{D}_s, \mathbf{v}_0 \in \mathbf{D}^H \quad (5.49)$$

and

$$\sum_{s=0}^J a(\mathbf{v}_s, \mathbf{v}_s) \leq C_0^2 a(\mathbf{v}, \mathbf{v}), \quad (5.50)$$

where C_0^2 is bounded independently of the mesh size h and number of subdomains J , and grows quadratically as $(1 + H/\delta)^2$.

Proof. For any $\mathbf{v} \in \mathbf{D}^h$, Girault and Raviart in [36] demonstrated the existence of a vector potential $\Phi \in H^1(\Omega)^3$ such that $\mathbf{v} = \mathbf{curl} \Phi$,

$$\|\Phi\|_{L^2(\Omega)^3} \leq C \|\mathbf{curl} \Phi\|_{L^2(\Omega)^3}, \quad (5.51)$$

and

$$\|\nabla\Phi\|_{L^2(\Omega)^3} \leq C\|\mathbf{curl}\ \Phi\|_{L^2(\Omega)^3}. \quad (5.52)$$

Let \mathbf{U}^H be the discrete vector potential space associated with the coarse grid partition Ω^H . Let \mathbf{Q}^H be the L^2 projection onto \mathbf{U}^H . Then, it can be shown (see [9]) that

$$\|\Phi - \mathbf{Q}^H\Phi\|_{L^2(\Omega)^3} \leq CH\|\nabla\Phi\|_{L^2(\Omega)^3}, \quad (5.53)$$

and

$$\|\mathbf{curl}\ (\mathbf{Q}^H\Phi)\|_{L^2(\Omega)^3} \leq C\|\nabla\Phi\|_{L^2(\Omega)^3}. \quad (5.54)$$

Let $\theta_s \in C_0^\infty(\Omega_s)$, $s = 1, \dots, J$, be a partition of unity such that

$$\|\nabla\theta_s\|_{L^\infty}^2 \leq C/\delta^2. \quad (5.55)$$

We then define

$$\mathbf{v}_0 = \mathbf{curl}\ \mathbf{Q}^H\Phi \in \mathbf{D}^H, \quad (5.56)$$

$$\Psi = \Phi - \mathbf{Q}^H\Phi, \quad (5.57)$$

and

$$\mathbf{v}_s = \Pi_h \mathbf{curl}\ (\theta_s\Psi) \in \mathbf{D}_s. \quad (5.58)$$

Note that $\mathbf{curl}\ \Psi = \mathbf{v} - \mathbf{v}_0$.

Since

$$\mathbf{curl}\ (\theta_s\Psi) = \Psi \times \nabla\theta_s + \theta_s\mathbf{curl}\ \Psi.$$

we can estimate the norm of \mathbf{v}_s over Ω as:

$$\|\mathbf{v}_s\|_{L^2(\Omega)^3}^2 = \|\mathbf{v}_s\|_{L^2(\Omega_s)^3}^2$$

$$\begin{aligned}
&\leq C \|\mathbf{curl}(\theta_s \Psi)\|_{L^2(Q)^3}^2 \\
&= C \|\Psi \times \nabla \theta_s + \theta_s \mathbf{curl} \Psi\|_{L^2(\Omega_s)^3}^2 \\
&\leq C \int_{\Omega_s} (|\Psi|^2 |\nabla \theta_s|^2 + \theta_s^2 |\mathbf{curl} \Psi|^2) dx dy dz \\
&\leq C \delta^{-2} \|\Psi\|_{L^2(\Omega_s)^3}^2 + C \|\mathbf{curl} \Psi\|_{L^2(\Omega_s)^3}^2.
\end{aligned} \tag{5.59}$$

For $s = 0$, note that

$$\begin{aligned}
\|\mathbf{v}_0\|_{L^2(\Omega)^3} &= \|\mathbf{curl}(\mathbf{Q}^H \Phi)\|_{L^2(\Omega)^3} \\
&\leq C \|\nabla \Phi\|_{L^2(\Omega)^3} \\
&\leq C \|\mathbf{curl} \Phi\|_{L^2(\Omega)^3} \\
&= C \|\mathbf{v}\|_{L^2(\Omega)^3},
\end{aligned} \tag{5.60}$$

hence

$$\begin{aligned}
\|\mathbf{curl} \Psi\|_{L^2(\Omega)^3} &= \|\mathbf{v} - \mathbf{v}_0\|_{L^2(\Omega)^3} \\
&\leq \|\mathbf{v}\|_{L^2(\Omega)^3} + \|\mathbf{v}_0\|_{L^2(\Omega)^3} \\
&\leq C \|\mathbf{v}\|_{L^2(\Omega)^3}.
\end{aligned} \tag{5.61}$$

Using (5.52), (5.53), (5.59), (5.60), (5.61), and a bound on the number of subdomain overlaps we get

$$\begin{aligned}
\sum_{s=0}^J \|\mathbf{v}_s\|_{L^2(\Omega)}^2 &\leq C \|\mathbf{v}\|_{L^2(\Omega)}^2 + C \delta^{-2} \|\Psi\|_{L^2(\Omega)}^2 + C \|\mathbf{curl} \Psi\|_{L^2(\Omega)}^2 \\
&\leq C \|\mathbf{v}\|_{L^2(\Omega)}^2 + C \delta^{-2} H^2 \|\nabla \Phi\|_{L^2(\Omega)}^2 \\
&\leq C \|\mathbf{v}\|_{L^2(\Omega)}^2 + C \frac{H^2}{\delta} \|\mathbf{curl} \Phi\|_{L^2(\Omega)}^2 \\
&\leq C \left(1 + \frac{H}{\delta}\right)^2 \|\mathbf{v}\|_{L^2(\Omega)}^2
\end{aligned}$$

■

Now we can estimate the condition number for \mathbf{BCMC}^T .

Theorem 5.6 The condition number of \mathbf{BCMC}^T is independent of h and the number of subdomains J and grows quadratically as $(1 + H/\delta)^2$.

Proof. The proof relies on some well known lemmas. For example, in [55, Lemma 1, pg. 154] it is shown that

$$a(\mathbf{B}^{-1}\mathbf{v}, \mathbf{v}) \leq \sum_{s=0}^J a(\mathbf{v}_s, \mathbf{v}_s) \quad (5.62)$$

where

$$\mathbf{v} = \sum_{s=0}^J \mathbf{v}_s. \quad (5.63)$$

Therefore, a direct result of Lemma 5.5 is a lower bound on the eigenvalues of \mathbf{BCMC}^T given by

$$\frac{C}{\left(1 + \frac{H}{\delta}\right)^2} \leq a(\mathbf{B}\mathbf{v}, \mathbf{v}). \quad (5.64)$$

Since the subdomains are nearly orthogonal (the number of overlaps is constant) we have an upper bound on the eigenvalues of \mathbf{BCMC}^T given by

$$a(\mathbf{B}\mathbf{v}, \mathbf{v}) \leq Ca(\mathbf{v}, \mathbf{v}) \quad (5.65)$$

where the positive constant C is independent of h and the number of subdomains. The constant may depend on the approximation properties of the subdomain solves. We are assuming exact solves on the subdomains and the coarse-grid. Again, this follows from standard arguments which can be found in [55, Chapter 5]. We conclude that the condition number of \mathbf{BCMC}^T is bounded by $C(1+H/\delta)^2$. ■

Remark 5.3 Domain decomposition algorithms with small overlap often have a bound on the condition number which grows only linearly with $1 + H/\delta$. This can be shown using a special Poincaré type lemma. Let $\Gamma = \bigcup \partial\tilde{\Omega}_s \setminus \partial\Omega$ and let $\Gamma_{\delta,s} \subset \tilde{\Omega}_s$ be the set of points which are within a distance δ of Γ . What this means is that we are considering a non-overlapping region, $\tilde{\Omega}_s$, and a strip of width δ inside $\tilde{\Omega}_s$ going around the boundary. Let Ψ be an arbitrary element of $H^1(\tilde{\Omega}_s)^3$. Then Dryja and Widlund showed in [27, Lemma 3.1] that

$$\|\Psi\|_{L^2(\Gamma_{\delta,s})^3}^2 \leq C\delta^2 \left((1 + H/\delta)\|\nabla\Psi\|_{L^2(\tilde{\Omega}_s)^3}^2 + 1/(H\delta)\|\Psi\|_{L^2(\tilde{\Omega}_s)^3}^2 \right). \quad (5.66)$$

Unfortunately, we cannot bound the gradient by the **curl** in 3-D. However, in two dimensions we would define the **curl** of a scalar stream function, p , as:

$$\mathbf{curl} p = \begin{pmatrix} \frac{\partial p}{\partial y} \\ -\frac{\partial p}{\partial x} \end{pmatrix}. \quad (5.67)$$

Now it is trivial to show that

$$\|\nabla p\|_{L^2(\tilde{\Omega}_s)^2}^2 = \|\mathbf{curl} p\|_{L^2(\tilde{\Omega}_s)^2}^2. \quad (5.68)$$

One can then use the analysis described in [55, Chapter 5] to get a bound on the condition number which only grows linearly as $1 + H/\delta$. It is not clear if we can get such a bound in 3-D. In other words, is Theorem 5.6 a tight bound on the condition number of \mathbf{BCMC}^T ?

5.4 Preconditioner for Bordered Matrix

If we have disconnected Dirichlet boundaries, then we will need to add a divergence-free pipe function to our divergence-free basis. This pipe function has

global support. Therefore, it will be represented as a dense column and row in the stiffness matrix. This type of matrix is called a bordered matrix and it does not lend itself to block Jacobian iterative methods such as domain decomposition.

The divergence-free subspace, \mathbf{D}^h , is spanned by basis vector functions with local support plus the pipe function. A divergence-free vector function, $\tilde{\mathbf{v}}_D^h \in \mathbf{D}^h$, is given by:

$$\tilde{\mathbf{v}}_D^h = \mathbf{v}_D^h + c * \mathbf{w}_G \quad (5.69)$$

where \mathbf{v}_D^h is a linear combination of divergence-free vector functions with local support, c is a constant, and \mathbf{w}_G is the pipe function. We let the test function, $\tilde{\mathbf{w}}$, be given by:

$$\tilde{\mathbf{w}} = \mathbf{w} + \mathbf{w}_G \quad (5.70)$$

where \mathbf{w} is a linear combination of divergence-free basis vector functions with local support. We then substitute (5.69) and (5.70) into (4.4) to get

$$a(\mathbf{v}_D^h + c * \mathbf{v}_G, \mathbf{w} + \mathbf{w}_G) = -a(\mathbf{v}_I^h + \mathbf{v}_g^h, \mathbf{w} + \mathbf{w}_G) - G(\mathbf{w} + \mathbf{w}_G). \quad (5.71)$$

This results in a bordered matrix equation given by:

$$\begin{bmatrix} \mathbf{C}\mathbf{M}\mathbf{C}^T & \mathbf{W}_1 \\ \mathbf{W}_1^T & W_2 \end{bmatrix} \begin{bmatrix} \mathbf{d} \\ c \end{bmatrix} = \begin{bmatrix} -\mathbf{C}\mathbf{M}\mathbf{u} + \mathbf{C}2\mathbf{q} \\ -\mathbf{u}_G^T\mathbf{M}\mathbf{u} + \mathbf{u}_g^T\mathbf{q} \end{bmatrix} \quad (5.72)$$

where \mathbf{d} are the coefficients corresponding to \mathbf{v}_D^h , \mathbf{u} are the coefficients corresponding to $\mathbf{v}_I^h + \mathbf{v}_g^h$, \mathbf{u}_G are the fluxes corresponding to the pipe function, and \mathbf{q} is the vector of specified pressures on the Dirichlet boundary. The \mathbf{W}_1 vector is given by:

$$\mathbf{W}_1 = \mathbf{C}\mathbf{M}\mathbf{w}_G \quad (5.73)$$

The scalar, W_2 , is given by:

$$W_2 = (\mathbf{w}_G)^T \mathbf{M} \mathbf{w}_G \quad (5.74)$$

We will denote the bordered matrix as \mathbf{CMC}_B^T . We would like to design a preconditioner for the bordered matrix which takes advantage of the domain decomposition preconditioner we implement for the \mathbf{CMC}^T matrix. One approach is to use the Schur complement method. The inverse of \mathbf{CMC}_B^T can be written as:

$$(\mathbf{CMC}_B^T)^{-1} = \begin{bmatrix} \mathbf{I} & -(\mathbf{CMC}^T)^{-1} \mathbf{W}_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} (\mathbf{CMC}^T)^{-1} & \mathbf{0} \\ 0 & S^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{W}_1^T (\mathbf{CMC}^T)^{-1} & 1 \end{bmatrix} \quad (5.75)$$

where the scalar, S , is given by:

$$S = W_2 - \mathbf{W}_1^T (\mathbf{CMC}^T)^{-1} \mathbf{W}_1 \quad (5.76)$$

We then define the preconditioner, \mathbf{B}_B , as:

$$\mathbf{B}_B = \begin{bmatrix} \mathbf{I} & -\mathbf{B} \mathbf{W}_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{B} & \mathbf{0} \\ 0 & B_S \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{W}_1^T \mathbf{B} & \mathbf{I} \end{bmatrix}. \quad (5.77)$$

The matrix, \mathbf{B} , is the domain decomposition preconditioner and B_S is an approximate inverse of S . To get an exact inverse of S we would have to invert \mathbf{CMC}^T . Instead, one might define B_S as being given by:

$$B_S = (W_2 - \mathbf{W}_1^T \mathbf{B} \mathbf{W}_1)^{-1}. \quad (5.78)$$

However, numerical tests showed that B_S was ill-conditioned. In some cases we would get a negative value for B_S . One could try different approximate inverses

for S . For example, the inverse of the diagonal of \mathbf{CMC}^T or perhaps a few iterations of the PCG method. However, we decided to take a different approach for implementing the bordered matrix preconditioner.

The key to our method will be to choose a specific pipe function which will make our preconditioned system well conditioned. There are many pipe functions we can choose. The pipe function shown in Figure 4.6 is just one example. If our grid is orthogonal and the hydraulic conductivities homogeneous and isotropic, then this pipe function will be $a(\cdot, \cdot)$ -orthogonal to all the divergence-free vector functions with local support. That means that \mathbf{W}_1 will be zero and we can use the preconditioner given by:

$$\mathbf{B}_B = \begin{bmatrix} \mathbf{B} & \mathbf{0} \\ 0 & W_2^{-1} \end{bmatrix} \quad (5.79)$$

If the grid is not orthogonal or the hydraulic conductivities are heterogeneous and possibly anisotropic, then we will need to compute a different pipe function which is approximately $a(\cdot, \cdot)$ -orthogonal to the divergence-free subspace consisting of vector functions with local support. Suppose we have a pipe function, for example, like the one shown in Figure 4.6. Then, we add a locally divergence-free vector function, \mathbf{w}_L , giving us a new divergence-free pipe function, $\tilde{\mathbf{w}}_G$, given by:

$$\tilde{\mathbf{w}}_G = \mathbf{w}_G + \mathbf{w}_L \quad (5.80)$$

where \mathbf{w}_L satisfies

$$a(\mathbf{w}_L, \mathbf{w}) = -a(\mathbf{w}_G, \mathbf{w}), \forall \mathbf{w} \in (\mathbf{D}^h \setminus c * \mathbf{w}_G). \quad (5.81)$$

The space, $(\mathbf{D}^h \setminus c * \mathbf{w}_G)$, is spanned by divergence-free basis vector functions in \mathbf{D}^h minus the pipe functions. Our pipe function, $\tilde{\mathbf{w}}_G$, is now $a(\cdot, \cdot)$ -orthogonal

to the locally divergence-free subspace and we can use the preconditioner given by (5.79). To compute the projection in (5.81) we need to invert the \mathbf{CMC}^T matrix. However, we can approximate the projection just enough to make the preconditioned system well conditioned. Numerical tests suggest that we need to reduce the preconditioned residual of the projection until it is about order 1.

5.4.1 Analysis of Bordered Matrix Preconditioner

We now analyze the effects of approximating (5.81) for the bordered matrix preconditioner. To shorten the notation, we denote the stiffness matrix and bordered stiffness matrix as:

$$\mathbf{A} = \mathbf{CMC}^T \quad (5.82)$$

$$\mathbf{A}_B = \mathbf{CMC}_B^T = \begin{bmatrix} \mathbf{A} & \mathbf{W}_1 \\ \mathbf{W}_1^T & W_1 \end{bmatrix}. \quad (5.83)$$

It can be shown that if $\tilde{\mu}_0$ and $\tilde{\mu}_1$ are two real positive numbers such that,

$$\tilde{\mu}_0 \leq \frac{\tilde{\mathbf{d}}^T \mathbf{A}_B \mathbf{B}_B \mathbf{A}_B \tilde{\mathbf{d}}}{\tilde{\mathbf{d}}^T \mathbf{A}_B \tilde{\mathbf{d}}} \leq \tilde{\mu}_1 \quad (5.84)$$

where

$$\tilde{\mathbf{d}} = \begin{bmatrix} \mathbf{d} \\ q \end{bmatrix} \quad (5.85)$$

then, the condition number of $\mathbf{B}_B \mathbf{A}_B$ is bounded by $\tilde{\mu}_1/\tilde{\mu}_0$.

Let \mathbf{w}_L^k be the approximation to (5.81). Then,

$$\mathbf{W}_1 = \mathbf{CM}(\mathbf{C}^T \mathbf{w}_L^k + \mathbf{w}_G) = \mathbf{r}_k \quad (5.86)$$

where \mathbf{r}_k is the residual of (5.81) after the k -th iteration of the PCG method. Given $\epsilon > 0$ we reduce the residual until $\mathbf{r}_k^T \mathbf{B} \mathbf{r}_k \approx \epsilon^2$. We think of \mathbf{B} as an

approximate inverse of \mathbf{A} which has eigenvalues bounded by a constant times h and a constant times h^{-1} . Therefore, we bound the Euclidean norm of \mathbf{r}_k in terms of ϵ as follows:

$$hc_0|\mathbf{r}_k|^2 \leq \epsilon^2 \leq h^{-1}c_1|\mathbf{r}_k|^2. \quad (5.87)$$

We will need a bound on the scalar W_2 . This is given by

Lemma 5.7 Let \mathbf{w}_G be the vector of coefficients (fluxes) corresponding to the pipe function which represents uniform flow from one Dirichlet boundary to another Dirichlet boundary on the opposite end of the domain. After approximating the $a(\cdot, \cdot)$ -orthogonal projection of the pipe function onto the space of locally divergence-free vector functions we obtain a vector of coefficients, \mathbf{w}_L^k , such that

$$W_2 = (\mathbf{C}^T \mathbf{w}_L^k + \mathbf{w}_G)^T \mathbf{M} (\mathbf{C}^T \mathbf{w}_L^k + \mathbf{w}_G), \quad (5.88)$$

and

$$W_2 \geq c_2 h^{-4}. \quad (5.89)$$

Proof. The scalar, W_2 , is given by

$$\tilde{\mathbf{w}}_G^T \mathbf{M} \tilde{\mathbf{w}}_G \quad (5.90)$$

where

$$\tilde{\mathbf{w}}_G = \mathbf{C}^T \mathbf{w}_L^k + \mathbf{w}_G. \quad (5.91)$$

Because \mathbf{M} comes from the $a(\cdot, \cdot)$ form (continuous and coercive), and its entries are $O(h^{-1})$, we have constants c_* and c^* such that

$$c_* h^{-1} \leq \frac{\tilde{\mathbf{w}}_G^T \mathbf{M} \tilde{\mathbf{w}}_G}{\tilde{\mathbf{w}}_G^T \tilde{\mathbf{w}}_G} \leq c^* h^{-1}. \quad (5.92)$$

We scale the pipe function \mathbf{w}_G in Figure 4.6 to have unit fluxes on faces same as the locally supported basis functions. Then the computed pipe function, $\tilde{\mathbf{w}}_G$, will be scaled such that the total flux across each plane perpendicular to the direction of flow (parallel to the Dirichlet faces) is $O(h^{-2})$. The scalar, $\tilde{\mathbf{v}}_G^T \tilde{\mathbf{v}}_G$, will be minimized if all the entries have the same magnitude of order 1. Therefore, after summing over all faces, $\tilde{\mathbf{w}}_G^T \tilde{\mathbf{v}}_G = O(h^{-3})$. Using (5.92) we have the desired result. ■

Concerning the condition number of the preconditioned bordered matrix we have the following:

Theorem 5.8 Let $\hat{\mu}_0$ and $\hat{\mu}_1$ be bounds on the minimum and maximum eigenvalues, respectively, of the preconditioned matrix, \mathbf{BA} . Then, there exist positive constants, $\tilde{\mu}_0(\epsilon)$ and $\tilde{\mu}_1(\epsilon)$, such that

$$\tilde{\mu}_0(\epsilon) \leq \frac{\tilde{\mathbf{d}}^T \mathbf{A}_B \mathbf{B}_B \mathbf{A}_B \tilde{\mathbf{d}}}{\tilde{\mathbf{d}}^T \mathbf{A}_B \tilde{\mathbf{d}}} \leq \tilde{\mu}_1(\epsilon) \quad (5.93)$$

where

$$\tilde{\mu}_0(\epsilon) \rightarrow \min\{\hat{\mu}_0, 1\} \quad (5.94)$$

and

$$\tilde{\mu}_1(\epsilon) \rightarrow \max\{\hat{\mu}_1, 1\} \quad (5.95)$$

as $\epsilon \rightarrow 0$.

Proof. The ratio of inner products in (5.93) is given by

$$\frac{\mathbf{d}^T \mathbf{A} \mathbf{B} \mathbf{A} \mathbf{d} + W_2^{-1} \mathbf{d}^T (\mathbf{r}_k \mathbf{r}_k^T) \mathbf{d} + 2c \mathbf{d}^T (\mathbf{A} \mathbf{B} + \mathbf{I}) \mathbf{r}_k + c^2 (\mathbf{r}_k^T \mathbf{B} \mathbf{r}_k + W_2)}{\mathbf{d}^T \mathbf{A} \mathbf{d} + 2c \mathbf{d}^T \mathbf{r}_k + c^2 W_2}. \quad (5.96)$$

We bound each of the positive terms in the numerator of (5.96) by a corresponding term in the denominator as follows:

$$\hat{\mu}_0 \mathbf{d}^T \mathbf{A} \mathbf{d} \leq \mathbf{d}^T \mathbf{A} \mathbf{B} \mathbf{A} \mathbf{d} \leq \hat{\mu}_1 \mathbf{d}^T \mathbf{A} \mathbf{d} \quad (5.97)$$

$$0 \leq W_2^{-1} \mathbf{d}^T (\mathbf{r}_k \mathbf{r}_k^T) \mathbf{d} \leq c_3 h^2 \epsilon^2 \mathbf{d}^T \mathbf{A} \mathbf{d} \quad (5.98)$$

$$(c_4 \epsilon^2 h^4 + 1) c^2 W_2 \leq c^2 (\mathbf{r}_k^T \mathbf{B} \mathbf{r}_k + W_2) \leq (c_5 \epsilon^2 h^4 + 1) c^2 W_2. \quad (5.99)$$

The bound (5.97) is a result of the domain decomposition analysis where $\hat{\mu}_1$ is bounded by a constant and $\hat{\mu}_0$ is bounded by a constant times $1/(1 + H/\delta)^2$. We use (5.89), (5.87), and an upper bound on the eigenvalue of \mathbf{A} to get (5.98). Relationship (5.99) is a direct result of the preconditioned residual and the bound on W_2 .

We now have

$$\tilde{\mu}_0 \frac{\left(1 + \frac{2c\mathbf{d}^T(\mathbf{A}\mathbf{B}+\mathbf{I})\mathbf{r}_k}{\mathbf{d}^T\mathbf{A}\mathbf{d}+c^2W_2}\right)}{\left(1 + \frac{2c\mathbf{d}^T\mathbf{r}_k}{\mathbf{d}^T\mathbf{A}\mathbf{d}+c^2W_2}\right)} \leq \frac{\tilde{\mathbf{d}}^T \mathbf{A}_B \mathbf{B}_B \mathbf{A}_B \tilde{\mathbf{d}}}{\tilde{\mathbf{d}}^T \mathbf{A}_B \tilde{\mathbf{d}}} \leq \tilde{\mu}_1 \frac{\left(1 + \frac{2c\mathbf{d}^T(\mathbf{A}\mathbf{B}+\mathbf{I})\mathbf{r}_k}{\mathbf{d}^T\mathbf{A}\mathbf{d}+c^2W_2}\right)}{\left(1 + \frac{2c\mathbf{d}^T\mathbf{r}_k}{\mathbf{d}^T\mathbf{A}\mathbf{d}+c^2W_2}\right)} \quad (5.100)$$

where

$$\tilde{\mu}_0 = \min\{\hat{\mu}_0, (c_4 \epsilon^2 h^4 + 1)\} \quad (5.101)$$

and

$$\tilde{\mu}_1 = \max\{\hat{\mu}_1 + c_3 \epsilon^2 h^2, (c_5 \epsilon^2 h^4 + 1)\}. \quad (5.102)$$

We want to show that the ratio

$$\frac{2c\mathbf{d}^T \mathbf{r}_k}{\mathbf{d}^T \mathbf{A} \mathbf{d} + c^2 W_2} \quad (5.103)$$

gets small as ϵ gets small. Since the matrix \mathbf{A} is symmetric positive definite we have

$$2c\mathbf{d}^T \mathbf{r}_k = 2(\mathbf{A}^{1/2} \mathbf{d})^T (c\mathbf{A}^{-1/2} \mathbf{r}_k) \quad (5.104)$$

and for any positive constant γ we get

$$|2c\mathbf{d}^T \mathbf{r}_k| \leq \gamma \mathbf{d}^T \mathbf{A} \mathbf{d} + \frac{1}{\gamma} c^2 \mathbf{r}_k^T \mathbf{A}^{-1} \mathbf{r}_k. \quad (5.105)$$

By using (5.87) and Lemma 5.7 along with $\mathbf{r}_k^T \mathbf{A}^{-1} \mathbf{r}_k \leq (h^{-1}/\mu_0) \mathbf{r}_k^T \mathbf{r}_k$ we get

$$\mathbf{r}_k^T \mathbf{A}^{-1} \mathbf{r}_k \leq \frac{h^2 \epsilon^2}{\mu_0 c_0 c_2} W_2. \quad (5.106)$$

If we choose $\gamma = \epsilon$, then we have

$$|2c\mathbf{d}^T \mathbf{r}_k| \leq \epsilon \mathbf{d}^T \mathbf{A} \mathbf{d} + \frac{h^2 \epsilon}{\mu_0 c_0 c_2} c^2 W_2 \quad (5.107)$$

which goes to zero as ϵ goes to zero. A similar argument can be made for the ratio

$$\frac{2c\mathbf{d}^T (\mathbf{A} \mathbf{B} + \mathbf{I}) \mathbf{r}_k}{\mathbf{d}^T \mathbf{A} \mathbf{d} + c^2 W_2}. \quad (5.108)$$

■

An interesting special case is when we have zero source/sink terms and constant pressure on the two Dirichlet boundaries. In this situation the vector \mathbf{d} is identically all zeros and the condition number of the preconditioned border matrix is bounded by

$$\kappa(\mathbf{B}_B \mathbf{A}_B) \leq \frac{c_5 \epsilon^2 h^4 + 1}{c_4 \epsilon^2 h^4 + 1} \quad (5.109)$$

which, for a fixed h , goes to 1 as ϵ goes to zero. This behavior is confirmed by numerical results.

6. Numerical Results

In this chapter we examine how the PCG method performs using domain decomposition as the preconditioner. We consider test cases I through V. For each case we observe the convergence rate for varying grid sizes. Case I is the Laplacian case where the hydraulic conductivity tensor is constant. An iteration count independent of grid size is observed. Cases II through V involve variable hydraulic conductivity coefficients. In most cases the iteration count does not increase in the presence of large jumps in the coefficients. The average reduction rate of the PCG method using the additive Schwarz preconditioner appears to be around 0.5. This is in agreement with [10]. A multiplicative Schwarz method may give an average reduction rate of about half of that for the additive method. The main observation remains that the iteration count is independent of grid-size and number of subdomains. However, in the presence of anisotropy the iteration count can increase significantly with respect to the grid size.

We assume an exact solve on the coarse-grid and subdomains within a tolerance equal to that of the outer iteration. An iterative method is used for the coarse-grid projection when the number of coarse-grid elements reaches $16 \times 16 \times 16$. This gives us a larger cost per iteration and the cost per iteration increases for heterogeneous coefficients because it becomes more difficult to solve the coarse-grid problem. We used the PCG method with a block-diagonal preconditioner on the coarse-grid problems of size $16 \times 16 \times 16$. The block-diagonal preconditioner is essentially domain decomposition without the coarse-grid.

We also perform two tests using a $128 \times 128 \times 128$ fine-grid. In this case, the coarse-grid is $32 \times 32 \times 32$. Again, we use block-diagonal preconditioning on the coarse-grid. However, it is no longer feasible to use direct solves on the subdomains due to memory restraints. At present, the only other choice is diagonal preconditioning. To put things into perspective, we note that for a $32 \times 32 \times 32$ coarse-grid we have 32768 subdomains. Each subdomain problem is small and it may require only one second to converge. However, with 32768 subdomains the total time for the subdomain solves will be about 10 minutes. If we require 30 outer iterations to converge, then the required CPU time will be about 5 hours. We will see that with a homogeneous conductivity coefficient the required CPU time on a problem of size $128 \times 128 \times 128$ is about 3.24 hours. With a low-permeability block in the center of the domain the CPU time is about 5.45 hours. Adding more heterogeneity to the problem will require CPU times measured in increments of days. It will be important to research how this problem scales to larger architectures with more memory and multiple processors.

The assembly time (in seconds) for the preconditioner is shown in Table 6.1 for different grid sizes. We can see that it takes zero time to assemble the coarse-grid problem. This means that the assembly time should be highly parallelizable since we can assemble each subdomain problem independently.

Fine-grid Size	Coarse-grid Size	Coarse-grid Assembly Time	Fine-grid Assembly Time
$16 \times 16 \times 16$	$4 \times 4 \times 4$	0	1
$32 \times 32 \times 32$	$8 \times 8 \times 8$	0	14
$64 \times 64 \times 64$	$16 \times 16 \times 16$	0	159

Table 6.1: Preconditioner Assembly Time

We also look at two model problems. The first model problem is a vertical cylinder of low hydraulic conductivity in the center of the domain. In this experiment we show how the preconditioner can be parallelized. The cylinder problem is an ideal problem for parallelization because we can partition the domain into horizontal slices each having an identical hydraulic conductivity coefficient. In this way we get perfect load balancing. In other words, each processor is working on an identical set of problems. Therefore, the work should be equal for each processor.

The other model problem models a tank experiment described in [32]. In this experiment, a pressure is imposed at two ends of the tank and homogeneous Neumann (no-flow) conditions on the other sides. Since we have disconnected Dirichlet boundaries we will have the bordered matrix problem. The first step is to compute, approximately, the $a(\cdot, \cdot)$ -orthogonal projection of the pipe function onto the locally divergence-free subspace. We show how the approximation properties of the pipe function projection effects the convergence of the bordered matrix problem. We also show the effects of introducing a source/sink term into the problem.

6.1 PCG Performance

For each test case we reduce the preconditioned residual by ten orders of magnitude. This allows us to examine the asymptotic behavior of the reduction. We also use a minimal overlap of one element for all the experiments. The tests were performed on an Alpha ev6 CPU at 500MHz [49]. All CPU times are given in seconds.

6.1.1 Test Case I

Test case I uses orthogonal grids of varying size with a constant hydraulic conductivity coefficient. The results in Table 6.2 show that the number of iterations is independent of the grid size. The convergence history is plotted in Figure 6.1.

Fine-grid Size	Coarse-grid Size	Iterations	Average Reduction Rate	Solver Time
$16 \times 16 \times 16$	$4 \times 4 \times 4$	31	0.47	1
$32 \times 32 \times 32$	$8 \times 8 \times 8$	31	0.47	14
$64 \times 64 \times 64$	$16 \times 16 \times 16$	32	0.48	152
$128 \times 128 \times 128$	$32 \times 32 \times 32$	31	0.47	11677

Table 6.2: PCG Method for Test Case I

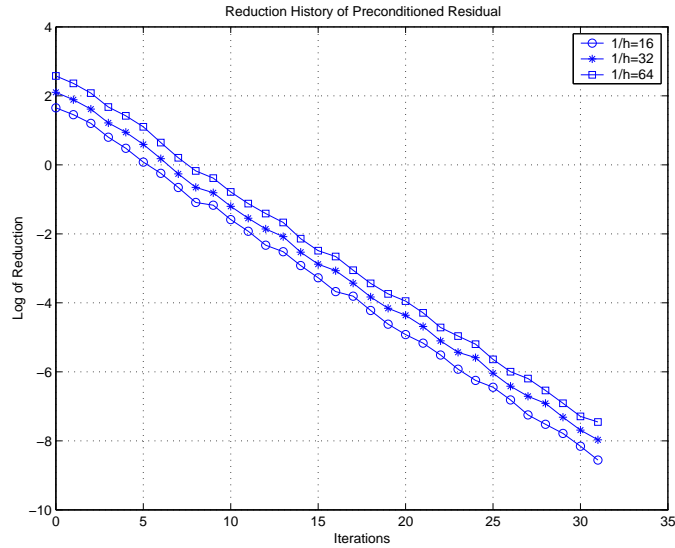


Figure 6.1: Reduction Histories for Test Case I

6.1.2 Test Case II

The next set of tests examine how a low hydraulic conductivity block of volume $1/8$ placed in the middle of the domain effects the convergence rate of the PCG method. The hydraulic conductivity is 10^{-5} inside the block and 1 outside. This is illustrated in Figure 6.2. We see from Table 6.3 that the low conductivity block does not effect the iteration count. When we compare the CPU times between the last row of Table 6.2 and the last row of Table 6.3 we see that the later is significantly larger. This is due to the iterative solver on the large number of subdomains having to do more work in the presence of a heterogeneous coefficient. We will not look at the performance of the PCG method on the $128 \times 128 \times 128$ grid for the remainder of the test cases since the CPU times would have to be measured in terms of days as the coefficient becomes more heterogeneous.

Fine-grid Size	Coarse-grid Size	Iterations	Average Reduction Rate	Solver Time
$16 \times 16 \times 16$	$4 \times 4 \times 4$	31	0.47	1
$32 \times 32 \times 32$	$8 \times 8 \times 8$	31	0.47	14
$64 \times 64 \times 64$	$16 \times 16 \times 16$	31	0.48	153
$128 \times 128 \times 128$	$32 \times 32 \times 32$	31	0.48	19638

Table 6.3: PCG Method for Test Case II

6.1.3 Test Case III

For test case III we use a random distribution of low-conductivity blocks. The low-conductivity blocks are aligned with the coarse-grid. Again, we use a hydraulic conductivity of 10^{-5} for the low-conductivity blocks and 1 otherwise. A

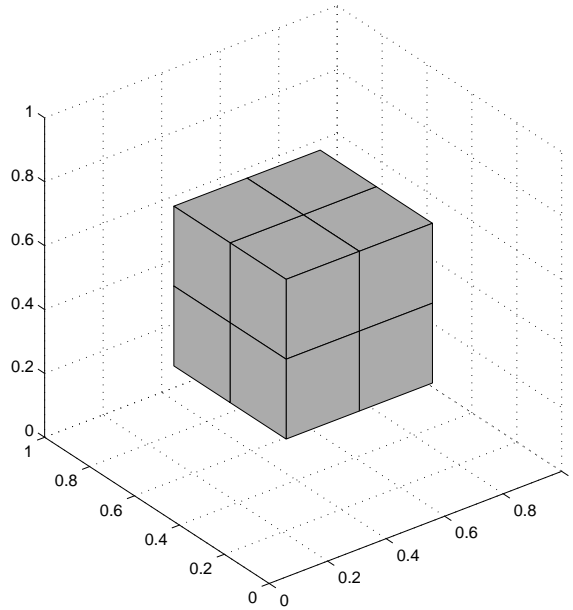


Figure 6.2: Low Conductivity Block

random distribution of low-conductivity blocks is shown in Figures 6.3 and 6.4 on a $4 \times 4 \times 4$ coarse-grid and a $8 \times 8 \times 8$ coarse-grid respectively. Table 6.4 shows a slight increase in the iteration count when we increase the number of coarse-grid elements and thus increase the number of randomly distributed low-conductivity blocks.

Fine-grid Size	Coarse-grid Size	Iterations	Average Reduction Rate	Solver Time
$16 \times 16 \times 16$	$4 \times 4 \times 4$	32	0.48	1
$32 \times 32 \times 32$	$8 \times 8 \times 8$	35	0.51	17
$64 \times 64 \times 64$	$16 \times 16 \times 16$	36	0.52	198

Table 6.4: PCG Method for Test Case III

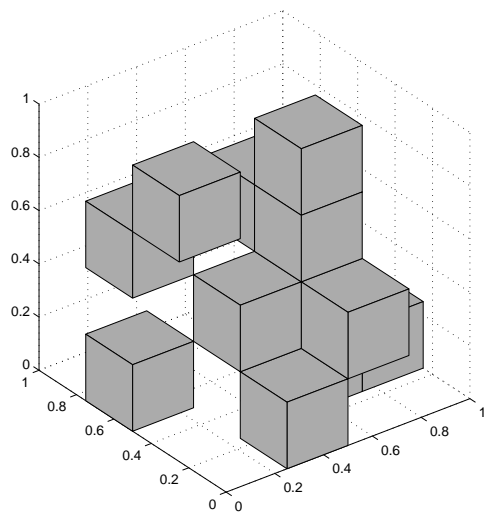


Figure 6.3: Test Case III on $4 \times 4 \times 4$ Coarse-Grid

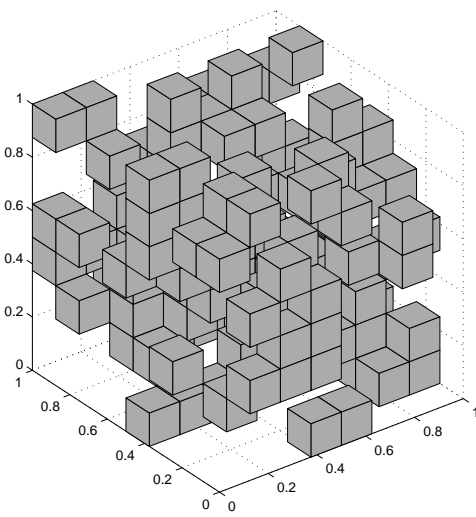


Figure 6.4: Test Case III on $8 \times 8 \times 8$ Coarse-Grid

6.1.4 Test Case IV

In the next set of tests we let the hydraulic conductivity tensor vary randomly. The conductivity tensor on a coarse-grid element is defined as

$$\mathbf{K} = \begin{bmatrix} 10^{p_x} & 0 & 0 \\ 0 & 10^{p_y} & 0 \\ 0 & 0 & 10^{p_z} \end{bmatrix}, \quad (6.1)$$

where p_x , p_y , and p_z are random integers between -5 and 0. This example shows the effects of heterogeneity and anisotropy. Table 6.5 shows the iteration counts for different grid sizes and Figure 6.5 shows the reduction histories.

Fine-grid Size	Coarse-grid Size	Iterations	Average Reduction Rate	Solver Time
$16 \times 16 \times 16$	$4 \times 4 \times 4$	97	0.79	4
$32 \times 32 \times 32$	$8 \times 8 \times 8$	155	0.86	70
$64 \times 64 \times 64$	$16 \times 16 \times 16$	259	0.91	1537

Table 6.5: PCG Method for Test Case IV

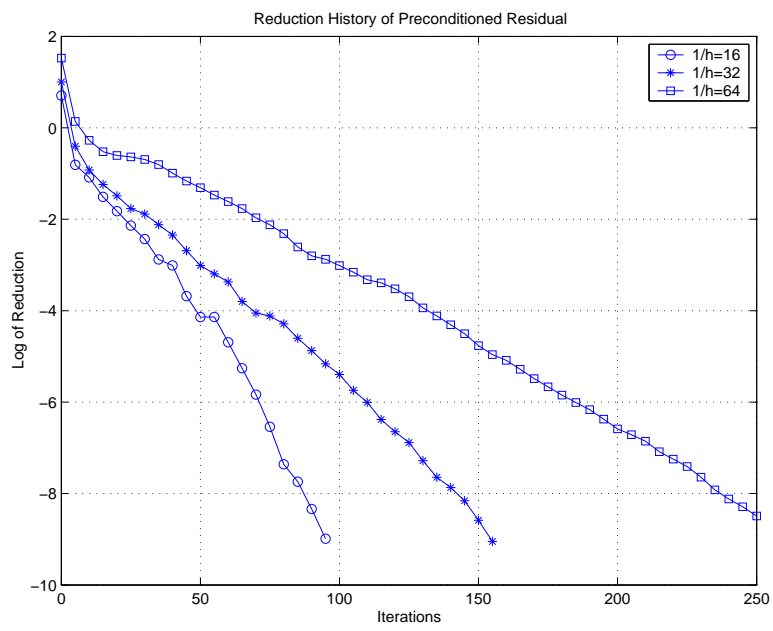


Figure 6.5: Reduction Histories for Test Case IV

6.2 Model Problem I

We now model the flow around a low conductivity vertical cylinder (parallel to the z axis) of radius a ; similar to what is described in [18]. The conductivity inside the cylinder is K_2 and outside the cylinder the conductivity is K_1 . The pressures p_2 inside the cylinder and p_1 outside the cylinder are given by

$$\begin{aligned} p_1 &= P(x - 1/2) \left(1 - \frac{(K_2 - K_1)a^2}{(K_2 + K_1)((x - 1/2)^2 + (y - 1/2)^2)} \right) \\ p_2 &= 2P \frac{K_1(x - 1/2)}{K_1 + K_2}. \end{aligned} \quad (6.2)$$

The pressure given in (6.2) satisfies the partial differential equation, given by

$$\begin{aligned} -\nabla \cdot (\mathbf{K} \nabla p) &= 0 \\ p &= \begin{cases} -\frac{1}{2}P \left(1 - \frac{(K_2 - K_1)a^2}{(K_2 + K_1)(1/4 + (y - 1/2)^2)} \right) & , \text{ for } x = 0 \\ \frac{1}{2}P \left(1 - \frac{(K_2 - K_1)a^2}{(K_2 + K_1)(1/4 + (y - 1/2)^2)} \right) & , \text{ for } x = 1 \\ P(x - 1/2) \left(1 - \frac{(K_2 - K_1)a^2(x - 1/2)}{(K_2 + K_1)((x - 1/2)^2 + 1/4)} \right) & , \text{ for } y = 0 \\ P(x - 1/2) \left(1 - \frac{(K_2 - K_1)a^2(x - 1/2)}{(K_2 + K_1)((x - 1/2)^2 + 1/4)} \right) & , \text{ for } y = 1 \end{cases} \\ \mathbf{K} \nabla p \cdot \mathbf{n} &= 0, \text{ for } z = 0, z = 1 \\ \mathbf{K} &= \begin{cases} \begin{bmatrix} K_1 & 0 & 0 \\ 0 & K_1 & 0 \\ 0 & 0 & K_1 \end{bmatrix} & , \text{ outside cylinder} \\ \begin{bmatrix} K_2 & 0 & 0 \\ 0 & K_2 & 0 \\ 0 & 0 & K_2 \end{bmatrix} & , \text{ inside cylinder} \end{cases} \end{aligned} \quad (6.3)$$

for a given P , and a given K_1 and K_2 .

The Darcy flow, $\mathbf{v} = -\mathbf{K}\nabla p$, is given by

$$\mathbf{v}_1 = \begin{bmatrix} -K_1 P \left(1 - \frac{(K_2 - K_1)a^2}{K_2 + K_1} \frac{(y-1/2)^2 - (x-1/2)^2}{((x-1/2)^2 + (y-1/2)^2)^2} \right) \\ -2K_1 P \frac{(K_2 - K_1)a^2}{K_2 + K_1} \frac{(x-1/2)(y-1/2)}{((x-1/2)^2 + (y-1/2)^2)^2} \\ 0 \end{bmatrix} \quad (6.4)$$

$$\mathbf{v}_2 = \begin{bmatrix} -K_2 2P \frac{K_1}{K_1 + K_2} \\ 0 \\ 0 \end{bmatrix} \quad (6.5)$$

where \mathbf{v}_1 is the velocity outside the cylinder and \mathbf{v}_2 is the velocity inside the cylinder.

Instead of pressure boundary conditions we can use flux boundary conditions by specifying (6.4) along the boundaries. In this experiment we use flux boundary conditions on all the boundaries. We start with a discretization of the cylinder on a $4 \times 4 \times 4$ coarse-grid and then, we refine the coarse-grid into a $16 \times 16 \times 16$ fine-grid problem. These grids are illustrated in Figures 6.6 and 6.7. The parameters used for equation (6.2) are given as:

$$\begin{aligned} P &= 1 \\ K_1 &= 1 \\ K_2 &= 10^{-5} \\ a &= 0.25 \end{aligned} \quad (6.6)$$

The numerically computed flow field is shown in Figure 6.8. The results of the PCG method for the cylinder problem are shown in Table 6.6.

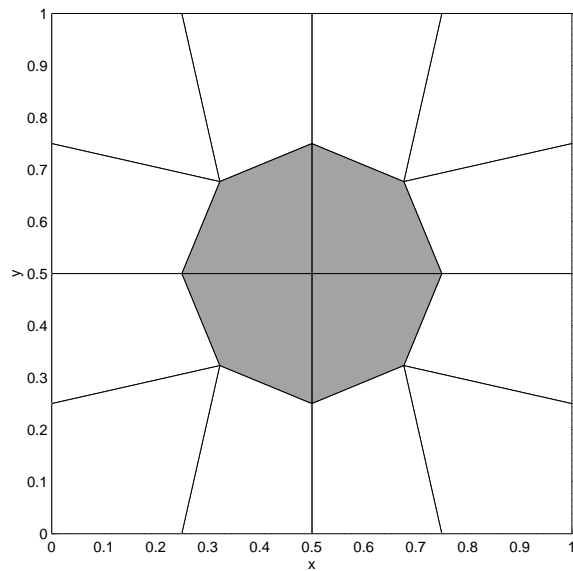


Figure 6.6: Cylinder on $4 \times 4 \times 4$ Grid

Fine-grid Size	Coarse-grid Size	Iterations	Average Reduction Rate	Solver Time
$16 \times 16 \times 16$	$4 \times 4 \times 4$	30	0.46	1
$32 \times 32 \times 32$	$8 \times 8 \times 8$	33	0.48	15
$64 \times 64 \times 64$	$16 \times 16 \times 16$	33	0.49	302

Table 6.6: PCG Method for Cylinder Problem

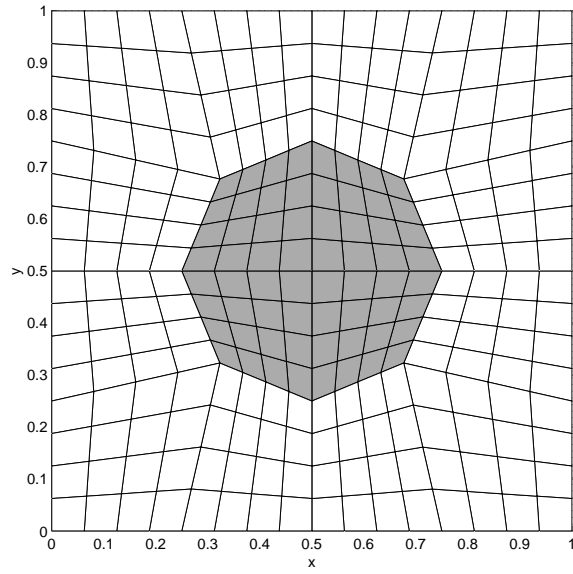


Figure 6.7: Cylinder on $16 \times 16 \times 16$ Grid

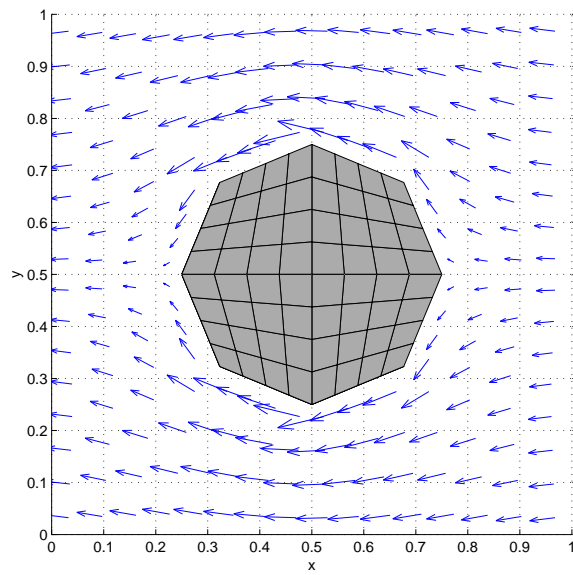


Figure 6.8: Flow Around Cylinder

6.2.1 Parallelization of Model Problem I

The cylinder problem is ideal for parallelization because it is homogeneous in the z -direction. This means that if we can slice the domain into x, y slices and distribute an equal number of the slices to different processors, then we should have a very balanced load for each processor. The domain decomposition preconditioner is ideal for this. We can group subdomain problems which all have a common z index. Then these groups can be computed independently on different processors.

We tested this type of parallelization on a 4 processor SGI Power Challenge using two different grids. First, we solve the $16 \times 16 \times 16$ cylinder problem which has a $4 \times 4 \times 4$ coarse-grid (64 subdomains). Next, we solve the $32 \times 32 \times 16$ cylinder problem which has a $8 \times 8 \times 4$ coarse-grid and 256 subdomains. The residual was again reduced by 10 orders of magnitude. We used a direct solve for the $4 \times 4 \times 4$ coarse grid and iterative solves on the subdomains using a diagonal preconditioner. An iterative solve was used on the $8 \times 8 \times 4$ coarse-grid. Tables 6.7 and 6.8 show near perfect speedups going from one processor to two processors. Using three processors does not show much improvement. This is probably due to not being able to distribute the work evenly with four coarse-grid slices being divided up amongst three processors. Four processors shows decent speedup.

Processors	CPU Time
1	214
2	109
3	107
4	64

Table 6.7: Speedup on $16 \times 16 \times 16$ Grid

Processors	CPU Time
1	961
2	489
3	487
4	286

Table 6.8: Speedup on $32 \times 32 \times 16$ Grid

Parallelization of this problem is very easy using the C compiler on the SGI Power Challenge [2]. The code used to compute the projections of the residual onto the subdomains is given below.

```

/* File : compute_SDd_sgi.c */
#include "SD.h"

int compute_SDd_sgi(d_vector* d2_ptr,d_vector* d1_ptr,
                   SD_matrix* SD_ptr)
{
    int i0,j0,k0;
    int l0,m0,n0;
    int l0m0;
    int j0l0,k0l0m0;
    int iter;

    d_vector *d1_list,*d2_list;
    PCG_operator* PCG_list;

    l0=SD_ptr->sdd.l0;
    m0=SD_ptr->sdd.m0;
    n0=SD_ptr->sdd.n0;

    l0m0=l0*m0;

    d1_list=SD_ptr->d1_list.d_list;
    d2_list=SD_ptr->d2_list.d_list;
    PCG_list=SD_ptr->PCG.PCG_list;

    sd_zero_sgi(&SD_ptr->d1_list);
    compute_G2Ld(&SD_ptr->d1_list,d1_ptr,&SD_ptr->sdd);

#pragma parallel
#pragma shared(d1_list,d2_list,PCG_list)
#pragma local(i0,j0,k0,j0l0,k0l0m0)
{
#pragma pfor iterate(k0=0;n0;1)
    for(k0=0;k0<n0;k0++)
    {
        k0l0m0=k0*l0m0;
        for(j0=0;j0<m0;j0++)
        {
            j0l0=j0*l0;
            for(i0=0;i0<l0;i0++)
            {
                compute_PCGd(&d2_list[i0+j0l0+k0l0m0],
                            &d1_list[i0+j0l0+k0l0m0],
                            &PCG_list[i0+j0l0+k0l0m0]);
            }
        }
    }
}

```

```

    }
  }
}
r_zero(&d2_ptr->r);
compute_L2Gd(d2_ptr,&SD_ptr->d2_list,&SD_ptr->sdd);

return 1;
}

```

The parallelization is done on the outer loop. Therefore, each processor gets its own i_0 , j_0 loop. For each i_0 and for each j_0 , processor k_0 computes an $a(\cdot, \cdot)$ -projection of the residual onto subdomain $i_0+j_0l_0+k_0l_0m_0$. This is done using the PCG method. Usually, the preconditioner for a subdomain is the inverse of the Cholesky factorization of the local stiffness matrix. In this case, the PCG method converges in one iteration. This is equivalent to doing a direct solve. If the stiffness matrix is stored locally on each processor then, there will not be any communication costs in these loops.

Communication costs are incurred when we map the global vector to the local subdomain vectors (`compute_G2Ld`) and when we map the local subdomain vectors to the global vector (`compute_L2Gd`). The SGI Power Challenge is essentially a shared memory system, i.e., we do not have direct control over communications between the processors and the memory. Most large parallel systems use distributed memory. That is, each processor has a local memory. In this situation we would need to write the `compute_G2Ld` and `compute_L2Gd` subroutines using, for example, MPI.

6.3 Model Problem II

Our next model problem is a laboratory tank experiment which was studied in [32]. As with the cylinder, the tank experiment is essentially a two-dimensional

problem. Sands of different conductivities are packed as blocks into the tank. A constant hydraulic head is imposed at two separate ends of the tank to induce a flow. The other boundaries are no-flow. We ran the experiment using a $20 \times 10 \times 3$ coarse-grid and a $80 \times 40 \times 12$ fine-grid. We ran the experiment again using a $40 \times 20 \times 3$ coarse-grid and a $160 \times 80 \times 12$ fine-grid. The computed flow-lines are shown in Figure 6.9.

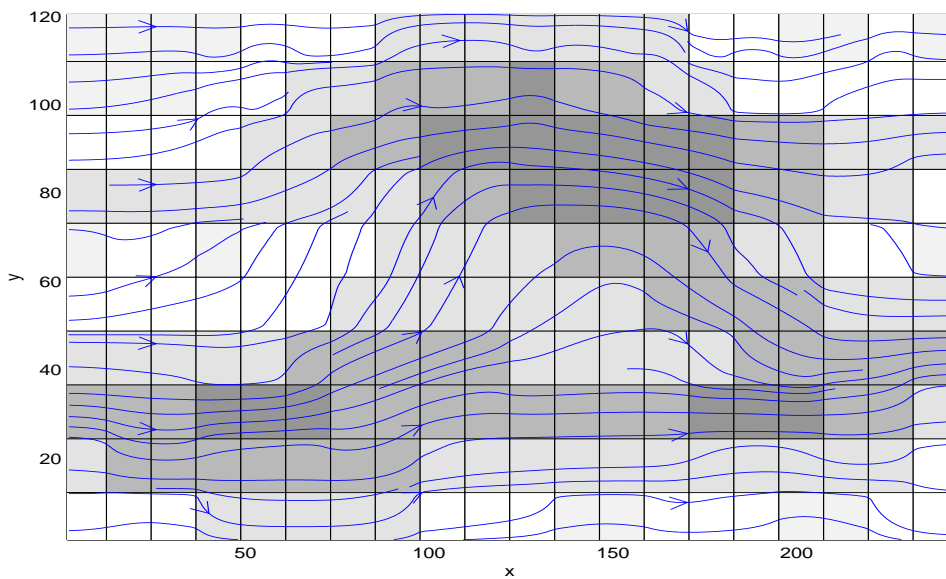


Figure 6.9: Computed Flow Lines for Tank Experiment

The shaded areas represent the different hydraulic conductivities. The lighter patches are higher conductivities and the darker patches are lower conductivities. Five different grades of crushed silica were used for the sands. The characteristics of the sands are summarized in Table 6.9.

To run the experiment, we first need to compute the approximate $a(\cdot, \cdot)$ -orthogonal projection of the pipe function onto the locally divergence-free sub-

Grade	Mean Diameter (mm)	Conductivity (m/day)
110	0.103	3.628
70	0.180	11.664
30	0.4102	100.224
16	0.880	371.52
8	1.390	1036.8

Table 6.9: Sand Characteristics

space. This is done using the PCG method with the domain decomposition preconditioner. We look at two different approximations of the pipe function. In one case we compute the pipe function so that the preconditioned residual is approximately 1. We will label this approximation P1. We also look at the case when we approximate the pipe function with a tolerance of 10^{-5} . We will label this approximation P2. For the $20 \times 10 \times 12$ coarse-grid we use a direct solve for the coarse-grid problem. On the $40 \times 20 \times 3$ coarse-grid we use an iterative solve with diagonal preconditioning. The computational costs of computing P1 and P2 are given in Tables 6.10 and 6.11 respectively.

Fine-grid Size	Coarse-grid Size	Iterations	Average Reduction Rate	Solver Time
$80 \times 40 \times 12$	$20 \times 10 \times 3$	8	0.48	4
$160 \times 80 \times 12$	$40 \times 20 \times 3$	9	0.47	56

Table 6.10: Computing P1

We then observe the convergence rate of the PCG method on the bordered matrix using the bordered matrix preconditioner with approximate pipe functions P1 and P2. We iterate until the residual is decreased by 10 orders of magnitude. The results for P1 are shown in Table 6.12 and the results for P2 are shown in Table 6.13. We see that the more we reduce the residual in the pipe function

Fine-grid Size	Coarse-grid Size	Iterations	Average Reduction Rate	Solver Time
$80 \times 40 \times 12$	$20 \times 10 \times 3$	14	0.51	13
$160 \times 80 \times 12$	$40 \times 20 \times 3$	25	0.49	146

Table 6.11: Computing P2

approximation the better the condition number for the preconditioned bordered matrix becomes.

Fine-grid Size	Coarse-grid Size	Iterations	Average Reduction Rate	Solver Time
$80 \times 40 \times 12$	$20 \times 10 \times 3$	31	0.47	15
$160 \times 80 \times 12$	$40 \times 20 \times 3$	27	0.43	159

Table 6.12: PCG Method Using P1

Fine-grid Size	Coarse-grid Size	Iterations	Average Reduction Rate	Solver Time
$80 \times 40 \times 12$	$20 \times 10 \times 3$	14	0.18	7
$160 \times 80 \times 12$	$40 \times 20 \times 3$	12	0.14	74

Table 6.13: PCG Method Using P2

Now we look at how a source/sink term will effect the tank experiment. We randomly place a vertical line-sink of magnitude 10. The computed flow-lines for this experiment are shown in Figure 6.10. We use the P2 pipe function approximation. However, we can see from Table 6.14 that the number of iterations is of the same order as in Table 6.12 where we used the P1 approximation for the pipe function. The way to understand this is to recall the statement made about an ideal pipe function. If we have a constant hydraulic conductivity tensor and

an orthogonal grid then, the pipe function is $a(\cdot, \cdot)$ -orthogonal to all the locally divergence-free functions and the PCG method will converge in one iteration. We need to qualify that statement by saying that the PCG method will converge in one iteration if we have zero source and sink terms and the correct boundary conditions corresponding to the pipe function. It is obvious that we cannot have a pipe function with the correct divergence which is divergence-free unless the source/sink term is zero. Table 6.15 shows the iteration count when using the P1 approximation of the pipe function. We conclude that an approximation of order one to the pipe function is more efficient in general than a more accurate approximation.

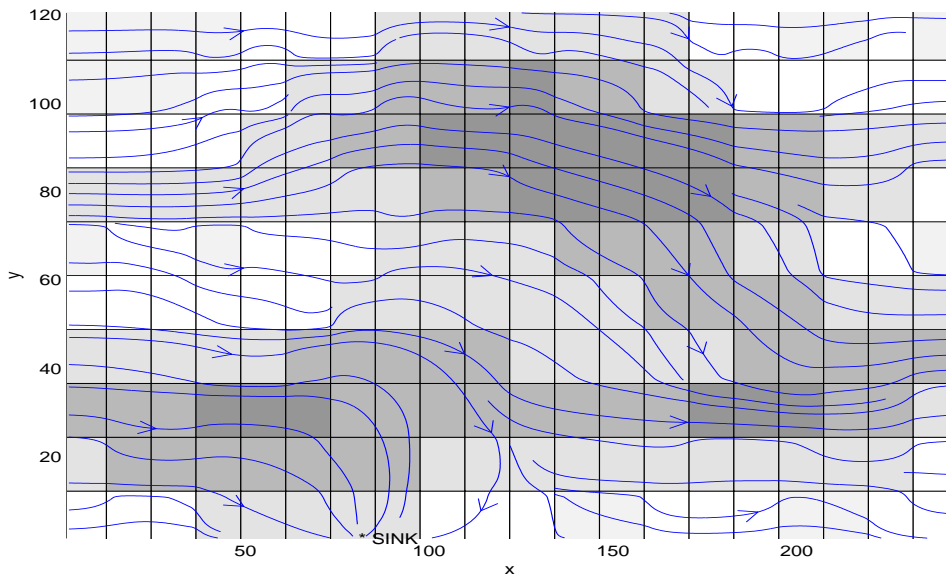


Figure 6.10: Computed Flow Lines for Tank Experiment with Sink Term

Fine-grid Size	Coarse-grid Size	Iterations	Average Reduction Rate	Solver Time
$80 \times 40 \times 12$	$20 \times 10 \times 3$	36	0.52	18
$160 \times 80 \times 12$	$40 \times 20 \times 3$	34	0.50	193

Table 6.14: PCG Method for Model Problem II with Sink Using P1

Fine-grid Size	Coarse-grid Size	Iterations	Average Reduction Rate	Solver Time
$80 \times 40 \times 12$	$20 \times 10 \times 3$	36	0.52	18
$160 \times 80 \times 12$	$40 \times 20 \times 3$	33	0.50	188

Table 6.15: PCG Method for Model Problem II with Sink Using P2

6.4 Error Analysis

In this section we approximate the error in the fluxes using an extrapolation technique. We impose Neumann boundary conditions at $x = 0$ and $x = 1$ such that the total flux across each of these boundaries is equal to one. Then for a given distribution of hydraulic conductivities we compute an approximation for grid sizes $h_0 = 1/16$, $h_1 = 1/32$, and $h_2 = 1/64$. We define the true value of the x -component of the velocity as u^x . The approximate values of the x -component for h_0 , h_1 , and h_2 are u_0^x , u_1^x , and u_2^x respectively. Then for some $\alpha > 0$ we assume the error can be approximated by

$$\begin{aligned} u_0^x - u^x &= h_0^\alpha c_0 \\ u_1^x - u^x &= h_1^\alpha c_0 = (1/2)^\alpha h_0^\alpha c_0 \\ u_2^x - u^x &= h_2^\alpha c_0 = (1/2)^{2\alpha} h_0^\alpha c_0. \end{aligned} \tag{6.7}$$

If we let $c = h_0^\alpha c_0$ and $\gamma = (1/2)^\alpha$ then we can write the approximate errors, e_0 , e_1 , and e_2 as:

$$\begin{aligned} e_0 &= u_0^x - u^x = c \\ e_1 &= u_1^x - u^x = \gamma c \\ e_2 &= u_2^x - u^x = \gamma^2 c. \end{aligned} \tag{6.8}$$

This gives us three equations with three unknowns. Solving for these unknowns we have

$$u^x = \frac{u_0^x u_2^x - (u^x)^2}{u_0^x + u_2^x - 2u_1^x} \tag{6.9}$$

$$c = \frac{(u_0^x - u_1^x)^2}{u_0^x + u_2^x - 2u_1^x} \tag{6.10}$$

$$\gamma = \frac{u_1^x - u_2^x}{u_0^x - u_1^x}. \tag{6.11}$$

The absolute values of the approximate errors are given as:

$$e_0 = \left| \frac{(u_0^x - u_1^x)^2}{u_0^x + u_2^x - 2u_1^x} \right| \quad (6.12)$$

$$e_1 = \left| \frac{(u_0^x - u_1^x)(u_1^x - u_2^x)}{u_0^x + u_2^x - 2u_1^x} \right| \quad (6.13)$$

$$e_2 = \left| \frac{(u_1^x - u_2^x)^2}{u_0^x + u_2^x - 2u_1^x} \right|. \quad (6.14)$$

We compute the the errors on faces of y, z -slices of the domain. These vertical slices coincide with the element boundaries on the $16 \times 16 \times 16$ grid. For each y, z -slice we compute errors e_0 , e_1 , and e_2 on 64×64 faces. For each set of errors we compute α and the average, $\bar{\alpha}$. Since the flux on the boundary is specified, we only consider the vertical slices interior to the domain. Therefore, we compute the average of α on slices $i = 1, \dots, 15$. We extrapolate the α 's for test problems II and III with Neumann boundary conditions described as above. With these Neumann boundary conditions we have a total flux of one across each y, z -slice. This is easily confirmed numerically and validates the mass conservation property of the MFE method. The averages of α on each slice for the different test cases are given in Table 6.16. We conclude that the error in the velocities is approximately bounded by h^1 in agreement with the approximation theory for lowest-order Raviart-Thomas vector functions.

Slice	$\bar{\alpha}$	
	Test II	Test III
1	1.22	1.02
2	1.39	1.07
3	1.33	0.95
4	1.11	1.05
5	1.16	1.22
6	1.07	1.07
7	1.08	1.10
8	1.10	1.05
9	1.08	1.01
10	1.07	1.05
11	1.16	1.14
12	1.11	0.99
13	1.33	0.95
14	1.39	1.07
15	1.22	0.98

Table 6.16: Extrapolated Error Estimate

A. Subdomain Restriction and Interpolation

Recall that subdomain, Ω_s , is $l_s \times m_s \times n_s$ and has a corresponding subspace $\mathbf{D}_s \subset \mathbf{D}^h$. We also stated that we can write a subspace vector \mathbf{w}_s in terms of basis vector functions in \mathbf{D}^h . This defines the restriction operator \mathbf{I}_h^s . We say that a vector function in \mathbf{D}^h is a global vector function defined on the global domain, Ω^h , and a vector function in \mathbf{D}_s is a local vector function defined on a subdomain. The index, s , is given as:

$$\begin{aligned} s &= i_0 + j_0 l_0 + k_0 l_0 m_0 \\ 0 &\leq i_0 \leq l_0 - 1, 0 \leq j_0 \leq m_0 - 1, 0 \leq k_0 \leq n_0 - 1 \end{aligned} \tag{A.1}$$

where

$$J = l_0 \times m_0 \times n_0 \tag{A.2}$$

is the number of subdomains.

Recall that a subdomain is the partition of a coarse-grid element into fine-grid elements with a fine-grid element overlap added. It is assumed that each coarse-grid element is partitioned into the same number of fine-grid elements. Therefore, the number of elements in subdomain s can be computed as:

$$l_s = \begin{cases} l/l_0 + 2t, & \text{for } 1 \leq i_0 \leq l_0 - 2 \\ l/l_0 + t, & \text{for } i_0 = 0 \text{ or } i_0 = l_0 - 1 \end{cases} \tag{A.3}$$

$$m_s = \begin{cases} m/m_0 + 2t, & \text{for } 1 \leq j_0 \leq m_0 - 2 \\ m/m_0 + t, & \text{for } j_0 = 0 \text{ or } j_0 = m_0 - 1 \end{cases} \tag{A.4}$$

$$n_s = \begin{cases} n/n_0 + 2t, & \text{for } 1 \leq k_0 \leq n_0 - 2 \\ n/n_0 + t, & \text{for } k_0 = 0 \text{ or } k_0 = n_0 - 1 \end{cases} \tag{A.5}$$

Where the parameter t is the number of elements in the overlap, i.e, $\delta = th$ where δ is the size of the overlap. We will also need to compute offsets into the coefficient vector corresponding to a global vector function. These offsets, x_s , y_s , and z_s are computed as:

$$x_s = \begin{cases} 0, & \text{for } i_0 = 0 \\ i_0(l/l_0) - t, & \text{for } 1 \leq i_0 \leq l_0 - 1 \end{cases} \quad (\text{A.6})$$

$$y_s = \begin{cases} 0, & \text{for } j_0 = 0 \\ j_0(m/m_0) - t, & \text{for } 1 \leq j_0 \leq m_0 - 1 \end{cases} \quad (\text{A.7})$$

$$z_s = \begin{cases} 0, & \text{for } k_0 = 0 \\ k_0(n/n_0) - t, & \text{for } 1 \leq k_0 \leq n_0 - 1 \end{cases} \quad (\text{A.8})$$

The computation of the restriction operator is straightforward for the most part. We match the local (subdomain) nodes with the global nodes. Let $\{w_{j_s+1/2, k_s+1/2, i_s}^{x_s}\}$ be the vector of coefficients representing x -slice functions on subdomain s and let $\{w_{j+1/2, k+1/2, i}^x\}$ be the vector of coefficients representing x -slice function on the global domain. We can then write the x -slice coefficients for subdomain s in terms of global x -slice coefficients as:

$$w_{j+1/2, k+1/2, i}^{x_s} = w_{j+1/2+y_s, k+1/2+z_s, i+x_s}^x \quad (\text{A.9})$$

$$0 \leq i \leq l_s - 1, 0 \leq j \leq m_s - 2, 0 \leq k \leq n_s - 2.$$

The y -slice and z -slice subdomain coefficients can be computed in an analogous fashion. In fact, we can use the same subroutine by using permutations of the data.

Recall that x -slice functions have coefficients on only one special x -slice. If subdomain s does not coincide with the special global x -slice, then the subdomain

x -slice nodes will have no corresponding global x -slice nodes. In this case, we have to use the linear dependence given by (4.116) to define the subdomain x -slice nodes. This is given by:

$$\begin{aligned}
w_{j+1/2,k+1/2}^{x_s} &= w_{j+1/2+y_s,k+1/2+z_s}^x \\
&+ \sum_{i=0}^{x_s} w_{i+1/2+x_s,j+1/2+y_s,k+z_s}^z \\
&- \sum_{i=0}^{x_s} w_{i+1/2+x_s,j+1/2+y_s,k+z_s+1}^z \\
&+ \sum_{i=0}^{x_s} w_{k+1/2+z_s,i+1/2+x_s,j+y_s}^y \\
&- \sum_{i=0}^{x_s} w_{k+1/2+z_s,i+1/2+x_s,j+y_s+1}^y
\end{aligned} \tag{A.10}$$

$$0 \leq j \leq m_s - 2, 0 \leq k \leq n_s - 2.$$

The restriction defined by (A.10) can be written in recursive form as:

$$\begin{aligned}
w_{j+1/2,k+1/2,0}^{x_{s_0}} &= w_{j+1/2+y_s,k+1/2+z_s,0}^x \\
0 \leq i \leq l_s - 1, 0 \leq j \leq m_s - 2, 0 \leq k \leq n_s - 2 \\
s_0 &= j_0 * l_0 + k_0 * l_0 * m_0 \\
0 \leq j_0 \leq m_0 - 1, 0 \leq k_0 \leq n_0 - 1
\end{aligned} \tag{A.11}$$

$$\begin{aligned}
w_{j+1/2,k+1/2,0}^{x_{s_{i_0}}} &= w_{j+1/2,k+1/2,0}^{x_{s_0}} \\
&+ \sum_{i=0}^{l_s-2*t-1} w_{k+1/2,i+1/2,j}^{y_{s_{i_0}-1}} \\
&- \sum_{i=0}^{l_s-2*t-1} w_{k+1/2,i+1/2,j+1}^{y_{s_{i_0}-1}} \\
&+ \sum_{i=0}^{l_s-2*t-1} w_{i+1/2,j+1/2,k}^{z_{s_{i_0}-1}} \\
&- \sum_{i=0}^{l_s-2*t-1} w_{i+1/2,j+1/2,k+1}^{z_{s_{i_0}-1}}
\end{aligned} \tag{A.12}$$

$$0 \leq j \leq m_s - 2, 0 \leq k \leq n_s - 2$$

$$s_{i_0-1} = (i_0 - 1) + j_0 * l_0 + k_0 * l_0 * m_0$$

$$s_{i_0} = i_0 + j_0 * l_0 + k_0 * l_0 * m_0$$

$$1 \leq i_0 \leq l_0 - 1, 0 \leq j_0 \leq m_0 - 1, 0 \leq k_0 \leq n_0 - 1.$$

The interpolation operator, \mathbf{I}_s^h , is defined as the transpose of the restriction operator. For example, the transpose of (A.9) is given by

$$\begin{aligned} w_{j+1/2+y_s, k+1/2+z_s, i+l_s}^x &= w_{j+1/2, k+1/2, i}^{x_s} \\ 0 \leq i \leq l_s - 1, 0 \leq j \leq m_s - 2, 0 \leq k \leq n_s - 2. \end{aligned} \quad (\text{A.13})$$

The transpose of (A.11) (A.12) is given by

$$\begin{aligned} w_{j+1/2+y_s, k+1/2+z_s, 0}^x &= w_{j+1/2, k+1/2, 0}^{x_{s_0}} \\ 0 \leq i \leq l_s - 1, 0 \leq j \leq m_s - 2, 0 \leq k \leq n_s - 2 \\ s_0 &= j_0 * l_0 + k_0 * l_0 * m_0 \\ 0 \leq j_0 \leq m_0 - 1, 0 \leq k_0 \leq n_0 - 1 \end{aligned} \quad (\text{A.14})$$

$$\begin{aligned} w_{j+1/2, k+1/2, 0}^{x_{s_{i_0-1}}} &= w_{j+1/2, k+1/2, 0}^{x_{s_{i_0}}} \\ w_{k+1/2, i+1/2, j}^{y_{s_{i_0-1}}} &= w_{j+1/2, k+1/2, 0}^{x_{s_{i_0}}} \\ w_{k+1/2, i+1/2, j+1}^{y_{s_{i_0-1}}} &= -w_{j+1/2, k+1/2, 0}^{x_{s_{i_0}}} \\ w_{i+1/2, j+1/2, k}^{z_{s_{i_0-1}}} &= w_{j+1/2, k+1/2, 0}^{x_{s_{i_0}}} \\ w_{i+1/2, j+1/2, k+1}^{z_{s_{i_0-1}}} &= -w_{j+1/2, k+1/2, 0}^{x_{s_{i_0}}} \\ 0 \leq i \leq l_s - 2 * t - 1, 0 \leq j \leq m_s - 2, 0 \leq k \leq n_s - 2 \\ s_{i_0-1} &= (i_0 - 1) + j_0 * l_0 + k_0 * l_0 * m_0 \\ s_{i_0} &= i_0 + j_0 * l_0 + k_0 * l_0 * m_0 \\ 1 \leq i_0 \leq l_0 - 1, 0 \leq j_0 \leq m_0 - 1, 0 \leq k_0 \leq n_0 - 1. \end{aligned} \quad (\text{A.15})$$

The restriction and interpolation of coefficients on the boundary is defined in an analogous fashion keeping in mind the various linear dependencies described in Sections 4.3 and 4.3.1. However, linear dependencies for the corner functions are not known a priori. It becomes impractical to define the restriction operator for these corner functions. The current implementation leaves these corner functions

out of the subdomain spaces, but they are included in the coarse-grid space. This does not seem to effect the performance of the preconditioner.

B. Coarse-Grid Restriction and Interpolation

We first define the interpolation operator, \mathbf{I}_H^h . Consider the coarse-grid partition, Ω^H , and the coarse-grid divergence-free subspace, \mathbf{D}^H . Recall that the divergence-free subspace is spanned by x -slice, y -slice, and z -slice divergence-free vector functions. We will need to consider special interpolations due to the special x -slice. Recall that the x -slice functions are restricted to only a single x -slice. Our strategy will be to define the interpolation for the x -slice functions as if there were no special x -slice. Then, a subroutine based on this interpolation can also be used for the y -slice functions and the z -slice functions by using permutations in the data. The y -slice and z -slice functions will not require any special interpolation formulas. We then go back and redefine the interpolation for the x -slice functions giving us a special interpolation formula. A special subroutine can then be written based on this formula.

Let $\mathbf{w}_{j_0+1/2, k_0+1/2, i_0}^{xH}$ be a coarse-grid x -slice divergence-free basis vector function and let $w_{j_0+1/2, k_0+1/2, i_0}^{xH}$ be the corresponding nodal value. This function has a support on four elements. We now partition the four coarse-grid elements into fine-grid elements as shown in Figure B.1.

The fine-grid nodal values are defined by evaluating the coarse-grid vector function at the fine-grid nodes. It can be shown that these fine-grid nodal values are a bilinear interpolation of the coarse-grid nodal value. This is because the coarse-grid vector function is the **curl** of a bilinear vector potential function.

The fine-grid is $l \times m \times n$ and the coarse-grid is $l_0 \times m_0 \times n_0$. We assume that the fine-grid partition is uniform in each coarse-grid element and each coarse-grid

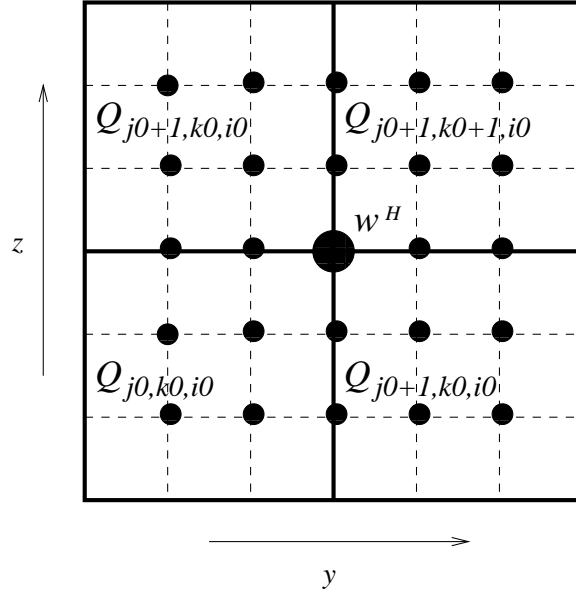


Figure B.1: Coarse Grid x -Slice Function

element is partitioned into an equal number of fine-grid elements. We define x , y , and z coordinates as follows:

$$x0_i = (i + 1)/l_H \tag{B.1}$$

$$x1_i = (l_H - i - 1)/l_H \tag{B.2}$$

$$y0_j = (j + 1)/m_H \tag{B.3}$$

$$y1_j = (m_H - j - 1)/m_H \tag{B.4}$$

$$z0_k = (k + 1)/n_H \tag{B.5}$$

$$z1_k = (n_H - k - 1)/n_H \tag{B.6}$$

where

$$l_H = l/l_0 \tag{B.7}$$

$$m_H = m/m_0 \tag{B.8}$$

$$n_H = n/n_0. \quad (\text{B.9})$$

Then, the fine-grid nodal values in coarse-grid element Q_{i_0, j_0, k_0} are given by

$$\begin{aligned} w_{j+1/2+j_0*m_H, k+1/2+k_0*n_H, i+i_0*l_H}^x &= \frac{y_0^j z_0^k}{l_H} w_{j_0+1/2, k_0+1/2, i_0}^{x_H} \\ 0 \leq i \leq l_H - 1, 0 \leq j \leq m_H - 1, 0 \leq k \leq n_H - 1. \end{aligned} \quad (\text{B.10})$$

In element Q_{i_0, j_0+1, k_0} the fine-grid nodal values are given by

$$\begin{aligned} w_{j+1/2+(j_0+1)*m_H, k+1/2+k_0*n_H, i+i_0*l_H}^x &= \frac{y_1^j z_0^k}{l_H} w_{j_0+1/2, k_0+1/2, i_0}^{x_H} \\ 0 \leq i \leq l_H - 1, 0 \leq j \leq m_H - 2, 0 \leq k \leq n_H - 1. \end{aligned} \quad (\text{B.11})$$

In element Q_{i_0, j_0+1, k_0+1} the fine-grid nodal values are given by

$$\begin{aligned} w_{j+1/2+(j_0+1)*m_H, k+1/2+(k_0+1)*n_H, i+i_0*l_H}^x &= \frac{y_1^j z_1^k}{l_H} w_{j_0+1/2, k_0+1/2, i_0}^{x_H} \\ 0 \leq i \leq l_H - 1, 0 \leq j \leq m_H - 2, 0 \leq k \leq n_H - 2. \end{aligned} \quad (\text{B.12})$$

Finally, in element Q_{i_0, j_0, k_0+1} the fine-grid nodal values are given by

$$\begin{aligned} w_{j+1/2+j_0*m_H, k+1/2+(k_0+1)*n_H, i+i_0*l_H}^x &= \frac{y_0^j z_1^k}{l_H} w_{j_0+1/2, k_0+1/2, i_0}^{x_H} \\ 0 \leq i \leq l_H - 1, 0 \leq j \leq m_H - 1, 0 \leq k \leq n_H - 2. \end{aligned} \quad (\text{B.13})$$

The coarse-grid y -slice and z -slice functions can be interpolated in the same manner. In fact, we can use the same subroutine using a permutation of the data.

Since w^{x_H} is restricted to only one coarse-grid x -slice, we will need to use a special interpolation based on linear dependencies described in Section 4.3. We now restrict the interpolation of coarse-grid x -slice functions to just the first coarse-grid

x -slice. The special interpolation for element Q_{0,j_0,k_0} is given by

$$\begin{aligned}
w_{j+1/2+j_0*m_H,k+1/2+k_0*n_H}^x &= \frac{y^0_j z^0_k}{1} w_{j_0+1/2,k_0+1/2,0}^{x_H} \\
w_{k+1/2+k_0*n_H,i+1/2,j+j_0*m_H}^y &= \frac{x^1_i y^0_j z^0_k}{1} w_{j_0+1/2,k_0+1/2,0}^{x_H} \\
w_{k+1/2+k_0*n_H,i+1/2,j+1+j_0*m_H}^y &= -\frac{x^1_i y^0_j z^0_k}{1} w_{j_0+1/2,k_0+1/2,0}^{x_H} \\
w_{i+1/2,j+1/2+j_0*m_H,k+k_0*n_H}^z &= \frac{x^1_i y^0_j z^0_k}{1} w_{j_0+1/2,k_0+1/2,0}^{x_H} \\
w_{i+1/2,j+1/2+j_0*m_H,k+1+k_0*n_H}^z &= -\frac{x^1_i y^0_j z^0_k}{1} w_{j_0+1/2,k_0+1/2,0}^{x_H} \\
0 \leq i \leq l_H - 2, 0 \leq j \leq m_H - 1, 0 \leq k \leq n_H - 1.
\end{aligned} \tag{B.14}$$

The special interpolation for element Q_{0,j_0+1,k_0} is given by

$$\begin{aligned}
w_{j+1/2+(j_0+1)*m_H,k+1/2+k_0*n_H}^x &= \frac{y^1_j z^0_k}{1} w_{j_0+1/2,k_0+1/2,0}^{x_H} \\
w_{k+1/2+k_0*n_H,i+1/2,j+(j_0+1)*m_H}^y &= \frac{x^1_i y^1_j z^0_k}{1} w_{j_0+1/2,k_0+1/2,0}^{x_H} \\
w_{k+1/2+k_0*n_H,i+1/2,j+1+(j_0+1)*m_H}^y &= -\frac{x^1_i y^1_j z^0_k}{1} w_{j_0+1/2,k_0+1/2,0}^{x_H} \\
w_{i+1/2,j+1/2+(j_0+1)*m_H,k+k_0*n_H}^z &= \frac{x^1_i y^1_j z^0_k}{1} w_{j_0+1/2,k_0+1/2,0}^{x_H} \\
w_{i+1/2,j+1/2+(j_0+1)*m_H,k+1+k_0*n_H}^z &= -\frac{x^1_i y^1_j z^0_k}{1} w_{j_0+1/2,k_0+1/2,0}^{x_H} \\
0 \leq i \leq l_H - 2, 0 \leq j \leq m_H - 2, 0 \leq k \leq n_H - 1.
\end{aligned} \tag{B.15}$$

The special interpolation for element Q_{0,j_0+1,k_0+1} is given by

$$\begin{aligned}
w_{j+1/2+(j_0+1)*m_H,k+1/2+(k_0+1)*n_H}^x &= \frac{y^1_j z^0_k}{1} w_{j_0+1/2,k_0+1/2,0}^{x_H} \\
w_{k+1/2+(k_0+1)*n_H,i+1/2,j+(j_0+1)*m_H}^y &= \frac{x^1_i y^1_j z^0_k}{1} w_{j_0+1/2,k_0+1/2,0}^{x_H} \\
w_{k+1/2+(k_0+1)*n_H,i+1/2,j+1+(j_0+1)*m_H}^y &= -\frac{x^1_i y^1_j z^0_k}{1} w_{j_0+1/2,k_0+1/2,0}^{x_H} \\
w_{i+1/2,j+1/2+(j_0+1)*m_H,k+(k_0+1)*n_H}^z &= \frac{x^1_i y^1_j z^0_k}{1} w_{j_0+1/2,k_0+1/2,0}^{x_H} \\
w_{i+1/2,j+1/2+(j_0+1)*m_H,k+1+(k_0+1)*n_H}^z &= -\frac{x^1_i y^1_j z^0_k}{1} w_{j_0+1/2,k_0+1/2,0}^{x_H} \\
0 \leq i \leq l_H - 2, 0 \leq j \leq m_H - 2, 0 \leq k \leq n_H - 2.
\end{aligned} \tag{B.16}$$

Finally, the special interpolation for element Q_{0,j_0,k_0+1} is given by

$$\begin{aligned}
w_{j+1/2+j_0*m_H,k+1/2+(k_0+1)*n_H}^x &= \frac{y_0^j z_0^k}{1} w_{j_0+1/2,k_0+1/2,0}^{x_H} \\
w_{k+1/2+(k_0+1)*n_H,i+1/2,j+j_0*m_H}^y &= \frac{x_1 y_0^j z_0^k}{1} w_{j_0+1/2,k_0+1/2,0}^{x_H} \\
w_{k+1/2+(k_0+1)*n_H,i+1/2,j+1+j_0*m_H}^y &= -\frac{x_1 y_0^j z_0^k}{1} w_{j_0+1/2,k_0+1/2,0}^{x_H} \\
w_{i+1/2,j+1/2+j_0*m_H,k+(k_0+1)*n_H}^z &= \frac{x_1 y_0^j z_0^k}{1} w_{j_0+1/2,k_0+1/2,0}^{x_H} \\
w_{i+1/2,j+1/2+j_0*m_H,k+1+(k_0+1)*n_H}^z &= -\frac{x_1 y_0^j z_0^k}{1} w_{j_0+1/2,k_0+1/2,0}^{x_H} \\
0 \leq i \leq l_H - 2, 0 \leq j \leq m_H - 1, 0 \leq k \leq n_H - 2.
\end{aligned} \tag{B.17}$$

We now define the restriction operator, \mathbf{I}_h^H , as the transpose of \mathbf{I}_H^h . For example, the transpose of (B.7) is given by

$$w_{j_0+1/2,k_0+1/2,i_0}^{x_H} = \sum_{i=0}^{l_H-1} \sum_{k=0}^{n_H-1} \sum_{j=0}^{m_H-1} \frac{y_0^j z_0^k}{l_H} w_{j+1/2+j_0*m_H,k+1/2+k_0*n_H,i+i_0*l_H}^x. \tag{B.18}$$

We also add the transposes of (B.11)-(B.13) to $w_{j_0+1/2,k_0+1/2,i_0}^{x_H}$ in a similar manner.

The transpose of the special interpolation, (B.14), is given by

$$\begin{aligned}
w_{j_0+1/2,k_0+1/2}^{x_H} &= \sum_{k=0}^{n_H-1} \sum_{j=0}^{m_H-1} \frac{y_0^j z_0^k}{1} w_{j+1/2+j_0*m_H,k+1/2+k_0*n_H}^x \\
&+ \sum_{k=0}^{n_H-1} \sum_{j=0}^{m_H-1} \sum_{i=0}^{l_H-2} \begin{pmatrix} \frac{x_1 y_0^j z_0^k}{1} w_{k+1/2+k_0*n_H,i+1/2,j+j_0*m_H}^y \\ -\frac{x_1 y_0^j z_0^k}{1} w_{k+1/2+k_0*n_H,i+1/2,j+1+j_0*m_H}^y \\ +\frac{x_1 y_0^j z_0^k}{1} w_{i+1/2,j+1/2+j_0*m_H,k+k_0*n_H}^z \\ -\frac{x_1 y_0^j z_0^k}{1} w_{i+1/2,j+1/2+j_0*m_H,k+1+k_0*n_H}^z \end{pmatrix}.
\end{aligned} \tag{B.19}$$

Again, we would also add the transposes of (B.15)-(B.17).

The interpolation of the boundary requires many special cases. These can be defined using the linear dependencies from Sections 4.3 and 4.3.1. Although numerous subroutines are required to compute the interpolations along with the many special cases, we can minimize the number of subroutines by using various permutations of the data.

References

- [1] Netlib repository. <http://www.netlib.org>. Go to <http://www.netlib.org/linpack>.
- [2] Sgi techpubs libray. <http://www.techpubs.sgi.com>.
- [3] K. E. Atkinson. *An Introduction to Numerical Analysis*. John Wiley & Sons Inc., 1988.
- [4] A. Behie, D. Collins, P. A. Forsyth Jr., and P. H. Sammon. Fully coupled multiblock wells in oil simulation. *SPEJ*, 25(4):535–542, August 1985.
- [5] Bentley, Sykes, Brebbia, Gray, and Pinder, editors. *Computational Methods in Water Resources*, volume 2, Old Post Road, Brookfield, VT 05036-9704, USA, 2000. Balkema. See the section on scaling and heterogeneity.
- [6] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring i. *Mathematics of Computation*, 47(175):103–134, July 1986.
- [7] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring ii. *Mathematics of Computation*, 49(179):1–16, July 1987.
- [8] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring iii. *Mathematics of Computation*, 51(184):415–430, October 1988.
- [9] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring iv. *Mathematics of Computation*, 53(187):1–24, July 1989.
- [10] J. H. Bramble, J. E. Pasciak, J. Wang, and J. Xu. Convergence estimates for product iterative methods with applications to domain decomposition. *Mathematics of Computation*, 57(195):1–21, July 1991.
- [11] J. H. Bramble and J. Xu. Some estimates for a weighted L^2 projection. *Mathematics of Computation*, 56(194):463–476, April 1991.
- [12] S. Brenner and Scott L. R. *The Mathematical Theory of Finite Element Methods*, pages 243–259. Springer-Verlag, 1994.

- [13] S. C. Brenner. Two-level additive schwarz preconditioners for nonconforming finite element methods. *Mathematics of Computation*, (65):897–921, 1996.
- [14] F. Brezzi, J. Douglas, M. Fortin, and L. D. Marini. Efficient rectangular mixed finite elements in two and three space variables. *Math. Model. Numerical Analysis*, (21):581–604, 1987.
- [15] F. Brezzi and M. Fortin. *Mixed and Hybrid Finite Element Methods*. Springer-Verlag, 1991.
- [16] Z. Cai, J.E. Jones, S.F. McCormick, and T.F. Russell. Control-volume mixed finite elements methods. *Computational Geosciences*, (1):289–315, 1997.
- [17] Z. Cai, R. Parashkevov, and T. Russell. Domain decomposition for a mixed finite element method in three dimensions. Technical Report 78, University of Colorado at Denver, Center for Computation Mathematics, February 1996.
- [18] H. S. Carslaw and J. C. Jaeger. *Conduction of Heat in Solids*, page 426. Oxford University Press, second edition, 1959.
- [19] P. G. Ciarlet and P. A. Raviart. General lagrange and hermite interpolation in \mathbb{R}^n with applications to finite element methods. *Arch. Rat. Mech. Anal.*, 46:177–199, 1972.
- [20] L. C. Cowsar, J. Mandel, and M. F. Wheeler. Balancing domain decomposition for mixed finite elements. *Mathematics of Computation*, 64(211):989–1015, July 1995.
- [21] M. Dryja. A method of domain decomposition for 3-d finite element problems. In R. Glowinski, G. H. Golub, G. A. Meurant, and J. Périaux, editors, *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 43–61. SIAM, 1988.
- [22] M. Dryja, B. F. Smith, and O. B. Widlund. Schwarz analysis of iterative substructuring algorithms for elliptic problems in three dimensions. *SIAM Journal of Numerical Analysis*, 31(6):1662–1694, December 1994.
- [23] M. Dryja and O. B. Widlund. Multilevel additive methods for elliptic finite element problems. In Wolfgang Hackbusch, editor, *Parallel Algorithms for Partial Differential Equations, Proceedings of the Sixth GAMM-Seminar*, pages 19–21. Kiel, 1990.

- [24] M. Dryja and O. B. Widlund. Some domain decomposition algorithms for elliptic problems. In David R. Kincaid and Linda J. Hayes, editors, *Iterative Methods for Large Linear Systems*, pages 274–288. Academic Press Inc., 1990.
- [25] M. Dryja and O. B. Widlund. Towards a unified theory of domain decomposition algorithms for elliptic problems. In Tony F. Chan, Roland Glowinski, Jacques Périaux, and Olof B. Widlund, editors, *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 3–21. SIAM, 1990.
- [26] M. Dryja and O. B. Widlund. Additive schwarz methods for elliptic finite element problems in three dimensions. In David E. Keyes, Tony F. Chan, Gérard A. Meurant, Jeffery S. Scroggs, and Robert G. Voigt, editors, *Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 3–18. SIAM, 1992.
- [27] M. Dryja and O. B. Widlund. Domain decomposition algorithms with small overlap. *SIAM Journal of Scientific Computing*, 15(3):604–620, May 1994.
- [28] F. Duboise. Discrete vector potential representation of a divergence-free vector field in three-dimensional domains: Numerical analysis of a model problem. *SIAM Journal on Numerical Analysis*, 27(5):1103–1141, October 1990.
- [29] R.E. Ewing and J. Wang. Analysis of the schwarz algorithm for mixed finite element methods. *Numerical Analysis*, (26):739–756, 1992.
- [30] C. Fetter. *Applied Hydrology*. Prentice Hall, 2001.
- [31] V.A. Garanzha and V.N. Konshin. Approximation schemes and discrete well models for the numerical simulation of the 2-d non-darcy fluid flows in porous media. Technical report, Russian Academy of Sciences Computer Centre, 1999.
- [32] J. E. Garcia. An experimental investigation of upscaling of water flow and solute transport in saturated porous media. Master’s thesis, University of Colorado, 1991. Department of Civil, Environmental, and Architectural Engineering.
- [33] R. Ghanem. Ingredients for a general purpose stochastic finite elements implementation. *Comput. Methods Appl. Mech. Engrg.*, 168:19–34, 1999.
- [34] R. Ghanem and S. Dham. Stochastic finite element analysis for multiphase flow in heterogeneous porous media. In *Transport in Porous Media*, pages 239–262. Kluwer Academic Publishers, 1998.

- [35] V. Girault. Incompressible finite element methods for navier-stokes equations with nonstandard boundary conditions in \mathbb{R}^3 . *Mathematics of Computation*, 51(183):55–74, July 1988.
- [36] V. Girault and P. A. Raviart. *Finite Element Methods for Navier-Stokes Equations: Theory and Algorithms*. Springer-Verlag, New York, 1986.
- [37] R. Glowinski and M. F. Wheeler. Domain decomposition and mixed finite element methods for elliptic problems. In Roland Glowinski, Gene H. Golub, Gérard A. Meurant, and Jacques Périaux, editors, *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 144–172. SIAM, 1988.
- [38] G. H. Golub and C. F. Van Loan. *Matrix Computations*. John Hopkins University Press, 1989.
- [39] R. B. Guenther and J. W. Lee. *Partial Differential Equations of Mathematical Physics and Integral Equations*, pages 5–7. Prentice-Hall, Englewood Cliffs, N.J. 07632, 1988.
- [40] M. J. Hagger. *DOUG User, Guide, Version 1.98*. University of Bath, 1998.
- [41] R. Healy and T. Russell. Analytical tracking along streamlines in temporally linear Raviart-Thomas velocity fields. In Bentley, Sykes, Brebbia, Gray, and Pinder, editors, *Computational Methods in Water Resources*, volume 2, pages 631–638. Balkema, Old Post Road, Brookfield, VT 05036-9704, USA, 2000.
- [42] C. Heberton, T. Russell, L. Konikow, and G. Hornberger. Three-dimensional finite-volume ellam implementation. In Bentley, Sykes, Brebbia, Gray, and Pinder, editors, *Computational Methods in Water Resources*, volume 2, pages 603–618. Balkema, Old Post Road, Brookfield, VT 05036-9704, USA, 2000.
- [43] I. Horton. *Beginning C++*. Wrox Press, 1998.
- [44] T. Hughes. *The Finite Element Method*, pages 123–125. Prentice-Hall, Englewood Cliffs, N.J. 07632, 1987.
- [45] K. Hwang. *Advanced Computer Architecture*. McGraw-Hill, 1993.
- [46] C. Johnson. *Numerical Solutions of Partial Differential Equations by the Finite Element Method*, pages 141–144. Cambridge University Press, 1987.
- [47] E. Kreyszig. *Introductory Functional Analysis With Applications*. Wiley, 1988.

- [48] W. Liu and A. H. Sherman. Comparative analysis of the cuthill-mckee and the reverse cuthill-mckee ordering algorithm for sparse matrices. *SIAM Journal of Numerical Analysis*, 13(2):198–213, April 1976.
- [49] J. Mandel. Jan mandel’s alpha page. <http://www-math.cudenver.esu/~jmandel/alpha>.
- [50] R. Naff, T. Russell, and J. Wilson. Test functions for three-dimensional control-volume finite-element methods on irregular grids. In Bentley, Sykes, Brebbia, Gray, and Pinder, editors, *Computational Methods in Water Resources*, volume 2, pages 677–684. Balkema, Old Post Road, Brookfield, VT 05036-9704, USA, 2000.
- [51] J. C. Nedelec. Mixed finite elements in \mathfrak{R}^3 . *Numer. Math.*, (35):315–341, 1980.
- [52] P.A. Raviart and J.M. Thomas. A mixed finite element method for 2nd order elliptic problems. In I. Galligani and E. Magenes, editors, *Mathematical Aspects of Finite Element Methods, Lecture Notes in Mathematics*, pages 292–315. Springer-Verlag, New York, 1977.
- [53] T. F. Russell and Trujillo R. V. The finite volume element method for elliptic and parabolic equations. Technical Report 76, University of Colorado at Denver, Center for Computation Mathematics, February 1996.
- [54] R. Scheichl. *Iterative Solution of Saddle Point Problems Using Divergence-free Finite Elements with Applications to Groundwater Flow*. PhD thesis, University of Bath, 2000.
- [55] B. Smith, P. Bjørstad, and William Gropp. *Domain Decomposition*. Cambridge University Press, 1996.
- [56] B. F. Smith. A parallel implementation of an iterative substructuring algorithm for problems in three dimensions. *SIAM Journal of Scientific Computing*, 14(2):406–423, March 1993.
- [57] J. M. Thomas. *Sur l’analyse numérique des méthodes d’éléments finis hybrides et mixtes*. PhD thesis, Université Pierre et Marie Curie, Paris, 1977.
- [58] J. Wilson and T. Russell. Efficient solver for mixed and control-volume mixed finite-element methods. In Bentley, Sykes, Brebbia, Gray, and Pinder, editors, *Computational Methods in Water Resources*, volume 2, pages 669–676. Balkema, Old Post Road, Brookfield, VT 05036-9704, USA, 2000.

- [59] J. Xu. Iterative methods by space decomposition and subspace corrections.
SIAM Review, 34(4):581–613, December 1992.