Matrix-free Implementation of the Weighted BFBT Preconditioner for Stokes Flow Problems with Highly Heterogeneous Viscosity

Sai Aakash Dasari

Master Thesis
Matrix-free Implementation of the Weighted BFBT Preconditioner for Stokes Flow Problems with Highly Heterogeneous Viscosity

Sai Aakash Dasari
Master Thesis

Aufgabensteller: Prof. Dr. U. Rüde
Betreuer: B. Mann (M.Sc.), F. Böhm (M.Sc.)
Bearbeitungszeitraum: 03.08.2023 – 05.02.2024
Erklärung:

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angege- benen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenom- men wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Der Universität Erlangen-Nürnberg, vertreten durch den Lehrstuhl für Systemsimulation (Informa- tik 10), wird für Zwecke der Forschung und Lehre ein einfaches, kostenloses, zeitlich und örtlich unbeschränktes Nutzungsrecht an den Arbeitsergebnissen der Master Thesis einschließlich etwaiger Schutzrechte und Urheberrechte eingeräumt.

Erlangen, den 3. Februar 2024

............................................
## Contents

Abstract 6

1 Introduction 7

2 Finite Element Discretization of Stokes Equations 9  
  2.1 Weak Formulation 9  
  2.2 Discretization Using Mixed Finite Elements 10  
  2.3 The Taylor-Hood Element 11  
  2.4 Properties of the Stokes Saddle-point Matrix 12

3 Solution Techniques for Discrete Stokes Systems 14  
  3.1 Krylov Subspace Methods 14  
  3.2 Convergence of Krylov Solvers 15  
  3.3 Preconditioning the Stokes System 16  
  3.4 The w-BFBT Approximation for Inverse Schur Complement 17  
  3.5 Geometric Multigrid Methods: Brief Introduction 19

4 HyTeG Preliminaries 20  
  4.1 Primitive Storage 20  
  4.2 HyTeG Functions 20  
  4.3 Elementwise Operators 21

5 Matrix-free Implementation of Weighted BFBT Operator 22  
  5.1 Incorporating Homogeneous Dirichlet Boundary Conditions 22  
  5.2 w-BFBT Weights 23  
  5.3 w-BFBT Sub-systems 25  
  5.4 The w-BFBT Operator 28

6 Numerical Experiments 29  
  6.1 The Multi-sinker Benchmark Problem 29  
  6.2 Comparison of Inverse Schur Complement Approximations 31  
  6.3 Solving w-BFBT Sub-systems 32

7 Conclusion 34

A Relevant Inequalities 37  
  A.1 Poincaré’s Inequality 37  
  A.2 Korn’s Inequality 37

B Benchmark Configuration 37  
  B.1 Location of Sinker Centers for the 20-Sinker Benchmark 37
List of Figures

1.1 Hybrid mantle convection model ............................................. 7
2.1 $P_1$ global basis function in 2D .......................................... 11
2.2 $P_2$-$P_1$ degrees of freedom ............................................. 12
3.1 Polynomial fitting for eigenvalues ........................................... 16
3.2 Sequence of steps in a two-grid V-cycle ................................... 19
4.1 Structured refinement of an unstructured triangular mesh ............... 20
5.1 Incorporating homogeneous Dirichlet boundary conditions in the Stokes system ........................................... 22
5.2 $P_2$ tetrahedral elements at grid corners ................................... 23
5.3 $P_2$ reference tetrahedron .................................................. 24
5.4 Overview of w-BFBT sub-operators ......................................... 27
6.1 Viscosity visualization on $P_2$ grid ........................................ 29
6.2 20-Sinker configuration ..................................................... 30
6.3 Comparison of inverse Schur complement approximations ............... 31
6.4 Comparison of sub-operators for VBQ1-BFBT ............................. 32

List of Tables

3.1 Characteristics of commonly utilized Krylov solvers .......................... 15
5.1 Possible weights for 3D node-based quadrature rules .......................... 25
5.2 Possible weights for 2D node-based quadrature rules .......................... 25
6.1 Comparison of inverse Schur complement approximations .................. 31
6.2 Mesh-dependent behavior of the Stokes solver ................................ 32
6.3 Performance improvement with sub-solver preconditioning: DiagA-BFBT .... 33
6.4 Performance improvement with sub-solver preconditioning: VBQ1-BFBT .... 33
6.5 Effect of sub-solver tolerance on outer iterations: DiagA-BFBT ............. 33
6.6 Effect of sub-solver tolerance on outer iterations: VBQ1-BFBT ............. 33
7.1 Differences in settings: Current work and Rudi et al. 2016 .................. 34

Listings

4.1 Creating a PrimitiveStorage with Dirichlet boundaries. ..................... 20
4.2 Incorporating boundary conditions in Function objects ...................... 21
4.3 Algebraic operations on Function objects ................................... 21
4.4 Example usage of an ElementwiseOperator .................................. 21

List of Algorithms

5.1 apply action of the BFBTOperator .......................................... 28
Abstract

One of the significant computational challenges in mantle convection simulations involves solving the stationary Stokes systems at each time step. Due to the highly heterogeneous rheological characteristics of the Earth’s mantle, these Stokes systems are extremely ill-conditioned and necessitate effective preconditioning techniques. In (Rudi et al. 2016), the authors introduce the weighted BFBT (w-BFBT) approximation as a robust alternative to the widely employed inverse-viscosity weighted pressure space mass matrix for use in Schur complement-based Stokes preconditioners. Through numerical experiments, they demonstrate the superior convergence and mesh-independent properties of the w-BFBT operator when employing the $Q_k$-$P_{k-1}^{disc}$ ($k > 2$) finite elements for discretizing the Stokes system. In this thesis, we implement and evaluate the w-BFBT operator for the $P_2$-$P_1$ Stokes systems with homogeneous Dirichlet boundary conditions within the matrix-free framework of HyTeG (Kohl et al. 2018). Such matrix-free operators are crucial for mantle convection simulations because the resulting systems are too large to be stored explicitly. As a part of our work, we explore different schemes for lumping the weighted velocity space mass matrix to derive the necessary weights. Furthermore, we investigate alternative sub-operators for the sub-systems that appear in the w-BFBT formulation to alleviate the computational workload of applying the operator. The implementation details conclude with steps for achieving the matrix-free action of the w-BFBT operator and we evaluate its effectiveness using the multi-sinker benchmark. Our experiments demonstrate that the w-BFBT operator facilitates improved convergence of the Stokes solver when approximating the inverse Schur complement in the Stokes preconditioner for $P_2$-$P_1$ systems with highly heterogeneous viscosity functions. However, further investigations into optimal weights for mesh-independent convergence and viable substitute sub-operators are required to effectively utilize the w-BFBT preconditioner in HyTeG.
1 Introduction

Gaining insights into plate tectonics and predicting geological processes necessitates a deep understanding of activity within the Earth’s mantle. A crucial process involves the transfer of heat and material from the planet’s core to the surface through the creeping motion of mantle material. Convection currents drive this process over a time scale of millions of years. Due to the inaccessible nature of the mantle, numerical models of mantle convection are necessary to test hypotheses against observational data and investigate various scenarios. Solving these models presents a challenge as the viscosity of the Earth’s mantle exhibits pronounced heterogeneity, leading to highly ill-conditioned linear systems. Viscosity contrasts in the mantle arise due to variations in temperature, composition, and the influence of geological features like plumes and tectonic plate boundaries. The frequently employed model for mantle convection, as detailed in (Schubert et al. 2001; Kronbichler et al. 2012; Zhong et al. 2007), assumes incompressibility and the Boussinesq approximation. This assumption leads to the formulation of the following system of partial differential equations for velocity ($\vec{u}$), pressure ($p$), and temperature ($T$):

\begin{align*}
-\nabla \cdot (2\mu \epsilon (\vec{u})) + \nabla p &= \rho (T) \vec{g} \quad (1.1) \\
\nabla \cdot \vec{u} &= 0 \quad (1.2) \\
\frac{\partial T}{\partial t} + \vec{u} \cdot \nabla T - \nabla \cdot (\kappa \nabla T) &= \gamma \quad (1.3)
\end{align*}

where

$\mu =$ viscosity \\
$\epsilon = \frac{1}{2} (\nabla + \nabla^T)$ \\
$\rho =$ density \\
$\vec{g} =$ gravity vector \\
$\kappa =$ thermal conductivity \\
$\gamma =$ heat sources.

In this work, we focus on a critical subset of the Boussinesq system: solving the ill-conditioned instantaneous Stokes flow problem given by eqs. (1.1) and (1.2).

Choosing an appropriate discretization technique and employing a suitable linear solver is essential to solve the Stokes system effectively. The finite element method (FEM) is capable of capturing the intricacies of geological structures and varying rheological characteristics within the mantle, making it a favourable choice for discretizing the Stokes system. To capture processes at relevant scales, a high-resolution spatial discretization is required, which leads to systems with $10^{12}$–$10^{13}$ degrees of freedom (DoFs) (Bauer 2019), rendering the sparse storage of system matrices impractical or even impossible due to memory limitations. Consequently, the adoption of matrix-free finite element methods becomes necessary. To address this challenge, we employ the HyTeG framework.

Figure 1.1: Hybrid mantle convection model (Chen 2016).
Kohl et al. 2018), specifically designed to support matrix-free methodologies on triangular and tetrahedral meshes, with a strong focus on geometric multigrid (GMG) methods. Moreover, it is essential to exercise caution when selecting the function spaces for approximating the velocity and the pressure fields to ensure a stable finite element discretization, given their coupling in the Stokes equations. The $P_2$-$P_1$ Taylor-Hood finite element, characterized by second-order polynomial elements for velocity components and first-order polynomial elements for pressure, stands out as a stable and popular choice for discretizing the Stokes system, owing to its straightforward implementation. We adopt this discretization due to its compelling convergence properties and ready availability in HyTeG.

The finite element discretization of the Stokes equations (1.1) and (1.2) using $P_2$-$P_1$ elements results in a $2 \times 2$ block saddle-point system of the form:

$$
\begin{bmatrix}
A & B^T \\
B & 0
\end{bmatrix}
\begin{bmatrix}
u \\
p
\end{bmatrix} =
\begin{bmatrix}
f \\
0
\end{bmatrix}.
$$

(1.4)

Krylov subspace methods are essential for solving such systems, especially in matrix-free formulations where it is only feasible to compute the action of an operator on a vector. The severe heterogeneity in viscosity and sharp viscosity gradients in the Earth’s mantle, along with the high-resolution spatial discretization, render the system represented by eq. (1.4) ill-conditioned and present a substantial challenge for Krylov solvers. Preconditioning this system is vital for accelerating the Krylov solvers and achieving mesh-independent convergence. To gain insights about common preconditioning strategies for the saddle-point system, refer to (Benzi et al. 2005, sec. 10). Block preconditioners are especially popular in fluid dynamics and require an effective approximation of $A^{-1}$ and $S^{-1}$, where the Schur complement $S$ is defined as:

$$
S := BA^{-1}B^T.
$$

(1.5)

The inverse-viscosity weighted pressure space mass matrix is widely employed to approximate $S$, but it proves unsatisfactory in the presence of highly heterogeneous viscosity functions (Rudi et al. 2016, sec. 1.4). In (Rudi et al. 2016), the authors introduce a weighted BFBT (w-BFBT) approximation for the inverse Schur complement as a superior alternative. Numerical experiments (for $Q_k$-$P_k^{\text{disc}}$ discretization, $k>2$) in their work demonstrate that the w-BFBT approximation exhibits robust convergence properties and optimal algorithmic scalability when tackling challenging problems. The primary objective of this thesis is the matrix-free implementation and performance assessment of this w-BFBT operator in HyTeG.
2 Finite Element Discretization of Stokes Equations

We start with the continuous Stokes equations in a bounded and connected domain $\Omega$ with Lipschitz boundary $\partial \Omega$:

\[
\begin{align*}
-\nabla \cdot (2\mu \epsilon (\vec{u})) + \nabla p &= \vec{f} \quad \text{in } \Omega \\
\nabla \cdot \vec{u} &= 0 \quad \text{in } \Omega 
\end{align*}
\]  

(2.1)

and for simplicity, limit ourselves to the homogeneous Dirichlet boundary conditions:

\[
\vec{u} = \vec{0} \quad \text{on } \partial \Omega. 
\]  

(2.3)

Note that the pressure solution of the Stokes problem represented by eqs. (2.1) to (2.3) is unique up to a constant. When examining the well-posedness of the Stokes equations, a common approach involves selecting the pressure solution from a space of functions with vanishing mean, i.e., $\int_{\Omega} p = 0$ (John 2016, Remark 3.25). Moreover, integrating the incompressibility constraint (2.2) over $\Omega$ gives us the compatibility condition:

\[
0 = \int_{\Omega} \nabla \cdot \vec{u} = \int_{\partial \Omega} \vec{u} \cdot \vec{n} 
\]  

(2.4)

which is automatically satisfied due to our choice of boundary conditions.

2.1 Weak Formulation

To proceed with the finite element discretization, we need a weak formulation of the Stokes problem. The first step in developing the weak formulation involves multiplying eqs. (2.1) and (2.2) by test functions $\vec{v}$ and $q$ chosen from appropriately selected spaces. Subsequently, to relax the regularity requirements on the solution $(\vec{u}, p)$, we integrate these equations over the domain and transfer the derivatives onto the test functions using integration by parts. We then end up with the following weak formulation (John 2016, sec. 4.1; Elman et al. 2014, sec. 3.2):

Given $\vec{f} \in H^{-1}(\Omega)$, find $(\vec{u}, p) \in H^1_0(\Omega) \times L^2_0(\Omega)$ such that

\[
\begin{align*}
\int_{\Omega} 2\mu \epsilon (\vec{u}) : \epsilon (\vec{v}) - \int_{\Omega} p \nabla \cdot \vec{v} &= \int_{\Omega} \vec{f} \cdot \vec{v} \quad \forall \vec{v} \in H^1_0(\Omega) \\
\int_{\Omega} \nabla \cdot \vec{u} &= 0 \quad \forall q \in L^2_0(\Omega)
\end{align*}
\]  

(2.5)

where

\[
\epsilon (\vec{u}) : \epsilon (\vec{v}) = \text{componentwise scalar product of } \epsilon (\vec{u}) \text{ and } \epsilon (\vec{v})
\]

\[
\begin{align*}
H^1_0(\Omega) &= \{ \vec{v} \in H^1(\Omega) \mid \vec{v} = \vec{0} \text{ on } \partial \Omega \} \\
L^2_0(\Omega) &= \{ q \in L^2(\Omega) \mid \int_{\Omega} q = 0 \} \\
H^{-1}(\Omega) &= \left( H^1_0(\Omega) \right)' \text{, the dual of } H^1_0(\Omega)
\end{align*}
\]

and the corresponding inner products, along with the induced norms, are given by:

\[
\begin{align*}
(\vec{u}, \vec{v})_{H^1_0(\Omega)} &= \int_{\Omega} \nabla \vec{u} : \nabla \vec{v}, \quad \lVert \vec{u} \rVert_{H^1_0(\Omega)} = \left( (\vec{u}, \vec{u})_{H^1_0(\Omega)} \right)^{\frac{1}{2}} \\
(p, q)_{L^2_0(\Omega)} &= \int_{\Omega} pq, \quad \lVert p \rVert_{L^2_0(\Omega)} = \left( (p, q)_{L^2_0(\Omega)} \right)^{\frac{1}{2}}
\end{align*}
\]  

(2.7)

(2.8)

Using the Korn’s (A.2) and the Poincaré’s inequalities (A.1), one can show that eq. (2.7) indeed defines an inner product and a norm.
2.2 Discretization Using Mixed Finite Elements

Before commencing with discretization, it is essential to consider the triangulation of the domain. Let \( \Omega \) be a polygonal subdomain of \( \mathbb{R}^d \) with \( d \in \{2, 3\} \). A triangulation \( T \) of \( \Omega \) is a set of simplices (triangles, tetrahedrons) \( \{K_i\}_{i=1}^N \) with the following properties:

1. \( \cup K_i = \bar{\Omega} \), the closure of \( \Omega \)
2. For \( i \neq j \), the intersection \( K_i \cap K_j \) is either:
   (a) empty
   (b) a common vertex
   (c) a common edge
   (d) a common face (in 3D).

The finite element method discretizes partial differential equations (PDEs) by approximating continuous spaces with finite-dimensional function spaces. These spaces consist of functions that are typically polynomials when restricted to each element of the triangulation. Consider \( V^h(\Omega) \subset H_0^1(\Omega) \) and \( Q^h(\Omega) \subset L_0^2(\Omega) \) as the finite element spaces for approximating the velocity and pressure fields. Using different finite-dimensional spaces for velocity and pressure gives rise to the nomenclature of mixed finite elements. Since the discrete spaces are a subset of the continuous ones, we achieve a conforming discretization. The discrete Stokes problem then reads as follows:

Given \( \vec{f} \), find \( (\vec{u}_h, p_h) \) such that

\[
a(\vec{u}_h, \vec{v}_h) + b(p_h, \vec{v}_h) = \sum_{K \in T} \int_K \vec{f} \cdot \vec{v}_h, \quad \forall \vec{v}_h \in V^h(\Omega) \tag{2.9}
\]

\[
b(q_h, \vec{u}_h) = 0, \quad \forall q_h \in Q^h(\Omega) \tag{2.10}
\]

with

\[
a(\vec{u}_h, \vec{v}_h) = \sum_{K \in T} \int_K 2 \mu \epsilon(\vec{u}_h) : \epsilon(\vec{v}_h),
\]

\[
b(q_h, \vec{v}_h) = -\sum_{K \in T} \int_K q_h \nabla \cdot \vec{v}_h
\]

where \( \vec{f} \) is assumed to be sufficiently smooth such that the right-hand side of eq. (2.9) is well defined. Proof for the existence, uniqueness, and stability of the finite element solution \((\vec{u}_h, p_h)\) can be found in (John 2016, sec. 4.2). In particular, the spaces \( V^h(\Omega) \) and \( Q^h(\Omega) \) must satisfy the discrete inf-sup condition (John 2016, Remark 3.51):

\[
\inf_{q_h \neq 0} \sup_{\vec{v}_h \neq 0} \frac{|b(q_h, \vec{v}_h)|}{\|\vec{v}_h\|_{H^1(\Omega)} \|q_h\|_{L_0^2(\Omega)}} \geq \beta^h_{\text{inf}} > 0, \quad \forall \vec{v}_h \in V^h(\Omega) \text{ and } \forall q_h \in Q^h(\Omega) \tag{2.11}
\]

for eqs. (2.9) and (2.10) to be well-posed.

To formulate the linear algebra problem corresponding to the discrete Stokes equations in three dimensions, we define the discrete velocity (vector-valued) and pressure spaces using the following basis functions (without considering the boundary conditions):

\[
V^h(\Omega) := \text{span} \left\{ \phi_i \right\}_{i=1}^{3N_u}
\]

\[
= \text{span} \left\{ \begin{array}{c} \phi_1 \\ 0 \\ 0 \\ \phi_1 \\ 0 \\ 0 \\ \phi_1 \\ 0 \\ 0 \end{array} \right\}
\]

\[
Q^h(\Omega) := \text{span} \left\{ \psi_i \right\}_{i=1}^{N_p}
\]

where \( N_u \) is the number of DoFs in each velocity component and \( N_p \) is the number of pressure DoFs. The discrete velocity and pressure functions find a unique representation through these basis functions:

\[
\vec{u}_h = \sum_{i=1}^{3N_u} u_i \phi_i, \quad p_h = \sum_{i=1}^{N_p} p_i \psi_i \tag{2.14}
\]
with real coefficients $\mathbf{u} = \{u_1, u_2, \ldots, u_{3N_u}\}$ and $\mathbf{p} = \{p_1, p_2, \ldots, p_{3N_p}\}$. Note that the coefficients $u_j$ corresponding to the boundary nodes must have a zero value to account for the boundary conditions. Introducing (2.14) in eqs. (2.9) and (2.10) and testing with each basis function separately gives us the saddle-point system:

$$
\begin{bmatrix}
\mathbf{A} & \mathbf{B}^T \\
\mathbf{B} & 0
\end{bmatrix}
\begin{bmatrix}
\mathbf{u} \\
\mathbf{p}
\end{bmatrix}
=
\begin{bmatrix}
\mathbf{f} \\
\mathbf{0}
\end{bmatrix}
$$

(2.15)

where

$$(\mathbf{A})_{ij} = \sum_{K \in T} \int_K 2\mu \epsilon(\vec{\phi}_j) : \epsilon(\vec{\phi}_i), \quad i, j = 1, 2, \ldots, 3N_u$$

(2.16)

$$(\mathbf{B})_{ij} = -\sum_{K \in T} \int_K \psi_i \nabla \cdot \vec{\phi}_j, \quad i = 1, 2, \ldots, N_p, \; j = 1, 2, \ldots, 3N_u$$

(2.17)

$$(\mathbf{f})_i = \sum_{K \in T} \int_K \vec{f} \cdot \vec{\phi}_i, \quad i = 1, 2, \ldots, 3N_u.$$  

(2.18)

### 2.3 The Taylor-Hood Element

The Taylor-Hood finite element, denoted as $P_2-P_1$, is prevalent in the numerical simulations of incompressible fluid flows, especially in the context of the Stokes equations. It finds extensive use in modeling phenomena such as mantle convection and ice sheet evolutions. Before describing the Taylor-Hood element, a discussion on the general polynomial space and Lagrange finite elements is necessary. The polynomial space $P_k$, for $\mathbf{x} = (x_1, x_2, \ldots, x_d)^T$, $k \in \mathbb{N} \cup 0$ and $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_d)^T$ is defined as:

$$P_k = \text{span} \left\{ \prod_{i=1}^d x_i^{\alpha_i} \left| \alpha_i \in \mathbb{N} \cup 0, \sum_{i=1}^d \alpha_i \leq k \right. \right\}$$

(2.19)

and the $P_k$ Lagrange finite element is characterized by:

1. The simplex element domain $K$
2. The space of shape functions $P_k$ defined on $K$
3. The set of basis functions $\phi_i \in P_k$ satisfying

$$
\phi_i(\mathbf{x}_j) = \begin{cases} 
1 & i = j \\
0 & i \neq j 
\end{cases}
$$

(2.20)

for node $\mathbf{x}_j \in K$.

Figure 2.1: $P_1$ global basis function in 2D.

The $P_2-P_1$ Taylor-Hood discretization approximates the components of the velocity space using $P_2$ elements and the pressure space using $P_1$ elements. Figure 2.2 illustrates the location of the nodes corresponding to velocity and pressure DoFs for the Taylor-Hood elements.
Remark 2.1 When using $P_1$ elements to approximate the pressure solution $p$, $p_h$ does not necessarily belong to $L^2_0(\Omega)$. The imposition of $p_h \in L^2_0(\Omega)$ involves additional constraints in eq. (2.15) and is not a common practice. Instead, $L^2(\Omega)$ space is chosen for $p_h$ and $q_h$ in the discrete weak formulation, in contrast to $L^2_0(\Omega)$. The change in the test function space poses no issues as $L^2_0(\Omega) \subset L^2(\Omega)$.

In this scenario, the finite element pressure solution is unique up to a constant. However, the Stokes weak formulation (2.5) and (2.6) remains well-posed. This is facilitated by employing the quotient space norm $\|q\|_{0,\Omega} = \|q - (1/|\Omega|) \int_\Omega q\|_{L^2(\Omega)}$ (Elman et al. 2014, sec. 3.2) in the finite element pressure space. The discrete inf-sup condition then transforms to:

$$\inf_{q_h \neq \text{constant}} \sup_{\vec{v}_h \neq \vec{0}} \frac{|b(q_h, \vec{v}_h)|}{\|q_h\|_{0,\Omega} \|\vec{v}_h\|_{H^1_0(\Omega)}} \geq \beta_{\text{is}} > 0 $$  \hspace{1cm} (2.21)

and the results for well-posedness can be verified to remain valid.

The following theorems outline the inf-sup stability of the $P_2-P_1$ discretization:

**Theorem 2.1** Let $\Omega \in \mathbb{R}^2$ be a polygonal domain and let $\mathcal{T}$ be a regular triangulation of $\Omega$. The $P_2-P_1$ pair satisfies the discrete inf-sup condition when $\mathcal{T}$ includes at least three triangles (John 2016, Theorem 3.128).

**Theorem 2.2** Let $\Omega \in \mathbb{R}^3$ be a polygonal domain and let $\mathcal{T}$ be a regular triangulation of $\Omega$. It is assumed that every element of $\mathcal{T}$ has at least one interior vertex. Then the $P_2-P_1$ pair satisfies the discrete inf-sup condition (John 2016, Theorem 3.129).

The Taylor-Hood element is computationally expensive compared to other inf-sup stable elements with fewer degrees of freedom. However, it compensates for this drawback through its favorable convergence properties. The finite element error estimates for the $P_2-P_1$ discretization can be found in (John 2016, corollary 4.30) and for a sufficiently regular solution of the Stokes problem (2.1) and (2.2), i.e, $\vec{u} \in (H^3(\Omega)^d \cap H^1_0(\Omega))$ and $p \in (H^2(\Omega) \cap L^2_0(\Omega))$:

$$\|\vec{u} - \vec{u}_h\|_{L^2(\Omega)} \leq Ch^3 \left( \|\vec{u}\|_{H^3(\Omega)^d} + \|p\|_{H^2(\Omega)} \right)$$ \hspace{1cm} (2.22)

$$\|p - p_h\|_{L^2(\Omega)} \leq Ch^2 \left( \|\vec{u}\|_{H^3(\Omega)^d} + \|p\|_{H^2(\Omega)} \right)$$ \hspace{1cm} (2.23)

where $h$ represents the characteristic length of the elements in the triangulation.

### 2.4 Properties of the Stokes Saddle-point Matrix

Analogous to (John 2016, Remark 4.69) and (John 2016, Lemma 4.67), one can show that the viscous block $A$ in the Stokes system of equations (2.15), is symmetric, positive definite, and possesses the
The viscous block in question differs from that of a Stokes system with the momentum equation given by:

$$-\nabla \cdot (\mu \nabla \vec{u}) + \nabla p = \vec{f}$$  \hspace{1cm} (2.24)$$

in that it possesses non-zero off-diagonal blocks. Due to the symmetry of $A$, the Stokes saddle-point matrix $A$ is symmetric. Moreover, $A$ is indefinite (John 2016, lemma 4.68), with $3N_u$ positive eigenvalues. We can observe this fact using the following congruence transform:

$$[\begin{array}{c|c}
A & B^T \\
\hline
B & 0
\end{array}] = [\begin{array}{c|c}
I & 0 \\
\hline
BA^{-1} & I
\end{array}] [\begin{array}{c|c}
A & 0 \\
\hline
0 & -BA^{-1}B^T
\end{array}] [\begin{array}{c|c}
I & A^{-1}B^T \\
\hline
0 & I
\end{array}]$$  \hspace{1cm} (2.25)$$

and applying Sylvester’s law of inertia. When looking for the pressure solution in $L^2(\Omega)$ (see Remark 2.1), the $P_2$-$P_1$ discretization ensures that $\text{null}(B^T) = \{1\}$ (Elman et al. 2014, section 3.3), faithfully representing the underlying physical problem. Consequently, $A$ possesses $N_p - 1$ negative eigenvalues with $u = 0$ and $p = 1$ in its kernel.
3 Solution Techniques for Discrete Stokes Systems

This segment focuses on techniques for solving the discrete Stokes system of equations. We start with an introduction to Krylov solvers, a vital component in matrix-free finite element frameworks. After that, we delve into preconditioning and introduce the w-BFBT approximation.

3.1 Krylov Subspace Methods

Krylov solvers are widely used iterative methods for solving large, sparse systems of equations encountered in various scientific domains. For the theory and algorithms implementing these methods, refer to (Trefethen et al. 1997; Saad 2003). When solving systems of the form $Ax = b$, Krylov solvers only require the action of $A$ on a vector $y$, enabling implementations that treat $A$ as a black box ($\text{HyTeG}$ operators in our case) with the following action:

$$y \rightarrow \text{black box} \rightarrow Ay.$$

This characteristic makes Krylov solvers an essential part of matrix-free frameworks. Before delving into Krylov solvers, exploring the underlying notion of Krylov subspaces is essential. The Krylov subspace of dimension $k$ associated with $A$ and $b$, denoted as $K_k(A,b)$, is defined as:

$$K_k(A,b) := \text{span}\{b, Ab, A^2b, \ldots, A^{k-1}b\}. \quad (3.1)$$

Constructing such a space is straightforward when utilizing the black box for matrix $A$, and one can verify the inclusion relation:

$$K_k(A,b) \subseteq K_{k+1}(A,b). \quad (3.2)$$

For an initial guess $x^{(0)} = 0$, the main idea behind Krylov solvers is to construct an iterative scheme such that $x^{(k)} \rightarrow x$ as $k \rightarrow \infty$, where $x^{(k)} \in K_k(A,b)$ is the "best" approximation of $x$ within the Krylov subspace $K_k(A,b)$. In a more formal sense:

$$x^{(k)} = \arg \min_{z \in K_k(A,b)} \|Az - b\|_P \quad (3.3)$$

for some chosen norm $\|\cdot\|_P$.

Remark 3.1 When starting with a non-zero initial guess $x^{(0)} = x_0$, we look for $x^{(k)} \in x_0 + K_k(A,r^{(0)})$, where $r^{(0)} = b - Ax$ is the initial residual.

An intuitive justification for using the Krylov subspace as a search space emerges from the Cayley-Hamilton theorem: every square matrix satisfies its own characteristic polynomial $c_A(z) = \prod_{i=1}^n (\lambda_i - z)$, where $\lambda_i$ is the $i$th eigenvalue of $A$. Bringing the characteristic polynomial into the following form:

$$c_A(A) = \alpha_n A^n + \alpha_{n-1} A^{n-1} + \cdots + \alpha_0 I = 0 \quad (3.4)$$

and multiplying it with $A^{-1}$ gives us:

$$A^{-1} = -\frac{1}{\alpha_0}(\alpha_n A^{n-1} + \alpha_{n-1} A^{n-2} + \cdots + \alpha_1 I). \quad (3.5)$$

We can now see that $x = A^{-1}b$ lies in the Krylov subspace $K_n(A,b)$.

In the $k$th iteration, implementations of Krylov solvers typically involve progressively generating an orthonormal basis for the Krylov subspace $K_k(A,b)$ (at least implicitly) through recurrence relations (Benzi et al. 2005, sec. 9.3). The Lanczos method (Trefethen et al. 1997, algorithm 36.1) achieves this objective using a three-term recurrence relation when the coefficient matrix $A$ is symmetric. In the case of a non-symmetric $A$, the Arnoldi method (Trefethen et al. 1997, algorithm 33.1) is employed, which utilizes a full-recurrence relation, resulting in increased memory requirements. Some of the frequently employed Krylov subspace solvers are listed as follows:

1. The conjugate gradient method (CG)
2. The minimal residual method (MINRES)
3. The generalized minimal residual method (GMRES)

and Table 3.1 summarizes the characteristics of these solvers to provide a concise overview.
3.2 Convergence of Krylov Solvers

When investigating Krylov solvers, a key consideration involves understanding their convergence behavior. Analyzing the convergence of these solvers not only sets expectations for their usage but also provides valuable insights into strategies for enhancing their performance. In (Liesen et al. 2004), the authors offer a thorough survey on this topic, and we draw relevant ideas from it to motivate the concept of preconditioning.

As mentioned in Remark 3.1, when starting with a non-zero initial guess $x_0$, the $k$th iterate $x^{(k)} \in x_0 + \mathbb{K}_k(A,r(0))$. This fact allows us to write the error for the $k$th iteration in the following polynomial form (Elman et al. 2014, sec. 2.1; Liesen et al. 2004, sec. 2):

$$x - x^{(k)} = p_k(A)(x - x_0), \quad p_k \in \pi_k \quad (3.6)$$

where $\pi_k$ is the set of polynomials of degree at most $k$ with a value of one at the origin. When the coefficient matrix $A$ is symmetric, there exists an eigendecomposition such that:

$$A = Q\Lambda Q^T, \quad QQ^T = I, \quad \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n) \quad (3.7)$$

and this decomposition simplifies eq. (3.6) to:

$$x - x^{(k)} = Qp_k(\Lambda)Q^T(x - x_0). \quad (3.8)$$

We now shift our focus to CG iterations (see Table 3.1) where the iterate $x^{(k)}$ satisfies:

$$\|x - x^{(k)}\|_A = \min_{p \in \pi_k} \|p(A)(x - x_0)\|_A. \quad (3.9)$$

Using eqs. (3.8) and (3.9), we can obtain the following convergence bound for the CG method through simple algebraic manipulations (Liesen et al. 2004, sec. 3.1; Trefethen et al. 1997, Theorem. 38.3):

$$\frac{\|x - x^{(k)}\|_A}{\|x - x_0\|_A} \leq \min_{p \in \pi_k} \max_i |p(\lambda_i)|. \quad (3.10)$$

This bound completely describes the worst-case behavior of the CG method and suggests that the convergence behavior depends on the spectrum of $A$. Generally, clustered eigenvalues imply faster convergence as a lower-order minimizing polynomial can effectively diminish the bound in eq. (3.10) (see Figure 3.1). This observation justifies using preconditioners to cluster the spectrum of $A$, and (Alger 2019, sec. 3.3) presents a similar discussion for MINRES.

Remark 3.2 Estimating the convergence rate of the CG method involves employing the following bound when estimates for $\lambda_{\text{max}}$ and $\lambda_{\text{min}}$ of $A$ are known (Trefethen et al. 1997, Theorem 38.5):

$$\frac{\|x - x^{(k)}\|_A}{\|x - x_0\|_A} \leq 2 \left( \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k \quad (3.11)$$

where $\kappa = \lambda_{\text{max}}/\lambda_{\text{min}}$ is the condition number of $A$. Equation 3.11 often represents an overestimation as it fails to consider the eigenvalue distribution, portraying the worst-case scenario for all distributions within the interval $[\lambda_{\text{min}}, \lambda_{\text{max}}]$. It is important to note that $\kappa \gg 1$ does not necessarily imply slower convergence; in contrast, a smaller $\kappa$ suggests better convergence.
3.3 Preconditioning the Stokes System

The central objective of preconditioning is to identify an operator $P$ (specifically, the action of $P^{-1}$) which ensures a significant reduction in the condition number of the original system, i.e., $\kappa(P^{-1}A) \ll \kappa(A)$. Intuitively, one can envision solving the modified system $P^{-1}Ax = P^{-1}b$, for which Krylov solvers exhibit improved convergence compared to the original system $Ax = b$. Preconditioners are vital to alleviate the computational workload when tackling large, ill-conditioned ($\kappa(A) \gg 1$) systems and an ideal preconditioner $P$ possesses the following properties:

1. The action of $P^{-1}$ on $A$ should induce a clustering effect on the eigenvalues of $A$
2. The computational cost associated with the action of $P^{-1}$ should be minimal.

For more details about the principles of preconditioning and possible approaches, refer to (Elman et al. 2014, sec. 2.2).

**Remark 3.3** It is well-known that the condition number of linear systems arising from finite element discretizations typically increases with mesh refinement (Zienkiewicz et al. 2005; Eisenträger et al. 2020). In this context, we aim to ensure mesh-independent convergence by seeking preconditioners that are spectrally equivalent to the coefficient matrix $A$.

Consider a symmetric positive definite operator $A$ resulting from finite element discretization of a partial differential operator on a grid with mesh size $h$. Then, a symmetric positive definite operator $P$ is said to be spectrally equivalent to $A$ if there exist constants $\alpha$ and $\beta$ independent of $h$, such that:

$$0 < \alpha \leq \frac{y^\top Ay}{y^\top Py} \leq \beta \quad \forall y \neq 0.$$  \hspace{1cm} (3.12)

Such a spectrally equivalent preconditioner ensures that the condition number of the preconditioned system does not deteriorate with mesh refinement.

As mentioned in Table 3.1, both CG and MINRES require a symmetric positive definite preconditioner. This requirement arises because the preconditioned coefficient matrix must adhere to the specifications of these solvers, i.e., the preconditioned system should be symmetric positive definite for CG and symmetric for MINRES. For more details on how a symmetric positive definite preconditioner meets these requirements, refer to (Elman et al. 2014, sec. 4.1).

Until now, our discussions have focused solely on solving non-singular systems. However, as discussed in subsection 2.4, the Stokes saddle-point system represented by eq. (2.15) is singular. Krylov solvers can still converge to one of the infinitely many solutions of such singular systems, provided the systems are consistent, i.e., the right-hand side $b \in \text{range}(A)$ (Elman et al. 2014, sec 2.3). When solving the Stokes saddle-point system, we employ the preconditioned MINRES.
solving (Elman et al. 2014, Algorithm 4.1) because $\mathbf{A}$ is symmetric and indefinite. In this case, the solutions to which the solver can converge differ by a constant value in the pressure component. The convergence of Krylov solvers for singular systems can be established (in exact arithmetic) by splitting the algorithms into the range and null space components (Hayami et al. 2021). This approach also reveals that the convergence rates depend solely on the non-zero eigenvalues. In (Hong et al. 2022), the effectiveness of singular preconditioners when solving singular systems using MINRES has been demonstrated, and one can find similar discussions for the preconditioned CG method (Elman et al. 2014, Algorithm 2.2) in (Kaasschieter 1988).

Now, the question remains: How do we precondition the Stokes system? When constructing a preconditioner for the Stokes saddle-point system, it is vital to consider the block structure of $\mathbf{A}$ and the properties of its constituent blocks (Elman et al. 2014, sec. 4.2). Let us consider the following block-diagonal preconditioner:

$$
\mathbf{P} = \begin{bmatrix} \mathbf{A} & 0 \\ 0 & \mathbf{S} \end{bmatrix} 
$$

(3.13)

where $\mathbf{S} = \mathbf{B} \mathbf{A}^{-1} \mathbf{B}^\top$ is the Schur complement of $\mathbf{A}$. When non-trivial Neumann and Dirichlet boundaries are present, both $\mathbf{A}$ and $\mathbf{P}$ are non-singular, and $\mathbf{P}^{-1} \mathbf{A}$ has only three distinct eigenvalues: $1, 1/2 \pm \sqrt{3}/2$ (Elman et al. 2014, sec. 4.2). In this case, a cubic polynomial with these eigenvalues as the roots exists, and consequently, MINRES will converge to the exact solution after three iterations. While $\mathbf{P}$ is an "ideal" preconditioner for $\mathbf{A}$, it is an impractical choice because $\mathbf{P}^{-1}$ requires the expensive action of $\mathbf{A}^{-1}$ and $\mathbf{S}^{-1}$. In practice, we replace $\mathbf{P}^{-1}$ with an approximation $\hat{\mathbf{P}}^{-1}$:

$$
\mathbf{P}^{-1} \approx \hat{\mathbf{P}}^{-1} = \begin{bmatrix} \hat{\mathbf{A}}^{-1} & 0 \\ 0 & \hat{\mathbf{S}}^{-1} \end{bmatrix}, \text{ where } \hat{\mathbf{A}}^{-1} \approx \mathbf{A}^{-1} \text{ and } \hat{\mathbf{S}}^{-1} \approx \mathbf{S}^{-1}. 
$$

(3.14)

MINRES may not converge in three iterations for this choice of preconditioner, but if $\hat{\mathbf{A}}^{-1}$ and $\hat{\mathbf{S}}^{-1}$ are sufficiently good approximations, the preconditioned system will exhibit improved convergence. Refer to (Benzi et al. 2005, sec. 10) for a comprehensive exploration of additional preconditioning strategies that can be employed when solving the Stokes system. Appropriate multigrid cycles can achieve the approximate action of $\mathbf{A}^{-1}$, and in this work, our primary focus is on obtaining a reliable approximation of $\mathbf{S}^{-1}$.

### 3.4 The w-BFBT Approximation for Inverse Schur Complement

As seen in subsection 3.3, a good approximation of the inverse Schur complement is necessary to precondition the Stokes system effectively. The inverse viscosity-weighted mass matrix in the pressure space, denoted as $\mathbf{M}_p(1/\mu)$, is widely employed for approximating the Schur complement of Stokes systems with variable viscosity functions (Kronbichler et al. 2012; May et al. 2008; Rudi et al. 2016). The entries of $\mathbf{M}_p(1/\mu)$ are given by:

$$
(\mathbf{M}_p(1/\mu))_{ij} = \int_{\Omega} \mu^{-1} \Psi_i \Psi_j 
$$

(3.15)

where $\Psi_i$ and $\Psi_j$ represent the global basis functions of the finite element space used for approximating the pressure space. To avoid solving systems when applying $\mathbf{M}_p(1/\mu)^{-1}$ during preconditioning, $\mathbf{M}_p(1/\mu)$ is usually diagonalized using lumping schemes (Rudi et al. 2016, sec. 1.3). When using $\mathbb{P}_1$ elements, row-sum lumping offers a good diagonal approximation of $\mathbf{M}_p(1/\mu)$. For a $n \times n$ lumped approximation $\tilde{\mathbf{M}}_p(1/\mu)$, this is formally expressed as:

$$
(\tilde{\mathbf{M}}_p(1/\mu))_{ii} = \sum_{j=1}^{n} (\mathbf{M}_p(1/\mu))_{ij}, \text{ for } i = 1 \ldots n. 
$$

(3.16)

However, $\mathbf{M}_p(1/\mu)$ fails to perform well as an approximation of $\mathbf{S}$ when the viscosity exhibits severe heterogeneity (Rudi et al. 2016, sec. 1.4). For this reason, the w-BFBT approximation for $\mathbf{S}^{-1}$ was proposed in (Rudi et al. 2016) as a better alternative to $\mathbf{M}_p(1/\mu)$.

A comprehensive introduction to the BFBT preconditioner, also commonly referred to as the Least-squares Commutator (LSC) preconditioner, can be found in (Elman et al. 2014, sec. 9.2).
We present the BFBT preconditioner and the weighting schemes as outlined in (Rudi et al. 2016). When implementing the BFBT preconditioner, the primary objective is to determine a matrix $X$ such that the following commutation vanishes, i.e., results in a null matrix:

$$\mathbf{A} \mathbf{D}^{-1} \mathbf{B}^\top - \mathbf{B}^\top \mathbf{X} \approx 0 \quad (3.17)$$

for a given diagonal matrix $\mathbf{D}$. The motivation for seeking such near-commutator $X$ is presented below:

$$\mathbf{A} \mathbf{D}^{-1} \mathbf{B}^\top - \mathbf{B}^\top \mathbf{X} \approx 0$$
$$\Rightarrow \mathbf{B} \mathbf{D}^{-1} \mathbf{B}^\top - \mathbf{B} \mathbf{A}^{-1} \mathbf{B}^\top \mathbf{X} \approx 0$$
$$\Rightarrow \mathbf{B} \mathbf{D}^{-1} \mathbf{B}^\top \mathbf{X}^{-1} - \mathbf{B} \mathbf{A}^{-1} \mathbf{B}^\top \approx 0 \quad (\text{assuming } \mathbf{X}^{-1} \text{ exists})$$
$$\Rightarrow \mathbf{B} \mathbf{D}^{-1} \mathbf{B}^\top \mathbf{X}^{-1} \approx \mathbf{S}.$$ 

The objective of finding a vanishing commutator in eq. (3.17) can be reformulated as the following least-squares minimization problem:

Find matrix $\mathbf{X}$ such that

$$\mathbf{X} = \arg \min_{\mathbf{Z}} \| \mathbf{A} \mathbf{D}^{-1} \mathbf{B}^\top \mathbf{e}_i - \mathbf{B}^\top \mathbf{Z} \mathbf{e}_i \|_{\mathbf{C}^{-1}}^2 \quad \forall i \quad (3.18)$$

where $\mathbf{e}_i$ denotes the $i$th Cartesian unit vector, and $\mathbf{C}$ is a symmetric positive definite matrix. The solution for eq. (3.18) is then given by $\mathbf{X} = (\mathbf{B} \mathbf{C}^{-1} \mathbf{B}^\top)^{-1} (\mathbf{B} \mathbf{C}^{-1} \mathbf{A} \mathbf{D}^{-1} \mathbf{B}^\top)$, from which one can derive the BFBT approximation as:

$$\tilde{\mathbf{S}}_{\text{BFBT}}^{-1} := (\mathbf{B} \mathbf{C}^{-1} \mathbf{B}^\top)^{-1}(\mathbf{B} \mathbf{C}^{-1} \mathbf{A} \mathbf{D}^{-1} \mathbf{B}^\top)(\mathbf{B} \mathbf{D}^{-1} \mathbf{B}^\top)^{-1}. \quad (3.19)$$

In (Elman et al. 2014, sec. 9.2), $\mathbf{C}$ and $\mathbf{D}$ were chosen to be the diagonal of the velocity space mass matrix $\mathbf{M}_u$. However, this choice does not capture the viscosity variations to meet our requirements (Rudi et al. 2016, sec. 1.3). Moreover, it is evident that applying the $\tilde{\mathbf{S}}_{\text{BFBT}}^{-1}$ is computationally expensive. Therefore, it is advisable to refrain from using any version of the BFBT approximation in simple applications where $\mathbf{M}_p(1/\mu)$ suffices for preconditioning.

In (Rudi et al. 2016, sec. 1.3), two different weighting schemes that enhance the $\tilde{\mathbf{S}}_{\text{BFBT}}^{-1}$ approximation were presented:

**diag(A)-BFBT**

In (May et al. 2008), incorporating entries from $\mathbf{A}$ into the weights for $\tilde{\mathbf{S}}_{\text{BFBT}}^{-1}$ has shown to improve the effectiveness of the preconditioner. When dealing with challenging applications, choosing $\mathbf{C} = \mathbf{D} = \text{diag}(\mathbf{A})$ results in an approximation superior to $\mathbf{M}_p(1/\mu)$. However, experiments in (Rudi et al. 2016, sec. 2.2) demonstrate that this scheme is unsatisfactory when using higher-order discretizations.

**w-BFBT**

Rudi et al. introduced the w-BFBT approximation as a robust alternative to $\mathbf{M}_p(1/\mu)$ and diag(A)-BFBT. This scheme utilizes $\mathbf{C} = \mathbf{D} = \bar{\mathbf{M}}_u(\sqrt{\mu})$, where $\bar{\mathbf{M}}_u(\sqrt{\mu})$ results from row-sum lumping of the $\sqrt{\mu}$-weighted mass matrix in the velocity space $\mathbf{M}_u(\sqrt{\mu})$. The resulting preconditioner shows only a mild dependency on the polynomial order in discretizations of the type $\mathcal{Q}_{k-1}^{h,k}$ for order $k \geq 2$ and significantly outperforms $\mathbf{M}_p(1/\mu)$ when confronted with high viscosity contrasts (Rudi et al. 2016, sec. 2.2).

We are specifically interested in the w-BFBT approximation because the w-BFBT operator is spectrally equivalent to the Schur complement (Rudi et al. 2016, Theorem 5). However, achieving mesh independence in practice requires damping the weights in $\mathbf{C}^{-1}$ and $\mathbf{D}^{-1}$ near the Dirichlet boundaries in an unsymmetric manner. Accomplishing such damping requires heuristic weight functions, and (Rudi et al. 2016, sec. 5) details this process.

Note that approximating the inverse action of the sub-operators in (Rudi et al. 2016, sec. 6), where $\mathcal{Q}_{k-1}^{h,k}$ discretization is employed, involves the following steps:

1. Regarding the sub-operators as discrete variable-coefficient Poisson operators with Neumann boundary conditions in $\mathcal{Q}_k$. 

18
2. Projecting the sub-systems from $P^{\text{disc}}_{k-1}$ to $Q_k$ and solving them using multigrid V-cycles.

3. Additional smoothing in $P^{\text{disc}}_{k-1}$ to address the high-frequency errors introduced after projecting the solutions from $Q_k$ to $P^{\text{disc}}_{k-1}$.

Here, it is essential to highlight that the coefficients in the Poisson operator depend on the weights used in the sub-operators, and implementing smoothing in the pressure space is not straightforward when dealing with matrix-free operators due to the lack of access to sub-operator entries.

### 3.5 Geometric Multigrid Methods: Brief Introduction

When preconditioning the Stokes systems, we often encounter Poisson-like systems that demand effective solution techniques to ensure the preconditioner is computationally inexpensive. In this context, multigrid methods leverage a hierarchy of discretizations to achieve optimal computational complexity. For a thorough understanding of the multigrid methods, refer to (Saad 2003, ch. 13; Elman et al. 2014, sec. 2.5). These methods exploit more information about the problem and demonstrate significant superiority over Krylov solvers. However, they may necessitate customizations specific to the particular physical problem.

Stationary iterative methods, such as weighted-Jacobi and Gauss-Seidel, are effective in damping high-frequency errors. However, they exhibit a slow reduction of low-frequency errors, leading to poor convergence rates. The core concept of multigrid methods involves transitioning to coarser grids, where these low-frequency errors resemble high-frequency errors and experience effective damping. Essential elements of the multigrid methods include:

**Hierarchy of Grids**
- Grids with progressively coarser resolutions and corresponding system operators.

**Smoothers**
- Relaxation methods to dampen high-frequency errors.

**Grid Transfer Operators**
- Restriction and prolongation operators facilitating information transfer between the grids.

**Cycle Strategies**
- Strategies like the V-cycle or W-cycle that dictate the sequence of operations.

**Coarse Grid Solver**
- Solver for systems of equations on the coarsest grid.

Figure 3.2 illustrates the steps in a single iteration (one V-cycle) of the two-grid algorithm, and the multigrid algorithm is a natural extension of these steps. Multigrid methods are particularly appealing due to their ability to achieve mesh-independent convergence when solving or preconditioning Poisson-like systems, provided appropriate multigrid cycles are employed (Elman et al. 2014, Remark 2.5; Bourne et al. 2022).

![Sequence of steps in a two-grid V-cycle.](image)

1. Apply smoothing steps
2. Compute residual: $r_{\text{fine}} = b - Ax$
3. Restrict residual: $r_{\text{fine}} \rightarrow r_{\text{coarse}}$
4. Coarse grid solve: $A e_{\text{coarse}} = r_{\text{coarse}}$
5. Prolongate error: $e_{\text{coarse}} \rightarrow e_{\text{fine}}$
6. Correction: $x = x + e_{\text{fine}}$
7. Apply smoothing steps

Figure 3.2: Sequence of steps in a two-grid V-cycle.
4 HyTeG Preliminaries

In this section, we provide a high-level overview of the HyTeG framework, intending to introduce the key elements necessary for understanding the matrix-free implementation of the w-BFBT operator. For a more in-depth exploration, refer to (Kohl et al. 2018).

4.1 Primitive Storage

A PrimitiveStorage represents the simulation domain and includes references to fundamental entities, such as vertices or edges, within the finite element grid. Marking different parts of the domain boundary by setting the corresponding MeshBoundaryFlags is necessary for incorporating boundary conditions when solving the Stokes system. The code segment presented in Listing 4.1 creates a 3D domain \( \Omega = (0,1)^3 \) with Dirichlet boundaries.

```cpp
std::shared_ptr<SetupPrimitiveStorage> setupStorage;
Point3D corner1(0, 0, 0);
Point3D corner2(1, 1, 1);
auto meshInfo3D = MeshInfo::meshSymmetricCuboid(corner1, corner2, 1, 1, 1);
// (corner1, corner2, nx, ny, nz)
setupStorage = std::make_shared<SetupPrimitiveStorage>(meshInfo3D, numProcesses);
setupStorage->setMeshBoundaryFlagsOnBoundary(1, 0, true);
// marks all the vertices on the boundary with a flag value of 1 (Dirichlet),
and flags the remaining vertices with a value of 0 (Inner)
auto storage = std::make_shared<PrimitiveStorage>(*setupStorage);
```

Listing 4.1: Creating a PrimitiveStorage with Dirichlet boundaries.

After setting up the domain, we look for a hierarchy of grids to work with geometric multigrid methods. HyTeG generates such a hierarchy through structured refinement of unstructured coarse meshes, as illustrated in Figure 4.1. The level parameter determines the refinement level at which we perform the current operations, and during multigrid operations, one works with levels from minLevel to maxLevel. This strategy enables the utilization of finite element stencils for most computations, which is particularly beneficial for problems characterized by at least patch-wise constant coefficients (Bergen et al. 2004).

![Figure 4.1: Structured refinement of an unstructured triangular mesh.](image)

4.2 HyTeG Functions

In HyTeG, a Function object represents scalar and vector fields on a grid by storing the degrees of freedom corresponding to a chosen finite element space. These Function objects also store information regarding the boundary conditions because we do not assemble the system matrices incorporating them. Listing 4.2 illustrates the incorporation of boundary conditions, while Listing 4.3 showcases some member functions of the Function class that facilitate algebraic operations.
Listing 4.2: Incorporating boundary conditions in Function objects.

// x, y, z : P2P1TaylorHoodFunction< real_t >
x.uvw().assign( {1.0, -2.0}, {y.uvw(), z.uvw()}, level );
// x.uvw() = y.uvw() - 2*z.uvw()
x.p().multElementwise( {y.p(), z.p()}, level, All );
// x.p() = y.p() .* z.p(), entrywise multiplication for all DoFs

Listing 4.3: Algebraic operations on Function objects.

4.3 Elementwise Operators

Matrix-vector multiplications in HyTeG are realized without the explicit assembly of finite element matrices through the action of a corresponding Operator on an appropriate source Function object. HyTeG provides elementwise and pointwise operators for this purpose, with our work primarily utilizing the former. The ElementwiseOperator, as implied by its name, achieves its action by iterating over all elements in a grid. Conceptually, it implicitly integrates the matrix assembly into matrix-vector multiplication, and this process is detailed as follows:

For each element

1. Formulate the local finite element matrix using the corresponding weak form
2. Read the element local DoFs from the source Function
3. Evaluate the local matrix-vector multiplication
4. Assemble the local contributions into the destination Function.

An example usage of ElementwiseOperator is illustrated in Listing 4.4. Objects of the class form bundle the weak forms and the quadrature rules for evaluating integrals during local matrix formulation. The HyTeG-Form-Generator (HFG) can be employed to generate the necessary kernels when a new form is required.

// mu: callback for variable coefficient mass operator
P2ElementwiseOperator< forms::p2_k_mass_affine_q4 > KMassOp( storage, minLevel, maxLevel, forms::p2_k_mass_affine_q4( mu, mu ) );
// x, b : P2Function< real_t >
KMassOp.apply( x, b, level, Inner );
// (source, destination, level, flag), only the inner DoFs of b are updated

Listing 4.4: Example usage of an ElementwiseOperator.
5 Matrix-free Implementation of Weighted BFBT Operator

This section details the matrix-free implementation of the w-BFBT operator. We begin by addressing the incorporation of Dirichlet boundary conditions in the Stokes saddle-point system, a critical step in ensuring correctness. Then, we delve into the intricate components of the w-BFBT operator, focusing on the weights and sub-systems. Finally, we elucidate how these components intertwine to form the w-BFBT operator, providing a comprehensive understanding of its matrix-free implementation.

5.1 Incorporating Homogeneous Dirichlet Boundary Conditions

Most finite element frameworks typically disregard Dirichlet boundary conditions during the assembly of the global system of equations. Consequently, one must correct the resulting system for rows corresponding to the DoFs on the Dirichlet boundary. This approach prioritizes flexibility in the assembly process, enabling the incorporation of various possible boundary conditions. For systems with explicitly assembled global matrices, the following steps outline the implementation of homogeneous Dirichlet boundary conditions:

1. Identifying rows in the global system corresponding to DoFs on the Dirichlet boundary and setting them to zero.
2. Identifying columns in the global system matrix corresponding to DoFs on the Dirichlet boundary and setting them to zero.
3. Setting the diagonal entries in the global system matrix for the identified rows to one.

Remark 5.1 Zeroing out columns in the system matrix is necessary to maintain symmetry, which allows the use of solvers like CG or MINRES. In the case of non-homogeneous Dirichlet boundary conditions, additional adjustments in the right-hand side vector are necessary to accommodate this change.

Figure 5.1 illustrates how the blocks in the Stokes saddle-point matrix are modified when implementing homogeneous Dirichlet boundary conditions. One must reflect such modifications when applying the matrix-free operators corresponding to these blocks in the w-BFBT operator. The conditions that must be satisfied for the correct implementation of boundary conditions, keeping in mind a matrix-free setting, are listed as follows:

![Figure 5.1: Modifications in the Stokes saddle-point system for a DoF on homogeneous Dirichlet boundary.](image-url)
When applying $A$, the DoFs corresponding to the Dirichlet boundary in the destination Function should receive the same values as those in the source Function.

After applying $B^\top$, the DoFs corresponding to the Dirichlet boundary in the destination Function should have a zero value.

When applying $B$, the DoFs corresponding to the Dirichlet boundary in the source Function should be treated as having a zero value.

Furthermore, particular caution concerning domain triangulation must be exercised in 3D when using $P_2-P_1$ elements. Ensuring every element has at least one DoF in the interior is crucial for inf-sup stability (John 2016, Theorem 3.129). It is easy to verify that in the presence of elements with all nodes on the Dirichlet boundary, as in Figure 5.2a, $\text{null}(B^\top) \neq \{1\}$ after applying the boundary conditions.

![Diagram of $P_2$ elements](image)

(a) $P_2$ element with all DoFs on the boundary
(b) An alternative arrangement ensuring that each $P_2$ element has at least one DoF in the interior

Figure 5.2: $P_2$ tetrahedral elements at grid corners.

### 5.2 w-BFBT Weights

As discussed in subsection 3.4, obtaining weights for the w-BFBT operator involves row-sum lumping of the $\sqrt{\mu}$-weighted velocity space mass matrix $M_u(\sqrt{\mu})$, followed by inversion of the resulting positive definite diagonal matrix. However, row-sum lumping results in a singular matrix when using $P_2$ elements for the velocity space. Due to this limitation, we explore alternative methods to formulate a diagonal approximation of $M_u(\sqrt{\mu})$. Before discussing these diagonalizing schemes, note that we employ objects of $P2VectorFunction$ to store the weights for the w-BFBT operator. In this case, simply multiplying the source Function with the weights in an entry-wise manner gives us the action of the weighting matrix. We now describe the lumping schemes used in this work:

**HRZ Lumping**

The HRZ (Hinton-Rock-Zienkiewicz) lumping scheme (Zienkiewicz et al. 2005, sec. 16.2.4) uses the scaled diagonal of $M_u(\sqrt{\mu})$ for deriving the entries in the lumped mass operator to conserve the "mass" of the system. If $M_{total}$ is the sum of all entries in $M_u(\sqrt{\mu})$, then the entries of the lumped mass operator are given by:

$$
(\tilde{M}_{HRZ})_{ii} = \frac{M_{total}}{\text{Trace}(M_u(\sqrt{\mu}))} \times (M_u(\sqrt{\mu}))_{ii}.
$$

(5.1)
**P₁ Row-Sum Lumping at Higher Level**

When working on the same grid, one can identify the P₁ DOFs at level +1 with the P₂ DOFs at level. Leveraging this fact, we obtain a diagonal approximation for $M_u(\sqrt{\mu})$ through the following steps:

1. Row-sum lumping of the weighted P₁ mass operator at level +1.
2. Storing the diagonal entries of the lumped mass operator in a P₁VectorFunction.
3. Converting the P₁VectorFunction at level +1 to a P₂VectorFunction at level.

**Node-Based Quadrature Rules**

This scheme achieves the mass lumping effect by employing quadrature rules that utilize nodes of the P₂ reference element as quadrature points. Evaluating $M_u(\sqrt{\mu})$ with such quadrature automatically results in a diagonal approximation. The motivation becomes apparent when formulating the local mass matrix over the reference element:

$$\begin{align*}
(M_{P_2})_{ij}^{local} &= \int_{K_{ref}} \phi_i \phi_j \approx \sum_{k=1}^{10} w_k \phi_i(\vec{n}_k)\phi_j(\vec{n}_k) = \begin{cases} 
    w_i & \text{if } i = j \\
    0 & \text{otherwise}
\end{cases} 
\end{align*}$$

where $\{\vec{n}_i\}_{i=1}^{10}$ are the nodes and $\{w_i\}_{i=1}^{10}$ are the corresponding weights (see Figure 5.3).

Consider the P₂ reference element depicted in Figure 5.3 to derive weights for node-based quadrature rules. The main idea behind these rules is to estimate integrals of functions over the reference element using the following approach:

$$\begin{align*}
\int_{K_{ref}} f(\vec{x}) &= \int_{0}^{1} \int_{0}^{1-z} \int_{0}^{1-z-y} f(\vec{x}) \, dx \, dy \, dz \\
&\approx \sum_{i=1}^{10} w_i f(\vec{n}_i).
\end{align*}$$

Integrating the basis of the P₂ polynomial space over $K_{ref}$ using eq. (5.3) and collecting the weights into a vector results in the following system of equations:

Figure 5.3: $P_2$ reference tetrahedron.
In eq. (5.4), the rows correspond to basis functions of the \( \mathbb{P}_2 \) polynomial space; the columns correspond to their evaluations at the nodes of \( K_{\text{ref}} \), and the right-hand side contains the results from exact integration of the basis functions over \( K_{\text{ref}} \). Solving the system (5.4) yields:

\[
\begin{align*}
\begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{pmatrix}
\begin{pmatrix}
f(\vec{n}_1) \\
f(\vec{n}_2) \\
f(\vec{n}_3) \\
f(\vec{n}_4) \\
f(\vec{n}_5) \\
f(\vec{n}_6) \\
f(\vec{n}_7) \\
f(\vec{n}_8) \\
f(\vec{n}_9) \\
f(\vec{n}_{10})
\end{pmatrix}
\begin{pmatrix}
w_1 \\
w_2 \\
w_3 \\
w_4 \\
w_5 \\
w_6 \\
w_7 \\
w_8 \\
w_9 \\
w_{10}
\end{pmatrix}
\end{align*}
\]

\[
= \int_{K_{\text{ref}}} f(\vec{x}) \begin{pmatrix}
1/6 \\
1/24 \\
1/24 \\
1/60 \\
1/60 \\
1/120 \\
1/120 \\
1/120
\end{pmatrix}^T.
\]

In eq. (5.4), the rows correspond to basis functions of the \( \mathbb{P}_2 \) polynomial space; the columns correspond to their evaluations at the nodes of \( K_{\text{ref}} \), and the right-hand side contains the results from exact integration of the basis functions over \( K_{\text{ref}} \). Solving the system (5.4) yields:

\[
\begin{align*}
w_1, w_2, w_3, w_4 &= -1/120 \\
w_5, w_6, w_7, w_8, w_9, w_{10} &= 1/30.
\end{align*}
\]

Although these weights result in a quadrature rule that is accurate up to quadratic functions, they cannot be used for diagonalizing the weighted mass operator because the negative weights render the lumped mass operator indefinite. If we restrict ourselves to quadratic rules that are accurate up to the first-order polynomial basis, then we have several possible combinations that result in positive weights because the system is underdetermined. Two such examples are shown in the table below:

<table>
<thead>
<tr>
<th>( \text{VBQ1} )</th>
<th>( w_1 )</th>
<th>( w_2 )</th>
<th>( w_3 )</th>
<th>( w_4 )</th>
<th>( w_5 )</th>
<th>( w_6 )</th>
<th>( w_7 )</th>
<th>( w_8 )</th>
<th>( w_9 )</th>
<th>( w_{10} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 1/60 )</td>
<td>( 1/60 )</td>
<td>( 1/60 )</td>
<td>( 1/60 )</td>
<td>( 1/60 )</td>
<td>( 1/60 )</td>
<td>( 1/60 )</td>
<td>( 1/60 )</td>
<td>( 1/60 )</td>
<td>( 1/60 )</td>
<td></td>
</tr>
</tbody>
</table>

| \( \text{VBQ2} \) | \( 1/240 \) | \( 1/240 \) | \( 1/240 \) | \( 1/240 \) | \( 1/40 \) | \( 1/40 \) | \( 1/40 \) | \( 1/40 \) | \( 1/40 \) | \( 1/40 \) |

Table 5.1: Possible weights for 3D node-based quadrature rules that are accurate up to linear functions.

**Remark 5.2** *In the case of 2D \( \mathbb{P}_2 \) elements, following a similar procedure leads to zero weights at the element vertices. These weights are again undesirable because lumping with such weights results in a singular operator. The table Table 5.2 presents possible weights that are accurate up to linear functions in 2D.*

<table>
<thead>
<tr>
<th>( \text{VBQ1} )</th>
<th>( w_1 )</th>
<th>( w_2 )</th>
<th>( w_3 )</th>
<th>( w_4 )</th>
<th>( w_5 )</th>
<th>( w_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 1/12 )</td>
<td>( 1/12 )</td>
<td>( 1/12 )</td>
<td>( 1/12 )</td>
<td>( 1/12 )</td>
<td>( 1/12 )</td>
<td>( 1/12 )</td>
</tr>
</tbody>
</table>

| \( \text{VBQ2} \) | \( 1/36 \) | \( 1/36 \) | \( 1/36 \) | \( 5/36 \) | \( 5/36 \) | \( 5/36 \) |

Table 5.2: Possible weights for 2D node-based quadrature rules that are accurate up to linear functions. In the table, \( w_1, w_2, \) and \( w_3 \) correspond to the weights at the element vertices, and the remaining weights correspond to the midpoints of the element edges.

### 5.3 w-BFBT Sub-systems

The choice of sub-operators and the solver for approximating their inverse action (sub-solver) is crucial for forming an effective w-BFBT operator. Solving the sub-systems that appear in eq. (3.19) using Krylov solvers is expensive because the sub-operator’s action involves applying two operators, and multiplication with weights. Therefore, we explore substitute sub-operators that are computationally inexpensive to apply and provide the possibility of using a multigrid solver to have a
mesh-independent sub-system. Different sub-operators implemented as a part of this work for the \( P_2-P_1 \) discretization are discussed below:

**P2P1BKBTSubOperator**

The **P2P1BKBTSubOperator** implements the sub-operator that appears in eq. (3.19). Achieving the action of this sub-operator involves the application of \( B^\top \), followed by entry-wise multiplication with the weights, and finally, the application of \( B \), all while incorporating the boundary conditions as detailed in subsection 5.1. Implementing matrix-free smoothers for this sub-operator is problematic because extracting its diagonal without explicitly forming the matrix is not straightforward.

**P1DivKGradSubOperator**

One can relate the steps in applying the **P2P1BKBTSubOperator** with those of a variable-coefficient Poisson operator in the pressure space subjected to homogeneous Neumann boundary conditions. For this reason, the **P1DivKGradSubOperator**, which is simply the **P1ElementwiseOperator** corresponding to the variable-coefficient Poisson operator, is implemented as a possible substitute for the **P2P1BKBTSubOperator**. The **P1DivKGradSubOperator** and other similar sub-operators in **HyTeG** allow for weighted-Jacobi multigrid smoothers while being less expensive to apply when compared to the **P2P1BKBTSubOperator**.

**P2P1MixedSubOperator**

The **P2P1MixedSubOperator** is a combination of the previous two sub-operators in the sense that it has the same action as a **P2P1BKBTSubOperator** when applying and a **P1DivKGradSubOperator** when smoothing. Through this operator, we aim to solve the original sub-systems using multigrid methods in which the smoothers employ the **P1DivKGradSubOperator**.

**P2DivKGradP1SubOperator**

The **P2DivKGradP1SubOperator** is built on the same idea as the **P1DivKGradSubOperator** but employs a **P2ElementwiseOperator** at level \( -1 \). The application action of this sub-operator involves the following steps:

1. Converting the right-hand side of the sub-system from a **P1Function** at level to a **P2Function** at level \( -1 \).
2. Solving the \( P_2 \) sub-system at level \( -1 \).
3. Converting the solution obtained from a **P2Function** at level \( -1 \) to a **P1Function** at level.

A summary of these sub-operators is available in Figure 5.4. When using the **DivKGrad** sub-operators, a natural consideration is the choice of an appropriate coefficient. An examination of (Rudi et al. 2016, sec. 3) suggests that using \( 1/\sqrt{\mu} \) as the coefficient is a sensible option when the weights for the w-BFBT operator are obtained by lumping \( M_u(\sqrt{\mu}) \) and inverting the result.

When solving the sub-systems, we recognize that the sub-operators are singular with the constant vector in the null space. This singularity arises as a requirement for the inf-sup stability of the \( P_2-P_1 \) discretization and the homogeneous Neumann boundary conditions on the pressure space. Moreover, the sub-systems are ill-conditioned due to the viscosity contrasts and require preconditioning. Consequently, utilizing CG as a sub-solver (or coarse-grid solver in a multigrid sub-solver) demands additional attention. In the presence of such singularity, the norms of the residuals during CG iterations tend to diverge after an initial period of convergence (Kronbichler et al. 2012; Van der Vorst 2003; May et al. 2008). We adopt a procedure similar to that described in (May et al. 2008, sec. 2.4) to ensure that the iteration vectors are orthogonal to the null space:

1. The sub-system is made consistent by projecting the right-hand side onto the range of the sub-operator.
2. Following the application of the preconditioner during CG iterations, the null space is projected out from the result.
In the steps described, we employ the following projection operator:

\[ P_{\text{ns}p} = \left( I - \frac{1}{1+1} \right) \]  

which subtracts the mean value of the source Function from its entries. Similar projection operations are necessary after the restriction and prolongation operations during multigrid iterations.

Figure 5.4: An overview of w-BFBT sub-operators.
5.4 The w-BFBT Operator

Having discussed the crucial components of the w-BFBT operator, the next step is to detail how these elements come together to achieve the operator’s action. Algorithm 5.1 outlines the sequence of steps executed by the *apply* method of the matrix-free *BFBTOperator* for Stokes systems with homogeneous Dirichlet boundary conditions.

**Algorithm 5.1 apply action of the BFBTOperator**

\[
W_l \quad : \text{Left weighting Function} \\
W_r \quad : \text{Right weighting Function} \\
K_l \quad : \text{Left matrix-free SubOperator} \\
K_r \quad : \text{Right matrix-free SubOperator} \\
A \quad : \text{Matrix-free variable-viscosity EpsilonOperator} \\
B \quad : \text{Matrix-free DivOperator} \\
B^\top \quad : \text{Matrix-free DivTOperator} \\
v_1, v_2 \quad : \text{Velocity space temporary Functions} \\
p \quad : \text{Pressure space temporary Function} \\
src \quad : \text{source Function} \\
dst \quad : \text{destination Function} \\
\odot \quad : \text{Component-wise multiplication}
\]

**ensure:** \( v_1, v_2 \) incorporate homogeneous Dirichlet boundary conditions  
**ensure:** \( p \) incorporates pure Neumann boundary conditions  
**ensure:** \( \text{flag} \) member in the sub-solver is set to *Inner*

**procedure** APPLY\((src, dst)\)

\[
\text{solve } K_r p = src \\
v_1 = B^\top p \ (\text{flag} = \text{Inner} | \text{Neumann}) \\
v_2 = W_r \odot v_1 \ (\text{flag} = \text{Inner} | \text{Neumann}) \\
v_1 = Av_2 \ (\text{flag} = \text{Inner} | \text{Neumann}) \\
v_2 = W_l \odot v_1 \ (\text{flag} = \text{Inner} | \text{Neumann}) \\
p = Bv_2 \ (\text{flag} = \text{Inner}) \\
\text{solve } K_l dst = p
\]

**end procedure**

We distinguish between the left and right sub-operators and the left and right weighting Functions to accommodate unsymmetric damping of the weights (as in Rudi et al. 2016, sec. 5). As of commit 782cc5464877d2a66c742de59685c8ae0dbef8e, operators and weights implemented during our work are present in the `hyteg/operators/BFBTOperators` directory on the branch `aakash/wBFBT`. Additionally, note that the forms employing the node-based quadrature rules for mass lumping (Table 5.1 and Table 5.2) bear the names `p2_k_mass_lumped_vbq1` and `p2_k_mass_lumped_vbq2` and are located in the `hyteg/forms/form_hyteg_generated/p2` directory.
6 Numerical Experiments

6.1 The Multi-sinker Benchmark Problem

To evaluate the effectiveness of the w-BFBT operator, we require a benchmark scenario in which the resulting Stokes system is challenging and reflective of the preconditioner’s performance for mantle convection problems. In this regard, we employ the multi-sinker benchmark as described in (Rudi et al. 2016, sec. 2.1). The multi-sinker benchmark involves high-viscosity sinkers in a low-viscosity fluid forced downward due to gravity. These sinkers introduce heterogeneity in the viscosity and have high viscosity gradients at their boundaries. Details of the multi-sinker benchmark are presented below:

**Simulation Domain**

The simulation domain is a unit square ($\Omega = (0, 1)^2$) in 2D and a unit cube ($\Omega = (0, 1)^3$) in 3D, both subjected to homogeneous Dirichlet boundary conditions.

**Sinker Indicator Function**

The sinker indicator function, denoted as $\chi_n(\vec{x}) \in [0, 1]$, captures the location of the $n$ Gaussian-like sinkers within the domain and is defined as:

$$
\chi_n(\vec{x}) := \prod_{i=1}^{n} 1 - \exp \left( -\delta \max \left( 0, |\vec{x} - \vec{c}_i| - \frac{\omega}{2} \right)^2 \right)
$$

(6.1)

where

- $\delta = \text{positive parameter for controlling the exponential decay of Gaussian smoothing}$
- $\vec{c}_i = \text{location of the } i\text{th sinker center}$
- $\omega = \text{diameter of the sinkers}$.

Note that larger values of $\delta$ imply steeper gradients in the indicator function. As in (Rudi et al. 2016), we set $\delta = 200$ and $\omega = 0.1$ in all our numerical experiments.

**Viscosity Function**

The viscosity function, defined using the sinker indicator function, attains its maximum value within the sinkers and its minimum value away from the sinkers:

$$
\mu(\vec{x}) := (\mu_{\text{max}} - \mu_{\text{min}})(1 - \chi_n(\vec{x})) + \mu_{\text{min}}
$$

(6.2)

where $\mu_{\text{max}}$ and $\mu_{\text{min}}$ are determined by the dynamic ratio DR as $\mu_{\text{max}} = \text{DR}^{\frac{1}{2}}$ and $\mu_{\text{min}} = \text{DR}^{-\frac{1}{2}}$.

![Viscosity visualization on $P_2$ grid for DR = $10^4$](image)

Figure 6.1: Viscosity visualization on $P_2$ grid for DR = $10^4$ with sinker centers at $(0.25, 0.75)$ and $(0.75, 0.5)$. 

29
**Force Vector**

The right-hand side of the Stokes momentum equation (2.1), representing the action of gravity attempting to force the sinkers downward, is given by:

\[ \vec{f} := (0, 0, \beta(x_n(\vec{x}) - 1))\top, \quad \beta = 10. \]  

(6.3)

**Benchmark Difficulty**

The difficulty of the multi-sinker problem, characterized by the viscosity contrasts in the domain, can be heightened by increasing the dynamic ratio, the number of randomly placed sinkers, or both.

In the upcoming experiments, we employ the 20-Sinker benchmark (see Figure 6.2 and Appendix B.1) to assess the performance of the w-BFBT operator. Unless explicitly stated, the following parameters are considered default:

- **Stokes solver**: MINRES with block-diagonal preconditioner (see subsection 3.3).
- **Stokes solver convergence criteria**: \( \|r\|_{sc} < 10^{-12} \), where \( r \) is the residual of the non-preconditioned system and \( \|r\|_{sc} = \sqrt{(r\top r)/\#\text{DoFs}} \).
- **BFBT sub-solver**: Preconditioned CG solver (see subsection 6.3).
- **BFBT sub-solver convergence criteria**: \( \|r\|_{sc} < 10^{-8} \).
- **Viscosity contrast (DR)**: \( 10^4 \).
- **Primitive storage**: meshSymmetricCuboid with \( n_x, n_y, n_z = 1 \).
- **Refinement level (level)**: 4.

To achieve the action of \( \hat{\mathbf{A}}^{-1} \) in the block preconditioner for the Stokes system, we utilize the PETScLUSolver. This decision stems from the convergence issues experienced by the current multigrid capabilities of HyTeG (geometric multigrid + Chebyshev smoother for \( \mathbf{A} \) does not converge to a solution) when attempting to approximate the action of \( \mathbf{A}^{-1} \) for the benchmark problem. We suspect that the severe heterogeneity in viscosity adversely affects the performance of the multigrid smoother. The benchmark application, along with the supporting source code, is available in the apps/2023-bfbt directory (For commit and branch details, check subsection 5.4).

![Figure 6.2: Configuration of sinkers in the 20-Sinker benchmark.](image-url)
6.2 Comparison of Inverse Schur Complement Approximations

In this segment, we compare the performance of various inverse Schur complement approximations based on the results of our experiments. Before going into the details of our observations, we introduce the following nomenclature to enhance clarity and maintain conciseness in our discussions:

1. **Outer iterations**: Stokes solver iterations.

2. **Inner iterations**: BFBT sub-solver iterations (for the left and right sub-systems).

3. **BFBT operators**: The prefix suggests the weights \( C = D \) used in the BFBT approximation. For example, in the case of P1HL-BFBT, the BFBT operator utilizes the weights derived from row-sum lumping of the \( P_1 \) weighted-mass operator \( M(\sqrt{\mu}) \) at a higher level (refer to subsection 5.2 for further details on the weights).

Figure 6.3 and Table 6.1 present a comparison of different inverse Schur complement approximations. In our experiments, the DiagA-BFBT consistently outperforms other alternatives when handling complex viscosity functions. Furthermore, increasing viscosity contrasts enhances the effectiveness of DiagA-BFBT. This behavior remains to be completely understood and requires further study. Following the DiagA-BFBT, VBQ1-BFBT emerges as the next best choice, highlighting the superiority of the VBQ1 scheme over other lumping schemes discussed in subsection 5.2. As viscosity contrasts increase, we observe a degradation in the performance of \( (\tilde{M}_p(1/\mu))^{-1} \), which interestingly exhibits performance similar to the non-lumped version \( (M_p(1/\mu))^{-1} \).

![Figure 6.3: Stokes Solver Performance: Comparison of inverse Schur complement approximations.](image)

<table>
<thead>
<tr>
<th>DR</th>
<th>( (M_p(1/\mu))^{-1} )</th>
<th>DiagA-BFBT</th>
<th>VBQ1-BFBT</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 10^4 )</td>
<td>203</td>
<td>81</td>
<td>95</td>
</tr>
<tr>
<td>( 10^8 )</td>
<td>723</td>
<td>67</td>
<td>101</td>
</tr>
</tbody>
</table>

**Table 6.1**: Comparison of inverse Schur complement approximations: Number of outer iterations to achieve \( \| r \|_\infty < 10^{-12} \).

Despite their excellent approximation properties, the BFBT operators are computationally expensive due to the sub-system solves, as explained in subsection 5.3. In an attempt to alleviate this issue, we assess the performance of \( P1DivKGradSubOperator \) and \( P2DivKGradP1SubOperator \) as substitute sub-operators in VBQ1-BFBT. Unfortunately, this does not yield promising results (see Figure 6.4), and the behavior of the substitute sub-operators is similar for other variations of the...
w-BFBT operator. It is important to note that this unsatisfactory behavior persists even when the viscosity is constant and the cause remains unknown.

The \texttt{P2P1MixedSubOperator} aims to render the original sub-system compatible with multigrid methods by employing the \texttt{P1DivKGradSubOperator} in the smoother. However, its status is unclear since the sub-systems do not converge when using a multigrid solver with weighted-Jacobi smoothing. We suspect that the \texttt{P2P1BKBTSubOperator} at lower levels may not faithfully represent the \texttt{P2P1BKBTSubOperator} at the base level due to its composition of three operators instead of a single operator. Moreover, one must ensure that the sub-operators used in the weighted-Jacobi smoother are diagonally dominant before employing them. Finally, the w-BFBT operator exhibits mesh-dependent behavior, as observed in Table 6.2, highlighting the necessity for heuristic damping of the weights as detailed in (Rudi et al. 2016, sec. 5).

![Graph showing solver performance for different sub-operators](image)

**Figure 6.4:** Stokes Solver Performance: Comparison of sub-operators for \texttt{VBQ1-BFBT}.

<table>
<thead>
<tr>
<th>level</th>
<th>DoFs</th>
<th>Outer It.</th>
<th>Outer It.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>((M_p(1/\mu))^{-1})</td>
<td>\texttt{Diag-A-BFBT}</td>
</tr>
<tr>
<td>2</td>
<td>7736</td>
<td>292</td>
<td>36</td>
</tr>
<tr>
<td>3</td>
<td>56364</td>
<td>218</td>
<td>52</td>
</tr>
<tr>
<td>4</td>
<td>429908</td>
<td>202</td>
<td>80</td>
</tr>
</tbody>
</table>

**Table 6.2:** Mesh-dependent behavior of the Stokes solver for different inverse Schur complement approximations.

### 6.3 Solving w-BFBT Sub-systems

The sub-operator sub-solver pair significantly influences the computational workload when applying the BFBT operator. Our investigations could not identify a sub-operator that can serve as a viable substitute for the \texttt{P2P1BKBTSubOperator} while being compatible with multigrid methods. Nonetheless, we discovered diagonal preconditioners for the sub-systems in \texttt{Diag-A-BFBT} and \texttt{VBQ1-BFBT} that accelerate the convergence of the CG sub-solver when applying these operators. When utilizing \texttt{diag}(A) as the weights in the BFBT operator, preconditioning the sub-systems with \(M_p(1/\mu)\) significantly reduces inner iterations (Table 6.3). This choice of preconditioning (referred to as \texttt{PCG1}) also proves to be effective for achieving convergence in cases where the non-preconditioned CG sub-solver stalls. Similarly, \(M_p(1/\sqrt{\mu})\) can be used to precondition (referred to as \texttt{PCG2}) the sub-systems when applying \texttt{VBQ1-BFBT} (Table 6.4).
Another critical aspect of the sub-system involves establishing the required accuracy of the sub-system solution for maintaining the approximation properties of the BFBT operator. Tables 6.5 and 6.6 illustrate the effect of subsolver tolerance on the number of outer iterations. These tables indicate that the inner iterations are not required to be more accurate than the outer iterations, and a lower accuracy of the sub-system solution only mildly increases the number of outer iterations. However, an excessively low sub-solver tolerance degrades the performance of the preconditioner.

### Table 6.5: Effect of sub-solver tolerance on outer iterations for a Stokes solver tolerance of $10^{-12}$: \texttt{DiagA-BFBT}.

<table>
<thead>
<tr>
<th>Sub-solver tolerance</th>
<th>Outer It.</th>
<th>Inner It. (Total)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-3}$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>84</td>
<td>6611</td>
</tr>
<tr>
<td>$10^{-8}$</td>
<td>80</td>
<td>11250</td>
</tr>
<tr>
<td>$10^{-10}$</td>
<td>78</td>
<td>13828</td>
</tr>
<tr>
<td>$10^{-12}$</td>
<td>78</td>
<td>15921</td>
</tr>
</tbody>
</table>

### Table 6.6: Effect of sub-solver tolerance on outer iterations for a Stokes solver tolerance of $10^{-12}$: \texttt{VBQ1-BFBT}.

<table>
<thead>
<tr>
<th>Sub-solver tolerance</th>
<th>Outer It.</th>
<th>Inner It. (Total)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-3}$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$10^{-6}$</td>
<td>108</td>
<td>14104</td>
</tr>
<tr>
<td>$10^{-8}$</td>
<td>94</td>
<td>16720</td>
</tr>
<tr>
<td>$10^{-10}$</td>
<td>90</td>
<td>19002</td>
</tr>
<tr>
<td>$10^{-12}$</td>
<td>84</td>
<td>20112</td>
</tr>
</tbody>
</table>
7 Conclusion

The main objective of this thesis was to implement the matrix-free w-BFBT operator in HyTeG for preconditioning the $P_2-P_1$ Stokes systems with heterogeneous viscosity functions. Commencing with a detailed explanation about incorporating homogeneous Dirichlet boundary conditions into the components of the BFBT operator, we subsequently introduced diverse schemes for lumping $M_u(\sqrt{\mu})$ to determine the necessary weights. Additionally, we explored various substitutes for the sub-operator in the w-BFBT formulation, aiming to minimize computational effort when solving the sub-systems. We then concluded the implementation details with a comprehensive outline of the procedural steps for applying the matrix-free w-BFBT operator.

Using the multi-sinker benchmark, we evaluated the effectiveness of the w-BFBT operator as an approximation of the inverse Schur complement. In our experiments, the w-BFBT operator demonstrates improved (but mesh-dependent) convergence when compared to the widely employed $M_p(1/\mu)$. However, concerns persist regarding its application cost due to the challenging and unfruitful search for suitable substitute sub-operators. We identified preconditioners for the original sub-systems that help alleviate the computational workload; nevertheless, the practical viability of the current implementation of the w-BFBT operator for mantle convection simulations is still a matter of debate.

Several significant questions requiring further investigation surfaced during our work. These include:

1. Feasibility of alternative sub-operators for the $P2P1BKBTSubOperator$.
2. Effectiveness of the w-BFBT approximation for continuous versus discontinuous pressure spaces.
3. Identifying heuristics for damping the weights at boundary elements (see Rudi et al. 2016, sec. 5) to achieve mesh-independent convergence.
4. Viability of employing the heuristically damped weights when simulating mantle convection in spherical domains.
5. Impact of quadrature rules employed in the $P1DivKGradSubOperator$ and $P2DivKGradP1-SubOperator$ on their viability as sub-operators.
6. The necessity of smoothing the sub-system solutions using the $P2P1BKBTSubOperator$ when a viable substitute sub-operator is employed.

Table 7.1 emphasizes the critical distinctions in the settings of our work and those of (Rudi et al. 2016). When building upon our work in future studies, it is essential to carefully consider these differences and adapt the necessary aspects from (Rudi et al. 2016).

<table>
<thead>
<tr>
<th></th>
<th>Current work</th>
<th>Rudi et al. 2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discretization</td>
<td>$P_2-P_1$</td>
<td>$Q_k-P^{disc}_{k-1}$</td>
</tr>
<tr>
<td>Stokes solver</td>
<td>MINRES</td>
<td>GMRES</td>
</tr>
<tr>
<td>Stokes preconditioner</td>
<td>$\hat{A} \ 0$</td>
<td>$[\hat{A} \quad B^T]$</td>
</tr>
<tr>
<td></td>
<td>$0 \ \hat{S}$</td>
<td>$0 \ \hat{S}$</td>
</tr>
</tbody>
</table>

Table 7.1: Differences in settings: Current work and Rudi et al. 2016.

In conclusion, our work has resulted in a matrix-free w-BFBT operator with potential for application in mantle convection simulations. Realizing this potential depends on identifying suitable weights for which the w-BFBT preconditioner demonstrates mesh-independent convergence behavior and addressing the computational workload concerns during preconditioning.
References


A Relevant Inequalities

A.1 Poincaré’s Inequality

Let $\Omega$ be an open, bounded, and connected subset of $\mathbb{R}^d$. The Poincaré’s inequality states that for $1 \leq p < \infty$, there exists a constant $C$, depending only on $\Omega$ and $p$, such that

$$\|g\|_{L^p(\Omega)} \leq C \|\nabla g\|_{L^p(\Omega)} \quad (A.1)$$

for all functions $g$ having zero trace on the boundary $\partial\Omega$ and in the Sobolev space $W^{1,p}(\Omega)$. Poincaré’s inequality stays valid for vector-valued functions if $\Omega$ is bounded with a locally Lipschitz boundary (John 2016, p. 686).

A.2 Korn’s Inequality

Let $\Omega$ be an open, bounded, and connected subset of $\mathbb{R}^d$. The Korn’s inequality states that for all $\vec{v} \in H^1(\Omega)^d$ having zero trace on the boundary $\partial\Omega$,

$$2 \|\epsilon(\vec{v})\|_{L^2(\Omega)}^2 = \|\nabla \vec{v}\|_{L^2(\Omega)}^2 + \|\nabla \cdot \vec{v}\|_{L^2(\Omega)}^2 \quad (A.2)$$

and consequently

$$\|\nabla \vec{v}\|_{L^2(\Omega)} \leq \sqrt{2} \|\epsilon(\vec{v})\|_{L^2(\Omega)} . \quad (A.3)$$

For proof, see (John 2016, p. 46-47).

B Benchmark Configuration

B.1 Location of Sinker Centers for the 20-Sinker Benchmark

1. 0.5900672813646731, 0.17138939996472613, 0.25363895188732954
2. 0.2854198880601549, 0.40375694572969834, 0.24028995315680046
3. 0.7019249192050013, 0.15565825402270983, 0.8975450920737098
4. 0.034256163333108, 0.8690849285825951, 0.6603153973352672
5. 0.7923162490812403, 0.9582319941495265, 0.11859617320653626
6. 0.10643745550774641, 0.26449063806395, 0.4911265885668187
7. 0.7678063644737774, 0.919037099192661, 0.281694108197999
8. 0.599101180648847, 0.4891392453652138, 0.5062824519051923
9. 0.7562392160645719, 0.5289631200720056, 0.4673487398965811
10. 0.4117150534087636, 0.3449098578619101, 0.20585370765837707
11. 0.01947828749232905, 0.0579428553892168, 0.7379949630505896
12. 0.00780804126439494, 0.3019561233246034, 0.012044164058422124
13. 0.2157195794749457, 0.8588295213643592, 0.8021685808349034
14. 0.817498358605668, 0.5400748851411242, 0.912020236241185
15. 0.6008621096767651, 0.6736483918650694, 0.343640086256177
16. 0.9353999139616266, 0.8421215870377751, 0.766115390734649
17. 0.506399750974906, 0.005290493469429003, 0.8567449626821255
18. 0.787705039287262, 0.5633716044534675, 0.446266066246617
19. 0.6843769614338335, 0.2286670618643638, 0.733958041362048
20. 0.9312953459697999, 0.9243132782398487, 0.980840697459257