Domain Decomposition And Time-Splitting Methods For The Biot System Of Poroelasticity

by

Manu Jayadharan

BS and MS in Mathematics, Indian Institute of Science Education and Research, Mohali, 2016

Submitted to the Graduate Faculty of

the Kenneth P. Dietrich School of Arts and Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2021

UNIVERSITY OF PITTSBURGH

KENNETH P. DIETRICH SCHOOL OF ARTS AND SCIENCES

This dissertation was presented

by

Manu Jayadharan

It was defended on

July 22nd 2021

and approved by

Prof. Ivan Yotov, Dept. of Mathematics, University of Pittsburgh

Prof. Michael Neilan, Dept. of Mathematics, University of Pittsburgh

Prof. Catalin Trenchea, Dept. of Mathematics, University of Pittsburgh

Prof. Martin Vohralík, INRIA, Paris

Domain Decomposition And Time-Splitting Methods For The Biot System Of Poroelasticity

Manu Jayadharan, PhD

University of Pittsburgh, 2021

In this thesis, we develop efficient mixed finite element methods to solve the Biot system of poroelasticity, which models the flow of a viscous fluid through a porous medium along with the deformation of the medium. We study non-overlapping domain decomposition techniques and sequential splitting methods to reduce the computational complexity of the problem. The solid deformation is modeled with a mixed three-field formulation with weak stress symmetry. The fluid flow is modeled with a mixed Darcy formulation.

We introduce displacement and pressure Lagrange multipliers on the subdomain interfaces to impose weakly the continuity of normal stress and normal velocity, respectively. The global problem is reduced to an interface problem for the Lagrange multipliers, which is solved by a Krylov space iterative method. We study both monolithic and split methods. For the monolithic method, the cases of matching and non-matching subdomain grid interfaces are analyzed separately. For both cases, a coupled displacement-pressure interface problem is solved, with each iteration requiring the solution of local Biot problems. For the case of matching subdomain grids, we show that the resulting interface operator is positive definite and analyze the convergence of the iteration. For the non-matching subdomain grid case, we use a multiscale mortar mixed finite element (MMMFE) approach.

We further study drained split and fixed stress Biot splittings, in which case we solve separate interface problems requiring elasticity and Darcy solves. We analyze the stability of the split formulations. We also use numerical experiments to illustrate the convergence of the domain decomposition methods and compare their accuracy and efficiency in the monolithic and time-splitting settings.

Finally, we present a novel space-time domain decomposition technique for the mixed finite element formulation of a parabolic equation. This method is motivated by the MMMFE method, where we split the space-time domain into multiple subdomains with space-time grids of different sizes. Scalar Lagrange multiplier (mortar) functions are introduced to enforce weakly the

continuity of the normal component of the mixed finite element flux variable over the space-time interfaces. We analyze the new method and numerical experiments are developed to illustrate and confirm the theoretical results.

Keywords: MFEM, MFE, Biot system, domain decomposition, poroelasticity, space-time methods, multiscale mortar method, MMMFEM, mortar, split schemes, splitting methods, partitioning schemes, computational geoscience.

Table of Contents

Preface . . . xii

1.0 Introduction . . . 1

1.1 Methodology . . . 1

1.2 Basic Notation . . . 6

1.3 Model Problems . . . 8

1.3.1 Biot system of poroelasticity . . . 8

1.3.2 Time-dependent parabolic PDE . . . 9

2.0 Domain Decomposition And Split-scheme Techniques For Biot System Of Poroelasticity Using Matching Subdomain Grids . . . 10

2.1 Introduction . . . 10

2.2 MFE Discretization . . . 11

2.3 Monolithic Domain Decomposition Method . . . 13

2.3.1 Time discretization . . . 15

2.3.2 Time-differentiated elasticity formulation . . . 16

2.3.3 Reduction to an interface problem . . . 17

2.3.4 Analysis of the interface problem . . . 18

2.4 Split Methods . . . 24

2.4.1 Drained split . . . 24

2.4.1.1 Stability analysis for drained split . . . 25

2.4.2 Fixed stress . . . 27

2.4.2.1 Stability analysis for fixed stress . . . 27

2.4.3 Domain decomposition for the split methods . . . 29

2.5 Numerical Results . . . 30

2.5.1 Example 1: convergence and stability . . . 31

2.5.2 Example 2: dependence on number of subdomains . . . 32

2.5.3 Example 3: heterogeneous benchmark . . . 34

2.6 Chapter Conclusions . . . 35

3.0 A Multiscale Mortar Domain Decomposition For Biot System Of Poroelasticity Using Non-matching Subdomain Grids 43
3.1 Introduction 43
3.2 Formulation of the Method 44
3.2.1 Multiscale mortar domain decomposition method 44
3.2.2 Projection and interpolation operators 47
3.2.3 Spaces of weakly continuous stress and velocity 51
3.3 Analysis of the MMMFE Method 52
3.3.1 Inf-sup stability for the weakly continuous spaces 52
3.3.2 Well-posedness of the semi-discrete MMMFE formulation 55
3.3.3 Stability analysis for MMMFE formulation 60
3.3.4 Error analysis 65
3.4 Implementation 78
3.4.1 Time discretization 78
3.4.2 Reduction to an interface problem 80
3.4.3 Solving the interface problem 82
3.4.4 Implementation with multiscale stress-flux basis (MSB) 83
3.5 Numerical Results 85
3.5.1 Example 1: testing convergence rates 86
3.5.2 Example 2: heterogeneous medium 88
3.6 Chapter Conclusions 90
4.0 A Multiscale Mortar Space-time Domain Decomposition Technique For Parabolic Equations 99
4.1 Introduction 99
4.2 Model Problem and Space-Time Domain Decomposition Formulation 100
4.2.1 Model problem 100
4.2.2 Space-time subdomains 100
4.2.3 Basic notation 101
4.2.4 Weak formulation 101
4.2.5 Domain decomposition weak formulation 102
4.3 Space-Time Mixed Finite Element Method 102

4.3.1 Space-time grids and spaces . . . 103
4.3.2 Space-time multiscale mortar mixed finite element method . . . 105
4.4 Well-Posedness Analysis . . . 106
4.4.1 Space-time interpolants . . . 106
4.4.2 Assumptions on the mortar grids . . . 107
4.4.3 Discrete inf–sup conditions . . . 107
4.4.4 Existence, uniqueness, and stability with respect to data . . . 110
4.5 A Priori Error Analysis . . . 111
4.5.1 Approximation properties of the space-time interpolants . . . 112
4.5.2 A priori error estimate . . . 113
4.5.3 Comments . . . 116
4.6 Reduction To An Interface Problem . . . 117
4.6.1 Decomposition of the solution . . . 117
4.6.2 Space-time Steklov–Poincaré operator . . . 118
4.6.3 GMRES convergence through the field-of-values estimates . . . 119
4.7 Numerical Results . . . 120
4.7.1 Example 1: convergence test . . . 121
4.7.2 Example 2: problem with a boundary layer . . . 125
4.8 Chapter Conclusions . . . 129
5.0 Conclusions . . . 130
5.1 Summary of Techniques Developed and Results . . . 130
5.2 Future Work . . . 131
Appendix. Code Gallery . . . 133
A.1 Note to the Reader . . . 133
A.2 Links to Open-source Packages Corresponding to Various Chapters . . . 133
A.3 Implementation of the Space-time Multiscale Mortar Decomposition Method . . . 134
A.3.1 User interface . . . 134
A.3.2 Source code . . . 137
Bibliography . . . 170

List of Tables

1 Example 1, physical and numerical parameters. 31

2 Example 1, convergence for $\Delta t = 10^{-3}$ and $c_0 = 1$, monolithic scheme (top), drained split (middle), fixed stress (bottom). 36

3 Example 1, convergence for $\Delta t = 10^{-2}$ and $c_0 = 1$, monolithic scheme (top), drained split (middle), fixed stress (bottom). 37

4 Example 1, convergence for $\Delta t = 10^{-1}$ and $c_0 = 1$, monolithic scheme (top), drained split (middle), fixed stress (bottom). 38

5 Example 1, convergence for $\Delta t = 10^{-2}$ and $c_0 = 10^{-3}$, monolithic scheme (top), drained split (middle), fixed stress (bottom). 39

6 Example 2, number of GMRES iterations in the monolithic scheme. 40

7 Example 2, number of CG elasticity iterations in the drained split and fixed stress schemes. 40

8 Example 2, number of CG Darcy iterations in the drained split and fixed stress schemes 40

9 Example 3, parameters (left) and boundary conditions (right) 41

10 Example 3, comparison of the number of interface iterations in the three methods. . 42

11 Degree of polynomials associated with FEM spaces used for numerical experiments. 85

12 Example 1, physical and numerical parameters. 87

13 Example 1, convergence table using linear mortar ($m = 1$) with $H = Ch$, $\Delta t = 10^{-4}$ and $c_0 = 1.0$. 91

14 Example 1, convergence table using quadratic mortar ($m = 2$) with $H = C\sqrt{h}$, $\Delta t = 10^{-4}$ and $c_0 = 1.0$. 91

15 Example 1, convergence table for linear mortar with $H = Ch$, $\Delta t = 10^{-4}$ and $c_0 = 10^{-3}$. 92

16 Example 1, convergence table for quadratic mortar with $H = C\sqrt{h}$, $\Delta t = 10^{-4}$ and $c_0 = 10^{-3}$. 92

17 Example 2, parameters (top) and boundary conditions (bottom). 94

18 Example 2, #GMRES iterations and maximum number of subdomain solves. . . . 94

19 Example 1, mesh size and #DoFs 122

20 Linear mortar convergence 122
21 Quadratic mortar convergence 122
22 Example 2, errors for the multiscale and fine-scale methods. 126

List of Figures

1 Example 1, computed solution at the final time step using the monolithic domain decomposition method with $h = 1/64$ and $\Delta t = 10^{-3}$, top: stress x (left), stress y (middle), displacement (right), bottom: rotation (left), velocity (middle), pressure (right). 33

2 Example 3, porosity, Young's modulus, permeability. 41

3 Example 3, computed solution at the final time using the monolithic domain decomposition scheme, top: pressure (left), velocity (right), middle: displacement (left), stress x (right), bottom: stress y. 42

4 Example 1, coarsest non matching subdomain grid on $(0, 1)^2$. 86

5 Example 1, computed solution at final time step using a linear mortar on non-matching subdomain grids, top: stress x (left), stress y (middle), displacement (right), bottom: rotation (left), velocity (middle), pressure (right). Mesh size, $h = 1/32$, $\Delta t = 10^{-3}$ and $c_0 = 1.0$. 93

6 Example 2, permeability, porosity, Young's modulus. 95

7 Example 2, pressure (color) and velocity (arrows): fine scale (left), single linear mortar per interface (middle), and two linear mortars per interface (right). 95

8 Example 2, pressure (color) and velocity (arrows): single quadratic mortar per interface (left), and two quadratic mortars per interface (right). 96

9 Example 2, velocity magnitude: fine scale (left), single linear mortar per interface (middle), and two linear mortars per interface (right). 96

10 Example 2, velocity magnitude: single quadratic mortar per interface (left), and two quadratic mortars per interface (right). 97

11 Example 2, displacement vector (arrows) and its magnitude: fine scale (left), single linear mortar per interface (middle), and two linear mortars per interface (right). . 97

12 Example 2, displacement vector (arrows) and its magnitude: single quadratic mortar per interface (left), and two quadratic mortars per interface (right). 98

13 Non-matching space-time subdomain and mortar grids in two spatial dimensions. . 103

14 Example 1, pressure computed using linear mortars shown on the space-time grid at refinement 2, top: on the whole space-time domain Ω^T, bottom: on $\Omega_1^T \cup \Omega_4^T$ (left), on $\Omega_2^T \cup \Omega_3^T$ (right). 123

15 Example 1, $x-$component of velocity computed using linear mortars shown on the space-time grid at refinement 2, on $\Omega_1^T \cup \Omega_4^T$ (left), on $\Omega_2^T \cup \Omega_3^T$ (right). 124

16 Example 1, $y-$component of velocity computed using linear mortars shown on the space-time grid at refinement 2, on $\Omega_1^T \cup \Omega_4^T$ (left), on $\Omega_2^T \cup \Omega_3^T$ (right). 124

17 Example 2, pressure from the multiscale method, cut along the plane $x = 0.25$ (top), velocity magnitude from the multiscale method, cut along the plane $x = 0.25$ (bottom).126

18 Example 2, left: pressure from the multiscale method, cut along the plane $t = 0.35$; right: pressure from the multiscale (top) and fine-scale (bottom) methods on the whole domain. 127

19 Example 2, left: velocity magnitude from the multiscale method, cut along the plane $t = 0.35$; right: velocity magnitude from the multiscale (top) and fine-scale (bottom) methods on the whole domain. 128

Preface

I would like to thank everyone who has helped me through this wonderful journey. I have to admit that wrapping up my research and completing this thesis during a global pandemic was more challenging than I initially thought. Not because it had any direct impact on my research, but because of other factors associated with the pandemic that weighed on me. Consequently, the successful completion of this thesis is that much more special to me.

I wish to express my most sincere gratitude and appreciation to my advisor Ivan Yotov for his guidance, patience, and unparalleled role in the completion of my research projects. I am indebted to him for understanding and working with the many time constraints I had and also putting up with my inability to find mistakes in some manuscripts I wrote and for correcting them. There was never a time when I was afraid to ask him a question or pitch ideas which is something I always cherished. I would like to extend my gratitude to professors Michael Neilan, Catalin Trenchea, and Martin Vohralík for serving on my thesis committee and providing invaluable feedback which helped greatly in the completion of this thesis.

I would like to thank all my friends and family who have supported me during my time at Pitt. Especially, I thank Eldar Khattatov for his invaluable help and positive influence on my projects and career. I would also like to express my love and gratitude towards Sruthi for being a rock and supporting me during stressful times. Last but not least, I would like to express my gratitude to my parents, Omana and Jayadharan, who never stopped believing in me.

1.0 Introduction

1.1 Methodology

The Biot system of poroelasticity [17] is used to model the flow of a viscous fluid through a poroelastic medium along with the deformation of the medium. Such flow occurs in many geophysics phenomena like earthquakes, landslides, and flow of oil inside mineral rocks and plays a key role in engineering applications such as hydrocarbon extraction through hydraulic or thermal fracturing. In this thesis we study efficient computational methods to numerically solve the classical Biot system of poroelasticity with the quasi-static assumption, which is particularly relevant in geoscience applications. The model consists of an equilibrium equation for the solid medium and a mass balance equation for the flow of the fluid through the medium. The system is fully coupled, with the fluid pressure contributing to the solid stress and the divergence of the solid displacement affecting the fluid content.

Numerical methods to solve the Biot system have been extensively studied in the literature. Many formulations have been considered, including two-field displacement-pressure formulations [34,60,69], three-field displacement–pressure–Darcy velocity formulations [45,56,65,66,70,79,80,82], and three-field displacement–pressure–total pressure formulations [57,62]. More recently, fully-mixed formulations of the Biot system have been studied. In [81], a four-field stress-displacement-pressure-Darcy velocity mixed formulation is developed. A posteriori error estimate for this formulation are obtained in [2]. In [54], a weakly symmetric stress–displacement–rotation elasticity formulation is considered, which is coupled with a mixed pressure–Darcy velocity flow formulation. Fully-mixed finite element approximations carry the advantages of local mass and momentum conservation, direct computation of the fluid velocity and the solid stress, as well as robustness and locking-free properties with respect to the physical parameters. Mixed finite element (MFE) methods can also handle discontinuous full tensor permeabilities and Lamé coefficients that are typical in subsurface flows. In our work we focus on the five-field weak-stress-symmetry formulation from [54], since weakly symmetric MFE methods for elasticity allow for reduced number of degrees of freedom. Moreover, a multipoint stress–multipoint flux mixed finite element approximation for this formulation has been recently developed in [6], which can

be reduced to a positive definite cell-centered scheme for pressure and displacement only, see also a related finite volume method in [61]. While our domain decomposition methods are developed for the weakly symmetric formulation from [54], the analysis carries over in a straightforward way to the strongly symmetric formulation from [81].

Discretizations of the Biot system of poroelasticity for practical applications typically result in large algebraic systems of equations. The efficient solution of these systems is critical for the ability of the numerical method to provide the desired resolution. In this work we focus on non-overlapping domain decomposition methods [67,77]. These methods split the computational domain into multiple non-overlapping subdomains with algebraic systems of lower complexity that are easier to solve. A global problem enforcing appropriate interface conditions is solved iteratively to recover the global solution. This approach naturally leads to scalable parallel algorithms. Despite the abundance of works on discretizations of the Biot system, there have been very few results on domain decomposition methods for this problem. In [35], a domain decomposition method using mortar elements for coupling the poroelastic model with an elastic model in an adjacent region is presented. In that work, the Biot region is not decomposed into subdomains. In [30,31], an iterative coupling method is employed for a two-field displacement–pressure formulation, and classical domain decomposition techniques are applied separately for the elasticity and flow equations. A monolithic domain decomposition method for the two-field formulation of poroelasticity combining primal and dual variables is developed in [40]. To the best of our knowledge, domain decomposition methods for mixed formulations of poroelasticity have not been studied in the literature. In this thesis, we study various efficient domain decomposition and split-scheme discretization techniques to efficiently solve the Biot system of poroelasticity in a five-field mixed formulation.

In Chapter 2, we develop a monolithic domain decomposition method for the Biot system using matching subdomain grids at the interface. We employ a physically heterogeneous Lagrange multiplier vector consisting of displacement and pressure variables to impose weakly the continuity of the normal components of stress and velocity, respectively. The algorithm involves solving at each time step an interface problem for this Lagrange multiplier vector. We show that the interface operator is positive definite, although it is not symmetric in general. As a result, a Krylov space solver such as GMRES can be employed for the solution of the interface problem. Each iteration requires solving monolithic Biot subdomain problems with specified Dirichlet

data on the interfaces, which can be done in parallel. We establish lower and upper bounds on the spectrum of the interface operator, which allows us to perform analysis of the convergence of the GMRES iteration using field-of-values estimates. In the second part of Chapter 2, we study split domain decomposition methods for the Biot system. Split or iterative coupling methods for poroelasticity have been extensively studied due to their computational efficiency. Four widely used sequential methods are drained split, undrained split, fixed stress split, and fixed strain split. Decoupling methods are prone to stability issues and a detailed stability analysis of the aforementioned schemes using finite volume methods can be found in [51, 52], see also [23] for stability analysis of several split methods using displacement–pressure finite element discretizations. Iterative coupling methods are based on similar splittings and involve iterating between the two sub-systems until convergence. Convergence for non-mixed finite element methods is analyzed in [59], while convergence for a four-field mixed finite element discretization is studied in [83]. An accelerated fixed stress splitting scheme for a generalized nonlinear consolidation of unsaturated porous medium is studied in [21]. Studies of the optimization and acceleration of the fixed stress decoupling method for the Biot consolidation model, including techniques such as multirate or adaptive time stepping and parallel-in-time splittings have been presented in [1, 3, 15, 20, 76]. In our work we consider drained split (DS) and fixed stress (FS) decoupling methods in conjunction with non-overlapping domain decomposition. In particular, at each time step we solve sequentially an elasticity and a flow problem in the case of DS or a flow and an elasticity problem in the case of FS splitting. We perform stability analysis for the two splittings using energy estimates and show that they are both unconditionally stable with respect to the time step and the physical parameters. We then employ separate non-overlapping domain decomposition methods for each of the decoupled problems, using the methods developed in [8, 38] for flow and [48] for mechanics.

In Chapter 3, we develop and study a multiscale mortar mixed finite element (MMMFE) method for the Biot system of poroelasticity. This technique is the generalization of the monolithic domain decomposition technique presented in the Chapter 2, where non-matching subdomain grids can be used instead of matching grids at the interface. MMMFE methods that allow non-matching subdomain grid blocks for second order elliptic problems have been studied in [8, 90] and a similar method for the problem of linear elasticity with weakly imposed symmetry has been studied in [48]. To the best of our knowledge, an MMMFE method for any

mixed formulations of poroelasticity has not been studied in the literature prior to our study. We study the adaptation of the non-overlapping domain decomposition technique studied in the Chapter 2 to enable the use of non-matching multiblock grids. This work is motivated by similar studies for the second order elliptic problems in [8,90] and for a linear system of elasticity in [48]. Similar to the matching-grid case, we use a physically heterogeneous Lagrange multiplier vector consisting of displacement and pressure variables to impose weakly the continuity of the normal components of stress and velocity, respectively. At each time step, we solve an interface problem for this Lagrange multiplier vector. In contrast to the matching-grid case, here we choose the Lagrange multiplier vector from a space of mortar finite elements, see e.g. [8,32,36,43,48–50,63]. This allows the interaction between the subdomain grids at the interface through projections onto the mortar finite element space. The mortar space can be chosen to be on a coarser scale, H (see [33,64,90]), compared to a finer subdomain grid size, h. We study the well-posedness and stability of this method under the appropriate condition on the richness of the mortar FE space. Further, we show a combined a priori error estimate for stress, displacement, rotation, pressure, and Darcy velocity, as well as how well the mortar function approximates the normal components of stress and velocity. The well-posedness, stability, and error analysis are motivated by the techniques discussed in the analysis of the MSMFE-MFMFE (multipoint stress and multipoint flux mixed fnite element) formulation for the Biot system studied in [6]. We also propose the construction and use of a multiscale stress-flux basis which makes the number of subdomain solves related to interface problem independent of the number of iterations required for the interface problem and the number of time steps used. The reuse of the multiscale basis could gain a significant performance advantage in the case of time-dependent coupled problems. This basis construction is motivated by the use of multiscale flux basis studied in [33] and multiscale stress basis in [48].

In Chapter 4, we introduce a *space-time domain decomposition discretization* technique for a model parabolic equation. This method is a generalization of the MMMFE technique discussed in Chapter 3 for a time-dependent parabolic system, where we allow multiscale mortar discretization in both space and time. This work is a starting point for developing such multiscale space-time techniques for the Biot system of poroelasticity, which will be pursued in future studies. We decompose the global space-time domain into multiple space-time subdomains and introduce a space-time mortar variable, on an independent interface space-time mesh. This space

is then used to couple the space-time subdomain problems and to ensure (a multiscale) weak continuity of the normal component of the mixed finite element flux variable over the space-time interfaces. This setting allows for high flexibility with individual discretizations of each space-time subdomain, and in particular for local time stepping, individually in each space-time subdomain. Moreover, space-time parallelization can be achieved, leading to solution of discrete problems on individual space-time subdomains, exchanging space-time boundary data through transmission conditions, in a spirit of space-time domain decomposition as in [106,112,113].These methods belong to an increasing body of parallel in time methods (see [105] for a review, or the web page Parallel in Time). Some methods emphasize time parallelism for a rather general class of problems, such as the celebrated Parareal algorithm [108, 118], or multigrid in time [103, 107, 124]. Others, including this work, are more tailored towards specific classes of PDES. Multi–rate methods [88, 122] have been used for flow and for poro-elasticity, as well as asynchronous methods [85, 96], or several variants of space time methods [93, 94, 97, 117, 120].

For all the methods presented in this thesis, we also report the results of several numerical tests designed to verify and compare the convergence, stability, and efficiency of these techniques. All numerical schemes are implemented using deal.II finite element package [91,92]. The mixed finite element spaces used for the numerical experiments related to the Biot system are: $\mathcal{BDM}_1^2 \times Q_0^2 \times Q_0$ [11] for elasticity and $\mathcal{BDM}_1 \times Q_0$ [22] for Darcy on quadrilateral meshes. Here Q_k denotes polynomials of degree k in each variable and $\mathcal{BDM}_k$ represents Brezzi-Douglas-Marini spaces containing polynomials of degree k. For the space-time domain decomposition for the parabolic problem, we have used stable $\mathcal{RT}_0 \times Q_0$ on a quadrilateral mesh, where $\mathcal{RT}_k$ denotes the Raviart-Thomas space containing polynomials of degree k. Combining this with the lowest-order DG (backward Euler) for time discretization on the mesh gives us a space-time mixed finite element space on the space-time domain.

As the implementation of novel algorithms are equally important as the theory itself, significant amount of time and effort went into the production of software capable of solving PDEs using the methods developed in this thesis. These packages are made open-source and are available on the GitHub page *https://github.com/mjayadharan*. Links to various code repositories developed as part of this thesis is given in Appendix A.2 and a brief overview of one of the core libraries is given in Appendix A.3.

1.2 Basic Notation

We tried to be consistent with our notation across different chapters in this thesis, but there were a few occasions where we had to use a slightly different notation for some variables, like the number of subdomains. Because of this and for the completeness of each chapter, we have defined the notation used within each chapter. Following is some of the notation which is consistent throughout the thesis.

Let $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, denote a simply connected domain. We use the notation $\mathbb{M}$, $\mathbb{S}$ and $\mathbb{N}$ for the spaces of $d \times d$ matrices, symmetric matrices, and skew-symmetric matrices, respectively, all over the field of real numbers. Let $I \in \mathbb{S}$ represents the $d \times d$ identity matrix. The partial derivative operator with respect to time, $\frac{\partial}{\partial t}$, is often abbreviated to ∂_t. C denotes a generic positive constant that is independent of the discretization parameters h and H. Throughout the thesis, the divergence operator is the usual divergence for vector fields, which produces vector field when applied to matrix field by taking the divergence of each row. The divergence operator is represented by either $\nabla\cdot$ or div.

For a set $G \subset \mathbb{R}^d$, the $L^2(G)$ inner product and norm are denoted by $(\cdot,\cdot)_G$ and $\|\cdot\|_G$, respectively, for scalar, vector, or tensor valued functions. For any $r \geq 0$, $\|\cdot\|_{r,G}$ denotes the $H^r(G)$-norm, with $\|\cdot\|_{0,G} = \|\cdot\|_G$. We omit subscript G if $G = \Omega$. For a section of the domain or element boundary $S \subset \mathbb{R}^{d-1}$, we write $\langle\cdot,\cdot\rangle_S$ and $\|\cdot\|_S$ for the $L^2(S)$ inner product (or duality pairing) and norm, respectively. For space-time norms we use the standard Bochner notation. For example, given a spatial norm $\|\cdot\|_X$, with respect with to a space X, we denote, for $p > 0$,

$$\|\cdot\|_{L^p(0,T;X)} = \left(\int_0^T \|\cdot\|_X^p\right)^{\frac{1}{p}}, \quad \|\cdot\|_{L^\infty(0,T;X)} = \operatorname{ess\,sup} \|\cdot\|_X,$$

with the usual extension for $\|\cdot\|_{W^{k,p}(0,T;X)}$ and $\|\cdot\|_{H^k(0,T;X)}$.

We will also use the spaces

$$H(\operatorname{div};\Omega) = \{\zeta \in L^2(\Omega,\mathbb{R}^d) : \operatorname{div}\zeta \in L^2(\Omega)\},$$
$$H(\operatorname{div};\Omega,\mathbb{M}) = \{\tau \in L^2(\Omega,\mathbb{M}) : \operatorname{div}\tau \in L^2(\Omega,\mathbb{R}^d)\},$$

with the norm $\|\tau\|_{\operatorname{div}} = \left(\|\tau\|^2 + \|\operatorname{div}\tau\|^2\right)^{1/2}$.

For an element K in dimension $d = 2$, let

$$P_k(K) = \left\{ p(x_1, x_2) \mid p(x_1, x_2) = \sum_{0 \le i+j \le k} a_{ij} x_1^i x_2^j \right\}$$

be the space polynomials of degree less than or equal to k and let

$$P_{k_1,k_2}(K) = \left\{ p(x_1, x_2) \mid p(x_1, x_2) = \sum_{\substack{0 \le i \le k_1, \\ 0 \le j \le k_2}} a_{ij} x_1^i x_2^j \right\}$$

be the space of polynomials of degree less than or equal to k_1 in x_1 and less than or equal to k_2 in x_2 on K. Similarly, for $d = 3$, we can $P_{k_1,k_2,k_3}(K)$.

The Raviart-Thomas elements of degree $k \ge 0$ on rectangular and cube elements are defined as follows:

$$\mathcal{RT}_k(K) := \begin{cases} P_{k+1,k}(K) \times P_{k,k+1}(K) & \text{for } d = 2, \\ P_{k+1,k,k}(K) \times P_{k,k+1,k}(K) \times P_{k,k,k+1}(K) & \text{for } d = 3. \end{cases}$$

For $d = 2$ and $k \ge 1$, the Brezzi-Douglas-Marini space $\mathcal{BDM}_k$ on a rectangular element is defined as

$$\mathcal{BDM}_k(K) := (P_k(K))^2 + \text{span } \left(\text{curl } x_1 x_2^{k+1}, \text{curl } x_1^{k+1} x_2 \right),$$

where curl $(\zeta) = (\frac{\partial \zeta}{\partial x_2}, -\frac{\partial \zeta}{\partial x_1})$ for a scalar function $\zeta(x_1, x_2)$. Similarly for $d = 3$, we define

$$\mathcal{BDM}_k(K) := (P_k(K))^3 + \text{span } \Big(\text{curl } \big\{ x_2 x_3 (w_2(x_1, x_3) - w_3(x_1, x_2)), \\ x_3 x_1 (w_3(x_1, x_2) - w_1(x_2, x_3)), x_1 x_1 (w_1(x_2, x_3) - w_2(x_1, x_3)) \big\} \Big),$$

where each w_i belongs to $P_k(K)$ and we have used the usual curl operator acting on a 3-dimensional vector field.

1.3 Model Problems

1.3.1 Biot system of poroelasticity

Given a vector field f representing body forces and a source term g, we consider the quasi-static Biot system of poroelasticity ([17]):

$$-\operatorname{div}\sigma(u) = f, \qquad \text{in } \Omega \times (0,T], \tag{1.3.1}$$

$$K^{-1}z + \nabla p = 0, \qquad \text{in } \Omega \times (0,T], \tag{1.3.2}$$

$$\frac{\partial}{\partial t}(c_0 p + \alpha \operatorname{div} u) + \operatorname{div} z = g, \qquad \text{in } \Omega \times (0,T], \tag{1.3.3}$$

where u is the displacement, p is the fluid pressure, z is the Darcy velocity, and σ is the poroelastic stress, defined as

$$\sigma = \sigma_e - \alpha p I. \tag{1.3.4}$$

Here I is the $d \times d$ identity matrix, $0 < \alpha \leq 1$ is the Biot-Willis constant, and σ_e is the elastic stress satisfying the stress-strain relationship

$$A\sigma_e = \epsilon(u), \quad \epsilon(u) := \frac{1}{2}\left(\nabla u + (\nabla u)^T\right), \tag{1.3.5}$$

where A is the compliance tensor, which is a symmetric, bounded and uniformly positive definite linear operator acting from $\mathbb{S} \to \mathbb{S}$, extendible to $\mathbb{M} \to \mathbb{M}$. In the special case of homogeneous and isotropic body, A is given by,

$$A\sigma = \frac{1}{2\mu}\left(\sigma - \frac{\lambda}{2\mu + d\lambda}\operatorname{tr}(\sigma) I\right), \tag{1.3.6}$$

where $\mu > 0$ and $\lambda \geq 0$ are the Lamé coefficients. In this case, $\sigma_e(u) = 2\mu\epsilon(u) + \lambda \operatorname{div} u\, I$. Finally, $c_0 \geq 0$ is the mass storativity and K stands for the permeability tensor that is spatially-dependent, uniformly bounded, symmetric, and positive definite, i.e, for constants $0 < k_{\min} \leq k_{\max} < \infty$,

$$\forall \text{ a.e. } x \in \Omega, \quad k_{\min}\zeta^T\zeta \leq \zeta^T K(x)\zeta \leq k_{\max}\zeta^T\zeta, \quad \forall \zeta \in \mathbb{R}^d. \tag{1.3.7}$$

To close the system, we impose the boundary conditions

$$u = g_u \quad \text{on } \Gamma_D^u \times (0,T], \qquad \sigma n = 0 \quad \text{on } \Gamma_N^\sigma \times (0,T], \tag{1.3.8}$$

$$p = g_p \quad \text{on } \Gamma_D^p \times (0,T], \qquad z \cdot n = 0 \quad \text{on } \Gamma_N^z \times (0,T], \tag{1.3.9}$$

where $\Gamma_D^u \cup \Gamma_N^\sigma = \Gamma_D^p \cup \Gamma_N^z = \partial\Omega$ and n is the outward unit normal vector field on $\partial\Omega$, along with the initial condition $p(x,0) = p_0(x)$ in Ω. Compatible initial data for the rest of the variables can be obtained from (1.3.1) and (1.3.2) at $t = 0$. Well posedness analysis for this system can be found in [72].

1.3.2 Time-dependent parabolic PDE

We consider a parabolic partial differential equation in a mixed form, modeling single phase flow in porous media. Following the notations defined above and using the time interval $(0, T]$ for $T > 0$, the model system of equations reads as follows:

$$\mathbf{u} = -K\nabla p, \qquad \partial_t p + \operatorname{div} \mathbf{u} = q \qquad \text{in } \Omega \times (0, T], \tag{1.3.10}$$

where p is the fluid pressure, $\mathbf{u}$ is the Darcy velocity, q is a source term, and K is a tensor representing the rock permeability divided by the fluid viscosity. We assume for simplicity that the homogeneous Dirichlet boundary condition

$$p(x,t) = 0 \qquad \text{on } \partial\Omega \times (0, T], \tag{1.3.11}$$

and assign the initial pressure

$$p(x,0) = p_0(x) \text{ on } \Omega. \tag{1.3.12}$$

2.0 Domain Decomposition And Split-scheme Techniques For Biot System Of Poroelasticity Using Matching Subdomain Grids

2.1 Introduction

In this chapter we study several non-overlapping domain decomposition methods for the Biot system of poroelasticity [17], which models the flow of a viscous fluid through a poroelastic medium along with the deformation of the medium.

The numerical solution of the Biot system has been extensively studied in the literature and various references are given in the introduction chapter. We study both monolithic and split non-overlapping domain decomposition methods for the five-field fully mixed formulation of poroelasticity with weak stress symmetry from [6, 54]. Monolithic methods require solving the fully coupled Biot system, while split methods only require solving elasticity and flow problems separately. Both methods have advantages and disadvantages. Monolithic methods involve solving larger and possibly ill-conditioned algebraic systems, but may be better suitable for problems with strong coupling between flow and mechanics, in which case split or iterative coupling methods may suffer from stability or convergence issues and require sufficiently small time steps. Our methods are motivated by the non-overlapping domain decomposition methods for MFE discretizations of Darcy flow developed in [8, 26, 38] and the non-overlapping domain decomposition methods for MFE discretizations of elasticity developed recently in [48].

In the first part of the chapter we develop a monolithic domain decomposition method. Physically heterogeneous Lagrange multiplier vector consisting of displacement and pressure variables to impose weakly the continuity of the normal components of stress and velocity, respectively. This leads to solving at each time step an interface problem for this Lagrange multiplier vector. We analyze the interface operator associated with the problem and explains how a Krylov space solver such as GMRES is suitable for solving the interface problem iteratively. Each iteration requires solving monolithic Biot subdomain problems with specified Dirichlet data on the interfaces, which can be done in parallel.

In the second part of this chapter, we study split domain decomposition methods for the Biot system. Split or iterative coupling methods for poroelasticity have been extensively studied due

to their computational efficiency and various references are given in the introduction chapter. Here we study the drained split (DS) and the fixed stress (FS) decoupling methods in conjunction with non-overlapping domain decomposition technique. In particular, at each time step we solve sequentially an elasticity and a flow problem in the case of DS or a flow and an elasticity problem in the case of FS splitting. We show that the DS and FS methods are unconditionally stable with respect to the time step and the physical parameters. We show the stability of these methods using energy estimates. We then employ separate non-overlapping domain decomposition methods for each of the decoupled problems, using the methods developed in [8,38] for flow and [48] for mechanics.

In the numerical section we present several computational experiments designed to verify and compare the accuracy, stability, and computational efficiency of the three domain decomposition methods for the Biot system of poroelasticity. In particular, we study the discretization error and the number of interface iterations, as well as the effect of the number of subdomains. We also illustrate the performance of the methods for a physically realistic heterogeneous problem with data taken from the Society of Petroleum Engineers 10th Comparative Solution Project.

The rest of this chapter is organized as follows. In Section 2.2 we introduce the mathematical model and its MFE discretization. The monolithic domain decomposition method is developed and analyzed in Section 2.3. In Section 2.4 we perform stability analysis of the DS and FS decoupling methods and present the DS and FS domain decomposition methods. The numerical experiments are presented in Section 2.5, followed by conclusions in Section 2.6.

2.2 MFE Discretization

We consider a mixed variational formulation for (1.3.1)–(1.3.9) with weak stress symmetry. We follow the approach in [54]. The motivation is that MFE elasticity spaces with weakly symmetric stress tend to have fewer degrees of freedom than strongly symmetric MFE spaces. Moreover, in a recent work, a multipoint stress–multipoint flux mixed finite element method has been developed for this formulation that reduces to a positive definite cell-centered scheme for pressure and displacement only [6]. Nevertheless, the domain decomposition methods in this chapter can be employed for strongly symmetric stress formulations, with the analysis carrying

over in a straightforward way. We introduce a rotation Lagrange multiplier $\gamma := \frac{1}{2}\left(\nabla u - \nabla u^T\right) \in \mathbb{N}$, which is used to impose weakly symmetry of the stress tensor σ. We rewrite (1.3.5) as

$$A\left(\sigma + \alpha p I\right) = \nabla u - \gamma. \tag{2.2.1}$$

Combining (1.3.5) and (1.3.4) gives $\operatorname{div} u = \operatorname{tr}(\epsilon(u)) = \operatorname{tr}(A\sigma_e) = \operatorname{tr} A(\sigma + \alpha p I)$, which can be used to rewrite (1.3.3) as

$$\partial_t(c_0 p + \alpha \operatorname{tr} A\left(\sigma + \alpha p I\right)) + \operatorname{div} z = g. \tag{2.2.2}$$

The combination of (2.2.1), (1.3.1), (1.3.2), and (2.2.2), along with the boundary conditions (1.3.8)–(1.3.9), leads to the variational formulation: find $(\sigma, u, \gamma, z, p) : [0, T] \to \mathbb{X}\times V\times\mathbb{Q}\times Z\times W$ such that $p(0) = p_0$ and for a.e. $t \in (0, T)$,

$$(A(\sigma + \alpha p I), \tau) + (u, \operatorname{div}\tau) + (\gamma, \tau) = \langle g_u, \tau\, n\rangle_{\Gamma_D^u}, \qquad \forall \tau \in \mathbb{X}, \tag{2.2.3}$$

$$(\operatorname{div}\sigma, v) = -\left(f, v\right), \qquad \forall v \in V, \tag{2.2.4}$$

$$(\sigma, \xi) = 0, \qquad \forall \xi \in \mathbb{Q}, \tag{2.2.5}$$

$$\left(K^{-1} z, q\right) - (p, \operatorname{div} q) = -\langle g_p, q \cdot n\rangle_{\Gamma_D^p}, \qquad \forall q \in Z, \tag{2.2.6}$$

$$c_0\left(\partial_t p, w\right) + \alpha\left(\partial_t A(\sigma + \alpha p I), wI\right) + (\operatorname{div} z, w) = \left(g, w\right), \qquad \forall w \in W, \tag{2.2.7}$$

where

$$\mathbb{X} = \left\{\tau \in H(\operatorname{div}; \Omega, \mathbb{M}) : \tau\, n = 0 \text{ on } \Gamma_N^\sigma\right\}, \quad V = L^2(\Omega, \mathbb{R}^d), \quad \mathbb{Q} = L^2(\Omega, \mathbb{N}),$$
$$Z = \left\{q \in H(\operatorname{div}; \Omega) : q \cdot n = 0 \text{ on } \Gamma_N^z\right\}, \quad W = L^2(\Omega).$$

It was shown in [6] that the system (2.2.3)–(2.2.7) is well posed.

Next, we present the MFE discretization of (2.2.3)–(2.2.7). For simplicity we assume that Ω is a Lipshicz polygonal domain. Let $\mathcal{T}_h$ be a shape-regular quasi-uniform finite element partition of Ω, consisting of simplices or quadrilaterals, with $h = \max_{E\in\mathcal{T}_h}\operatorname{diam}(E)$. The MFE method for

solving (2.2.3)–(2.2.7) is: find $(\sigma_h, u_h, \gamma_h, z_h, p_h) : [0,T] \to \mathbb{X}_h \times V_h \times \mathbb{Q}_h \times Z_h \times W_h$ such that, for a.e. $t \in (0,T)$,

$$(A(\sigma_h + \alpha p_h I), \tau) + (u_h, \operatorname{div} \tau) + (\gamma_h, \tau) = \langle g_u, \tau\, n\rangle_{\Gamma_D^u}, \qquad \forall \tau \in \mathbb{X}_h, \tag{2.2.8}$$

$$(\operatorname{div} \sigma_h, v) = -(f, v), \qquad \forall v \in V_h, \tag{2.2.9}$$

$$(\sigma_h, \xi) = 0, \qquad \forall \xi \in \mathbb{Q}_h, \tag{2.2.10}$$

$$\left(K^{-1} z_h, q\right) - (p_h, \operatorname{div} q) = -\langle g_p, q \cdot n\rangle_{\Gamma_D^p}, \qquad \forall q \in Z_h, \tag{2.2.11}$$

$$c_0 (\partial_t p_h, w) + \alpha (\partial_t A(\sigma_h + \alpha p_h I), wI) + (\operatorname{div} z_h, w) = (g, w), \qquad \forall w \in W_h, \tag{2.2.12}$$

with discrete initial data obtained as the elliptic projection of the continuous initial data. Here $\mathbb{X}_h \times V_h \times \mathbb{Q}_h \times Z_h \times W_h \subset \mathbb{X} \times V \times \mathbb{Q} \times Z \times W$ is a collection of suitable finite element spaces. In particular, $\mathbb{X}_h \times V_h \times \mathbb{Q}_h$ could be chosen from any of the known stable triplets for linear elasticity with weak stress symmetry, e.g. [5,7,11–13,18,19,25,29,39,55,75], satisfying the inf-sup condition

$$\forall v \in V_h, \xi \in \mathbb{Q}_h, \quad \|v\| + \|\xi\| \le C \sup_{0 \neq \tau \in \mathbb{X}_h} \frac{(v, \operatorname{div} \tau) + (\xi, \tau)}{\|\tau\|_{\operatorname{div}}}. \tag{2.2.13}$$

For the flow part, $Z_h \times W_h$ could be chosen from any of the known stable velocity-pressure pairs of MFE spaces such as the Raviart-Thomas ($\mathcal{RT}$) or Brezzi-Douglas-Marini ($\mathcal{BDM}$) spaces, see [22], satisfying the inf-sup condition

$$\forall w \in W_h, \quad \|w\| \le C \sup_{0 \neq q \in Z_h} \frac{(\operatorname{div} q, w)}{\|q\|_{\operatorname{div}}}. \tag{2.2.14}$$

2.3 Monolithic Domain Decomposition Method

Let $\Omega = \cup_{i=1}^m \Omega_i$ be a union of non-overlapping shape-regular polygonal subdomains, where each subdomain is a union of elements of $\mathcal{T}_h$. Let $\Gamma_{i,j} = \partial\Omega_i \cap \partial\Omega_j$, $\Gamma = \cup_{i,j=1}^m \Gamma_{i,j}$, and $\Gamma_i = \partial\Omega_i \cap \Gamma = \partial\Omega_i \setminus \partial\Omega$ denote the interior subdomain interfaces. Denote the restriction of the spaces $\mathbb{X}_h$, V_h, $\mathbb{Q}_h$, Z_h, and W_h to Ω_i by $\mathbb{X}_{h,i}$, $V_{h,i}$, $\mathbb{Q}_{h,i}$, $Z_{h,i}$, and $W_{h,i}$, respectively. Let $\mathcal{T}_{h,i,j}$ be a finite element partition of $\Gamma_{i,j}$ obtained from the trace of $\mathcal{T}_h$, and let $n_{i,j}$ be a unit normal vector on $\Gamma_{i,j}$ with an arbitrarily fixed direction. In the domain decomposition formulation we utilize a vector Lagrange multiplier $\lambda_h = (\lambda_h^u, \lambda_h^p)^T$ approximating the displacement and the pressure on the

interface and used to impose weakly the continuity of the normal components of the poroelastic stress tensor σ and the velocity vector z, respectively. We define the Lagrange multiplier space on $\mathcal{T}_{i,j}$ and $\cup_{i<j}\mathcal{T}_{i,j}$ as follows:

$$\Lambda_{h,i,j} := \begin{pmatrix} \Lambda^u_{h,i,j} \\ \Lambda^p_{h,i,j} \end{pmatrix} := \begin{pmatrix} \mathbb{X}_h\, n_{i,j} \\ Z_h \cdot n_{i,j} \end{pmatrix}, \quad \Lambda^u_h := \bigoplus_{1\le i<j\le m} \Lambda^u_{h,i,j}, \quad \Lambda^p_h := \bigoplus_{1\le i<j\le m} \Lambda^p_{h,i,j}, \quad \Lambda_h := \begin{pmatrix} \Lambda^u_h \\ \Lambda^p_h \end{pmatrix}.$$

The domain decomposition formulation for the mixed Biot problem in a semi-discrete form reads as follows: for $1 \le i \le m$, find $(\sigma_{h,i}, u_{h,i}, \gamma_{h,i}, z_{h,i}, p_{h,i}, \lambda_h) : [0,T] \to \mathbb{X}_{h,i} \times V_{h,i} \times \mathbb{Q}_{h,i} \times Z_{h,i} \times W_{h,i} \times \Lambda_h$ such that, for a.e. $t \in (0,T)$,

$$\begin{aligned}
&(A(\sigma_{h,i} + \alpha p_{h,i} I), \tau)_{\Omega_i} + (u_{h,i}, \operatorname{div} \tau)_{\Omega_i} + (\gamma_{h,i}, \tau)_{\Omega_i} \\
&\qquad = \langle g_u, \tau\, n_i\rangle_{\partial\Omega_i \cap \Gamma^u_D} + \langle \lambda^u_h, \tau\, n_i\rangle_{\Gamma_i}, && \forall \tau \in \mathbb{X}_{h,i}, && (2.3.1)\\
&(\operatorname{div} \sigma_{h,i}, v)_{\Omega_i} = -(f, v)_{\Omega_i}, && \forall v \in V_{h,i}, && (2.3.2)\\
&(\sigma_{h,i}, \xi)_{\Omega_i} = 0, && \forall \xi \in \mathbb{Q}_{h,i}, && (2.3.3)\\
&\left(K^{-1} z_{h,i}, q\right)_{\Omega_i} - (p_{h,i}, \operatorname{div} q)_{\Omega_i} = -\langle g_p, q \cdot n_i\rangle_{\partial\Omega_i \cap \Gamma^p_D} - \langle \lambda^p_h, q \cdot n_i\rangle_{\Gamma_i}, && \forall q \in Z_{h,i}, && (2.3.4)\\
&c_0 (\partial_t p_{h,i}, w)_{\Omega_i} + \alpha (\partial_t A(\sigma_{h,i} + \alpha p_{h,i} I), wI)_{\Omega_i} + (\operatorname{div} z_{h,i}, w)_{\Omega_i} = (g, w)_{\Omega_i}, && \forall w \in W_{h,i}, && (2.3.5)\\
&\sum_{i=1}^m \langle \sigma_{h,i}\, n_i, \mu^u\rangle_{\Gamma_i} = 0, && \forall \mu^u \in \Lambda^u_h, && (2.3.6)\\
&\sum_{i=1}^m \langle z_{h,i} \cdot n_i, \mu^p\rangle_{\Gamma_i} = 0, && \forall \mu^p \in \Lambda^p_h, && (2.3.7)
\end{aligned}$$

where n_i is the outward unit normal vector field on Ω_i. We note that both the elasticity and flow subdomain problems in the above method are of Dirichlet type. It is easy to check that (2.3.1)–(2.3.7) is equivalent to the global formulation (2.2.8)–(2.2.12) with $(\sigma_h, u_h, \gamma_h, z_h, p_h)|_{\Omega_i} = (\sigma_{h,i}, u_{h,i}, \gamma_{h,i}, z_{h,i}, p_{h,i})$.

2.3.1 Time discretization

For time discretization we employ the backward Euler method. Let $\{t_n\}_{n=0}^N$, $t_n = n\Delta t$, $\Delta t = T/N$, be a uniform partition of $(0,T)$. The fully discrete problem corresponding to (2.3.1)–(2.3.7) reads as follows: for $0 \le n \le N-1$ and $1 \le i \le m$, find $(\sigma_{h,i}^{n+1}, u_{h,i}^{n+1}, \gamma_{h,i}^{n+1}, z_{h,i}^{n+1}, p_{h,i}^{n+1}, \lambda_h^{n+1}) \in \mathbb{X}_{h,i} \times V_{h,i} \times \mathbb{Q}_{h,i} \times Z_{h,i} \times W_{h,i} \times \Lambda_h$ such that:

$$\begin{aligned}
&\left(A(\sigma_{h,i}^{n+1} + \alpha p_{h,i}^{n+1} I), \tau\right)_{\Omega_i} + \left(u_{h,i}^{n+1}, \operatorname{div} \tau\right)_{\Omega_i} + \left(\gamma_{h,i}^{n+1}, \tau\right)_{\Omega_i} \\
&\qquad = \langle g_u^{n+1}, \tau\, n_i\rangle_{\partial\Omega_i \cap \Gamma_D^u} + \langle \lambda_h^{u,n+1}, \tau\, n_i\rangle_{\Gamma_i}, && \forall \tau \in \mathbb{X}_{h,i}, \quad (2.3.8)\\
&\left(\operatorname{div} \sigma_{h,i}^{n+1}, v\right)_{\Omega_i} = -\left(f^{n+1}, v\right)_{\Omega_i}, && \forall v \in V_{h,i}, \quad (2.3.9)\\
&\left(\sigma_{h,i}^{n+1}, \xi\right)_{\Omega_i} = 0, && \forall \xi \in \mathbb{Q}_{h,i}, \quad (2.3.10)\\
&\left(K^{-1} z_{h,i}^{n+1}, q\right)_{\Omega_i} - \left(p_{h,i}^{n+1}, \operatorname{div} q\right)_{\Omega_i} = -\langle g_p^{n+1}, q \cdot n_i\rangle_{\partial\Omega_i \cap \Gamma_D^p} - \langle \lambda_h^{p,n+1}, q \cdot n_i\rangle_{\Gamma_i}, && \forall q \in Z_{h,i}, \quad (2.3.11)\\
&c_0 \left(\frac{p_{h,i}^{n+1} - p_{h,i}^n}{\Delta t}, w\right)_{\Omega_i} + \alpha \left(\frac{A\left(\sigma_{h,i}^{n+1} - \sigma_{h,i}^n\right)}{\Delta t}, wI\right)_{\Omega_i} \\
&\qquad + \alpha \left(A\alpha \frac{p_{h,i}^{n+1} - p_{h,i}^n}{\Delta t} I, wI\right)_{\Omega_i} + \left(\operatorname{div} z_{h,i}^{n+1}, w\right)_{\Omega_i} = \left(g^{n+1}, w\right)_{\Omega_i}, && \forall w \in W_{h,i}, \quad (2.3.12)\\
&\sum_{i=1}^m \langle \sigma_{h,i}^{n+1}\, n_i, \mu^u\rangle_{\Gamma_i} = 0, && \forall \mu^u \in \Lambda_h^u, \quad (2.3.13)\\
&\sum_{i=1}^m \langle z_{h,i}^{n+1} \cdot n_i, \mu^p\rangle_{\Gamma_i} = 0, && \forall \mu^p \in \Lambda_h^p. \quad (2.3.14)
\end{aligned}$$

Remark 2.3.1. *We note that the scheme requires initial data $p_{h,i}^0$ and $\sigma_{h,i}^0$. Such data can be obtained by taking $p_{h,i}^0$ to be the L^2-projection of p_0 onto $W_{h,i}$ and solving a mixed elasticity domain decomposition problem obtained from (2.3.8)–(2.3.10) and (2.3.13) with $n = -1$.*

2.3.2 Time-differentiated elasticity formulation

In the monolithic domain decomposition method we will utilize a related formulation in which the first elasticity equation is differentiated in time. The reason for this will become clear in the analysis of the resulting interface problem. We introduce new variables $\dot{u} = \partial_t u$ and $\dot{\gamma} = \partial_t \gamma$ representing the time derivatives of the displacement and the rotation, respectively. The time-differentiated equation (2.2.3) is

$$(\partial_t A(\sigma + \alpha p I), \tau) + (\dot{u}, \operatorname{div} \tau) + (\dot{\gamma}, \tau) = \langle \partial_t g_u, \tau\, n \rangle_{\Gamma_D^u}, \quad \forall\, \tau \in \mathbb{X}.$$

The semi-discrete equation (2.2.8) is replaced by

$$(\partial_t A(\sigma_h + \alpha p_h I), \tau) + (\dot{u}_h, \operatorname{div} \tau) + (\dot{\gamma}_h, \tau) = \langle \partial_t g_u, \tau\, n \rangle_{\Gamma_D^u}, \forall\, \tau \in \mathbb{X}_h.$$

We note that the original variables u_h and γ_h can be recovered easily from the solution of the time-differentiated problem. In particular, given compatible initial data $\sigma_{h,0}$, $u_{h,0}$, $\gamma_{h,0}$ that satisfy (2.2.8), the expressions

$$u_h(t) = u_{h,0} + \int_0^t \dot{u}_h(s)\, ds, \quad \gamma_h(t) = \gamma_{h,0} + \int_0^t \dot{\gamma}_h(s)\, ds,$$

provide a solution to (2.2.8) at any $t \in (0, T]$.

In the domain decomposition formulation we now consider the Lagrange multiplier $\lambda_h = (\lambda_h^{\dot{u}}, \lambda_h^p) \in \Lambda_h$, where $\lambda_h^{\dot{u}} \in \Lambda_h^u$ approximates the trace of $\dot{u}$ on Γ. Then the semi-discrete domain decomposition equation (2.3.1) is replaced by

$$(\partial_t A(\sigma_{h,i} + \alpha p_{h,i} I), \tau)_{\Omega_i} + (\dot{u}_{h,i}, \operatorname{div} \tau)_{\Omega_i} + (\dot{\gamma}_{h,i}, \tau)_{\Omega_i} = \langle \partial_t g_u, \tau\, n_i \rangle_{\partial\Omega_i \cap \Gamma_D^u} + \langle \lambda_h^{\dot{u}}, \tau\, n_i \rangle_{\Gamma_i}, \forall \tau \in \mathbb{X}_{h,i}.$$

Finally, the fully discrete equation (2.3.8) is replaced by

$$\begin{aligned} &\left(A(\sigma_{h,i}^{n+1} + \alpha p_{h,i}^{n+1} I), \tau\right)_{\Omega_i} + \Delta t \left(\dot{u}_{h,i}^{n+1}, \operatorname{div} \tau\right)_{\Omega_i} + \Delta t \left(\dot{\gamma}_{h,i}^{n+1}, \tau\right)_{\Omega_i} \\ &\quad = \Delta t \langle \partial_t g_u^{n+1}, \tau\, n_i \rangle_{\partial\Omega_i \cap \Gamma_D^u} + \Delta t \langle \lambda_h^{\dot{u},n+1}, \tau\, n_i \rangle_{\Gamma_i} + \left(A(\sigma_{h,i}^n + \alpha p_{h,i}^n I), \tau\right)_{\Omega_i}, \quad \forall \tau \in \mathbb{X}_{h,i}. \end{aligned} \tag{2.3.15}$$

The original variables can be recovered from

$$u_h^n = u_h^0 + \Delta t \sum_{k=1}^n \dot{u}_h^k, \quad \gamma_h^n = \gamma_h^0 + \Delta t \sum_{k=1}^n \dot{\gamma}_h^k, \quad \lambda_h^{u,n} = \lambda_h^{u,0} + \sum_{k=1}^n \lambda_h^{\dot{u},k}. \tag{2.3.16}$$

2.3.3 Reduction to an interface problem

The non-overlapping domain decomposition algorithm for the solution of (2.3.15), (2.3.9)–(2.3.14) at each time step is based on reducing it to an interface problem for the Lagrange multiplier λ_h. To this end, we introduce two sets of complementary subdomain problems. The first set of problems reads as follows: for $1 \leq i \leq m$, find $(\bar{\sigma}_{h,i}^{n+1}, \bar{u}_{h,i}^{n+1}, \bar{\dot{\gamma}}_{h,i}^{n+1}, \bar{z}_{h,i}^{n+1}, \bar{p}_{h,i}^{n+1}) \in \mathbb{X}_{h,i} \times V_{h,i} \times \mathbb{Q}_{h,i} \times Z_{h,i} \times W_{h,i}$ such that

$$\begin{aligned}
&\left(A(\bar{\sigma}_{h,i}^{n+1} + \alpha\bar{p}_{h,i}^{n+1} I), \tau\right)_{\Omega_i} + \Delta t \left(\bar{u}_{h,i}^{n+1}, \operatorname{div} \tau\right)_{\Omega_i} + \Delta t \left(\bar{\dot{\gamma}}_{h,i}^{n+1}, \tau\right)_{\Omega_i} \\
&\qquad = \Delta t \langle \partial_t g_u^{n+1}, \tau\, n_i \rangle_{\partial\Omega_i \cap \Gamma_D^u} + \left(A(\sigma_{h,i}^n + \alpha p_{h,i}^n I), \tau\right)_{\Omega_i}, && \forall \tau \in \mathbb{X}_{h,i}, && (2.3.17)\\
&\left(\operatorname{div} \bar{\sigma}_{h,i}^{n+1}, v\right)_{\Omega_i} = -\left(f^{n+1}, v\right)_{\Omega_i}, && \forall v \in V_{h,i}, && (2.3.18)\\
&\left(\bar{\sigma}_{h,i}^{n+1}, \xi\right)_{\Omega_i} = 0, && \forall \xi \in \mathbb{Q}_{h,i}, && (2.3.19)\\
&\left(K^{-1}\bar{z}_{h,i}^{n+1}, q\right)_{\Omega_i} - \left(\bar{p}_{h,i}^{n+1}, \operatorname{div} q\right)_{\Omega_i} = -\langle g_p^{n+1}, q \cdot n_i \rangle_{\partial\Omega_i \cap \Gamma_D^p}, && \forall q \in Z_{h,i}, && (2.3.20)\\
&c_0 \left(\bar{p}_{h,i}^{n+1}, w\right)_{\Omega_i} + \alpha \left(A(\bar{\sigma}_{h,i}^{n+1} + \alpha\bar{p}_{h,i}^{n+1} I), wI\right)_{\Omega_i} + \Delta t \left(\operatorname{div} \bar{z}_{h,i}^{n+1}, w\right)_{\Omega_i} \\
&\qquad = \Delta t \left(g^{n+1}, w\right)_{\Omega_i} + c_0 \left(p_{h,i}^n, w\right)_{\Omega_i} + \alpha \left(A(\sigma_{h,i}^n + \alpha p_{h,i}^n I), wI\right)_{\Omega_i}, && \forall w \in W_{h,i}. && (2.3.21)
\end{aligned}$$

These subdomain problems have zero Dirichlet data on the interfaces and incorporate the true source terms f and g and outside boundary conditions g_u and g_p, as well as initial data $\sigma_{h,i}^n$ and $p_{h,i}^n$.

The second problem set reads as follows: given $\lambda_h \in \Lambda_h$, for $1 \leq i \leq m$, find $\left(\sigma_{h,i}^{*,n+1}(\lambda_h), \dot{u}_{h,i}^{*,n+1}(\lambda_h), \dot{\gamma}_{h,i}^{*,n+1}(\lambda_h), z_{h,i}^{*,n+1}(\lambda_h), p_{h,i}^{*,n+1}(\lambda_h)\right) \in \mathbb{X}_{h,i} \times V_{h,i} \times \mathbb{Q}_{h,i} \times Z_{h,i} \times W_{h,i}$ such that:

$$\begin{aligned}
&\left(A\big(\sigma_{h,i}^{*,n+1}(\lambda_h) + \alpha p_{h,i}^{*,n+1}(\lambda_h) I\big), \tau\right)_{\Omega_i} + \Delta t \left(\dot{u}_{h,i}^{*,n+1}(\lambda_h), \operatorname{div} \tau\right)_{\Omega_i} \\
&\qquad + \Delta t \left(\dot{\gamma}_{h,i}^{*,n+1}(\lambda_h), \tau\right)_{\Omega_i} = \Delta t \langle \lambda_h^{\dot{u}}, \tau\, n_i \rangle_{\Gamma_i}, && \forall \tau \in \mathbb{X}_{h,i}, && (2.3.22)\\
&\left(\operatorname{div} \sigma_{h,i}^{*,n+1}(\lambda_h), v\right)_{\Omega_i} = 0, && \forall v \in V_{h,i}, && (2.3.23)\\
&\left(\sigma_{h,i}^{*,n+1}(\lambda_h), \xi\right)_{\Omega_i} = 0, && \forall \xi \in \mathbb{Q}_{h,i}, && (2.3.24)\\
&\left(K^{-1} z_{h,i}^{*,n+1}(\lambda_h), q\right)_{\Omega_i} - \left(p_{h,i}^{*,n+1}(\lambda_h), \operatorname{div} q\right)_{\Omega_i} = -\langle \lambda_h^p, q \cdot n_i \rangle_{\Gamma_i}, && \forall q \in Z_{h,i}, && (2.3.25)\\
&c_0 \left(p_{h,i}^{*,n+1}(\lambda_h), w\right)_{\Omega_i} + \alpha \left(A\big(\sigma_{h,i}^{*,n+1}(\lambda_h) + \alpha p_{h,i}^{*,n+1}(\lambda_h) I\big), wI\right)_{\Omega_i} \\
&\qquad + \Delta t \left(\operatorname{div} z_{h,i}^{*,n+1}(\lambda_h), w\right)_{\Omega_i} = 0, && \forall w \in W_{h,i}. && (2.3.26)
\end{aligned}$$

These problems have λ_h as Dirichlet interface data, along with zero source terms, zero outside boundary conditions, and zero data from the previous time step.

Define the bilinear forms $a_i^{n+1} : \Lambda_h \times \Lambda_h \to \mathbb{R}$, $1 \le i \le m$, $a^{n+1} : \Lambda_h \times \Lambda_h \to \mathbb{R}$, and the linear functional $g^{n+1} : \Lambda_h \to \mathbb{R}$ for all $0 \le n \le N-1$ by

$$a_i^{n+1}(\lambda_h, \mu) = \langle \sigma_{h,i}^{*,n+1}(\lambda_h)\, n_i, \mu^u \rangle_{\Gamma_i} - \langle z_{h,i}^{*,n+1}(\lambda_h) \cdot n_i, \mu^p \rangle_{\Gamma_i}, \quad a^{n+1}(\lambda_h, \mu) = \sum_{i=1}^{m} a_i^{n+1}(\lambda_h, \mu), \tag{2.3.27}$$

$$g^{n+1}(\mu) = \sum_{i=1}^{m} \left(-\langle \bar{\sigma}_{h,i}^{n+1}\, n_i, \mu^u \rangle_{\Gamma_i} + \langle \bar{z}_{h,i}^{n+1} \cdot n_i, \mu^p \rangle_{\Gamma_i} \right). \tag{2.3.28}$$

It follows from (2.3.13)–(2.3.14) that, for each $0 \le n \le N-1$, the solution to the global problem (2.3.15), (2.3.9)–(2.3.14) is equivalent to solving the interface problem for $\lambda_h^{n+1} \in \Lambda_h$:

$$a^{n+1}(\lambda_h^{n+1}, \mu) = g^{n+1}(\mu), \quad \forall \mu \in \Lambda_h, \tag{2.3.29}$$

and setting

$$\begin{aligned} &\sigma_{h,i}^{n+1} = \sigma_{h,i}^{*,n+1}(\lambda_h^{n+1}) + \bar{\sigma}_{h,i}^{n+1}, \quad \dot{u}_{h,i}^{n+1} = \dot{u}_{h,i}^{*,n+1}(\lambda_h^{n+1}) + \bar{\dot{u}}_{h,i}^{n+1}, \quad \dot{\gamma}_{h,i}^{n+1} = \dot{\gamma}_{h,i}^{*,n+1}(\lambda_h^{n+1}) + \bar{\dot{\gamma}}_{h,i}^{n+1}, \\ &z_{h,i}^{n+1} = z_{h,i}^{*,n+1}(\lambda_h^{n+1}) + \bar{z}_{h,i}^{n+1}, \quad p_{h,i}^{n+1} = p_{h,i}^{*,n+1}(\lambda_h^{n+1}) + \bar{p}_{h,i}^{n+1}. \end{aligned}$$

2.3.4 Analysis of the interface problem

We next show that the interface bilinear form $a^{n+1}(\cdot,\cdot)$ is positive definite, which implies that the interface problem (2.3.29) is well-posed and can be solved using a suitable Krylov space method such as GMRES. We further obtain bounds on the spectrum of $a^{n+1}(\cdot,\cdot)$ and establish rate of convergence for GMRES. We start by obtaining an expression for $a^{n+1}(\cdot,\cdot)$ in terms of the subdomain bilinear forms.

Proposition 2.3.1. *For $\lambda_h, \mu \in \Lambda_h$, the interface bilinear form can be expressed as*

$$\begin{aligned} a^{n+1}(\lambda_h, \mu) = \frac{1}{\Delta t} \sum_{i=1}^{m} \Big[&\left(A\sigma_{h,i}^{*,n+1}(\mu), \sigma_{h,i}^{*,n+1}(\lambda_h) \right) + 2 \left(A\alpha p_{h,i}^{*,n+1}(\mu) I, \sigma_{h,i}^{*,n+1}(\lambda_h) \right)_{\Omega_i} \\ &+ \left(A\alpha p_{h,i}^{*,n+1}(\mu) I, \alpha p_{h,i}^{*,n+1}(\lambda_h) I \right)_{\Omega_i} + c_0 \left(p_{h,i}^{*,n+1}(\mu), p_{h,i}^{*,n+1}(\lambda_h) \right)_{\Omega_i} \\ &+ \Delta t \left(K^{-1} z_{h,i}^{*,n+1}(\mu), z_{h,i}^{*,n+1}(\lambda_h) \right) \Big]. \end{aligned} \tag{2.3.30}$$

Proof. To see this, consider the second set of complementary equations (2.3.22−2.3.26) with data μ, use the test functions: $\sigma_{h,i}^{*,n+1}(\lambda_h)$ in (2.3.22) and $z_{h,i}^{*,n+1}(\lambda_h)$ in equation (2.3.25) . □

Remark 2.3.2. *The non-differentiated formulation results in a missing scaling of* $\frac{1}{\Delta t}$ *in the term* $\big(A\big(\sigma_{h,i}^{*,n+1}(\lambda_h)+\alpha p_{h,i}^{*,n+1}(\lambda_h)I\big),\tau\big)_{\Omega_i}$ *in* (2.3.22) *compared to the similar term in* (2.3.26)*. Hence the two terms cannot be combined, resulting in a non-coercive expression for* $a^{n+1}(\cdot,\cdot)$*.*

Recalling the properties of A and K, there exist constants $0 < a_{\min} \leq a_{\max} < \infty$ and $0 < k_{\min} \leq k_{\max} < \infty$ such that

$$a_{\min}\|\tau\|^2 \leq (A\tau,\tau) \leq a_{\max}\|\tau\|^2, \quad \forall\, \tau \in \mathbb{X}, \tag{2.3.31}$$

$$k_{\min}\|q\|^2 \leq (Kq,q) \leq k_{\max}\|q\|^2, \quad \forall\, q \in Z. \tag{2.3.32}$$

We will also utilize suitable mixed interpolants in the finite element spaces $\mathbb{X}_{h,i}$ and $Z_{h,i}$. It is shown in [48] that there exists an interpolant $\tilde{\Pi}_i : H^\epsilon(\Omega_i,\mathbb{M}) \cap \mathbb{X}_i \to \mathbb{X}_{h,i}$ for any $\epsilon > 0$ such that for all $\sigma \in H^\epsilon(\Omega_i,\mathbb{M}) \cap \mathbb{X}_i$, $\tau \in \mathbb{X}_{h,i}$, $v \in V_{h,i}$, and $\xi \in \mathbb{Q}_{h,i}$,

$$(\operatorname{div}(\tilde{\Pi}_i\sigma - \sigma), v)_{\Omega_i} = 0, \quad (\tilde{\Pi}_i\sigma - \sigma, \xi)_{\Omega_i} = 0, \quad \langle(\tilde{\Pi}_i\sigma - \sigma)n_i, \tau\, n_i\rangle_{\partial\Omega_i} = 0, \tag{2.3.33}$$

and

$$\|\tilde{\Pi}_i\sigma\|_{\Omega_i} \leq C(\|\sigma\|_{\epsilon,\Omega_i} + \|\operatorname{div}\sigma\|_{\Omega_i}). \tag{2.3.34}$$

For the Darcy problem we use the canonical mixed interpolant [22], $\Pi : H^\epsilon(\Omega_i,\mathbb{R}^d) \cap Z_i \to Z_{h,i}$ such that for all $z \in H^\epsilon(\Omega_i,\mathbb{R}^d) \cap Z_i$, $q \in Z_{h,i}$, and $w \in W_{h,i}$,

$$(\operatorname{div}(\Pi_i z - z), w)_{\Omega_i} = 0, \quad \langle(\Pi_i z - z)\cdot n_i, q\cdot n_i\rangle_{\partial\Omega_i} = 0, \tag{2.3.35}$$

and

$$\|\Pi_i z\|_{\Omega_i} \leq C(\|z\|_{\epsilon,\Omega_i} + \|\operatorname{div} z\|_{\Omega_i}). \tag{2.3.36}$$

Lemma 2.3.2. *The interface bilinear form* $a^{n+1}(\cdot,\cdot)$ *is positive definite over* Λ_h*.*

Proof. Using the representation of the interface bilinear form (2.3.30), we get

$$a^{n+1}(\lambda_h,\lambda_h) = \frac{1}{\Delta t}\sum_{i=1}^{m}\Big[\left(A(\psi_\sigma(\lambda_h)+\alpha p_{h,i}^{*,n+1}(\lambda_h)I),\sigma_{h,i}^{*,n+1}(\lambda_h)+\alpha p_{h,i}^{*,n+1}(\lambda_h)I\right)_{\Omega_i} + c_0\left(p_{h,i}^{*,n+1}(\lambda_h),p_{h,i}^{*,n+1}(\lambda_h)\right)_{\Omega_i} + \Delta t\left(K^{-1}z_{h,i}^{*,n+1}(\lambda_h),z_{h,i}^{*,n+1}(\lambda_h)\right)_{\Omega_i}\Big], \quad (2.3.37)$$

which, combined with (2.3.31)–(2.3.32), gives $a^{n+1}(\lambda_h,\lambda_h)\geq 0$, and hence $a^{n+1}(\cdot,\cdot)$ is positive semidefinite. We next show that $a(\lambda_h,\lambda_h)=0$ implies $\lambda_h=0$. We use a two-part argument to control separately $\lambda_h^{\dot{u}}$ and λ_h^{p}. Let Ω_i be a domain adjacent to Γ_D^u such that $|\partial\Omega_i\cap\Gamma_D^u|>0$ and let $(\psi^{\dot{u}},\phi^{\dot{u}})$ be the solution of the auxiliary elasticity problem

$$\begin{aligned} &A\psi_i^{\dot{u}} = \epsilon(\phi_i^{\dot{u}}), \quad \operatorname{div}\psi_i^{\dot{u}} = 0 \quad \text{in } \Omega_i,\\ &\phi_i^{\dot{u}} = 0 \quad \text{on } \partial\Omega_i\cap\Gamma_D^u,\\ &\psi_i^{\dot{u}}\,n_i = \begin{cases} 0 & \text{on } \partial\Omega_i\cap\Gamma_N^\sigma\\ \lambda_h^{\dot{u}} & \text{on } \Gamma_i. \end{cases}\end{aligned}$$

Elliptic regularity [27] implies that $\psi_i^{\dot{u}}\in H^\epsilon(\Omega_i,\mathbb{M})\cap\mathbb{X}_i$ for some $\epsilon>0$, and therefore the mixed interpolant $\tilde{\Pi}_i\psi_i^{\dot{u}}$ is well defined. Taking $\tau=\tilde{\Pi}_i\psi_i^{\dot{u}}$ in (2.3.22) and using (2.3.33) and (2.3.34) gives

$$\begin{aligned}\|\lambda_h^{\dot{u}}\|_{\Gamma_i}^2 &= \langle\lambda_h^{\dot{u}},\psi_i^{\dot{u}}\,n_i\rangle_{\Gamma_i} = \langle\lambda_h^{\dot{u}},\tilde{\Pi}\psi_i^{\dot{u}}\,n_i\rangle_{\Gamma_i}\\ &= \frac{1}{\Delta t}\left(A(\sigma_{h,i}^{*,n+1}(\lambda_h)+\alpha p_{h,i}^{*,n+1}(\lambda_h)I),\tilde{\Pi}\psi_i^{\dot{u}}\right)_{\Omega_i} + \left(\dot{u}_{h,i}^{*,n+1}(\lambda_h),\operatorname{div}\tilde{\Pi}\psi_i^{\dot{u}}\right)_{\Omega_i}\\ &+ \left(\dot{\gamma}_{h,i}^{*,n+1}(\lambda_h),\tilde{\Pi}\psi_i^{\dot{u}}\right)_{\Omega_i}\\ &= \frac{1}{\Delta t}\left(A^{1/2}(\sigma_{h,i}^{*,n+1}(\lambda_h)+\alpha p_{h,i}^{*,n+1}(\lambda_h)I),A^{1/2}\tilde{\Pi}\psi_i^{\dot{u}}\right)_{\Omega_i}\\ &\leq \frac{C}{\Delta t}\|A^{1/2}(\sigma_{h,i}^{*,n+1}(\lambda_h)+\alpha p_{h,i}^{*,n+1}(\lambda_h)I)\|_{\Omega_i}\|\psi_i^{\dot{u}}\|_{\epsilon,\Omega_i}\\ &\leq \frac{C}{\Delta t}\|A^{1/2}(\sigma_{h,i}^{*,n+1}(\lambda_h)+\alpha p_{h,i}^{*,n+1}(\lambda_h)I)\|_{\Omega_i}\|\lambda_h^{\dot{u}}\|_{\Gamma_i}, \end{aligned} \quad (2.3.38)$$

where in the last inequality we used the elliptic regularity bound [27]

$$\|\psi_i^{\dot{u}}\|_{\epsilon,\Omega_i}\leq C\|\lambda_h\|_{\epsilon-1/2,\Gamma_i}. \quad (2.3.39)$$

Using the representation of the interface bilinear form (2.3.30), we obtain

$$\|\lambda_h^{\dot{u}}\|_{\Gamma_i}^2 \leq \frac{C}{\Delta t} a_i^{n+1}(\lambda_h, \lambda_h) \quad \forall\, \lambda_h \in \Lambda_h. \tag{2.3.40}$$

Next, consider an adjacent subdomain Ω_j such that $|\Gamma_{ij}| > 0$. Let $(\psi_j^{\dot{u}}, \phi_j^{\dot{u}})$ be the solution to

$$\begin{aligned}
&A\psi_j^{\dot{u}} = \epsilon(\phi_j^{\dot{u}}), \quad \operatorname{div} \psi_j^{\dot{u}} = 0 \quad \text{in } \Omega_j,\\
&\phi_i^{\dot{u}} = 0 \quad \text{on } \Gamma_{ij},\\
&\psi_i^{\dot{u}}\, n_i = \begin{cases} 0 & \text{on } \partial\Omega_j \cap \partial\Omega \\ \lambda_h^{\dot{u}} & \text{on } \Gamma_j \setminus \Gamma_{ij}. \end{cases}
\end{aligned}$$

Taking $\tau = \tilde{\Pi}_j \psi_j^{\dot{u}}$ in (2.3.22) and using (2.3.33) gives

$$\begin{aligned}
\|\lambda_h^{\dot{u}}\|_{\Gamma_j \setminus \Gamma_{ij}}^2 &= \frac{1}{\Delta t}\left(A(\sigma_{h,j}^{*,n+1}(\lambda_h) + \alpha p_{h,j}^{*,n+1}(\lambda_h) I), \tilde{\Pi}\psi_j^{\dot{u}}\right)_{\Omega_j} - \langle \lambda_h^{\dot{u}}, \psi_j^{\dot{u}}\, n_j\rangle_{\Gamma_{ij}} \\
&\leq C\left(\frac{1}{\Delta t}\|A^{1/2}(\sigma_{h,j}^{*,n+1}(\lambda_h) + \alpha p_{h,j}^{*,n+1}(\lambda_h) I)\|_{\Omega_j} + \|\lambda_h^{\dot{u}}\|_{\Gamma_{ij}}\right)\|\psi_j^{\dot{u}}\|_{\epsilon,\Omega_j} \\
&\leq \frac{C}{\sqrt{\Delta t}}\left(a_j^{n+1}(\lambda_h, \lambda_h)^{1/2} + a_i^{n+1}(\lambda_h, \lambda_h)^{1/2}\right)\|\lambda_h^{\dot{u}}\|_{\Gamma_j \setminus \Gamma_{ij}},
\end{aligned}$$

where in the first inequality we used (2.3.34) and the trace inequality [58]

$$\langle \tau\, n_j, \mu\rangle_{\Gamma_{ij}} \leq C(\|\tau\|_{\epsilon,\Omega_j} + \|\operatorname{div}\tau\|_{\Omega_j})\|\mu\|_{\Gamma_{ij}}, \quad \forall\, \tau \in H^{\epsilon}(\Omega_j, \mathbb{M}) \cap \mathbb{X}_j,\ \mu \in L^2(\Gamma_{ij}, \mathbb{R}^d),$$

and for the second inequality we used the representation (2.3.30) and the bound from Ω_i (2.3.40), along with the elliptic regularity bound (2.3.39). Iterating over all subdomains in a similar fashion results in

$$\|\lambda_h^{\dot{u}}\|_{\Gamma}^2 \leq \frac{C}{\Delta t} a^{n+1}(\lambda_h, \lambda_h) \quad \forall\, \lambda_h \in \Lambda_h. \tag{2.3.41}$$

The argument for λ_h^p is similar. We start with a subdomain Ω_i adjacent to Γ_D^p such that $|\partial\Omega_i \cap \Gamma_D^p| > 0$ and let (ψ^p, ϕ^p) be the solution of the auxiliary flow problem

$$K^{-1}\psi_i^p = \nabla\phi_i^p, \quad \nabla\cdot\psi_i^p = 0 \quad \text{in } \Omega_i, \tag{2.3.42}$$

$$\phi_i^p = 0 \quad \text{on } \partial\Omega_i \cap \Gamma_D^p, \tag{2.3.43}$$

$$\psi_i^p \cdot n_i = \begin{cases} 0 & \text{on } \partial\Omega_i \cap \Gamma_N^z, \\ \lambda_h^p & \text{on } \Gamma_i. \end{cases} \tag{2.3.44}$$

Taking $q = \Pi_i \psi_i^p$ in (2.3.25) and using (2.3.35), (2.3.36), and elliptic regularity similar to (2.3.39) gives

$$\begin{aligned}\|\lambda_h^p\|_{\Gamma_i}^2 &= \langle \lambda_h^p, \psi_i^p \cdot n_i \rangle_{\Gamma_i} = \langle \lambda_h^p, \Pi_i \psi_i^p \cdot n_i \rangle_{\Gamma_i} = (K^{-1} z_{h,i}^{*,n+1}(\lambda_h), \Pi_i \psi_i^p)_{\Omega_i} \\ &\leq C \|K^{-1/2} z_{h,i}^{*,n+1}(\lambda_h)\|_{\Omega_i} \|\psi_i^p\|_{\epsilon,\Omega_i} \leq C \|K^{-1/2} z_{h,i}^{*,n+1}(\lambda_h)\|_{\Omega_i} \|\lambda_h^p\|_{\Gamma_i},\end{aligned}$$

which, together with (2.3.30) implies

$$\|\lambda_h^p\|_{\Gamma_i}^2 \leq C a_i^{n+1}(\lambda_h, \lambda_h).$$

Iterating over all subdomains in a way similar to the argument for $\lambda_h^{\dot{u}}$, we obtain

$$\|\lambda_h^p\|_{\Gamma}^2 \leq C a^{n+1}(\lambda_h, \lambda_h) \quad \forall\, \lambda_h \in \Lambda_h. \tag{2.3.45}$$

A combination of (2.3.41) and (2.3.45) implies that $a^{n+1}(\cdot,\cdot)$ is positive definite on Λ_h. □

Theorem 2.3.3. *There exist positive constants C_0 and C_1 independent of h and Δt such that*

$$\forall \lambda_h \in \Lambda_h, \qquad C_0(\Delta t \|\lambda_h^{\dot{u}}\|_{\Gamma}^2 + \|\lambda_h^p\|_{\Gamma}^2) \leq a^{n+1}(\lambda_h, \lambda_h) \leq C_1 h^{-1}(\Delta t \|\lambda_h^{\dot{u}}\|_{\Gamma}^2 + \|\lambda_h^p\|_{\Gamma}^2). \tag{2.3.46}$$

In addition, there exist positive constants $\tilde{C}_0$ and $\tilde{C}_1$ independent of h, Δt, and c_0 such that

$$\forall \lambda_h \in \Lambda_h, \qquad \tilde{C}_0(\Delta t \|\lambda_h^{\dot{u}}\|_{\Gamma}^2 + \|\lambda_h^p\|_{\Gamma}^2) \leq a^{n+1}(\lambda_h, \lambda_h) \leq \tilde{C}_1 h^{-1} \Delta t^{-1}(\Delta t \|\lambda_h^{\dot{u}}\|_{\Gamma}^2 + \|\lambda_h^p\|_{\Gamma}^2). \tag{2.3.47}$$

Proof. The left inequality in (2.3.46) and (2.3.47) follows from (2.3.41) and (2.3.45). To prove the right inequality, we use the definition of the interface operator (2.3.27) to obtain

$$\begin{aligned}a_i^{n+1}(\lambda_h, \lambda_h) &= \langle \sigma_{h,i}^{*,n+1}(\lambda_h)\, n_i, \lambda_h^{\dot{u}} \rangle_{\Gamma_i} - \langle z_{h,i}^{*,n+1}(\lambda_h) \cdot n_i, \lambda_h^p \rangle_{\Gamma_i} \\ &\leq \|\sigma_{h,i}^{*,n+1}(\lambda_h)\, n_i\|_{\Gamma_i} \|\lambda_h^{\dot{u}}\|_{\Gamma_i} + \|z_{h,i}^{*,n+1}(\lambda_h) \cdot n_i\|_{\Gamma_i} \|\lambda_h^p\|_{\Gamma_i} \\ &\leq C h^{-1/2} \left(\|\sigma_{h,i}^{*,n+1}(\lambda_h)\|_{\Omega_i} \|\lambda_h^{\dot{u}}\|_{\Gamma_i} + \|z_{h,i}^{*,n+1}(\lambda_h)\|_{\Omega_i} \|\lambda_h^p\|_{\Gamma_i} \right) \\ &\leq C h^{-1/2} \big(\big(\|\sigma_{h,i}^{*,n+1}(\lambda_h) + \alpha p_{h,i}^{*,n+1}(\lambda_h) I\|_{\Omega_i} + \|\alpha p_{h,i}^{*,n+1}(\lambda_h) I\| \big) \|\lambda_h^{\dot{u}}\|_{\Gamma_i} \\ &+ \|z_{h,i}^{*,n+1}(\lambda_h)\|_{\Omega_i} \|\lambda_h^p\|_{\Gamma_i} \big) \leq C h^{-1/2} a_i^{n+1}(\lambda_h, \lambda_h)^{1/2} \left(\Delta t^{1/2} \|\lambda_h^{\dot{u}}\|_{\Gamma_i} + \|\lambda_h^p\|_{\Gamma_i} \right),\end{aligned} \tag{2.3.48}$$

where for the second inequality we used the discrete trace inequality for finite element functions φ,

$$\|\varphi\|_{\Gamma_i} \leq C h^{-1/2} \|\varphi\|_{\Omega_i}, \tag{2.3.49}$$

and the last inequality follows from (2.3.37). We note that the constant in the last inequality depends on c_0. This implies the right inequality in (2.3.46).

To obtain the right inequality in (2.3.47) with a constant independent of c_0, we use the inf-sup condition (2.2.14) and (2.3.25):

$$\begin{aligned}\|p_{h,i}^{*,n+1}(\lambda_h)\|_{\Omega_i} &\leq C \sup_{0\neq q\in Z_{h,i}} \frac{\langle \operatorname{div} q, p_{h,i}^{*,n+1}(\lambda_h)\rangle_{\Omega_i}}{\|q\|_{\operatorname{div},\Omega_i}} = C \sup_{0\neq q\in Z_{h,i}} \frac{(K^{-1}z_{h,i}^{*,n+1}(\lambda_h), q)_{\Omega_i} + \langle \lambda_h^p, q\cdot n_i\rangle_{\Gamma_i}}{\|q\|_{\operatorname{div},\Omega_i}} \\ &\leq C\left(\|z_{h,i}^{*,n+1}(\lambda_h)\|_{\Omega_i} + h^{-1/2}\|\lambda_h^p\|_{\Gamma_i}\right), \end{aligned} \tag{2.3.50}$$

where the last inequality uses (2.3.49). Combining (2.3.50) with the next to last inequality in (2.3.48) and using (2.3.37), we get:

$$\begin{aligned} &a_i^{n+1}(\lambda_h,\lambda_h) \\ &\quad\leq Ch^{-1/2}\left(\left(\Delta t^{1/2}a_i(\lambda_h,\lambda_h)^{1/2} + a_i(\lambda_h,\lambda_h)^{1/2} + h^{-1/2}\|\lambda_h^p\|_{\Gamma_i}\right)\|\lambda_h^{\dot{u}}\|_{\Gamma_i} + a_i(\lambda_h,\lambda_h)^{1/2}\|\lambda_h^p\|_{\Gamma_i}\right) \\ &\quad\leq C\left(\epsilon a_i^{n+1}(\lambda_h,\lambda_h) + \frac{1}{\epsilon}h^{-1}\Delta t^{-1}(\Delta t\|\lambda_h^{\dot{u}}\|_{\Gamma_i}^2 + \|\lambda_h^p\|_{\Gamma_i})\right), \end{aligned}$$

using Young's inequality in the last inequality. Taking ϵ sufficiently small implies the right inequality in (2.3.47). □

Theorem 2.3.3 provides upper and lower bounds on the field of values of the interface operator, which can be used to estimate the convergence of the interface GMRES solver. In particular, let $\mathbf{r}_k = (\mathbf{r}_k^{\dot{u}}, \mathbf{r}_k^p)$ be the k-th residual of the GMRES iteration for solving the interface problem (2.3.29). Define $|\mathbf{r}_k|_\star^2 = \Delta t|\mathbf{r}_k^{\dot{u}}|^2 + |\mathbf{r}_k^p|^2$, where $|\cdot|$ denotes the Euclidean vector norm. The following corollary to Theorem 2.3.3 follows from the field-of-values analysis in [125].

Corollary 2.3.4. *For the k-th GMRES residual for solving (2.3.29), it holds that*

$$|\mathbf{r}_k|_\star \leq \left(\sqrt{1-(C_0/C_1)^2h^2}\right)^k |\mathbf{r}_0|_\star \tag{2.3.51}$$

and

$$|\mathbf{r}_k|_\star \leq \left(\sqrt{1-(\tilde{C}_0/\tilde{C}_1)^2h^2\Delta t^2}\right)^k |\mathbf{r}_0|_\star. \tag{2.3.52}$$

Remark 2.3.3. *Bounds (2.3.51) and (2.3.52) imply the convergence of the interface GMRES iteration that is independent of either Δt or c_0, but not both. In Section 2.5 we present numerical results showing that the GMRES convergence is robust with respect to both c_0 and Δt.*

2.4 Split Methods

In this section, we consider two popular splitting methods to decouple the fully coupled poroelastic problem, namely the drained split (DS) and fixed stress (FS) methods [51, 52]. We show, using energy bounds, that these two methods are unconditionally stable in our MFE formulation. We then define, at each time step, a domain decomposition algorithm for the flow and mechanics equations separately. Domain decomposition techniques for the flow [38] and mechanics [48] components have already been studied in previous works.

2.4.1 Drained split

The DS method consists of solving the mechanics problem first, with the value of pressure from the previous time step. Afterward, the flow problem is solved using the new values of the stress tensor. The DS method for the classical Biot formulation of poroelasticity is known to require certain conditions on the parameters for stability [51]. In the setting of our mixed formulation, we show that this is not necessary and the method is unconditionally stable, see also [83]. For simplicity, we do the analysis with zero source terms.This method results in the problem: for $n = -1, 0, \ldots, N-1$, find $(\sigma_h^{n+1}, u_h^{n+1}, \gamma_h^{n+1}, z_h^{n+1}, p_h^{n+1}) \in \mathbb{X}_h \times V_h \times \mathbb{Q}_h \times Z_h \times W_h$ such that

$$\left(A\sigma_h^{n+1}, \tau\right) + \left(u_h^{n+1}, \operatorname{div} \tau\right) + \left(\gamma_h^{n+1}, \tau\right) = -\left(A\alpha p_h^n I, \tau\right), \qquad \forall \tau \in \mathbb{X}_h, \tag{2.4.1}$$

$$\left(\operatorname{div} \sigma_h^{n+1}, v\right) = 0, \qquad \forall v \in V_h, \tag{2.4.2}$$

$$\left(\sigma_h^{n+1}, \xi\right) = 0, \qquad \forall \xi \in \mathbb{Q}_h, \tag{2.4.3}$$

and

$$\left(K^{-1} z_h^{n+1}, q\right) - \left(p_h^{n+1}, \operatorname{div} q\right) = 0, \qquad \forall q \in Z_h, \tag{2.4.4}$$

$$c_0 \left(\frac{p_h^{n+1} - p_h^n}{\Delta t}, w\right) + \alpha \left(A\alpha \frac{p_h^{n+1} - p_h^n}{\Delta t} I, wI\right) + \left(\operatorname{div} z_h^{n+1}, w\right) = -\alpha \left(A \frac{\sigma_h^{n+1} - \sigma_h^n}{\Delta t}, wI\right), \qquad \forall w \in W_h, \tag{2.4.5}$$

where (2.4.1)–(2.4.4) hold for $n = -1, 0, \ldots, N-1$ with $p_h^{-1} := p_h^0$, and (2.4.5) holds for $n = 0, \ldots, N-1$. We note that solving (2.4.1)–(2.4.4) for $n = -1$ provides initial data σ_h^0, u_h^0, γ_h^0, and z_h^0.

2.4.1.1 Stability analysis for drained split

The following theorem shows that the drained split scheme is unconditionally stable.

Theorem 2.4.1. *For the solution* $(\sigma_h^{n+1}, u_h^{n+1}, \gamma_h^{n+1}, z_h^{n+1}, p_h^{n+1})_{0\le n\le N-1}$ *of the system (2.4.1)–(2.4.5), there exists a constant* C *independent of* h, *and* Δt, c_0, *and* $a_{\min}$ *such that*

$$\sum_{n=0}^{N-1} \frac{c_0}{\Delta t}\|p_h^{n+1}-p_h^n\|^2 + \max_{0\le n\le N-1}\left(\|z_h^{n+1}\|^2 + \|p_h^{n+1}\|^2 + \|A^{1/2}\sigma_h^{n+1}\|^2 + \|u_h^{n+1}\|^2 + \|\gamma_h^{n+1}\|^2\right)$$
$$\le C\left(\|p_h^0\|^2 + \|z_h^0\|^2\right).$$

Proof. We subtract two successive time steps for equations (2.4.1)–(2.4.4), obtaining, for $n = 0, \dots, N-1$,

$$\left(A(\sigma_h^{n+1}-\sigma_h^n),\tau\right) + \left(u_h^{n+1}-u_h^n, \operatorname{div}\tau\right) + \left(\gamma_h^{n+1}-\gamma_h^n,\tau\right)$$
$$= -\left(A\alpha(p_h^n - p_h^{n-1})I,\tau\right), \qquad \forall\,\tau\in\mathbb{X}_h, \tag{2.4.6}$$
$$\left(\operatorname{div}(\sigma_h^{n+1}-\sigma_h^n), v\right) = 0, \qquad \forall\, v\in V_h, \tag{2.4.7}$$
$$\left(\sigma_h^{n+1}-\sigma_h^n,\xi\right) = 0, \qquad \forall\,\xi\in\mathbb{Q}_h, \tag{2.4.8}$$
$$\left(K^{-1}(z_h^{n+1}-z_h^n), q\right) - \left(p_h^{n+1}-p_h^n, \operatorname{div} q\right) = 0, \qquad \forall\, q\in Z_h. \tag{2.4.9}$$

Taking $\tau = \sigma_h^{n+1}-\sigma_h^n$, $v = u_h^{n+1}-u_h^n$ and $\xi = \gamma_h^{n+1}-\gamma_h^n$ in (2.4.6)–(2.4.8) and summing gives

$$\left(A(\sigma_h^{n+1}-\sigma_h^n), \sigma_h^{n+1}-\sigma_h^n\right) = -\left(A\alpha(p_h^n-p_h^{n-1})I, \sigma_h^{n+1}-\sigma_h^n\right),$$

implying

$$\|A^{\frac12}(\sigma_h^{n+1}-\sigma_h^n)\| \le \alpha\|A^{\frac12}(p_h^n-p_h^{n-1})I\|. \tag{2.4.10}$$

Taking $q = z_h^{n+1}$ in (2.4.9) and $w = p_h^{n+1}-p_h^n$ in (2.4.5) and summing results in

$$c_0\left(\frac{p_h^{n+1}-p_h^n}{\Delta t}, p_h^{n+1}-p_h^n\right) + \alpha\left(A\alpha\frac{p_h^{n+1}-p_h^n}{\Delta t}I, (p_h^{n+1}-p_h^n)I\right) + \left(K^{-1}(z_h^{n+1}-z_h^n), z_h^{n+1}\right)$$
$$= \alpha\left(A\frac{\sigma_h^{n+1}-\sigma_h^n}{\Delta t}, (p_h^{n+1}-p_h^n)I\right) \le \frac{1}{2\Delta t}\|A^{\frac12}(\sigma_h^{n+1}-\sigma_h^n)\|^2 + \frac{\alpha^2}{2\Delta t}\|A^{\frac12}(p_h^{n+1}-p_h^n)I\|^2,$$

which, combined with (2.4.10), implies

$$\frac{c_0}{\Delta t}\|p_h^{n+1}-p_h^n\|^2 + \frac{\alpha^2}{2\Delta t}\|A^{\frac12}(p_h^{n+1}-p_h^n)I\|^2 + \frac12(\|K^{-\frac12}(z_h^{n+1}-z_h^n)\|^2 + \|K^{-\frac12}z_h^{n+1}\|^2$$
$$- \|K^{-\frac12}z_h^n\|^2) \le \frac{\alpha^2}{2\Delta t}\|A^{\frac12}(p_h^n-p_h^{n-1})I\|^2.$$

Summing over n from 0 to $k-1$ for any $k = 1, \dots, N$ and using that $p_h^{-1} = p_h^0$ gives

$$\sum_{n=0}^{k-1} \frac{2c_0}{\Delta t} \|p_h^{n+1} - p_h^n\|^2 + \frac{\alpha^2}{\Delta t} \|A^{\frac{1}{2}}(p_h^k - p_h^{k-1})I\|^2 + \|K^{-\frac{1}{2}} z_h^k\|^2 + \sum_{n=0}^{k-1} \|K^{-\frac{1}{2}}(z_h^{n+1} - z_h^n)\|^2$$
$$\leq \|K^{-\frac{1}{2}} z_h^0\|^2.$$

We note that the second and fourth terms are suboptimal with respect to Δt. Neglecting these terms and using (2.3.32), we obtain

$$\sum_{n=0}^{k-1} \frac{c_0}{\Delta t} \|p_h^{n+1} - p_h^n\|^2 + \|z_h^k\|^2 \leq C \|z_h^0\|^2, \quad k = 1, \dots, N. \tag{2.4.11}$$

To obtain control on p_h independent of c_0, we use the inf-sup condition (2.2.14) and (2.4.4):

$$\|p_h^{n+1}\| \leq C \sup_{0 \neq q \in Z_h} \frac{(\operatorname{div} q, p_h^{n+1})}{\|q\|_{\operatorname{div}}} = C \sup_{0 \neq q \in Z_h} \frac{(K^{-1} z_h^{n+1}, q)}{\|q\|_{\operatorname{div}}} \leq C \|z_h^{n+1}\|, \quad n = 0, \dots, N-1. \tag{2.4.12}$$

Taking $\tau = \sigma_h^{n+1}$, $v = u_h^{n+1}$, and $\xi = \gamma_h^{n+1}$ in (2.4.1)–(2.4.3) gives

$$\|A^{1/2} \sigma_h^{n+1}\| \leq C \|p_h^n\|, \quad n = 0, \dots, N-1. \tag{2.4.13}$$

For the stability of u_h and γ_h, the inf-sup condition (2.2.13) combined with (2.4.1) gives:

$$\|u_h^{n+1}\| + \|\gamma_h^{n+1}\| \leq C \sup_{0 \neq \tau \in \mathbb{X}_h} \frac{(u_h^{n+1}, \operatorname{div} \tau) + (\gamma_h^{n+1}, \tau)}{\|\tau\|_{\operatorname{div}}} = -C \sup_{0 \neq \tau \in \mathbb{X}_h} \frac{(A\sigma_h^{n+1}, \tau) + (A\alpha p_h^n I, \tau)}{\|\tau\|_{\operatorname{div}}}$$
$$\leq C \left(\|A^{\frac{1}{2}} \sigma_h^{n+1}\| + \|p_h^n\| \right), \quad n = 0, \dots, N-1. \tag{2.4.14}$$

A combination of bounds (2.4.11)–(2.4.14) completes the proof of the theorem.

□

2.4.2 Fixed stress

The FS decoupling method solves the flow problem first, with the value of σ fixed from the previous time step. After that, the mechanics problem is solved using the new values of the pressure as data [52]. We again assume in the analysis zero source terms for simplicity. The method is: for $n = -1, 0, \ldots, N-1$, find $(\sigma_h^{n+1}, u_h^{n+1}, \gamma_h^{n+1}, z_h^{n+1}, p_h^{n+1}) \in \mathbb{X}_h \times V_h \times \mathbb{Q}_h \times Z_h \times W_h$ such that

$$\left(K^{-1} z_h^{n+1}, q\right) - \left(p_h^{n+1}, \operatorname{div} q\right) = 0, \qquad \forall q \in Z_h, \tag{2.4.15}$$

$$\begin{aligned} c_0 \left(\frac{p_h^{n+1} - p_h^n}{\Delta t}, w\right) + \alpha \left(A\alpha \frac{p_h^{n+1} - p_h^n}{\Delta t} I, wI\right) + \left(\operatorname{div} z_h^{n+1}, w\right) \\ = -\alpha \left(A \frac{\sigma_h^n - \sigma_h^{n-1}}{\Delta t}, wI\right), \qquad \forall w \in W_h, \end{aligned} \tag{2.4.16}$$

and

$$\left(A\sigma_h^{n+1}, \tau\right) + \left(u_h^{n+1}, \operatorname{div} \tau\right) + \left(\gamma_h^{n+1}, \tau\right) = -\left(A\alpha p_h^{n+1} I, \tau\right), \qquad \forall \tau \in \mathbb{X}_h, \tag{2.4.17}$$

$$\left(\operatorname{div} \sigma_h^{n+1}, v\right) = 0, \qquad \forall v \in V_h, \tag{2.4.18}$$

$$\left(\sigma_h^{n+1}, \xi\right) = 0, \qquad \forall \xi \in \mathbb{Q}_h, \tag{2.4.19}$$

where the equations (2.4.15) and (2.4.17)–(2.4.19) hold for $n = -1, 0, \ldots, N-1$ and (2.4.16) holds for $n = 0, \ldots, N-1$ with $\sigma_h^{-1} := \sigma_h^0$. Solving (2.4.15) and (2.4.17)–(2.4.19) for $n = -1$ provides initial data σ_h^0, u_h^0, γ_h^0, and z_h^0.

2.4.2.1 Stability analysis for fixed stress

The following theorem shows that the fixed stress scheme is unconditionally stable.

Theorem 2.4.2. *For the solution $(\sigma_h^{n+1}, u_h^{n+1}, \gamma_h^{n+1}, z_h^{n+1}, p_h^{n+1})_{0 \le n \le N-1}$ of the system (2.4.15)–(2.4.19), there exists a constant C independent of h, and Δt, c_0, and $a_{\min}$ such that*

$$\begin{aligned} \sum_{n=0}^{N-1} \frac{c_0}{\Delta t} \|p_h^{n+1} - p_h^n\|^2 + \max_{0 \le n \le N-1} \left(\|z_h^{n+1}\|^2 + \|p_h^{n+1}\|^2 + \|A^{1/2}\sigma_h^{n+1}\|^2 + \|u_h^{n+1}\|^2 + \|\gamma_h^{n+1}\|^2\right) \\ \le C \|z_h^0\|^2. \end{aligned}$$

Proof. The proof is similar to that of the drained split scheme. Taking the difference of two successive time steps for equations (2.4.17)–(2.4.19) and (2.4.15), we obtain, for $n = 0, \dots, N-1$,

$$\begin{aligned}
&\big(A(\sigma_h^{n+1} - \sigma_h^n), \tau\big) + \big(u_h^{n+1} - u_h^n, \operatorname{div} \tau\big) + \big(\gamma_h^{n+1} - \gamma_h^n, \tau\big) \\
&\qquad + \big(A\alpha(p_h^{n+1} - p_h^n)I, \tau\big) = 0, && \forall \tau \in \mathbb{X}_h, && (2.4.20)\\
&\big(\operatorname{div}(\sigma_h^{n+1} - \sigma_h^n), v\big) = 0, && \forall v \in V_h, && (2.4.21)\\
&\big(\sigma_h^{n+1} - \sigma_h^n, \xi\big) = 0, && \forall \xi \in \mathbb{Q}_h, && (2.4.22)\\
&\big(K^{-1}(z_h^{n+1} - z_h^n), q\big) - \big(p_h^{n+1} - p_h^n, \operatorname{div} q\big) = 0, && \forall q \in Z_h, && (2.4.23)
\end{aligned}$$

Taking $\tau = \sigma_h^{n+1} - \sigma_h^n$, $v = u_h^{n+1} - u_h^n$ and $\xi = \gamma_h^{n+1} - \gamma_h^n$ in (2.4.20)–(2.4.22) and adding the equations results in

$$\|A^{\frac{1}{2}}(\sigma_h^{n+1} - \sigma_h^n)\| \le \alpha \|A^{\frac{1}{2}}(p_h^{n+1} - p_h^n)I\|. \tag{2.4.24}$$

Taking test functions $q = z_h^{n+1}$ in (2.4.23) and $w = p_h^{n+1} - p_h^n$ in (2.4.16) and adding the equations gives

$$\begin{aligned}
&c_0 \left(\frac{p_h^{n+1} - p_h^n}{\Delta t}, p_h^{n+1} - p_h^n\right) + \alpha \left(A\alpha \frac{p_h^{n+1} - p_h^n}{\Delta t} I, (p_h^{n+1} - p_h^n)I\right) + \big(K^{-1}(z_h^{n+1} - z_h^n), z_h^{n+1}\big) \\
&= \alpha \left(A \frac{\sigma_h^n - \sigma_h^{n-1}}{\Delta t}, (p_h^{n+1} - p_h^n)I\right) \le \frac{1}{2\Delta t} \|A^{\frac{1}{2}}(\sigma_h^n - \sigma_h^{n-1})\|^2 + \frac{\alpha^2}{2\Delta t} \|A^{\frac{1}{2}}(p_h^{n+1} - p_h^n)I\|^2,
\end{aligned}$$

which, combined with (2.4.24), implies, for $n = 0, \dots, N-1$,

$$\begin{aligned}
&\frac{c_0}{\Delta t} \|p_h^{n+1} - p_h^n\|^2 + \frac{\alpha^2}{2\Delta t} \|A^{\frac{1}{2}}(p_h^{n+1} - p_h^n)I\|^2 + \frac{1}{2}\Big(\|K^{-\frac{1}{2}}(z_h^{n+1} - z_h^n)\|^2 + \|K^{-\frac{1}{2}} z_h^{n+1}\|^2 \\
&- \|K^{-\frac{1}{2}} z_h^n\|^2\Big) \le \frac{\alpha^2}{2\Delta t} \|A^{\frac{1}{2}}(p_h^n - p_h^{n-1})I\|^2,
\end{aligned}$$

where for $n = 0$ we have set $p_h^{-1} := p_h^0$. Summing over n from 0 to $k-1$ for any $k = 1, \dots, N$ gives

$$\sum_{n=0}^{k-1} \frac{c_0}{\Delta t} \|p_h^{n+1} - p_h^n\|^2 + \|z_h^k\|^2 \le C \|z_h^0\|^2, \quad k = 1, \dots, N. \tag{2.4.25}$$

Next, similarly to the arguments in Theorem 2.4.1, we obtain

$$\|p_h^{n+1}\| \le C \|z_h^{n+1}\|, \quad n = 0, \dots, N-1, \tag{2.4.26}$$

$$\|A^{1/2} \sigma_h^{n+1}\| \le C \|p_h^{n+1}\|, \quad n = 0, \dots, N-1. \tag{2.4.27}$$

and

$$\|u_h^{n+1}\| + \|\gamma_h^{n+1}\| \le C \left(\|A^{\frac{1}{2}} \sigma_h^{n+1}\| + \|p_h^{n+1}\|\right), \quad n = 0, \dots, N-1. \tag{2.4.28}$$

The proof is completed by combining (2.4.25)–(2.4.28). □

2.4.3 Domain decomposition for the split methods

In this subsection, we present a non-overlapping domain decomposition method for the drained split decoupled formulation discussed in subsection 2.4.1, with non-zero source terms. The domain decomposition algorithm for the fixed stress decoupled formulation is similar; it can be obtained by modifying the order of the coupling terms accordingly. We omit the details.

Following the notation used in Section 2.3 for the monolithic domain decomposition method, the domain decomposition method for the DS formulation with non-zero source terms reads as follows: for $1 \leq i \leq m$ and $n = 0, \ldots, N-1$, find $(\sigma_{h,i}^{n+1}, u_{h,i}^{n+1}, \gamma_{h,i}^{n+1}, \lambda_h^{u,n+1}) \in \mathbb{X}_{h,i} \times V_{h,i} \times \mathbb{Q}_{h,i} \times \Lambda_h^u$ and $(z_{h,i}^{n+1}, p_{h,i}^{n+1}, \lambda_h^{p,n+1}) \in Z_{h,i} \times W_{h,i} \times \Lambda_h^p$ such that:

$$\begin{aligned}
&\left(A\sigma_{h,i}^{n+1}, \tau\right)_{\Omega_i} + \left(u_{h,i}^{n+1}, \operatorname{div}\tau\right)_{\Omega_i} + \left(\gamma_{h,i}^{n+1}, \tau\right)_{\Omega_i} \\
&\qquad = \left(A\alpha p_{h,i}^{n} I, \tau\right)_{\Omega_i} + \langle g_u^{n+1}, \tau\, n_i\rangle_{\partial\Omega_i \cap \Gamma_D^u} + \langle \lambda_h^{u,n+1}, \tau\, n_i\rangle_{\Gamma_i}, && \forall \tau \in \mathbb{X}_{h,i}, && (2.4.29)\\
&\left(\operatorname{div}\sigma_{h,i}^{n+1}, v\right)_{\Omega_i} = -\left(f^{n+1}, v\right)_{\Omega_i}, && \forall v \in V_{h,i}, && (2.4.30)\\
&\left(\sigma_{h,i}^{n+1}, \xi\right)_{\Omega_i} = 0, && \forall \xi \in \mathbb{Q}_{h,i}, && (2.4.31)\\
&\sum_{i=1}^{m} \left(\sigma_{h,i}^{n+1} n_i, \mu^u\right)_{\Gamma_i} = 0, && \forall \mu^u \in \Lambda_h^u, && (2.4.32)
\end{aligned}$$

and

$$\begin{aligned}
&\left(K^{-1} z_{h,i}^{n+1}, q\right)_{\Omega_i} - \left(p_{h,i}^{n+1}, \operatorname{div} q\right)_{\Omega_i} = -\langle g_p^{n+1}, q\cdot n_i\rangle_{\partial\Omega_i\cap\Gamma_D^p} - \langle \lambda_h^{p,n+1}, q\cdot n_i\rangle_{\Gamma_i}, && \forall q \in Z_{h,i},\\
&c_0\left(\frac{p_{h,i}^{n+1} - p_{h,i}^{n}}{\Delta t}, w\right)_{\Omega_i} + \alpha\left(A\alpha\frac{p_{h,i}^{n+1} - p_{h,i}^{n}}{\Delta t} I, wI\right)_{\Omega_i} + \left(\operatorname{div} z_{h,i}^{n+1}, w\right)_{\Omega_i}\\
&\qquad = -\alpha\left(\frac{A\left(\sigma_{h,i}^{n+1} - \sigma_{h,i}^{n}\right)}{\Delta t}, wI\right)_{\Omega_i} + \left(g^{n+1}, w\right)_{\Omega_i}, && \forall w \in W_{h,i},\\
&\sum_{i=1}^{m}\left(z_{h,i}^{n+1}\cdot n_i, \mu^p\right)_{\Gamma_i} = 0, && \forall \mu^p \in \Lambda_h^p.
\end{aligned}$$

The above split domain decomposition formulation consists of separate domain decomposition methods for mechanics and flow at each time step. Such methods have been studied in detail for the flow [38] and mechanics [48] components. It is shown that in both cases the global problem can be reduced to an interface problem with a symmetric and positive definite operator with condition number $O(h^{-1})$. Therefore, we employ the conjugate gradient (CG) method for the solution of the interface problem in each case.

2.5 Numerical Results

In this section we report the results of several numerical tests designed to verify and compare the convergence, stability, and efficiency of the three domain decomposition methods developed in the previous sections. The numerical schemes are implemented using deal.II finite element package [91, 92].

In all examples the computational domain is the unit square $(0,1)^2$ and the mixed finite element spaces are $\mathbb{X}_h \times V_h \times \mathbb{Q}_h = \mathcal{BDM}_1^2 \times Q_0^2 \times Q_0$ [11] for elasticity and $Z_h \times W_h = \mathcal{BDM}_1 \times Q_0$ [22] for Darcy on quadrilateral meshes. Here Q_k denotes polynomials of degree k in each variable. For solving the interface problem in the monolithic scheme we use non-restarted unpreconditioned GMRES and in the sequential decoupled methods we use unpreconditioned CG for the flow and mechanics parts separately. We use a tolerance on the relative residual $\frac{r_k}{r_0}$ as the stopping criteria for both iterative solvers. For Examples 1 and 2, the tolerance is taken to be 10^{-12}. For Example 3, the tolerance is taken to be 10^{-6} due to relatively smaller initial residual r_0. For the monolithic method, Theorem 2.3.3 implies that that the spectral ratio $\frac{\lambda_{\max}}{\lambda_{\min}} = \mathcal{O}(h^{-1})$, where $\lambda_{\min}$ and $\lambda_{\max}$ are the smallest and largest real eigenvalues of the interface operator, respectively. Depending on the deviation of the operator from a normal matrix [115, 116], the growth rate for the number of iterations required for GMRES to converge could be bounded. In particular, if the interface operator is normal, then the expected growth rate of the number of GMRES iterations is $\mathcal{O}\left(\sqrt{\frac{\lambda_{max}}{\lambda_{min}}}\right)$ [116], which in our case is $\mathcal{O}(h^{-0.5})$. On the other hand, the interface operators in the decoupled mechanics and flow systems in the DS and FS schemes are symmetric and positive definite [38, 48]. A well known result [116] is that the number of CG iterations required for convergence is $\mathcal{O}(\sqrt{\kappa})$, where κ is the condition number for the interface operator. Furthermore, it is shown in [48, 78] that the condition numbers κ_{mech} and κ_{flow} for the interface operators corresponding to the mechanics and flow parts respectively are $\mathcal{O}(h^{-1})$ as well and hence the expected growth rate for the number of CG iterations is also $\mathcal{O}(h^{-0.5})$.

2.5.1 Example 1: convergence and stability

In this example we test the convergence and stability of the three domain decomposition schemes. We consider the analytical solution

$$p = \exp(t)(\sin(\pi x)\cos(\pi y) + 10), \quad u = \exp(t)\begin{pmatrix} x^3y^4 + x^2 + \sin((1-x)(1-y))\cos(1-y) \\ (1-x)^4(1-y)^3 + (1-y)^2 + \cos(xy)\sin(x) \end{pmatrix}.$$

The physical and numerical parameters are given in Table 1. Using this information, we derive the right hand side and boundary and initial conditions for the system (1.3.1)–(1.3.9). The

Table 1: Example 1, physical and numerical parameters.

Parameter	Value
Permeability tensor (K)	I
Lame coefficient (μ)	100.0
Lame coefficient (λ)	100.0
Mass storativity (c_0)	$1.0, 10^{-3}$
Biot-Willis constant (α)	1.0
Time step (Δt)	$10^{-3}, 10^{-2}, 10^{-1}$
Number of time steps	100

global mesh is divided into 2×2 square subdomains. We run a sequence of refinements from $h = 1/4$ to $h = 1/64$. The initial grids in the bottom left and top right subdomains are perturbed randomly, resulting in general quadrilateral elements. The computed solution for the monolithic scheme with $h = 1/64$ and $\Delta t = 10^{-3}$ on the final time step is given in Figure 1.

To study and compare the convergence and stability of the three methods, we run tests with time steps $\Delta t = 10^{-3}, 10^{-2}$ and 10^{-1}. The results with $c_0 = 1$ are presented in Tables 2–4. We report the average number of iterations over 100 time steps. The numerical errors are relative to the corresponding norms of the exact solution. We use standard Bochner space notation to denote the space-time norms. Convergence results for the case with $c_0 = 0.001$ and $\Delta t = 0.01$ are given in Table 5.

The main observation is that all three methods exhibit growth in the number of interface iterations at the rate of $\mathcal{O}(h^{-0.5})$. This is consistent with the theoretical bounds on the spectrum of the interface operator, cf. the discussion at the beginning of Section 2.5. This behavior is robust with respect to both Δt and c_0. We further note that in both split schemes, the Darcy interface solver requires fewer number of iterations than the elasticity solver. We attribute this to the fact that the Darcy formulation involves a contribution to the diagonal from the time derivative term, resulting in a smaller condition number of the interface operator.

Another important conclusion from the tables is that two split schemes are stable uniformly in Δt and c_0, in accordance with Theorem 2.4.1 and Theorem 2.4.2.

In terms of accuracy, all three methods yield $\mathcal{O}(h)$ convergence for all variables in their natural norms, which is optimal convergence for the approximation of the Biot system with the chosen finite element spaces, cf. [6, 54]. In some cases, especially for larger Δt, we observe reduction in the convergence rate for certain variables due to the effect of the time discretization and/or splitting errors, most notably for the Darcy velocity in the fixed stress scheme. The accuracy of the three methods is comparable for smaller Δt.

In terms of efficiency, the split schemes have a clear advantage, due to the smaller total number of interface iterations, the more efficient CG interface solver

compared to GMRES for the monolithic scheme, as well as the less costly subdomain problems - individual physics solves versus the coupled Biot solves in the monolithic scheme.

2.5.2 Example 2: dependence on number of subdomains

The objective of this example is to study how the number of GMRES and CG iterations required for the different schemes depend on the number (and diameter) of subdomains used in the domain decomposition. For this example, we use the same test case as in Example 1. We solve the system using 4 (2×2), 16 (4×4), and 64 (8×8) square subdomains of identical size. The physical parameters are as in Example 1, with $c_0 = 1$, $\Delta t = 10^{-3}$, and $T = 100 \times \Delta t$. The average number and growth rate of iterations in the three methods are reported in Tables 6–8, where A denotes the subdomain diameter. We note that the number of iterations for the drained split and fixed stress schemes are identical, so we give one table for both methods. For a fixed A, the growth rate with respect to h is averaged over all mesh refinements. For a fixed mesh

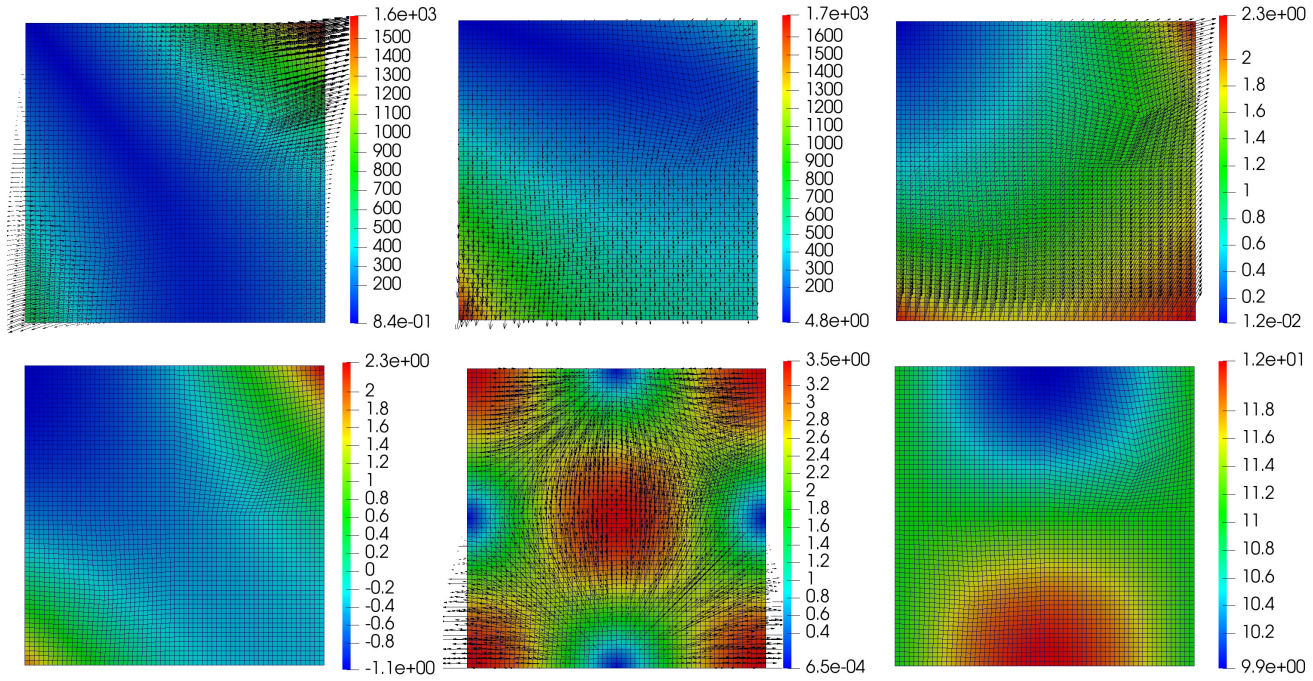

Figure 1: Example 1, computed solution at the final time step using the monolithic domain decomposition method with $h = 1/64$ and $\Delta t = 10^{-3}$, top: stress x (left), stress y (middle), displacement (right), bottom: rotation (left), velocity (middle), pressure (right).

size h, the growth rate with respect to A is averaged over the different domain decompositions. For all three methods, we observe that for a fixed number of subdomains, the growth rate in the number of iterations with respect to mesh refinement is approximately $\mathcal{O}(h^{-0.5})$, being slightly better for the Darcy solver in the split schemes. As this is the same as the growth rate in Example 1, the conclusion from Example 1 that the growth rate is consistent with the theory extends to domain decompositions with varying number of subdomains, see also the discussion at the beginning of Section 2.5. We further observe that for a fixed mesh size, the growth rate in number of iterations with respect to subdomain diameter A is approximately $\mathcal{O}(A^{-0.5})$, again being somewhat better for the Darcy solves. This is consistent with theoretical results bounding the spectral ratio of the unpreconditioned interface operator as $\mathcal{O}\left((hA)^{-1}\right)$ [77]. The dependence on A can be eliminated with the use of a coarse solve preconditioner [77, 78].

2.5.3 Example 3: heterogeneous benchmark

This example illustrates the performance of the methods for highly heterogeneous media. We use porosity and permeability fields from the Society of Petroleum Engineers 10th Comparative Solution Project (SPE10)[1]. The computational domain is $\Omega = (0,1)^2$, which is partitioned into a 128×128 square grid. We decompose the domain into 4×4 square subdomains. From the porosity field data, the Young's modulus is obtained using the relation $E = 10^2\left(1-\frac{\phi}{c}\right)^{2.1}$, where $c = 0.5$, refers to the porosity at which the Young's modulus vanishes, see [53] for details. The porosity, Young's modulus and permeability fields are given in Figure 2. The parameters and boundary conditions are given in Table 9. The source terms are taken to be zero. These conditions describe flow from left to right, driven by a pressure gradient. Since in this example analytical solution is not available, we need to prescribe suitable initial data. The initial condition for the pressure is taken to be $p_0 = 1 - x$, which is compatible with the prescribed boundary conditions. We then follow the procedure described in Remark 2.3.1 to obtain discrete initial data. In particular, we set p_h^0 to be the L^2-projection of p_0 onto W_h and solve a mixed elasticity domain decomposition problem at $t = 0$ to obtain σ_h^0. We note that this solve also gives u_h^0, γ_h^0, and $\lambda_h^{u,0}$. In the case of the monolithic scheme where the time-differentiated elasticity equation (2.3.8) is solved, the computed initial data is used to recover u_h^n, γ_h^n, and $\lambda_h^{u,n}$ using (2.3.16). The computed solution using the monolithic domain decomposition scheme is given in Figure 3. The solutions from the two split methods look similar.

In Table 10, we compare the average number of interface iterations per time step in the three methods. All three methods converge for this highly heterogeneous problem with realistic physical parameters. While the three methods provide similar solutions, the split methods are more efficient than the monolithic method, as they require smaller number of interface iterations. We further note that in the split methods the Darcy solve is more expensive, which is likely due to the fact that the permeability varies over seven orders of magnitude, affecting the condition number of the interface operator.

[1] https://www.spe.org/web/csp/datasets/set02.htm

2.6 Chapter Conclusions

We presented three non-overlapping domain decomposition methods for the Biot system of poroelasticity in a five-field fully mixed formulation. The monolithic method involves solving an interface problem for a composite displacement-pressure Lagrange multiplier, which requires coupled Biot subdomain solves at each iteration. The two split methods are based on the drained split and fixed stress splittings. They involve two separate elasticity and Darcy interface iterations requiring single-physics subdomain solves. We analyze the spectrum of the monolithic interface operator and show unconditional stability for the split methods. A series of numerical experiments illustrate the efficiency, accuracy, and robustness of the three methods. Our main conclusion is that the split methods provide accuracy comparable to the monolithic method, while being more computationally efficient in terms of smaller number of interface iterations and simpler subdomain solves.

Table 2: Example 1, convergence for $\Delta t = 10^{-3}$ and $c_0 = 1$, monolithic scheme (top), drained split (middle), fixed stress (bottom).

h	#GMRES		$\|z - z_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|p - p_h\|_{L^\infty(L^2)}$		$\|\sigma - \sigma_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|u - u_h\|_{L^\infty(L^2)}$	
1/4	24	rate	2.13e+00	rate	7.05e-02	rate	6.95e-01	rate	6.88e-01	rate
1/8	33	-0.46	1.13e+00	0.92	3.56e-02	0.98	3.57e-01	0.96	3.48e-01	0.98
1/16	44	-0.42	4.84e-01	1.22	1.79e-02	1.00	1.79e-01	0.99	1.75e-01	1.00
1/32	62	-0.49	2.01e-01	1.27	8.94e-03	1.00	8.99e-02	1.00	8.74e-02	1.00
1/64	87	-0.49	9.15e-02	1.14	4.47e-03	1.00	4.50e-02	1.00	4.37e-02	1.00

h	#CGElast		#CGDarcy		$\|z - z_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|p - p_h\|_{L^\infty(L^2)}$		$\|\sigma - \sigma_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|u - u_h\|_{L^\infty(L^2)}$	
1/4	19	rate	10	rate	2.00e+00	rate	7.07e-02	rate	7.01e-01	rate	6.88e-01	rate
1/8	23	-0.28	10	0.00	1.11e+00	0.85	3.57e-02	0.99	3.59e-01	0.96	3.48e-01	0.98
1/16	34	-0.56	11	-0.14	4.89e-01	1.18	1.79e-02	1.00	1.81e-01	0.99	1.75e-01	1.00
1/32	47	-0.47	15	-0.45	2.06e-01	1.25	8.94e-03	1.00	9.06e-02	1.00	8.74e-02	1.00
1/64	65	-0.47	20	-0.42	9.29e-02	1.15	4.47e-03	1.00	4.53e-02	1.00	4.37e-02	1.00

h	#CGElast		#CGDarcy		$\|z - z_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|p - p_h\|_{L^\infty(L^2)}$		$\|\sigma - \sigma_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|u - u_h\|_{L^\infty(L^2)}$	
1/4	19	rate	10	rate	1.93e+00	rate	7.06e-02	rate	7.01e-01	rate	6.88e-01	rate
1/8	23	-0.28	10	0.00	1.05e+00	0.88	3.56e-02	0.99	3.59e-01	0.96	3.48e-01	0.98
1/16	34	-0.56	11	-0.14	4.46e-01	1.23	1.79e-02	1.00	1.81e-01	0.99	1.75e-01	1.00
1/32	47	-0.47	15	-0.45	2.63e-01	0.76	8.95e-03	1.00	9.06e-02	1.00	8.74e-02	1.00
1/64	65	-0.47	20	-0.42	2.17e-01	0.28	4.49e-03	0.99	4.53e-02	1.00	4.37e-02	1.00

Table 3: Example 1, convergence for $\Delta t = 10^{-2}$ and $c_0 = 1$, monolithic scheme (top), drained split (middle), fixed stress (bottom).

h	#GMRES		$\|z - z_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|p - p_h\|_{L^\infty(L^2)}$		$\|\sigma - \sigma_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|u - u_h\|_{L^\infty(L^2)}$	
1/4	18	rate	1.58e+00	rate	6.98e-02	rate	6.97e-01	rate	6.88e-01	rate
1/8	23	-0.35	7.47e-01	1.08	3.55e-02	0.97	3.58e-01	0.96	3.48e-01	0.98
1/16	32	-0.48	3.58e-01	1.06	1.79e-02	0.99	1.80e-01	0.99	1.75e-01	0.99
1/32	44	-0.46	1.77e-01	1.02	8.97e-03	0.99	9.02e-02	1.00	8.88e-02	0.98
1/64	63	-0.52	8.98e-02	0.98	4.54e-03	0.98	4.53e-02	1.00	4.66e-02	0.93

h	#CGElast		#CGDarcy		$\|z - z_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|p - p_h\|_{L^\infty(L^2)}$		$\|\sigma - \sigma_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|u - u_h\|_{L^\infty(L^2)}$	
1/4	19	rate	10	rate	1.57e+00	rate	6.98e-02	rate	7.01e-01	rate	6.88e-01	rate
1/8	23	-0.28	12	-0.26	7.46e-01	1.07	3.55e-02	0.97	3.59e-01	0.96	3.48e-01	0.98
1/16	34	-0.56	16	-0.42	3.58e-01	1.06	1.79e-02	0.99	1.81e-01	0.99	1.75e-01	1.00
1/32	47	-0.47	23	-0.52	1.77e-01	1.02	8.97e-03	0.99	9.06e-02	1.00	8.74e-02	1.00
1/64	65	-0.47	32	-0.48	8.96e-02	0.98	4.53e-03	0.98	4.53e-02	1.00	4.37e-02	1.00

h	#CGElast		#CGDarcy		$\|z - z_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|p - p_h\|_{L^\infty(L^2)}$		$\|\sigma - \sigma_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|u - u_h\|_{L^\infty(L^2)}$	
1/4	19	rate	10	rate	1.48e+00	rate	6.97e-02	rate	7.01e-01	rate	6.88e-01	rate
1/8	23	-0.28	12	-0.26	7.64e-01	0.96	3.56e-02	0.97	3.59e-01	0.96	3.48e-01	0.98
1/16	34	-0.56	16	-0.42	4.88e-01	0.65	1.81e-02	0.98	1.81e-01	0.99	1.75e-01	1.00
1/32	47	-0.47	23	-0.52	3.80e-01	0.36	9.37e-03	0.95	9.06e-02	1.00	8.74e-02	1.00
1/64	65	-0.47	32	-0.48	3.44e-01	0.14	5.26e-03	0.83	4.53e-02	1.00	4.37e-02	1.00

Table 4: Example 1, convergence for $\Delta t = 10^{-1}$ and $c_0 = 1$, monolithic scheme (top), drained split (middle), fixed stress (bottom).

h	#GMRES		$\|z - z_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|p - p_h\|_{L^\infty(L^2)}$		$\|\sigma - \sigma_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|u - u_h\|_{L^\infty(L^2)}$	
1/4	40	rate	1.38e+00	rate	6.99e-02	rate	7.04e-01	rate	7.17e-01	rate
1/8	59	-0.56	7.20e-01	0.94	3.63e-02	0.94	3.65e-01	0.95	4.26e-01	0.75
1/16	88	-0.58	3.97e-01	0.86	1.94e-02	0.90	1.92e-01	0.93	3.09e-01	0.46
1/32	128	-0.54	2.57e-01	0.63	1.17e-02	0.72	1.09e-01	0.81	2.72e-01	0.19
1/64	180	-0.49	2.08e-01	0.31	8.84e-03	0.41	7.40e-02	0.56	2.62e-01	0.06

h	#CGElast		#CGDarcy		$\|z - z_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|p - p_h\|_{L^\infty(L^2)}$		$\|\sigma - \sigma_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|u - u_h\|_{L^\infty(L^2)}$	
1/4	19	rate	11	rate	1.38e+00	rate	6.99e-02	rate	7.01e-01	rate	6.88e-01	rate
1/8	23	-0.28	14	-0.35	7.17e-01	0.94	3.62e-02	0.95	3.59e-01	0.96	3.48e-01	0.98
1/16	34	-0.56	20	-0.51	3.92e-01	0.87	1.92e-02	0.91	1.81e-01	0.99	1.75e-01	1.00
1/32	47	-0.47	28	-0.49	2.50e-01	0.65	1.15e-02	0.75	9.07e-02	1.00	8.74e-02	1.00
1/64	65	-0.47	38	-0.44	1.99e-01	0.33	8.48e-03	0.44	4.56e-02	0.99	4.37e-02	1.00

h	#CGElast		#CGDarcy		$\|z - z_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|p - p_h\|_{L^\infty(L^2)}$		$\|\sigma - \sigma_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|u - u_h\|_{L^\infty(L^2)}$	
1/4	19	rate	11	rate	1.42e+00	rate	7.00e-02	rate	7.00e-01	rate	6.88e-01	rate
1/8	23	-0.28	14	-0.35	8.38e-01	0.76	3.63e-02	0.95	3.59e-01	0.96	3.48e-01	0.98
1/16	34	-0.56	20	-0.51	5.83e-01	0.52	1.93e-02	0.91	1.81e-01	0.99	1.75e-01	1.00
1/32	47	-0.47	28	-0.49	4.87e-01	0.26	1.15e-02	0.74	9.06e-02	1.00	8.74e-02	1.00
1/64	65	-0.47	38	-0.44	4.56e-01	0.09	8.53e-03	0.44	4.53e-02	1.00	4.37e-02	1.00

Table 5: Example 1, convergence for $\Delta t = 10^{-2}$ and $c_0 = 10^{-3}$, monolithic scheme (top), drained split (middle), fixed stress (bottom).

h	#GMRES		$\|z - z_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|p - p_h\|_{L^\infty(L^2)}$		$\|\sigma - \sigma_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|u - u_h\|_{L^\infty(L^2)}$	
$h/4$	21	rate	1.86e+00	rate	7.12e-02	rate	6.97e-01	rate	6.88e-01	rate
$h/8$	28	-0.42	7.87e-01	1.24	3.57e-02	1.00	3.58e-01	0.96	3.48e-01	0.98
$h/16$	38	-0.44	3.63e-01	1.12	1.79e-02	1.00	1.80e-01	0.99	1.75e-01	0.99
$h/32$	53	-0.48	1.77e-01	1.04	8.94e-03	1.00	9.02e-02	1.00	8.88e-02	0.98
$h/64$	73	-0.46	8.78e-02	1.01	4.47e-03	1.00	4.53e-02	1.00	4.66e-02	0.93

h	#CGElast		#CGDarcy		$\|z - z_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|p - p_h\|_{L^\infty(L^2)}$		$\|\sigma - \sigma_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|u - u_h\|_{L^\infty(L^2)}$	
1/4	19	rate	11	rate	1.90e+00	rate	7.16e-02	rate	7.01e-01	rate	6.88e-01	rate
1/8	23	-0.28	15	-0.45	7.91e-01	1.26	3.58e-02	1.00	3.59e-01	0.96	3.48e-01	0.98
1/16	34	-0.56	20	-0.42	3.64e-01	1.12	1.79e-02	1.00	1.81e-01	0.99	1.75e-01	1.00
1/32	47	-0.47	28	-0.49	1.78e-01	1.03	8.98e-03	1.00	9.06e-02	1.00	8.74e-02	1.00
1/64	65	-0.47	41	-0.55	9.01e-02	0.98	4.55e-03	0.98	4.53e-02	1.00	4.37e-02	1.00

h	#CGElast		#CGDarcy		$\|z - z_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|p - p_h\|_{L^\infty(L^2)}$		$\|\sigma - \sigma_h\|_{L^\infty(H_{\mathrm{div}})}$		$\|u - u_h\|_{L^\infty(L^2)}$	
1/4	19	rate	11	rate	1.88e+00	rate	7.16e-02	rate	7.01e-01	rate	6.88e-01	rate
1/8	23	-0.28	15	-0.45	8.84e-01	1.09	3.72e-02	0.94	3.59e-01	0.96	3.48e-01	0.98
1/16	34	-0.56	20	-0.42	6.43e-01	0.46	2.08e-02	0.84	1.81e-01	0.99	1.75e-01	1.00
1/32	47	-0.47	28	-0.49	5.60e-01	0.20	1.36e-02	0.61	9.06e-02	1.00	8.74e-02	1.00
1/64	65	-0.47	41	-0.55	5.36e-01	0.06	1.10e-02	0.31	4.53e-02	1.00	4.37e-02	1.00

Table 6: Example 2, number of GMRES iterations in the monolithic scheme.

h	2×2	4×4	8×8	Rate
1/8	33	53	76	$\mathcal{O}(A^{-0.60})$
1/16	45	68	97	$\mathcal{O}(A^{-0.55})$
1/32	63	93	126	$\mathcal{O}(A^{-0.50})$
1/64	88	125	164	$\mathcal{O}(A^{-0.45})$
Rate	$\mathcal{O}(h^{-0.47})$	$\mathcal{O}(h^{-0.41})$	$\mathcal{O}(h^{-0.36})$	

Table 7: Example 2, number of CG elasticity iterations in the drained split and fixed stress schemes.

h	2×2	4×4	8×8	Rate
1/8	23	40	60	$\mathcal{O}(A^{-0.69})$
1/16	34	51	73	$\mathcal{O}(A^{-0.55})$
1/32	47	68	95	$\mathcal{O}(A^{-0.51})$
1/64	65	95	124	$\mathcal{O}(A^{-0.46})$
Rate	$\mathcal{O}(h^{-0.50})$	$\mathcal{O}(h^{-0.42})$	$\mathcal{O}(h^{-0.35})$	

Table 8: Example 2, number of CG Darcy iterations in the drained split and fixed stress schemes

h	2×2	4×4	8×8	Rate
1/8	10	11	14	$\mathcal{O}(A^{-0.24})$
1/16	11	12	14	$\mathcal{O}(A^{-0.17})$
1/32	15	16	18	$\mathcal{O}(A^{-0.13})$
1/64	20	23	24	$\mathcal{O}(A^{-0.13})$
Rate	$\mathcal{O}(h^{-0.34})$	$\mathcal{O}(h^{-0.36})$	$\mathcal{O}(h^{-0.25})$	

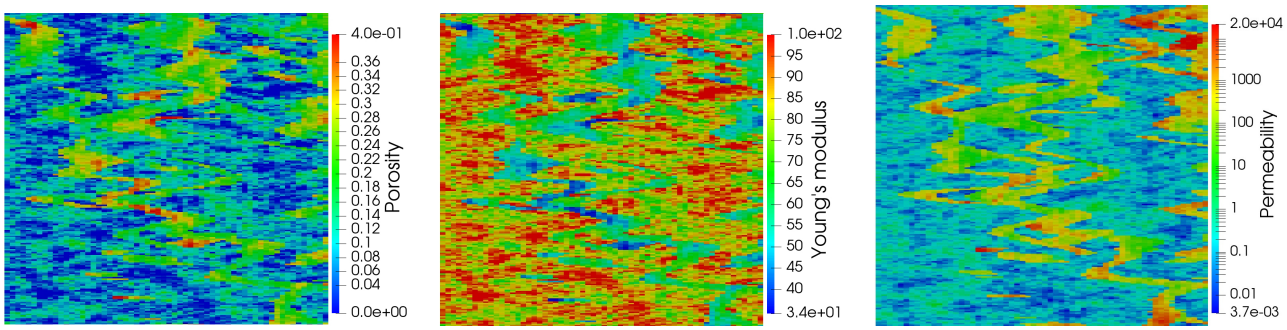

Figure 2: Example 3, porosity, Young's modulus, permeability.

Table 9: Example 3, parameters (left) and boundary conditions (right) .

Parameter	Value
Mass storativity (c_0)	1.0
Biot-Willis constant (α)	1.0
Time step (Δt)	10^{-2}
Total time (T)	1.0

Boundary	σ	u	p	z
Left	$\sigma n = -\alpha p n$	-	1	-
Bottom	$\sigma n = 0$	-	-	$z \cdot n = 0$
Right	$-$	0	0	-
Top	$\sigma n = 0$	-	-	$z \cdot n = 0$

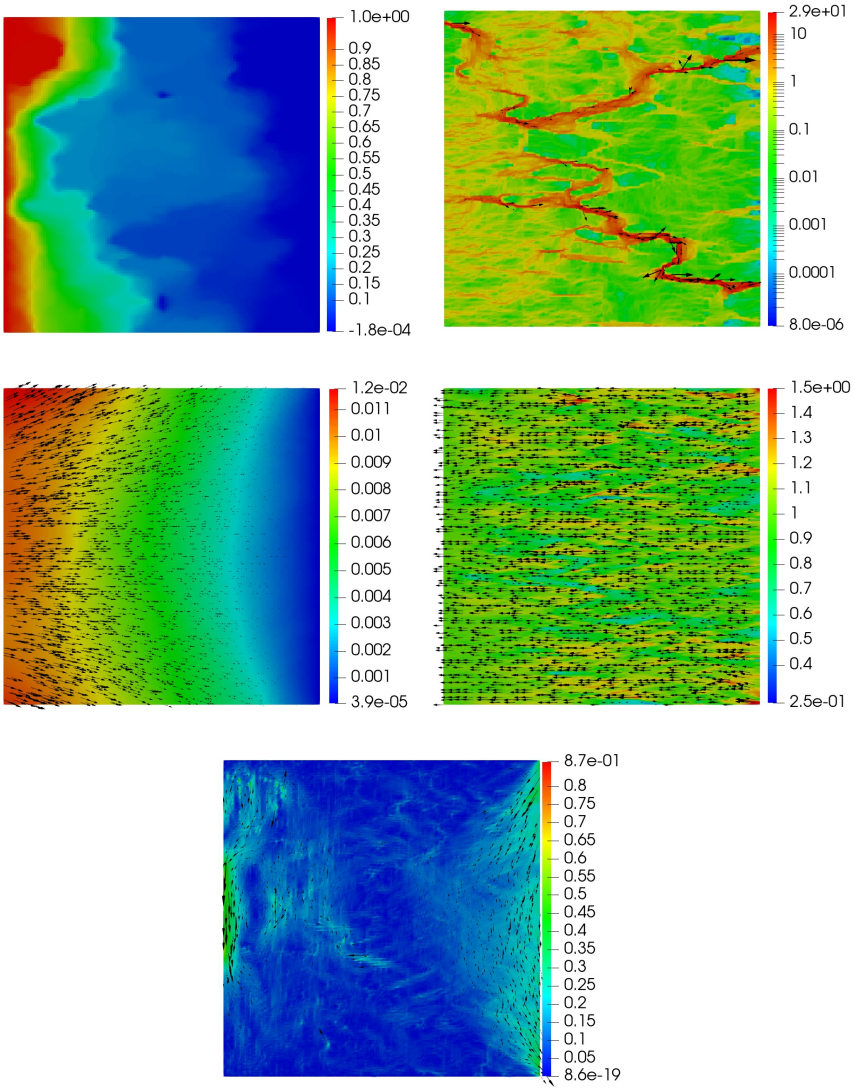

Figure 3: Example 3, computed solution at the final time using the monolithic domain decomposition scheme, top: pressure (left), velocity (right), middle: displacement (left), stress x (right), bottom: stress y.

Table 10: Example 3, comparison of the number of interface iterations in the three methods.

	Monolithic	Drained Split		Fixed Stress	
h	#GMRES	#CGElast	#CGDarcy	#CGElast	#CGDarcy
1/128	565	297	464	297	464

3.0 A Multiscale Mortar Domain Decomposition For Biot System Of Poroelasticity Using Non-matching Subdomain Grids

3.1 Introduction

In this chapter we develop and study a multiscale mortar mixed finite element (MMMFE) method for the Biot system of poroelasticity [17]. This technique is the generalization of the non-overlapping domain decomposition technique discussed in the previous chapter, where non-matching subdomain grids can be used instead of matching grids at the interface.

The MFE domain decomposition methods discussed in the previous chapter required the subdomain grids to match at the interface, which may not be the ideal setting for problems where it is advantageous to use a computational domain consisting of multiple blocks of multiscale subdomain grids. In this chapter, we study the adaptation of these methods to enable the use of non-matching multiblock grids. This work is motivated by similar studies for the second order elliptic problems in [8, 90] and for a linear system of elasticity in [48]. Following the ideas from the previous chapter, we use a physically heterogeneous Lagrange multiplier vector consisting of displacement and pressure variables to impose weakly the continuity of the normal components of stress and velocity, respectively. At each time step, we solve an interface problem for this Lagrange multiplier vector. In contrast to the previous chapter, we choose the Lagrange multiplier vector from a space of mortar finite elements, see e.g. [8, 32, 36, 43, 48–50, 63]. This allows for the interaction between multiscale subdomain grids at the interface through projections onto the mortar finite element space. This allows for the mortar space to be on a coarser scale, H (see [33, 64, 90]), compared to a finer subdomain grid size, h. The multiscale capability adds an extra layer of flexibility over the monolithic DD method discussed in the previous chapter. We study the well-posedness and stability of the method under the appropriate condition on the richness of the mortar FE space. We also show a combined a priori error estimate for stress, displacement, rotation, pressure, and Darcy velocity, as well as how well the mortar function approximates the normal components of stress and velocity. We further propose the construction and use of a multiscale stress-flux basis which makes the number of subdomain solves related to interface problem independent of the number of iterations required for the interface problem

and the number of time steps used. The reuse of the multiscale basis could gain a significant performance advantage in the case of time-dependent coupled problems. Finally, we report the results of several numerical tests designed to verify and compare the well-posedness, stability, and convergence of the multiscale domain decomposition method we have developed. We compare the computational efficiency in different cases using matching and non-matching grids on the subdomain interfaces and also, discuss the advantages of using a multiscale basis.

The rest of the chapter is organized as follows. Section 3.2 introduces the mathematical model, its MFE formulation and the domain decomposition formulation. Analysis of well-posedness, stability, and error bounds for the DD formulation is discussed in Section 3.3. In Section 3.4, we discuss the implementation details of the method along with the construction of the multiscale stress-flux basis. Numerical results are reported in Section 3.5.

3.2 Formulation of the Method

In this section, we develop the framework for the multiscale mortar mixed finite element (MMMFE) domain decomposition method based on the MFE formulation introduced in Section 2.2 of Chapter 2. Projection operators critical in the analysis of the method and various bounds associated with them are introduced. Finally, we introduce the weakly continuous spaces of stress, $\mathbb{X}_{h,0}$, and velocity, $Z_{h,0}$, and reformulate the MMMFE method in terms of these spaces. Note that some of the notations and formulations that will be introduced in this section have already been covered in the previous chapters, nevertheless, we present them here for the sake of completeness and self-containment of the chapter.

3.2.1 Multiscale mortar domain decomposition method

Let $\Omega = \cup_{i=1}^{N} \Omega_i$ be a union of non-overlapping shape regular polygonal subdomains, where each subdomain is a union of elements of finite element partition $\mathcal{T}_h$. Let $\Gamma_{i,j} = \partial\Omega_i \cap \partial\Omega_j$, $\Gamma = \cup_{i,j=1}^{N} \Gamma_{i,j}$, and $\Gamma_i = \partial\Omega_i \cap \Gamma = \partial\Omega_i \setminus \partial\Omega$ denote the interior subdomain interfaces. The domain discretization technique we develop in this section is the generalization of the monolithic non-overlapping domain decomposition technique developed in the previous chapter, where the sub-

domains are allowed to have multiscale non-matching grids at their interfaces. Let h_i be the diameter of the maximal element in the mesh on Ω_i and define $h = \max_i h_i$. For $1 \leq i \leq N$, let $\mathbb{X}_{h,i} \times V_{h,i} \times \mathbb{Q}_{h,i} \times Z_{h,i} \times W_{h,i}$ be a family of stable mixed finite elements defined on the subdomain Ω_i. These spaces could be chosen from any of the stable family of spaces discussed in the previous chapter. Let the finite element spaces $\mathbb{X}_{h,i}$, $V_{h,i}$, $\mathbb{Q}_{h,i}$, $Z_{h,i}$, and $W_{h,i}$ contain polynomials of degree less than or equal to $k \geq 1$, $l \geq 0$, $j \geq 0$, $r \geq 0$, and $s \geq 0$, respectively. We define the global FE spaces, defined on Ω, as follows:

$$\mathbb{X}_h = \bigoplus_{1\leq i\leq N} \mathbb{X}_{h,i}, \quad V_h = \bigoplus_{1\leq i\leq N} V_{h,i}, \quad \mathbb{Q}_h = \bigoplus_{1\leq i\leq N} \mathbb{Q}_{h,i}, \quad Z_h = \bigoplus_{1\leq i\leq N} Z_{h,i}, \quad W_h = \bigoplus_{1\leq i\leq N} W_{h,i},$$

with norms $\| \quad \|_{\mathbb{X}_h} := \left(\sum_{i=1}^N \| \quad \|^2_{\mathbb{X}_i}\right)^{\frac{1}{2}}$, $\| \quad \|_{V_h} := \left(\sum_{i=1}^N \| \quad \|^2_{V_i}\right)^{\frac{1}{2}}$, $\| \quad \|_{\mathbb{Q}_h} := \left(\sum_{i=1}^N \| \quad \|^2_{\mathbb{Q}_i}\right)^{\frac{1}{2}}$, $\| \quad \|_{Z_h} := \left(\sum_{i=1}^N \| \quad \|^2_{Z_i}\right)^{\frac{1}{2}}$, and $\| \quad \|_{W_h} := \left(\sum_{i=1}^N \| \quad \|^2_{W_i}\right)^{\frac{1}{2}}$, respectively, where $\mathbb{X}_i = \mathbb{X}|_{\Omega_i}$ with similar definitions for other spaces.

Note that the definitions of the global spaces $\mathbb{X}_h$ and Z_h do not impose continuity of normal components of the stress tensor or velocity vector across the sub-domain interfaces, though these normal components are continuous across element interfaces within a subdomain. This discontinuity is addressed using Lagrange multipliers defined on suitable mortar spaces on the interface Γ. We use relatively coarser mortar finite elements satisfying certain coarseness conditions (which will be discussed in the later sections) to approximate the traces of the displacement vector and the pressure at the interfaces. Let $\mathcal{T}_{H,i,j}$ be a shape regular quasi-uniform finite element partition of $\Gamma_{i,j}$ constructed using a simplicial or quadrilateral mesh in $d-1$ dimensions with maximal element diameter H. Define the global mortar fine element spaces on the union of sub-domain interfaces, Γ, to be,

$$\Lambda_H = \bigoplus_{1\leq i<j\leq N} \begin{pmatrix} \Lambda^u_{H,i,j} \\ \Lambda^p_{H,i,j} \end{pmatrix}, \; \Lambda^u_H = \bigoplus_{1\leq i<j\leq N} \Lambda^u_{H,i,j}, \text{ and } \Lambda^p_H = \bigoplus_{1\leq i<j\leq N} \Lambda^p_{H,i,j},$$

where $\Lambda^u_{H,i,j} \subset \left(L^2(\Gamma_{i,j})\right)^d$ and $\Lambda^p_{H,i,j} \subset L^2(\Gamma_{i,j})$ are mortar finite element spaces on $\Gamma_{i,j}$ representing the displacement and pressure Lagrange multipliers, respectively. We assume that these mortar spaces contain either continuous or discontinuous polynomials of degree up to $m \geq 0$. Conditions on the degree and richness of the mortar spaces in order to get a well-posed and stable method will be discussed in the later sections.

The multiscale mortar domain decomposition formulation for the mixed Biot problem in a semi-discrete form reads as follows: for $1 \leq i \leq N$, find $(\sigma_{h,i}, u_{h,i}, \gamma_{h,i}, z_{h,i}, p_{h,i}, \lambda_H) : [0,T] \to \mathbb{X}_{h,i} \times V_{h,i} \times \mathbb{Q}_{h,i} \times Z_{h,i} \times W_{h,i} \times \Lambda_H$ such that $p_{h,i}(0) = p_{h,0}|_{\Omega_i}$ and for a.e. $t \in (0,T)$,

$$
\begin{aligned}
&\left(A\left(\sigma_{h,i} + \alpha p_{h,i} I\right), \tau\right)_{\Omega_i} + \left(u_{h,i}, \operatorname{div} \tau\right)_{\Omega_i} + \left(\gamma_{h,i}, \tau\right)_{\Omega_i} \\
&\qquad = \langle g_u, \tau\, n_i\rangle_{\partial\Omega_i \cap \Gamma_D^u} + \langle \lambda_H^u, \tau\, n_i\rangle_{\Gamma_i}, && \forall \tau \in \mathbb{X}_{h,i}, \quad (3.2.1)\\
&\left(\operatorname{div} \sigma_{h,i}, v\right)_{\Omega_i} = -\left(f, v\right)_{\Omega_i}, && \forall v \in V_{h,i}, \quad (3.2.2)\\
&\left(\sigma_{h,i}, \xi\right)_{\Omega_i} = 0, && \forall \xi \in \mathbb{Q}_{h,i}, \quad (3.2.3)\\
&\left(K^{-1} z_{h,i}, \zeta\right)_{\Omega_i} - \left(p_{h,i}, \operatorname{div} \zeta\right)_{\Omega_i} = -\langle g_p, \zeta \cdot n_i\rangle_{\partial\Omega_i \cap \Gamma_D^p} - \langle \lambda_H^p, \zeta \cdot n_i\rangle_{\Gamma_i}, && \forall \zeta \in Z_{h,i}, \quad (3.2.4)\\
&c_0\left(\partial_t p_{h,i}, w\right)_{\Omega_i} + \alpha\left(\partial_t A\left(\sigma_{h,i} + \alpha p_{h,i} I\right), w I\right)_{\Omega_i} + \left(\operatorname{div} z_{h,i}, w\right)_{\Omega_i} = \left(g, w\right)_{\Omega_i}, && \forall w \in W_{h,i}, \quad (3.2.5)\\
&\sum_{i=1}^{N} \langle \sigma_{h,i}\, n_i, \mu^u\rangle_{\Gamma_i} = 0, && \forall \mu^u \in \Lambda_H^u, \quad (3.2.6)\\
&\sum_{i=1}^{N} \langle z_{h,i} \cdot n_i, \mu^p\rangle_{\Gamma_i} = 0, && \forall \mu^p \in \Lambda_H^p, \quad (3.2.7)
\end{aligned}
$$

where n_i is the outward unit normal vector field on Ω_i. Note that equations (3.2.6)−(3.2.7) enforces a notion of weak continuity of normal components of the stress tensor and velocity vector across the interface Γ and that both the flow and the elasticity problems are of Dirichlet type.

For simplicity of the analysis, we assume that $\Gamma_D^u = \Gamma_D^p = \partial\Omega$ and $g_u = g_p = 0$, to get the following reformulation of (3.2.1)−(3.2.7): find $(\sigma_h, u_h, \gamma_h, z_h, p_h, \lambda_H) : [0,T] \to \mathbb{X}_h \times V_h \times \mathbb{Q}_h \times$

$Z_h \times W_h \times \Lambda_H$ such that $p_{h,i}(0) = p_{h,0}|_{\Omega_i}$ and for a.e. $t \in (0,T)$,

$$(A(\sigma_h + \alpha p_h I), \tau) + \sum_{i=1}^{N} (u_h, \operatorname{div} \tau)_{\Omega_i} + (\gamma_h, \tau) = \sum_{i=1}^{N} \langle \lambda_H^u, \tau\, n_i \rangle_{\Gamma_i}, \qquad \forall \tau \in \mathbb{X}_h, \tag{3.2.8}$$

$$\sum_{i=1}^{N} (\operatorname{div} \sigma_h, v)_{\Omega_i} = -(f, v), \qquad \forall v \in V_h, \tag{3.2.9}$$

$$(\sigma_h, \xi) = 0, \qquad \forall \xi \in \mathbb{Q}_h, \tag{3.2.10}$$

$$\left(K^{-1} z_h, \zeta\right) - \sum_{i=1}^{N} (p_h, \operatorname{div} \zeta)_{\Omega_i} = \sum_{i=1}^{N} -\langle \lambda_H^p, \zeta \cdot n_i \rangle_{\Gamma_i}, \qquad \forall \zeta \in Z_h, \tag{3.2.11}$$

$$c_0 \left(\frac{\partial p_h}{\partial t}, w \right) + \alpha \left(\frac{\partial}{\partial t} A(\sigma_h + \alpha p_h I), wI \right) + \sum_{i=1}^{N} (\operatorname{div} z_h, w)_{\Omega_i} = (g, w), \quad \forall w \in W_h, \tag{3.2.12}$$

$$\sum_{i=1}^{N} \langle \sigma_h\, n_i, \mu^u \rangle_{\Gamma_i} = 0, \qquad \forall \mu^u \in \Lambda_H^u, \tag{3.2.13}$$

$$\sum_{i=1}^{N} \langle z_h \cdot n_i, \mu^p \rangle_{\Gamma_i} = 0, \qquad \forall \mu^p \in \Lambda_H^p. \tag{3.2.14}$$

3.2.2 Projection and interpolation operators

In this subsection, we discuss various interpolation and projection operators useful in the analysis of the method.

Let $\mathcal{Q}_{h,i}^u : (L^2(\partial\Omega_i))^d \to \mathbb{X}_{h,i} n_i$ and $\mathcal{Q}_{h,i}^p : L^2(\partial\Omega_i) \to Z_{h,i} \cdot n_i$ be projection operators onto the trace of the normal components of $\mathbb{X}_{h,i}$ and $Z_{h,i}$, respectively such that for any $\phi_u \in (L^2(\partial\Omega_i))^d$ and $\phi_p \in L^2(\partial\Omega_i)$,

$$\left\langle \phi_u - \mathcal{Q}_{h,i}^u \phi_u,\, \tau n_i \right\rangle_{\partial\Omega_i} = 0, \qquad \forall \tau \in \mathbb{X}_{h,i}, \tag{3.2.15}$$

$$\left\langle \phi_p - \mathcal{Q}_{h,i}^p \phi_p,\, \zeta \cdot n_i \right\rangle_{\partial\Omega_i} = 0, \qquad \forall \zeta \in Z_{h,i}. \tag{3.2.16}$$

We define $\mathcal{Q}_{h,i} : (L^2(\partial\Omega_i))^d \times L^2(\partial\Omega_i) \to \mathbb{X}_{h,i} n_i \times Z_{h,i} \cdot n_i$ as

$$\mathcal{Q}_{h,i} = \begin{pmatrix} \mathcal{Q}_{h,i}^u \\ \mathcal{Q}_{h,i}^p \end{pmatrix}. \tag{3.2.17}$$

For any inf-sup stable pair of finite element spaces, $\mathbb{X}_{h,i} \times V_{h,i}$, with $\operatorname{div} \mathbb{X}_{h,i} = V_{h,i}$, there exists a mixed canonical interpolant [22], $\Pi_i^\sigma : H^\epsilon(\Omega_i, \mathbb{M}) \cap \mathbb{X}_i \to \mathbb{X}_{h,i}$, for any $\epsilon > 0$, such that for any $\tau \in H^\epsilon(\Omega_i, \mathbb{M}) \cap \mathbb{X}_{h,i}$,

$$(\operatorname{div}(\Pi_i^\sigma \tau - \tau),\, v)_{\Omega_i} = 0, \qquad \forall v \in V_{h,i}, \tag{3.2.18}$$

$$\langle (\Pi_i^\sigma \tau - \tau) n_i,\, \hat{\tau} n_i \rangle_{\Gamma_i} = 0, \qquad \forall \hat{\tau} \in \mathbb{X}_{h,i}, \tag{3.2.19}$$

$$\|\Pi_i^\sigma \tau\|_{\Omega_i} \leq C \left(\|\tau\|_{L^\epsilon(\Omega_i)} + \|\operatorname{div} \tau\|_{\Omega_i} \right). \tag{3.2.20}$$

Similarly for any inf-sup stable pair, $Z_{h,i} \times W_{h,i}$, with $\operatorname{div} Z_{h,i} = W_{h,i}$, there exists a mixed canonical interpolant $\Pi_i^z : (H^\epsilon(\Omega_i))^d \cap Z_i \to Z_{h,i}$ such that for any $\zeta \in (H^\epsilon(\Omega_i))^d \cap Z_i$, the following holds

$$(\operatorname{div}(\Pi_i^z \zeta - \zeta),\, w)_{\Omega_i} = 0, \qquad \forall w \in W_{h,i}, \tag{3.2.21}$$

$$\left\langle (\Pi_i^z \zeta - \zeta) \cdot n,\, \hat{\zeta} \cdot n \right\rangle_{\Gamma_i} = 0, \qquad \forall \hat{\zeta} \in Z_{h,i}, \tag{3.2.22}$$

$$\|\Pi_i^z \zeta\|_{Z_i} \leq C \left(\|\zeta\|_{H^\epsilon(\Omega_i)} + \|\operatorname{div} \zeta\|_{\Omega_i} \right). \tag{3.2.23}$$

Let $\mathcal{P}_{h,i}^p$ denote the L^2 orthogonal projection, $\mathcal{P}_{h,i}^p : L^2(\Omega_i) \to W_{h,i}$, such that for any $w \in L^2(\Omega_i)$,

$$\left(\mathcal{P}_{h,i}^p w - w,\, \hat{w}\right)_{\Omega_i} = 0, \qquad \forall \hat{w} \in W_{h,i}. \tag{3.2.24}$$

Let $\mathcal{P}_{h,i}^u$ denote the L^2 orthogonal projection, $\mathcal{P}_{h,i}^u : (L^2(\Omega_i))^d \to V_{h,i}$, such that for any $v \in (L^2(\Omega_i))^d$,

$$\left(\mathcal{P}_{h,i}^u v - v,\, \hat{v}\right)_{\Omega_i} = 0, \qquad \forall \hat{v} \in V_{h,i}. \tag{3.2.25}$$

We also use $\mathcal{R}_{h,i}$ to denote the orthogonal projection, $\mathcal{R}_{h,i} : L^2(\Omega_i, \mathbb{N}) \to \mathbb{Q}_{h,i}$ such that for any $\xi \in L^2(\Omega_i, \mathbb{N})$,

$$\left(\mathcal{R}_{h,i} \xi - \xi,\, \hat{\xi}\right)_{\Omega_i} = 0, \qquad \forall \hat{\xi} \in \mathbb{Q}_{h,i}. \tag{3.2.26}$$

For the analysis of the method, we will use an elliptic projection operator, $\hat{\Pi}_i^\sigma$ onto $\mathbb{X}_{h,i}$ as defined in [48]. Define $\hat{\Pi}_i^\sigma : H^\epsilon(\Omega_i, \mathbb{M}) \cap \mathbb{X}_i \to \mathbb{X}_{h,i}$ as the operator that takes $\sigma \in H^\epsilon(\Omega_i, \mathbb{M}) \cap \mathbb{X}_i$ to

the finite element approximation, $\hat{\sigma}$, of the following Neumann problem: for any $\sigma \in H^{\epsilon}(\Omega_i, \mathbb{M})$, find $(\hat{\sigma}, \hat{u}, \hat{\gamma}) \in \mathbb{X}_{h,i} \times V_{h,i} \times \mathbb{Q}_{h,i}$ such that

$$
\begin{aligned}
&(\hat{\sigma}, \tau)_{\Omega_i} + (\hat{u}, \operatorname{div} \tau)_{\Omega_i} + (\hat{\gamma}, \tau)_{\Omega_i} = (\sigma, \tau)_{\Omega_i}, && \forall \tau \in \mathbb{X}_{h,i}, \\
&(\operatorname{div} \hat{\sigma}, v)_{\Omega_i} = (\operatorname{div} \sigma, v)_{\Omega_i}, && \forall v \in V_{h,i}, \\
&(\hat{\sigma}, \xi)_{\Omega_i} = (\sigma, \xi)_{\Omega_i}, && \forall \xi \in \mathbb{Q}_{h,i}, \\
&\hat{\sigma} n_i = (\Pi_i^{\sigma} \sigma) n_i \quad \text{on } \partial\Omega_i.
\end{aligned}
$$

More details on the well-posedness and properties of $\hat{\Pi}_i^{\sigma}$ can be found in [48]. In particular, the following bounds hold

$$
\begin{aligned}
&\|\sigma - \hat{\Pi}_i^{\sigma} \sigma\|_{\Omega_i} \leq C \|\sigma - \Pi_i \sigma\|_{\Omega_i}, && \sigma \in H^1(\Omega_i, \mathbb{M}), \\
&\|\hat{\Pi}_i^{\sigma} \sigma\|_{\Omega_i} \leq C \left(\|\sigma\|_{H^{\epsilon}(\Omega_i)} + \|\operatorname{div} \sigma\|_{\Omega_i} \right). && \sigma \in H^{\epsilon}(\Omega_i, \mathbb{M}) \cap \mathbb{X}_i,\ 0 < \epsilon \leq 1.
\end{aligned}
$$

We also use $\mathcal{I}_H^C$ to denote the Scott-Zhang interpolation operator (see [123]) into Λ_H^C, the subset of the mortar space Λ_H that contains continuous functions. Note that $\mathcal{I}_H^C = \begin{pmatrix} \mathcal{I}_H^{C,u} \\ \mathcal{I}_H^{C,p} \end{pmatrix}$, where $\mathcal{I}_H^{C,u}$ and $\mathcal{I}_H^{C,p}$ denote projections onto Λ_H^u and Λ_H^p, respectively. The operators defined above satisfy the following approximation bounds:

$$\|\psi - \mathcal{I}_H^C \psi\|_{t,\Gamma_{i,j}} \leq C H^{\hat{m}-t} \|\psi\|_{\hat{m},\Gamma_{i,j}}, \qquad 0 \leq \hat{m} \leq m+1,\ 0 \leq t \leq 1, \tag{3.2.27}$$

$$\|v - \mathcal{P}_{h,i}^u v\|_{\Omega_i} \leq C h^{\hat{l}} \|v\|_{\hat{l},\Omega_i}, \qquad 0 \leq \hat{l} \leq l+1, \tag{3.2.28}$$

$$\|\zeta - \mathcal{P}_{h,i}^p \zeta\|_{\Omega_i} \leq C h^{\hat{s}} \|\zeta\|_{\hat{s},\Omega_i}, \qquad 0 \leq \hat{s} \leq s+1, \tag{3.2.29}$$

$$\|\xi - \mathcal{R}_{h,i} \xi\|_{\Omega_i} \leq C h^{\hat{j}} \|\xi\|_{\hat{j},\Omega_i}, \qquad 0 \leq \hat{j} \leq j+1, \tag{3.2.30}$$

$$\|\psi - \mathcal{Q}_{h,i}^u \psi\|_{-t,\Gamma_{i,j}} \leq C h^{\hat{k}+t} \|\psi\|_{\hat{k},\Gamma_{i,j}}, \qquad 0 \leq \hat{k} \leq k+1,\ 0 \leq t \leq k+1, \tag{3.2.31}$$

$$\|\psi - \mathcal{Q}_{h,i}^p \psi\|_{-t,\Gamma_{i,j}} \leq C h^{\hat{r}+t} \|\psi\|_{\hat{r},\Gamma_{i,j}}, \qquad 0 \leq \hat{r} \leq r+1,\ 0 \leq t \leq r+1, \tag{3.2.32}$$

$$\|\tau - \hat{\Pi}_i^\sigma \tau\|_{\Omega_i} \leq C h^{\hat{k}} \|\tau\|_{\hat{k},\Omega_i}, \qquad 0 \leq \hat{k} \leq k+1, \tag{3.2.33}$$

$$\|\zeta - \Pi_i^z \zeta\|_{\Omega_i} \leq C h^{\hat{r}} \|\zeta\|_{\hat{r},\Omega_i}, \qquad 0 \leq \hat{r} \leq r+1, \tag{3.2.34}$$

$$\|\operatorname{div}(\tau - \hat{\Pi}_i^\sigma \tau)\|_{\Omega_i} \leq C h^{\hat{l}} \|\operatorname{div} \tau\|_{\hat{l},\Omega_i}, \qquad 0 \leq \hat{l} \leq l+1, \tag{3.2.35}$$

$$\|\operatorname{div}(\zeta - \Pi_i^z \zeta)\|_{\Omega_i} \leq C h^{\hat{s}} \|\operatorname{div} \tau\|_{\hat{s},\Omega_i}, \qquad 0 \leq \hat{s} \leq s+1, \tag{3.2.36}$$

$$\|(\tau - \hat{\Pi}_i^\sigma \tau) n_i\|_{-t,\Gamma_{i,j}} \leq C h^{\hat{k}+t} \|\tau\|_{\hat{k},\Gamma_{i,j}}, \qquad 0 \leq \hat{k} \leq k+1,\ 0 \leq t \leq k+1, \tag{3.2.37}$$

$$\|(\zeta - \Pi_i^z \zeta) \cdot n_i\|_{-t,\Gamma_{i,j}} \leq C h^{\hat{r}+t} \|\zeta\|_{\hat{r},\Gamma_{i,j}}, \qquad 0 \leq \hat{r} \leq r+1,\ 0 \leq t \leq r+1, \tag{3.2.38}$$

where the functions ψ, v, ζ, τ, and ξ are taken from the domains of appropriate operators acting on them. Bound (3.2.27) can be found in [123], bounds (3.2.28)−(3.2.32) and (3.2.35)−(3.2.38) can be found in [24], and bounds (3.2.33)−(3.2.34) can be found in [18, 48, 68].

We will also use the following trace inequalities in the analysis of the method

$$\|\psi\|_{t,\Gamma_{i,j}} \leq C \|\psi\|_{t+\frac{1}{2},\Omega_i}, \qquad t > 0, \tag{3.2.39}$$

$$\langle \psi, \tau n \rangle_{\partial\Omega_i} \leq C \|\psi\|_{\frac{1}{2},\partial\Omega_i} \|\tau\|_{H(\operatorname{div};\Omega_i)}, \tag{3.2.40}$$

which can be found in [111] and [18, 68], respectively.

Finally, define the projection operators $\hat{\Pi}^E$, Π^z, $\mathcal{P}_h^p$, $\mathcal{P}_h^u$, $\mathcal{R}_h$ on respective spaces defined in global domain, Ω, to be the piece-wise application of $\hat{\Pi}_i^\sigma$, Π_i^z, $\mathcal{P}_{h,i}^p$, $\mathcal{P}_{h,i}^p$, $\mathcal{R}_{h,i}$, respectively on subdomains Ω_i for $i = 1, \ldots, N$.

3.2.3 Spaces of weakly continuous stress and velocity

In this section, we introduce the spaces of weakly continuous stress tensors and velocity vectors, which are defined as follows:

$$\mathbb{X}_{h,0} = \left\{ \tau \in \mathbb{X}_h : \sum_{i=1}^{N} \langle \tau n_i, \mu^u \rangle_{\Gamma_i} = 0, \ \ \forall \mu^u \in \Lambda_H^u \right\}$$

and

$$Z_{h,0} = \left\{ \zeta \in Z_h : \sum_{i=1}^{N} \langle \zeta \cdot n_i, \mu^p \rangle_{\Gamma_i} = 0, \ \ \forall \mu^p \in \Lambda_H^p \right\}.$$

In order to find a priori error estimates for the method (3.2.8)−(3.2.14) using techniques developed for single domain system (2.2.8)−(2.2.12) in [6], we restate (3.2.8)−(3.2.14) in terms of $\mathbb{X}_{h,0}$ and $Z_{h,0}$ as follows: find $(\sigma_h, u_h, \gamma_h, z_h, p_h) : [0,T] \to (\mathbb{X}_{h,0}, V_h, \mathbb{Q}_h, Z_{h,0}, W_h)$ such that $p_h(0) = p_{h,0}$ and

$$\left(A\left(\sigma_h + \alpha p_h I\right), \tau\right) + \sum_{i=1}^{N} \left(u_h, \operatorname{div} \tau\right)_{\Omega_i} + \left(\gamma_h, \tau\right) = 0, \qquad \forall \tau \in \mathbb{X}_{h,0}, \tag{3.2.41}$$

$$\sum_{i=1}^{N} \left(\operatorname{div} \sigma_h, v\right)_{\Omega_i} = -\left(f, v\right), \qquad \forall v \in V_h, \tag{3.2.42}$$

$$\left(\sigma_h, \xi\right) = 0, \qquad \forall \xi \in \mathbb{Q}_h, \tag{3.2.43}$$

$$\left(K^{-1} z_h, \zeta\right) - \sum_{i=1}^{N} \left(p_h, \operatorname{div} \zeta\right)_{\Omega_i} = 0, \qquad \forall \zeta \in Z_{h,0}, \tag{3.2.44}$$

$$c_0 \left(\partial_t p_h, w\right) + \alpha \left(\partial_t A\left(\sigma_h + \alpha p_h I\right), w I\right) + \sum_{i=1}^{N} \left(\operatorname{div} z_h, w\right)_{\Omega_i} = \left(g, w\right), \qquad \forall w \in W_h. \tag{3.2.45}$$

Note that constructing basis functions for function spaces, $\mathbb{X}_{h,0}$ and $Z_{h,0}$, is difficult and we use the above formulation only for the sake of error analysis. In the later sections, we will present a reduction to interface problem approach to design the numerical algorithm using any of the popular sub-domain spaces $\mathbb{X}_{h,i}$ and $Z_{h,i}$ discussed earlier.

3.3 Analysis of the MMMFE Method

In this section, we present well-posedness and error analysis of the DD formulation developed in the previous section. We start out by proving inf-sup stability bounds for weakly continuous stress, $\mathbb{X}_{h,0}$, and velocity, $Z_{h,0}$, spaces under appropriate conditions on the mortar space, Λ_H. Under the same conditions, we show that the multiscale mortar DD method is well-posed and stable. We finish the section by proving a combined a priori error bound for all the variables in the formulation.

3.3.1 Inf-sup stability for the weakly continuous spaces

In this subsection, we give inf-sup stability bounds for the weakly continuous stress, $\mathbb{X}_{h,0}$, and velocity, $Z_{h,0}$, spaces under appropriate conditions on the mortar space, Λ_H.

Assumption 1. *The mortar space Λ_H is chosen so that there exists a positive constant C independent of H and h such that the following inequality holds:*

$$\|\mu\|_{\Gamma_{i,j}} \leq C\left(\|\mathcal{Q}_{h,i}\mu\|_{\Gamma_{i,j}} + \|\mathcal{Q}_{h,j}\mu\|_{\Gamma_{i,j}}\right), \quad \forall \mu \in \Lambda_H, \quad 1 \leq i < j \leq n. \tag{3.3.1}$$

Remark 3.3.1. *Note that assumption (3.3.1) implies that the space Λ_H cannot be too rich compared to subdomain stress-velocity FE spaces (similar approach to [9]) in the sense that Λ_H^u and Λ_H^p are well controlled by their projections on to the normal traces of stress and velocity sub-domain spaces respectively. In practice, this condition can be easily obtained by taking a coarser mortar mesh satisfying $h < H \leq 1$(see [8, 9, 63]).*

Lemma 3.3.1. *Under the assumption (3.3.1), there exists a constant $\beta_D > 0$, independent of h and H such that for any $\mu^p \in \Lambda_H^p$, the following holds:*

$$\|\mu^p\|_\Gamma \leq \beta_D \sup_{0 \neq \zeta \in Z_h} \frac{\sum_{i=1}^N \langle \zeta \cdot n_i, \mu^p \rangle_{\Gamma_i}}{\|\zeta\|_{Z_h}}. \tag{3.3.2}$$

Proof. We start with any $\mu^p \in \Lambda_H^p$ and extend it by zero on $\partial\Omega$. Let ϕ_i be the solution to the following auxiliary problem

$$\operatorname{div} \nabla \phi_i = \overline{\mathcal{Q}_{h,i}^p \mu^p}, \qquad \text{in } \Omega_i, \tag{3.3.3}$$

$$\nabla \phi_i \cdot n_i = \mathcal{Q}_{h,i}^p \mu^p, \qquad \text{on } \partial\Omega_i, \tag{3.3.4}$$

where $\overline{\mathcal{Q}^p_{h,i}\mu^p}$ denotes the mean value of $\mathcal{Q}^p_{h,i}\mu^p$ on $\partial\Omega_i$. The above problem can be reformulated in the mixed form by defining $\psi_i = \nabla\phi_i$. The aforementioned elliptic problem is well-posed and the elliptic regularity (see [111]) gives

$$\|\psi_i\|_{1/2,\Omega_i} + \|\operatorname{div}\psi\|_{\Omega_i} \leq C\|\mathcal{Q}^p_{h,i}\mu^p\|_{\partial\Omega_i}. \tag{3.3.5}$$

Take $\zeta_{h,i} = \Pi^z\psi_i \in Z_{h,i}$ and (3.2.22) combined with (3.3.4) implies that $\zeta_{h,i}\cdot n_i = \mathcal{Q}^p_{h,i}\mu$ on $\partial\Omega_i$. This equality along with the definition of $\mathcal{Q}^p_{h,i}$ imply

$$\begin{aligned}\sum_{i=1}^N\langle\zeta_{h,i}\cdot n_i,\mu^p\rangle_{\Gamma_i} &= \sum_{i=1}^N\langle\Pi^z\psi_i\cdot n_i,\mu^p\rangle_{\partial\Omega_i} = \sum_{i=1}^N\langle\Pi^z\psi_i\cdot n_i,\mathcal{Q}^p_{h,i}\mu^p\rangle_{\partial\Omega_i}\\ &= \sum_{i=1}^N\langle\mathcal{Q}^p_{h,i}\mu^p,\mathcal{Q}^p_{h,i}\mu^p\rangle_{\partial\Omega_i} \geq C\sum_{i=1}^N\|\mu\|_{\Gamma_i},\end{aligned} \tag{3.3.6}$$

where we have used the mortar coarseness assumption (3.3.1).

Next, we note that

$$\|\zeta_{h,i}\|_{Z_i} \leq C\sum_{i=1}^N\|\mu\|_{\Gamma_i}, \tag{3.3.7}$$

which follows from the stability of the canonical projection Π^z_i, (3.2.23), with $\epsilon = 1/2$, (3.3.5) and the stability of $\mathcal{Q}^p_{h,i}$.

Finally, (3.3.7) combined with (3.3.6) and defining $\zeta := \zeta_{h,i}$ on Ω_i completes the proof. □

Lemma 3.3.2. *Under the assumption* (3.3.1)*, there exists a constant $\beta_E > 0$, independent of h and H such that for any $\mu^u \in \Lambda^u_H$, the following bound holds*

$$\|\mu^u\|_\Gamma \leq \beta_E \sup_{0\neq\tau\in\mathbb{X}_h}\frac{\sum_{i=1}^N\langle\tau n_i,\mu^u\rangle_{\Gamma_i}}{\|\tau\|_{\mathbb{X}_h}}. \tag{3.3.8}$$

Proof. The proof follows similar arguments as in the proof of the previous lemma, starting with any $\mu^u \in \Lambda^u_H$ and using elliptic regularity of corresponding elliptic problem and stability of the projections Π^σ_i and $\mathcal{Q}^u_{h,i}$. □

Lemma 3.3.3. *Under the assumption (3.3.1), there exists a linear operator* $\Pi_0^\sigma : H^{\frac{1}{2}+\epsilon}(\Omega, \mathbb{M}) \cap \mathbb{X} \to \mathbb{X}_{h,0}$ *for any* $\epsilon > 0$, *such that for any* $1 \leq i \leq N$ *and* $\tau \in H^{\frac{1}{2}+\epsilon}(\Omega, \mathbb{M}) \cap \mathbb{X}$,

$$\sum_{i=1}^{N} \left(\operatorname{div}\left(\Pi_0^\sigma \tau - \tau\right), v\right)_{\Omega_i} = 0, \qquad \forall v \in V_{h,i}, \tag{3.3.9}$$

$$\left(\Pi_0^\sigma \tau - \tau, \xi\right) = 0, \qquad \forall \xi \in \mathbb{Q}_h, \tag{3.3.10}$$

$$\|\Pi_0^\sigma \tau\| \leq C \left(\|\tau\|_{\frac{1}{2}+\epsilon} + \|\operatorname{div} \tau\| \right), \tag{3.3.11}$$

$$\|\Pi_0^\sigma \tau - \tau\| \leq C \left(h^{\tilde{t}} \|\tau\|_{\tilde{t}} + h^{\tilde{k}} H^{\frac{1}{2}} \|\tau\|_{\tilde{k}+\frac{1}{2}} \right), \qquad 0 \leq \tilde{t} \leq k+1,\ 0 < \tilde{k} \leq k+1. \tag{3.3.12}$$

Proof. The proof is based on a construction, $\Pi_0^\sigma|_{\partial\Omega_i} = \hat{\Pi}_i\left(\tau + \delta\tau_i\right)$, where the correction $\delta\tau_i$ is designed to generate weak continuity of the normal components. A complete proof is given in [48, Lemma 4.6]. □

Lemma 3.3.4. *Under the assumption (3.3.1), there exists a linear operator* $\Pi_0^z : \left(H^{\frac{1}{2}+\epsilon}(\Omega)\right)^d \cap Z \to Z_{h,0}$ *such that for any* $1 \leq i \leq N$ *and* $\zeta \in \left(H^{\frac{1}{2}+\epsilon}(\Omega)\right)^d \cap Z$,

$$\sum_{i=1}^{N} \left(\operatorname{div}\left(\Pi_0^z \zeta - \zeta\right), w\right)_{\Omega_i} = 0, \qquad \forall w \in W_h, \tag{3.3.13}$$

$$\|\Pi_0^z \zeta\|_{Z_h} \leq C \left(\|\zeta\|_{\frac{1}{2}+\epsilon} + \|\operatorname{div} \zeta\| \right), \tag{3.3.14}$$

$$\|\Pi_0^z \zeta - \zeta\| \leq C \sum_{i=1}^{N} \left(h^{\tilde{r}} \|\zeta\|_{\tilde{r},\Omega_i} + h^{\tilde{r}} H^{\frac{1}{2}} \|\zeta\|_{\tilde{r}+\frac{1}{2},\Omega_i} \right), \qquad 1 \leq \tilde{r} \leq r+1. \tag{3.3.15}$$

$$\tag{3.3.16}$$

Proof. A detailed proof of the lemma can be found in [8] and [9, Section 3]. □

Lemma 3.3.3 and Lemma 3.3.4 along with a simple variant of Fortin's Lemma [18, 37] gives the following theorem, which essentially gives inf-sup stability bounds with respect to the weakly continuous spaces of stress and velocity.

Lemma 3.3.5. *Under the assumption (3.3.1), there exists positive constants* C_E *and* C_D *independent of the discretization parameters* h *and* H *such that for any* $v \in V_h$ *and* $\xi \in \mathbb{Q}_h$,

$$\|v\| + \|\xi\| \leq C_E \sup_{0 \neq \tau \in \mathbb{X}_{h,0}} \frac{\sum_{i=1}^{N} \left(v, \operatorname{div} \tau\right)_{\Omega_i} + \left(\xi, \tau\right)}{\|\tau\|_{\mathbb{X}_h}}, \tag{3.3.17}$$

$$\|w\| \leq C_D \sup_{0 \neq \zeta \in Z_{h,0}} \frac{\sum_{i=1}^{N} (\operatorname{div} \zeta,\, w)_{\Omega_i}}{\|\zeta\|_{Z_h}}, \tag{3.3.18}$$

for any $w \in W_h$.

3.3.2 Well-posedness of the semi-discrete MMMFE formulation

In this subsection, we show the existence of a unique solution to the system of equations (3.2.8)−(3.2.14) under the assumption (3.3.1). We follow closely the proof for the well-posedness of the multipoint flux method for the Biot system given in [6]. We base our proof on the theory for showing the existence of solution to a degenerate parabolic system [73]. In particular, we use [73, IV, Theorem 6.1(b)] which is stated as follows:

Theorem 3.3.6. *Let the linear, symmetric, and monotone operator $\mathcal{N}$ be given for the real vector space E to its algebraic dual E^*, and let E_b' be the Hilbert space which is the dual of E with the seminorm $|x|_b = \sqrt{\mathcal{N}x(x)}$ for $x \in E$. Let $\mathcal{M} \subset E \times E_b'$ be a relation with the domain $D = \{x \in E : \mathcal{M}(x) \neq \emptyset\}$. Assume that $\mathcal{M}$ is monotone and Range$(\mathcal{N} + \mathcal{M}) = E_b'$. Then for each $x_0 \in D$ and for each $\mathcal{F} \in W^{1,1}\left(0, T; E_b'\right)$, there is a solution x of*

$$\frac{d}{dt}\left(\mathcal{N}x(t) + \mathcal{M}\left(x(t)\right)\right) \ni \mathcal{F}(t), \qquad a.e.\ 0 < t < T,$$

with

$$\mathcal{N}x \in W^{1,\infty}\left(0, T; E_b'\right), x(t) \in D, \text{ for all } 0 \leq t \leq T, \text{ and } \mathcal{N}x(0) = \mathcal{N}x_0.$$

Using the above theorem, we now prove that the semi-discrete system (3.2.8)−(3.2.14) is well-posed.

Theorem 3.3.7. *For each $(f, g) \in W^{1,\infty}\left(0, T; \left(L^2(\Omega)\right)^d\right) \times W^{1,\infty}\left(0, T; L^2(\Omega)\right)$ and compatible initial data $(\sigma_{h,0}, u_{h,0}, \gamma_{h,0}, z_{h,o}, p_{h,0}, \lambda_{H,0})$, the system of equations (3.2.8)−(3.2.14) has a unique solution $(\sigma_h, u_h, \gamma_h, z_h, p_h, \lambda_H)$ provided that the assumption (3.3.1) holds.*

Proof. We start by reformulating (3.2.8)−(3.2.14) to fit the setting of Theorem 3.3.6. For this purpose, we define operators

$$(A_{\sigma\sigma}\sigma_h,\, \tau) = (A\sigma_h,\, \tau)\,, \quad (A_{\sigma p}\sigma_h,\, w) = \alpha\,(A\sigma_h,\, wI)\,, \quad (A_{\sigma u}\sigma_h,\, v) = \sum_{i=1}^{N} (\operatorname{div}\sigma_{h,i},\, v)\,,$$

$$(A_{\sigma\gamma}\sigma_h,\, \xi) = (\sigma_h,\, \xi)\,, \quad (A_{\sigma\lambda}\sigma_h,\, \mu^u) = \sum_{i=1}^{N} \langle \sigma_h n_i,\, \mu^u\rangle_{\Gamma_i}\,, \quad (A_{zz}z_h,\, \zeta) = \left(K^{-1}z_h,\, \zeta\right),$$

$$(A_{zp}z_h,\, w) = -\sum_{i=1}^{N} (\operatorname{div} z_{h,i},\, w)\,, \quad (A_{z\lambda}z_h,\, \mu^p) = \sum_{i=1}^{N} \langle z_h\cdot n_i,\, \mu^p\rangle_{\Gamma_i}\,,$$

$$(A_{pp}p_h,\, w) = c_0\,(p_h,\, w) + \alpha^2\,(Ap_hI,\, wI)\,.$$

Let us introduce the new variables $\dot{u}_h$, $\dot{\gamma}_h$, and $\dot{\lambda}_H^u$ representing $\partial_t u_h$, $\partial_t\gamma_h$, and $\partial_t\lambda_H^u$, respectively. We differentiate equation (3.2.8) in time to get

$$(\partial_t A\,(\sigma_h + \alpha p_h I)\,,\tau) + \sum_{i=1}^{N} (\dot{u}_h, \operatorname{div}\tau)_{\Omega_i} + (\dot{\gamma}_h,\tau) = \sum_{i=1}^{N} \left(\dot{\lambda}_H^u, \tau n_i\right)_{\Gamma_i}, \qquad \forall\tau\in\mathbb{X}_h. \tag{3.3.19}$$

Using the above definitions of operators and using (3.3.19) instead of equation (3.2.8), we can reformulate the problem as a system of linear equations

$$\frac{d}{dt}\left(\mathcal{N}\dot{x}(t) + \mathcal{M}\left(\dot{x}(t)\right)\right) \;= \mathcal{F}(t) \quad 0 < t < T, \tag{3.3.20}$$

where

$$\dot{x} = \begin{pmatrix} \sigma_h \\ \dot{u}_h \\ \dot{\gamma}_h \\ z_h \\ p_h \\ \dot{\lambda}_H^u \\ \lambda_H^p \end{pmatrix}, N = \begin{pmatrix} A_{\sigma\sigma} & 0 & 0 & 0 & A_{\sigma p}^T & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A_{\sigma p} & 0 & 0 & 0 & A_{pp} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$M = \begin{pmatrix} 0 & A_{\sigma u}^T & A_{\sigma\gamma}^T & 0 & 0 & -A_{\sigma\lambda}^T & 0 \\ -A_{\sigma u} & 0 & 0 & 0 & 0 & 0 & 0 \\ -A_{\sigma\gamma} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & A_{zz} & A_{zp}^T & 0 & A_{z\lambda}^T \\ 0 & 0 & 0 & -A_{zp} & 0 & 0 & 0 \\ A_{\sigma\lambda} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & A_{z\lambda} & 0 & 0 & 0 \end{pmatrix}, \mathcal{F} = \begin{pmatrix} 0 \\ -f \\ 0 \\ 0 \\ g \\ 0 \\ 0 \end{pmatrix}.$$

The dual space $E_b^{'}$ is given by $L^2(\Omega, \mathbb{M}) \times 0 \times 0 \times 0 \times L^2(\Omega) \times 0 \times 0$ and the condition $\mathcal{F} \in W^{1,1}\left(0, T; E_b^{'}\right)$ implies that non-zero source terms can appear only in equations with time derivatives. This means we have to take $f = 0$ in our case. We can fix this issue by considering an auxiliary problem that, for each $t \in (0, T]$, solves the system

$$\begin{pmatrix} A_{\sigma\sigma} & A_{\sigma u}^T & A_{\sigma\gamma}^T & -A_{\sigma\lambda}^T \\ -A_{\sigma u} & 0 & 0 & 0 \\ -A_{\sigma\gamma} & 0 & 0 & 0 \\ A_{\sigma\lambda} & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \sigma_h^f \\ \partial_t u_h^f \\ \partial_t \gamma_h^f \\ \partial_t \lambda_H^{u,f} \end{pmatrix} = \begin{pmatrix} 0 \\ -f \\ 0 \\ 0 \end{pmatrix}. \tag{3.3.21}$$

Such an auxiliary system (3.3.21) is well-posed and the proof can be found in [48]. Now we can subtract the solution to (3.3.21) from the original system of equations (3.2.8)−(3.2.14) to obtain the modified RHS $\mathcal{F} = \left(A_{\sigma\sigma}\left(\sigma_h^f - \partial_t \sigma_h^f\right), 0, 0, 0, q - A_{\sigma p}\partial_t \sigma_h^f, 0, 0\right)^T$.

Next we show that $Range(\mathcal{N}+\mathcal{M}) = E_b'$. This can be established by showing that the following square linear homogeneous system has only the trivial solution: $(\hat{\sigma}_h, \hat{u}_h, \hat{\gamma}_h, \hat{z}_h, \hat{p}_h, \hat{\lambda}_H) \in \mathbb{X}_h \times V_h \times \mathbb{Q}_h \times Z_h \times W_h \times \Lambda_H$ such that

$$\left(A\left(\hat{\sigma}_h + \alpha\hat{p}_h I\right), \tau\right) + \sum_{i=1}^{N} \left(\hat{u}_h, \operatorname{div}\tau\right)_{\Omega_i} + \left(\hat{\gamma}_h, \tau\right) - \sum_{i=1}^{N} \langle \hat{\lambda}_H^u, \tau\, n_i\rangle_{\Gamma_i} = 0, \qquad \forall \tau \in \mathbb{X}_h, \tag{3.3.22}$$

$$\sum_{i=1}^{N} \left(\operatorname{div}\hat{\sigma}_h, v\right)_{\Omega_i} = 0, \qquad \forall v \in V_h,$$

$$\left(\hat{\sigma}_h, \xi\right) = 0, \qquad \forall \xi \in \mathbb{Q}_h,$$

$$\left(K^{-1}\hat{z}_h, \zeta\right) - \sum_{i=1}^{N} \left(\hat{p}_h, \operatorname{div}\zeta\right)_{\Omega_i} + \sum_{i=1}^{N} \langle \hat{\lambda}_H^p, \zeta \cdot n_i\rangle_{\Gamma_i} = 0, \qquad \forall \zeta \in Z_h, \tag{3.3.23}$$

$$c_0\left(\partial_t \hat{p}_h, w\right) + \alpha\left(A\left(\hat{\sigma}_h + \alpha\hat{p}_h I\right), wI\right) + \sum_{i=1}^{N} \left(\operatorname{div}\hat{z}_h, w\right)_{\Omega_i} = 0, \qquad \forall w \in W_h,$$

$$\sum_{i=1}^{N} \langle \hat{\sigma}_h n_i, \mu^u\rangle_{\Gamma_i} = 0, \qquad \forall \mu^u \in \Lambda_H^u,$$

$$\sum_{i=1}^{N} \langle \hat{z}_h \cdot n_i, \mu^p\rangle_{\Gamma_i} = 0, \qquad \forall \mu^p \in \Lambda_H^p.$$

Taking appropriate test functions $(\tau, v, \xi, \zeta, w, \mu^u, \mu^p) = (\hat{\sigma}_h, \hat{u}_h, \hat{\gamma}_h, \hat{z}_h, \hat{p}_h, \hat{\lambda}_H^u, \hat{\lambda}_H^p)$ in the above system and adding the equations together gives $\|A^{\frac{1}{2}}\left(\hat{\sigma}_h + \alpha\hat{p}_h I\right)\|^2 + c_0\|\hat{p}_h\|^2 + \|K^{-\frac{1}{2}}\hat{z}_h\|^2 = 0$. The coercivity of A, (2.3.31), and K, (2.3.32), give $\hat{\sigma}_h + \alpha\hat{p}_h I = 0$ and $\hat{z}_h = 0$ respectively. Further, the inf-sup condition with respect to the weakly continuous space $Z_{h,0}$, (3.3.18), implies $\hat{p}_h = 0$ and hence we also have $\hat{\sigma}_h = 0$. Inf-sup condition with respect to the weakly continuous space $\mathbb{X}_{h,0}$, (3.3.17), implies $\hat{u}_h = 0$ and $\hat{\gamma}_h = 0$. Finally, (3.3.2) combined with (3.3.22) implies $\hat{\lambda}_H^u = 0$, and (3.3.8) combined with (3.3.23) implies $\hat{\lambda}_H^p = 0$. Similar arguments can be used to show that $\mathcal{N}$ and $\mathcal{M}$ are non-negative and therefore due to linearity, monotone.

Now to completely satisfy the hypothesis of Theorem 3.3.6, we need compatible initial data $\dot{x}_0 \in D$ which implies $\mathcal{M}\dot{x}_0 \in E_b'$. We first construct compatible initial data to the continuous system (2.2.3)-(2.2.7), $(\sigma_0, u_0, \gamma_0, z_0, p_0)^T$, from continuous initial condition, p_0, as follows:

1. Solve equations (2.2.3)$-$(2.2.5) using $p = p_0$ as given initial data to get σ_0, u_0, γ_0.
2. Set $z_0 = -K\nabla p_o$ and it is easy to show using integration by parts that this choice satisfies equation (2.2.6) with $p = p_0$.

Define $\tilde{x}_0 = (\sigma_0, u_0, \gamma_0, z_0, p_0, \lambda^u_{H,0}, \lambda^p_{H,0})^T$, where $\lambda^u_{H,0} = u_0|_\Gamma$ and $\lambda^p_{H,0} = p_0|_\Gamma$. Take the initial data to the system (3.2.8)−(3.2.7), x_0, to be the elliptic projection of $\tilde{x}_0$. With the reduction of the problem to the case with $f = 0$, we have $(\mathcal{N} + \mathcal{M})\,\tilde{x}_0 \in E_b'$. We also have

$$(\mathcal{N} + \mathcal{M})\,x_0 = (\mathcal{N} + \mathcal{M})\,\tilde{x}_0, \tag{3.3.24}$$

which implies $\mathcal{M}x_0 = (\mathcal{N} + \mathcal{M})\,\tilde{x}_0 - \mathcal{N}x_0 \in E_b'$. Now for the modified system (3.3.20), we take the initial data, $\dot{x}_0$ to be $(\sigma_{h,0}, 0, 0, z_{h,0}, p_{h,0}, 0, 0)$, which also satisfies $\mathcal{M}\dot{x}_0 \in E_b'$. Note that initial data $u_{h,0}$, $\gamma_{h,0}$ and $\lambda^u_{H,0}$ are not needed to solve (3.3.20), but will be used later to recover solution to the original problem.

Now we can apply Theorem 3.3.6 to prove the existence of a unique solution

$$\dot{x} = (\sigma_h, \dot{u}_h, \dot{\gamma}_h, z_h, p_h, \lambda_H),$$

such that $\sigma_h(0) = \sigma_{h,0}$ and $p_h(0) = p_{h,0}$. It is also easy to see that $z_h(0) = z_{h,0}$ by taking $t \to 0$ in (3.2.44) and using the fact that $z_{h,0}$ and $p_{h,0}$ satisfy (3.2.44). Finally for each $t \in [0, T]$, we define u_h , γ_h, and λ^u_H as follows:

$$u_h(t) = u_{h,0} + \int_0^t \dot{u}_h(s)ds,$$

$$\gamma_h(t) = \gamma_{h,0} + \int_0^t \dot{\gamma}_h(s)ds,$$

and

$$\lambda^u_H(t) = \lambda^u_{H,0} + \int_0^t \dot{\lambda}^u_H(s)ds.$$

It is easy to show that $u_h(t)$, $\gamma_h(t)$ and $\lambda^u_H(t)$ satisfy equation (3.2.8). We can indeed verify this by integrating the differentiated version of this equation namely equation (3.3.19) with respect to time from 0 to any $t \in (0, T]$ and using the fact that $\sigma_{h,0}$, $u_{h,0}$, $\gamma_{h,0}$, and $\lambda^u_{H,0}$ satisfy equation (3.2.8). This completes the proof that (3.2.18)−(3.2.7) has a solution $(\sigma_h, u_h, \gamma_h, z_h, p_h, \lambda_H)$. Uniqueness of the above constructed solution follows from the stability bound for the solution variables which will be discussed in the next section. □

3.3.3 Stability analysis for MMMFE formulation

In this subsection, we give a stability bound for the system (3.2.8)−(3.2.14).

Theorem 3.3.8. *Under the assumption (3.3.1), there exists a constant $C > 0$, independent of discretization parameters h and H, and c_0 such that the following stability bound holds for the solution of (3.2.8)−(3.2.14):*

$$\begin{aligned}
&\|\sigma_h\|_{L^\infty(0,T;\mathbb{X}_h)} + \|u_h\|_{L^\infty(0,T;L^2(\Omega))} + \|\gamma_h\|_{L^\infty(0,T;L^2(\Omega))} + \|z_h\|_{L^\infty(0,T;L^2(\Omega))} + \|p_h\|_{L^\infty(0,T;L^2(\Omega))} \\
&+ \|\lambda_H^u\|_{L^\infty(0,T;L^2(\Gamma))} + \|\lambda_H^p\|_{L^\infty(0,T;L^2(\Gamma))} + \|\sigma_h\|_{L^2(0,T;\mathbb{X}_h)} + \|u_h\|_{L^2(0,T;L^2(\Omega))} + \|\gamma_h\|_{L^2(0,T;L^2(\Omega))} \\
&+ \|z_h\|_{L^2(0,T;Z_h)} + \|p_h\|_{L^2(0,T;L^2(\Omega))} + \|\lambda_H^u\|_{L^2(0,T;L^2(\Gamma))} + \|\lambda_H^p\|_{L^2(0,T;L^2(\Gamma))} \\
&\qquad \le C\big(\|f\|_{H^1(0,T;L^2(\Omega))} + \|g\|_{H^1(0,T;L^2(\Omega))} + \|p_0\|_{H^1(\Omega)} + \|\nabla K p_0\|_{H(\mathrm{div},\Omega)}\big).
\end{aligned}$$

Proof. We start out by choosing the test functions

$$(\tau, v, \xi, \zeta, w, \mu^u, \mu^p) = (\sigma_h, \partial_t u_h, \partial_t \gamma_h, z_h, p_h, \partial_t \lambda_H^u, \lambda_H^p),$$

in equations (3.3.19) and (3.2.9)−(3.2.14) and combining them to get

$$\left(\partial_t A\left(\sigma_h + \alpha p_h I\right), \sigma_h + \alpha p_h I\right) + c_0\left(\partial_t p_h, p_h\right) + \left(K^{-1} z_h, z_h\right) = \left(f, \partial_t u_h\right) + \left(q, p_h\right).$$

The above equation can be rewritten as

$$\frac{1}{2}\partial_t\left(\|A^{\frac{1}{2}}\left(\sigma_h + \alpha p_h I\right)\|^2 + c_0\|p_h\|^2\right) + \|K^{-\frac{1}{2}} z_h\|^2 = \partial_t\left(f, u_h\right) - \left(\partial_t f, u_h\right) + \left(g, p_h\right). \tag{3.3.25}$$

For any $t \in (0, T]$, we integrate equation (3.3.25) with respect to time from 0 to t to get

$$\begin{aligned}
&\frac{1}{2}\left(\|A^{\frac{1}{2}}\left(\sigma_h + \alpha p_h I\right)(t)\|^2 + c_0\|p_h(t)\|^2\right) + \int_0^t \|K^{-\frac{1}{2}} z_h\|^2 ds \\
&= \frac{1}{2}\left(\|A^{\frac{1}{2}}\left(\sigma_h + \alpha p_h I\right)(0)\|^2 + c_0\|p_h(0)\|^2\right) + \int_0^t \left(\left(g, p_h\right) - \left(\partial_t f, u_h\right)\right) ds + \left(f, u_h\right)(t) - \left(f, u_h\right)(0).
\end{aligned}$$

On applications of the Cauchy-Schwartz and Young's inequalities, we get

$$\begin{aligned}
&\|A^{\frac{1}{2}}\left(\sigma_h + \alpha p_h I\right)(t)\|^2 + c_0\|p_h(t)\|^2 + 2\int_0^t \|K^{-\frac{1}{2}} z_h\|^2 ds \\
&\le \|A^{\frac{1}{2}}\left(\sigma_h + \alpha p_h I\right)(0)\|^2 + c_0\|p_h(0)\|^2 + \epsilon_1\left(\int_0^t \left(\|p_h\|^2 + \|u_h\|^2\right) ds + \|u_h(t)\|^2\right) \\
&\quad + \frac{1}{\epsilon_1}\left(\int_0^t \left(\|g\|^2 + \|\partial_t f\|^2\right) ds + \|f(t)\|^2\right) + \|f(0)\|^2 + \|u_h(0)\|^2.
\end{aligned} \tag{3.3.26}$$

Bounds for $\|u_h\|$ and $\|\gamma_h\|$ follow from the inf-sup condition (3.3.17) as follows:

$$\|u_h\| + \|\gamma_h\| \leq C_E \sup_{0 \neq \tau \in \mathbb{X}_{h,0}} \frac{\sum_{i=1}^{N} (u_h, \operatorname{div} \tau)_{\Omega_i} + (\gamma_h, \tau)}{\|\tau\|_{\mathbb{X}_h}}.$$

We combine the above equation along with (3.2.8) and the Cauchy-Schwartz inequality to obtain the bound

$$\|u_h\| + \|\gamma_h\| \leq C_E \sup_{0 \neq \tau \in \mathbb{X}_{h,0}} \frac{\left(A^{\frac{1}{2}} (\sigma_h + \alpha p_h I), A^{\frac{1}{2}} \tau\right)}{\|\tau\|_{\mathbb{X}_h}} \leq C \|A^{\frac{1}{2}} (\sigma_h + \alpha p_h I)\|. \tag{3.3.27}$$

Equation (3.3.27) also gives,

$$\int_0^t \left(\|u_h\|^2 + \|\gamma_h\|^2\right) ds \leq C \int_0^t \left(\|\sigma_h\|^2 + \|p_h\|^2\right) ds. \tag{3.3.28}$$

Further, choose test functions $(\tau, v, \xi, \mu^u) = (\sigma_h, u_h, \gamma_h, \lambda_H^u)$ in (3.2.8)−(3.2.10), (3.2.13) and combine the equations along with Cauchy-Schwartz inequality to get

$$\|\sigma_h\|^2 \leq C \left(\|p_h^2\| + \epsilon_2 \|u_h\|^2 + \frac{1}{\epsilon_2} \|f\|^2 \right).$$

Combining the above inequality with inequality (3.3.28) yields the following bound

$$\int_0^t \left(\|u_h\|^2 + \|\gamma_h\|^2\right) ds \leq C \int_0^t \left(\|p_h^2\| + \epsilon_2 \|u_h\|^2 + \frac{1}{\epsilon_2} \|f\|^2 \right) ds. \tag{3.3.29}$$

Bound for $\|p_h\|$ can be obtained from the inf-sup condition (3.3.18) and equation (3.2.11) as follows:

$$||p_h|| \leq C_D \sup_{0 \neq \zeta_h \in Z_{h,0}} \frac{\sum_{i=1}^{N} (\operatorname{div} \zeta_h, p_h)_{\Omega_i}}{||\zeta_h||_{Z_h}} = C_D \sup_{0 \neq \zeta_h \in Z_{h,0}} \frac{(K^{-1} z_h, \zeta_h)}{||\zeta_h||_{Z_h}} \leq C \|K^{-1} z_h\|, \tag{3.3.30}$$

where the last inequality follows from the Cauchy-Schwartz inequality.

Further, taking test function $v|_{\Omega_i} = \operatorname{div} \sigma_h|_{\Omega_i}$ in (3.2.9) and using Cauchy-Schwartz inequality yields

$$\sum_{i=1}^{N} \|\operatorname{div} \sigma_h\|_{\Omega_i}^2 \leq \|f\|^2. \tag{3.3.31}$$

Finally, combining inequalities (3.3.26)−(3.3.27) and (3.3.29)−(3.3.31) and taking ϵ_1 and ϵ_2 small enough give the following bound

$$\begin{aligned}
&\sum_{i=1}^{N}\Big(\|A^{\frac{1}{2}}\left(\sigma_h+\alpha p_h I\right)(t)\|_{\Omega_i}^2+\|\text{div }\sigma_h(t)\|_{\Omega_i}^2\Big)+\|u_h(t)\|^2+\|\gamma_h(t)\|^2+c_0\|p_h(t)\|^2\\
&\quad+\sum_{i=1}^{N}\int_0^t\left(\|\sigma_h\|_{\Omega_i}^2+\|\text{div }\sigma_h\|_{\Omega_i}^2\right)ds+\int_0^t\Big(\|u_h\|^2+\|\gamma_h\|^2+\|K^{-\frac{1}{2}}z_h\|^2+\|p_h\|^2\Big)\,ds\\
&\leq C\Big(\int_0^t\left(\|g(s)\|^2+\|\partial_t f(s)\|^2+\|f(s)\|^2\right)ds+\|f(t)\|^2+\|\sigma_h(0)\|^2\\
&\qquad+\|u_h(0)\|^2+\|p_h(0)\|^2+\|f(0)\|^2\Big).
\end{aligned}\tag{3.3.32}$$

Next we give bounds for $\|\text{div }z_h\|$, $\|K^{-\frac{1}{2}}z_h(t)\|$ and $\|p_h(t)\|$ for all $t\in(0,t]$, which are independent of c_0.

We start by choosing test function $w=\text{div }z_h$ in (3.2.12) and apply Cauchy-Schwartz inequality to obtain

$$\sum_{i=1}^{N}\|\text{div }z_h\|_{\Omega_i}\leq C\left(c_0\|\partial_t p_h\|+\|\partial_t A^{\frac{1}{2}}\left(\sigma_h+\alpha p_h I\right)(t)\|+\|q\|\right).\tag{3.3.33}$$

To bound the right hand side of (3.3.33), differentiate equations (3.2.8)−(3.2.11), and (3.2.13)-−(3.2.14) with respect to time and take appropriate test functions, namely

$$(\tau,v,\xi,\zeta,w,\mu^u,\mu^p)=(\partial_t\sigma_h,\partial_t u_h,\partial_t\gamma_h,z_h,\partial_t p_h,\partial_t\lambda_H^u,\lambda_H^p),$$

in the differentiated equations and equation (3.2.12). Further, combining the resulting equations and integrating in time from 0 to $t\in(0,T]$ similar to what we did for equations (3.3.25)−(3.3.26) previously, we obtain

$$\begin{aligned}
&2\int_0^t\Big(\|\partial_t A^{\frac{1}{2}}\left(\sigma_h+\alpha p_h I\right)\|^2+c_0\partial_t\|p_h\|^2\Big)\,ds+\|K^{-\frac{1}{2}}z_h(t)\|^2\\
&\leq\epsilon_3\left(\int_0^t\|\partial_t u_h\|^2ds+\|p_h(t)\|^2\right)+\frac{1}{\epsilon_3}\left(\int_0^t\|\partial_t f\|^2ds+\|g(t)\|^2\right)\\
&\quad+\int_0^t\left(\|p_h\|^2+\|\partial_t g\|^2\right)ds+\|K^{-\frac{1}{2}}z_h(0)\|^2+\|p_h(0)\|^2+\|g(0)\|^2.
\end{aligned}\tag{3.3.34}$$

To bound $\|\partial_t u_h\|$ and $\|\partial_t\gamma_h\|$, we use the inf-sup condition (3.3.17) and equation (3.2.8) as before, but now in their time differentiated forms to obtain

$$\|\partial_t u_h\|+\|\partial_t\gamma_h\|\leq C\|\partial_t A^{\frac{1}{2}}\left(\sigma_h+\alpha p_h I\right)\|.\tag{3.3.35}$$

Combining inequalities (3.3.30), (3.3.34), and (3.3.34), and using small enough ϵ_3 give

$$\int_0^t \left(\|\partial_t A^{\frac{1}{2}} \left(\sigma_h + \alpha p_h I\right)\|^2 + c_0 \|\partial_t p_h\|^2 + \|\partial_t u_h\|^2 + \|\partial_t \gamma_h\|^2 \right) ds + \|K^{-\frac{1}{2}} z_h(t)\|^2 + \|p_h(t)\|^2$$
$$\leq C \left(\int_0^t \left(\|\partial_t f\|^2 + \|p_h\|^2 + \|\partial_t g\|^2 \right) ds + \|g(t)\|^2 + \|p_h(0)\|^2 + \|g(0)\|^2 + \|z_h(0)\|^2 \right). \tag{3.3.36}$$

Integrating inequality (3.3.33) with respect to time from 0 to $t \in (0, T]$ and combining the resulting integral inequality with inequalities (3.3.30) and (3.3.36) give

$$\|p_h(t)\|^2 + \|K^{-\frac{1}{2}} z_h(t)\|^2 + \sum_{i=1}^N \int_0^t \|\text{div } z_h\|_{\Omega_i}^2 ds \leq C \Big(\int_0^t \left(\|f(s)\|^2 + \|\partial_t f(s)\|^2 + \|g(s)\|^2 \right.$$
$$\left. + \|\partial_t q(s)\|^2 \right) ds + \|f(t)\|^2 + \|g(t)\|^2 + \|\sigma_h(0)\|^2 + \|p_h(0)\|^2 + \|z_h(0)\|^2 + \|f(0)\|^2 + \|g(0)\|^2 \Big).$$

Also, the coercivity of A and K given in inequalities (2.3.31) and (2.3.32), respectively gives

$$\|z_h\| \leq C \|K^{-\frac{1}{2}} z_h\|, \tag{3.3.37}$$
$$\|\sigma_h\| \leq C \left(\|A^{\frac{1}{2}} \left(\sigma_h + \alpha p_h I\right)(t)\| + \|p_h\| \right). \tag{3.3.38}$$

Next, we give a bound for $\|\lambda_H^u\|$. Combining (3.3.8) and (3.2.8) gives

$$\|\lambda_H^u\|_\Gamma \leq C \sup_{0 \neq \tau \in \mathbb{X}_h} \frac{\sum_{i=1}^N \langle \tau n_i, \lambda_H^u \rangle_{\Gamma_i}}{\|\tau\|_{\mathbb{X}_h}}$$
$$= C \sup_{0 \neq \tau \in \mathbb{X}_h} \frac{1}{\|\tau\|_{\mathbb{X}_h}} \left(\left(A \left(\sigma_h + \alpha p_h I\right), \tau\right) + \sum_{i=1}^N \left(u_h, \text{div } \tau\right)_{\Omega_i} + \left(\gamma_h, \tau\right) \right)$$
$$\leq C \left(\|A^{\frac{1}{2}} \left(\sigma_h + \alpha p_h I\right)\| + \|u_h\| + \|\gamma_h\| \right).$$

Squaring the above inequality and integrating from time 0 to t, we get the following set of inequalities

$$\|\lambda_H^u\|_\Gamma^2 \leq C \left(\|A^{\frac{1}{2}} \left(\sigma_h + \alpha p_h I\right)\|^2 + \|u_h\|^2 + \|\gamma_h\|^2 \right), \tag{3.3.39}$$
$$\int_0^t \|\lambda_H^u(s)\|_\Gamma^2 ds \leq C \int_0^t \left(\|A^{\frac{1}{2}} \left(\sigma_h + \alpha p_h I\right)\|^2 + \|u_h\|^2 + \|\gamma_h\|^2 \right) ds. \tag{3.3.40}$$

Similarly, we can bound $\|\lambda_H^p\|$ combining (3.3.2) and (3.2.11) to obtain

$$\begin{aligned}\|\lambda_H^p\|_\Gamma &\leq C \sup_{0\neq\zeta\in Z_h} \frac{\sum_{i=1}^N \langle \zeta\cdot n_i, \lambda_H^p\rangle_{\Gamma_i}}{\|\zeta\|_{Z_h}} \\ &= C \sup_{0\neq\zeta\in Z_h} \frac{1}{\|\zeta\|_{Z_h}} \left(-\left(K^{-1}z_h,\zeta\right) + (p_h, \operatorname{div}\zeta)\right) \leq C\left(\|K^{-\frac{1}{2}}z_h\| + \|p_h\|\right),\end{aligned}$$

and using the above inequality to obtain

$$\|\lambda_H^p\|_\Gamma^2 \leq C\left(\|K^{-\frac{1}{2}}z_h\|^2 + \|p_h\|^2\right), \tag{3.3.41}$$

$$\int_0^t \|\lambda_H^p\|_\Gamma^2 ds \leq C\int_0^t \left(\|K^{-\frac{1}{2}}z_h\|^2 + \|p_h\|^2\right) ds. \tag{3.3.42}$$

In order to bound the initial data, $\sigma_h(0)$, $u_h(0)$, $z_h(0)$, and $p_h(0)$, note that we obtain the discrete initial data by taking elliptic projection of the continuous initial data $(\sigma_0, u_0, \gamma_0, z_0, p_0)$ to the continuous problem (2.2.3)−(2.2.7), see (3.3.24). Further note that the continuous initial data is constructed using the original pressure initial data $p_0 \in H^1(\Omega)$ using the procedure mentioned in Section 3.2. Following the arguments used in the proof so far for the steady-state version with $t = 0$ gives

$$\begin{aligned}&\|\sigma_h(0)\| + \|u_h(0)\| + \|\gamma_h(0)\| + \|z_h(0)\| + \|p_h(0)\| \leq C\left(\|\sigma_0\| + \|u_0\| + \|\gamma_0\| + \|z_0\| + \|p_0\|\right) \\ &\qquad \leq C\left(\|p_0\|_{H^1(\Omega)} + \|K\nabla p_0\|_{H(\operatorname{div};\Omega)}\right).\end{aligned} \tag{3.3.43}$$

Finally, we combine inequalities (3.3.32), (3.3.37)−(3.3.43) along with the fact that all the results derived so far hold for a general $t \in (0, T]$, to arrive at the stability bound in the theorem. □

3.3.4 Error analysis

In this subsection, we will establish a combined a priori error estimate for all the unknowns in the formulation.

Theorem 3.3.9. *Let* $(\sigma_h(t), u_h(t), \gamma_h(t), z_h(t), p_h(t), \lambda_H) \in \mathbb{X}_h \times V_h \times \mathbb{Q}_h \times Z_h \times W_h \times \Lambda_H$ *be the solution to the system of equations* (3.2.8)−(3.2.14) *under the assumption* (3.3.1) *for* $t \in [0, T]$, *and suppose the solution of* (2.2.3)−(2.2.7) *is sufficiently smooth, then there exists a positive constant* C, *independent of* h, H *and* c_0 *such that the following holds:*

$$
\begin{aligned}
&\|\sigma - \sigma_h\|_{L^\infty(0,T;\mathbb{X}_h)} + \|u - u_h\|_{L^\infty(0,T;L^2(\Omega))} + \|\gamma - \gamma_h\|_{L^\infty(0,T;L^2(\Omega))} + \|z - z_h\|_{L^\infty(0,T;L^2(\Omega))} \\
&+ \|p - p_h\|_{L^\infty(0,T;L^2(\Omega))} + \|u - \lambda_H^u\|_{L^\infty(0,T;L^2(\Gamma))} + \|p - \lambda_H^p\|_{L^\infty(0,T;L^2(\Gamma))} + \|\sigma - \sigma_h\|_{L^2(0,T;\mathbb{X}_h)} \\
&+ \|u - u_h\|_{L^2(0,T;L^2(\Omega))} + \|\gamma - \gamma_h\|_{L^2(0,T;L^2(\Omega))} + \|z - z_h\|_{L^2(0,T;Z_h)} + \|p - p_h\|_{L^2(0,T;L^2(\Omega))} \\
&+ \|u - \lambda_H^u\|_{L^2(0,T;L^2(\Gamma))} + \|u - \lambda_H^p\|_{L^2(0,T;L^2(\Gamma))} \le C\Big(h^{k_1}\|\sigma\|_{H^1\left(0,T;H^{k_1}(\Omega)\right)} \\
&\quad + h^{k_2}H^{\frac{1}{2}}\|\sigma\|_{H^1\left(0,T;H^{k_2+\frac{1}{2}}(\Omega)\right)} + h^{l_1}\|\operatorname{div}\sigma\|_{L^\infty\left(0,T;H^{l_1}(\Omega)\right)} + h^{l_2}\|\operatorname{div}\sigma\|_{L^2\left(0,T;H^{l_2}(\Omega)\right)} \\
&\quad + h^{l_3}\|u\|_{L^2\left(0,T;H^{l_3}(\Omega)\right)} + h^{l_4}\|u\|_{L^\infty\left(0,T;H^{l_4}(\Omega)\right)} + h^{j_1}\|\gamma\|_{H^1\left(0,T;H^{j_1}(\Omega)\right)} \\
&\quad + h^{r_1}\|z\|_{H^1(0,T;H^{r_1}(\Omega))} + h^{r_2}H^{\frac{1}{2}}\|z\|_{H^1\left(0,T;H^{r_2+\frac{1}{2}}(\Omega)\right)} + h^{s_1}\|\operatorname{div} z\|_{L^2(0,T;H^{s_1}(\Omega))} \\
&\quad + h^{s_2}\|p\|_{H^1(0,T;H^{s_2}(\Omega))} + H^{m_1-\frac{1}{2}}\|u\|_{H^2\left(0,T;H^{m_1+\frac{1}{2}}(\Omega)\right)} + H^{m_2-\frac{1}{2}}\|p\|_{H^1\left(0,T;H^{m_2+\frac{1}{2}}(\Omega)\right)}\Big),
\end{aligned}
$$

$$
1 \le k_1 \le k+1, \quad 0 \le k_2 \le k+1, \quad 0 \le l_1, l_2, l_3, l_4 \le l+1, \quad 0 \le j_1 \le j+1,
$$

$$
1 \le r_1, r_2 \le r+1, \quad 0 \le s_1, s_2 \le s+1, \quad 0 \le m_1, m_2 \le m+1.
$$

Proof. First, note that the solution to (2.2.3)−(2.2.7) satisfies for $1 \le i \le N$,

$$
(A(\sigma + \alpha p I), \tau)_{\Omega_i} + (u, \operatorname{div}\tau)_{\Omega_i} + (\gamma, \tau)_{\Omega_i} - \langle u, \tau\, n_i\rangle_{\Gamma_i} = 0, \qquad \forall \tau \in \mathbb{X}_i, \tag{3.3.44}
$$

$$
(\operatorname{div}\sigma, v)_{\Omega_i} = -(f, v)_{\Omega_i}, \qquad \forall v \in V_i, \tag{3.3.45}
$$

$$
(\sigma, \xi)_{\Omega_i} = 0, \qquad \forall \xi \in \mathbb{Q}_i, \tag{3.3.46}
$$

$$
\left(K^{-1}z, \zeta\right)_{\Omega_i} - (p, \operatorname{div}\zeta)_{\Omega_i} + \langle p, \zeta \cdot n_i\rangle_{\Gamma_i} = 0, \qquad \forall \zeta \in Z_i, \tag{3.3.47}
$$

$$
c_0\,(\partial_t p, w)_{\Omega_i} + \alpha\,(\partial_t A(\sigma + \alpha p I), wI)_{\Omega_i} + (\operatorname{div} z, w)_{\Omega_i} = (g, w)_{\Omega_i}, \qquad \forall w \in W_i, \tag{3.3.48}
$$

Subtracting (3.2.41)−(3.2.45) from (3.3.44)−(3.3.48) gives

$$\begin{aligned}
&\left(A\left((\sigma-\sigma_h)+\alpha\left(p-p_h\right)I\right),\tau\right)+\sum_{i=1}^{N}\left(\left(u-u_h\right),\operatorname{div}\tau\right)_{\Omega_i}\\
&\qquad+\left(\left(\gamma-\gamma_h\right),\tau\right)=\sum_{i=1}^{N}\langle u,\tau n_i\rangle_{\Gamma_i}, && \forall\tau\in\mathbb{X}_{h,0}, && (3.3.49)\\
&\sum_{i=1}^{N}\left(\operatorname{div}\left(\sigma-\sigma_h\right),v\right)_{\Omega_i}=0, && \forall v\in V_h, && (3.3.50)\\
&\left(\left(\sigma-\sigma_h\right),\xi\right)=0, && \forall\xi\in\mathbb{Q}_h, && (3.3.51)\\
&\left(K^{-1}\left(z-z_h\right),\zeta\right)-\sum_{i=1}^{N}\left(\left(p-p_h\right),\operatorname{div}\zeta\right)_{\Omega_i}=-\sum_{i=1}^{N}\langle p,\zeta\cdot n_i\rangle_{\Gamma_i}, && \forall\zeta\in Z_{h,0}, && (3.3.52)\\
&c_0\left(\partial_t(p-p_h),w\right)+\alpha\left(\partial_t A\left((\sigma-\sigma_h)+\alpha\left(p-p_h\right)I\right),wI\right)\\
&+\sum_{i=1}^{N}\left(\operatorname{div}\left(z-z_h\right),w\right)_{\Omega_i}=0, && \forall w\in W_h. && (3.3.53)
\end{aligned}$$

Next, rewrite the above error equations in terms of the approximation errors $\psi_\star$ and discretization errors $\phi_\star$, for $\star\in\{\sigma,u,\gamma,z,p\}$ as follows:

$$\begin{aligned}
\sigma-\sigma_h&=(\sigma-\Pi_0^\sigma\sigma)+(\Pi_0^\sigma\sigma-\sigma_h):=\psi_\sigma+\phi_\sigma,\\
u-u_h&=(u-\mathcal{P}_h^u u)+(\mathcal{P}_h^u u-u_h):=\psi_u+\phi_u,\\
\gamma-\gamma_h&=(\gamma-\mathcal{R}_h\gamma)+(\mathcal{R}_h\gamma-\gamma_h):=\psi_\gamma+\phi_\gamma,\\
z-z_h&=(z-\Pi_0^z z)+(\Pi_0^z z-z_h):=\psi_z+\phi_z,\\
p-p_h&=(p-\mathcal{P}_h^p p)+(\mathcal{P}_h^p p-p_h):=\psi_p+\phi_p,\\
u-\lambda_H^u&=(u-\mathcal{Q}_h^u u)+(\mathcal{Q}_h^u u-\lambda_H^u):=\psi_{\lambda^u}+\phi_{\lambda^u},\\
p-\lambda_H^p&=(p-\mathcal{Q}_h^p p)+(\mathcal{Q}_h^p p-\lambda_H^p):=\psi_{\lambda^p}+\phi_{\lambda^p}.
\end{aligned}$$

Note that combining equations (3.3.50) an (3.3.9) gives

$$\operatorname{div}\phi_\sigma=0,\ \text{ in }\Omega_i, \tag{3.3.54}$$

and (3.3.51) combined with (3.3.10) gives

$$(\phi_\sigma,\xi)=0,\ \text{ for }\xi\in\mathbb{Q}_h. \tag{3.3.55}$$

We rewrite error equation (3.3.49) as

$$\left(A\left(\phi_\sigma+\alpha\phi_p I\right),\tau\right)+\sum_{i=1}^{N}\left(\phi_u,\operatorname{div}\tau\right)_{\Omega_i}+\left(\phi_\gamma,\tau\right)=-\left(A\left(\psi_\sigma+\alpha\psi_p I\right),\tau\right)-\left(\psi_\gamma,\tau\right)$$
$$+\sum_{i=1}^{N}\langle u-\mathcal{I}_H^{C,u}u,\tau n_i\rangle_{\Gamma_i},\tag{3.3.56}$$

where we have used

$$\sum_{i=1}^{N}\langle\mathcal{I}_H^{C,u}u,\tau n_i\rangle_{\Gamma_i}=0,\tag{3.3.57}$$

for any $\tau\in\mathbb{X}_{h,0}$. Differentiating the above equation with respect to time, t gives

$$\left(\partial_t A\left(\phi_\sigma+\alpha\phi_p I\right),\tau\right)+\sum_{i=1}^{N}\left(\partial_t\phi_u,\operatorname{div}\tau\right)_{\Omega_i}+\left(\phi_\gamma,\tau\right)$$
$$=-\left(\partial_t A\left(\psi_\sigma+\alpha\psi_p I\right),\tau\right)-\left(\partial_t\psi_\gamma,\tau\right)+\sum_{i=1}^{N}\langle\partial_t\left(u-\mathcal{I}_H^{C,u}u\right),\tau n_i\rangle_{\Gamma_i}.\tag{3.3.58}$$

Taking $\tau=\phi_\sigma$ in (3.3.58) and using (3.3.54) and (3.3.55) gives

$$\left(\partial_t A\left(\phi_\sigma+\alpha\phi_p I\right),\phi_\sigma\right)=-\left(\partial_t A\left(\psi_\sigma+\alpha\psi_p I\right),\phi_\sigma\right)-\left(\partial_t\psi_\gamma,\phi_\sigma\right)+\sum_{i=1}^{N}\left(\partial_t\left(u-\mathcal{I}_H^{C,u}u\right),\phi_\sigma n_i\right)_{\Gamma_i}.\tag{3.3.59}$$

Error equation (3.3.53) can be written as

$$c_0\left(\partial_t\phi_p,w\right)+\alpha\left(\partial_t A\left(\phi_\sigma+\alpha\phi_p I\right),wI\right)+\sum_{i=1}^{N}\left(\operatorname{div}\phi_z,w\right)_{\Omega_i}=-c_0\left(\partial_t\psi_p,w\right)$$
$$-\alpha\left(\partial_t A\left(\psi_\sigma+\alpha\psi_p I\right),wI\right)-\sum_{i=1}^{N}\left(\operatorname{div}\psi_z,w\right)_{\Omega_i}.$$

Using the definition of the L^2 projection $\mathcal{P}_h^p$ and Π_0^z (see (3.3.13)), we can further simplify the above equation to

$$c_0\left(\partial_t\phi_p,w\right)+\alpha\left(\partial_t A\left(\phi_\sigma+\alpha\phi_p I\right),wI\right)+\sum_{i=1}^{N}\left(\operatorname{div}\phi_z,w\right)_{\Omega_i}=-\alpha\left(\partial_t A\left(\psi_\sigma+\alpha\psi_p I\right),wI\right)$$
$$-\sum_{i=1}^{N}\left(\operatorname{div}\phi_z,w\right)_{\Omega_i}.\tag{3.3.60}$$

Taking $w = \phi_p$ in (3.3.60) and combining the resulting equation with (3.3.59) give

$$\frac{1}{2}\partial_t \left(\|A^{\frac{1}{2}} \left(\phi_\sigma + \alpha\phi_p I\right)\|^2 + c_0\|\phi_p\|^2 \right) + \sum_{i=1}^{N} \left(\operatorname{div}\phi_z, \phi_p\right)_{\Omega_i} = -\left(\partial_t A \left(\psi_\sigma + \alpha\psi_p I\right), \phi_\sigma + \alpha\phi_p I\right)$$
$$- \left(\partial_t\psi_\gamma, \phi_\sigma\right) + \sum_{i=1}^{N} \langle \partial_t \left(u - \mathcal{I}_H^{C,u} u \right), \phi_\sigma n_i \rangle_{\Gamma_i}. \tag{3.3.61}$$

Finally error equation (3.3.52) can be written as

$$\left(K^{-1}\phi_z, \zeta\right) - \sum_{i=1}^{N} \left(\phi_p, \operatorname{div}\zeta\right)_{\Omega_i} = -\left(K^{-1}\psi_z, \zeta\right) + \sum_{i=1}^{N} \langle \mathcal{I}_H^{C,p} p - p, \zeta \cdot n_i \rangle_{\Gamma_i}, \tag{3.3.62}$$

where we have used for $\zeta \in Z_{h,0}$,

$$\sum_{i=1}^{N} \langle \mathcal{I}_H^{C,p} p, \zeta \cdot n_i \rangle_{\Gamma_i} = 0, \tag{3.3.63}$$

and the definition of the $L^2(\Omega)$-projection, $\mathcal{P}_h^p$ onto space W_h.

Taking test function $\zeta = \phi_z$ in equation (3.3.62) and combining the resulting equation with (3.3.61) gives

$$\frac{1}{2}\partial_t \left(\|A^{\frac{1}{2}} \left(\phi_\sigma + \alpha\phi_p I\right)\|^2 + c_0\|\phi_p\|^2 \right) + \|K^{-\frac{1}{2}}\phi_z\|^2 = -\left(\partial_t A \left(\psi_\sigma + \alpha\psi_p I\right), \phi_\sigma + \alpha\phi_p I\right)$$
$$- \left(\partial_t\psi_\gamma, \phi_\sigma\right) - \left(K^{-1}\psi_z, \phi_z\right) + \left(\psi_\sigma, \partial_t\phi_\gamma\right) - \sum_{i=1}^{N} \langle \partial_t \left(\mathcal{I}_H^{C,u} u - u \right), \phi_\sigma n_i \rangle_{\Gamma_i}$$
$$+ \sum_{i=1}^{N} \langle \mathcal{I}_H^{C,p} p - p, \phi_z \cdot n_i \rangle_{\Gamma_i}, \tag{3.3.64}$$

where we have used (3.3.55) to conclude that $\left(\phi_\sigma, \partial_t\phi_\gamma\right) = 0$.

Next, we bound the first three terms on the right hand side of (3.3.64).

$$-\left(\partial_t A \left(\psi_\sigma + \alpha\psi_p I\right), \phi_\sigma + \alpha\phi_p I\right) - \left(\partial_t\psi_\gamma, \phi_\sigma\right) - \left(K^{-1}\psi_z, \phi_z\right)$$
$$\leq \|\partial_t A \left(\psi_\sigma + \alpha\psi_p I\right)\| \|\phi_\sigma + \alpha\phi_p I\| + \|\partial_t\psi_\gamma\| \|\phi_\sigma\| + \|K^{-1}\psi_z\| \|\phi_z\|$$
$$\leq \frac{C}{\epsilon_1} \left(\|\partial_t\psi_\sigma\|^2 + \|\partial_t\psi_p\|^2 + \|\partial_t\psi_\gamma\|^2 + \|\psi_z\|^2 \right) + \epsilon_1 \left(\|\phi_\sigma\|^2 + \|\phi_p\|^2 + \|\phi_z\|^2 \right), \tag{3.3.65}$$

where we have used the operator bounds (2.3.31) and (2.3.32) along with Young's inequality for a positive constant $\epsilon_1 > 0$.

Next, we give a bound on the last two boundary terms in the right hand side of equation (3.3.64). For this, we note that the following bounds hold for any $(\tau, v) \in \mathbb{X}_{h,0} \times V$ and $(\zeta, w) \in Z_h \times W$,

$$\begin{aligned}\langle \mathcal{I}_H^{C,u} v - v, \tau n_i \rangle_{\Gamma_i} = \langle E_i \left(\mathcal{I}_H^{C,u} v - v \right), \tau n_i \rangle_{\partial\Omega_i} &\leq C \| E_i \left(\mathcal{I}_H^{C,u} v - v \right) \|_{\frac{1}{2},\partial\Omega_i} \| \tau \|_{H(\mathrm{div};\Omega_i)} \\ &\leq C \| \mathcal{I}_H^{C,u} v - v \|_{\frac{1}{2},\Gamma_i} \| \tau \|_{H(\mathrm{div};\Omega_i)}, \qquad (3.3.66) \\ \langle \mathcal{I}_H^{C,p} w - w, \zeta \cdot n_i \rangle_{\Gamma_i} = \langle E_i \left(\mathcal{I}_H^{C,p} \zeta - \zeta \right), \zeta \cdot n_i \rangle_{\partial\Omega_i} &\leq C \| E_i \left(\mathcal{I}_H^{C,p} \zeta - \zeta \right) \|_{\frac{1}{2},\partial\Omega_i} \| \zeta \|_{H(\mathrm{div};\Omega_i)} \\ &\leq C \| \mathcal{I}_H^{C,p} \zeta - \zeta \|_{\frac{1}{2},\Gamma_i} \| \zeta \|_{H(\mathrm{div};\Omega_i)}, \qquad (3.3.67)\end{aligned}$$

where $E_i \left(\mathcal{I}_H^{C,u} v - v \right)$ and $E_i \left(\mathcal{I}_H^{C,p} w - w \right)$ denote the continuous extension by zero to the entire subdomain boundary $\partial\Omega_i$ and we have used the trace inequality (3.2.40) in order to get the bounds on the right hand side of the above inequalities.

Combining inequalities (3.3.64)−(3.3.65) and taking $(\tau, v) = (\phi_\sigma, \partial_t u)$ and $(\zeta, w) = (\phi_z, p)$ in (3.3.66) and (3.3.67), respectively and integrating with respect to time from 0 to $t \in (0, T]$ gives

$$\begin{aligned}&\| A^{\frac{1}{2}} \left(\phi_\sigma + \alpha \phi_p I \right)(t) \|^2 + c_0 \| \phi_p(t) \|^2 + \int_0^t \| K^{-1} \phi_z \|^2 \leq \\ &C \int_0^t \left(\| \partial_t \psi_\sigma \|^2 + \| \partial_t \psi_p \|^2 + \| \partial_t \psi_\gamma \|^2 + \| \psi_z \|^2 + \sum_{i=1}^N \| \mathcal{I}_H^{C,u} \partial_t u - \partial_t u \|^2_{\frac{1}{2},\Gamma_i} \right. \\ &\left. + \sum_{i=1}^N \| \mathcal{I}_H^{C,p} p - p \|^2_{\frac{1}{2},\Gamma_i} \right) ds + \epsilon_1 \int_0^t \left(\| \phi_\sigma \|^2 + \| \phi_p \|^2 + \| \phi_z \|^2 \right) ds + C \int_0^t \sum_{i=1}^N \| \mathrm{div}\, \phi_z \|^2_{\Omega_i} ds \\ &+ \left| \int_0^t \left(\psi_\sigma, \partial_t \phi_\gamma \right) ds \right| + \| A^{\frac{1}{2}} \left(\phi_\sigma + \alpha \phi_p I \right)(0) \|^2 + c_0 \| \phi_p(0) \|^2, \qquad (3.3.68)\end{aligned}$$

where we also used (3.3.54).

To bound the term $\int_0^t \left(\psi_\sigma, \partial_t \phi_\gamma \right) ds$, we use integration by parts as follows:

$$\begin{aligned}&\int_0^t \left(\psi_\sigma, \partial_t \phi_\gamma \right) ds = \left(\psi_\sigma(t), \phi_\gamma(t) \right) - \left(\psi_\sigma(0), \phi_\gamma(0) \right) - \int_0^t \left(\partial_t \psi_\sigma, \phi_\gamma \right) ds \\ &\leq \frac{C}{\epsilon_2} \left(\int_0^t \| \partial_t \psi_\sigma \|^2 ds + \| \psi_\sigma(t) \|^2 \right) + \epsilon_2 \left(\int_0^t \| \phi_\gamma \|^2 ds + \| \phi_\gamma(t) \|^2 \right) + C \left(\| \psi_\sigma(0) \|^2 + \| \phi_\gamma(0) \|^2 \right), \qquad (3.3.69)\end{aligned}$$

where we have used the Cauchy-Schwartz and Young's inequality for $\epsilon_2 > 0$.

Next, we bound the discrete errors of the form $\phi_\star$ for $\star \in \{\sigma, \gamma, u, z, p\}$.

Using the inf-sup condition (3.3.17) and the error equation (3.3.54) gives

$$\begin{aligned}
\|\phi_u\| + \|\phi_\gamma\| &\leq C \sup_{0\neq\tau\in\mathbb{X}_{h,0}} \frac{\sum_{i=1}^{N} (\phi_u, \operatorname{div}\tau)_{\Omega_i} + (\phi_\gamma, \tau)}{\|\tau\|_{\mathbb{X}_h}} \\
&\leq C \sup_{0\neq\tau\in\mathbb{X}_{h,0}} \frac{1}{\|\tau\|_{\mathbb{X}_h}} \Big| \left(A\left(\phi_\sigma + \alpha\phi_p I\right), \tau\right) + \left(A\left(\psi_\sigma + \alpha\psi_p I\right), \tau\right) + \left(\psi_\gamma, \tau\right) \\
&- \sum_{i=1}^{N} \left(\left(\mathcal{I}_H^{C,u} u - u \right), \tau n_i \right)_{\Gamma_i} \Big| \\
&\leq C \left(\|A^{\frac{1}{2}} \left(\phi_\sigma + \alpha\phi_p I\right)\| + \|\psi_\sigma\| + \|\psi_\gamma\| + \|\psi_p\| + \sum_{i=1}^{N} \|\mathcal{I}_H^{C,u} u - u\|_{\frac{1}{2},\Gamma_i} \right), \qquad (3.3.70)
\end{aligned}$$

where we have used Cauchy-Schwartz inequality, (3.3.54) and (3.3.66) with $v = u$ to get last inequality. Integrating (3.3.70) with respect to time from 0 to $t \in (0, T]$ gives

$$\begin{aligned}
\int_0^t \left(\|\phi_u\|^2 + \|\phi_\gamma\|^2 \right) ds &\leq C \int_0^t \left(\|\phi_\sigma\|^2 + \alpha \|\phi_p I\|^2 \right) ds \\
&+ C \int_0^t \left(\|\psi_\sigma\|^2 + \|\psi_\gamma\|^2 + \|\psi_p\|^2 + \sum_{i=1}^{N} \|\mathcal{I}_H^{C,u} u - u\|_{\frac{1}{2},\Gamma_i}^2 \right) ds. \qquad (3.3.71)
\end{aligned}$$

To bound $\|\phi_p\|$, we use the inf-sup stability condition (3.3.18) to get

$$\begin{aligned}
\|\phi_p\| &\leq C \sup_{0\neq\zeta\in Z_{h,0}} \frac{\sum_{i=1}^{N} (\operatorname{div}\zeta, \phi_p)_{\Omega_i}}{\|\zeta\|_{Z_h}} \\
&\leq C \sup_{0\neq\zeta\in Z_{h,0}} \frac{(K^{-1}\phi_z, \zeta) + (K^{-1}\psi_z, \zeta) - \sum_{i=1}^{N} \langle \mathcal{I}_H^{C,p} p - p, \zeta \cdot n_i \rangle_{\Gamma_i}}{||\zeta||_{Z_h}} \\
&\leq C \left(\|\psi_z\| + \|K^{-\frac{1}{2}}\phi_z\| + \sum_{i=1}^{N} \|\mathcal{I}_H^{C,p} p - p\|_{\frac{1}{2},\Gamma_i} \right), \qquad (3.3.72)
\end{aligned}$$

where we have used (3.3.62) and (3.3.63) to obtain second inequality and (3.3.67) with $w = p$ to obtain the last one. Integrating (3.3.72) in time from 0 to $t \in (0, T]$ yields

$$\int_0^t \|\phi_p\|^2 ds \leq C \int_0^t \left(\|\psi_z\|^2 + \|K^{-\frac{1}{2}}\phi_z\|^2 + \sum_{i=1}^{N} \|\mathcal{I}_H^{C,p} p - p\|_{\frac{1}{2},\Gamma_i}^2 \right) ds. \qquad (3.3.73)$$

To bound the term $\int_0^t \|\phi_\sigma\|^2 ds$, we take $\tau = \phi_\sigma$ in (3.3.56) and $\xi = \phi_\gamma$ in (3.3.51), and use (3.3.54)−(3.3.55) to get, for $\epsilon_3 > 0$,

$$\|A^{\frac{1}{2}}\phi_\sigma\|^2 = -\left(A^{\frac{1}{2}}\alpha\phi_p I, \phi_\sigma\right) - \left(A\left(\psi_\sigma + \alpha\psi_p I\right), \phi_\sigma\right) - (\psi_\gamma, \phi_\sigma) - \sum_{i=1}^{N}\langle \mathcal{I}_H^{C,u}u - u, \phi_\sigma n_i\rangle_{\Gamma_i}$$

$$+ (\psi_\sigma, \phi_\gamma) \leq C\Big(\big(\|\phi_p\| + \|\psi_\sigma\| + \|\psi_p\| + \|\psi_\gamma\|\big)\|\phi_\sigma\| + \sum_{i=1}^{N}\|\mathcal{I}_H^{C,u}u - u\|_{\frac{1}{2},\Gamma_i}\|\phi_\sigma\|_{H(\mathrm{div};\Omega_i)}$$

$$+ \|\psi_\sigma\|\|\phi_\gamma\|\Big) \leq \frac{C}{\epsilon_3}\left(\|\psi_\sigma\|^2 + \|\psi_p\|^2 + \|\psi_\gamma\|^2 + \sum_{i=1}^{N}\|\mathcal{I}_H^{C,u}u - u\|^2_{\frac{1}{2},\Gamma_i} + \|\phi_p\|^2\right)$$

$$+ \epsilon_3\left(\|\phi_\sigma\|^2 + \|\phi_\gamma\|^2\right), \tag{3.3.74}$$

where we have used (3.3.66) with $(\tau, v) = (\phi_\sigma, u)$ to arrive at the first inequality and (2.3.31) with Young's inequality to justify the last inequality. Now again using (2.3.31), integrating (3.3.74) with respect to time from 0 to $t \in (0, T]$, and taking ϵ_3 small enough, we get

$$\int_0^t \|\phi_\sigma\|^2 ds \leq C\Big(\int_0^t \big(\|\psi_\sigma\|^2 + \|\psi_p\|^2 + \|\psi_\gamma\|^2 + \sum_{i=1}^{N}\|\mathcal{I}_H^{C,u}u - u\|^2_{\frac{1}{2},\Gamma_i} + \|\phi_p\|^2\big)ds\Big)$$

$$+ \epsilon_3\int_0^t \|\phi_\gamma\|^2 ds. \tag{3.3.75}$$

Combining (3.3.68)−(3.3.75), using (3.3.54), and taking ϵ_1, ϵ_2, and ϵ_3 small enough gives

$$\|A^{\frac{1}{2}}\left(\phi_\sigma + \alpha\phi_p I\right)\|^2 + \|\phi_u\|^2 + \|\phi_\gamma\|^2 + \|c_0^{\frac{1}{2}}\phi_p\|^2 + \|\operatorname{div}\phi_\sigma\|^2$$

$$+ \int_0^t \left(\|\phi_\sigma\|^2 + \|\phi_u\|^2 + \|\phi_\gamma\|^2 + \|K^{-\frac{1}{2}}\phi_z\|^2 + \|\phi_p\|^2 + \|\operatorname{div}\phi_\sigma\|^2\right)$$

$$\leq C\int_0^t \left(\|\partial_t\psi_\sigma\|^2 + \|\partial_t\psi_p\|^2 + \|\partial_t\psi_\gamma\|^2 + \|\psi_\sigma\|^2 + \|\psi_p\|^2 + \|\psi_\gamma\|^2 + \|\psi_z\|^2\right)ds + C(\|\psi_\sigma\|^2$$

$$+ \|\psi_p\|^2 + \|\psi_\gamma\|^2) + C\int_0^t \sum_{i=1}^{N}\|\operatorname{div}\phi_z\|^2_{\Omega_i} ds + C\sum_{i=1}^{N}\left(\mathcal{I}_H^{C,u}u - u(t)\|^2_{\frac{1}{2},\Gamma_i} + \|\mathcal{I}_H^{C,p}p - p(t)\|^2_{\frac{1}{2},\Gamma_i}\right)$$

$$+ C\sum_{i=1}^{N}\int_0^t \left(\|\mathcal{I}_H^{C,u}\partial_t u - \partial_t u\|^2_{\frac{1}{2},\Gamma_i} + \|\mathcal{I}_H^{C,u}u - u\|^2_{\frac{1}{2},\Gamma_i} + \|\mathcal{I}_H^{C,p}p - p\|^2_{\frac{1}{2},\Gamma_i}\right)ds$$

$$+ C\left(\|\phi_\sigma(0)\|^2 + \|\phi_p(0)\|^2 + \|\phi_\gamma(0)\|^2\right). \tag{3.3.76}$$

Bound on $\sum_{i=1}^{N}\|\operatorname{div}\phi_z\|_{\Omega_i}$.

Take $w = \phi_z$ in (3.3.53) to get, for $i = 1, 2, \ldots, N$,

$$\begin{aligned}
\| \operatorname{div} \phi_z \|_{\Omega_i}^2 &= - \left(c_0 \partial_t \phi_p, \operatorname{div} \phi_z \right)_{\Omega_i} - \left(c_0 \partial_t \phi_p, \operatorname{div} \phi_z \right)_{\Omega_i} - \alpha \left(\partial_t A \left(\phi_\sigma + \alpha \phi_p I \right), \operatorname{div} \phi_z I \right)_{\Omega_i} \\
&- \alpha \left(\partial_t A \left(\psi_\sigma + \alpha \psi_p I \right), \operatorname{div} \phi_z I \right)_{\Omega_i} + \left(\psi_z, \operatorname{div} \phi_z \right)_{\Omega_i} = - \left(c_0 \partial_t \phi_p, \operatorname{div} \phi_z \right)_{\Omega_i} \\
&- \alpha \left(\partial_t A \left(\phi_\sigma + \alpha \phi_p I \right), \operatorname{div} \phi_z I \right)_{\Omega_i} - \alpha \left(\partial_t A \left(\psi_\sigma + \alpha \psi_p I \right), \operatorname{div} \phi_z I \right)_{\Omega_i},
\end{aligned}$$

where the last equality follows from (3.2.24) and (3.3.13). Finally, summing the above equation over all the subdomain indices and using (2.3.31), we get, for $i = 1, 2, \ldots, N$,

$$\| \operatorname{div} \phi_z \|_{\Omega_i} \leq C \left(\| c_0^{\frac{1}{2}} \partial_t \phi_p \|_{\Omega_i} + \| \partial_t A^{\frac{1}{2}} \left(\phi_\sigma + \alpha \phi_p I \right) \|_{\Omega_i} + \| \psi_p \|_{\Omega_i} + \| \psi_\sigma \|_{\Omega_i} \right). \tag{3.3.77}$$

Squaring the above equation and summing over all the subdomain indices give

$$\sum_{i=1}^{N} \| \operatorname{div} \phi_z \|_{\Omega_i}^2 \leq C \left(\| c_0^{\frac{1}{2}} \partial_t \phi_p \|^2 + \| \partial_t A^{\frac{1}{2}} \left(\phi_\sigma + \alpha \phi_p I \right) \|^2 + \| \psi_p \|^2 + \| \psi_\sigma \|^2 \right). \tag{3.3.78}$$

In order to bound $\| c_0^{\frac{1}{2}} \partial_t \phi_p \|^2$ and $\| \partial_t A^{\frac{1}{2}} \left(\phi_\sigma + \alpha \phi_p I \right) \|^2$, we use arguments similar to the ones used in the stability analysis for the method. Differentiate (3.3.54), (3.3.55), and (3.3.62) in time, combine them with (3.3.58), and take test functions $\tau = \partial_t \phi_\sigma$, $\xi = \partial_t \phi_\gamma$, $q = \phi_z$, and $w = \partial_t \phi_p$ to get the following time differentiated version of (3.3.64):

$$\begin{aligned}
\| \partial_t A^{\frac{1}{2}} \left(\phi_\sigma + \alpha \phi_p I \right) \|^2 + \| c_0^{\frac{1}{2}} \partial_t \phi_p \|^2 + \frac{1}{2} \partial_t \| K^{-\frac{1}{2}} \phi_z \|^2 &= - \left(\partial_t A \left(\psi_\sigma + \alpha \psi_p I \right), \partial_t \left(\phi_\sigma + \alpha \phi_p I \right) \right) \\
- \left(\partial_t \psi_\gamma, \partial_t \left(\phi_\sigma + \alpha \phi_p I \right) \right) - \left(\partial_t K^{-1} \psi_z, \phi_z \right) + \left(\partial_t \psi_\sigma, \partial_t \phi_\gamma \right) - \sum_{i=1}^{N} \langle \partial_t \left(\mathcal{I}_H^{C,u} u - u \right), \partial_t \phi_\sigma n_i \rangle_{\Gamma_i} \\
+ \sum_{i=1}^{N} \langle \mathcal{I}_H^{C,p} \partial_t p - \partial_t p, \phi_z \cdot n_i \rangle_{\Gamma_i},
\end{aligned} \tag{3.3.79}$$

where we have used the fact that $\left(\partial_t \psi_\gamma, \partial_t \alpha \phi_p I \right) = 0$ to write

$$\left(\partial_t \psi_\gamma, \partial_t \phi_\sigma \right) = \left(\partial_t \psi_\gamma, \partial_t \left(\phi_\sigma + \alpha \phi_p I \right) \right).$$

To bound $\sum_{i=1}^{N}\langle\partial_t\left(\mathcal{I}_H^{C,u}u-u\right),\partial_t\phi_\sigma n_i\rangle_{\Gamma_i}$, we use integration by parts to rewrite

$$\sum_{i=1}^{N}\langle\partial_t\left(\mathcal{I}_H^{C,u}u-u\right),\partial_t\phi_\sigma n_i\rangle_{\Gamma_i}=\frac{\partial}{\partial t}\left(\sum_{i=1}^{N}\langle\partial_t\left(\mathcal{I}_H^{C,u}u-u\right),\phi_\sigma n_i\rangle_{\Gamma_i}\right)$$
$$-\sum_{i=1}^{N}\langle\partial_t^2\left(\mathcal{I}_H^{C,u}u-u\right),\phi_\sigma n_i\rangle_{\Gamma_i}. \tag{3.3.80}$$

To bound the last term on the right hand side of the above equation, we take $(\tau,v)=(\phi_\sigma,\partial_t^2u)$ in (3.3.66) to get

$$\Big|\sum_{i=1}^{N}\langle\partial_t^2\left(\mathcal{I}_H^{C,u}u-u\right),\phi_\sigma n_i\rangle_{\Gamma_i}\Big|\leq C\sum_{i=1}^{N}\|\mathcal{I}_H^{C,u}\partial_t^2u-\partial_t^2u\|_{\frac{1}{2},\Gamma_i}\|\phi_\sigma\|_{H(\mathrm{div};\Omega_i)}. \tag{3.3.81}$$

To bound $\sum_{i=1}^{N}\langle\mathcal{I}_H^{C,p}\partial_tp-\partial_tp,\partial_t\phi_z\cdot n_i\rangle_{\Gamma_i}$, we take $(\zeta,w)=(\phi_z,\partial_tp)$ in (3.3.67), to get

$$\Big|\sum_{i=1}^{N}\langle\mathcal{I}_H^{C,p}\partial_tp-\partial_tp,\phi_z\cdot n_i\rangle_{\Gamma_i}\Big|\leq C\sum_{i=1}^{N}\|\mathcal{I}_H^{C,p}\partial_tp-\partial_tp\|_{\frac{1}{2},\Gamma_i}\|\phi_z\|_{H(\mathrm{div};\Omega_i)}. \tag{3.3.82}$$

Combining (3.3.79)−(3.3.82), using (2.3.31)−(2.3.32) and integrating with respect to time from 0 to $t\in(0,T]$, we get

$$\|K^{-\frac{1}{2}}\phi_z\|^2+\int_0^t\left(\|\partial_tA^{\frac{1}{2}}\left(\phi_\sigma+\alpha\phi_pI\right)\|^2+\|c_0^{\frac{1}{2}}\partial_t\phi_p\|^2\right)ds$$
$$\leq C\int_0^t(\|\partial_t\psi_\sigma\|^2+\|\partial_t\psi_p\|^2+\|\partial_t\psi_\gamma\|^2+\|\partial_t\psi_z\|^2+\sum_{i=1}^{N}\|\mathcal{I}_H^{C,u}\partial_t^2u-\partial_t^2u\|_{\frac{1}{2},\Gamma_i}^2$$
$$+\sum_{i=1}^{N}\|\mathcal{I}_H^{C,p}\partial_tp-\partial_tp\|_{\frac{1}{2},\Gamma_i}^2)ds+C\sum_{i=1}^{N}\|\mathcal{I}_H^{C,u}\partial_tu-\partial_tu(t)\|_{\frac{1}{2},\Gamma_i}^2$$
$$+\epsilon\left(\int_0^t\left(\|\phi_\sigma\|^2+\|\partial_t\phi_\gamma\|^2+\|\phi_z\|^2+\sum_{i=1}^{N}\|\operatorname{div}\phi_z\|_{\Omega_i}^2\right)ds+\|\phi_\sigma(t)\|^2\right)$$
$$+C\left(\|K^{-\frac{1}{2}}\phi_z(0)\|^2+\|\phi_\sigma(0)\|^2+\sum_{i=1}^{N}\|\mathcal{I}_H^{C,u}\partial_tu-\partial_tu(0)\|_{\frac{1}{2},\Gamma_i}^2\right), \tag{3.3.83}$$

where we have used Young's inequality for $\epsilon>0$ and (3.3.66) with $(\tau,v)=(\phi_\sigma,\partial_tu)$.

Using the inf-sup condition (3.3.17) with $v = \partial_t \phi_u$, $\xi = \partial_t \phi_\gamma$, the time-differentiated (3.3.56), and following the steps similar to the ones used to arrive at (3.3.71), we get

$$\int_0^t \left(\|\partial_t \phi_u\|^2 + \|\partial_t \phi_\gamma\|^2\right) ds \leq C\Big(\int_0^t \|\partial_t A^{\frac{1}{2}} \left(\phi_\sigma + \alpha \phi_p I\right)\|^2 ds \\ + \int_0^t \left(\|\partial_t \psi_\sigma\|^2 + \|\partial_t \psi_\gamma\|^2 + \|\partial_t \psi_p\|^2 + \sum_{i=1}^N \|\mathcal{I}_H^{C,u} \partial_t u - \partial_t u(t)\|^2_{\frac{1}{2},\Gamma_i}\right) ds\Big). \tag{3.3.84}$$

Combining (3.3.83)−(3.3.84), taking ϵ small enough, and using (3.3.72) implies

$$\|K^{-\frac{1}{2}} \phi_z(t)\|^2 + \|\phi_p(t)\|^2 + \int_0^t \left(\|\partial_t A^{\frac{1}{2}} \left(\phi_\sigma + \alpha \phi_p I\right)\|^2 + \|c_0^{\frac{1}{2}} \partial_t \phi_p\|^2\right) ds \\ \leq C \int_0^t (\|\partial_t \psi_\sigma\|^2 + \|\partial_t \psi_p\|^2 + \|\partial_t \psi_\gamma\|^2 + \|\partial_t \psi_z\|^2 + \sum_{i=1}^N \|\mathcal{I}_H^{C,u} \partial_t^2 u - \partial_t^2 u\|^2_{\frac{1}{2},\Gamma_i} \\ + \sum_{i=1}^N \|\mathcal{I}_H^{C,p} \partial_t p - \partial_t p\|^2_{\frac{1}{2},\Omega_i}) ds + C \sum_{i=1}^N (\|\mathcal{I}_H^{C,u} \partial_t u - \partial_t u(t)\|^2_{\frac{1}{2},\Gamma_i} \\ + \|\mathcal{I}_H^{C,p} p - p(t)\|^2_{\frac{1}{2},\Gamma_i}) + \epsilon \left(\int_0^t \left(\|\phi_\sigma\|^2 + \|\phi_z\|^2 + \sum_{i=1}^N \|\operatorname{div} \phi_z\|^2_{\Omega_i} \right) ds \right) \\ + \epsilon \left(\|\phi_\sigma(t)\|^2\right) + C \left(\|\psi_z(t)\|^2 + \|K^{-\frac{1}{2}} \phi_z(0)\|^2 + \|\phi_\sigma(0)\|^2\right) \\ + C \left(\sum_{i=1}^N \|\mathcal{I}_H^{C,u} \partial_t u - \partial_t u(0)\|^2_{\frac{1}{2},\Gamma_i} \right). \tag{3.3.85}$$

Combining (3.3.78) and (3.3.85) gives

$$\|K^{-\frac{1}{2}} \phi_z(t)\|^2 + \|\phi_p(t)\|^2 + \int_0^t \sum_{i=1}^N \|\operatorname{div} \phi_z\|^2_{\Omega_i} ds \\ \leq C \int_0^t (\|\partial_t \psi_\sigma\|^2 + \|\partial_t \psi_p\|^2 + \|\partial_t \psi_\gamma\|^2 + \|\partial_t \psi_z\|^2 + \sum_{i=1}^N \|\mathcal{I}_H^{C,u} \partial_t^2 u - \partial_t^2 u\|^2_{\frac{1}{2},\Gamma_i} \\ + \sum_{i=1}^N \|\mathcal{I}_H^{C,p} \partial_t p - \partial_t p\|^2_{\frac{1}{2},\Gamma_i}) ds + C \sum_{i=1}^N \Big(\|\mathcal{I}_H^{C,u} \partial_t u - \partial_t u\|^2_{\frac{1}{2},\Gamma_i} \\ + \|\mathcal{I}_H^{C,p} p - p(t)\|^2_{\frac{1}{2},\Gamma_i}\Big) + \epsilon_3 \Big(\int_0^t \left(\|\phi_\sigma\|^2 + \|\phi_z\|^2 + \sum_{i=1}^N \|\operatorname{div} \phi_z\|^2_{\Omega_i}\right) ds + \|\phi_\sigma(t)\|^2 \Big) \\ + C\Big(\|\psi_z(t)\|^2 + \|K^{-\frac{1}{2}} \phi_z(0)\|^2 + \|\phi_\sigma(0)\|^2 + \sum_{i=1}^N \|\mathcal{I}_H^{C,u} \partial_t u - \partial_t u(0)\|^2_{\frac{1}{2},\Gamma_i}\Big). \tag{3.3.86}$$

Coercivity of A, (2.3.31), also implies

$$\|\phi_\sigma\| \leq C \left(\|A^{\frac{1}{2}} \left(\phi_\sigma + \alpha \phi_p I\right)\| + \|\phi_p\| \right). \tag{3.3.87}$$

Bound on $\|\lambda_H^u\|_\Gamma$ and $\|\lambda_H^p\|_\Gamma$.

In order to bound $\|\lambda_H^u\|_\Gamma$, we take the difference between equations (2.2.3) and (3.2.8) to get

$$\begin{aligned}&\left(A\left((\sigma-\sigma_h)+\alpha\left(p-p_h\right)I\right),\tau\right)+\sum_{i=1}^{N}\left(\left(u-u_h\right),\operatorname{div}\tau\right)_{\Omega_i}+\left(\left(\gamma-\gamma_h\right),\tau\right)\\&\quad-\sum_{i=1}^{N}\left(u,\tau n_i\right)_{\Gamma_i}=\sum_{i=1}^{N}\langle u-\lambda_H^u,\tau\, n_i\rangle_{\Gamma_i},\qquad\qquad\forall\tau\in\mathbb{X}_h.\end{aligned}$$

We can split the error terms in the above equation and use (3.2.15) to rewrite the above equation as

$$\begin{aligned}&\sum_{i=1}^{N}\langle\phi_{\lambda^u},\tau\, n_i\rangle_{\Gamma_i}=\left(A\left(\phi_\sigma+\alpha\phi_p\right),\tau\right)+\left(A\left(\psi_\sigma+\alpha\psi_p\right),\tau\right)+\sum_{i=1}^{N}\left(\phi_u,\operatorname{div}\tau\right)_{\Omega_i}\\&+\sum_{i=1}^{N}\left(\psi_u,\operatorname{div}\tau\right)_{\Omega_i}+\left(\phi_\gamma,\tau\right)+\left(\psi_\gamma,\tau\right)+\sum_{i=1}^{N}\langle\mathcal{I}_H^{C,u}u-u,\tau n_i\rangle_{\Gamma_i},\qquad\qquad\forall\tau\in\mathbb{X}_h.\end{aligned}\tag{3.3.88}$$

Inf-sup stability bound (3.3.8) combined with (3.3.88) implies

$$\begin{aligned}&\|\phi_{\lambda^u}\|_\Gamma\le C\sup_{0\neq\tau\in\mathbb{X}_h}\frac{\sum_{i=1}^{N}\langle\tau n_i,\phi_{\lambda^u}\rangle_{\Gamma_i}}{\|\tau\|_{\mathbb{X}_h}}=C\sup_{0\neq\tau\in\mathbb{X}_h}\Big(\frac{\left(A\left(\phi_\sigma+\alpha\phi_p\right),\tau\right)+\left(A\left(\psi_\sigma+\alpha\psi_p\right),\tau\right)}{\|\tau\|_{\mathbb{X}_h}}\\&+\frac{\sum_{i=1}^{N}\left(\phi_u,\operatorname{div}\tau\right)_{\Omega_i}+\sum_{i=1}^{N}\left(\psi_u,\operatorname{div}\tau\right)_{\Omega_i}+\left(\phi_\gamma,\tau\right)+\left(\psi_\gamma,\tau\right)+\sum_{i=1}^{N}\langle\mathcal{I}_H^{C,u}u-u,\tau n_i\rangle_{\Gamma_i}}{\|\tau\|_{\mathbb{X}_h}}\Big)\\&\le C\big(\|A^{\frac12}\left(\phi_\sigma+\alpha\phi_pI\right)\|+\|\phi_u\|+\|\phi_\gamma\|\big)+C\Big(\|\psi_\sigma\|+\|\psi_p\|+\|\psi_u\|+\|\psi_\gamma\|\\&\quad+\sum_{i=1}^{N}\|\mathcal{I}_H^{C,u}u-u\|_{\frac12,\Gamma_i}\Big),\end{aligned}$$

where in the last inequality, we have used (2.3.31) and (3.3.66) with $v=u$. Squaring the above inequality and then integrating with respect to time from 0 to t yields the following two bounds

$$\begin{aligned}\|\phi_{\lambda^u}\|_\Gamma^2&\le C\left(\|A^{\frac12}\left(\phi_\sigma+\alpha\phi_pI\right)\|^2+\|\phi_u\|^2+\|\phi_\gamma\|^2\right)\\&\quad+C\left(\|\psi_\sigma\|^2+\|\psi_p\|^2+\|\psi_u\|^2+\|\psi_\gamma\|^2+\sum_{i=1}^{N}\|\mathcal{I}_H^{C,u}u-u\|_{\frac12,\Gamma_i}^2\right),\end{aligned}\tag{3.3.89}$$

$$\begin{aligned}\int_0^t\|\phi_{\lambda^u}\|_\Gamma^2ds&\le C\int_0^t\left(\|A^{\frac12}\left(\phi_\sigma+\alpha\phi_pI\right)\|^2+\|\phi_u\|^2+\|\phi_\gamma\|^2\right)ds\\&\quad+C\int_0^t\left(\|\psi_\sigma\|^2+\|\psi_p\|^2+\|\psi_u\|^2+\|\psi_\gamma\|^2+\sum_{i=1}^{N}\|\mathcal{I}_H^{C,u}u-u\|_{\frac12,\Gamma_i}^2\right)ds.\end{aligned}\tag{3.3.90}$$

Following similar arguments, we can bound $\|\lambda_H^p\|_\Gamma$ as well. To achieve this, take the difference between equations (3.3.47) and (3.2.11) to get the following error equation

$$\begin{aligned}
&\left(K^{-1}\phi_z,\zeta\right)+\left(K^{-1}\psi_z,\zeta\right)-\sum_{i=1}^{N}\left(\phi_p,\operatorname{div}\zeta\right)_{\Omega_i}-\sum_{i=1}^{N}\left(\psi_p,\operatorname{div}\zeta\right)_{\Omega_i}\\
&-\sum_{i=1}^{N}\langle\mathcal{I}_H^{C,p}p-p,\zeta\cdot n_i\rangle_{\Gamma_i}=\sum_{i=1}^{N}-\langle\lambda_H^p,\zeta\cdot n_i\rangle_{\Gamma_i}=\sum_{i=1}^{N}-\langle\phi_{\lambda^p},\zeta\cdot n_i\rangle_{\Gamma_i},\quad\forall\zeta\in Z_h,
\end{aligned}\tag{3.3.91}$$

where the last equality follows from (3.2.16). Inf-sup stability bound (3.3.2) combined with (3.3.91) implies

$$\begin{aligned}
\|\phi_{\lambda^p}\|_\Gamma &\le C\sup_{0\neq\zeta\in Z_h}\frac{\sum_{i=1}^{N}\langle\zeta\cdot n_i,\phi_{\lambda^p}\rangle_{\Gamma_i}}{\|\zeta\|_{Z_h}}\\
&=C\sup_{0\neq\zeta\in Z_h}\Big(\frac{(K^{-1}\phi_z,\zeta)+(K^{-1}\psi_z,\zeta)-\sum_{i=1}^{N}\left(\phi_p,\operatorname{div}\zeta\right)_{\Omega_i}-\sum_{i=1}^{N}\left(\psi_p,\operatorname{div}\zeta\right)_{\Omega_i}}{\|\zeta\|_{Z_h}}\\
&\quad+\frac{-\sum_{i=1}^{N}\langle\mathcal{I}_H^{C,p}p-p,\zeta\cdot n_i\rangle_{\Gamma_i}}{\|\zeta\|_{Z_h}}\Big)\le C\Big(\|K^{-\frac12}\phi_z\|+\|\phi_p\|+\|\psi_z\|+\|\psi_p\|\\
&\qquad+\sum_{i=1}^{N}\|\mathcal{I}_H^{C,p}p-p\|_{\frac12,\Gamma_i}\Big),
\end{aligned}\tag{3.3.92}$$

where we have used (2.3.32) and (3.3.67) with $w=p$. Squaring the above bound and then integrating with respect to time from 0 to t give the following

$$\|\phi_{\lambda^p}\|_\Gamma^2\le C\left(\|K^{-\frac12}\phi_z\|^2+\|\phi_p\|^2\right)+C\left(\|\psi_z\|^2+\|\psi_p\|^2+\sum_{i=1}^{N}\|\mathcal{I}_H^{C,p}p-p\|_{\frac12,\Gamma_i}^2\right),\tag{3.3.93}$$

$$\begin{aligned}
\int_0^t\|\phi_{\lambda^p}\|_\Gamma^2ds&\le C\int_0^t\left(\|K^{-\frac12}\phi_z\|^2+\|\phi_p\|^2\right)ds\\
&\quad+C\int_0^t\left(\|\psi_z\|^2+\|\psi_p\|^2+\sum_{i=1}^{N}\|\mathcal{I}_H^{C,p}p-p\|_{\frac12,\Gamma_i}^2\right)ds.
\end{aligned}\tag{3.3.94}$$

Finally, combining (3.3.76), (3.3.86)−(3.3.87), (3.3.89)−(3.3.90) and (3.3.93)−(3.3.94), and taking ϵ small enough, we get

$$\begin{aligned}
&\|A^{\frac{1}{2}}\left(\phi_\sigma+\alpha\phi_p I\right)(t)\|^2+\|\phi_\sigma(t)\|^2+\|\phi_u(t)\|^2+\|\phi_\gamma(t)\|^2+\|K^{-\frac{1}{2}}\phi_z(t)\|^2+\|\phi_p(t)\|^2\\
&+\|\phi_{\lambda^u}(t)\|_\Gamma^2+\|\phi_{\lambda^p}(t)\|_{L^2(\Gamma}^2+\int_0^t(\|\phi_\sigma\|_{\mathbb{X}_h}^2+\|\phi_u\|^2+\|\phi_\gamma\|^2+\|K^{-\frac{1}{2}}\phi_z\|^2+\sum_{i=1}^N\|\operatorname{div}\phi_z\|_{\Omega_i}^2\\
&+\|\phi_p\|^2+\|\phi_{\lambda^u}\|_\Gamma^2+\|\phi_{\lambda^p}\|_\Gamma^2)ds\le C\int_0^t(\|\partial_t\psi_\sigma\|^2+\|\partial_t\psi_p\|^2+\|\partial_t\psi_\gamma\|^2+\|\partial_t\psi_z\|^2+\|\psi_\sigma\|^2\\
&\quad+\|\psi_p\|^2+\|\psi_\gamma\|^2+\|\psi_z\|^2)ds+C\sum_{i=1}^N\int_0^t(\|\mathcal{I}_H^{C,p}u-u\|_{\frac{1}{2},\Gamma_i}^2+\|\mathcal{I}_H^{C,u}\partial_t u-\partial_t u\|_{\frac{1}{2},\Gamma_i}^2\\
&\quad+\|\mathcal{I}_H^{C,u}\partial_t^2 u-\partial_t^2 u\|_{\frac{1}{2},\Gamma_i}^2+\|\mathcal{I}_H^{C,p}p-p\|_{\frac{1}{2},\Gamma_i}^2+\|\mathcal{I}_H^{C,p}\partial_t p-\partial_t p\|_{\frac{1}{2},\Gamma_i}^2)ds\\
&\quad+C\left(\|\psi_\sigma(t)\|^2+\|\psi_p(t)\|^2+\|\psi_\gamma(t)\|^2+\|\psi_z(t)\|^2\right)\\
&\quad+C\sum_{i=1}^N(\|\mathcal{I}_H^{C,u}u-u(t)\|_{\frac{1}{2},\Gamma_i}^2+\|\mathcal{I}_H^{C,u}\partial_t u-\partial_t u(t)\|_{\frac{1}{2},\Gamma_i}^2+\|\mathcal{I}_H^{C,p}p-p(t)\|_{\frac{1}{2},\Gamma_i}^2)\\
&\quad+C\left(\|\phi_\sigma(0)\|^2+\|\phi_p(0)\|^2+\|\phi_\gamma(0)\|^2+\|\phi_z(0)\|^2+\sum_{i=1}^N\|\mathcal{I}_H^{C,u}\partial_t u-\partial_t u(0)\|_{\frac{1}{2},\Gamma_i}^2\right). \qquad (3.3.95)
\end{aligned}$$

To bound terms of the form $\sum_{i=1}^N\|\mathcal{I}_H^{C,u}v-v]\|_{\frac{1}{2},\Gamma_i}$ and $\sum_{i=1}^N\|\mathcal{I}_H^{C,p}w-w\|_{\frac{1}{2},\Gamma_i}$, we use (3.2.27) and (3.2.39) to obtain

$$\sum_{i=1}^N\|\mathcal{I}_H^{C,u}v-v]\|_{\frac{1}{2},\Gamma_i}\le CH^{\hat{m}-\frac{1}{2}}\|v\|_{\hat{m}+\frac{1}{2},\Omega_i},\qquad 0\le\hat{m}\le m+1,\ \forall v\in V, \qquad (3.3.96)$$

$$\sum_{i=1}^N\|\mathcal{I}_H^{C,p}w-w]\|_{\frac{1}{2},\Gamma_i}\le CH^{\hat{m}-\frac{1}{2}}\|w\|_{\hat{m}+\frac{1}{2},\Omega_i},\qquad 0\le\hat{m}\le m+1,\ \forall w\in W. \qquad (3.3.97)$$

In order to bound the initial errors, $\|\phi_\sigma(0)\|$, $\|\phi_p(0)\|$, $\|\phi_\gamma(0)\|$, and $\|\phi_z(0)\|$, we recall that we obtain the discrete initial data from the elliptic projection of the continuous initial data (see (3.3.24)). Following the arguments similar to the ones used to arrive at (3.3.43), we get

$$\|\phi_\sigma(0)\|+\|\phi_p(0)\|+\|\phi_\gamma(0)\|+\|\phi_z(0)\|\le C\left(\|\psi_\sigma(0)\|+\|\psi_p(0)\|+\|\psi_\gamma(0)\|+\|\psi_z(0)\|+\|\psi_u(0)\|\right). \qquad (3.3.98)$$

Finally, error bounds (3.3.95)−(3.3.98) combined with the approximation results (3.2.27)-−(3.2.34), (3.3.12) and (3.3.15), as well as (3.3.62) completes the proof. □

Remark 3.3.2. *The above theorem implies that for sufficiently smooth solution variables, the error in using our method is of* $\mathcal{O}\left(h^{k+1}+h^{l+1}+h^{j+1}+h^{r+1}+h^{s+1}+H^{m+\frac{1}{2}}\right)$. *Assuming we use inf-sup stable pairs of FE spaces containing polynomials of degree* $l=j=s$, *and* $k=r$, *and* $l\leq k$, *we could choose* $H=\mathcal{O}\left(h^{\frac{l+1}{m+1/2}}\right)$ *to get a total error bound of order* $\mathcal{O}\left(h^{l+1}\right)$. *For example, for the choice of* $l=0$ *and* $m=1$, *we could choose* $H=\mathcal{O}\left(h^{\frac{2}{3}}\right)$ *and for* $l=0$ *and* $m=2$, *we could choose* $H=\mathcal{O}\left(h^{\frac{2}{5}}\right)$ *to obtain a total convergence rate of* $\mathcal{O}(h)$. *We will demonstrate the results for different choices of* $H(h)$ *in the numerical results section.*

3.4 Implementation

In this section, we discuss the implementation of the multiscale mortar domain decomposition technique discussed in this chapter. First, we provide a fully discrete version of the system (3.2.8)−(3.2.14) using backward Euler time discretization. We use a related formulation where a time differentiated elasticity equation (2.2.3) is used. The reason for such a reformulation was discussed in the previous chapter. The fully discrete formulation of the technique is then reduced to an interface problem which can then be solved using an iterative solver like GMRES. Finally, we discuss the possibility of computing and storing a multiscale basis which will help in increasing the efficiency of the method.

3.4.1 Time discretization

For time discretization, we use the backward Euler method. Let $\{t_n\}_{n=0}^{N_T}$, $t_n=n\Delta t$, $\Delta t=T/N_T$, be a uniform partition of $(0,T)$. With these choices, the fully discrete problem corresponding to (3.2.8)−(3.2.14) reads as follows: for $0\leq n\leq N_T-1$ and $1\leq i\leq N$, find

$(\sigma_{h,i}^{n+1}, u_{h,i}^{n+1}, \gamma_{h,i}^{n+1}, z_{h,i}^{n+1}, p_{h,i}^{n+1}, \lambda_H^{n+1}) \in \mathbb{X}_{h,i} \times V_{h,i} \times \mathbb{Q}_{h,i} \times Z_{h,i} \times W_{h,i} \times \Lambda_H$ such that:

$$\begin{aligned}
&\left(A\left(\sigma_{h,i}^{n+1} + \alpha p_{h,i}^{n+1} I\right), \tau\right)_{\Omega_i} + \left(u_{h,i}^{n+1}, \operatorname{div} \tau\right)_{\Omega_i} + \left(\gamma_{h,i}^{n+1}, \tau\right)_{\Omega_i} \\
&\qquad = \langle g_u^{n+1}, \tau\, n_i\rangle_{\partial\Omega_i \cap \Gamma_D^u} + \langle \lambda_H^{u,n+1}, \tau\, n_i\rangle_{\Gamma_i}, && \forall \tau \in \mathbb{X}_{h,i}, \quad (3.4.1)\\
&\left(\operatorname{div} \sigma_{h,i}^{n+1}, v\right)_{\Omega_i} = -\left(f^{n+1}, v\right)_{\Omega_i}, && \forall v \in V_{h,i}, \quad (3.4.2)\\
&\left(\sigma_{h,i}^{n+1}, \xi\right)_{\Omega_i} = 0, && \forall \xi \in \mathbb{Q}_{h,i}, \quad (3.4.3)\\
&\left(K^{-1} z_{h,i}^{n+1}, \zeta\right)_{\Omega_i} - \left(p_{h,i}^{n+1}, \operatorname{div} \zeta\right)_{\Omega_i} = -\langle g_p^{n+1}, \zeta \cdot n_i\rangle_{\partial\Omega_i \cap \Gamma_D^p} - \langle \lambda_H^{p,n+1}, \zeta \cdot n_i\rangle_{\Gamma_i}, && \forall \zeta \in Z_{h,i}, \quad (3.4.4)\\
&c_0 \left(\frac{p_{h,i}^{n+1} - p_{h,i}^{n}}{\Delta t}, w\right)_{\Omega_i} + \alpha \left(\frac{A\left(\sigma_{h,i}^{n+1} - \sigma_{h,i}^{n}\right)}{\Delta t}, wI\right)_{\Omega_i} \\
&\qquad + \alpha \left(A\alpha \frac{p_{h,i}^{n+1} - p_{h,i}^{n}}{\Delta t} I, wI\right)_{\Omega_i} + \left(\operatorname{div} z_{h,i}^{n+1}, w\right)_{\Omega_i} = \left(g^{n+1}, w\right)_{\Omega_i}, && \forall w \in W_{h,i}, \\
& && (3.4.5)\\
&\sum_{i=1}^{N} \langle \sigma_{h,i}^{n+1}\, n_i, \mu^u\rangle_{\Gamma_i} = 0, && \forall \mu^u \in \Lambda_H^u, \\
& && (3.4.6)\\
&\sum_{i=1}^{N} \langle z_{h,i}^{n+1} \cdot n_i, \mu^p\rangle_{\Gamma_i} = 0, && \forall \mu^p \in \Lambda_H^p. \\
& && (3.4.7)
\end{aligned}$$

Note that similarly to the matching grid case in the previous chapter, the non-matching multiscale grid method also requires initial data $p_{h,i}^0$ and $\sigma_{h,i}^0$. Such data can be obtained by taking $p_{h,i}^0$ to be the L^2-projection of p_0 onto $W_{h,i}$ and solving a mixed elasticity non-matching grid non-overlapping domain decomposition problem obtained from (3.4.1)−(3.4.3) and (3.4.6) with $n = -1$ (see [48]). Also, following the case of the matching grid domain decomposition method, we will utilize a related formulation in which the elasticity equation, (2.2.3), is differentiated in time. The reason for this was discussed in the analysis of the resulting interface problem in the previous chapter. We introduce new variables $\dot{u} = \partial_t u$, $\dot{\gamma} = \partial_t \gamma$, and $\lambda_H^{\dot{u}} = \partial_t \lambda_H^u$ representing the time derivatives of the displacement, rotation, and displacement-Lagrange multiplier function, respectively. Using time-differentiated equation (2.2.3) and backward Euler (see the subsection on time discretization in the previous chapter for details), we get

$$\begin{aligned}
&\left(A(\sigma_{h,i}^{n+1}+\alpha p_{h,i}^{n+1}I),\tau\right)_{\Omega_i}+\Delta t\left(\dot{u}_{h,i}^{n+1},\operatorname{div}\tau\right)_{\Omega_i}+\Delta t\left(\dot{\gamma}_{h,i}^{n+1},\tau\right)_{\Omega_i}\\
&\quad=\Delta t\langle\partial_t g_u^{n+1},\tau\, n_i\rangle_{\partial\Omega_i\cap\Gamma_D^u}+\Delta t\langle\dot{\lambda}_H^{u,n+1},\tau\, n_i\rangle_{\Gamma_i}+\left(A(\sigma_{h,i}^{n}+\alpha p_{h,i}^{n}I),\tau\right)_{\Omega_i},\qquad\forall\tau\in\mathbb{X}_{h,i}.
\end{aligned}\tag{3.4.8}$$

The original variables can be recovered using

$$u_h^n=u_h^0+\Delta t\sum_{k=1}^{n}\dot{u}_h^k,\quad \gamma_h^n=\gamma_h^0+\Delta t\sum_{k=1}^{n}\dot{\gamma}_h^k,\quad \lambda_H^{u,n}=\lambda_H^{u,0}+\sum_{k=1}^{n}\dot{\lambda}_H^{u,k}.\tag{3.4.9}$$

3.4.2 Reduction to an interface problem

Similar to the non-overlapping matching grid domain decomposition algorithm discussed in the previous chapter, we solve the system resulting from (3.4.8), (3.4.2)−(3.4.7) at each time step by reducing it to an interface problem for the Lagrange multiplier function λ_H. Following similar arguments as in the previous chapter, we introduce two sets of complementary subdomain problems.

The first set of problems reads as follows: for $1\le i\le N$, find $(\bar{\sigma}_{h,i}^{n+1},\bar{u}_{h,i}^{n+1},\bar{\gamma}_{h,i}^{n+1},\bar{z}_{h,i}^{n+1},\bar{p}_{h,i}^{n+1})\in\mathbb{X}_{h,i}\times V_{h,i}\times\mathbb{Q}_{h,i}\times Z_{h,i}\times W_{h,i}$ such that

$$\begin{aligned}
&\left(A(\bar{\sigma}_{h,i}^{n+1}+\alpha\bar{p}_{h,i}^{n+1}I),\tau\right)_{\Omega_i}+\Delta t\left(\bar{\dot{u}}_{h,i}^{n+1},\operatorname{div}\tau\right)_{\Omega_i}+\Delta t\left(\bar{\dot{\gamma}}_{h,i}^{n+1},\tau\right)_{\Omega_i}\\
&\quad=\Delta t\langle g_u^{n+1},\tau\, n_i\rangle_{\partial\Omega_i\cap\Gamma_D^u}+\left(A(\sigma_{h,i}^{n}+\alpha p_{h,i}^{n}I),\tau\right)_{\Omega_i},&&\forall\tau\in\mathbb{X}_{h,i},&(3.4.10)\\
&\left(\operatorname{div}\bar{\sigma}_{h,i}^{n+1},v\right)_{\Omega_i}=-\left(f^{n+1},v\right)_{\Omega_i},&&\forall v\in V_{h,i},&(3.4.11)\\
&\left(\bar{\sigma}_{h,i}^{n+1},\xi\right)_{\Omega_i}=0,&&\forall\xi\in\mathbb{Q}_{h,i},&(3.4.12)\\
&\left(K^{-1}\bar{z}_{h,i}^{n+1},\zeta\right)_{\Omega_i}-\left(\bar{p}_{h,i}^{n+1},\operatorname{div}\zeta\right)_{\Omega_i}=-\langle g_p^{n+1},\zeta\cdot n_i\rangle_{\partial\Omega_i\cap\Gamma_D^p},&&\forall\zeta\in Z_{h,i},&(3.4.13)\\
&c_0\left(\bar{p}_{h,i}^{n+1},w\right)_{\Omega_i}+\alpha\left(A(\bar{\sigma}_{h,i}^{n+1}+\alpha\bar{p}_{h,i}^{n+1}I),wI\right)_{\Omega_i}+\Delta t\left(\operatorname{div}\bar{z}_{h,i}^{n+1},w\right)_{\Omega_i}\\
&\quad=\Delta t\left(g^{n+1},w\right)_{\Omega_i}+c_0\left(p_{h,i}^{n},w\right)_{\Omega_i}+\alpha\left(A(\sigma_{h,i}^{n}+\alpha p_{h,i}^{n}I),wI\right)_{\Omega_i},&&\forall w\in W_{h,i}.&(3.4.14)
\end{aligned}$$

Note that these subdomain problems have zero Dirichlet data at the subdomain interface, the true source terms f and g, outside boundary conditions g_u and g_p, and initial data $\sigma_{h,i}^n$ and $p_{h,i}^n$.

The second set of equations reads as follows: given $\lambda_H \in \Lambda_H$, for $1 \le i \le N$, find $(\sigma_{h,i}^{*,n+1}(\lambda_H), \dot{u}_{h,i}^{*,n+1}(\lambda_H), \dot{\gamma}_{h,i}^{*,n+1}(\lambda_H), z_{h,i}^{*,n+1}(\lambda_H), p_{h,i}^{*,n+1}(\lambda_H)) \in \mathbb{X}_{h,i} \times V_{h,i} \times \mathbb{Q}_{h,i} \times Z_{h,i} \times W_{h,i}$ such that:

$$\begin{aligned}
&\left(A\big(\sigma_{h,i}^{*,n+1}(\lambda_H) + \alpha p_{h,i}^{*,n+1}(\lambda_H) I\big), \tau\right)_{\Omega_i} + \Delta t \left(\dot{u}_{h,i}^{*,n+1}(\lambda_H), \operatorname{div} \tau\right)_{\Omega_i} \\
&\qquad + \Delta t \left(\dot{\gamma}_{h,i}^{*,n+1}(\lambda_H), \tau\right)_{\Omega_i} = \Delta t \left\langle \lambda_H^{\dot{u}}, \tau\, n_i \right\rangle_{\Gamma_i}, \qquad \forall \tau \in \mathbb{X}_{h,i}, \qquad (3.4.15)\\
&\left(\operatorname{div} \sigma_{h,i}^{*,n+1}(\lambda_H), v\right)_{\Omega_i} = 0, \qquad \forall v \in V_{h,i}, \qquad (3.4.16)\\
&\left(\sigma_{h,i}^{*,n+1}(\lambda_H), \xi\right)_{\Omega_i} = 0, \qquad \forall \xi \in \mathbb{Q}_{h,i}, \qquad (3.4.17)\\
&\left(K^{-1} z_{h,i}^{*,n+1}(\lambda_H), \zeta\right)_{\Omega_i} - \left(p_{h,i}^{*,n+1}(\lambda_H), \operatorname{div} \zeta\right)_{\Omega_i} = -\langle \lambda_H^{p,n+1}, \zeta \cdot n_i \rangle_{\Gamma_i}, \qquad \forall \zeta \in Z_{h,i}, \qquad (3.4.18)\\
&c_0 \left(p_{h,i}^{*,n+1}(\lambda_H), w\right) + \alpha \left(A\big(\sigma_{h,i}^{*,n+1}(\lambda_H) + \alpha p_{h,i}^{*,n+1}(\lambda_H) I\big), wI\right)_{\Omega_i} \\
&\qquad + \Delta t \left(\operatorname{div} z_{h,i}^{*,n+1}(\lambda_H)\right) = 0, \qquad \forall w \in W_{h,i}. \qquad (3.4.19)
\end{aligned}$$

Note that this set of problems has λ_H as the Dirichlet boundary data on the interface Γ, compared to λ_h in the matching grid case. This system also has zero source terms, zero boundary data on part of the outside boundary $\partial\Omega$, and zero data from the previous time step.

Define the bilinear form $a_{H,i}^{n+1} : \lambda_H \times \lambda_H \to \mathbb{R}$, $1 \le i \le N$, $a_H^{n+1} : \lambda_H \times \lambda_H \to \mathbb{R}$, and the linear functional $g_H^{n+1} : \lambda_H \to \mathbb{R}$ for all $0 \le n \le N_T - 1$ by

$$a_{H,i}^{n+1}(\lambda_H, \mu) = \langle \sigma_{h,i}^{*,n+1}(\lambda_H)\, n_i, \mu^u \rangle_{\Gamma_i} - \langle z_{h,i}^{*,n+1}(\lambda_H) \cdot n_i, \mu^p \rangle_{\Gamma_i}, \quad a_H^{n+1}(\lambda_H, \mu) = \sum_{i=1}^{N} a_{H,i}^{n+1}(\lambda_H, \mu), \tag{3.4.20}$$

$$g_H^{n+1}(\mu) = \sum_{i=1}^{N} \left(-\langle \bar{\sigma}_{h,i}^{n+1}\, n_i, \mu^u \rangle_{\Gamma_i} + \langle \bar{z}_{h,i}^{n+1} \cdot n_i, \mu^p \rangle_{\Gamma_i} \right). \tag{3.4.21}$$

It follows from (3.4.6)–(3.4.7) that for each $0 \le n \le N_T - 1$, the solution to the global problem (3.4.8), (3.4.2)–(3.4.7) is equivalent to solving the interface problem for $\lambda_H^{n+1} \in \Lambda_H$:

$$a_H^{n+1}(\lambda_H^{n+1}, \mu) = g_H^{n+1}(\mu), \quad \forall \mu \in \Lambda_H, \tag{3.4.22}$$

and setting

$$\begin{aligned}
&\sigma_{h,i}^{n+1} = \sigma_{h,i}^{*,n+1}(\lambda_H^{n+1}) + \bar{\sigma}_{h,i}^{n+1}, \quad \dot{u}_{h,i}^{n+1} = \dot{u}_{h,i}^{*,n+1}(\lambda_H^{n+1}) + \bar{\dot{u}}_{h,i}^{n+1}, \quad \dot{\gamma}_{h,i}^{n+1} = \dot{\gamma}_{h,i}^{*,n+1}(\lambda_H^{n+1}) + \bar{\dot{\gamma}}_{h,i}^{n+1},\\
&z_{h,i}^{n+1} = z_{h,i}^{*,n+1}(\lambda_H^{n+1}) + \bar{z}_{h,i}^{n+1}, \quad p_{h,i}^{n+1} = p_{h,i}^{*,n+1}(\lambda_H^{n+1}) + \bar{p}_{h,i}^{n+1}.
\end{aligned}$$

3.4.3 Solving the interface problem

In order to solve the interface problem (3.4.22), we introduce linear operators $\mathcal{A}_{H,i}^{n+1} : \Lambda_{H,i} \to \Lambda_{H,i}$, for $1 \leq i \leq N$ and $\mathcal{A}_H^{n+1} : \Lambda_H \to \Lambda_H$ such that for any $\lambda_H \in \Lambda_H$,

$$\langle \mathcal{A}_{H,i}^{n+1} \lambda_{H,i},\, \mu \rangle_{\Gamma_i} = \langle \sigma_{h,i}^{*,n+1}(\lambda_H)\, n_i, \mu^u \rangle_{\Gamma_i} - \langle z_{h,i}^{*,n+1}(\lambda_H) \cdot n_i, \mu^p \rangle_{\Gamma_i}, \qquad \forall \mu \in \Lambda_{H,i}, \tag{3.4.23}$$

$$\mathcal{A}_H^{n+1} \lambda_H = \sum_{i=1}^{N} \mathcal{A}_{H,i}^{n+1} \lambda_{H,i}, \tag{3.4.24}$$

where $\lambda_{H,i}$ and $\Lambda_{H,i}$ denote the restrictions of λ_H and Λ_H to Γ_i, respectively. We also define the vector $G_H^{n+1} \in \Lambda_H$ as

$$\langle G_H^{n+1}, \mu \rangle_\Gamma = \sum_{i=1}^{N} \left(-\langle \bar{\sigma}_{h,i}^{n+1}\, n_i, \mu^u \rangle_{\Gamma_i} + \langle \bar{z}_{h,i}^{n+1} \cdot n_i, \mu^p \rangle_{\Gamma_i} \right), \qquad \forall \mu \in \Lambda_{H,i}. \tag{3.4.25}$$

Interface problem (3.4.22) can now be reformulated as finding $\lambda_H \in \Lambda_H$ such that

$$\mathcal{A}_H^{n+1} \lambda_H = G_H^{n+1}. \tag{3.4.26}$$

Consider the L^2 orthogonal projections, $\mathcal{Q}_{h,i}^{u,T} : \mathbb{X}_{h,i} n_i \to \Lambda_H^u$ and $\mathcal{Q}_{h,i}^{p,T} : Z_{h,i} \cdot n_i :\to \Lambda_H^p$ such that for any $\tau \in \mathbb{X}_{h,i}$ and $\zeta \in Z_{h,i}$,

$$\begin{aligned} \langle \mathcal{Q}_{h,i}^{u,T} (\tau\, n_i) - \tau\, n_i, \mu^u \rangle_{\Gamma_i} &= 0, && \forall \mu^u \in \Lambda_H^u, \\ \langle \mathcal{Q}_{h,i}^{p,T} (\zeta \cdot n_i) - \zeta \cdot n_i, \mu^p \rangle_{\Gamma_i} &= 0, && \forall \mu^p \in \Lambda_H^p. \end{aligned}$$

where n_i is the unit outward normal to $\partial\Omega_i$. Define $\mathcal{Q}_{h,i}^T : \mathbb{X}_{h,i} n_i \times Z_{h,i} \cdot n_i \to \Lambda_H^u \times \Lambda_H^p$ such that for any $\tau \in \mathbb{X}_{h,i}$ and $\zeta \in Z_{h,i}$

$$\mathcal{Q}_{h,i}^T \begin{pmatrix} \tau\, n_i \\ \zeta \cdot n_i \end{pmatrix} = \begin{pmatrix} \mathcal{Q}_{h,i}^{u,T} \tau\, n_i \\ \mathcal{Q}_{h,i}^{p,T} \zeta \cdot n_i \end{pmatrix}. \tag{3.4.27}$$

Using the above notations, we note that

$$\mathcal{A}_{H,i}^{n+1} \lambda_{H,i} = \mathcal{Q}_{h,i}^T \begin{pmatrix} \sigma_{h,i}^{*,n+1}(\lambda_H)\, n_i \\ -z_{h,i}^{*,n+1}(\lambda_H) \cdot n_i \end{pmatrix}, \qquad G_H^{n+1} = \begin{pmatrix} \sum_{i=1}^N -\mathcal{Q}_{h,i}^{u,T} \bar{\sigma}_{h,i}^{n+1}\, n_i \\ \sum_{i=1}^N \mathcal{Q}_{h,i}^{p,T} \bar{z}_{h,i}^{n+1} \cdot n_i \end{pmatrix}, \tag{3.4.28}$$

for $i = 1, 2, \ldots N$.

To solve the interface problem (3.4.26), we use an iterative method like GMRES with an initial guess $\lambda_{H,0} \in \Lambda_H$ at each time step, $t_n = n\Delta t$. A detailed description is given in Algorithm 1 (also see [33]).

Algorithm 1 Solving interface problem using GMRES algorithm.

1. Solve the first set of complementary equations, (3.4.10)−(3.4.14), and compute G_H^{n+1} using (3.4.28).
2. Pick an initial guess $\lambda_{H,0} \in \Lambda_H$.
3. Project the mortar function onto the subdomain boundaries, $\lambda_{H,0,i} \longrightarrow \mathcal{Q}_{h,i}\left(\lambda_{H,0,i}\right)$.
4. Solve the second set of complementary equations, (3.4.15)−(3.4.19), using the projected function $\mathcal{Q}_{h,i}\left(\lambda_{H,0,i}\right)$ as Dirichlet boundary data to obtain $\sigma_{h,i}^{*,n+1}(\lambda_{H,0})$ and $z_{h,i}^{*,n+1}(\lambda_{H,0})$.
5. Project the solution variables to the mortar space, $\sigma_{h,i}^{*,n+1}(\lambda_{H,0})\, n_i \longrightarrow \mathcal{Q}_{h,i}^{u,T}\sigma_{h,i}^{*,n+1}(\lambda_{H,0})\, n_i$ and $z_{h,i}^{*,n+1}(\lambda_{H,0}) \cdot n_i \longrightarrow \mathcal{Q}_{h,i}^{p,T} z_{h,i}^{*,n+1}(\lambda_{H,0}) \cdot n_i$.
6. Compute the action $\mathcal{A}_H^{n+1}\lambda_{H,0}$ using (3.4.28).
7. Update λ_H using $\mathcal{A}_H^{n+1}\lambda_{H,0}$ in the GMRES algorithm.
8. Repeat steps $3-7$, with updated values of λ_H, until the residual for the GMRES algorithm goes below a predetermined tolerance.

This method has the performance advantage over the similar method for matching grids discussed in the previous chapter that a coarse mortar mesh could be used to obtain a smaller interface problem due to the reduction in the mortar degrees of freedom. We implement this algorithm and study various test cases in the numerical results section.

3.4.4 Implementation with multiscale stress-flux basis (MSB)

A coarser mortar mesh can lead to a smaller interface problem, but even in that case, the number of subdomain solves of the type (3.4.15)−(3.4.19) is directly proportional to both the number of mortar space degrees of freedom and the number of time steps used. We propose the construction and use of a multiscale stress-flux basis (MSB) which makes the number of subdomain solves independent of the number of iterations required for the interface problem and the number of time steps used.

Let $\left\{\beta_{H,i}^k\right\}_{k=0}^{N_H}$ be a basis for $\Lambda_{H,i}$, where N_H denotes the number of degrees of freedom associated with the finite element space $\Lambda_{H,i}$. We calculate and store the action of the interface operator of the form

$$\mathcal{A}_{H,i}\beta_{H,i}^k = \mathcal{Q}_{h,i}^T \begin{pmatrix} \sigma_{h,i}^*(\beta_{H,i}^k)\, n_i \\ -z_{h,i}^*(\beta_{H,i}^k) \cdot n_i \end{pmatrix}, \tag{3.4.29}$$

for $k = 1, 2, \ldots N_H$, where we obtain $\sigma_{h,i}^*(\beta_{H,i}^k)$ and $z_{h,i}^*(\beta_{H,i}^k)$ by solving (3.4.15)−(3.4.19) with $\beta_{H,i}^k$ as the Dirichlet boundary data. A detailed description of the construction of the multiscale sbasis elements $\left\{\phi_{H,i}^k\right\}_{k=0}^{N_H}$, where $\phi_{H,i}^k = \mathcal{A}_{H,i}\beta_{H,i}^k$ is given in Algorithm 2, (also see [33, 48] for similar constructions).

Algorithm 2 Construction of multiscale stress-flux basis

for $k = 1, \ldots, N_H$:

1. Project $\beta_{H,i}^k$ onto the subdomain boundary, $\beta_{H,i}^k \longrightarrow \mathcal{Q}_{h,i}\left(\beta_{H,i}^k\right)$.
2. Solve the system (3.4.15)−(3.4.19) using the projected function $\mathcal{Q}_{h,i}\left(\beta_{H,i}^k\right)$ as the Dirichlet boundary data, to obtain $\sigma_{h,i}^*(\beta_{H,i}^k)$ and $z_{h,i}^*(\beta_{H,i}^k)$.
3. Project the solution variables to the mortar space to obtain $\phi_{H,i}^k = \begin{pmatrix} \mathcal{Q}_{h,i}^{u,T}\sigma_{h,i}^*(\beta_{H,i}^k)\, n_i \\ -\mathcal{Q}_{h,i}^{p,T} z_{h,i}^*(\beta_{H,i}^k) \cdot n_i \end{pmatrix}$.

end for

For any $\mu \in \Lambda_{H,i}$, consider the mortar basis decomposition, $\mu = \sum_{k=0}^{N_H} \mu_i \beta_{H,i}^k$. We can calculate the action of the interface operator on μ as follows:

$$\mathcal{A}_{H,i}\mu = \sum_{k=0}^{N_H} \mu_i \phi_{H,i}^k. \tag{3.4.30}$$

We can use the multiscale basis to replace steps $3-6$ in Algorithm 1, by taking linear combinations of the form (3.4.30). Note that the multiscale stress-flux basis is computed and saved once and can be reused over all time steps, which gains a significant performance advantage in the case of time-dependent parabolic problems like the one we are studying.

We further discuss and compare the efficiency of using the multiscale stress-flux basis with other methods in Example 2 in the numerical section of this chapter.

3.5 Numerical Results

In this section, we report the results of various numerical tests designed to verify and compare the well-posedness, stability, and convergence of the multiscale mortar non-overlapping domain decomposition method for the Biot system of poroelasticity that we have developed in this chapter. We compare the computational efficiency in different cases, including the matching and non-matching grids on the subdomains, and also discuss the advantages of using a multiscale basis. The numerical schemes are implemented using deal.II finite element package [91,92].

In all the examples, we have used the FE triplet $\mathbb{X}_h \times V_h \times \mathbb{Q}_h = \mathcal{BDM}_1^2 \times Q_0^2 \times Q_0$ ([11]) for elasticity and the FE pair $Z_h \times W_h = \mathcal{BDM}_1 \times Q_0$ ([22]) for Darcy on quadrilateral meshes. Here Q_k denotes polynomials of degree k in each variable. For the mortar spaces, λ_H^u is taken to be DQ_m^2, and λ_H^p is taken to be DQ_m with $m = 1$ or 2, where DQ_k represents the discontinuous finite element spaces containing polynomials of degree k, which lives on the subdomain interface. The degrees of FEM spaces used in this section is given in Table 11. For solving the interface problem, we use non-restarted unpreconditioned GMRES with a tolerance on the relative residual $\frac{r_k}{r_0}$ as the stopping criteria. For all the examples, this tolerance is taken to be 10^{-6}.

Table 11: Degree of polynomials associated with FEM spaces used for numerical experiments.

$\Lambda_H : m$	$\mathbb{X}_h : k$	$V_h : l$	$\mathbb{Q}_h : j$	$Z_h : r$	$W_h : s$
1 (linear)	1	0	0	1	0
2 (quadratic)	1	0	0	1	0

In Example 1, we test and compare the convergence, stability, and efficiency of the multiscale mortar DD method using linear ($m = 1$) and quadratic ($m = 2$) mortar spaces. We do this by solving a system of equations with a known solution on successively refined meshes.

In Example 2, we apply the multiscale DD method to solve a more practical problem, using a highly heterogeneous medium. We compare the efficiency of the multiscale versus fine scale methods and study the computational advantage of constructing a multiscale stress-flux basis (MSB) discussed in Section 3.4.4 of this chapter.

In Example 1, we solve the system of PDEs on a checkerboard global mesh, which consists of non-matching grids on all subdomain interfaces. In particular, the coarsest multiscale mesh

in all examples follows a subdomain mesh size ratio $\frac{1}{4} : \frac{1}{6} : \frac{1}{6} : \frac{1}{4}$ as shown in Figure 4. The corresponding coarsest mortar-interface mesh consists of two elements with mesh size $\frac{1}{2}$.

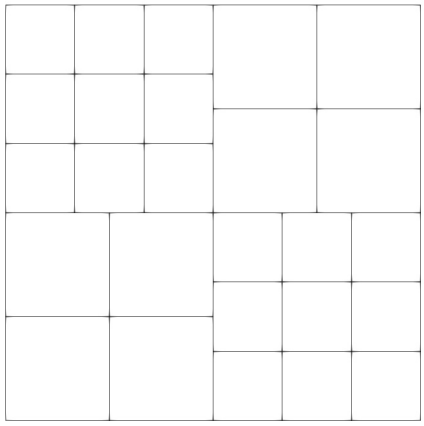

Figure 4: Example 1, coarsest non matching subdomain grid on $(0,1)^2$.

3.5.1 Example 1: testing convergence rates

In this example, we test the well-posedness, convergence, and stability of the multiscale mortar DD method using linear ($m = 1$) and quadratic ($m = 2$) mortar spaces. The global computational domain Ω is taken to be the unit square $(0,1)^2$. We consider the following analytical solution

$$p = \exp(t)(\sin(\pi x)\cos(\pi y) + 10), \quad u = \exp(t)\begin{pmatrix} x^3y^4 + x^2 + \sin((1-x)(1-y))\cos(1-y) \\ (1-x)^4(1-y)^3 + (1-y)^2 + \cos(xy)\sin(x) \end{pmatrix}.$$

The physical and numerical parameters are given in Table 12. Using this information, we derive the right hand side and initial conditions essential to solve the system (1.3.1)−(1.3.9). We partition Ω into four square subdomains with non-matching grids as shown in Figure 4.

We consider two different cases, with linear and quadratic mortar spaces, where the former contains polynomials of degree 1 and the latter contains polynomials of degree 2. To test the convergence and verify the theoretical apriori error estimates, we successively refine the subdomain and mortar meshes. In the linear mortar case, we maintain a subdomain to mortar

mesh ratio such that $H = Ch$, and in the quadratic mortar case, we maintain the ratio such that $H = C\sqrt{h}$.

The convergence tables for the cases with linear and quadratic mortar spaces with $\Delta t = 10^{-4}$ and $c_0 = 1.0$ are given in Tables 13 and 14, respectively. Tables 15−16 present the convergence table in the case of linear mortar and quadratic mortar spaces, respectively with $\Delta t = 10^{-4}$, and $c_0 = 10^{-3}$. We present the number of interface iterations, relative errors, and their convergence rates in these tables. Solution plots in the case of linear mortar with an intermediate level of refinement, $h = 1/32$, $\Delta t = 10^{-3}$ and $c_0 = 1.0$ is given in Figure 5 in order to compare with the plots obtained in the previous chapter using monolithic domain decomposition technique using matching grids on subdomain interfaces. Note that plots in the case of quadratic mortar space look similar.

Table 12: Example 1, physical and numerical parameters.

Parameter	Value
Permeability tensor (K)	I
Lame coefficient (μ)	100.0
Lame coefficient (λ)	100.0
Mass storativity (c_0)	$1.0, 10^{-3}$
Biot-Willis constant (α)	1.0
Time step (Δt)	$10^{-3}, 10^{-4}$
Number of time steps	100

The numerical results that we observe are consistent with the theoretical results from the previous sections. In particular, we demonstrate the stability of the method over a 100 time steps, and Tables 13 and 14 confirm convergence rates that follow from Theorem 3.3.9 and Table 11. With linear mortar $m = 1$ and $H = Ch$, the interface error is $\mathcal{O}\left(h^{\frac{3}{2}}\right)$. With quadratic mortar $m = 2$ and $H = \sqrt{h}$, the interface error is $\mathcal{O}\left(h^{\frac{5}{4}}\right)$. In both the cases, it is dominated by the subdomain error, which is $\mathcal{O}(h)$. As a result, we expect at least $\mathcal{O}(h)$ convergence in both cases, which is what we observe. Comparison of the number of interface iterations required in the case of linear and quadratic mortars in Tables 13 and 14, respectively shows that both mortar degrees result in similar accuracy for the same level of subdomain mesh refinement. This

is despite the fact that the quadratic mortar case requires smaller number of interface iterations compared to the linear mortar case with the same level of subdomain mesh refinement. This is due to the choice of a coarser mortar mesh in the case of quadratic mortar case. This points towards a way to decrease the number of interface iterations by using a coarser mesh and higher mortar space degree, without any loss in accuracy. Tables 15−16 confirm that the stability and error bounds proved in previous sections are not affected by smaller values of c_0. Further, Figure 5 demonstrates the efficacy of the method in enforcing continuity of solution variables across subdomain interfaces, weakly using coarse mortar spaces. In fact, the solution looks almost identical to the ones obtained using matching-subdomain grids in the previous chapter, with a smaller number of interface iterations for same level of finest subdomain mesh refinement. This demonstrates the advantage of using the multiscale technique we have developed over the completely matching case that was discussed in the previous chapter.

3.5.2 Example 2: heterogeneous medium

In this example, we demonstrate the performance of our method in a practical application with highly heterogeneous medium. First, we compare the efficiency of our multiscale mortar method, where $H > h$, with a fine scale method, where $H = h$. We expect the former to be more efficient than the latter because of weaker enforcement of continuity across subdomain interfaces using a coarser mortar space in the case of the multiscale method. We then study the computational advantage of using a multiscale stress-flux basis (MSB) over not using an MSB. In the case of no-MSB, the number of subdomain solves is total #GMRES iterations across all time steps + 2×number of time steps, where the last term comes from two extra solves required to solve the system (3.4.10)−(3.4.19) initially and recovering the final solution once the GMRES converges. Similarly, in the case of using MSB, total number of subdomain solves equals the $\dim(\Lambda_H)$+2×number of time steps. Note that the first term in the number of solves in the case of no-MSB is directly proportional to the number of time steps used in time discretization, while the same in the case of MSB method is independent of the number of time steps used. This leads to MSB method being far more efficient than the no-MSB method with any choice of mortar, as long as enough number of time steps are used.

To obtain the desired level of heterogeneity in the medium, we use the porosity and the

permeability data from the Society of Petroleum Engineers 10th Comparative Solution Project (SPE10)[1]. The porosity and permeability fields are given on a 60×220 grid and we use the rectangular region $(0, 60) \times (0, 220)$ as the computational domain. We decompose the global domain into 3×5 subdomains consisting of identical rectangular blocks. The Young's modulus is obtained from the porosity field data using the relation $E = 10^2 \left(1 - \frac{\phi}{c}\right)^{2.1}$, where $c = 0.5$, refers to the porosity at which the Young's modulus vanishes, see [53] for details. These input fields are presented in Figure 6. We use parameters and boundary conditions as mentioned in Table 17, along with zero source terms. These conditions describe a flow from left to right, driven by the gradient in the pressure. We use a compatible initial condition for pressure, $p_0 = 1 - x$. To obtain the essential discrete initial data, we take the elliptic projection of the continuous initial data, see (3.3.24). In particular, we set p_h^0 to be the L^2-projection of p_0 onto W_h and solve a mixed elasticity domain decomposition problem at $t = 0$ to obtain σ_h^0. We also obtain u_h^0, γ_h^0, and $\lambda_H^{u,0}$ from this solve.

We use a global 60×220 grid and solve the problem using both fine scale ($H = h$) and coarse ($H > h$) mortar spaces. For the coarse mortar case, we use both linear ($m = 1$) and quadratic mortars ($m = 2$) with one and two mortars per subdomain interface. The comparison of the computed solution using different choices of mortars is given in Figures 7–12. Comparison of the number of solves required for different choices of mortar, both in the no-MSB and MSB cases are given in Table 18. We report the number of subdomain solves which dominates the computational complexity of the method.

Table 18 clearly shows that using the multiscale mortar method requires fewer number of solves compared to fine scale method and hence the former is computationally less expensive than the latter. Comparison of the computed solution for various choices of mortars in Figures 7–12 shows that we retain good amount of accuracy even in the case of the coarsest mortar case with one linear mortar per interface. We also note that using a single quadratic mortar per interface yields almost identical results as the fine scale solution which emphasizes our observation from the previous example that a coarse mortar can be compensated by choosing a higher degree for mortar space Λ_H. Table 18 also demonstrates the superiority of using MSB for a time-dependent multiscale problem like the one in our case. The number of solves in the case of no MSB is at least an order of magnitude bigger than the MSB case which implies that the construction of

[1] https://www.spe.org/web/csp/datasets/set02.htm

MSB is an excellent tool to make our multiscale mortar method even more efficient than it already is compared to the fine scale methods discussed in the previous chapter.

3.6 Chapter Conclusions

In this chapter we presented a multiscale mortar mixed finite element technique (MMMFE) for the Biot system of poroelasticity in a five-filed fully mixed formulation. This method is the generalization of the monolithic domain decomposition technique discussed in the previous chapter, with the extra capability to use non-matching subdomain grids at the interface. This capability is obtained by using composite multiscale mortar Lagrange multiplier spaces that approximates displacement and pressure on a coarse mortar grid at the interface. The global problem can be reduced into a series of parallel Dirichlet type problems and an interface problem for the composite displacement-pressure Lagrange multiplier spaces which requires subdomain solves at each iteration. We showed the well-posedness and stability of the method under proper assumptions. We have also carried out an extensive error analysis of the method to get a combined a priori error estimate for all the unknowns in the formulation. To complete the analysis, we have done a series of numerical experiments to put the theory into test. We observed stability and convergence results as predicted by the theory and also demonstrated the application of the method to a highly heterogeneous medium. We noted that in practice, a coarser mesh with higher mortar space degree can be used to get a smaller interface problem and hence faster convergence without compromising the accuracy of the method. We conclude the chapter by recalling the effectiveness of the construction and use of a pre-computed multiscale stress-flux basis (MSB), which makes the MMMFE method far more superior than the fine scale monolithic methods, especially when a coarse mortar mesh is used.

Table 13: Example 1, convergence table using linear mortar ($m = 1$) with $H = Ch$, $\Delta t = 10^{-4}$ and $c_0 = 1.0$.

h	# gmres		$\|\sigma - \sigma_h\|_{L^\infty(L^2)}$		$\|\operatorname{div}(\sigma - \sigma_h)\|_{L^\infty(L^2)}$		$\|\gamma - \gamma_h\|_{L^\infty(L^2)}$		$\|u - u_h\|_{L^\infty(L^2)}$	
1/4	16	rate	1.23e-01	rate	6.09e-01	rate	1.39e+00	rate	5.78e-01	rate
1/8	28	-0.81	3.24e-02	1.92	3.11e-01	0.97	7.07e-01	0.97	2.92e-01	0.99
1/16	46	-0.72	8.20e-03	1.98	1.56e-01	0.99	3.55e-01	0.99	1.46e-01	1.00
1/32	73	-0.67	2.08e-03	1.98	7.82e-02	1.00	1.78e-01	1.00	7.31e-02	1.00
1/64	122	-0.74	5.39e-04	1.94	3.91e-02	1.00	8.89e-02	1.00	3.65e-02	1.00

h	$\|z - z_h\|_{L^\infty(L^2)}$		$\|\operatorname{div}(z - z_h)\|_{L^2(L^2)}$		$\|p - p_h\|_{L^\infty(L^2)}$		$\|u - {\lambda^u}_H\|_{L^\infty(L^2)}$		$\|p - {\lambda^p}_H\|_{L^\infty(L^2)}$	
1/4	1.04e+00	rate	4.15e-01	rate	5.91e-02	rate	7.50e-01	rate	2.06e-01	rate
1/8	3.72e-01	1.48	1.89e-01	1.14	2.96e-02	1.00	1.90e-01	1.98	5.30e-02	1.96
1/16	1.19e-01	1.64	8.50e-02	1.15	1.48e-02	1.00	4.76e-02	1.99	1.33e-02	2.00
1/32	3.56e-02	1.74	3.97e-02	1.10	7.39e-03	1.00	1.19e-02	2.00	3.33e-03	2.00
1/64	1.08e-02	1.72	1.92e-02	1.05	3.70e-03	1.00	3.04e-03	1.97	8.37e-04	1.99

Table 14: Example 1, convergence table using quadratic mortar ($m = 2$) with $H = C\sqrt{h}$, $\Delta t = 10^{-4}$ and $c_0 = 1.0$.

h	# gmres		$\|\sigma - \sigma_h\|_{L^\infty(L^2)}$		$\|\operatorname{div}(\sigma - \sigma_h)\|_{L^\infty(L^2)}$		$\|\gamma - \gamma_h\|_{L^\infty(L^2)}$		$\|u - u_h\|_{L^\infty(L^2)}$	
1/4	22	rate	1.26e-01	rate	6.09e-01	rate	1.39e+00	rate	5.79e-01	rate
1/16	40	-0.43	8.25e-03	1.97	1.56e-01	0.98	3.55e-01	0.99	1.46e-01	0.99
1/64	65	-0.35	5.62e-04	1.93	3.91e-02	1.00	8.89e-02	1.00	3.65e-02	1.00

h	$\|z - z_h\|_{L^\infty(L^2)}$		$\|\operatorname{div}(z - z_h)\|_{L^2(L^2)}$		$\|p - p_h\|_{L^\infty(L^2)}$		$\|u - {\lambda^u}_H\|_{L^\infty(L^2)}$		$\|p - {\lambda^p}_H\|_{L^\infty(L^2)}$	
1/4	6.72e-01	rate	3.92e-01	rate	5.92e-02	rate	7.55e-01	rate	9.70e-02	rate
1/16	8.20e-02	1.52	8.36e-02	1.11	1.48e-02	1.00	4.82e-02	1.99	6.83e-03	1.91
1/64	7.03e-03	1.77	1.92e-02	1.06	3.70e-03	1.00	3.31e-03	1.93	5.91e-04	1.77

Table 15: Example 1, convergence table for linear mortar with $H = Ch$, $\Delta t = 10^{-4}$ and $c_0 = 10^{-3}$.

h	# gmres		$\|\sigma - \sigma_h\|_{L^\infty(L^2)}$		$\|\operatorname{div}(\sigma - \sigma_h)\|_{L^\infty(L^2)}$		$\|\gamma - \gamma_h\|_{L^\infty(L^2)}$		$\|u - u_h\|_{L^\infty(L^2)}$	
1/4	16	rate	1.25e-01	rate	6.09e-01	rate	1.39e+00	rate	5.78e-01	rate
1/8	29	-0.86	3.30e-02	1.92	3.11e-01	0.97	7.07e-01	0.97	2.92e-01	0.99
1/16	50	-0.79	8.34e-03	1.98	1.56e-01	0.99	3.55e-01	0.99	1.46e-01	1.00
1/32	87	-0.80	2.09e-03	1.99	7.82e-02	1.00	1.78e-01	1.00	7.31e-02	1.00
1/64	157	-0.85	5.38e-04	1.96	3.91e-02	1.00	8.89e-02	1.00	3.65e-02	1.00

h	$\|z - z_h\|_{L^\infty(L^2)}$		$\|\operatorname{div}(z - z_h)\|_{L^2(L^2)}$		$\|p - p_h\|_{L^\infty(L^2)}$		$\|u - \lambda^u{}_H\|_{L^\infty(L^2)}$		$\|p - \lambda^p{}_H\|_{L^\infty(L^2)}$	
1/4	4.18e+01	rate	2.31e+00	rate	8.81e-01	rate	7.52e-01	rate	8.48e+00	rate
1/8	9.68e+00	2.11	7.14e-01	1.69	2.33e-01	1.92	1.90e-01	1.98	2.11e+00	2.00
1/16	2.31e+00	2.07	2.00e-01	1.84	5.93e-02	1.98	4.77e-02	1.99	5.08e-01	2.06
1/32	5.68e-01	2.02	6.02e-02	1.73	1.62e-02	1.87	1.19e-02	2.00	1.25e-01	2.02
1/64	1.42e-01	2.00	2.22e-02	1.44	5.22e-03	1.64	2.98e-03	2.00	3.12e-02	2.00

Table 16: Example 1, convergence table for quadratic mortar with $H = C\sqrt{h}$, $\Delta t = 10^{-4}$ and $c_0 = 10^{-3}$.

h	# gmres		$\|\sigma - \sigma_h\|_{L^\infty(L^2)}$		$\|\operatorname{div}(\sigma - \sigma_h)\|_{L^\infty(L^2)}$		$\|\gamma - \gamma_h\|_{L^\infty(L^2)}$		$\|u - u_h\|_{L^\infty(L^2)}$	
1/4	23	rate	1.28e-01	rate	6.09e-01	rate	1.39e+00	rate	5.79e-01	rate
1/16	41	-0.41	8.39e-03	1.97	1.56e-01	0.98	3.55e-01	0.96	1.46e-01	0.99
1/64	72	-0.41	5.61e-04	1.95	3.91e-02	1.00	8.89e-02	1.00	3.65e-02	1.00

h	$\|z - z_h\|_{L^\infty(L^2)}$		$\|\operatorname{div}(z - z_h)\|_{L^2(L^2)}$		$\|p - p_h\|_{L^\infty(L^2)}$		$\|u - \lambda^u{}_H\|_{L^\infty(L^2)}$		$\|p - \lambda^p{}_H\|_{L^\infty(L^2)}$	
1/4	4.24e+01	rate	2.42e+00	rate	9.97e-01	rate	7.57e-01	rate	1.07e+01	rate
1/16	2.33e+00	2.01	2.01e-01	1.79	6.01e-02	2.06	4.83e-02	1.98	5.17e-01	2.19
1/64	1.50e-01	1.97	2.25e-02	1.58	5.40e-03	1.74	3.26e-03	1.95	3.38e-02	1.97

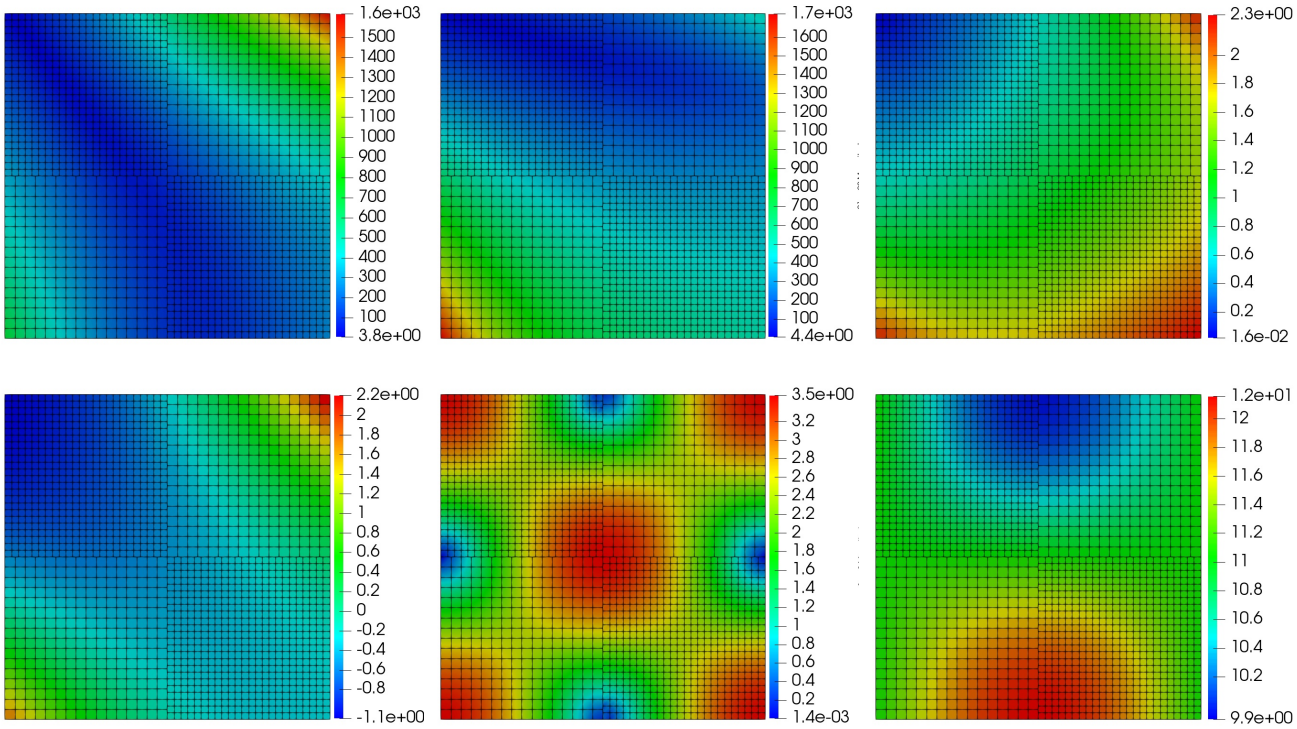

Figure 5: Example 1, computed solution at final time step using a linear mortar on non-matching subdomain grids, top: stress x (left), stress y (middle), displacement (right), bottom: rotation (left), velocity (middle), pressure (right). Mesh size, $h = 1/32$, $\Delta t = 10^{-3}$ and $c_0 = 1.0$.

Table 17: Example 2, parameters (top) and boundary conditions (bottom).

Parameter	Value
Mass storativity (c_0)	1.0
Biot-Willis constant (α)	1.0
Time step (Δt)	10^{-3}
Total time (T)	0.1

Boundary	σ	u	p	z
Left	$\sigma n = -\alpha p n$	-	1	-
Bottom	$\sigma n = 0$	-	-	$z \cdot n = 0$
Right	$-$	0	0	-
Top	$\sigma n = 0$	-	-	$z \cdot n = 0$

Table 18: Example 2, #GMRES iterations and maximum number of subdomain solves.

mortar	Average #GMRES	Total #GMRES	Total #Solves	
			No MSB	MSB
linear fine scale	343	34375	34575	968
1 linear per interface	41	4149	4349	224
1 quadratic per interface	61	6184	6384	236
2 linear per interface	80	8010	8210	248
2 quadratic per interface	123	12302	12502	272

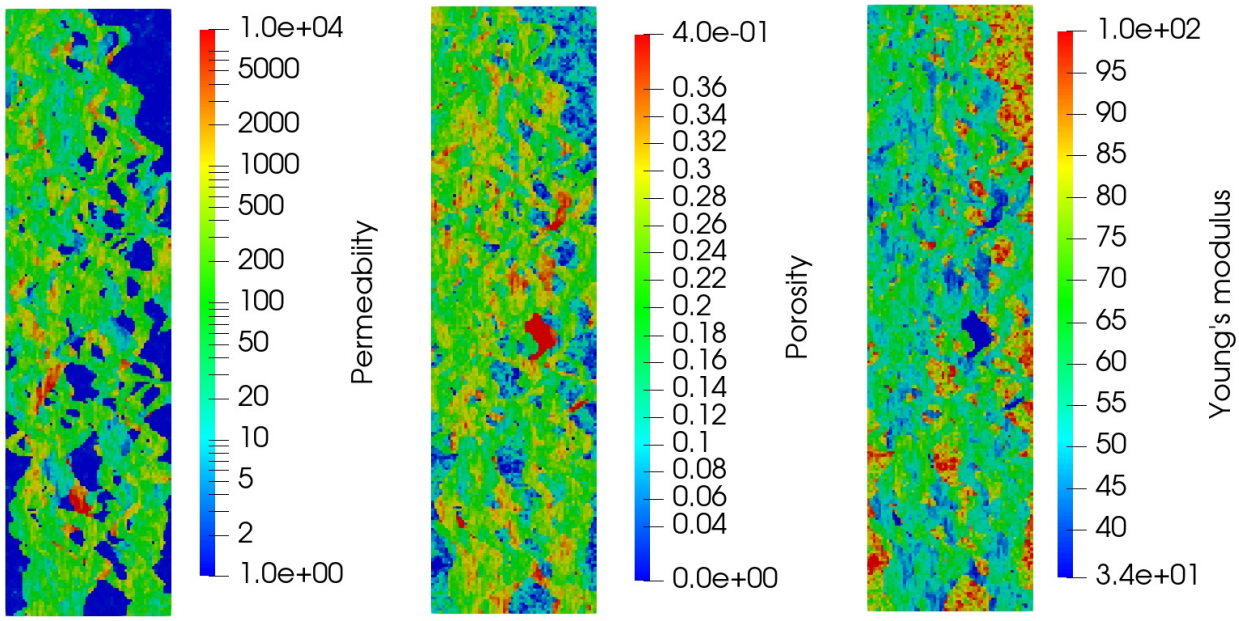

Figure 6: Example 2, permeability, porosity, Young's modulus.

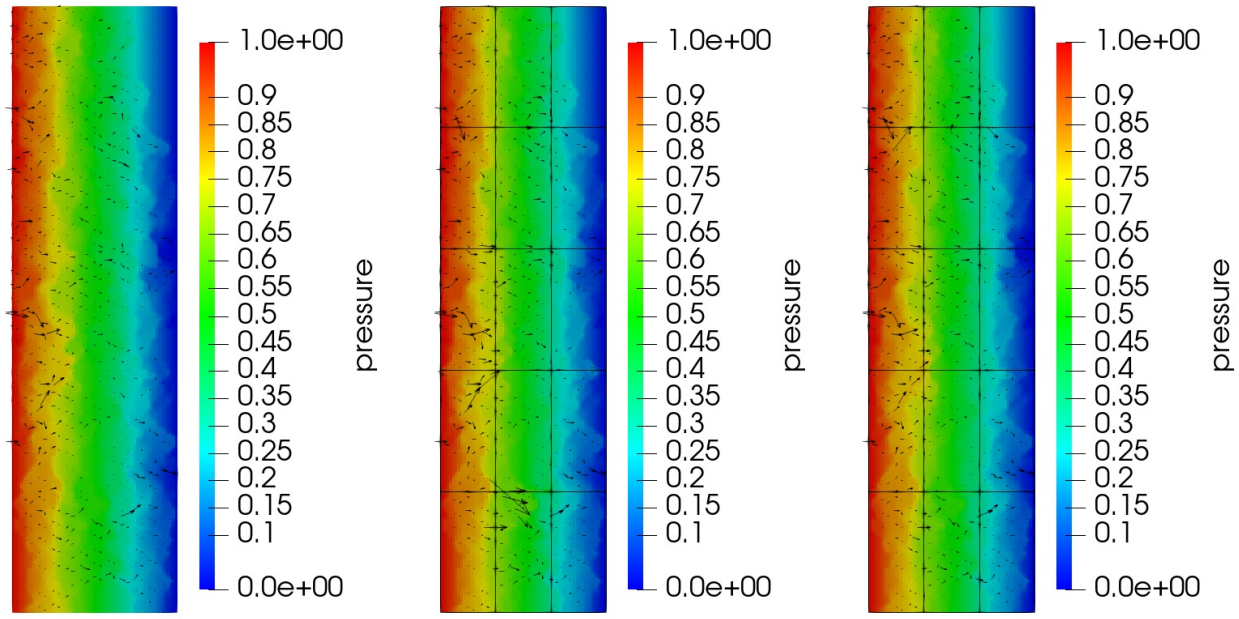

Figure 7: Example 2, pressure (color) and velocity (arrows): fine scale (left), single linear mortar per interface (middle), and two linear mortars per interface (right).

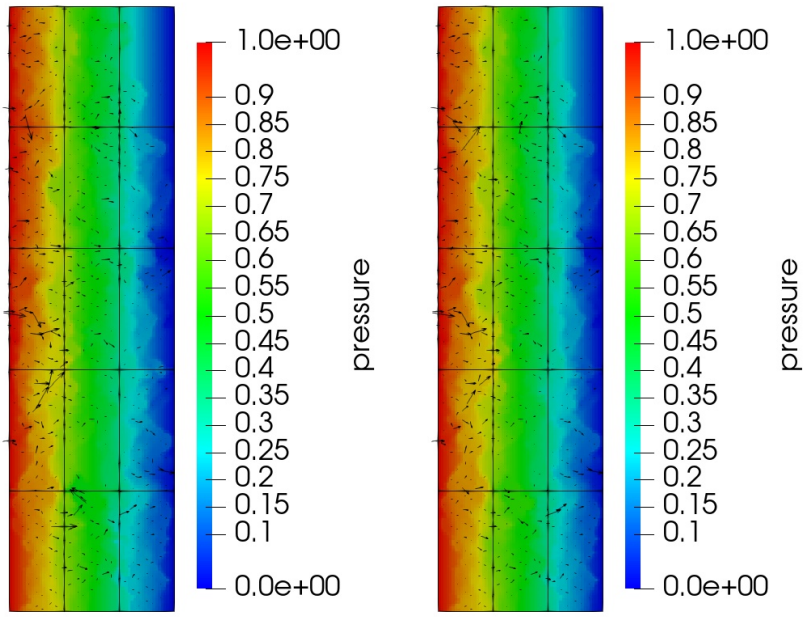

Figure 8: Example 2, pressure (color) and velocity (arrows): single quadratic mortar per interface (left), and two quadratic mortars per interface (right).

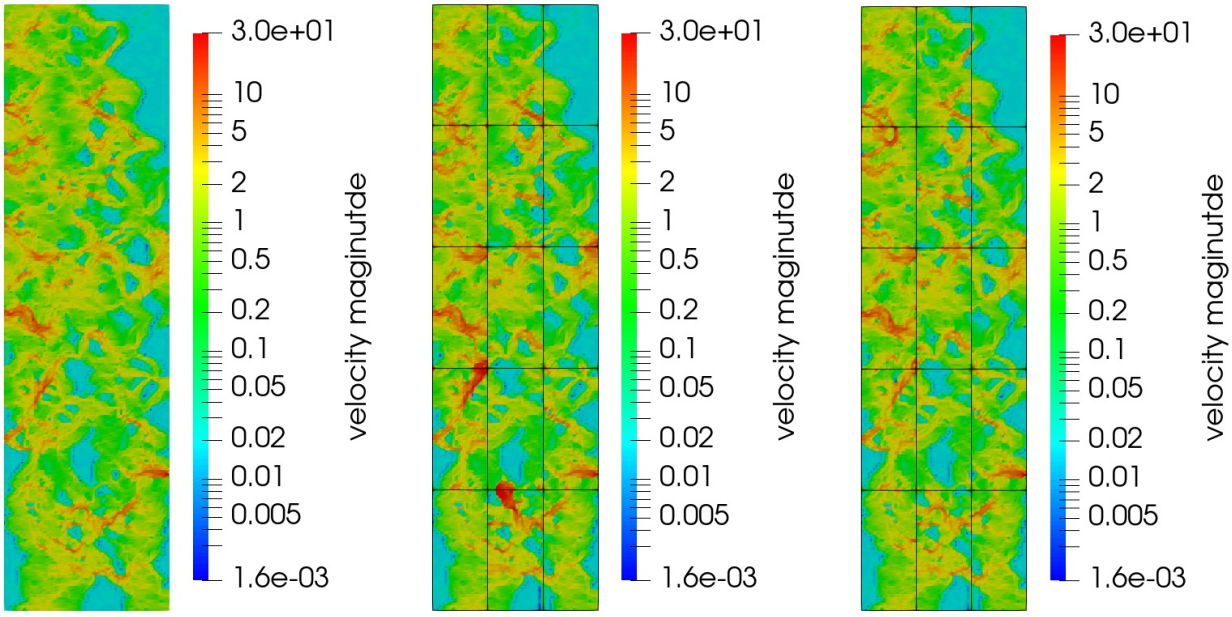

Figure 9: Example 2, velocity magnitude: fine scale (left), single linear mortar per interface (middle), and two linear mortars per interface (right).

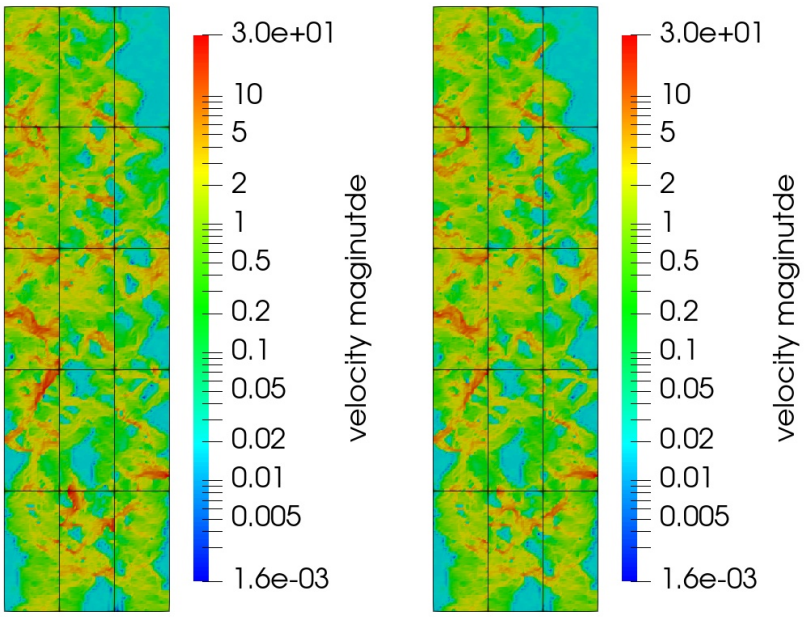

Figure 10: Example 2, velocity magnitude: single quadratic mortar per interface (left), and two quadratic mortars per interface (right).

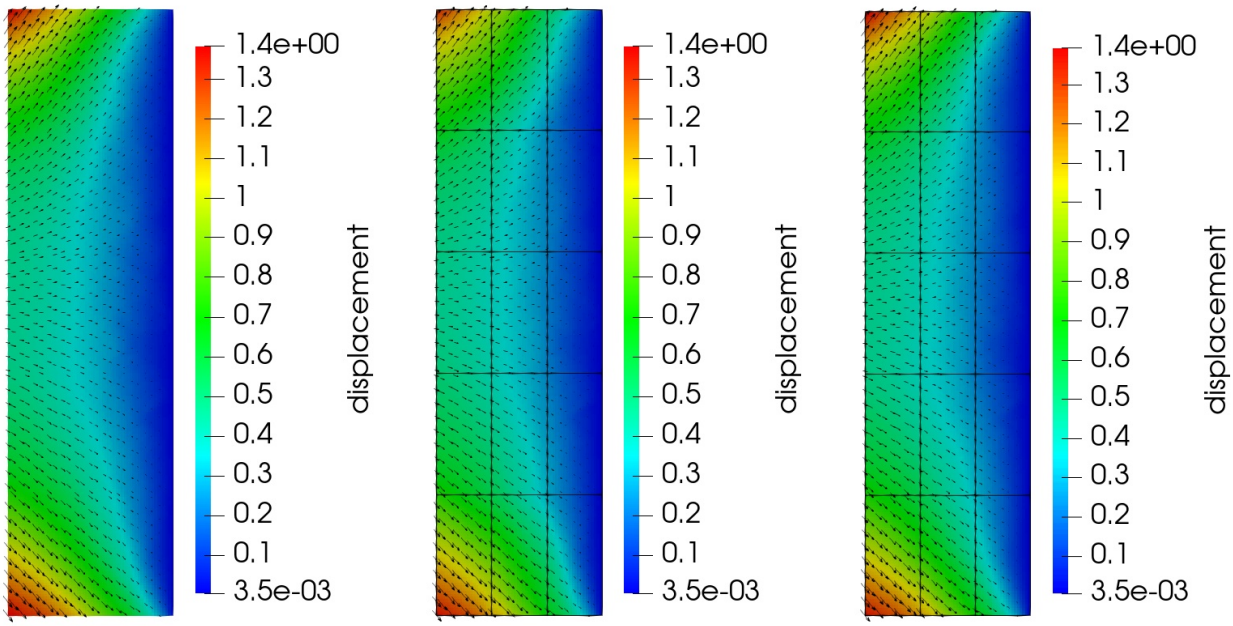

Figure 11: Example 2, displacement vector (arrows) and its magnitude: fine scale (left), single linear mortar per interface (middle), and two linear mortars per interface (right).

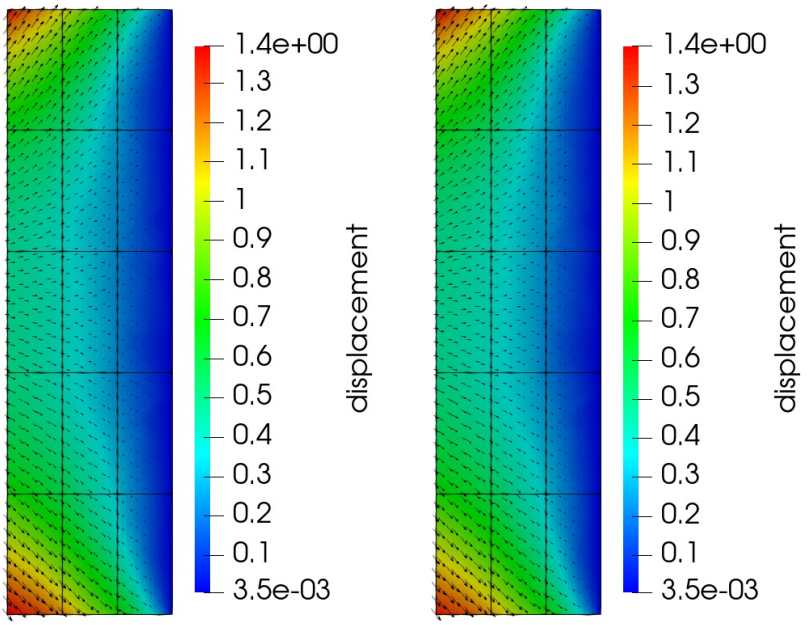

Figure 12: Example 2, displacement vector (arrows) and its magnitude: single quadratic mortar per interface (left), and two quadratic mortars per interface (right).

4.0 A Multiscale Mortar Space-time Domain Decomposition Technique For Parabolic Equations

4.1 Introduction

In this chapter, we study a more general version of multiscale mortar mixed finite element (MMMFE) technique discussed in Chapter 3, where we allow multiscale discretization in both time and space.

As usual, we divide the global domain Ω into a union of non-overlapping subdomains Ω_i. For each subdomain Ω_i, our approach considers an individual space mesh of Ω_i along with individual time steps on $(0,T]$. On each *space-time subdomain* $\Omega_i \times (0,T]$, any standard mixed finite element scheme is combined with the discontinuous Galerkin time discretization. Then a stand-alone mortar variable is introduced, on an independent *interface space-time mesh* which is typically coarse and where possibly higher polynomial degrees are used. This is then used to couple the space-time subdomain problems and to ensure (a multiscale) weak continuity of the normal component of the mixed finite element flux variable over the space-time interfaces. This setting allows for high flexibility with individual discretizations of each space-time subdomain $\Omega_i \times (0,T]$, and in particular for *local time stepping*, individually in each space-time subdomain $\Omega_i \times (0,T]$. Moreover, *space-time parallelization* can be achieved, leading to solution of discrete problems on individual space-time subdomains $\Omega_i \times (0,T]$, exchanging space-time boundary data through transmission conditions.

Remaining part of the chapter is organized as follows. In Section 4.2, we describe the model problem and basic notation. Our space-time multiscale mortar discretization is introduced in Section 4.3, and we prove its existence, uniqueness, and stability with respect to data in Section 4.4. Section 4.5 then derives a priori error estimates. We rewrite equivalently our method under a form of a space-time interface problem for the mortar variable in Section 4.6, which in particular allows for the space-time parallelization. We finally present numerical illustrations in Section 4.7. Future works may include as well as deriving a posteriori error estimates, possibly building upon the ideas from [86, 87, 102].

4.2 Model Problem and Space-Time Domain Decomposition Formulation

4.2.1 Model problem

We consider a parabolic partial differential equation in a mixed form, modeling single phase flow in porous media. Let $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, be a spatial polytopal domain with Lipschitz boundary and let $(0, T]$ be a time interval. The governing equations are

$$\mathbf{u} = -K\nabla p, \quad \frac{\partial p}{\partial t} + \nabla \cdot \mathbf{u} = q \quad \text{in } \Omega \times (0, T], \tag{4.2.1a}$$

where p is the fluid pressure, $\mathbf{u}$ is the Darcy velocity, q is a source term, and K is a tensor representing the rock permeability divided by the fluid viscosity. We assume for simplicity the homogeneous Dirichlet boundary condition

$$p(x, t) = 0 \text{ on } \partial\Omega \times (0, T] \tag{4.2.1b}$$

and assign the initial pressure

$$p(x, 0) = p_0(x) \text{ on } \Omega. \tag{4.2.1c}$$

We assume that $q \in L^2(0, T; L^2(\Omega))$, $p_0 \in H_0^1(\Omega)$, $\nabla \cdot K\nabla p_0 \in L^2(\Omega)$, and that K is spatially-dependent, uniformly bounded, symmetric, and positive definite tensor, i.e., for constants $0 < k_{\min} \leq k_{\max} < \infty$,

$$\forall \text{ a.e. } x \in \Omega, \quad k_{\min}\zeta^T\zeta \leq \zeta^T K(x)\zeta \leq k_{\max}\zeta^T\zeta \quad \forall \zeta \in \mathbb{R}^d. \tag{4.2.2}$$

Moreover, suppose a scaling such that the diameter of Ω and the final time T are of order one.

4.2.2 Space-time subdomains

Let Ω be a union of non-overlapping polytopal subdomains with Lipschitz boundary, $\overline{\Omega} = \cup\overline{\Omega}_i$. Let $\Gamma_i = \partial\Omega_i \setminus \partial\Omega$ be the interior boundary of Ω_i, let $\Gamma_{ij} = \Gamma_i \cap \Gamma_j$ be the interface between two adjacent subdomains Ω_i and Ω_j, and let $\Gamma = \cup\Gamma_{ij}$ be the union of all subdomain interfaces. We also introduce the space-time counterparts $\Omega^T = \Omega \times (0, T)$, $\Omega_i^T = \Omega_i \times (0, T)$, $\Gamma_i^T = \Gamma_i \times (0, T)$, and $\Gamma_{ij}^T = \Gamma_{ij} \times (0, T)$. We will introduce space-time domain decomposition discretizations based on Ω_i^T.

4.2.3 Basic notation

We will utilize the following notation. For a domain $\mathcal{O} \subset \mathbb{R}^d$, the $L^2(\mathcal{O})$ inner product and norm for scalar and vector-valued functions are denoted by $(\cdot,\cdot)_{\mathcal{O}}$ and $\|\cdot\|_{\mathcal{O}}$, respectively. The norms and seminorms of the Sobolev spaces $W^{k,p}(\mathcal{O})$, $k \in \mathbb{R}, p > 0$ are denoted by $\|\cdot\|_{k,p,\mathcal{O}}$ and $|\cdot|_{k,p,\mathcal{O}}$, respectively. The norms and seminorms of the Hilbert spaces $H^k(\mathcal{O})$ are denoted by $\|\cdot\|_{k,\mathcal{O}}$ and $|\cdot|_{k,\mathcal{O}}$, respectively. For a section of a subdomain boundary $S \subset \mathbb{R}^{d-1}$ we write $\langle\cdot,\cdot\rangle_S$ and $\|\cdot\|_S$ for the $L^2(S)$ inner product (or duality pairing) and norm, respectively. By $\mathbf{M}$ we denote the vectorial counterpart of a generic scalar space M.

The above notation is extended to space-time domains as follows. For $\mathcal{O}^T = \mathcal{O} \times (0,T)$ and $S^T = S \times (0,T)$, let $(\cdot,\cdot)_{\mathcal{O}^T} = \int_0^T (\cdot,\cdot)_{\mathcal{O}}$ and $\langle\cdot,\cdot\rangle_{S^T} = \int_0^T \langle\cdot,\cdot\rangle_S$. For space-time norms we use the standard Bochner notation. For example, given a spatial norm $\|\cdot\|_V$, we denote, for $p > 0$,

$$\|\cdot\|_{L^p(0,T;V)} = \left(\int_0^T \|\cdot\|_V^p\right)^{\frac{1}{p}}, \quad \|\cdot\|_{L^\infty(0,T;V)} = \operatorname{ess\,sup} \|\cdot\|_V,$$

with the usual extension for $\|\cdot\|_{W^{k,p}(0,T;V)}$ and $\|\cdot\|_{H^k(0,T;V)}$. We will also use the notation $\|\cdot\|_{S^T} = \|\cdot\|_{L^2(0,T;L^2(S))}$. Finally, we will use the space

$$\mathbf{H}(\operatorname{div};\mathcal{O}) = \left\{\mathbf{v} \in \mathbf{L}^2(\mathcal{O}) : \nabla\cdot\mathbf{v} \in L^2(\mathcal{O})\right\},$$

equipped with the norm

$$\|\mathbf{v}\|_{\operatorname{div};\mathcal{O}} = \left(\|\mathbf{v}\|_{\mathcal{O}}^2 + \|\nabla\cdot\mathbf{v}\|_{\mathcal{O}}^2\right)^{\frac{1}{2}}.$$

4.2.4 Weak formulation

The weak formulation of problem (4.2.1) reads: find $(\mathbf{u},p) : [0,T] \mapsto \mathbf{H}(\operatorname{div};\Omega) \times L^2(\Omega)$ such that $p(x,0) = p_0$ and for a.e. $t \in (0,T)$,

$$(K^{-1}\mathbf{u},\mathbf{v})_\Omega - (p,\nabla\cdot\mathbf{v})_\Omega = 0 \quad \forall\,\mathbf{v} \in \mathbf{H}(\operatorname{div};\Omega), \tag{4.2.3a}$$

$$(\partial_t p, w)_\Omega + (\nabla\cdot\mathbf{u}, w)_\Omega = (q,w)_\Omega \quad \forall\, w \in L^2(\Omega). \tag{4.2.3b}$$

The following well-posedness result is rather standard and presented in, e.g., [112, Theorem 2.1].

Theorem 4.2.1 (Well-posedness). *Problem (4.2.3) has a unique solution* $\mathbf{u} \in L^2(0,T;\mathbf{H}(\operatorname{div};\Omega)) \cap L^\infty(0,T;\mathbf{L}^2(\Omega))$, $p \in H^1(0,T;H_0^1(\Omega))$.

We note that in particular the inclusion $p \in H^1(0,T;H_0^1(\Omega))$ follows from (4.2.3a), which implies that for a.e. $t \in (0,T)$, $\nabla p = -K^{-1}\mathbf{u}$ in a sense of distributions.

4.2.5 Domain decomposition weak formulation

We now give a domain decomposition weak formulation of (4.2.3). Introduce the subdomain velocity and pressure spaces

$$\mathbf{V}_i = \mathbf{H}(\mathrm{div};\Omega_i),\ \mathbf{V} = \bigoplus \mathbf{V}_i, \quad W_i = L^2(\Omega_i),\ W = \bigoplus W_i = L^2(\Omega),$$

endowed with the norms

$$\|\mathbf{v}\|_{\mathbf{V}_i} = \|\mathbf{v}\|_{\mathrm{div};\Omega_i}, \quad \|\mathbf{v}\|_{\mathbf{V}} = \left(\sum_i \|\mathbf{v}\|^2_{\mathbf{V}_i}\right)^{\frac{1}{2}}, \quad \|w\|_W = \|w\|_\Omega.$$

We also introduce the following spatial bilinear forms, which will turn useful below:

$$a_i(\mathbf{u},\mathbf{v}) = (K^{-1}\mathbf{u},\mathbf{v})_{\Omega_i}, \quad a(\mathbf{u},\mathbf{v}) = \sum_i a_i(\mathbf{u},\mathbf{v}), \tag{4.2.4a}$$

$$b_i(\mathbf{v},w) = -(\nabla\cdot\mathbf{v},w)_{\Omega_i}, \quad b(\mathbf{v},w) = \sum_i b_i(\mathbf{v},w), \tag{4.2.4b}$$

$$b_\Gamma(\mathbf{v},\mu) = \sum_i \langle \mathbf{v}\cdot\mathbf{n}_i,\mu\rangle_{\Gamma_i}, \tag{4.2.4c}$$

In addition, for any spatial bilinear form $s(\cdot,\cdot)$, let $s^T(\cdot,\cdot) = \int_0^T s(\cdot,\cdot)$.

Now, since $p \in H^1(0,T;H^1_0(\Omega))$, we can consider the trace of the pressure p on the interfaces, $\lambda = p|_\Gamma$. Thus, integrating in time, it is easy to see that the solution $(\mathbf{u},p)$ of (4.2.3) satisfies

$$a^T(\mathbf{u},\mathbf{v}) + b^T(\mathbf{v},p) + b^T_\Gamma(\mathbf{v},\lambda) = 0 \quad \forall\,\mathbf{v}\in L^2(0,T;\mathbf{V}), \tag{4.2.5a}$$

$$(\partial_t p, w)_{\Omega^T} - b^T(\mathbf{u},w) = (q,w)_{\Omega^T} \quad \forall\, w\in L^2(0,T;W). \tag{4.2.5b}$$

4.3 Space-Time Mixed Finite Element Method

We consider a space-time discretization of (4.2.5), motivated by [112]. It employs mortar finite elements to approximate the pressure trace λ from (4.2.5) and uses it as a Lagrange multiplier to impose weakly the continuity of flux across space-time interfaces.

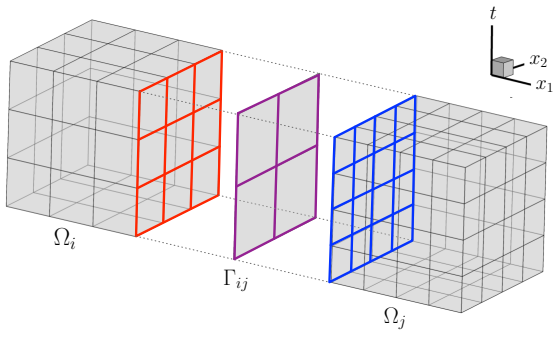

Figure 13: Non-matching space-time subdomain and mortar grids in two spatial dimensions.

4.3.1 Space-time grids and spaces

Let $\mathcal{T}_{h,i}$ be a shape-regular partition of the subdomain Ω_i into parallelepipeds or simplices in the sense of [99]. We stress that this allows for grids that do not match along the interfaces Γ_{ij} between subdomains Ω_i and Ω_j. Similarly, let $\mathcal{T}_i^{\Delta t} : 0 = t_i^0 < t_i^1 < \cdots < t_i^{N_i} = T$ be a partition of the time interval $(0, T)$ corresponding to subdomain Ω_i; this means that we consider different time discretizations on different subdomains. Let $h = \max_i \max_{E \in \mathcal{T}_{h,i}} \operatorname{diam} E$ and $\Delta t = \max_i \max_{1 \le k \le N_i} |t_i^k - t_i^{k-1}|$ be respectively the space and time mesh sizes. Composing $\mathcal{T}_{h,i}$ and $\mathcal{T}_i^{\Delta t}$ by tensor product results in a space-time partition

$$\mathcal{T}_{h,i}^{\Delta t} = \mathcal{T}_{h,i} \times \mathcal{T}_i^{\Delta t}$$

of the space-time subdomain Ω_i^T. An illustration is given in Figure 13, where yet a different, mortar space-time grid, is also shown in the middle.

For discretization in space, we consider any of the inf–sup stable mixed finite element spaces $\mathbf{V}_{h,i} \times W_{h,i} \subset \mathbf{V}_i \times W_i$ such as the Raviart–Thomas (RT) or the Brezzi–Douglas–Marini (BDM) spaces, see, e.g., [98]. For discretization in time, we will in turn utilize the discontinuous Galerkin (DG) method, cf. [126], which is based on a discontinuous piecewise polynomial approximation of the solution on the mesh $\mathcal{T}_i^{\Delta t}$. Denote by $\mathbf{V}_i^{\Delta t} = [W_i^{\Delta t}]^d$ and $W_i^{\Delta t}$ the subdomain time discretizations of the velocity and pressure, respectively. Composing the space and time discretizations

$$\mathbf{V}_{h,i}^{\Delta t} = \mathbf{V}_{h,i} \times \mathbf{V}_i^{\Delta t}, \quad W_{h,i}^{\Delta t} = W_{h,i} \times W_i^{\Delta t}$$

results in the space-time mixed finite element spaces $\mathbf{V}_{h,i}^{\Delta t} \times W_{h,i}^{\Delta t}$ in each space-time subdomain Ω_i^T. We will also need the spatial variable only space

$$W_h = \bigoplus W_{h,i}.$$

Let $\mathcal{T}_{H,ij}$ be a finite element partition of Γ_{ij}, where $H = \max_{i,j} \max_{e \in \mathcal{T}_{H,ij}} \operatorname{diam} e$, see Figure 13, middle. The use of index H indicates a possibly coarser interface grid compared to the subdomain grids, resulting in a multiscale approximation. Let $\mathcal{T}_{ij}^{\Delta T} : 0 = t_{ij}^0 < t_{ij}^1 < \cdots < t_{ij}^{N_{ij}} = T$ be a partition of $(0, T)$ corresponding to Γ_{ij}, which may be different from (and again possibly coarser than) the time-partitions for the neighboring subdomains. Let $\Delta T = \max_{i,j} \max_{1 \le k \le N_{ij}} |t_{ij}^k - t_{ij}^{k-1}|$. Composing $\mathcal{T}_{H,ij}$ and $\mathcal{T}_{ij}^{\Delta T}$ by tensor product gives a space-time partition

$$\mathcal{T}_{H,ij}^{\Delta T} = \mathcal{T}_{H,ij} \times T_{ij}^{\Delta T}$$

of the space-time interface Γ_{ij}^T. Finally, let

$$\Lambda_{H,ij}^{\Delta T} = \Lambda_{H,ij} \times \Lambda_{ij}^{\Delta T}$$

be a space-time mortar finite element space on $\mathcal{T}_{H,ij}^{\Delta T}$ consisting of continuous or discontinuous piecewise polynomials in space and in time. We will also need the spatial variable only space

$$\Lambda_H = \bigoplus \Lambda_{H,ij}.$$

Finally, the global space-time finite element spaces are defined as

$$\mathbf{V}_h^{\Delta t} = \bigoplus \mathbf{V}_{h,i}^{\Delta t}, \quad W_h^{\Delta t} = \bigoplus W_{h,i}^{\Delta t}, \quad \Lambda_H^{\Delta T} = \bigoplus \Lambda_{H,ij}^{\Delta T}. \tag{4.3.1}$$

In particular, the Lagrange multiplier will be sought for in the mortar space $\Lambda_H^{\Delta T}$. For the purpose of the analysis, we also define the space of velocities with space-time weakly continuous normal components

$$\mathbf{V}_{h,0}^{\Delta t} = \left\{ \mathbf{v} \in \mathbf{V}_h^{\Delta t} : b_\Gamma(\mathbf{v}, \mu) = 0 \quad \forall\, \mu \in \Lambda_H^{\Delta T} \right\}. \tag{4.3.2}$$

The discrete velocity and pressure spaces inherit the norms $\|\cdot\|_{\mathbf{V}}$ and $\|\cdot\|_W$, respectively. The mortar space is equipped with the spatial norm $\|\mu\|_{\Lambda_H} = \|\mu\|_{L^2(\Gamma)}$.

4.3.2 Space-time multiscale mortar mixed finite element method

For the DG time discretization, we introduce the notation for $p_h^{\Delta t}$, $w \in W_h^{\Delta t}$, see [126],

$$\begin{aligned}\int_0^T (\tilde{\partial}_t p_h^{\Delta t}, w)_{\Omega_i} &= \sum_{k=1}^{N_i} \int_{t_i^{k-1}}^{t_i^k} (\partial_t p_h^{\Delta t}, w)_{\Omega_i} + \sum_{k=1}^{N_i} ([p_h^{\Delta t}]_{k-1}, w_{k-1}^+)_{\Omega_i} \\ &\equiv \int_0^T (\hat{\partial}_t p_h^{\Delta t}, w)_{\Omega_i} + \sum_{k=1}^{N_i} ([p_h^{\Delta t}]_{k-1}, w_{k-1}^+)_{\Omega_i},\end{aligned} \tag{4.3.3}$$

where $\hat{\partial}_t$ denotes the step-wise time derivative and $[w]_k = w_k^+ - w_k^-$, with $w_k^+ = \lim_{t \to t_i^{k,+}} w$ and $w_k^- = \lim_{t \to t_i^{k,-}} w$. We note that the last term for $k = 1$ is $\left((p_h^{\Delta t})_0^+ - (p_h^{\Delta t})_0^-, w_0^+\right)_{\Omega_i}$. Here, $(p_h^{\Delta t})_0^+$ is computed by the method, while $(p_h^{\Delta t})_0^-$ is determined by the initial condition. More precisely, we will take as initial data $(p_h^{\Delta t})_0^- = \mathcal{P}_h p_0$, where $\mathcal{P}_h$ is the L^2-orthogonal projection onto W_h.

Remark 4.3.1 (Initial value)**.** *In what follows, we will tacitly assume that a function $w \in W_h^{\Delta t}$ has an associated initial value w_0^-, which will be defined if it is explicitly used.*

The space-time multiscale mortar mixed finite element method for approximating (4.2.5) is: find $\mathbf{u}_h^{\Delta t} \in \mathbf{V}_h^{\Delta t}$, $p_h^{\Delta t} \in W_h^{\Delta t}$, and $\lambda_H^{\Delta T} \in \Lambda_H^{\Delta T}$ such that $(p_h^{\Delta t})_0^- = \mathcal{P}_h p_0$ and

$$a(\mathbf{u}_h^{\Delta t}, \mathbf{v}) + b(\mathbf{v}, p_h^{\Delta t}) + b_\Gamma(\mathbf{v}, \lambda_H^{\Delta T}) = 0 \quad \forall\, \mathbf{v} \in \mathbf{V}_h^{\Delta t}, \tag{4.3.4a}$$

$$(\tilde{\partial}_t p_h^{\Delta t}, w)_{\Omega^T} - b(\mathbf{u}_h^{\Delta t}, w) = (q, w)_{\Omega^T} \quad \forall\, w \in W_h^{\Delta t}, \tag{4.3.4b}$$

$$b_\Gamma(\mathbf{u}_h^{\Delta t}, \mu) = 0 \quad \forall\, \mu \in \Lambda_H^{\Delta T}, \tag{4.3.4c}$$

where the obvious notation $(\tilde{\partial}_t p_h^{\Delta t}, w)_{\Omega^T} = \sum_i (\tilde{\partial}_t p_h^{\Delta t}, w)_{\Omega_i^T}$ has been used.

The above method provides a highly general and flexible framework, allowing for different spatial and temporal discretizations in different subdomains. We note that according to (4.3.4c), continuity of the flux is imposed weakly on the space-time interfaces Γ_{ij}^T, requiring that the jump in flux is orthogonal to the space-time mortar space $\Lambda_{H,ij}^{\Delta T}$. This formulation results in a correct notion of mass conservation across interfaces for time-dependent domain decomposition problems with non-matching grids in both space and time. In the case of discontinuous mortars, (4.3.4c) implies that the total flux across any space-time interface cell $e \times (t_{ij}^{k-1}, t_{ij}^k)$, $e \in \mathcal{T}_{H,ij}$, is continuous.

4.4 Well-Posedness Analysis

In this section we analyze the existence, uniqueness, and stability of the solution to (4.3.4).

4.4.1 Space-time interpolants

We will make use of several space-time interpolants. Let $\mathcal{P}_{h,i}$ be the L^2-orthogonal projection onto $W_{h,i}$ and let $\mathcal{P}_i^{\Delta t}$ be the L^2-orthogonal projection onto $W_i^{\Delta t}$. We then define the L^2-orthogonal projection in space and time on subdomain Ω_i by

$$\mathcal{P}_{h,i}^{\Delta t} = \mathcal{P}_{h,i} \times \mathcal{P}_i^{\Delta t} : L^2(0,T;L^2(\Omega_i)) \to W_{h,i}^{\Delta t}$$

and globally by

$$\mathcal{P}_h^{\Delta t} : L^2(0,T;L^2(\Omega)) \to W_h^{\Delta t}, \quad \mathcal{P}_h^{\Delta t}|_{\Omega_i} = \mathcal{P}_{h,i}^{\Delta t}.$$

Setting $\mathcal{P}^{\Delta t}|_{\Omega_i} = \mathcal{P}_i^{\Delta t}$, we will also write $\mathcal{P}_h^{\Delta t} = \mathcal{P}_h \times \mathcal{P}^{\Delta t}$. Since $\nabla \cdot \mathbf{V}_{h,i} = W_{h,i}$, we have, for all $\varphi \in L^2(0,T;L^2(\Omega_i))$,

$$(\mathcal{P}_h^{\Delta t}\varphi - \varphi, \nabla \cdot \mathbf{v})_{\Omega_i^T} = 0 \quad \forall\, \mathbf{v} \in \mathbf{V}_{h,i}^{\Delta t}. \tag{4.4.1}$$

For $\epsilon > 0$, denote $\mathbf{H}^\epsilon(\mathrm{div};\Omega_i) := \mathbf{H}^\epsilon(\Omega_i) \cap \mathbf{H}(\mathrm{div};\Omega_i)$. Let $\Pi_{h,i} : \mathbf{H}^\epsilon(\mathrm{div};\Omega_i) \to \mathbf{V}_{h,i}$ be the canonical mixed interpolant [98] and let

$$\Pi_{h,i}^{\Delta t} = \Pi_{h,i} \times \mathcal{P}_i^{\Delta t} : L^2(0,T;\mathbf{H}^\epsilon(\mathrm{div};\Omega_i)) \to \mathbf{V}_{h,i}^{\Delta t}.$$

In particular, this space-time interpolant satisfies, for all $\boldsymbol{\psi} \in L^2(0,T;\mathbf{H}^\epsilon(\mathrm{div};\Omega_i))$,

$$(\nabla \cdot (\Pi_{h,i}^{\Delta t}\boldsymbol{\psi} - \boldsymbol{\psi}), w)_{\Omega_i^T} = 0 \quad \forall\, w \in W_{h,i}^{\Delta t}, \tag{4.4.2a}$$

$$\langle (\Pi_{h,i}^{\Delta t}\boldsymbol{\psi} - \boldsymbol{\psi}) \cdot \mathbf{n}_i, \mathbf{v} \cdot \mathbf{n}_i \rangle_{\partial\Omega_i^T} = 0 \quad \forall\, \mathbf{v} \in \mathbf{V}_{h,i}^{\Delta t}, \tag{4.4.2b}$$

$$\|\Pi_{h,i}^{\Delta t}\boldsymbol{\psi}\|_{L^2(0,T;\mathbf{V}_i)} \le C(\|\boldsymbol{\psi}\|_{L^2(0,T;\mathbf{H}^\epsilon(\Omega_i))} + \|\nabla \cdot \boldsymbol{\psi}\|_{L^2(0,T;L^2(\Omega_i))}). \tag{4.4.2c}$$

Let $\mathcal{Q}_{h,i} : L^2(\partial\Omega_i) \to \mathbf{V}_{h,i} \cdot \mathbf{n}_i$ be the L^2-orthogonal projection and let

$$\mathcal{Q}_{h,i}^{\Delta t} = \mathcal{Q}_{h,i} \times \mathcal{P}_i^{\Delta t} : L^2(0,T;L^2(\partial\Omega_i)) \to \mathbf{V}_{h,i}^{\Delta t} \cdot \mathbf{n}_i. \tag{4.4.3}$$

Finally, let $\mathcal{P}_{H,\Gamma_{ij}} : L^2(\Gamma_{ij}) \to \Lambda_{H,ij}$ and $\mathcal{P}_{ij}^{\Delta T} : L^2(0,T) \to \Lambda_{ij}^{\Delta T}$ be the L^2-orthogonal projections and let

$$\mathcal{P}_{H,\Gamma_{ij}}^{\Delta T} = \mathcal{P}_{H,\Gamma_{ij}} \times \mathcal{P}_{ij}^{\Delta T} : L^2(0,T;L^2(\Gamma_{ij})) \to \Lambda_{H,ij}^{\Delta T}, \quad \mathcal{P}_{H,\Gamma}^{\Delta T}|_{\Gamma_{ij}} = \mathcal{P}_{H,\Gamma_{ij}}^{\Delta T} \tag{4.4.4}$$

be the mortar space-time L^2-orthogonal projection.

4.4.2 Assumptions on the mortar grids

We make the following assumptions on the mortar grids, which are needed to guarantee that the method (4.3.4) is well posed: there exists a positive constant C independent of the spatial mesh sizes h and H such that

$$\forall\, \mu \in \Lambda_H,\ \forall\, i,j, \quad \|\mu\|_{\Gamma_{ij}} \le C(\|\mathcal{Q}_{h,i}\,\mu\|_{\Gamma_{ij}} + \|\mathcal{Q}_{h,j}\,\mu\|_{\Gamma_{ij}}), \tag{4.4.5a}$$

$$\forall\, i,j, \quad \Lambda_{ij}^{\Delta T} \subset W_i^{\Delta t} \cap W_j^{\Delta t}. \tag{4.4.5b}$$

The spatial mortar assumption (4.4.5a) is the same as the assumption made in [89,90]. Note that it is in particular satisfied with $C = \frac{1}{2}$ when $\mathcal{T}_{H,ij}$ is a coarsening of both $\mathcal{T}_{h,i}$ and $\mathcal{T}_{h,j}$ on the interface Γ_{ij} and the space $\Lambda_{H,ij}$ consists of discontinuous piecewise polynomials contained in $\mathbf{V}_{h,i}\cdot\mathbf{n}_i$ and $\mathbf{V}_{h,j}\cdot\mathbf{n}_j$ on Γ_{ij}. In general, it requires that the mortar space Λ_H is sufficiently coarse, so that it is controlled by the normal traces of the neighboring subdomain velocity spaces.

The temporal mortar assumption (4.4.5b) similarly provides control of the mortar time discretization by the subdomain time discretizations. It requires that each subdomain time discretization be a refinement of the mortar time discretization. We also note that (4.4.5a) and (4.4.5b) imply

$$\forall\, \mu \in \Lambda_H^{\Delta T}, \forall\, i,j, \quad \|\mu\|_{L^2(0,T;L^2(\Gamma_{ij}))} \le C(\|\mathcal{Q}_{h,i}^{\Delta t}\,\mu\|_{L^2(0,T;L^2(\Gamma_{ij}))} + \|\mathcal{Q}_{h,j}^{\Delta t}\,\mu\|_{L^2(0,T;L^2(\Gamma_{ij}))}) \tag{4.4.6}$$

for a constant C independent of h, H, Δt, and ΔT.

4.4.3 Discrete inf–sup conditions

Recall the form $b(\cdot,\cdot)$ from (4.2.4b). Under the above assumptions on the mortar grids, the weakly continuous velocity space $\mathbf{V}_{h,0}^{\Delta t}$ of (4.3.2) satisfies the following inf–sup condition.

Lemma 4.4.1 (Discrete divergence inf–sup condition on $\mathbf{V}_{h,0}^{\Delta t}$)**.** *Let (4.4.5) hold. Then there exists a constant $\beta > 0$, independent of h, H, Δt, and ΔT, such that*

$$\forall\, w \in W_h^{\Delta t}, \qquad \sup_{0\neq \mathbf{v}\in \mathbf{V}_{h,0}^{\Delta t}} \frac{b(\mathbf{v},w)}{\|\mathbf{v}\|_{L^2(0,T;\mathbf{V})}} \ge \beta \|w\|_{L^2(0,T;L^2(\Omega))}. \tag{4.4.7}$$

Proof. Let $\mathbf{V}_{h,0} = \{\mathbf{v} \in \mathbf{V}_h : \sum_i \langle \mathbf{v}\cdot\mathbf{n}_i, \mu\rangle_{\Gamma_i} = 0 \quad \forall\, \mu \in \Lambda_H\}$. It is shown in [89, 90] that if (4.4.5a) holds, then there is an interpolant $\Pi_{h,0} : \mathbf{H}^{\frac{1}{2}+\epsilon}(\mathrm{div};\Omega) \to \mathbf{V}_{h,0}$ such that, for all $\boldsymbol{\psi} \in \mathbf{H}^{\frac{1}{2}+\epsilon}(\mathrm{div};\Omega)$,

$$\sum_i (\nabla\cdot(\Pi_{h,0}\boldsymbol{\psi} - \boldsymbol{\psi}), w)_{\Omega_i} = 0 \quad \forall\, w \in W_h, \tag{4.4.8a}$$

$$\|\Pi_{h,0}\boldsymbol{\psi}\|_{\mathbf{V}} \le C(\|\boldsymbol{\psi}\|_{\mathbf{H}^{\frac{1}{2}+\epsilon}(\Omega)} + \|\nabla\cdot\boldsymbol{\psi}\|_{L^2(\Omega)}), \tag{4.4.8b}$$

for a constant C independent of h and H. Define

$$\Pi_{h,0}^{\Delta t} = \Pi_{h,0} \times \mathcal{P}^{\Delta t}.$$

We claim that $\Pi_{h,0}^{\Delta t} : L^2(0,T;\mathbf{H}^{\frac{1}{2}+\epsilon}(\mathrm{div};\Omega)) \to \mathbf{V}_{h,0}^{\Delta t}$. To see this, note that, for all functions $\boldsymbol{\psi} \in L^2(0,T;\mathbf{H}^{\frac{1}{2}+\epsilon}(\mathrm{div};\Omega))$, clearly $\Pi_{h,0}^{\Delta t}\boldsymbol{\psi} \in \mathbf{V}_h^{\Delta t}$. Thus (4.4.5b) implies

$$b_\Gamma(\Pi_{h,0}^{\Delta t}\boldsymbol{\psi}, \mu) = \sum_i \int_0^T \langle \Pi_{h,0}^{\Delta t}\boldsymbol{\psi}\cdot\mathbf{n}_i, \mu\rangle_{\Gamma_i} = \sum_i \int_0^T \langle \Pi_{h,0}\boldsymbol{\psi}\cdot\mathbf{n}_i, \mu\rangle_{\Gamma_i} = 0 \quad \forall\, \mu \in \Lambda_H^{\Delta T},$$

i.e., indeed $\Pi_{h,0}^{\Delta t}\boldsymbol{\psi} \in \mathbf{V}_{h,0}^{\Delta t}$ by virtue of (4.3.2). Moreover, (4.4.8a) and (4.4.8b) imply

$$\sum_i (\nabla\cdot(\Pi_{h,0}^{\Delta t}\boldsymbol{\psi} - \boldsymbol{\psi}), w)_{\Omega_i^T} = 0 \quad \forall\, w \in W_h^{\Delta t}, \tag{4.4.9a}$$

$$\|\Pi_{h,0}^{\Delta t}\boldsymbol{\psi}\|_{L^2(0,T;\mathbf{V})} \le C(\|\boldsymbol{\psi}\|_{L^2(0,T;\mathbf{H}^{\frac{1}{2}+\epsilon}(\Omega))} + \|\nabla\cdot\boldsymbol{\psi}\|_{L^2(0,T;L^2(\Omega))}). \tag{4.4.9b}$$

The inf–sup condition (4.4.7) then follows from the classical continuous inf–sup condition for $b(\cdot,\cdot)$, the existence of the interpolant $\Pi_{h,0}^{\Delta t}$, and Fortin's lemma [98]. □

To control the mortar variable, we need the following mortar inf–sup condition.

Lemma 4.4.2 (Discrete mortar inf–sup condition on $\mathbf{V}_{h,0}^{\Delta t}$)**.** *Let (4.4.6) hold. Then there exists a constant $\beta_\Gamma > 0$, independent of h, H, Δt, and ΔT, such that*

$$\forall\, \mu \in \Lambda_H^{\Delta T}, \qquad \sup_{0\neq\mathbf{v}\in\mathbf{V}_h^{\Delta t}} \frac{b_\Gamma(\mathbf{v},\mu)}{\|\mathbf{v}\|_{L^2(0,T;\mathbf{V})}} \ge \beta_\Gamma \|\mu\|_{L^2(0,T;L^2(\Gamma))}. \tag{4.4.10}$$

Proof. Let $\mu \in \Lambda_H^{\Delta T}$ be given. In the following we assume that μ is extended by zero on $\partial\Omega$. We consider a set of auxiliary subdomain problems. Let $\varphi_i(x,t)$ be the solution for each $t \in (0,T]$ of the problem

$$\nabla \cdot \nabla \varphi_i(\cdot,t) = \overline{(\mathcal{Q}_{h,i}^{\Delta t}\,\mu)(\cdot,t)} \quad \text{in } \Omega_i, \tag{4.4.11a}$$

$$\nabla \varphi_i(\cdot,t) \cdot \mathbf{n}_i = (\mathcal{Q}_{h,i}^{\Delta t}\,\mu)(\cdot,t) \quad \text{on } \partial\Omega_i, \tag{4.4.11b}$$

where $\overline{\mathcal{Q}_{h,i}^{\Delta t}\,\mu}$ denotes the mean value of $\mathcal{Q}_{h,i}^{\Delta t}\,\mu$ on $\partial\Omega_i$. Let $\boldsymbol{\psi}_i = \nabla\varphi_i$. Elliptic regularity [111,119] implies that for all $t \in (0,T]$,

$$\|\boldsymbol{\psi}_i\|_{\frac{1}{2},\Omega_i} + \|\nabla \cdot \boldsymbol{\psi}_i\|_{\Omega_i} \leq C\|\mathcal{Q}_{h,i}^{\Delta t}\,\mu\|_{\partial\Omega_i}. \tag{4.4.12}$$

Let $\mathbf{v}_i = \Pi_{h,i}^{\Delta t}\boldsymbol{\psi}_i \in \mathbf{V}_{h,i}^{\Delta t}$. Note that (4.4.2b) together with (4.4.3) and (4.4.11b) imply that $\mathbf{v}_i \cdot \mathbf{n}_i = \mathcal{Q}_{h,i}^{\Delta t}\,\mu$ on $\partial\Omega_i$. Thus, using definition (4.2.4c) of b_Γ, the fact that μ is extended by zero on $\partial\Omega_i \setminus \Gamma_i$, and definition (4.4.3) of the projection $\mathcal{Q}_{h,i}^{\Delta t}$, we have

$$\begin{aligned} b_\Gamma(\mathbf{v},\mu) &= \sum_i \langle \Pi_{h,i}^{\Delta t}\boldsymbol{\psi}_i \cdot \mathbf{n}_i, \mu\rangle_{\Gamma_i^T} = \sum_i \langle \Pi_{h,i}^{\Delta t}\boldsymbol{\psi}_i \cdot \mathbf{n}_i, \mu\rangle_{\partial\Omega_i^T} = \sum_i \langle \Pi_{h,i}^{\Delta t}\boldsymbol{\psi}_i \cdot \mathbf{n}_i, \mathcal{Q}_{h,i}^{\Delta t}\,\mu\rangle_{\partial\Omega_i^T} \\ &= \sum_i \|\mathcal{Q}_{h,i}^{\Delta t}\,\mu\|^2_{L^2(0,T;L^2(\partial\Omega_i))} \geq C \sum_i \|\mu\|^2_{L^2(0,T;L^2(\Gamma_i))}, \end{aligned} \tag{4.4.13}$$

where we used (4.4.6) in the inequality. On the other hand, (4.4.2c) with $\epsilon = \frac{1}{2}$ and (4.4.12), along with the stabilty of L^2-orthogonal projection $Q_{h,i}^{\Delta t}$, imply

$$\|\mathbf{v}_i\|_{L^2(0,T;\mathbf{V}_i)} \leq C\|\mu\|_{L^2(0,T;L^2(\Gamma_i))}. \tag{4.4.14}$$

The assertion of the lemma follows from combining (4.4.13) and (4.4.14). □

4.4.4 Existence, uniqueness, and stability with respect to data

In the analysis we will utilize the following auxiliary result.

Lemma 4.4.3 (Summation in time). *For any $w \in W_h^{\Delta t}$ and for all Ω_i, there holds*

$$\int_0^T (\tilde{\partial}_t w, w)_{\Omega_i} = \frac{1}{2}\left(\|w_{N_i}^-\|_{\Omega_i}^2 - \|w_0^-\|_{\Omega_i}^2\right) + \frac{1}{2}\sum_{k=1}^{N_i} \|[w]_{k-1}\|_{\Omega_i}^2. \tag{4.4.15}$$

Proof. Using the definition (4.3.3) of $\tilde{\partial}_t w$, we have

$$\begin{aligned}
\int_0^T (\tilde{\partial}_t w, w)_{\Omega_i} &= \sum_{k=1}^{N_i} \int_{t_i^{k-1}}^{t_i^k} \frac{1}{2}\frac{\partial}{\partial t}\|w\|_{\Omega_i}^2 + \sum_{k=1}^{N_i} ([w]_{k-1}, w_{k-1}^+)_{\Omega_i} \\
&= \frac{1}{2}\sum_{k=1}^{N_i} \left(\|w_k^-\|_{\Omega_i}^2 - \|w_{k-1}^+\|_{\Omega_i}^2 + \|w_{k-1}^+\|_{\Omega_i}^2 - \|w_{k-1}^-\|_{\Omega_i}^2 + \|w_{k-1}^+ - w_{k-1}^-\|_{\Omega_i}^2\right) \\
&= \frac{1}{2}\left(\|w_{N_i}^-\|_{\Omega_i}^2 - \|w_0^-\|_{\Omega_i}^2\right) + \frac{1}{2}\sum_{k=1}^{N_i} \|w_{k-1}^+ - w_{k-1}^-\|_{\Omega_i}^2.
\end{aligned}$$

□

To simplify the presentation, we introduce the notation

$$\|\varphi\|_{\mathrm{DG}}^2 = \sum_i \left(\|\varphi_{N_i}^-\|_{\Omega_i}^2 + \sum_{k=1}^{N_i} \|[\varphi]_{k-1}\|_{\Omega_i}^2\right). \tag{4.4.16}$$

Theorem 4.4.1 (Existence and uniqueness of the discrete solution, stability with respect to data). *Assume that conditions (4.4.5) hold. Then the space-time mortar method (4.3.4) has a unique solution. Moreover, for some constant $C > 0$ independent of h, H, Δt, and ΔT,*

$$\|p_h^{\Delta t}\|_{\mathrm{DG}} + \|\mathbf{u}_h^{\Delta t}\|_{L^2(0,T;\mathbf{L}^2(\Omega))} + \|p_h^{\Delta t}\|_{L^2(0,T;L^2(\Omega))} + \|\lambda_H^{\Delta T}\|_{L^2(0,T;L^2(\Gamma))} \le C(\|q\|_{L^2(0,T;L^2(\Omega))} + \|p_0\|_{\Omega}). \tag{4.4.17}$$

Proof. We begin with establishing the stability bound (4.4.17). Taking $\mathbf{v} = \mathbf{u}_h^{\Delta t}$, $w = p_h^{\Delta t}$, and $\mu = \lambda_H^{\Delta T}$ in (4.3.4) and combining the equations, we obtain, using (4.4.15) and Young's inequality,

$$\frac{1}{2}\sum_i \left(\|(p_h^{\Delta t})^-_{N_i}\|^2_{\Omega_i} + \sum_{k=1}^{N_i} \|[p_h^{\Delta t}]_{k-1}\|^2_{\Omega_i} \right) + \|K^{-\frac{1}{2}}\mathbf{u}_h^{\Delta t}\|^2_{L^2(0,T;\mathbf{L}^2(\Omega))}$$
$$\leq \frac{\epsilon}{2}\|p_h^{\Delta t}\|^2_{L^2(0,T;L^2(\Omega))} + \frac{1}{2\epsilon}\|q\|^2_{L^2(0,T;L^2(\Omega))} + \frac{1}{2}\|\mathcal{P}_h p_0\|^2.$$

The inf–sup condition for the weakly continuous velocity (4.4.7) and (4.3.4a) imply

$$\|p_h^{\Delta t}\|_{L^2(0,T;L^2(\Omega))} \leq C\|K^{-\frac{1}{2}}\mathbf{u}_h^{\Delta t}\|_{L^2(0,T;\mathbf{L}^2(\Omega))}.$$

Furthermore, the mortar inf–sup condition (4.4.10) and (4.3.4a) imply

$$\|\lambda_H^{\Delta T}\|_{L^2(0,T;L^2(\Gamma))} \leq C(\|K^{-\frac{1}{2}}\mathbf{u}_h^{\Delta t}\|_{L^2(0,T;\mathbf{L}^2(\Omega))} + \|p_h^{\Delta t}\|_{L^2(0,T;L^2(\Omega))}).$$

Combining above three inequalities, taking ϵ sufficiently small, and using equation (4.2.2), we obtain (4.4.17). The existence and uniqueness of a solution follows from (4.4.17) by taking $q = 0$ and $p_0 = 0$. □

Remark 4.4.1 (Control of divergence)**.** *Control on* $\|\nabla \cdot \mathbf{u}_h^{\Delta t}\|_{L^2(0,T;L^2(\Omega_i))}$ *could be obtained following the approach in [94, Lemma 4.4]. It requires first obtaining bound on* $\|\hat{\partial}_t p_h^{\Delta t}\|_{L^2(0,T;L^2(\Omega))}$ *by taking* $w = \hat{\partial}_t p_h^{\Delta t}$ *in* (4.3.4b) *and using a time-differentiated version of* (4.3.4a)*. Then a bound on* $\|\nabla \cdot \mathbf{u}_h^{\Delta t}\|_{L^2(0,T;L^2(\Omega_i))}$ *would follow from* (4.3.4b) *by taking* $w|_{\Omega_i} = \nabla \cdot \mathbf{u}_h^{\Delta t}|_{\Omega_i}$*. For sake of space, we do not pursue this here.*

4.5 A Priori Error Analysis

In this section we derive a priori error estimates for the solution of the space-time mortar MFE method (4.3.4).

4.5.1 Approximation properties of the space-time interpolants

Assume that the spaces $\mathbf{V}_h^{\Delta t}$ and $W_h^{\Delta t}$ from (4.3.1) contain respectively on each space-time element polynomials of degree k and l in space and polynomials of degree q in time. Let $\Lambda_H^{\Delta T}$ contain on each space-time mortar element polynomials of degree m in space and polynomials of degree s in time. We have the following approximation properties for the space-time interpolants $\mathcal{P}_h^{\Delta t}$ and $\mathcal{P}_{H,\Gamma}^{\Delta T}$ of Section 4.4.1 and $\Pi_{h,0}^{\Delta t}$ of the proof of Lemma 4.4.1:

$$\begin{aligned}\|\boldsymbol{\psi} - \Pi_{h,0}^{\Delta t}\boldsymbol{\psi}\|_{\Omega^T} &\leq C\sum_i \|\boldsymbol{\psi}\|_{H^{r_q}(0,T;\mathbf{H}^{r_k}(\Omega_i))}(h^{r_k} + \Delta t^{r_q}) \\ &+ C\|\boldsymbol{\psi}\|_{H^{r_q}(0,T;\mathbf{H}^{\tilde{r}_k+\frac{1}{2}}(\Omega))}(h^{\tilde{r}_k}H^{\frac{1}{2}} + \Delta t^{r_q}), \\ &\qquad 0 < r_k \leq k+1, \quad 0 < \tilde{r}_k \leq k+1, \quad 0 \leq r_q \leq q+1, \end{aligned} \tag{4.5.1a}$$

$$\|\varphi - \mathcal{P}_h^{\Delta t}\varphi\|_{\Omega_i^T} \leq C\|\varphi\|_{H^{r_q}(0,T;H^{r_l}(\Omega_i))}(h^{r_l} + \Delta t^{r_q}), \quad 0 \leq r_l \leq l+1, \quad 0 \leq r_q \leq q+1, \tag{4.5.1b}$$

$$\|\varphi - \mathcal{P}_{H,\Gamma}^{\Delta T}\|_{\Gamma_{ij}^T} \leq C\|\varphi\|_{H^{r_s}(0,T;H^{r_m}(\Gamma_{ij}))}(H^{r_m} + \Delta T^{r_s}), \quad 0 \leq r_m \leq m+1, \quad 0 \leq r_s \leq s+1. \tag{4.5.1c}$$

Bound (4.5.1a) follows from the approximation properties of $\Pi_{h,0}$ obtained in [89, 90]. Bounds (4.5.1b) and (4.5.1c) are standard approximation properties of the L^2 projection [99].

In the analysis we will also use the following approximation property, which follows from the stability of the L^2 projection in L^∞ [100]:

$$\|\varphi - \mathcal{P}_h^{\Delta t}\varphi\|_{L^\infty(0,T;L^2(\Omega_i))} \leq C\|\varphi\|_{W^{r_q,\infty}(0,T;H^{r_l}(\Omega_i))}(h^{r_l} + \Delta t^{r_q}), \quad 0 \leq r_l \leq l+1, \quad 0 \leq r_q \leq q+1. \tag{4.5.2}$$

We also recall the well-known discrete trace inequality, for all $\mathbf{v} \in \mathbf{V}_{h,i}$, $\|\mathbf{v}\cdot\mathbf{n}_i\|_{\Gamma_i} \leq Ch^{-\frac{1}{2}}\|\mathbf{v}\|_{\Omega_i}$, which implies

$$\forall\, \mathbf{v} \in \mathbf{V}_{h,i}^{\Delta t}, \quad \|\mathbf{v}\cdot\mathbf{n}_i\|_{\Gamma_i^T} \leq Ch^{-\frac{1}{2}}\|\mathbf{v}\|_{\Omega_i^T}. \tag{4.5.3}$$

4.5.2 A priori error estimate

We proceed with the error estimate for the space-time mortar MFE method (4.3.4).

Theorem 4.5.1 (A priori error estimate)**.** *Assume that conditions* (4.4.5) *hold and that the solution to* (4.2.5) *is sufficiently smooth. Then there exists a constant* $C > 0$ *independent of the mesh sizes* h, H, Δt, *and* ΔT, *such that the solution to the space-time mortar MFE method* (4.3.4) *satisfies*

$$
\begin{aligned}
&\|p - p_h^{\Delta t}\|_{\mathrm{DG}} + \|\mathbf{u} - \mathbf{u}_h^{\Delta t}\|_{\Omega^T} + \|p - p_h^{\Delta t}\|_{\Omega^T} + \|\lambda - \lambda_H^{\Delta T}\|_{\Gamma^T} \\
&\quad \le C\Big(\sum_i \|\mathbf{u}\|_{H^{r_q}(0,T;\mathbf{H}^{r_k}(\Omega_i))}(h^{r_k} + \Delta t^{r_q}) + \|\mathbf{u}\|_{H^{r_q}(0,T;\mathbf{H}^{\tilde{r}_k+\frac{1}{2}}(\Omega))}(h^{\tilde{r}_k} H^{\frac{1}{2}} + \Delta t^{r_q}) \\
&\qquad + \sum_i \|p\|_{W^{r_q,\infty}(0,T;H^{r_l}(\Omega_i))} \Delta t^{-\frac{1}{2}}(h^{r_l} + \Delta t^{r_q}) + \sum_{i,j} \|\lambda\|_{H^{r_s}(0,T;H^{r_m}(\Gamma_{ij}))} h^{-\frac{1}{2}}(H^{r_m} + \Delta T^{r_s}) \Big), \\
&\quad 0 < r_k \le k+1,\ 0 < \tilde{r}_k \le k+1,\ 0 \le r_q \le q+1,\ 0 \le r_l \le l+1, \\
&\quad 0 \le r_m \le m+1,\ 0 \le r_s \le s+1.
\end{aligned}
\tag{4.5.4}
$$

Proof. For the purpose of the analysis, we consider the following equivalent formulation of (4.3.4) in the space of weakly continuous velocities $\mathbf{V}_{h,0}^{\Delta t}$ given by (4.3.2): find $\mathbf{u}_{h,0}^{\Delta t} \in \mathbf{V}_{h,0}^{\Delta t}$ and $p_h^{\Delta t} \in W_h^{\Delta t}$ such that $(p_{h,0}^{\Delta t})^- = \mathcal{P}_h p_0$ and

$$
a(\mathbf{u}_h^{\Delta t}, \mathbf{v}) + b(\mathbf{v}, p_h^{\Delta t}) = 0 \quad \forall\, \mathbf{v} \in \mathbf{V}_{h,0}^{\Delta t}, \tag{4.5.5a}
$$

$$
(\tilde{\partial}_t p_h^{\Delta t}, w)_{\Omega^T} - b(\mathbf{u}_h^{\Delta t}, w) = (q, w)_{\Omega^T} \quad \forall\, w \in W_h^{\Delta t}. \tag{4.5.5b}
$$

The fact that $\mathcal{P}_{H,\Gamma}^{\Delta T}$ defined in (4.4.4) maps to $\Lambda_H^{\Delta T}$ and definition (4.3.2) imply that $b_\Gamma(\mathbf{v}, \mathcal{P}_{H,\Gamma}^{\Delta T}\lambda) = 0$ for all $\mathbf{v} \in \mathbf{V}_{h,0}^{\Delta t}$, where $\lambda = p|_\Gamma$ is the pressure trace from (4.2.5). Then, subtracting (4.5.5a)–(4.5.5b) from (4.2.5a)–(4.2.5b), we obtain the error equations

$$
a(\mathbf{u} - \mathbf{u}_h^{\Delta t}, \mathbf{v}) + b(\mathbf{v}, \mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}) + b_\Gamma(\mathbf{v}, \lambda - \mathcal{P}_{H,\Gamma}^{\Delta T}\lambda) = 0 \quad \forall\, \mathbf{v} \in \mathbf{V}_{h,0}^{\Delta t}, \tag{4.5.6a}
$$

$$
\Big(\partial_t p - \tilde{\partial}_t p_h^{\Delta t}, w\Big)_{\Omega^T} - b(\Pi_{h,0}^{\Delta t}\mathbf{u} - \mathbf{u}_h^{\Delta t}, w) = 0 \quad \forall\, w \in W_h^{\Delta t}, \tag{4.5.6b}
$$

where we have also used (4.4.1) and (4.4.9a) to incorporate the interpolants $\mathcal{P}_h^{\Delta t}$ and $\Pi_{h,0}^{\Delta t}$. We take $\mathbf{v} = \Pi_{h,0}^{\Delta t}\mathbf{u} - \mathbf{u}_h^{\Delta t}$ and $w = \mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}$ and sum the two equations, resulting in

$$
\begin{aligned}
&a(\Pi_{h,0}^{\Delta t}\mathbf{u} - \mathbf{u}_h^{\Delta t}, \Pi_{h,0}^{\Delta t}\mathbf{u} - \mathbf{u}_h^{\Delta t}) + \Big(\partial_t p - \tilde{\partial}_t p_h^{\Delta t}, \mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}\Big)_{\Omega^T} \\
&\qquad = a(\Pi_{h,0}^{\Delta t}\mathbf{u} - \mathbf{u}, \Pi_{h,0}^{\Delta t}\mathbf{u} - \mathbf{u}_h^{\Delta t}) - b_\Gamma(\Pi_{h,0}^{\Delta t}\mathbf{u} - \mathbf{u}_h^{\Delta t}, \lambda - \mathcal{P}_{H,\Gamma}^{\Delta T}\lambda).
\end{aligned}
\tag{4.5.7}
$$

For the second term on the left of (4.5.7), restricted to a subdomain, we write, using (4.3.3),

$$\begin{aligned}
&\int_0^T \left(\partial_t p - \tilde{\partial}_t p_h^{\Delta t}, \mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}\right)_{\Omega_i} \\
&\quad = \sum_{k=1}^{N_i} \int_{t_i^{k-1}}^{t_i^k} \left(\partial_t (p - p_h^{\Delta t}), \mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}\right)_{\Omega_i} - \sum_{k=1}^{N_i} \left([p_h^{\Delta t}]_{k-1}, (\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t})_{k-1}^+\right)_{\Omega_i} =: I_1 + I_2.
\end{aligned} \tag{4.5.8}$$

For the first term, we develop

$$\begin{aligned}
I_1 &= -\sum_{k=1}^{N_i} \int_{t_i^{k-1}}^{t_i^k} \left(p - p_h^{\Delta t}, \partial_t (\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t})\right)_{\Omega_i} + \sum_{k=1}^{N_i} (p - p_h^{\Delta t}, \mathcal{P}_h^{\Delta t} p - p_h^{\Delta t})_{\Omega_i} \Big|_{t_i^{k-1}}^{t_i^k} \\
&= -\sum_{k=1}^{N_i} \int_{t_i^{k-1}}^{t_i^k} \left(\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}, \partial_t (\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t})\right)_{\Omega_i} + \sum_{k=1}^{N_i} (p - p_h^{\Delta t}, \mathcal{P}_h^{\Delta t} p - p_h^{\Delta t})_{\Omega_i} \Big|_{t_i^{k-1}}^{t_i^k} \\
&= \sum_{k=1}^{N_i} \int_{t_i^{k-1}}^{t_i^k} \left(\partial_t (\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}), \mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}\right)_{\Omega_i} + \sum_{k=1}^{N_i} (p - \mathcal{P}_h^{\Delta t} p, \mathcal{P}_h^{\Delta t} p - p_h^{\Delta t})_{\Omega_i} \Big|_{t_i^{k-1}}^{t_i^k} \\
&=: I_{1,1} + I_{1,2},
\end{aligned}$$

where we used integration by parts in the first and third equalities and the orthogonality property of $\mathcal{P}_h^{\Delta t}$ in the second equality. The term $I_{1,2}$ will be combined with I_2. To this end, we use the following algebraic manipulations, for sequences $\{\alpha_k^\pm\}$ and $\{\beta_k^\pm\}$:

$$\begin{aligned}
\sum_{k=1}^{N_i} \alpha\beta \Big|_{t_i^{k-1}}^{t_i^k} &= \sum_{k=1}^{N_i} (\alpha_k^- \beta_k^- - \alpha_{k-1}^+ \beta_{k-1}^+) = \sum_{k=1}^{N_i} (\alpha_k^- \beta_k^- - [\alpha]_{k-1} \beta_{k-1}^+ - \alpha_{k-1}^- \beta_{k-1}^+) \\
&= \sum_{k=1}^{N_i} (\alpha_k^- \beta_k^- - [\alpha]_{k-1} \beta_{k-1}^+ - \alpha_{k-1}^- [\beta]_{k-1} - \alpha_{k-1}^- \beta_{k-1}^-) \\
&= \alpha_{N_i}^- \beta_{N_i}^- - \sum_{k=1}^{N_i} ([\alpha]_{k-1} \beta_{k-1}^+ + \alpha_{k-1}^- [\beta]_{k-1}) - \alpha_0^- \beta_0^- .
\end{aligned}$$

In order to apply this formula for $I_{1,2}$, we need to make sure that the quantities α_0^- and β_0^- are defined. Recall that $(p_h^{\Delta t})_0^- = \mathcal{P}_h p_0$. We set $p_0^- = p_0$ and $(\mathcal{P}_h^{\Delta t} p)_0^- = \mathcal{P}_h p_0$. Then, we have

$$\begin{aligned}
I_{1,2} = ((p - \mathcal{P}_h^{\Delta t} p)_{N_i}^-, (\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t})_{N_i}^-)_{\Omega_i} - \sum_{k=1}^{N_i} ([p - \mathcal{P}_h^{\Delta t} p]_{k-1}, (\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t})_{k-1}^+)_{\Omega_i} \\
- \sum_{k=1}^{N_i} ((p - \mathcal{P}_h^{\Delta t} p)_{k-1}^-, [\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}]_{k-1})_{\Omega_i}.
\end{aligned} \tag{4.5.9}$$

Combining (4.5.8)–(4.5.9), and using that $[p]_{k-1} = 0$, we obtain

$$\begin{aligned}
&\int_0^T \left(\partial_t p - \tilde{\partial}_t p_h^{\Delta t}, \mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}\right)_{\Omega_i} \\
&\quad = \sum_{k=1}^{N_i} \int_{t_i^{k-1}}^{t_i^k} \left(\partial_t(\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}), \mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}\right)_{\Omega_i} + \sum_{k=1}^{N_i} ([\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}]_{k-1}, (\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t})_{k-1}^+)_{\Omega_i} \\
&\qquad\quad + ((p - \mathcal{P}_h^{\Delta t} p)_{N_i}^-, (\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t})_{N_i}^-)_{\Omega_i} - \sum_{k=1}^{N_i} ((p - \mathcal{P}_h^{\Delta t} p)_{k-1}^-, [\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}]_{k-1})_{\Omega_i} \\
&\quad = \frac{1}{2} \|(\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t})_{N_i}^-\|_{\Omega_i}^2 + \frac{1}{2} \sum_{k=1}^{N_i} \|[\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}]_{k-1}\|_{\Omega_i}^2 \\
&\qquad\quad + ((p - \mathcal{P}_h^{\Delta t} p)_{N_i}^-, (\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t})_{N_i}^-)_{\Omega_i} - \sum_{k=1}^{N_i} ((p - \mathcal{P}_h^{\Delta t} p)_{k-1}^-, [\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}]_{k-1})_{\Omega_i},
\end{aligned} \tag{4.5.10}$$

where, recalling notation (4.3.3), we have used (4.4.15) for the second equality.

Now, combining (4.5.7) and (4.5.10), and using notation (4.4.16) together with the Cauchy–Schwarz and Young's inequalities, we obtain,

$$\begin{aligned}
&\|K^{-\frac{1}{2}}(\Pi_{h,0}^{\Delta t}\mathbf{u} - \mathbf{u}_h^{\Delta t})\|_{\Omega^T}^2 + \frac{1}{2}\|\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}\|_{\mathrm{DG}}^2 \\
&\quad \le \|K^{-\frac{1}{2}}(\Pi_{h,0}^{\Delta t}\mathbf{u} - \mathbf{u})\|_{\Omega^T} \|K^{-\frac{1}{2}}(\Pi_{h,0}^{\Delta t}\mathbf{u} - \mathbf{u}_h^{\Delta t})\|_{\Omega^T} + \sum_i \|(\Pi_{h,0}^{\Delta t}\mathbf{u} - \mathbf{u}_h^{\Delta t}) \cdot \mathbf{n}_i\|_{\Gamma_i^T} \|\lambda - \mathcal{P}_{H,\Gamma}^{\Delta T}\lambda\|_{\Gamma_i^T} \\
&\qquad + \sum_i \left(\|(p - \mathcal{P}_h^{\Delta t} p)_{N_i}^-\|_{\Omega_i} \|(\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t})_{N_i}^-\|_{\Omega_i} + \sum_{k=1}^{N_i} \|(p - \mathcal{P}_h^{\Delta t} p)_{k-1}^-\|_{\Omega_i} \|[\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}]_{k-1}\|_{\Omega_i} \right) \\
&\quad \le \epsilon \left(\|\Pi_{h,0}^{\Delta t}\mathbf{u} - \mathbf{u}_h^{\Delta t}\|_{\Omega^T}^2 + \|\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}\|_{\mathrm{DG}}^2 \right) \\
&\qquad + C_\epsilon \left(\|\Pi_{h,0}^{\Delta t}\mathbf{u} - \mathbf{u}\|_{\Omega^T}^2 + h^{-1}\|\lambda - \mathcal{P}_{H,\Gamma}^{\Delta T}\lambda\|_{\Gamma^T}^2 + \sum_i \sum_{k=1}^{N_i} \|(p - \mathcal{P}_h^{\Delta t} p)_{k-1}^-\|_{\Omega_i}^2 \right),
\end{aligned}$$

where we used the trace inequality (4.5.3) in the last estimate. Taking ϵ sufficiently small and using (4.2.2) gives

$$\begin{aligned}
&\|\Pi_{h,0}^{\Delta t}\mathbf{u} - \mathbf{u}_h^{\Delta t}\|_{\Omega^T} + \|\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}\|_{\mathrm{DG}} \\
&\qquad \le C \left(\|\Pi_{h,0}^{\Delta t}\mathbf{u} - \mathbf{u}\|_{\Omega^T} + h^{-\frac{1}{2}}\|\lambda - \mathcal{P}_{H,\Gamma}^{\Delta T}\lambda\|_{\Gamma^T} + \Delta t^{-\frac{1}{2}} \sum_i \|p - \mathcal{P}_h^{\Delta t} p\|_{L^\infty(0,T;L^2(\Omega_i))} \right).
\end{aligned} \tag{4.5.11}$$

Next, the inf–sup condition for the weakly continuous velocity (4.4.7) and (4.5.6a) imply, using (4.5.3),

$$\|\mathcal{P}_h^{\Delta t} p - p_h^{\Delta t}\|_{\Omega^T} \leq C \left(\|\mathbf{u} - \mathbf{u}_h^{\Delta t}\|_{\Omega^T} + h^{-\frac{1}{2}} \|\lambda - \mathcal{P}_{H,\Gamma}^{\Delta T} \lambda\|_{\Gamma^T} \right). \tag{4.5.12}$$

Finally, to obtain a bound on $\lambda_H^{\Delta T}$, we subtract (4.3.4a) from (4.2.5a), to obtain the error equation

$$a(\mathbf{u} - \mathbf{u}_h^{\Delta t}, \mathbf{v}) + b(\mathbf{v}, p - p_h^{\Delta t}) + b_\Gamma(\mathbf{v}, \mathcal{P}_{H,\Gamma}^{\Delta T} \lambda - \lambda_H^{\Delta T}) = b_\Gamma(\mathbf{v}, \mathcal{P}_{H,\Gamma}^{\Delta T} \lambda - \lambda) \quad \forall \mathbf{v} \in \mathbf{V}_h^{\Delta t}. \tag{4.5.13}$$

The mortar inf–sup condition (4.4.10) and (4.5.13) imply, using (4.5.3),

$$\|\mathcal{P}_{H,\Gamma}^{\Delta T} \lambda - \lambda_H^{\Delta T}\|_{\Gamma^T} \leq C \left(\|\mathbf{u} - \mathbf{u}_h^{\Delta t}\|_{\Omega^T} + \|p - p_h^{\Delta t}\|_{\Omega^T} + h^{-\frac{1}{2}} \|\lambda - \mathcal{P}_{H,\Gamma}^{\Delta T} \lambda\|_{\Gamma^T} \right). \tag{4.5.14}$$

The assertion of the theorem follows from combining (4.5.11), (4.5.12), and (4.5.14) and using the triangle inequality and the approximation bounds (4.5.1)–(4.5.2). □

4.5.3 Comments

Remark 4.5.1 (The factor $h^{-\frac{1}{2}}$ and appropriate choice of the polynomial degrees m and s)**.** *The term $h^{-\frac{1}{2}}(H^{r_m} + \Delta T^{r_s})$ in the error bound appears due the use of the discrete trace inequality (4.5.3) to control the consistency error $b_\Gamma(\Pi_{h,0}^{\Delta t}\mathbf{u} - \mathbf{u}_h^{\Delta t}, \lambda - \mathcal{P}_{H,\Gamma}^{\Delta T}\lambda)$. This term can be made comparable to the other error terms in (4.5.4) by choosing m and s sufficiently large. Alternatively, this term can be improved if a bound on $\|\nabla \cdot (\mathbf{u} - \mathbf{u}_h^{\Delta t})\|_{\Omega_i^T}$ is available. In particular, using a suitable interpolant $\tilde{\mathcal{P}}_{H,\Gamma}^{\Delta T}$ in the continuous subspace of $\Lambda_H^{\Delta T}$ that can be extended continuously by zero to $\partial\Omega$, we have for a.e. $t \in (0,T)$,*

$$\begin{aligned}
\langle (\Pi_{h,0}^{\Delta t}\mathbf{u} - \mathbf{u}_h^{\Delta t}) \cdot \mathbf{n}_i, \lambda - \tilde{\mathcal{P}}_{H,\Gamma}^{\Delta T}\lambda \rangle_{\partial\Omega_i} &\leq \|(\Pi_{h,0}^{\Delta t}\mathbf{u} - \mathbf{u}_h^{\Delta t}) \cdot \mathbf{n}_i\|_{H^{-\frac{1}{2}}(\partial\Omega_i)} \|\lambda - \tilde{\mathcal{P}}_{H,\Gamma}^{\Delta T}\lambda\|_{H^{\frac{1}{2}}(\partial\Omega_i)} \\
&\leq C \|\Pi_{h,0}^{\Delta t}\mathbf{u} - \mathbf{u}_h^{\Delta t}\|_{\mathrm{div};\Omega_i} \|\lambda - \tilde{\mathcal{P}}_{H,\Gamma}^{\Delta T}\lambda\|_{H^{\frac{1}{2}}(\partial\Omega_i)},
\end{aligned}$$

thus avoiding the $h^{-\frac{1}{2}}$ factor. We refer the reader to [90] for further details. Since we do not bound $\|\nabla \cdot (\mathbf{u} - \mathbf{u}_h^{\Delta t})\|_{\Omega_i^T}$, we do not pursue this approach.

4.6 Reduction To An Interface Problem

In this section we combine the time-dependent Steklov–Poincaré operator approach from [112] with the mortar domain decomposition algorithm from [89,90] to reduce the global problem (4.3.4) to a space-time interface problem.

4.6.1 Decomposition of the solution

Consider a decomposition of the solution to (4.3.4) in the form

$$\mathbf{u}_h^{\Delta t} = \mathbf{u}_h^{\Delta t,*}(\lambda_H^{\Delta T}) + \overline{\mathbf{u}}_h^{\Delta t}, \quad p_h^{\Delta t} = p_h^{\Delta t,*}(\lambda_H^{\Delta T}) + \overline{p}_h^{\Delta t}. \tag{4.6.1}$$

Here, $\overline{\mathbf{u}}_h^{\Delta t} \in \mathbf{V}_h^{\Delta t}$, $\overline{p}_h^{\Delta t} \in W_h^{\Delta t}$ are such that for each Ω_i^T, $(\overline{\mathbf{u}}_h^{\Delta t}|_{\Omega_i^T} \in \mathbf{V}_{h,i}^{\Delta t}, \overline{p}_h^{\Delta t}|_{\Omega_i^T} \in W_{h,i}^{\Delta t})$ is the solution to the space-time subdomain problem in Ω_i^T with zero Dirichlet data on the space-time interfaces and the prescribed source term, initial data, and boundary data on the external boundary:

$$a_i(\overline{\mathbf{u}}_h^{\Delta t}, \mathbf{v}) + b_i(\mathbf{v}, \overline{p}_h^{\Delta t}) = 0 \quad \forall\, \mathbf{v} \in \mathbf{V}_{h,i}^{\Delta t} \tag{4.6.2a}$$

$$(\tilde{\partial}_t \overline{p}_h^{\Delta t}, w)_{\Omega_i^T} - b_i(\overline{\mathbf{u}}_h^{\Delta t}, w) = (q, w)_{\Omega_i^T} \quad \forall\, w \in W_{h,i}^{\Delta t}. \tag{4.6.2b}$$

Furthermore, for a given $\mu \in \Lambda_H^{\Delta T}$, $\mathbf{u}_h^{\Delta t,*}(\mu) \in \mathbf{V}_h^{\Delta t}$, $p_h^{\Delta t,*}(\mu) \in W_h^{\Delta t}$ are such that for each Ω_i^T, $(\mathbf{u}_h^{\Delta t,*}(\mu)|_{\Omega_i^T} \in \mathbf{V}_{h,i}^{\Delta t}, p_h^{\Delta t,*}(\mu)|_{\Omega_i^T} \in W_{h,i}^{\Delta t})$ is the solution to the space-time subdomain problem in Ω_i^T with Dirichlet data μ on the space-time interfaces and zero source term, initial data, and boundary data on the external boundary:

$$a_i(\mathbf{u}_h^{\Delta t,*}(\mu), \mathbf{v}) + b_i(\mathbf{v}, p_h^{\Delta t,*}(\mu)) = -\langle \mathbf{v} \cdot \mathbf{n}_i, \mu \rangle_{\Gamma_i^T} \quad \forall\, \mathbf{v} \in \mathbf{V}_{h,i}^{\Delta t}, \tag{4.6.3a}$$

$$(\tilde{\partial}_t p_h^{\Delta t,*}(\mu), w)_{\Omega_i^T} - b_i(\mathbf{u}_h^{\Delta t,*}(\mu), w) = 0 \quad \forall\, w \in W_{h,i}^{\Delta t}. \tag{4.6.3b}$$

Note that both (4.6.2) and (4.6.3) are posed in the individual space-time subdomains Ω_i^T and can thus be solved in parallel (on the entire space-time subdomains Ω_i^T, without any synchronization on time steps). It is easy to check that (4.3.4) is equivalent to solving the space-time interface problem: find $\lambda_H^{\Delta T} \in \Lambda_H^{\Delta T}$ such that

$$-b_\Gamma(\mathbf{u}_h^{\Delta t,*}(\lambda_H^{\Delta T}), \mu) = b_\Gamma(\overline{\mathbf{u}}_h^{\Delta t}, \mu) \quad \forall\, \mu \in \Lambda_H^{\Delta T}, \tag{4.6.4}$$

and obtaining $\mathbf{u}_h^{\Delta t}$ and $p_h^{\Delta t}$ from (4.6.1)–(4.6.3).

4.6.2 Space-time Steklov–Poincaré operator

The above problem can be written in an operator form: find $\lambda_H^{\Delta T} \in \Lambda_H^{\Delta T}$ such that

$$S\,\lambda_H^{\Delta T} = g, \tag{4.6.5}$$

where $S : \Lambda_H^{\Delta T} \to \Lambda_H^{\Delta T}$ is the space-time Steklov–Poincaré operator defined as

$$\langle S\lambda, \mu\rangle_{\Gamma^T} = \sum_i \langle S_i\lambda, \mu\rangle_{\Gamma_i^T}, \quad \langle S_i\lambda, \mu\rangle_{\Gamma_i^T} = -\langle \mathbf{u}_h^{\Delta t,*}(\lambda)\cdot\mathbf{n}_i, \mu\rangle_{\Gamma_i^T} \quad \forall\,\lambda,\mu \in \Lambda_H^{\Delta T}, \tag{4.6.6}$$

and $g \in \Lambda_H^{\Delta T}$ is defined as $\langle g, \mu\rangle_{\Gamma^T} = b_\Gamma(\overline{\mathbf{u}}_h^{\Delta t}, \mu)\,\forall\,\mu \in \Lambda_H^{\Delta T}$.

Lemma 4.6.1 (Space-time Steklov–Poincaré operator)**.** *Assume that conditions* (4.4.5) *hold. Then the operator S defined in* (4.6.6) *is positive definite.*

Proof. For $\lambda, \mu \in \Lambda_H^{\Delta T}$, consider (4.6.3a) with data μ and test function $\mathbf{v} = \mathbf{u}_h^{\Delta t,*}(\lambda)$. This implies, using (4.6.6),

$$\begin{aligned}\langle S\lambda, \mu\rangle_{\Gamma^T} &= a(\mathbf{u}_h^{\Delta t,*}(\mu), \mathbf{u}_h^{\Delta t,*}(\lambda)) + b(\mathbf{u}_h^{\Delta t,*}(\lambda), p_h^{\Delta t,*}(\mu)) \\ &= a(\mathbf{u}_h^{\Delta t,*}(\mu), \mathbf{u}_h^{\Delta t,*}(\lambda)) + (\tilde{\partial}_t p_h^{\Delta t,*}(\lambda), p_h^{\Delta t,*}(\mu))_{\Omega^T},\end{aligned} \tag{4.6.7}$$

where we have used (4.6.3b) with data λ and test function $p_h^{\Delta t,*}(\mu)$ in the second equality. Lemma 4.4.3 together with $p_h^{\Delta t,*}(\mu)(x,0) = 0$ (recall that zero initial data is supposed in (4.6.3)) imply that $\langle S\mu, \mu\rangle_{\Gamma^T} \geq 0$ for all $\mu \in \Lambda_H^{\Delta T}$. Moreover, assume that $\langle S\mu, \mu\rangle_{\Gamma^T} = 0$. Then $\mathbf{u}_h^{\Delta t,*}(\mu) = 0$. The inf–sup condition for the weakly continuous velocity (4.4.7) and (4.6.3a) imply $p_h^{\Delta t,*}(\mu) = 0$. Then the mortar inf–sup condition (4.4.10) and (4.6.3a) imply $\mu = 0$. □

Due to Lemma 4.6.1, GMRES can be employed to solve the interface problem (4.6.5). On each GMRES iteration, the dominant computational cost is the evaluation of the action of S, which requires solving space-time problems with prescribed Dirichlet interface data in each individual space-time subdomain $\Omega_i \times (0,T]$. The following result can be used to provide a bound on the number of GMRES iterations.

Theorem 4.6.1 (Spectral bound)**.** *Assume that conditions* (4.4.5) *hold. Then there exist positive constants C_0 and C_1 independent of the mesh sizes h, H, Δt, and ΔT, such that*

$$\forall\,\mu \in \Lambda_H^{\Delta T}, \quad C_0\|\mu\|_{\Gamma^T}^2 \leq \langle S\mu, \mu\rangle_{\Gamma^T} \leq C_1 h^{-1}\|\mu\|_{\Gamma^T}^2. \tag{4.6.8}$$

Proof. Using (4.6.6), the Cauchy–Schwarz inequality, and (4.5.3), we obtain

$$\langle S_i\mu,\mu\rangle_{\Gamma_i^T} \le \|\mathbf{u}_h^{\Delta t,*}(\mu)\cdot\mathbf{n}_i\|_{\Gamma_i^T}\|\mu\|_{\Gamma_i^T} \le Ch^{-\frac{1}{2}}\|\mathbf{u}_h^{\Delta t,*}(\mu)\|_{\Omega_i^T}\|\mu\|_{\Gamma_i^T} \le Ch^{-\frac{1}{2}}\langle S_i\mu,\mu\rangle_{\Gamma_i^T}^{\frac{1}{2}}\|\mu\|_{\Gamma_i^T},$$

where we used (4.6.7), also valid on each Ω_i^T, in the last inequality. This implies the upper bound in (4.6.8).

To prove the lower bound in (4.6.8), we consider the set of auxiliary subdomain problems (4.4.11) with data μ. Let $\mathbf{v}_i = \Pi_{h,i}^{\Delta t}\boldsymbol{\psi}_i$ and recall that $\mathbf{v}_i\cdot\mathbf{n}_i = \mathcal{Q}_{h,i}^{\Delta t}\mu$. Using (4.4.6) and (4.6.3a), we have

$$\begin{aligned}
\|\mu\|_{\Gamma^T}^2 &\le C\sum_i \langle \mathcal{Q}_{h,i}^{\Delta t}\mu, \mathcal{Q}_{h,i}^{\Delta t}\mu\rangle_{\Gamma_i^T} = C\sum_i \langle \mathcal{Q}_{h,i}^{\Delta t}\mu,\mu\rangle_{\Gamma_i^T} = C\sum_i \langle \mathbf{v}_i\cdot\mathbf{n}_i,\mu\rangle_{\Gamma_i^T},\\
&= -C\sum_i \Big(a_i(\mathbf{u}_h^{\Delta t,*}(\mu),\mathbf{v}_i) + b_i(\mathbf{v}_i, p_h^{\Delta t,*}(\mu))\Big)\\
&\le C\sum_i \Big(\|\mathbf{u}_h^{\Delta t,*}(\mu)\|_{\Omega_i^T}^2 + \|p_h^{\Delta t,*}(\mu)\|_{\Omega_i^T}^2\Big)^{\frac{1}{2}}\|\mathbf{v}_i\|_{L^2(0,T;\mathbf{V}_i)}\\
&\le C\left\{\sum_i \|\mathbf{u}_h^{\Delta t,*}(\mu)\|_{\Omega_i^T}^2\right\}^{\frac{1}{2}}\left\{\sum_i \|\mu\|_{\Gamma_i^T}^2\right\}^{\frac{1}{2}}\\
&\le C\langle S\mu,\mu\rangle_{\Gamma^T}^{\frac{1}{2}}\|\mu\|_{\Gamma^T}.
\end{aligned}$$

In the next to last inequality above, we used the Cauchy–Schwarz inequality together with the inf–sup condition (4.4.7) and (4.6.3a) to bound $\|p_h^{\Delta t,*}(\mu)\|_{\Omega^T}$ and the elliptic regularity (4.4.14) to bound $\|\mathbf{v}_i\|_{L^2(0,T;\mathbf{V}_i)}$. In the last inequality we used (4.6.7). This concludes the proof. □

4.6.3 GMRES convergence through the field-of-values estimates

Theorem 4.6.1 leads to convergence estimates for solving the interface problem (4.6.5) with GMRES. In [101, Theorem 3.3], a bound is shown for the k-th residual $\mathbf{r}_k$ of the generalized conjugate residual method for solving a system with a positive definite matrix $\mathbf{S}\in\mathbb{R}^{n\times n}$, which also applies to GMRES. It can be stated in terms of angle $\beta\in[0,\pi/2)$, see [95]:

$$\|\mathbf{r}_k\| \le \sin^k(\beta)\|\mathbf{r}_0\|, \quad \text{where } \cos(\beta) = \frac{\lambda_{\min}((\mathbf{S}+\mathbf{S}^T)/2)}{\|\mathbf{S}\|}, \tag{4.6.9}$$

where $\|\cdot\|$ denotes the Euclidean vector norm and the induced matrix norm. The quantities in (4.6.9) can be interpreted in terms of the field-of-values of $\mathbf{S}$, defined as

$$W(\mathbf{S}) = \{\boldsymbol{\zeta}^T \mathbf{S}\, \boldsymbol{\zeta} : \boldsymbol{\zeta} \in \mathbb{C}^n, \|\boldsymbol{\zeta}\| = 1\}.$$

It is known (see [114, Chapter 15]) that $W(\mathbf{S})$ is a compact and convex set in the complex plane that contains (but is usually much larger than) the eigenvalues of $\mathbf{S}$. Because $\mathbf{S}$ is positive definite, $\mathbf{0} \notin W(\mathbf{S})$, and because $\mathbf{S}$ is real, the smallest eigenvalue of the symmetric part of $\mathbf{S}$ is actually the distance from $\mathbf{0}$ to $W(\mathbf{S})$, so that the angle β can be improved to, see [95] or [110, Theorem 2.2.2],

$$\cos(\beta) = \frac{\operatorname{dist}(\mathbf{0}, W(\mathbf{S}))}{\|\mathbf{S}\|}.$$

The above bound, together with inequalities (4.6.8) obtained in Theorem 4.6.1, imply that the reduction in the k-th GMRES residual for solving the interface problem (4.6.5) is bounded by

$$\|\mathbf{r}_k\| \leq \left(\sqrt{1 - (C_0/C_1)^2 h^2}\right)^k \|\mathbf{r}_0\|. \tag{4.6.10}$$

A similar inequality, allowing for an explicit preconditioning matrix, has been obtained in [125].

4.7 Numerical Results

In this section, we present several numerical results obtained from implementation of the space-time mortar method developed in Section 4.3.2, confirming the convergence rate and illustrating other theoretical results obtained in the previous sections.

In all the examples, we take the mixed finite element spaces $\mathbf{V}_{h,i} \times W_{h,i}$ on the space subdomain Ω_i to be the stable $\mathcal{RT}_0 \times Q_0$ (i.e., $k = l = 0$) on a quadrilateral mesh, as discussed in [98]. Combining this with the lowest-order DG (backward Euler, $q = 0$) for time discretization on the mesh $\mathcal{T}_i^{\Delta t}$ gives us a space-time mixed finite element space $\mathbf{V}_{h,i}^{\Delta t} \times W_{h,i}^{\Delta t}$ in Ω_i^T as detailed in Section 4.3.1. Depending on the mortar space-polynomial degree m, we have implemented two different mortar finite element spaces on the space-time interface mesh $\mathcal{T}_{H,ij}^{\Delta T}$, with ΔT suitably chosen as a function of Δt. These are linear mortars $\Lambda_{H,ij,1}^{\Delta T}(m = s = 1)$ and quadratic mortars $\Lambda_{H,ij,2}^{\Delta t}(m = s = 2)$ which are Q_1 and Q_2 respectively.

For solving the interface problem identified in Section 4.6.2, we have implemented the GMRES algorithm with identity preconditioner. Adding a preconditioner to the iterative solver which could significantly reduce the number of iterations and its theoretical analysis could be included in a future project.

All the numerical examples are implemented using the deal.II finite element package [91,92].

4.7.1 Example 1: convergence test

In this example, we solve the parabolic problem (4.2.1) in two spatial dimensions with a known solution to verify the accuracy of the space-time mortar method. We also observe how the number of iterations required for the convergence of GMRES solver is in accordance with the theory discussed. In addition, we also compare the accuracy and computational cost of using discontinuous linear vs quadratic mortar spaces. Finally, we present the computed solutions as 3-dimensional space-time plots to visualize how the weak continuity is enforced across the subdomain interfaces and how continuity is preserved in all three directions. Note that the $z-$axis corresponds to direction in time, t.

We use the known pressure function $p(x,y,t) = \sin(8t)\sin(11x)\cos(11y - \frac{\pi}{4})$ along with permeability $K = I_{2\times 2}$ to manufacture the right-hand side q in (4.2.1) and impose a natural Dirichlet boundary condition on a unit square domain $\Omega = (0,1)^2$. The problem is solved over the time interval $(0, 0.5]$ which gives $\Omega^T = (0,1)^2 \times (0, 0.5]$.

We partition the space domain, Ω, into four identical squares Ω_i and correspondingly Ω^T into four space-time subdomains Ω_i^T, $i = 1,2,3,4$. We start with an initial grid for each Ω_i^T and Γ^T and refine it successively 4 times to test the convergence rate of the solutions with respect to the actual known solution. The subdomains Ω_i^T maintain a checkerboard non-matching mesh structure with $\frac{1}{h_1} : \frac{1}{h_2} : \frac{1}{h_3} : \frac{1}{h_4} = \frac{1}{t_1} : \frac{1}{t_2} : \frac{1}{t_3} : \frac{1}{t_4} = 3 : 2 : 4 : 3$ throughout the refinement cycles (see Table 19). In the case of linear mortars, we employ $H = 2h$ and $\Delta T = 2\Delta t$ and halve the mesh sizes on each refinement cycle. For quadratic mortars, we start with $H : h = \Delta T : \Delta t = 2 : 1$ and refine the mortar mesh only every other time to maintain $H = \sqrt{2h}$ and $\Delta T = \sqrt{2\Delta t}$. We expect the coarser mesh for Γ^T in the quadratic mortar case to be compensated by the higher degree of quadratic mortar space, $\Lambda_{H,ij,2}^{\Delta t}$. More details on the mesh refinement and number of degrees of freedom of spaces $\mathcal{RT}_0 \times Q_0$ on Ω_i and Γ^T is given in Table 19.

Table 19: Example 1, mesh size and #DoFs

Ref.	Ω_1^T			Ω_2^T			Ω_3^T			Ω_4^T			$\Gamma^T(m=1)$			$\Gamma^T(m=2)$		
No.	$\frac{1}{h_1}$	$\frac{1}{\Delta t_1}$	#DoF	$\frac{1}{h_2}$	$\frac{1}{\Delta t_2}$	#DoF	$\frac{1}{h_3}$	$\frac{1}{\Delta t_3}$	#DoF	$\frac{1}{h_4}$	$\frac{1}{\Delta t_4}$	#DoF	$\frac{1}{H}$	$\frac{1}{\Delta T}$	#DoF	$\frac{1}{H}$	$\frac{1}{\Delta T}$	#DoF
0	3	6	33	2	4	16	4	8	56	3	6	33	1	2	8	1	2	18
1	6	12	120	4	8	56	8	16	208	6	12	120	2	4	32			
2	12	24	456	8	16	208	16	32	800	12	24	456	4	8	128	2	4	72
3	24	48	1776	16	32	800	32	64	3136	24	48	1776	8	16	512			
4	48	96	7008	32	64	3136	64	128	12416	48	96	7008	16	32	2048	4	8	288

Table 20: Linear mortar convergence

Ref.	# GMRES		$\|\mathbf{u}-\mathbf{u}_h^{\Delta t}\|_{L^2(0,T;\mathbf{L}^2(\Omega))}$		$\|p-p_h^{\Delta t}\|_{\mathrm{DG}}$		$\|p-p_h^{\Delta t}\|_{L^2(0,T;W)}$		$\|\lambda-\lambda_H^{\Delta T}\|_{L^2(0,T;\Lambda_H)}$	
0	11	Rate	6.50e-01	Rate	1.21e+00	Rate	7.91e-01	Rate	7.98e-01	Rate
1	23	-1.06	3.63e-01	0.84	7.21e-01	0.75	4.76e-01	0.73	5.11e-01	0.64
2	39	-0.76	1.74e-01	1.06	3.19e-01	1.18	2.46e-01	0.95	2.34e-01	1.13
3	59	-0.60	8.63e-02	1.02	1.46e-01	1.13	1.25e-01	0.98	1.20e-01	0.96
4	86	-0.54	4.29e-02	1.01	6.93e-02	1.08	6.25e-02	1.00	6.11e-02	0.97

Table 21: Quadratic mortar convergence

Ref.	# GMRES		$\|\mathbf{u}-\mathbf{u}_h^{\Delta t}\|_{L^2(0,T;\mathbf{L}^2(\Omega))}$		$\|p-p_h^{\Delta t}\|_{\mathrm{DG}}$		$\|p-p_h^{\Delta t}\|_{L^2(0,T;W)}$		$\|\lambda-\lambda_H^{\Delta T}\|_{L^2(0,T;\Lambda_H)}$	
0	18	Rate	6.81e-01	Rate	1.35e+00	Rate	8.39e-01	Rate	2.13e+00	Rate
2	34	-0.46	1.70e-01	1.00	3.51e-01	0.97	2.51e-01	0.87	2.82e-01	1.46
4	57	-0.37	4.48e-02	0.96	8.59e-02	1.02	6.59e-02	0.96	9.20e-02	0.81

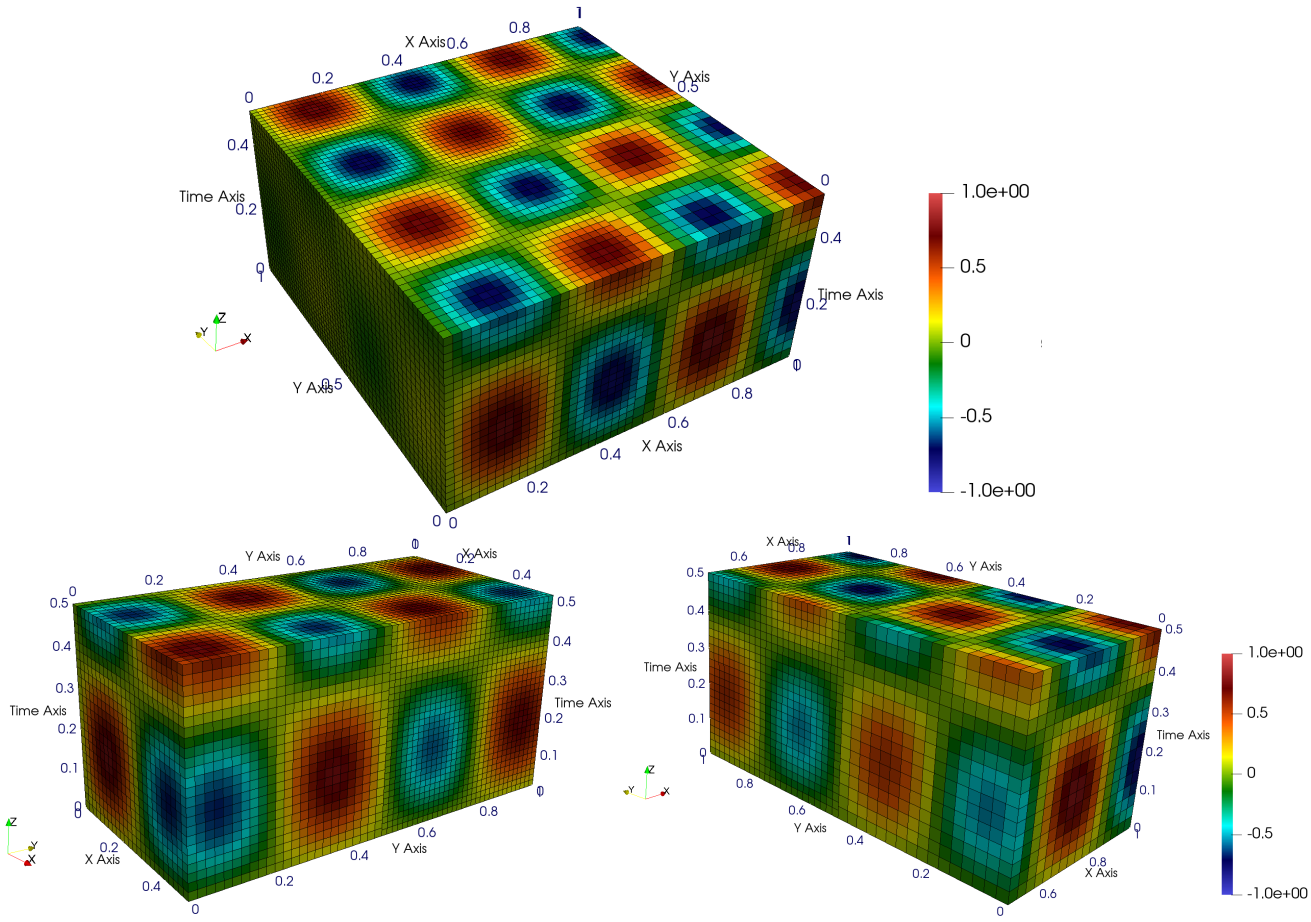

Figure 14: Example 1, pressure computed using linear mortars shown on the space-time grid at refinement 2, top: on the whole space-time domain Ω^T, bottom: on $\Omega_1^T \cup \Omega_4^T$ (left), on $\Omega_2^T \cup \Omega_3^T$ (right).

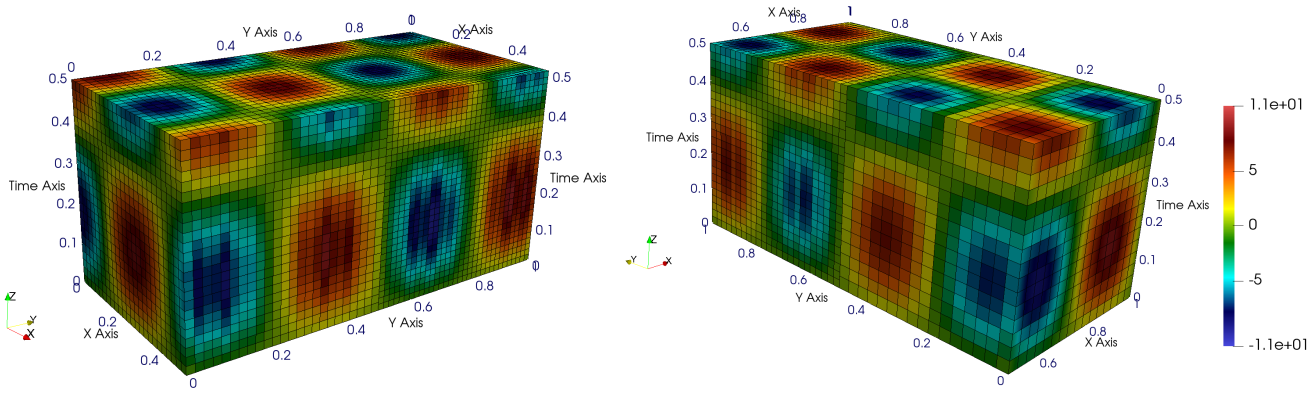

Figure 15: Example 1, $x-$component of velocity computed using linear mortars shown on the space-time grid at refinement 2, on $\Omega_1^T \cup \Omega_4^T$ (left), on $\Omega_2^T \cup \Omega_3^T$ (right).

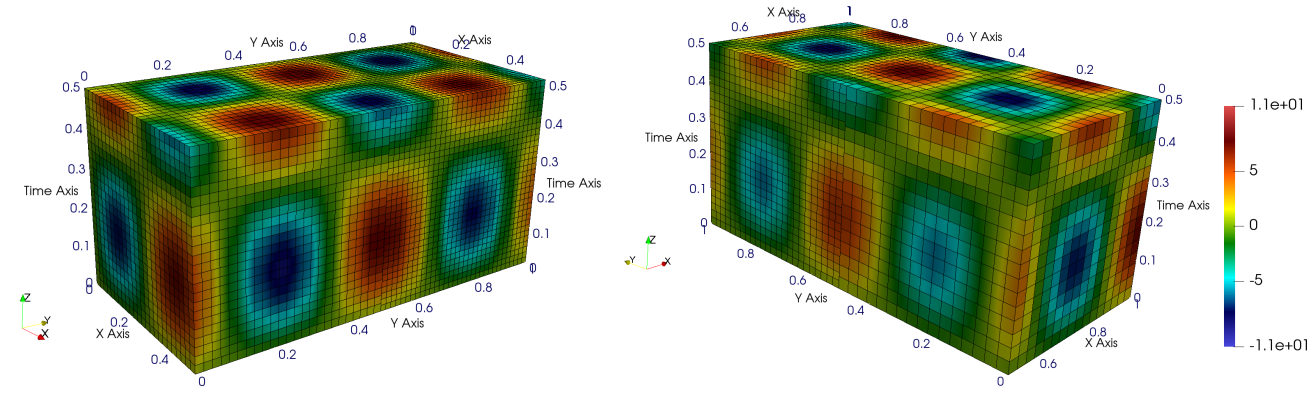

Figure 16: Example 1, $y-$component of velocity computed using linear mortars shown on the space-time grid at refinement 2, on $\Omega_1^T \cup \Omega_4^T$ (left), on $\Omega_2^T \cup \Omega_3^T$ (right).

All the errors reported in Tables 20 and 21 are relative with respective to the norm of the true solution. Also note that the rate of convergence reported are with respect to the orders of h and Δt. We observe optimal rate of convergence of the method with respect to the $\mathcal{RT}_0 \times Q_0$ finite element spaces using both linear and quadratic mortars. Theorem 4.6.1 bounds the spectral ratio of the interface operator, S, by $\mathcal{O}(h^{-1})$ and depending on the deviation of this operator from a normal matrix [115, 116], the growth rate for the number of GMRES iterations required for converge could be bounded by the square root of the spectral ratio, i.e. of $\mathcal{O}(h^{-0.5})$ in our case. This is close to what we observe in the case of linear and quadratic mortars from Tables 20 and 21, respectively. Figures 14–16 clearly show the local conservation of mass (imposition of continuity of the normal flux in the weak multiscale sense) across different subdomain interfaces. Even though quadratic mortar space, $\Lambda^{\Delta t}_{H,ij,2}$ has far fewer degrees of freedom compared to linear mortar space $\Lambda^{\Delta T}_{H,ij,1}$ at the same refinement level for subdomains, we see very comparable errors for these two cases. Also the former results in less number of GMRES iterations (see Tables 20–21). Thus, from a computational point of view, higher mortar degrees m, s will give a computationally less intense and efficient method compared to using smaller m, s. Also the extra $h^{-\frac{1}{2}}$ loss in convergence rate that we see in the convergence result is not observed in the numerical results.

4.7.2 Example 2: problem with a boundary layer

In this example, we demonstrate the advantages of using our multiscale space-time domain decomposition method to a problem where the solution variables, pressure and velocity, vary on different scales across the space-time domain. For this, we use the known solution, $p(x, y, t) = 1000xyte^{-10\left(x^2+y^2+\frac{1}{4}t^2\right)}$ along with permeability $K = I_{2\times 2}$ to manufacture the right-hand side q in (4.2.1) and impose a natural Dirichlet boundary condition on a unit square domain $\Omega = (0, 1)^2$. The problem is solved over the time interval $(0, 0.5]$ which gives $\Omega^T = (0, 1)^2 \times (0, 0.5]$. By construction, $p(x, y, t)$ varies rapidly along the lower-left corner of Ω^T, with almost zero pressure on majority of other corners. This calls for an efficient multiscale method which would take advantage of the multiscale nature of the problem and gives more resolution around the lower-left corner compared to the rest of Ω^T.

Table 22: Example 2, errors for the multiscale and fine-scale methods.

Method	# GMRES	$\|\mathbf{u}-\mathbf{u}_h^{\Delta t}\|_{L^2(0,T;\mathbf{L}^2(\Omega))}$	$\|p-p_h^{\Delta t}\|_{\mathrm{DG}}$	$\|p-p_h^{\Delta t}\|_{L^2(0,T;W)}$	$\|\lambda-\lambda_H^{\Delta T}\|_{L^2(0,T;\Lambda_H)}$
multiscale	102	5.657e-02	8.425e-02	6.319e-02	5.796e-02
fine-scale	140	1.524e-02	2.234e-02	2.154e-02	3.016e-02

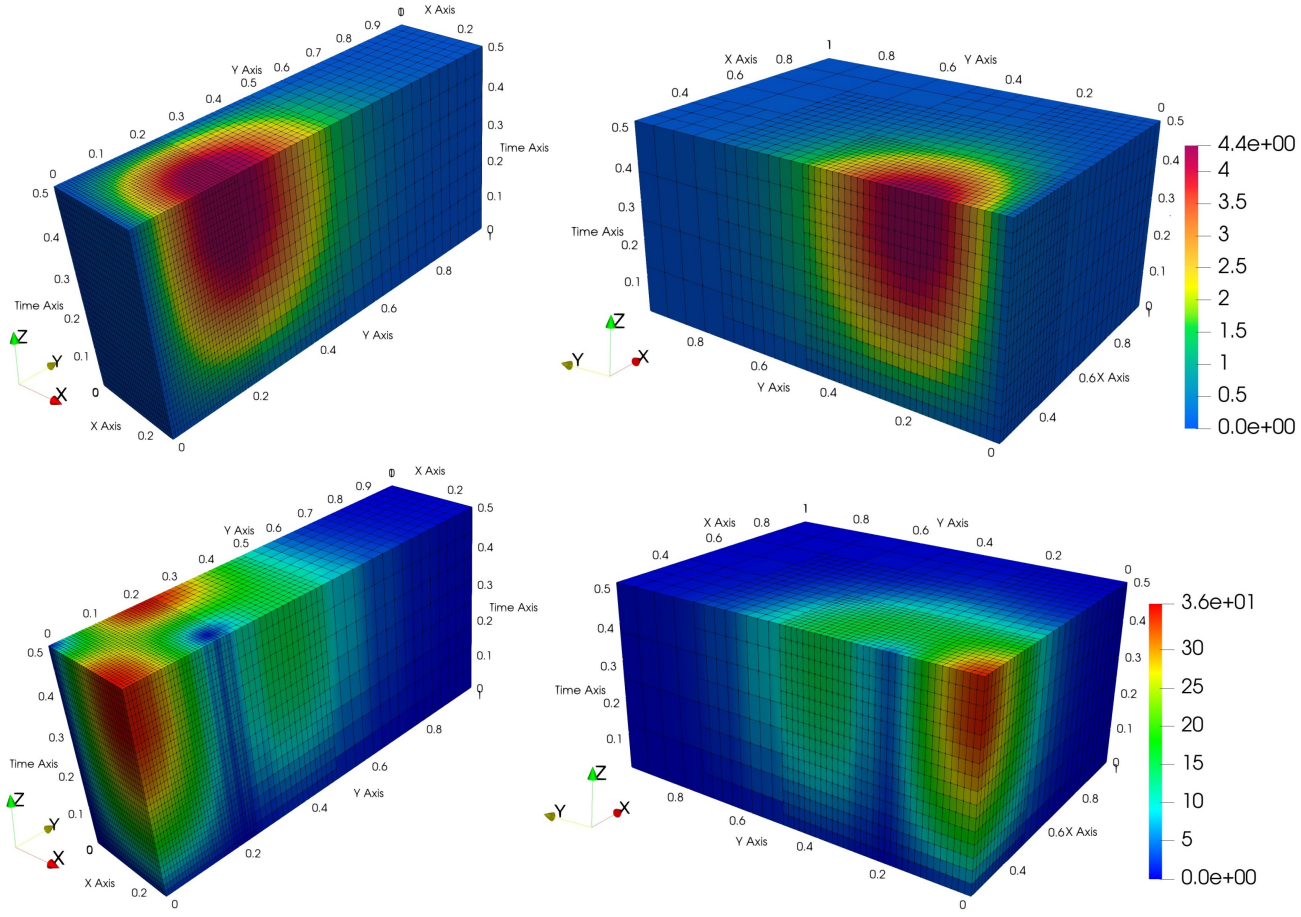

Figure 17: Example 2, pressure from the multiscale method, cut along the plane $x = 0.25$ (top), velocity magnitude from the multiscale method, cut along the plane $x = 0.25$ (bottom).

We partition Ω^T into 4×4 identical square space-time subdomain blocks Ω_i^T. From the knowledge about variation of the true pressure, we use a multiscale space-time grid on Ω^T, where refinement of the grid on each Ω_i^T is proportional to the amount of pressure variation.

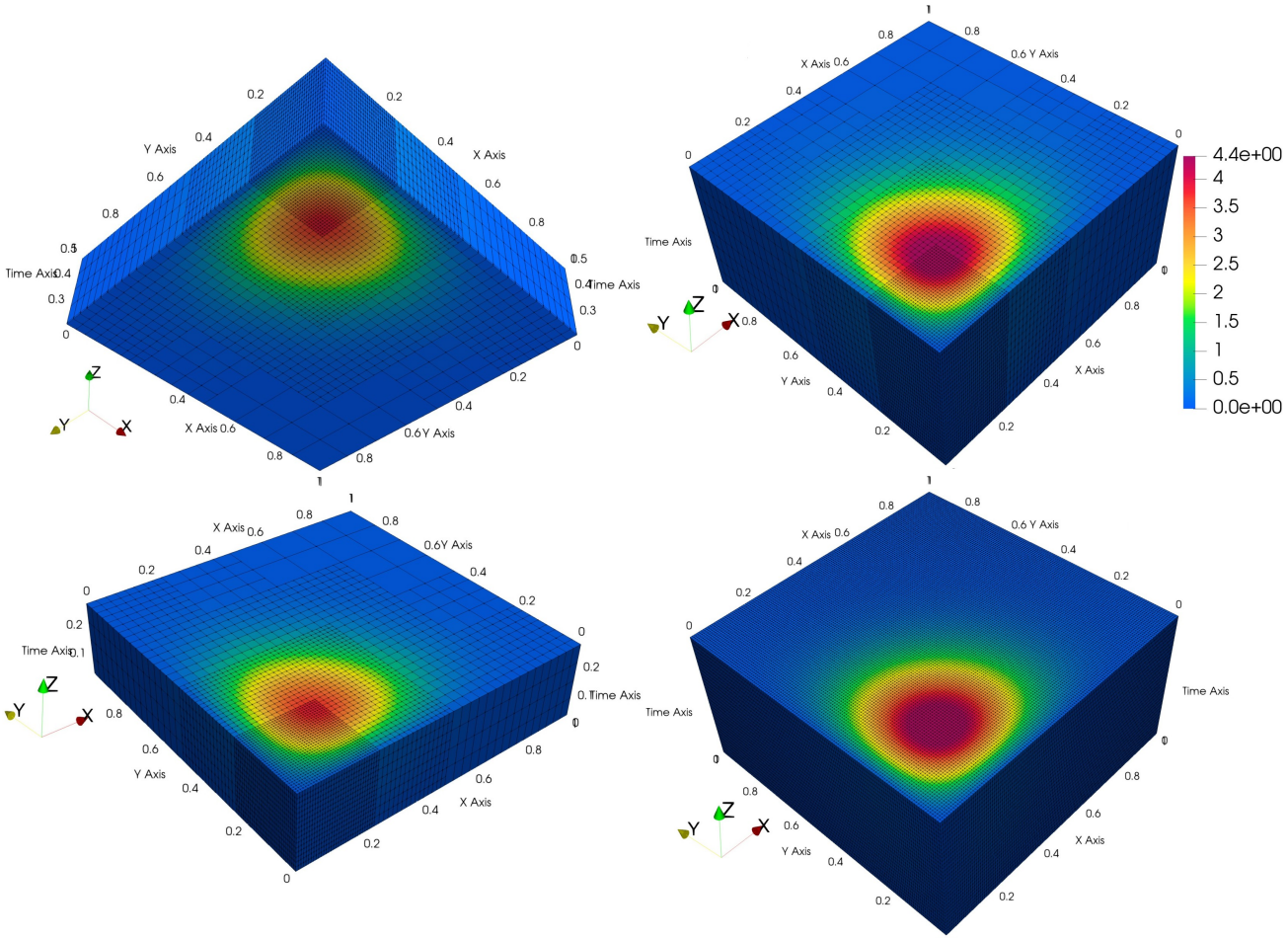

Figure 18: Example 2, left: pressure from the multiscale method, cut along the plane $t = 0.35$; right: pressure from the multiscale (top) and fine-scale (bottom) methods on the whole domain.

The finest mesh on Ω_i^T has $h_{fine} = 1/128$ and $\Delta t_{fine} = 1/64$, and the coarsest mesh on Ω_i^T has $h_{coarse} = 1/8$ and $\Delta t_{coarse} = 1/8$, see Figures 17–18 for the mesh refinement. The coarser meshes on the majority of the space-time subdomains bring down the computational complexity arising from the subdomain solves associated with them. We use a linear mortar ($m = s = 1$) on the subdomain interfaces. The mortar mesh sizes in space are chosen as follows. For vertical interfaces (fixed x) between subdomains on the bottom row, the one along the boundary layer, we set $H = 1/32$. For the next row of subdomains we set $H = 1/16$, and for the other two rows, $H = 1/8$. Similarly, for the horizontal interfaces (fixed y) between subdomains on the left column we set $H = 1/32$, for the second column, $H = 1/16$, and for the other two columns, $H = 1/8$. We

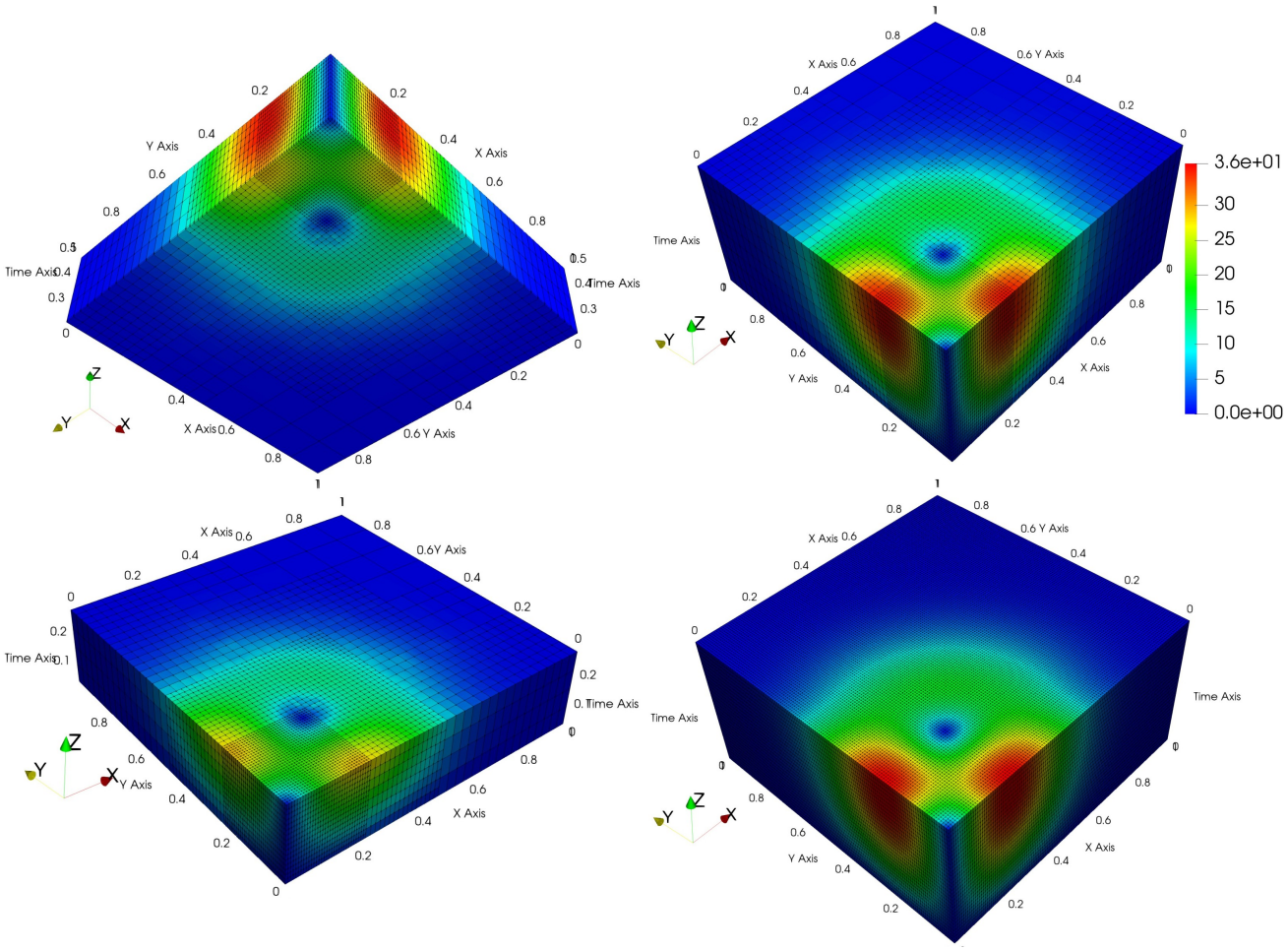

Figure 19: Example 2, left: velocity magnitude from the multiscale method, cut along the plane $t = 0.35$; right: velocity magnitude from the multiscale (top) and fine-scale (bottom) methods on the whole domain.

choose $\Delta T = 1/8$ on all interfaces. These choices guarantee that the mortar assumption (4.4.6) is satisfied and that the dimension of the interface problem is reduced, while at the same time provide suitable resolution to enforce weakly flux continuity across the space-time subdomain interfaces.

For comparison, we solve the problem using a uniformly fine and matching subdomain mesh with $h = H = 1/128$ and $\Delta t = \Delta T = 1/64$. A comparison of the number of GMRES iterations and the relative errors from the multiscale and the fine-scale methods is given in Table 22. A detailed demonstration of the enforcement of continuity of pressure and velocity computed using

the multiscale method is given in Figures 17–19. Side to side comparison of the multiscale and fine-scale solutions are given on the right sides in Figure 18 and Figure 19.

Table 22 shows that both the multiscale and the fine-scale solution attains comparable accuracy with the former being computationally far less expensive than the latter. We observe smaller relative error in the case of fine-scale solution method because of the matching grids and higher resolution throughout the space-time domain, Ω^T. Slightly higher error for the multiscale method is compensated with cheaper subdomain solves and smaller interface problem which converges faster compared to the fine-scale method. Figures 17–19 show good enforcement of continuity across various space and time interfaces for the multiscale method. The comparisons in Figure 18 and Figure 19 show that the multiscale method provides good resolution where it matters and once again confirm that the less expensive multiscale method provides comparable accuracy to the more expensive fine-scale method.

4.8 Chapter Conclusions

In this chapter, we presented a multiscale space-time discretization technique for efficiently solving a model parabolic problem. This method is the generalization of the multiscale mortar mixed finite element (MMMFE) technique introduced in Chapter 3 for a time-dependent parabolic system, where we allow multiscale discretization in both space and time. We decompose the global space-time domain into multiple space-time subdomains and introduce a space-time mortar variable, on an independent interface space-time mesh. This method involves solving an interface problem to ensure (a multiscale) weak continuity of the normal component of the mixed finite element flux variable over the space-time interfaces. We have shown the well-posedness and stability of the technique along with a combined a priori error estimate. Various numerical experiments were conducted to confirm the theoretical results and demonstrate the advantage of using a multiscale space-time domain decomposition method. We conclude that methods like this offer a high level of flexibility in choosing the level of discretization in both space and time dimensions. This flexibility can be exploited to our advantage while developing a numerical method to solve multiphysics problems where the solution varies on extremely different scales across the spatio-temporal domain, as demonstrated in the numerical results section.

5.0 Conclusions

5.1 Summary of Techniques Developed and Results

In this thesis, we have developed various numerical techniques to efficiently solve the Biot system of poroelasticity in the setting of the mixed finite element (MFE) methods. We have also developed a novel space-time domain decomposition technique for the time-dependent second-order parabolic equation, which can be extended to the setting of the monolithic Biot system of poroelasticity in future studies.

In Chapter 2, we have presented three non-overlapping domain decomposition methods for the Biot system of poroelasticity in a five-field fully mixed formulation using matching subdomain grids at the interface. The monolithic method involves solving an interface problem for a composite displacement-pressure Lagrange multiplier, which requires coupled Biot subdomain solves at each iteration. The two split methods are based on the drained split and fixed stress splittings. They involve two separate elasticity and Darcy interface iterations requiring single-physics subdomain solves. We analyze the spectrum of the monolithic interface operator and show unconditional stability for the split methods. A series of numerical experiments illustrate the efficiency, accuracy, and robustness of the three methods. Our main conclusion from this chapter is that the split methods provide accuracy comparable to the monolithic method while being more computationally efficient in terms of the smaller number of interface iterations and simpler subdomain solves.

In Chapter 3, we presented a multiscale mortar mixed finite element technique (MMMFE) for the Biot system of poroelasticity in a five-filed fully mixed formulation. This method is the generalization of the monolithic domain decomposition technique discussed in the previous chapter, with the extra capability to use non-matching subdomain grids at the interface. This is achieved by using composite multiscale mortar Lagrange multiplier spaces approximating displacement and pressure on a coarse mortar grid at the interface. The global problem can be reduced into a series of parallel Dirichlet type problems and an interface problem for the composite displacement-pressure Lagrange multiplier spaces which requires subdomain solves at each iteration. We showed the well-posedness and stability of the method under proper

assumptions. We have also carried out an extensive error analysis of the method to get a combined a priori error estimate for all the unknowns in the formulation. To complete the analysis, we have done a series of numerical experiments to put the theory to test. We observed stability and convergence results as predicted by the theory and also demonstrated the application of the method to a highly heterogeneous medium. We noted that in practice, a coarser mesh with a higher mortar space degree can be used to get a smaller interface problem and hence faster convergence without compromising the accuracy of the method. We conclude the chapter by recalling the effectiveness of the construction and use of a pre-saved multiscale stress-flux basis (MSB), which makes the MMMFE method far more superior than the fine-scale monolithic methods, especially when a coarse mortar mesh is used.

In Chapter 4, we developed a multiscale space-time discretization technique for efficiently solving a model parabolic equation. This method is the generalization of the multiscale mortar mixed finite element (MMMFE) technique introduced in Chapter 3 for a time-dependent parabolic system, where we allow multiscale discretization in both space and time. We decompose the global space-time domain into multiple space-time subdomains and introduce a space-time mortar variable, on an independent interface space-time mesh. This method involves solving an interface problem to ensure (a multiscale) weak continuity of the normal component of the mixed finite element flux variable over the space-time interfaces. We have shown the well-posedness and stability of the technique along with a combined a priori error estimate. Various numerical experiments were conducted to confirm the theoretical results and demonstrate the advantage of using a multiscale space-time domain decomposition method. We conclude that methods like this offer a high level of flexibility in choosing the level of discretization in both space and time dimensions. This flexibility can be exploited to our advantage while developing a numerical method to solve multiphysics problems where the solution varies on extremely different scales across the spatio-temporal domain, as demonstrated in the numerical results section.

5.2 Future Work

Techniques developed in Chapter 4 for the parabolic equation paves the way for future works on developing a space-time discretization technique for the Biot system of poroelastic equations.

While the Biot system of equations is certainly more challenging and complex than the parabolic system, we believe the techniques developed to mathematically analyze the latter will prove to be crucial in the analysis of the former.

Other possible works include analyzing the condition number for the interface operator developed in Chapter 3. It will be also interesting to study how the condition number for the interface operators for all the techniques developed in this thesis depends on the subdomain size or in the case of multiscale methods, the mortar element size H. The use of a coarse solve preconditioner to speed up the convergence of the interface iterations is also worth pursuing.

We are currently engaged in the study of employing machine learning (ML) techniques in our multiscale domain decomposition algorithms for improved computational efficiency. We are specifically interested in the recently developed physics-informed neural networks (PINNs) which incorporate PDE information into the loss function of a neural network. After training, these methods provide fast PDE solvers that can be used as an alternative to finite element solvers. Despite the growing evidence in the scientific literature of the robustness and efficiency of PINNs, there is still an incomplete understanding of their accuracy as PDE solvers, as well as the sense in which the computed solution satisfies fundamental physical laws such as mass conservation. We plan to investigate these issues in the context of PINNs based on mixed formulations of the underlying PDEs. We believe that the PINNs in it's mixed form could tackle the issue of vanishing/ exploding gradients which allows the use of a more robust activation function like the rectified linear unit (ReLU) function. We have recently developed an open-source software package FluidLearn (see Appendix A.2), designed to solve PDEs using supervised deep learning techniques, and specifically feed-forward PINNs in the mixed form. We plan to employ the package as a subdomain solver as part of our space-time domain decomposition algorithm. Another possible path of study is to explore the use of transfer learning techniques to improve the training efficiency of the subdomain PINNs by reusing previously trained neural nets.

Appendix Code Gallery

A.1 Note to the Reader

As it is with the development of any software based on novel algorithms, a significant amount of time and effort went into the production of software capable of solving PDEs using the methods developed in this thesis. All the packages are written in *C++* using deal.II finite element package [91, 92]. These packages are made open-source and are available on the GitHub page *https://github.com/mjayadharan*. All the numerical results presented in this thesis are generated using simulators published as open-source repositories on the aforementioned web page. Anyone who is interested in using or do development based on these packages is encouraged to do so with citation to this thesis wherever relevant. Details on installation and usage of these packages can be found through the *README* file in the appropriate repositories (e.g. see this link).

As it is an impossible task to list the thousands of lines of code written to implement our methods, we give hyperlinks to some of the core packages and we also give a stripped-down version of the main algorithm implementation for the multiscale mortar space-time domain decomposition method (see Chapter 4). The reason for presenting this particular implementation in the thesis is that the multiscale space-time domain decomposition method implementation is the most generic method we have developed in this thesis and incorporates techniques used in the other methods.

A.2 Links to Open-source Packages Corresponding to Various Chapters

1. **Chapter 2**: A base-repository that implements the non-overlapping domain decomposition technique for the Biot system of poroelasticity using matching subdomain grids, with the option to use sequential splitting.
2. **Chapter 3**: Package to solve the Biot system of poroelasticity using non-overlapping domain decomposition method that is also capable of using multiscale non-matching subdomain grids, with the option to use sequential splitting. Note that this package is inherited from

the above base-repository.

3. **Chapter 4**: Package implementing the multiscale space-time mortar domain decomposition method for a time-dependent parabolic model.
4. **Ongoing work**: **FluidLearn**: software package with python interface, capable of solving non-linear fluid flow problems using supervised deep learning (DL) techniques.

A.3 Implementation of the Space-time Multiscale Mortar Decomposition Method

Here we give a stripped-down version of the implementation of the space-time multiscale mortar decomposition technique for the time-dependent parabolic problem (1.3.10). A complete version of the package can be found on GitHUB. Detailed user instructions are given in the README file.

A.3.1 User interface

Once the package is compiled using the instructions given in the README file, the user can use an interface *.txt* file to interact with the simulator. The implementation details are hidden from the user. An example of a user interaction file is given below.

```
/*
* User input file
*--------------------------------
*/
   c0: 1.0

   alpha: 2.0

   coe_a: 0.5

   space_degree: 0

   mortar_degree: 1

   num_refinement: 1

   final_time: 1.0

   tolerence: 1.e-6
```

```
max_iteration: 500

need_plot_at_each_time_step(bool): 0

left_bc: D 0.0
bottom_bc: N 0.0
right_bc: D 0.0
top_bc: N 1.0

is_manufact_soln(bool): 0

mesh_pattern_sub_d0: 8 8 16
mesh_pattern_sub_d1: 12 12 24
mesh_pattern_sub_d2: 12 12 24
mesh_pattern_sub_d3: 8 8 16
mesh_pattern_mortar: 2 2 2

mesh_pattern_sub_d0: 8 8 16
mesh_pattern_sub_d1: 8 8 16
mesh_pattern_sub_d2: 8 8 16
mesh_pattern_sub_d3: 8 8 16
mesh_pattern_mortar: 8 8 16
```

We use the following main function to drive the simulator.

```
// Utilities, data, etc.
#include "../inc/darcy_vtdd.h"
#include "../inc/filecheck_utility.h"
#include <fstream>
#include <string>
#include <cassert>

int main (int argc, char *argv[])
{
    try
    {
        using namespace dealii;
        using namespace vt_darcy;

        MultithreadInfo::set_thread_limit(4);
        Utilities::MPI::MPI_InitFinalize mpi_initialization(argc, argv, 1);
        double c_0, alpha, coe_a, final_time, tolerence;
        int space_degree, mortar_degree, num_refinement, max_iteration;

        //declaring mesh refinement structure for space-time mortar
        std::vector<int> zeros_vector(3,0);
        std::vector<std::vector<int>> mesh_m3d, mesh_m3d_mortar;
```

```
//boundary condition vector. 'D':Dirichlet, 'N': Neumann
std::vector<char> bc_con(4,'D');
std::vector<double> nm_bc_con_funcs(4,0.0);

bool is_manufact_solution, need_each_time_step_plot;
std::string dummy_string;

/*
 * Block for pulling in parameters and other desired program features from a
    parameter file
 */
{
   MPI_Comm mpi_communicator_1(MPI_COMM_WORLD);
   MPI_Status mpi_status_1;
   int mpi_send_bool(0), mpi__rec_bool(0);
 const unsigned int this_mpi =
    Utilities::MPI::this_mpi_process(mpi_communicator_1);
 const unsigned int n_processes =
    Utilities::MPI::n_mpi_processes(mpi_communicator_1);

 mesh_m3d.resize(n_processes+1, zeros_vector);
 mesh_m3d_mortar.resize(n_processes+1, zeros_vector);


 if(this_mpi!=0)
 {
     MPI_Recv(&mpi__rec_bool, 1, MPI_INT, this_mpi-1, this_mpi-1,
        mpi_communicator_1, &mpi_status_1);

 }
 parameter_pull_in (c_0, alpha, coe_a, space_degree, mortar_degree,
    num_refinement,
      final_time, tolerence, max_iteration, need_each_time_step_plot,
      bc_con, nm_bc_con_funcs, is_manufact_solution, mesh_m3d,
         mesh_m3d_mortar, n_processes, "parameter.txt");

 if(this_mpi!=n_processes-1)
        {
            MPI_Send(&mpi_send_bool, 1, MPI_INT, this_mpi+1, this_mpi,
               mpi_communicator_1);

        }

}

BiotParameters bparam (1.0, 1, final_time, c_0, alpha, coe_a);
//Instantiating the class
DarcyVTProblem<2> problem_2d(space_degree, bparam, 1, mortar_degree, bc_con,
   nm_bc_con_funcs, is_manufact_solution, need_each_time_step_plot);
```

```
        //Solving the problem
        problem_2d.run(num_refinement, mesh_m3d, mesh_m3d_mortar,
            tolerence, max_iteration, mortar_degree+1);
    }
    catch (std::exception &exc)
    {
        "..catching exceptions.."
   }
}
```

A.3.2 Source code

Here we present the source code for the *DarcyVTProblem* class which encapsulates all the data structures and algorithms needed to implement the method. Note that this class depends on various other utility and data files which can be found in the code repository. Also, here we omit the implementation of various class methods, like the one for error calculations, that are not part of the core algorithm.

```
/* ---------------------------------------------------------------------
 * Importing dependencies
 * ---------------------------------------------------------------------
 */
#include <deal.II/base/quadrature_lib.h>
#include <deal.II/base/logstream.h>
#include <deal.II/lac/block_vector.h>
#include <deal.II/lac/full_matrix.h>
#include <deal.II/base/function.h>
#include <deal.II/lac/block_sparse_matrix.h>
#include <deal.II/lac/solver_cg.h>
#include <deal.II/lac/sparse_direct.h>
#include <deal.II/lac/precondition.h>
#include <deal.II/grid/tria.h>
#include <deal.II/grid/grid_generator.h>
#include <deal.II/grid/tria_accessor.h>
#include <deal.II/grid/tria_iterator.h>
#include <deal.II/grid/grid_in.h>
#include <deal.II/grid/grid_tools.h>
#include <deal.II/dofs/dof_handler.h>
#include <deal.II/dofs/dof_renumbering.h>
#include <deal.II/dofs/dof_accessor.h>
#include <deal.II/dofs/dof_tools.h>
#include <deal.II/fe/fe_dgq.h>
#include <deal.II/fe/fe_dgp.h>
```

```
#include <deal.II/fe/fe_face.h>
#include <deal.II/fe/fe_raviart_thomas.h>
#include <deal.II/fe/fe_bdm.h>
#include <deal.II/fe/fe_nothing.h>
#include <deal.II/numerics/vector_tools.h>
#include <deal.II/numerics/matrix_tools.h>
#include <deal.II/numerics/data_out.h>
// Extra for MPI and mortars
#include <deal.II/numerics/fe_field_function.h>
#include <deal.II/base/timer.h>
// C++
#include <fstream>
#include <iostream>
#include <random>
// Utilities, data, etc.
#include "../inc/darcy_vtdd.h"
#include "../inc/utilities.h"
#include "../inc/data.h"
/* ---------------------------------------------------------------------
 * Declaration of the DarcyVTProblem class
 * ---------------------------------------------------------------------
 */
template<int dim = 2>
class DarcyVTProblem {
public:
   DarcyVTProblem(const unsigned int degree, const BiotParameters& bprm,
         const unsigned int mortar_flag = 0,
         const unsigned int mortar_degree = 0,
         std::vector<char> bc_condition_vect = { 'D', 'D', 'D', 'D' },
         std::vector<double> bc_const_functs = { 0., 0., 0., 0. },
         const bool is_manufact_soln = true,
         const bool need_each_time_step_plot = false);

   void run(const unsigned int refine,
         const std::vector<std::vector<int>> &reps_st,
         const std::vector<std::vector<int>> &reps_st_mortar, double tol,
         unsigned int maxiter, unsigned int quad_degree = 3);

private:
   MPI_Comm mpi_communicator;
   MPI_Status mpi_status;

   Projector::Projector<dim + 1> P_coarse2fine;
   Projector::Projector<dim + 1> P_fine2coarse;

   void make_grid_and_dofs();
   void assemble_system();
   void get_interface_dofs();
   void get_interface_dofs_st();
```

```
void assemble_rhs_bar();
void assemble_rhs_star();
void solve_bar();
void solve_star();
void solve_timestep(int star_bar_flag, unsigned int time_level);
void solve_darcy_vt(unsigned int maxiter);

void compute_multiscale_basis();
std::vector<double> compute_interface_error_dh();
double compute_interface_error_l2();
double compute_jump_error();
void compute_errors(const unsigned int refinement_index,
     unsigned int time_level);
void output_results(const unsigned int cycle, const unsigned int refine,
     const unsigned int time_level);
void set_current_errors_to_zero();
void reset_mortars();
//For implementing GMRES
void
givens_rotation(double v1, double v2, double &cs, double &sn);
void
apply_givens_rotation(std::vector<double> &h, std::vector<double> &cs,
     std::vector<double> &sn, unsigned int k_iteration);
void
back_solve(std::vector<std::vector<double>> H, std::vector<double> beta,
     std::vector<double> &y, unsigned int k_iteration);
void
local_gmres(const unsigned int maxiter);
double vect_norm(std::vector<double> v);
//distribute solution vectors between 2-d space and 3-d space-time subdomain
   meshes.
void st_to_subdom_distribute(BlockVector<double> &vector_st,
     BlockVector<double> &vector_subdom, unsigned int &time_level,
     double scale_factor);
void subdom_to_st_distribute(BlockVector<double> &vector_st,
     BlockVector<double> &vector_subdom, unsigned int &time_level,
     double scale_factor);
//distribute local to global solution.
void final_solution_transfer(BlockVector<double> &solution_st,
     BlockVector<double> &solution_subdom, unsigned int &time_level,
     double scale_factor);
// Number of subdomains in the computational domain
std::vector<unsigned int> n_domains;
// Physical parameters
BiotParameters prm;
BiotErrors err;
std::vector<char> bc_condition_vect;
std::vector<double> bc_const_functs;
const bool is_manufact_solution;
```

```
const bool need_each_time_step_plot;
std::vector<int> dir_bc_ids, nm_bc_ids;
// FE degree and DD parameters
const unsigned int degree;
const unsigned int mortar_degree;
const unsigned int mortar_flag;
unsigned int gmres_iteration;
double grid_diameter;
unsigned int cg_iteration;
unsigned int max_cg_iteration;
double tolerance;
unsigned int qdegree;
unsigned int refinement_index;
unsigned int total_refinements;
// Neighbors and interface information
std::vector<int> neighbors;
std::vector<unsigned int> faces_on_interface;
std::vector<unsigned int> faces_on_interface_mortar;
std::vector<unsigned int> faces_on_interface_st;
std::vector<std::vector<unsigned int>> interface_dofs;
std::vector<std::vector<unsigned int>> interface_dofs_subd;
std::vector<std::vector<unsigned int>> interface_dofs_st;
std::vector<std::vector<unsigned int>> face_dofs_st;
std::vector<std::vector<unsigned int>> face_dofs_subdom;

unsigned long n_flux;
unsigned long n_pressure;
unsigned long n_flux_st;
unsigned long n_pressure_st;

// Subdomain coordinates
Point<dim> p1;
Point<dim> p2;
// space-time grid diagonal coordinates
Point<dim + 1> p1_st;
Point<dim + 1> p2_st;

// Fine triangulation
Triangulation<dim> triangulation;
FESystem<dim> fe;
DoFHandler<dim> dof_handler;

//3d Space time triangulation for subdomain.
Triangulation<dim + 1> triangulation_st;
FESystem<dim + 1> fe_st;
DoFHandler<dim + 1> dof_handler_st;

// Mortar triangulation
Triangulation<dim + 1> triangulation_mortar;
```

```
  FESystem<dim + 1> fe_mortar;
  DoFHandler<dim + 1> dof_handler_mortar;

  // Star and bar problem data structures
  BlockSparsityPattern sparsity_pattern;
  BlockSparseMatrix<double> system_matrix;
  SparseDirectUMFPACK A_direct;

  BlockVector<double> solution_bar;
  BlockVector<double> solution_star;
  BlockVector<double> solution;
  BlockVector<double> solution_st;

  BlockVector<double> old_solution;
  BlockVector<double> old_solution_for_jump;
  BlockVector<double> initialc_solution;

  BlockVector<double> pressure_projection;
  BlockVector<double> old_pressure_projection;

  BlockVector<double> system_rhs_bar;
  BlockVector<double> system_rhs_bar_bc;
  BlockVector<double> system_rhs_star;
  BlockVector<double> interface_fe_function_subdom;
  AffineConstraints<double> constraint_bc;

  // Mortar data structures
  BlockVector<double> interface_fe_function_mortar;
  BlockVector<double> solution_bar_mortar;
  BlockVector<double> solution_star_mortar;
  std::vector<BlockVector<double>> multiscale_basis;

  // 3d Space-time data structures
  BlockVector<double> interface_fe_function_st;
  BlockVector<double> solution_bar_st;
  BlockVector<double> solution_star_st;
  std::vector<BlockVector<double>> solution_bar_collection;

  // Output extra
  ConditionalOStream pcout;
  ConvergenceTable convergence_table;
  TimerOutput computing_timer;
};
}

#endif

/*
 * ---------------------------------------------------------------------
```

```
 * Definition of class methods
 * ---------------------------------------------------------------------
 */


namespace vt_darcy {
using namespace dealii;

// DarcyVTProblem class constructor
template<int dim>
DarcyVTProblem<dim>::DarcyVTProblem(
   const unsigned int degree,
   const BiotParameters &bprm,
   const unsigned int mortar_flag,
   const unsigned int mortar_degree,
   std::vector<char> bc_condition_vect,
   std::vector<double> bc_const_functs,
   const bool is_manufact_soln,
   const bool need_each_time_step_plot,
   mpi_communicator(MPI_COMM_WORLD),
   P_coarse2fine(false),
   P_fine2coarse(false),
   n_domains(dim, 0),
   prm(bprm), bc_condition_vect(
   bc_condition_vect),
   bc_const_functs(bc_const_functs),
   is_manufact_solution(
   is_manufact_soln),
   need_each_time_step_plot(
   need_each_time_step_plot),
   degree(degree), mortar_degree(mortar_degree),
   mortar_flag(mortar_flag),
   gmres_iteration(0),
   grid_diameter(0),
   cg_iteration(0),
   max_cg_iteration(0),
   qdegree(11),
   fe(FE_RaviartThomas<dim>(degree), 1, FE_DGQ<dim>(degree), 1),
   dof_handler(triangulation),
   fe_st(FE_RaviartThomas<dim + 1>(degree), 1,FE_DGQ<dim + 1>(degree), 1),
   dof_handler_st(triangulation_st),
   fe_mortar(FE_RaviartThomas<dim + 1>(mortar_degree), 1, FE_Nothing<dim + 1>(), 1),
   dof_handler_mortar(triangulation_mortar),
   pcout(std::cout, (Utilities::MPI::this_mpi_process(mpi_communicator) == 0)),
   computing_timer(mpi_communicator, pcout, TimerOutput::summary,
   TimerOutput::wall_times)
{}
//Method to make grids and DOFs
template<int dim>
```

```
void DarcyVTProblem<dim>::make_grid_and_dofs() {

   TimerOutput::Scope t(computing_timer, "Make grid and DoFs");
   system_matrix.clear();
   const unsigned int this_mpi = Utilities::MPI::this_mpi_process(
         mpi_communicator);
   // Find neighbors
   neighbors.resize(GeometryInfo<dim>::faces_per_cell, 0);
   find_neighbors(dim, this_mpi, n_domains, neighbors);

   // Make interface data structures
   faces_on_interface.resize(GeometryInfo<dim>::faces_per_cell, 0);
   faces_on_interface_mortar.resize(GeometryInfo<dim>::faces_per_cell, 0);
   faces_on_interface_st.resize(GeometryInfo<dim>::faces_per_cell, 0);
   mark_interface_faces(triangulation, neighbors, p1, p2, faces_on_interface);
   if (mortar_flag) {
      mark_interface_faces_space_time(triangulation_mortar, neighbors, p1, p2,
            faces_on_interface_mortar);
      mark_interface_faces_space_time(triangulation_st, neighbors, p1, p2,
            faces_on_interface_st);
   }
   dof_handler.distribute_dofs(fe);
   DoFRenumbering::component_wise(dof_handler);
   if (mortar_flag) {
      dof_handler_mortar.distribute_dofs(fe_mortar);
      DoFRenumbering::component_wise(dof_handler_mortar);

      dof_handler_st.distribute_dofs(fe_st);
      DoFRenumbering::component_wise(dof_handler_st);
   }
   std::vector<types::global_dof_index> dofs_per_component(dim + 1);
   DoFTools::count_dofs_per_component(dof_handler, dofs_per_component);
   unsigned int n_z = dofs_per_component[0];
   unsigned int n_p = dofs_per_component[dim];
   n_flux = n_z;
   n_pressure = n_p;
   //Adding essential Neumann BC
   {
      constraint_bc.clear();
      if (!is_manufact_solution) {
         for (unsigned int i = 0; i < bc_condition_vect.size(); ++i) {
            if (bc_condition_vect[i] == 'D')
               dir_bc_ids.push_back(100 + i + 1);
            else if (bc_condition_vect[i] == 'N')
               nm_bc_ids.push_back(100 + i + 1);

            std::map<types::boundary_id, const Function<dim> *> velocity_bc;
            std::map<types::global_dof_index, double> boundary_values_velocity;
            std::vector<double> zero_std_vect(3, 0.);
```

```
   Vector<double> zero_dealii_vect(zero_std_vect.begin(),
        zero_std_vect.end());
   std::vector<Vector<double>> const_funct_base(4,
        zero_dealii_vect);
   const_funct_base[0][0] = -1.0 * bc_const_functs[0];
   const_funct_base[1][1] = -1.0 * bc_const_functs[1];
   const_funct_base[2][0] = 1.0 * bc_const_functs[2];
   const_funct_base[3][1] = 1.0 * bc_const_functs[3];

   Functions::ConstantFunction<dim> const_fun_left(
        const_funct_base[0]), const_fun_bottom(
        const_funct_base[1]);
   Functions::ConstantFunction<dim> const_fun_right(
        const_funct_base[2]), const_fun_top(
        const_funct_base[3]);
   std::vector<Functions::ConstantFunction<dim>> velocity_const_funcs(
        4, const_fun_left);
   velocity_const_funcs[0] = const_fun_left;
   velocity_const_funcs[1] = const_fun_bottom;
   velocity_const_funcs[2] = const_fun_right;
   velocity_const_funcs[3] = const_fun_top;

   //Feeding the Neumann boundary values into the constraint matrix
   Functions::ZeroFunction<dim> velocity_bc_func(dim + 1);
   for (unsigned int i = 0; i < nm_bc_ids.size(); ++i)
      velocity_bc[nm_bc_ids[i]] =
           &velocity_const_funcs[nm_bc_ids[i] - 101];
   VectorTools::project_boundary_values(dof_handler, velocity_bc,
        QGauss<dim - 1>(degree + 3), boundary_values_velocity);

   typename std::map<types::global_dof_index, double>::const_iterator
       boundary_value_vel =
        boundary_values_velocity.begin();
   for (; boundary_value_vel != boundary_values_velocity.end();
        ++boundary_value_vel) {
      if (!constraint_bc.is_constrained(
           boundary_value_vel->first)) {
        constraint_bc.add_line(boundary_value_vel->first);
        constraint_bc.set_inhomogeneity(
             boundary_value_vel->first,
             boundary_value_vel->second);
      }
   }
}
else
for (int i = 0; i < 4;
     ++i
   )
   dir_bc_ids.push_back(101 + i);
```

```
}
constraint_bc.close();
BlockDynamicSparsityPattern dsp(2, 2);
dsp.block(0, 0).reinit(n_z, n_z);
dsp.block(1, 0).reinit(n_p, n_z);
dsp.block(0, 1).reinit(n_z, n_p);
dsp.block(1, 1).reinit(n_p, n_p);
dsp.collect_sizes();
DoFTools::make_sparsity_pattern(dof_handler, dsp, constraint_bc,
     false);

// Initialize system matrix
sparsity_pattern.copy_from(dsp);
system_matrix.reinit(sparsity_pattern);

// Reinit solution and RHS vectors
solution_bar.reinit(2);
solution_bar.block(0).reinit(n_z);
solution_bar.block(1).reinit(n_p);
solution_bar.collect_sizes();
solution_bar = 0;

// Reinit solution and RHS vectors
solution_star.reinit(2);
solution_star.block(0).reinit(n_z);
solution_star.block(1).reinit(n_p);
solution_star.collect_sizes();
solution_star = 0;

system_rhs_bar.reinit(2);
;
system_rhs_bar.block(0).reinit(n_z);
system_rhs_bar.block(1).reinit(n_p);
system_rhs_bar.collect_sizes();
system_rhs_bar = 0;

// Required for essential(Neumann bc)
system_rhs_bar_bc.reinit(2);
system_rhs_bar_bc.block(0).reinit(n_z);
system_rhs_bar_bc.block(1).reinit(n_p);
system_rhs_bar_bc.collect_sizes();
system_rhs_bar_bc = 0;

system_rhs_star.reinit(2);
system_rhs_star.block(0).reinit(n_z);
system_rhs_star.block(1).reinit(n_p);
system_rhs_star.collect_sizes();
system_rhs_star = 0;
```

```
//adding vectors required for storing mortar and space-time subdomain
    solutions.
if (mortar_flag) {
   //Mortar part.
   std::vector<types::global_dof_index> dofs_per_component_mortar(
        dim + 1 + 1);
   DoFTools::count_dofs_per_component(dof_handler_mortar,
        dofs_per_component_mortar);
   unsigned int n_z_mortar = dofs_per_component_mortar[0]; //For RT mortar
      space
   unsigned int n_p_mortar = dofs_per_component_mortar[dim + 1];
   solution_bar_mortar.reinit(2);
   solution_bar_mortar.block(0).reinit(n_z_mortar);
   solution_bar_mortar.block(1).reinit(n_p_mortar);
   solution_bar_mortar.collect_sizes();
   solution_bar_mortar = 0;
   solution_star_mortar.reinit(2);
   solution_star_mortar.block(0).reinit(n_z_mortar);
   solution_star_mortar.block(1).reinit(n_p_mortar);
   solution_star_mortar.collect_sizes();
   solution_star_mortar = 0;

   //Space-time part.
   std::vector<types::global_dof_index> dofs_per_component_st(
        dim + 1 + 1);
   DoFTools::count_dofs_per_component(dof_handler_st,
        dofs_per_component_st);

   n_flux_st = dofs_per_component_st[0];
   n_pressure_st = dofs_per_component_st[dim + 1];

   solution_bar_st.reinit(2);
   solution_bar_st.block(0).reinit(n_flux_st);
   solution_bar_st.block(1).reinit(n_pressure_st);
   solution_bar_st.collect_sizes();
   solution_bar_st = 0;
   solution_star_st.reinit(2);
   solution_star_st.block(0).reinit(n_flux_st);
   solution_star_st.block(1).reinit(n_pressure_st);
   solution_star_st.collect_sizes();
   solution_star_st = 0;
   solution_st.reinit(solution_bar_st);
   solution_st.collect_sizes();
   solution_st = 0;
   solution_bar_collection.resize(prm.num_time_steps, solution_bar);
}

solution.reinit(2);
solution.block(0).reinit(n_z);
```

```
        solution.block(1).reinit(n_p);
        solution.collect_sizes();
        solution = 0;
        old_solution.reinit(solution);
        initialc_solution.reinit(solution);
        old_solution_for_jump.reinit(solution);
        pressure_projection.reinit(solution);
        old_pressure_projection.reinit(solution);
        initialc_solution = 0;
        old_solution_for_jump = 0;
        pressure_projection = 0;
        old_pressure_projection = 0;
}
//Assembing the main system
template<int dim>
void DarcyVTProblem<dim>::assemble_system() {
   TimerOutput::Scope t(computing_timer, "Assemble system");
   system_matrix = 0;
   system_rhs_bar_bc = 0;
   QGauss<dim> quadrature_formula(degree + 2);
   FEValues<dim> fe_values(fe, quadrature_formula,
         update_values | update_gradients | update_quadrature_points
         | update_JxW_values);
   const unsigned int dofs_per_cell = fe.dofs_per_cell;
   const unsigned int n_q_points = quadrature_formula.size();
   FullMatrix<double> local_matrix(dofs_per_cell, dofs_per_cell);
   std::vector<types::global_dof_index> local_dof_indices(dofs_per_cell);
   const KInverse<dim> k_inverse;
   std::vector<Tensor<2, dim>> k_inverse_values(n_q_points);

   // Velocity and Pressure DoFs
   const FEValuesExtractors::Vector velocity(0);
   const FEValuesExtractors::Scalar pressure(dim);
   typename DoFHandler<dim>::active_cell_iterator cell =
   dof_handler.begin_active(), endc = dof_handler.end();
   for (; cell != endc; ++cell) {
      fe_values.reinit(cell);
      local_matrix = 0;
      k_inverse.value_list(fe_values.get_quadrature_points(),
            k_inverse_values);
      // Velocity and pressure
      std::vector<Tensor<1, dim>> phi_u(dofs_per_cell);
      std::vector<double> div_phi_u(dofs_per_cell);
      std::vector<double> phi_p(dofs_per_cell);

      for (unsigned int q = 0; q < n_q_points; ++q) {
         for (unsigned int k = 0; k < dofs_per_cell; ++k) {
            // Evaluate test functions
            phi_u[k] = fe_values[velocity].value(k, q);
```

```
            phi_p[k] = fe_values[pressure].value(k, q);
            div_phi_u[k] = fe_values[velocity].divergence(k, q);
         }
         for (unsigned int i = 0; i < dofs_per_cell; ++i) {
            for (unsigned int j = 0; j < dofs_per_cell; ++j) {
               local_matrix(i, j) += (phi_u[i] * k_inverse_values[q]
                     * phi_u[j] - phi_p[j] * div_phi_u[i]
                     + prm.time_step * div_phi_u[j] * phi_p[i]
                     + prm.c_0 * phi_p[i] * phi_p[j]) * fe_values.JxW(q);
            }
         }
      }
      cell->get_dof_indices(local_dof_indices);
      Vector<double> local_rhs(dofs_per_cell);
      local_rhs = 0;
      constraint_bc.distribute_local_to_global(local_matrix, local_rhs,
            local_dof_indices, system_matrix, system_rhs_bar_bc);
   }
   pcout << " ...factorized..." << "\n";
   A_direct.initialize(system_matrix);
}
template<int dim>
void DarcyVTProblem<dim>::get_interface_dofs() {
   TimerOutput::Scope t(computing_timer, "Get interface DoFs");
   {
      interface_dofs.resize(GeometryInfo<dim>::faces_per_cell,
            std::vector<types::global_dof_index>());
      std::vector<types::global_dof_index> local_face_dof_indices;
      if (mortar_flag == 0) {
         typename DoFHandler<dim>::active_cell_iterator cell, endc;
         cell = dof_handler.begin_active(), endc = dof_handler.end();
         local_face_dof_indices.resize(fe.dofs_per_face);
         for (; cell != endc; ++cell) {
            for (unsigned int face_n = 0;
                  face_n < GeometryInfo<dim>::faces_per_cell; ++face_n)
            if (cell->at_boundary(face_n)
                  && cell->face(face_n)->boundary_id() < 100) {
               cell->face(face_n)->get_dof_indices(
                     local_face_dof_indices, 0);
               for (auto el : local_face_dof_indices)
               interface_dofs[cell->face(face_n)->boundary_id() - 1].push_back(
                     el);
            }
         }
      } else {
         typename DoFHandler<dim + 1>::active_cell_iterator cell, endc;
         cell = dof_handler_mortar.begin_active(), endc =
         dof_handler_mortar.end();
         local_face_dof_indices.resize(fe_mortar.dofs_per_face);
```

```
        for (; cell != endc; ++cell) {
          for (unsigned int face_n = 0;
               face_n < GeometryInfo<dim>::faces_per_cell; ++face_n)
          if (cell->at_boundary(face_n)
              && cell->face(face_n)->boundary_id() < 100) {
            cell->face(face_n)->get_dof_indices(
                local_face_dof_indices, 0);
            for (auto el : local_face_dof_indices)
            interface_dofs[cell->face(face_n)->boundary_id() - 1].push_back(
                el);
          }
        }
      }
    }
    if (mortar_flag) {
      interface_dofs_subd.resize(GeometryInfo<dim>::faces_per_cell,
          std::vector<types::global_dof_index>());
      face_dofs_subdom.resize(GeometryInfo<dim>::faces_per_cell,
          std::vector<types::global_dof_index>());
      std::vector<types::global_dof_index> local_face_dof_indices;
      typename DoFHandler<dim>::active_cell_iterator cell, endc;
      cell = dof_handler.begin_active(), endc = dof_handler.end();
      local_face_dof_indices.resize(fe.dofs_per_face);
      for (; cell != endc; ++cell) {
        for (unsigned int face_n = 0;
             face_n < GeometryInfo<dim>::faces_per_cell; ++face_n) {
          //start of getting face dofs.
          cell->face(face_n)->get_dof_indices(local_face_dof_indices, 0);
          for (auto el : local_face_dof_indices) {
            face_dofs_subdom[face_n].push_back(el);
          }
          if (cell->at_boundary(face_n)
              && cell->face(face_n)->boundary_id() < 100) {

            for (auto el : local_face_dof_indices)
            interface_dofs_subd[cell->face(face_n)->boundary_id()
            - 1].push_back(el);
          }
        }
      }
    }
}
//collecting interface DoFs
template<int dim>
void DarcyVTProblem<dim>::get_interface_dofs_st() {
    TimerOutput::Scope t(computing_timer, "Get interface DoFs S-T");
    unsigned int n_faces = GeometryInfo<dim>::faces_per_cell;
    interface_dofs_st.resize(GeometryInfo<dim>::faces_per_cell,
          std::vector<types::global_dof_index>());
```

```
   face_dofs_st.resize(GeometryInfo<dim>::faces_per_cell,
         std::vector<types::global_dof_index>());
   std::vector<types::global_dof_index> local_face_dof_indices;
   typename DoFHandler<dim + 1>::active_cell_iterator cell, endc;
   cell = dof_handler_st.begin_active(), endc = dof_handler_st.end();
   local_face_dof_indices.resize(fe_st.dofs_per_face);

   for (; cell != endc; ++cell) {
      for (unsigned int face_n = 0; face_n < n_faces; ++face_n) {
         cell->face(face_n)->get_dof_indices(local_face_dof_indices, 0);
         for (auto el : local_face_dof_indices) {
            face_dofs_st[face_n].push_back(el);
         }
         if (cell->at_boundary(face_n)
               && cell->face(face_n)->boundary_id() < 100) {
            for (auto el : local_face_dof_indices) {
               interface_dofs_st[cell->face(face_n)->boundary_id() - 1].push_back(
                     el);
            }
         }
      }
   }
}
//Assembling RHS for bar problem
template<int dim>
void DarcyVTProblem<dim>::assemble_rhs_bar() {
   system_rhs_bar = 0;

   QGauss<dim> quadrature_formula(degree + 2);
   QGauss<dim - 1> face_quadrature_formula(qdegree);
   FEValues<dim> fe_values(fe, quadrature_formula,
         update_values | update_gradients | update_quadrature_points
         | update_JxW_values);
   FEFaceValues<dim> fe_face_values(fe, face_quadrature_formula,
         update_values | update_normal_vectors | update_quadrature_points
         | update_JxW_values);
   const unsigned int dofs_per_cell = fe.dofs_per_cell;
   const unsigned int n_q_points = fe_values.get_quadrature().size();
   const unsigned int n_face_q_points = fe_face_values.get_quadrature().size();

   Vector<double> local_rhs(dofs_per_cell);
   std::vector<types::global_dof_index> local_dof_indices(dofs_per_cell);

   //Pressure value for Dirichlet (natural) bc in case of manufactured solution
   PressureBoundaryValues<dim> pressure_boundary_values(prm.coe_a);
   pressure_boundary_values.set_time(prm.time);

   //Dirichlet bc picked up from parameter files. For real applicatins.
   std::vector<Functions::ConstantFunction<dim>> dirichlet_boundary_values_vect;
```

```
//adding dirichlet bc corresponding to each side
//left boundary
Functions::ConstantFunction<dim> dirichlet_boundary_values_left(
     bc_const_functs[0]);
dirichlet_boundary_values_vect.push_back(dirichlet_boundary_values_left);
//bottom boundary
Functions::ConstantFunction<dim> dirichlet_boundary_values_bottom(
     bc_const_functs[1]);
dirichlet_boundary_values_vect.push_back(dirichlet_boundary_values_bottom);
//right boundary
Functions::ConstantFunction<dim> dirichlet_boundary_values_right(
     bc_const_functs[2]);
dirichlet_boundary_values_vect.push_back(dirichlet_boundary_values_right);
//top boundary
Functions::ConstantFunction<dim> dirichlet_boundary_values_top(
     bc_const_functs[1]);
dirichlet_boundary_values_vect.push_back(dirichlet_boundary_values_top);
std::vector<double> boundary_values_flow(n_face_q_points);

RightHandSidePressure<dim> right_hand_side_pressure(prm.c_0, prm.alpha,
     prm.coe_a);
right_hand_side_pressure.set_time(prm.time);
std::vector<double> rhs_values_flow(n_q_points);

typename DoFHandler<dim>::active_cell_iterator cell =
dof_handler.begin_active(), endc = dof_handler.end();
for (; cell != endc; ++cell) {
   local_rhs = 0;
   fe_values.reinit(cell);
   right_hand_side_pressure.value_list(fe_values.get_quadrature_points(),
        rhs_values_flow);

   // Velocity and Pressure DoFs
   const FEValuesExtractors::Vector velocity(0);
   const FEValuesExtractors::Scalar pressure(dim);
   std::vector<double> phi_p(dofs_per_cell);
   std::vector<double> old_pressure_values(n_q_points);
   if (std::fabs(prm.time - prm.time_step) < 1.0e-10)
   fe_values[pressure].get_function_values(initialc_solution,
        old_pressure_values);
   else
   fe_values[pressure].get_function_values(old_solution,
        old_pressure_values);

   for (unsigned int q = 0; q < n_q_points; ++q) {

      for (unsigned int k = 0; k < dofs_per_cell; ++k) {
         // Evaluate test functions
         phi_p[k] = fe_values[pressure].value(k, q);
```

```
        }
        for (unsigned int i = 0; i < dofs_per_cell; ++i) {
          local_rhs(i) += (prm.time_step * phi_p[i] * rhs_values_flow[q]
                + prm.c_0 * old_pressure_values[q] * phi_p[i])
          * fe_values.JxW(q);
        }
      }

      Tensor<2, dim> sigma;
      Tensor<1, dim> sigma_n;
      for (unsigned int face_no = 0;
           face_no < GeometryInfo<dim>::faces_per_cell; ++face_no) {
        if (cell->at_boundary(face_no)) {
          bool at_dir_boundary;
          at_dir_boundary = is_inside<int>(dir_bc_ids,
                cell->face(face_no)->boundary_id());
          if (at_dir_boundary)/
          {
            fe_face_values.reinit(cell, face_no);
            if (is_manufact_solution)
            pressure_boundary_values.value_list(
                  fe_face_values.get_quadrature_points(),
                  boundary_values_flow);
            else if (!is_manufact_solution)
            dirichlet_boundary_values_vect[cell->face(face_no)->boundary_id()
            - 101].value_list(
                  fe_face_values.get_quadrature_points(),
                  boundary_values_flow);
            for (unsigned int q = 0; q < n_face_q_points; ++q)
            for (unsigned int i = 0; i < dofs_per_cell; ++i) {
              local_rhs(i) += -(fe_face_values[velocity].value(i,
                          q) * fe_face_values.normal_vector(q)
                    * boundary_values_flow[q]
                    * fe_face_values.JxW(q));
            }
          }
        }
      }
      cell->get_dof_indices(local_dof_indices);
      FullMatrix<double> local_matrix(dofs_per_cell);
      local_matrix = 0;
      constraint_bc.distribute_local_to_global(local_matrix, local_rhs,
            local_dof_indices, system_matrix, system_rhs_bar);
    }
}
//Assembling RHS for star problem
template<int dim>
void DarcyVTProblem<dim>::assemble_rhs_star() {
```

```
system_rhs_star = 0;
QGauss<dim> quadrature_formula(degree + 2);
QGauss<dim - 1> face_quadrature_formula(qdegree);
FEValues<dim> fe_values(fe, quadrature_formula,
     update_values | update_quadrature_points | update_JxW_values);
FEFaceValues<dim> fe_face_values(fe, face_quadrature_formula,
     update_values | update_normal_vectors | update_quadrature_points
     | update_JxW_values);

const unsigned int dofs_per_cell = fe.dofs_per_cell;
const unsigned int n_q_points = fe_values.get_quadrature().size();
const unsigned int n_face_q_points = fe_face_values.get_quadrature().size();

Vector<double> local_rhs(dofs_per_cell);
std::vector<types::global_dof_index> local_dof_indices(dofs_per_cell);
const FEValuesExtractors::Vector velocity(0);
const FEValuesExtractors::Scalar pressure(dim);
std::vector<Tensor<1, dim>> interface_values_flux(n_face_q_points);
typename DoFHandler<dim>::active_cell_iterator cell =
dof_handler.begin_active(), endc = dof_handler.end();
for (; cell != endc; ++cell) {
   local_rhs = 0;
   fe_values.reinit(cell);
   std::vector<double> phi_p(dofs_per_cell);
   std::vector<double> old_pressure_values(n_q_points);
   if (std::fabs(prm.time - prm.time_step) > 1.0e-10) {
      fe_values[pressure].get_function_values(old_solution,
           old_pressure_values);
      for (unsigned int q = 0; q < n_q_points; ++q) {
         for (unsigned int k = 0; k < dofs_per_cell; ++k) {
            // Evaluate test functions
            phi_p[k] = fe_values[pressure].value(k, q);
         }

         for (unsigned int i = 0; i < dofs_per_cell; ++i)
         local_rhs(i) +=
         (prm.c_0 * old_pressure_values[q] * phi_p[i])
         * fe_values.JxW(q);
      }
   }
   for (unsigned int face_n = 0;
        face_n < GeometryInfo<dim>::faces_per_cell; ++face_n)
   if (cell->at_boundary(face_n)
        && cell->face(face_n)->boundary_id() < 100) {
      fe_face_values.reinit(cell, face_n);
      fe_face_values[velocity].get_function_values(
           interface_fe_function_subdom, interface_values_flux);
      for (unsigned int q = 0; q < n_face_q_points; ++q)
      for (unsigned int i = 0; i < dofs_per_cell; ++i) {
```

```
            local_rhs(i) += -(fe_face_values[velocity].value(i, q)
                  * fe_face_values.normal_vector(q)
                  * interface_values_flux[q]
                  * get_normal_direction(
                        cell->face(face_n)->boundary_id() - 1)
                  * fe_face_values.normal_vector(q)
                  * fe_face_values.JxW(q));
          }
      }
      cell->get_dof_indices(local_dof_indices);
      FullMatrix<double> local_matrix(dofs_per_cell);
      local_matrix = 0;
      constraint_bc.distribute_local_to_global(local_matrix, local_rhs,
            local_dof_indices, system_matrix, system_rhs_star);
   }
}
//Solving bar problem
template<int dim>
void DarcyVTProblem<dim>::solve_bar() {
   system_rhs_bar.sadd(1.0, system_rhs_bar_bc);
   A_direct.vmult(solution_bar, system_rhs_bar);
   constraint_bc.distribute(solution_bar);
}
//Solving star problem
template<int dim>
void DarcyVTProblem<dim>::solve_star() {

   A_direct.vmult(solution_star, system_rhs_star);

}
//Method to drive the solver over all time steps
template<int dim>
void DarcyVTProblem<dim>::solve_darcy_vt(unsigned int maxiter) {
   prm.time = 0.0;
   for (unsigned int time_level = 0; time_level < prm.num_time_steps;
         time_level++) {
      prm.time += prm.time_step;
      solve_timestep(0, time_level);
   }
   prm.time = 0.0;
   pcout << "\nStarting GMRES iterations.........\n";
   if (Utilities::MPI::n_mpi_processes(mpi_communicator) != 1)
   local_gmres(maxiter);
}
//Methods to sovle for each tiem step
template<int dim>
void DarcyVTProblem<dim>::solve_timestep(int star_bar_flag,
      unsigned int time_level) {
   switch (star_bar_flag) {
```

```
case 0:
assemble_rhs_bar();
solve_bar();
if (Utilities::MPI::n_mpi_processes(mpi_communicator) == 1) {
   solution = solution_bar;
   if (is_manufact_solution)
   compute_errors(refinement_index, time_level);
   output_results(refinement_index, total_refinements, time_level + 1);
}
old_solution = solution_bar;
system_rhs_bar = 0;
if (mortar_flag) {
   solution_bar_collection[time_level] = solution_bar;
   subdom_to_st_distribute(solution_bar_st, solution_bar, time_level,
        prm.time_step);
   solution_bar = 0;
}
break;

case 1:
st_to_subdom_distribute(interface_fe_function_st,
     interface_fe_function_subdom, time_level, prm.time_step);
assemble_rhs_star();
solve_star();

interface_fe_function_subdom = 0;
old_solution = solution_star;

subdom_to_st_distribute(solution_star_st, solution_star, time_level,
     prm.time_step);

solution_star = 0;

break;

case 2:
st_to_subdom_distribute(interface_fe_function_st,
     interface_fe_function_subdom, time_level, prm.time_step);
assemble_rhs_star();
solve_star();
interface_fe_function_subdom = 0;
old_solution = solution_star;
solution = 0;
solution.sadd(1.0, solution_star);
solution.sadd(1.0, solution_bar_collection[time_level]);
final_solution_transfer(solution_st, solution, time_level,
     prm.time_step);
if (is_manufact_solution)
```

```
      compute_errors(refinement_index, time_level);
      output_results(refinement_index, total_refinements, time_level + 1);
      old_solution_for_jump = solution;
      break;
   }

}
// Methods to distribute DoFs from space-time subdomain mesh to 2d sub-domain space
    mesh
template<int dim>
void DarcyVTProblem<dim>::st_to_subdom_distribute(
      BlockVector<double> &vector_st, BlockVector<double> &vector_subdom,
      unsigned int &time_level, double scale_factor) {
   for (unsigned int side = 0; side < GeometryInfo<dim>::faces_per_cell;
        ++side)
   if (neighbors[side] >= 0) {
      int interface_dofs_side_size = interface_dofs_subd[side].size();
      for (int i = 0; i < interface_dofs_side_size; i++)
      vector_subdom[interface_dofs_subd[side][i]] =
      (1 / scale_factor)
      * vector_st[interface_dofs_st[side][interface_dofs_side_size
      * time_level + i]];
   }
}
//Methods to distribute DoFs from 2-d subdomain space meshes to 3-d subdomain
    space-time meshes
template<int dim>
void DarcyVTProblem<dim>::subdom_to_st_distribute(
      BlockVector<double> &vector_st, BlockVector<double> &vector_subdom,
      unsigned int &time_level, double scale_factor) {
   for (unsigned int side = 0; side < GeometryInfo<dim>::faces_per_cell;
        ++side)
   if (neighbors[side] >= 0) {
      int interface_dofs_side_size = interface_dofs_subd[side].size();
      for (int i = 0; i < interface_dofs_side_size; i++)
      vector_st[interface_dofs_st[side][interface_dofs_side_size
      * time_level + i]] = scale_factor
      * vector_subdom[interface_dofs_subd[side][i]];
   }
}

//Method to transfer solution from 2d to space-time 3d mesh
template<int dim>
void DarcyVTProblem<dim>::final_solution_transfer(
      BlockVector<double> &solution_st, BlockVector<double> &solution_subdom,
      unsigned int &time_level, double scale_factor) {
   Assert(n_pressure_st == prm.num_time_steps*n_pressure,
        ExcDimensionMismatch(n_pressure_st, prm.num_time_steps*n_pressure ));
   for (unsigned int i = 0; i < n_pressure; i++) {
```

```
      solution_st.block(1)[(time_level * n_pressure) + i] =
      solution_subdom.block(1)[i];
   }
   for (unsigned int side = 0; side < GeometryInfo<dim>::faces_per_cell;
        ++side) {
      int face_dofs_side_size = face_dofs_subdom[side].size();
      for (int i = 0; i < face_dofs_side_size; i++)
      solution_st[face_dofs_st[side][face_dofs_side_size * time_level + i]] =
      scale_factor * solution_subdom[face_dofs_subdom[side][i]];
   }
}
//Auxilliary methods used in the GMRES algorithm
template<int dim>
void DarcyVTProblem<dim>::givens_rotation(double v1, double v2, double &cs,
      double &sn) {
   if (fabs(v1) < 1e-15) {
      cs = 0;
      sn = 1;
   } else {
      double t = sqrt(v1 * v1 + v2 * v2);
      cs = fabs(v1) / t;
      sn = cs * v2 / v1;
   }
}
template<int dim>
void DarcyVTProblem<dim>::apply_givens_rotation(std::vector<double> &h,
      std::vector<double> &cs, std::vector<double> &sn,
      unsigned int k_iteration) {
   unsigned int k = k_iteration;
   AssertThrow(h.size() > k + 1, ExcDimensionMismatch(h.size(), k + 2));
   double temp;
   for (unsigned int i = 0; i < k; ++i) {
      temp = cs[i] * h[i] + sn[i] * h[i + 1];
      h[i + 1] = -sn[i] * h[i] + cs[i] * h[i + 1];
      h[i] = temp;
   }
   AssertThrow(h.size() == k + 2, ExcDimensionMismatch(h.size(), k + 2));
   double cs_k = 0, sn_k = 0;
   givens_rotation(h[k], h[k + 1], cs_k, sn_k);
   h[k] = cs_k * h[k] + sn_k * h[k + 1];
   h[k + 1] = 0.0;
   cs[k] = cs_k;
   sn[k] = sn_k;
}

template<int dim>
void DarcyVTProblem<dim>::back_solve(std::vector<std::vector<double>> H,
      std::vector<double> beta, std::vector<double> &y,
      unsigned int k_iteration) {
```

```
   int k = k_iteration;
   AssertThrow(y.size() == k_iteration + 1,
         ExcDimensionMismatch(y.size(), k_iteration + 1));
   for (unsigned int i = 0; i < k_iteration; i++)
   y[i] = 0;
   for (int i = k - 1; i >= 0; i--) {
      y[i] = beta[i] / H[i][i];
      for (int j = i + 1; j <= k - 1; j++) {
         y[i] -= H[j][i] * y[j] / H[i][i];
      }
   }
}
//GMRES to solve interface problem, working across different processors
template<int dim>
void DarcyVTProblem<dim>::local_gmres(const unsigned int maxiter) {

   const unsigned int this_mpi = Utilities::MPI::this_mpi_process(
         mpi_communicator);
   const unsigned int n_faces_per_cell = GeometryInfo<dim>::faces_per_cell;
   std::vector<std::vector<double>> interface_data_receive(n_faces_per_cell);
   std::vector<std::vector<double>> interface_data_send(n_faces_per_cell);
   std::vector<std::vector<double>> interface_data(n_faces_per_cell);
   std::vector<std::vector<double>> lambda(n_faces_per_cell);
   for (unsigned int side = 0; side < n_faces_per_cell; ++side)
   if (neighbors[side] >= 0) {
      interface_data_receive[side].resize(interface_dofs[side].size(), 0);
      interface_data_send[side].resize(interface_dofs[side].size(), 0);
      interface_data[side].resize(interface_dofs[side].size(), 0);
   }
   Quadrature<dim> quad;
   quad = QGauss<dim>(qdegree);
   Quadrature<dim> quad_project;
   quad_project = QGauss<dim>(qdegree);
   AffineConstraints<double> constraints;
   constraints.clear();
   constraints.close();
   unsigned int temp_array_size = maxiter / 4;
   //GMRES structures and parameters
   std::vector<double> sn(temp_array_size);
   std::vector<double> cs(temp_array_size);
   std::vector<double> Beta(temp_array_size);
   std::vector<std::vector<double>> H(temp_array_size, Beta);
   std::vector<double> e_all_iter(temp_array_size + 1);
   double combined_error_iter = 0;
   std::vector<std::vector<double>> r(n_faces_per_cell);
   std::vector<double> r_norm_side(n_faces_per_cell, 0);
   std::vector<std::vector<std::vector<double>>> Q_side(n_faces_per_cell);
   std::vector<std::vector<double>> Ap(n_faces_per_cell);
   std::vector<std::vector<double>> q(n_faces_per_cell);
```

```
interface_fe_function_st.reinit(solution_bar_st);
interface_fe_function_subdom.reinit(solution_bar);
if (mortar_flag == 1) {
   interface_fe_function_mortar.reinit(solution_bar_mortar);
   interface_fe_function_mortar = 0;
   project_mortar<dim>(P_fine2coarse, dof_handler_st, solution_bar_st,
         quad_project, constraints, neighbors, dof_handler_mortar,
         solution_bar_mortar);
}
for (unsigned side = 0; side < n_faces_per_cell; ++side)
if (neighbors[side] >= 0) {
   Ap[side].resize(interface_dofs[side].size(), 0);
   lambda[side].resize(interface_dofs[side].size(), 0);
   q[side].resize(interface_dofs[side].size());
   r[side].resize(interface_dofs[side].size(), 0);
   std::vector<double> r_receive_buffer(r[side].size());
   Q_side[side].resize(temp_array_size + 1, q[side]);
   if (mortar_flag)
   for (unsigned int i = 0; i < interface_dofs[side].size(); ++i) {
      r[side][i] = get_normal_direction(side)
      * solution_bar_mortar[interface_dofs[side][i]];
   }
   else
   for (unsigned int i = 0; i < interface_dofs[side].size(); ++i)
   r[side][i] = get_normal_direction(side)
   * solution_bar[interface_dofs[side][i]];
   MPI_Sendrecv(&r[side][0], r[side].size(), MPI_DOUBLE,
         neighbors[side], this_mpi,
         &r_receive_buffer[0], r_receive_buffer.size(), MPI_DOUBLE,
         neighbors[side], neighbors[side], mpi_communicator,
         &mpi_status);
   for (unsigned int i = 0; i < interface_dofs[side].size(); ++i) {
      r[side][i] += r_receive_buffer[i];
   }
   r_norm_side[side] = vect_norm(r[side]);
}
double r_norm = 0;
for (unsigned int side = 0; side < n_faces_per_cell; ++side)
if (neighbors[side] >= 0)
r_norm += r_norm_side[side] * r_norm_side[side];
double r_norm_buffer = 0;
MPI_Allreduce(&r_norm, &r_norm_buffer, 1, MPI_DOUBLE, MPI_SUM,
      mpi_communicator);
r_norm = sqrt(r_norm_buffer);
for (unsigned int side = 0; side < n_faces_per_cell; ++side)
if (neighbors[side] >= 0) {
   for (unsigned int i = 0; i < interface_dofs[side].size(); ++i)
   q[side][i] = r[side][i] / r_norm;
   Q_side[side][0] = q[side];
```

```
}
e_all_iter[0] = 1;
pcout << "\n\n r_norm is " << r_norm << " target is " << r_norm * tolerance
<< "\n\n";
Beta[0] = r_norm;
unsigned int k_counter = 0;
while (k_counter < maxiter) {
   if (temp_array_size < k_counter + 2) {
      temp_array_size *= 2;
      cs.resize(temp_array_size);
      sn.resize(temp_array_size);
      e_all_iter.resize(temp_array_size);
      Beta.resize(temp_array_size);
      H.resize(temp_array_size, Beta);
      for (unsigned int side = 0; side < n_faces_per_cell; ++side)
      if (neighbors[side] >= 0) {
         std::vector<double> tmp_vector(interface_dofs[side].size());
         Q_side[side].resize(temp_array_size + 1, tmp_vector);
      }
   }
   for (unsigned int side = 0; side < n_faces_per_cell; ++side)
   if (neighbors[side] >= 0)
   interface_data[side] = Q_side[side][k_counter];
   if (mortar_flag == 1) {
      interface_fe_function_mortar = 0;
      for (unsigned int side = 0; side < n_faces_per_cell; ++side)
      for (unsigned int i = 0; i < interface_dofs[side].size(); ++i)
      interface_fe_function_mortar[interface_dofs[side][i]] =
      interface_data[side][i];

      project_mortar(P_coarse2fine, dof_handler_mortar,
           interface_fe_function_mortar, quad_project, constraints,
           neighbors, dof_handler_st, interface_fe_function_st);
      prm.time = 0.0;
      for (unsigned int time_level = 0; time_level < prm.num_time_steps;
           time_level++)
      {
         prm.time += prm.time_step;
         solve_timestep(1, time_level);
      }
      prm.time = 0.0;
   }
   else {
      for (unsigned int side = 0; side < n_faces_per_cell; ++side)
      for (unsigned int i = 0; i < interface_dofs[side].size(); ++i)
      interface_fe_function_subdom[interface_dofs[side][i]] =
      interface_data[side][i];
      assemble_rhs_star();
      solve_star();
```

```
}
cg_iteration++;
if (mortar_flag == 1) {
   project_mortar<2>(P_fine2coarse, dof_handler_st, solution_star_st,
         quad_project, constraints, neighbors, dof_handler_mortar,
         solution_star_mortar);
}
std::vector<double> h(k_counter + 2, 0);
for (unsigned int side = 0; side < n_faces_per_cell; ++side)
if (neighbors[side] >= 0) {
   if (mortar_flag)
   for (unsigned int i = 0; i < interface_dofs[side].size();
         ++i)
   interface_data_send[side][i] = get_normal_direction(
         side)
   * solution_star_mortar[interface_dofs[side][i]];
   else
   for (unsigned int i = 0; i < interface_dofs[side].size();
         ++i)
   interface_data_send[side][i] = get_normal_direction(
         side) * solution_star[interface_dofs[side][i]];
   MPI_Sendrecv(&interface_data_send[side][0],
         interface_dofs[side].size(), MPI_DOUBLE,
         neighbors[side], this_mpi,
         &interface_data_receive[side][0],
         interface_dofs[side].size(), MPI_DOUBLE,
         neighbors[side], neighbors[side], mpi_communicator,
         &mpi_status);
   // Compute Ap and with it compute alpha
   for (unsigned int i = 0; i < interface_dofs[side].size(); ++i) {
      Ap[side][i] = -(interface_data_send[side][i]
            + interface_data_receive[side][i]);
   }
   q[side].resize(Ap[side].size(), 0);
   AssertThrow(Ap[side].size() == Q_side[side][k_counter].size(),
         ExcDimensionMismatch(Ap[side].size(),
               Q_side[side][k_counter].size()));
   q[side] = Ap[side];
   for (unsigned int i = 0; i <= k_counter; ++i) {
      for (unsigned int j = 0; j < q[side].size(); ++j) {
         h[i] += q[side][j] * Q_side[side][i][j];
      }
   }
}
std::vector<double> h_buffer(k_counter + 2, 0);
MPI_Allreduce(&h[0], &h_buffer[0], k_counter + 2, MPI_DOUBLE, MPI_SUM,
      mpi_communicator);
h = h_buffer;
for (unsigned int side = 0; side < n_faces_per_cell; ++side)
```

```
   if (neighbors[side] >= 0)
   for (unsigned int i = 0; i <= k_counter; ++i)
   for (unsigned int j = 0; j < q[side].size(); ++j) {
      q[side][j] -= h[i] * Q_side[side][i][j];
   }
   double h_dummy = 0;
   for (unsigned int side = 0; side < n_faces_per_cell; ++side)
   if (neighbors[side] >= 0)
   h_dummy += vect_norm(q[side]) * vect_norm(q[side]);
   double h_k_buffer = 0;
   MPI_Allreduce(&h_dummy, &h_k_buffer, 1, MPI_DOUBLE, MPI_SUM,
         mpi_communicator);
   h[k_counter + 1] = sqrt(h_k_buffer);
   for (unsigned int side = 0; side < n_faces_per_cell; ++side)
   if (neighbors[side] >= 0) {
      for (unsigned int i = 0; i < q[side].size(); ++i)
      q[side][i] /= h[k_counter + 1];
      Q_side[side][k_counter + 1] = q[side];
   }
   H[k_counter] = h;
   apply_givens_rotation(H[k_counter], cs, sn, k_counter);
   Beta[k_counter + 1] = -sn[k_counter] * Beta[k_counter];
   Beta[k_counter] *= cs[k_counter];
   combined_error_iter = fabs(Beta[k_counter + 1]) / r_norm;
   e_all_iter[k_counter + 1] = (combined_error_iter);
   << " iterations completed, (relative residual = "
   << combined_error_iter << ")..." << std::flush;
   // Exit criterion
   if (combined_error_iter < tolerance) {
       << "\n GMRES converges in " << cg_iteration
      << " iterations!\n and residual is "
      << e_all_iter[k_counter + 1] * r_norm << "\n";
      break;
   } else if (k_counter > maxiter - 2)
   pcout << "\n GMRES doesn't converge after " << k_counter
   << " iterations!\n";
   for (unsigned int side = 0; side < n_faces_per_cell; ++side) {
      for (unsigned int i = 0; i < interface_data_send[side].size();
            i++) {
         interface_data_receive[side][i] = 0;
         interface_data_send[side][i] = 0;
      }
   }
   Ap.resize(n_faces_per_cell);
   k_counter++;
}
std::vector<double> y(k_counter + 1, 0);
back_solve(H, Beta, y, k_counter);
for (unsigned int side = 0; side < n_faces_per_cell; ++side)
```

```
   if (neighbors[side] >= 0)
   for (unsigned int i = 0; i < interface_data[side].size(); ++i)
   for (unsigned int j = 0; j <= k_counter; ++j)
   lambda[side][i] += Q_side[side][j][i] * y[j];
   if (mortar_flag) {
      interface_data = lambda;
      for (unsigned int side = 0; side < n_faces_per_cell; ++side)
      for (unsigned int i = 0; i < interface_dofs[side].size(); ++i)
      interface_fe_function_mortar[interface_dofs[side][i]] =
      interface_data[side][i];
      project_mortar(P_coarse2fine, dof_handler_mortar,
            interface_fe_function_mortar, quad_project, constraints,
            neighbors, dof_handler_st, interface_fe_function_st);
   }
   else {
      interface_data = lambda;
      for (unsigned int side = 0; side < n_faces_per_cell; ++side)
      for (unsigned int i = 0; i < interface_dofs[side].size(); ++i)
      interface_fe_function_subdom[interface_dofs[side][i]] =
      interface_data[side][i];
   }
   //Finally solving star problems.
   max_cg_iteration = cg_iteration;
   for (unsigned int time_level = 0; time_level < prm.num_time_steps;
         time_level++) {
      prm.time += prm.time_step;
      solve_timestep(2, time_level);
   }
   prm.time = 0.0;
}
//Method to output the soluton for visualization and other purposes.
//This method is capable of producing 3-D space-time plots.
template<int dim>
void DarcyVTProblem<dim>::output_results(const unsigned int cycle,
      const unsigned int refine, const unsigned int time_level) {
   unsigned int n_processes = Utilities::MPI::n_mpi_processes(
         mpi_communicator);
   unsigned int this_mpi = Utilities::MPI::this_mpi_process(mpi_communicator);
   if (cycle == total_refinements - 1) {
      std::vector<std::string> solution_names;
      switch (dim) {
         case 2:
         solution_names.push_back("u1");
         solution_names.push_back("u2");
         solution_names.push_back("p");
         break;
         case 3:
         solution_names.push_back("u1");
         solution_names.push_back("u2");
```

```
      solution_names.push_back("u3");
      solution_names.push_back("p");
      break;
      default:
      AssertThrow(false, ExcNotImplemented());
   }
   if (need_each_time_step_plot) {
      std::vector<DataComponentInterpretation::DataComponentInterpretation>
      data_component_interpretation(
            dim,
            DataComponentInterpretation::component_is_part_of_vector);
      data_component_interpretation.push_back(
            DataComponentInterpretation::component_is_scalar);
      DataOut<dim> data_out;
      data_out.attach_dof_handler(dof_handler);
      data_out.add_data_vector(solution, solution_names,
            DataOut<dim>::type_dof_data, data_component_interpretation);
      data_out.build_patches();
      std::ofstream output(
            "time-step-plots/solution_d" + Utilities::to_string(dim)
            + "_p" + Utilities::to_string(this_mpi, 4) + "-"
            + std::to_string(time_level) + ".vtu");
      data_out.write_vtu(output);
      if (this_mpi == 0) {
         std::vector<std::string> filenames;
         for (unsigned int i = 0; i < n_processes; ++i)
         filenames.push_back(
               "solution_d" + Utilities::to_string(dim) + "_p"
               + Utilities::to_string(i, 4) + "-"
               + std::to_string(time_level) + ".vtu");
         std::ofstream master_output(
               ("time-step-plots/solution_d"
                     + Utilities::to_string(dim) + "-"
                     + std::to_string(time_level) + ".pvtu").c_str());
         data_out.write_pvtu_record(master_output, filenames);
      }
   }
   if (std::fabs(prm.time - prm.final_time) < 1.0e-12) {
      std::vector<std::string> solution_names_st;
      switch (dim) {
         case 2:
         solution_names_st.push_back("u1");
         solution_names_st.push_back("u2");
         solution_names_st.push_back("u3");
         solution_names_st.push_back("p");
         break;
         default:
         AssertThrow(false, ExcNotImplemented())
         ;
```

```
        }
        std::vector<DataComponentInterpretation::DataComponentInterpretation>
        data_component_interpretation_st(
              dim + 1,
              DataComponentInterpretation::component_is_part_of_vector);
        data_component_interpretation_st.push_back(
              DataComponentInterpretation::component_is_scalar);
        DataOut<dim + 1> data_out_2;
        data_out_2.attach_dof_handler(dof_handler_st);
        data_out_2.add_data_vector(solution_st, solution_names_st,
              DataOut<dim + 1>::type_dof_data,
              data_component_interpretation_st);
        data_out_2.build_patches();
        std::ofstream output_st(
              "space-time-plots/st_solution_d"
              + Utilities::to_string(dim + 1) + "_p"
              + Utilities::to_string(this_mpi, 4) + ".vtu");
        data_out_2.write_vtu(output_st);
        if (this_mpi == 0) {
           std::vector<std::string> filenames_st;
           for (unsigned int i = 0;
                i < Utilities::MPI::n_mpi_processes(mpi_communicator);
                ++i)
           filenames_st.push_back(
                "st_solution_d" + Utilities::to_string(dim + 1)
                + "_p" + Utilities::to_string(i, 4)
                + ".vtu");
           std::ofstream master_output_st(
                ("space-time-plots/st_solution_d"
                     + Utilities::to_string(dim + 1) + ".pvtu").c_str());
           data_out_2.write_pvtu_record(master_output_st, filenames_st);
        }
}
\\Resetting mortars in case of multiple levels of refinement
template<int dim>
void DarcyVTProblem<dim>::reset_mortars() {
   triangulation.clear();
   dof_handler.clear();
   convergence_table.clear();
   faces_on_interface.clear();
   faces_on_interface_mortar.clear();
   interface_dofs.clear();
   interface_dofs_st.clear();
   interface_dofs_subd.clear();
   face_dofs_subdom.clear();
   face_dofs_st.clear();
   interface_fe_function_subdom = 0;
   interface_fe_function_st = 0;
```

```
   if (mortar_flag) {
      triangulation_mortar.clear();
      triangulation_st.clear();
   }
   dof_handler_mortar.clear();
   dof_handler_st.clear();
}
//Run method: public member of the class which calls other methods in appropriate
    order
template<int dim>
void DarcyVTProblem<dim>::run(const unsigned int refine,
      const std::vector<std::vector<int>> &reps_st,
      const std::vector<std::vector<int>> &reps_st_mortar, double tol,
      unsigned int maxiter, unsigned int quad_degree) {
   tolerance = tol;
   qdegree = quad_degree;
   total_refinements = refine;
   const unsigned int this_mpi = Utilities::MPI::this_mpi_process(
         mpi_communicator);
   const unsigned int n_processes = Utilities::MPI::n_mpi_processes(
         mpi_communicator);
   pcout << "\n\n Total number of processes is " << n_processes << "\n\n";
   AssertThrow(reps_st[0].size() == dim + 1,
         ExcDimensionMismatch(reps_st[0].size(), dim));
   std::vector<std::vector<unsigned int>> reps_local(reps_st.size()),
   reps_st_local(reps_st.size()), reps_st_local_mortar(reps_st.size());
   for (unsigned int i = 0; i < reps_st_local.size(); i++) {
      reps_local[i].resize(2);
      reps_st_local[i].resize(3);
      reps_st_local_mortar[i].resize(3);
      reps_st_local[i][0] = reps_st[i][0];
      reps_st_local[i][1] = reps_st[i][1];
      reps_st_local[i][2] = reps_st[i][2];
      reps_st_local_mortar[i][0] = reps_st_mortar[i][0];
      reps_st_local_mortar[i][1] = reps_st_mortar[i][1];
      reps_st_local_mortar[i][2] = reps_st_mortar[i][2];
      reps_local[i][0] = reps_st_local[i][0];
      reps_local[i][1] = reps_st_local[i][1];
   }
   if (mortar_flag) {
      pcout << "number of processors is " << n_processes << std::endl;
      AssertThrow(n_processes > 1,
            ExcMessage("Mortar MFEM is impossible with 1 subdomain"));
      AssertThrow(reps_st.size() >= n_processes + 1,
            ExcMessage("Some of the mesh parameters were not provided"));
   }
   for (refinement_index = 0; refinement_index < total_refinements;
         ++refinement_index) {
      cg_iteration = 0;
```

```
interface_dofs.clear();
interface_dofs_st.clear();
interface_dofs_subd.clear();
face_dofs_subdom.clear();
face_dofs_st.clear();
if (refinement_index == 0) {
   prm.num_time_steps = reps_st_local[this_mpi][2];
   prm.time_step = prm.final_time / double(prm.num_time_steps);
   pcout << "Final time= " << prm.final_time << "\n";
   // Partitioning into subdomains (simple bricks)
   find_divisors<dim>(n_processes, n_domains);
   // Dimensions of the domain (unit hypercube)
   std::vector<double> subdomain_dimensions(dim);
   for (unsigned int d = 0; d < dim; ++d)
   subdomain_dimensions[d] = 1.0 / double(n_domains[d]);
   get_subdomain_coordinates(this_mpi, n_domains, subdomain_dimensions,
         p1, p2);
   //corners of the space time sub-domain.
   p1_st = {p1[0],p1[1],0}, p2_st= {p2[0],p2[1],prm.final_time};
   if (mortar_flag) {
      GridGenerator::subdivided_hyper_rectangle(triangulation,
            reps_local[this_mpi], p1, p2);
      GridGenerator::subdivided_hyper_rectangle(triangulation_st,
            reps_st_local[this_mpi], p1_st, p2_st);
      GridGenerator::subdivided_hyper_rectangle(triangulation_mortar,
            reps_st_local_mortar[this_mpi], p1_st, p2_st);
      pcout << "Mortar mesh has "
      << triangulation_mortar.n_active_cells() << " cells"
      << std::endl;
   } else {
      GridGenerator::subdivided_hyper_rectangle(triangulation,
            reps_local[0], p1, p2);
      if (this_mpi == 0 || this_mpi == 3)
      GridTools::distort_random(0.1 * (1 + this_mpi),
            triangulation, true);
   }
}
else {
   if (mortar_flag == 0)
   triangulation.refine_global(1);
   else
   {
      triangulation.clear();
      triangulation_st.clear();
      triangulation_mortar.clear();
      for (unsigned int dum_i = 0; dum_i < reps_st_local.size() - 1;
            dum_i++) {
         reps_st_local[dum_i][0] *= 2;
         reps_st_local[dum_i][1] *= 2;
```

```
            reps_st_local[dum_i][2] *= 2;
            if (mortar_degree == 1) {
               reps_st_local_mortar[dum_i][0] *= 2;
               reps_st_local_mortar[dum_i][1] *= 2;
               reps_st_local_mortar[dum_i][2] *= 2;
            } else if (refinement_index != 0
                  && refinement_index % 2 == 0) {
               reps_st_local_mortar[dum_i][0] *= 2;
               reps_st_local_mortar[dum_i][1] *= 2;
               reps_st_local_mortar[dum_i][2] *= 2;
            }
         }
         //refining mortar mesh
         if (mortar_degree == 1) {
            reps_st_local[reps_st_local.size() - 1][0] *= 2;
            reps_st_local[reps_st_local.size() - 1][1] *= 2;
            reps_st_local[reps_st_local.size() - 1][2] *= 2;
         } else if (refinement_index != 0 && refinement_index % 2 == 0)
         {
            reps_st_local[reps_st_local.size() - 1][0] *= 2;
            reps_st_local[reps_st_local.size() - 1][1] *= 2;
            reps_st_local[reps_st_local.size() - 1][2] *= 2;
         }
         for (unsigned int dum_i = 0; dum_i < reps_local.size();
               dum_i++) {
            reps_local[dum_i][0] = reps_st_local[dum_i][0];
            reps_local[dum_i][1] = reps_st_local[dum_i][1];
         }
         prm.num_time_steps = reps_st_local[this_mpi][2];
         prm.time_step = prm.final_time / double(prm.num_time_steps);
         pcout << "Final time= " << prm.final_time << "\n";
         pcout << "number of time_steps for subdomain is: "
         << prm.num_time_steps << "\n";
         GridGenerator::subdivided_hyper_rectangle(triangulation,
               reps_local[this_mpi], p1, p2);
         GridGenerator::subdivided_hyper_rectangle(triangulation_st,
               reps_st_local[this_mpi], p1_st, p2_st);
         GridGenerator::subdivided_hyper_rectangle(triangulation_mortar,
               reps_st_local_mortar[this_mpi], p1_st, p2_st);
         pcout << "Mortar mesh has "
         << triangulation_mortar.n_active_cells() << " cells"
         << std::endl;
      }
   }
   pcout << "Making grid and DOFs...\n";
   make_grid_and_dofs();
   pcout << "Projecting the initial conditions...\n";
   {
      InitialCondition<dim> ic(prm.coe_a);
```

```
        AffineConstraints<double> constraints;
        constraints.clear();
        constraints.close();
        VectorTools::project(dof_handler, constraints,
              QGauss<dim>(degree + 5), ic, initialc_solution);
        solution = initialc_solution;
        unsigned int time_level = 0;
        output_results(refinement_index, refine, time_level);
      }
      pcout << "Assembling system..." << "\n";
      assemble_system();
      if (Utilities::MPI::n_mpi_processes(mpi_communicator) != 1) {
        get_interface_dofs();
        get_interface_dofs_st();
      }
      solve_darcy_vt(maxiter);
      max_cg_iteration = 0;
      set_current_errors_to_zero();
      prm.time = 0.0;
      computing_timer.print_summary();
      computing_timer.reset();
    }
    reset_mortars();
}
```

Bibliography

[1] Elyes Ahmed, Jan Martin Nordbotten, and Florin Adrian Radu. Adaptive asynchronous time-stepping, stopping criteria, and a posteriori error estimates for fixed-stress iterative schemes for coupled poromechanics problems. *J. Comput. Appl. Math.*, 364:112312, 25, 2020.

[2] Elyes Ahmed, Florin Adrian Radu, and Jan Martin Nordbotten. Adaptive poromechanics computations based on a posteriori error estimates for fully mixed formulations of Biot's consolidation model. *Comput. Methods Appl. Mech. Engrg.*, 347:264–294, 2019.

[3] T. Almani, K. Kumar, A. Dogru, G. Singh, and M. F. Wheeler. Convergence analysis of multirate fixed-stress split iterative schemes for coupling flow with geomechanics. *Comput. Methods Appl. Mech. Engrg.*, 311:180–207, 2016.

[4] G. Alzetta, D. Arndt, W. Bangerth, V. Boddu, B. Brands, D. Davydov, R. Gassmoeller, T. Heister, L. Heltai, K. Kormann, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, and D. Wells. The `deal.II` library, version 9.0. *Journal of Numerical Mathematics*, 26(4):173–183, 2018.

[5] M. Amara and J. M. Thomas. Equilibrium finite elements for the linear elastic problem. *Numer. Math.*, 33(4):367–383, 1979.

[6] I. Ambartsumyan, E. Khattatov, and I. Yotov. A coupled multipoint stress - multipoint flux mixed finite element method for the Biot system of poroelasticity. *Comput. Methods Appl. Mech. Engrg.*, 372:113407, 2020.

[7] Ilona Ambartsumyan, Eldar Khattatov, Jan M. Nordbotten, and Ivan Yotov. A multipoint stress mixed finite element method for elasticity on quadrilateral grids. *Numer. Methods Partial Differential Equations*, https://doi.org/10.1002/num.22624, 2020.

[8] Todd Arbogast, Lawrence C. Cowsar, Mary F. Wheeler, and Ivan Yotov. Mixed finite element methods on nonmatching multiblock grids. *SIAM J. Numer. Anal.*, 37(4):1295–1315, 2000.

[9] Todd Arbogast, Gergina Pencheva, Mary Wheeler, and Ivan Yotov. A multiscale mortar mixed finite element method. *Multiscale Modeling and Simulation*, 6, 01 2007.

[10] Todd Arbogast, Gergina Pencheva, Mary F. Wheeler, and Ivan Yotov. A multiscale mortar mixed finite element method. *Multiscale Model. Simul.*, 6(1):319–346, 2007.

[11] Douglas N. Arnold, Gerard Awanou, and Weifeng Qiu. Mixed finite elements for elasticity on quadrilateral meshes. *Adv. Comput. Math.*, 41(3):553–572, 2015.

[12] Douglas N. Arnold, Richard S. Falk, and Ragnar Winther. Mixed finite element methods for linear elasticity with weakly imposed symmetry. *Math. Comp.*, 76(260):1699–1723, 2007.

[13] Gerard Awanou. Rectangular mixed elements for elasticity with weakly imposed symmetry condition. *Adv. Comput. Math.*, 38(2):351–367, 2013.

[14] W. Bangerth, R. Hartmann, and G. Kanschat. deal.II – a general purpose object oriented finite element library. *ACM Trans. Math. Softw.*, 33(4):24/1–24/27, 2007.

[15] M. Bause, F.A. Radu, and U. Kocher. Space-time finite element approximation of the Biot poroelasticity system with iterative coupling. *Comput. Methods Appl. Mech. Engrg.*, 320:745–768, 2017.

[16] B. Beckermann, S. A. Goreinov, and E. E. Tyrtyshnikov. Some remarks on the Elman estimate for GMRES. *SIAM J. Matrix Anal. Appl.*, 27(3):772–778, 2005.

[17] Maurice A Biot. General theory of three-dimensional consolidation. *J. Appl. Phys.*, 12(2):155–164, 1941.

[18] Daniele Boffi, Franco Brezzi, Leszek F. Demkowicz, Ricardo G. Durán, Richard S. Falk, and Michel Fortin. *Mixed finite elements, compatibility conditions, and applications*, volume 1939 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin; Fondazione C.I.M.E., Florence, 2008.

[19] Daniele Boffi, Franco Brezzi, and Michel Fortin. Reduced symmetry elements in linear elasticity. *Commun. Pure Appl. Anal.*, 8(1):95–121, 2009.

[20] Manuel Borregales, Kundan Kumar, Florin Adrian Radu, Carmen Rodrigo, and Francisco Jose Gaspar. A partially parallel-in-time fixed-stress splitting method for Biot's consolidation model. *Comput. Math. Appl.*, 77(6):1466–1478, 2019.

[21] Jakub Wiktor Both, Kundan Kumar, Jan Martin Nordbotten, and Florin Adrian Radu. Anderson accelerated fixed-stress splitting schemes for consolidation of unsaturated porous media. *Comput. Math. Appl.*, 77(6):1479–1502, 2019.

[22] Franco Brezzi and Michel Fortin. *Mixed and hybrid finite element methods*, volume 15 of *Springer Series in Computational Mathematics*. Springer-Verlag, New York, 1991.

[23] Martina Bukac, William Layton, Marina Moraiti, Hoang Tran, and Catalin Trenchea. Analysis of partitioned methods for the Biot system. *Numer. Methods Partial Differential Equations*, 31(6):1769–1813, 2015.

[24] Philippe G. Ciarlet. *The finite element method for elliptic problems*, volume 40 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2002.

[25] Bernardo Cockburn, Jayadeep Gopalakrishnan, and Johnny Guzmán. A new elasticity element made for enforcing weak stress symmetry. *Math. Comp.*, 79(271):1331–1349, 2010.

[26] Lawrence C. Cowsar, Jan Mandel, and Mary F. Wheeler. Balancing domain decomposition for mixed finite elements. *Math. Comp.*, 64(211):989–1015, 1995.

[27] Monique Dauge. *Elliptic boundary value problems on corner domains*, volume 1341 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1988.

[28] Stanley C. Eisenstat, Howard C. Elman, and Martin H. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM J. Numer. Anal.*, 20(2):345–357, 1983.

[29] Mohamed Farhloul and Michel Fortin. Dual hybrid methods for the elasticity and the Stokes problems: a unified approach. *Numer. Math.*, 76(4):419–440, 1997.

[30] Horacio Florez. About revisiting domain decomposition methods for poroelasticity. *Mathematics*, 6(10):187, 2018.

[31] Horacio Florez and Mary Wheeler. A mortar method based on NURBS for curved interfaces. *Comput. Methods Appl. Mech. Engrg.*, 310:535–566, 2016.

[32] A. Fritz, S. Hüeber, and B. I. Wohlmuth. A comparison of mortar and Nitsche techniques for linear elasticity. *Calcolo*, 41(3):115–137, 2004.

[33] Benjamin Ganis and Ivan Yotov. Implementation of a mortar mixed finite element method using a multiscale flux basis. *Comput. Methods Appl. Mech. Engrg.*, 198(49-52):3989–3998, 2009.

[34] F. J. Gaspar, F. J. Lisbona, and P. N. Vabishchevich. A finite difference analysis of Biot's consolidation model. *Appl. Numer. Math.*, 44(4):487–506, 2003.

[35] V Girault, G Pencheva, Mary F Wheeler, and T Wildey. Domain decomposition for poroelasticity and elasticity with DG jumps and mortars. *Math. Mod. Meth. Appl. S.*, 21(01):169–213, 2011.

[36] Vivette Girault, Gergina V. Pencheva, Mary F. Wheeler, and Tim M. Wildey. Domain decomposition for linear elasticity with DG jumps and mortars. *Comput. Methods Appl. Mech. Engrg.*, 198(21-26):1751–1765, 2009.

[37] Vivette Girault and Pierre-Arnaud Raviart. *Finite element methods for Navier-Stokes equations*, volume 5 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1986. Theory and algorithms.

[38] Roland Glowinski and Mary F Wheeler. Domain decomposition and mixed finite element methods for elliptic problems. In *First international symposium on domain decomposition methods for partial differential equations*, pages 144–172, 1988.

[39] J. Gopalakrishnan and J. Guzmán. A second elasticity element using the matrix bubble. *IMA J. Numer. Anal.*, 32(1):352–372, 2012.

[40] Pierre Gosselet, Vincent Chiaruttini, Christian Rey, and Frederic Feyel. A monolithic strategy based on an hybrid domain decomposition method for multiphysic problems. Application to poroelasticity. *Revue Europeenne des Elements Finis*, 13:523–534, 2012.

[41] Anne Greenbaum. *Iterative methods for solving linear systems*, volume 17 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[42] Pierre Grisvard. *Elliptic problems in nonsmooth domains*, volume 69 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2011.

[43] Patrice Hauret and Patrick Le Tallec. A discontinuous stabilized mortar method for general 3D elastic problems. *Comput. Methods Appl. Mech. Engrg.*, 196(49-52):4881–4900, 2007.

[44] Roger A. Horn and Charles R. Johnson. *Topics in matrix analysis*. Cambridge University Press, Cambridge, 1994. Corrected reprint of the 1991 original.

[45] Xiaozhe Hu, Carmen Rodrigo, Francisco J. Gaspar, and Ludmil T. Zikatanov. A nonconforming finite element method for the Biot's consolidation model in poroelasticity. *J. Comput. Appl. Math.*, 310:143–154, 2017.

[46] I. C. F. Ipsen. Expressions and bounds for the GMRES residual. *BIT Numerical Mathematics*, 40(3):524–535, 2000.

[47] C. T. Kelley. *Iterative methods for linear and nonlinear equations*, volume 16 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics, Philadelphia, 1995.

[48] Eldar Khattatov and Ivan Yotov. Domain decomposition and multiscale mortar mixed finite element methods for linear elasticity with weak stress symmetry. *ESAIM Math. Model. Numer. Anal.*, 53(6):2081–2108, 2019.

[49] Hyea Hyun Kim. A BDDC algorithm for mortar discretization of elasticity problems. *SIAM J. Numer. Anal.*, 46(4):2090–2111, 2008.

[50] Hyea Hyun Kim. A FETI-DP formulation of three dimensional elasticity problems with mortar discretization. *SIAM J. Numer. Anal.*, 46(5):2346–2370, 2008.

[51] Jihoon Kim, Hamdi Tchelepi, and R Juanes. Stability and convergence of sequential methods for coupled flow and geomechanics: Drained and undrained splits. *Comput. Methods Appl. Mech. Engrg.*, 200:2094–2116, 2011.

[52] Jihoon Kim, Hamdi Tchelepi, and R Juanes. Stability and convergence of sequential methods for coupled flow and geomechanics: Fixed-stress and fixed-strain splits. *Comput. Methods Appl. Mech. Engrg.*, 200:1591–1606, 2011.

[53] J. Kovacik. Correlation between Young's modulus and porosity in porous materials. *J. Mater. Sci. Lett.*, 18(13):1007–1010, 1999.

[54] Jeonghun J. Lee. Robust error analysis of coupled mixed methods for Biot's consolidation model. *J. Sci. Comput.*, 69(2):610–632, 2016.

[55] Jeonghun J. Lee. Towards a unified analysis of mixed methods for elasticity with weakly symmetric stress. *Adv. Comput. Math.*, 42(2):361–376, 2016.

[56] Jeonghun J. Lee. Robust three-field finite element methods for Biot's consolidation model in poroelasticity. *BIT*, 58(2):347–372, 2018.

[57] Jeonghun J. Lee, Kent-Andre Mardal, and Ragnar Winther. Parameter-robust discretization and preconditioning of Biot's consolidation model. *SIAM J. Sci. Comput.*, 39(1):A1–A24, 2017.

[58] T. P. Mathew. *Domain decomposition and iterative refinement methods for mixed finite element discretizations of elliptic problems*. PhD thesis, Courant Institute of Mathematical Sciences, New York University, 1989. Tech. Rep. 463.

[59] Andro Mikelić and Mary F. Wheeler. Convergence of iterative coupling for coupled flow and geomechanics. *Comput. Geosci.*, 17(3):455–461, 2013.

[60] Márcio A. Murad and Abimael F. D. Loula. Improved accuracy in finite element analysis of Biot's consolidation problem. *Comput. Methods Appl. Mech. Engrg.*, 95(3):359–382, 1992.

[61] Jan Martin Nordbotten. Stable cell-centered finite volume discretization for Biot equations. *SIAM J. Numer. Anal.*, 54(2):942–968, 2016.

[62] Ricardo Oyarzúa and Ricardo Ruiz-Baier. Locking-free finite element methods for poroelasticity. *SIAM J. Numer. Anal.*, 54(5):2951–2973, 2016.

[63] Gergina Pencheva and Ivan Yotov. Balancing domain decomposition for mortar mixed finite element methods. *Numer. Linear Algebra Appl.*, 10(1-2):159–180, 2003.

[64] Malgorzata Peszynska, Mary F. Wheeler, and Ivan Yotov. Mortar upscaling for multiphase flow in porous media. *Comput. Geosci.*, 6(1):73–100, 2002.

[65] P. J. Phillips and M. F. Wheeler. A coupling of mixed and discontinuous Galerkin finite-element methods for poroelasticity. *Comput. Geosci.*, 12(4):417–435, 2008.

[66] Phillip Joseph Phillips and Mary F. Wheeler. A coupling of mixed and continuous Galerkin finite element methods for poroelasticity. I. The continuous in time case. *Comput. Geosci.*, 11(2):131–144, 2007.

[67] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential equations*. Clarendon Press, Oxford, 1999.

[68] J. E. Roberts and J.-M. Thomas. Mixed and hybrid methods. In *Handbook of numerical analysis, Vol. II*, Handb. Numer. Anal., II, pages 523–639. North-Holland, Amsterdam, 1991.

[69] C. Rodrigo, F. J. Gaspar, X. Hu, and L. T. Zikatanov. Stability and monotonicity for some discretizations of the Biot's consolidation model. *Comput. Methods Appl. Mech. Engrg.*, 298:183–204, 2016.

[70] C. Rodrigo, X. Hu, P. Ohm, J. H. Adler, F. J. Gaspar, and L. T. Zikatanov. New stabilized discretizations for poroelasticity and the Stokes' equations. *Comput. Methods Appl. Mech. Engrg.*, 341:467–484, 2018.

[71] L. Ridgway Scott and Shangyou Zhang. Finite element interpolation of nonsmooth functions satisfying boundary conditions. *Math. Comput.*, 54(190):483–493, 1990.

[72] R. E. Showalter. Diffusion in poro-elastic media. *J. Math. Anal. Appl.*, 251(1):310 – 340, 2000.

[73] R. E. Showalter. Monotone operators in banach space and nonlinear partial differential equations. 2013.

[74] Gerhard Starke. Field-of-values analysis of preconditioned iterative methods for nonsymmetric elliptic problems. *Numer. Math.*, 78(1):103–117, 1997.

[75] Rolf Stenberg. A family of mixed finite elements for the elasticity problem. *Numer. Math.*, 53(5):513–538, 1988.

[76] Erlend Storvik, Jakub W. Both, Kundan Kumar, Jan M. Nordbotten, and Florin A. Radu. On the optimization of the fixed-stress splitting for Biot's equations. *Int. J. Numer. Methods. Eng.*, 120(2):179–194, 2019.

[77] Andrea Toselli and Olof Widlund. *Domain decomposition methods—algorithms and theory*, volume 34 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2005.

[78] Danail Vassilev, Changqing Wang, and Ivan Yotov. Domain decomposition for coupled Stokes and Darcy flows. *Comput. Methods. Appl. Mech. Eng.*, 268:264–283, 2014.

[79] Mary F. Wheeler, Guangri Xue, and Ivan Yotov. Coupling multipoint flux mixed finite element methods with continuous Galerkin methods for poroelasticity. *Comput. Geosci.*, 18(1):57–75, 2014.

[80] Son-Young Yi. A coupling of nonconforming and mixed finite element methods for Biot's consolidation model. *Numer. Meth. Partial. Differ. Equ.*, 29(5):1749–1777, 2013.

[81] Son-Young Yi. Convergence analysis of a new mixed finite element method for Biot's consolidation model. *Numer. Meth. Partial. Differ. Equ.*, 30(4):1189–1210, 2014.

[82] Son-Young Yi. A study of two modes of locking in poroelasticity. *SIAM J. Numer. Anal.*, 55(4):1915–1936, 2017.

[83] Son-Young Yi and Maranda Bean. Iteratively coupled solution strategies for a four-field mixed finite element method for poroelasticity. *Int. J. Numer. Anal. Meth. Geomech.*, 2016.

[84] E. Abreu, P. Ferraz, A. M. E. Santo, F. Pereira, L. G. C. Santos, and F. S. Sousa. Recursive formulation and parallel implementation of multiscale mixed methods, 2020. arXiv:2009.07965.

[85] E. Ahmed, J. M. Nordbotten, and F. A. Radu. Adaptive asynchronous time-stepping, stopping criteria, and a posteriori error estimates for fixed-stress iterative schemes for coupled poromechanics problems. *Journal of Computational and Applied Mathematics*, 364:112312, 2020.

[86] S. Ali Hassan, C. Japhet, M. Kern, and M. Vohralík. A posteriori stopping criteria for optimized Schwarz domain decomposition algorithms in mixed formulations. *Comput. Methods Appl. Math.*, 18(3):495–519, 2018.

[87] S. Ali Hassan, C. Japhet, and M. Vohralík. A posteriori stopping criteria for space-time domain decomposition for the heat equation in mixed formulations. *Electron. Trans. Numer. Anal.*, 49:151–181, 2018.

[88] T. Almani, K. Kumar, A. Dogru, G. Singh, and M. Wheeler. Convergence analysis of multirate fixed-stress split iterative schemes for coupling flow with geomechanics. *Computer Methods in Applied Mechanics and Engineering*, 311:180–207, 2016.

[89] T. Arbogast, L. C. Cowsar, M. F. Wheeler, and I. Yotov. Mixed finite element methods on nonmatching multiblock grids. *SIAM J. Numer. Anal.*, 37(4):1295–1315, 2000.

[90] T. Arbogast, G. Pencheva, M. F. Wheeler, and I. Yotov. A multiscale mortar mixed finite element method. *Multiscale Model. Simul.*, 6(1):319–346, 2007.

[91] W. Bangerth, R. Hartmann, and G. Kanschat. deal.II—a general-purpose object-oriented finite element library. *ACM Trans. Math. Software*, 33(4):Art. 24, 27, 2007.

[92] W. Bangerth, G. Kanschat, T. Heister, and M. Maier. *deal.II*.

[93] M. Bause, F. Radu, and U. Köcher. Space-time finite element approximation of the Biot poroelasticity system with iterative coupling. *Computer Methods in Applied Mechanics and Engineering*, 320:745–768, 2017.

[94] M. Bause, F. A. Radu, and U. Köcher. Error analysis for discretizations of parabolic problems using continuous finite elements in time and mixed finite elements in space. *Numer. Math.*, 137(4):773–818, 2017.

[95] B. Beckermann, S. A. Goreinov, and E. E. Tyrtyshnikov. Some remarks on the Elman estimate for GMRES. *SIAM J. Matrix Anal. Appl.*, 27(3):772–778, 2005.

[96] M. Bener**r**A¡, A. Nekvinda, and M. K. Yadav. Multi-time-step domain decomposition method with non-matching grids for parabolic problems. *Applied Mathematics and Computation*, 267:571–582, 2015. The Fourth European Seminar on Computing (ESCO 2014).

[97] M. Borregales, K. Kumar, F. A. Radu, C. Rodrigo, and F. J. Gaspar. A partially parallel-in-time fixed-stress splitting method for Biot's consolidation model. *Computers & Mathematics with Applications*, 77(6):1466–1478, 2019. 7th International Conference on Advanced Computational Methods in Engineering (ACOMEN 2017).

[98] F. Brezzi and M. Fortin. *Mixed and hybrid finite element methods*. Springer-Verlag, New York, 1991.

[99] P. G. Ciarlet. *The finite element method for elliptic problems*. North-Holland Publishing Co., Amsterdam-New York-Oxford, 1978. Studies in Mathematics and its Applications, Vol. 4.

[100] M. Crouzeix and V. Thomée. The stability in L_p and W_p^1 of the L_2-projection onto finite element function spaces. *Math. Comp.*, 48(178):521–532, 1987.

[101] S. C. Eisenstat, H. C. Elman, and M. H. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM J. Numer. Anal.*, 20(2):345–357, 1983.

[102] A. Ern, I. Smears, and M. Vohralík. Guaranteed, locally space-time efficient, and polynomial-degree robust a posteriori error estimates for high-order discretizations of parabolic problems. *SIAM J. Numer. Anal.*, 55(6):2811–2834, 2017.

[103] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, and J. B. Schroder. Parallel time integration with multigrid. *SIAM Journal on Scientific Computing*, 36(6):C635–C661, 2014.

[104] V. Faucher and A. Combescure. A time and space mortar method for coupling linear modal subdomains and non-linear subdomains in explicit structural dynamics. *Computer Methods in Applied Mechanics and Engineering*, 192(5):509–533, 2003.

[105] M. J. Gander. 50 years of time parallel time integration. In *Multiple shooting and time domain decomposition methods. MuS-TDD, Heidelberg, Germany, May 6–8, 2013*, pages 69–113. Cham: Springer, 2015.

[106] M. J. Gander, F. Kwok, and B. C. Mandal. Dirichlet-Neumann and Neumann-Neumann waveform relaxation algorithms for parabolic problems. *ETNA, Electron. Trans. Numer. Anal.*, 45:424–456, 2016.

[107] M. J. Gander and M. Neumüller. Analysis of a new space-time parallel multigrid algorithm for parabolic problems. *SIAM J. Sci. Comput.*, 38(4):a2173–a2208, 2016.

[108] M. J. Gander and S. Vandewalle. Analysis of the parareal time-parallel time-integration method. *SIAM J. Sci. Comput.*, 29(2):556–578, 2007.

[109] B. Ganis and I. Yotov. Implementation of a mortar mixed finite element method using a multiscale flux basis. *Computer Methods in Applied Mechanics and Engineering*, 198(49):3989–3998, 2009.

[110] A. Greenbaum. *Iterative methods for solving linear systems*, volume 17 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[111] P. Grisvard. *Elliptic problems in nonsmooth domains*, volume 69 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2011.

[112] T.-T.-P. Hoang, J. Jaffré, C. Japhet, M. Kern, and J. E. Roberts. Space-time domain decomposition methods for diffusion problems in mixed formulations. *SIAM J. Numer. Anal.*, 51(6):3532–3559, 2013.

[113] T.-T.-P. Hoang, C. Japhet, M. Kern, and J. E. Roberts. Space-time domain decomposition for reduced fracture models in mixed formulation. *SIAM J. Numer. Anal.*, 54(1):288–316, 2016.

[114] R. A. Horn and C. R. Johnson. *Topics in matrix analysis*. Cambridge University Press, Cambridge, 1994. Corrected reprint of the 1991 original.

[115] I. C. F. Ipsen. Expressions and bounds for the GMRES residual. *BIT Numerical Mathematics*, 40(3):524–535, 2000.

[116] C. T. Kelley. *Iterative methods for linear and nonlinear equations*, volume 16 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics, Philadelphia, 1995.

[117] D. Krause and R. Krause. Enabling local time stepping in the parallel implicit solution of reaction-diffusion equations via space-time finite elements on shallow tree meshes. *Applied Mathematics and Computation*, 277:164–179, 2016.

[118] J.-L. Lions, Y. Maday, and G. Turinici. Résolution d'EDP par un schéma en temps "pararéel". *C. R. Acad. Sci., Paris, Sér. I, Math.*, 332(7):661–668, 2001.

[119] J.-L. Lions and E. Magenes. *Non-homogeneous boundary value problems and applications. Vol. I*. Springer-Verlag, New York-Heidelberg, 1972.

[120] M. NeumÃŒller and I. Smears. Time-parallel iterative solvers for parabolic evolution equations. *SIAM Journal on Scientific Computing*, 41(1):C28–C51, 2019.

[121] M. A. Puscas, G. Enchéry, and S. Desroziers. Application of the mixed multiscale finite element method to parallel simulations of two-phase flows in porous media. *Oil Gas Sci. Technol. - Rev. IFP Energies nouvelles*, 73:38, 2018.

[122] V. Savcenco, W. Hundsdorfer, and J. Verwer. A multirate time stepping strategy for parabolic PDE. Modelling, Analysis and Simulation report E 0516, Centrum voor Wiskunde en Informatica, 2005.

[123] L. R. Scott and S. Zhang. Finite element interpolation of nonsmooth functions satisfying boundary conditions. *Math. Comput.*, 54(190):483–493, 1990.

[124] R. Speck, D. Ruprecht, M. Emmett, M. Minion, M. Bolten, and R. Krause. A multi-level spectral deferred correction method. *BIT*, 55(3):843–867, 2015.

[125] G. Starke. Field-of-values analysis of preconditioned iterative methods for nonsymmetric elliptic problems. *Numer. Math.*, 78(1):103–117, 1997.

[126] V. Thomée. *Galerkin finite element methods for parabolic problems*, volume 25 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 2006.

[127] H. Yu. A local space-time adaptive scheme in solving two-dimensional parabolic problems based on domain decomposition methods. *SIAM Journal on Scientific Computing*, 23(1):304–322, 2001.