A Weak Galerkin Finite Element Method for Parallel Solutions of Linear Elasticity Problems on Unstructured Meshes

by Liangwei Li

B.S. in Mechanical Engineering, June 2011, Sun Yat-sen University
M.S. in Mechanical Engineering, August 2013, the George Washington University

A dissertation submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

May 1, 2018

Dissertation directed by

Dr. Chunlei Liang
Associate Professor of Engineering and Applied Science
Dr. Junping Wang
National Science Foundation

The School of Engineering and Applied Science of The George Washington University certifies that Liangwei Li has passed the Final Examination for the degree of Doctor of Philosophy as of February 26, 2018. This is the final and approved form of the dissertation.

A Weak Galerkin Finite Element Method for Parallel Solutions of Linear Elasticity Problems on Unstructured Meshes

Liangwei Li

Dissertation Research Committee:

Chunlei Liang, Associate Professor of Engineering and Applied Science, Dissertation Director

Junping Wang, National Science Foundation, Co-advisor & Committee Member

James Lee, Professor of Engineering and Applied Science, Committee Member

Michael Plesniak, Chair and Professor of Engineering and Applied Science, Chair of committee

Murli M. Gupta, Professor of Mathematics, Committee Member

Dedication

I dedicate this work to my family, for their boundless love.

Acknowledgements

I am deeply thankful for my advisor, Professor Chunlei Liang. His broad knowledge inspired me, his restless hardworking encouraged me, his constant advice enlightened me and his generous support helped me through my entire graduate study at the George Washington University. It is my best gratitude to have Dr. Junping Wang, a pioneer of the weak Galerkin finite element method, as my co-advisor. His research contribution not only inspired a lot of researchers, but also created a vibrant community. He pointed a promising direction for my entire research life. I would like to thank all other committee members, Prof. James Lee, Prof. Murli Gupta and Prof. Michael Plesniak for their insightful advice on this piece of work. Many thanks for my lab officemates and best friends: Junfeng, Bin, Jingjing, Xiaoliang, Mao, Zihua, Zhen and Rongguang. Additional thanks to MAE department for the long-term support.

Abstract

A Weak Galerkin Finite Element Method for Parallel Solutions of Linear Elasticity Problems on Unstructured Meshes

In this dissertation, we present a novel parallel computing method for efficiently solving elasticity equations on unstructured meshes. The numerical method of our parallel computing scheme is based on the weak Galerkin (WG) finite element method which is newly developed by Dr. Junping Wang and Dr. Xiu Ye. The WG finite element method refers to a general finite element method on tackling a variety of partial differential equations. The distinct feature of the WG method is that the differential operators are discretized and replaced by the weak operators. Through the weak functions, a connection between the weak operators and unknown variables is connected by solving the elemental matrices for each element. Utilizing the WG method for solving the elasticity equation is converting the weak strain and stress tensors to the weak operators by employing the concept of discrete weak gradients and weak divergences.

A long-term goal of this research is to solve large-scale fluid-structure interaction problems on parallel computers. On the purpose of solving large-scale fluid-structure interaction problems, we choose to employ a partitioned approach and coupled by the Implicit-Explicit coupling method. In this approach, the governing equations of the fluid and structural parts are calculated by two individual solvers. Since an in-house fluid solver for solving the Naiver-Stokes equations has been developed and published, an accurate and efficient solid solver, which is capable of solving elasticity equation on parallel computers, is needed now.

To achieve a highly efficient computational process dealing with the structural dynamics, a non-overlapping domain decomposition scheme is introduced to parallelize the WG solver. We present two different approaches to implement the parallel

computing. Initially, we combine the classic continuous Galerkin (CG) finite element method with the WG finite element method and design a hybrid WG-CG element. The hybrid element inherits the discontinuous feature from the WG method and the computational efficiency from the CG method. After implementing the Schur complement method, the accuracy and scalability have been demonstrated. A more advanced approach for 2 dimensional problem is to implement the duality concept to split the computational space into primal and dual spaces. The connection of adjacent subdomains is implemented through the balancing domain decomposition with constraints (BDDC) method which is originally proposed by Mandel[40]. Locally over each subdomain, matrices are constructed for interior and interface quantities separately. An efficient preconditioner for interface problem is constructed in the parallel fashion through a global primal space and a local dual space. Therefore, such interface related quantities are passed over to their adjacent subdomains through inter-processor communication library, Message Passing Interface (MPI). After the convergence of the interface problem, the remaining local unknown variables could be recovered locally.

This dissertation is structured as follow:

In Chapter 1, we introduce the background and objectives of this dissertation. We provide the preliminaries which are important and necessary for the following discussions. We derive the bilinear form of second order elliptical equation and the elasticity equation.

In Chapter 2, we discuss the weak Galerkin finite element method and the bilinear form of linear elasticity equation.The WG finite element method is based on the variational form of the governing equations. Different types of polygons can be applied as finite element for the WG method. The stiffness matrix derived from the WG method is symmetric and positive definite. Due to the flexibility of the polynomials basis functions, it is simple to construct high order element. The order of accuracy for

the WG method is determined by the highest order of basis function in the element.

In Chapter 3, we design a novel parallel computing method for solving the elasticity equation. We develop a novel hybrid WG-CG element which combines the elements of weak Galerkin (WG) finite element method and classic continuous Galerkin (CG) finite element method. The new hybrid element inherits the discontinuous feature of the WG method. In the hybrid element, we can insert multiple CG elements in one WG element. Each hybrid element is treated as a computational subdomain calculated by individual processor. In our numerical examples, second order of accuracy is demonstrated for interior and boundary unknown variables. A superlinear speedup is obtaind with the scalability up to 60 processors.

In Chapter 4, we develop a novel parallel solver for the linear elasticity problems on unstructured meshes. To enable parallel computation, the computational domain is partitioned into multiple subdomains. The connection of adjacent subdomains is implemented through the balancing domain decomposition with constraints (BDDC) method which was originally proposed by Mandel [40]. Locally, over each subdomain, the matrices are constructed for interior and interface quantities concurrently. Consequently, the interface related matrices are computed by their adjacent subdomains through MPI libraries. MPI communications are employed to construct an efficient global preconditioner and distribute the major computational cost to local processors. The novel WG-BDDC parallel algorithm achieves superlinear speedup up to 600 processors. The highest speedup ratio achieved at 123%. Our numerical results demonstrate that the WG-BDDC method obtained optimal order of accuracy for both 2nd-order and 3rd-order spatial discretization schemes. In addition, the condition numbers of the Lanczos matrix from PCG iteration for all test problems are well bounded.

In Chapter 5, we conclude the dissertation and outlook the future work.

Keywords : weak Galerkin, finite element method, parallel computing, linear

elasticity, message passing interface, continuous Galerkin, domain decomposition, balancing domain decomposition by constraints, polygonal meshes

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Background

1.1.1 Engineering Background

Peripheral artery disease (PAD) is a narrowing of arteries along legs. It is a major cause of amputation in United States [11]. It is prevalent among smokers, diabetics and patients with dyslipidemia. Meanwhile, the long waiting time and expensive cost are two major problems that affect patients. In the modern clinic, doctors are actively looking for a solution which can provides both high fidelity and high resolution images to analyze the PAD. The present CT scan technology can only render a two-dimensional, monochrome, static and low-resolution pictures. In recent decades, many researchers have contributed numerous efforts on improving the diagnosis of stenoses and the quality of images [10, 48, 60, 52, 39]. With the fast development of both hardware and computing algorithms, the computational simulations are aiming to provide 3D, dynamic, high resolution and patient-specific results.

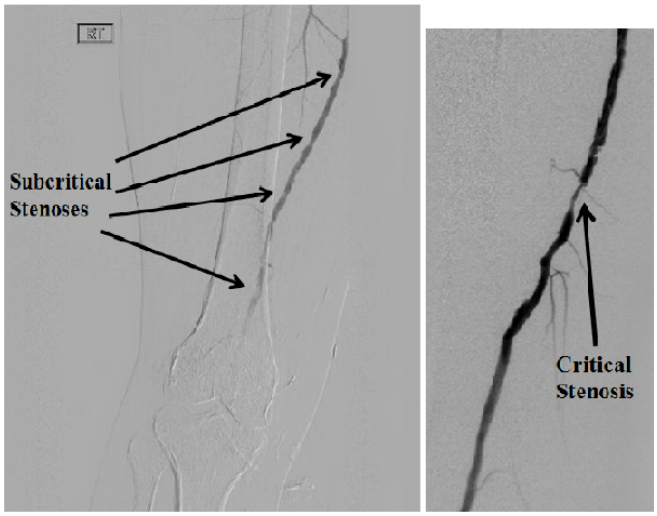


Figure 1.1: Critical stenosis and subcritical stenoses.

Fig. 1.1 is the CT scan image. It shows the difference between single critical

stenosis and multiple subcritical stenosis. Currently, doctors employ stent to treat the critical stenosis. However, the impact of multiple subcritical stenoses is unclear yet. Computational simulation is a great tool to assist doctors to evaluate and make decision. Compared to current CT scans, it is significantly faster, less expensive and more accurate. In addition, it is also very important that the simulation results provide the dynamic growth animations of current stenoses and the developing path after the clinic treatment to doctors and patients.

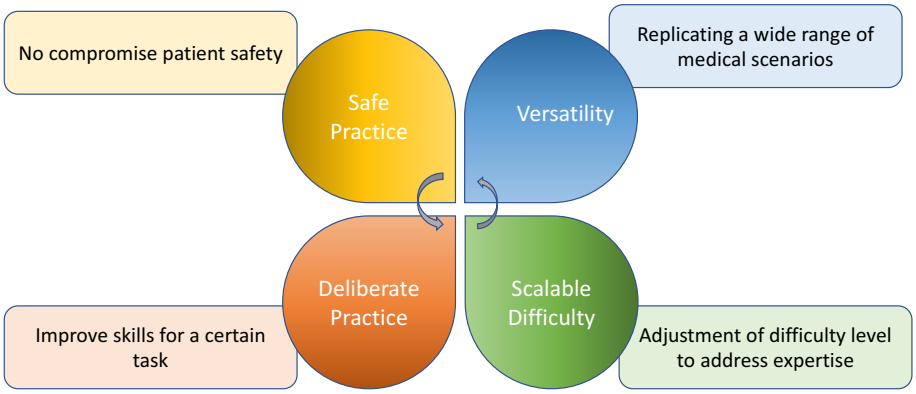


Figure 1.2: Different factors affecting computer-based medical simulation.

Fig 1.2 summarizes some factors which contribute to the related components in computational medical simulation technology [3]. Many investigations have been conducted through last decades [24, 5, 47, 50]. We are interested in investigating the impact comparison between single critical stenosis with multiple sequential subcritical stenoses. We aims to discover the high fidelity simulation results of comparison between single critical stenosis with multiple subcritical stenoses along peripheral artery.

In Fig 1.2, it shows the factors affecting the computational simulation applying

on PAD study. Safe practice refers to contact-less and safety; Versatility refers to the flexibility to construct patient specific case; Deliberate practice refers to the accuracy of diagnosis from doctors and scalability difficulty refers to the learning curve for doctors to employ.

Physically, the PAD can be abstracted as a fluid-structural interaction (FSI) problem by using fluid and structural dynamics equations, respectively. There are two approaches for solving such FSI problems, the monolithic method [28, 15] and the partitioned method [33, 55]. For the monolithic method, the two sets of equations are solved in one large linear system simultaneously. This mutual influence can be considered as internal forces. One significant advantage of this scheme is simplicity. Only a single global matrix is constructed so that both the fluid and structural parts can be solved under the same spatial discretization and time marching scheme. For the trade-off, we loose the flexibility to control fluid and structural parts individually. The other approach is the partitioned method. The two sets of equations are solved separately by passing results as boundary conditions to each other. The solution of fluid equations is calculated while the structural part waits for new boundary condition input, and vice versa. The partitioned method requires a coupling algorithm to exchange the interaction between the two phases as a pair of modules. The Implicit-Explicit (IMEX) Runge-Kutta (RK) time integration approach has been illustrated to be accurate and efficient [61].

A Finite-Volume fluid solver [37, 36, 38] is accomplished and a series of CFD studies has been implemented on idealized geometries. To implement a more high-fidelity simulation, the tissue of blood vessel wall shall be considered as elastic material. Therefore, an accurate and efficient structural solver for solving the elasticity equation is needed. The solver should be capable of calculating the material behavior according to the stiffness property, which is the response of the deformable model reacting to the external forces. The simplest model is a mass-spring model, which is

easy to compute but can not accurately determine the details of material behavior. A more precise approach is the finite element method (FEM), which is based on the continuum mechanics, is more popular. The FEM can specify the stiffness of the model by only using a few characteristic parameters, such as Young's modulus, Poisson ratio and the geometries. However, the challenge for 3D FEM simulation is its parallelization in large-scale parallel computational fluid-structure interaction frameworks is to be used in real-time application due to the very expensive computational cost.

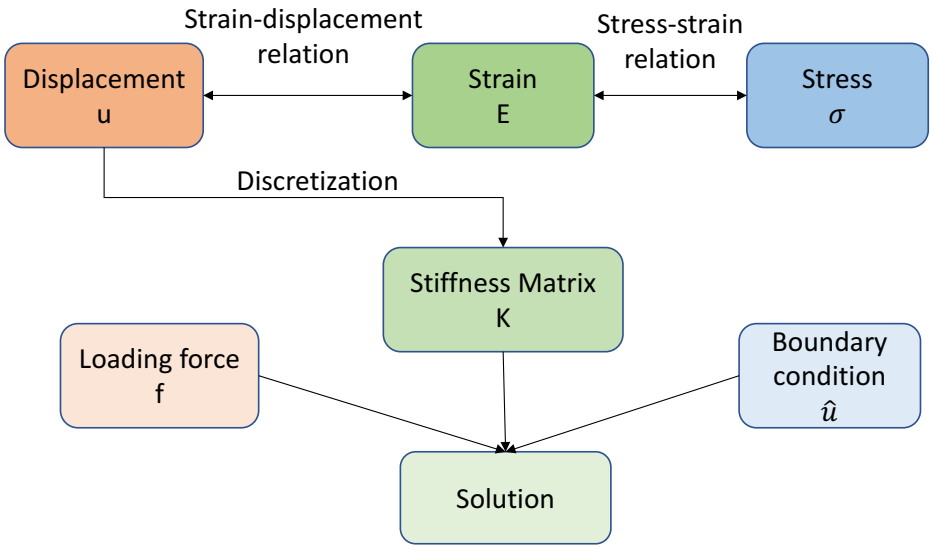


Figure 1.3: Construct stiffness matrix.

Fig 1.3 shows a general process of calculating a deformable material object. Applying stress-strain relation, the unknown variable of governing equation is converted to strain tensor ε. Then after applying the strain-displacement relation, the unknown variable is displacement vector u. The FEM employs basis function to convert the continuous function into construct the discrete matrix K. Combine with the boundary condition $\hat{u}$ and loading force f, we can solve the linear system and obtain the

solution.

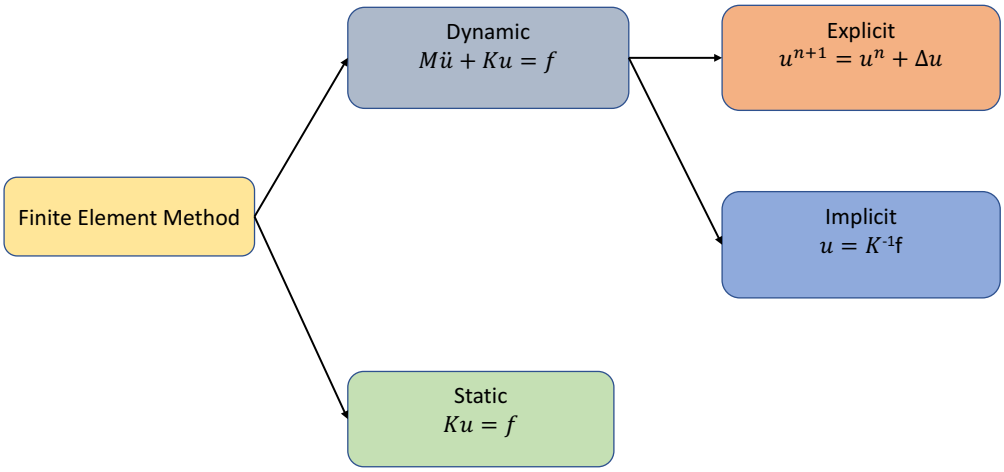


Figure 1.4: Static and dynamic finite element method.

The Fig 1.4 displays both static analysis and dynamic analysis for elasticity equations. The steady-state and transient finite element problem are studied with static analysis and dynamic analysis, respectively. The latter one is time dependent. In dynamic analysis, we shall consider the spatial discretization by using both explicit and implicit time marching schemes [4]. The implicit time marching scheme is computational costly, because a matrix inversion for large linear system is required every time-step. The benefit is unconditionally stable for large time steps. On the other hand, the explicit time marching scheme requires less computational cost for each time-step. However, the size of each time step has to satisfy the numerical stability criteria.

Commonly, the linear system of elasticity equations derived from the FEM method is positive definite . For real engineering problem, the size is too large to be directly computed. The most time consuming part in the FEM analysis in Fig 1.3 is solving

the linear system. The complex system can be written in a matrix form $\mathbf{Ax} = \mathbf{b}$. To solve it, there are two approaches: the direct method and the iterative method. Considering the better performance on memory usage and shorter computing time, the iterative method is preferred for large scale problems [8]. The Preconditioned Conjugate Gradient (PCG) is one of the most popular methods because of its robustness and fast convergence rate. However, the PCG requires positive definite matrices. It is still very challenging to implement finite element analysis using an implicit scheme to study the material real-time response behavior due to the excessive computational cost. Therefore, the domain decomposition method combined with a parallel computing scheme is one emerging choice to reduce computational cost.

Parallel computing desires optimal load balance to achieve high concurrent execution in terms of computational time. The challenges to reach that purpose include learning and understanding programming paradigms, balancing the loads to maximize the usage of bandwidth, minimizing the overhead on data communication and avoiding potential data racing problems. To achieve a good level of parallelism, we employ some modern techniques such as MPI, Intel Math-Kernel library and LAPACK-BLAS for optimal scalability by reducing the overhead latency with a large number of active processors and ensure the balance of loads on each processor.

1.1.2 Overview of Related Numerical Methods

The Weak Galerkin Finite Element Method (WG-FEM) is an efficient numerical method for solving partial differential equations. The WG method is recently developed by Dr. Junping Wang and Dr. Xiu Ye in 2011 and applied for solving second order elliptic equations[58]. The WG method introduces a series of weak operators such as weak gradient, weak divergence and weak curl operators for the computation of corresponding strong forms of differential equations. The WG finite element method provides a new perspective to solve numerical problems. The WG method can be applied on a variety of partial differential equations, such as second

order elliptic equation, elasticity equation[56], Stokes equations [59] and Maxwell's equations [43], etc. The WG method is a discontinuous method. It has two distinct components, weak gradient function and stabilizer. We defined weak function which divides an element into interior and boundary space. Different order of polynomials can be applied on two spaces independently. The weak gradient function connects the unknown variables and weak gradient operator and constructs the elemental matrix. The stabilizer connect the interior and boundary space in a form of boundary integral of projected value difference. The details of the WG method will be discussed in Chapter 2.

1.2 Objectives

The objective of this dissertation is to develop an efficient parallel structure solver by using WG method on unstructured meshes. We deliver it in two steps: designing an improved stabilizer for time marching scheme and developing the WG-BDDC method to solve elasticity problems. The stabilizer is designed for completing mass matrix to enable dynamic study. The WG-BDDC is the first attempt to bring the WG method in the engineering filed and implement it on parallel computers.

The classic continuous Galerkin (CG) finite element analysis of continuum mechanics is widely used to study and predict material deformations. The results are generally accurate and reliable for analyzing the relationship between load and deformation. However, the computational complexity prevents wide adoption to solve large-scale FSI problems. Massively parallel computing is required for obtaining large linear system. In this work, we develop efficient parallel schemes, which is capable for solving large scale elasticity equation.

We employ the WG-FEM to convert the bilinear form equations into a positive definite linear system. The WG method can use multiple types of elements. The selection of polynomials on interior or boundary region is flexible which enhances the freedom of computational simulation for dealing with complex geometries. More-

over, the discontinuous features in the WG method facilitates the implementations of parallel computations.

The Balancing Domain Decomposition by Constraints (BDDC) method is a new approach for the WG-FEM to achieve parallelism on distributed-memory computers. BDDC identifies the two spaces (primal and dual) and applies the divide and conquer strategy (Block Cholesky) to the original computational domain. By minimizing the global communication and synchronization overhead, WG-BDDC can effectively solve equations with superlinear scalability on parallel computers.

Fig 1.5 shows the paths to accomplish the objectives in this work. This research aims to significantly improve the computational efficiency of the elastic material response, including in clinic judgment and other medical applications.

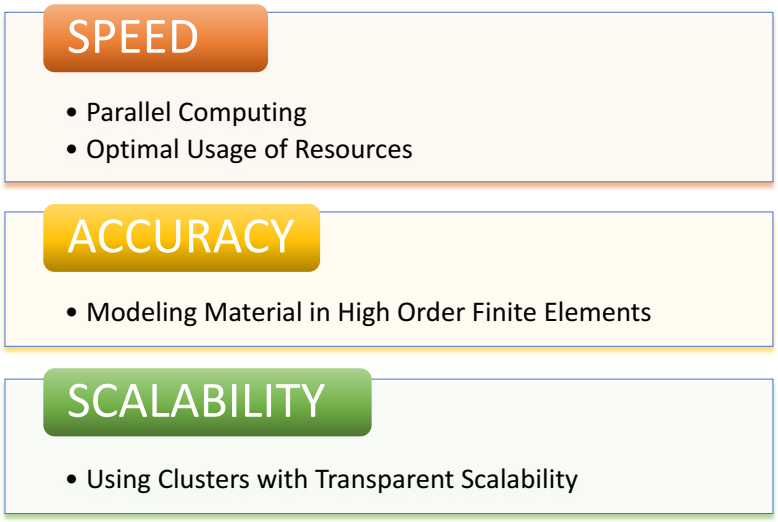


Figure 1.5: Objectives and methodology.

To validate the WG-BDDC scheme and develop a powerful software, we first design a hybrid WG-CG element. The results are consistent with the benchmark of serial CG solver. Then, we extend the WG elements to integrate with BDDC method and verify the convergence and the order of accuracy of the new method (WG-BDDC) on

parallel computers. Finally, the scalability is discussed under the parallel computing framework.

The rest of this dissertation is organized as :

- Chapter 2 presents the background and basics of the WG method.
- Chapter 3 discusses the hybrid WG-CG finite element method. We also tested the performance of the hybrid method for nonlinear elasticity equation.
- Chapter 4 introduces the WG-BDDC method for second order elliptic and elasticity equation.
- Chapter 5 concludes this work and directions for future works.

Chapter 2: Numerical Method

2.1 Mathematical Models and Numerical Method

2.1.1 Linear Elastic Equation and Weak Galerkin Method

Consider an elastic body subject to an exterior force $\mathbf{f}$, and denote the computational domain as Ω and its continuous boundary as $\Gamma = \partial\Omega$. The governing elasticity equation can be written as

$$\nabla \cdot \sigma(\mathbf{u}) = \mathbf{f}, \qquad in \quad \Omega \tag{2.1}$$

$$\mathbf{f} = \hat{\mathbf{f}}, \qquad in \quad \Omega \tag{2.2}$$

$$\mathbf{u} = \hat{\mathbf{u}}, \qquad on \quad \Gamma \tag{2.3}$$

where $\sigma(\mathbf{u})$ is the symmetric Cauchy stress tensor. For linear, isotropic and homogeneous materials, the stress-strain relation is

$$\sigma(\mathbf{u}) = 2\mu\varepsilon(\mathbf{u}) + \lambda(\nabla \cdot \mathbf{u})\mathbf{I} \tag{2.4}$$

where $\varepsilon(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T)$, μ and λ are Lame indices which can be written as

$$\lambda = \frac{E\mu}{(1+\mu)(1-2\mu)} \tag{2.5}$$

$$\mu = \frac{E}{2(1+\mu)} \tag{2.6}$$

where E is the elasticity modulus and μ is the Poisson's ratio.

The weak function on the domain is $\mathbf{u} = \{\mathbf{u}_0, \mathbf{u}_b\}$, $\mathbf{u}_0 \in L^2(T)$. The first

function $\mathbf{u}_0$ represents the interior domain of the function $\mathbf{u}$. The second function $\mathbf{u}_b$ represents the value of function $\mathbf{u}$ on the boundary of domain T . The key notion is that the basis functions of $\mathbf{u}_0$ and $\mathbf{u}_b$ are independent with each other . The weak function is defined as

$$V_h = \{\mathbf{v} = \{\mathbf{v}_0, \mathbf{v}_b\} : \mathbf{v}_0 \in P_j(T^0), \mathbf{v}_b \in P_l(e), e \subset \partial T\} \tag{2.7}$$

The key of the weak Galerkin method is to approximate the solution in the weak discrete space $S(T)$. The discrete weak gradient $\nabla_w \mathbf{u} \in [P_r(T)]^d$ for $\mathbf{u} \in V_h$ on each element T:

$$(\nabla_w \mathbf{u}, \mathbf{q})_T = -(\mathbf{u}_0, \nabla \cdot \mathbf{q})_T + \langle \mathbf{u}_b, q \cdot \mathbf{n} \rangle_{\partial T} \tag{2.8}$$

For the discrete weak divergence, $\nabla_w \cdot \mathbf{u} \in [P_r(T)]^d$ is defined

$$(\nabla_w \cdot \mathbf{u}, \mathbf{q})_T = -(\mathbf{u}_0, \nabla \mathbf{q})_T + \langle \mathbf{u}_b \cdot \mathbf{n}, q \rangle_{\partial T} \tag{2.9}$$

Then we can define the weak strain tensor by using the weak gradient

$$\varepsilon_w(\mathbf{u}) = \frac{1}{2}(\nabla_w \mathbf{u} + \nabla_w \mathbf{u}^T) \tag{2.10}$$

Analogously, we can define the weak stress tensor as

$$\sigma_w(\mathbf{u}) = 2\mu\varepsilon_w(\mathbf{u}) + \lambda(\nabla_w \cdot \mathbf{u})\mathbf{I} \tag{2.11}$$

The bilinear form of governing equation of continuous Galerkin method is following

$$a(\mathbf{u}, \mathbf{v}) = (\mathbf{f}, \mathbf{v}) \tag{2.12}$$

In WG method, to complete the elemental stiffness matrix, we need the stabilizer to connect the interior and boundary unknown variables. The matrix form of

governing equation is

$$a(\mathbf{u}_w, \mathbf{v}_w) + s(\mathbf{u}, \mathbf{v}) = (\mathbf{f}, \mathbf{u}) \tag{2.13}$$

the term $s(\mathbf{u}, \mathbf{v})$ is a stabilizer enforcing a weak continuity which measures the discontinuity of the finite element solution. The governing equation in weak form can be introduced by two bilinear equations

$$s(\mathbf{u}, \mathbf{u}) = \sum_{T \in \Omega}^{N} h_T^{-1} \langle Q_b \mathbf{u}_0 - \mathbf{u}_b, Q_b \mathbf{v}_0 - \mathbf{v}_b \rangle_{\partial T} \tag{2.14}$$

where Q_b is the projection from the interior unknown variables to boundary unknown variables. Commonly it is taken as 1. The matrix form of governing equation is assembled as

$$a(\mathbf{u}_w, \mathbf{u}_w) = \sum_{T \in \Omega}^{N} 2(\mu \varepsilon_w(\mathbf{u}), \varepsilon_w(\mathbf{v}))_T + \sum_{T \in \Omega}^{N} (\lambda \nabla \cdot \mathbf{u}, \nabla_w \cdot \mathbf{v})_T \tag{2.15}$$

2.2 Existing Numerical Methods Review

In this section, we present and analyze several most widely used numerical method for solving finite element problems. The details of each method are presented in the following subsections.

2.2.1 Classic Continuous Galerkin Finite Element Method

Back to 1950s and 1960s, finite element method is invented to solve complex elasticity and structural analysis engineering problems in mechanical and aeronautical field. A. Hrennikoff [27], R. Courant [12] and K.Feng [**?**] are the earliest pioneers who established this subject. FEM is then proposed as a systematic numerical method to solve variety of partial differential equations. The core characteristic of FEM is that it employs mesh discretization to divide a continuous computational domain. Therefore, a big problem is then converted to a set of discrete small problems. That

is the source of finite element. Each element represents a small piece of computational sub-domain.

FEM is an efficient solution to solve partial differential equations. It converts the original partial differential equation to an equivalent bilinear form as the weak function. Then we partition the computational domain into polygon meshes. In each mesh element, we construct the finite element space. The bilinear form is discretized into a summation of finite elemental matrices. The solution is approximated based on the calculation of assembled matrix. More details can be found in [62, 9, 29, 51].

The variational formulation is derived from the governing equation. It determines the characteristic of the finite element method. To obtain the variational form, mathematicians derived several different paths such as Galerkin method, the discontinuous Galerkin method, mixed method, etc. In this chapter, we introduce two most popular method, DG and MFEM. The WG methods is inspired from these two method and shares many similarities with them.

2.2.2 Discontinuous Galerkin Finite Element Method

We shall go through a simple example to review the common features of FEM and the new feature of DG-FEM.

We have a 1-dimensional convection equation with domain $[0, \pi]$

$$a\frac{du}{dx} = cos(x) \tag{2.16}$$

the exact solution is

$$u = \frac{sin(x)}{a} \tag{2.17}$$

where the Dirichlet boundary condition is applied on both end.

For the classic CG FEM, we shall first apply the basis function ϕ_i on each element.

The solution becomes as

$$u(x) = \sum_{i=1}^{N} \phi_i(x) U_i \tag{2.18}$$

We integrate the governing equation and obtain the weak form

$$\int_\Omega a \frac{du}{dx} \phi_i dx = \int_\Omega cos(x) \phi_i dx \tag{2.19}$$

Then we apply integration by parts

$$-\int_\Omega au\phi_{i,x} dx + [au\phi_i]_{left}^{right} = \int_\Omega cos(x)\phi_i dx \tag{2.20}$$

Discrete form of the continuous unknown function is

$$u(x) = \sum_{i=1}^{N} \phi_i(x) U_i \tag{2.21}$$

So we obtain the new governing equation in matrix form

$$-\sum_{i=1}^{N} U_i a \int_\Omega \phi_i \phi_{j,x} dx + BC = \int_\Omega cos(x)\phi_j dx \tag{2.22}$$

The linear system is simplified as

$$\mathbf{Ax} = \mathbf{b} \tag{2.23}$$

In DG-FEM, there is no continuity constraint between element. In another word, the basis functions in the computational domain is discontinuous.

The discretized governing equation in matrix form is

$$a \int_\Omega \frac{du}{dx} \phi_i dx = \int_\Omega cos(x)\phi_i dx \tag{2.24}$$

For the integration by parts, the equation is written in

$$\sum_j \int_j a\frac{du}{dx}\phi_i dx = \sum_j \int_j cos(x)\phi_i dx \tag{2.25}$$

Then we apply the integration by parts

$$\sum_j \{-\int_j au\phi_{i,x}dx + [(au)\phi_i]_{j-\frac{1}{2}}^{j+\frac{1}{2}}\} = \sum_j \int_j cos(x)\phi_i dx \tag{2.26}$$

For every element, we have the elemental matrix form

$$-\int_j au\phi_{j,x}dx + [(au)\phi_i]_{j-\frac{1}{2}}^{j+\frac{1}{2}} = \int_j cos(x)\phi_j dx \tag{2.27}$$

A penalty term is required in the above equation representing a jump of the interface value. There are several methods to calculate it. Upwinding method is a common useful tool to calculate the u value on the interface. Comparing to the CG method, the DG method introduces additional DOFs to maintain the continuity as jump penalty term.

There are several widely applied iterative schemes such as block Jacobi, Conjugate Gradient method are stable for elements with high order polynomials. Domain decomposition is compatible with the DG method as parallel computing scheme.

2.2.3 Mixed Finite Element Method

The mixed finite element method is a new type of the FEM in which an extra independent variable is introduced which discretization of a partial differential equation. It is characterized by a variational equation. Usually, the extra variables are constrained by an external vector, Lagrange multiplier. The main difference between FEM and mixed FEM is that the changing of bilinear form of governing equation. The mixed FEM has an advantage on computing the elasticity equation and the stress and strain field.

To illustrate the mixed FEM, a simple elliptic equation is

$$-\nabla \cdot (a\nabla u) = f \quad in \quad \Omega \tag{2.28}$$

$$u = 0 \quad on \quad \partial\Omega \tag{2.29}$$

we shall introduce an vector $\mathbf{G}$ that

$$\mathbf{G} = -a\nabla u \tag{2.30}$$

the elliptic problem then can be decomposed into a first order linear system that

$$\mathbf{G} + a\nabla u = 0 \quad in \quad \Omega \tag{2.31}$$

$$\nabla \cdot \mathbf{G} = f \quad in \quad \Omega \tag{2.32}$$

$$u = 0 \quad on \quad \partial\Omega \tag{2.33}$$

we can rewrite the first equation as

$$a^{-1}\mathbf{G} + \nabla u = 0 \quad in \quad \Omega \tag{2.34}$$

then we apply the integration by parts

$$\int_\Omega a^{-1}\mathbf{G} \cdot \mathbf{v} d\Omega - \int_\Omega c\nabla \cdot \mathbf{v} d\Omega = 0 \quad \mathbf{v} \in H(div, \Omega) \tag{2.35}$$

$H(div, \Omega)$ represents the integral function is defined in Hilbert space as Ω.

$$\int_{\Omega} \psi \nabla \cdot \mathbf{G} d\Omega = \int_{\Omega} f \psi d\Omega \tag{2.36}$$

where

$$H(div, \Omega) = \{\mathbf{v} \in L^2(\Omega)^d : \nabla \cdot \mathbf{v} \in L^2(\Omega)\} \tag{2.37}$$

We use the Raviart-Thomas spaces to define a generic element $T \in \Omega$

$$RT_k = (P_k)^d + xP_k \tag{2.38}$$

where k is the order of polynomials degrees.

the variational function $\tilde{G}$ and $\mathbf{u}$ can be approximated by

$$\tilde{G} = \sum_{i=1}^{N} g_i \mathbf{v}_i \tag{2.39}$$

$$\mathbf{u} = \sum_{i=1}^{N} u_i \psi_i \tag{2.40}$$

both $\mathbf{v}$ and ϕ_i are vector basis functions. If we consider the RT_0 space,

$$\mathbf{v} = \begin{pmatrix} ax + b \\ ay + c \end{pmatrix} \tag{2.41}$$

for each element $T_i, (i = 1, 2, 3)$

$$v_i = \begin{pmatrix} a_i x + b_i \\ a_i y + c_i \end{pmatrix} \tag{2.42}$$

the above equation follows the Kronecker property [14]

$$\int_T v_i \cdot \mathbf{n} dS = \int_T (ax+b)n_x + (ay+c)n_y dS \tag{2.43}$$

$$= a\int_{x_1}^{x_2} (x_1 n_x + y_1 n_y)\frac{1}{n_y}dx + \int_{x_1}^{x_2}(bn_x + cn_y)\frac{1}{n_y}dx \tag{2.44}$$

$$= a(x_1 n_x + y_1 n_y)\frac{x_1 - x_2}{n_y} + (bn_x + cn_y)\frac{x_1 - x_2}{n_y} \tag{2.45}$$

$$= \delta_{ij} \tag{2.46}$$

Then we apply the divergence theorem, we have

$$\int_T \nabla \cdot \mathbf{v} dxdy = \int_T (\frac{\partial \mathbf{v}}{\partial x} + \frac{\partial \mathbf{v}}{\partial y})dS \tag{2.47}$$

$$= \int_T 2a dS \tag{2.48}$$

$$= 2a|T_i| \tag{2.49}$$

2.3 Weak Galerkin Finite Element Methods

2.3.1 Preliminary

In this section, all computation is in Sobolev space[7, 9]. The Lipschitz boundary is in open area $D \subset R^d$, $d = 2, 3$. The inner product is defined as

$$|v|_{s,D} = \left(\sum_{|\alpha|=s} \int_D |\partial^2 v|^2 dD\right)^{1/2} \tag{2.50}$$

where

$$\alpha = (\alpha_1, \ldots, \alpha_d) \tag{2.51}$$

$$|\alpha| = \alpha_1 + \cdots + \alpha_d \tag{2.52}$$

The definition of divergence is defined by

$$H(div; D) = \{\mathbf{v} : \mathbf{v} \in [L^2(D)]^d, \nabla \cdot \mathbf{v} \in L^2(D)\} \tag{2.53}$$

the norm is defined as

$$||\mathbf{v}||_{H(div,D)} = (||\mathbf{v}||_D^2 + ||\nabla \cdot \mathbf{v}||_D^2)^{1/2} \tag{2.54}$$

the curl of $H(curl; D)$ in $L^2(D)$ is defined as

$$H(curl; D) = \{\mathbf{v} : \mathbf{v} \in [L^2(D)]^d, \nabla \times \mathbf{v} \in L^2(D)\} \tag{2.55}$$

the norm of it is defined as

$$||\mathbf{v}||_{H(curl;D)} = (||\mathbf{v}||_D^2 + ||\nabla \times \mathbf{v}||_D^2)^{1/2} \tag{2.56}$$

2.3.2 Weak operators

Let's assume that $T \subset R^d$ is an arbitrary polygon domain, the boundary is ∂T. The weak function in the domain is $v = \{v_0, v_b\}$, so that $v_0 \in L^2(T)$ and $v_b \in L^2(\partial T)$. The v_0 represents the unknown variable vector belongs to the interior domain, and v_b represents the unknown variable vector along the boundary of T. The keynote is that the v_0 is independent with v_b on ∂T. The weak space for all T is $S(T)$

$$S(T) = \{v = \{v_0, v_b\} : v_0 \in L^2(T), v_b \in L^2(\partial T)\} \tag{2.57}$$

Now we define some commonly used weak operators

The weak gradient operator, for any $v \in S(T)$, the weak gradient of v is $\nabla_w v$

$$(\nabla_w v, \mathbf{q})_T = -(v_0, \nabla \cdot \mathbf{q})_T + \langle v_b, \mathbf{q} \cdot \mathbf{n} \rangle_{\partial T} \tag{2.58}$$

the discrete weak gradient operator is $\nabla_{w,r} v$

$$(\nabla_{w,r} v, \mathbf{q})_T = -(v_0, \nabla \cdot \mathbf{q})_T + \langle v_b, \mathbf{q} \cdot \mathbf{n} \rangle_{\partial T} \tag{2.59}$$

where

$$V(T) = \{\mathbf{v} = \{\mathbf{v}_0, \mathbf{v}_b\} : \mathbf{v}_0 \in [L^2(T)]^d, \mathbf{v}_b \in [L^2(\partial T)]^d\} \tag{2.60}$$

The weak divergence operator, for any $v \in S(T)$, is $\nabla \cdot \mathbf{v}$

$$\langle \nabla_w \cdot \mathbf{v}, \varphi \rangle_T = -(\mathbf{v}_0, \nabla \varphi)_K + \langle \mathbf{v}_b \cdot \mathbf{n}, \varphi \rangle_{\partial T} \tag{2.61}$$

the discrete weak divergence operator is

$$(\nabla_{w,r,T} \cdot \mathbf{v}, \varphi)_T = -(\mathbf{v}_0, \nabla \varphi)_T + \langle \mathbf{v}_b \cdot \mathbf{n}, \varphi \rangle_{\partial T} \tag{2.62}$$

the curl operator , for any $v \in S(T)$, is $\nabla_w \times \mathbf{v}$ which is defined as

$$\langle \nabla_w \times \mathbf{v}, \varphi \rangle_T = (\mathbf{v}_0, \nabla \times \varphi)_T - \langle \mathbf{v}_b \times \mathbf{n}, \varphi \rangle_{\partial T} \tag{2.63}$$

the discrete weak divergence operator is

$$(\nabla_{w,r,T} \times \mathbf{v}, \varphi)_T = (\mathbf{v}_0, \nabla \times \varphi)_T - \langle \mathbf{v}_b \times \mathbf{n}, \varphi \rangle_{\partial K} \tag{2.64}$$

2.3.3 Weak Galerkin method for second order elliptic equation

In this section, we use the weak operators to solve second order elliptic equation. In domain Ω, we have the equation in form

$$-\nabla \cdot (a \nabla u) = f \tag{2.65}$$

Considering the Dirichlet boundary condition, we have

$$u = -g, \quad on \quad \partial\Omega \tag{2.66}$$

For Neumann boundary condition, we have

$$(a\nabla u) \cdot \mathbf{n} = -g, \quad on \quad \partial\Omega \tag{2.67}$$

The primal formulation for Dirichlet boundary condition is

$$(a\nabla u, \nabla v) = (f, v) \tag{2.68}$$

For Neumann boundary condition

$$(a\nabla u, \nabla v) = (f, v) - \langle g, v\rangle_{\partial\Omega} \tag{2.69}$$

The Primal-Mixed formulation is to make $\mathbf{q} = -a\nabla u$, the elliptic equation is

$$a^{-1}\mathbf{q} + \nabla u = 0, \tag{2.70}$$

$$\nabla \cdot \mathbf{q} = f. \tag{2.71}$$

we choose assistant function $\mathbf{p} \in [L^2(\Omega)]^d$, the bilinear form is

$$(a^{-1}\mathbf{q}, \mathbf{p}) + (\nabla u, \mathbf{p}) = 0; \tag{2.72}$$

for any $v \in H_0^1(\Omega)$

$$(\mathbf{q}, \nabla v) = -(f, v) \tag{2.73}$$

For Dirichlet boundary condition, $u \in H^1(\Omega)$, $\mathbf{q} \in [L^2(\Omega)]^d$, so that we enforce the boundary condition $u = -g$ on $\partial\Omega$

$$(a^{-1}, \mathbf{q}, \mathbf{p}) + (\nabla u, \mathbf{p}) \tag{2.74}$$

$$(\mathbf{q}, \nabla v) = -(f, v), \tag{2.75}$$

for Neumann boundary condition

$$(a^{-1}\mathbf{q}, \mathbf{p}) + (\nabla u, \mathbf{p}) = 0, \tag{2.76}$$

$$(\mathbf{q}, \nabla v) = \langle g, v\rangle_{\partial\Omega} - (f, v), \tag{2.77}$$

Let's define the weak function on each element space

$$S(k, T) = \{v = \{v_0, v_b\} : v_0 \in P_k(T), v_b|_e \in P_k\} \tag{2.78}$$

where k is the order of polynomial for each function, and e is the boundary edge of each element.

The weak space S_h is defined as

$$S_h = \{v = \{v_0, v_b\} : v|_T \in S(k, T), v_b|_{\partial T}\} \tag{2.79}$$

for each element T, we use Q_0 to represent the L^2 projection from L^T to $P_k(T)$ and Q_b is the L^2 projection from $L^2(T)$ to $P_k(\partial T)$. The local weak discrete space Q_h is

$$Q_h v = \{Q_0 v_0, Q_b, v_b\} \tag{2.80}$$

in space S_h we have the bilinear form

$$a(u,v) = \sum (a\nabla_w u, \nabla_w v)_T \tag{2.81}$$

To connect the two independent spaces, interior and boundary, a stabilizer is introduced. The stabilizer describes the difference of boundary values and the projected interior value to the boundary. Since the interior space and boundary space is independent, two sets of basis functions are applied independently. When we project the value from interior basis function to boundary basis function, the two values are different even though they share the same location. The stabilizer is a boundary integral which measure the difference and connect the spaces.

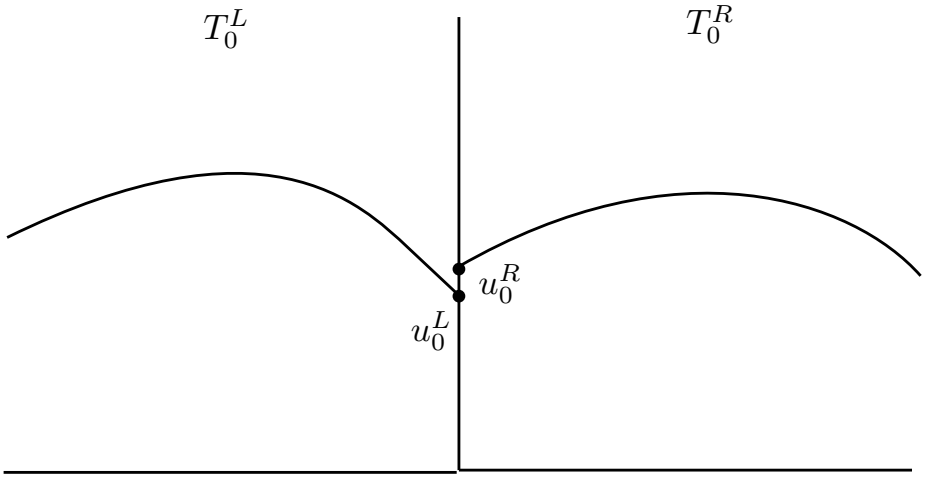


Figure 2.1: 1-D interior basis functions of two adjcent elements.

The above figure shows the adjacent elements using identical interior basis functions. The curves in two neighbor elements are interior basis functions. The boundary basis function are constant locates on the boundary edge. However, the projection values, u_0^L and u_0^R, from interior to boundary are different. The following equation is the stabilizer to measure the difference and connect the interior values to the bound-

ary value.

$$s(u,v) = \rho \sum h_T^{-1} \langle Q_b u_0 - u_b, Q_b v_0 - v_b \rangle_{\partial T} \tag{2.82}$$

where ρ is commonly set as 1, and h is the characteristic length of each element. The governing equation is

$$a_s(u,v) = a(u,v) + s(u,v) \tag{2.83}$$

The above equation is the final elemental stiffness matrix. It consists elemental matrix from bilinear form and the stabilizer. After the superposition with connectivity, we construct the global stiffness matrix.

2.3.4 Weak Galerkin Triangular Meshes

Consider triangular element linear type basis function for both interior and boundary subspaces $P_1(T)/P_1(\partial T)$

$$\phi_k = \{\lambda_k, 0\}, \qquad k = 1,2,3 \tag{2.84}$$

$$\phi_{3+l} = \{0, \mu_l\}, \qquad l = 1,2,\cdots,2N \tag{2.85}$$

where N is the number of element boundaries.

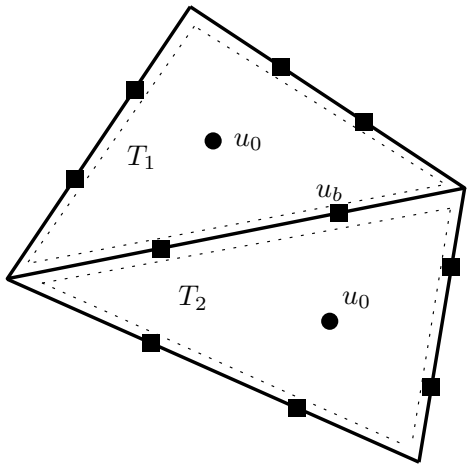


Figure 2.2: Weak Galerkin triangular elements and solution points.

For Fig. **??**, we set linear basis functions, two unknown variables, on each edge. The dash line represents the fictitious boundary of interior space. We can apply different order of polynomials on it. The two spaces are independent of each other.

In this section, we present a triangular WG element with linear interior and boundary space. There are three interior basis functions representing the x, y and xy respectively.

2.3.5 Weak Galerkin Quadrilateral Meshes

Consider triangular element linear type basis function for both interior and boundary subspaces $Q_1(T)/Q_1(\partial T)$

$$\phi_k = \{\lambda_k, 0\}, \qquad k = 1, 2, 3 \tag{2.86}$$

$$\phi_{3+l} = \{0, \mu_l\}, \qquad l = 1, 2, \cdots, 2N \tag{2.87}$$

where N is the number of element boundaries.

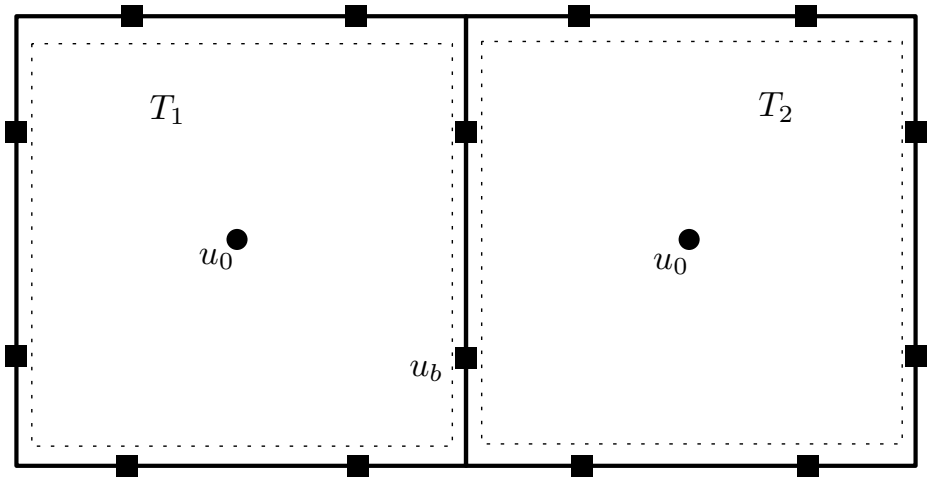


Figure 2.3: Weak Galerkin quadrilateral elements and solution points.

In above figure, we present a quadrilateral WG element with linear interior and boundary space. There are three interior basis functions representing the x, y and xy respectively. Meanwhile, there are eight boundary basis functions represents six solution points based on the location of Gaussian points.

2.3.6 Numerical Examples

We also consider the linear elasticity Equation in the square domain $\Omega = (0, 1)^2$. For each subdomain, it is partitioned into uniform triangular and quadrilateral mesh wish mesh size h. The right-hand side function f is chosen. The exact solution is given by

$$u = \begin{pmatrix} sin(2\pi x)sin(2\pi y) \\ 1 \end{pmatrix} \tag{2.88}$$

the solution is shown as below

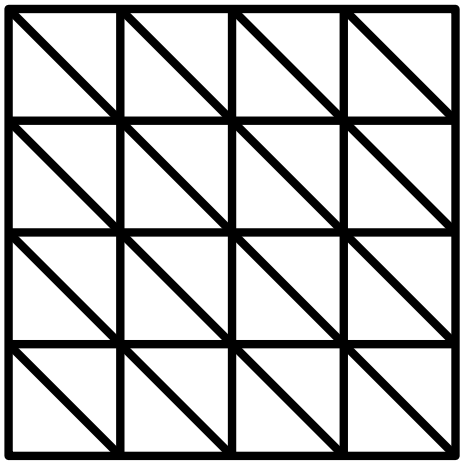


Figure 2.4: Triangle mesh elements

Table 2.1: Numerical results for triangular element.

# CG = 20				
#WG	$E(u_0)$	$O(u_0)$	$E(u_b)$	$O(u_b)$
4	$9.484e-2$	-	$8.484e-2$	-
16	$2.678e-3$	1.9	$2.178e-3$	1.9
64	$5.570e-4$	2.0	$5.470e-4$	1.9
256	$1.437e-4$	2.0	$1.367e-4$	2.0

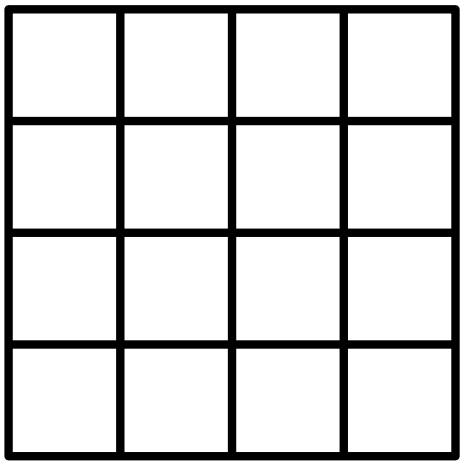


Figure 2.5: Quadrilateral mesh elements

Table 2.2: Numerical results for quadrilateral element.

# CG = 20				
#WG	$E(u_0)$	$O(u_0)$	$E(u_b)$	$O(u_b)$
4	$8.368e-2$	-	$9.103e-2$	-
16	$1.944e-3$	1.9	$2.058e-3$	1.9
64	$5.337e-4$	2.0	$5.280e-4$	2.0
256	$1.086e-4$	2.0	$1.451e-4$	2.0

We tested the WG method for solving elasticity equation. Second order of accuracy is achieved for interior unknown variables u_0 and boundary unknown variables u_b in the triangular and quadrilateral elements.

Chapter 3: Hybrid Weak Galerkin - Continuous Galerkin Finite Element Method

In this chapter, we present a novel parallel computing method to effectively solve linear and nonlinear elasticity equations. The WG finite element method is newly developed by Dr. Junping Wang and Dr. Xiu Ye [57]. We develop a novel hybrid element which combines the elements of both WG and CG finite element. The new hybrid element inherits the discontinuous feature of the WG method. To create a hybrid element, we treat the WG element as cargo ship and the CG element as the container. We insert multiple CG elements in one single WG element.

The hybrid element takes advantages from both the WG and the CG element. For the WG element, it is discontinuous and can be calculated as a non-overlapping subdomain without extra unknown variables, such as Lagrangian multiplier. For the CG element, it's simple to construct and compute. The hybrid element use WG element as the subdomain boundary interface and CG and interior refine elements.

Like the standard FEM, the WG method can be used to solve generic partial differential equations. An advanced feature of the WG finite element method is that the entire problem can be decomposed into multiple local problems. In these local problems, the differential operators are approximated and reconstructed by small-size matrices. The WG method provides optimal orders of accuracy in spatial discretization for difficult problems on serial computers. However, the performance of the WG method on parallel computers has not yet been examined.

This chapter is organized as following. We first introduce the weak Galerkin method and the derivations of the bilinear form of nonlinear elasticity equation. Then we discuss the details of matrix construction. An improved stabilizer is designed for implementing time marching schemes. For the numerical method, we present the details of the parallel scheme including the explicit Central Difference method and

the implicit time marching scheme, the Newmark-Beta method [1]. At the end of this chapter, we present the numerical results of one-dimensional linear and nonlinear elastic equation. The second order accuracy are achieved with observed superlinear speedup.

3.1 Nonlinear Elasticity Equation

Consider an elastic body subject to an exterior force $\mathbf{f}$, we denote the computational domain as Ω and its continuous boundary as $\Gamma = \partial\Omega$. The governing equation for elasticity can be written as

$$-\nabla \cdot \boldsymbol{\sigma}(\boldsymbol{u}) = \mathbf{f}, \quad in \quad \Omega \tag{3.1}$$

$$\mathbf{u} = \hat{\mathbf{u}}, \qquad on \quad \Gamma \tag{3.2}$$

where $\boldsymbol{\sigma}(\boldsymbol{u})$ is Cauchy stress. For isotropic and homogeneous materials, the stress-strain relation is

$$\boldsymbol{\sigma}(\boldsymbol{u}) = 2\mu\varepsilon(\mathbf{u}) + \lambda(\nabla \cdot \mathbf{u})\mathbf{I} \tag{3.3}$$

the μ and λ are Lame indices which can be written as

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \tag{3.4}$$

where E is the elasticity modulus and ν is Poisson's ratio.

The geometric nonlinear strain-displacement relation can be written as following format

$$\boldsymbol{\varepsilon}(\boldsymbol{u}) = \frac{1}{2}(\nabla u + \nabla u^T - (\nabla u^T) \cdot \nabla u) \tag{3.5}$$

The weak function on domain is $u = \{u_0, u_b\}$, $u_0 \in L^2(T)$. The first function u_0 represents the interior domain of the function u. The second function u_b represents

the boundary value of function u. The key notion is that for two functions u_0 and u_b are independent with each other. The weak function is defined as

$$U_h = \{u = \{u_0, u_b\} : u_0 \in P_j(T^0), u_b \in P_l(e), e \subset \partial T\} \tag{3.6}$$

The key of the weak Galerkin method is to approximate the solution in weak discrete space $S(T)$. The discrete weak gradient $\nabla_w u \in [P_r(T)]^d$ for $u \in V_h$ on each element T:

$$(\nabla_w, q)_T = -(u_0, \nabla \cdot q)_T + \langle u_b, q \cdot \mathbf{n} \rangle_{\partial T} \tag{3.7}$$

for the discrete weak divergence, $\nabla_w \cdot \mathbf{u} in [P_r(T)]^d$ is defined

$$(\nabla_w \cdot u, q)_T = -(u_0, \nabla q)_T + \langle u_b \cdot \mathbf{n}, q \rangle_{\partial T} \tag{3.8}$$

Then we can define the weak linear strain tensor by using the weak gradient

$$\varepsilon_w(u) = \frac{1}{2}(\nabla_w u + \nabla_w u^T) \tag{3.9}$$

Analogously, the nonlinear weak strain tensor is defined by

$$\varepsilon_w(u) = \frac{1}{2}(\nabla_w u + \nabla_w u^T + (\nabla_w u^T) \cdot \nabla_w u) \tag{3.10}$$

The weak stress can be defined as

$$\sigma_w(u) = 2\mu\epsilon_w(u) + \lambda(\nabla_w \cdot u)\mathbf{I} \tag{3.11}$$

The weak form of governing equation of continuous Galerkin method as

$$a(\mathbf{u}, \mathbf{v}) = (\mathbf{f}, \mathbf{v}) \tag{3.12}$$

the above approximation function v and the gradient ∇v is not well defined for the discontinuous feature of weak Galerkin method. The new form is

$$a(\mathbf{u}_w, \mathbf{v}_w) + s(\mathbf{u}, \mathbf{v}) = (\mathbf{f}, \mathbf{v}) \tag{3.13}$$

the term $s(\mathbf{u}, \mathbf{v})$ is a stabilizer enforcing a weak continuity which measures the discontinuity of the finite element solution. The governing equation in weak form can be introduced by two bilinear equations

$$s(\mathbf{u}, \mathbf{v}) = \sum_{T \in \Omega}^{N} h_T^{-1} \langle Q_b u_0 - u_b, Q_b v_0 - b_b \rangle_{\partial T} \tag{3.14}$$

where Q_b is the projection parameter defined by the interior and boundary value. h_T is the characteristic length of mesh element. It is a constant number which usually is taken as 1.

The governing equation has the weak form as

$$a(\mathbf{u}_w, \mathbf{v}_w) = \sum_{T \in \Omega}^{N} 2(\mu \varepsilon_w(u), \varepsilon_w(v))_T + \lambda(\nabla_w \cdot \mathbf{u}, \nabla_w \cdot \mathbf{v})_T \tag{3.15}$$

3.2 Hybrid WG-CG Element

In this section, we introduce the design of hybrid element, the construction of the basis functions. The one dimensional WG element has the shape function

$$\phi_0^1 = \frac{x - x_{i+1}}{x_i - x_{i+1}}, \quad \phi_b^1 = 1 \tag{3.16}$$

$$\phi_0^2 = \frac{x - x_i}{x_{i+1} - x_i}, \quad \phi_b^2 = 1 \tag{3.17}$$

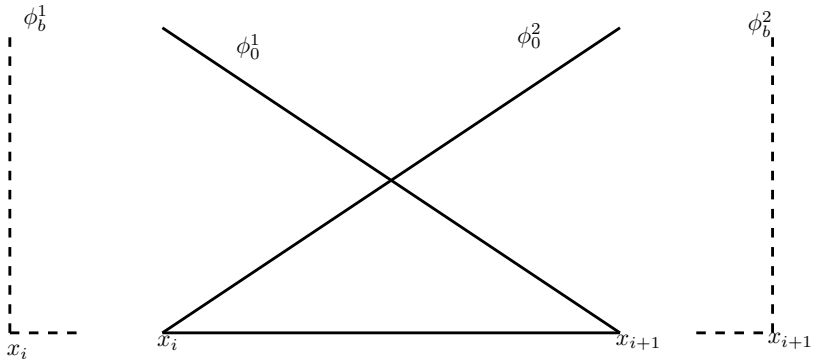


Figure 3.1: Basis function of one dimensional weak Galerkin element

Based on the weak gradient function, the boundary basis functions are independent of the interior basis functions, Therefore, we can insert an arbitrary number of CG element basis functions into the WG element. Each weak Galerkin element is considered as one sub-element that neighbored with adjacent elements. The interior CG elements are corrected by the coarse mesh which is the WG element. The boundary basis function of WG element is shared by adjacent elements and constitute the coarse mesh. The newly designed basis function of hybrid element can be illustrated as

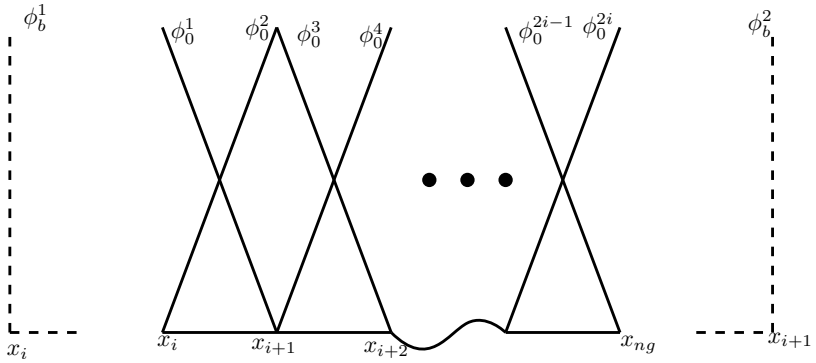


Figure 3.2: Basis function of hybrid WG-CG element, an arbitrary number of CG elements are inserted into WG element

As above figure, in one WG element, there are multiple CG element basis functions. All the interior basis functions are linear and shared the same boundary basis functions of WG element. This is the illustration of the hybrid element basis func-

tions.

3.3 The Improved Stabilizer

3.3.1 Deficiency of Original Stabilizer

Here we present the parallel computing scheme for one dimensional linear and nonlinear elastic equations. The governing equation for dynamic equation is

$$M\ddot{u} + Ku = f \tag{3.18}$$

In the hybrid elements, the information of mass is missing for the corresponding boundary values. The original governing equation in matrix form is

$$\begin{bmatrix} M_{00} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{u}_0 \\ \ddot{u}_b \end{bmatrix} + \begin{bmatrix} K_{00} & K_{0b} \\ K_{b0} & K_{bb} \end{bmatrix} \begin{bmatrix} u_0 \\ u_b \end{bmatrix} = \begin{bmatrix} f_0 \\ 0 \end{bmatrix} \tag{3.19}$$

from above equation we can find that the mass matrix for $\ddot{u}_b$ is not complete. The information of mass is missing on the boundary basis functions. Under this circumstance, the explicit time marching scheme is unable to initiate.

3.3.2 Improvement

To avoid the deficit of time marching scheme, we introduce a newly designed stabilizer, namely, the generic stabilizer.

$$s(\mathbf{u}, \mathbf{v}) = \sum_{T \in \Omega}^{N} h_T^{-1} \langle Q_b \ddot{u}_0 - \ddot{u}_b, Q_b v_0 - v_b \rangle_{\partial T} \tag{3.20}$$

Consider the Central difference method as

$$\frac{\partial^2 u}{\partial^2 t} = \frac{u^{n+1} - 2u^n + u^{n-1}}{(\Delta t)^2} \tag{3.21}$$

For one stabilizer matrix $h^{-1}\langle Q_b\ddot{u} - \ddot{u}_b, Q_b v_0 - v_b\rangle$, we apply Eq. (3.21) on it and

rewrite it as

$$\langle h^{-1}\frac{Q_b(u_0^{n+1}-2u_0^n+u_0^{n-1})}{(\Delta t)^2}, Q_b v_0 - v_b\rangle - h^{-1}\langle\frac{u_b^{n+1}-2u_b^n+u_b^{n-1}}{(\Delta t)^2}, Q_b v_0 - v_b\rangle \quad (3.22)$$

Then we calculate the boundary integral and write the result in matrix form S. The stabilizer is written in

$$\frac{h^{-1}}{(\Delta t)^2}\langle Q_b u_0^{n+1}-u_b^{n+1}, Q_b v_0-v_b\rangle-\frac{2h^{-1}}{(\Delta t)^2}\langle Q_b u_0^n-u_b^n, Q_b v_0-v_b\rangle+\frac{h^{-1}}{(\Delta t)^2}\langle Q_b u_0^{n-1}-u_b^{n-1}, Q_b v_0-v_b\rangle \quad (3.23)$$

$$\frac{h^{-1}}{(\Delta t)^2}\left(\begin{bmatrix} S_{00} & S_{0b} \\ S_{b0} & S_{bb}\end{bmatrix}\cdot\begin{bmatrix} u_0^{n+1} \\ u_b^{n+1}\end{bmatrix} - 2\begin{bmatrix} S_{00} & S_{0b} \\ S_{b0} & S_{bb}\end{bmatrix}\cdot\begin{bmatrix} u_0^{n} \\ u_b^{n}\end{bmatrix} + \begin{bmatrix} S_{00} & S_{0b} \\ S_{b0} & S_{bb}\end{bmatrix}\cdot\begin{bmatrix} u_0^{n-1} \\ u_b^{n-1}\end{bmatrix}\right) \quad (3.24)$$

We combine the discretized stabilizer with the incomplete equation Eq. (3.19), we can construct the governing equation in a new matrix form

$$h^{-1}\begin{bmatrix} M_{00}+S_{00} & S_{0b} \\ S_{b0} & S_{bb}\end{bmatrix}\cdot\begin{bmatrix} u_0^{n+1} \\ u_b^{n+1}\end{bmatrix} + (\Delta t)^2\begin{bmatrix} K_{00} & K_{0b} \\ K_{b0} & K_{bb}\end{bmatrix}\cdot\begin{bmatrix} u_0^{n+1} \\ u_b^{n+1}\end{bmatrix} \quad (3.25)$$

$$= 2h^{-1}\begin{bmatrix} S_{00} & S_{0b} \\ S_{b0} & S_{bb}\end{bmatrix}\cdot\begin{bmatrix} u_0^{n} \\ u_b^{n}\end{bmatrix} - h^{-1}\begin{bmatrix} S_{00} & S_{0b} \\ S_{b0} & S_{bb}\end{bmatrix}\cdot\begin{bmatrix} u_0^{n-1} \\ u_b^{n-1}\end{bmatrix} + (\Delta t)^2\begin{bmatrix} f_0 \\ 0\end{bmatrix} \quad (3.26)$$

In Eq. (3.25), the mass matrix has completed with $M_{bb} = S_{bb}$ information for boundary unknown variables. In the time marching scheme, the M_{bb}^{-1} produces valid result.

The original stabilizer requires that the values in the equation must be the same category. We extend the rule to all the unknown variables in the bilinear form. After

the injection of the new stabilizer, the above equation becomes

$$\begin{bmatrix} M_{00} & 0 \\ 0 & M_{bb} \end{bmatrix} \begin{bmatrix} \ddot{u}_0 \\ \ddot{u}_b \end{bmatrix} + \begin{bmatrix} K_{00} & K_{0b} \\ K_{b0} & K_{bb} \end{bmatrix} \begin{bmatrix} u_0 \\ u_b \end{bmatrix} = \begin{bmatrix} f_0 \\ 0 \end{bmatrix} \tag{3.27}$$

Consequently, both explicit and implicit time marching scheme can be implemented on above matrices.

3.3.3 Parallel Computing Method

For both linear and nonlinear equations, we implement the implicit time marching scheme, the Newmark-Beta method. It has the following steps

$$\ddot{u}_0^{n+1} = \frac{(u_0^{n+1} - \tilde{u}_0)}{\beta \Delta t^2} \tag{3.28}$$

$$\ddot{u}_b^{n+1} = \frac{(u_b^{n+1} - \tilde{u}_b)}{\beta \Delta t^2} \tag{3.29}$$

The $\tilde{u}$ represents the intermediate step, then we can use it to calculate the value of next step

$$\tilde{u}_0 = u_0^n + \dot{u}_0^n \Delta t + \frac{1}{2} \Delta t^2 \ddot{u}_0^n (1 - 2\beta) \tag{3.30}$$

$$\tilde{u}_b = u_b^n + \dot{u}_b^n \Delta t + \frac{1}{2} \Delta t^2 \ddot{u}_b^n (1 - 2\beta) \tag{3.31}$$

To reduce the local unknown variable u_0 into a smaller size global matrix of unknown variable u_b, we apply the Schur complement method. The correlation between interior unknown variables and boundary variables is following equation

$$u_0^{n+1} = -K_{00}^{-1} K_{0b} u_b^{n+1} + K_{00}^{-1} g_1 \tag{3.32}$$

Therefore, the original large sparse global matrix is transformed into a small condensed global matrix with the only one set of unknown variables, u_b. The governing equation has the new form

$$u_b^{n+1} = [K_{b0}(-K_{00}^{-1}K_{0b}) + K_{bb}]^{-1}(g_2 - K_{b0}(K_{00}^{-1}g_1)) \tag{3.33}$$

Then we can simplify the above equation into

$$u_b^{n+1} = K^{'}f^{'} \tag{3.34}$$

From above equations, the boundary unknown variable u_b is updated. In each time step, we convert local unknown variable u_0 to u_b through local Schur complement method to reduce the computational cost. After condensing the matrix, we can calculate the solution by matrix inversion.

3.3.4 The Parallel Computing Work Flow

For one-dimensional elastic equation, the original governing equation can be converted to a sparse tri-diagonal stiffness matrix to a blocked decomposed submatrices

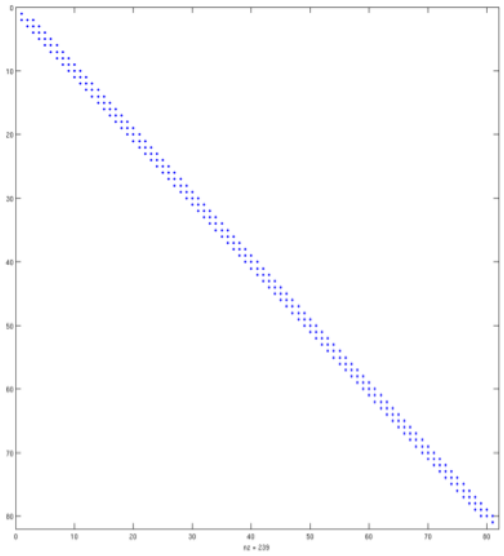


Figure 3.3: Global sparse tri-diagonal stiffness matrix

For the WG element, the global matrix is constructed as above figure. The elements are continuously arranged. The parallel computing scheme such as Schur complement method cannot be directly applied. An external loading vector like Lagrangian multiplier is required. Our hybrid WG-CG element method is discontinuous, it will provide a discontinuous matrix form as below figure.

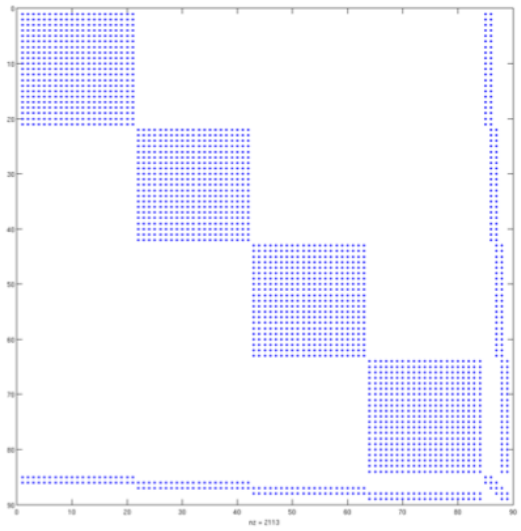


Figure 3.4: Blocked stiffness matrix after domain decomposition

The original large sparse stiffness matrix is compressed in a very small matrix on the right bottom corner by using Schur complement method. In another word, the main computational load is split into the calculation of square matrices. Each square matrix represents the interior unknown variables in one single hybrid element. The size of hybrid element is determined by the number of interior CG elements since the boundary basis function remains the same. The entire computational domain is decomposed into multiple subdomains and each subdomain corresponds to one single processor. After the global calculation, the interior unknown variables can be recovered through local computing.

The parallel computing work flow chart is described as following

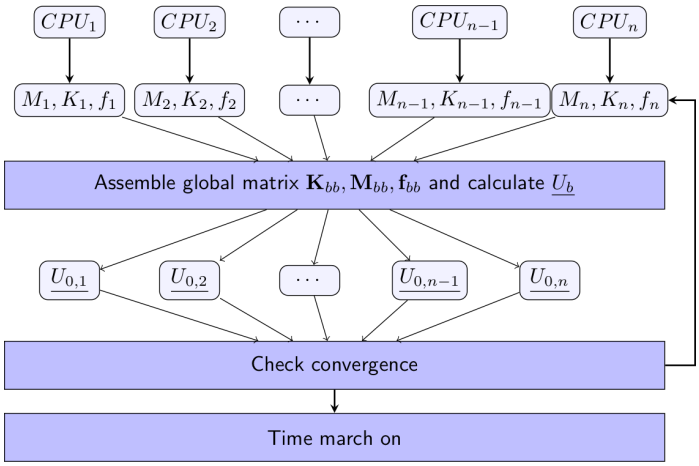


Figure 3.5: Parallel computing workflow chart for WG-CG method

1. Each CPU reads in preprocessed file and initiates MPI communication[25].

2. Every hybrid element is dealt by each individual processor. Local stiffness matrices and load vectors are constructed on parallel machine.

3. The global stiffness boundary matrix is assembled additively. The global boundary unknown variables are obtained and passed to each processor through MPI communication.

4. Once the global residual is less than the tolerant value, current Newmark-Beta time iteration complete and move to the next time step.

To accelerate the performance, we use the open source library such as LAPACK/BLAS to calculate the computationally expensive operators in terms of the block Cholesky elimination, matrix multiplication and vector operations. This implementation largely benefits the programming process and works very well among different high performance computing platforms.

The software has been tested on the George Washington University super computing cluster, ColonialOne, with Xeon E5-2650v2 2.6GHz 8-core processors with 128 GB of RAM each.

3.4 Numerical Results

3.4.1 Geometric linear elastic equation

We design a numerical test to verify the hybrid element with analytical solution $u = sin(2\pi x)$. In each WG element, we set 20 CG elements and then observe the accuracy of u_0, u_b and the effect of stabilizer.

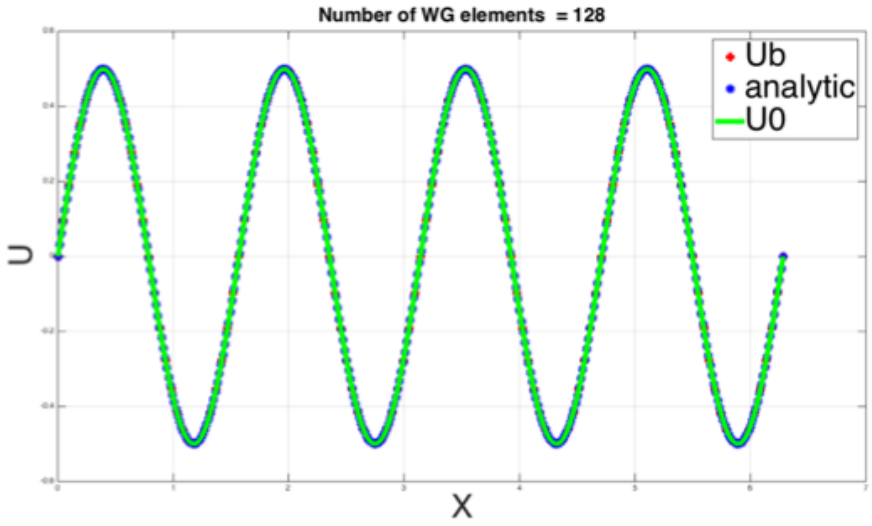


Figure 3.6: Linear elastic equation results for hybrid WG-CG element

The above table demonstrates the accuracy of the hybrid element on linear elastic-

Table 3.1: Error and accuracy of hybrid WG-CG element for linear elasticity.

# CG = 20				
#WG	$E(u_0)$	$O(u_0)$	$E(u_b)$	$O(u_b)$
16	$1.7636e-4$	-	$3.8909e-4$	-
32	$4.4745e-5$	1.98	$9.8515e-5$	1.99
64	$1.1271e-5$	1.99	$2.4744e-5$	1.99
128	$2.8286e-6$	1.99	$6.1941e-6$	2.00

ity equation by using central difference time marching scheme. With the increasing of the number of hybrid WG-CG elements, we observed a second order accuracy of both interior and boundary unknown variables. This test shows that the hybrid element has the excellent compatibility for linear elastic equation.

To explore the capability in dynamic problem using time marching schemes, we extend the test case with the same analytic solution. By implementing the generic stabilizer above, both explicit and implicit time marching scheme are ready to test. For explicit time marching scheme, we choose the forward Euler method [30] and define the relative small time step to fit the critical time marching step [13]. For the implicit scheme, we choose the Newmark-Beta method [26]. For every test case, a constant loading force is applied on the right-hand side as a Neumann boundary condition [49]. Meanwhile, we fix the left-hand side as Dirichlet boundary condition. The results are shown as following charts

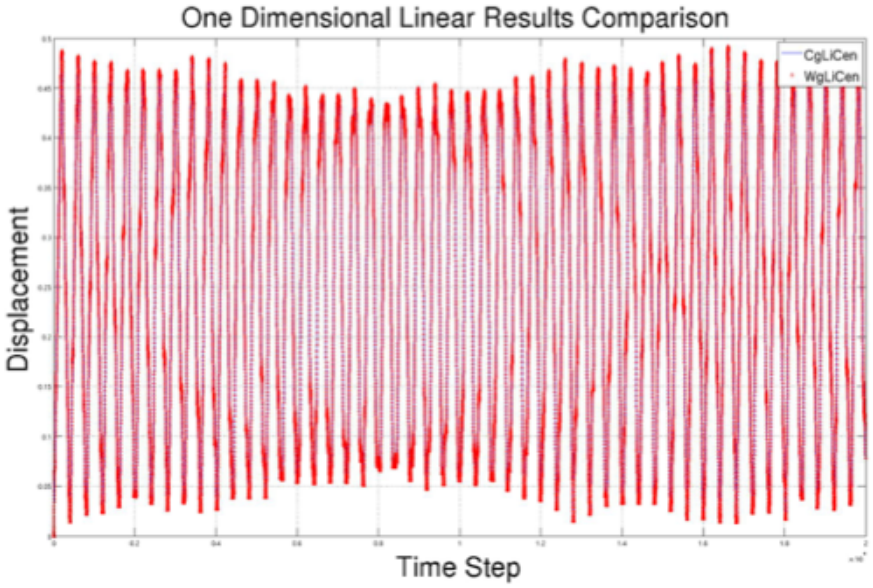


Figure 3.7: The results comparison between CG only and hybrid WG-CG element for explicit scheme

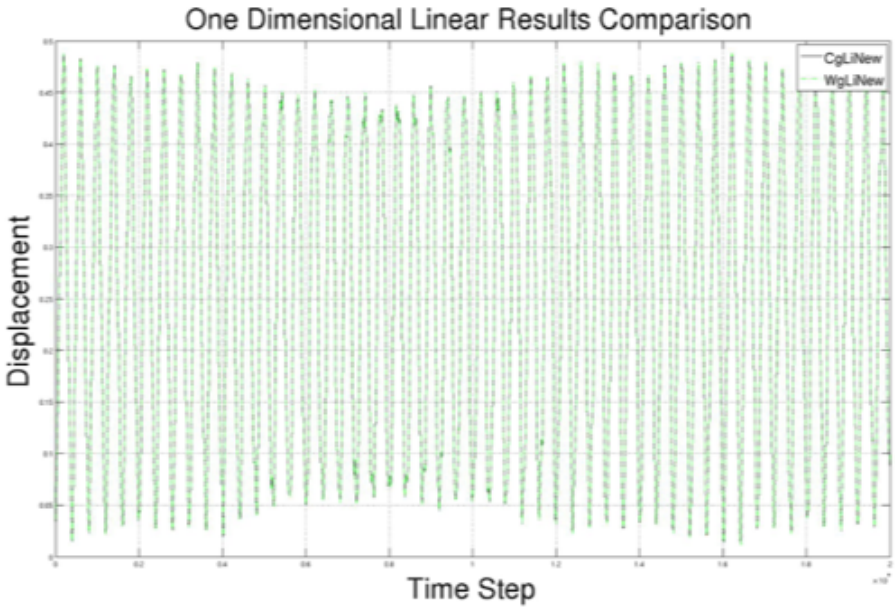


Figure 3.8: The results comparison between CG only and hybrid WG-CG element for implicit scheme

The above two figures illustrates the comparison of deflections. We compare the solutions between our hybrid WG-CG element and classic CG element. In both im-

plicit and explicit time marching scheme, the average difference between two methods is less than 10^{-5}. Based on the results, our hybrid WG-CG solver shows optimal convergence to the established CG solver. The lower error value encouraged us to extend the hybrid element to handle geometric nonlinear problems.

3.4.2 Geometric nonlinear elasticity equation

Both the large deflection and body rotation require the nonlinear strain-displacement relation. We test the performance of accuracy and efficiency of hybrid WG-CG element. We continue using the classic CG element solver as the reference and compare our new method to the reliable existing solutions.

By following the linear elastic equation test cases, we load constant force as the boundary condition and compare the solutions between the serial CG solver and parallel implicit hybrid WG-CG solver. We divide the entire computational domain into 20 subdomains. In another word, each subdomain corresponds to an individual hybrid WG-CG element. In each hybrid element, we insert 20 standard CG elements. Each processor handles one hybrid element and the communicates with each other through MPI library.

The CG solver choose 400 standard linear CG elements to maintain the same resolution. We track the deflection on the right tip of the domain and plot the two sets of solutions in the same figure to compare the difference as following

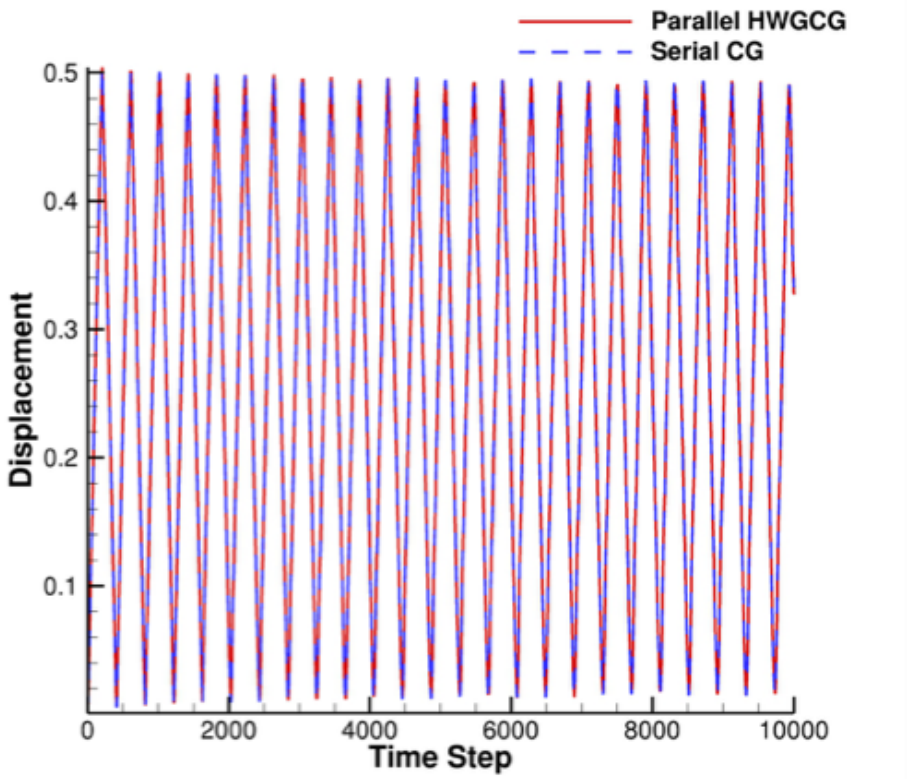


Figure 3.9: The results comparison between CG only and hybrid WG-CG element by using parallel implicit scheme with constant boundary condition

An optimal convergence of the two independent solutions shows an excellent accuracy of the parallel hybrid solver. The average difference is less than 10^{-4}, which is same as the previous linear test.

To verify the the robustness of the new solver, a periodic boundary condition is enforced to substitute the constant variable where $f = sin(2\pi t)$. The comparing results is as following chart

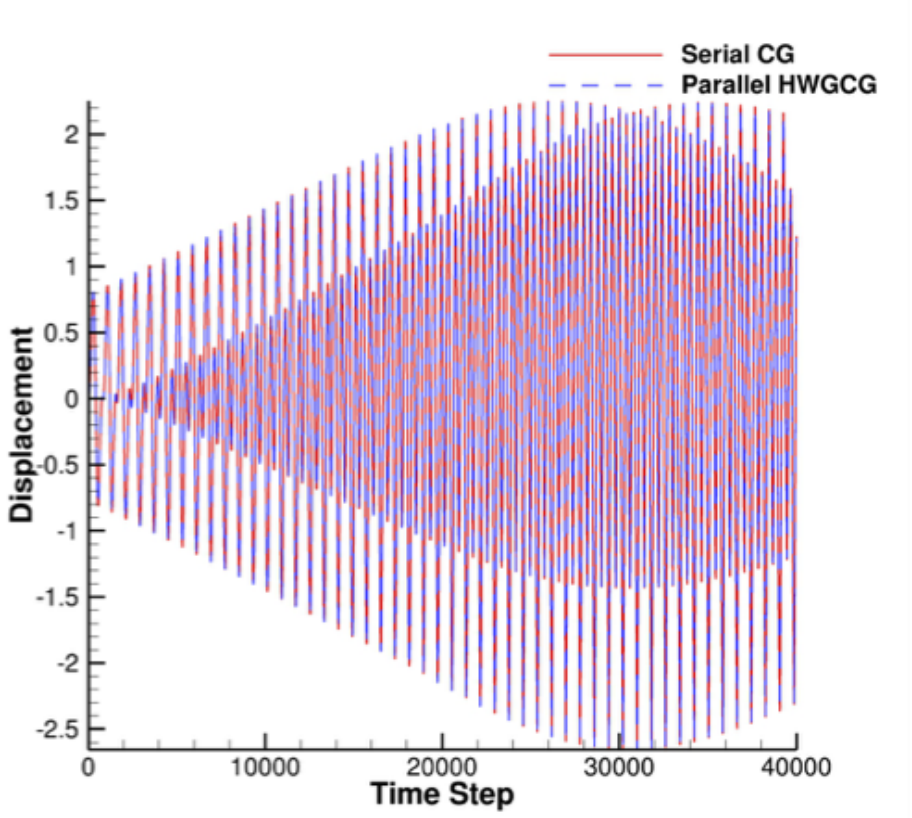


Figure 3.10: The results comparison between CG only and hybrid WG-CG element by using parallel implicit scheme with periodic boundary condition

Analogously, an optimal convergence is also observed from this test case. We have a strong confidence on the accuracy of our nonlinear parallel computing solver.

After the verification of accuracy and precision, we want to test the performance and parallel computing scalability on high performance clusters. We increase the size of the problem to a higher level, 20 times larger than the original test case, up to 100,000 unknown variables. Meanwhile, we gradually increase the number of processor applied for the problem from 10 to 60. The time using graph is plotted as following

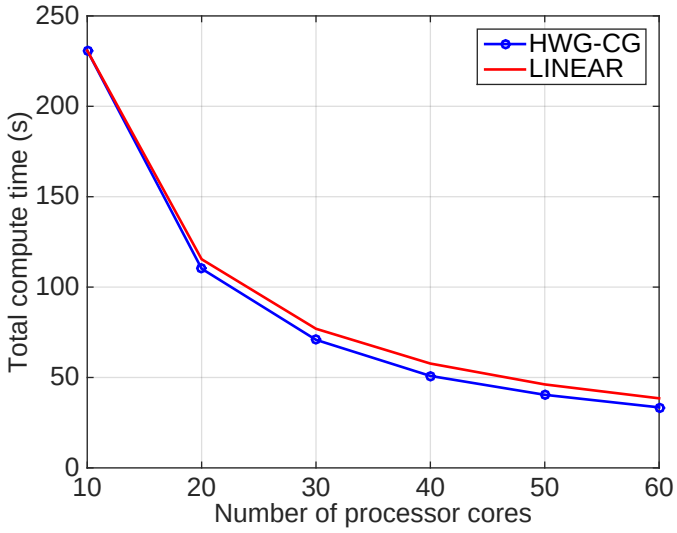


Figure 3.11: Time decreasing .vs. number of processors increasing

then we also plot the speedup curvature is proportional to the increasing number of processors

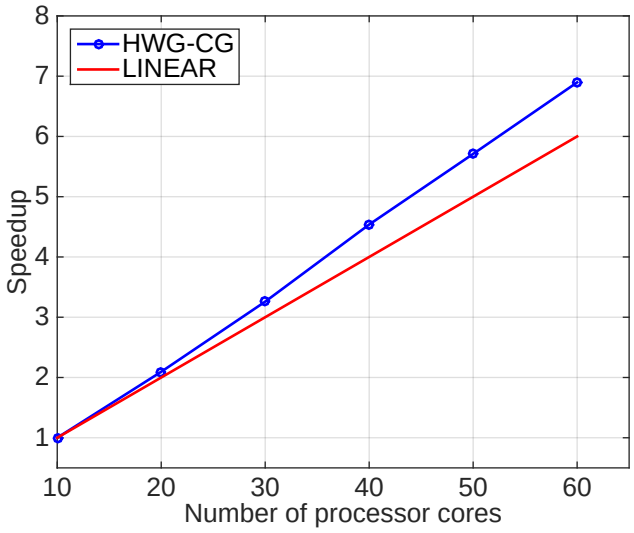


Figure 3.12: Speedup .vs. number of processors increasing

A superlinear speedup is observed from the above test cases. The reason is that the computational effort to solve local matrices decreases faster than communication overhead. The computational cost for stiffness matrix inverion is $O(n^3)$, where n is the size of stiffness matrix. Meanwhile, the communication overhead grows linearly with time complexity $O(n)$. Overall, the trade-off benefit for domain decomposition is larger than MPI communication which leads to the superlinear results.

3.5 Summary

In summary, we present a newly developed hybrid element combining the weak Galerkin finite element and continuous Galerkin finite element. It inherits the discontinuous feature from WG method and the computational simplicity from CG method. The implementation of hybrid element for parallel computing is through Schur complement method. The second order of accuracy and superlinear speedup are observed for the hybrid element.

We test the hybrid WG-CG element for linear and nonlinear elasticity equations. We implement explicit (Central Difference) and implicit (Newmark-Beta) methods to obtain the results. Comparing with analytical solution and published results, the error is lower than 10^{-5} for both Dirichlet and Neumann boundary condition.

After the one dimensional examples, we present the test cases of the WG method for solving two dimensional problems.We obtain expected order of accuracy. The next step is to implement parallel computing scheme on it. The details of parallel computing for larger problem will be discussed in the next chapter.

Chapter 4: Weak Galerkin Parallel Solutions of Linear Elasticity on Unstructured Meshes

This chapter focuses on solving linear elasticity problem on parallel computer by combining a novel finite element method with an efficient parallel computing scheme. Specifically, this combination involves a discontinuous weak Galerkin finite element method [44, 35, 58, 42] and a non-overlapping domain decomposition scheme, namely, the balancing domain decomposition with constraints (BDDC) [16, 54, 53]. The WG method is considered as a newly developed robust numerical method and inherits the locking-free feature for the linear elastic equation [56].

Like the standard Finite Element method (FEM), the WG method can be used to solve generic partial differential equations. An advanced feature of the WG finite element method is that the entire problem is decomposed into multiple elemental problems. Such elemental problems are often derived from weak formulations of the corresponding differential operators after integration by parts. In these elemental problems, the differential operators are approximated and reconstructed by smaller-size matrices. The WG method has been proven robust and possessing optimal orders of accuracy in spatial discretization on serial computers [45, 46]. Wang et al [56] recently extended the WG method to solve linear elasticity problems and also successfully demonstrated its locking-free property. However, the performance of the WG method on parallel computers has not yet been examined. In this chapter, we design and develop the WG-BDDC scheme to bring the WG method to solve engineering problem in parallel fashion. This is the first attempt in our knowledge. The successful results indicate that it's a promising direction to develop an efficient and parallel structural solver.

4.1 Domain Decomposition Scheme

The basic idea of domain decomposition is to split the computational mesh of an entire domain into many smaller meshes for a set of non-overlapping subdomains. Each subdomain contains its own set of grid elements. For finite element methods, after domain decomposition, a remaining challenging task is to connect these subdomains' interfaces by satisfying continuity constraints to correctly represent the solution of the original set of equations over the complete domain. In this work, the BDDC method is used to serve this purpose. The original balancing domain decomposition (BDD) method [40] has only considered two level meshes. It used a multiplicative coarse domain to correct the local fine mesh subdomain. However, the significant difference between BDDC and BDD is that the method in this paper applies the coarse problem in an additive routine rather than multiplicative manner. In this case, a more flexible of constraints will reduce the complexity and improve the efficiency. In our BDDC method, we assemble the preconditioner matrix additively in contrast to the multiplicative coarse grid correction used in the BDD method. In the BDDC method, the flexibility of choosing constraining points leads to reduced complexity of implementation and improved efficiency of computations in comparison to the standard BDD method. The details of the choice of constraints for BDDC will be discussed in this section.

4.1.1 FETI-DP Method

Finite Element Tearing and Interconnect (FETI) , first proposed by Farhat [23, 22, 31, 21, 34, 32], is a popular method for solving large finite element problems. Originally, FETI is proposed to solve the discontinuity when applying domain decomposition on second order elliptic partial differential equations, particularly elasticity equations. FETI method partitioned the entire computational domain into two level meshes, the coarse grid and fine grid. Lagrangian multiplier is employed to resolve the

singularity for the matrix of every subdomain after the domain decomposition. For the fine mesh, since local matrix is not positive definite, the pseudo-inverse is applied to invert the local matrix. Combining the Lagrangian multiplier and pseudo-inverse, FETI method bypass the rigid body motion problem and partitioned the continuous problem to multiple non-overlapping subdomains.

Then FETI-DP (Dual Primal) is introduced for two-dimensional problems by Farhat and Rixen [20]. In this method, the unknown variables on the interface is partitioned into primal and dual spaces. The continuity of the primal unknown variables along the interface, the vertices of each subdomain, is maintained by assemblage. For the dual space, the constraints are represented by Lagrange multipliers.

In this section, we begin with an example of a two-dimensional, two partitioned subdomain case. The unknown variables $\mathbf{u}$ are represented by the solid squares on the interface. We enforce the Dirichlet boundary condition on the boundary. The original geometry and computational domain is

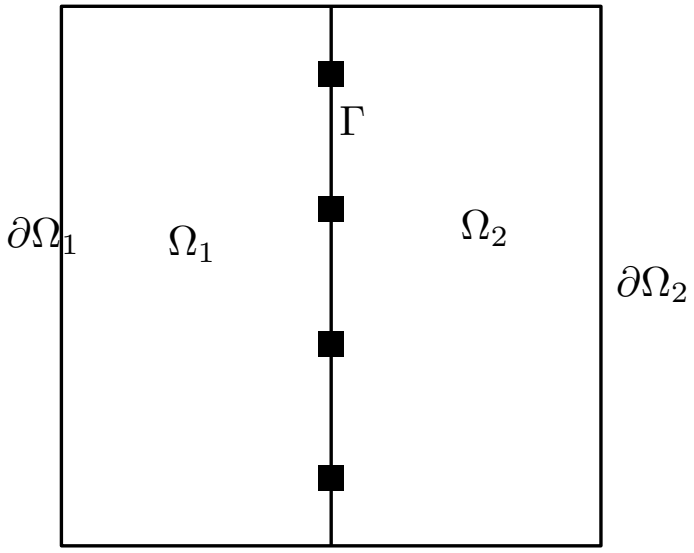


Figure 4.1: Computational domain partitioned into two nonoverlapping subdomains.

the original governing equation in matrix form is $\mathbf{Au} = \mathbf{f}$. We begin to compute

the stiffness matrix for each subdomain that

$$\begin{pmatrix} A_{II}^{(j)} & A_{I\Gamma}^{(j)} \\ A_{\Gamma I}^{(j)} & A_{\Gamma\Gamma}^{(j)} \end{pmatrix} \begin{pmatrix} u_I^{(j)} \\ u_\Gamma^{(j)} \end{pmatrix} = \begin{pmatrix} f_I^{(j)} \\ f_\Gamma^{(j)} \end{pmatrix} \tag{4.1}$$

where $j = 1, 2$.

The boundary condition along $\partial\Omega$ is Dirichlet condition, a homogeneous Neumann condition is applied on Γ.

The global problem after assemblage is that

$$\begin{pmatrix} A_{II}^{(1)} & 0 & A_{I\Gamma}^{(1)} \\ 0 & A_{II}^{(2)} & A_{I\Gamma}^{(2)} \\ A_{\Gamma I}^{(1)} & A_{\Gamma I}^{(2)} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} u_I^{(1)} \\ u_I^{(2)} \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} f_I^{(1)} \\ f_I^{(2)} \\ f_\Gamma \end{pmatrix} \tag{4.2}$$

where $A_\Gamma = A_{\Gamma\Gamma}^{(1)} + A_{\Gamma\Gamma}^{(2)}$ and $f_\Gamma = f_\Gamma^{(1)} + f_\Gamma^{(2)}$. The unknown variables are decomposed into two computational subdomains. The degree of freedoms are represented by $u_I^{(1)}$, u_I^2 and u_Γ corresponding to the domains Ω_1, Ω_2 and Γ respectively.

Since the interior matrices are square and can be inverted. Then we can eliminate the interior unknown variables by Schur complement operators. The interface unknown DOFs shall have the new form as

$$S^{(j)} := A_{\Gamma\Gamma}^{(j)} - A_{\Gamma I}^{(j)} {A_{II}^{(j)}}^{-1} A_{I\Gamma}^{(j)} \tag{4.3}$$

$$g_\Gamma^{(j)} := f_\Gamma^{(j)} - A_{\Gamma I}^{(j)} {A_{II}^{(i)}}^{-1} f_I^{(j)} \tag{4.4}$$

based on the given matrix $Au = f$, we can reduce the equation matrices into interface DOF only system

$$(S^{(1)} + S^{(2)})u_\Gamma = g_\Gamma^{(1)} + g_\Gamma^2 \tag{4.5}$$

After the Schur complement, the original global matrix A is reduced the unknown variables on the interface only g_Γ. The major cost of local matrix inversion has been distributed to each local processors. The local unknown variables can be recovered locally with the solution of interface.

For FETI method, we introduce the Lagrange multiplier to enforce the continuity along the interface. The governing equation in matrix form is

$$\begin{pmatrix} A_{II}^{(j)} & A_{I\Gamma}^{(j)} \\ A_{\Gamma I}^{(j)} & A_{\Gamma\Gamma}^{(j)} \end{pmatrix} \begin{pmatrix} u_I^{(j)} \\ u_\Gamma^{(j)} \end{pmatrix} = \begin{pmatrix} f_I^{(j)} \\ f_\Gamma^{(j)} + \lambda_\Gamma^{(j)} \end{pmatrix} \tag{4.6}$$

where $j = 1, 2$ and $\lambda_\Gamma = \lambda_\Gamma^{(1)} = -\lambda_\Gamma^{(2)}$. λ_Γ is an unknown flux vector, namely, Lagrange multiplier. We can solve this linear system following Eq. (4.3) to Eq. (4.5)

$$g_\Gamma^{(j)} = f_\Gamma^{(j)} - A_{\Gamma I}^{(j)} {A_{II}^{(j)}}^{-1} f_I^{(j)} \tag{4.7}$$

$$u_\Gamma^{(j)} = {S^{(j)}}^{-1} (g_\Gamma^{(j)} + \lambda_\Gamma^{(j)}) \tag{4.8}$$

The interface condition between adjacent subdomains is

$$u_\Gamma^{(1)} = u_\Gamma^{(2)} \tag{4.9}$$

the value of DOFs for the same position has to be same. To resolve the rigid body motion problem, we obtain

$$F\lambda_\Gamma = d_\Gamma \tag{4.10}$$

$$F = {S^{(1)}}^{-1} + {S^{(2)}}^{-1} \tag{4.11}$$

An improved version of FETI is FETI-DP. We further divide the interface to

primal and dual spaces,

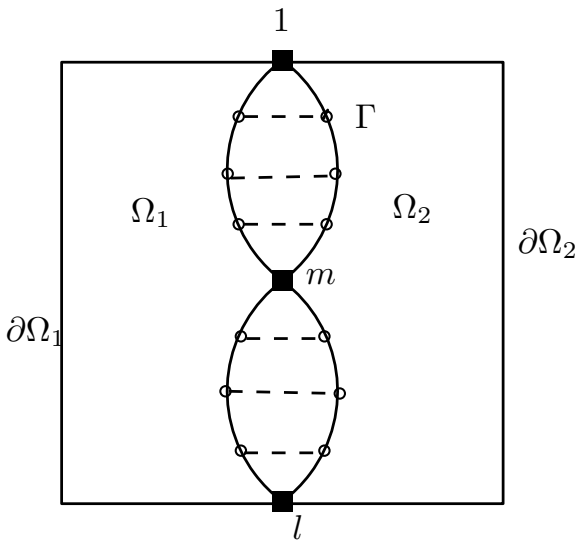


Figure 4.2: Computational domain partitioned into two nonoverlapping subdomains with floating constaint.

From the Fig 4.2, the interface edge is continued to split into $(u_1, \cdots, u_m, \cdots, u+l)$. The solid squares represent the unknown variables belong to primal space, the hollow dots represent the unknown variables belong to the dual space. The original interface is partially teared. The curve and dots mean the new local unknown variables after the tearing. The connecting squares represent the reconnecting. This graph shows the origin of the name of FETI-DP. The linear system can be written as

$$\begin{pmatrix} A_{II}^{(j)} & A_{1I}^{(j)} & \cdots & A_{mI}^{(j)} & \cdots & A_{lI}^{(j)} \\ A_{1I}^{(j)} & A_{11}^{(j)} & \cdots & A_{1m}^{(j)} & \cdots & A_{1l}^{(j)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ A_{mI}^{(j)} & A_{m1}^{(j)} & \cdots & A_{mm}^{(j)} & \cdots & A_{ml}^{(j)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ A_{lI}^{(j)} & A_{l1}^{(j)} & \cdots & A_{lm}^{(j)} & \cdots & A_{ll}^{(j)} \end{pmatrix} \begin{pmatrix} u_I^{(j)} \\ u_1^{(j)} \\ \vdots \\ u_m^{(j)} \\ \vdots \\ u_l^{(j)} \end{pmatrix} = \begin{pmatrix} f_I^{(j)} \\ f_1^{(j)} \\ \vdots \\ f_m^{(j)} \\ \vdots \\ f_l^{(j)} \end{pmatrix} \tag{4.12}$$

The variables along the interface between the two nonoverlapping subdomians can

be written as

$$\begin{pmatrix} u_1^{(j)} \\ \vdots \\ u_m^{(j)} \\ \vdots \\ u_l^{(j)} \end{pmatrix} = T_E \begin{pmatrix} \hat{u}_1^{(j)} \\ \vdots \\ \hat{u}_m^{(j)} \\ \vdots \\ \hat{u}_l^{(j)} \end{pmatrix} = \begin{pmatrix} 1 & \cdots & 1 & 0 & \cdots \\ \vdots & \ddots & \vdots & 0 & \cdots \\ -1 & \cdots & 1 & \cdots & -1 \\ \cdots & 0 & \vdots & \ddots & \vdots \\ \cdots & 0 & 1 & \cdots & 1 \end{pmatrix} \begin{pmatrix} \hat{u}_1^{(j)} \\ \vdots \\ \hat{u}_m^{(j)} \\ \vdots \\ \hat{u}_l^{(j)} \end{pmatrix} \tag{4.13}$$

After the multiplication between vector and matrix we have

$$T_E \begin{pmatrix} \hat{u}_1^{(j)} \\ \vdots \\ \hat{u}_m^{(j)} \\ \vdots \\ \hat{u}_l^{(j)} \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{pmatrix} \hat{u}_m^{(j)} + \begin{pmatrix} & \hat{u}_1^{(j)} & & & \\ & \vdots & & & \\ -\hat{u}_1^{(j)} & -\cdots & -\hat{u}_{m-1}^{(j)} & \hat{u}_{m+1}^{(j)} & -\hat{u}_l^{(j)} \\ & \vdots & & & \\ & \hat{u}_l^{(j)} & & & \end{pmatrix} \tag{4.14}$$

the T_E is a square matrix and each columns represents the new space of interface unknown variables. The original interface is divided into two spaces. One is the local space and another is the interface space. The the problem can be written as

$$T^T \begin{pmatrix} A_{II}^{(j)} & A_{1I}^{(j)} & \cdots & A_{mI}^{(j)} & \cdots & A_{lI}^{(j)} \\ A_{1I}^{(j)} & A_{11}^{(j)} & \cdots & A_{1m}^{(j)} & \cdots & A_{1l}^{(j)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ A_{mI}^{(j)} & A_{m1}^{(j)} & \cdots & A_{mm}^{(j)} & \cdots & A_{ml}^{(j)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ A_{lI}^{(j)} & A_{l1}^{(j)} & \cdots & A_{lm}^{(j)} & \cdots & A_{ll}^{(j)} \end{pmatrix} T \begin{pmatrix} u_I^{(j)} \\ \hat{u}_1^{(j)} \\ \vdots \\ \hat{u}_m^{(j)} \\ \vdots \\ \hat{u}_l^{(j)} \end{pmatrix} = T^T \begin{pmatrix} f_I^{(j)} \\ f_1^{(j)} \\ \vdots \\ f_m^{(j)} \\ \vdots \\ f_l^{(j)} \end{pmatrix} \tag{4.15}$$

same as previous equation, T is a diagonal block matrix

$$T = \begin{pmatrix} I & 0 \\ 0 & T_E \end{pmatrix} \tag{4.16}$$

We can assume that the unknown variables on each subdomain have been changed when using primal space. We can see that the primal space is the only connection along the interface. The other DOFs are in dual space including interior unknowns and the rest interface nodes.

$$\begin{pmatrix} A_{II}^{(1)} & A_{\Delta I}^{(1)} & 0 & 0 & A_{\Pi I}^{(1)} & 0 \\ A_{\Delta I}^{(1)} & A_{\Delta\Delta}^{(1)} & 0 & 0 & A_{\Pi\Delta}^{(1)} & B_{\Delta}^{(1)} \\ 0 & 0 & A_{II}^{(2)} & A_{\Delta I}^{(2)} & A_{\Pi I}^{(2)} & 0 \\ 0 & 0 & A_{\Delta I}^{(2)} & A_{\Delta\Delta}^{(2)} & A_{\Pi\Delta}^{(2)} & B_{\Delta}^{(2)} \\ A_{\Pi I}^{(1)} & A_{\Pi\Delta}^{(1)} & A_{\Pi I}^{(2)} & A_{\Pi\Delta}^{(2)} & A_{\Pi\Pi}^{(1)} + A_{\Pi\Pi}^{(2)} & 0 \\ 0 & B_{\Delta}^{(1)} & 0 & B_{\Delta}^{(2)} & 0 & 0 \end{pmatrix} \begin{pmatrix} u_I^{(1)} \\ u_{\Delta}^{(1)} \\ u_I^{(2)} \\ u_{\Delta}^{(2)} \\ u_{\Pi} \\ \lambda \end{pmatrix} = \begin{pmatrix} f_I^{(1)} \\ f_{\Delta}^{(1)} \\ f_I^{(2)} \\ f_{\Delta}^{(2)} \\ f_{\Pi}^{(1)} + f_{\Pi}^{(2)} \\ 0 \end{pmatrix} \tag{4.17}$$

We can further eliminate the local variables $u_I^{(1)}, u_{\Delta}^{(1)}, u_I^{(2)}$ and $u_{\Delta}^{(2)}$, so that we can obtain

$$\begin{pmatrix} S_{\Pi\Pi} & \tilde{B}_{\Lambda\Pi}^T \\ \tilde{B}_{\Lambda\Pi} & \tilde{B}_{\Lambda\Lambda} \end{pmatrix} \begin{pmatrix} u_{\Pi} \\ \lambda \end{pmatrix} = \begin{pmatrix} g_{\Pi} \\ d_{\Lambda} \end{pmatrix} \tag{4.18}$$

the details of above equation is

$$S_{\Pi\Pi} = \sum_{j=1}^{2} R_{\Pi}^{(j)} \left(A_{\Pi\Pi}^{(j)} - \begin{bmatrix} A_{\Pi I}^{(j)} & A_{\Pi\Delta}^{(j)} \end{bmatrix} \begin{bmatrix} A_{II}^{(j)} & A_{I\Delta}^{(j)} \\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)} \end{bmatrix}^{-1} \begin{bmatrix} A_{\Pi I}^{(j)} \\ A_{\Pi\Delta}^{(j)} \end{bmatrix} \right) R_{\Pi}^{(j)}, \tag{4.19}$$

$$\tilde{B}_{\Lambda\Pi} = -\sum_{j=1}^{2} \begin{bmatrix} 0 & B_{\Pi}^{(j)} \end{bmatrix} \begin{bmatrix} A_{II}^{(j)} & A_{\Delta I}^{(j)} \\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)} \end{bmatrix}^{-1} \begin{bmatrix} A_{\Pi I}^{(j)} \\ A_{\Pi\Delta}^{(j)} \end{bmatrix} R_{\Pi}^{(j)}, \tag{4.20}$$

$$\tilde{B}_{\Lambda\Lambda} = -\sum_{j=1}^{2} \begin{bmatrix} 0 & B_{\Delta}^{(j)} \end{bmatrix} \begin{bmatrix} A_{II}^{(j)} & A_{\Delta I}^{(j)} \\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ B_{\Delta}^{(j)} \end{bmatrix}, \tag{4.21}$$

$$g_{\Pi} = \sum_{j=1}^{2} R_{\Pi}^{(j)} \left(f_{\Pi}^{(j)} - \begin{bmatrix} A_{\Pi I}^{(j)} & A_{\Pi\Delta}^{(j)} \end{bmatrix} \begin{bmatrix} A_{II}^{(j)} & A_{\Pi\Delta}^{(j)} \end{bmatrix} \begin{bmatrix} A_{II}^{(j)} & A_{\Delta I}^{(j)} \\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)} \end{bmatrix}^{-1} \begin{bmatrix} f_{I}^{(j)} \\ f_{\Delta}^{(j)} \end{bmatrix} \right), \tag{4.22}$$

$$d_{\Lambda} = -\sum_{j=1}^{2} \begin{bmatrix} 0 & B_{\Delta}^{(j)} \end{bmatrix} \begin{bmatrix} A_{II}^{(j)} & A_{\Delta I}^{(j)} \\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)} \end{bmatrix}^{-1} \begin{bmatrix} f_{I}^{(j)} \\ f_{\Delta}^{(j)} \end{bmatrix} \tag{4.23}$$

whre the $R_{\Pi}^{(j)}$ is a projection matrix that mapping the local subdomain element to global elements with $\{0, 1\}$ values. In this example, both projection values are equal to 1.

For the linear system, the Lagrange multiplier λ is

$$\left(\tilde{B}_{\Lambda\Lambda} - \tilde{B}_{\Lambda\Pi} S_{\Pi\Pi}^{-1} \tilde{B}_{\Lambda\Pi}^{T} \right) \lambda = d_{\Lambda} - \tilde{B}_{\Lambda\Pi} S_{\Pi\Pi}^{-1} g_{\Pi} \tag{4.24}$$

A Dirichlet preconditioner is applied in FETI-DP method for solving above equation.

Based on above linear system, Lagrange multiplier is an external unknown vector λ and requires lots of computational effort to assemble the matrix. When we try to solve the such a linear system, it's common to obtain the result along the dual space that $u_{\Delta}^{(1)} \neq u_{\Delta}^{(2)}$. To maintain the continuity, we shall balance the computational cost for solving the unknown vectors. Therefore, an assembled residual of resulting vector is needed. The residue is mapped into the appropriate space of enforced vector on

the right-hand side of the equation. Then we can use the residue vector to correct the solution and gradually obtain the final results. An improved algorithm, BDDC method, will be discussed in the following section.

4.1.2 Block Cholesky elimination

First, we consider how to represent the inverse of a matrix M, which is symmetric and positive definite block matrix as

$$M = \begin{bmatrix} A & B^T \\ B & C \end{bmatrix} \tag{4.25}$$

where M is the global stiffness matrix, A is the square matrix represents the interior values of each subdomain, C is the values on the interface and B represents the connectivity between A and C.

We employ the block Cholesky elimination and obtain the following equation that

$$\begin{bmatrix} A & B^T \\ B & C \end{bmatrix} = \begin{bmatrix} I_A & 0 \\ BA^{-1} & I_C \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & C - BA^{-1}B^T \end{bmatrix} \begin{bmatrix} I_A & A^{-1}B^T \\ 0 & I_C \end{bmatrix} \tag{4.26}$$

Both I_A and I_C are identity matrices. The Schur complement is represented as

$$S = C - BA^{-1}B^T \tag{4.27}$$

the S can be seen as a smaller global matrix containing the information among A, B and C. A Schur complement matrix S , represents a smaller global stiffness matrix partially assembled by each individual subdomains. However, after the decomposition, the size of S is still too large to be directly inversed. Consequently, we introduce the primal and dual spaces to calculate a preconditioner in parallel fashion to distribute the computational load and accelerate the overall speed. We redefine the Schur complement matrix S and compute the preconditioner S_{BDDC} by a parallel

fashion. The number of primal variables, the constraints through the interface, is much less than the size of Schur complement matrix.

The inverse of original matrix M is

$$M^{-1} = \begin{bmatrix} A & B^T \\ B & C \end{bmatrix}^{-1} = \begin{bmatrix} I_A & -A^{-1}B^T \\ 0 & I_C \end{bmatrix} \begin{bmatrix} A^{-1} & 0 \\ 0 & S^{-1} \end{bmatrix} \begin{bmatrix} I_A & 0 \\ -BA^{-1} & I_C \end{bmatrix} \quad (4.28)$$

$$= \begin{bmatrix} A^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \Phi S^{-1} \Phi^T \quad (4.29)$$

where $\Phi = \begin{bmatrix} -A^{-1}B^T \\ I_C \end{bmatrix}$ where I_A and I_C is diagonal matrix with same size as matrix A and C respectively.

The original inverse global matrix is partitioned into one matrix containing A^{-1} and the other matrix associated with the S^{-1}. The interface matrix S takes from adjacent local subdomains, corresponding to the degrees of freedom on the solid squares shown in Figure 4.2.

After decomposing the original computational domain into subdomains, we can demonstrate the effectiveness of the BDDC method in solving the governing equation

$$M\mathbf{u} = \mathbf{f} \quad (4.30)$$

4.1.3 Balancing Domain Decomposition by Constraints

The balancing domain decomposition method was introduced by Mandel [41]. The original idea of BDD method is applying a coarse correction to guarantee the convergence of residuals. The BDDC is a domain decomposition method for solving large symmetric, positive definite equations of linear systems. Comparing to BDD method, the substructure spaces and the coarse spaces are connected by the corner cell as constraints only. The main difference is that the BDDC method applies the

coarse problem in an additive routine, which makes it possible to use a different bilinear form on the coarse problem. In this way, the BDDC method is considered as a simpler alternative to FETI-DP domain decomposition method [34]. We only consider the corner connections of subdomain as the only constraints. Comparing with FETI-DP, BDDC method adds coarse degrees of freedom involving averages over edges and faces of elements. This improvement causes an obvious simplification through domain decomposition and matrix calculation.

After the Schur complement, the preconditioner for interface is

$$\hat{S}u_\Gamma = \sum_{j=1}^{N} R_\Gamma^{(j)} g_\Gamma \tag{4.31}$$

where

$$\tilde{S} = \tilde{R}_\Gamma^T \tilde{S}_\Gamma \tilde{R}_\Gamma \tag{4.32}$$

In the BDDC algorithm, a two-level Neumann-Neumann type preconditioner for solving this interface problem. In the BDDC preconditioner, the coarse grid is assembled from coarse basis functions. We can apply the minimum energy method on the subdomains to obtain primal constrains. The primal constraints maintain the continuity along the edge interface between two subdomains, as in FETI-DP algorithm.

Dohrmann's BDDC preconditioner [16, 17, 41] has the form

$$M_{BDDC}^{-1} = R_{D,\Gamma}^T T R_{D,\Gamma} \tag{4.33}$$

for the coarse-level T is defined by

$$T = \Psi(\Psi^T S \Psi)^{-1} \Psi^T \tag{4.34}$$

the coarse level basis function vector is defined by

$$\Psi = \begin{pmatrix} \Psi^{(1)} \\ \vdots \\ \Psi^{(N)} \end{pmatrix} \tag{4.35}$$

Then the Schur complement coarse-level matrix can be written as

$$\begin{pmatrix} S^{(j)} & C^{(j)^T} \\ C^{(j)} & 0 \end{pmatrix} \begin{pmatrix} \Psi^{(j)} \\ V^{(j)} \end{pmatrix} = \begin{pmatrix} 0 \\ R_{\Pi}^{(j)} \end{pmatrix} \tag{4.36}$$

where $C^{(j)}$ is the primal constraints of each subdomain and $V^{(j)}$ is Lagrange multiplier vector. Since the variables are changing for each subdomain, then the Lagrange multiplier vector is no longer needed to enforce the primal continuity constraints and the new BDDC preconditioner can be designed as

$$M_{BDDC}^{-1} = R_{D,\Gamma}^T \{ R_{\Gamma\Delta}^T S_{\Delta}^{-1} R_{\Gamma\Delta} + \Psi (\Psi^T S \Psi)^{-1} \Psi^T \} R_{D,\Gamma} \tag{4.37}$$

The primal space DOFs are used to enforce the continuity by restricting the operators to the dual interface space Δ. The governing matrix equation can be designed as

$$\begin{pmatrix} A_{II}^{(j)} & A_{\Delta I}^{(j)} & A_{\Pi I}^{(j)} \\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)} & A_{\Pi\Delta}^{(j)} \\ A_{\Pi I}^{(j)} & A_{\Pi\Delta}^{(j)} & A_{\Pi\Pi}^{(j)} \end{pmatrix} \begin{pmatrix} u_I^{(j)} \\ \Psi_{\Delta}^{(j)} \\ R_{\Pi}^{(j)} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ S_{\Pi\Pi}^{(j)} R_{\Pi}^{(j)} \end{pmatrix} \tag{4.38}$$

where

$$\Psi^{(j)} = \begin{pmatrix} \Psi_{\Delta}^{(j)} \\ R_{\Pi}^{(j)} \end{pmatrix} \tag{4.39}$$

$$\Psi^{(j)} = \begin{pmatrix} -\begin{pmatrix} 0 & I_{\Delta}^{(j)} \end{pmatrix} \begin{pmatrix} A_{II}^{(j)} & A_{I\Delta}^{(j)} \\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)} \end{pmatrix}^{-1} \begin{pmatrix} A_{I\Pi}^{(j)} \\ A_{\Pi\Delta}^{(j)} \end{pmatrix} R_{\Pi}^{(j)} \\ R_{\Pi}^{(j)} \end{pmatrix} \tag{4.40}$$

and

$$S_{\Pi\Pi}^{(j)} = A_{\Pi\Pi}^{(j)} - \begin{pmatrix} A_{\Pi I}^{(j)} & A_{\Pi\Delta}^{(j)} \end{pmatrix} \begin{pmatrix} A_{II}^{(j)} & A_{I\Delta}^{(j)} \\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)} \end{pmatrix}^{-1} \begin{pmatrix} A_{I\Pi}^{(j)} \\ A_{\Delta\Pi}^{(j)} \end{pmatrix} \tag{4.41}$$

the global Schur complement operator $S_{\Pi\Pi}$ is assembled by every subdomain that

$$\Psi^T S \Psi = \sum_{j=1}^{N} {\Psi^{(j)}}^T S^{(j)} \Psi^{(j)} \tag{4.42}$$

this equals to

$$\sum_{j=1}^{N} {R_{\Pi}^{(j)}}^T S_{\Pi\Pi}^{(j)} R_{\Pi}^{(j)} = S_{\Pi\Pi} \tag{4.43}$$

where Ψ represents the interface vectors with distributed coarse-level variables. Φ represents the unknown variable vectors on coarse-level.

The averaging scaling vector is

$$\delta_j^{\dagger}(x) = 1, \quad x \in \Gamma \tag{4.44}$$

$$R_{D,\Gamma}^T \Psi = \tilde{R}_{D,\Gamma}^T \Phi \tag{4.45}$$

therefore

$$M_{BDDC}^{-1} = R_{D,\Gamma}^T R_{\Gamma\Delta}^T S_{\Delta}^{-1} R_{\Gamma\Delta} R_{D,\Gamma} + R_{D,\Gamma}^T \Psi \left(\Psi^T S \Psi \right)^{-1} \Psi^T R_{D,\Gamma} \tag{4.46}$$

after replacing Ψ by Φ, we can derive

$$M_{BDDC}^{-1} = \tilde{R}_{D,\Gamma}^T R_{\Gamma\Delta}^T S_{\Delta}^{-1} R_{\Gamma\Delta} \tilde{R}_{\Delta,\Gamma} + \tilde{R}_{D,\Gamma}^T \Phi S_{\Pi\Pi}^{-1} \Phi^T \tilde{R}_{D,\Gamma} \tag{4.47}$$

we can simplify the equation by

$$M_{BDDC}^{-1} = \tilde{R}_{D,\Gamma}^T \tilde{S}_{\Gamma}^{-1} \tilde{R}_{D,\Gamma} \tag{4.48}$$

the preconditioned BDDC operator is designed by

$$\tilde{R}_{D,\Gamma}^T \tilde{S}_{\Gamma}^{-1} \tilde{R}_{D,\Gamma} \tilde{R}_{\Gamma}^T \tilde{S}_{\Gamma} \tilde{R}_{\Gamma} \tag{4.49}$$

4.2 WG-BDDC Method

In this section, we discuss the details of combining WG method with BDDC method for solving elasticity equations. The preconditioned conjugate gradient method is adopted as the linear solver for BDDC method. The construction of preconditioner is crucial in the problem. The BDDC preconditioner combines the solution of the local problem on each subdomain with the solution of a global coarse problem and the coarse degrees of freedoms as unknowns.

The preconditioned conjugate gradient method is adopted as the linear solver for BDDC method. The construction of preconditioner is crucial in the problem. The BDDC preconditioner combines the solution of the local problem on each subdomain with the solution of a global coarse problem and the coarse degrees of freedoms as unknowns.

In FETI method, local matrices after domain decomposition are singular and the pseudo-inverses must be computed. On the contrary, the WG-BDDC has the advantage to bypass this difficulty.

BDDC shall be processed by the following steps:

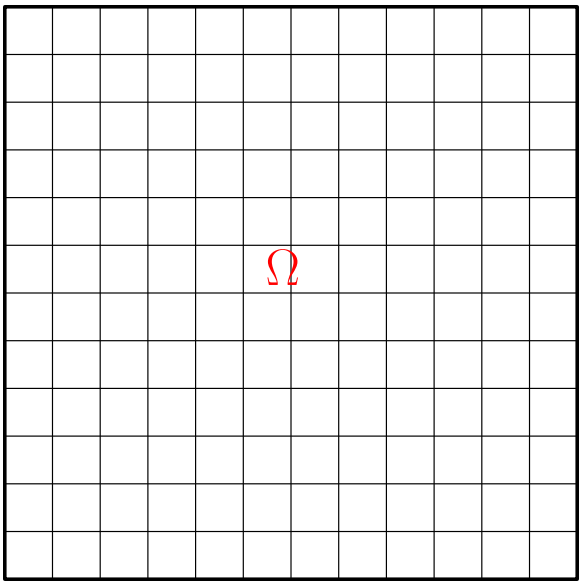


Figure 4.3: The total computational domain.

1. Compute Schur complement matrix S.

2. Further divide the unknowns of the interface into primal and dual subspaces and construct the preconditioner S_{BDDC} in a parallel fashion.

3. Solve the linear system for interface unknowns by using preconditioned conjugate gradient solver.

4. Calculate the interior unknown through a recovery process.

4.2.1 Schur complement

In Fig. 4.4 , the solid squares contain the unknown variables on the interface between each subdomain. These variables are on the boundary edge which is shared by adjacent elements on corresponding subdomains and should be calculated in the global matrix through Schur complement. The calculation process is given in Eq. (4.28). The goal of this step is to distribute the multiple square sub-matrices onto each processor for computing the inverse matrix locally. In doing so, the size of the global matrix decreases and includes information from the interface only.

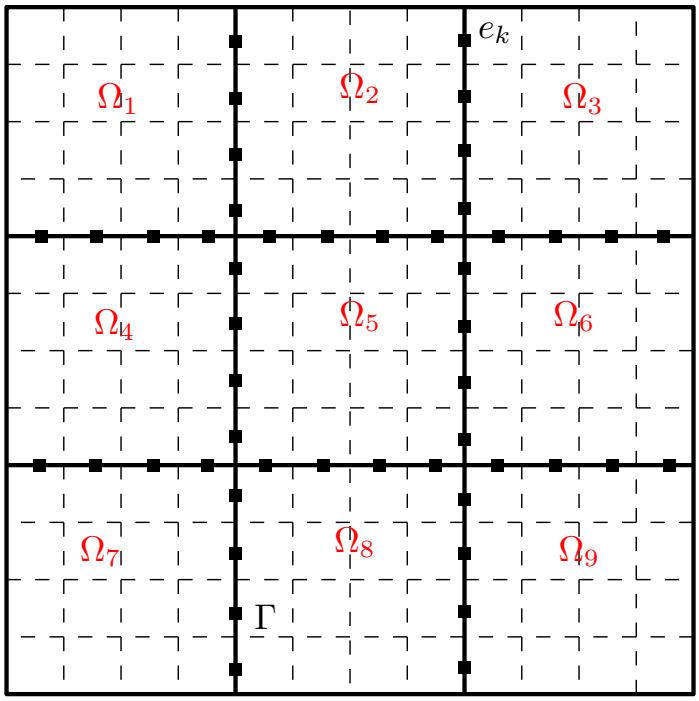


Figure 4.4: Decomposed computational .

Denote the weak Galerkin solution on each subdomain Ω_j. For the consistency with Eq. (1), we use $\mathbf{u}_h$ to represent the weak Galerkin solution on each Ω_j. To define the Schur complement system, the DOFs on every subdomain are partitioned by interior and interface categories. The unknown function becomes $\mathbf{u}_h = [\mathbf{u}_{I0}, \mathbf{u}_{Ib}, \mathbf{u}_{\Gamma b}]$, $\mathbf{v}_h = [\mathbf{v}_{I0}, \mathbf{v}_{Ib}, \mathbf{v}_{\Gamma b}]$ and denote the interior unknown variable $\mathbf{u}_i = [\mathbf{u}_{I0}, \mathbf{u}_{Ib}]$ on the interface we have $\mathbf{u}_\Gamma = \mathbf{u}_{\Gamma b}$. The same rule applies to the other known variable $\mathbf{v}_I = [\mathbf{v}_{I0}, \mathbf{v}_{Ib}]$, $\mathbf{v}_\Gamma = \mathbf{v}_{\Gamma b}$.

The governing equation in matrix form can be written as

$$\begin{pmatrix} A^u_{II} & (A^u_{\Gamma I})^T & 0 & 0 \\ A^u_{\Gamma I} & A^u_{\Gamma\Gamma} & 0 & A^{uv}_{\Gamma\Gamma} \\ 0 & 0 & A^v_{II} & (A^v_{\Gamma I})^T \\ 0 & A^{uv}_{\Gamma\Gamma} & A^v_{\Gamma I} & A^v_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} \mathbf{u}_I \\ \mathbf{u}_\Gamma \\ \mathbf{v}_I \\ \mathbf{v}_\Gamma \end{pmatrix} = \begin{pmatrix} \mathbf{f}^u_I \\ \mathbf{f}^u_\Gamma \\ \mathbf{f}^v_i \\ \mathbf{f}^v_\Gamma \end{pmatrix} \tag{4.50}$$

The interface stiffness matrix has the form as following

$$S_{\Gamma\Gamma}^{(j)} = A_{\Gamma\Gamma}^{(j)} - \left[A_{\Gamma I}^{(j)}\right]\left[A_{II}^{(j)}\right]^{-1}\left[A_{\Gamma I}^{(j)}\right]^{T}. \tag{4.51}$$

The assembled matrix form is then

$$S_{\Gamma\Gamma} = \sum_{j=1}^{N} {R_{\Gamma}^{(j)}}^{T} S_{\Gamma\Gamma}^{(j)} R_{\Gamma}^{(j)}, \tag{4.52}$$

where R_{Γ}^{j} is the mapping vector to convert unknown variables between the global interface Γ and the local interface Γ_i on subdomains Ω_j

Therefore, the global interface problem is constructed as

$$S_{\Gamma\Gamma}\begin{pmatrix}\mathbf{u}_{\Gamma}\\ \mathbf{v}_{\Gamma}\end{pmatrix} = \begin{pmatrix}\mathbf{f}_{\Gamma}^{u}\\ \mathbf{f}_{\Gamma}^{v}\end{pmatrix} \tag{4.53}$$

After the Schur complement, the global matrix size decreases containing information from the interface only. Even though the number of DOFs in the global matrix S_{Γ} is drastically decreased, the size of the global matrix and amount of parallel computing overhead is still too large. The computational space and time limitations remain a bottleneck that prevent us from demonstrating the effectiveness and efficiency of the method. Therefore, we introduce the preconditioner to accelerate performance. To obtain the preconditioner within the parallel framework, we divide the interface problem into primal and dual subspaces, the details of which are provided in the following section.

4.2.2 Primal and dual spaces

In Fig. 4.5, we split the interface into two spaces - primal and dual. The primal space consists of the solid squares, which are the constraints between each subdomain. These are the only unknown variables globally. The dual spae consists of the hollow

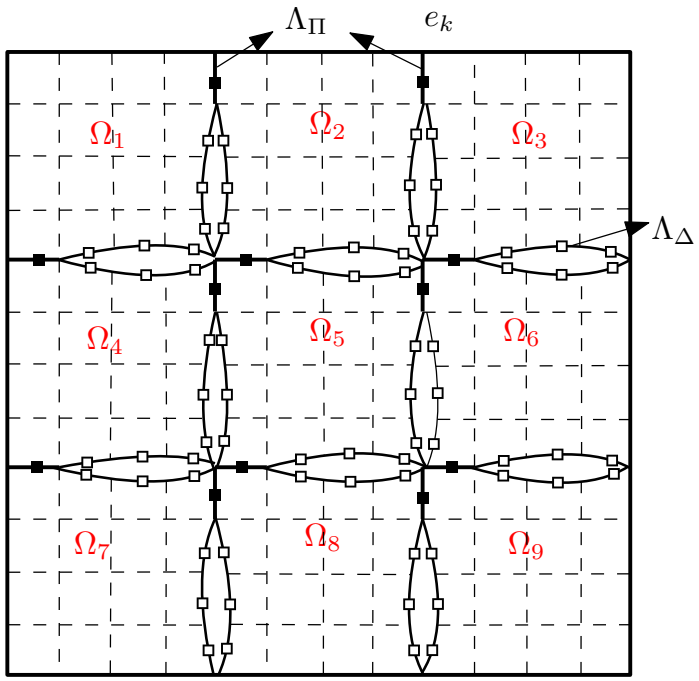


Figure 4.5: BDDC computational domain with only one cell boundary.

squares which represent the unknown variables local to each subspace. We connect the information from hollow squares in dual space to those in primal space through the preconditioner. The remaining solid squares are unknown variables in primal space. They are the only degrees of freedom which are communicated and calculated through MPI functions.

One important feature of the BDDC method is the boundedness of the condition number when the number of subdomains increases due to an appropriate choice of preconditioner. The condition number grows very slowly with increasing number of elements in each subdomain. In other words, the total number of iterations within the linear solver is also bounded. This helps the method scale well with the number of subdomains and problem size.

In the BDDC formulation, primal constraints are introduced over edges/faces. To define the preconditioner for the Schur complement problem, the interface space $\Lambda_\Gamma^{(j)}$ is a partitioned into two spaces, dual, $\Lambda_\Delta^{(j)}$ and primal, $\Lambda_\Pi^{(j)}$. The dual space, $\Lambda_\Delta^{(j)}$, corresponds to the subset of function in $\Lambda_\Gamma^{(j)}$.

We define the partially assembled space as:

$$\hat{\Lambda}_\Gamma = \hat{\Lambda}_\Pi \oplus (\sum_{i=1}^{N} \Lambda_\Delta^{(j)}) \tag{4.54}$$

where $\hat{\Lambda}_\Pi$ is the assembled global primal space, single valued on Γ, which is formed by assembling the local primal space, $\Gamma_\Pi^{(j)}$. The BDDC preconditioner has been viewed as solving a finite element problem on partially assembled finite element space, $\hat{\Lambda}_\Pi$, to precondition the Schur complement problem whose solution lies in the fully assembled space $\hat{\Lambda}_\Gamma$.

The key components of the BDDC preconditioner [19] are the following:

- An averaging operator which restricts functions from Λ_Γ to $\hat{\Lambda}_\Gamma$
- A positive scaling factor $\delta_i^\dagger(e_k)$ is defined for each interface e_k of the subdomain Ω_j such that $\delta_i^\dagger(e_k) + \delta_j^\dagger(e_k) = 1$ where $e_k = \partial\Omega_i \cap \partial\Omega_j$
- Define $D_\Gamma^{(i)}$ as the diagonal matrix formed by setting the diagonal entries corresponding to each nodal degree of freedom on e_k to $\delta_i^\dagger(e_k)$
- Define $R_{D,\Gamma} : \hat{\Lambda}_\Gamma \to \Lambda_\Gamma$ as the product of $R_{D,\Gamma} := D_\Gamma R_\Gamma$

We interpret the Eq. (**??**) by using the unknown variable function $\mathbf{u}_\Gamma = [\mathbf{u}_r, \mathbf{u}_c]^T$. The matrix can be written as

$$\begin{pmatrix} A_{rr}^{(1)} & 0 & 0 & \cdots & 0 & A_{rc}^{(1)} \\ 0 & A_{rr}^{(2)} & 0 & \cdots & 0 & A_{rc}^{(2)} \\ 0 & 0 & A_{rr}^{(3)} & \cdots & 0 & A_{rc}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_{rr}^{(N)} & A_{rc}^{(N)} \\ A_{cr}^{(1)} & A_{cr}^{(2)} & A_{cr}^{(3)} & \cdots & A_{cr}^{(N)} & A_{cc} \end{pmatrix} \begin{pmatrix} \underline{\mathbf{u}}_r^{(1)} \\ \underline{\mathbf{u}}_r^{(2)} \\ \underline{\mathbf{u}}_r^{(3)} \\ \vdots \\ \underline{\mathbf{u}}_r^{(N)} \\ \underline{\mathbf{u}}_\Pi \end{pmatrix} = \begin{pmatrix} f_r^{(1)} \\ f_r^{(2)} \\ f_r^{(3)} \\ \vdots \\ f_r^{(N)} \\ f_\Pi \end{pmatrix} \tag{4.55}$$

where the subscript c represents the unknown variables of black square constraints and the subscript r denotes the remaining unknown variable. For example, on subdomain (j), $\mathbf{u}_r^{(j)} = \mathbf{u}_I^{(j)} \cup \mathbf{u}_\Delta^{(j)}$.

From the interface unknowns in the primal spaces, in Eq. (4.54), we assemble all primal unknowns together and formulate a global constraints matrix

$$A_{cc} = \sum_{i=1}^{N} A_{\Pi\Pi}^{(j)} \tag{4.56}$$

where c represents the corner constraints located in only primal space.

Similar to Eq. (4.26), the rest unknown variables matrix can be written in

$$A_{rr}^{(j)} = \begin{bmatrix} A_{II}^{(j)} & A_{I\Delta}^{(j)} \\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)} \end{bmatrix} \tag{4.57}$$

The implementation of the WG-BDDC algorithm is presented in the following manner:

$$S_{\Gamma_{BDDC}}^{-1} = \hat{R}_{D,\Gamma}^T \{ R_{\Gamma,\Delta}^T (\sum_{j=1}^{N} \begin{bmatrix} 0 & R_\Delta^{(j)T} \end{bmatrix} \begin{bmatrix} A_{II}^{(j)} & A_{I\Delta}^{(j)} \\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ R_\Delta^{(j)} \end{bmatrix}) R_{\Gamma\Delta} + \Phi S_\Pi^{-1} \Phi^T \} \hat{R}_{D,\Gamma} \tag{4.58}$$

Similar to block Cholesky method in Eq. (4.28), the preconditioner is assembled by local inverse and global inverse where the matrix Φ represents the connection between Schur complement S and global stiffness matrix. In Eq. (40), the main difference is that we assemble the global matrix from multiple non-overlapped subdomains and the mapping matrix $\hat{R}$ is applied on each subdomain.

The matrix Φ is computed using

$$\Phi = R_{\Gamma\Pi}^T - R_{\Gamma\Delta}^T \sum_{j=1}^{N} \begin{bmatrix} 0 & R_\Delta^{(j)T} \end{bmatrix} \begin{bmatrix} A_{II}^{(j)} & A_{I\Delta}^{(j)} \\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)} \end{bmatrix}^{-1} \begin{bmatrix} A_{\Pi I}^{(j)T} \\ A_{\Pi\Delta}^{(j)\ T} \end{bmatrix} R_\Pi^{(j)}, \tag{4.59}$$

The matrix S_Π is the global coarse system matrix computed by

$$S_\Pi = \sum_{j=1}^{N} R_\Pi^{(j)^T} \{ A_{\Pi\Pi}^{(j)} - \begin{bmatrix} A_{\Pi I}^{(j)} & A_{\Pi\Delta}^{(j)} \end{bmatrix} \begin{bmatrix} A_{II}^{(j)} & A_{I\Delta}^{(j)} \\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)} \end{bmatrix}^{-1} \begin{bmatrix} A_{\Pi I}^{(j)^T} & A_{\Pi\Delta}^{(j)^T} \end{bmatrix} \} R_\Pi^{(j)}. \quad (4.60)$$

The major computational cost of calculating the global preconditioner lies in the inversion of the local matrix $\begin{bmatrix} A_{II}^{(j)} & A_{I\Delta}^{(j)} \\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)} \end{bmatrix}^{-1}$, and which is distributed and calculated on each processor (j). Thus the synchronization overhead is reduced to a very low level.

The BDDC preconditioner has the following form

$$S_{\Gamma_{BDDC}}^{-1} = R_{D,\Gamma}^T \tilde{S}_{\Gamma\Gamma}^{-1} R_{D,\Gamma}. \quad (4.61)$$

The global interface problem after preconditioning can be written as

$$S_{\Gamma_{BDDC}}^{-1} S_{\Gamma\Gamma} \mathbf{u}_\Gamma = S_{\Gamma_{BDDC}}^{-1} \mathbf{f}_\Gamma. \quad (4.62)$$

The preconditioned conjugate gradient is applied to solve the above linear system in Eq. (44). Theoretically, the condition number should be bounded, where

$$\kappa(S_{\Gamma_{BDDC}}^{-1} \hat{S}_{\Gamma\Gamma}) \leq C(1 + log(\frac{kH}{h}))^2 \quad (4.63)$$

for a second order elliptic problem. The constant C is independent of solution order p, element size h, and subdomain size H . Thus, the condition number and the number of iterations to achieve convergence are independent of the number of subdomains and only depend upon the accuracy order and subdomain size.

4.2.3 Preconditioned Conjugate Gradient Method

The preconditioned conjugate gradient (PCG) method is adopted as the linear solver for BDDC method. The construction of preconditioner is crucial for PCG method. The BDDC preconditioner combines the solution of the local problem on each subdomain with the solution of a global coarse problem while treating the coarse degrees of freedoms as unknowns.

The conjugate gradient (CG) method is an effective iterative numerical method for solving large-scale symmetric and positive definite linear systems. The method is straightforward to implement and has the capability to handle complex domains and boundary conditions.

The preconditioned conjugate gradient method has been reported by Bramble and Pasciak [6] to iteratively solve the symmetric saddle point problems. It inherits beneficial features of CG and extends it to a better performance. This method is applied to a sparse system which is too large to handle by a direct method such as the Cholesky decomposition.

The details of the PCG method are summarized by Algorithm I. The major effort in preconditioning is to assemble the global preconditioner matrix. Then, the CG method is applied to solve the preconditioned global matrix. By doing this, we can obtain the global interface solution with minimal iterations. Both global and local matrices are large and sparse, and we directly take advantage of the open source library LAPACK/BLAS [2].

The algorithm of PCG method is following:

Algorithm 1: Preconditioned Conjugate Gradient Algorithm

Input : $r_0 := b - M\underline{\mathbf{u}}_0$

$z_0 := S^{-1}_{\Gamma_{BDDC}} r_0$

$p_0 := z_0$

$k := 0$

1 Repeat ;

2

$$\alpha_k := \frac{r_k^T z_k}{p_k^T A p_k}$$

$$\underline{U}_{k+1} := \underline{\mathbf{u}}_k + \alpha_k p_k$$

$$r_{k+1} := r_k - \alpha_k A p_k$$

if r_k *is sufficiently small* **then**

3 return $\mathbf{u}_{k+1}$;

4 **else**

5

$$z_{k+1} := S^{-1}_{\Gamma_{BDDC}} r_{k+1}$$

$$\beta_k := \frac{z_{k+1}^T r_{k+1}}{z_k^T r_k}$$

$$p_{k+1} := z_{k+1} + \beta_k p_k$$

$$k := k + 1$$

6 **end**

where r is the residue vector, p is the orthogonal direction vector and k is the iteration number.

The Lanczos[**?**] matrix is applied to estimate the upper and lower eigenvalue

bounds. The matrix is in a tridiagonal form and generated from the PCG iterations. The global preconditioner matrix $S^{-1}_{\Gamma_{BDDC}}$ is converted into a tridiagonal matrix L_{mm}. When the m equals to the dimension of M, L_{mm} is similar to $S^{-1}_{\Gamma_{BDDC}}$. Then we calculate the eigenvalues of L_{mm} and obtain the condition number from calculating the ratio of the maximum and minimum eigenvalues.

4.2.4 Recover local information

The solution on the interface, $\mathbf{u}_\Gamma$, is obtained iteratively from the PCG solver. The small global vector result is communicated and shared by all processors. For each processor, the local interior DOFs, $\mathbf{u}_I$, can be recovered through the following equations. The local DOFs of interior unknowns and interface unknowns have the following relation

$$A_{II}^{u^{(j)}}\mathbf{u}_I^{(j)} + (A_{\Gamma I}^{u^{(j)}})^T\mathbf{u}_\Gamma^{(j)} = f_I^{(j)} \tag{4.64}$$

Since $u_\Gamma^{(j)}$ has been obtained from the PCG linear solver and $f_I^{(j)}$ is known, the local recovery is obtained by

$$\mathbf{u}_I^{(j)} = (A_{II}^{u^{(j)}})^{-1}(f_I^{(j)} - (A_{\Gamma I}^{u^{(j)}})^T\mathbf{u}_\Gamma^{(j)}) \tag{4.65}$$

where each local processor inverts the local positive definite square matrix $A_{II}^{u^{(j)}}$. The compute time for $(A_{II}^{u^{(j)}})^{-1}$ decreases significantly as we increase the number of subdomains.

4.2.5 Workflow of the parallel computing scheme

Message Passing Interface (MPI) [25] is portable and widely used as the communicator. We applied MPI to exchange the information between each non-overlapping subdomain. MPI provides a standard set of Fortran subprogram definitions. Intel MKL supports the modern MPI version which allows us to migrate the software on a

variety of platforms. Besides, the MPI subprograms introduce the minimum overhead in both coding and testing stages.

The workflow of parallel computing scheme is following:

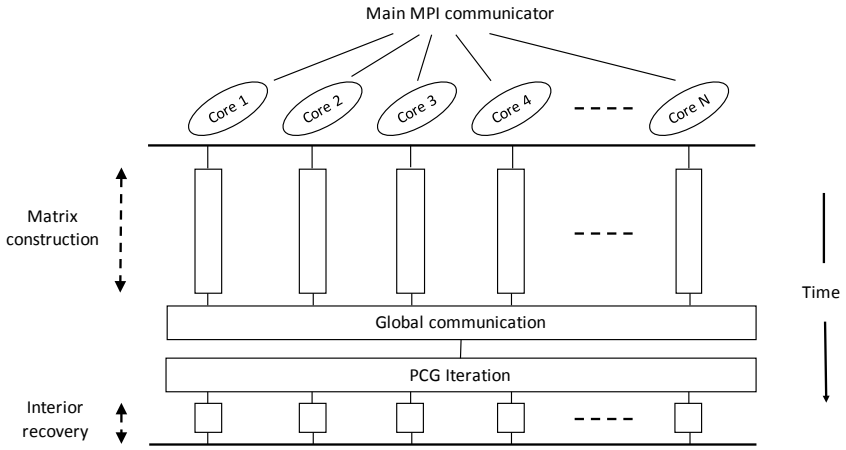


Figure 4.6: Parallel computing work flow.

1. MPI communicator initiates work and distributes the parameters, configuration and mesh data to all the processors in the same time. The global matrix M is divided into two spaces primal and dual. For each processor, it reads the corresponding geometry and connectivity information.

2. For each processor, the local elemental matrices are constructed individually, in Fig. 4.6 the matrix construction section. The primal matrix S_{Π} is constructed in this step. Every processor deals with one non-overlapping subdomain refers to Eqs. (4.59) and (4.60). The local inverse is calculated by using block Cholesky elimination and communicate through interface with global mapping vectors $R_{D,\Gamma}$.

3. Through MPI subprograms, the local matrices are communicated and the global preconditioner is constructed and assembled through each processor. The pre-

conditioner $S^{-1}_{\Gamma_{BDDC}}$ refers to Eq. (4.58) which is the assemblage from each individual processor.

4. The global problem, is calculated on every processor with PCG linear solver. The details of PCG solver illustrated in Algorithm 1.

5. The global solution is reduced to each processor for the local solution recovery, in Fig. 4.6 interior recovery section. When the global solution in primal space shared with each processors, the local unknown DOFs can be recovered individually by Eq. (4.65)

Throughout this paper, the results from our WG-BDDC code has been obtained on Colonial One, a supercomputing facility at the George Washington University. ColonialOne has total 200+ nodes of Xeon E5-2650 8-core processors. Each processor has 2.6GHz processing speed.

4.3 Numerical Results

4.3.1 Poisson Equation

We can choose different order of basis functions in a WG element for the weak gradient equation among gradient, interior and boundary basis functions. We test different combination orders of interior, boundary and weak gradient shape functions.

In the first test, we consider the Poisson equation $-\nabla\cdot(\nabla\mathbf{u}) = \mathbf{f}$. Let $\Omega = (0,1)\times(0,1)$, $a = I$, and $\mathbf{f}$ are chosen such that the exact solution is $u = sin(\pi x)sin(\pi y)$

We choose different weak Galerkin elements for validating our WG-BDDC numerical scheme. The unit square is first decomposed into $N \times N$ subdomains as the coarse mesh with length $H = 1/N$. In every subdomain, all elements are further triangulated into $2n \times n$ triangles, and the finite mesh has the size $h = 1/(N \times n)$. The preconditioned system is solved by the PCG solver. In every iteration, the L^2-norm of the residual is reduced by a factor of 10^{-7}.

#sub	$k=1$				$k=2$			
	Cond.	Iter.	L^2-error	O	Cond.	Iter.	L^2-error	O
4×4	2.217	5	1.6013e-3	-	3.528	8	7.1456e-5	-
8×8	2.390	9	3.9939e-4	2.0	3.803	10	8.9214e-6	3.0
16×16	2.335	8	9.9789e-5	2.0	3.768	10	1.1150e-6	3.0
32×32	2.325	8	2.4944e-5	2.0	3.758	10	1.3938e-7	3.0

Table 4.1: The accuracy and convergence properties of the $P_kP_{k-1}P_{k-1}^2$ element on different numbers of subdomains where each subdomain has 128 triangular elements.

H/h	$k=1$				$k=2$			
	Cond.	Iter.	L^2-error	O	Cond.	Iter.	L^2-error	O
4	1.722	7	1.6013e-3	-	2.900	10	7.1456e-5	-
8	2.390	9	3.9939e-4	2.0	3.803	10	8.9214e-6	3.0
16	3.245	10	9.9789e-5	2.0	4.957	12	1.1150e-6	3.0
32	4.239	11	2.4944e-5	2.0	6.218	13	1.3938e-7	3.0

Table 4.2: The accuracy and convergence $P_kP_{k-1}P_{k-1}^2$ scheme on different number of fine element in each subdomain. The entire computational domain is partitioned into 64 non-overlapping subdomains.

In above examples, the largest test case contains more than 1 million degree of freedoms. The first test is implemented for weak Galerkin element with $u_0 \in P_k, u_b \in P_{k-1}$, and $\nabla_w u \in P_{k-1}$. The Tables 4.1 and 4.2 show the condition number of Lanczos matrix and the iteration number in the PCG solver. From the Table 4.1 and Table

4.2, we can see that the condition number does not increase proportionally with the increasing number of subdomains. It depends upon the resolution of mesh, H/h, as $(1 + log(\frac{H}{h}))^2$. With the increasing number of subdomains, we obtain stable second and third order accuracy results. Similarly, the iteration number is also well-bounded. Due to the fact that the size of global matrix grows at a slower rater than the DOFs in primal space.

#sub	$k=1$				$k=2$			
	Cond.	Iter.	L^2-error	O	Cond.	Iter.	L^2-error	O
4×4	2.451	7	1.0109e-3	-	3.805	8	6.6333e-5	-
8×8	2.648	9	2.5117e-4	2.0	4.003	12	8.2709e-6	3.0
16×16	2.629	9	6.2696e-5	2.0	3.943	12	1.0334e-6	3.0
32×32	2.617	9	1.5668e-5	2.0	3.917	12	1.2917e-7	3.0

Table 4.3: The accuracy and convergence properties of the $P_k P_k P_{k-1}^2$ element on different numbers of subdomains where each subdomain has 128 triangular elements.

H/h	$k=1$ and #sub=64				$k=2$ and #sub=64			
	Cond.	Iter.	L^2-error	$O1$	Cond.	Iter.	L^2-error	O
4	1.968	8	1.0109e-3	-	3.926	11	6.6333e-5	-
8	2.648	9	2.5117e-4	2.0	4.003	12	8.2709e-6	3.0
16	3.529	10	6.2696e-5	2.0	5.084	13	1.0334e-6	3.0
32	4.619	12	1.5668e-5	2.0	6.329	13	1.2918e-7	3.0

Table 4.4: The accuracy and convergence $P_kP_kP_{k-1}^2$ scheme on different number of fine element in each subdomain. The entire computational domain is partitioned into 64 non-overlapping subdomains.

The second test is the weak Galerkin element with order $u_0 \in P_k$, $u_b \in P_k$ and $\nabla_w u \in P_{k-1}$. In Table 4.3 and Table 4.4 the condition number has the identical pattern of the theoretical convergence rate. Comparing to the first example, we find that convergence rates and the orders of accuracy have optimal agreement to the degree of polynomial in u_0. The maximum number of processors for this example is up 1024. The WG-BDDC method shows optimal scalability.

#sub	$k=1$				$k=2$			
	Cond.	Iter.	L^2-error	O	Cond.	Iter.	L^2-error	O
4×4	3.671	8	2.1451e-4	-	4.620	8	6.7627e-6	-
8×8	3.965	10	5.2129e-5	2.0	4.987	12	7.8998e-7	3.0
16×16	3.934	10	1.2937e-5	2.0	4.921	12	9.6925e-8	3.0
32×32	3.922	10	3.2281e-6	2.0	4.901	12	1.2058e-8	3.0

Table 4.5: The accuracy and convergence properties of the $P_kP_kP_k^2$ element on different numbers of subdomains where each subdomain has 128 triangular elements.

H/h	$k=1$				$k=2$			
	Cond.	Iter.	L^2-error	O	Cond.	Iter.	L^2-error	O
4	3.024	10	2.1451e-4	-	3.859	11	6.7628e-6	-
8	3.965	10	5.2129e-5	2.0	4.987	12	7.8998e-7	3.0
16	5.153	12	1.2937e-5	2.0	6.235	13	9.6931e-8	3.0
32	6.472	14	3.2281e-6	2.0	7.673	15	1.2060e-8	3.0

Table 4.6: The accuracy and convergence $P_kP_kP_k^2$ scheme on different number of fine element in each subdomain. The entire computational domain is partitioned into 64 non-overlapping subdomains.

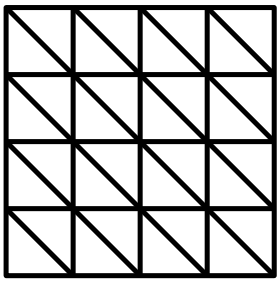
In Tables 4.5 and 4.6, we implement the WG element $u_0 \in P_k$, $u_b \in P_k$ and $\nabla_w u \in P_k$. The $P_kP_kP_k^2$ element delivers the lowest error result among all three types of element. The reason for this lies in the fact that all three shape functions of interior variable, boundary variable and weak gradient have the same order.

4.3.2 Linear Elastic Equation

We consider the linear elastic (3.1) in the square domain $\Omega = (0,1)^2$ which is decomposed into uniform square subdomains with size H. The exact solution is given by

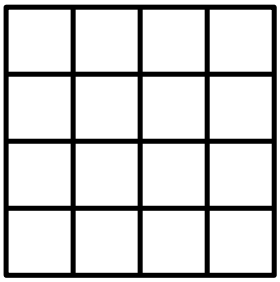
$$\mathbf{u} = \begin{pmatrix} sin(2\pi x)sin(2\pi y) \\ 1 \end{pmatrix} \tag{4.66}$$

To show the WG method for unstructured meshes, we present accurate results in Table 7 calculated for triangular elements. We also demonstrate implementation of the WG method for quadrilateral elements and show results in Table 8.



k = 1				
#WG	$Error(u_0)$	$O(u_0)$	$Error(u_b)$	$O(u_b)$
4	$9.484e-2$	-	$8.484e-2$	-
16	$2.678e-3$	1.9	$2.178e-3$	1.9
64	$5.570e-4$	2.0	$5.470e-4$	1.9
256	$1.437e-4$	2.0	$1.367e-4$	2.0

Table 4.7: Numerical results and accuracy of triangular elements.



k = 1				
#WG	$Error(u_0)$	$O(u_0)$	$Error(u_b)$	$O(u_b)$
4	$8.368e-2$	-	$9.103e-2$	-
16	$1.944e-3$	1.9	$2.058e-3$	1.9
64	$5.337e-4$	2.0	$5.280e-4$	2.0
256	$1.086e-4$	2.0	$1.451e-4$	2.0

Table 4.8: Numerical results and accuracy of quadrilateral elements.

After applying the BDDC scheme, we can partition the original computational domain into several non-overlapping subdomains, and, we present the scalability of parallel computing for solving Eq. (48) in Fig. 4.7.

#sub	$H/h = 8$				H/h	$k = 1$ and #sub=64			
	Cond.	Iter.	L_{Max}-error	O		Cond.	Iter.	L^2-error	O
2×2	2.281	8	8.484e-2	-	4	2.212	11	2.993e-1	-
4×4	3.922	12	2.1787e-2	1.96	8	3.069	12	8.567e-2	1.9
8×8	4.895	17	5.4706e-3	1.99	16	4.143	13	2.217e-2	2.0
16×16	5.238	17	1.3675e-3	2.00	32	5.437	15	5.575e-3	2.0

Table 4.9: The accuracy and convergence properties with $\lambda = 1, \mu = 0.5$ on quadrilateral linear elements.

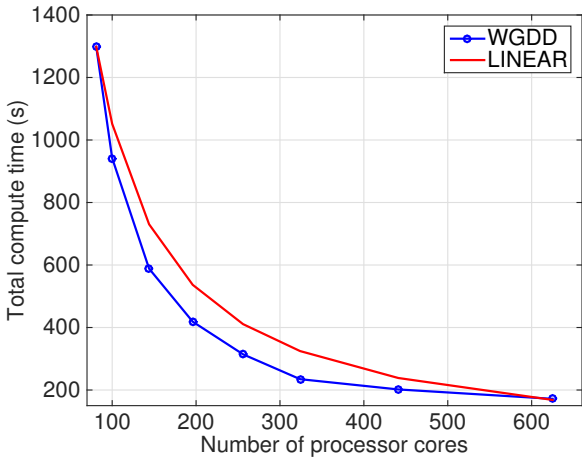


Figure 4.7: The running time .vs. number of processors.

In this figure, we plot the average compute time of each processor for a mesh first decomposed into 2×2 subdomains and then uniformly partitioned to 25×25 subdomains. the circle line represents WG-BDDC method. The total number of mesh elements remains the same. This figure indicates the reduction in compute time for increasing number of processors.

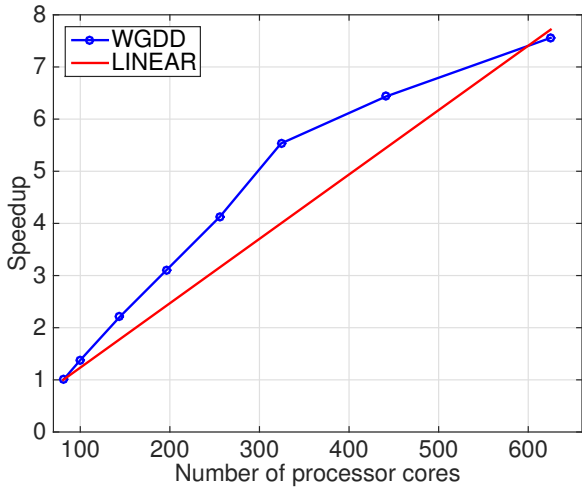


Figure 4.8: The speedup .vs. number of processors.

We can see from Fig. 4.8 that superlinear speedup is obtained with the increasing number of subdomains. When the number of subdomains reaches 300, the highest speedup rate is achieved because it is the best ratio of global to local cost produces. The communication overhead takes the lowest portion in the total computational

effort. In each local subdomain, the critical ration exists between the number of the degree of freedoms in primal space (global matrix) to these in dual space (local matrix). The costs for Cholesky elimination and the PCG solver are $O(n^3)$ and $O(n\sqrt{(k)})$, respectively. We can obtain significant superlinear speedup through domain decomposition. On the other hand, the growth of global matrix introduces increasing overhead. It is the main reason of the speedup curve flatting with the increasing number of processor. The main factor of the WG-BDDC method is the balance between the global matrix, the preconditioner, and the local matrices. An appropriate estimate on balancing load distributions will produce optimal performance on modern computer structure.

4.3.3 Locking-Free Tests

To investigate the locking-free feature of the WG-BDDC method on linear elasticity equation, we consider the exact solution given by

$$u = \begin{pmatrix} sin(2\pi x)sin(2\pi y) \\ cos(2\pi x)cos(2\pi y) \end{pmatrix} + \lambda^{-1} \begin{pmatrix} x \\ y \end{pmatrix} \tag{4.67}$$

The locking-free feature refers to lack of constraints applied on Lame index λ and μ. For instance, the Poisson's ratio ν is close to $\frac{1}{2}$ (the material property is nearly incompressible, Lame indices λ is very large and μ is $\frac{1}{2}$) which indicates that the displacement of the exact solution must fit $\nabla \cdot u = 0$. This is a very challenging condition for continuous Galerkin FEM. However, this difficulty can be addressed by the WG-BDDC method. We design a series of numerical tests to explain that WG-BDDC provides good locking-free convergence.

#sub	$H/h=8$				H/h	$k=1$ and #sub=64			
	Cond.	Iter.	L_{Max}-error	O		Cond.	Iter.	L^2-error	O
2×2	2.281	8	7.522e-2	-	4	2.374	11	2.535e-1	-
4×4	3.922	13	2.543e-2	1.56	8	3.272	12	8.967e-2	1.50
8×8	4.895	16	7.533e-3	1.75	16	4.538	13	2.853e-2	1.66
16×16	5.238	18	2.017e-3	1.90	32	5.751	15	7.834e-3	1.86

Table 4.10: The accuracy and convergence properties of quadrilateral linear element, $\lambda=1,\mu=0.5$

#sub	$H/h=8$				H/h	$k=1$ and #sub=64			
	Cond.	Iter.	L_{Max}-error	O		Cond.	Iter.	L^2-error	O
2×2	2.281	8	7.523e-2	-	4	2.374	11	2.535e-1	-
4×4	3.922	13	2.543e-2	1.56	8	3.272	12	8.967e-2	1.50
8×8	4.895	16	7.533e-3	1.75	16	4.538	13	2.851e-2	1.66
16×16	5.238	18	2.017e-3	1.90	32	5.751	15	7.834e-3	1.86

Table 4.11: The accuracy and convergence properties of quadrilateral linear element, $\lambda=1,000,\mu=0.5$

#sub	$H/h=8$				H/h	$k=1$ and #sub=64			
	Cond.	Iter.	L_{Max}-error	O		Cond.	Iter.	L^2-error	O
2×2	2.281	8	7.52e-2	-	4	2.374	11	2.535e-1	-
4×4	3.922	13	2.545e-2	1.56	8	3.272	12	8.967e-2	1.50
8×8	4.895	16	7.533e-3	1.75	16	4.538	13	2.854e-2	1.66
16×16	5.238	18	2.017e-3	1.90	32	5.751	15	7.834e-3	1.86

Table 4.12: The accuracy and convergence properties of quadrilateral linear element, $\lambda=1,000,000,\mu=0.5$

Tabless 4.10, 4.11 and 4.12 show that the order of accuracy is not affected by the values of Lame indices. The locking-free feature is inherited from the WG method

and is also successfully demonstrated by the WG-BDDC scheme.

4.4 Summary

In this chapter, a novel parallel computing method is presented for solving linear elasticity problems. This method integrates the newly developed weak Galerkin (WG) finite element method with the balancing domain decomposition with constraints (BDDC). The WG-BDDC method is implemented using Fortran and MPI. The WG-BDDC method is demonstrated to have optimal order of accuracy for both 2nd and 3rd order numerical discretizations. The method is also proven to have outstanding scalability with superlinear speedup when the number of processors increases to over 600 processors. The condition numbers of the Lanczos matrix from the global interface for all test cases presented in this paper are well bounded demonstrating a fast and robust convergence. The WG-BDDC method has locking-free feature. It provides accurate results for almost incompressible materials.

Chapter 5: Conclusions and Future Work

5.1 Conclusions

In this dissertation, we successfully implement the weak Galerkin finite element method for solving elasticity equation on parallel computers. We obtained optimal convergence rate, expected order of accuracy and excellent scalability from our WG solvers. The results showed solid performance and great capability of the WG method for solving large scale structural problems on supercomputer cluster. The WG-BDDC solver is an optimal candidate to construct a robust and efficient partitioned parallel fluid-structural interaction solver.

Firstly, we developed a novel hybrid element combining the weak Galerkin finite element with the classic continuous Galerkin finite element. It is designed by inserting multiple CG elements into one WG elements and considered the hybrid element as a non-overlapping subdomain. This hybrid element owns the advantages of the discontinuous feature from the WG method and the computational convenience from the CG method. The hybrid WG-CG element is implemented by using Schur complement method and MPI library for parallel computing. The second order of accuracy is illustrated with different number of CG elements in the hybrid WG-CG element for interior and boundary unknown variables. For linear and nonlinear elasticity equation, we compare the results between our parallel hybrid WG-CG element and an established serial CG solver. The results are identical with an average difference down to 10^{-6} for Dirichlet and Neumann boundary conditions. The hybrid element method achieved superlinear speedup up to 60 processors on supercomputing cluster, ColonialOne, GWU.

We designed an improved stabilizer for implementing implicit time marching scheme for the WG method and the hybrid WG-CG element method for solving the equation of motion. In this improved stabilizer, the acceleration vector replaces

the displacement vector as the unknown variable to calculate the boundary integral. The result matrix completes the information of the mass matrix for the boundary unknown variables. In every time step, the new acceleration can be calculated with the help of the new mass matrix. By using the improved stabilizer, we can compute the implicit time marching scheme with the very low computational cost.

The hybrid WG-CG element method and improved stabilizer built a solid foundation for developing 2-D parallel WG solver. To overcome the new challenge of increasing interfacial unknown variables, we have successfully designed a novel parallel computing method for solving linear elasticity problems. This method integrates the newly developed weak Galerkin (WG) finite element method with a balancing domain decomposition with constraints (BDDC). The WG-BDDC method is implemented by using Fortran and MPI libraries. The WG-BDDC method is demonstrated to have optimal order of accuracy and convergence properties for both 2nd- and 3rd- order numerical discretization. This method is also proven to have outstanding scalability with superlinear speedup when the number of processors increases by testing up to 600 processors. The condition numbers of the Lanczos matrix from the global primal problem for all test cases presented in this paper are well bounded demonstrating fast and robust convergence properties.

5.2 Future Work

The Future Work consists of two major steps: 1) further develop the WG-BDDC scheme for nonlinear elasticity problems, 2) extend the current WG-BDDC to handle 3 dimensional geometry and meshes including the tetrahedron and the hexahedron elements.

The present WG-BDDC method is built on triangular and quadrilateral 2 dimensional elements for solving linear elasticity problems. To solve more complicated real-world problems, the nonlinear elasticity equation and three dimensional geometry model are needed. Our ultimate goal is to simulate complex real-world engineering

problems with our efficient high-order accuracy structural and fluid solver. The IMEX coupling scheme will combine the two independent solvers together and are expected to produce high-fidelity results.

References

[1] Adeli, H., Weaver, W., Gere, J. M., 1978. Algorithms for nonlinear structural dynamics. Journal of the Structural Division 104 (2), 263–280.

[2] Anderson, E., Bai, Z., Bischof, C., Blackford, L. S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., et al., 1999. LAPACK Users' guide. SIAM.

[3] Barry Issenberg, S., Mcgaghie, W. C., Petrusa, E. R., Lee Gordon, D., Scalese, R. J., 2005. Features and uses of high-fidelity medical simulations that lead to effective learning: a beme systematic review. Medical teacher 27 (1), 10–28.

[4] Bathe, K.-J., 2008. Finite element method. Wiley Online Library.

[5] Bertram, C., 2010. Evaluation by fluid/structure-interaction spinal-cord simulation of the effects of subarachnoid-space stenosis on an adjacent syrinx. Journal of biomechanical engineering 132 (6), 061009.

[6] Bramble, J. H., Pasciak, J. E., 1988. A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems. Mathematics of Computation 50 (181), 1–17.

[7] Brenner, S., Scott, R., 2007. The mathematical theory of finite element methods. Vol. 15. Springer Science & Business Media.

[8] Brussino, G., Sonnad, V., 1989. A comparison of direct and preconditioned iterative techniques for sparse, unsymmetric systems of linear equations. International journal for numerical methods in engineering 28 (4), 801–815.

[9] Ciarlet, P. G., 2002. The finite element method for elliptic problems. SIAM.

[10] Clark, C., 1976. The fluid mechanics of aortic stenosisi. theory and steady flow experiments. Journal of biomechanics 9 (8), 521–528.

[11] Conte, M., Teraa, M., Conte, M., Moll, F., Verhaar, M., 2016. Critical limb ischemia: Current trends and future directions.

[12] Courant, R., 1994. Variational methods for the solution of problems of equilibrium and vibrations. Lecture Notes in Pure and Applied Mathematics, 1–1.

[13] Cundall, P. A., Strack, O. D., 1979. A discrete numerical model for granular assemblies. geotechnique 29 (1), 47–65.

[14] Davio, M., 1981. Kronecker products and shuffle algebra. IEEE Transactions on Computers 100 (2), 116–125.

[15] Degroote, J., Bathe, K.-J., Vierendeels, J., 2009. Performance of a new partitioned procedure versus a monolithic procedure in fluid–structure interaction. Computers & Structures 87 (11-12), 793–801.

[16] Dohrmann, C. R., 2003. A preconditioner for substructuring based on constrained energy minimization. SIAM Journal on Scientific Computing 25 (1), 246–258.

[17] Dohrmann, C. R., 2003. A study of domain decomposition preconditioners. Tech. rep., Technical Report SAND2003-4391, Sandia National Laboratories, Albuquerque, New Mexico.

[18] Duff, I. S., Erisman, A. M., Reid, J. K., 1986. Direct methods for sparse matrices. Clarendon press Oxford.

[19] Eisenstat, S. C., 1981. Efficient implementation of a class of preconditioned conjugate gradient methods. SIAM Journal on Scientific and Statistical Computing 2 (1), 1–4.

[20] Farhat, C., Lesoinne, M., LeTallec, P., Pierson, K., Rixen, D., 2001. Feti-dp: a dual–primal unified feti methodpart i: A faster alternative to the two-level feti method. International journal for numerical methods in engineering 50 (7), 1523–1544.

[21] Farhat, C., Mandel, J., 1998. The two-level feti method for static and dynamic plate problems part i: An optimal iterative solver for biharmonic systems. Computer methods in applied mechanics and engineering 155 (1-2), 129–151.

[22] Farhat, C., Mandel, J., Roux, F. X., 1994. Optimal convergence properties of the feti domain decomposition method. Computer methods in applied mechanics and engineering 115 (3-4), 365–385.

[23] Farhat, C., Roux, F.-X., 1991. A method of finite element tearing and interconnecting and its parallel solution algorithm. International Journal for Numerical Methods in Engineering 32 (6), 1205–1227.

[24] Feng, R., Xenos, M., Girdhar, G., Kang, W., Davenport, J. W., Deng, Y., Bluestein, D., 2012. Viscous flow simulation in a stenosis model using discrete particle dynamics: a comparison between dpd and cfd. Biomechanics and modeling in mechanobiology 11 (1-2), 119–129.

[25] Gropp, W., Lusk, E., Doss, N., Skjellum, A., 1996. A high-performance, portable implementation of the mpi message passing interface standard. Parallel computing 22 (6), 789–828.

[26] Hahn, G., 1991. A modified euler method for dynamic analyses. International Journal for Numerical Methods in Engineering 32 (5), 943–955.

[27] Hrennikoff, A., 1941. Solution of problems of elasticity by the framework method. Journal of applied mechanics 8 (4), 169–175.

[28] Hübner, B., Walhorn, E., Dinkler, D., 2004. A monolithic approach to fluid–structure interaction using space–time finite elements. Computer methods in applied mechanics and engineering 193 (23-26), 2087–2104.

[29] Hughes, T. J., 2012. The finite element method: linear static and dynamic finite element analysis. Courier Corporation.

[30] Jameson, A., 1991. Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings. In: 10th Computational Fluid Dynamics Conference. p. 1596.

[31] Klawonn, A., Widlund, O. B., 2001. Feti and neumann-neumann iterative substructuring methods: connections and new results. Communications on pure and applied Mathematics 54 (1), 57–90.

[32] Klawonn, A., Widlund, O. B., 2006. Dual-primal feti methods for linear elasticity. Communications on pure and applied mathematics 59 (11), 1523–1572.

[33] Küttler, U., Wall, W. A., 2008. Fixed-point fluid–structure interaction solvers with dynamic relaxation. Computational Mechanics 43 (1), 61–72.

[34] Li, J., Widlund, O. B., 2006. Feti-dp, bddc, and block cholesky methods. International journal for numerical methods in engineering 66 (2), 250–271.

[35] Li, Q. H., Wang, J., 2013. Weak galerkin finite element methods for parabolic equations. Numerical Methods for Partial Differential Equations 29 (6), 2004–2024.

[36] Liang, C., Papadakis, G., 2007. Large eddy simulation of cross-flow through a staggered tube bundle at subcritical reynolds number. Journal of Fluids and Structures 23 (8), 1215–1230.

[37] Liang, C., Papadakis, G., 2007. Large eddy simulation of pulsating flow over a circular cylinder at subcritical reynolds number. Computers & fluids 36 (2), 299–312.

[38] Liang, C., Papadakis, G., Luo, X., 2009. Effect of tube spacing on the vortex shedding characteristics of laminar flow past an inline tube array: a numerical study. Computers & Fluids 38 (4), 950–964.

[39] Long, Q., Xu, X., Ramnarine, K., Hoskins, P., 2001. Numerical investigation of physiologically realistic pulsatile flow through arterial stenosis. Journal of biomechanics 34 (10), 1229–1242.

[40] Mandel, J., 1993. Balancing domain decomposition. Communications in numerical methods in engineering 9 (3), 233–241.

[41] Mandel, J., Dohrmann, C. R., Tezaur, R., 2005. An algebraic theory for primal and dual substructuring methods by constraints. Applied numerical mathematics 54 (2), 167–193.

[42] Mu, L., Wang, J., Wang, Y., Ye, X., 2013. A computational study of the weak galerkin method for second-order elliptic equations. Numerical Algorithms 63 (4), 753–777.

[43] Mu, L., Wang, J., Wang, Y., Ye, X., 2013. A weak galerkin mixed finite element method for biharmonic equations. In: Numerical Solution of Partial Differential Equations: Theory, Algorithms, and Their Applications. Springer, pp. 247–277.

[44] Mu, L., Wang, J., Ye, X., 2012. Weak galerkin finite element methods on polytopal meshes. arXiv preprint arXiv:1204.3655.

[45] Mu, L., Wang, J., Ye, X., 2014. Weak galerkin finite element methods for the

biharmonic equation on polytopal meshes. Numerical Methods for Partial Differential Equations 30 (3), 1003–1029.

[46] Mu, L., Wang, J., Ye, X., 2015. A new weak galerkin finite element method for the helmholtz equation. IMA Journal of Numerical Analysis 35 (3), 1228–1255.

[47] Nadeem, S., Akbar, N. S., 2010. Simulation of the second grade fluid model for blood flow through a tapered artery with a stenosis. Chinese Physics Letters 27 (6), 068701.

[48] Nesbitt, W. S., Westein, E., Tovar-Lopez, F. J., Tolouei, E., Mitchell, A., Fu, J., Carberry, J., Fouras, A., Jackson, S. P., 2009. A shear gradient–dependent platelet aggregation mechanism drives thrombus formation. Nature medicine 15 (6), 665–673.

[49] Ng, M. K., Chan, R. H., Tang, W.-C., 1999. A fast algorithm for deblurring models with neumann boundary conditions. SIAM Journal on Scientific Computing 21 (3), 851–866.

[50] Ogulu, A., Abbey, T., 2005. Simulation of heat transfer on an oscillatory blood flow in an indented porous artery. International communications in heat and mass transfer 32 (7), 983–989.

[51] Reddy, J. N., 1993. An introduction to the finite element method. Vol. 2. McGraw-Hill New York.

[52] Stergiopulos, N., Young, D., Rogge, T., 1992. Computer simulation of arterial flow with applications to arterial and aortic stenoses. Journal of biomechanics 25 (12), 1477–1488.

[53] Tu, X., 2007. Three-level bddc in three dimensions. SIAM Journal on Scientific Computing 29 (4), 1759–1780.

[54] Tu, X., 2007. Three-level bddc in two dimensions. International journal for numerical methods in engineering 69 (1), 33–59.

[55] Vierendeels, J., Lanoye, L., Degroote, J., Verdonck, P., 2007. Implicit coupling of partitioned fluid–structure interaction problems with reduced order models. Computers & structures 85 (11-14), 970–976.

[56] Wang, C., Wang, J., Wang, R., Zhang, R., 2016. A locking-free weak galerkin finite element method for elasticity problems in the primal formulation. Journal of Computational and Applied Mathematics 307, 346–366.

[57] Wang, J., Ye, X., 2013. A weak galerkin finite element method for second-order elliptic problems. Journal of Computational and Applied Mathematics 241, 103–115.

[58] Wang, J., Ye, X., 2014. A weak galerkin mixed finite element method for second order elliptic problems. Mathematics of Computation 83 (289), 2101–2126.

[59] Wang, J., Ye, X., 2016. A weak galerkin finite element method for the stokes equations. Advances in Computational Mathematics 42 (1), 155–174.

[60] Wardlaw, J., Chappell, F., Best, J., Wartolowska, K., Berry, E., et al., 2006. Non-invasive imaging compared with intra-arterial angiography in the diagnosis of symptomatic carotid stenosis: a meta-analysis. The Lancet 367 (9521), 1503–1512.

[61] Zhang, X., Liang, C., Li, L., Zhang, Z., Lee, J., 2016. A high order spectral difference method for fluid-structure interaction using an implicit-explicit rk coupling scheme. In: 46th AIAA Fluid Dynamics Conference. p. 3354.

[62] Zienkiewicz, O. C., Taylor, R. L., Taylor, R. L., 1977. The finite element method. Vol. 3. McGraw-hill London.