**A New Finite Element Method Enabling Parallel Solutions for Second Order Elliptic Equation**

by Liangwei Li

B.S. in Mechanical Engineering, June 2011, Sun Yat-San 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 31, 2017

Dissertation directed by

Chunlei Liang, Junping Wang
Associate Professor of Engineering and Applied Science

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 March 15, 2017. This is the final and approved form of the dissertation.

**A New Finite Element Method Enabling Parallel Solutions for Second Order Elliptic Equation**

Liangwei Li

Dissertation Research Committee:

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

Adam Wickenheiser, Assistant Professor of Engineering and Applied Science, Committee Member

Junping Wang, Professor of Engineering and Applied Science, Committee Member

Lin Mu, Professor of Mathematics, Committee Member

Mortan Friedman, Assistant Professor of Mechanical Engineering, Committee Member

## Dedication

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

**Acknowledgements**

Great thanks to Dr. Chunlei Liang, Dr Junping Wang and Dr Lin Mu.

## Abstract

### A New Finite Element Method Enabling Parallel Solutions for Second Order Elliptic Equation

In this paper, we present a novel parallel computing method to efficiently solve linear elasticity problems on unstructured meshes. The implementation of our parallel method is based on the weak Galerkin (WG) finite element method which has been recently developed by Wang and Ye and Mu et al. The core idea of the WG method for solving linear elasticity equation is to introduce weak strain and stress tensors by using the concept of discrete weak gradients. The weak Galerkin finite element method refers to a general finite element method to solve partial differential equations. The main feature is that the differential operators are calculated through the weak functions and reconstructed by solving the relatively inexpensive elemental matrices on each element.

Linear elasticity is the equation describe how solid objects stress and strain distribute due to the external and internal prescribed loading condition. This equation requires the materials as continua. On the purpose of solving large-scale fluid structure interaction problems, the partitioned approach, the governing equation for fluid and displacement of the structure are calculated in two different solver. We present an accurate and efficient solid solver which is compatible for high order fluid solvers for Navier-Stokes equations.

To enable parallel computation, the computational grid is split into arbitrary number of subdomains.We present two different approaches to implement the parallel computing. Firstly, we combine the classic continuous Galerkin (CG) finite element with weak Galerkin finite element method together and develop a hybrid element. The hybrid element inherit the discontinuous feature from WG method and the computational efficiency from CG method. The other method is to implement

the primal and dual spaces to split the computational spaces for ech subdomain. The connection of adjacent subdomains is realized through the balancing domain decomposition with constraints (BDDC) which was originally proposed in. Locally over each subdomain, matrices are constructed for interior and interface quantities separately. Subsequently, interface related ma- trices are passed over to their adjacent subdomains through MPI.

In Chapter 1, we introduce the background and meaning for this thesis. We provide the preliminaries which are necessary for the following presentation. We derived the bilinear form of second order elliptical equation and linear elasticity equation.

In Chapter 2, we discuss the weak Galerkin finite elemet method and the bilinear form of linear elasticity equation.The WG finite element method is based on the variational form of equations. It is compatible for general polygons on a finite element computational domain. The computational matrix derived from WG method is symmetric and positive definite. Due to the flexibility of the polynomials basis functions, it's convenient to obtain high order accuracy solutions. The convergence rate for WG method is bounded by the lowest order.

In Chapter 3, we design a novel parallel computing method to efficiently solve elastic equation. The core idea of the WG method for solving linear elastic equation is to replace its gradients after the integration by parts by discrete weak strain and stress tensors. We develop a novel hybrid element which combines the elements of both weak Galerkin (WG) finite element method and continuous Galerkin (CG) finite element method. The new hybrid element inherits the discontinuous feature of the WG method. We insert an arbitrary number of CG elements in one single WG element. The hybrid element provides a second order of accuracy for both linear and nonlinear elastic equation. The superlinear speedup is observed.

In Chapter 4, we develop a novel parallel computing method to efficiently solve linear elasticity problems on unstructured meshes. The core idea of the WG method for

solving linear elasticity equation is to introduce weak strain and stress tensors by using the concept of discrete weak gradients. To enable parallel computation, the computational grid is split into arbitrary number of subdomains. The connection of adjacent subdomains is realized through the balancing domain decomposition with constraints (BDDC) which was originally proposed in [24]. Locally over each subdomain, matrices are constructed for interior and interface quantities separately. Subsequently, interface related matrices are passed over to their adjacent subdomains through MPI. MPI communications are used to help construct a smaller global matrix leaving most of computational operations locally to each processor. The designed WG-BDDC parallel algorithm achieves outstanding scalability by testing over 600 processors. Our numerical results also demonstrate that the WG-BDDC method pos- sesses designed orders of accuracy for both 2nd-order and 3rd-order spatial discretization schemes. Moreover, condition numbers for all test problems are well bounded demonstrating the stability of WG-BDDC method for parallel processing.

In Chapter 5, We conclude the current stage and explore the future potential work.

**Keyword** : 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 Backgrounds

### 1.1.1 Engineering Background

### 1.1.2 Numerical Background

## 1.2 Objectives of this Work

In this work, xxx

# 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 $\mathbf{f}$, we denote the computational domain as $\Omega$ and its continuous boundary as $\Gamma = \partial\Omega$. The governing equation for linear elasticity can be written as

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

$$\mathbf{u} = \hat{\mathbf{u}}, \qquad on \quad \Gamma \tag{2.2}$$

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

$$\sigma(\mathbf{u}) = 2\mu\varepsilon(\mathbf{u}) + \lambda(\nabla \cdot \mathbf{u})\mathbf{I} \tag{2.3}$$

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

$$\lambda = \frac{E\mu}{(1+\mu)(1-2\mu)} \tag{2.4}$$

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

where $E$ is the elasticity modulus and $\mu$ 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 for two function $\mathbf{u}_0$ and $\mathbf{u}_b$ are independent with each other along . The weak function is defined as

$$V_h = \{\mathbf{v} = \{\mathbf{v}_0, \mathbf{v}_b\} : \mathbf{v}_0 \in P_j(T^0), \mathbf{v}_b \in P_l(e), e \subset \partial T\} \tag{2.6}$$

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 $T$:

$$(\nabla_w \mathbf{u}, \mathbf{q})_T = -(\mathbf{u}_0, \nabla \cdot \mathbf{q})_T + \langle \mathbf{u}_b, q \cdot \mathbf{n} \rangle_{\partial T} \tag{2.7}$$

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

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

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

$$\varepsilon_w(\mathbf{u}) = \frac{1}{2}(\nabla_w \mathbf{u} + \nabla_w \mathbf{u}^T) \tag{2.9}$$

Analogously, we can define the weak stress tensor as

$$\sigma_w(\mathbf{u}) = 2\mu \varepsilon_w(\mathbf{u}) + \lambda(\nabla_w \cdot \mathbf{u})\mathbf{I} \tag{2.10}$$

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

$$a(\mathbf{u}, \mathbf{v}) = (\mathbf{f}, \mathbf{v}) \tag{2.11}$$

The above approximation function $\mathbf{u}$ and gradient $\nabla \mathbf{u}$ is not well defined for the discontinuous feature of weak Galerkin method, the new form is

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

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

$$s(\mathbf{u}, \mathbf{u}) = \sum_{T \in \Omega}^{N} h_T^{-1} \langle Q_b \mathbf{u}_0 - \mathbf{u}_b, Q_b \mathbf{v}_0 - \mathbf{v}_b \rangle_{\partial T} \tag{2.13}$$

where $Q_b$ is the projection from the interior unknown variables to boundary unknown variables. Commonly it is taken as 1. The discrete bilinear equation has the assemblage form

$$a(\mathbf{u}_w, \mathbf{u}_w) = \sum_{T \in \Omega}^{N} 2(\mu \varepsilon_w(\mathbf{u}), \varepsilon_w(\mathbf{v}))_T + \sum_{T \in \Omega}^{N} (\lambda \nabla \cdot \mathbf{u}, \nabla_w \cdot \mathbf{v})_T \tag{2.14}$$

## 2.2 Existing Numerical Methods Review

In this section, we present and analyze several most widely used numerical method to solve 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 arised to solve complex elasticity and structural analysis engineering problems in mechanical and aeronautical field. A. Hrennikoff[17], R. Courant[5] 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 so that 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.

Finite element method is an efficient solution to solve partial differential equations. The core idea is to convert the original partial differential equation to a equivalent

bilinear form weak function. Then we partition the computational domain into polygon meshes. In each mesh element, we construct the finite element space. Then the bilinear form is discretized into a summation of finite elemental spaces. The solution is approximated based on the calculation of assembled matrix. More details can be found in [37, 4, 18, 31]

The variational formulation which generated by the governing equation determines the characteristic of the finite element method. To obtain the variational form, many 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 method is inspired from these two method and shares many similarities with them.

### 2.2.2 Discontinuous Galerkin Finite Element Method

### 2.2.3 Mixed Finite Element Method

## 2.3 Weak Galerkin Finite Element Methods

### 2.3.1 Weak Galerkin triangular meshes

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

$$\phi_k = \{\lambda_k, 0\}, \qquad k = 1, 2, 3 \tag{2.15}$$

$$\phi_{3+l} = \{0, \mu_l\}, \qquad l = 1, 2, \cdots, 2N \tag{2.16}$$

where N is the number of element boundaries.

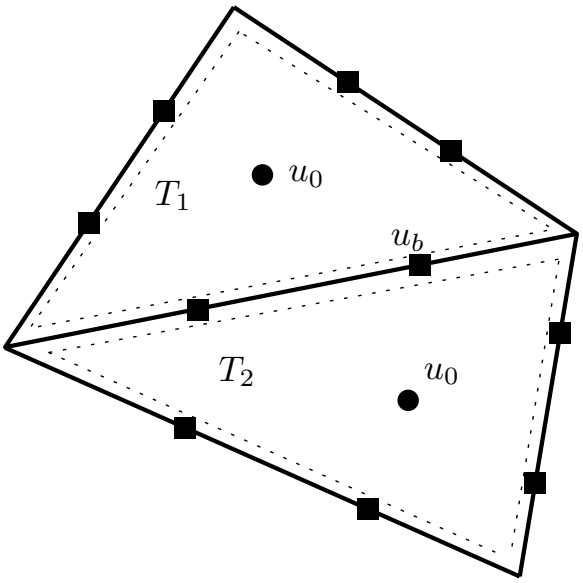


Figure 2.1: Weak Galerkin triangular elements and solution points.

### 2.3.2 Weak Galerkin quadrilateral meshes

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

$$\phi_k = \{\lambda_k, 0\}, \qquad k = 1, 2, 3 \tag{2.17}$$

$$\phi_{3+l} = \{0, \mu_l\}, \qquad l = 1, 2, \cdots, 2N \tag{2.18}$$

where N is the number of element boundaries.

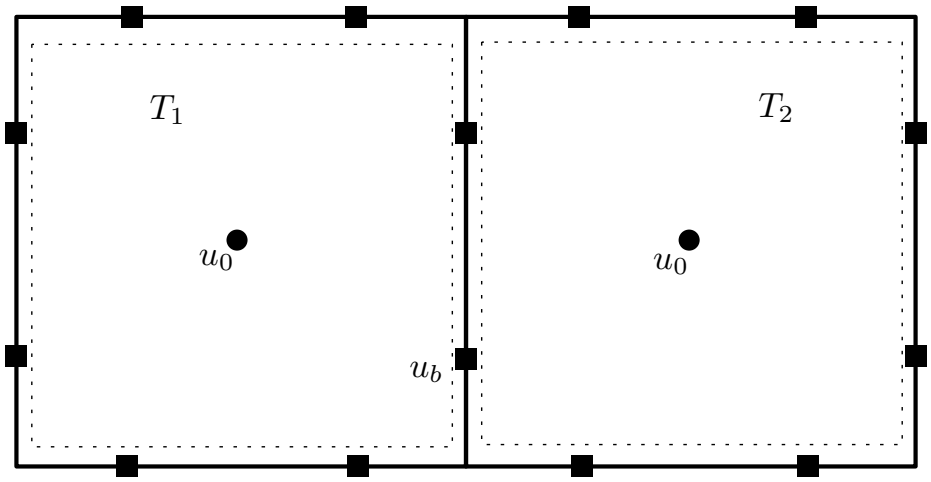


Figure 2.2: Weak Galerkin quadrilateral elements and solution points.

## Chapter 3: Hybrid Weak Galerkin and 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 Junping Wang and Xiu Ye [35]. The core idea of the WG method for solving linear elastic equation is to replace its gradients after the integration by parts by discrete weak strain and stress tensors. We develop a novel hybrid element which combines th 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 insert an arbitrary number of CG elements in one single WG element. The hybrid element provides a second order of accuracy for both linear and nonlinear elastic equation. The superlinear speedup is observed.

The objective of this chapter is to explore a new efficient path to the massively parallel computing method with domain decomposition scheme. Specifically, this combination involves a discontinuous weak Galerkin finite element method[35, 23, 36] and implement a hybrid element which enable multiple continuous Galerkin finite elements inserted into the WG element. The WG method is considered as a newly developed robust numerical method and inherits the locking-free feature for the linear elastic equation[34].

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 is proven robust and has optimal orders of accuracy in spatial discretization for even 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 follows. We introduce the weak Galerkin method and the derivations of the bilinear form of nonlinear elasticity equation. Then we discuss the details of matrix construction and a generic stabilizer. For the numerical method, we present the details of the parallel scheme which including the implicit time marching scheme, namely, the Newmark-Beta method[1]. At the end of this chapter, we show the numerical results of one-dimensional linear and nonlinear elastic equation. The second and third order accuracy 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 $\Omega$ 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 $\mu$ and $\lambda$ 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 $\nu$ is Poisson's ratio.

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

format

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

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

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

The we may 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 the form as

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

the above approximation function $v$ and the gradient $\nabla v$ 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 $Q_b$ the projection parameter bound by the interior and boundary value. 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

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

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

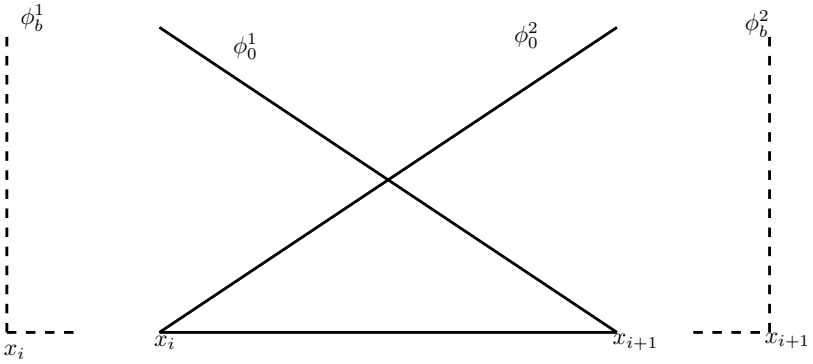


Figure 3.1: Basis function of one dimensional weak Galerkin element

Based on the weak gradient function, the boundary basis functions are independent with the interior basis functions, Therefore, we can insert an arbitrary number of CG elements into the WG element. Each weak Galerkin element is treated as one subarea that neighbored with adjacent elements. The interior CG elements are found mesh level with are corrected by the coarse mesh. 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

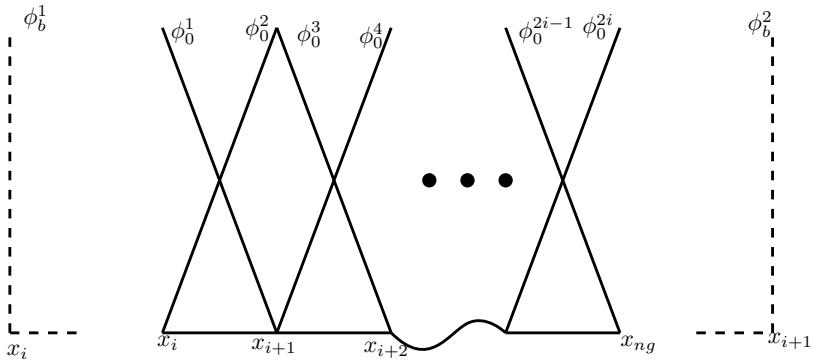


Figure 3.2: Basis function of hybrid WG-CG element, an arbitrary number of CG elements are inserted into WG element

### 3.3 Generic 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

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

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

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

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

$$\begin{bmatrix} M_{00} & 0 \\ 0 & M_{bb} \end{bmatrix} \begin{bmatrix} \ddot{u}_0 \\ \ddot{u}_b \end{bmatrix} + \begin{bmatrix} K_{00} & K_{0b} \\ K_{b0} & K_{bb} \end{bmatrix} \begin{bmatrix} u_0 \\ u_b \end{bmatrix} = \begin{bmatrix} f_0 \\ 0 \end{bmatrix} \tag{3.21}$$

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

### 3.4 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

$$\ddot{u}_0^{n+1} = \frac{(u_0^{n+1} - \tilde{u}_0)}{\beta \Delta t^2} \tag{3.22}$$

$$\ddot{u}_b^{n+1} = \frac{(u_b^{n+1} - \tilde{u}_b)}{\beta \Delta t^2} \tag{3.23}$$

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

$$\tilde{u}_0 = u_0^n + \dot{u}_0^n \Delta t + \frac{1}{2} \Delta t^2 \ddot{u}_0^n (1 - 2\beta) \tag{3.24}$$

$$\tilde{u}_b = u_b^n + \dot{u}_b^n \Delta t + \frac{1}{2} \Delta t^2 \ddot{u}_b^n (1 - 2\beta) \tag{3.25}$$

To reduce the local unknown variable $u_0$ into a global smaller size unknown variable $u_b$, we apply the Schur complement method. The correlation between interior

unknown variables and boundary variables is following

$$u_0^{n+1} = -G_{00}^{-1} G_{0b} u_b^{n+1} + G_{00}^{-1} g_1 \tag{3.26}$$

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

$$u_b^{n+1} = [G_{b0}(-G_{00}^{-1} G_{0b}) + G_{bb}]^{-1} (g_2 - G_{b0}(G_{00}^{-1} g_1)) \tag{3.27}$$

Then we can simplify the above equation into

$$u_b^{n+1} = G' f' \tag{3.28}$$

### 3.5 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

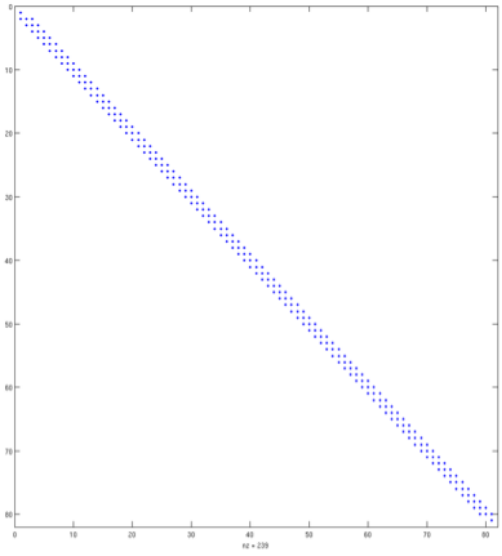

Figure 3.3: Global sparse tri-diagonal stiffness matrix

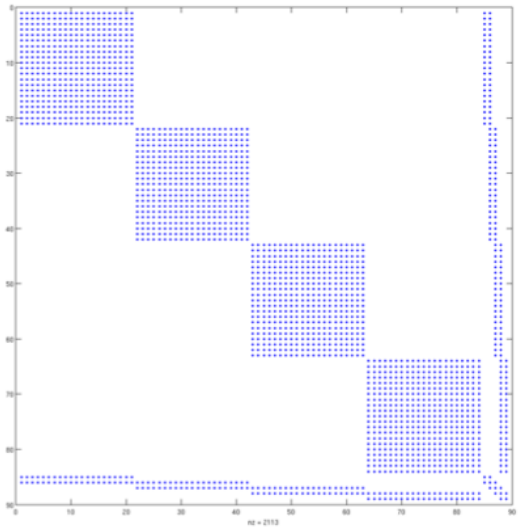

Figure 3.4: Blocked stiffness matrix after domain decomposition

The original large sparse stiffness matrix is compressed in a very small matrix on the right bottom corner. Consequently, the main computational load is split into the calculation of square matrices. Each square matrix represents one single hybrid element. The size of hybrid element is determined by the number of CG elements which inserted into one WG element. We can compare the WG element as the ship and CG elements are the cargo on the board. The entire computational domain is decomposed into multiple subdomains naturally since the discontinuous feature of hybrid WG-CG element.

The parallel computing work flow chart is described as following

1. Each CPU reads in preprocessed file and initiates MPI communication[15].

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.

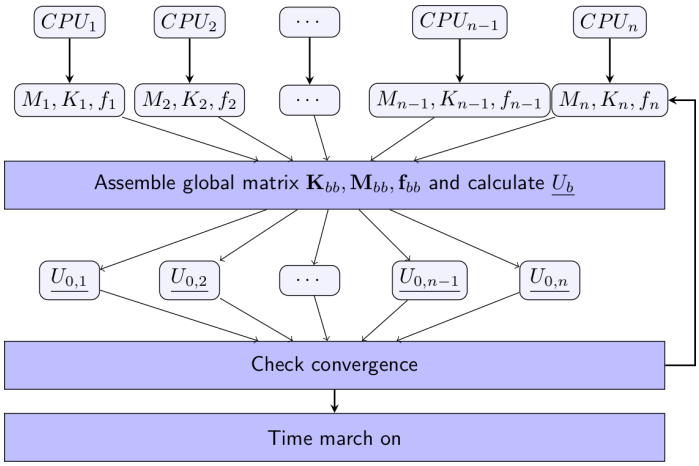


Figure 3.5: Parallel computing workflow chart for WG-CG method

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 very expensive operators including Cholesky transportation, 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 cluster, ColonialOne, with Xeon E5-2650v2 2.6GHz 8-core processors with 128 GB of RAM each.

## 3.6 Numerical Results

### 3.6.1 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 insert 20 CG elements and then observe the accuracy of $u_0$, $u_b$ and the effect of stabilizer.

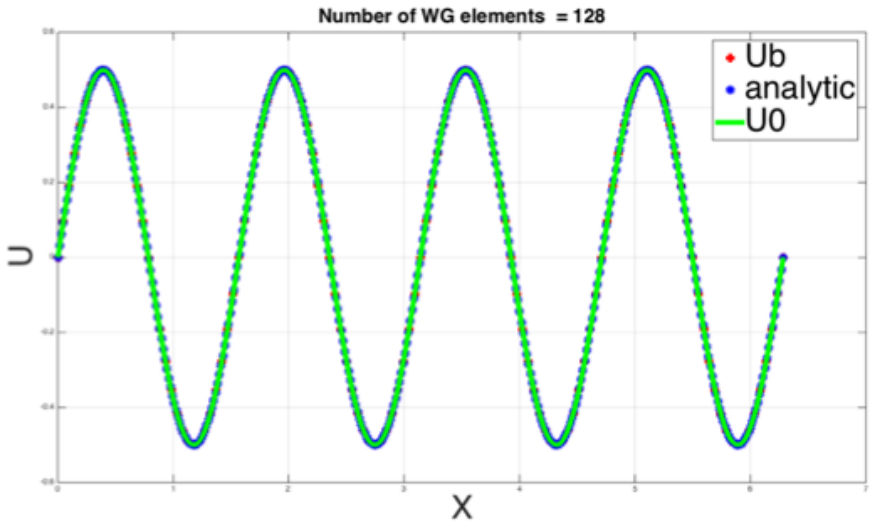


Figure 3.6: Linear elastic equation results for hybrid WG-CG element

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 |

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 probes that the hybrid element has the excellent compatibility for linear elastic equation.

To explore the capability in dynamic problem, 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[19] and define the relative small time step to fit the critical time marching step[6]. For the implicit scheme, we choose the

Newmark-Beta method[16]. For every test case, a constant loading force is applied on the right-hand side as a Neumann boundary condition[30]. Meanwhile, we fix the left-hand side as Dirichlet boundary condition. The results are shown as following charts

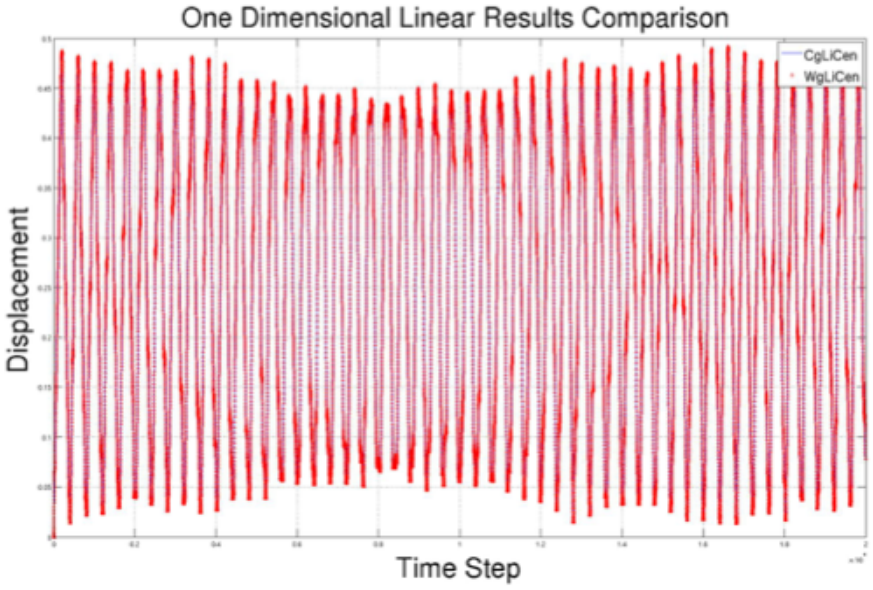


Figure 3.7: The results comparison between CG only and hybrid WG-CG element for explicit scheme

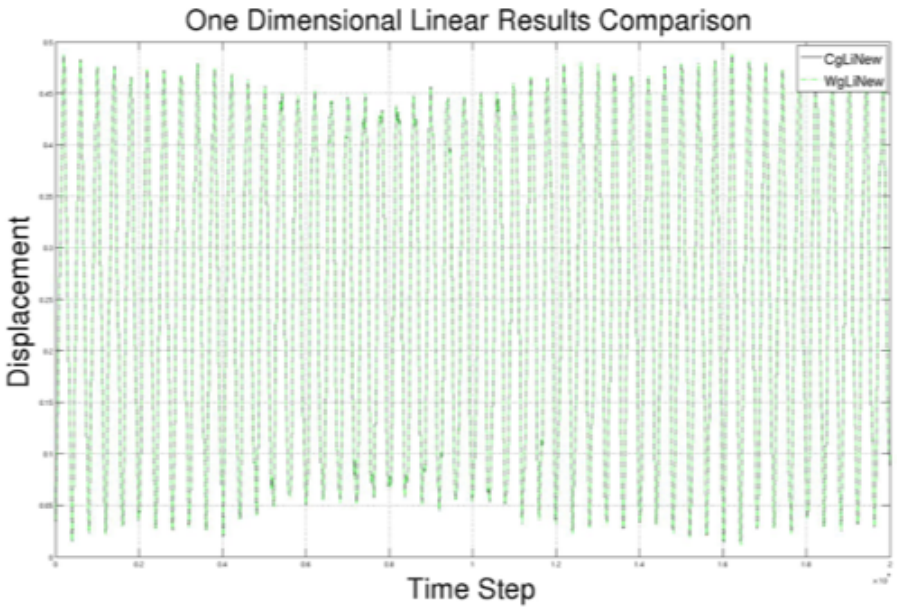


Figure 3.8: The results comparison between CG only and hybrid WG-CG element for implicit scheme

We compare the solutions between our hybrid WG-CG element and classic CG element. In both implicit and explicit time marching scheme, the average difference between two methods is less than $1e-5$. We can safely draw the conclusion that hybrid element achieves an optimal accuracy for not only static case, by also dynamic problems.

### 3.6.2 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.

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 operates 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

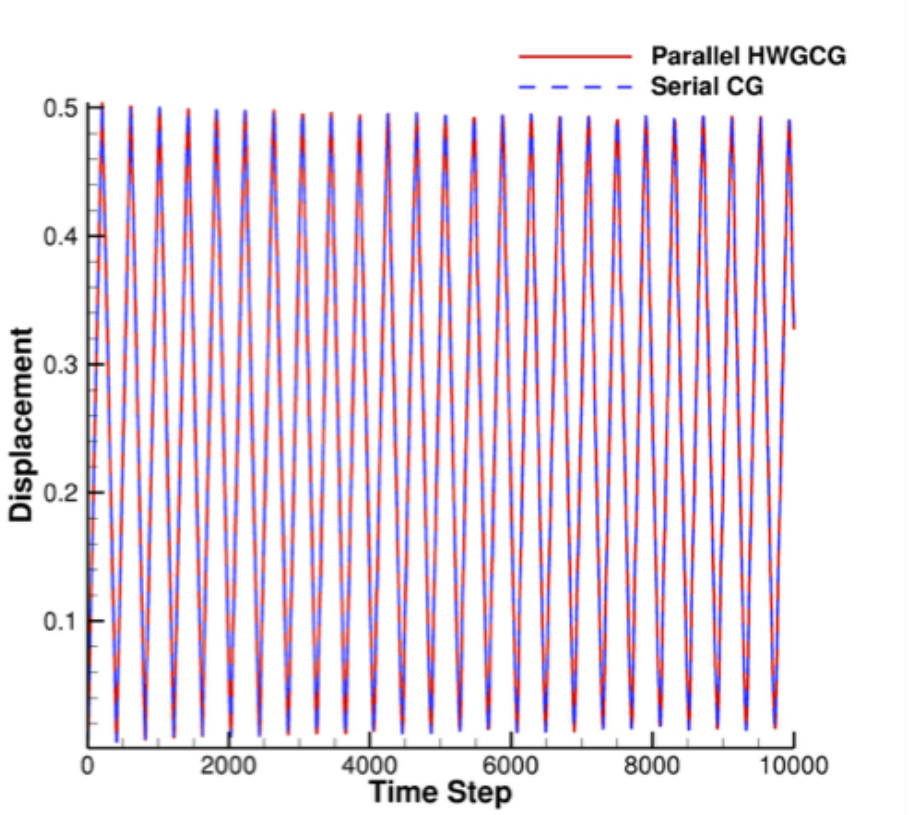


Figure 3.9: The results comparison between CG only and hybrid WG-CG element by using parallel implicit scheme with constant boundary condition

An optimal convergence of the two independent solutions shows an excellent accuracy of the parallel hybrid solver. The average difference is less than $1e-4$, 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

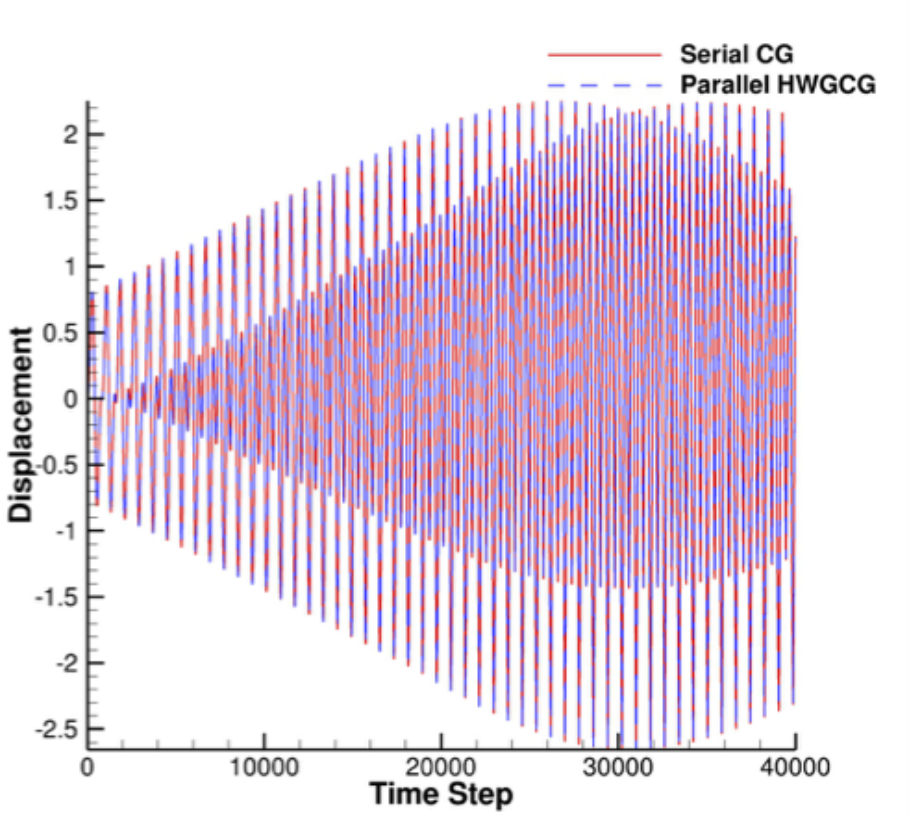


Figure 3.10: The results comparison between CG only and hybrid WG-CG element by using parallel implicit scheme with periodic boundary condition

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. Meanwhile, we gradually increase the number of processor applied for the problem from 10 to 60. The time using graph is plotted as following

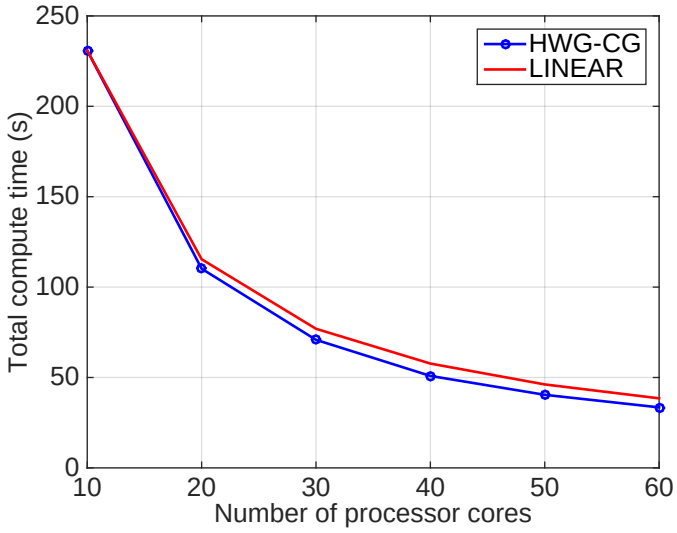


Figure 3.11: Time decreasing .vs. number of processors increasing

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

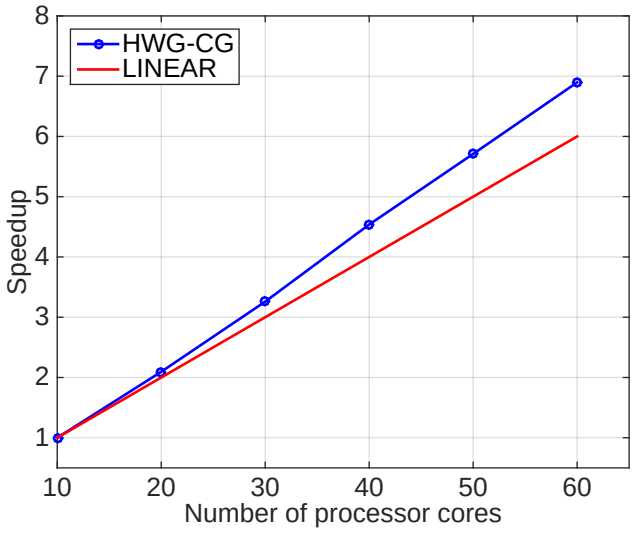


Figure 3.12: Speedup .vs. number of processors increasing

A superlinear speedup is observed from the above test cases. The reason is that the computational effort to solve local matrices decreases nonlienarly when the local problem is divided into a series of small subdomains. The computational expense for converting stiffness matrix decreased cubics to the size of matrix. Meanwhile, the communication overhead growing relatively slow. The time complexity is $O(n^3)$ reducing ans $O(n)$ increasing. Overall, the trade-off benefit for domain decomposition is larger than MPI communication which leads to the superlinear results.

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

$$u = \begin{pmatrix} sin(2\pi x)sin(2\pi y) \\ 1 \end{pmatrix} \tag{3.29}$$

the solution is shown as below

Table 3.2: 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 |

Table 3.3: 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 |

In summary, we present a newly developed hybrid element combine the weak Galerkin finite element and continuous Galerkin finite element. It inherits the discontinuity from WG method and the convenience on implementation from CG method. The implementation of hybrid element for parallel computing is compatible for MPI library platform. The second order accuracy and superlinear speedup are observed

for the hybrid element.

Both geometric linear and nonlinear test cases are studied. The results of new hybrid element provides and optimal convergence which is accordance with WG and CG elements. The hybrid elements proved a strong robustness for Dirichlet and Neumann boundary condition.

We present the unstructured mesh results of two-dimensional solution for linear elasticity equation. We will extend the parallel computing to two dimensional with more advanced technicals. More details of parallel computing for larger problem will be discussed in the following 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 [27, 23, 36, 26] and a non-overlapping domain decomposition scheme, namely, the balancing domain decomposition with constraints (BDDC) [7, 33, 32]. The WG method is considered as a newly developed robust numerical method and inherits the locking-free feature for the linear elastic equation [34].

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 [28, 29]. Wang et al [34] 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.

## 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 [24] 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[14, 13, 20, 12, 22, 21], is an iterative method for solving large finite element problems generated by linear equations. Originally, FETI is proposed to solve the discontinuity when apply domain decomposition on second order elliptic partial differential equations, particularly linear elasticity equations. FETI method partitioned the entire computational domain into two level meshes, the coarse grid and fine grid. Lagrangian multiplier is employed to conquer the discontinuity between each subdomain. For the fine mesh, since local matrix is not positive definite, the pseudo-inverse is applied to convert local matrix.

Then FETI-DP (Dual Primal) is introduced for two-dimensional problems by Farhat and Rixen[11]. In this new method, the unknown variables along 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 as-

semblage. For the rest dual spaces, the constrains are still controlled by Lagrange multipliers. However, this constrain is only at the convergence of the method.

In this section, we begin with an example of a two-dimensional, two partitioned subdomain case. We enforce the Dirichlet boundary condition on the boundary. The original geometry and computational domain is

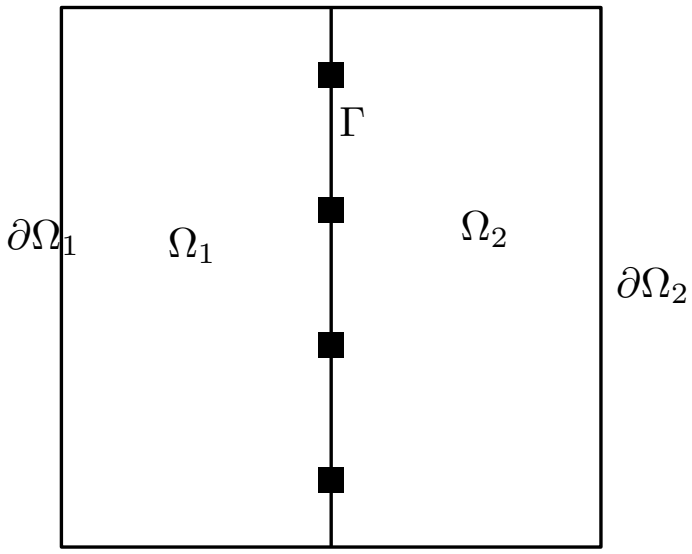


Figure 4.1: Computational domain partitioned into two nonoverlapping subdomains.

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 $\Gamma$.

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 DOFs are represented by $\Omega_1, \Omega_2$ and $\Gamma$ respectively.

Since the interior matrix are square and can be inversed. 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

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

Now we can find that the original large matrix $A$ is reduced to a small global matrix $S^{(j)}$ which times unknown variable vector along the interface. The interior unknown variables can be recovered locally based 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 $\lambda_\Gamma = \lambda_\Gamma^{(1)} = -\lambda_\Gamma^{(2)}$. $\lambda_\Gamma$ is an unknown flux vector, namely, Lagrange multiplier. We can solve this linear system following previous fashion that

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

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

to maintain the continuity along the interface, the condition is set as

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

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 can partitioned the subdomains with floating constraints.

From the above figure, the interface edge is continued to split into $(u_1, \cdots, u_m, \cdots, u+$

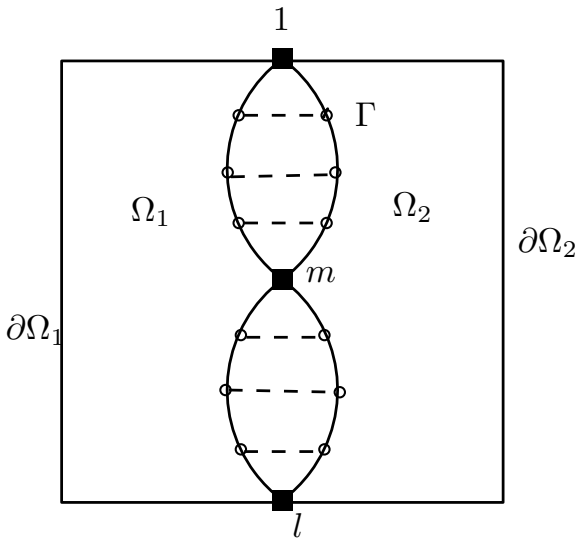


Figure 4.2: Computational domain partitioned into two nonoverlapping subdomains with floating constaint.

$l$) then 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 variable along the interface of two nonoverlapping subdomian can be elaborate 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 the columns representing the new space of interface unknown variables. The original interface is divided into two parts. One part is that the value of unknown variables has a value on its own subdomain. The other part is that the function has zero interface averages. The the problem can be rewrite 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 find out that the change is an individual interface edge local procedure. We can assume that the unknown variables on each subdomain have been changed when using primal edges or faces. In last figure, 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 $\lambda$ 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, we can find out that the Lagrange multiplier is an external unknown vector and requires lots of computational effort to assemble the matrix. When we 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 restore the computing a weighted average of the vectors. Therefore, an assembled residual of resulting vector needs to be compute. 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 Balancing Domain Decomposition by Constraints

The balancing domain decomposition method was introduced by Mandel [25]. 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. The main function is to solve problems arises from the finite element method, including WG method. It is

inspired by FETI-DP method of Farhat et al [13, 11] which has been extended multi-dimension varying problems. 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 primal alternative to FETI-DP domain decomposition method [22]. In this paper, we only consider the corner connections of subdomain as the only constraints. The substructure spaces, coarse space, and the substructure bilinear forms are same as Mandels paper. 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

$$\hat{S}u_\Gamma = \sum_{j=1}^{N} R_\Gamma^{(j)} g_\Gamma \tag{4.25}$$

where

$$\tilde{S} = \tilde{R}_\Gamma^T \tilde{S}_\Gamma \tilde{R}_\Gamma \tag{4.26}$$

In the BDDC algorithm, a two-level Neumann-Neumann type preconditioner for solving this interface problem is as above equation. 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 [7, 8, 25] has the form

$$M_{BDDC}^{-1} = R_{D,\Gamma}^T T R_{D,\Gamma} \tag{4.27}$$

for the coarse-level $T$ is defined by

$$T = \Psi(\Psi^T S \Psi)^{-1} \Psi^T \tag{4.28}$$

the coarse level basis function vector is defined by

$$\Psi = \begin{pmatrix} \Psi^{(1)} \\ \vdots \\ \Psi^{(N)} \end{pmatrix} \tag{4.29}$$

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

$$\begin{pmatrix} S^{(j)} & C^{(j)^T} \\ C^{(j)} & 0 \end{pmatrix} \begin{pmatrix} \Psi^{(j)} \\ V^{(j)} \end{pmatrix} = \begin{pmatrix} 0 \\ R_{\Pi}^{(j)} \end{pmatrix} \tag{4.30}$$

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

$$M_{BDDC}^{-1} = R_{D,\Gamma}^T \{ R_{\Gamma\Delta}^T S_{\Delta}^{-1} R_{\Gamma\Delta} + \Psi(\Psi^T S \Psi)^{-1} \Psi^T \} R_{D,\Gamma} \tag{4.31}$$

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

$$\begin{pmatrix} A_{II}^{(j)} & A_{\Delta I}^{(j)} & A_{\Pi I}^{(j)} \\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)} & A_{\Pi\Delta}^{(j)} \\ A_{\Pi I}^{(j)} & A_{\Pi\Delta}^{(j)} & A_{\Pi\Pi}^{(j)} \end{pmatrix} \begin{pmatrix} u_I^{(j)} \\ \Psi_{\Delta}^{(j)} \\ R_{\Pi}^{(j)} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ S_{\Pi\Pi}^{(j)} R_{\Pi}^{(j)} \end{pmatrix} \tag{4.32}$$

where

$$\Psi^{(j)} = \begin{pmatrix} \Psi_{\Delta}^{(j)} \\ R_{\Pi}^{(j)} \end{pmatrix} \tag{4.33}$$

$$\Psi^{(j)} = \begin{pmatrix} -\begin{pmatrix} 0 & I_{\Delta}^{(j)} \end{pmatrix} \begin{pmatrix} A_{II}^{(j)} & A_{I\Delta}^{(j)} \\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)} \end{pmatrix}^{-1} \begin{pmatrix} A_{I\Pi}^{(j)} \\ A_{\Pi\Delta}^{(j)} \end{pmatrix} R_{\Pi}^{(j)} \\ R_{\Pi}^{(j)} \end{pmatrix} \tag{4.34}$$

and

$$S_{\Pi\Pi}^{(j)} = A_{\Pi\Pi}^{(j)} - \begin{pmatrix} A_{\Pi I}^{(j)} & A_{\Pi\Delta}^{(j)} \end{pmatrix} \begin{pmatrix} A_{II}^{(j)} & A_{I\Delta}^{(j)} \\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)} \end{pmatrix}^{-1} \begin{pmatrix} A_{I\Pi}^{(j)} \\ A_{\Delta\Pi}^{(j)} \end{pmatrix} \tag{4.35}$$

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

$$\Psi^T S \Psi = \sum_{j=1}^{N} {\Psi^{(j)}}^T S^{(j)} \Psi^{(j)} \tag{4.36}$$

this equals to

$$\sum_{j=1}^{N} {R_{\Pi}^{(j)}}^T S_{\Pi\Pi}^{(j)} R_{\Pi}^{(j)} = S_{\Pi\Pi} \tag{4.37}$$

where $\Psi$ represents the interface vectors with distributed coarse-level variables. $\Phi$ represents the DOFs vectors on coarse-level.

If we have the assumption that

$$\delta_j^{\dagger}(x) = 1, \quad x \in \Gamma \tag{4.38}$$

$$R_{D,\Gamma}^T \Psi = \tilde{R}_{D,\Gamma}^T \Phi \tag{4.39}$$

therefore

$$M_{BDDC}^{-1} = R_{D,\Gamma}^T R_{\Gamma\Delta}^T S_\Delta^{-1} R_{\Gamma\Delta} R_{D,\Gamma} + R_{D,\Gamma}^T \Psi \left( \Psi^T S \Psi \right)^{-1} \Psi^T R_{D,\Gamma} \tag{4.40}$$

after substitute $\Psi$ by $\Phi$, we can derive

$$M_{BDDC}^{-1} = \tilde{R}_{D,\Gamma}^T R_{\Gamma\Delta}^T S_\Delta^{-1} R_{\Gamma\Delta} \tilde{R}_{\Delta,\Gamma} + \tilde{R}_{D,\Gamma}^T \Phi S_{\Pi\Pi}^{-1} \Phi^T \tilde{R}_{D,\Gamma} \tag{4.41}$$

we can simplify the equation by

$$M_{BDDC}^{-1} = \tilde{R}_{D,\Gamma}^T \tilde{S}_\Gamma^{-1} \tilde{R}_{D,\Gamma} \tag{4.42}$$

the preconditioned BDDC operator is designed by

$$\tilde{R}_{D,\Gamma}^T \tilde{S}_\Gamma^{-1} \tilde{R}_{D,\Gamma} \tilde{R}_\Gamma^T \tilde{S}_\Gamma \tilde{R}_\Gamma \tag{4.43}$$

## 4.2 WG-BDDC Method

In this section, we discuss the details of design the parallel computing scheme by combining WG method with BDDC method.

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:

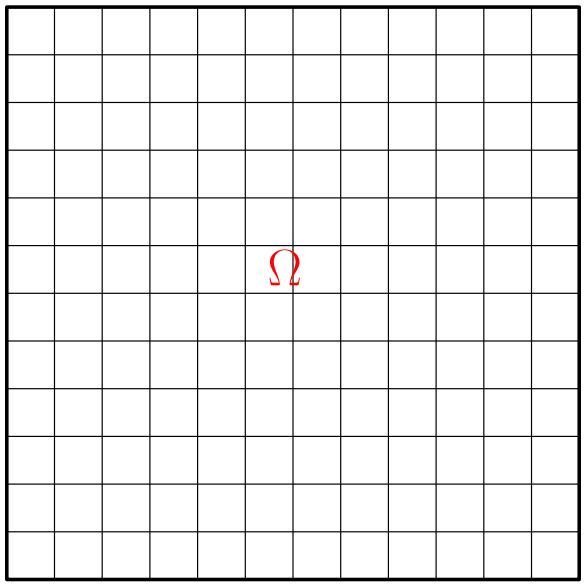


Figure 4.3: The total computational domain.

1. Schur Complement [9] of problems on each subdomain will eliminate all the interior unknowns and only retain the unknowns on the interface of . Denote the interface by .

2. Reduce the unknowns on the interface to construct the preconditioner.

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

In the second step, the solid dot represents the unknown variables along the interface. They are shared by adjacent subdomains and should be calculated in the global matrix through Schur complement method. Even though the number of DOF in global matrix is drastically decreased, the communication overhead and scale of global matrix are still not satisfied the standard for high performance computing. In this way, we shall continuously split the interface into two spaces.

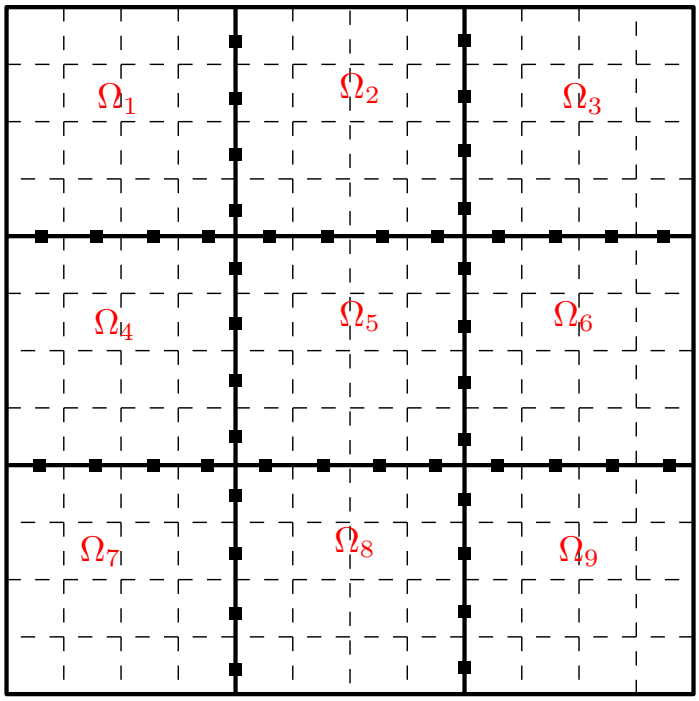


Figure 4.4: After the Schur complement method the computational domain becomes interior and interface.

In the third graph, we split the interface into primal and dual spaces. The circle represents the unknown variables belongs to dual space. They are calculated only in local matrices. We bridged the information from dual space to primal space through preconditioner. The remain dots are unknown variables in the primal spaces. They are the only information shall be communicated and calculated through MPI functions. Now the global matrix has been decreased to an optimal level which benefit us significantly in speedup test.

One significant feature of this method is that when the number of subdomains increasing, the condition number of this method is bounded under the circumstance that an appropriate choice of the coarse DOFs and with regular subdomain shapes. The condition number grows only very slowly with the number of elements in each subdomain.

The number of iterations is also bounded in the same fashion. Meanwhile, the method scales well with the number of subdomains and size of the problem.

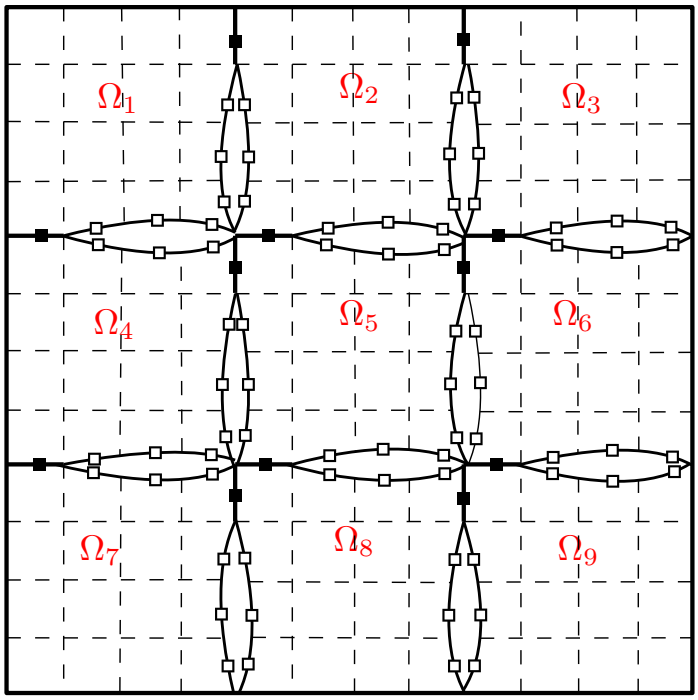


Figure 4.5: BDDC computational domain with only one cell boundary.

### 4.3 Schur Complement Method for subdomain $\Omega_j$

Denote the weak Galerkin solution on each subdomain $\Omega_j$ For the consistency with Equation (1), we will use $u_h$ to represent the weak Galerkin solution on each $\Omega_j$.

To define the Schur complement system, the degrees of freedom $u_h$ on each subdomain $\Omega_j$ are partitioned into interior $u_I$and interface $u_\Gamma$ parts . Then, we can rewrite the unknown variable function as

$$u_h = [u_{I0}, u_{Ib}, u_{\Gamma b}] \tag{4.44}$$

and denote $u_I = [u_{I0}, u_{Ib}]$, meanwhile, the $u_\Gamma = u_{\Gamma b}$ . Consequently, the local Schur complements can be applied to each subdomain in the following form

$$\begin{pmatrix} A_{II} & A_{\Gamma I}^T \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} \times \begin{pmatrix} u_I \\ u_\Gamma \end{pmatrix} \tag{4.45}$$

To define the Schur complement system, the DOFs on each subdomain are partitioned by interior and interface categories. Now the unknown function becomes $u_h = [u_{I0}, u_{Ib}, u_{\Gamma b}]$, $v_h = [v_{I0}, v_{Ib}, v_{\Gamma b}]$ and denote the interior unknown variable $u_i = [u_{I0}, u_{Ib}]$ on the interface we have $u_\Gamma = u_{\Gamma b}$ . For the assistant function the same rule applied to the assistant function $v_I = [v_{I0}, v_{Ib}]$ , $v_\Gamma = v_{\Gamma b}$ . The matrix form includes the assistant function should be following

$$\begin{pmatrix} A^u_{II} & (A^u_{\Gamma I})^T & 0 & 0 \\ A^u_{\Gamma I} & A^u_{\Gamma\Gamma} & 0 & A^{uv}_{\Gamma\Gamma} \\ 0 & 0 & A^v_{II} & (A^v_{\Gamma I})^T \\ 0 & A^{uv}_{\Gamma\Gamma} & A^v_{\Gamma I} & A^v_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} u_I \\ u_\Gamma \\ v_I \\ v_\Gamma \end{pmatrix} \tag{4.46}$$

then we apply Schur complement method to eliminate the interior unknowns which will give the following equations

The interface stiffness matrix has the form as following

$$S^j_{\Gamma\Gamma} = A^{(j)}_{\Gamma\Gamma} - \left[A^{(j)}_{\Gamma I}\right] \times \left[A^{(j)}_{II}\right]^{-1} \times \left[A^{(j)}_{\Gamma I}\right]^T \tag{4.47}$$

The loading force along the interface has the form

$$f^{(j)}_\Gamma = b^{(j)}_\Gamma - \left[A^{(}_{\Gamma I} j)\right] \times \left[A^{(j)}_{II}\right]^{(-1)} \times b^{(j)}_I \tag{4.48}$$

Then denote the assembled matrix form

$$S_{\Gamma\Gamma} = \sum_{j=1}^{N} R^{(j)^T}_\Gamma S^{(j)}_\Gamma R^{(j)}_\Gamma \tag{4.49}$$

where $R^j_\Gamma$ is the mapping vector to convert unknown variables between $\Gamma$ global interface to $\Gamma_i$ interfaces on subdomains $\Omega_j$

Therefore, the global interface problem is constructed as

$$S_{\Gamma\Gamma}\begin{pmatrix} u_\Gamma \\ v_\Gamma \end{pmatrix} = \begin{pmatrix} f_\Gamma^u \\ f_\Gamma^v \end{pmatrix} \tag{4.50}$$

### 4.4 BDDC Preconditioner

Now, we eliminate most of the continuity across the interfaced, refers to Fig 5, and construct the BDDC preconditioner for the inverse of matrix $S_\Gamma$.

In our BDDC formulation, the primal constraints are introduced over edges/faces. To define the BDDC preconditioner for the Schur complement problem, the interface space $\Lambda_\Gamma^{(j)}s$ 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:

$$\hat{\Lambda}_\Gamma = \hat{\Lambda}_\Pi \oplus (\sum_{i=1}^{N} \Lambda_\Delta^{(j)}) \tag{4.51}$$

where $\hat{\Lambda}_\Pi$ is the assembled global primal space, single valued on $\Gamma$, 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, $\hat{\Lambda}_\Gamma$, to precondition the Schur complement problem whose solution lies in the fully assembled space $\hat{\Lambda}_\Gamma$.

The key component of BDDC preconditioner [10]:

- An averaging operator which restricts functions from $\Lambda_\Gamma$ to $\hat{\Lambda}_\Gamma$
- A positive scaling factor $\delta_i^\dagger(e_k)$ is defined for each interface $e_k$ of the subdomain $\Omega_j$ 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 $R_{D,\Gamma} := D_\Gamma R_\Gamma$

The BDDC preconditioner has the following form that

$$M^{-1}_{\Gamma_{BDDC}} = R^T_{D,\Gamma} \tilde{S}^{-1}_{\Gamma\Gamma} R_{D,\Gamma} \tag{4.52}$$

We interpret the above equation by using the unknown variable function $u_\Gamma = [u_r, u_c]^T$ and the matrix can be written as

$$M = \begin{pmatrix} S^{(1)}_{rr} & 0 & 0 & \cdots & 0 & S^{(1)}_{rc} \\ 0 & S^{(2)}_{rr} & 0 & \cdots & 0 & S^{(2)}_{rc} \\ 0 & 0 & S^{(3)}_{rr} & \cdots & 0 & S^{(3)}_{rc} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & S^{(N)}_{rr} & S^{(N)}_{rc} \\ S^{(1)}_{cr} & S^{(2)}_{cr} & S^{(3)}_{cr} & \cdots & S^{(N)}_{cr} & S_{cc} \end{pmatrix} \tag{4.53}$$

the subscript $c$ represents the unknown variables of constraints. The $r$ represents the rest of unknown variables in computational subdomains.

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 BDDC method can be written as the form with preconditioner as

$$M^{-1}_{\Gamma_{BDDC}} S_{\Gamma\Gamma} u_\Gamma = M^{-1}_{\Gamma_{BDDC}} f_\Gamma \tag{4.54}$$

## 4.5 WG-BDDC Method Implementation

In the Equaion (28), we can expand the constraints matrix as

$$S_{cc} = \sum_{i=1}^{N} A^{(j)}_{\Pi\Pi} \tag{4.55}$$

meanwhile, the rest unknown variables matrix can be written in

$$S_{rr}^{(i)} = \begin{bmatrix} A_{II}^{(j)} & A_{I\Delta}^{(}j) \\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)} \end{bmatrix} \tag{4.56}$$

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

$$\hat{R}_{D,\Gamma}^T \{ R_{\Gamma,\Delta}^T ( \sum_{j=1}^{N} \begin{bmatrix} 0 & R_{\Delta}^{(j)^T} \end{bmatrix} \begin{bmatrix} A_{II}^{(j)} & A_{I\Delta}^{(j)} \\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ R_{\Delta}^{(j)} \end{bmatrix} ) R_{\Gamma\Delta} + \Phi S_{\Pi}^{-1} \Phi^T \} \hat{R}_{D,\Gamma} \tag{4.57}$$

with

$$\Phi = R_{\Gamma\Pi}^T - R_{\Gamma\Delta}^T \sum_{j=1}^{N} \begin{bmatrix} 0 & R_{\Delta}^{(i)^T} \end{bmatrix} \begin{bmatrix} A_{II}^{(j)} & A_{I\Delta}^{(j)} \\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)} \end{bmatrix}^{-1} \begin{bmatrix} A_{\Pi I}^{(j)^T} \\ A_{\Pi\Delta}^{(j)\ T} \end{bmatrix} R_{\Pi}^{(j)} \tag{4.58}$$

and

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

here the $S_{\Pi}$ is the global coarse system matrix.

The preconditioned conjugate gradient is applied to solving above linear system. Theoretically, the condition number should be bounded as

$$\kappa(M_{\Gamma_{BDDC}}^{-1} \hat{S}_{\Gamma\Gamma}) \leq C(1 + log(\frac{kH}{h}))^2 \tag{4.60}$$

for a second order elliptic problem. The constant $C$ is independent of solution order, $p$, element size $h$, and the subdomain size $H$ . Thus, the condition number and hence number of iteration required to converge are independent of the number of subdomains and only weakly dependent on the solution order and the size of subdomains.

## 4.6 Preconditioned Conjugate Gradient Method

The conjugate gradient (CG) method is a well-known iterative 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 [3] to iteratively solving the symmetric saddle point problems. It inherits all great features of CG method and extends it to a higher level. 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 PCG method is discussed by the following chart step by step. In terms of preconditioned, the major effort is to assemble the global preconditioner matrix. Then CG similar method is applied to solve the small global matrix. Thus, we can obtain the global corner solution with minimum overhead. Since both global and local matrices are sparse, the open source library LAPACK/BLAS [2] benefits the matrices calculation substantially.

The algorithm of PCG method is following:

**Algorithm 1:** Preconditioned Conjugate Gradient Algorithm

**Input** : $r_0 := b - Ax_0$

$z_0 := M^{-1} r_0$

$p_0 := z_0$

$k := 0$

Repeat ;

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

$$x_{k+1} := x_k + \alpha_k p_k$$

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

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

    return $x_{k+1}$;

**else**

$$z_{k+1} := M^{-1} r_{k+1}$$

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

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

$$k := k + 1$$

**end**

The condition number is calculated from Lanczos matrix. The global preconditioner matrix $M$ is transfered into a tridiagonal matrix $T_{mm}$. When the $m$ is equal to the dimension of $M$, $T_{mm}$ is similar to $M$. Then we calculate the eigenvalues of

$T_{mm}$ and obtain the condition number from calculating the ratio of the maximum and minimum eigenvalue.

## 4.7 Parallel Computing Scheme

Message Passing Interface (MPI) [15] 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:

1. MPI communicator initiates work and distribute the parameters and mesh data to all the processors.

2. In every processor, the connectivity is analyzed and the local elemental matrices are constructed.

3. Through MPI subprograms, the local matrices are communicated and global preconditioner is constructed on every processor.

4. The global problem, whose size is significantly small, is calculated on every processor with PCG linear solver.

5. The global solution is reduced to each processor for the local solution recovery.

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

## 4.8 Numerical Results

### 4.8.1 Poisson Equation

The WG element can choose different order of basis function in weak gradient equation. Hence, we test the combination order of interior, boundary and weak

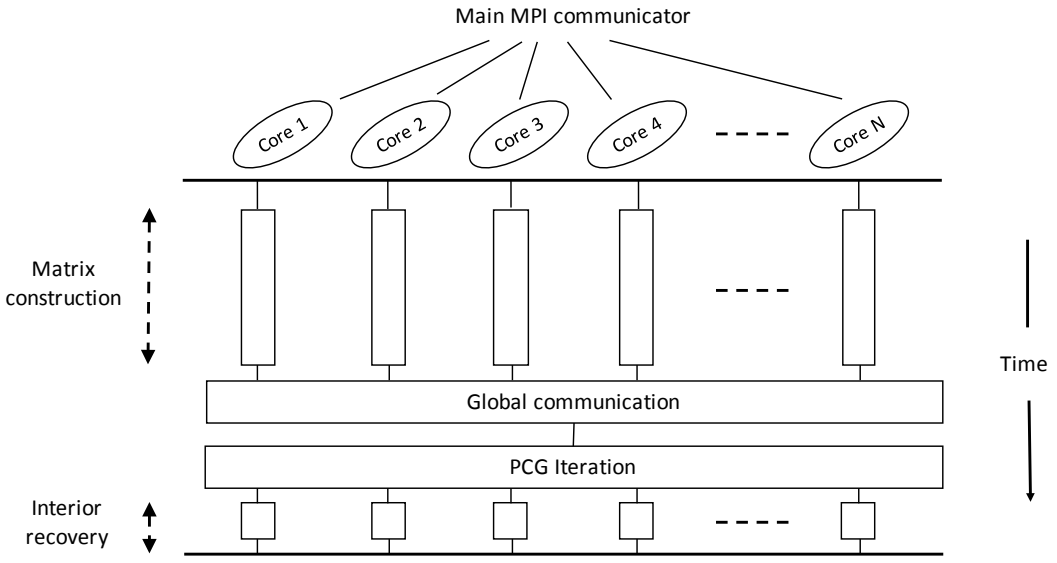


Figure 4.6: Parallel computing work flow.

gradient shape functions.

The Poisson equation $-\nabla \cdot (\nabla \mathbf{u}) = \mathbf{f}$ is considered in the test. Let $\Omega = (0,1) \times (0,1)$, $a = I$, and $f$ 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 firstly decomposed into $N \times N$ subdomains as the coarse mesh with length $H = 1/N$. In every subdomain, all elements are further triangulated into a $2n \times n$ 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 $10^{-6}$. The $L^2$ error is calculated by using the equation $e_{L^2} = \sqrt{\sum_{k=1}^{n}(u - u_t)^2}$

The first test is implemented for weak Galerkin element $u_0 \in P_k, u_b \in P_{k-1}$, and $\nabla_w u \in P_{k-1}$. 4.1 shows the condition number of lanczos matrix and the iteration number in PCG solver. From the 4.1, we can see that the condition number is independent of the number of subdomain, while it depends on $H/h$ as $(1 + log(\frac{H}{h}))^2$.

Table 4.1: Performance with $P_k P_{k-1} P_{k-1}^2$.

| #sub | $k=1$ and $H/h=8$ | | | | $H/h$ | $k=1$ and #sub=64 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Cond. | Iter. | $L^2$-error | $O$ | | Cond. | Iter. | $L^2$-error | $O$ |
| $4\times 4$ | 2.217 | 5 | 1.6013e-3 | - | 4 | 1.722 | 7 | 1.6013e-3 | - |
| $8\times 8$ | 2.390 | 9 | 3.9939e-4 | 2.0 | 8 | 2.390 | 9 | 3.9939e-4 | 2.0 |
| $16\times 16$ | 2.335 | 8 | 9.9789e-5 | 2.0 | 16 | 3.245 | 10 | 9.9789e-5 | 2.0 |
| $32\times 32$ | 2.325 | 8 | 2.4944e-5 | 2.0 | 32 | 4.239 | 11 | 2.4944e-5 | 2.0 |
| #sub | $k=2$ and $H/h=8$ | | | | $H/h$ | $k=2$ and #sub=64 | | | |
| | Cond. | Iter. | $L^2$-error | $O$ | | Cond. | Iter. | $L^2$-error | $O$ |
| $4\times 4$ | 3.528 | 8 | 7.1456e-5 | - | 4 | 2.900 | 10 | 7.1456e-5 | - |
| $8\times 8$ | 3.803 | 10 | 8.9214e-6 | 3.0 | 8 | 3.803 | 10 | 8.9214e-6 | 3.0 |
| $16\times 16$ | 3.768 | 10 | 1.1150e-6 | 3.0 | 16 | 4.957 | 12 | 1.1150e-6 | 3.0 |
| $32\times 32$ | 3.758 | 10 | 1.3938e-7 | 3.0 | 32 | 6.218 | 13 | 1.3938e-7 | 3.0 |

Meanwhile, the communication between each adjacent subdomain does not introduce any error to the results. With the increasing of number of subdomains, we obtain stable second and third order of results. The iteration number of global matrix increases slowly to the number of subdomains.

The second test is the weak Galerkin element with order $u_0 \in P_k$, $u_b \in P_k$ and $\nabla_w \in P_{k-1}$. The condition number has the identical pattern of the theoretical convergence rate. Comparing with the first example, we can find the convergence rates and accuracy have optimal agreement to the degree of polynomial in $u_0$. The parallel scalability is up to 1,000 processors.

In the third test, we implement the weak Galerkin element $u_0 \in P_k$, $u_b \in P_k$ and $\nabla_w u \in P_k$. The $P_k P_k P_k^2$ element deliveries the most accurate result among all three types of element. The reason is that all three shape functions of interior variable, boundary variable and weak gradient have the same order.

Table 4.2: Performance with $P_k P_k P_{k-1}^2$.

| #sub | $k=1$ and $H/h=8$ | | | | $H/h$ | $k=1$ and #sub=64 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Cond. | Iter. | $L^2$-error | $O1$ | | Cond. | Iter. | $L^2$-error | $O$ |
| $4\times 4$ | 2.451 | 7 | 1.0109e-3 | - | 4 | 1.968 | 8 | 1.0109e-3 | - |
| $8\times 8$ | 2.648 | 9 | 2.5117e-4 | 2.0 | 8 | 2.648 | 9 | 2.5117e-4 | 2.0 |
| $16\times 16$ | 2.629 | 9 | 6.2696e-5 | 2.0 | 16 | 3.529 | 10 | 6.2696e-5 | 2.0 |
| $32\times 32$ | 2.617 | 9 | 1.5668e-5 | 2.0 | 32 | 4.619 | 12 | 1.5668e-5 | 2.0 |
| #sub | $k=2$ and $H/h=8$ | | | | $H/h$ | $k=2$ and #sub=64 | | | |
| | Cond. | Iter. | $L^2$-error | $\lambda_1$ | | Cond. | Iter. | $L^2$-error | $\lambda_1$ |
| $4\times 4$ | 3.805 | 8 | 6.6333e-5 | - | 4 | 3.926 | 11 | 6.6333e-5 | - |
| $8\times 8$ | 4.003 | 12 | 8.2709e-6 | 3.0 | 8 | 4.003 | 12 | 8.2709e-6 | 3.0 |
| $16\times 16$ | 3.943 | 12 | 1.0334e-6 | 3.0 | 16 | 5.084 | 13 | 1.0334e-6 | 3.0 |
| $32\times 32$ | 3.917 | 12 | 1.2917e-7 | 3.0 | 32 | 6.329 | 13 | 1.2918e-7 | 3.0 |

### 4.8.2 Linear Elastic Equation

We consider the linear elastic equation (1) in the square domain $\Omega = (0,1)^2$ which is decomposed into uniform square subdomains with size $H$. For each subdomain, it is partitioned into uniform quadrilateral mesh with size $h$. The exact solution is given by

$$u = \begin{pmatrix} sin(2\pi x)sin(2\pi y) \\ 1 \end{pmatrix} \tag{4.61}$$

We test the performance of WG-BDDC on cluster and present the speedup figure which indicates the superlinear acceleration and good scalability.

The blue dot line called WGDD represents WG-BDDC method. we can see from the above figures that the superlinear speedup is captured within the increasing of the number of subdomains. The main concern for BDDC method is that the balance between global matrix, the preconditioner, and the local matrices. It is important to estimate the size of preconditioner and choose proper number of processors.

we report a novel parallel computing method. This method integrated the newly

Table 4.3: Case 1: Performance with $P_k P_k P_k^2$.

| #sub | $k=1$ and $H/h=8$ | | | | $H/h$ | $k=1$ and #sub=64 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Cond. | Iter. | $L^2$-error | $O$ | | Cond. | Iter. | $L^2$-error | $O$ |
| $4\times 4$ | 3.671 | 8 | 2.1451e-4 | - | 4 | 3.024 | 10 | 2.1451e-4 | - |
| $8\times 8$ | 3.965 | 10 | 5.2129e-5 | 2.0 | 8 | 3.965 | 10 | 5.2129e-5 | 2.0 |
| $16\times 16$ | 3.934 | 10 | 1.2937e-5 | 2.0 | 16 | 5.153 | 12 | 1.2937e-5 | 2.0 |
| $32\times 32$ | 3.922 | 10 | 3.2281e-6 | 2.0 | 32 | 6.472 | 14 | 3.2281e-6 | 2.0 |
| #sub | $k=2$ and $H/h=8$ | | | | $H/h$ | $k=2$ and #sub=64 | | | |
| | Cond. | Iter. | $L^2$-error | $O$ | | Cond. | Iter. | $L^2$-error | $O$ |
| $4\times 4$ | 4.620 | 8 | 6.7627e-6 | - | 4 | 3.859 | 11 | 6.7628e-6 | - |
| $8\times 8$ | 4.987 | 12 | 7.8998e-7 | 3.0 | 8 | 4.987 | 12 | 7.8998e-7 | 3.0 |
| $16\times 16$ | 4.921 | 12 | 9.6925e-8 | 3.0 | 16 | 6.235 | 13 | 9.6931e-8 | 3.0 |
| $32\times 32$ | 4.901 | 12 | 1.2058e-8 | 3.0 | 32 | 7.673 | 15 | 1.2060e-8 | 3.0 |

Table 4.4: Case 1: Performance with $P_1 P_1$, $\lambda = 1, \mu = 0.5$

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

developed weak Galerkin finite element method with balancing domain decomposition with constraints. The MPI library is set up for information communication between each processor. The optimal convergence rate and promising scalability of this WG-BDDC method are observed.

To this method, it is convenient to implement high order element. We have multiple choices on interior, boundary and gradient basis functions which is an obvious flexibility to numerical calculation. We can design the element type based on our need. From the optimal convergence rate, we can conclude that this method is robust and highly compatible to different element orders.

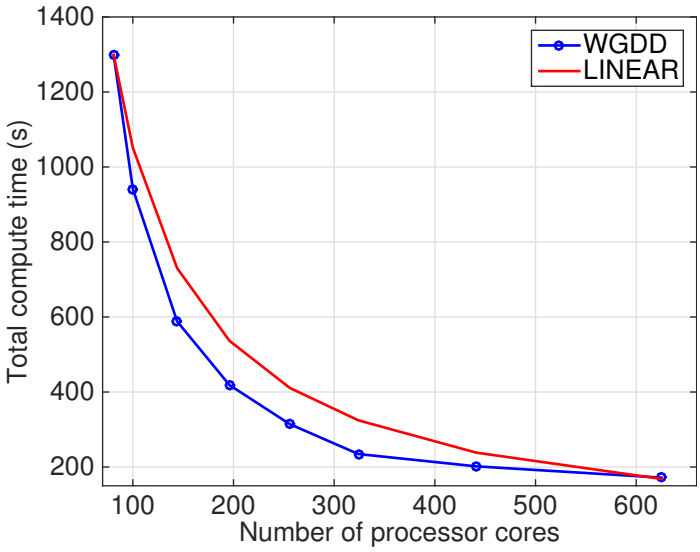


Figure 4.7: The running time .vs. number of processors.

For the results, all test cases have well bounded condition numbers for the global matrices which fits our initial assumption properly. Meanwhile, the superlinear feature from the speedup graph is also in favor of high performance computing. This is the first attempt to introduce weak Galerkin to engineering purpose implementation. The optimal performance on parallel computing indicates a promising future of this method.

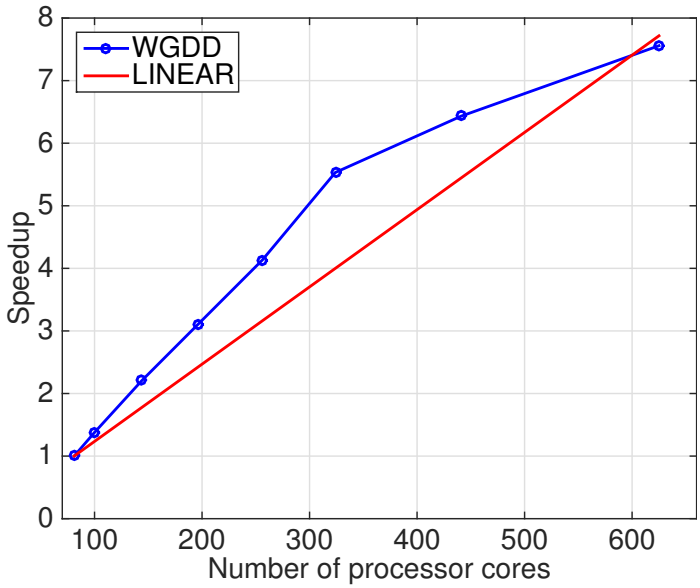


Figure 4.8: The speedup .vs. number of processors.

## 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] 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.

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

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

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

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

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

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

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

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

[19] 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.

[20] 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.

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

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

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

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

[25] 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.

[26] 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.

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

[28] 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.

[29] 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.

[30] 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.

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

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

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

[34] 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.

[35] 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.

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

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