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 [13]. 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 Ω 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 $\varepsilon(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T)$, μ and λ 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 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 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

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 [24, 3, 10, 19]

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.

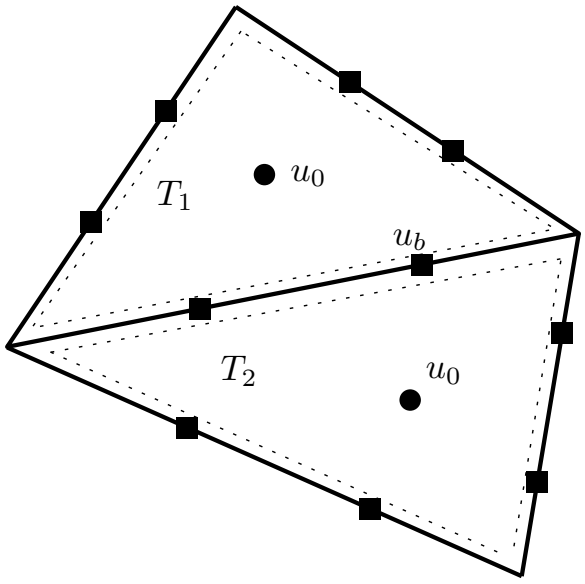

Figure 2.1: Weak Galerkin triangular elements and solution points.

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.

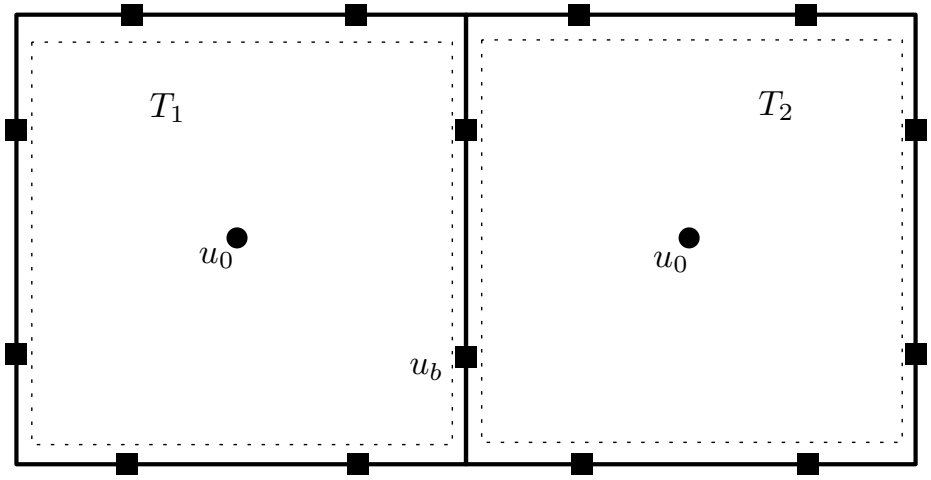

Figure 2.2: Weak Galerkin quadrilateral elements and solution points.

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

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 [16, 12, 23, 15] and a non-overlapping domain decomposition scheme, namely, the balancing domain decomposition with constraints (BDDC) [4, 21, 20]. The WG method is considered as a newly developed robust numerical method and inherits the locking-free feature for the linear elastic equation [22].

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 [17, 18]. Wang et al [22] 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 [13] 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 Method

4.1.2 Balancing Domain Decomposition by Constraints

The balancing domain decomposition method was introduced by Mandel [14]. 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 [8, 7] 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 [11]. 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.

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:

1. Schur Complement [5] 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.

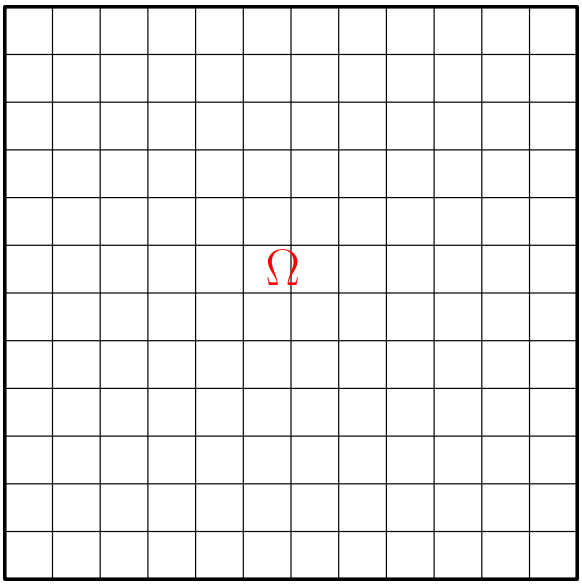

Figure 4.1: The total computational domain.

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.

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

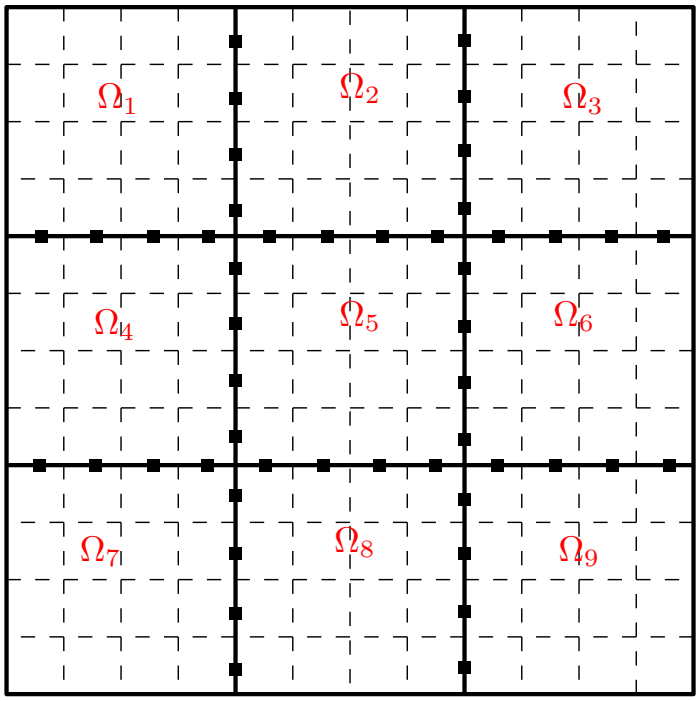

Figure 4.2: After the Schur complement method the computational domain becomes interior and interface.

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.

4.3 Schur Complement Method for subdomain Ω_j

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

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

$$u_h = [u_{I0}, u_{Ib}, u_{\Gamma b}] \tag{4.1}$$

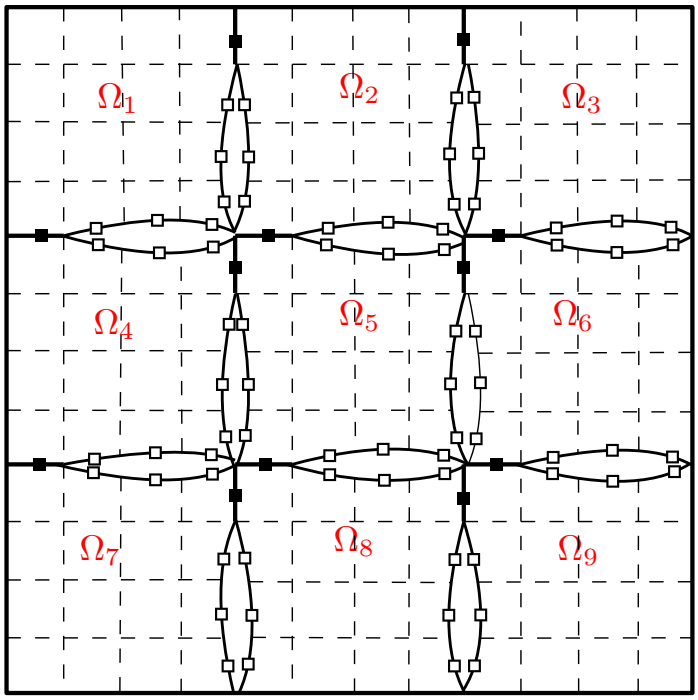

Figure 4.3: BDDC computational domain with only one cell boundary.

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

$$\begin{pmatrix} A_{II} & A_{\Gamma I}^T \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} \times \begin{pmatrix} u_I \\ u_\Gamma \end{pmatrix} \tag{4.2}$$

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

$$\begin{pmatrix} A_{II}^{u} & (A_{\Gamma I}^{u})^{T} & 0 & 0 \\ A_{\Gamma I}^{u} & A_{\Gamma\Gamma}^{u} & 0 & A_{\Gamma\Gamma}^{uv} \\ 0 & 0 & A_{II}^{v} & (A_{\Gamma I}^{v})^{T} \\ 0 & A_{\Gamma\Gamma}^{uv} & A_{\Gamma I}^{v} & A_{\Gamma\Gamma}^{v} \end{pmatrix} \begin{pmatrix} u_I \\ u_\Gamma \\ v_I \\ v_\Gamma \end{pmatrix} \tag{4.3}$$

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

$$S_{\Gamma\Gamma}^{j} = A_{\Gamma\Gamma}^{(j)} - \left[A_{\Gamma I}^{(j)}\right] \times \left[A_{II}^{(j)}\right]^{-1} \times \left[A_{\Gamma I}^{(j)}\right]^{T} \tag{4.4}$$

The loading force along the interface has the form

$$f_{\Gamma}^{(j)} = b_{\Gamma}^{(j)} - \left[A_{\Gamma I}^{(}j)\right] \times \left[A_{II}^{(j)}\right]^{(-1)} \times b_{I}^{(j)} \tag{4.5}$$

Then denote the assembled matrix form

$$S_{\Gamma\Gamma} = \sum_{j=1}^{N} R_{\Gamma}^{(j)^{T}} S_{\Gamma}^{(j)} R_{\Gamma}^{(j)} \tag{4.6}$$

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

Therefore, the global interface problem is constructed as

$$S_{\Gamma\Gamma} \begin{pmatrix} u_\Gamma \\ v_\Gamma \end{pmatrix} = \begin{pmatrix} f_{\Gamma}^{u} \\ f_{\Gamma}^{v} \end{pmatrix} \tag{4.7}$$

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_Γ.

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:

$$\hat{\Lambda}_\Gamma = \hat{\Lambda}_\Pi \oplus (\sum_{i=1}^{N} \Lambda_\Delta^{(j)}) \tag{4.8}$$

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, $\hat{\Lambda}_\Gamma$, to precondition the Schur complement problem whose solution lies in the fully assembled space $\hat{\Lambda}_\Gamma$.

The key component of BDDC preconditioner [6]:

- 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 $R_{D,\Gamma} := D_\Gamma R_\Gamma$

The BDDC preconditioner has the following form that

$$M_{\Gamma_{BDDC}}^{-1} = R_{D,\Gamma}^T \tilde{S}_{\Gamma\Gamma}^{-1} R_{D,\Gamma} \tag{4.9}$$

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

$$M = \begin{pmatrix} S_{rr}^{(1)} & 0 & 0 & \cdots & 0 & S_{rc}^{(1)} \\ 0 & S_{rr}^{(2)} & 0 & \cdots & 0 & S_{rc}^{(2)} \\ 0 & 0 & S_{rr}^{(3)} & \cdots & 0 & S_{rc}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & S_{rr}^{(N)} & S_{rc}^{(N)} \\ S_{cr}^{(1)} & S_{cr}^{(2)} & S_{cr}^{(3)} & \cdots & S_{cr}^{(N)} & S_{cc} \end{pmatrix} \tag{4.10}$$

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

$$M_{\Gamma_{BDDC}}^{-1} S_{\Gamma\Gamma} u_\Gamma = M_{\Gamma_{BDDC}}^{-1} f_\Gamma \tag{4.11}$$

4.5 WG-BDDC Method Implementation

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

$$S_{cc} = \sum_{i=1}^{N} A_{\Pi\Pi}^{(j)} \tag{4.12}$$

meanwhile, the rest unknown variables matrix can be written in

$$S_{rr}^{(i)} = \begin{bmatrix} A_{II}^{(j)} & A_{I\Delta}^{(}j) \\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)} \end{bmatrix} \tag{4.13}$$

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

$$\hat{R}_{D,\Gamma}^{T}\{R_{\Gamma,\Delta}^{T}(\sum_{j=1}^{N}\begin{bmatrix}0 & R_{\Delta}^{(j)^T}\end{bmatrix}\begin{bmatrix}A_{II}^{(j)} & A_{I\Delta}^{(j)}\\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)}\end{bmatrix}^{-1}\begin{bmatrix}0\\ R_{\Delta}^{(j)}\end{bmatrix})R_{\Gamma\Delta}+\Phi S_{\Pi}^{-1}\Phi^{T}\}\hat{R}_{D,\Gamma} \tag{4.14}$$

with

$$\Phi = R_{\Gamma\Pi}^{T} - R_{\Gamma\Delta}^{T}\sum_{j=1}^{N}\begin{bmatrix}0 & R_{\Delta}^{(i)^T}\end{bmatrix}\begin{bmatrix}A_{II}^{(j)} & A_{I\Delta}^{(j)}\\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)}\end{bmatrix}^{-1}\begin{bmatrix}A_{\Pi I}^{(j)^T}\\ A_{\Pi\Delta}^{(j)\ T}\end{bmatrix}R_{\Pi}^{(j)} \tag{4.15}$$

and

$$S_{\Pi} = \sum_{j=1}^{N} R_{\Pi}^{(j)^T}\{A_{\Pi\Pi}^{(j)} - \begin{bmatrix}A_{\Pi I}^{(j)} & A_{\Pi\Delta}^{(j)}\end{bmatrix}\begin{bmatrix}A_{II}^{(j)} & A_{I\Delta}^{(j)}\\ A_{\Delta I}^{(j)} & A_{\Delta\Delta}^{(j)}\end{bmatrix}^{-1}\begin{bmatrix}A_{\Pi I}^{(j)^T} & A_{\Pi\Delta}^{(j)\ T}\end{bmatrix}\}R_{\Pi}^{(j)} \tag{4.16}$$

here the S_{Π} 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

$$\kappa(M_{\Gamma_{BDDC}}^{-1}\hat{S}_{\Gamma\Gamma}) \leq C(1+log(\frac{kH}{h}))^2 \tag{4.17}$$

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 [2] 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 [1] 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) [9] 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

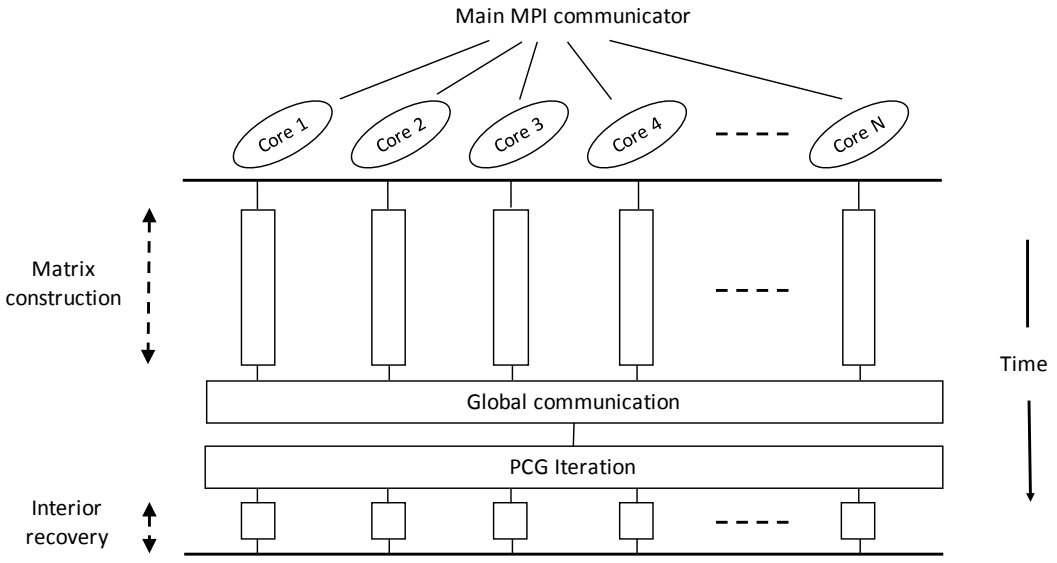

Figure 4.4: Parallel computing work flow.

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
4×4	2.217	5	1.6013e-3	-	4	1.722	7	1.6013e-3	-
8×8	2.390	9	3.9939e-4	2.0	8	2.390	9	3.9939e-4	2.0
16×16	2.335	8	9.9789e-5	2.0	16	3.245	10	9.9789e-5	2.0
32×32	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
4×4	3.528	8	7.1456e-5	-	4	2.900	10	7.1456e-5	-
8×8	3.803	10	8.9214e-6	3.0	8	3.803	10	8.9214e-6	3.0
16×16	3.768	10	1.1150e-6	3.0	16	4.957	12	1.1150e-6	3.0
32×32	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
4×4	2.451	7	1.0109e-3	-	4	1.968	8	1.0109e-3	-
8×8	2.648	9	2.5117e-4	2.0	8	2.648	9	2.5117e-4	2.0
16×16	2.629	9	6.2696e-5	2.0	16	3.529	10	6.2696e-5	2.0
32×32	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	λ_1		Cond.	Iter.	L^2-error	λ_1
4×4	3.805	8	6.6333e-5	-	4	3.926	11	6.6333e-5	-
8×8	4.003	12	8.2709e-6	3.0	8	4.003	12	8.2709e-6	3.0
16×16	3.943	12	1.0334e-6	3.0	16	5.084	13	1.0334e-6	3.0
32×32	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

$$u = \begin{pmatrix} sin(2\pi x)sin(2\pi y) \\ 1 \end{pmatrix} \tag{4.18}$$

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
4×4	3.671	8	2.1451e-4	-	4	3.024	10	2.1451e-4	-
8×8	3.965	10	5.2129e-5	2.0	8	3.965	10	5.2129e-5	2.0
16×16	3.934	10	1.2937e-5	2.0	16	5.153	12	1.2937e-5	2.0
32×32	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
4×4	4.620	8	6.7627e-6	-	4	3.859	11	6.7628e-6	-
8×8	4.987	12	7.8998e-7	3.0	8	4.987	12	7.8998e-7	3.0
16×16	4.921	12	9.6925e-8	3.0	16	6.235	13	9.6931e-8	3.0
32×32	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
2×2	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

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.

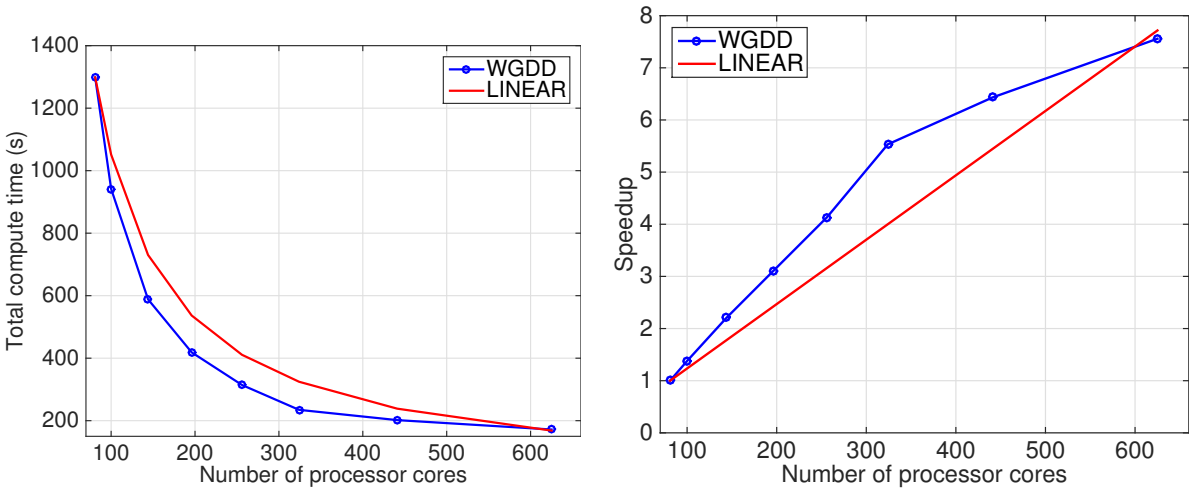

Figure 4.5: The running time and speedup .vs. number of processors.

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.

References

[1] 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.

[2] 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.

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

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

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

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

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

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

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

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

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

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

[22] 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.

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

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