The Adaptive Finite Element Method for Poisson Equation with Algebraic Multigrid Solver

Wenqiang Feng
wfeng1@utk.edu
Department of Mathematics,
University of Tennessee, Knoxville, TN, 37909

September 1, 2014

Abstract

This report is for my MATH 673 final project. In this report, I give some details for implementing the Adaptive Finite Element Method (AFEM) via Matlab. Moreover, I also describe how to implement the Algebraic Multigrid Solver with Matlab. Some functions are from my previous Finite Element Method(FEM) package [8] [9, 10] and some functions are from Long Chen' package [4][5].

Contents

List of Figures

List of Tables

List of Notations

$\mathcal{E}_h^B$	Boundary edges
$\mathcal{E}_h^D$	Dirichlet edges
$\mathcal{E}_h^I$	Interior edges
$\mathcal{E}_h^N$	Neumann edges
Γ_D	The Dirichlet boundary
Γ_N	The Dirichlet boundary
$\mathbb{P}(K)$	Finite dimension smooth function (typically polynomial) on the region K
$\mathbb{V}_h$	The finite dimension space is used to approximate the variable u
$\mathcal{T}_h$	The partition (triangulation or rectangular partition) of Ω
h_e	The diameter of the edge

1 The Model Problem

For simplicity, I consider the following pure dirichlet boundary condition poisson equation:

$$-\Delta u = f \quad \text{in} \quad \Omega, \quad u = g \quad \text{on} \quad \partial\Omega, \tag{1}$$

where Ω is assumed to be a polygonal domain, f a given function in $L^2(\Omega)$ and g a given function in $H^{\frac{1}{2}}(\Omega)$.

Before I give the details, I would like to introduce some useful notations in this report. Let $\mathcal{T}_h$ to be the partition (Figure.1)(typically triangulation or rectangular partition) of Ω, with piecewise constant mesh size h, i.e., $h_K = \text{diam}(K)$, similarly, to be $h_e = \text{diam}(e)$, Γ_D to be the Dirichlet boundary, Γ_N to be the Neumann boundary , and $\mathbb{P}(K)$ to be a finite dimension smooth function (typically polynomial) on the region K. This space $\mathbb{V}_h$ ($\subset H_0^1(\Omega)$) will be used to approximate the variable u.

$$\mathbb{V}_h := \{v \in L^2(\Omega) \quad | \quad v\,|_K \in \mathbb{P}(K) \ \ \forall K \in \mathcal{T}_h,\ v = 0 \ on\ \Gamma_D\}. \tag{2}$$

After we have the above notations, we can define the Jump and Average as following:

1. If $e \in \mathcal{E}$ (See Figure.2)

$$\begin{aligned} [\![v]\!] &= v^+ n^+ + v^- n^-; \\ \{\{\nabla v\}\} &= \frac{1}{2}(\nabla v^+ + \nabla v^-). \end{aligned}$$

where $v^+ = v^- = v|_K$.

2. If $e \in \mathcal{E}$ (See Figure.3)

$$\begin{aligned} [\![v]\!] &= v^+ n^\tau; \\ \{\{\nabla v\}\} &= \nabla v^\tau. \end{aligned}$$

Given the discontinuous nature of the piecewise polynomial functions, we define interior edges $\mathcal{E}_h^I$, boundary edges $\mathcal{E}_h^B$, Dirichlet boundary $\mathcal{E}_h^D$ and Neumann boundary $\mathcal{E}_h^N$ for $\mathcal{T}_h$ (Similarly, we can give the definitions for $\mathcal{T}_H$) as following

$$\begin{aligned} \mathcal{E}_h^I &= \left\{e = \partial K_j \cap \partial K_l,\ \mu(K_j \cap K_l) > 0\right\} \\ \mathcal{E}_h^B &= \{e = \partial K \cap \Omega,\ \mu(K \cap \Omega) > 0\} \\ \mathcal{E}_h^D &= \{e = \partial K \cap \Gamma_D,\ \mu(K \cap \Gamma_D) > 0\} \\ \mathcal{E}_h^N &= \{e = \partial K \cap \Gamma_N,\ \mu(K \cap \Gamma_N) > 0\} \end{aligned}$$

We also set $\mathcal{E}_h = \mathcal{E}_h^I \cup \mathcal{E}_h^B$ and $\mathcal{E}_h^B = \mathcal{E}_h^D \cup \mathcal{E}_h^N$. If $e \in \mathcal{E}_h^I$, then $e = \partial K_- \cap \partial K_+$ for $\partial K_-,\ \partial K_+ \in \mathcal{T}_h$.

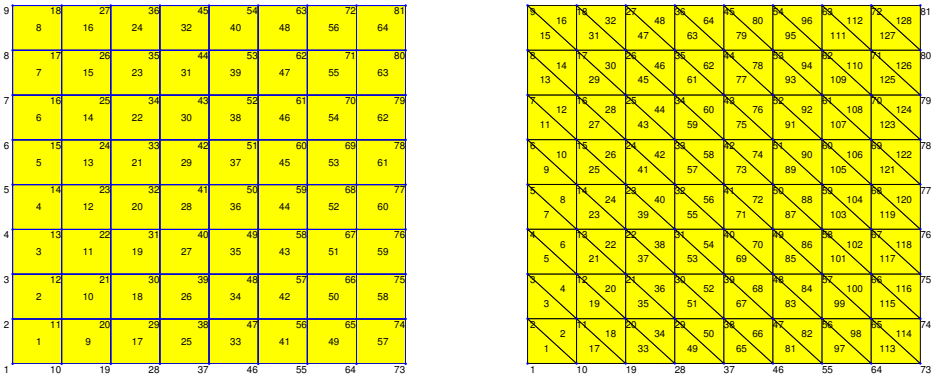

Figure 1: The rectangular partition and triangulation

1.1 The weak formulation for the model problem

For the standard FEM, the weak formulations can be written as as follows:

$$\text{find } u \in H^1(\Omega) \text{ with } u|_{\partial\Omega} = g \text{ and} \tag{3}$$

$$a(u,\phi) := \int_\Omega \nabla u \nabla \phi = \int_\Omega f\phi \quad \text{for} \quad \forall \phi \in H_0^1, \tag{4}$$

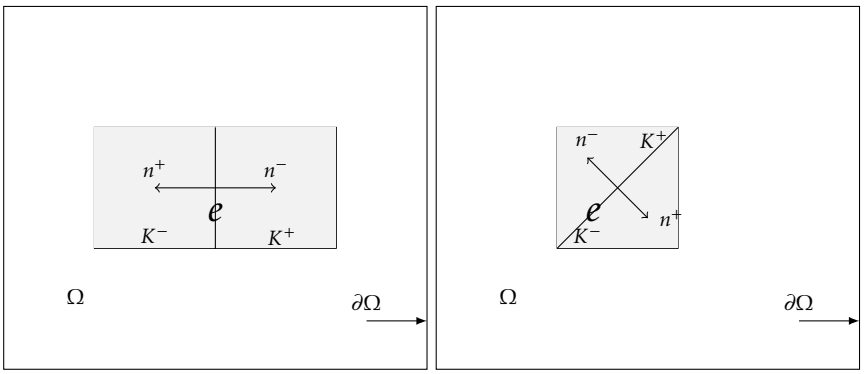

Figure 2: The interior edge and normal vectors for rectangular partition and triangulation.

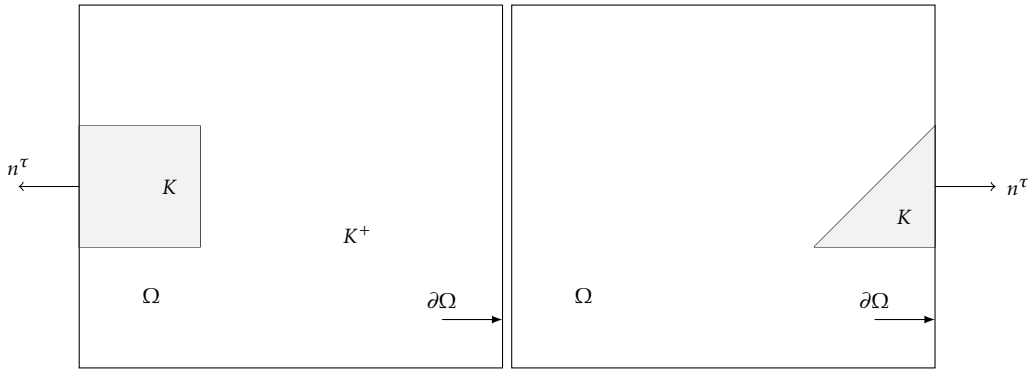

Figure 3: The boundary edge and normal vectors for rectangular partition and triangulation.

1.2 The Galerkin approximation formula

$$\text{find } u_h \in \mathbb{V}_h \text{ with } u_h|_{\partial\Omega} = g \text{ and} \tag{5}$$

$$a(u_h, \phi_h) := \int_\Omega \nabla u_h \nabla \phi = \int_\Omega f_h \phi_h \quad \text{for} \quad \forall \phi_h \in \mathbb{V}_h, \tag{6}$$

2 AFEM implementation

In this section, I will give the implement details for AFEM via Matlab. I will follow the general form in [6][7][12][13][19][20][21]. I will also follow the standard local mesh refinement loops of AFEM:

$$\text{SOLVE} \longrightarrow \text{ESTIMATE} \longrightarrow \text{MARK} \longrightarrow \text{REFINE}.$$

2.1 Poisson Solver

2.1.1 Data structure

Before I give the Poisson solver, I would like to introduce the data structure in Matlab. I will use the initial mesh (Figure.4) as an example to illustrate the concept of the components.

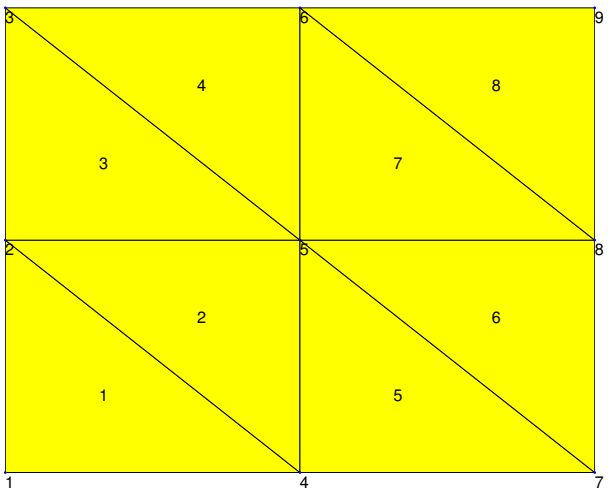

Figure 4: The initial mesh partition

1. The basic data structure (See Table (1)) is mesh which contains
   ◇mesh.node: The *node* vector is just the xy-value of node.
   ◇mseh.elem: In the *elem* matrix, the first and the second column represent the start nodal indices vector and the end nodal indices vector, respectively.
   ◇mesh.Dirichlet: The *Dirichlet* is the Dirichlet boundary condition edges.
   ◇mesh.Neumann: The *boundary* is the Neumann boundary condition edges.
2. Another main data structure is indices (See Table (2))which will provide useful indices.
   ◇ indices.neighbor: *indices.neighbor(1:NT,1:3)*: the indices map of neighbor information of elements, where neighbor(t,i) is the global index of the element opposite to the i-th vertex of the t-th element.
   ◇ indices.elem2edge: *indices.elem2edge(1:NT,1:3)*: the indices map from elements to edges, elem2edge(t,i) is the edge opposite to the i-th vertex of the t-th element.
   ◇ indices.edge: *indices.edge(1:NE,1:2)*: all edges, where edge(e,i) is the global index of the i-th vertex of the e-th edge, and edge(e,1) < edge(e,2).
   ◇ indices.bdEdge: *indices.bdEdge(1:Nbd,1:2)*: boundary edges with positive oritentation, where bdEdge(e,i) is the global index of the i-th vertex of the e-th edge for i=1,2. The positive oritentation means that the interior of the domain is on the

left moving from bdEdge(e,1) to bdEdge(e,2). Note that this requires elem is positive ordered, i.e., the signed area of each triangle is positive. If not, use elem = fixorder(node,elem) to fix the order.

◇ indices.edge2elem: *indices.edge2elem(1:NE,1:4)*: the indices map from edge to element, where edge2elem(e,1:2) are the global indices of two elements sharing the e-th edge, and edge2elem(e,3:4) are the local indices of e (See Figure. 5 and Table. 2).

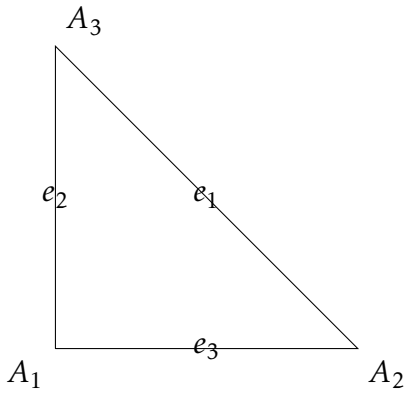

Figure 5: The local indices of vetrices and edges .

2.1.2 Poisson solver Process

The Poisson solver contains three main steps:

◇*Step 1* Pre-process step: In this phase, we should get the information of the nodal, element and indices. In this project, I use my own function *InitialMesh* to generate for the square domain. For the complex domain, you can use Matlab's PDE tool or the package in [14] to generate the mesh.

◇*Step 2* Process step: This phase contains four sub-steps as follow:

- ◇*step 2.1* Compute the stiffness matrix and load vector of the elements
- ◇*step 2.2* Assemble the global stiffness matrix and global load vector
- ◇*step 2.3* Deal with the boundary condition
- ◇*step 2.4* Solve the linear system $AU = F$

◇*Step 3* post-process step: this phase is just to output the solution and give it visual form.

2.2 Posterior Error Estimation

The posteriori error estimators and indicator are essential components for the Estimation part. In this project, I will use the residual-type error estimator [1][7][12][13][20][21].

Table 1: MESH Data structure in two dimension

Nodal NO.	node(:,1)	node(:,2)
1	-1	-1
2	-1	0
3	-1	1
4	0	-1
5	0	0
6	0	1
7	1	-1
8	1	0
9	1	1

i	Dirichlet(:,1)	Dirichlet(:,2)
	1	2
	1	4
	2	3
	3	6
	4	7
	6	9
	7	8
	8	9

	Element			neighbor			elem2edge		
NO.	1	2	3	1	2	3	1	2	3
1	1	4	2	2	1	1	4	1	2
2	5	2	4	1	5	3	4	8	5
3	2	5	3	4	3	2	6	3	5
4	6	3	5	3	7	4	6	10	7
5	4	7	5	6	2	5	11	8	9
6	8	5	7	5	6	7	11	15	12
7	5	8	6	8	4	6	13	10	12
8	9	6	8	7	8	8	13	16	14

Table 2: Indices data structure in two dimension

edge			edge2elem				
Edge NO.	edge(:,1)	edge(:,2)	EdgeNO.	elem 1	elem 2	local	local
1	1	2	1	1	1	2	2
2	1	4	2	1	1	3	3
3	2	3	3	3	3	2	2
4	2	4	4	1	2	1	1
5	2	5	5	2	3	3	3
6	3	5	6	3	4	2	2
7	3	6	7	4	4	3	3
8	4	5	8	2	5	2	2
9	4	7	9	5	5	3	3
10	5	6	10	4	7	2	2
11	5	7	11	5	6	1	1
12	5	8	12	6	7	3	3
13	6	8	13	7	8	1	1
14	6	9	14	8	8	3	3
15	7	8	15	6	6	2	2
16	8	9	16	8	8	2	2

Boundary Element	BdEdge NO.	bdedge(:,1)	bdedge(:,2)
1	1	2	1
1	2	3	2
3	3	7	8
4	4	8	9
5	5	1	4
8	6	6	3
6	7	4	7
8	8	9	6

Before I give the derivation of the residual error estimator, I would to give the definition of the domains (See Figure. 6) of inter point w_p, interior edge w_e and element w_k in [19].

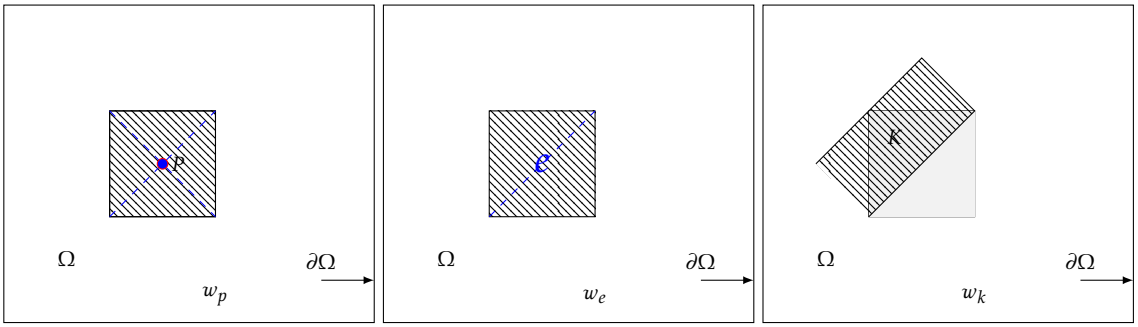

Figure 6: The domains of inter point w_p, interior edge w_e and element w_k[19].

Now, I will recall the residual-type error estimator for (4) and (6). For a given partition $\mathcal{T}_H$, let u_H be the finite element approximation of the solution u for the poisson equation (1). Then subtracting (6) from (4) and integrating by parts yields the following well-known relation between the error $u - u_H$ and the residuals:

$$a(u - u_H, \phi) = \sum_{T \in \mathcal{T}_H} (f, \phi - \mathcal{I}_H \phi)_T + \sum_{e \in \mathcal{E}_H^I} \langle J_e, \phi - \mathcal{I}_H \phi \rangle_e, \quad \forall \phi \in H_0^1. \tag{7}$$

Where $\mathcal{I}_H$ is the Clément interpolation operator and $J_e = [\![\nabla u_H]\!]_e \cdot \vec{n}$. The derivation of (7) needs the following two facts: First one is the Galerkin orthogonality,

$$a(u - u_H, \mathcal{I}_H \phi) = 0. \tag{8}$$

The other one is

$$[\![\nabla u]\!] \cdot \vec{n} = 0. \tag{9}$$

Lemma 2.1. ***Trace Theorem****[2]*
Let $\phi \in H^1(\Omega)$. Then there exists a constant $C > 0$ such that

$$\|\phi\|_{L^2(\partial\Omega)} \leq \sqrt[4]{8} \|\phi\|_{L^2(\Omega)}^{1/2} \|\phi\|_{H^1(\Omega)}^{1/2}. \tag{10}$$

Lemma 2.2. ***Trace Theorem with Scaling argument****[2]*
Let $\phi \in H^1(K)$ and $e \subset \partial K$. Then there exists a constant $C > 0$ such that

$$\|\phi\|_{0,e} \leq C \left\{ \|\phi\|_{0,K} \left(H^{-1} \|\phi\|_{0,K} + \|\nabla \phi\|_{0,K} \right) \right\}^{1/2} \leq C \left(H^{-1} \|\phi\|_{0,K}^2 + H \|\nabla \phi\|_{0,K}^2 \right)^{1/2} \tag{11}$$

Lemma 2.3. ***Clément Interpolation Theorem****[21]*
Let $\mathcal{I}_H$ be the quasi-interpolation operator. Then the operator $\mathcal{I}_H$ satises the following local error estimates for all $\phi \in \mathbb{V}_H$ and all elements $K \in \mathcal{T}_H$

$$\left\|\phi - \mathcal{I}_H\phi\right\|_{0,K} \leq CH_k \left\|\phi\right\|_{1,w_k} \tag{12}$$

$$|\phi - \mathcal{I}_H\phi|_{0,e} \leq CH_e^{1/2} \left\|\phi\right\|_{1,w_e} \tag{13}$$

Where the constant C only depending on the shape regularity of $\mathcal{T}_H$.

By using the facts (8) and (9) together with the trace theorem with scaling argument (Lemma 2.2) and the interpolation theory (Lemma 2.3) , we can get the following theorem.

Theorem 2.1. *For a given partition $\mathcal{T}_H$, let u_H be the finite element approximation of the solution u for the poisson equation (1). Then there exists a constant C_1 only depending depending on the shape regularity of $\mathcal{T}_H$ such that*

$$|u - u_H|^2_{1,\Omega} \leq C_1 \left(\sum_{K\in\mathcal{T}_H} \|Hf\|^2_{0,K} + \sum_{e\in\mathcal{E}_H^I} \left\|H^{1/2}[\![\nabla u_H \cdot \vec{n}]\!]\right\|^2_{0,e} \right). \tag{14}$$

Proof. For any $\phi \in H_0^1$ and any $\mathcal{I}_H\phi \in \mathbb{V}_H$, we have

$$\begin{aligned}
& a(u - u_H, \phi) \\
= \;& a(u - u_H, \phi - \mathcal{I}_H\phi) \;\text{ (Galerkin orthogonality (8))} \\
= \;& \sum_{K\in\mathcal{T}_H} \int_K \nabla(u - u_H)\nabla(\phi - \mathcal{I}_H\phi)dx \\
= \;& \sum_{K\in\mathcal{T}_H} \int_{\partial K} \nabla(u - u_H)\cdot\vec{n}(\phi - \mathcal{I}_H\phi)dS - \sum_{K\in\mathcal{T}_H} \int_K \Delta(u - u_H)(\phi - \mathcal{I}_H\phi)dx \\
= \;& \sum_{e\in\mathcal{E}_H^I} \int_e [\![\nabla u_H \cdot \vec{n}_e]\!](\phi - \mathcal{I}_H\phi)dS + \sum_{K\in\mathcal{T}_H} \int_K f(\phi - \mathcal{I}_H\phi)dx \;\text{ (Using fact (9))} \\
= \;& \sum_{T\in\mathcal{T}_H} (f, \phi - \mathcal{I}_H\phi)_T + \sum_{e\in\mathcal{E}_H^I} \langle J_e, \phi - \mathcal{I}_H\phi\rangle_e \;\text{ (Result (7))} \\
\leq \;& \sum_{T\in\mathcal{T}_H} \|H_K f\|^2_{0,K} \left\|H_K^{-1}(\phi - \mathcal{I}_H\phi)\right\|^2_{0,K} + \sum_{e\in\mathcal{E}_H^I} \left\|H_e^{1/2}[\![\nabla u_H \cdot \vec{n}_e]\!]\right\|^2_{0,e} \left\|H_e^{-1/2}(\phi - \mathcal{I}_H\phi)\right\|^2_{0,e} \\
\leq \;& \left(\sum_{T\in\mathcal{T}_H} \|H_K f\|^2_{0,K} + \sum_{e\in\mathcal{E}_H^I} \left\|H_e^{1/2}[\![\nabla u_H \cdot \vec{n}_e]\!]\right\|^2_{0,e} \right)^{1/2} \left(\sum_{T\in\mathcal{T}_H} \left\|H_K^{-1}(\phi - \mathcal{I}_H\phi)\right\|^2_{0,K} + \left\|\nabla(\phi - \mathcal{I}_H\phi)\right\|^2_{0,K} \right)^{1/2}.
\end{aligned}$$

For the last step, we used the trace theorem with scaling argument (Lemma 2.2). And now, by using the quasi-interpolation operator theorem (Lemma 2.3), we can choose $\mathcal{I}_H\phi$ such that

$$\left(\sum_{T\in\mathcal{T}_H} \left\|H_K^{-1}(\phi - \mathcal{I}_H\phi)\right\|^2_{0,K} + \left\|\nabla(\phi - \mathcal{I}_H\phi)\right\|^2_{0,K} \right)^{1/2} \leq |\phi|_{1,\Omega}. \tag{15}$$

Then we get

$$|u-u_H|_{1,\Omega} = \sup_{\phi\in H_0^1} \frac{a(u-u_H,\phi)}{|\phi|_{1,\Omega}} \leq \left(\sum_{T\in\mathcal{T}_H} \|H_K f\|_{0,K}^2 + \sum_{e\in\mathcal{E}_H^I} \left\| H_e^{1/2} [\![\nabla u_H \cdot \vec{n}_e]\!] \right\|_{0,e}^2 \right)^{1/2} \tag{16}$$

□

From theorem (2.1), we can get the local residual-type error indictor

$$\eta_e^2 = \sum_{T\in\mathcal{T}_H} \|H_K f\|_{0,K}^2 + \sum_{e\in\mathcal{E}_H^I} \left\| H_e^{1/2} [\![\nabla u_H \cdot \vec{n}_e]\!] \right\|_{0,e}^2 \tag{17}$$

$$:= \|H_K f\|_{w_e}^2 + \left\| H_e^{1/2} J_e \right\|_e^2 \tag{18}$$

2.3 Marking

I will use the following Dörfler marking strategy [7] in this project.

Algorithm 1 Dörfler marking strategy

Process:

1: Choose the parameter $0 < \theta < 0$
2: For a given subset $\mathcal{E}_h$ of $\mathcal{E}_H$
3: **while** $\eta(u_H, \mathcal{E}_h) \geq \theta\eta(u_H, \mathcal{E}_H)$ **do**
4: Mark the subset $\mathcal{T}_h$ of $\mathcal{T}_H$ of the element with the longest side in $\mathcal{E}_h$.
5: **end while**

2.4 Refinement

I will use the newest vertex bisection (See Figure (7)) to do the refinement. For more details, you can find in[11].

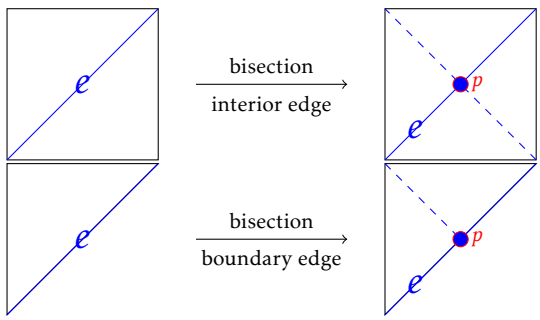

Figure 7: The newest vertex bisection for interior edge and boundary edge.

3 Algebraic Multigrid Method

In the following we will introduce the AMG method [15, 16, 18] that is appropriate for solving the linear system which arises from the AFE method. Let $A_h^1 = A_h$, $\vec{u}_h^1 = \vec{u}_h$, $\vec{b}_h^1 = \vec{b}_h$. Then in one V-cycle a sequence of linear systems

$$A_h^m \vec{u}_h^m = \vec{b}_h^m,\ m = 1, \cdots, M,$$

can be generated from different grid levels. Here $A_h^m = \left(a_{ij}^m\right)_{n_m \times n_m}$, $\vec{b}_h = \left(b_i^m\right)_{n_m \times 1}$, $\vec{u}_h = \left(u_i^m\right)_{n_m \times 1}$, and $n = n_1 > n_2 > \cdots > n_m$. Now we discuss two main phases of the AMG algorithm: setup phase and solution phase [15].

3.1 Setup phase of AMG

In the setup phase, let Ω^m denote the set of unknowns $u_i^m (1 \le i \le n_m)$ of the m^{th} grid level. And the coarser grid Ω^{m+1} is chosen as a subset of Ω^m, which is denoted as C^m in the m^{th} grid level. The remaining subset $\Omega^m - C^m$ will be denoted by F^m. A point u_i^m is said to be strongly connected to u_j^m, if

$$|a_{ij}^m| \ge \eta \cdot \max_{i \ne j} |a_{ij}^m|,\ \ 0 < \eta \le 1. \tag{19}$$

Let S_i^m denote the set of all strongly connected points of u_i^m and let the coarse interpolatory set be $C_i^m = C^m \cap S_i^m$. In general, C^m and F^m are chosen by the following criteria:

(C1) For each $u_i^m \in F^m$, each point $u_j^m \in S_i^m$ should be either in C_i^m itself or should be strongly connected to at least one point in C_i^m;
(C2) C^m should be maximal subset of all points with the property that no two Coarse points are strongly connected to each other.

Define the set of points which are strongly connected to u_i^m to be

$$S_i^{m,T} \equiv \{u_j^m :\ u_i^m \in S_j^m\}. \tag{20}$$

For a set P, let $|P|$ denote the number of the elements in P. Then Algorithm 2 is proposed by Ruge and Stüben in [16, 17] can be used to chose the coarse grid $\Omega^{m+1} = C^m$ and F^m.

Once the coarse grid Ω^{m+1} is chosen, the interpolation operators I_{m+1}^m, restriction operators I_m^{m+1} and the coarse grid equation can be constructed as follows. Let $N_i^m = \{u_j^m \in \Omega^m : j \ne i, a_{ij}^m \ne 0\}$ denote the neighborhood of a point $u_i^m \in \Omega^m$, and $D_i^m = N_i^m - C_i^m$. Then the set of the fine grid neighborhood points which are strongly connected to u_i^m will be $D_i^{m,s} = D_i^m \cap S_i^m$, and the rest set of the neighborhood points which are weakly

Algorithm 2 The construction of coarse grid

Input: Ω^m.
Output: C^m and F^m.
Method:

$C^m \ \emptyset, F^m \ \emptyset, \vec{u}_h^m \ \Omega^m$ and $\lambda_k^m = |S_k^{m,T}| \ (1 \le k \le n_m)$
for $(1 \le i \le n_m)$ **do**
  if $(\vec{u}_h^m \ne \emptyset)$ **then**
    Pick the $u_i^m \in \vec{u}_h^m$ such that $\lambda_i^m = \max\limits_{1 \le k \le n_m} \lambda_k^m$, and set $C^m = C^m \cup \{u_i^m\}, \vec{u}_h^m = \vec{u}_h^m - \{u_i^m\}$
    for (all $u_j^m \in S_i^{m,T} \cap \vec{u}_h^m$) **do**
      Set $F^m = F^m \cup \{j\}$ and $\vec{u}_h^m = \vec{u}_h^m - \{j\}$
      for (all $u_l^m \in S_j^m \cap \vec{u}_h^m$) **do**
        set $\lambda_l^m = \lambda_l^m + 1$
      end for
    end for
    for (all $u_j^m \in S_i^m \cap \vec{u}_h^m$) **do**
      set $\lambda_j^m = \lambda_j^m - 1$
    end for
  else
    Stop.
  end if
end for

connected (not strongly connected) to u_i^m will be $D_i^{m,w} = D_i^m - D_i^{m,s}$. Each $u_i^m \in C^m$ can be directly interpolated from the corresponding variable in Ω^{m+1} with unity weight. Each $u_i^m \in F^m$ can be interpolated as a weighted summation of the points in the coarse interpolatory set C_i^m. Assume $u_i^m \in C^m$ is corresponding to $u_{k_i}^{m+1} \in \Omega^{m+1}$. Ruge and Stüben proposed the corresponding interpolation formula [16]:

$$I_{m+1}^m \{u_k^{m+1}\}_{k=1}^{n_{m+1}} = \begin{cases} u_{k_i}^{m+1} & \forall u_i^m \in C^m \\ \sum\limits_{\{j:u_j^m \in C_i^m\}} w_{ij}^m u_{k_j}^{m+1} & \forall u_i^m \in F^m \end{cases} \tag{21}$$

where

$$w_{ij}^m = -\frac{1}{a_{ii}^m + \sum\limits_{\{r:u_r^m \in D_i^{m,w}\}} a_{ir}^m} \left[a_{ij}^m + \sum_{\{r:u_r^m \in D_i^{m,s}\}} a_{ir}^m a_{rj}^m \Big/ \sum_{\{l:u_l^m \in C_i^m\}} a_{il}^m \right] \tag{22}$$

The Galerkin type method in [16] is a simple approach to define the restriction operator I_m^{m+1}

$$I_m^{m+1} = (I_{m+1}^m)^T \tag{23}$$

and

$$A_h^{m+1} = I_m^{m+1} A_h^m I_{m+1}^m, \ \ \vec{b}_h^{m+1} = I_m^{m+1} \vec{b}_h^m I_{m+1}^m.$$

3.2 Solution phase of AMG

In the solution phase, the smoothing operator needs to be chosen with proper parameters ν_1 and ν_2, which are the number of the pre-smoothing and post-smoothing steps. In the next section, we will investigate the influence of the type of the operator (Gauss-Seidel and incomplete LU) and these two parameters. Furthermore, we will consider V-cycle only with the maximum number of levels M in this article. Another critical component of AMG is the stopping tolerance, which may have significant effect on the accuracy. Our study in the next section shows that the tolerance needs to be small enough for the chosen mesh size. Once all the above components are specified, the recursively defined AMG algorithm (Algorithm 3) with V-cycle can be proposed in the usual framework as follows [16].

Algorithm 3 The AMG algorithm of the elliptic interface problem

Input: Model parameters and AMG parameters
Output: AMG approximation solution $\vec{u}_h$.
Method:

Assemble the matrix from the AFE formulation: A_h
Assemble the vector from the AFE formulation: $\vec{b}_h$
relative residual= 1, $\vec{u}_h = 0$
while relative residual>tolerance **do**
    $m = 1$, $A_h^1 = A_h$, $\vec{b}_h^1 = \vec{b}_h$, $\vec{u}_h^1 = \vec{u}_h$
    Call algorithm $MG(A_h^m, \vec{u}_h^m, \vec{b}_h^m, \Omega^m, \nu_1, \nu_2, m)$ as follows:
        Call Algorithm 2 with Ω^m to obtain the C^m and F^m
        Set $\Omega^{m+1} = C^m$
        Define I_{m+1}^m, $I_m^{m+1} = (I_{m+1}^m)^T$
        Pre-smooth: $\vec{u}_h^m$:=smooth$(A_h^m, \vec{u}_h^m, \vec{b}_h^m, \nu_1)$
        Residual: $\vec{r}_h^m = \vec{b}_h^m - A_h^m \vec{u}_h^m$
        Coarsening: $\vec{r}_h^{m+1} = I_m^{m+1} \vec{r}^m I_{m+1}^m$, $A_h^{m+1} = I_m^{m+1} A_h^m I_{m+1}^m$, $\vec{b}_h^{m+1} = I_m^{m+1} \vec{b}_h^m I_{m+1}^m$
        If m ≡ M
            Solve: $A_h^{m+1} \delta^{m+1} = \vec{r}_h^{m+1}$
        Else
            Recursion: $\delta^{m+1} = MG(A_h^{m+1}, 0, \vec{r}_h^{m+1}, \Omega^{m+1}, \nu_1, \nu_2, m+1)$
        EndIf
        Correction: $\vec{u}_h^m = \vec{u}_h^m + I_{m+1}^m \delta^{m+1} I_m^{m+1}$
        Post-smooth: $\vec{u}_h^m$:=smooth$(A_h^m, \vec{u}_h^m, \vec{b}_h^m, \nu_2)$
    END of MG
    $\vec{u}_h = \vec{u}_h^1$
    relative residual $= \left\| \vec{b}_h - A_h \vec{u}_h \right\| / \left\| \vec{b}_h \right\|$
end while

4 Numerical Experiments

For the first experiment, I consider the following poisson problem with pure Dirichlet boundary in $[-1,-1]\times[1,1]$:

$$\begin{aligned} -\Delta u &= -\pi^3 sin(\pi x)(-y + cos(\pi(1-y))) - (2 - \pi sin(\pi x))(-\pi^2 cos(\pi(1-y))) \quad \text{in} \quad \Omega, \\ u &= exactsolution \quad \text{on} \quad \partial\Omega,. \end{aligned}$$

For the residual-type error estimator, I set $\theta = 0.25$, the maximum refinement to be 20 . And for the AMG, I set the initial vector u^0 to be 0 and he strong connection threshold $\eta = 0.25$. We denote number of W-cycles by W's, the size of the coarsest mesh by N_c, and the stopping tolerance on residual by tol. The Conjugate Gradient Method (CG) preconditioner will be used as pre-smoothing and post-smoothing operations.

Table 3: Errors of the AFEM solution for poisson with CG smoother.

$iter$	$\#dof$	N_c	$\|u-u_h\|_{L^2}$	$\|u-u_h\|_{H^1}$	w's
5	57	21	2.72×10^{-1}	4.16	6
10	184	61	1.24×10^{-1}	2.84	7
15	454	148	5.09×10^{-2}	1.88	8
20	1173	408	2.34×10^{-2}	1.21	10

Figure (9)-(12) show the changes of the mesh partition and the numerical solution. From Table (3) and Figure (8), I get the quasi-optimal convergence rates for the Adaptive Finite Element which verify the results in [3].

References

[1] I. Babuska and W. C. Rheinboldt, *Error estimates for adaptive finite element computations*, SIAM Journal on Numerical Analysis, 15 (1978), pp. pp. 736–754. 7

[2] S. C. Brenner and L. R. Scott, *The mathematical theory of finite element methods*, vol. 15, Springer, 2008. 10

[3] J. M. Cascon, C. Kreuzer, R. H. Nochetto, and K. G. Siebert, *Quasi-optimal convergence rate for an adaptive finite element method*, SIAM Journal on Numerical Analysis, 46 (2008), pp. 2524–2550. 16

[4] L. Chen, *AFEM@MATLAB: a matlab package of adaptive finite element methods*, Technique Report, (2006). 1

[5] ——, *iFEM: an innovative finite element methods package in MATLAB*, Technique Report, (2009). 1

[6] W. Dörfler, *A robust adaptive strategy for the nonlinear poisson equation*, Computing, 55 (1995), pp. 289–304. 5

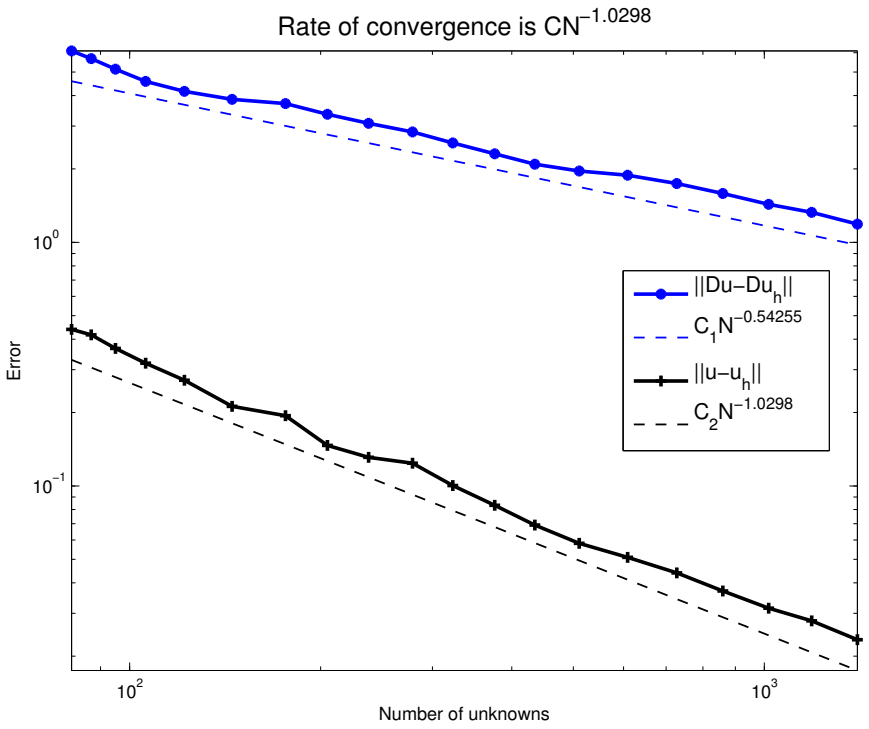

Figure 8: The rate of convergence for H^1 and L^2

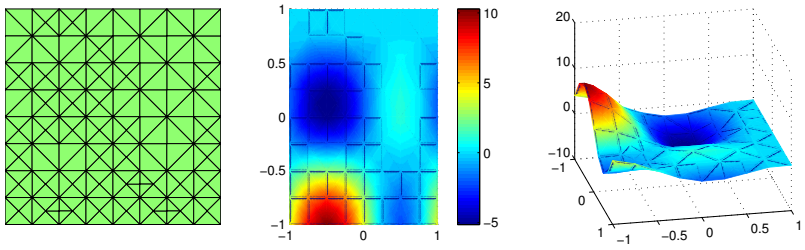

Figure 9: The mesh partition and numerical solution at 5_{th} refinement

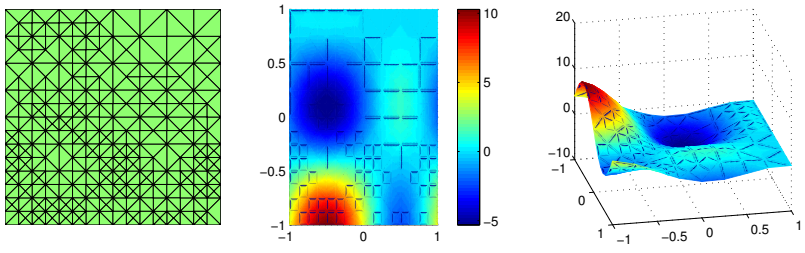

Figure 10: The mesh partition and numerical solution at 10_{th} refinement

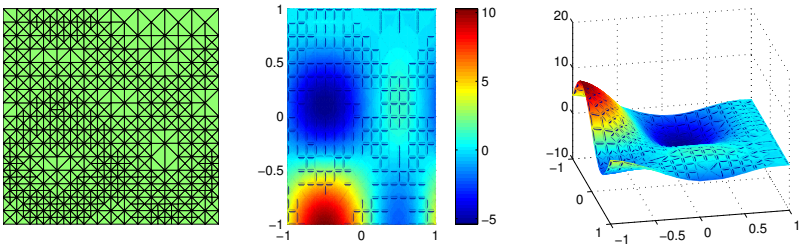

Figure 11: The mesh partition and numerical solution at 15_{th} refinement

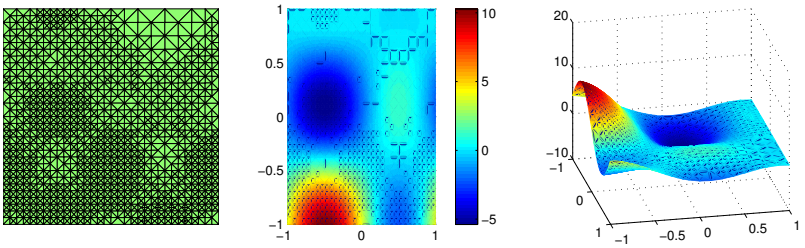

Figure 12: The mesh partition and numerical solution at 20_{th} refinement

[7] ——, *A convergent adaptive algorithm for poisson's equation*, SIAM Journal on Numerical Analysis, 33 (1996), pp. pp. 1106–1124. 5, 7, 12

[8] W. Feng, *Immersed finite element method for interface problems with algebraic multigrid solver*, 2013. 1

[9] W. Feng, X. He, Y. Lin, and X. Zhang, *Immersed finite element method for interface problems with algebraic multigrid solver*, Commun. Comput. Phys., 15 (2014), pp. 1045–1067. 1

[10] W. Feng, X. He, and Y. L. X. Zhang, *Immersed finite element method for interface problems with algebraic multigrid solver*, Commun.Comput.Phys., (To appear). 1

[11] W. F. Mitchell, *A comparison of adaptive refinement techniques for elliptic problems*, ACM Trans. Math. Softw., 15 (1989), pp. 326–347. 12

[12] P. Morin, R. H. Nochetto, and K. G. Siebert, *Convergence of adaptive finite element methods*, SIAM Review, 44 (2002), pp. pp. 631–658. 5, 7

[13] R. Nochetto, K. Siebert, and A. Veeser, *Theory of adaptive finite element methods: An introduction*, Springer Berlin Heidelberg, 2009. 5, 7

[14] P. olof Persson and G. Strang, *A simple mesh generator in matlab*, SIAM Review, 46 (2004), p. 2004. 7

[15] H. F. Q. Chang, Y. Wong, *On the algebraic multigrid method*, Journal of Computational Physics, 125 (1996), pp. 279–292. 13

[16] J. W. Ruge and K. Stüben, *"Algebraic multigrid". In S. F. McCormick, editor, multigrid methods*, SIAM, Philadelphia, 4 (1987), pp. 73–130. 13, 14, 15

[17] K. Stüben, *Algebraic multigrid (AMG): experiences and comparisons*, Applied Mathematics and Computation, 13 (1983), pp. 419–451. 13

[18] C. O. U. Trottenberg and A.schuller, *Multigrid*, vol. 631, Academic Press, London, 2001. 13

[19] R. Verfürth, *A posteriori error estimation and adaptive mesh-refinement techniques*, Journal of Computational and Applied Mathematics, 50 (1994), pp. 67–83. 2, 5, 10

[20] ——, *A review of a posteriori error estimation and adaptive mesh-refinement techniques*, Wiley-Teubner, 1996. 5, 7

[21] ——, *Adaptive finite element methods*. University Lecture, 2013-14. 5, 7, 11