NESTED ADAPTIVE VERTEX-CENTERED FINITE VOLUME METHODS FOR DIFFUSION PROBLEMS

T. Ghoudi 1, I. Kissami 1

1 Department of Mathematics, Faculty of science University of Paris 13, 99 Avenue J.B. Clement, 93430 Villetaneuse, France

Abstract

The goal of this paper is to develop a new technique of mesh adaptation for finite volume solution of diffusion problems. Vertex-centered finite volume discretization is used for the elliptic operator and error estimates in the energy norm are considered for adaptation of meshes. In contrast to the conventional approach in which the diffusion equation is solved after each error estimation, the current method allows for multiple adaptations of meshes within a single error estimation. This nested adaptive finite volume method requires only treatment of conformity in meshes for multiple adaptations which is dealt with using the new bisection procedure. For a prescribed accuracy, the new method offers a substantial reduction of computational cost compared to standard adaptive finite volume methods.

Keywords: Vertex-centered finite volume, Diffusion equations, Error estimates, Mesh adaptation, Adapt solver, Nested refinement, Newest-vertex-bissection.

1 Introduction

In this paper we consider second-order elliptic equations with heterogeneous coefficients. For this class of problems, the diffusion in a two-dimensional closed medium \( \Omega \subset \mathbb{R}^2 \) with boundary \( \partial \Omega \) is described by the following equation:

\[
-\nabla \cdot (K \nabla u(x)) = f(x), \quad \forall x \in \Omega, \\
u(x) = g(x), \quad \forall x \in \partial \Omega,
\]

(1)

where \( f \) is the external force, \( g \) is the boundary source, and \( K \) is a piecewise constant diffusion coefficient. The objective of the present study is to put forward a numerical comparison between different mesh adaptation techniques of vertex-centered finite volume solution of the diffusion problem (1). The adaptive vertex-centered finite volume method is commonly used to approach complex problems arising from several physical phenomena. In many of these applications, adaptive techniques using a posteriori error estimators have become very useful, see for instance [3]. These estimators measure the quality of the calculated solution and provide information to control the mesh adaptation algorithms. In this paper we propose a new multilevel mesh adaptation method that combines two adaptation methods, namely Adapt strategy (for cutting triangles) and Newest-vertex-bissection strategy (for conformity) (see section 4). We propose an algorithm of our strategy. We have succeeded to integrate the a posteriori error estimators into our code which is used as a refinement criterion. This estimator provides an accurate estimate of error in a certain standard norm (energy). We have done numerical tests that confirm the efficiency of our new strategy of refinement.

2 Adaptive vertex-centered finite volume method

We cover the computational domain \( \Omega \) with a regular triangular mesh \( T_h \) and using the notations given in Figure 1. We also construct the associated dual mesh \( D_h \), \( S_h \) is a fine simplicial mesh obtained from the dual mesh \( D_h \), and for each element \( K \) with vertices \( S_i \ (1 \leq i \leq 3) \) of the primal mesh we call \( B \) its barycentre, we call respectively \( \Sigma_1^{opp} \), \( \Sigma_2^{opp} \) and \( \Sigma_3^{opp} \) the edges \([S_2S_3], [S_1S_3] et [S_1S_2] \); and \( \vec{n}_1^{opp} \), \( \vec{n}_2^{opp} \) et \( \vec{n}_3^{opp} \) their outgoing unit normals. The system (1) is discretized using a so-called Finite Volume Finite Element method (FVFE) (see [4]), which is based on a Galerkin method to discretize time and variables, together with a Finite Volume method using the Godunov...
scheme for the convection term. Detailed description of the numerical scheme can be found in [5].

Hence, the finite volume discretization of the problem (1) on the control volumes \( D \in \mathcal{D}_h \) yields

\[
- \sum_{K \in D} \int_{\partial D \cap K} \mathbb{K}_K \nabla u \cdot \mathbf{n}_K \, d\sigma = \int_D f(x) \, dx, \tag{2}
\]

where \( \mathbb{K}_K \) is an approximation of the diffusion tensor on the triangle \( K \). We note the elementary diffusion terms by

\[
a_{12}(K) = |K| \mathbb{K}_K \frac{|\Sigma_{opp}^{12}|}{2|K|} \mathbf{n}_1^{opp} \cdot \mathbf{n}_2^{opp}, \tag{3}
\]

\[
a_{13}(K) = |K| \mathbb{K}_K \frac{|\Sigma_{opp}^{13}|}{2|K|} \mathbf{n}_1^{opp} \cdot \mathbf{n}_3^{opp}. \tag{4}
\]

Finally, the finite volume scheme for the diffusion equation is written as

\[
\sum_{K \in D} a_{12}(K)(u_2 - u_1) + a_{13}(K)(u_3 - u_1) = \int_D f(x) \, dx. \tag{5}
\]

3 A posteriori error estimates

The interest of these estimates is to provide bounds on the error that can be evaluated as soon as the approximate solution has been calculated as proposed by Vohralik [3]. In this section, we present and analyze our a posteriori error estimator. It is founded on a post-processing of the finite element solution \( u_h \). The set of vertices of \( \mathcal{T}_h \) is designated by \( \mathcal{X}_h \). We examine a dual mesh \( \mathcal{D}_h \) in which we define the centers of gravity of the triangles and the midpoints of the edges; see Figure 2 where solid lines represents the primal mesh and dashed lines the dual one. We point out that each \( D \in \mathcal{D}_h \) is associated with one center vertex \( S_i \in \mathcal{X}_h \). The corners of an element \( D \) of the dual mesh are the midpoints of the edges of the primal mesh which have the vertex \( S_i \) as an endpoint and the centers of gravity of all triangles \( K \in \mathcal{T}_h \) with \( S_i \) as vertex. Within the framework of finite volume methods, this dual mesh is widely known and largely approved see [6].

A third mesh is generated from the intersection of the primal \( \mathcal{T}_h \) and the dual mesh \( \mathcal{D}_h \). If we want to obtain a simplicial triangulation \( \mathcal{S}_h \), the introduction of additional edges is required. The elements \( t \in \mathcal{S}_h \) refer to sub-triangles of the elements \( K \in \mathcal{T}_h \) which are subdivided into six sub-triangles. This method is detailed as follows. The center of gravity of \( K \) is connected with the three vertices and with the three midpoints of the edges of \( K \), in such a manner that six subelements are obtained;
refer to the right picture of Figure 2. By construction the area of each subelement $t$ of $K$ is equal, and thus $|t| = |K|/6$.

![Figure 2: Reconstruction of dual control volumes and simplicial mesh.](image)

We note the space of Raviart-Thomas-Nedelec of order 0 by $RTN^0$, and for each $t_1, t_2 \in S_h$ we have $\sigma = \sigma_{t_1, t_2}$ is a common face between $t_1$ and $t_2$.

### 3.1 Construction of $t_h$ by direct prescription

We define $t_h \in RTN^0(S_h) = \{ v \in H(div; \Omega); v|_{t_i} \in RTN^0(t_i) \}$ by,

$$ t_h \cdot n_\sigma = -\left( -\frac{\mathbb{K}_{t_2}}{\mathbb{K}_{t_1} + \mathbb{K}_{t_2}} (t_h \cdot n_{\sigma|t_1})|_{\sigma} - \frac{\mathbb{K}_{t_1}}{\mathbb{K}_{t_1} + \mathbb{K}_{t_2}} (t_h \cdot n_{\sigma|t_2})|_{\sigma} \right) $$(6)

$t_h$ verifies the following condition on each control volume $D \in \mathcal{D}_h$ given by:

$$ -\sum_{\gamma \in \partial D} \int_{\gamma} t_h \cdot \vec{n}_{\gamma} ds = \int_D f(x) dx, \quad (7) $$

Here we use harmonic averaging for the construction of $t_h$ which gives more robustness of estimates and respects discontinuous coefficients in the energy norm. Also the diffusion coefficient $\mathbb{K}$ is piecewise constant on $\mathcal{D}_h$. Thus, the error estimator we consider is given as:

$$ |||u - u_h||| \leq \left( \sum_{D \in \mathcal{D}_h} (\eta_{R,D} + \eta_{DF,D})^2 \right)^{\frac{1}{2}} $$

where the energy norm is defined by:

$$ |||u - u_h||| = ||\mathbb{K}^{\frac{1}{2}} \nabla (u - u_h)||| $$

and the a posteriori error estimator is

$$ \eta(T_h, D, u_h) = \left\{ \sum_{D \in \mathcal{D}_h} (\eta_{R,D} + \eta_{DF,D})^2 \right\}^{\frac{1}{2}}. $$

The residual error estimate $\eta_{R,D}$ and the flux error estimate $\eta_{DF,D}$ are defined for each element $D \in \mathcal{D}_h$ as:

$$ \eta_{DF,D} = ||\mathbb{K}^{\frac{1}{2}} \nabla V_h + \mathbb{K}^{-\frac{1}{2}} t_h||_D, \quad \eta_{R,D} = m_{D,a} ||f - \nabla \cdot t_h||_D, $$

where the constant $m_{D,a}$ is defined using the constant of the Poincaré and the Friedrichs inequalities, see [3] for more details.
4 Mesh adaptation techniques

Two techniques have been considered in this work for mesh adaptation namely the Newest-Vertex-Bisection (NewestVB) method studied in [1] and the so-called Adapt technique proposed in [2]. In the NewestVB method, for each element $T \in \mathcal{T}_h$ we define a newest vertex denoted by $V_T$ and $E_T$ is the longest edge of $T$ with $V_T$ is the vertex opposite to $E_T$. Then we divide the elements by joining $V_T$ to the middle $I$ of $E_T$ and $I$ becomes the “newest vertex” of each of the two created triangles, see Figure 3 for an illustration.

![Figure 3: Illustration of Newest-Vertex-Bisection adaptation.](image)

The main idea for the Adapt strategy for refinement is to subdivide the reference triangle into four new triangles by joining the faces midpoints. Then the conformity satisfies that two neighboring triangles have only one common face. The algorithm is repeated until level of refinement of all cells is equal to zero see Figure 4. If the levels of refinement among neighboring cells are different, we have to satisfy the mesh conformity. For each cell, we check the level of refinement of neighboring cells with common face. In case that at least two neighboring cells have higher level of refinement, we increase the level of refinement for given cell and it’s divided by standard algorithm. In case that only one neighboring cell has higher level of refinement, we split the given cell into two triangles (see Figure 4, part Level 2).

![Figure 4: Multi-level refinement in 2D](image)

In this present work we also combine both NewestVB and Adapt method for mesh adaptation and call it AdaptNVB. First we proceed by refining our mesh by the Adapt strategy, then for the conformity one uses the NewestVB method. One notices that there is no more propagation of the refinement with this strategy compared to the other classical version of Adapt method. It is observed that there is no more propagation of the refinement on the triangles $T_1, T_2, T_3, T_4$ and $T_5,$ see Figure 5.
5 Numerical results

We solve the two-dimensional diffusion problem \((1)\) with high heterogeneous diffusion coefficient \(K\) proposed in \([3]\). Here, the domain \(\Omega = [-1,1]^2\) is divided into four subdomains \(\Omega_i\) \((i = 1, \ldots, 4)\) and the diffusion coefficient \(K\) is constant in each subdomain \((K|_{\Omega_i} = \kappa_i, \ i = 1, \ldots, 4)\). Finally, functions \(f\) and \(g\) are selected such that the exact solution \(u\) in polar coordinates \((r,\theta)\) is

\[
  u|_{\Omega_i}(r,\theta) = r^\alpha \left( a_i \sin (\alpha \theta) + b_i \cos (\alpha \theta) \right), \quad i = 1, \ldots, 4, \tag{8}
\]

where \(\alpha\) is a fixed parameter in our simulations, \(a_i\) and \(b_i\) are constants related to the subdomains \(\Omega_i\).

Permeability values \(\kappa_1 = \kappa_3 = 100\) and \(\kappa_2 = \kappa_4 = 1\):

\[
\begin{align*}
  &\alpha = 0.12690207, \\
  &a_1 = 1, \quad b_1 = 0.1 \\
  &a_2 = 2.96039604, \quad b_2 = -9.60396040 \\
  &a_3 = -0.882756592, \quad b_3 = -0.480354867 \\
  &a_4 = -6.45646175, \quad b_4 = 7.70156488
\end{align*}
\]

In what follows we denote by DoF the number of elements in the considered mesh, and by \(\epsilon_1\) the energy error, by \(\epsilon_2\) the \(L^2\)-error, and by \(f_\eta\) the effective index defined as the ratio between the related estimator and the true error. In all the computations reported herein, the linear systems of algebraic equations are solved using a direct solver.

First we present a numerical comparison between the NewestVB approach and our approach AdaptNVB for mesh refinements. To this end we display in Figure 6 the obtained final meshes using both approaches along the initial mesh. For this case, both approaches show similar features and capture the singularity around the origin. However, by comparing the errors and the effective indices in Figure 7, one can easily see oscillations in the results obtained using the NewestVB approach which are completely absent from those obtained using the AdaptNVB approach see Figure 8. To further quantify these effects we summarize in Table 1 for selected iterations the number of elements DoFs, the error estimator \(\eta\), the energy error \(\epsilon_1\), the \(L^2\) error \(\epsilon_2\), the effective index \(f_\eta\) and the CPU time for NewestVB and Adapt approaches. As expected the NewestVB approach requires about twice the CPU times needed for the Adapt approach. This is consistent with the fact that to divide a triangle in four sub-triangles, the NewestVB approach uses two iterations compared to one single iteration in our Adapt approach. The reduction of estimator and errors \(\epsilon_1\) and \(\epsilon_2\) in Table 1 is also faster in AdaptNVB approach than in the NewestVB approach.
Figure 6: Refined mesh, error distribution using NewestVB approach (left), and refined mesh, error distribution using Adapt approach (right).

Figure 7: Errors and effective indices for NewestVB approach
Table 1: Comparison between the NewestVB and AdaptNVB approaches. CPU times are in seconds.

<table>
<thead>
<tr>
<th>Iter</th>
<th>DoFs</th>
<th>$\eta$</th>
<th>$\epsilon_1$</th>
<th>$\epsilon_2$</th>
<th>$f_0$</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>128</td>
<td>103.3915</td>
<td>15.836</td>
<td>0.30586</td>
<td>6.5289</td>
<td>0.679</td>
</tr>
<tr>
<td>12</td>
<td>942</td>
<td>44.08</td>
<td>8.8284</td>
<td>0.074364</td>
<td>4.993</td>
<td>0.074</td>
</tr>
<tr>
<td>45</td>
<td>4608</td>
<td>9.7158</td>
<td>6.5221</td>
<td>0.027443</td>
<td>1.2734</td>
<td>2.575</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>112.741</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iter</th>
<th>DoFs</th>
<th>$\eta$</th>
<th>$\epsilon_1$</th>
<th>$\epsilon_2$</th>
<th>$f_0$</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>240</td>
<td>103.3915</td>
<td>15.836</td>
<td>0.30586</td>
<td>6.5289</td>
<td>0.671</td>
</tr>
<tr>
<td>6</td>
<td>1040</td>
<td>42.7369</td>
<td>10.677</td>
<td>0.081534</td>
<td>4.0027</td>
<td>0.918</td>
</tr>
<tr>
<td>29</td>
<td>7296</td>
<td>6.9263</td>
<td>5.104</td>
<td>0.022277</td>
<td>1.357</td>
<td>3.995</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>56.647</td>
</tr>
</tbody>
</table>

Next we compare the numerical results obtained using the conventional AdaptNVB approach to a proposed nested approach using three levels for adaptNVB. Figure 9 illustrates the obtained results for adapted meshes, errors and effective indices for both methods. Again, the refinement is concentrated in the centre of the computational domain in both methods but the conventional approach exhibits refinements in many locations where the solution is smooth while the refinement in our nested approach remains concentrated around the center as expected. The profiles of errors in Figure 9 show a faster convergence in our nested approach compared to the conventional AdaptNVB method. These features can be clearly seen in Table 2 where errors and CPU times are presented for several iterations using both approaches. It is evident that for a fixed error in both methods, the NewestVB approach requires more iterations and CPU times than the AdaptNVB approach. This is mainly due to the fact that in the NewestVB approach each refinement is followed by solving the diffusion problem on the refined mesh whereas the AdaptNVB approach requires the solution of the diffusion problem only after three levels of refinements.
Figure 9: Results using conventional approach (left) and using nested approach (right).
Table 2: Comparison between the conventional and nested approaches. CPU times are in seconds.

<table>
<thead>
<tr>
<th>Iter</th>
<th>DoFs</th>
<th>$\eta_1$</th>
<th>$\epsilon_1$</th>
<th>$\epsilon_2$</th>
<th>$f_1$</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>240</td>
<td>103.3915</td>
<td>15.836</td>
<td>0.30586</td>
<td>6.5289</td>
<td>0.71</td>
</tr>
<tr>
<td>6</td>
<td>2256</td>
<td>42.0556</td>
<td>10.298</td>
<td>0.079115</td>
<td>4.0838</td>
<td>1.45</td>
</tr>
<tr>
<td>20</td>
<td>9903</td>
<td>10.617</td>
<td>5.2289</td>
<td>0.020684</td>
<td>2.0305</td>
<td>5.5</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>60.691</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iter</th>
<th>DoFs</th>
<th>$\eta_1$</th>
<th>$\epsilon_1$</th>
<th>$\epsilon_2$</th>
<th>$f_1$</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1046</td>
<td>103.3915</td>
<td>15.836</td>
<td>0.30586</td>
<td>6.5289</td>
<td>0.39</td>
</tr>
<tr>
<td>3</td>
<td>3054</td>
<td>37.5234</td>
<td>9.6343</td>
<td>0.065962</td>
<td>3.8948</td>
<td>1.5</td>
</tr>
<tr>
<td>10</td>
<td>10082</td>
<td>7.9307</td>
<td>5.5007</td>
<td>0.024378</td>
<td>1.4418</td>
<td>6.8</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>35.13</td>
</tr>
</tbody>
</table>

6 Algorithm

Algorithm parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Signification</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_{\text{adv}}$</td>
<td>Table have the values 0 or 1, 1: the triangle that contains this vertex must be refined, and 0 otherwise</td>
</tr>
<tr>
<td>Marker</td>
<td>Table that indicates whether an edge should be marked or not.</td>
</tr>
<tr>
<td>$N_{\text{lev}}$</td>
<td>maximum number of multi-level refinement.</td>
</tr>
<tr>
<td>$N_{\text{adv}}$</td>
<td>contains 0 or 1, 0: the edge is not yet created, 1: the node is already created.</td>
</tr>
<tr>
<td>$N_{\text{Ref}}$</td>
<td>Maximum level of refinement.</td>
</tr>
</tbody>
</table>

First, we construct the data structure: primal mesh $T_h$, dual mesh $D_h$, simplicial mesh $S_h$, then we compute the estimators on the edges of simplicial mesh which will enable us to calculate the estimator in the node surrounded by these edges, using the estimator we define a sill to obtain the $I_{\text{adv}}$ in the node. After that, we compute the $I_{\text{adv}}$ in the cell this information will allow us to know if the cell must be adapted or not and its level of refinement, see Algorithm 1.

We give the following table which contains the different parameters of our algorithm:

```
Algorithm 1: Nested Algorithm

1 for $t:=1$ to $N_{\text{Ref}}$ do
2     Construction of the data structure of the new mesh
3     Computation of the numerical solution
4     Computation of the estimator
5     Computation of the refinement criterion according to the estimators
6     Flags on of the elements to be refined
7 for $k:=1$ to $N_{\text{lev}}$ do
8     Mesh refinement with Adapt strategy
9     Conformity with NewestVB strategy
10    Flags on new created triangles to be refined
11 end
12 end
```
Conclusion

In this work we describe our coupled mesh adaptation strategy "AdaptNVB", we compare it with the new proposed "Nested" approach and we show the gain in terms of CPU time and number of iterations. In addition, the conservation of the property of non-propagation of refinement levels is guaranteed by our AdaptNVB coupled strategy.

References


