On Multi-Mesh H-Adaptive Methods

1. INTRODUCTION

It is a challenging problem to design an efficient adaptive mesh method in high dimensions, especially when the underlying solution develops complicated structures and becomes singular or nearly singular. In practice, there are three types of adaptive methods, namely h-method (mesh refinement), p-method (order enrichment) and r-method (mesh redistribution) (see, e.g., [1,2,9,11,13]). The r-method is also known as moving mesh method. An adaptive h-method adjusts meshes for the solution of a problem by refining or coarsening the meshes locally and thus inserting or removing elements locally. An adaptive p-method adjusts local approximation function spaces on selected elements by varying the dimension of the local spaces. An adaptive r-method moves mesh grid points to where they are most needed while keeping the total number of grid points unchanged. Finally,
An adaptive $h$-$p$-method combines the $h$-and $p$-method by refining meshes locally in some parts of domains and enlarging local approximation function spaces in other parts. Successful implementation of the adaptive mesh strategy can enhance the accuracy of the numerical approximations and also reduce the computational cost. Over the past two decades, several very efficient adaptive techniques for the numerical solution of partial differential equations have been developed. The existing adaptive libraries include ALBERT [14], deal.II [3,4] and UG [5]. These libraries are based on the $h$-adaptive strategy utilizing information extracted from a posteriori error estimators, which are computable indicators to the local error between the true solution and the finite element approximations. The indicator often plays a key role in the adaptive method. (see [8] for some recent results to get the indicator by superconvergence analysis)

This work also uses the $h$-refinement method. It is noted that the libraries mentioned above have only considered mesh adaptation on a single mesh. However, solutions of practical problems may involve several components, which may have different natures or singularities. For example, the 2D Navier–Stokes equations can be expressed in terms of two variables, i.e., vorticity and streamfunction, and these two variables often have different regularities. In general, the streamfunction is smoother than the vorticity, and as a result, more grid points are often required for the vorticity approximation, and less for the streamfunction. In such a case, it may be inefficient to approximate the two solution components on a single mesh. Multi-mesh strategies proved particularly useful in computational optimal control, where the optimal control and the state often have very different natures. It was shown in [7] that multi-meshes can greatly reduce computational work in solving optimal control problems. Although the idea of using multi-meshes seems to be very natural, it has been considered very expensive to implement adaptive multi-meshes in practical computations. A good implementation has to be very efficient as otherwise the benefits brought by this strategy can easily be consumed by the extra work introduced. The aim of this research is to develop efficient adaptive multi-mesh implementation strategies for the solutions whose components have very different natures or singularities. These strategies have been implemented in a finite element package AFEpack, and have been successfully used in computational optimal control. The key idea here is to construct multi-meshes in the way that it is convenient to retrieve the essential information among the elements in different meshes.

The paper is organized as follow. In Sec. 2, we provide the motivations of the multi-mesh adaptive approach by considering a model elliptic system. In Sec. 3, the implementation details for the multi-mesh strategy...
To illustrate the motivation of the multi-mesh $h$-adaptive approach, we consider a simple model equation

\[
-\nabla \left( A_0 \nabla u_0 \right) + \sum_{i=0}^{c_0} c_0_i u_i = f_0, \\
-\nabla \left( A_1 \nabla u_1 \right) + \sum_{i=0}^{c_1} c_1_i u_i = f_1, 
\]

on a two-dimensional domain $\Omega$ with homogeneous Dirichlet boundary condition:

\[
u_i \bigg|_{\partial \Omega} = 0, \
i = 0, 1, 
\]

where the coefficient functions $A_0$ and $A_1$ have different locations of singularity in the domain and the coefficients $c_0$ and $c_1$ are chosen such that the system is well-posed. A simple choice of the coefficients $c_0_i$ is to let the matrix $(c_0_i)$ symmetric positive definite.

Although this is an elliptic system, the two components of the solution $u_0$ and $u_1$ are only weakly coupled. The standard weak formulation of the system is as follows:

Find $(u_0, u_1) \in H^1_0(\Omega) \times H^1_0(\Omega)$ such that

\[
\int_\Omega \left( A_0 \nabla u_0 \nabla v_0 + \sum_{i=0}^{c_0} c_0_i u_i v_0 \right) \, dx = \int_\Omega f_0 v_0 \, dx, \quad \forall v_0 \in H^1_0(\Omega),
\]

\[
\int_\Omega \left( A_1 \nabla u_1 \nabla v_1 + \sum_{i=0}^{c_1} c_1_i u_i v_1 \right) \, dx = \int_\Omega f_1 v_1 \, dx, \quad \forall v_1 \in H^1_0(\Omega).
\]

Since the coefficient matrices $A_0$ and $A_1$ have different locations of singularity, in general the solution $u_0$ and $u_1$ are singular at different positions in $\Omega$. To save the computational cost, it is desirable to approximate $u_0$ and $u_1$ on two different meshes. Let us discretize (2.3) formally.
\[ \int_{\Omega} \left( A_0 \nabla u_0^h \nabla v_0^h + \sum_{i=0}^{c_0} c_0^i u_i^h v_0^h \right) \, dx = \int_{\Omega} f_0 v_0^h \, dx, \quad \forall v_0^h \in V_0^h, \]

\[ \int_{\Omega} \left( A_1 \nabla u_1^h \nabla v_1^h + \sum_{i=0}^{c_1} c_1^i u_i^h v_1^h \right) \, dx = \int_{\Omega} f_1 v_1^h \, dx, \quad \forall v_1^h \in V_1^h. \]

It is obvious that the above problem is well-posed. It can be shown that the leading terms of the posterior error estimator for this discretization are given by

\[ \| u_0 - u_0^h \|_{\partial \tau_0, \tau_0} \quad \text{and} \quad \sum_{l_0 \in \partial \tau_0, \tau_0} \int_{l_0} \left[ A_0 \nabla u_0^h \cdot n_{l_0} \right]^2 h_{l_0} \, dl, \]

\[ \| u_1 - u_1^h \|_{\partial \tau_1, \tau_1} \quad \text{and} \quad \sum_{l_1 \in \partial \tau_1, \tau_1} \int_{l_1} \left[ A_1 \nabla u_1^h \cdot n_{l_1} \right]^2 h_{l_1} \, dl, \]

where \( n_{l_0} \) is the outward unit normal of \( \tau_i \) and \( [\cdot] \) denotes the jump along the edge of the element. In principle, we can adopt the standard \( h \)-adaptive procedure as:

1. solve the PDEs on the current mesh;
2. calculate the posterior error estimators;
3. adapt the mesh based on the computable estimators;
4. go to (1) if the solution is not good enough.

\( u_0^h \) and \( u_1^h \) are defined on two different meshes then there is no straightforward method to calculate the mixed terms involving \( c_0^1 u_1^h v_0^h \) and \( c_1^0 u_0^h v_1^h \). This example is only for illustration as here we at least know the locations of the singularities. There seems to be two possible ways to do this: the first one is to build up an inter-relationship data structure beforehand and the second one is to integrate all the quantities to a much finer reference mesh. If both meshes are almost uniform, or one of the two meshes is extremely coarse, both ways will be feasible. Unfortunately, this is not the case for problem (2.1). A satisfactory mesh to approximate a function with a local singularity will generally locate most of its grid nodes to a small part of the domain. In principle, the use of the highest possible order basis functions in these elements is expected to increase the smoothness of the approximate solution. The same remarks is the stated every element of one mesh has to contain part with mostly finite element of the other mesh.
On Multi-Mesh H-Adaptive Methods

As a result, we can not save much by using such data structure. The second method above is also expensive either, since the reference mesh is generally a very fine uniform one which is finer than both solution meshes at every part of the domain.

The key problem in implementing adaptive multi-meshes efficiently is to build-up the relationship among the different meshes used. This issue will be addressed in Sec 3.

3. ALGORITHM AND IMPLEMENTATION

As discussed in the last section, for arbitrary two meshes it is very costly to build up useful relationship between them. To construct efficient adaptive multi-mesh algorithms, the meshes must have some known relationship. If the relationship is too close or too loose, we either lose flexibility of multi-mesh, or fail to implement it efficiently. It suggests that a low level data structure should be used, which can hold the common information of different meshes. To this end, our choice is the hierarchy geometry tree or hierarchical grids. By hierarchical we mean that the structures of the grid are described hierarchically (points, lines, quadrilaterals etc.) and the refinement has to be made hierarchically as well, as opposed to unstructured grids. The hierarchical data structure serves the simplicity of programming and allows fast algorithms for grid refinement and the numerics, such as multi-grid solvers.

For ease of description, we only describe the algorithm for 2D simplex triangulation. The algorithms described below can be extended to other elements, and three dimensions.

3.1. Hierarchy Geometry Tree

Let us consider a given regular simplex triangulation \( T \) on a 2D domain \( \Omega \) and denote an element as \( \tau \). The terminology "regular" here means that two triangles will share a whole edge if they share part of an edge. The triangulation is composed by triangles, edges and points. To distinguish the element in the triangulation and the element in the finite element space, we call the element in the triangulation "element geometry". All the triangles, edges and points are called "geometries". If an edge is one of the edges of a triangle, we say it belongs to the triangle and if a point is one of the end-points of an edge, we say the point belongs to the edge. All geometries in the triangulation then have a belonging-to relationship.

A triangle in the triangulation can be refined into four smaller triangles, and in this refinement operation every edge of the triangle is refined...
Fig. 1. Refine model for interval (left) and triangle (right).

Fig. 2. Sequential uniform refined mesh (up) and its hierarchy tree (bottom).

to two smaller edges, (see Fig. 1). After refining every triangle in the triangulation, we can get a finer regular triangulation $T_1$. This is in fact a uniform refinement of $T_0$. Doing this successively gives a series of triangulations $\{T_n\}$. A triangle $\tau_1$ in $T_n$ is called a child of the triangle $\tau_0$ in $T_{n-1}$ if $\tau_1$ is located in $\tau_0$. This gives a four-fork tree rooted from every triangle in $T_0$. We give a virtual parent for each triangle in $T_0$ and all these four-fork trees rooted from the triangles in $T_0$ is constructed as a tree data structure, as shown in Fig. 2. This tree is called “hierarchy geometry tree.”
It is obvious that we should not construct a uniform hierarchy geometry in the adaptive algorithm. We adopted a "refine when required" strategy to maintain the hierarchy geometry tree in our implementation. A triangle in the hierarchy geometry tree will be refined only when a mesh adaptation is necessary based on some rules we mentioned.

3.2. Mesh

A mesh is a cluster of elements which can cover the whole domain but can be composed of different elements. The refinement of the common element will cause the refinement of all the others from the hierarchy geometry tree, or we say all the elements in the whole domain will be refined. A mesh obtained from a finite sub-tree of the hierarchy geometry is called a finite mesh. The considered mesh will be composed by the finest elements from the hierarchy that satisfy the rules. A considered mesh should be composed of the finest element, the other element in the hierarchy that is the finest one in the whole domain. The intersection corresponding to this mesh is then a mesh in general meaning (see Fig. 3). A mesh obtained by the leaf nodes of a sub-tree of the hierarchy geometry tree is called a semi-regular mesh. Obviously, a mesh is uniquely determined by the complete sub-tree and versa vita. For all those meshes coming from this hierarchy geometry tree, we in fact have already got the inclusion relation from the data structure.

Since the irregular meshes can have extremely complex structure, it will be a difficult work to build efficiently a finite element space on. Even if we can build such a finite element space, it is generally not good to approximate the problem under consideration in this finite element space because the mesh size in this mesh will vary very dramatically. Therefore, it is suggested to accept only those meshes with relatively simple and smooth structure such that building finite element spaces on such meshes will be simple and efficient. We call these acceptable meshes as semi-regular meshes.

As seen before, there is a 1-1 mapping from the complete sub-tree of the hierarchy geometry tree to the mesh coming from the hierarchy geometry so that we can regard the complete sub-tree and the corresponding mesh as the same thing. We denote the set of all meshes from the hierarchy geometry tree as \( T \).
Fig. 3. A mesh is an intersection of the hierarchy geometry tree with respect to a complete sub-tree. According to the inclusion relation of those complete sub-trees, there is a partial order on the set of all those complete sub-trees. A precise definition on these orderings will be given as follows.

Definition 3.1. Denote two meshes in \( M(T) \) as \( M_1 \) and \( M_2 \). We say \( M_1 \) less than \( M_2 \), denoted by \( M_1 \prec M_2 \), if and only if the complete sub-tree of \( M_1 \) is a sub-tree of the complete sub-tree of \( M_2 \). Moreover, the intersection of \( M_1 \) and \( M_2 \), denoted by \( M_1 \cap M_2 \), is the unique element in the set \( \{ \tilde{M} \in \text{Child}(M_1) \cap \text{Child}(M_2) : \tilde{M} \prec M \} \), where \( \text{Child}(M) = \{ \tilde{M} \in M(T) : \tilde{M} \prec M \} \).

3.3. Adaptation

The irregular meshes are very flexible since there are no requirement at all to neighboring elements. It is easy to adapt an irregular mesh to...
On Multi-Mesh H-adaptive Methods

Another irregular mesh can be adapted by using the same number of triangles as before, requiring the adaptive refinement of the mesh. The adaptation operation can be achieved by refining and coarsening every single element without considering its neighbors. The mesh adaptation algorithm is based on the equi-distribution principle and consists of two steps. Step 1 is to adapt the mesh according to the indicator to form a new irregular mesh and step 2 is to process the irregular mesh into a semi-regular mesh. The step 1 is discussed here and the step 2 is left to the next subsection.

One of the basic ingredients of an adaptive finite element method is an a posteriori error estimator. While a general theory exists for these estimators in the case of linear and mildly nonlinear problems [15], highly nonlinear problems usually still need special treatments [6, 12]. An adaptive method is guided by such an error estimator, and optimizes the mesh by equi-distributing the local indicator values over all mesh elements, while the total estimate is below a given tolerance. An error estimate is normally of the form

$$\|u - u_h\| \leq \sum_{\tau \in T} \tau C(u_h),$$

where \(\alpha\) is the accuracy order and \(C(u_h)\) is of order \(O(1)\). If the indicator is large enough on \(\tau\), then the triangle will be refined, and if on a patch of triangles the indicator are all very small, then these triangles will be coarsened into a larger triangle. More precisely, assume that a tolerance \(\epsilon\) is given and the dimension of \(\Omega\) is two. The adaptation algorithm is given as follows:

1. **Local refinement.** For an element \(\tau\) with an indicator \(E_{\tau,i}\), if \(E_{\tau,i} > 2^{N + \alpha} \epsilon\) then refine \(\tau\) and set the indicators of all its children as \(E_{\tau,i}/2^{N + \alpha}\).

2. **Local coarsening.** For a triangle \(\tau\) whose children are all elements of the given mesh, if \(\sum_{\tilde{\tau} \in \text{Child}(\tau)} E_{\tilde{\tau}} < 2^{-\alpha} \epsilon\), where \(\text{Child}(\tau)\) denotes the set of children of \(\tau\), then set \(\tau\) as a single element of the mesh and set the indicator of \(\tau\) as \(2^{\alpha} \sum_{\tilde{\tau} \in \text{Child}(\tau)} E_{\tilde{\tau}}\).

In the implementation, the algorithm for coarsening the mesh can be simplified as follows.

1. **Lump the indicator.** For each node in the sub-tree of the geometry hierarchy tree, if the node is an element geometry, then lump the indicator of all its children as

$$\sum_{A \in \text{Child}(\tau)} A_{ij}.\$$

where \(A_{ij}\) denotes the set of children of \(\tau\), there are \(i\) as a single element of the mesh and set the indicator of \(\tau\).

In the implementation, the algorithm for coarsening the mesh can be simplified as follows:

1. **Lump the indicator.** For each node in the sub-tree of the geometry hierarchy tree, if the node is an element geometry, then lump the indicator of all its children as

$$\sum_{A \in \text{Child}(\tau)} A_{ij}.$$
Fig. 4. Indicator distribution and lumping.

In the mesh, the indicator is given; if the node is not an element geometry of the mesh, the indicator is set as (see Fig. 4)

$$\alpha \sum_{\tilde{\tau} \in \text{Child}(\tau)} E_{\tilde{\tau}}.$$ 

By such an operation, the indicator can be obtained from the top (leaf node) to the bottom (root node) level by level in the sub-tree.

(3.2.2) Coarsening. Search from the bottom to the top of the sub-tree, if the indicator for a node $$E_\tau$$ satisfies $$E_\tau < 2N + \alpha \epsilon$$, then all of its descendant are removed.

For the triangle refined, its children will inherit an indicator as in (3.1.1). If the indicator inherited from the parent node is still too large, then the children can be refined again by also repeating Algorithm (3.1.1). Of course, the operation should not be too intensive since the indicator will not be so reliable after several continuous inheritings. Our purpose is to make full use of the information provided by the indicator. The algorithm described above does provide us such ability in making full use of the information, which can be expected to be more efficient than those algorithm with one step adaptation. The advantage of this adaptation is as follows.

- easy to be implemented. It is seen from the algorithm that the refining and coarsening are in fact implemented by the same algorithm;
On Multi-Mesh H-adaptive Methods

- Easy to be parallelized. The adaptation can be applied to different parts of the mesh at the same time;
- Easy to do multi-level adaptations. The proposed algorithm can perform mesh refining and coarsening more than one step. This indicates that the information of the indicator can be fully used.

3.4 Semi-regularization

The mesh generated by the algorithm described in the last subsection is generally so complex that building a finite element space on it may be expensive and inefficient. Now we propose another operation called semi-regularization to get a semi-regular mesh on which a finite element space can be built. In principle, a semi-regular mesh should satisfy the following requirements:

(a): It should be only slightly more complex than a regular mesh;
(b): The intersection of two semi-regular meshes should be still semi-regular.

The requirement (a) above is a natural requirement. The requirement (b) can guarantee that the semi-regularization is a natural extension of the regularization.

In our 2D case, a regular simplex mesh is filled with triangles that two neighboring triangles will share a whole edge. However, for our 2D semi-regular simplex mesh a triangle can have one special edge shared by two smaller neighboring triangles. This may lead to hanging nodes: a triangle with an additional node on one of its edges. To build finite element spaces on such meshes, we add another basis function at the additional node so that the finite element space is conform. This is illustrated in Fig. 5. It is emphasized that one of the key ingredients of our multi-mesh h-adaptive method is to relax the regularity requirement of the mesh. It allows that different meshes are included with each other locally instead of globally, and at the same time we may build the finite element space on a semi-regular mesh instead of on a regular one.

Below we outline the main steps of the semi-regularization for 2D simplex triangulation:

(3.3.1) on each edge in the mesh, there allows at most one level jump;
(3.3.2) for an element geometry, there allows at most one of its neighbors to be refined.
It can be verified that this definition satisfies the "Intersection Close Condition". It can be shown that the element geometries should not be coarsened. The semi-regularization is then a pure refinement operation. Different from the coarsening operation, refinement is applied on certain single element geometry in the irregular mesh. All coarsening operations in this algorithm have been taken in the previous step, so we only need to operate on some single elements in this step. We call a refinement operation on a single element geometry an "Atom-Op" and denote it by \( P \). The following properties can be verified.
Assume that every Atom-Op $P$ in the semi-regularization algorithm satisfies:

$$M_1 \prec M_2 \Rightarrow P_{M_1} \prec M_2.$$ (3.1)

If $M_2$ is a semi-regular mesh, then the semi-regularization can be achieved within a finite number of operations. We call this condition the Atom-Op Close Condition.

Furthermore, if the "Intersection Close Condition" holds, then the semi-regular mesh obtained by semi-regularizing the irregular mesh $M$ is unique and is in fact the minimal element in the set $\{\tilde{M} \in M(T) | M \prec \tilde{M}, \tilde{M} \text{ is semi-regular}\}$.

We close this section by outlining the 2D semi-regularization algorithm.

Step 1. Traverse the element geometry of the tree, check if its edge is refined and if the children of that edge are refined. If yes, refine this element geometry and then go to next element geometry;

Step 2. If there are no refinement committed in Step 1, go to next step; else go to Step 1;

Step 3. The mesh now satisfies (3.3.1). Now traverse the element geometry of the tree and check if it has more than one refined neighbors. If yes, refine the geometry and then go to next element geometry;

Step 4. If there is no operation committed in Step 3, then stop; else go to Step 1.

It is easy to verify that the above semi-regularization algorithm for two space dimension satisfies "Atom-Op Close Condition".

4. TIME-DEPENDENT PROBLEMS

We point out that the strategies proposed in this paper can be generalized to time-dependent problems:

$$\frac{\partial u}{\partial t} = Lu.$$ (4.1)

For simplicity, we consider the forward-Euler type temporal discretization:

$$u(n+1) - u(n) \Delta t = L(n).$$ (4.2)

The standard single-mesh $h$-adaptive algorithm is as follows:
(4.1.1): Solve (4.2) on the current mesh to get a temporary $u^{(n+1)}$.

(4.1.2): Check if the mesh is satisfactory — often this is done by checking if the mesh is good to approximate $u^{(n+1)}$: if yes, go to next time step; if not, adapt the current mesh to a new one and interpolate $u^{(n)}$ on the new mesh, (4.1.3): go to 4.1.1).

The above procedure implies that the solution on a new time step is obtained by an iteration procedure. In part 4.1.2) above, if the new mesh is not satisfactory then only refinement operation is performed until the overall mesh is satisfactory. In the latter case, i.e., after we get a satisfactory solution $u^{(n+1)}$ on the new mesh — of course the mesh is now generally over-refined — a coarsening will be done based on some coarsening indicator. Often the coarsening indicator can be obtained by patch recovery interpolation. The purpose that coarsening is not undertaken during the intermediate iteration procedure is to keep enough information of $u^{(n)}$ — coarsening may yield the lose of some useful data of $u^{(n)}$ when is it interpolated to a new mesh. The interpolation of $u^{(n)}$ from the old mesh to a new mesh is for the convenience of solving (4.2) whose weak formulation is of the form:

$$
(u^{(n+1)} - u^{(n)}, v^{(n+1)}) = \Delta t \langle L^{(n)}, v^{(n+1)} \rangle,
$$

(4.3)

where $v^{(n+1)}$ is a test function which is defined on a mesh different with $u^{(n)}$. In this case, it is generally expensive to compute the terms $(u^{(n)}, v^{(n+1)})$ and $\langle L^{(n)}, v^{(n+1)} \rangle$. Therefore, some appropriate interpolation for $u^{(n)}$ seems necessary.

Now with the multi-mesh adaptation algorithm, the over-refinement in the iteration procedure can be avoided. Moreover, this procedure is interpolation free since all quantities can be computed directly. The solution procedure can be outlined below:

(4.2.1): Solve (4.2) on the current mesh $M$ to get a temporary $u^{(n+1)}$ using $u^{(n)}$ on mesh $M^{(n)}$,

(4.2.2): Check if the approximation solution for (4.2) is satisfactory at $t = t^{(n+1)}$: if yes, set $M^{(n+1)} = M$ and go to next time step; if not, adapt the current mesh $M$ to a new one;

(4.2.3): go to (4.2.1).

In the above algorithm, the mesh $M$ is initially set as $M^{(n)}$. 
5. EXAMPLES OF APPLICATIONS

In this section we present some numerical results in order to demonstrate the flexibility of our multi-mesh algorithms.

5.1. Weakly Coupled Elliptical System

The model problem in Sec. 2 is designed specifically for the multi-mesh algorithm. It is the simplest model problem to demonstrate the ability of the multi-meshes. The coefficient matrices of Eq. (2.1) are given by

\[ A_0 = a_0 I \]
\[ A_1 = a_1 I \]

with

\[ a_0 = \begin{cases} 10, & x^2 - y^2 \geq 0, \\ 1, & x^2 - y^2 < 0, \end{cases} \]

\[ a_1 = \begin{cases} 10, & x^2 + y^2 < 1/2, \\ 1, & x^2 + y^2 \geq 1/2. \end{cases} \]  

(5.1)

Moreover, the matrix \( C = (c_{i,j}) \) is defined by

\[ C = \begin{pmatrix} 11/4 & 1/4 \\ 1/4 & 1 \end{pmatrix}. \]  

(5.2)

It can be shown that \( u_0 \) has singularity along the line \( x + y = 0 \) and \( x - y = 0 \) while \( u_1 \) has singularity along the circle \( x^2 + y^2 = 1/2 \). Figures 6 and 7 present the mesh and the solution for \( u_0 \) and \( u_1 \), respectively, from which it can be seen that the meshes for the two solution components are very different.

\[ \text{Fig. 6. Weakly coupled elliptical system: mesh (left) and solution (right) for } u_0. \]  

\[ \text{Fig. 7. Weakly coupled elliptical system: mesh (left) and solution (right) for } u_1. \]
5.2. Optimal Control Problem

Optimal control problems have been studied extensively. In fact, our multi-mesh adaptive method is originally developed for solving the optimal control problems where it is common to see different behaviors of solution singularity. A model optimal control problem with an elliptic constraint equation and a distributed control is as follows:

$$
\begin{align*}
\min_{g(y), h(u)} & \quad g(y) + h(u) \\
\text{s.t.} & \quad -\Delta y = u + f, \quad y|_{\partial \Omega} = 0, \quad u \geq 0,
\end{align*}
$$

(5.3)

where $y$ is the state, $u$ is control and the objection functions are given by

$$
\begin{align*}
g(y) &= \frac{1}{2} \int_{\Omega} (y - y_0)^2, \\
h(u) &= \frac{1}{2} \int_{\Omega} (u - u_0)^2.
\end{align*}
$$

(5.4)

It is well known that the optimal control problem (5.3) is equivalent to its optimality condition:

$$
\begin{align*}
-\Delta y &= u + f, \quad y|_{\partial \Omega} = 0, \\
-\Delta p &= y - y_0, \quad p|_{\partial \Omega} = 0, \\
(u - u_0 + p, v - u_0) &= 0, \quad \forall v \geq 0,
\end{align*}
$$

(5.5)

where $p$ is the co-state. In [7], sharp a posteriori error estimators for the discretization of (5.3) and (5.5) are established. In [10], a moving grid algorithm is also proposed to solve the control problem (5.3).
The weak formulation of the optimal control problem (5.3) is as follows: Find \((y,u) \in H^1_0(\Omega) \times L^2(\Omega)\) such that
\[
\min g(y) + h(u) s.t. \int (\nabla y, \nabla w) = (u + f, w), \quad \forall w \in H^1_0(\Omega), u \geq 0.
\] (5.6)

Similarly, the weak formulation of the optimality condition (5.5) is as follows: Find \((y,u,p) \in H^1_0(\Omega) \times L^2(\Omega) \times H^1_0(\Omega)\) such that
\[
\int (\nabla y, \nabla w) = (u + f, w), \quad \forall w \in H^1_0(\Omega),
\int (\nabla p, \nabla q) = (y - y_0, q), \quad \forall q \in H^1_0(\Omega),
\int (u - u_0 + p, v - u_0) \geq 0, \quad \forall v \geq 0, v \in L^2(\Omega).
\] (5.7)

Given a simplex mesh \(T_h\) for \(y\) and \(p\) and a simplex mesh \(T_{hU}\) for \(u\). The corresponding finite element spaces are \(Y_h\) and \(U_h\) for \(y\) and \(u\). \(Y_h\) and \(U_h\) are chosen as subspaces of \(H^1_0(\Omega)\) and \(L^2(\Omega)\), respectively. The a posteriori error estimator is given by
\[
\|u - u_h\|_{L^2(\Omega)} + \|y - y_h\|_{H^1(\Omega)} + \|p - p_h\|_{H^1(\Omega)} \leq C \sum \eta^2_i.
\] (5.8)

where
\[
\eta_1 = \int_{\Omega} (\nabla (u_h + p), \nabla (y - y_0)) \quad \text{and} \quad \eta_2 \text{ denotes the jump along the segment edge}.
\] (5.9)
In our computations, $\eta$ is adopted as the indicator to adapt the mesh for $u$ and $\eta^2 + \eta^3$ is adopted as the indicator to adapt the mesh for $y$ and $p$.

If the data in (5.3) are chosen as

$$z = \begin{cases} 0.5, & x_1 + x_2 > 1.0, \\ 0.0, & x_1 + x_2 \leq 1.0 \end{cases}$$

$$u_0 = 1.0 - \sin \pi x_1^2 - \sin \pi x_2^2 + y,$$

$$v_0 = 100 \sqrt{(x_1 - 1)^2 + (x_2 - 1)^2},$$

$$f = 4\pi^4 p - u - 10000/y,$$

then the solutions for (5.3) and (5.5) are given by

$$u = \max(u_0 - p, 0), y = 2\pi^2 p + y_0, p = \sin \pi x_1 \sin \pi x_2.$$ (5.11)

The above solutions indicate that there exists a jump in the control $u$ introduced by $u_0$. Moreover, there is a weak singular point for the state $y$, i.e., $y$ is singular in $H^1$ space. In Fig. 8, we show adaptively refined meshes used for $u$ and $y$, $p$. It is clear that the strong singularity for $u$ requires mesh refinement, and the mesh for $y$ and $p$ is almost uniform, except at the top-right corner where the derivative of $y$ is large.

### 5.3. 3D Examples

This is an optimal control problem for the same as the last example except the domain is three dimensional. The domain $\Omega$ is the unit cube.
On Multi-Mesh H-Adaptive Methods

Fig. 9. Meshes for the 3D control problem: the left is the mesh for $u$ and the right is for $y$ and $p$. The graylevel in the figures denote the value of $u$ and $y$, respectively.

$\Omega = \[0,1\] \times [0,1] \times [0,1]$. The problem and data are as follows.

$$\begin{align*}
\min \frac{1}{2} \int_{\Omega} (y - y_0)^2 \, dx + \frac{1}{2} \int_{\Omega} (u - u_0)^2 \, dx \\
\text{s.t.} \quad -\Delta y = u + f, \quad u \geq 0, \quad (5.12)
\end{align*}$$

where

$$z = \begin{cases} 0.4, & x_1 + x_2 + x_3 > 1.0 \\ 0.0, & x_1 + x_2 + x_3 \leq 1.0 \\ \end{cases}$$

$u_0 = 1.0 - \sin \pi x_1^2 - \sin \pi x_2^2 - \sin \pi x_3^2 + z, \quad y_0 = 0, \quad f = 9\pi^2 p - u_0. \quad (5.13)$

The solution of the problem is given by

$$u = \max(u_0 - p, 0), \quad y = 3\pi^2 p + y_0, \quad p = \sin \pi x_1 \sin \pi x_2 \sin \pi x_3. \quad (5.15)$$

There are both strong and weak discontinuity in the control $u$ while the state $y$ and co-state $p$ are smooth. Fig. 9 shows the adaptive refined meshes for $u$, $y$ and $p$. It is seen that the area of the singularity is well-refined.

6. CONCLUSIONS

We described in this paper an efficient implementation strategy for multi-mesh adaptive finite element method. The proposed method allows...
The efficient use of multiple meshes and proved useful in solving problems whose solution components have different singularity behaviors. A C++ library, AFEPack—Adaptive Finite Element Package—can be obtained at http://www.ukc.ac.uk/cbs/staff/homepage/wbl/data/AFELAB-AFEPACK.htm.

Some application examples on multi-mesh computations can be found at the author's homepage: http://circus.math.pku.edu.cn

ACKNOWLEDGMENTS

The author is grateful to Prof. Wenbin Liu of the University of Kent and Prof. T. Tang of Hong Kong Baptist University for many useful discussions. The first version of AFEPack was completed while the author carried out postdoctoral research work at University of Kent, UK, supported by the EPSRC grant GR/R3198. This research is supported in part by the Joint Research Institute for Applied Mathematics (JRIAM) between Peking University and Hong Kong Baptist University.

REFERENCES


