Available online at www.sciencedirect.com
SCIENCE DIRECT®
Comput. Methods Appl. Mech. Engrg. 194 (2005) 4915–4950
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

3D anisotropic mesh adaptation by mesh modification

Xiangrong Li [a,*], Mark S. Shephard [a], Mark W. Beall [b]

[a] *Scientific Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, USA*
[b] *Simmetrix Inc., 10 Halfmoon Executive Park Drive, Clifton Park, NY 12065, USA*

Received 26 November 2003; received in revised form 9 August 2004; accepted 26 November 2004

Abstract

This paper describes an effective anisotropic mesh adaptation procedure for general 3D geometries using mesh modification operations. The procedure consists of four interacted high level components: refinement, coarsening, projecting boundary vertices and shape correction. All components are governed by an anisotropic mesh metric field that represents the desired element size and shape distribution. The paper presents the application for the procedure in anisotropic adaptive 3D simulations.

Keywords: Mesh adaptation; Anisotropic; Mesh modification; Mesh metric

1. Introduction

It is well recognized that the application of adaptive mesh refinements based on a posteriori mesh discretization error is an effective means to ensure the accuracy of a finite element solution process. The majority of the adaptive mesh refinement procedures focus on the construction of isotropic elements. In general the desired level of solution accuracy can be more efficiently obtained by the adaptive construction of anisotropic meshes. Since the construction of adaptive anisotropic meshes is more complex than the application of isotropic refinement, the anisotropic adaptive procedures developed to date have employed remeshing where an anisotropic mesh is constructed over the entire domain given an anisotropic mesh metric field. This paper presents an alternative approach to adaptive anisotropic mesh adaptation for curved 3D domains based on the application of generalized mesh modification operators that convert a mesh into one that satisfies the given adaptive anisotropic mesh metric field.

* Corresponding author.
E-mail address: xli@scorec.rpi.edu (X. Li).

0045-7825/$ - see front matter
doi:10.1016/j.cma.2004.11.019

The desired size and shape of elements are strongly influenced by the details of the solution field which is constantly evolving in transient problems [21,37]. Although equilateral elements are well suited for solution fields that vary equally in all directions, most physical problems exhibit anisotropic phenomena, which can be extremely strong in cases such as high Reynolds number flow problems and phase change problems. The use of isotropic elements in these cases will force the creation of small elements in all directions, therefore leading to prohibitively large meshes. Adaptively constructing anisotropic meshes requires a procedure capable of effectively creating and maintaining control of anisotropic elements that satisfy both element size and shape distribution as indicated by the error estimation/indication procedures. Curved domains require the adaptivity procedure handle the geometric approximation issue [23,15,26]. In particular, in refining a mesh, new nodes created on curved boundary need to be placed on that boundary.

The current mesh adaptivity techniques can be decomposed into three categories:

- Adaptive re-meshing methods [36,20,1,33];
- Element subdivision methods [2,3,5,7,24,29,30,39,41];
- Fixed order mesh modification procedures [12,13,17,15,10,22,35].

Adaptive re-meshing methods construct an adaptive mesh by regenerating the entire mesh through the application of automatic mesh generation algorithms governed by desired element size and shape information. For example, Peraire et al. [36] and Moller et al. [34] have used adaptive re-meshing based on anisotropic advancing front mesh generation technology and the Inria group [8,20,18,19] applied adaptive re-meshing in terms of anisotropic Delaunay kernel. The benefit of adaptive re-meshing is that curved domains can be considered by the mesh generation algorithm and mesh coarsening is accounted for. However, it is inefficient for refining only a few elements in an already refined mesh, and introduces complexities in the transfer of solution fields from one mesh to another. Element subdivision methods can control element shape by specific split orders, for example, the longest-edge refinement based algorithm by Rivara [38–40], the successive bisection procedure that controls split orders in terms of an affine transformation by Liu and Joe [27,29,30], the red–green algorithm [5,7,24] and newest vertex bisection algorithm [32]. Although effective, these methods tend to over refine and do not consider curved geometry domains and the generalization of changing elements into desired shape. The coarsening process of these algorithms is limited to undoing refinements [24,38,41], therefore, they cannot coarsen past the initial mesh. The third mesh adaptation techniques apply local mesh modifications in a fixed order. For example, Briere et al. [10], de Cougny et al. [15] and Joe et al. [22] have improved the quality of an existing mesh using procedures consisting of four local mesh modification operations: swap, collapse, split and relocation. Buscaglia et al. [12], Castro-Diaz et al. [13], Dompierre et al. [17] and Pain et al. [35] have made applications of these four local mesh modification operations governed by a desired mesh size and shape distribution for adaptivity purpose in two dimension [12,13,17] and recently in three dimensions [35]. The advantage of the third methods is that it is a local process in which curved geometry domains can be accounted for [26], collapse-based mesh coarsening can be applied [15], and the transfer of solution fields can be executed incrementally as each modification is applied. However, in three dimensions, the effectiveness of local mesh modification procedure in terms of efficiency and quality is a strong function of the manner in which various mesh modification operations are considered.

The purpose of the paper is to present a general mesh adaptation procedure that applies mesh modification operations to yield a mesh of the same quality as would be obtained by an adaptive anisotropic remeshing procedure but at less computational cost. In particular, given:

- a general 3D non-manifold domain;
- any tetrahedral mesh of the domain;
- a desired element size and shape distribution throughout the domain;

apply mesh modification operations effectively until the given element size and shape distribution is satisfied with the curved domain boundaries properly approximated.

The paper is structured as follows: Section 2 reviews the representation of anisotropic element shape and size distribution, and presents a feasible alignment criterion between the mesh and the element size/shape distribution for mesh modification purpose. Section 3 describes an anisotropic mesh adaptation procedure governed by the element size/shape distribution, and discusses the technical aspects to achieve effectiveness. Specifically, Section 3.1 presents the overall algorithm, which is an integration of four interacted processes: refinement, coarsening, projecting new boundary vertices and shape correction. Consideration is then given to specific components of mesh coarsening (Section 3.2), mesh refinement (Section 3.4) and shape correction (Section 3.3). The algorithm that projects new boundary vertices has been described in Ref. [26]. Section 4 provides examples to demonstrate the application of the mesh adaptation procedure.

1.1. Nomenclature

M_i^d the ith mesh entity of dimension d, $d = 0$ for a vertex, $d = 1$ for an edge, $d = 2$ for a face and $d = 3$ for a region
G_i^d the ith entity of dimension d in geometry model
$\sqsubset$ classification symbol used to indicate the association of one or more mesh entities with an entity in geometry model [4]
$\{M^d\}$ unordered set of topological mesh entities of dimension d
$\{M_i^d\{M^D\}\}$ the set of mesh entities of order D adjacent to M_i^d
$\mathbf{T}(P)$ mesh metric tensor at point P
$\mathbf{Q}(P)$ transformation representation of $\mathbf{T}(P)$
$h(P, \vec{e})$ desired edge length at point P in the direction of unit vector $\vec{e}$

2. Specification of anisotropic mesh size and shape distribution

An essential requirement for an anisotropic mesh adaptation procedure is a means to define the desired anisotropic element size and shape distribution. The mesh metric field can be used to represent an anisotropic mesh size field that specifies the distribution of element size and stretching (see for example [18,34,36]). This section reviews the definition of the mesh metric field, then discusses the criterion of determining the alignment between a mesh and a mesh metric field that can guide mesh modifications.

2.1. Definition

The anisotropic mesh size at point P is defined as a symmetric positive definite mesh metric tensor $\mathbf{T}(P)$ such that the desired directional mesh edge length distribution at this point follows an ellipsoidal surface (refer to Fig. 1).

Remark 1. Mathematically, the quadratic surface a mesh metric tensor can describe is $\mathbf{X}\mathbf{T}(P)\mathbf{X}^{\mathrm{T}} = 1$, where $\mathbf{X} = [x_1, x_2, x_3]$ is a coordinate vector. The mesh metric tensor must be symmetric positive definite to ensure the geometric representation of the quadratic surface is an ellipsoid [6].

Remark 2. Let $\vec{e}$ denote a unit vector emanating from P in an arbitrary direction and $h(P, \vec{e})$ be the desired edge length along $\vec{e}$ at P, then vector $h(P, \vec{e})\vec{e}$ should point to a point on the ellipsoidal surface and satisfy the quadratic surface. Thus

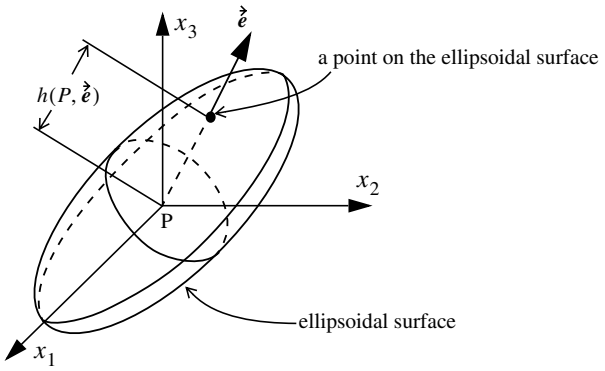

Fig. 1. Definition of mesh metric tensor at point P.

$$h(P,\vec{e})\vec{e}\mathbf{T}(P)h(P,\vec{e})\vec{e}^{\mathrm{T}} = 1 \tag{1}$$

therefore

$$h(P,\vec{e}) = \frac{1}{\sqrt{\vec{e}\mathbf{T}(P)\vec{e}^{\mathrm{T}}}} \tag{2}$$

which is the explicit expression of directional variation of desired mesh edge length at point P.

2.2. Alignment between mesh and mesh metric field

In general, the mesh metric varies with position and a criterion is required to measure the degree of alignment between a mesh entity and the mesh metric field. To develop such a criterion, it is useful to introduce the transformation associated with each mesh metric tensor which maps the mesh entity of interest into a space where the mesh metric tensor becomes identity [34,36,44]. Measuring satisfaction of the mesh entity is done in the transformed space.

2.2.1. Transformation representation of mesh metric tensor

Due to symmetry and positive definiteness, the mesh metric tensor $\mathbf{T}(P)$ can be decomposed as follows:

$$\mathbf{T}(P) = \mathbf{Q}(P)\mathbf{Q}(P)^{\mathrm{T}}, \tag{3}$$

where

$$\mathbf{Q}(P) = \underbrace{\begin{bmatrix}\vec{e}_1\\ \vec{e}_2\\ \vec{e}_3\end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix}\sqrt{\lambda_1} & 0 & 0\\ 0 & \sqrt{\lambda_2} & 0\\ 0 & 0 & \sqrt{\lambda_3}\end{bmatrix}}_{\text{distortion}} \tag{4}$$

with $\vec{e}_1, \vec{e}_2, \vec{e}_3$ being eigenvectors (row vectors) and $\lambda_1, \lambda_2, \lambda_3$ being eigenvalues of $\mathbf{T}(P)$.[1] Note that $\mathbf{Q}(P)$ can be considered as a matrix representing rotation and distortion transformation, and we can define a new coordinate vector after the distortion and rotation as $\mathbf{X}' = \mathbf{X}\mathbf{Q}(P)$. Since

[1] In terms of Eq. (2), eigenvalue λ_i is related to the desired mesh edge length h_i in direction $\vec{e}_i$, namely $h_i = 1/\sqrt{\lambda_i}$.

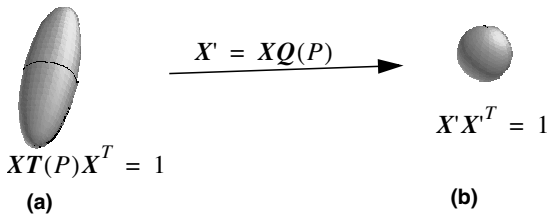

Fig. 2. Geometric illustration of the transformation a mesh metric defines: (a) before transformation; (b) after transformation.

$$\mathbf{X}'\mathbf{X}'^{\mathrm{T}} = \mathbf{X}Q(P)(\mathbf{X}Q(P))^{\mathrm{T}} = \mathbf{X}Q(P)Q(P)^{\mathrm{T}}\mathbf{X}^{\mathrm{T}} = \mathbf{X}\mathbf{T}(P)\mathbf{X}^{\mathrm{T}} = 1,$$

the ellipsoidal surface is transformed into a unit sphere surface by $\mathbf{Q}(P)$, namely, the desired element shape and size is isotropic and normalized under the transformation. Fig. 2 illustrates the geometric significance of this transformation.

2.2.2. Ideal alignment criterion

In terms of the transformation interpretation of the mesh metric, the desired mesh edge length in transformed space is unitary, and any tetrahedron ideally aligned to the mesh metric field will be mapped into a unit regular tetrahedron. Fig. 3 demonstrates this concept. The left figure depicts two desired anisotropic tetrahedra of different size and shape in physical space, while the transformation associated with the mesh metric field is indicated by the two matrices.

Given an arbitrary tetrahedron in physical space and the mesh metric field over the tetrahedron, the degree of satisfaction of the tetrahedron to the mesh metric field can be measured by examining the difference between a unit regular tetrahedron and the mapped tetrahedron in the transformed space. One component of this is done by examining the lengths of the six mesh edges that bound the tetrahedron in the transformed space. The second component is to ensure the resulting tetrahedron is not flat in the transformed space (see Section 2.2.3). A tetrahedron is considered as possibly satisfying the mesh metric field if the six mesh edges are close to one in the transformed space. Note that, it is assumed that a straight/planar tetrahedron in physical space can still be considered straight/planner after the transformation.

Consider edge AB on a local x-axis as depicted in Fig. 4, where ellipses indicate the mesh metric field over AB. In general, the length of AB in the transformed space can be computed by [25,20]

$$L'(AB) = \int_A^B \sqrt{\vec{e}Q(x)\cdot(\vec{e}Q(x))^{\mathrm{T}}}\,\mathrm{d}x. \tag{5}$$

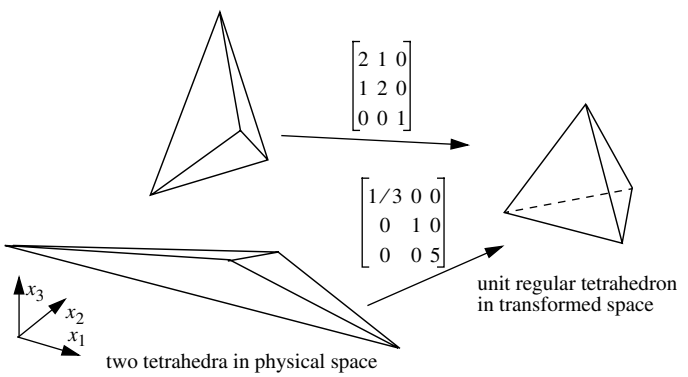

Fig. 3. Desired tetrahedra are mapped into unit regular tetrahedra.

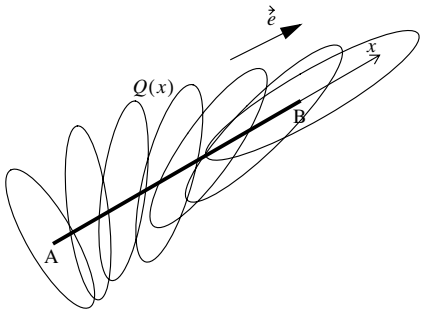

Fig. 4. Illustration of length computation in transformed space.

Remark 3. We view the mesh metric field as a field that can have any appropriate representation. In the examples presented in this paper, we represent it by piecewise liners defined in terms of nodal values. At element level, we simply consider the metric of highest aspect ratio over the element to be efficient and respect anisotropy. Eq. (5) can be simplified into Eq. (6) under the piecewise linear representation [25].

$$L'(AB) = \begin{cases} \dfrac{L}{h(A,\vec{e})} & \text{if } h(A,\vec{e}) = h(B,\vec{e}), \\ \dfrac{L}{h(A,\vec{e}) - h(B,\vec{e})} \ln \dfrac{h(A,\vec{e})}{h(B,\vec{e})} & \text{otherwise} \end{cases} \tag{6}$$

with L being the physical length of AB.

2.2.3. Relaxed alignment criterion for mesh modifications

The lack of any packing proofs for elements of given size and shape ensures that any mesh created will not perfectly match the given mesh metric field. Therefore, the definition of an acceptable representation of the mesh metric field is needed. The one used here is based on:

- Considering a mesh edge to satisfy the mesh metric field if its transformed length falls into an appropriate interval close to one where the interval is defined so that infinite loop of refinement and coarsening is avoided.
- When edges that bound a tetrahedron satisfy the relaxed length criterion, the tetrahedron must be in good quality (not flat) in the transformed space.

The first criterion relaxes edge length requirement from a value into an interval, referred to as $[L_{\text{low}}, L_{\text{up}}]$ hereafter. To prevent the possible oscillation between mesh refinement and mesh coarsening, the selection of L_{low} and L_{up} should satisfy

$$L_{\text{low}} \leqslant 0.5 L_{\text{up}}. \tag{7}$$

Proof. Let AB be an edge with length L_{up} in transformed space and be split into two sub-edges, AC and CB. In terms of the addition property of definite integral,

$$L'(AB) = L'(AC) + L'(CB). \tag{8}$$

To ensure that coarsening will not un-do the splitting, both edge AC and CB must be longer than L_{low} in the transformed space, i.e.

$$L'(AC) \geqslant L_{\text{low}}, \quad L'(CB) \geqslant L_{\text{low}}. \tag{9}$$

Eq. (7) is proved by substituting inequalities in Eq. (9) and $L'(AB) = L_{\text{up}}$ into Eq. (8). □

The selection of the interval is not unique after satisfying Eq. (7). It can be set to many reasonable values, for example, $[\frac{\sqrt{2}}{2}, \sqrt{2}]$, [0.4, 1.0], etc. The procedure presented in this paper does allow the user to adjust one of the parameters, within limits, while the second is calculated to ensure Eq. (7).

The second element level criterion is required since, in three dimensions, sliver (see Fig. 5 for an example) may exist even if the edge length criterion are met. To minimize computation, the cubic of mean ratio shape measure [27] in transformed space (see Eq. (10)) is used to identify slivers.

$$\eta' = 15552 \frac{V'^2}{\left(\sum_{i=1}^{6} l_i'^2\right)^3} \tag{10}$$

with V' and $l_i'^2$ being the volume and edge length square in transformed space that can be computed as follows:

$$V' = \frac{1}{6}\sqrt{\lambda_1\lambda_2\lambda_3}(\vec{v}_1 \times \vec{v}_2) \cdot \vec{v}_3, \tag{11}$$

$$l_i'^2 = \sum_{j=1}^{3} \sqrt{\lambda_j}(\vec{v}_i \cdot \vec{e}_j), \tag{12}$$

where $\vec{v}_i$ ($i = 1, \ldots, 6$) are the vectors associated with the edges of a tetrahedron, $\vec{e}_i$ and λ_i are eigenvectors and eigenvalues of the highest aspect ratio metric defined over the tetrahedron. It is worth noting that the total cost of computing η' is 15 additions, 28 multiplications, 3 subtractions and 1 division if we represent metrics by their eigenvectors and square root eigenvalues, and the computing of $l_i'^2$ can be further saved since the control of element size is performed via edge length computing.

η' has been normalized to interval [0, 1] with 0 for flat tetrahedron and 1 for regular tetrahedron in transformed space. In Section 3.3, we consider a tetrahedron as sliver that must be corrected if its η' is less than a given threshold.

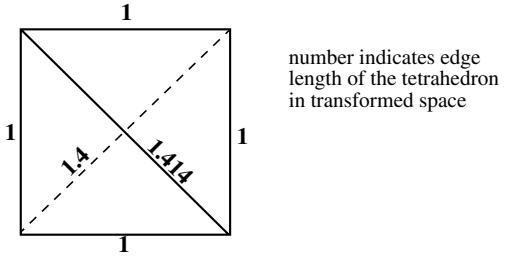

Fig. 5. Example of sliver tetrahedron in transformed space.

3. Mesh adaptation procedure

Given any initial mesh, the geometry domain the mesh is classified onto and a mesh metric field, the goal of the mesh adaptation procedure is to effectively modify the mesh using mesh modifications to satisfy the mesh metric field while maintaining appropriate approximation to the curved geometry domain. By performing all geometric computations that control the application of mesh modifications in the transformed space, the procedure generates an anisotropic mesh when the mesh metric field is anisotropic.

3.1. Overall procedure

Fig. 6 shows the pseudo-code of the overall mesh adaptation procedure. It consists of two stages: coarsening/shape-correction (lines 1–5) and iterative refinement (lines 7–16).

The first stage applies local mesh modification operations to eliminate as many short edges in the initial mesh as possible. Here, a mesh edge is referred to as "short" if its length in the transformed space is smaller than L_{low}. It differs from the coarsening inside refinement iterations (see line 13 of Fig. 6) in two aspects: allowing element shape drop and no control on edge length increase. It is important to note that the number of applicable coarsening operations can be depended on the allowed element shape drop. In the extreme, no coarsening operation can be applied if we want to coarsen a mesh consisting of equilateral triangles but do not allow any shape decrease. Hence, element shape decrease must be allowed to coarsen meshes in good quality. Similar is the control of mesh edge length. Another consideration of the coarsening stage is to filter out the refinement patterns in which the triangle or tetrahedron to be refined has short mesh edge(s). For example, the middle refinement pattern in Fig. 8 will be filtered out if M_c^1 is a short edge.

The shape-dropped elements as well as those poorly-shaped in the initial mesh are handled by the element shape correction process that follows. Details of this process are described in Section 3.3. It is important to eliminate poorly-shaped tetrahedra before refinements to avoid the possible duplication of slivers during refinement. Fig. 7 depicts a simple 3D example to demonstrate the undesired sliver duplication that would occur during refinement if slivers were not eliminated. Fig. 7(a) shows a sliver tetrahedron (vertex M_0^0 is close to the center of its opposite face) and indicates the length of its edges in the transformed space. Fig. 7(b) shows the tetrahedra created by subdividing the tetrahedron with the three long edges split. It can be seen that four undesired sliver tetrahedra are created (one associated with each vertex of the tetrahedron).

The second iterative refinement stage is an integration of four components: refinement, projecting new boundary vertices, coarsening short edges and correcting slivers. The goal is to reduce the maximal mesh

```
1   determine all mesh edges shorter than L_low in transformed space
2   eliminate these edges using the coarsening algorithm
3
4   determine all sliver tetrahedra in transformed space
5   eliminate these slivers using the shape correction algorithm
6
7   compute L'_max, the maximal mesh edge length in transformed space
8   while L'_max > L_up
9     compute L'_ref = max(αL'_max, L_up), where α is a given constant
10    mark all edges longer than L'_ref as refinement edges
11    split all marked edges and their adjacent entities through subdivisions
12    project all newly created boundary vertices onto their boundaries
13    process all newly created short edges using the coarsening algorithm
14    process sliver tetrahedra projecting creates using the shape correction algorithm
15    update L'_max
16  end while
```

Fig. 6. Overall mesh adaptation procedure.

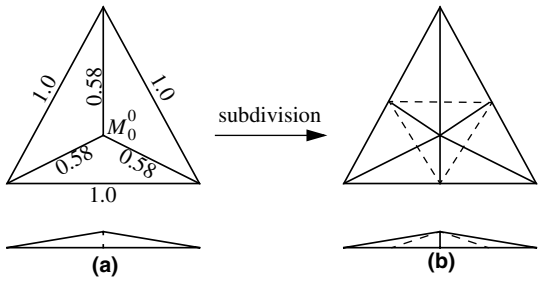

Fig. 7. Illustration to possible sliver duplication in subdivision.

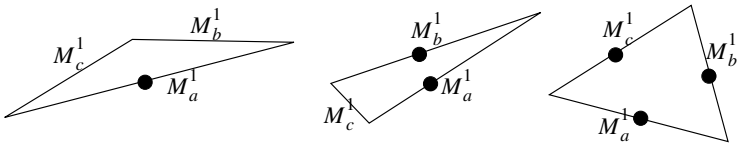

Fig. 8. Desired refinement patterns to refine a triangle.

edge length in transformed space, denoted as $L'_{\max}$, until it is less than L_{up}. In particular, the length reduction in transformed space is performed through multiple iterations. In each iteration, the procedure refines a set of longest mesh edges in transformed space using the full set of edge-based subdivision templates [15,25], projects all new vertices on curved boundaries onto their classified boundaries [26], coarsens short mesh edges subdivision may create, and eliminates sliver tetrahedra projecting new vertices may create.

To achieve intelligent refinement patterns, a refinement criterion, refining a set of longest mesh edges in the transformed space in each iteration, is adopted and $L'_{\max}$ is reduced in an incremental manner. The set of longest mesh edges are efficiently determined by iteratively marking edges longer than

$$\max(\alpha L'_{\max}, L_{\text{up}}) \tag{13}$$

as refinement edges, where α is a given constant that controls the length reduction rate. α should be a value at interval $[\sqrt{2}/2, 1]$ so that refinement edges can be selected in a way that the type of sliver tetrahedron depicted in Fig. 5 can be eliminated during subdivision and coarsening iterations. Fig. 8 illustrates that the "a set of longest edges criterion" can always select the desired refinement edges to split a triangle. In this example, three typical triangles are depicted: a triangle with an angle close 180 degrees but without short edge (left), a triangle with a relatively shorter edge (middle) and a triangle close to equilateral triangle (right). Solid black bullets indicate refinement edges when the criterion of Eq. (13) is applied. It can be seen that, in the case of left triangle, only edge M_a^1 can be the refinement edge; While, for the middle triangle, edge M_c^1 cannot be the refinement edge[2]; And all edges are refinement edges for the right triangle. Similar are the patterns of tetrahedral subdivisions.

Since the edge-based refinement templates may create short mesh edges when subdividing poorly shaped faces or regions, a coarsening process that does not allow element shape dropping is followed to achieve a refined mesh in which newly created vertices are not connected to any short mesh edge. Fig. 9 gives three typical situations where short mesh edges (indicated by solid black bullets) may be created in subdividing

[2] Note that the middle pattern is also desired since: (i) $L'(M_c^1) > L_{\text{low}}$, otherwise it should be filtered out by previous coarsening; (ii) if the two new vertices are close, one of them will be collapsed out, which is equivalent to a split-swap operation.

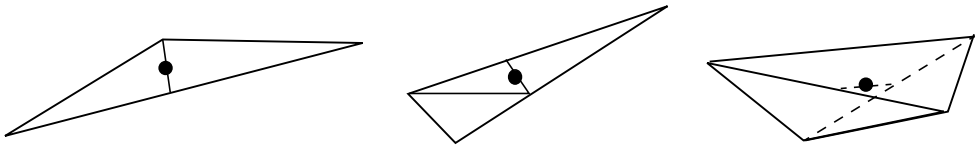

Fig. 9. Possible short edges to be created in refinement templates.

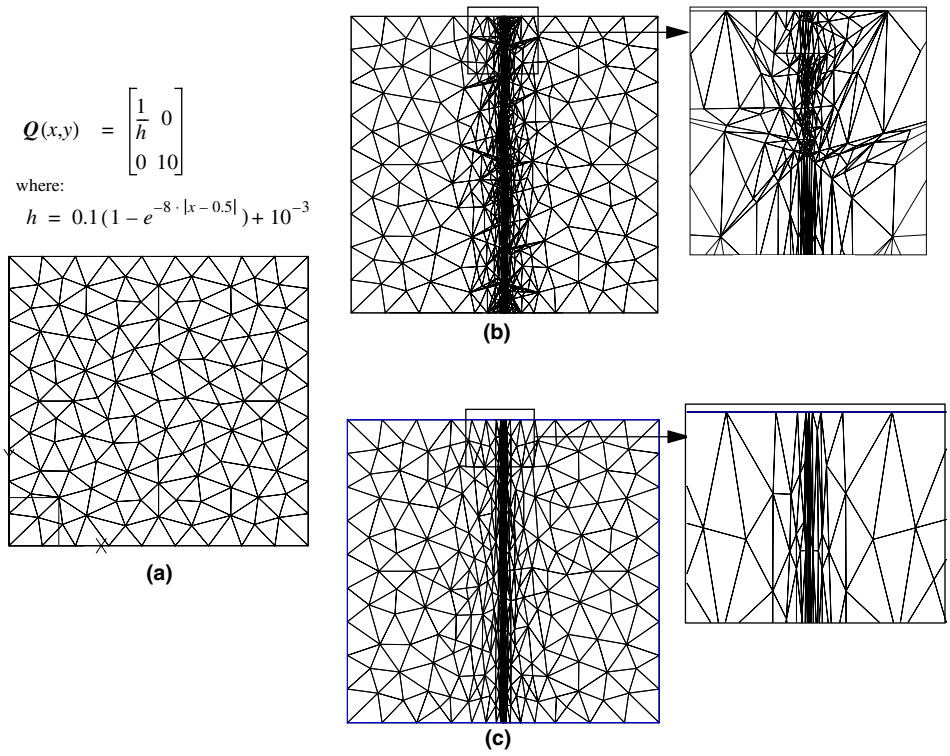

Fig. 10. 2D example to show the importance of eliminating short edges: (a) initial mesh; (b) result mesh without coarsening; (c) result mesh with coarsening.

the two triangles or the tetrahedron. Fig. 10 shows the importance of eliminating these short edges after the element subdivision process. Fig. 10(a) gives the initial mesh covering two-dimensional domain $[0, 1] \times [0, 1]$ with origin at its left-down corner. The analytical expression specifies the target anisotropic mesh size (transformation matrix) field to catch a discontinuity at $x = 0.5$. Fig. 10(b) gives the mesh generated using the template refinement process without mesh coarsening. It can be seen that the refined mesh has many short edges and is not close to optimal. Similar refined meshes have been reported by Lo for tetrahedral meshes [31]. Lo also indicated that the poorly refined mesh cannot be improved using edge/face swap and vertex relocation procedures. Fig. 10(c) gives the adapted mesh using the algorithm in which subdivision process collects short mesh edges it creates, and the coarsening process that follows eliminates them. It can be seen that the coarsening algorithm is effective in solving the problem.

It is important to remark that, by optimal selection of subdivision patterns and diagonals (see Section 3.4.2), as well as denying any coarsening operation that drops element shape, the splits-collapses procedure

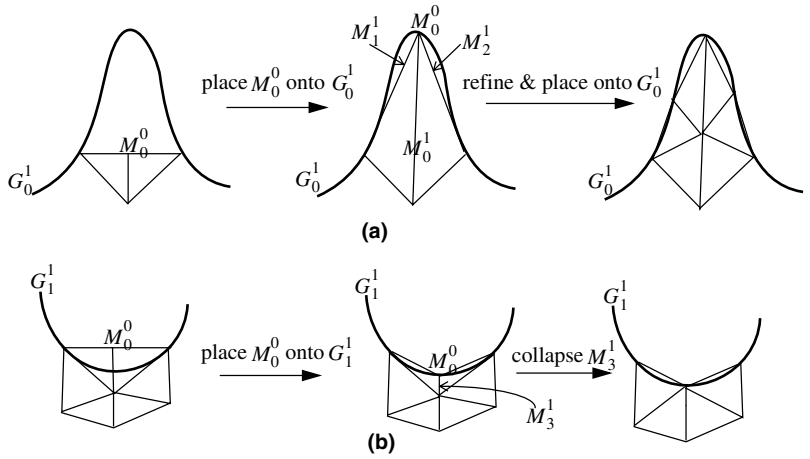

Fig. 11. Short/long edges projecting vertices generates are addressed in refinement iterations.

avoids the creation of slivers, the critical issue that must be explicitly accounted for in order to apply three-dimensional local Delaunay reconnection algorithm [18,33].

To account for curved geometry domains, the edge-based template refinement algorithm maintains a list of new vertices to be placed onto curved boundaries. Boundary vertices are identified in terms of the classification information between the mesh and the geometry model [4]. When splitting a mesh edge classified onto a model face or a model edge, the new vertex inherits the classification of the parent mesh edge, but is temporally placed at a point on the parent mesh edge and inserted into the vertex list. After finishing the current iteration of refinement, the vertex list is processed using the algorithm given in Refs. [25,26], which places those vertices onto their boundaries.

The short/long mesh edges or sliver tetrahedra possibly created in projecting boundary vertices are addressed by the coarsening, shape correction or refinement process that follows. Fig. 11 depicts two 2D examples to demonstrate this. In Fig. 11(a), placing vertex M_0^0 onto curved boundary G_0^1 generates three long edges (M_0^1, M_1^1 and M_2^1). These three edges are refined with new vertices placed onto G_0^1 in the next iteration. While in Fig. 11(b), projecting vertex M_0^0 onto G_1^1 generates a short edge M_3^1, which is collapsed in coarsening process.

The iterative second stage terminates when $L'_{\max}$ is less than L_{up}. This termination condition is ensured since: (i) splitting is always performable, and (ii) any collapse operation inside the iterative loop is denied if it creates a mesh edge longer than the current $L'_{\max}$ value.

The mesh modification operations described in this paper do not include vertex repositioning. Vertex repositioning is a useful operator that is most effective in local improvements when the local mesh topology is already satisfactory. Just like any of the other individual operators, it is not sufficient to support all the needs of anisotropic mesh adaptation. However, its inclusion would provide specific benefits. How to most effectively include vertex repositioning in the overall procedure will be investigated in the future.

3.2. Coarsening

The mesh coarsening algorithm accepts a list of short mesh edges and a value of maximal mesh edge length allowed in transformed space, $L'_{\max}$, as inputs. The goal is to eliminate these short mesh edges if possible while not creating any mesh edge longer than $L'_{\max}$.

Coarsening can be performed by three local mesh modifications:

- edge collapse;
- swap(s) collapse compound operations [25];
- vertex relocation.

Fig. 12 illustrates the edge collapse operation, which is a simple transform that collapses edge M_i^1 by "pulling vertex M_d^0 and merging it with vertex M_r^0". The collapse can also be executed as a compound operation that chains one or several swap operations with an edge collapse operation. Fig. 13 depicts an example to show the need for this compound operation, where collapsing vertex M_i^0 inside the prism is not possible however, the compound operation, swapping edge M_l^1 into M_m^1 followed by collapsing vertex M_i^0 to M_j^0, can yield the desired result. The pre-swap operation can be effectively determined by investigating M_k^3, the tetrahedron to be flat, using the flat element analysis technique described on pages 257–258 of Ref. [26]. The vertex relocation is a useful operation if edge collapse operation would create edge(s) longer than $L'_{\max}$ in transformed space.

Note that these coarsening local mesh modifications are not always performable since they may produce flat/inverted elements or violate the topological association between mesh and its model domain. However, the validity of their executions, the interested quantity of the result mesh configuration (for example the maximal edge length in transformed space to be produced) can be predicted without the actual execution.

Fig. 15 describes the pseudo-code of the mesh coarsening algorithm. The first nine lines of the algorithm simply initialize a dynamic vertex list with the end vertices of the given short mesh edge list. To ensure uniqueness and be efficient, a tag process is used to determine (in the complexity of O(1)) if a vertex is already in the dynamic list.

The coarsening process, starting at line 11 in Fig. 15, repeatedly processes vertices in the dynamic list until the list is empty or all remaining vertices in the list have been tagged as processed. The relative order of vertices processed is important to maintain a good vertex distribution. Section 3.2.2 discusses this issue in detail. Since vertices in the dynamic list may not bound a short edge any more due to the mesh modifications carried out in previous steps, line 15 is needed to remove such vertices from the dynamic list.

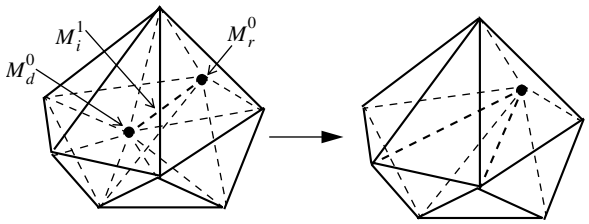

Fig. 12. Edge collapse in three dimensions.

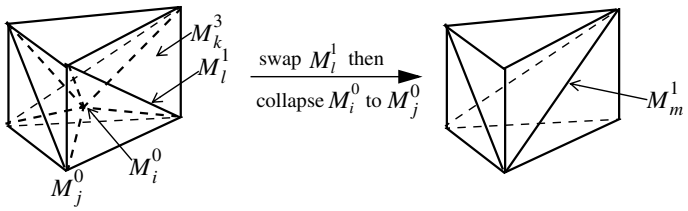

Fig. 13. 3D example of swap collapse compound operation.

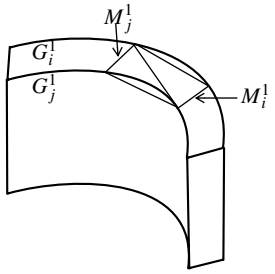

Fig. 14. Collapsing edge M_i^1 or M_j^1 is topologically invalid since crossing model edge G_i^1 and G_j^1.

```
1   initialize a dynamic vertex list
2   loop over the given short edge list
3     for each vertex that bounds the current edge
4       if the vertex is not yet tagged to be in the dynamic list
5         append the vertex into the dynamic list
6         tag the vertex to be in the dynamic list
7       end if
8     end for
9   end loop
10
11  while there are vertices not tagged processed in the dynamic vertex list
12    get an unprocessed vertex M_i^0 from the list in a controlled order (see section 3.2.2)
13    get M_j^1, the shortest mesh edge in transformed space connected to M_i^0
14    if the transformed length of M_j^1 is greater than L_low
15      remove M_i^0 from the dynamic list
16    else
17      evaluate edge collapse operation of collapsing M_j^1 with M_i^0 removed
18      if the edge collapse would create an edge longer than L'_max in transformed space
19        evaluate relocating vertex M_i^0
20      else if the edge collapse would lead to flat/inverted elements
21        evaluate the swap(s)/collapse compound operation
22      end if
23      if any local mesh modification is determined
24        tag neighboring vertices of M_i^0 in the dynamic list as unprocessed
25        apply the local mesh modification
26        remove M_i^0 from the dynamic list if it is collapsed
27      else
28        tag M_i^0 as processed
29      end if
30    end if
31  end while
```

Fig. 15. Pseudo-code of mesh coarsening algorithm.

Line 17–29 process a specific vertex, denoted as M_i^0, that bounds a short edge. The algorithm first evaluates an edge collapse operation, collapsing the shortest mesh edge in transformed space M_i^0 bounds. Other operations are selectively evaluated in terms of the results of the edge collapse evaluation. In case the

predicted maximal mesh edge length in transformed space after the edge collapse is longer than $L'_{\max}$, relocating M_i^0 is further evaluated. The compound operation is further evaluated if edge collapse would produce flat/inverted elements. No further action is necessary if the edge collapse is topologically invalid (see Fig. 14 for an example). Once a local mesh modification is determined, execute it, update the dynamic list as well as tag all vertices connected to M_i^0 through a mesh edge as unprocessed. If no local mesh modification can be applied (this can be the best choice in some cases), then simply proceed to the next.

Since the success of eliminating a short mesh edge cannot be ensured, a process that tags vertices in the dynamic list as processed has been incorporated to guarantee the termination of the algorithm.

3.2.1. Collapsing the shortest edge

The edge collapse operation has been used as the major local mesh modification in the coarsening process. To be efficient, it is critical to realize that, given a mesh vertex connected to short edges, only one edge collapse operation needs to be checked, namely, collapsing the shortest edge connected to the vertex. Here, the shortest edge of a vertex is referred to as the edge of smallest length in transformed space among all edges connected to the vertex.

The reason for applying this "collapsing shortest edge" criterion can be explained from two aspects. First, since collapsing a shorter edge causes less changes to the local mesh, collapsing the shortest edge has a higher success rate than collapsing a longer one. The second justification can be illustrated by the simple 2D example depicted in Fig. 16. In this example, collapsing the shortest edge M_0^1 with vertex M_0^0 removed is not possible while collapsing a longer edge (for example M_2^1) can succeed. However, it can be seen that the compound operation, swapping M_1^1 then collapsing M_0^1, succeeds and yields the same local mesh configuration as directly collapsing vertex M_0^0 to M_2^0.

3.2.2. Processing vertices every other one

By processing vertices in the dynamic list in an order that is topologically every other one, a good vertex distribution during mesh coarsening can be maintained. Consider Fig. 17 for an example to demonstrate this idea. Fig. 17(a) shows a two-dimensional mesh to be coarsened, where M_0^0 represents the current vertex being processing and black bullets indicate its topologically neighboring (edge connected) vertices. Fig. 17(b) shows the locally coarsened mesh with M_0^0 collapsed. In terms of the "every other vertex" criterion, the next vertex to be processed should be a vertex indicated by a circle in Fig. 17(b). Fig. 17(c) shows such a result mesh (vertex M_2^0 is collapsed). It can be seen that both vertex distribution and local mesh quality are maintained. Fig. 17(d) shows the result mesh where vertex M_1^0 were allowed to be collapsed next. In this case, coarsening would be too concentrated on the left-bottom corner, while other parts of the mesh are untouched. Furthermore, even additional swap operation(s) is applied to regain a good local mesh connectivity, vertices are not in good distribution.

De Cougny has pointed out the needs of maintaining vertex distribution control during mesh coarsening and proposed a method in terms of an independent set paradigm [15]. Two deficiencies exist in his method.

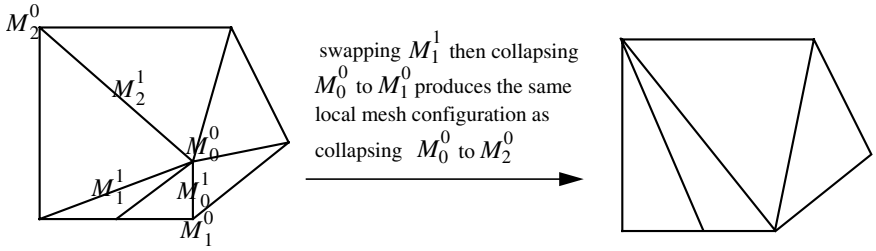

Fig. 16. 2D example to justify "collapsing shortest edge" criterion.

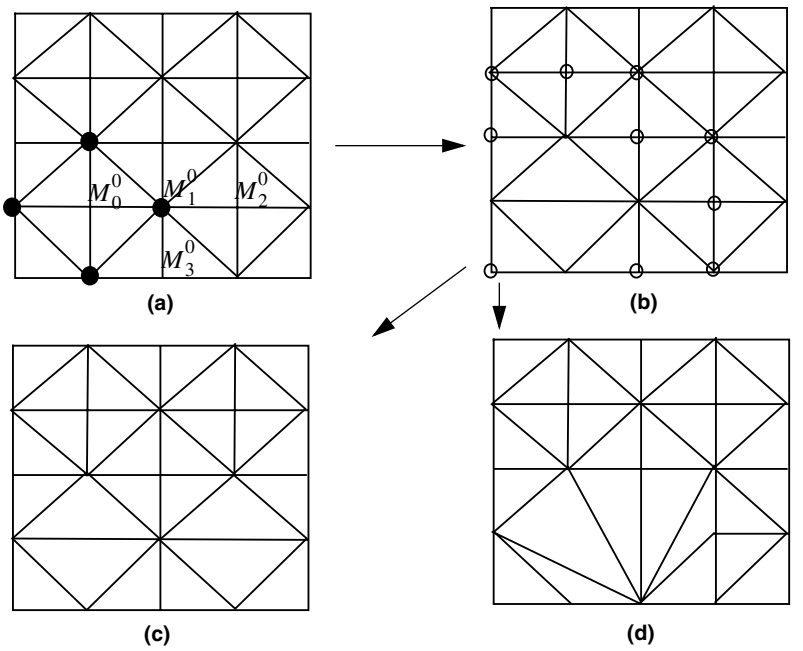

Fig. 17. 2D example to show the need of processing vertices "every other one": (a) start mesh; (b) after collapsing M_0^0; (c) after collapsing M_2^0 to M_1^0; (d) if M_1^0 were collapsed to M_3^0.

First, although coarsening is a local activity in adaptive analysis, the independent set of vertices is determined in a global sense, and it has to be determined more than once in cases where more than one level of coarsening is required. Second, constraining vertices in the independent set as not collapsible is unnecessary and may preclude desired edge collapse operations. An algorithm that overcomes these two deficiencies focuses on changing the order of processing vertices to follow "topologically every other vertex" rule.

Figs. 18 and 19 gives the pseudo-code that changes the order of processing vertices in the dynamic list by properly tagging them as "not collapsible". Fig. 20 depicts a part of a 2D mesh to be coarsened to demonstrate this algorithm, where integers (1, 2, ..., 13) indicate mesh vertices in the dynamic list and the bottom line shows the order of these vertices. The first vertex to be processed is M_1^0 since it is the first vertex in

```
1   while there are vertices not tagged "processed" in the dynamic vertex list
3     loop over vertices not tagged "processed" in the dynamic list
4       let M_i^0 be the current vertex not tagged "processed"
5       if no vertex in dynamic list is tagged as "not collapsible" yet
6         process M_i^0 and update tags (see figure 19)
7       else if M_i^0 is not tagged but connected to a vertex tagged as "not collapsible"
8         process M_i^0 and update tags (see figure 19)
9       end if
10    end loop
11    clear "not collapsible" tags on vertices in the dynamic list
12  end while
```

Fig. 18. Pseudo-code to enfore vertices processed "topologically every other one".

```
1  if M_i^0 can be removed by edge collapsing or compound operation
2    tag neighboring vertices of M_i^0 as "unprocessed" and "not collapsible"
3    apply the edge collapsing or compound operation
4    remove M_i^0 from the dynamic list
5  else
6    tag M_i^0 as "processed"
7  end if
```

Fig. 19. Pseudo-code to process vertex M_i^0 and update tages.

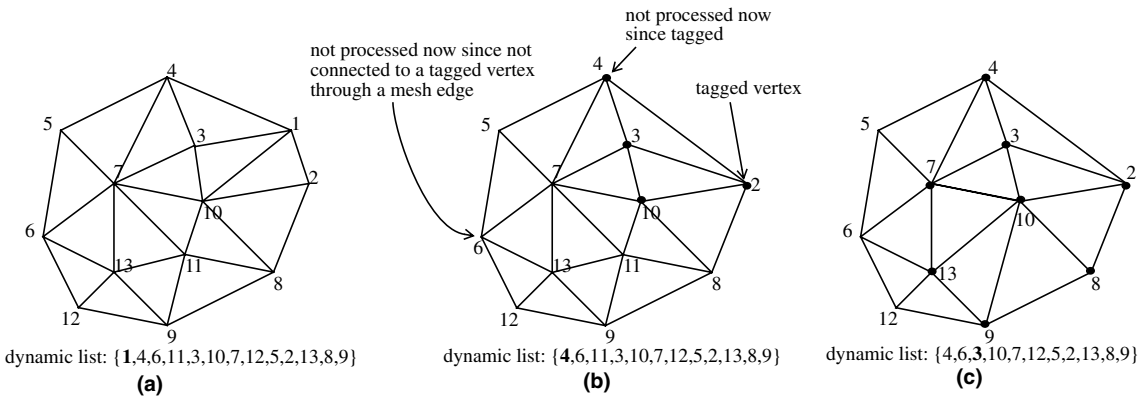

Fig. 20. 2D example for the algorithm to ensure processing vertex "every other one": (a) a portion of a 2D mesh to be coarsened; (b) after vertex 1 collapsed; (c) after vertex 11 collapsed.

the dynamic list and no vertex has been tagged as "not collapsible" yet. Fig. 20(b) shows the resulting mesh and the updated dynamic list after M_1^0 is collapsed, where black bullets indicate the vertices tagged as "not collapsible" (connected to M_1^0 by mesh edges). The next two vertices, M_4^0 and M_6^0, in the dynamic list cannot be processed since M_4^0 has been tagged as "not collapsible" and M_6^0 is not connected to any tagged vertex by a mesh edge. M_{11}^0 is processed next. Fig. 20(c) shows the resulting mesh after M_{11}^0 is collapsed and vertex $M_7^0, M_8^0, M_9^0, M_{13}^0$ are further tagged. Following the same logic, in the first traversing over the dynamic list, the next two vertices that can be processed in turn is M_{12}^0 and M_5^0. Then, the "not collapsible" tags on remaining vertices of the dynamic list are cleared and the list is traversed from the start again until all vertices in the dynamic list are processed.

3.3. Shape correction

Because the initial mesh may have sliver tetrahedra and element quality may degrade during coarsening or projecting mesh vertices to curved boundaries, local operations are included to allow the mesh to be modified to better match the desired mesh metric field.

Local mesh modifications, particularly edge and face swap operators, can be used to improve the mesh quality by replacing poor elements with better ones. In case swap operations cannot be applied, additional local mesh modifications, e.g. split, relocation and collapse operations have shown to be useful [15,10,35]. The key issue in applying these local mesh modifications lies in the effective determination of the most appropriate one without the cost of examining a large number of possible operations.

```
1   initialize a dynamic list
2   insert all elements with quality below a given threshold into the dynamic list
3   while the dynamic list is not empty
4     pop an element from the list
5     analyze the popped element (see section 3.3.1)
6     determine the mesh modification improving element quality most (section 3.3.2)
7     if such a mesh modification exists
8       apply the mesh modification
9       update the dynamic list
10    end if
11  end while
```

Fig. 21. Pseudo-code of the shape correction algorithm.

Fig. 21 outlines the shape correction algorithm. Given a quality threshold, the algorithm first collects all elements with quality below the threshold into a dynamic list. Then process each element in this dynamic list using the sliver tetrahedron analysis technique described in next two subsections until the list is empty. Whenever a tetrahedron in this list is processed, the dynamic list is updated to remove the tetrahedron from the list. If a local mesh modification is executed, the dynamic list is updated to remove all deleted elements and insert any newly created elements with quality below the threshold.

3.3.1. Analysis of sliver tetrahedra

Sliver tetrahedra are characterized by small volume while not being bounded by any relatively short edge both measured in the transformed space. The possible types of sliver tetrahedra are limited. To support the intelligent selection of local mesh modifications, it is useful to classify tetrahedra into two types [10,28] (see Fig. 22). A tetrahedron is classified as type I sliver if two opposite edges of the tetrahedron almost intersect. A tetrahedron is classified as type II if one vertex of the tetrahedron is close to the centroid of its opposite face.

Such a classification is useful to the determination of local mesh modification since it indicates the reason why the tetrahedron is of poor quality, therefore, points to the possible operations to eliminate the sliver tetrahedron. In particular, for type I, the reason is the small distance between a pair of opposite edges of the

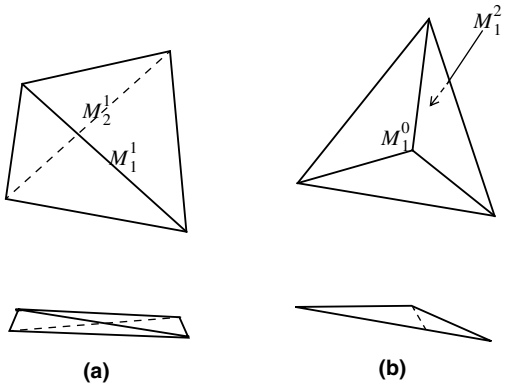

Fig. 22. Classification of sliver tetrahedron: (a) type I, (b) type II.

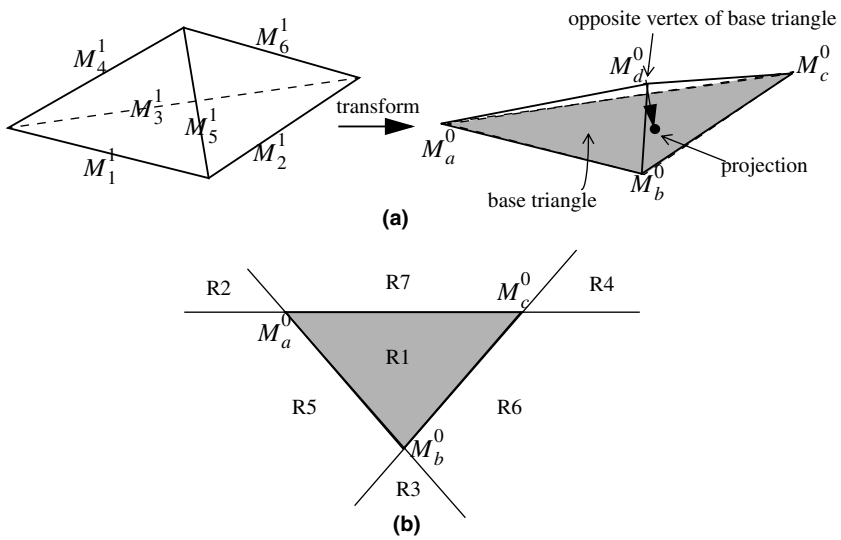

Fig. 23. Determination of sliver type in terms of projection: (a) a sliver tetrahedron in physical (left) and transformed (right) space; (b) top view of the base triangle.

Table 1
Type and key mesh entities in terms of projection location

Projection location	Type	Key mesh entities
R1	I	M_d^0 and its opposite face
R2		M_a^0 and its opposite face
R3		M_b^0 and its opposite face
R4		M_c^0 and its opposite face
R5	II	M_1^1 and M_6^1
R6		M_2^1 and M_4^1
R7		M_3^1 and M_5^1

tetrahedron, so one (or both) of the edges must be eliminated (the key entities are the two edges). For type II, the reason is that a vertex is too close to its opposite face, so either the vertex or the face must be eliminated (key entities are the vertex and its opposite face).

The qualification of the two sliver types is determined based on projection in transformed space. Fig. 23(a) shows a simple construct to explain the projection based method in which the first step is the selection of a base triangle (shaded), which can be any mesh face of the tetrahedron. The vertex of the tetrahedron opposite to the base face is projected onto the base face plane in the transformed space. The type and key mesh entities of the tetrahedron are then determined in terms of the relative location of this projected point. Fig. 23(b) shows the top view of the base triangle plane, which is subdivided into seven subareas (R1 to R7) defined by extending the mesh edges of the base triangle. There is one subarea associated with each edge, one associated with each vertex, and one with the base triangle itself. Table 1 indicates the tetrahedron type and key mesh entities in terms of which the projection is in. The projection will not be close to any of the mesh vertices of the base triangle since this would correspond to the existence of a short edge.

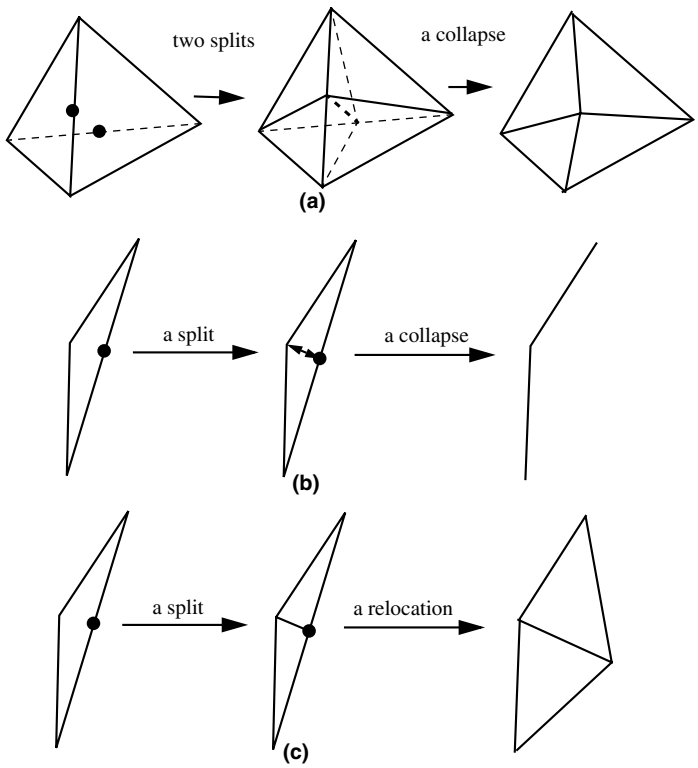

Fig. 24. Illustration of compound operations: (a) double split collapse operation; (b) split collapse operation; (c) split relocation operation (black bullets indicate edges to be split and all figures only show the sliver element with adjacent tetrahedra not depicted).

Note that, as long as the volume of the tetrahedron is almost zero, the selection of base triangle has no effect on the analysis results given in Table 1. For instance in the example depicted in Fig. 23, if the face bounded by vertex M_a^0, M_b^0 and M_d^0 were selected as the base triangle and vertex M_c^0 were projected onto it, the projection would be in a subarea associated with vertex M_d^0, and the key entities would still be M_d^0 and its opposite face.

It should be pointed out that this projection-based method is efficient. Given the three vectors that define a tetrahedron, identifying the projected subarea or the closure can be implemented with 10 addition, 24 subtraction, 42 multiplication, 1 divisions and 2–6 comparison operations.

Table 2
Determination of local mesh modification in terms of sliver tetrahedron analysis

Sliver type	Priority	Possible local mesh modifications
I	1	Swap either key edges
	2	Split either key edges, then relocate or collapse the new vertex
		Split both key edges, then collapse the new interior edge
	3	Relocate a vertex of the tetrahedron
II	1	Swap the key face, or any edge that bounds the key face
	2	Split the key face, then collapse or relocate the new vertex
	3	Relocate a vertex of the tetrahedron

3.3.2. Intelligent selection of local mesh modifications

The used local mesh modification operators include: edge swap, face swap, vertex relocation and compound operators. The target location of relocation is computed using de Cougny's methodology to explicitly improve the worst shape of connected elements [16]. Fig. 24 illustrates the used compound operators. The double split collapse operation reduces a sliver tetrahedron into four triangles by splitting a pair of opposite edges and collapsing the new interior edge. The split collapse operation first splits an edge or a face, then collapses the new interior edge. The split relocation operation first splits an edge or a face, then relocates the new vertex.

Given a sliver tetrahedron, Table 2 indicates a possible effective determination of the appropriate local mesh modification based on the results of sliver tetrahedron analysis. To avoid extensive evaluations, these mesh modification operations are evaluated in three priority levels with the most effective ones evaluated first and the operations in the next level considered only if operations in previous level(s) could not be performed.

Since the swap operation is the most effective local mesh modification to eliminate slivers (see results of the first examples in Section 4), and the possible swap operations are limited (only two possibilities for a type I sliver and four possibilities for type II), the swap operation is set as the first to be evaluated. The second level evaluates specific sets of compound mesh modifications to increase the chance of success. It involves evaluating two split collapse operations, two split relocation operations and one double split collapse operation for a type I sliver tetrahedron, or one split collapse operation and one split relocation operation for a type II sliver tetrahedron. The third level evaluates relocating a vertex of the sliver tetrahedron. Although in practice these three levels of local mesh modification usually are enough to improve mesh quality to a reasonable level, the elimination of the sliver cannot be ensured theoretically due to the heuristic nature of local mesh modifications.

The determined local mesh modification is applied if it improves local mesh quality and does not violate the length criterion. Also, after the application of swap or compound operations, the length of new mesh edges is examined in transformed space. Attempts are made to collapse them in case of new short mesh edges.

3.4. Refinement

Given a mesh size field and refinement threshold $L'_{\max}$, the refinement algorithm marks all mesh edges longer than $L'_{\max}$ in the transformed space, then applies edge-based subdivision templates to split all marked edges and their adjacent elements. Although a wide variety of algorithms can be used for refinement, for example insertion using Delaunay criterion [9,18,33], edge splitting [10,31,35], etc., the method using templates is favored since it is relatively efficient (linear complexity), can minimize over-refinement and does not create slivers.

Fig. 25 presents the refinement algorithm in pseudo-code form. It performs *create* and *delete* operations to modify the mesh. The *create* operations are applied from bottom-up, i.e., first create mesh entities to replace refinement edges, then create mesh entities to replace faces refinement edges bound, finally create mesh entities to replace regions refinement edges bound. The reason for using such a bottom-up strategy is to make this algorithm applicable to non-manifold geometry models consisting of general combinations of solids, surface and wires. In the process of creation, the algorithm creates child mesh entities for each parent mesh entity (the mesh entity being split) one by one and does not immediately delete the parent mesh entity but properly attaches creation information onto it if it still bounds any higher dimension mesh entities. The attached information is retrieved and used when the higher dimension mesh entities are processed. For example, in case of subdividing face M_j^2 that bounds two mesh regions, the algorithm first retrieves all creation information attached to edges in $\{M_j^2\{M^1\}\}$, then creates a triangulation of M_j^2 using retrieved information, finally attaches the information of the triangulation onto M_j^2. When any mesh region M_j^2 bounds is processed, the attached information on M_j^2 will be retrieved and used. After the creation process, a deletion

```
initialize a list of faces to be split
loop over mesh edges in the mesh
  let M_i^1 be the current edge and L' be its length in transformed space
  if L' > L'_max, then
    create new mesh entities to replace M_i^1 (M_i^1 is not deleted)
    if M_i^1 bounds any mesh faces, then
      attach a pointer to all new mesh entities onto M_i^1
      for each mesh face M_i^1 bounds
        insert it into the face list and tag it as inserted if not tagged as inserted
      end for
    else
      delete M_i^1
    end if
  end if
end loop

initialize a list of regions to be split
loop over the mesh face list to be split
  let M_j^2 be the current mesh face to be split
  retrieve all new mesh entities attached to edges in {M_j^2{M^1}}
  create a new triangulation to replace M_j^2 with retrieved information used
  if M_j^2 bounds any mesh region
    attach a pointer to the new triangulation onto M_j^2
    for each region M_j^2 bounds
      insert it into the region list and tag it as inserted if not tagged as inserted
    end for
  else
    delete M_j^2 and edges in {M_j^2{M^1}} that do not bound any face
  end if
end loop

loop over the mesh region list to be split
  let M_k^3 be the current mesh region to be split
  retrieve all new triangulations attached to the faces in {M_k^3{M^2}}
  create a new tetrahedralization to replace M_k^3 using retrieved information
  delete M_k^3 and its downward entities that do not bound any higher dimension entity
end loop
```

Fig. 25. Pseudo-code of the refinement algorithm.

process deletes any subdivided mesh entity and its downward mesh entities that do not bound any higher dimension mesh entities. The final mesh is always conforming after all subdivided mesh entities are deleted.

Fig. 26 illustrates the refinement algorithm through a simple two-dimensional example. In this example, the initial mesh consists of four triangles as depicted in Fig. 26(a), where refinement edges are indicated by four black bullets. Fig. 26(b) shows the non-conforming mesh after processing refinement edges, where eight new mesh edges (indicated by the eight segments) are created and attached onto the four refinement edges, and the circle indicates the collected mesh faces to be processed later. Fig. 26(c) gives the non-conforming mesh after the shaded face is replaced by three child faces (the three shrunk triangles) and deleted.

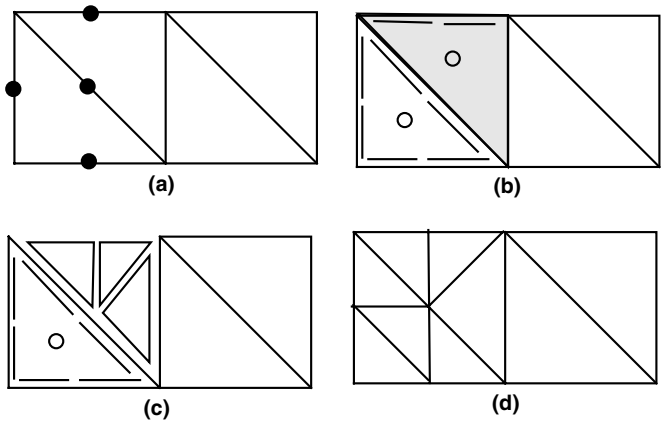

Fig. 26. Illustration of the refinement algorithm.

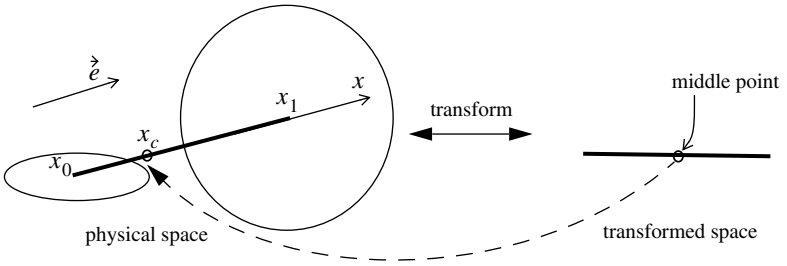

Fig. 27. Middle point in transformed space.

The top edge that bounds the shaded face is also deleted since it does not bound any mesh face any more. Fig. 26(d) gives the refined conforming mesh after processing the remaining face indicated by a circle, i.e., replacing it with four child triangles and deleting it and its downward mesh entities.

3.4.1. Selection of vertex insert point

We refine mesh edges at the middle point in transformed space.

Fig. 27 illustrates the middle point in transformed space, where an edge is depicted in both physical and transformed space with ellipses indicating the desired directional mesh size distribution at the two ends of the edge. In terms of Eq. (14), generally, the physical position of middle point in transformed space can be computed by solving the following equation:

$$\int_{x_0}^{x_c} \sqrt{\vec{e}Q(x)Q(x)^{\mathrm{T}}\vec{e}^{\mathrm{T}}}\,\mathrm{d}x = 0.5\int_{x_0}^{x_1} \sqrt{\vec{e}Q(x)Q(x)^{\mathrm{T}}\vec{e}^{\mathrm{T}}}\,\mathrm{d}x, \tag{14}$$

where x_c is the coordinate of the middle point and $\vec{e}$ is the unit vector in the direction of the edge.

To be effective, we assumed that the desired mesh edge length along the edge changes linearly, namely:

$$h(x,\vec{e}) = \frac{x_1 - x}{x_1 - x_0}h_0 + \frac{x - x_0}{x_1 - x_0}h_1, \tag{15}$$

where h_0 and h_1 are the desired mesh edge length along $\vec{e}$ at the two ends of the edge computed using Eq. (2). Thus [25],

$$x_c = x_0 + \frac{1}{1+\sqrt{h_1/h_0}}(x_1 - x_0). \tag{16}$$

Remark 4. Although the insertion position in terms of Eq. (16) meets our requirements, others are possible and may work well. For example, when refining an edge of length 3.3 in the transformed space, we will obtain more edge length close to one as desired if we create two subedges of length 1.1 and 2.2 in the first refinement operation.

3.4.2. Selection of diagonals

Another issue in refinement is the appropriate selection of diagonal mesh edges. Fig. 28 depicts ambiguous situations met in applying subdivision templates and indicates their corresponding diagonal edges. In Fig. 28(a), two edges of a triangle are marked as refinement edges. The triangulation of the quadrilateral area (shaded) is not uniquely determined, and two options of creating the diagonal edge are possible (creating M_0^1 or M_1^1). In Fig. 28(b), all edges of a tetrahedron are marked as refinement edges, and three options of creating the interior edge exist (creating M_2^1, M_3^1 or M_4^1).

Fig. 29 shows the importance of creating the proper diagonal using a simple 3D construct. In this example, the initial mesh is a cube consisting of six tetrahedra (Fig. 29(a) is a wireframe view of the mesh). Fig. 29(b) shows a refined mesh after one uniform refinement iteration without the control of diagonal creation (three tetrahedra at top-left are not shown so that a view of the interior is exposed). It can be seen that a long edge across the top and bottom face is created, which yields a refinement that does not reduce the maximal mesh edge length. Fig. 29(c) shows the result mesh after three uniform refinements where the shortest diagonal edge is always created in case of ambiguity. It can be seen that all new tetrahedra are similar to the six initial ones and the maximal mesh edge length decreases to 1/8.

Considering the anisotropic mesh metric field, the extension is to create the shortest diagonal in the transformed space. Consider the 2D example depicted in Fig. 30, where the ellipse indicates the given mesh metric over the shaded quadrilateral and the arrow shows the direction of anisotropy. The diagonal

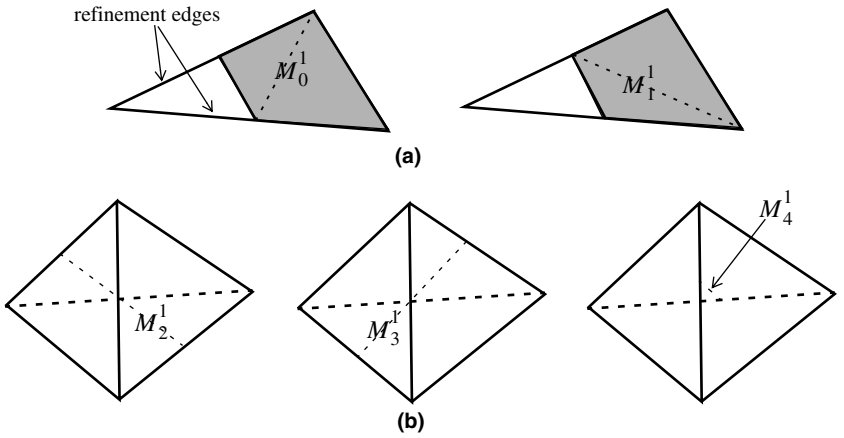

Fig. 28. Ambiguity in creating diagonal edges. Possible diagonals are indicated by the thin dot lines.

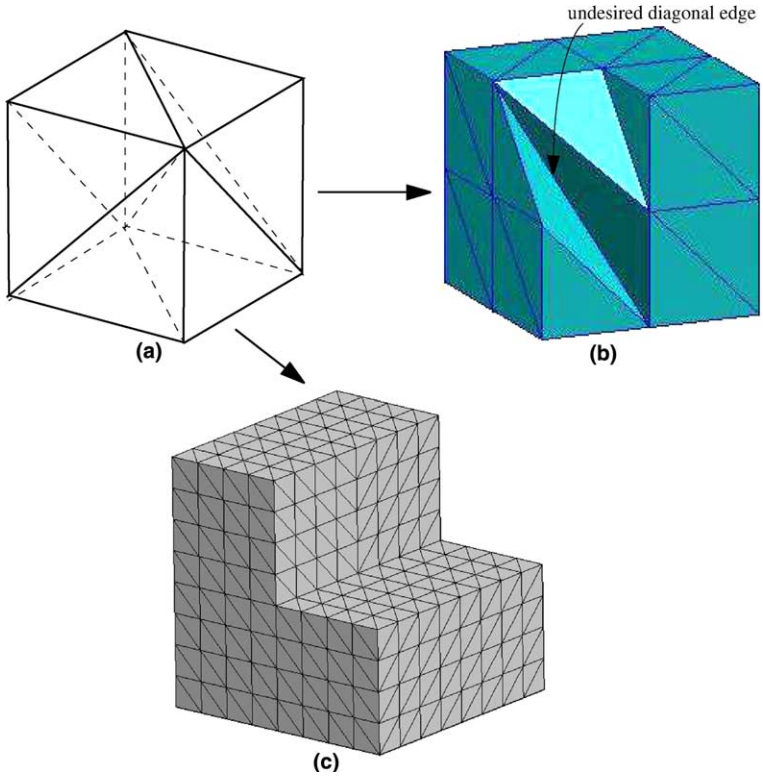

Fig. 29. Example to show the need of creating the short diagonal edge.

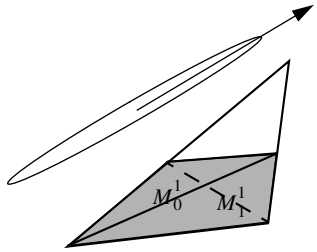

Fig. 30. The diagonal aligning to anisotropy.

M_0^1 is shorter than M_1^1 in transformed space and aligns to the anisotropy better, therefore represents a better option.

4. Examples

This section provides four examples to demonstrate the application of the anisotropic mesh adaptation procedure. In the first example, the mesh metric field is specified using analytical functions to show the

technical aspects of the adaptation procedure. The rest examples show applications in 3D anisotropic adaptive flow simulations, where linear tetrahedral elements are used and the mesh metric field is adaptively constructed by decomposing, scaling the matrix of second derivatives to equilibrate the directional and spatial distribution of the leading interpolation error [25] and smoothing it using the anisotropic mesh metric field gradation procedure described in Ref. [45]. The fourth example also uses a shock detector to catch the solution discontinuity and explicitly specifies discontinuity aligned mesh metric [37]. In all examples, the initial volume meshes are given and the mesh metric fields are piecewise represented by attaching the

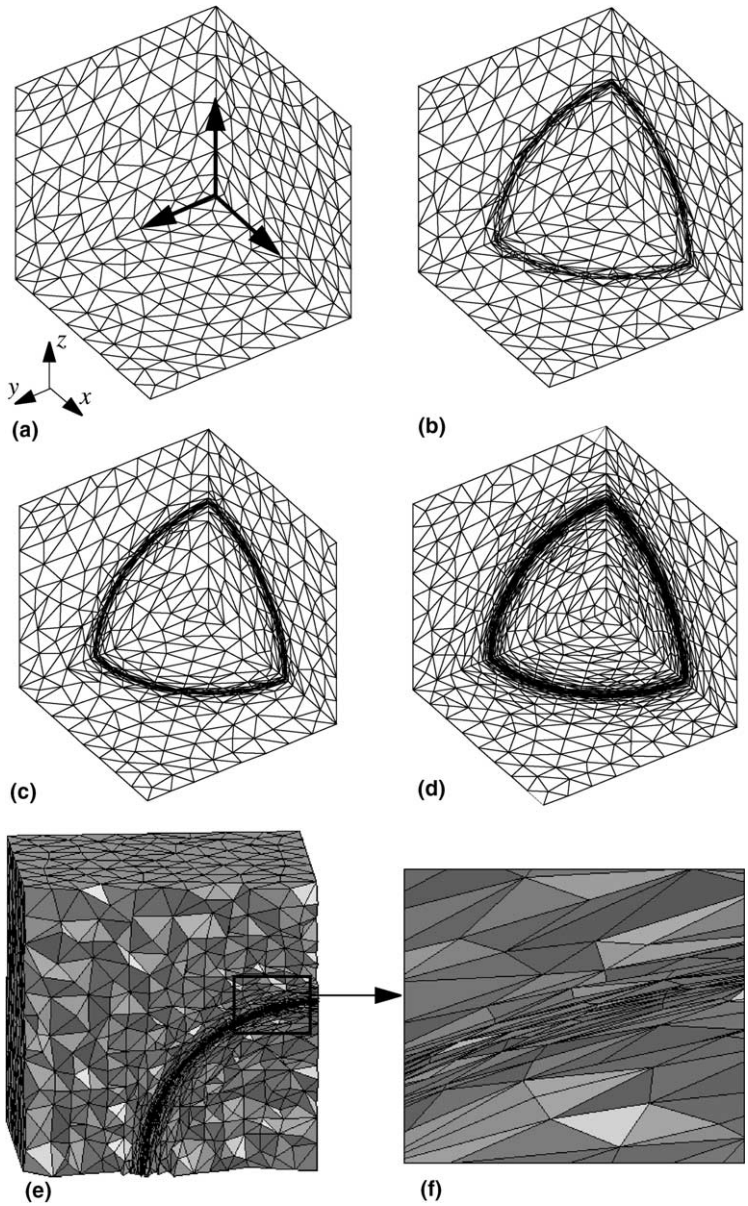

Fig. 31. Evolution of meshes during adaptation.

transformation matrices to all vertices of this volume mesh [25]. When the volume mesh changes in mesh adaptation, the mesh metric attached to new vertex (or moved vertex) is computed if analytical functions are used, otherwise, it is interpolated based on the existing vertex values.

4.1. Expanding spherical shock

Fig. 31(a) shows a coordinate system originated at a corner of a $1 \times 1 \times 1$ cubic domain and a uniform initial mesh of the domain. The desired transformation matrix field to capture an expanding spherical shock is

$$\mathbf{Q}(x,y,z) = \begin{bmatrix} \vec{e}_r \\ \vec{e}_\theta \\ \vec{e}_\varphi \end{bmatrix} \begin{bmatrix} 1/h_r & 0 & 0 \\ 0 & 1/h_\theta & 0 \\ 0 & 0 & 1/h_\varphi \end{bmatrix} \tag{17}$$

with

$$\begin{cases} h_r = 0.125(1 - \mathrm{e}^{-3|r^2 - t^2|}) + 0.00125, \\ h_\theta = h_\varphi = 0.125, \end{cases} \tag{18}$$

where $r = \sqrt{x^2 + y^2 + z^2}$, t is a given parameter related to shock expansion, $\vec{e}_r$ is a unit vector in radial direction, particularly $\vec{e}_r = [x/r, y/r, z/r]$, $\vec{e}_\theta$ is any unit vector orthogonal to $\vec{e}_r$ and $\vec{e}_\varphi = \vec{e}_\theta \times \vec{e}_r$. Note that $\vec{e}_r$ would be a zero vector at the origin, where an isotropic mesh size of 0.125 can be used instead since $h_r \simeq h_\theta = h_\varphi$.

Fig. 31 shows the evolution of the mesh towards satisfying the spherical anisotropic mesh size field of $t = 0.6$. Fig. 31(b) and (c) show two intermediate meshes and 31(d) gives the final adapted mesh. Fig. 31(e) shows the adapted mesh from another viewpoint with tetrahedra in front of a plane not shown to expose a view of the interior. Fig. 31(f) is a zoomed view of the rough interior tetrahedra. The anisotropy

Table 3
Statistics of performed mesh modifications in shape correction process

	Edge swap	Split + relocation	Double split + collapse	Split + collapse	Face swap
# of operations	2283	63	19	11	10

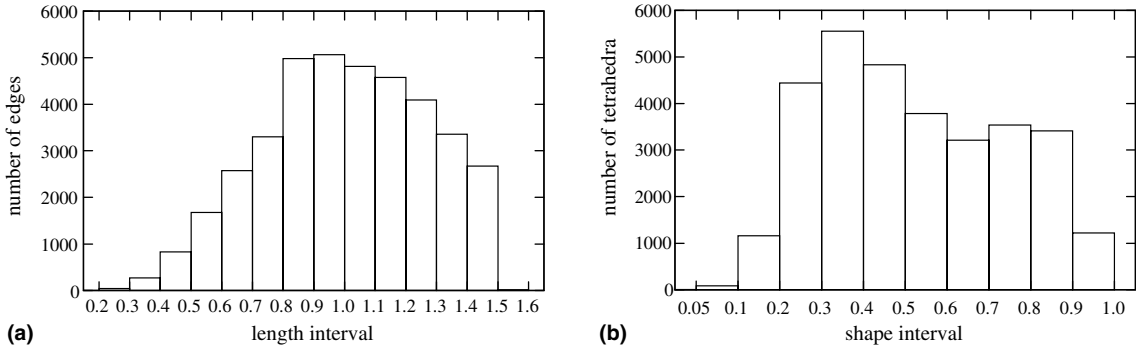

Fig. 32. Adapted length and mean ratio histogram in transformed space.

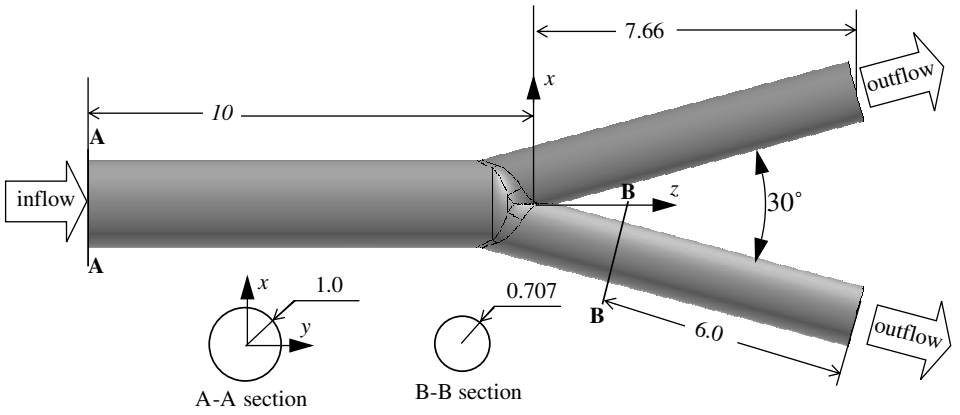

Fig. 33. Schematic diagram of bifurcation blood vessel.

of the mesh along the spherical surface can be clearly seen. The number of tetrahedra in the mesh is increased from 6860 to 40,250 during the anisotropic refinement.

To show that edge swap is the most effective operation in eliminating sliver tetrahedra while other local mesh operations may also be useful, Table 3 gives statistics of the performed local mesh modifications in the shape correction process to eliminate tetrahedra with mean ratio less than 0.2 in transformed space. It can be seen that edge swap is the most effective, followed by splitting plus relocation operation, double split plus collapsing operation, single split plus collapsing operation and face swap.

To show how well the given analytic mesh metric field has been met, Fig. 32 provides the edge length and mean ratio histogram of the adapted mesh in transformed space. The interval of desired mesh edge length is set to [0.707, 1.414] in this example. From Fig. 32(a), it can be seen that the majority (86%) of the mesh edges have length at interval [0.7, 1.5]. For element shapes of the adapted mesh, the worst mean ratio value is at interval [0.05, 0.1], and 99.7% tetrahedra have shape greater than 0.1. Here, it should be pointed out that, the shape interval with the peak number of tetrahedra is [0.3, 0.4] and increasing the mean ratio threshold cannot move the peak closer to 1 since the curvature of the curved anisotropy is small compared with the desired tangential edge length.[3]

To demonstrate the capability of capturing the expanding spherical shock, a new mesh metric field is defined by changing parameter t from 0.6 to 0.62, then the mesh adaptation algorithm is applied once again using the adapted mesh in Fig. 31(d) as initial mesh. As expected, the mesh is updated to align to the new mesh metric field by performing coarsening near $r = 0.6$ and refinement near $r = 0.62$. In particular, since the desired radial mesh edge length changes from 0.00125 to 0.011 at $r = 0.6$, 3758 edge collapse operations are applied there in the first coarsening stage of the mesh adaptation algorithm (collapse edges shorter than 0.707 in transformed space), decreasing the number of tetrahedra in the mesh from 40,250 to 19,563. While nearby $r = 0.62$, since desired radial mesh edge length changes from 0.011 to 0.00125, incremental refinements are applied, increasing the number of tetrahedra in the mesh to 36,839.

4.2. Bifurcation of blood vessel

This example demonstrates the controlled anisotropic meshes appropriate for cardiovascular system models. Fig. 33 illustrates the geometric model domain approximating a portion of blood vessel with a

[3] For planar anisotropy over the cubic domain, statistics have shown that the peak interval is [0.8, 0.9] and 60% tetrahedra have mean ratio value greater than 0.7 in transformed space.

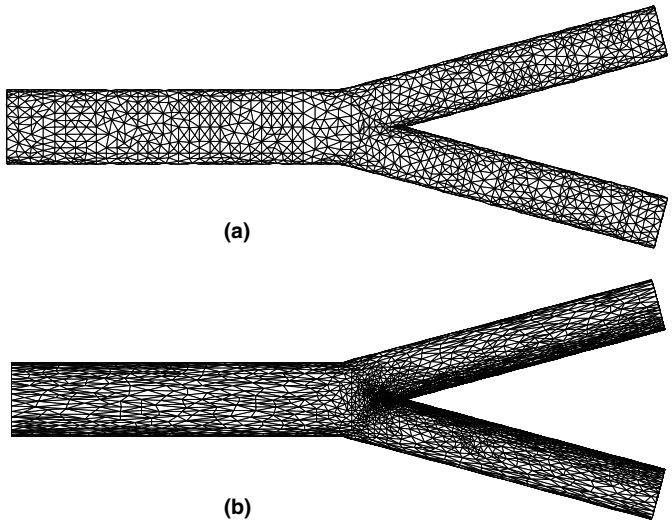

Fig. 34. Top view of the initial mesh and anisotropically refined mesh: (a) initial mesh; (b) refined mesh after the 4th mesh adaptations.

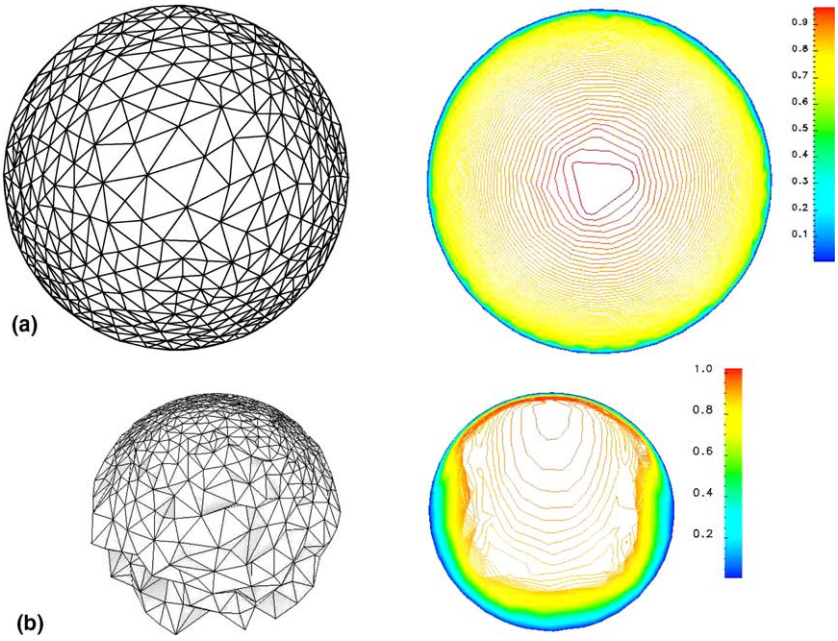

Fig. 35. Mesh and flow speed contour on two blood vessel sections: (a) inlet; (b) B–B section.

symmetric bifurcation. At the inlet boundary, the inlet velocity profile indicated below is specified in the z direction and all other velocity components are set to zero.

$$u_3 = \min(25(1-r), (1-r)^{1/7}), \tag{19}$$

where u_3 is the velocity in z direction and $r = \sqrt{x^2+y^2}$ ($r \leqslant 1$). At the two outlet boundaries, zero pressure boundary conditions are applied. At all other boundaries, no slip wall boundary conditions are enforced.

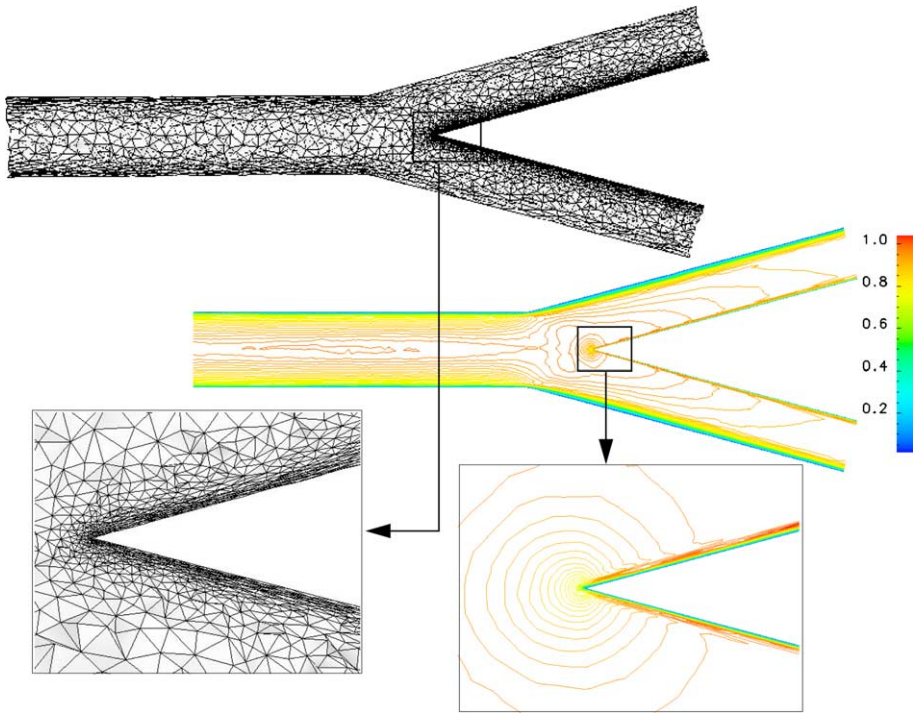

Fig. 36. Interior mesh and flow speed contour on two xz plane.

The advective form of time-dependent, incompressible Navier–Stokes equations is [11,43]:

$$\begin{cases} u_{i,i} = 0, \\ \dot{u}_i + u_i u_{i,j} = (-p_{,i} + \tau_{ij,j} + f_i)/\rho, \end{cases} \tag{20}$$

where p is the pressure, ρ is the density (assumed constant), u_i is the ith component of velocity, f_i is the prescribed body force, and τ_{ij} is the viscous stress tensor given by

$$\tau_{ij} = \mu(u_{i,j} + u_{j,i}), \tag{21}$$

where μ is the kinematic viscosity. The blood vessel flow is simulated by solving this PDE using stabilized finite element formulations (SUPG) [11].

Fig. 34 gives the top view of initial mesh and the anisotropically refined mesh after the fourth application of mesh adaptations. The initial mesh is uniform and isotropic. It is used to solve the flow problem in solution steps from 0 to 50 (0.5 s for each solution step). The refined mesh is achieved after the application of four anisotropic mesh adaptations at solution step 50, 80, 110, 140 respectively to equilibrate the spatial and directional distribution of leading interpolation error [25], and it is used to solve the flow problem from step 140 to step 170. The number of elements is increased from the initial 38,903 to 47,257 to 71,082 to 129,990 to 270,753 in these four mesh adaptations.

Fig. 35 shows the refined mesh and its associated flow speed contours on two sections of the blood vessel model (see Fig. 33 for definition of the two sections). Fig. 35(a) shows the surface mesh and the flow speed contour at inlet while (b) shows the interior mesh faces and speed contour related to B–B section. Fig. 36 shows the interior mesh faces interacting with the plane of $y = 0$ and the speed contour on the plane, including close-up views to the area near the bifurcation point. It can be seen that mesh

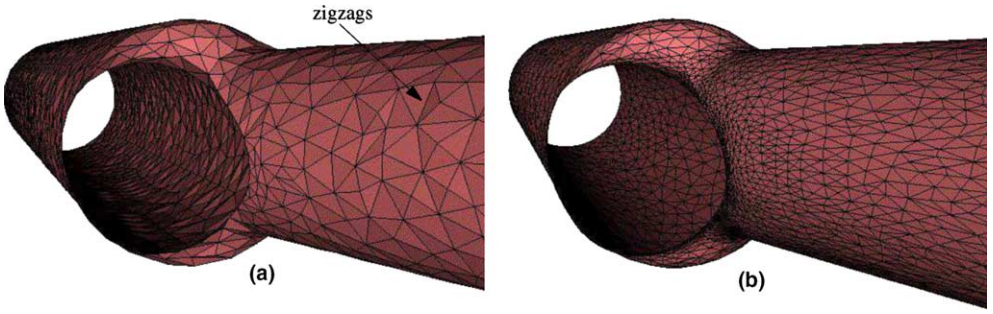

Fig. 37. Mesh better aligned to curved geometries in adaptation: (a) initial mesh faces on wall boundary; (b) adapted mesh faces.

adaptation has created anisotropic elements in the boundary layer of wall surfaces (particularly, elements of a very small radial spacing, somewhat larger azimuthal spacing and large axial spacing), small isotropic elements at bifurcation and quickly developing anisotropy in the boundary layers downstream of the bifurcation, which catches both singularity and boundary layers as indicated by the speed contours.

Fig. 37 shows the surface meshes on wall boundaries with the front cylindrical wall boundary in the downstream of the bifurcation hidden. Fig. 37(a) shows the initial mesh faces. It can be seen that the wall boundary is rough (there are zigzags as indicated) and the geometry at bifurcation is not well approximated. Fig. 37(b) shows the refined and snapped mesh faces. It can be seen that geometric discretization error has been improved when boundary edges in azimuthal direction are refined and modified to better align with the direction of zero curvature, and so is the boundary mesh near bifurcation.

4.3. Flow around a parachute

The interactions between parachute system and the surrounding fluid field are dominant in parachute operations. 3D flow simulation to predict the surrounding flow using stabilized finite element formulations is of interest and feasible using a carefully defined tetrahedral mesh that was not changed during simulation (see for example [42]). Experience is needed to generate such a mesh and, to be conservative, the mesh is usually over-refined. This example presents alternative for the mesh control of the fluid domain near a parachute using the mesh adaptation procedure.

Consider a parachute with four cut-off at corners in a box domain as illustrated in Fig. 38. Let u_i be the ith component of velocity, τ represent stress tensor, $(\vec{\mathbf{n}} \cdot \tau)$ be the stress acting on the surface with normal $\vec{\mathbf{n}}$, and $(\vec{\mathbf{n}} \cdot \tau)_i$ indicate the ith component. The initial velocity field is simply $u_1 = u_2 = 0$, $u_3 = 1$ everywhere. The following boundary conditions are applied:

- On the parachute, no slip wall, i.e. $u_1 = u_2 = u_3 = 0$.
- At the inlet boundary, $u_1 = u_2 = 0$, $u_3 = 1$.
- At the outflow boundary, zero pressure, i.e. $(\vec{\mathbf{n}} \cdot \tau) = 0$.
- At the two side box faces normal to x-axis, no slip in x direction, i.e. $u_1 = 0$, $(\vec{\mathbf{n}} \cdot \tau)_2 = 0$ and $(\vec{\mathbf{n}} \cdot \tau)_3 = 0$.
- At the two side box faces normal to y-axis, no slip in y direction, i.e. $u_2 = 0$, $(\vec{\mathbf{n}} \cdot \tau)_1 = 0$ and $(\vec{\mathbf{n}} \cdot \tau)_3 = 0$.

Fig. 39 visualizes the initial mesh by showing a slice of the 3D mesh interacting with the plane of $x = y$. It consists of 123,948 tetrahedra. To allow pressure discontinuity on the two sides of the parachute, the volume mesh has been split by the canopy surface such that two duplicated pieces of surface meshes exist, one is classified on the upper side of the canopy surface and another is classified on the lower side.

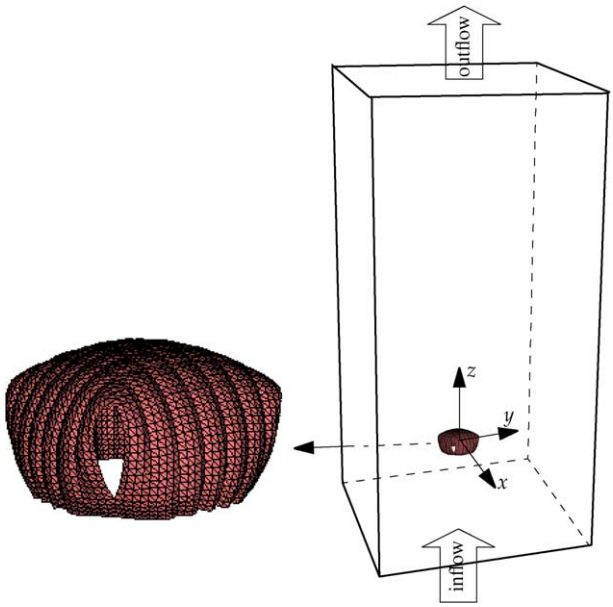

Fig. 38. Schematic diagram for the parachute problem.

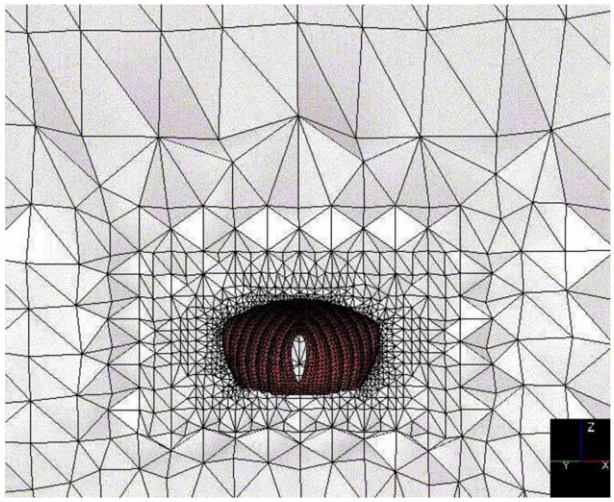

Fig. 39. Initial mesh on plane $x = y$ for the parachute problem.

This unsteady solution has been obtained by solving the same time-dependent, incompressible Navier–Stokes equations given in Section 4.2. The initial mesh is used in the first 60 solution steps, then the mesh is adapted every 30 steps. Fig. 40 shows the mesh after the 8th mesh adaptation, which is used to solve the flow from step 270 to step 300. The total number of tetrahedra has reached 563,487. Fig. 40(a) shows the side view of the evolving mesh (the slice of the 3D mesh faces interacting with plane $x = y$), while (b) shows the top view (the slice of interior mesh faces interacting with plane $z = -0.6$). Fig. 41 gives the speed contour at solution step 300 on these two planes. It can be seen that mesh

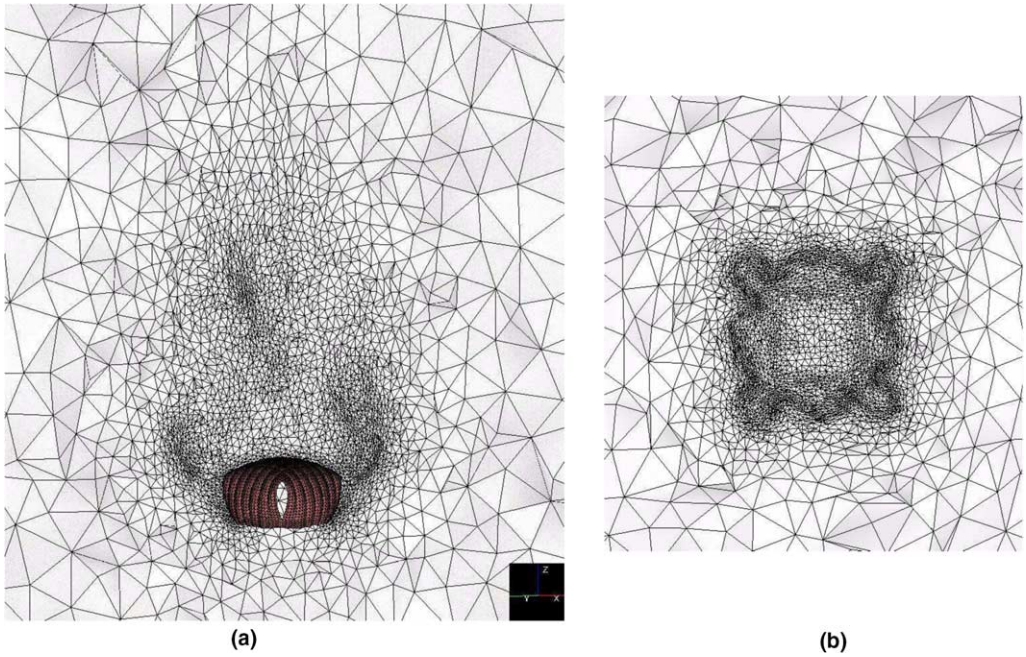

Fig. 40. Evolving mesh after the 8th mesh adaptation: (a) side view; (b) top view.

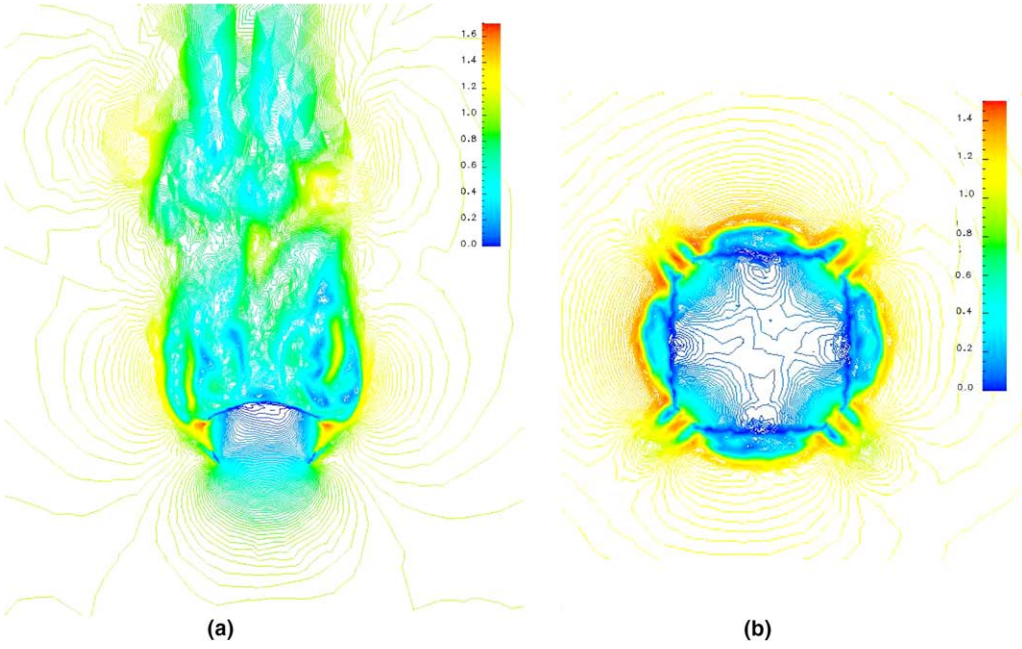

Fig. 41. Evolving speed contour at solution step 300: (a) side view, (b) top view.

adaptation has created small elements near the parachute and in the downstream out of the cut-off, and refined the region on top of the parachute in a way that can catch unsteady flow around step 300. Also the result mesh is smoothly graded in all directions. Fig. 42 shows a closed-up view to the downstream mesh and velocity near one of the cut-off corners. One can see anisotropic elements aligned to the flow.

4.4. 3D cannon blast

This example applies the mesh adaptation procedure to 3D cannon blast problem governed by Euler's equation. Fig. 43 shows a perforated cannon inside a box domain. The cut plane indicates a slice of the 3D

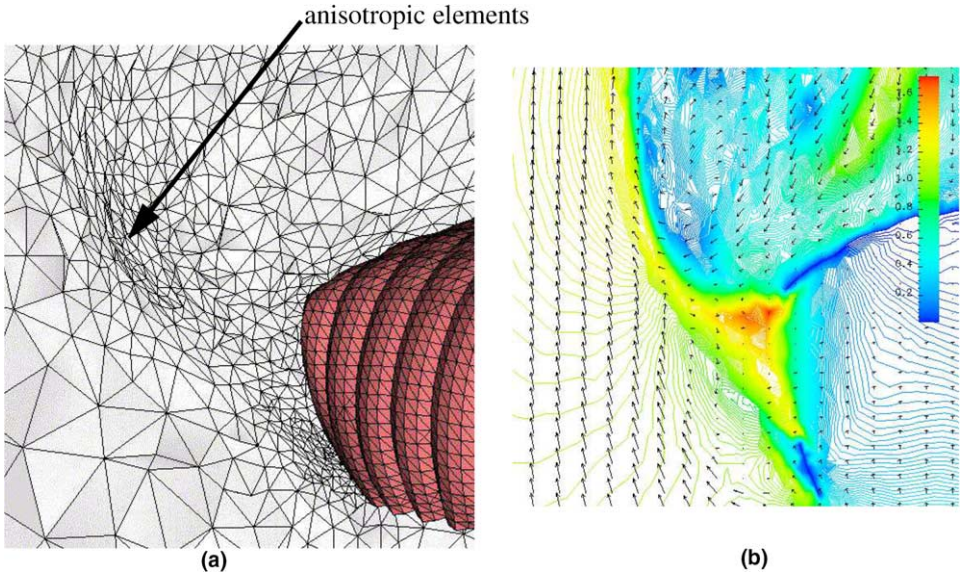

Fig. 42. Zoom near one cut-off corner: (a) mesh, (b) speed contour and velocity vector field.

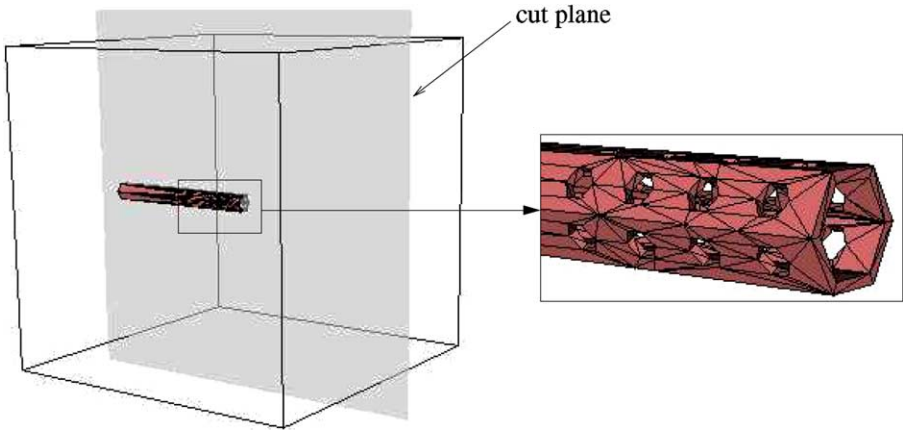

Fig. 43. Analysis domain: a cannon with 24 perforated holes inside a box.

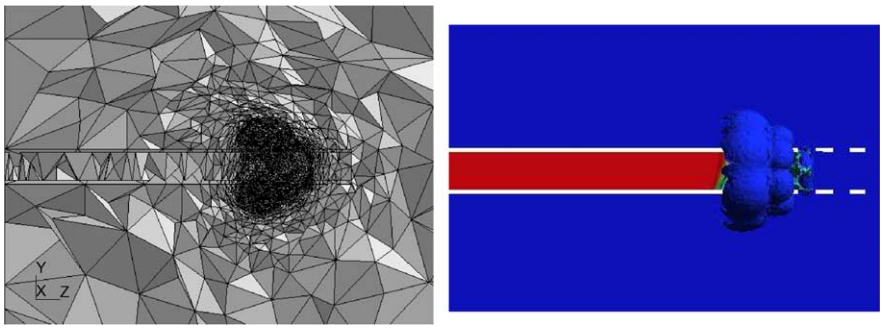

Fig. 44. Result mesh and density distribution after 700 adaptive cycles: the slice of the tetrahedral mesh associated with the cut plane (left), and contour surfaces of the density field (right).

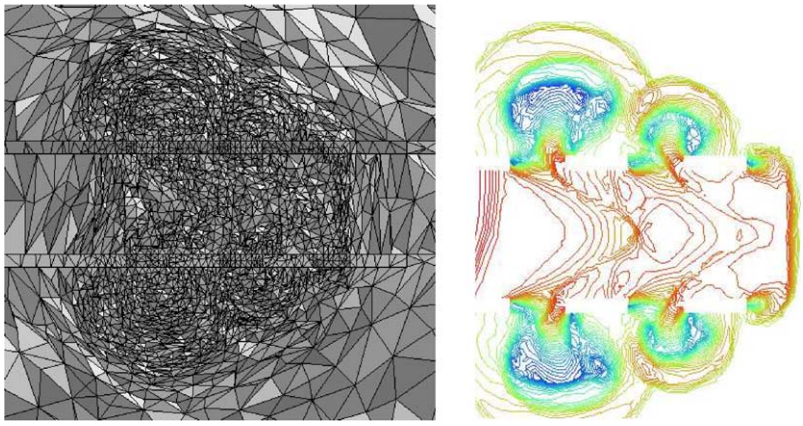

Fig. 45. Close-up view of the slice of the tetrahedral mesh (left) and the density contour on the cut plane (right).

domain where mesh and density results are shown. Fig. 44 shows the slice of mesh associated with the cut plane and the density contour surfaces after 700 cycles of solutions and mesh adaptations, where the shock inside the cannon passed half of the perforated hole. Fig. 45 is a close-up view near the perforated holes.

5. Closing remarks

This paper presented an anisotropic mesh modification procedure for general 3D geometries. A mesh metric field has been used to represent the desired element shape and size distribution. The procedure is an effective integration of four related components: refinement, coarsening, projecting boundary vertices and shape correction. At the low level, it operates as a sequence of mesh modifications efficiently determined to make the mesh satisfy the given mesh metric field and properly represent curved geometries. Key technical features presented to ensure effective alignment include:

- incremental refinement that refines a set of longest mesh edges with respect to the given mesh metric field using all possible element subdivision patterns;

- coarsening that eliminates the shortest local mesh edge in an order of topologically every other vertex using edge collapse dominated local mesh modifications;
- tetrahedral shape analysis technique to guide the intelligent selection of local mesh modifications.

The application of the procedure was demonstrated using priori analytical mesh metric field and adaptively determined mesh metric field in 3D flow simulations.

Acknowledgments

The authors would like to thank Prof. Kenneth E. Jansen for providing the needed incompressible flow simulation software, Prof. Tayfun E. Tezduyar for providing the surface mesh of the parachute problem and Nicolas Chevaugeon for providing the pictures of the cannon blast problem. The contributions from the reviewers are gratefully acknowledged.

References

[1] R.C. Almeida, P.A. Feijoo, A.C. Galeao, C. Padra, R.S. Silva, Adaptive finite element computational fluid dynamics using an anisotropic error estimator, Comput. Methods Appl. Mech. Engrg. 182 (3–4) (2000) 379–400.
[2] D.N. Arnold, A. Mukherjee, L. Pouly, Locally adapted tetrahedral meshes using bisection, SIAM J. Sci. Comput. 22 (2000) 431–448.
[3] E. Bansch, Refinement in 2 and 3 dimensions, Impact Comput. Sci. Engrg. 3 (1991) 181–191.
[4] M.W. Beall, M.S. Shephard, A general topology-based mesh data structure, Int. J. Numer. Methods Engrg. 40 (1997) 1573–1596.
[5] J. Bey, Tetrahedral grid refinement, Computing 55 (1995) 271–288.
[6] W.H. Beyer, CRC Standard Mathematical Tables, 28th ed., CRC Press, Boca Raton, FL, 1987.
[7] F. Bornemann, B. Erdmann, R. Kornhuber, Adaptive multilevel methods in three-dimension spaces, Int. J. Numer. Methods Engrg. 36 (1993) 3187–3203.
[8] H. Borouchaki, P.L. George, F. Hecht, P. Laug, E. Saltel, Delaunay mesh generation governed by metric specifications—part i: Algorithms and part ii: Applications, Finite Elements Anal. Design 25 (1997) 61–83, 85–109.
[9] A. Bowyer, Computing Dirchlet tessellations, Comput. J. 24 (1981) 162–166.
[10] E. Briere de l'Isle, P.L. George, Optimization of tetrahedral meshes, in: I. Babuska, J.E. Flaherty, W.D. Henshaw, J.E. Hopcroft, J.E. Oliger, T. Tezduyar (Eds.), Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations, Springer-Verlag, 1993, pp. 97–128.
[11] A.N. Brooks, T.J.R. Hughes, Streamline upwind/Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations, Comput. Methods Appl. Mech. Engrg. 32 (1982) 199–259.
[12] G.C. Buscaglia, E.A. Dari, Anisotropic mesh optimization and its application in adaptivity, Int. J. Numer. Methods Engrg. 40 (1997) 4119–4136.
[13] M.J. Castro-Diaz, F. Hecht, B. Mohammadi, Anisotropic unstructured grid adaptation for flow simulations, Int. J. Numer. Methods Fluids 25 (4) (1997) 475–491.
[15] H.L. de Cougny, M.S. Shephard, Parallel refinement and coarsening of tetrahedral meshes, Int. J. Numer. Methods Engrg. 46 (1999) 1101–1125.
[16] H.L. de Cougny, M.S. Shephard, M.K. Georges, Explicit node point mesh smoothing within the octree mesh generator, SCOREC Report 1990-10, Rensselaer Polytechnic Institute, 1990.
[17] J. Dompierre, M.-G. Vallet, Y. Bourgault, M. Fortin, W.G. Habashi, Anisotropic mesh adaptation: towards user-independent, mesh-independent and solver independent cfd. Part iii: Unstructured meshes, Int. J. Numer. Methods Fluids 39 (2002) 675–702.
[18] P.J. Frey, P.L. George, Mesh Generation, HERMES Science Europe Ltd., 2000.
[19] P.L. George, H. Borouchaki, P. Laug, An efficient algorithm for 3d adaptive meshing, in: B.H.V. Topping (Ed.), Finite Elements: Techniques and Developments, Civil-Comp Press, Edinburgh, Scotland, 2000, pp. 1–11.
[20] P.L. George, F. Hecht, Non isotropic grids, in: J. Thompson, B.K. Soni, N.P. Weatherill (Eds.), CRC Handbook of Grid Generation 20.1-20.29, CRC Press, Boca Raton, FL, 1999.
[21] K.E. Jansen, M.S. Shephard, M.W. Beall, On anisotropic mesh generation and quality control in complex flow problems, in: Tenth International Meshing Roundtable, 2001.
[22] B. Joe, Three-dimensional triangulations from local transformations, SIAM J. Sci. Comput. 10 (1989) 718–741.

[23] M.T. Jones, P.E. Plassmann, Adaptive refinement of unstructured finite-element meshes, Finite Elements Anal. Design 40 (1997) 41–60.
[24] Y. Kallinderis, P. Vijayan, Adaptive refinement-coarsening scheme for three dimensional unstructured meshes, AIAA J. 31 (8) (1993) 1440–1447.
[25] X. Li, Mesh modification procedures for general 3-D non-manifold domains, Ph.D. thesis, Rensselear Polytechnic Institute, August 2003.
[26] X. Li, M.S. Shephard, M.W. Beall, Accounting for curved domains in mesh adaptation, Int. J. Numer. Methods Engrg. 58 (2003) 247–276.
[27] A. Liu, B. Joe, On the shape of tetrahedra from bisection, Math. Comput. 63 (1994) 141–154.
[28] A. Liu, B. Joe, Relationship between tetrahedron shape measures, BIT 34 (1994) 268–287.
[29] A. Liu, B. Joe, Quality local refinements of tetrahedral meshed based on bisection, SIAM J. Sci. Comput. 16 (1995) 1269–1291.
[30] A. Liu, B. Joe, Quality local refinements of tetrahedral meshed based on 8-subtetrahedron subdivision, Math. Comput. 65 (1996) 1183–2000.
[31] S.H. Lo, 3-d anisotropic mesh refinement in compliance with a general metric specification, Finite Elements Anal. Design 38 (2001) 3–19.
[32] J.M. Maubach, Local bisection refinement for *n*-simplicial grids generated by reflection, SIAM J. Sci. Comput. 16 (1995) 210–227.
[33] D.J. Mavriplis, Adaptive mesh generation for viscous flows using Delaunay triangulation, J. Comput. Phys. 90 (1990) 271–291.
[34] P. Moller, P. Hansbo, On advancing front mesh generation in three dimensions, Int. J. Numer. Methods Engrg. 38 (1995) 3551–3569.
[35] C.C. Pain, A.P. Umpleby, C.R.E. de Oliveria, A.J.H. Goddard, Tetrahedral mesh optimization and adaptivity for steady-state and transient finite element calculations, Comput. Methods Appl. Mech. Engrg. 190 (2001) 3771–3796.
[36] J. Peraire, J. Peiro, K. Morgan, Adaptive remeshing for three dimensional compressible flow computation, J. Comput. Phys. 103 (1992) 269–285.
[37] J.-F. Remacle, X. Li, M.S. Shephard, J.E. Flaherty, Anisotropic adaptive simulation of transient flows using discontinuous Galerkin methods, Int. J. Numer. Methods Engrg. 62 (2005) 899–923.
[38] M.C. Rivara, Selective refinement/derefinement algorithms for sequences of nested triangulations, Int. J. Numer. Methods Engrg. 28 (1989) 2889–2906.
[39] M.C. Rivara, A 3-d refinement algorithm suitable for adaptive and multi-grid techniques, Comm. Appl. Numer. Methods 8 (1992) 281–290.
[40] M.C. Rivara, New longest-edge algorithms for the refinement and/or improvement of unstructured triangulations, Int. J. Numer. Methods Engrg. 40 (1997) 3313–3324.
[41] W. Speares, M. Berzins, A 3d unstructured mesh adaptation algorithm for time dependent shock-dominated problems, Int. J. Numer. Methods Fluids 25 (1997) 81–104.
[42] K.B. Stein, V. Kalro, T.E. Tezduyar, M. Accorsi, Parachute fluid–structure interactions: 3-d computation, Comput. Methods Appl. Mech. Engrg. 190 (2000) 373–386.
[43] C.H. Whiting, K.E. Jansen, A stabilized finite element method for incompressible Navier–Stokes equations using a hierarchical basis, Int. J. Numer. Methods Fluids 35 (2001) 93–116.
[44] S. Yamakaw, K. Shimada, High quality anisotropic mesh generation via ellipsoidal bubble packing, in: Ninth International Meshing Roundtable, 2000.
[45] X. Li, J.-F. Remacle, N. Chevaugeon, M.S. Shephard, Anisotropic mesh gradation control, in: Proceedings of the 13th International Meshing Roundtable, 2004.