An Improved Hexahedral Mesh Matching Algorithm

Jinming Chen, Hua Zhu, Shuming Gao*, and Haiyan Wu

CAD&CG State Key Laboratory, Zhejiang University
smgao@cad.zju.edu.cn

Summary. Mesh matching is an effective way to convert the non-conforming interfaces between two hexahedral meshes into conforming ones, which is very important for achieving high quality finite element analysis. However, the existing mesh matching algorithm is not robust and efficient enough. In this paper, the algorithm is improved in three aspects: by introducing a more precise criteria for chord matching and the concept of partition chord set, complex interfaces with internal loops can be handled more efficiently; by putting forward a mesh quality evaluation method, the sheet extraction operation during mesh matching is sped up; by bringing up a new solution, self-intersecting sheet can be inflated locally. Test results demonstrate the effectiveness of the improved mesh matching algorithm.

Keywords: Mesh matching, Sheet operation, Hexahedral mesh, Finite method analysis.

1 Introduction

For finite element analysis, hexahedral meshes are usually better than tetrahedral meshes[1]. Thus a lot of work has been conducted on hexahedral mesh generation. Up to now, however, it is still too difficult to automatically generate hexahedral meshes of complex CAD models. As a result, a complex model that cannot be well meshed as a whole is usually decomposed into a number of components each of which can be meshed automatically by certain algorithms, and then the hexahedral meshes of all components are combined together as the resultant meshes of the whole CAD model. One problem of this method is that the interfaces between the hexahedral meshes of the components are usually non-conforming.

The assembly meshing is a typical example. We usually mesh the assembly components, i.e. parts, separately, instead of the whole assembly. Because

* Corresponding author.

J. Sarrate & M. Staten (eds.), *Proceedings of the 22nd International Meshing Roundtable*,
DOI: 10.1007/978-3-319-02335-9_11, © Springer International Publishing Switzerland 2013

each part of the assembly may be meshed by different FEA engineers, or using different meshing techniques, or taking different element density, the meshes of the parts often become non-conforming on the assembly interfaces.

In addition to assembly meshing, the reusing of the existing hexahedral mesh models also faces the non-conforming interfaces problem. Because the high quality hexahedral mesh generation of complex CAD models is very difficult and time-consuming, reusing of the existing hexahedral mesh models in the library whose size keeps expanding is becoming more and more important. In reuse-based scenario, for a complex CAD model, instead of meshing it from scratch, its hexahedral meshes are generated by combining some existing mesh models searched and some new generated meshes. Obviously the searched mesh models are hardly conforming to the new generated one on their interfaces.

Currently, in order to obtain an integral solution based on the mesh model with non-conforming interfaces, it is necessary to impose constraints on the non-conforming interfaces, like gap elements, multi-point constraints etc[2, 3, 4]. Nevertheless, when mesh models are large and interfaces are complex, the tasks of correctly imposing the required constraints become tedious and erroneous, and the analysis result is usually not as accurate as that obtained based on the mesh model with conforming interfaces.

Several algorithms have been brought out to generate conforming meshes of complex CAD models. Cooper Tool projects all the barrels' outlines on the interfaces to generate conforming hexahedral meshes[5]. However, this method can only handle situations of sweepable models conjoining in the sweeping direction. Graft Tool[6] and Multi-axis Cooper Tool[7] relax the conjoining direction restriction, but the former needs the branch models to be sweepable and the latter not only needs the child models to be sweepable but also has limitations on the angle formed by two child models.

Staten et al. proposed an algorithm called mesh matching, which gradually converts the non-conforming hexahedral mesh interfaces into conforming ones using some dual operations like pillowing, dicing etc[8, 9]. Staten, later in his dissertation, brought up an automatic strategy in order to get rid of manual intervention[1]. They are the first to use sheet technologies to provide an alternative method dealing with non-conforming interfaces. Although it is an excellent work, there are still some aspects to improve. For example, the efficiency of the sheet extraction operation during mesh matching is low and it seems that self-intersecting sheets can not be locally dealt with.

Recently, Lo proposed another algorithm for merging non-conforming hexahedral meshes[10]. However, the resultant meshes contain non-hexahedral elements like tetrahedra, pyramids. For finite element analyzing, a pure hexahedral mesh is preferred.

Recognizing the promise of hexahedral mesh matching, the objective of this paper is to make improvements to make the algorithm more capable and efficient. By introducing the concept of partition chord set and adopting a new chord matching criteria, we can deal with complex interfaces with internal

loops more effectively. And we also provide a method to locally inflate self-intersecting sheets. Moreover by adopting a mesh quality prediction strategy, we speed up the sheet extraction operation during mesh matching.

The rest of this paper is organized as follows: Section 2 introduces basic concepts and gives the overview of our algorithm. Section 3 describes the details of the key steps of our improvements. Section 4 shows some examples. Finally, Section 5 gives the conclusions and future work.

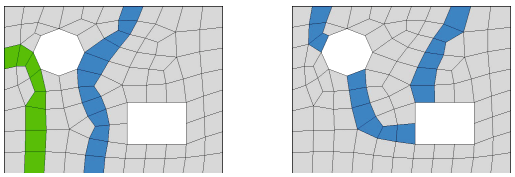
2 Basic Concepts and Overview of Algorithm

2.1 Basic Concepts

In this paper, we bring out new chord matching criteria to make the algorithm to match chords on interfaces with internal loops more efficiently and the concept of partition chord set to enable localization of sheet operations when interfaces contain internal loops. Some related concepts are given in this section. Due to the limited space of this paper, we leave out some common concepts like hexahedral mesh, sheet and quadrilateral mesh etc which can be found in [1, 8, 9, 11, 12].

Definition 1 (Chord). Given a quadrilateral mesh Q and a edge e, a chord c can be represented as a set of edges $c = E_c, E_c \subset Q, e \in E_c$, where E_c is the maximum set of edges that can be recursively found by getting the topological parallel edge of e.

On a quadrilateral mesh, a chord must be associated to a sheet, while a sheet may associate to several chords on this quadrilateral mesh.



(a) (b)

Fig. 1 Partition chord and partition chord set: (a)Two partition chords;(b)One partition chord set

Definition 2 (Partition chord). Given a quadrilateral mesh Q and the edge set E of Q, a chord $c = E_c$ is a partition chord if edges in $E' = E - E_c$ can form two unconnected groups.

For interfaces without inner loops, all the chords are partition chords except for self-intersecting ones, which separate the mesh into more than two

unconnected groups. For interfaces with multiple loops, chords connecting to the same loop are partition chords. Fig. 1(a) shows two partition chords.

Definition 3 (Partition chord set). Given a quadrilateral mesh Q and the edge set E of Q, a chord set $C = \{c_1, c_2, \cdots, c_n\}$ is a partition chord set if $E' = E - E_{cs}$ can form two unconnected groups, where E_{cs} are the edges of C, while $E' = E - E'_{cs}$ cannot form two unconnected groups where E'_{cs} are the edges of $C' \subset C$.

Fig. 1(b) shows one partition chord set. It can be seen that all the chords in the partition chord set connect different loops on the interface.

Definition 4 (Direction of chord). Given a chord $c = E_c$, its direction is determined by the order of the edges in E_c, which can be sorted as $(e_c^1, e_c^2, \cdots, e_c^n)$, where $e_c^i \in E$, and e_c^i and e_c^{i+1} are adjacent to a same quadrilateral face, $i = 1, 2, \cdots, n-1$.

Definition 5 (Closed chord and open chord). Given a chord $c = E_c$, c is called an open chord if $\exists e \in E_c$ on the boundary curves, otherwise c is called a closed chord.

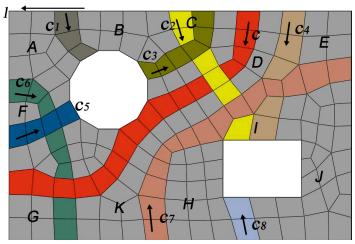


Fig. 2 Illustration of cross sequence, decomposed regions, chord segments and curve segments

Definition 6 (Cross sequence of a chord). The cross sequence of a chord c with respect to a chord set C refers to the sequence of all the chords in C that cross with c, sorted according to c's direction. It can be expressed as $CrossSeq(c|C) = (c_c^1, c_c^2, \cdots, c_c^n)$, where $c_c^i \in C, i = 1, 2, \cdots, n$.

Fig. 2 gives an example. In the figure, the red chord is the chord c whose direction is marked by the black arrows, the chord set C is $\{c_i | i = 1, 2, \cdots, 8\}$, and the cross sequence of c with respect to C is (c_2, c_6).

Definition 7 (Chord segment). The cross sequence of a chord c with respect to a chord set C separates c into a number of sections. These sections are termed as the chord segments of chord c.

Each chord segment of a chord is represented by the pair of chords that separates it from the chord. To express the beginning and ending of the chord, we add two virtual chords, *Start* and *End*. In Fig. 2, the chord segments of chord c with respect to C are $(Start, c_2)$, (c_2, c_6), and (c_6, End).

Definition 8 (Curve segment). Given a chord set C and a boundary curve l, chords in C that intersects with l separate l into a number of sections. These sections are called the curve segments of the boundary curve.

The expression of the curve segment of a boundary curve is similar to the chord segment. Taking the boundary curve l in Fig. 2 as an example, given the direction of l as shown in the figure, the curve segments of l with respect to C are $(Start, c_4)$, (c_4, c_3), (c_3, c_2), (c_2, c_1), and (c_1, End).

Definition 9 (Decomposed regions of mesh). Given a set of chords of a mesh, the mesh is decomposed into a number of regions by the chords, which are defined as decomposed regions of the mesh.

Each decomposed region can be represented by a 3-tuple (c, s, p), where c is one of the chords that surround this region, s is a chord segment of c that is adjacent to the decomposed region, and p stands for the left side or right side of c that the region lies in.

Fig. 2 shows an example of decomposed regions. In this example, chord set C decomposes the quadrilateral mesh into 11 regions A - K. Region A can be expressed as $A = (c_1, (Start, End), right)$. Similarly, $B = (c_2, (Start, c_3), right)$, $C = (c_3, (c_2, End), left)$ etc.

A decomposed region may have several expressions, e.g. decomposed region B can also be expressed as $B = (c_1, (Start, End), left)$ besides $(c_2, (Start, c_3), right)$. All these expressions are interchangeable. In addition, if the set of chords is empty, there will be only one decomposed region, i.e. the whole mesh.

Definition 10 (Polyline of a chord). Given a chord $c = (e_c^1, e_c^2, \cdots, e_c^n)$, its chord polyline is defined as $(m_1, m_2, \cdots, m_n)$, where m_i is the middle point of e_c^i, $i = 1, 2, \cdots, n$.

Fig. 3(a) shows the polylines (dashed lines) of two chords c_1 and c_2.

Definition 11 (Polyline for chord inflation). The polyline for chord inflation refers to the polyline in the mesh along which the chord inflation will be conducted.

Fig. 3(a) shows a polyline for chord inflation with the inflating directions indicated in bold black arrows. And the new chord generated by the chord inflation along the polyline is shown in Fig. 3(b).

Definition 12 (Topological 4-tuple of a chord). Given a set of chords C, the topological 4-tuple of a chord c is defined as (T, E, I, R), where T

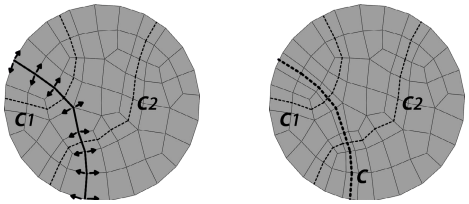


Fig. 3 Two types of polylines: (a)The polyline for chord inflation (in bold black); (b)The new chord generated by chord inflation (in bold black)

stands for c's type, i.e. open chord or closed chord; E stands for its starting and ending boundary curves and the curve segments where c intersects with the curves if c is an open chord or E is empty set if c is a closed chord; I refers to the cross sequence of c with respect to C and the chord segments where these chords intersects with c; R refers to the decomposed regions that c passes created by chords in C.

In essence, topological 4-tuple of a chord is used to dynamically and precisely characterize a chord during mesh matching. Like the chord, the polyline of the chord also has the topological information, so a similar topological 4-tuple can also be defined on the polyline.

Definition 13 (Polygon associated to two chords). Let chords c_1 and c_2 come from quadrilateral mesh Q_A and Q_B respectively and both are open chords, the polygon associated to these two chords refers to the polygon formed by the polylines of c_1 and c_2 and the segments of the boundary curves between their intersection points with the polylines of c_1 and c_2.

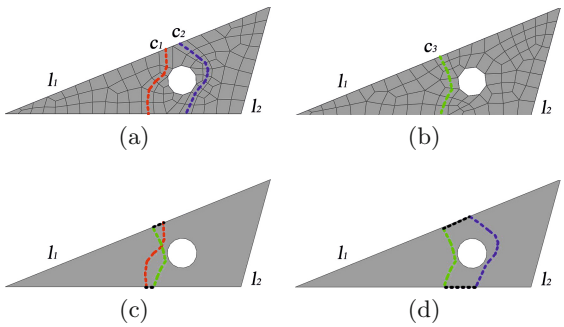


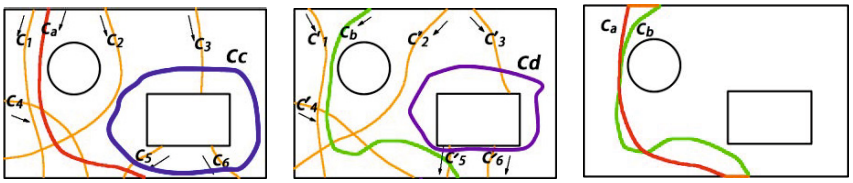
Fig. 4 Polygon associated to two chords: (a)Chords c_1 and c_2 in Q_A; (b)Chord c_3 in Q_B; (c)Polygon associated to c_1 and c_3; (d)Polygon associated to c_2 and c_3

Fig. 4(c) shows the polygon associated to chord c_1 and chord c_3 where the black dashed lines are the segment on l_1 and l_2 between the intersection

points with the polylines of c_1 and c_3. It can be seen that the polygon shown is self-intersecting and contains no internal loops while the polygon associated to c_2 and c_3 shown in Fig. 4(d) contains a circular loop.

Definition 14 (Topological equivalence of two chords). Chord c_a and c_b are in quadrilateral mesh Q_A and Q_B respectively, the sets of matched chords on Q_A and Q_B are C_{pair-A} and C_{pair-B} respectively, and $P_{pair} = \{(c_i, c_i')|c_i \in C_{pair-A}, c_i' \in C_{pair-B}$, where c_i and c_i' are matched chords$\}$. c_a and c_b are topologically equivalent with respect to P_{pair} if and only if they meet the following conditions:

1. The two topological 4-tuples of c_a with regard to C_{pair-A} and c_b with regard to C_{pair-B} are homeomorphic;
2. If c_a and c_b both are open chords, then the polygon associated to these two chords should contain no internal loops;
3. if c_a and c_b both are closed chords, then their polylines should surrounds the same internal loops.



(a) (b) (c)

Fig. 5 Illustration of topological equivalence: (a)Chord c_a in mesh A; (b)Chord c_b in mesh B; (c)Polygon associated to c_a and c_b

Fig. 5 shows an example of topological equivalence of two chords, where chords c_i and c_i' $(i = 1, 2, \cdots, 6)$ are matched pairs and c_a and c_b are both open chords. It can be seen that c_a and c_b meet all the conditions in Definition 14, so they are topologically equivalent and can form a matched pair. Similarly, two closed chords c_c and c_d are also topologically equivalent.

2.2 Overview of Algorithm

The algorithm takes two hexahedral meshes (denoted as H_A and H_B) and their corresponding B-Rep models as input.

The two hexahedral meshes contain non-conforming interfaces. After conducting the initial matching, mesh matching gradually changes the topology of the two meshes using sheet operations including sheet extraction and sheet inflation. In order to limits the topological modification of the sheet operations within a certain layers of hexahedra, depth control can be enabled. At

last conforming interfaces will be generated, and then the two hexahedral meshes can be connected on the conforming interfaces.

Algorithm 1 below shows the main procedures of the algorithm when depth control is required. If depth control is not required, the algorithm flow is almost the same except that sheet operations need not to be localized.

Algorithm 1. Mesh matching

1: Find the interface between two input models
2: Get the quadrilateral meshes Q_A and Q_B on the interfaces from H_A and H_B
3: Retrieve all the chords in Q_A and Q_B, store them in C_A and C_B
4: Make initial matching, put the matched chord pairs in P, and put the unmatched chords into C_{uc-A} and C_{uc-B}
5: **while** $C_{uc-A} \neq \emptyset$ **or** $C_{uc-B} \neq \emptyset$ **do**
6: Get an unmatched chord c_a from C_{uc-A} or C_{uc-B}
7: **if** c_a needs to be grouped into a partition chord set **then**
8: Form a partition chord set pcs_a
9: **end if**
10: Choose sheet extraction or sheet inflation
11: **if** Sheet extraction is chosen **then**
12: Do quality evaluation and choose the best strategy for localization
13: Execute sheet extraction on pcs_a or c_a
14: **else**
15: Execute sheet inflation, generating a new sheet on the other mesh
16: Put new paired chords into P
17: **end if**
18: Remove c_a or all chords in pcs_a from C_{uc-A} or C_{uc-B}
19: Perform mesh smoothing
20: **end while**

3 Key Steps of Algorithm

In this section, four key steps of the algorithm are described in detail, including initial matching, check and form partition chord set, sheet inflation and sheet extraction based on mesh quality prediction.

3.1 Initial Matching

After gathering all the chords in quadrilateral mesh Q_A and quadrilateral mesh Q_B, and storing them in $C_{unpair-A}$ and $C_{unpair-B}$ respectively, the initial matching is performed. This step finds out the existing chords that can be matched between the two meshes without doing any dual mesh operations. In this process, whether two chords can be matched is determined by the topological equivalence criteria. If more than one chord meet the topological equivalence criteria, then chord distance is used to help select the best one.

At the very beginning, since there is no matched chords, only T and E in the topological 4-tuple of the chords and the other two requirements in the definition of topological equivalence are checked. Since boundary sheets are more important to the quality of mesh, they are matched prior. For example in Fig. 6(a) and Fig. 6(b) both chords c_1 and c'_1 start from E_1 and end at E_3, and since currently there are no matched chords, so the topological 4-tuples of c_1 and c'_1 are equal. Additionally, their associated polygon contains no internal loops, so they become a matched chord pair.

As the algorithm continues, matched chords become position reference of chords that are to be matched. For example, in Fig. 6(c) and Fig. 6(d) chords c_i and $c'_i (i = 1, 2, \cdots, 5)$ are matched chords and the black arrows show the chords' directions. Topological 4-tuples of chord c_a (denoted in red in Fig. 6(c)) and chord c_b (denoted in red in Fig. 6(d)) are (T =open chord, $E = ((E_1, 1), (E_5, 0))$, $I = ((c_4, (c_3, c_1)), (c_6, (c_5, c_1)))$, $R = (c_4, (c_3, c_1, Left)))$, and ($T$ =open chord, $E = ((E_1, 1), (E_5, 0))$, $I = ((c'_4, (c'_3, c'_1)), (c'_6, (c'_5, c'_1)))$, $R = (c'_4, (c'_3, c'_1, Left)))$. With the matched chord pairs, it can be seen that c_a and c_b are topological equivalent, so they become a new pair of matched chords.

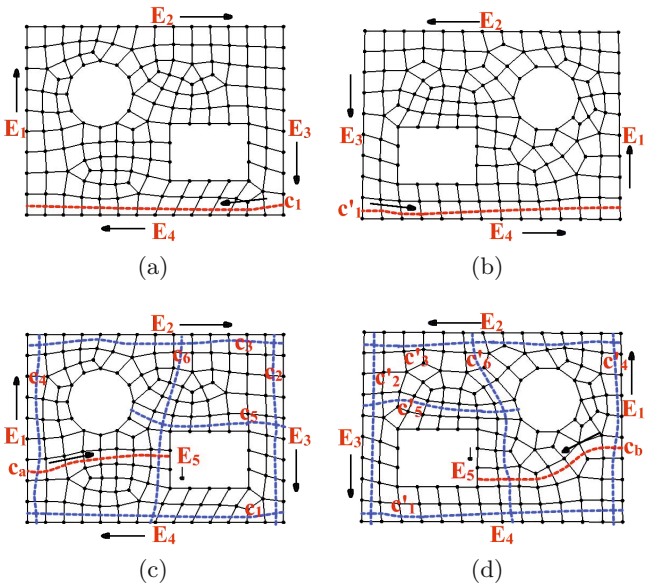


Fig. 6 Initial matching: (a) First matched chord c_1 on quadrilateral mesh Q_A; (b) First matched chord c'_1 on quadrilateral mesh Q_B; (c) Chord c_a on mesh Q_A; (d) Chord c_b on mesh Q_B

3.2 Establishing Form Partition Chord Set

In Algorithm 1 line 7, if c_a is not a partition chord, it need checking whether it should be grouped into a partition chord set. There are two situations when the chord needs to be grouped into a partition chord set: depth control is required, or there exist other chords that associate to the same sheet of c_a.

In the first case, suppose c_a starts from loop l_1 and ends at loop l_2, $l_1 \neq l_2$, we try to find a set of chords $c_2, \cdots, c_n$ whose starting and ending loops can be sorted in a sequence $(l_2, \cdots, l_1)$, using a greedy-based algorithm. If all the chords in the set $C = \{c_a, c_2, \cdots, c_n\}$ do not intersect with each other, C is a partition chord set. If there is no chord in the unpaired chord set that can form a partition chord set with c_1, then we will go further to check the matched chords.

The second case happens when blind holes exist. Chords belong to the same sheet should be grouped into a partition chord set and be dealt together.

3.3 Sheet Inflation

Given an unmatched partition chord or partition chord set on one interface, one way to match it is to create a corresponding chord or a corresponding chord set on the other interface through sheet inflation. In order to correctly do that, firstly we determine the polylines for chord inflation which define the topological positions of the chords to be created, then conduct the sheet inflation to create the sheet based on the polylines determined.

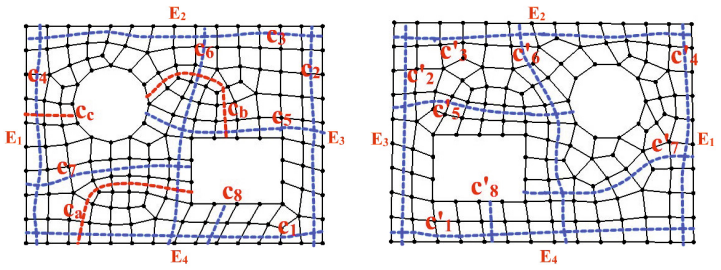
Determination of Polylines for Sheet Inflation Based on Topological Equivalence

In our algorithm, we find these polylines based on the following observation: if chord c is in Q_A and we want to create c' on Q_B to match with c, a polyline in Q_B is the polyline for chord inflation of c' if and only if it is topologically equivalent to c. We first set up a topological 4-tuple in Q_B equivalent to that of c, and then use it as the constraints to search for the polyline.

The specific procedures are described below, illustrated by a simple example shown in Fig. 7-8. Suppose the matched chord set of Q_A and Q_B are C_{pair-A} and C_{pair-B} respectively and the matched chords pairs set is P. In Fig. 7(a), the red chord set $\{c_a, c_b, c_c\}$ is the partition chord set we are now dealing with, and the green chords refer to the matched chord pairs set denoted as $P = \{(c_i, c_i') | i = 1, 2, \cdots, 8\}$.

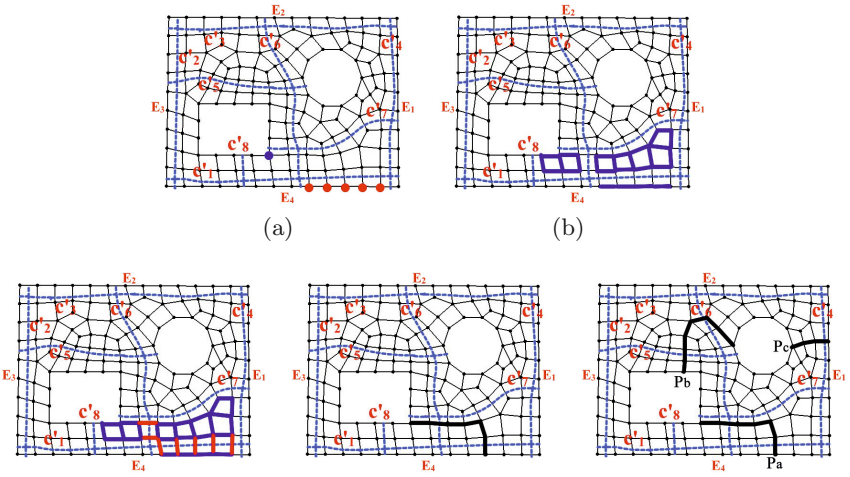
For each chord in $\{c_a, c_b, c_c\}$, we find its corresponding polyline in following 6 steps, taking c_a as an example here:

1. Set up the topological 4-tuple. Get c_a's topological 4-tuple, say t, with respect to C_{pair-A}. According to the chords matching relation in P, we get a topological 4-tuple, say t', with respect to C_{pair-B}.



(a) (b)

Fig. 7 Two quadrilateral meshes before chord inflation: (a)Mesh Q_A with partition chord set $\{c_a, c_b, c_c\}$ and matched chords ;(b)Mesh Q_B with matched chords.



(a) (b) (c) (d) (e)

Fig. 8 Process of finding polylines: (a)Starting and ending nodes (in blue and red respectively);(b)Decomposed regions that new chord should pass (in bold blue); (c)Edges candidates (in bold blue and bold red); (d)The searched polyline for chord inflation (in bold black); (e)All polylines for partition chord set.

2. Find starting and ending nodes. Obtain the starting boundary curve and its curve segment from t'. Mesh nodes in Q_B that lie on the curve segment of the starting boundary curve form a nodes set which is called the starting nodes set N_{start}. Similarly, we get the ending nodes set N_{end} on the ending boundary curve. In Fig. 8(a), the starting nodes are marked as blue while the ending nodes as red.
3. Gather edges in decomposed regions. Find the decomposed regions in Q_B according to t' and collect all the mesh edges in these regions into an edges set E_{candi}. These edges are highlighted in blue in Fig. 8(b).

4. Get edges on chord segments. Get the cross sequence and the chord segments in t'. Retrieve all the edges that lie on the chord segments of the chords in the cross sequence, and put them into E_{candi}. In Fig. 8(c), these edges are highlighted in red.
5. Search the polyline for the chord inflation. Find a polyline from E_{candi}, which starts from one of the nodes in N_{start}, then crosses all the chords in cross sequence in the same order, at last ends in one of the nodes in N_{end}. Note that when the polyline overlaps with other polylines in the partition chord set, it is modified by manual interaction to avoid the overlaps.

Using the above algorithm, the two polylines p_b and p_c for c_b and c_c are gotten, as shown in Fig.8(e).

Sheet Inflation Capable of Handling Self-intersecting Sheets

After finding out all the polylines of $\{c_a, c_b, c_c\}$ in Fig. 7(a), now we create the new sheet using the sheet inflation operation. These polylines will guarantee that the new chords on Q_B associated with the new sheet created are topologically equivalent to the corresponding chords on Q_A.

If what we are currently dealing with is partition chord or partition chord set, pillowing can be used to generate sheets[13]. It first select a number of hexahedra to form a shrink set. Then the shrink set shrinks, creating blank space between the set and the rest of the mesh. At last, a new sheet is formed in the gap, so as its associated chord on the interface. Since $\{c_a, c_b, c_c\}$ is a partition chord set, we use pillowing to generate the sheet. The processes are shown in Fig. 9.

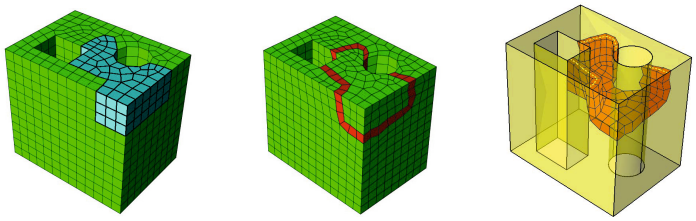


Fig. 9 Pillowing: (a)A shrink set is selected; (b)The shrink set shrinks and a new sheet is created filling the gap; (c)The new sheet and its three associated chords on the interface (marked as dashed yellow line)

If what we are currently dealing with is a self-intersecting sheet, then generalized sheet inflation can be used[8, 9]. However this method seems cannot locally generate the sheet, when depth control is required in many situations. In this paper, we present a method to locally generate a self-intersecting sheet

by extending the generalized sheet inflation algorithm. Since sheets that intersect itself once are the most common, we currently only deal with this kind of self-intersecting sheets. The specific steps are as follows:

1. Determine a polyline $poly_a$ on the interface with one self-intersecting point A based on the algorithm of finding polyline for chord inflation described previously. Then we find another polyline $poly_b$ with one self-intersecting point B on the side face within the required depth. $poly_a$ and $poly_b$ are connected, shown in Fig.9(a) as blue lines.
2. Use path finding algorithms like A* to find a path from A to B, all edges of which have valences no less than 4, as the red lines in Fig.9(a).
3. Retrieve 4 sequences of faces along the path, shown in Fig.10(c) as yellow faces.
4. Get two sets of hexahedra based on the 4 sequences and polyline on the interface. Then do some optimization on these two hexahedra sets according to edge valences and number of hexahedra.
5. Perform pillowing on the two sets respectively, shown in Fig.10(d).
6. Then make a column inflation between the two newly generated sheets, creating a self-intersecting sheet, shown in Fig.10(e).

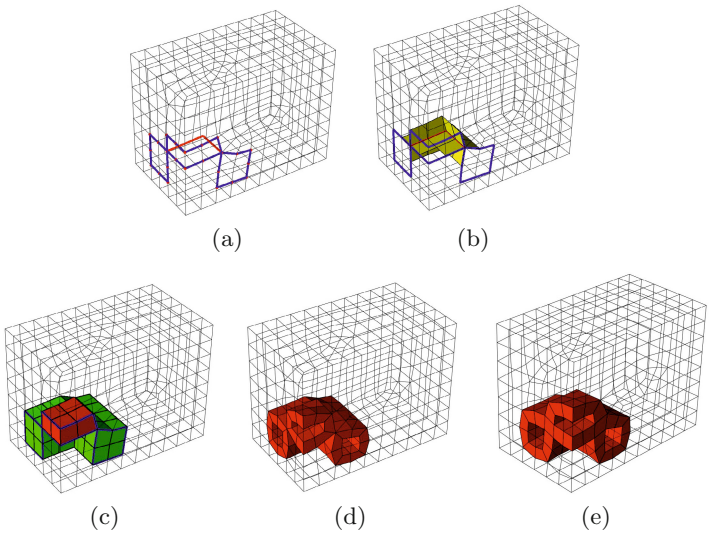


Fig. 10 Localized inflation of self-intersecting sheet: (a)Find self-intersecting polylines and the shortest path between two intersecting points; (b)Four sequences of faces along the path; (c)Two hexahedra sets; (d)Pillowing results of two hexahedra sets; (e)Self-intersecting sheet is created by a column inflation

3.4 Sheet Extraction Based on Mesh Quality Prediction

Another strategy of dealing unmatched partition chord or partition chord set is extracting it from the interface using sheet extraction. One problem with sheet extraction is how to get it executed locally, i.e. meets the depth control requirement. For example, given a partition chord set *pcs* in Fig.11(a), associated sheets S_{pcs} cannot be extracted together locally. To solve this problem the sheet extraction is executed in three steps in the previous work: firstly, an assistant sheet S_{assist} at the required depth which intersects with S_{pcs} in Fig.11(c) needs to be determined. Secondly, columns at the intersections between S_{assist} and S_{pcs} are collapsed to get a localized sheet S'_{assist}.

Thirdly, this sheet is extracted to have the partition chord set extracted, as illustrated in Fig.11(e).

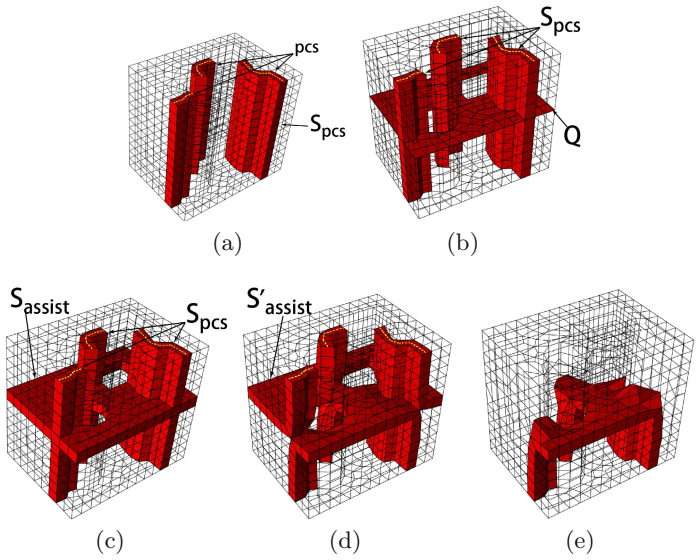


Fig. 11 Procedures of sheet extraction for partition chord set: (a)Sheets associated with the partition chord set; (b)Inflatable face set; (c)Assistant sheet; (d)Column collapse; (e)Extraction of sheet

In the first step mentioned above, there exist more than one possible assistant sheet satisfying the required depth, also it can be generated by inflating a quadrilateral face set Q from Fig.11(b) to Fig.11(c). Thus, the best assistant sheet which admits the highest quality of the resultant mesh needs to be found out. To obtain the best one, every possible assistant sheet is assessed. In the original mesh matching algorithm, in order to backtrack to the original

mesh model, the mesh model is copied to a new one to perform each assessment. This brings about many time-consuming mesh model copy operations, making the efficiency of the sheet extraction greatly decreased.

In order to improve the efficiency of the sheet extraction operation, we propose a method to predict the quality of the resultant mesh for each possible assistant sheet without conducting the mesh model copying and the real sheet extraction. Here the mesh quality is computed using the 3D topology score EEVS[1], thus edge valence (count of adjacent faces) and hexahedra number variation are taken into consideration. Before describing our prediction method, we first give the analysis of edges valence variation and hexahedra number variation.

Analysis of edges valence variation

During the process of sheet extraction, the valence variation for existing edges and new edges can be classified into following three situations:

1. In sheet inflation, edges of the quadrilateral face set Q are doubled, and new edges are inserted between Q and the new quadrilateral face set Q'. Then the new edge valences for the preserved edge $e_{preserve}$ and the newly inflated edge $e_{inflate}$ are the counts of the adjacent faces at their individual side of Q plus 1, and the edge valence for the newly inserted edge e_{insert} is the valence of node n_{insert} on Q, as illustrated in Fig.12(a).
2. In sheet extraction, Q and Q' are the two quadrilateral face sets of the sheet, and every pair of edges come from them individually is collapsed to one edge, and edges connecting Q and Q' are removed. As illustrated in Fig.12(b), the valence of edge collapsing from edge $e_{collapse}$ to $e_{preserve}$ is the count of their connected faces minus 3, and removed edge e_{remove} is not counted.
3. In column collapse, edges are collapsed or preserved with valence decreased by 1. Fig.12(c) illustrates a hexahedron of the column for collapse, and q_{AEFB} and q_{HGCD} are two of the quadrilateral faces of the column which are topologically perpendicular to the column. After column collapse, edges are collapsed, e.g. AE and AB are collapsed to $A'B'$, EH and BC are collapsed to $B'C'$. In addition, AD is preserved as $A'D'$, and its connected faces q_{AEHD} and q_{ABCD} are collapsed to $q_{A'B'C'D'}$, so its valence is decreased by 1.

Analysis of hexahedra number variation

The variation of hexahedra number is relatively simple and clear: in sheet inflation, the number is increased by the same number of quadrilateral faces in Q, then decreased by the number of hexahedral elements in the columns at the intersection of S_{assist} and S_{pcs}, and finally hexahedral elements in S'_{assist} are all removed in sheet extraction.

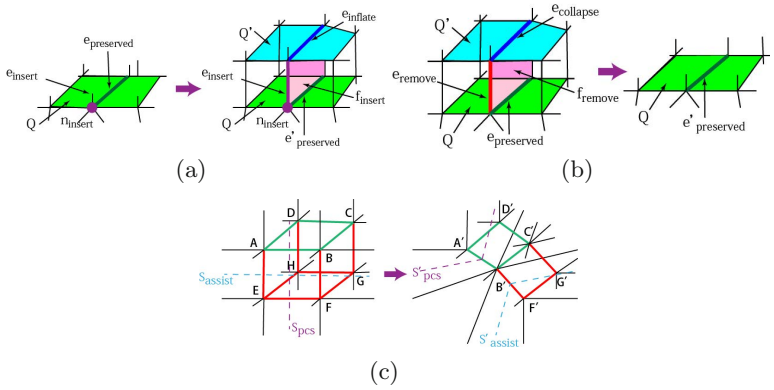


Fig. 12 Edge valence variation: (a)Edge valence variation during sheet inflation; (b)Edge valence variation during sheet extraction;(c)Edge valence variation during column collapse.

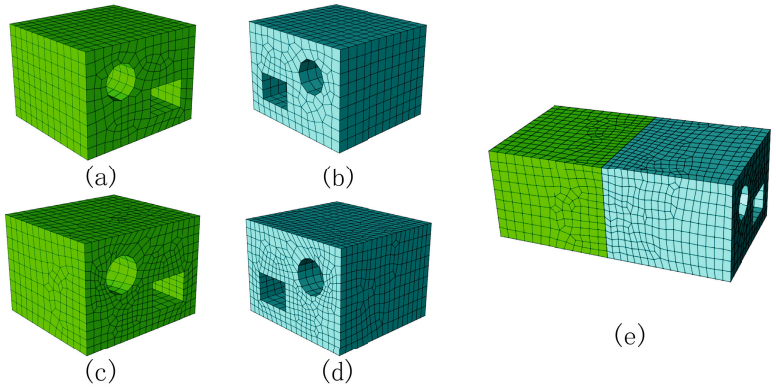


Fig. 13 Mesh matching example of assembly

Based on the above analysis, our method of mesh quality prediction is as follows: firstly, predict the resultant edge valences according the above analysis. Then the variation of hexahedra number is predicted. Finally, 3D topology score EEVS is predicted using edge valences and hexahedra number variation as input, in order to select the best assistant sheet efficiently.

4 Examples

We implement our algorithm in C++ on the platform of 32-bit Windows 7, Visual Studio 2010 and test the algorithm on several meshes. Fig.13 and Fig.14 are two examples.

Fig.13 shows an example of applying mesh matching on an assembly model. The two hexahedral meshes of two parts are generated by different size functions, so the interfaces between them are non-conforming. All the sheet operations applied to them during mesh matching are localized.

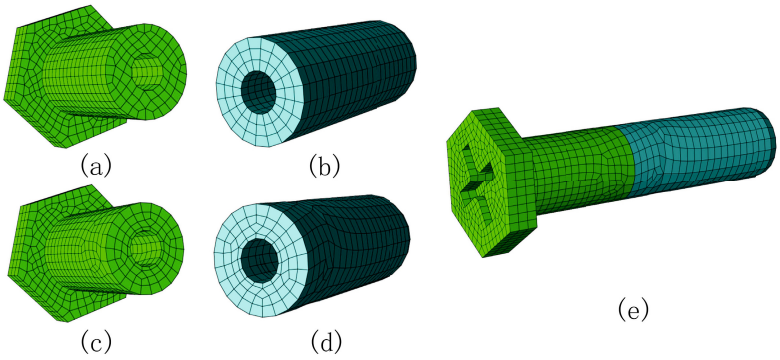


Fig. 14 Mesh matching example of the screw part

Fig.14 shows another kind of situation. The original screw part shown in Fig.14(e) is hard to be directed meshed. So it is cut into two parts, both of which can be meshed much more easily, shown in Fig.14(a) and Fig.14(b). Then we perform the mesh matching algorithm on the two meshes, and finally get the conforming interfaces.

The testing results show that mesh qualities degenerate due to sheet operations, however they are still within the acceptable tolerance for analysis. The sheet extraction with our prediction method is about 7 times faster than the original one.

5 Conclusion

In this paper, we have made some improvements to the original mesh matching algorithm to extend its ability and efficiency. By bringing out the new chord matching criteria, we can deal with complex interfaces with internal loops more effectively. Based on the new concept of partition chord set, we can now locally inflate or extract sheets associated with a partition chord set, sheets that cannot be locally inflated or extracted alone. For a sheet that intersects itself once, we present a solution to locally create a new self-intersecting sheet to match with it. And we also put forward an algorithm of predicting the quality of the resultant mesh model after sheet extraction without actually doing the operation, which greatly improves the efficiency of sheet extraction by avoiding frequently copying the whole mesh model.

Future research will focus on four aspects:

1. Find method to localize sheet extraction of self-intersecting sheets.

2. Make the algorithm be fully automatic. As described previously, when polylines of the chords in a partition chord set overlap, human interactions are needed to correct the polylines. A better optimization algorithm should be applied during searching the polylines in the future.
3. For a chord that is not a partition chord it can be grouped into different partition chord set, and different combinations of partition chord set have different impact on the quality of the resultant meshes. Thus we will study how to determine the optimal partition chord sets.
4. Extend the current algorithm so that it can deal with the case that the two interfaces only partly overlap.

References

1. Staten, M.L.: Sheet-based Generation and Modification of Unstructured Conforming All-hexahedral Finite Element Meshes. Dissertation, Carnegie Mellon University, Department of Civil and Environmental Engineering (2010)
2. Dohrmann, C.R., Key, S.W.: A Transition Element for Uniform Strain Hexahedral and Tetrahedral Finite Elements. International Journal for Numerical Methods in Engineering 44, 1933–1950 (1999)
3. Trujillo, D.M., Pappoff, C.G.: A General Thermal Contact Resistance Finite Element. Finite Elements in Analysis and Design 38(3), 263–276 (2002)
4. Dohrmann, C.R., Key, S.W., Heinstein, M.W.: Methods for Connecting Dissimilar Three-Dimentional Finite Element Meshes. International Journal for Numerical Methods in Engineering 47, 1057–1080 (2008)
5. Blacker, T.D.: The Cooper Tool. In: Proceedings of 5th International Meshing Roundtable, pp. 13–29 (1996)
6. Jankovich, S.R., Benzley, S.E., Shepherd, J.F., Mitchell, S.: The Graft Tool: an All-Hexahedral Transition Algorithm for Creating a Multi-Directional Swept Volume Mesh. In: Proceedings of 8th International Meshing Roundtable, pp. 387–392 (1999)
7. Miyoshi, K., Blacker, T.D.: Hexahedral Mesh Generation Using Multi-Axis Cooper Algorithm. In: Proceedings of 9th International Meshing Roundtable, pp. 89–97 (2000)
8. Staten, M.L., Shepherd, J.F., Shimada, K.: Mesh Matching - Creating Conforming Interfaces between Hexahedral Meshes. In: Proceedings of 17th International Meshing Roundtable, pp. 467–484 (2008)
9. Staten, M.L., Shepherd, J.F., Ledoux, F., Shimada, K.: Hexahedral Mesh Matching: Converting Non-conforming Hexahedral-to-hexahedral Interfaces into Conforming Interfaces. International Journal for Numerical Methods in Engineering 82, 1475–1509 (2010)
10. Lo, S.H.: Automatic Merging of Hexahedral Meshes. Finite Elements in Analysis and Design 55, 7–22 (2012)
11. Shepherd, J.F.: Topologic and Geometric Constraint-based Hexahedral Mesh Generation. Dissertation, The University of Utah, School of Computing (2007)
12. Ledoux, F., Shepherd, J.F.: Topological Modifications of Hexahedral Meshes via Sheet Operations: a Theoretical Study. Engineering with Computers 26, 433–447 (2010)

13. Merkley, K., Ernst, C.D., Shepherd, J.F., Borden, M.J.: Methods and Applications of Generalized Sheet Insertion for Hexahedral meshing. In: Proceedings of 16th International Meshing Roundtable, pp. 233–250 (2007)
14. Borden, M.J., Benzley, S.E., Shepherd, J.F.: Coarsening and Sheet Extraction for All-Hexahedral Meshes. In: Proceedings of 11th International Meshing Roundtable, pp. 147–152 (2002)
15. Staten, M.L., Benzley, S.E., Scott, M.A.: A Methodology for Quadrilateral Finite Element Mesh Coarsening. Engineering with Computers 24, 241–251 (2008)
16. Woodbury, A.C., Shepherd, J.F., Staten, M.L., Benzley, S.E.: Localized Coarsening of Conforming All-hexahedral Meshes. In: Proceedings of 17th International Meshing Roundtable, pp. 603–619 (2008)