SAND2007-3586C

Localized Coarsening of Conformal All-Quadrilateral Meshes

Mark Dewey[1], Steven Benzley[2], and Matt Staten[3]

[1] Brigham Young University markwdewey@gmail.com
[2] Brigham Young University seb@et.byu.edu
[3] Sandia National Labs mlstate@sandia.gov

Summary. Only a small amount of work has been done in the area of localized quadrilateral mesh coarsening. A previously presented algorithm that coarsens by forming a closed ”adaptive” chord around a region and subsequently removing the chord is reviewed in this paper and its implementation is discussed. This method, though robust, has proved to provide only modest amounts of coarsening in the desired region. A new algorithm to accomplish quadrilateral coarsening is presented. This new method is more general, proves to be faster and allows for more significant levels of coarsening. Examples of the two methods are provided.

1 Introduction

Despite exponentially increasing speed and memory in modern computers, finite element analysis continues to push to limits of computing power. To balance a need for faster computation times, an appropriate balance between element quantity and accurately representing the geometry of the model needs to be attained on a case by case basis. In many analysis situations the areas of interest are localized, so the need for a higher element count in specific regions may not transfer to other regions. A great deal of research has been put into finding out how to increase the number of elements in a specific region so that a mesh can have high accuracy in the area of interest while maintaining low computation time. Most common among this research is refinement algorithms which take an existing mesh and perform localized refinement, dividing the area of interest into smaller elements [1-7].

The research presented in this paper is a complement to mesh-refinement research. Rather than refining regions of interest, the algorithm presented in this paper coarsens a localized area, presumably an area not of interest. The ability to manipulate a mesh both ways should increase the rapidity with which analysis of different parts of the same mesh may be closely analyzed. On a particular model different boundary conditions may require that different areas of the model are finely meshed while other areas may be coarse. Using both refinement and coarsening tools the analyst would not have to re-mesh the model each time but could operate off of one original model, refining and coarsening different areas for use in analysis of different boundary conditions.

The algorithms presented in this paper relies heavily on the concept of the dual or spatial twist continuum as presented by P. Murdoch et al. [8]; Section 2 of the paper contains a review of the key principles from their research which apply to this algorithm. Section 3 reviews the implementation of a previously presented algorithm which creates a closed chord as a means of coarsening [9]. Section 4 discusses a more general algorithm that was inspired by insights gained during the implementation of the previous algorithm. Finally, Section 5 discusses areas of future research that these algorithms make immediately available.

2 Coarsening Process

The development of coarsening has been largely based on the concept of the dual of the mesh presented by Murdoch et al. [8]. The dual of the mesh is formed by connecting the centroids of each face in the mesh to their four neighbors. Each edge of a face corresponds to exactly one edge of the dual. The dual can be seen as the aggregate of *chords* which are formed by connecting a series of face centroids through the a series of edges opposite to each other as shown by the dashed lines in Figure 1. Murdoch points out several key properties of the chords of a valid conformal quad mesh. Reiterated here are those concepts most important to coarsening research. (1) A chord that begins on the boundary of the input region must terminate on the boundary of the input region. (Straight Chord) (2) A chord that does not begin on the boundary of the region must be a closed curve. (Circular Chord) (3) The centroid of each quad is crossed by exactly two chords. Borden has established that any chord (the analog of a hexahedral sheet) can be removed from the mesh [10]. An example of such a removal is shown in Figure 2. The removal of these quads results in coarsening the mesh around the chord.

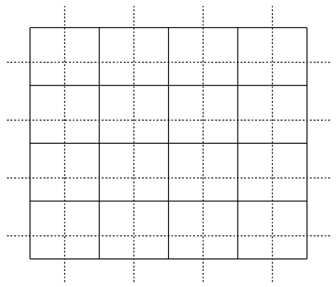


Fig. 1. Dual of the Mesh

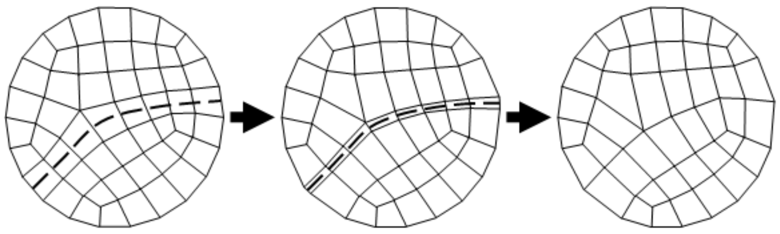
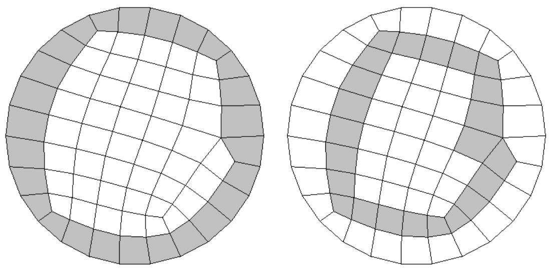


Fig. 2. Pulling a Chord from a Mesh

Two methods of localized refinement have been developed and are presented in this paper. The first method is to alter the structure of the mesh until there is a circular chord in the region of the mesh to be coarsened. The development of this method is completely described by Staten et al. [9] This circular chord is called an *adaptive chord*. Once the adaptive chord has been created it is pulled from the mesh. The removal of the chord is the 2-D analog of the extraction of a hex sheet [10]. A simple, natural example of an adaptive chord is shown in Figure 3(a). The second method is to identify a more general *lasso* which is a closed loop of quads that are not necessarily a single chord and remove them. An example of a lasso is shown in Figure 3(b). The lasso method is a generalized method that includes the concept of the adaptive chord method. The adaptive chord method begins with a lasso and performs operations (described in Section 3) at the intersections of the lasso's chords. Once either a lasso or an adaptive chord have been *pulled*, or removed, the mesh must be *stitched*, or reconnected, back together. Despite the simplicity of this concept at first, a large number of difficult cases develop that create degenerate elements. These cases must be addressed. Further, the coarsening process naturally results in a lower quality mesh. The specific methods of stitching the mesh back together and applying measures to improve the quality of the remaining elements are addressed in this paper.



(a) Adaptive Chord (b) Lasso Chord Set

Fig. 3. Lasso versus Adaptive Chord

3 Adaptive Chord Method of Localized Coarsening

A complete description of the adaptive chord method is described by Staten et al. [9]. A summary of the method is presented here to show the relationship of this method with the improved method presented in Section 4. The basic idea of the adaptive chord method is to coarsen the mesh by removing a chord. However, the removal of a chord will alter the mesh in areas remote from the region to be coarsened; this drawback can be eliminated by finding or making a circular chord in a specified region [7]. In implementing an algorithm to take advantage of this insight, it is necessary to identify a region to be coarsened and then decide which chords should be removed to perform the coarsening. These chords are then combined into one chord by the use of localized chord operations and the newly generated circular chord can be removed. Research into artificially forming an adaptive chord in preparation for localized coarsening has so far made use of four simple chord operations: doublet-insertion, face-close, face-open, and edge-swap. While there are many other operations that could accomplish the manipulation of chords, these operations have very little impact on the surrounding mesh and can be applied to any valid mesh regardless of connectivity. A fifth operation, not implemented in the original version of this algorithm, is presented here called template-insertion. In all cases except the face-open operation, the goal of the chord operations is to take two previously intersecting chords and make them into one continuous chord. Essentially this causes the chords to turn a tight corner. In the case of the face-open operation the goal is to take two previously parallel chords and combine them into one. In all cases the application of these operations lowers the local shape quality of the mesh. However, in most cases where coarsening would be considered, the region to be coarsened is of less interest to the analysis and errors due to lesser (but still acceptable) shape quality in that region would not cause problems for the overall analysis.

3.1 Doublet-Insertion

A doublet insertion divides a quad by placing a new node in the center of a quad and creating two new edges from two nodes of the quad opposite each other. The mesh is no longer valid in this location, but the doublet insertion is only a temporary state (the two new quads will later be pulled out as part of the adaptive chord). The operation and effect on the chords is shown in Figure 4.

As noted in Section 2, each face has two chords which intersect at its centroid. In a valid mesh "chords may cross each other multiple times, but such crossings may not be consecutive" [8]. Doublet insertion violates this rule in order to manipulate the direction of these chords; however, as noted above, the degenerate elements will be removed later in the coarsening process. As shown in Figure 4 one chord enters the original quad from the left changes course and leaves through the top rather than through the right side of the

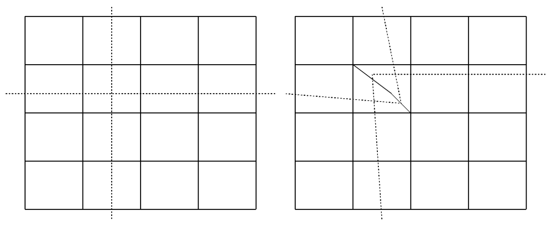

(a) Chord Intersection (b) Doublet-Insertion

Fig. 4. Doublet-Insertion Operation to Combine Chords

new quads. Similarly, a second chord now exits through the right rather than exiting through the top.

3.2 Face-Close

The face close operation is accomplished by bringing two nodes opposite each other on a quad together. Each pair of edges associated with the other two nodes merge and the original quad vanishes. The operation is shown in Figure 5.

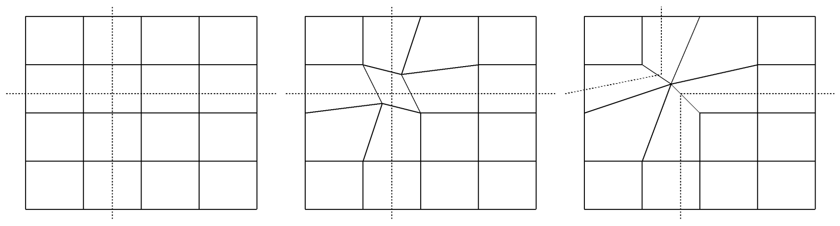
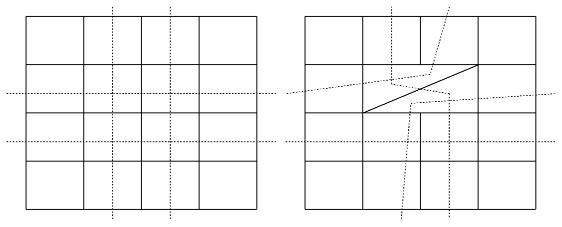

(a) Chord Intersection (b) Face Collapsing (c) Face Collapsed

Fig. 5. Face-Collapse Results Results

As with doublet insertion the two chords that entered into the original quad are modified: one chord coming from the left now goes up and the second chord coming from the bottom now goes right. Once again, chords initially going ”straight” are now caused to turn a sharp corner. Note also that two chords which previously intersected no longer intersect. We note here that the face-close operation and the doublet-insertion operation result in the same topology after coarsening. Thus, in the case of coarsening, they are equivalent operations.

3.3 Edge-Swap

The edge swap operation begins by selecting two adjacent quads. The edge separating the quads is deleted and replaced by another edge extending between two different nodes. The new edge between the two quads may be placed in two different locations. One seems to spin the original edge clockwise while the other spins it counter-clockwise. The clockwise version is shown in Figure 6.



(a) Chord Intersection (b) Edge-Swap

Fig. 6. Edge-Swap Results

The edge swap operation manipulates three chords, two are made to go around a tight corner, as in the doublet insertion although not quite as tight. The other chord simply shifts courses. While the counter-clockwise and clockwise edge swaps do not differ in mechanics, the use of one or the other changes which chords are affected. Consequentially, if two specific chords are being targeted, different quads may be used for each operation (the quad at their intersection will always be the same, but the other quad will change). The difference of which other quad is used can make a difference in the quality of the resulting mesh. Thus, both operations need to be implemented and used in appropriate situations. Edge-swap operations have proven advantageous over face-close and doublet-insertion methods because the final mesh has higher quality elements. However, they have a slight disadvantage in that they reduce the net number of elements removed by the coarsening process by one every time the operation is used.

3.4 Face-Open

The face open operation begins with a node. Two edges that are not part of the same quad are selected (this means that a three valent node may not be selected). Each of these edges and the original node is duplicated and then the two nodes are separated to form a quad (see Figure 7). Each of the two nodes takes some of the edges from the original node with it.

Examination of the chords around the original node reveals that the two chords associated with the edges selected above now cross rather than running

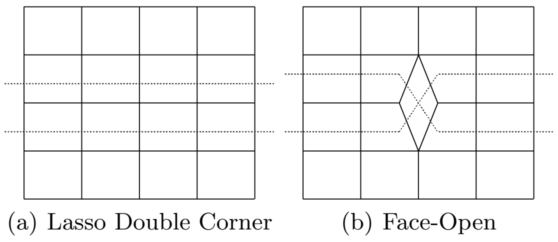

(a) Lasso Double Corner (b) Face-Open

Fig. 7. Face-Open Results

parallel past the node. Unlike the above three operations, this operation does not result in turning a corner. Instead, it turns two consecutive corners. Like the edge-swap operation, this operation adds one element to the mesh to accomplish the quality improvement. However, this single addition resolves two corner problems simultaneously. Further, the neglect of this method causes very bad quality issues, which none of the other methods address as effectively. The face-open method is specifically used to resolve *jagged* regions of the bounding chords: places where the set of chords forming the boundary of the region (or in Section 4, the lasso) have two or more consecutive corners.

3.5 Template-Insertion

A recently developed method for merging chords is the insertion of a template in the element just inside the corner of the bounding chords. Figure 8 shows an example of the template insertion. This operation is more complicated to implement than the other methods and has a higher computational overhead. However, it guarantees valid mesh elements where other methods would create degenerate elements. Unfortunately, this operation adds two elements back into the mesh as opposed to one or none.

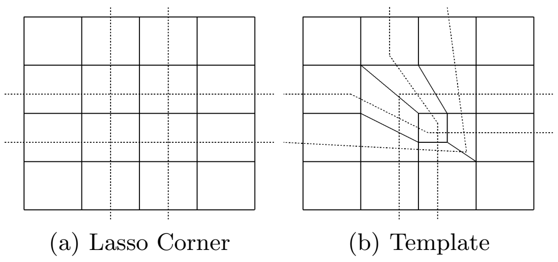

(a) Lasso Corner (b) Template

Fig. 8. Template-Insertion Results

3.6 Adaptive Chord Example

The following is an example of the adaptive chord removal algorithm. The region selected for coarsening is part of a circular surface with a paved mesh. The region shown in Figure 9(a) shows the lasso that is to be pulled. Each corner of the region is resolved so that the bounding chords are formed into an adaptive chord. Most of the corners are dealt with by edge-swap operations. However, there is a jagged pair of corners in the top right portion of the figure that is resolved by the use of a face-open operation. Edge-swap operations were utilized as the preferred method of combining chords because they tended to create higher quality meshes after coarsening took place. Figure 9(b) shows the fully formed adaptive chord. The adaptive chord is pulled in Figure 9(c) and the final, smoothed mesh is shown in Figure 9(d).

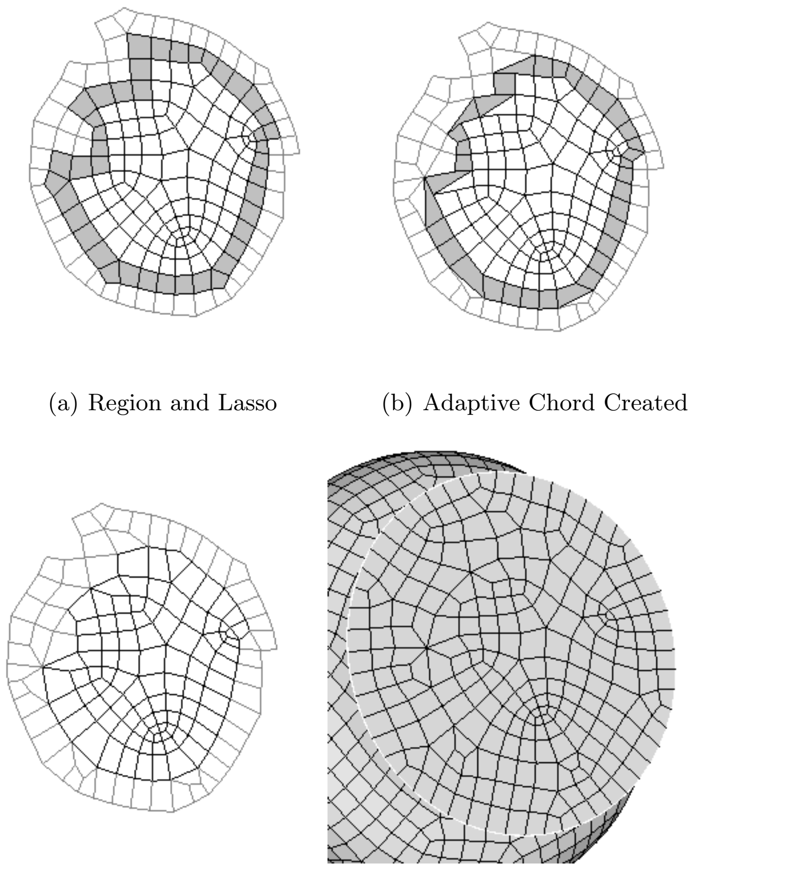
The adaptive chord algorithm is a robust means of coarsening. However, note how little the mesh is coarsened by pulling a single adaptive chord. The process can easily be put into an iterative loop until sufficient coarsening is accomplished; however, the degradation of the quality of the mesh tends to accelerate with each pass.

4 Lasso Method of Localized Coarsening

The initial goal of adaptive chord coarsening was to remove only the outermost boundary layer of quads. However, we will show that the removal of several, non-adjacent sets of chords provides large amounts of coarsening with minimal shape quality degradation. Furthermore, we will show that it is unnecessary to form a circular adaptive chord.

With the development of the new lasso coarsening method, several new goals took shape. Firstly, the degree of coarsening on the interior of the specified region needed to be increased. As can be seen in Figure 9, the coarsening only effects the quads near where the adaptive chord has been removed. In cases where the region was very large, the quads near the center of the region are not coarsened at all. Secondly, the issue of element quality and the creation of valid meshes was addressed. Low quality and invalid quads were major issues with adaptive chord coarsening. The new algorithm, presented here, resolves these issues.

The lasso coarsening algorithm consists of 3 basic generalized steps. (1) Select the lassos to pull. This selection process includes: (a) Identifying a set of lassos that may be removed from the mesh and (b) choosing a feasible subset of those lassos to accomplish the desired level of coarsening (preferably while maximizing element quality). (2) Apply quality-improving operations to the lasso corners (optional). (3) Pull the Lassos. Pulling the lassos is accomplished by (a) removing the quads forming the lassos from the mesh and (b) stitching the mesh back together.



(a) Region and Lasso (b) Adaptive Chord Created

(c) Adaptive Chord Pulled (d) Smoothed Mesh

Fig. 9. Template-Insertion Results

While the method is simple, there are several complicated cases and difficulties that arise due to the complex nature of meshes. Research into this algorithm has only begun; Step 2 has not been implemented in the more generalized approach. However, the adaptive chord example above can be viewed in light of these steps. First, the lasso to be removed is selected, in this case simply the outermost lasso. Second, the quality improvement operations are applied, in this case primarily edge-swap operations at the corners and one face-open operation at a jagged part of the lasso. Third, the adaptive chord is removed. If the algorithm is not able to remove enough quads the first time it is run, multiple iterations of the same process can be executed until the mesh is of desired coarseness.

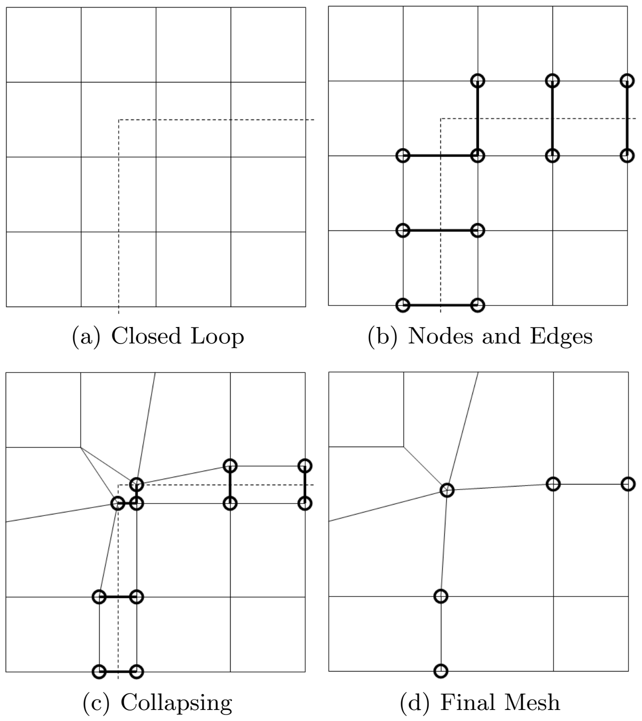
4.1 Basic Algorithm for Removing a Loop of Quads

The simplest method for stitching the mesh back together into a conformal mesh is to collapse the edges that have been removed from the mesh. Most of the nodes at the boundary of the region where the quads have been removed have a single corresponding node on the opposite side. These nodes were previously attached to each other by an edge that formed the boundary between two adjacent quads in the now removed closed loop. The simplest method of stitching the mesh back together is by simply merging all of the nodes that were so attached to each other previously. Figure 10 shows how this process works at a corner of a closed loop. The loop is indicated by the dashed line (Figure 10(a)), the nodes at the boundary are subsequently identified by a bold circle and the edges connecting them are also bolded (Figure 10(b)). Each group of nodes, including the group with three nodes attached by two edges, are drawn together toward their centroids (Figure 10(c)). The resulting mesh (Figure 10(d)) is identical to the the mesh that would be produced by using a doublet-insertion or a face-close operation at the corner element in the lasso. This process is simply repeated throughout the bounding loop selected. The method of pulling a loop by matching groups of nodes by their edge-connectivity is essentially identical to the method of pulling a hexahedral sheet proposed by Borden et al. [10] with the exception of what happens at the corner element. The recognition that the algorithm did not need to change at the corner of the lasso was the first piece of inspiration that led to the development of the lasso coarsening process.

4.2 Chord Operations as Quality Operations

While this most basic stitching process is very fast and simple to program, it has a tendency to create poor or degenerate element shape quality. Recognizing that it is unnecessary to form a closed adaptive chord raises a question: why does the use of edge-swap operations tend to create better element quality in the final, smoothed mesh? Shepherd seems to answer this question in his research on webcutting [11]. Essentially, the nodes on either side of a continuous chord have a one-to-one ratio. If the mesh is simply removed as shown in Figure 10 there is not a one-to-one relationship. This causes the elements collapsing into the void to become cramped around the vanishing corner elements. Jagged edges are a worst-case scenario of this problem. Several corners in immediate succession could cause dozens of nodes to collapse into a single node causing inverted and extremely poor quality elements in the jagged region.

In light of this new insight, the edge-swap, face-open, and template-insertion operations take on a new meaning, they are simply ways of restoring the one-to-one ratio of nodes on either side of the removed lasso. In this capacity they serve to increase the quality of the final mesh. To avoid the jagged scenario described above, the face-open operation must be implemented to


(a) Closed Loop (b) Nodes and Edges

(c) Collapsing (d) Final Mesh

Fig. 10. Basic Coarsening Process

break apart most of the corners. To increase mesh quality at sharp corners the edge-swap or template insertion methods may be employed. Comparing the use of these two operations, it can be noted that while the template-insertion operation shows potential to render the best quality meshes, it also reinserts the largest number of quads back into the mesh, diminishing the overall coarsening if it is used consistently.

The quality of the mesh after the removal of the lasso is a fairly easy metric to project. The use of these operations can be made conditional on poor element quality, thus maximizing the amount of coarsening taking place while maintaining acceptable element quality. In other words, the chord operations can be viewed as quality improving operations that can be applied as preprocessing coarsening operations. Projecting the quality of the mesh is not strictly necessary in the use of the lasso algorithm; however, the use of it in deciding where to apply quality operations and in the optimization of loop selection described below make the implementation of an algorithm to determine future element quality very important.

4.3 Optimization of Loop Selection

One of the most difficult parts of automating the coarsening process for a general, specified region is choosing which lassos to remove. Naturally, the user could specify one or more lassos to remove. However, there are automated methods for choosing a lasso or set of lassos to remove from a general region specified by the user. To spread the coarsening evenly throughout the region, several lassos may be selected to be removed simultaneously. Lasso selection is restricted in two ways. Lassos being pulled simultaneously cannot run parallel and adjacent to each other: such lassos pulled simultaneously create high valency nodes and low quality elements. This includes the situation of a lasso running parallel and adjacent to itself. Second, lassos must contain only one closed loop: a three way intersection does not appear to have a localized stitching method. Simultaneous lassos can, however, cross or even have a single corner quad in common. Beyond user selection, there are several methods by which a set of potential lassos can be formed. In addition to developing an algorithm to identify potential lassos, an algorithm for selecting the best set of lassos to pull simultaneously from a list of available lassos has been developed. These methods are discussed below.

Forming a Set of Available Lassos

Given a large, arbitrary region, the simplest method to form lassos is to grab concentric loops of the outside quads at the boundary of the region. The code used in the example below simply finds all of the quads on the border of the selected region and puts them into a data structure that represents the lasso. These quads are then temporarily marked as being outside the selected region and the next lasso is formed from the new set of boundary quads. Often, specific quads are not selected in order to ensure that a lasso can be pulled effectively. For example, border quads that run two wide would be excluded from a given lasso because lassos that run parallel and adjacent to each other cannot be pulled simultaneously.

Alternately, random nodes or quads might be selected within the region, the lassos formed could be the set of quads one layer of quads distant from the central node or quad. The set of all of these small lassos would form an appropriate base for the optimization algorithm that follows. Naturally, the two sets just described (the concentric set and the set of all small lassos) are not mutually exclusive, both sets could be included to maximize the potential options. Naturally, the more lassos that are in the set to be selected from, the longer the run-time of the algorithm.

Choosing from the Available Lassos

A set of lassos to be pulled simultaneously is feasible if it meets three conditions: first, it must not pull more quads than a given goal number of quads to be removed plus some tolerance; second, the projected quality of the mesh when the lasso is pulled meets a given minimum shape quality standard; and third, no two of the lassos in the set run adjacent to each other as mentioned above. The projected minimum quality of the mesh and the number of quads pulled should be determined beforehand for each lasso so that the algorithm can quickly check the minimum quality metric and sum of quads to be pulled of a given lasso set. The algorithm then loops through the list of available lassos until it finds one that is feasible. This lasso is placed in a current list of lassos. It then recursively loops through each available lasso again checking whether the set of lassos in the current list would be feasible if a given lasso were added to the list. Each time it finds a lasso that can be added it recurses again, checking if more lassos can be added. Once it finds a feasible set of lassos to which no lasso can be added it takes note of how many quads would be pulled and the lowest quality projected by any of the lassos. This list of lassos is stored out, along with the number of quads pulled and the quality projected. The recursion continues, other potential sets of lassos are compared to the current best list, possibly replacing the best list with a new list of lassos.

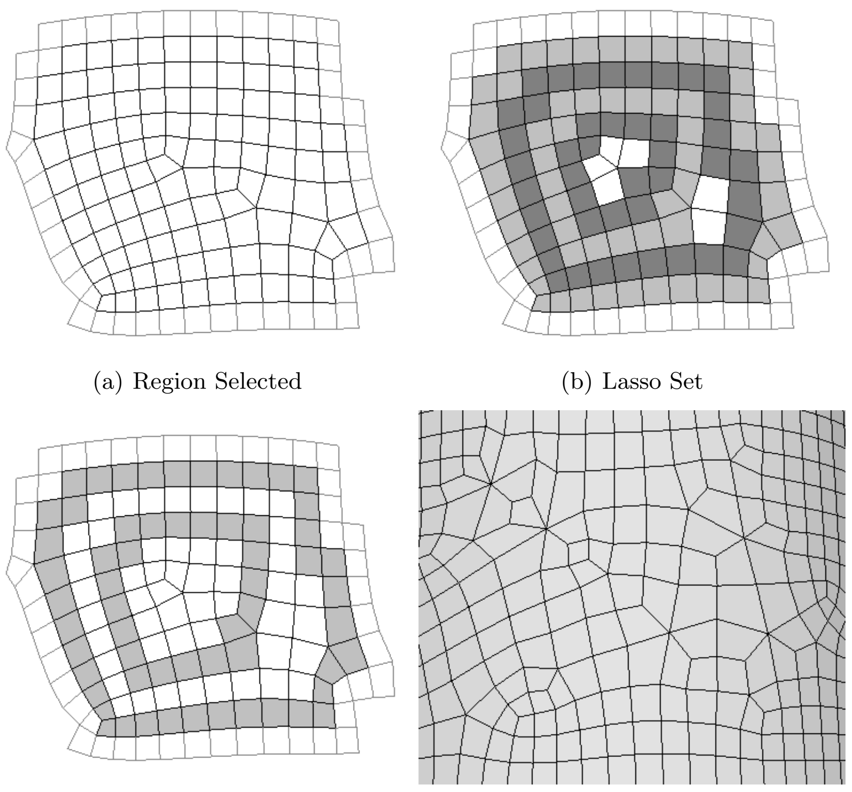
In determining if a new feasible set is better, several conditions must be met. First, if a list is found that has a number of quads in it greater than the goal number of quads minus a tolerance, this list is selected as the best list. Second, if there are multiple lists that respond to this description then the list with the highest minimum quality is selected. Third, if there are no lists that have sufficient quads then the feasible list that removes the most quads is selected. In this case a second iteration of the entire lasso algorithm would have to be performed after the removal of the current best lasso.

This algorithm mirrors the branch and bound algorithm utilized in many design optimization programs [12]. Every time it tries to add a lasso that would make the current list less viable than the best list, it does not have to proceed down the recursion. For example, if a list has been found that has sufficient quads in it, then any list considered later that includes a lasso with lower quality than the current best list can be immediately eliminated and the recursion can stop. Further, if a new lasso would make the current list unfeasible then further recursions including that lasso do not need to be attempted. Thus, not every subset of lassos needs to be considered. Further, not every feasible set of lassos needs to be considered, only those that are potentially better than the most recent best lasso set encountered. The ability to eliminate many of the options by these methods makes this algorithm orders of magnitude faster than an exhaustive search of all possible combinations of lassos. Only a small fraction of the combinatorial space need be considered.

4.4 Lasso Example

The following example shows how the lasso method is applied to coarsening a paved mesh. Figure 11 shows the decision making process of the algorithm.

Figure 11(a) shows the region selected by the user with the addition of one layer of quads around the outside to show connectivity to the rest of the mesh. Figure 11(b) shows the lasso set developed by the concentric lasso method. The white quads have been ruled untenable as part of one of the concentric lassos. Figure 11(c) shows the lassos that were selected to accomplish the desired level of coarsening. Finally, Figure 11(d) shows the mesh after the lassos have been pulled and the mesh has been smoothed. Fifty-nine out of the 109 quads in the original region were removed from the mesh.



(a) Region Selected

(b) Lasso Set

(c) Lassos Chosen

(d) Coarsened Mesh

Fig. 11. Lasso example 1

A close examination of the mesh reveals that apart from a few specific problem areas, the quality of the mesh has stayed well within normal bounds. There are a few six-valent nodes immediately noticeable. These nodes all correspond with places where the lasso is jagged, making more than one turn at a time. This problem can be resolved by the use of a face-open operation at those corners, reducing the valency of these nodes. In fact, the use of the face open operation in a structured mesh will result in a structured mesh around the area of the operation.

5 Future Research

The field of quad coarsening is rich with possibilities and potential for further improvement. Further research is immediately warranted in developing local quality improving operations. Adaptation of this process to hexahedral coarsening demands attention now that the groundwork of quadrilateral coarsening has been lain. Finally, developing this algorithm to facilitate time-dependent analysis should be explored. While a great deal of attention could be paid to coarsening in general, these three objectives seem to be readily obtainable from the current state of the research.

5.1 Template Insertion

While the use of template insertion is promising, it has not yet been implemented and tested. The implementation of this localized operation is the clear next step in the research of quad coarsening. Additionally, other localized operations ought to be pursued and considered to give the coarsening process a larger tool-set. Another topic of research related to the insertion of templates is a reliable quality projection scheme. In choosing which lassos to remove, projected quality and quantity are dynamically compared in the decision of which lassos to remove. If a reliable method of projecting the quality of the mesh taking into consideration the use of quality operations were developed, then the selection of the best set of lassos could be markedly improved.

5.2 Hexahedral Coarsening

Having developed a promising solution to the quad-coarsening problem, the results discovered should be applied to hex coarsening. Benzley et al have already given an initial explanation of how to accomplish hex coarsening [7]. Hexahedral coarsening is not a trivial expansion of the algorithms already developed. While some of the same principles seem to apply, there is no guarantee that every closed volume of hexes will be adaptable in the same way.

5.3 Reversible Coarsening

Time-dependent analysis relies heavily on refining and un-refining meshes to closely model specific dynamic effects. Because of the constant change in the location of crucial analysis points the reversibility of the refinement processes in paramount to the usefulness of the refinement process during time-dependent analysis. If coarsening is to be used in this field, a similar ability to reverse the process must be shown. There is great potential available to reverse the coarsening involved in this approach. Each edge that was once two edges on the opposite sides of a quadrilateral could be re-expanded into a quadrilateral. The nodes on either end of that edge correspond to edges that were removed from the mesh. The nodes at the corners of the lasso can be face-opened into the corner quadrilateral. Templates and face-open quads inserted into the mesh can be removed. Each of these operations to reverse the coarsening algorithm needs research and development. However, it is likely that the development of reversible coarsening would be a straightforward, though not trivial, task.

6 Conclusion

The ability to automatically coarsen a mesh is now well within the grasp of meshing tools. Although its maturity and development for commercial markets is still a few years off, it shows great promise as a means of improving FEA run time while maintaining numerical accuracy. Localized quad coarsening has been shown to be a straightforward process of selecting a closed loop of quads, removing them from the mesh, and stitching the mesh back together. While there are a number of useful localized operations to improve the resulting quality of the mesh, the algorithms to use these tools are merely subsets of one generalized method. Further research in the field of quad coarsening, hex coarsening and reversible coarsening have been opened wide to research possibilities.

References

1. Tchon K, Hirsh C, Schneiders R (1997) Octree-Based Hexahedral Mesh Generator for Viscous Flow Simulations. 13th AIAA Computational Fluid Dynamics Conference, No. AIAA-97-1980, Snowmass, CO.
2. Marechal L (2001) A New Approach to Octree-Based Hexahedral Meshing. Proceedings 10th International Meshing Roundtable, Sandia National Laboratories, 209-221.
3. Schneiders R (1996) Refining Quadrilateral and Hexahedral Element Meshes. 5th International Conference on Numerical Grid Generation in Computational Field Simulations, Mississippi State University, 679-688.
4. Schneiders R (2000) Octree Based Hexahedral Mesh Generation. Int Journal Comput Geom Appl. 10, No. 4, 383-398.
5. Tchon K, Dompierre J and Camerero R (2002) Conformal Refinement of All-Quadrilateral and All-Hexahedral Meshes According to an Anisotropic Metrid. Proceedings 11th International Meshing Roundtable, Sandia National Laboratories, 231-242.

6. Tchon K, Dompierre J and Camarero R (2004) Automated Refinement of Conformal Quadrilateral and Hexahedral Meshes. Int Journal Numer Math Engng 59:1539-1462.
7. Benzley S, Harris N, Scott M, Borden M, and Owen S (2005) Conformal Refinement and Coarsening of Unstructured Hexahedral Meshes. Journal of Computing and Information Science Engineering, December 2005, Vol 5, 330-337.
8. Murdoch P, Benzley S, Blacker T, and Mitchell S (1997) The Spatial Twist Continuum: A Connectivity Based Method for Representing All Hexahedral Finite Element Meshes. Finite Element Analysis and Design, 28(2), 137-149.
9. Staten M, Benzley S and Scott M (2006) A Methodology for Quadrilateral Finite Element Mesh Coarsening. To appear in: Engineering with Computers, WCCM2006 Special Edition.
10. Borden M, Benzley S, Shepherd J (2002) Hexahedral Sheet Extraction. 11th International Meshing Roundtable, Sandia National Laboratories, September 2002 147-152.
11. Shepherd J (2007) Topological and Geometric Constraint-Based Hexahedral Mesh Generation. PhD Thesis, University of Utah, Utah, May 2007.
12. Land A and Doig A (1960) An Automatic Method of Solving Discrete Programming Problems. Econometrica, Vol. 28, No. 3 (Jul., 1960), pp. 497-520.