2012-12-06

Two-Refinement by Pillowing for Structured Hexahedral Meshes

J. Bruce Malone
Brigham Young University - Provo

Two-Refinement by Pillowing for Structured Hexahedral Meshes

J. Bruce Malone

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Paul W. Richards, Chair
Steven J. Owen
Steven E. Benzley
Michael A. Scott

Department of Civil and Environmental Engineering

Brigham Young University

December 2012

ABSTRACT

Two-Refinement by Pillowing for Structured
Hexahedral Meshes

J. Bruce Malone
Department of Civil and Environmental Engineering, BYU
Master of Science

A number of methods for adaptation of existing all-hexahedral grids by localized refinement have been developed; however, none ideally fit all refinement needs. This thesis presents the structure to a method of two-refinement developed for conformal, structured, all-hexahedral grids that offers flexibility beyond what has been offered to date. The method is fundamentally based on pillowing pairs of sheets of hexes. This thesis also suggests an implementation of the method, shows the results of examples refined using it and compares these results to results from implementing three-refinement on the same examples.

Keywords: hexahedral meshing, pillowing, two refinement, eight refinement, finite element analysis preprocessing, adaptive meshing, mesh improvement

ACKNOWLEDGMENTS

I thank my professors Dr. Steven J. Owen and Dr. Steven E. Benzley for their support in getting this thesis together. Dr. Owen helped me develop its topic and oversee its content. Dr. Paul W. Richards and Dr. Michael A. Scott, as well as Dr. Fernando S. Fonseca and the secretaries of the department, also demonstrated care toward my research and the presentation of it. I thank Sandia National Laboratories for funding this research and providing Cubit so that I could implement the algorithm this thesis proposes. Finally, I thank my lovely wife Emily for her unfailing encouragement as I put this thesis together.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 MOTIVATION

1.1 Adaptation of Hexahedral Meshes for Finite Element Analysis

A number of physical problems in physics and engineering may be modelled mathematically as partial differential equations. The solutions to these governing equations are the continuous functions from which the derivatives are taken [1]. In general, closed-form solutions to these equations cannot be determined. Instead, numerical techniques are used. They return approximate evaluations of the continuous-function solutions of these governing equations on a finite domain of points. These evaluations are the solutions to the numerical approach to solving differential equations [2].

One numerical method, finite element analysis, was initially developed for use by civil engineers. Finite element analysis relies on a physical problem having been modelled mathematically with a set of governing differential equations; modelled physically with geometric entities; given boundary conditions and material properties; and discretized into a mesh of some finite number of smaller regions—elements—that give the method its name. These elements form the points at which the solutions to the finite element analysis are taken [3] [4].

While many such elements exist, this thesis will focus on hexahedrons (also referred herein as hexes) and their two-dimensional counterparts, quadrilaterals (also referred herein as quads). Hexes are box-shaped elements formed by at least eight nodes at the corners (higher-order hexes

contain more nodes on the faces, edges and interior, but this distinction is beyond the scope of this thesis).

1.1.1 Advantages and Challenges of Uniformly Hexahedral Meshes

Hexahedral mesh elements are general three-dimensional elements that may be chosen in order to produce numerically reliable results from a finite element analysis with relatively low computational cost. For example, a finite element analysis over a mesh consisting of hexahedral mesh elements is expected to yield numerically acceptable results with significantly fewer elements than would be required with a mesh composed of the more popular pyramid-shaped tetrahedral elements [5] [6]. This fewer number of elements is desirable because potentially, it significantly lessens the processing time during finite element analysis; computational demand from finite element analysis increases significantly with the number of elements in the mesh [7].

However, creation of hexahedral meshes is much less automatic than creation of tetrahedral meshes [5] [8]. Methods for creating and adapting hexahedral meshes are heuristic in nature [9]; hence developing all-encompassing solutions is difficult. The development of two-refinement for this thesis was likewise a heuristic process.

Mesh element uniformity is desirable. Exploring all reasons for this desirability is beyond the scope of this thesis. One reason, however, is that not all finite-element solvers can operate on mixed-element meshes [10]. Moreover, despite controversy over whether hexahedral-based meshes are always superior to their tetrahedral-based counterparts [11], the numerical results of finite element analyses over some real physical phenomena are not reliable when performed over a mesh with tetrahedrons due to a phenomenon called tet-locking [8], as explained in [12]. As hexes

form a viable alternative [13], this thesis adds to the quest to develop useful tools for generating and adapting meshes composed entirely of hexahedrons.

1.1.2 Finite Element Analysis Requires a Quality Mesh

In order for hexahedral meshes to give the finite element method reliable results, each hexahedron must be well-shaped. For example, a hexahedron formed by its eight corner nodes into the shape of a tetrahedron will lose its computational advantage of being a hexahedron. On the other hand, perfectly shaped hexahedrons (i.e. cubes) usually cannot be confined adequately to some geometric shape. Most elements will form some compromise between perfectly-shaped hexahedrons and poorly shaped ones, leaning as far toward the perfectly-shaped ones as possible.

For example, consider an overlay grid of quads over a circle. An overlay grid is an independent mesh of perfectly-shaped elements that is placed to overlay some geometry (Figure 1-2(a) on page 11) and then adapted to conform to the geometry (Figure 1-2(b) and Figure 1-2(c)—note that an extra layer of quads was added in order to improve quality) [14]. In the middle of the circle, the quads can be perfect squares. But on the circle boundary, nodes must be moved in order to conform to the boundary of the circle, forming more poorly shaped elements. Smoothing then can be used to adjust all quads, slightly reducing the quality of those nearest the center but overall increasing the quality of the quads.

Numerous metrics, or measures, exist for determining the quality of shape of a hexahedron. The metric used by this thesis will be the minimum scaled Jacobian determinant, or simply the

scaled Jacobian. The following algorithm demonstrates how to calculate the scaled Jacobian given a hexahedron as input.

$S \leftarrow$ set of size 8 of scalars in R^1;
for *Each node, n, of the 8 hexahedron nodes* **do**
  $V \leftarrow$ set of size three of vectors, V_1, V_2, V_3, in R^3;
  for *Each edge, e of the 3 hexahedron edges sharing n* **do**
    Normalize (as a unit vector) the vector from n to its opposite node with respect to e, and add this normalized vector to V;
  end
  Add $|V_1, V_2, V_3|$ to S; *[This is the determinant of the scaled Jacobian]*
end
return min S;

Figure 1-1: Computing the scaled Jacobian

The values of the scaled Jacobian range from -1 to 1. However, an accepted minimum value for this metric is typically no less than 0.2; the closer it is to 1, the better the hexahedron [15] [16]. The motivation behind the scaled Jacobian metric is that during finite element analysis "a necessary and sufficient condition for a valid finite element solution to exist is that the Jacobian determinant ... be non-negative at every point in the parametric space" [17]; the shape of the element is an indicator of how well that parametric space is modelled. A motive of the algorithm proposed by this thesis is to keep the scaled Jacobian relatively close to 1.

1.1.3 Localized Refinement Helps Adapt a Mesh

In general, an initial mesh will not initially be adequate for reliable numerical results during a finite element analysis. For example, some regions may critically need accurate approximations compared to other regions; the required mesh density may be unknown before analysis. In fact,

the error analysis resulting from finite element analysis will reveal critical areas [18] such as those of high physical gradient [19]—for example, a high rate of change of stresses—in a region of a physical model. Such areas may need more elements in order for a finite element analysis to yield acceptable values.

Likewise, geometric features such as those that are small or having high curvature may require a finer mesh in order to be captured [20] [21]. A coarse mesh may not yield enough resolution through nodes to capture these boundary features [22], similar to how a low-resolution photograph may not correctly capture complex details. Also, a coarse hexahedron may need to be warped significantly in order to match boundary details. Subdivision of hexes into smaller hexes may be ideal in order to capture geometry.

The mesh may be adapted—modified—to have more hexes in the critical areas where more are needed (refinement) or fewer where they are not needed (coarsening). In general, the goal of adaptation is to produce a mesh with as few elements as possible but yielding an adequate solution from a finite element analysis in order to optimize computational efficiency during finite element analysis.

One may simply adapt via refinement the entire mesh to the density of elements needed at critical locations. However, doing so may significantly yet unnecessarily increase the computation time of the analysis due to the increased size of the mesh.

Instead, only the critical regions of the mesh may be refined locally [23], leaving the rest of the mesh alone. This type of refinement is called localized refinement, and it is preferable to refining the entire mesh because it yields fewer elements. Fewer elements may significantly decrease computational time during finite element analysis [10].

For example, consider in the overlay-grid example in Section 1.1.2 a relatively coarse overlay grid placed over a circle. In order to help conform to the circle's boundary, the conforming quads could be adapted through refinement to give a higher resolution of conforming quads [14].

1.2 Three Refinement May Over-refine

One commonly implemented technique for localized refinement is 3-refinement. After hexes requiring refinement are selected, this technique divides each hex by three in each of the three Cardinal directions, forming a total of 27 new hexes.

In addition, an outside layer of hexes, referred to herein as the transition zone, also subdivides hexes. This subdivision occurs in a way to maintain conformal connectivity between the hexes selected for refinement and the rest of the mesh. *Conformal* in this context means that each face of each hex shares exactly one face of another hex unless on a geometric boundary. Otherwise, the mesh connectivity can introduce inaccuracies in the finite element solution [6] (assuming the finite element software could accept such a mesh; many require conformal meshes [10]). The transition zone is the challenge to implementing conformal, localized refinement.

Three refinement is robust and well-established, and it can be used to refine unstructured meshes. Its disadvantage lies in the dramatic 1-to-27 change in hex density. When this large amount of adaptation is required, three refinement is a working solution. However, when significantly less adaptation is required, the analyst is left with too many hexes—similar to when the entire mesh is refined. Also, Edgel [24] shows that some of the commonly-used three-refinement templates developed by Schneiders contain elements lacking in quality with respect to the scaled Jacobian.

The aim of two-refinement is to provide meshers and analysts another tool for localized mesh refinement—one that will not over-refine when only a slight refinement is required and one that will provide relatively high-quality elements. With both two- and three-refinement algorithms at hand, an analyst or mesher has greater control over the localized refinement levels of a hexahedral mesh.

1.3 Two Refinement Produces Fewer Hexes than Three Refinement

Two refinement divides a hex into 8 sub-hexes instead of 27, or less than a third as many as three-refinement. From this stand point, refining via two-refinement can help keep the computational cost resulting from refinement, during finite element analysis, to a minimum.

We acknowledge previous work done on conformal two refinement for hexahedral meshes. The juxtaposition of challenges and advantages given by two-refinement have made for research by a number of meshing researchers; but none have yet found an all-encompassing solution. While this thesis also does not give such a solution, it addresses a need that has not previously been resolved.

Edgel [24] describes a method for two-refinement using templates to modify structured, hexahedral meshes. While his method may successfully and efficiently be applied to any region of selected hexes that is convex and carries an even number of hexes in all three directions, his solution to concavities requires large templates containing many, low-quality elements. His method uses two refinement zones.

Ebeida et al [25] also developed twin techniques based on two-refinement templates and a framework developed by Schneiders. These techniques boast the capacity to do two-refinement on unstructured hexahedral meshes with high-quality hexes, previously not accomplished. Moreover,

through use of an octree, they allow multiple levels of localized two-refinement to be applied. This control over the amount of refinement in a localized region will result in an effective subdivision per direction of 2^{l+1} where l is the level of localized refinement ($l = 1$ corresponds to halving each hex in each direction, for eight hexes). However, these flexible methods are intended for heavy refinement of an initially coarse mesh: before any localized refinement can be applied, the entire mesh must be two-refined. Therefore, if the techniques are applied at level $l = 1$ to some localized region, the hexes of that region have effectively been subdivided into $\left(2^{1+1}\right)^3 = 64$ hexes instead of 8. For applications where 3-refinement over-refines, these techniques clearly are not the solution.

While the capacity to refine unstructured meshes offers tremendous flexibility with an algorithm, the capacity to refine structured meshes only is not without application. For example, meshes generated from an overlay grid are inherently structured. This work has been designed to target the overlay grid method as described in [26].

1.3.1 The Pairing Rule Binds the Size of Transition Zones in Two Refinement

Two refinement methods refine mesh elements in pairs in order to maintain connectivity in the transition zones [24]. We refer to this requirement as the pairing rule. While the pairing rule does not bind the shape of the region requested for refinement, it does limit how few hexes may be used as part of a transition zone. The pairing rule also has limited the scope of this thesis to structured grids.

By observation, for any uniform refinement method (i.e. a method that subdivides each hex selected for refinement in one way), at a minimum, each modified face of the hexes selected for refinement—the uniform refinement zone or $\Omega_{\mathbf{URZ}}$—on the surface formed by these hexes must

touch another hex that is not in $\Omega_{\mathbf{URZ}}$, unless it is at a geometric boundary. Let us call these transition zone hexes not in $\Omega_{\mathbf{URZ}}$ that share faces with $\Omega_{\mathbf{URZ}}$, $\Omega_{\mathbf{TZ}}^{\min}$.

If the mesh elements selected for two-refinement do not satisfy the pairing rule, meaning that sufficient pairs of hexes cannot be identified from $\Omega_{\mathbf{URZ}} \cup \Omega_{\mathbf{TZ}}^{\min}$ (or, equivalently, that $\Omega_{\mathbf{TZ}}^{\min}$ is insufficient for maintaining conformal, all-hexahedral connectivity with the rest of the mesh), additional hex elements outside of $\Omega_{\mathbf{URZ}} \cup \Omega_{\mathbf{TZ}}^{\min}$ will be required.

For example, consider a two-dimensional case with quads instead of hexes (and edges instead of faces). If a region selected for refinement perfectly satisfies the pairing rule, such as a 2×2 region of quads, as given in Figure 1-3 on page 11, then the minimum number of transition zone quads required is the number of outside edges (analogous to faces of hexes), or the size of $\Omega_{\mathbf{TZ}}^{\min}$. However, in Figure 1-4, not only is one transition zone quad necessary for each edge of $\Omega_{\mathbf{URZ}}$, but an additional 2 quads are necessary in the transition zone—a transition zone requiring quads besides those in $\Omega_{\mathbf{TZ}}^{\min}$. Hence, 6 quads are necessary for refinement. Again, consider Figure 1-5. Besides a transition quad necessary for each of the 6 outside edges, two more are needed in order to satisfy the pairing rule, for a total of 8 quads—the same number necessary for the 2×2 case!

Though in two dimensions two-refinement is simple (even if the $\Omega_{\mathbf{URZ}}$ does not naturally fit the pairing rule, e.g. $n \times n$ quads where n is even), in three dimensions, regions that do not naturally satisfy the pairing rule (e.g. $n \times n \times n$ hexes where n is even) do not render themselves easily solvable. Connectivities through the third dimension encounter obstacles that inhibit an approach that simply extends the method applied in two dimensions into three dimensions in all cases.

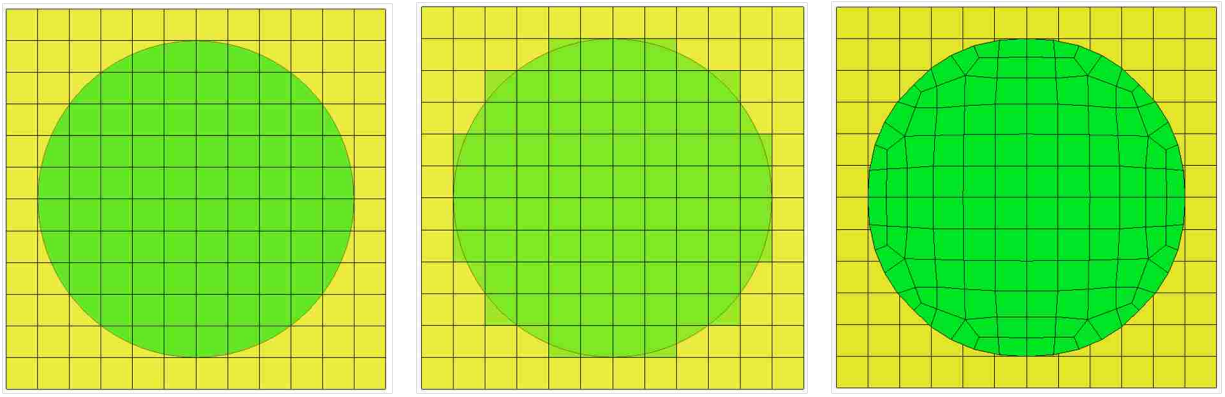
These obstacles stem directly from the fact that the newly created nodes at the outside faces of the original hexes have others nodes they must connect to in the third dimension. In reviewing Figures 1-3, 1-4 and 1-5, imagine looking straight-on at a region of hexes. On top of these, immediately outside of the page, is another transition zone of hexes. If the quads drawn in these figures are the outside faces of a $\Omega_{\mathbf{URZ}}$ that extends into the page, no known method, in general, can simply connect the nodes to the transition layer out of the page and maintain a conformal, all-hexahedral mesh.

1.3.2 What this Thesis Resolves to Contribute

This thesis presents the framework for a class of implementations of conformal two-refinement on arbitrary structured, all-hexahedral meshes. Moreover, it demonstrates and provides details about a recommended implementation that is heuristically demonstrated to yield mesh elements of desirable quality.

The basic framework potentially could be implemented more efficiently using templates. The framework and its implementation were created as a general approach with such further research in mind. However, such research is beyond the scope of this document.

As given in this document, the basic framework consists of a refinement technique, pillowing, serially applied three times. While this repetition may make the presented algorithm unattractive from a computational standpoint, the implementation given was designed so that much of the remaining computational work could, potentially, be programmed to take advantage of parallel processing. Also, the technique presented may offer flexibility in level of refinement; the refinement technique could be only done once or repeated twice instead of thrice, for example, resulting in a less-refined mesh.



(a) Geometry to mesh with overlay grid

(b) Selecting overlay grid quads

(c) Overlay grid conforming to geometry

Figure 1-2: Demonstration of overlay-grid

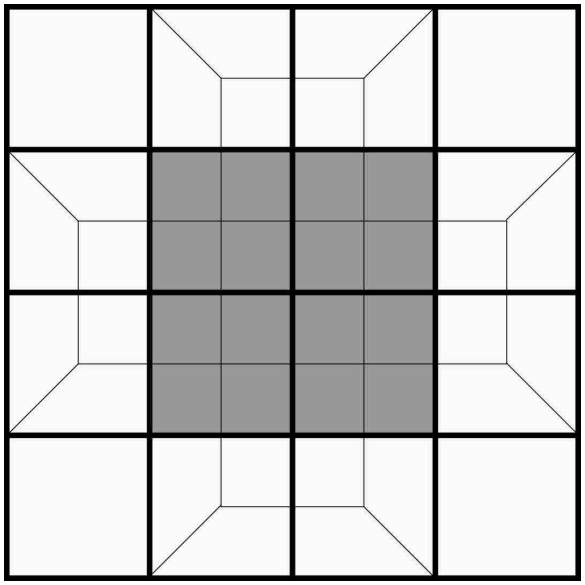


Figure 1-3: A 2×2 region of quads selected for two refinement (shaded). The bold lines indicate the original grid.

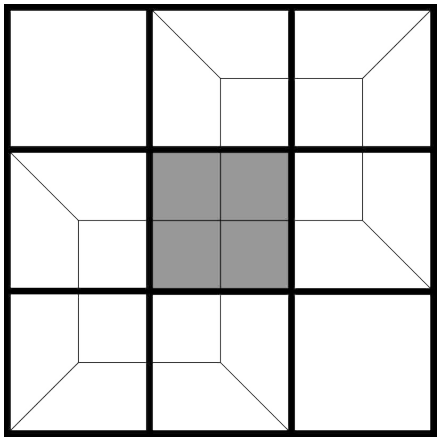


Figure 1-4: A 1×1 region of quads selected for two refinement (shaded). The bold lines indicate the original grid.

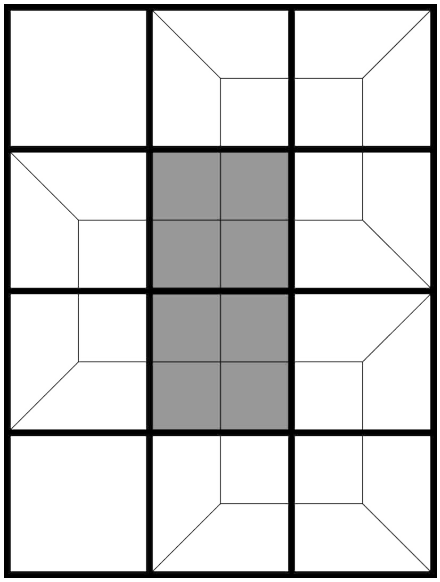


Figure 1-5: A 2×1 region of quads selected for two refinement (shaded). The bold lines indicate the original grid.

2 OVERVIEW OF ALGORITHM

This chapter demonstrates graphically the structure for doing two-refinement that this thesis suggests. It does so through step-by-step figures on an example showing the recommended implementation for the two-refinement algorithm. Figures in this chapter were created using the meshing software Cubit developed by Sandia National Laboratories.

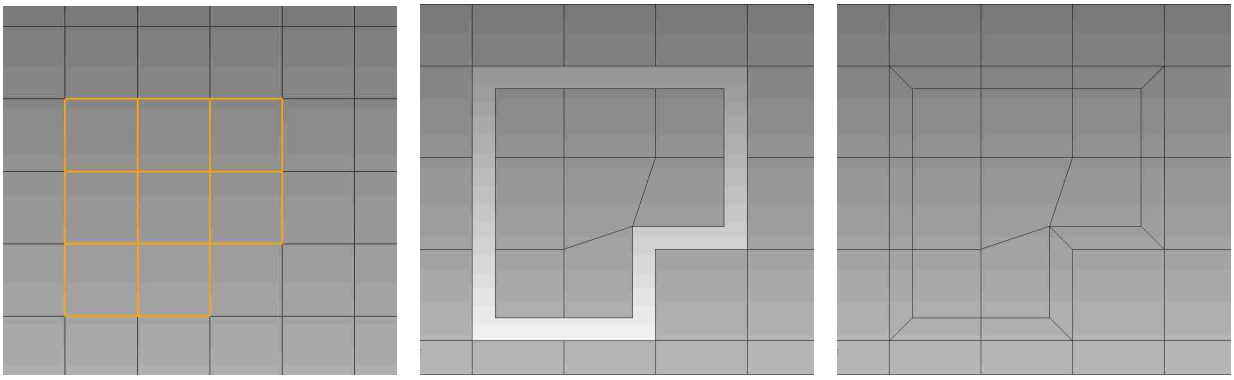
2.1 Explanation of Pillowing

Pillowing simply is inserting a sheet, or pillow, of hexes between selected hexes and their neighboring hexes. This process is effectively accomplished by shrinking the selected hexes—known as a shrink-set—and connecting the nodes pulled inward to the nodes in their original positions, as shown in Figure 2-1 on the following page (for simplicity, quads are used instead of hexes).

A nice property of pillowing is that it conformally retains element uniformity—it will pillow a quad mesh with quads and a hex mesh with hexes. A topology operation, pillowing combined with smoothing generally produces elements of sufficient quality [27].

2.2 Demonstration in Two Dimensions

We first present the two-refinement procedure using a two-dimensional example in order to facilitate understanding of the algorithm in three dimensions.



(a) Select elements for refinement (b) Shrink elements (c) Connect nodes pulled inward to original nodes

Figure 2-1: Process of pillowing

2.2.1 Pillowing in the X_1 Direction

We will begin with a few quads selected for refinement from a domain of quads:

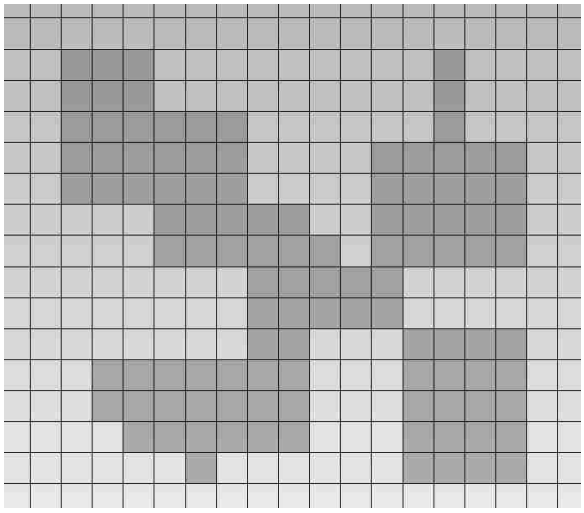


Figure 2-2: Two-dimensional example of quads to be refined. The shaded region is the quads selected for refinement or the Uniform Refinement Zone.

Figure 2-3 on the next page shows how a quality two-dimensional two-refining algorithm such as the one in Cubit would refine this region. It utilized 95 quads in the transition zone (12

more than minimally necessary by a generic uniform refinement algorithm as described in Section 1.3.1 on page 8) and produced 279 new quads within the transition zones. The method that will be presented is less efficient in terms of how many quads it produces, but it will work in the three-dimensional case, unlike the method used to produce Figure 2-3.

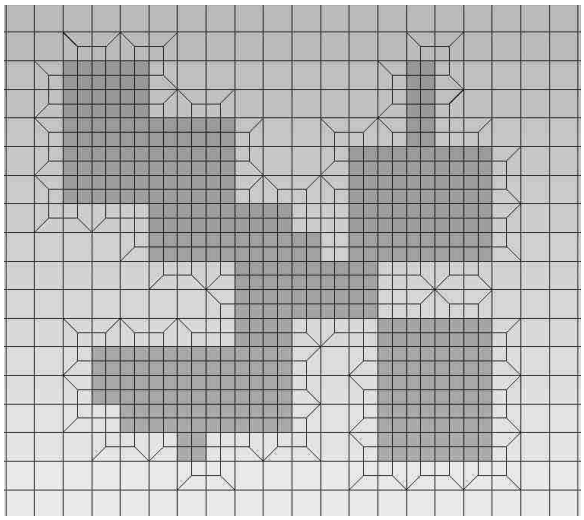


Figure 2-3: Two-dimensional example two-refined by Cubit

Let us define the X_1 direction as going from left to right and the X_2 direction from top to bottom. Going from the left of the region to the right, we will select two columns of quads at a time and pillow them. Figure 2-4 on the next page shows the first pair of columns selected for refinement based on the $\Omega_{\mathbf{URZ}}$, or quads selected for refinement. Notice that the region selected for pillowing extends beyond the $\Omega_{\mathbf{URZ}}$ by one quad in the positive and negative X_2 direction.

Figure 2-5 on page 17 shows the second pair of columns of quads selected. Note that in order again to extend the region selected for pillowing beyond the $\Omega_{\mathbf{URZ}}$ by one quad in the positive and negative X_2 directions, a rectangular, or convex, set of quads was chosen despite the

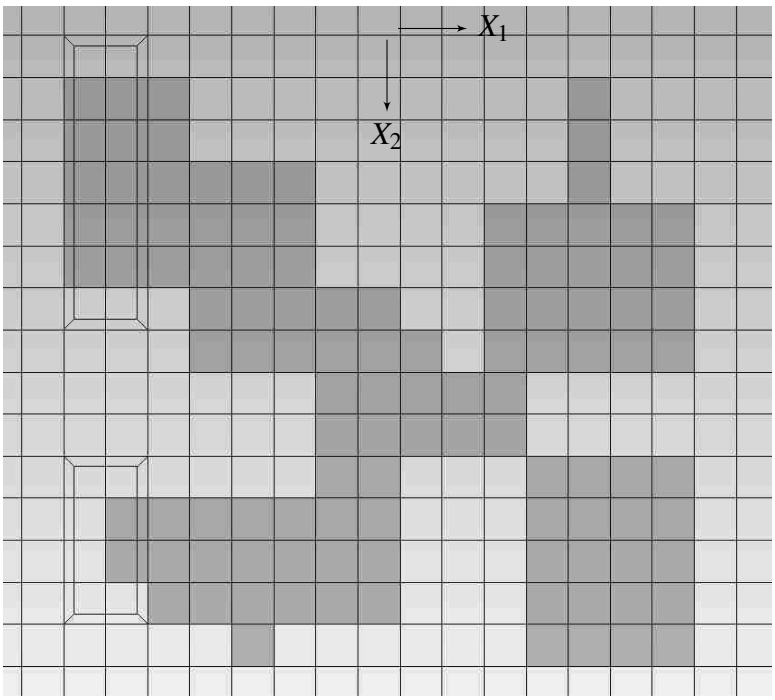


Figure 2-4: First pair of columns of quads selected in the X_1 direction

concavities of the $\Omega_{\mathbf{URZ}}$ in these two columns. Figure 2-6 on the next page shows pillowing in the X_1 direction completed.

2.2.2 Pillowing in the X_2 Direction

We now proceed with the same procedure in the X_2 direction, as follows.

Figure 2-7 on page 18 shows the first pair of columns pillowed in the X_2 direction, and Figure 2-8 shows the first two pairs of columns pillowed in the X_2 direction. Note that only all $\Omega_{\mathbf{URZ}}$ quads modified by pillowing in the X_1 direction have been added to the set of $\Omega_{\mathbf{URZ}}$ quads. Figure 2-9 on page 19 shows the finished product.

Note on pillowing through a transition zone Note the zoom-in of Figures 2-10 and 2-11 on page 20. In order to pillow some regions in the X_2 direction, quads resulting from a pillow in the

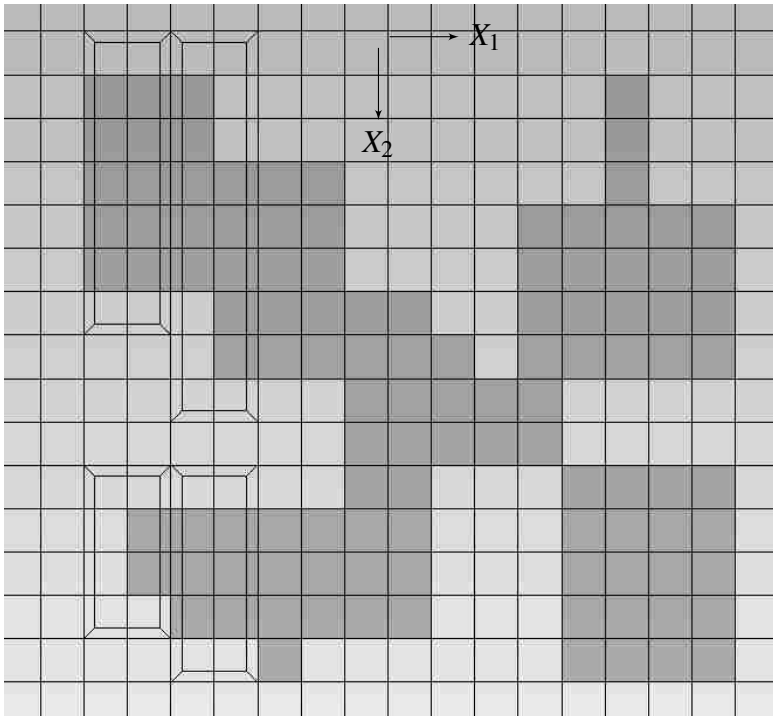


Figure 2-5: Two pairs of columns refined in the X_1 direction

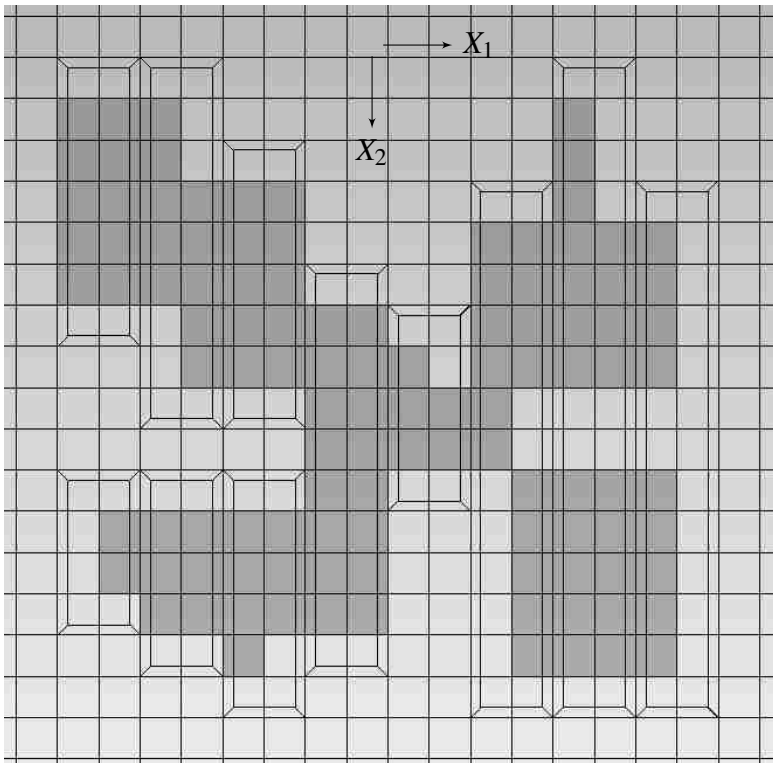


Figure 2-6: All columns refined in the X_1 direction

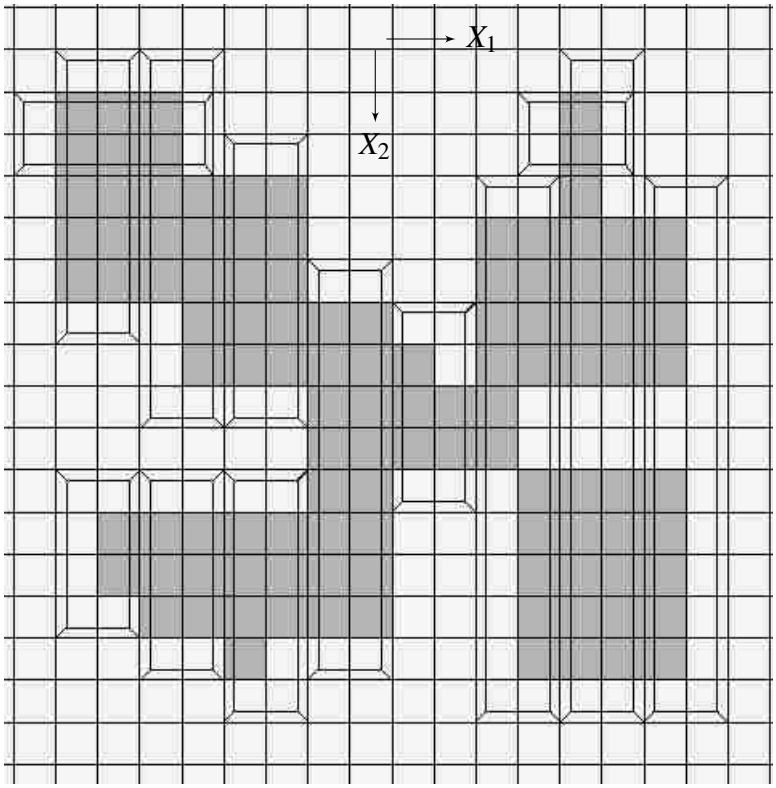


Figure 2-7: First pair of columns refined in the X_2 direction

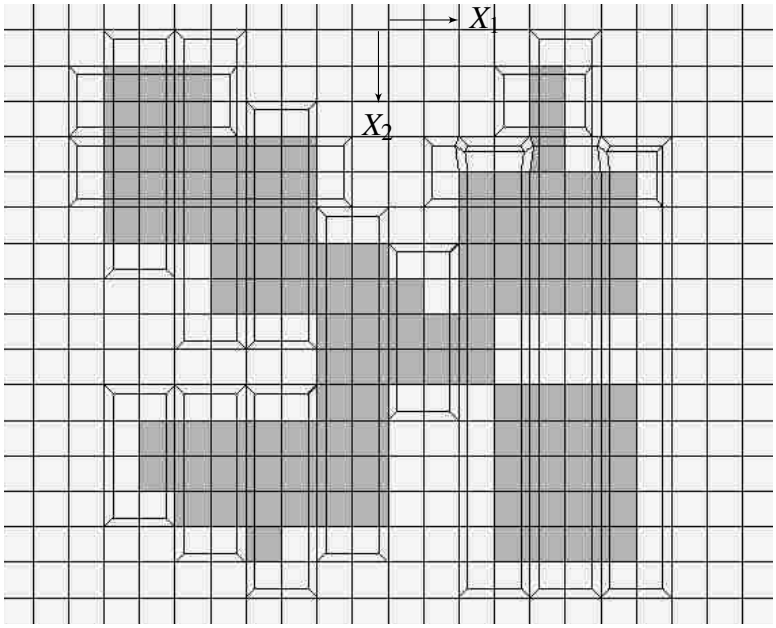


Figure 2-8: Two pairs of columns refined in the X_2 direction

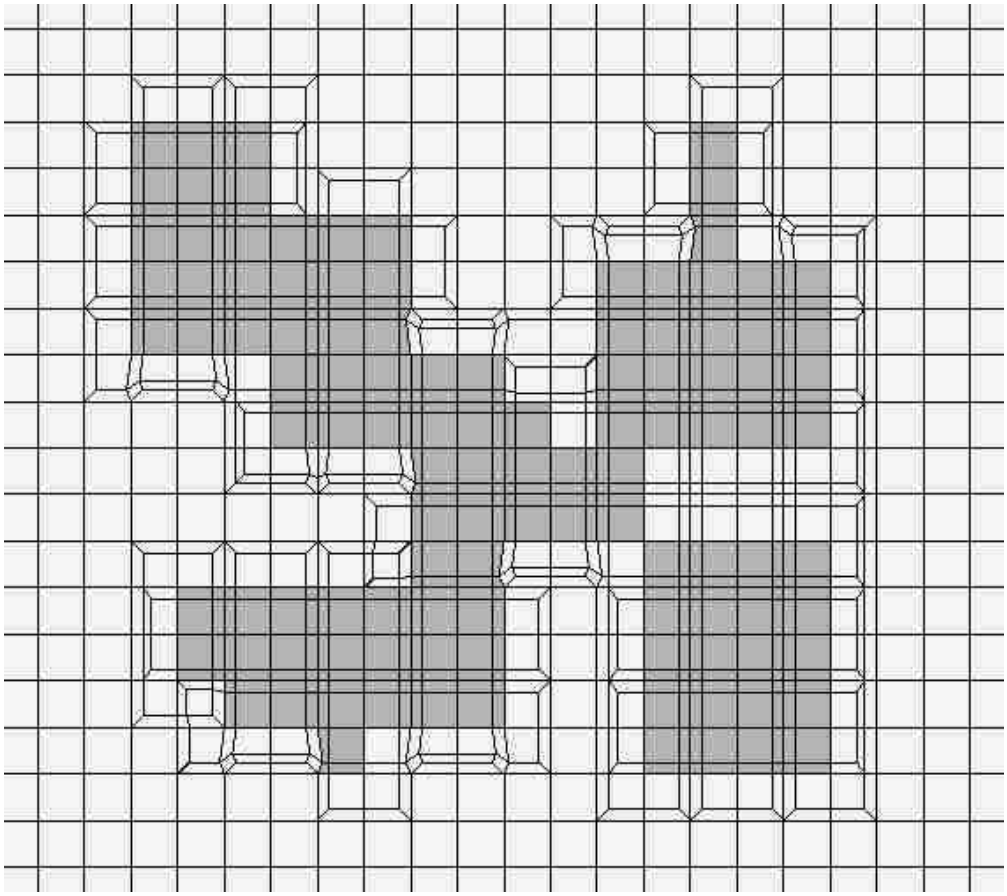


Figure 2-9: All pairs of columns refined in the X_1 and X_2 directions

X_1 direction had to be included in the shrink set. Ironically, this necessity produced quads with more elements than those being refined; that is, some of the original quads ended up with 5 or even 7—nearly double the target refinement—instead of 4.

Note on number of hexes refined

In counting quads from the final product of the current example, one will count 111 quads selected and 99 original quads used for the transition zone, or four more than were used in producing Figure 2-3. Other algorithms for 2-D meshes certainly could accomplish this refinement

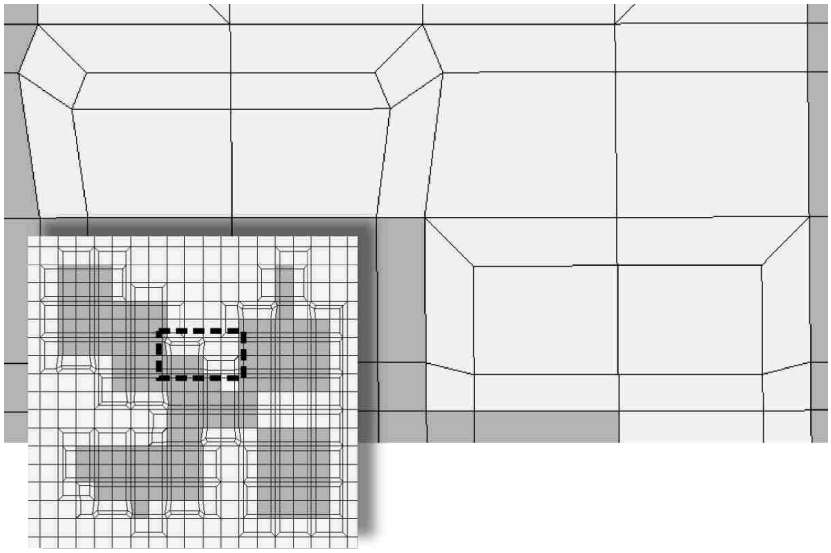


Figure 2-10: Heavy refinement zone in 2-D with 5 quads

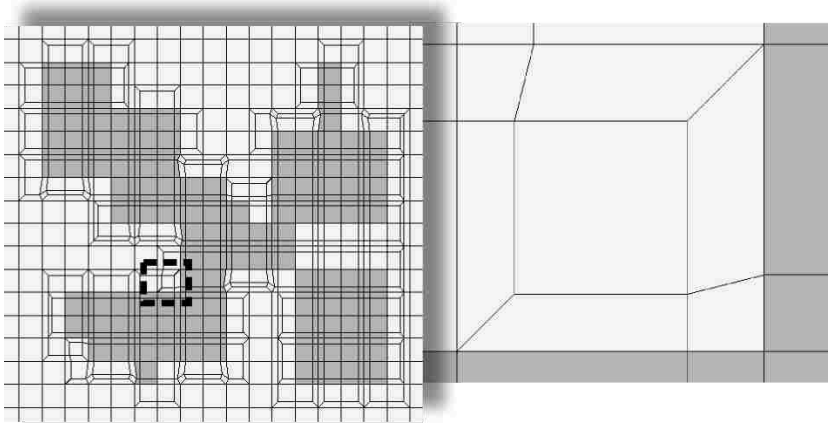


Figure 2-11: Heavy refinement zone in 2-D with 7 quads

without the use of the heavy transition-zone hexes; but this algorithm is designed for three dimensions.

In this example, of the 99 original transition zone hexes, 2 (2%) became 2 hexes; 40 (41%) became 3 hexes; 35 (35%) became 4 hexes; 20 (20%) became 5 hexes; and 2 (2%) became 7 hexes, for a total of 279 new hexes in the transition zone, or 78 more (26.4% more) than created for Figure 2-3. In other words, just over a fifth of the original transition zone hexes became heavy transition

zone regions. Other examples would produce different results, but this example suggests that the algorithm used is prone to producing heavy transition zone regions as a minority compared to the rest of the transition zones. Though such transitions are not ideal, they are unavoidable using the pillowing approach, and as-is, they are acceptable.

2.3 Implementing in Three Directions

Let us consider Figure 2-9 on page 19 such that the X_3 direction is into the page. Shown is a cross-section taken from a domain of hexes that has been refined in two directions. Selecting the sheet of hexes in the current view and an identical sheet immediately behind it, then pillowing the appropriate $\Omega_{\mathbf{URZ}}$, would result in Figure 2-12 on the following page, which is then shown in Figure 2-13 on page 23 after smoothing in order to visually indicate the quality of the resulting grid in terms of both element shape and gradient of element sizes. Note that the selection criteria differs from that used in the X_2 direction: all new hexes from previous refinement are included as part of $\Omega_{\mathbf{URZ}}$, whereas in the X_2 direction only the hexes replacing the original $\Omega_{\mathbf{URZ}}$ were included in $\Omega_{\mathbf{URZ}}$.

If, instead, Figure 2-3 on page 15 were viewed as a cross-section taken from a domain of hexes, the hex connectivity with hexes in the third dimension would not be conformal. However, using the pillowing approach shown in the previous section resulting in Figure 2-9, the hexahedral mesh remains conformal and is ready for refinement in the third direction.

2.3.1 Pillowing in the X_1 Direction

Using a simple geometry (the green volume in the domain of hexes of Figure 2-14 on page 24), we will begin at the origin (least-positive hex, of the entire domain, with respect to the

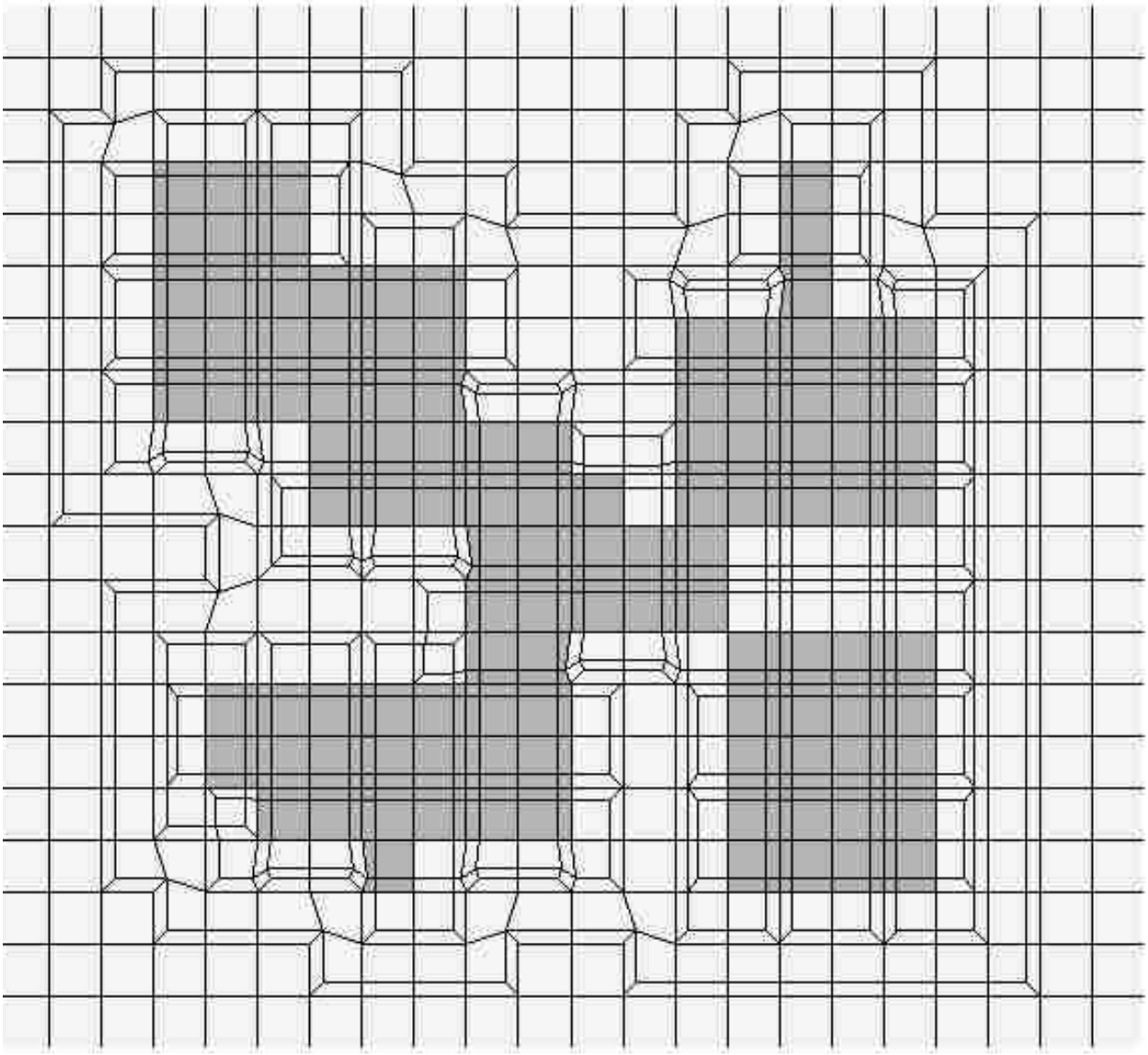


Figure 2-12: Pillowing the current sheet in the X_3 direction

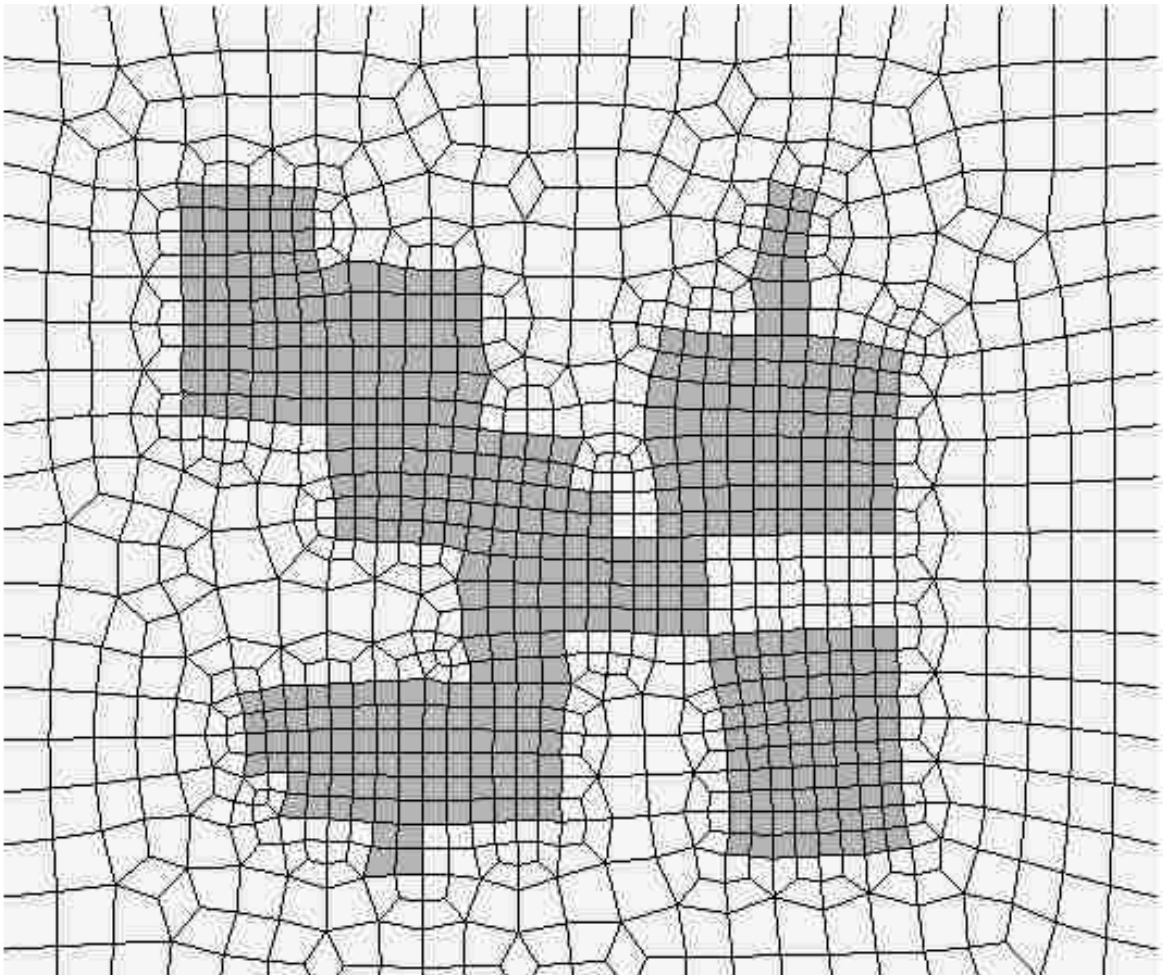


Figure 2-13: Pillowing the current sheet in the X_3 direction, smoothed

coordinate axes) and work our way down the X_1 direction, grabbing two sheets of hexes at a time. The first pair of sheets having hexes selected for refinement is given by Figure 2-15 on page 25. The second pair of sheets and corresponding shrink-set are given by Figure 2-16, and the final two are given by Figures 2-17 and 2-18 on pages 25–26. Figure 2-19 slightly shrinks the hexes of the first slice-pair in the X_1 direction in order to show results after pillowing. The shrink sets are

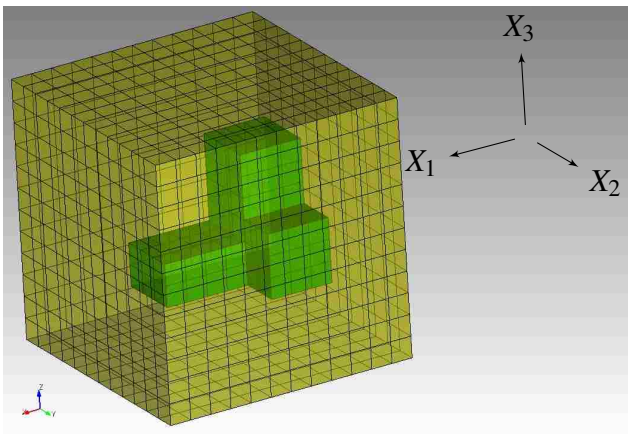
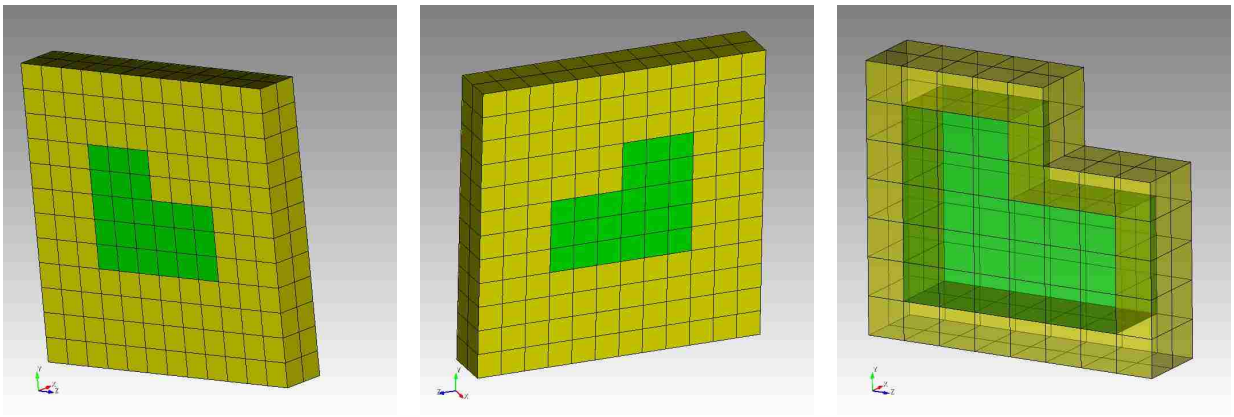


Figure 2-14: Initial example region to be pillowed in three directions

generally composed of all hexes immediately surrounding the volume in the X_2 and X_3 directions. However, note that in all pictures of the shrink-set, each slice-pair of hexes has uniform shape in the X_1 direction. For example, in Figure 2-16(c), the size of the volume on the rear slice of the shrink-set is smaller than the size of the volume on the front slice, yet the shape of the rear slice matches the front slice because additional hexes outside the volume are selected. In this case, these extra hexes are necessary so that the uniform refinement zone is completely surrounded within the slice-pair by hexes; otherwise, pillowing will produce unwanted hexes within the $\Omega_{\mathbf{URZ}}$. This requirement for additional hexes in order to make both sheets congruent is also necessary in order to prevent the production of low-quality hexes from concavities orthogonal to the direction that determines the sheets of hexes (currently X_1), as shown in Figure 3-1 on page 34.

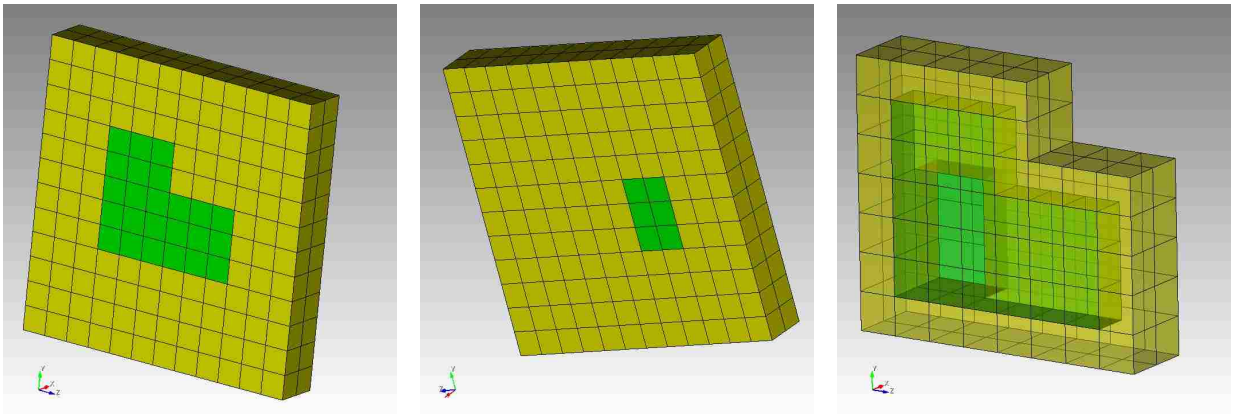
2.3.2 Pillowing in the X_2 Direction

Pillowing in the X_2 direction is identical to pillowing in the X_1 direction except that the relative directions change. During pillowing in the X_2 direction, the hexes that replace those in the original $\Omega_{\mathbf{URZ}}$ are added to the $\Omega_{\mathbf{URZ}}$, and those that replace those hexes in the transition zones



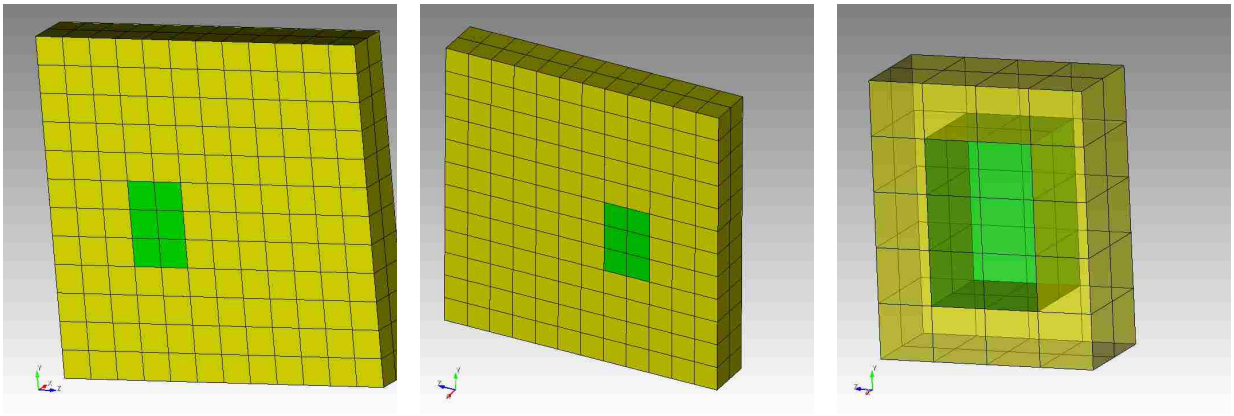
(a) Front of slice-pair (b) Rear of slice-pair (c) Shrink set

Figure 2-15: First slice-pair in X_1 direction



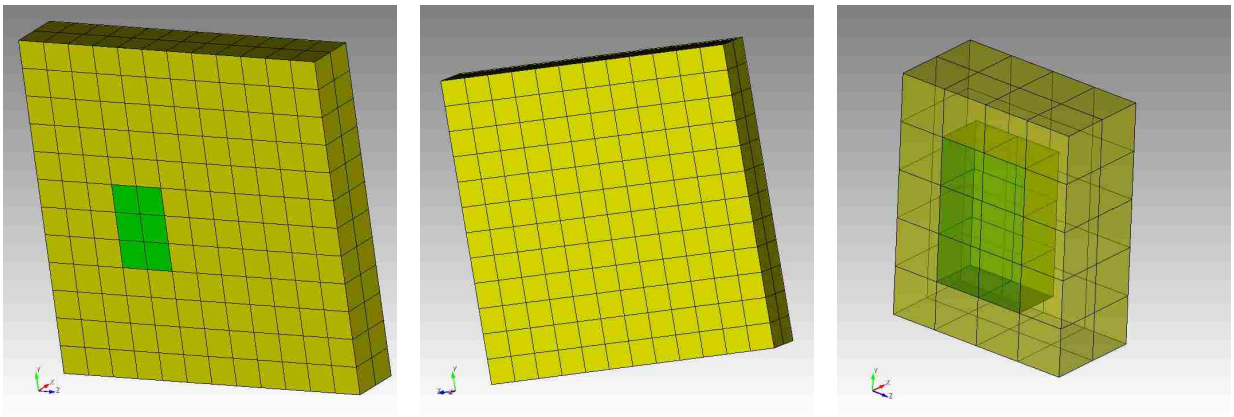
(a) Front of slice-pair (b) Rear of slice-pair (c) Shrink set

Figure 2-16: Second slice-pair in X_1 direction



(a) Front of slice-pair (b) Rear of slice-pair (c) Shrink set

Figure 2-17: Third slice-pair in X_1 direction



(a) Front of slice-pair (b) Rear of slice-pair (c) Shrink set

Figure 2-18: Fourth slice-pair in X_1 direction

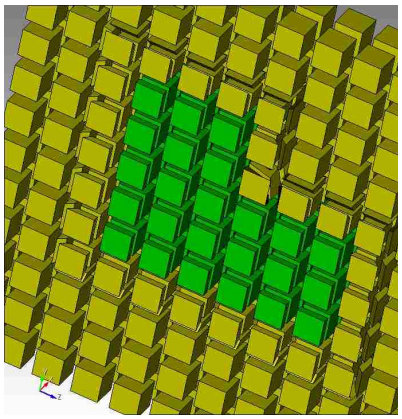
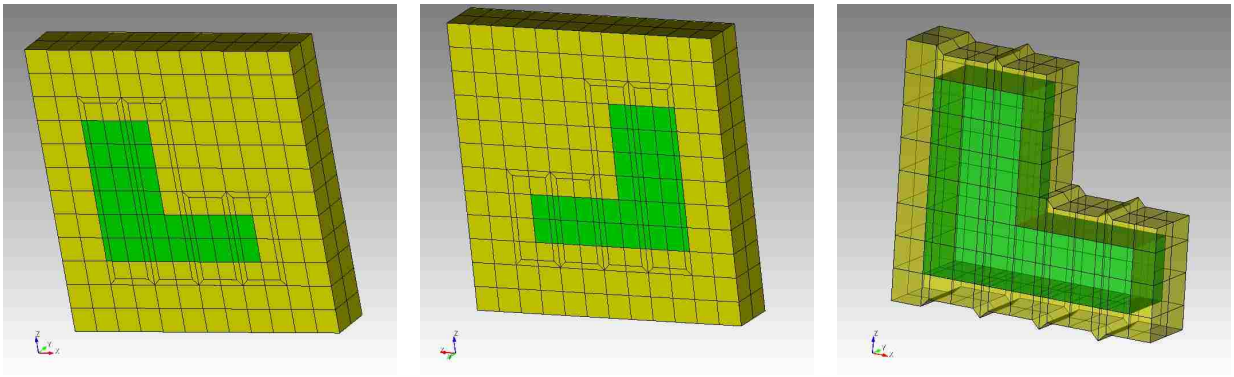


Figure 2-19: First slice-pair in X_1 direction after pillowing

are simply considered part of the rest of the mesh. The sequence of sets of hexes gathered for pillowing are illustrated in order by Figures 2-20, 2-21 and 2-22 on the next page.

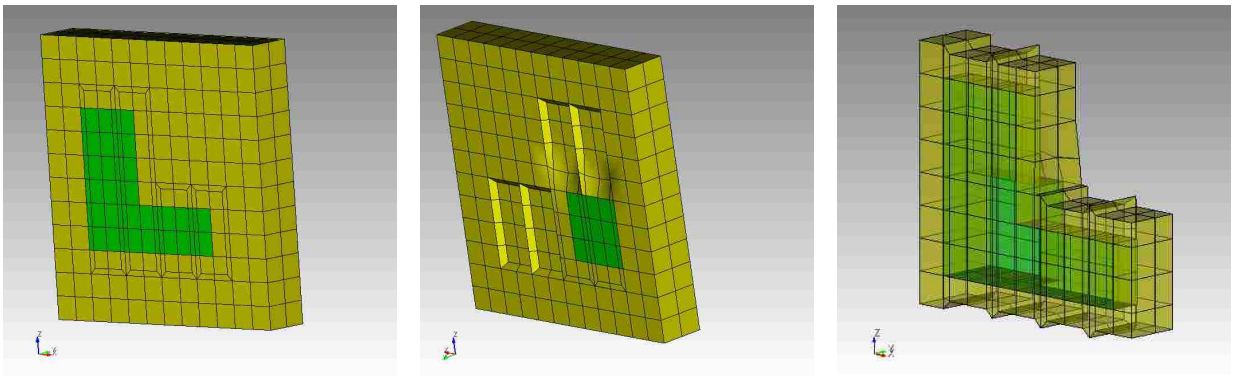
2.3.3 Pillowing in the X_3 Direction

Pillowing in the X_3 direction is a little different than pillowing in the X_2 direction. All hexes created in the X_1 and X_2 directions—regardless of if they were in the $\Omega_{\mathbf{URZ}}$ or if they were in the transition zone—are counted as part of the $\Omega_{\mathbf{URZ}}$. Of course, the relative directions also change.



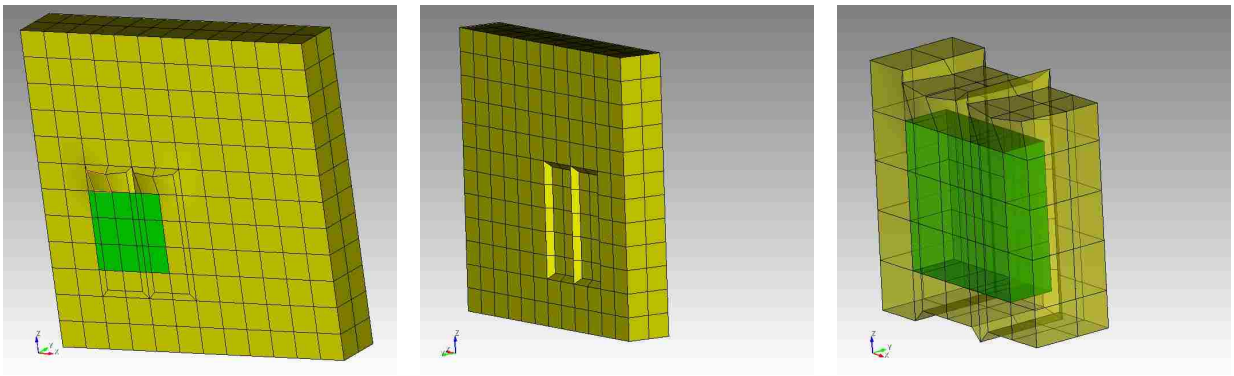
(a) Front of slice-pair (b) Rear of slice-pair (c) Shrink set

Figure 2-20: First slice-pair in X_2 direction



(a) Front of slice-pair (b) Rear of slice-pair (c) Shrink set

Figure 2-21: Second slice-pair in X_2 direction

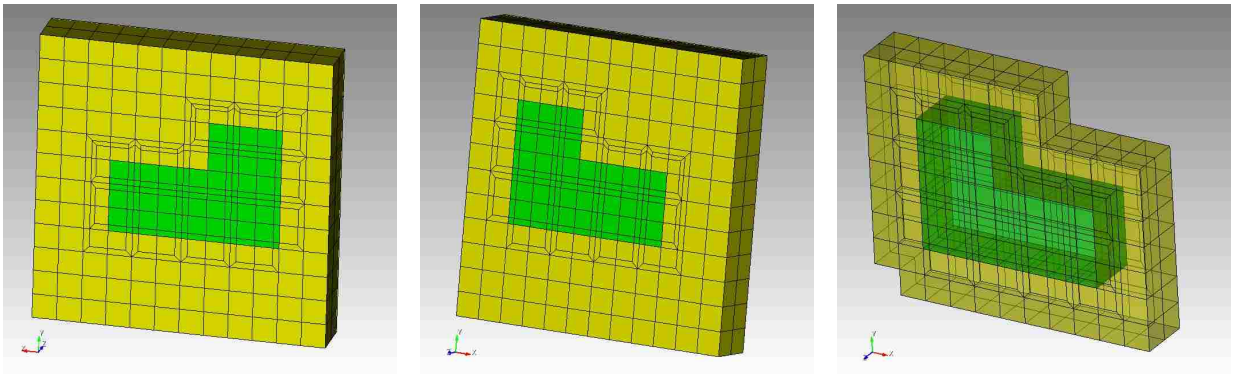


(a) Front of slice-pair (b) Rear of slice-pair (c) Shrink set

Figure 2-22: Third slice-pair in X_2 direction

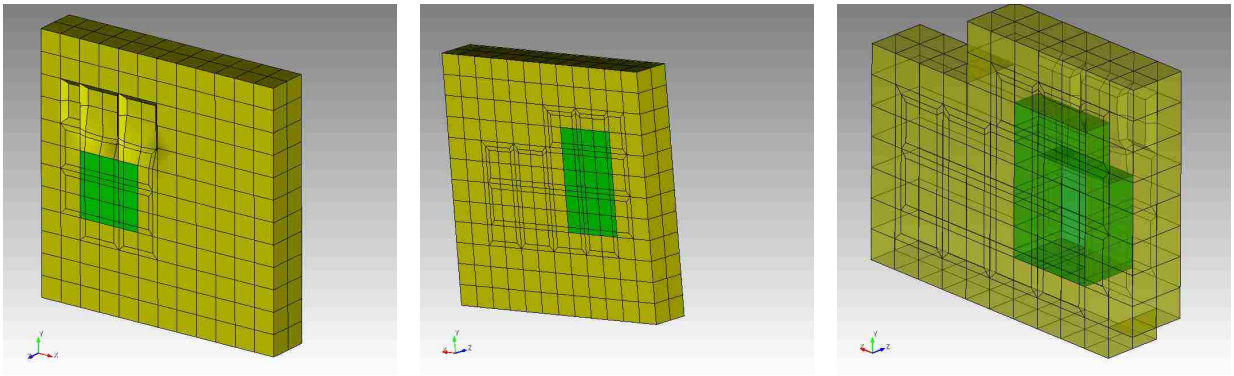
As shown in Chapter 3, this change in procedure increases mesh quality, though it does create more hexes than would following the procedure as given for the X_2 direction.

Figures 2-23 to 2-25 on pages 28–29 show the sequence of gathering shrink-sets for the X_3 direction. Upon shrinking, the algorithm is done, though the user likely will want to smooth the resulting mesh (Figure 2-26 on the next page).



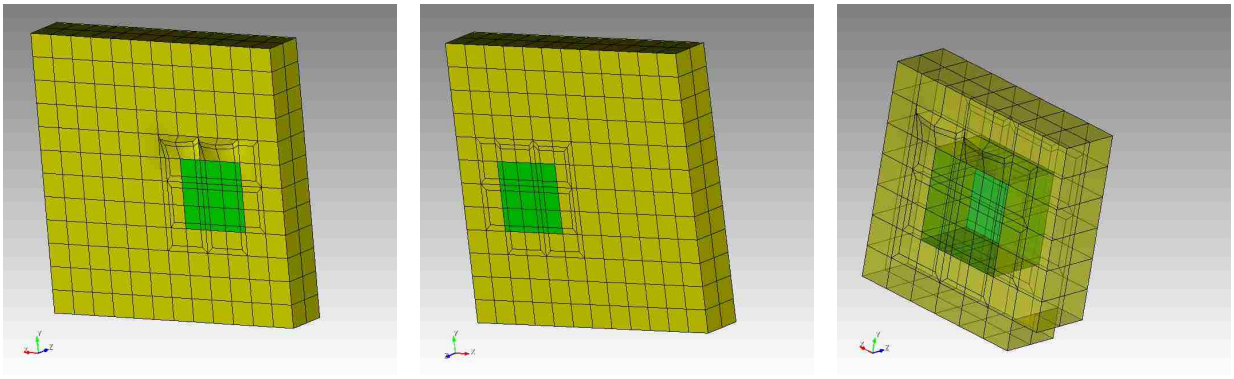
(a) Front of slice-pair (b) Rear of slice-pair (c) Shrink set

Figure 2-23: First slice-pair in X_3 direction



(a) Front of slice-pair (b) Rear of slice-pair (c) Shrink set

Figure 2-24: Second slice-pair in X_3 direction



(a) Front of slice-pair (b) Rear of slice-pair (c) Shrink set

Figure 2-25: Third slice-pair in X_3 direction

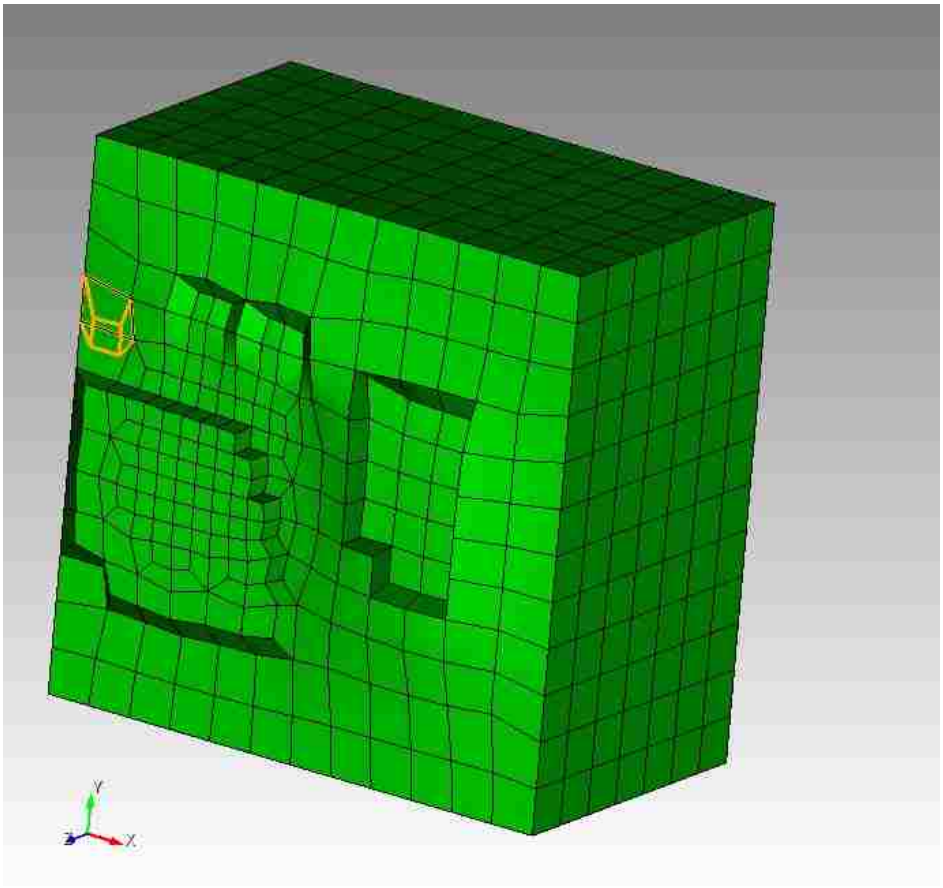


Figure 2-26: The finished mesh. The cut-away shown includes the lowest-quality hex (by the scaled-Jacobian metric), outlined in yellow.

Comparison to two dimensions In two dimensions, a $2 \times n$ region of hexes was selected for pillowing; this region may be thought of as two identically-shaped, adjacent columns of hexes, and they formed a rectangularly-shaped region. In three directions, however, two sheets instead were selected. While again both sheets were identically-shaped, they did not need to be rectangular (i.e. convex): concavities parallel to the direction of travel were permitted.

2.4 Recommended Algorithm

The algorithm as shown is open to a variety of implementations. A logical sequence of primary operations is shown in Figure 2-27. The diagram refers to traversing over hexes. This idea is simply to take a hex and designate opposite pairs of faces as corresponding to directions X_1, X_2 and X_3, respectively. Traversing in some direction is to get the face of a current hex corresponding to that direction, to grab the hex sharing that face and to change the current hex to it.

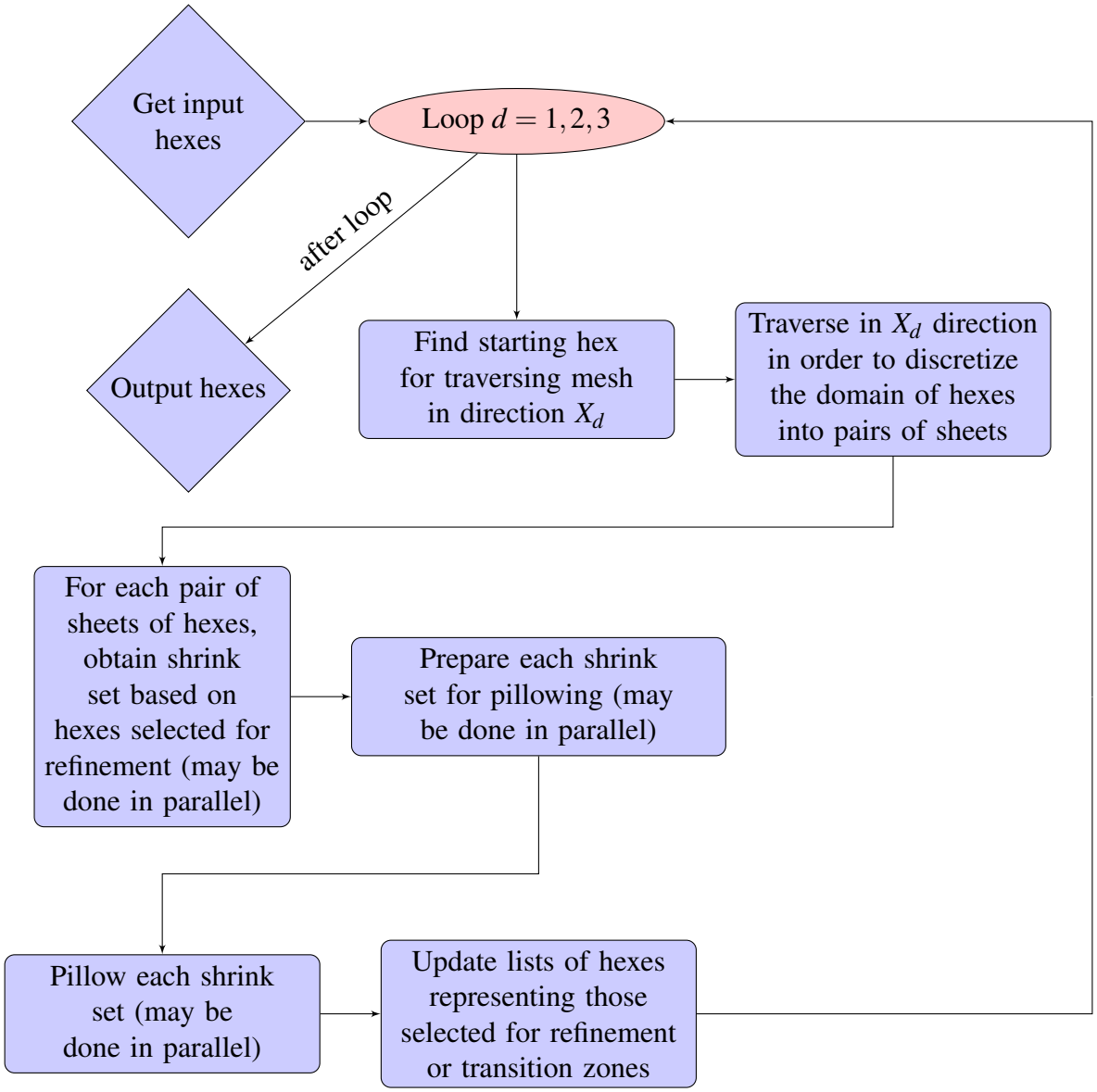


Figure 2-27: Flow chart of algorithm

3 DERIVATION OF ALGORITHM

3.1 Alternatives

The basic structure of the algorithm is to pillow consecutive pairs of sheets of hexes chosen for refinement in each of three directions. However, finding an implementation employed in selecting shrink sets offers numerous alternatives.

A class of these alternatives deals with how the algorithm should treat hexes that replace the shrink set. In particular: the shrink sets consist of both the hexes selected for refinement and an outer transition zone or two. These hexes all may be replaced by smaller hexes. While the hexes replacing those originally chosen for refinement may certainly be classified as hexes chosen for refinement, how to classify those hexes replacing the transition zones is ambiguous.

These alternatives inherently apply only to the algorithm running in the second and third directions. In order to choose the implementation chosen for this thesis, a number of alternatives were considered and tested using Cubit. They were tried on a few sample models and then smoothed using Cubit's mean ratio smoother that optimizes the hexes of a mesh with respect to a metric other than the scaled Jacobian metric; however, the testing assumed that the scaled Jacobian would typically follow the metric of the mean ratio smoother. In reality, the scaled Jacobian would sometimes decrease after smoothing.

The alternatives chosen for this thesis are as follows. The referenced figures indicate the respective shrink-sets that were each taken from identical sheets of hexes and hexes chosen for refinement during pillowing in the third direction.

Alternative A The hexes replacing the transition zone hexes are not considered with those chosen for refinement. Also, with respect to the domain of two sheets of hexes from which the shrink set is taken, all hexes that are node-connected with those chosen for refinement make up the transition zone; however, traversing in the direction perpendicular to the sheets, each shrink-set hex is not necessarily face-connected to another hex in the domain of two sheets of hexes. For an example, see Figure 3-1 on the following page.

Alternative B The hexes replacing the transition zone hexes are not considered with those chosen for refinement. Also, with respect to the domain of two sheets of hexes from which the shrink set is taken, all hexes that are node-connected with those chosen for refinement make up the transition zone. Furthermore, traversing in the direction perpendicular to the sheets, each shrink-set hex is face-connected to another hex in the domain of two sheets of hexes. For an example, see Figure 3-2 on the next page.

Alternative C The hexes replacing the transition zone hexes are considered with those chosen for refinement. In effect, a second transition zone is used. With respect to the domain of two sheets of hexes, all hexes that are node-connected with those chosen for refinement make up the transition zone. Furthermore, traversing in the direction perpendicular to the sheets, each shrink-set hex is face-connected to another hex in the domain of two sheets of hexes. For an example, see Figure 3-3 on page 35.

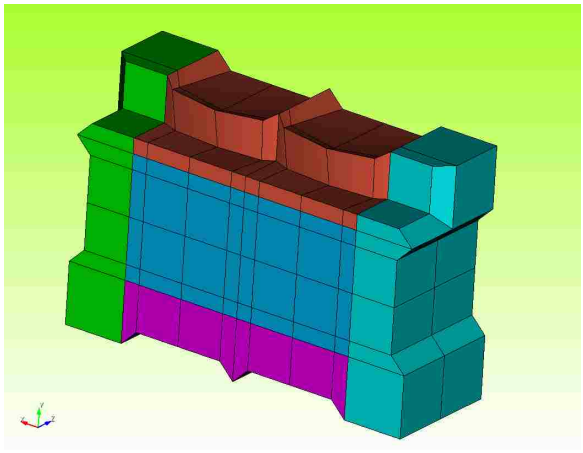


Figure 3-1: Shrink-set selected by Alternative A

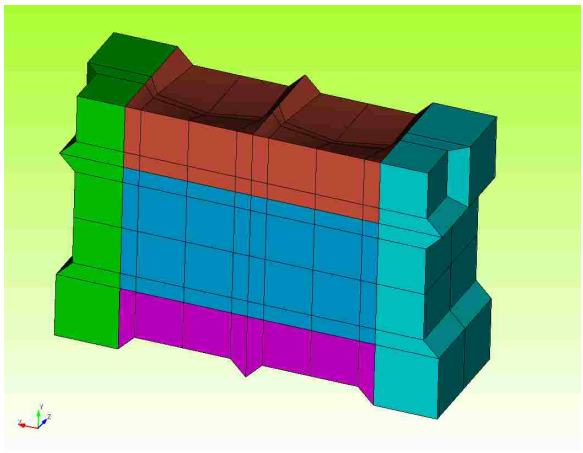


Figure 3-2: Shrink-set selected by Alternative B

Alternative D When traversing in the second direction, Alternative B is followed. When traversing in the third direction, Alternative C is followed. This is the alternative chosen for the presentation of this thesis.

3.2 Comparison of Approaches on Test Models

In choosing an appropriate implementation, two criteria were considered: the minimum scaled Jacobian hex quality after running the algorithm and the number of hexes generated. We

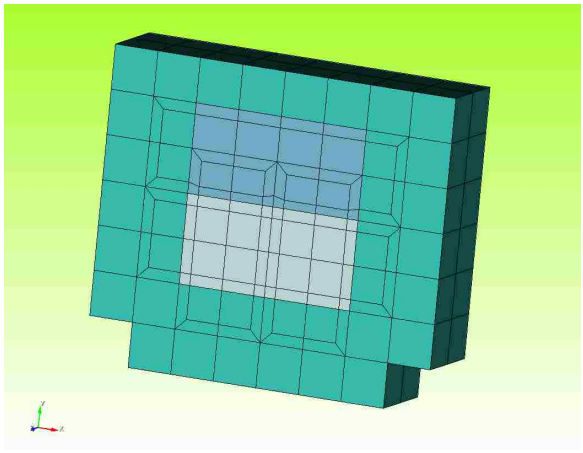


Figure 3-3: Shrink-set selected by Alternative C

first searched for an implementation that generates high-valued minimum scaled Jacobian hexes and as a second priority searched for an algorithm that would generate the fewest hexes.

The approach for selecting the appropriate implementation alternative, A, B, C or D, was entirely heuristic. Simple test regions of hexes were developed, as shown in Figures 3-4 to 3-8 on pages 37–39. The four alternatives were manually applied to each, using Cubit. Results from the testing are tabulated in Table 3-1 on page 38. The results from using Cubit's three-refinement feature are also included for comparison.

In considering these results, note the following:

- No one alternative provided the largest minimum scaled Jacobian every time, before or after smoothing.
- Before smoothing, Alternative B, Alternative D and three refinement did not generate negative-Jacobian hexes.
- Before smoothing, only Alternative D produced a minimum scaled Jacobian greater than three-refinement every time.

- After smoothing, only Alternative D produced a minimum scaled Jacobian greater than three-refinement every time.

- Before or after smoothing, Alternative D produced the largest minimum scaled Jacobian for the experiments Simple Cube and Stair Step, only. Alternative B produced the largest minimum scaled Jacobian for the experiment Chopped-corner, odd. Alternative C produced the largest minimum scaled Jacobian for the remaining two experiments.

- The two-transition zone alternatives, C and D, produced more hexes than the one-transition zone alternatives, A and B. However, Alternative D produced fewer than Alternative C.

Alternative D was chosen because it reliably produced relatively high-quality hexes. Furthermore, the number of hexes it produced was a compromise between Alternative B and Alternative C.

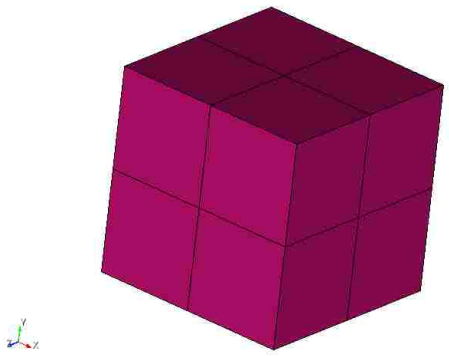


Figure 3-4: Simple Cube: A simple, convex test region for testing two-refinement implementations

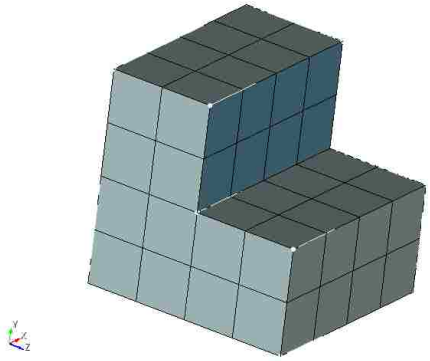


Figure 3-5: Stair-step: A test region for two refinement featuring concavity in one direction

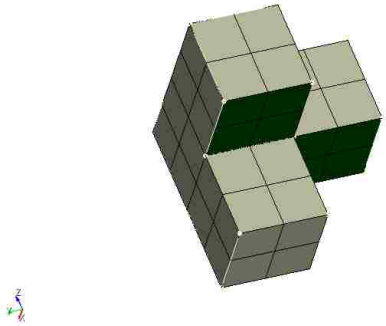


Figure 3-6: Triple-axis: A test region with multiple concavities in one direction

Table 3-1: Results from alternatives

Experiment Shape	Alternative	Number of transition zones used	Initial number of hexes selected for refinement	Number of hexes replacing those selected for refinement	Increase in number of transition zone hexes	Minimum scaled Jacobian before smoothing	Minimum scaled Jacobian after smoothing (mean ratio smoother)
Simple Cube	Alt. A	1	8	64	216	0.4243	0.5141
	Alt. B	1		64	216	0.4243	0.5141
	Alt. C	2		64	288	0.3959	0.5249
	Alt. D	2		64	264	0.3959	0.5536
	3-ref	1		216	384	0.3076	0.2839
Stair Step	Alt. A	1	48	384	694	0.1489	0.4266
	Alt. B	1		384	696	0.3919	0.3703
	Alt. C	2		384	988	-0.2475	0.4813
	Alt. D	2		384	864	0.3919	0.5169
	3-ref	1		1296	1248	0.2365	0.4316
Triple Axis	Alt. A	1	32	256	622	0.1489	0.4091
	Alt. B	1		256	630	0.3919	0.3472
	Alt. C	2		256	972	-0.2475	0.4446
	Alt. D	2		256	808	0.3919	0.3652
	3-ref	1		864	1056	0.1399	0.2880
Chopped Corner, Odd	Alt. A	1	189	1512	1575	0.0000	0.4242
	Alt. B	1		1512	1575	0.2484	0.4556
	Alt. C	2		1512	1991	0.3957	0.4394
	Alt. D	2		1512	1835	0.2838	0.4121
	3-ref	1		5103	2916	0.1159	0.3382
Chopped Corner, Even	Alt. A	1	448	3584	2644	-0.0710	0.3202
	Alt. B	1		3584	2644	0.1106	0.3715
	Alt. C	2		3584	3348	0.3959	0.2547
	Alt. D	2		3584	3096	0.2112	0.3867
	3-ref	1		3584	5040	0.1159	0.3098

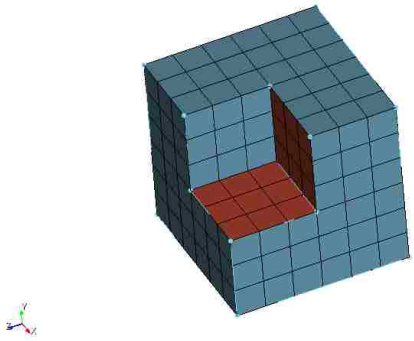


Figure 3-7: Chopped-corner, odd: A test region with a concavity in two directions and an odd number of hexes

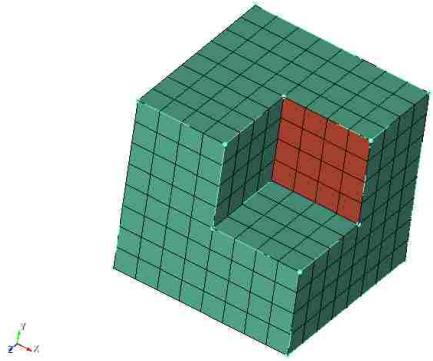


Figure 3-8: Chopped-corner, even: A test region with a concavity in two directions and an even number of hexes

4 EXAMPLES AND CONCLUSIONS

We coded the algorithm described in Chapter 2 into Cubit and ran test models through it and Cubit's three-refining algorithm in order to compare results and ensure that the algorithm proposed by this thesis works as predicted. Parallel processing was not utilized in our code.

The uniform refinement zones of two test models are given by Figures 4-1 and 4-2 on pages 41–42.

Results from testing are tabulated in Table 4-1 on page 44. From the table, the following remarks may be made:

- Both this thesis' two-refinement algorithm and Cubit's three-refinement algorithm produced topologically conformal meshes that could be smoothed into better-quality meshes.
- Before and after smoothing, the two-refinement implementation yielded a higher quality mesh.
- The two-refinement algorithm yielded fewer elements by greater than a factor of 2.

In other words, the two-refinement algorithm proposed by this thesis compares well with the current available option for refinement in Cubit. Because it yields relatively high-quality results and produces relatively few hexes, it meets the objectives proposed by this thesis. To our knowledge, no other two-refinement algorithm has demonstrated as effectively the ability to handle concavities in structured meshes while maintaining relatively-low refinement density.

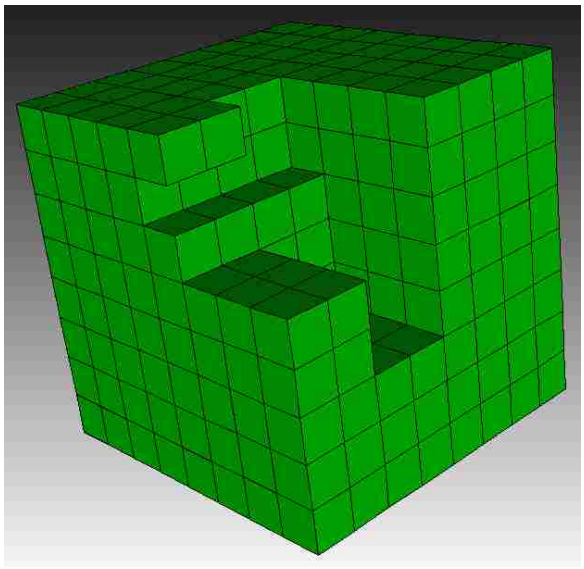


Figure 4-1: Test case A: with concavities on corner

Cutaway sections after running each refinement method and smoothing are illustrated in Figures 4-3 to 4-6. They show that the transition from refined hexes to unrefined hexes is smoother in the two-refinement case because the change in hex-density is lower.

The framework of the algorithm provides the basis for further research. Examples of such research include more efficient methods of implementation through templates and handling the case of unstructured meshes. The current algorithm is intended to further meshing research involving overlay-grids by helping adapt them to geometries.

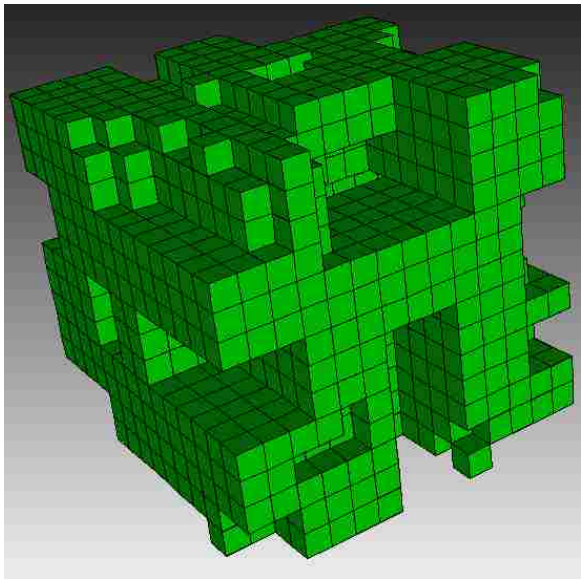


Figure 4-2: Test case B: with concavities all over

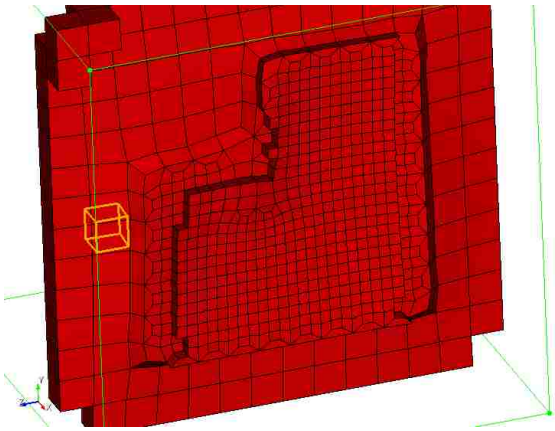


Figure 4-3: Cutaway from test case A after applying 3-refinement

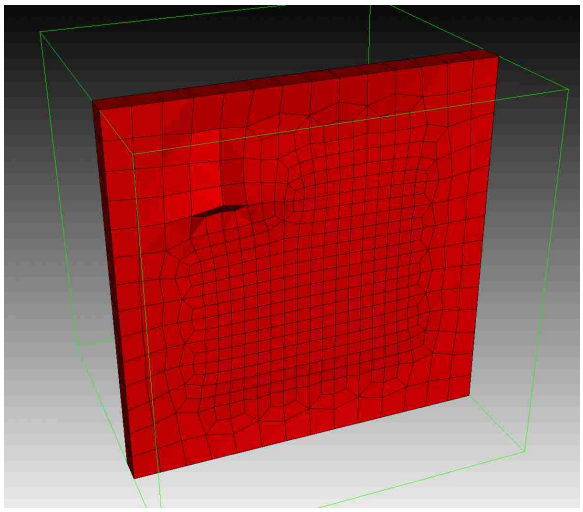


Figure 4-4: Cutaway from test case A after applying 2-refinement

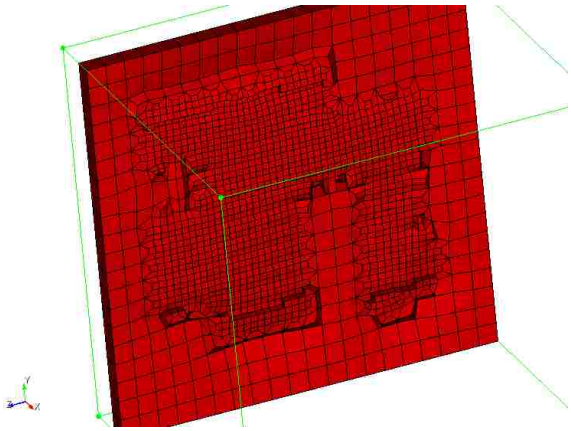


Figure 4-5: Cutaway from test case B after applying 3-refinement

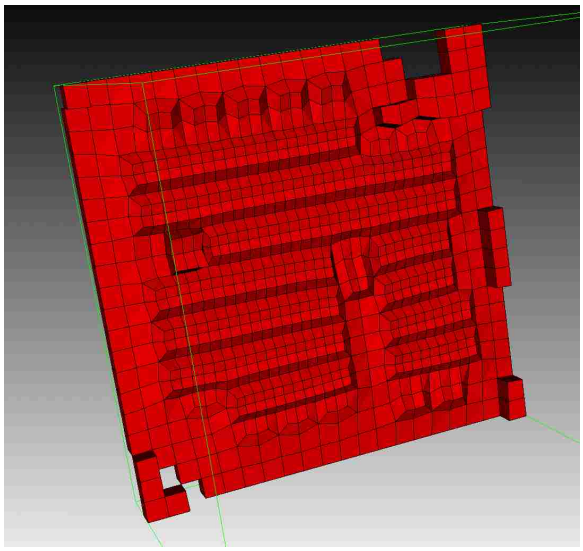


Figure 4-6: Cutaway from test case B after applying 2-refinement

Table 4-1: Results of tests

		Test	
		A	B
Thesis algorithm (2-refinement)	Increase in number of hexes	6432	26472
	Scaled Jacobian before smoothing	0.2182	0.0000
	Scaled Jacobian after smoothing	0.4999	0.3810
	Smoothing methods employed	mean ratio	untangle, mean ratio
Cubit algorithm (3-refinement)	Increase in number of hexes	17236	72444
	Scaled Jacobian before smoothing	-0.07213	-0.07230
	Scaled Jacobian after smoothing	0.2981	0.2659
	Smoothing methods employed	untangle, mean ratio	untangle, mean ratio, condition number

REFERENCES

[1] Boyce, W. E., and DiPrima, R. C., 2009. *Elementary Differential Equations*., 9 ed. John Wiley & Sons, Inc., Danvers, MA, ch. 1.3. 1

[2] Burden, R. L., and Faires, J. D., 2005. *Numerical Analysis*., 8 ed. Thomson Brooks/Cole, Belmont, CA, ch. 5.1. 1

[3] Burden, R. L., and Faires, J. D., 2005. *Numerical Analysis*., 8 ed. Thomson Brooks/Cole, Belmont, CA, ch. 12.4. 1

[4] Rao, S. S., 2011. *The finite element method in engineering*., 5 ed. Butterworth-Heinemann, Burlington, MA. 1

[5] Schneiders, R., 1999. "Quadrilateral and hexahedral element meshes." In *Handbook of Grid Generation*, J. F. Thompson, B. K. Soni, and N. P. Weatherill, eds. CRC Press LLC, New York, pp. 21-1–21-27. 2

[6] Blacker, T., 2001. "Automated conformal hexahedral meshing constraints, challenges and opportunities." *Engineering with Computers,* **17**(3), p. 201–210. 2, 6

[7] Dewey, M. W., Benzley, S. E., Shepherd, J. F., and Staten, M. L., 2008. "Automated quadrilateral coarsening by ring collapse." In *Proceedings of the 17th International Meshing Roundtable*, R. V. Garimella, ed. Springer Berlin Heidelberg, p. 93. 2

[8] Shepherd, J. F., and Johnson, C. R., 2008. "Hexhedral mesh generation constraints." *Finite Elements in Analysis and Design,* **24**(3), pp. 195–213. 2

[9] Thompson, J. F., Soni, B. K., and Weatherill, N. P., eds., 1999. *Handbook of Grid Generation*. CRC Press LLC, New York, ch. P3.2.3. 2

[10] Woodbury, A. C., Shepherd, J. F., Staten, M. L., and Benzley, S. E., 2008. "A mesh optimization algorithm to decrease the maxiumum error in finite element computations." In *Proceedings of the 17th International Meshing Roundtable*, R. V. Garimella, ed. Springer Berlin Heidelberg, pp. 603–619. 2, 5, 6

[11] Weingarten, V. I., 1994. "The controversy over hex or tet meshing." *Machine Design,* **66**(8), April, p. 74. 2

[12] Alves, M., Fernandes, J., Rodrigues, J., and Martins, P., 2003. "Finite element remeshing in metal forming using hexahedral elements." *Journal of Materials Processing Technology,* **141**(3), February, p. 395–403. 2

[13] Benzley, S. E., Perry, E., Merkley, K., Clark, B., and Sjaardama, G., 1995. "A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic

analysis." In *Proceedings, 4th International Meshing Roundtable*, Vol. 17, Sandia National Laboratories, pp. 179–191. 3

[14] Hu, J., Lee, Y. K., Blacker, T., and Zhu, J., 2002. "Overlay grid based geometry cleanup." In *Proceedings of the 11th International Meshing Roundtable*. Sandia National Laboratories, Springfield, VA, pp. 313–322. 3, 6

[15] Owen, S. J., 2012. "Parallel smoothing for grid-based methods." In *Proceedings of the 21st International Meshing Roundtable* research note. 4

[16] Kremer, M., Bommes, D., and Kobbelt, L., 2012. "Open volume mesh–a versatile index-based data structure for 3d polytopal complexes." In *Proceedings of the 21st International Meshing Roundtable*, X. Jiao and J.-C. Weill, eds. Springer Berlin Heidelberg. 4

[17] Kwok, W., and Chen, Z., 2002. "A simple and effective mesh quality metric for hexahedral and wedge elements." In *Proceedings of the 9th International Meshing Roundtable*. Sandia National Laboratories, Springfield, VA, pp. 325–333. 4

[18] Tristano, J. R., Chen, Z., Hancq, D. A., and Kwok, W., 2003. "Fully automatic adaptive mesh refinement integrated into the solution process." In *Proceedings of the 12th International Meshing Roundtable*. Sandia National Laboratories, Springfield, VA, pp. 307–314. 5

[19] Shepherd, J. F., Dewey, M. W., Woodbury, A. C., Benzley, S. E., Staten, M. L., and Owen, S. J., 2010. "Adaptive mesh coarsening for quadrilateral and hexahedral meshes." *Finite Elements in Analysis and Design,* **46**(1), pp. 17–32. 5

[20] Miranda, A. C. O., and Martha, L. F., 2002. "Mesh generation on high-curvature surfaces based on a background quadtree structure." In *Proceedings of the 11th International Meshing Roundtable*. Sandia National Laboratories, Springfield, VA, pp. 333–341. 5

[21] Zhu, J., Blacker, T., and Smith, R., 2002. "Background overlay grid size functions." In *Proceedings of the 11th International Meshing Roundtable*. Sandia National Laboratories, Springfield, VA, pp. 65–73. 5

[22] Zhang, Y., and Bajaj, C., 2004. "Adaptive and quality quadrilateral/hexahedral meshing from volumetric data." In *Proceedings of the 13th International Meshing Roundtable*. Sandia National Laboratories, Springfield, VA, pp. 365–376. 5

[23] Hetmaniuk, U., and Knupp, P., 2008. "A mesh optimization algorithm to decrease the maxiumum error in finite element computations." In *Proceedings of the 17th International Meshing Roundtable*, R. V. Garimella, ed. Springer Berlin Heidelberg, pp. 533–550. 5

[24] Edgel, J., 2010. "An adaptive grid-based all hexahedral meshing algorithm based on 2-refinement." Master's thesis, Brigham Young University, December. 6, 7, 8

[25] Ebeida, M. S., Patney, A., Owens, J. D., and Mestreau, E., 2011. "Isotropic conforming refinement of quadrilateral and hexahedral meshes using two-refinement templates." *International Journal for Numerical Methods in Engineering,* **88**(10), May, p. 974–985. 7

[26] Owen, S. J., and Shepherd, J. F., 2009. "Embedding features in a cartesian grid." In *Proceedings of the 18th International Meshing Roundtable*, B. W. Clark, ed. Springer Berlin Heidelberg, pp. 117–138. 8

[27] Mitchell, S. A., and Tautges, T., 1995. "Pillowing doublets: refining a mesh to ensure that faces share at most one edge." In *4th International Meshing Roundtable*, Sandia National Laboratories, pp. 231–240. 13