HEXHOOP: MODULAR TEMPLATES FOR CONVERTING A HEX-DOMINANT MESH TO AN ALL-HEX MESH*

Soji Yamakawa[1] and Kenji Shimada[2**]

[1]Carnegie Mellon University, Pittsburgh, PA, U.S.A., soji@andrew.cmu.edu
[2]Carnegie Mellon University, Pittsburgh, PA, U.S.A., shimada@cmu.edu

ABSTRACT

This paper presents a new mesh conversion template, called HEXHOOP, that fully automates a conversion from a hex-dominant mesh to an all-hex mesh. A HEXHOOP template subdivides a hex/prism/pyramid element to a set of smaller hex elements while maintaining the topological conformity with neighboring elements. A HEXHOOP template is constructed by assembling sub-templates, *cores* and *caps*. A dicing template for a hex and a prism is constructed by choosing the appropriate combination of a *core* and *caps*. A template that dices a pyramid without losing conformity to the adjacent element is derived from a HEXHOOP template. Some experimental results show that the HEXHOOP templates successfully convert a hex-dominant mesh to an all-hex mesh.

Keywords: mesh conversion, template, hexahedral mesh, hex-dominant, pyramid

1. INTRODUCTION

We propose new mesh conversion templates, called HEXHOOP, that fully-automate a conversion from a hex-dominant mesh to an all-hex mesh. A hex-dominant mesh is a three-dimensional mesh consisting of four types of elements, hexahedral elements, prism elements, pyramid elements and tetrahedral elements, as illustrated in Figure 1 (a). An all-hex mesh is a mesh consisting of exclusively hexahedral elements. Figure 1 (b) shows an example of an all-hex mesh converted from a hex-dominant mesh shown in Figure 1 (a) by using our HEXHOOP templates.

Such conversion templates make all-hex meshing possible for a complicated geometry for which the existing direct all-hex meshing methods are inadequate. Although it would be ideal if an all-hex mesh could be generated for an arbitrary three-dimensional shape without going through a hex-dominant mesh [1, 2], the direct all-hex meshing problem is known to be highly challenging, and none of the exiting methods always succeeds to create a high-quality all-hex mesh for a complex three-dimensional geometry. Although there exists a trivial solution, creating a tetrahedral mesh first and subdividing each of the tet elements into four smaller hex elements, such an all-hex mesh is topologically so irregular that this method is not used in practice. Creating a quality hex-dominant mesh, on the other hand, is an easier problem to solve, either by hand or by an automated algorithm [3, 4]. Our HEXHOOP templates take as input any type of hex-dominant mesh and convert it to an all-hex mesh automatically.

In order to highlight the difficulty in developing conversion templates for all-hex meshing, let us first examine a much easier, two-dimensional problem of converting a quad-dominant mesh to an all-quad mesh, as illustrated in Figure 2 (a). The solution to this problem is well known—we need only two types of templates shown in Figure 2 (b). With

* Patent pending

** Corresponding author:
Kenji Shimada, Department of Mechanical Engineering, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213-3890, U.S.A.
Tel: (412)268-3614, Fax: (412)268-3348, Email: shimada@cmu.edu

these two types of templates, it is guaranteed that we convert any quad-dominant mesh to an all-quad mesh.

During this all-quad mesh conversion, it is important to maintain the interface conformity, or the topological and geometric conformity between adjacent mesh elements. To maintain conformity, each of the all interior-edges of a final mesh must be shared by exactly two elements. By using the two templates shown in Figure 2 (b) it is easy to satisfy such conformity in the all-quad mesh conversion because all the edges of an input quad-dominant mesh are always split into two segments.

Let us now consider the all-hex mesh conversion problem. A similar interface comformity requirement exists. The common method for converting a hex-dominant mesh into an all-hex mesh is to subdivide, or dice, a non-hex element into a set of smaller hexes. A hex in the original mesh is also subdivided into a set of smaller hexes. In a final all-hex mesh, all the interfaces between adjacent hexes must be quadrilaterals, and each of the quadrilaterals must be shared by exactly two hexes in order to maintain the conformity.

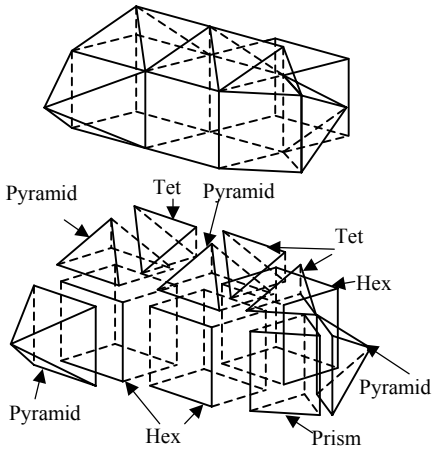
Despite the apparent similarity between the problem statements, this all-hex mesh conversion problem is significantly more challenging than the all-quad mesh conversion problem for the following two reasons:

- An input hex-dominant mesh consists of four different types of elements, hexes, prisms, pyramids and tets, as opposed to only two types of elements in a quad-dominant mesh.
- A hex-dominant mesh has two types of interfaces, triangles and quadrilaterals, which makes it more difficult to maintain the topological and geometric conformity at the interfaces, compared with the all-quad mesh conversion problem, in which there is only one type of interfaces, a line segment.

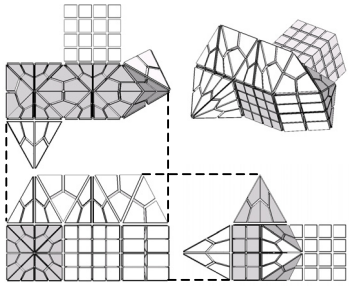
Among the four types of elements in a hex-dominant mesh, hexes, tets, and prisms have the following well-known, simple conversion templates:

- A hex element can be split into eight smaller hex elements by adding a node at the center of the volume, six nodes at the centers of six quadrilateral faces, twelve nodes at the centers of twelve edges of the original hex.
- A tet element can be split into four smaller hex elements by adding a node at the center of the volume, four nodes at the centers of four triangular faces, and six nodes at the centers of six edges of the original tet.
- A prism can be split into six smaller hex elements by adding a node at the center of the volume, five nodes at the centers of two triangular faces and three quadrilateral faces, and nine nodes at the centers of nine edges of the original prism.

Note that all of these three templates apply aforementioned all-quad templates, shown in Figure 2 (b), —splitting a triangular face of the original element into three smaller quadrilaterals, and a quadrilateral face into four smaller quadrilaterals.

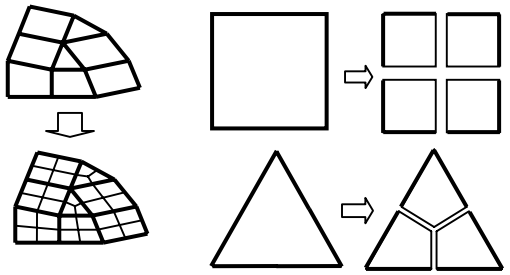


(a) an example of a hex-dominant mesh



(b) an all-hex mesh

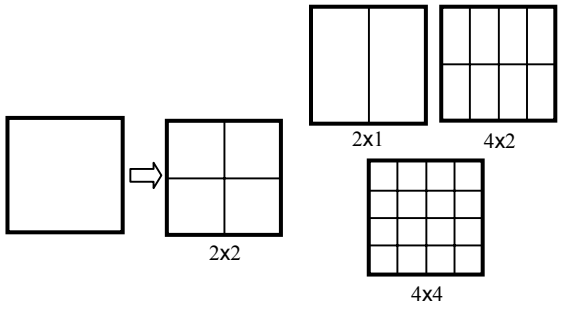
Figure 1 Converting a hex-dominant mesh to an all-hex mesh



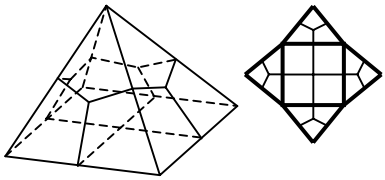
(a) conversion of a quad-dominant mesh to an all-quad mesh

(b) two all-quad conversion templates

Figure 2 Templates for converting a quad-dominant mesh to an all-quad mesh

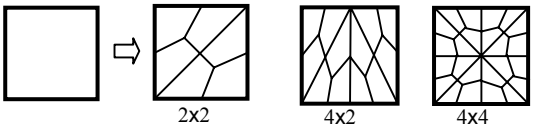


(a) basic rectangular pattern (b) variations of rectangular pattern

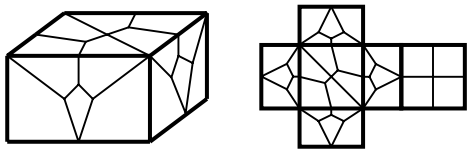


(c) Schneiders' Open Problem: a valid dicing of this pyramid has not been published.

Figure 3 Rectangular pattern and Schneiders' Open Problem



(a) basic triangular pattern (b) variations of triangular pattern



(c) Mitchell's Geode template: the side face has an irregular pattern.

Figure 4 Triangular pattern and Mitchell's Geode template

One problem is that there is no such template known for a pyramid; if it exists, such a template should subdivide four triangular faces of the pyramid into three smaller quadrilateral faces and the bottom quadrilateral face to four smaller quadrilaterals as shown in Figure 3 (a). We call this subdivision pattern of the bottom quadrilateral face and its variations *rectangular patterns*. Given this boundary mesh consisting of 16 quadrilaterals, finding a valid internal structure that dices the pyramid into a set of smaller hex elements is difficult, and there is no valid solution published for this open problem as Schneiders discusses in his paper [5] and one of his web pages [6]. This problem is referred to as "Schneiders' Open Problem". Although Carboner proposes a solution to the problem [7], his solution is not valid because some interior faces are not shared by two hexes.

One known simple template for a pyramid first splits the pyramid into two tet elements, and then applies the known tet template to each tet. With this template the bottom quadrilateral face of a pyramid is split into a pattern shown in Figure 4 (a). This pattern and its variations are referred to as *triangular patterns*. If we subdivide a bottom face of a pyramid into a triangular pattern, however, a hex element or a prism element adjacent to the pyramid must have a triangular pattern on one face in order to maintain the interface conformity. This brings up another unsolved problem of finding conversion templates for a hex and a prism that have both rectangular patterns and triangular patterns mixed on the exterior surface of the hex and the prism. Mitchell presents a partial solution to this problem as shown in Figure 4 (c) [8], but this template has an irregular subdivision pattern on the side faces of a hex, which limits its application and practical value.

In summary, there are two approaches to the all-hex conversion template, but no complete solution to these two approaches has been published:

(1) to find templates for a pyramid with a rectangular subdivision pattern on the bottom face, as Schneiders pointed out in [5] and [6], and

(2) to find a template for a hex and a prism that have mixed subdivision patterns, rectangular and triangular, as Mitchell attempted [8].

In this paper we propose a family of new mesh conversion templates, called HEXHOOP, that provide solutions to both unsolved problems. Unlike previously published templates, HEXHOOP is not a single specific template; it is a systematic method for constructing a family of modular sub-templates that can be assembled to form all-hex conversion templates for hexes, pyramids, and prisms. This new template design uses two types of modular sub-templates, called a *core* and a *cap*. For a hex or prism element we define one core, which specifies the subdivision patterns of two opposite faces of the input hex or prism. We then define four caps for a hex and three caps for a prism to specify the subdivision patterns of the other faces. The advantage of the HEXHOOP method is that two subdivision patterns, rectangular and triangular, can be mixed and matched freely on the exterior surfaces of a hex, prism, and a pyramid.

The rest of this document is organized as follows. Section 2 discusses the basic concept of the HEXHOOP template, and Section 3 explains detailed construction methods of HEXHOOP templates. We show possible variations of cores in Section 4 and present two solutions to Schneiders' Open Problem in Section 5.

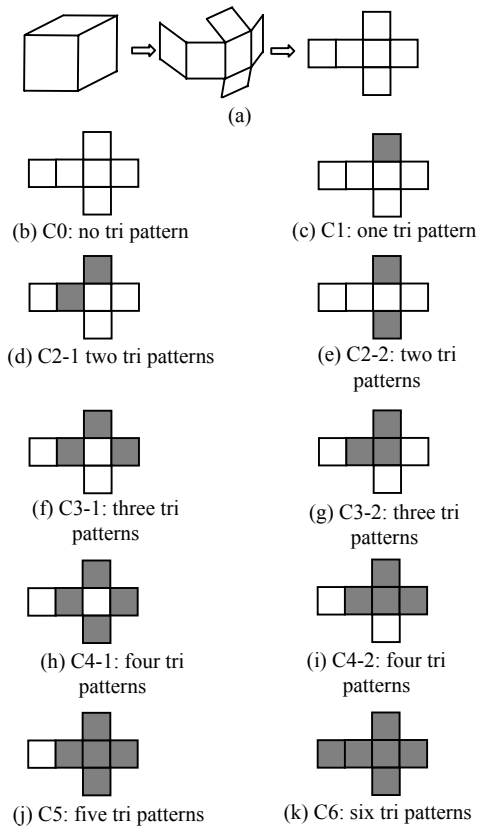


Figure 5 Ten possible combinations of triangular patterns and rectangular patterns on a template (shaded faces are triangular patterned faces)

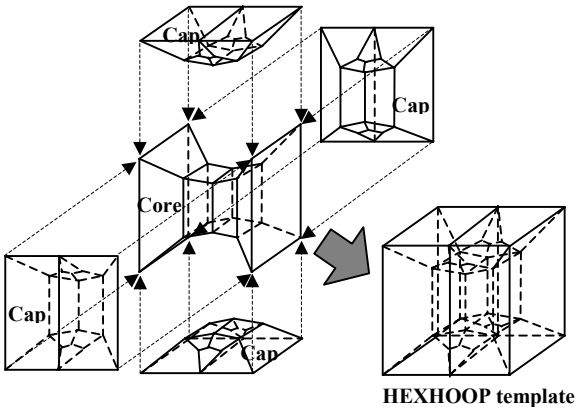


Figure 6 HEXHOOP's modular approach to a all-hex template

2. MODULAR APPROACH TO ALL-HEX TEMPLATES

Our goal is to develop a system of all-hex templates for hexes, prisms, and pyramids. As discussed in Sections 1 the difficulty is that there exists two face subdivision patterns, rectangular pattern and triangular pattern, mixed on the exterior faces of a hex, prism, and pyramid. In this section and the next two sections we primarily discuss templates for hexes in detail to explain our new modular design approach of HEXHOOP. Templates for prisms and pyramids are discussed later in Sections 4 and 5.

Because a hex template has six exterior faces, and each of the faces has either a rectangular or triangular pattern, there are ten different hex templates. The number of exterior faces with a triangular pattern ranges from 0 to 6, and in the cases where the number of such exterior faces is 2, 3, and 4, there exists two topologically different ways to choose the exterior faces as illustrated in Figure 5. Among the ten cases there are two cases for which there exists a known, simple solution: a hex template with six rectangular patterned faces (Figure 5 (b)), and a hex template with two triangular patterned faces, one on the top and one on the bottom (Figure 5 (e)). Solutions to the other eight cases are not trivial, and to our knowledge there has been no published solution to creating a valid template for all of the cases.

To tackle this challenging all-hex template problem, we propose a new modular approach that provides a systematic method of constructing a family of modular sub-templates that can be assembled to form all-hex conversion templates. Unlike previously published templates, HEXHOOP is not a single specific template. Instead, we first define sub-templates, each of which has either rectangular or triangular subdivision patterns on its external faces, and then assemble them together. Our new template design uses two types of modular sub-templates, called a *core* and a *cap*. For a hex template we define one core, which specifies the subdivision patterns of two opposite faces and then define four caps to specify the subdivision patterns of the other four faces, as shown in Figure 6. This new modular approach to all-hex template generation is superior because two subdivision patterns, rectangular and triangular, can be mixed and combined freely on the exterior surfaces of a hex.

The design of the cap is key to the modular template design. Each cap has two faces, front and back, with an irregular subdivision pattern. Thus it exhibits the same problem as Mitchell's Geode template—all the irregular faces must be matched and shared with the next cap. Mitchell addressed this problem by assuming that all the Geodes are laid out in a closed shell-like volume, which is a strong restriction on the applicability of the template. In our HEXHOOP approach we solve this problem by arranging four caps to form a hoop, or a ring. In this way, all the irregular faces are matched and shared by adjacent caps without exposing them to the exterior of the template. This is the single most important feature of our modular template design. Therefore the template is named "HEXHOOP", meaning a hoop of hex elements.

Each cap has one exterior face to be subdivided into either a rectangular or triangular pattern. The construction process

for each type of cap is as follows: We begin with an all-hex mesh consisting of four elements and subdivide one end into a triangular pattern as shown in Figure 7 (a). This triangular pattern marches through a mesh and reaches the other end as shown in Figure 7 (b). Next, the corners of the two ends are joined as also shown in Figure 7 (b). By joining two corners we obtain a volumetric region surrounded by inside faces of the mesh as shown in Figure 7 (c). This region is called a *pipe*. We then add some hexes (in this case two hexes) and fill the pipe. Finally, an appropriate deformation scheme is applied so that neither gaps nor overlaps remain when the cap is assembled with a core and other caps. The final shape of the cap is shown in Figure 7 (d). We call this all-hex sub-template a cap of the HEXHOOP template. Because the top faces form a triangular pattern, we can connect a diced tet, pyramid, or prism without losing the mesh conformity. In a similar way we can subdivide one end face into a rectangular pattern. This way, we obtain a cap with a rectangular pattern on its external face. To distinguish two different types of caps, we call a cap with a triangular pattern a *triangular cap* and a cap with a rectangular pattern a *rectangular cap*.

An important property of this cap design is that both rectangular and triangular caps share the identical boundary face subdivision patterns except on the top face. It is especially convenient to have a simple rectangular pattern on both types of caps on the bottom face. A volumetric region surrounded by four 'hooped' caps is therefore a region surrounded by four rectangular patterns. A sub-template for this volume is called a *core*, and the core is easily constructed by sweeping a quad mesh shown in Figure 3 (a), Figure 3 (b), Figure 4 (a) or Figure 4 (b) four times.

3. CONSTRUCTION OF HEXHOOP TEMPLATES FOR A HEX ELEMENT

3.1. Construction of a Standard Core

HEXHOOP's core has two *wing faces* and four *slots* as shown in Figure 8. Four slots are labeled as slot 0, slot 1, slot 2 and slot 3. The two wing faces and three vertical, cross-sectional faces are all rectangles. In order to dice such a core into a set of smaller hexes we subdivide two sets of parallel edges, vertical and horizontal, into a same number of line segments. This will subdivide the two wing faces and three cross-sectional faces into a structured rectangular grid pattern. We denote as n_1 the number of sub-edges in horizontal direction, and as n_2 the number of sub-edges in vertical direction as shown in Figure 9. Then the number of subdivision of slot 0 and slot 1 is n_1, and slot 2 and slot 3 n_2. We call this type of core a $n_1 \times n_2$ *rectangular core*. An example of 2×4 rectangular core is shown in Figure 9 (a).

Another type of core is a *triangular core*, which has a triangular subdivision pattern on the two wing faces. Figure 9 (b) shows an example of such a core. Because this core has 4×4 triangular pattern on wing faces we call the core 4×4 *triangular core*. By choosing a different triangular subdivision pattern, shown in Figures 4 (a) and 4(b), and applying it to the wing faces, we can create different types of triangular cores such as a 4×2 triangular core and a 2×2 triangular core.

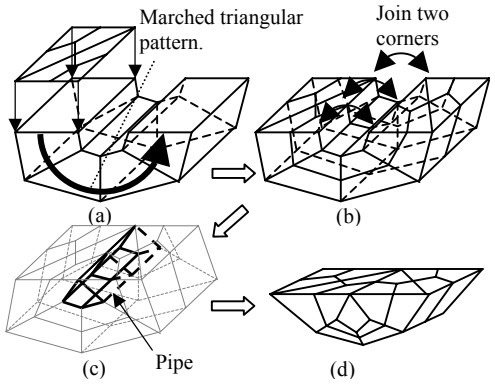


Figure 7 Construction of a triangular cap

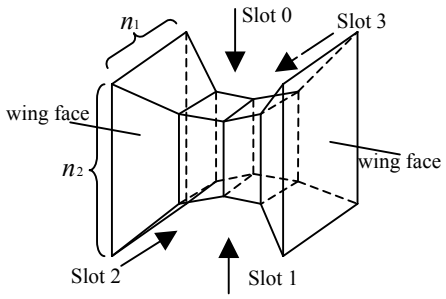


Figure 8 A core has two wing faces and four slots.

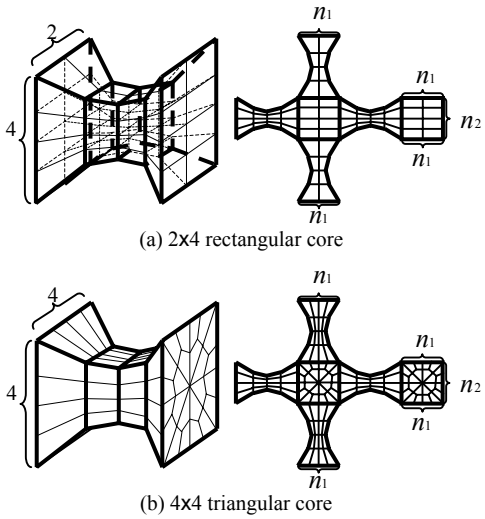


Figure 9 Rectangular core

3.2. Construction of a Cap

A cap has three types of faces: T-faces, B-faces, and F-faces, as shown in Figure 10. T-faces become a part of the exterior faces of a hex. B-faces are connected to a core, and F-faces are connected to an adjacent cap. As illustrated in Figure 11, the characteristic of a cap is fully described by three factors:

- the number of subdivision, n_f , of the two edges shared by a T-face and a B-face
- the number of subdivision, n_s , of the four edges shared by a T-face and a F-face
- the type of the subdivision pattern of two T-faces, rectangular or triangular

A cap is either a $n_s \times n_f$ triangular cap or a $n_s \times n_f$ rectangular cap.

Any two caps with the same n_f can be connected each other with F-faces without losing the mesh conformity. Any cap whose n_s is the same as n_1 of the core can be connected to slot 0 or slot 1 of a core. Similarly any cap whose n_s is the same as n_2 of the core can be connected to slot 2 or slot 3. If T-faces have a triangular pattern, a triangular face of a pyramid, tet or prism can be attached to these T-faces. On the other hand if T-faces have a rectangular pattern, the cap a quadrilateral face of a hex or a prism can be attached to these T-faces.

There are five types of caps that are practically most useful, illustrated in Figure 12. Since all caps shown in Figure 12 have $n_f = 4$, any of the caps shown in Figure 12 can be connected to each other without losing the mesh conformity.

3.3. Assembly of a HEXHOOP Template

Having discussed the construction of various caps and cores, we now consider assembling one core and four caps to construct a complete HEXHOOP template for a hex. In order to assemble a core and caps we need following node tables:

- nodes on each slot of the core
- nodes on B-face of a cap
- nodes on F-faces of a cap

We denote the node tables of slot p , $0 \le p \le 3$, as s_i^p , nodes on the B-faces as b_j^p , nodes on the front F-faces as f_k^{pf} and nodes on the back F-faces as f_k^{pb} . n_1 is the number of subdivision of slot 0 and slot 1, and n_2 is the number of subdivision of slot 2 and slot 3. And we also denote n_s^p for n_s of cap p and n_f^p for n_f of cap p .

The node tables are used to generate a core and a cap, and all the nodes must be ordered consistently according to the order illustrated in Figure 13. The length of table s_i^p is $5(n_1+1)$ for $p = 0,1$ and $5(n_2+1)$ for $p = 2,3$. The length of table b_j^p is $5(n_s^p+1)$. The length of f_k^{cf} and f_k^{cb} is $5\left(\frac{n_f^p}{2}-1\right)+4$.

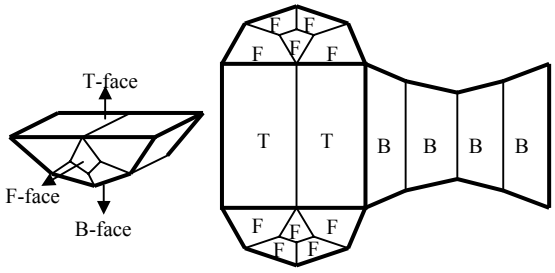


Figure 10 Faces of a Cap

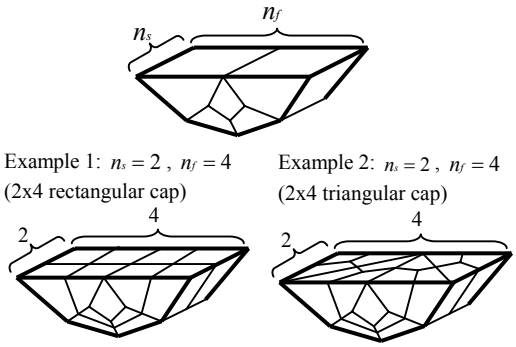


Figure 11 Subdivision of a Cap

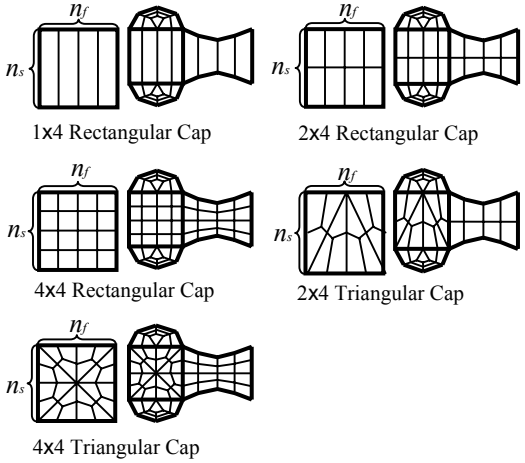


Figure 12 Five practically most useful caps

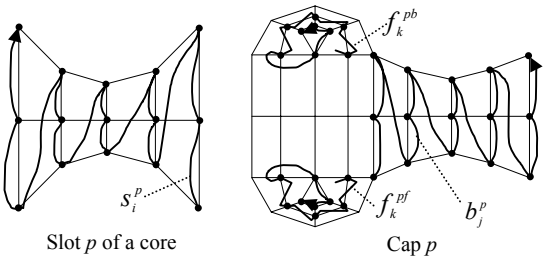


Figure 13 Orders of nodes in node tables

Because cap p is connected to slot p, the following conditions must be satisfied:

$$\text{Condition 1: } n_s^0 = n_s^1 = n_1, \quad n_s^2 = n_s^3 = n_2$$

If the above conditions are not satisfied, the length of table s_i^p does not match the length of table b_j^p, and thus the cap and the slot cannot be connected.

Also, in order to form a hoop of caps, we have to satisfy the following conditions:

$$\text{Condition 2: } n_f^0 = n_f^1 = n_f^2 = n_f^3$$

If the above conditions are not satisfied, the length of tables f_k^{pf} does not match corresponding f_k^{pb} (and vice-versa), and thus caps cannot be connected.

Now we have sufficient information to assemble a HEXHOOP template. After each cap is formed and oriented properly so that it fits its matching slot, nodes are joined together as shown in Figure 14.

We stress again the fact that the interior pattern of T-faces has nothing to do with this assembly process. The combination of caps can therefore be chosen arbitrarily as long as Conditions 1 and 2 are both satisfied.

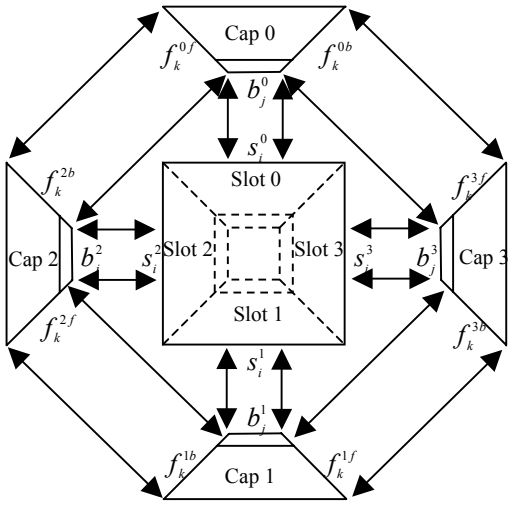


Figure 14 Joining nodes of a core and a cap

In this section we have explained how to generate two types of standard cores, triangular core and rectangular cores, and two types of caps, triangular caps and rectangular caps. By combining these two types of cores and caps we can obtain various all-hex template for a hex. This covers all the ten triangular and rectangular pattern combinations listed in Figure 5 except that shown in Figure 5 (g). This combination of three triangular patterned faces and three rectangular patterned faces require a non-standard core that has one triangular wing face and one rectangular wing face. Section 4.1 discusses how to realize such a core.

4. VARIATIONS OF A CORE

This section discusses three variations of a core. These non-standard cores are required in: (1) covering all the ten combinations of triangular and rectangular patterned faces shown in Figure 5, and (2) generating HEXHOOP templates for a prism.

4.1. Double Hoop Core

As pointed out earlier, a combination of three triangular patterned faces and three rectangular patterned faces, shown in Figure 5 (g), requires a non-standard core with one triangular patterned face and one rectangular patterned face. Figure 15 shows an example of such a template with three 4×4 triangular patterns and three 4×4 rectangular patterns.

In order to make this new core with one triangular wing face and one rectangular wing face, we assemble a HEXHOOP template with a 4×4 rectangular core, a 4×4 triangular cap and three 4×4 rectangular caps. If we assemble and rotate the template 90 degree to the left, we obtain a HEXHOOP template as shown in Figure 16 (a). Only one face of a template has a 4×4 triangular pattern and all other faces have a 4×4 rectangular pattern. Now, we narrow the center of the mesh as shown in Figure 16 (b). The mesh then becomes a 4×4 core with only one wing face 4×4 triangular pattern. We call this core a 4×4 *double hoop core* because two hoops of caps exist in this template.

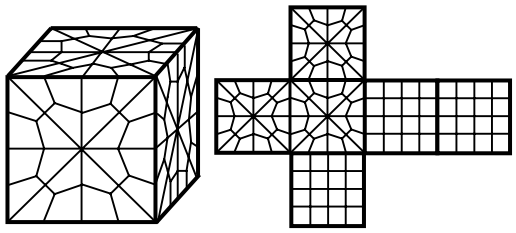


Figure 15 A combination of three 4x4 triangular patterns and three 4x4 rectangular patterns require a non-standard core with one triangular patterned face and one rectangular patterned face.

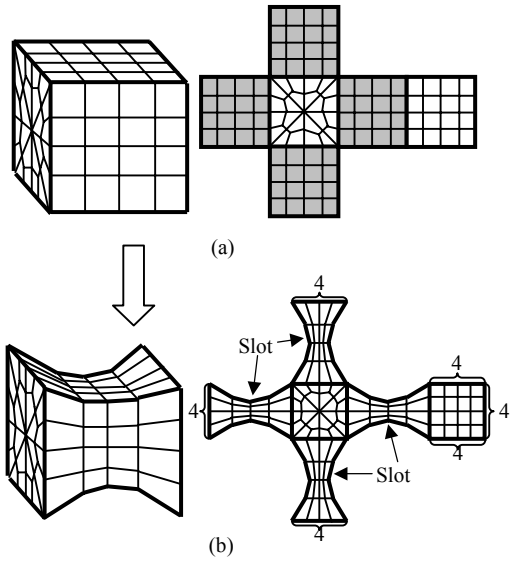


Figure 16 4x4 double hoop core

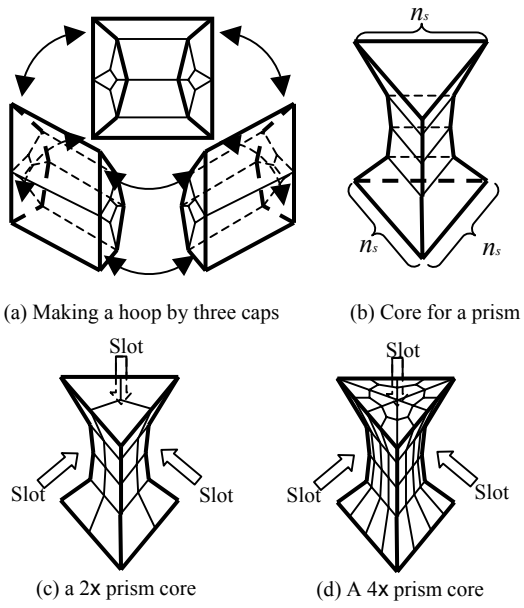


Figure 17 Core for a prism element

4.2. Core for a Prism Element

Up to this point we have discussed HEXHOOP templates for a hex element. HEXHOOP templates can be developed in a similar way for a prism element. Two differences, also illustrated in Figure 17, are: (1) a core's two wing faces for a prism are triangles, and (2) a hoop of caps for a prism consists of only three caps.

Two wing faces can be subdivided in various ways. Figure 17 (c) and Figure 17 (d) show two such examples. By sweeping the two wing faces' subdivision pattern throughout the core, we obtain an all-hex mesh. We can then attach three $4 \times n$ caps to the core's three slots. Like the HEXHOOP templates for a hex, the subdivision pattern of each cap's T-faces can be either triangular or rectangular. We call the core shown in Figure 17 (c) as a $2\times$ *prism core* and the core shown in Figure 17 (d) $4\times$ *prism core*.

5. A SOLUTION TO SCHNEIDERS' OPEN PROBLEM WITH HEXHOOP

In this section we provide a solution to Schneiders' Open Problem [6] using a HEXHOOP template for hex.

We first create a sub-template shown in Figure 18 (a). This is a HEXHOOP template that combines a 2×1 rectangular core, two 1×4 rectangular caps, a 2×4 rectangular cap, and a 2×4 triangular cap.

Consider such a template whose bounding box is (-1.0,-1.0,-1.0) to (1.0,1.0,1.0). We then deform the template by following coordinate transformation:

$$s_x = \frac{|2-y|}{2}$$

$$s_z = \frac{s_x \cdot |x+2|}{3}$$

$$d_y = \frac{1-|x| \cdot s_x}{5}$$

$$x' = x \cdot s_x$$

$$y' = \frac{y+1}{4} + d_y$$

$$z' = z \cdot s_z$$

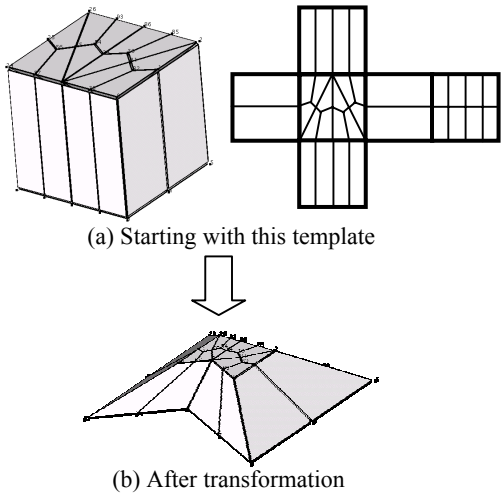


Figure 18 A HEXHOOP template for a hex

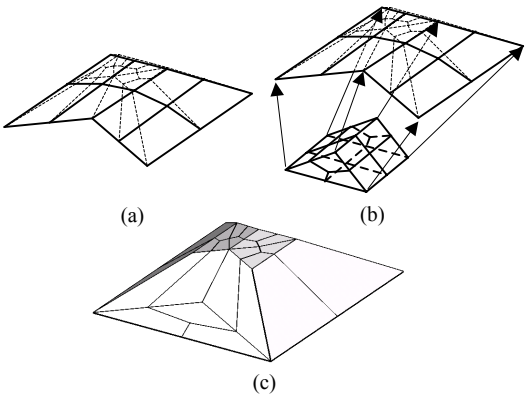


Figure 19 Bottom face of the deformed template

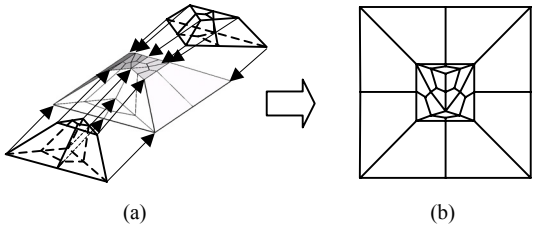


Figure 20 Attaching two diced prisms from the front and from the back

This deformation transforms the HEXHOOP template to the shape shown in Figure 18 (b).

At this point, the bottom face has a 2x4 rectangular pattern as shown in Figure 19 (a); now we attach a diced prism from the bottom as shown in Figure 19 (b). This gives a mesh shown in Figure 19 (c)

After the diced prism is attached to the bottom, the bottom face becomes 2x2 rectangular pattern; this is the pattern that Schneiders' Open Problem demands. Next, we attach two more diced prisms as shown in Figure 20 (a) from the front and from the back.

At this point, if we look at the mesh from the top, the mesh shows the pattern illustrated in Figure 20 (b). The inner square has a triangular pattern. Thus, obviously we can attach six diced tets on the inner square. We make a tet mesh as shown in Figure 21, dice it into hexes, and attach them to the inner square of the current mesh.

Now, the mesh shows the pattern illustrated in Figure 22. Finally, adding four diced prisms from the four sides as shown in Figure 23 gives a solution to Schneiders' Open Problem. Because each interior face is always shared by two hexahedral elements during the above procedure, the mesh is valid.

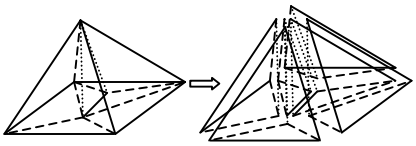


Figure 21 Tet mesh which is attached to the center square

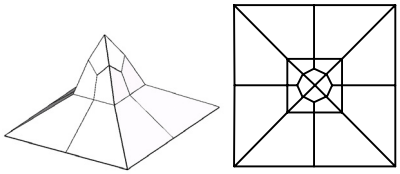


Figure 22 After a diced tet mesh is attached

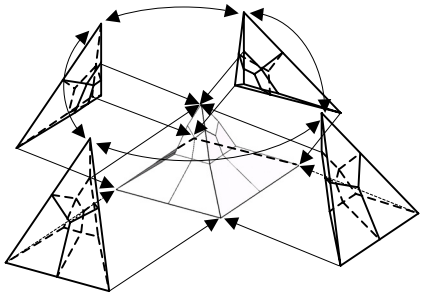


Figure 23 A solution to Rob Schneiders' Open Problem

6. EXAMPLES

This section shows some examples of the conversion from a hex-dominant mesh to an all-hex mesh. The examples shown in this section are tested by a program to confirm that they are topologically valid, and that there are no gaps and overlaps. The qualities of the meshes, however, are not optimized in these examples, because the goal of the HEXHOOP template is to provide templates that create a topologically valid all-hex mesh, and the geometric quality of the mesh must be improved by appropriate post-processes.

Figure 24 (a) shows a hex-dominant mesh, which consists of 3 hexes, 1 prism, 4 pyramids, and 3 tets, and Figure 24 (b) shows a resultant all-hex mesh consisting of 542 hexes. Figure 24 (c) and Figure 24 (d) show cross-sections after the conversion.

The hex element that is adjacent to two pyramids and a hex is replaced with a HEXHOOP template made from a 4x2 rectangular core, a 2x4 triangular cap, a 4x4 triangular cap, a 2x4 rectangular cap, and a 4x4 rectangular cap. The hex element that is adjacent to a pyramid, two hexes, and a prism

is replaced with a HEXHOOP template made from a 4x2 rectangular core, a 2x4 triangular cap, a 2x4 rectangular cap, and two 4x4 rectangular caps. The prism that is adjacent to a hex and a pyramid is replaced with a HEXHOOP template made from a 2x prism core, a 2x4 triangular cap, and two 2x4 rectangular caps. Pyramids and tets are diced accordingly.

Figure 25 (a) shows a hex-dominant mesh of a geometry, made by combining three octagonal prisms, and Figure 25 (b) shows its cross-section. Hexes and prisms that are adjacent to a pyramid are replaced with HEXHOOP templates of $n_1 = n_2 = n_f = 4$, and the other elements are subdivided accordingly. The hex-dominant mesh is converted to the all-hex mesh shown in Figure 25 (c). A cross-section of the all-hex mesh is shown in Figure 25 (d).

Figure 26 (a) shows a hex-dominant mesh of a mechanical part, and Figure 26 (b) shows its cross-section. Again, hexes and prisms that are adjacent to a pyramid are replaced with HEXHOOP templates of $n_1 = n_2 = n_f = 4$, and the other elements are subdivided accordingly. The hex-dominant mesh is converted to the all-hex mesh shown in Figure 26 (c). A cross-section of the all-hex mesh is shown in Figure 26 (d).

7. DISCUSSION

The proposed all-hex conversion templates have two limitations: (1) the number of elements in a final all-hex mesh increases drastically, and (2) the quality of some of the elements in the templates is relatively low.

If $n_1 = n_2 = n_f = 4$ is chosen for all elements in a hex-dominant mesh, the total number of elements in the final all-hex mesh increases to about 60 times more than the number of elements in the original hex-dominant mesh. In order to keep the number of elements in the final all-hex mesh low, the original hex-dominant mesh must be made as coarse as possible. A method for creating such a coarse hex-dominant mesh is one of our future research issues.

Alternatively, different values of n_1, n_2, and n_f can be chosen for each element so that the number of elements in the all-hex mesh remains low. Developing an algorithm for choosing an optimal combination of n_1, n_2, and n_f for each element of a hex-dominant mesh, however, is not trivial and would require future work.

Although the quality of the elements in the templates is relatively low, the quality of the final all-hex mesh could be improved by appropriate post-process such as optimization based smoothing [9]. We are currently working on developing a new mesh-smoothing scheme suitable for an all-hex mesh created with the HEXHOOP template.

It is also important that an input hex-dominant mesh be of reasonably good quality. As in all other template-based methods, if an element in the hex-dominant mesh is highly distorted all the hexes resulting from this element will also be distorted. One possible remedy for this problem is to try limiting the usage of tets and pyramids, for which the hexes in the templates are more distorted, to non-critical regions of a FEM analysis.

8. CONCLUSIONS

We proposed a new modular approach to designing all-hex mesh conversion templates called HEXHOOP. Our method provides a systematic method of generating various templates when triangular and rectangular subdivision patterns are arbitrarily combined on the exterior faces of a template, which could not be realized with previously published methods.

REFERENCES

[1] T. D. Blacker and R. J. Meyers, "Seams and Wedges in Plastering: A 3-D Hexahedral Mesh Generation Algorithm," *Engineering with Computers*, vol. 2, pp. 83-93, 1993.

[2] T. J. Tautges, T. Blacker, and S. A. Mitchell, "The Whisker Weaving Algorithm: A Connectivity-Based Method for Constructing All-Hexahedral Finite Element Meshes," *International Journal for Numerical Methods in Engineering*, vol. 39, pp. 3327-3349, 1996.

[3] S. J. Owen and S. Sagal, "H-Morph: an indirect approach to advancing front hex meshing," *International Journal for Numerical Methods in Engineering*, vol. 49, pp. 289-312, 2000.

[4] S. J. Owen, S. A. Canann, and S. Sagal, "Pyramid Elements for Maintaining Tetrahedra to Hexahedra Conformability," *AMD-Vol. 220 Trend in Unstructured Mesh Generation, ASME*, pp. pp. 123-129, 1997.

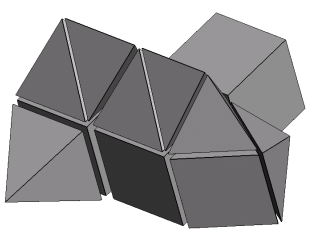
[5] R. Schneiders, "A grid-based algorithm for the generation of hexahedral element meshes," *Engineering with Computers*, vol. 12, pp. 168-177, 1996.

[6] R. Schneiders, "Open Problem, http://www-users.informatik.rwth-aachen.de/~roberts/open.html" .

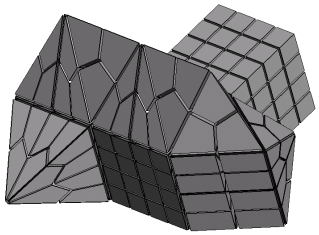
[7] C. D. Carboner, "Constrained Mesh Generation, http://www-users.informatik.rwth-aachen.de/~roberts/SchPyr/index.html" .

[8] S. A. Mitchell, "The All-Hex Geode-Template for Conforming a Diced Tetrahedral Mesh to any Diced Hexahedral Mesh," presented at 7th International Meshing Roundtable, pp. 295-305, 1998.

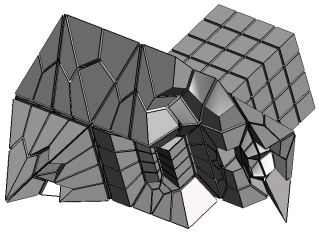
[9] P. M. Knupp, "Hexahedral Mesh Untangling and Algebraic Mesh Quality Metrics," presented at 9th International Meshing Roundtable, pp. 173-183, 2000.



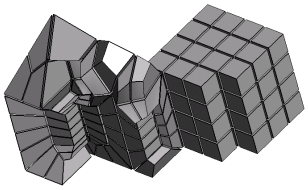
(a) A hex-dominant mesh
(3 hexes, 1 prism, 4 pyramids, and 3 tets)



(b) An all-hex mesh (542 hexes)

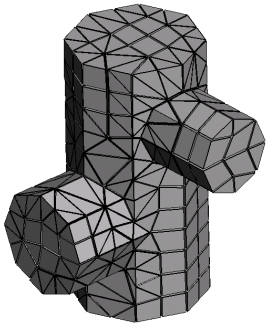


(c) A cross-section of the all-hex mesh

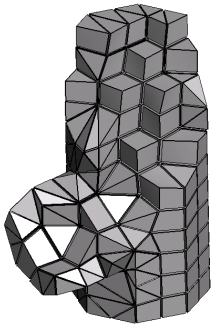


(d) A cross-section of the all-hex mesh

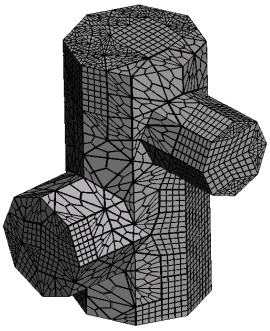
Figure 24 An example of the conversion from a hex-dominant mesh to an all-hex mesh



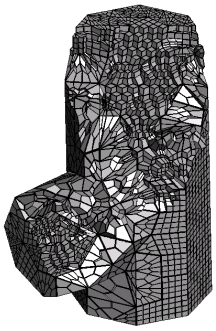
(a) A hex-dominant mesh of an object consisting of three circular bars
(93 hexes, 127 prisms, 127 prisms, and 417 tets)



(b) A cross-section of the hex-dominant mesh

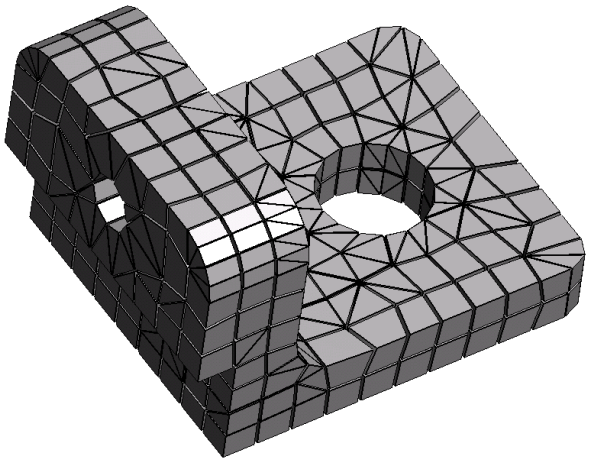


(c) An all-hex mesh, which is converted from (a)
(35,120 hexes)

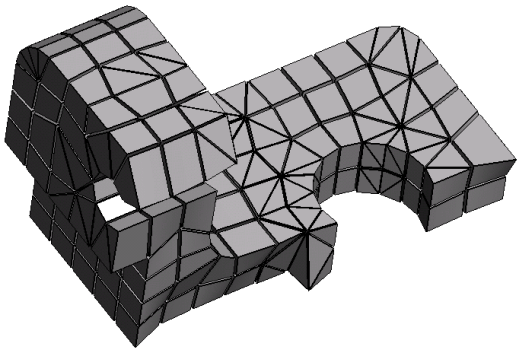


(d) A cross-section of the all-hex mesh

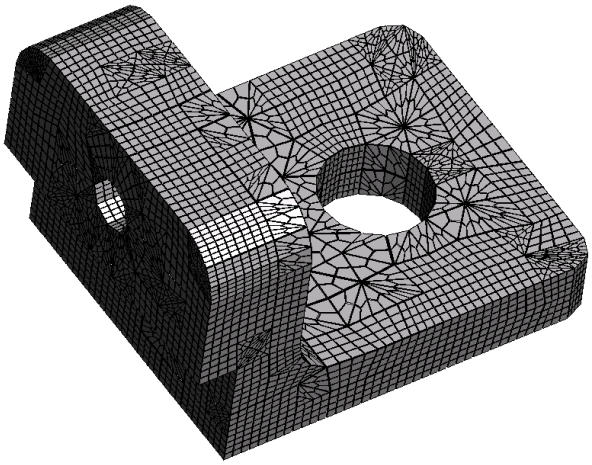
Figure 25 A hex-dominant mesh and an all-hex mesh of an object consisting of three circular bars



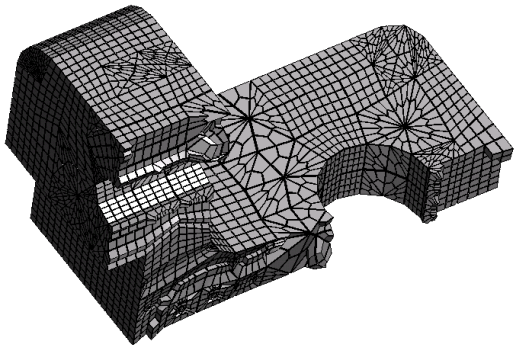
(a) A hex-dominant mesh of a mechanical part
(162 hexes, 137 prisms, 113 pyramids, and 402 tets)



(b) A cross-section of the hex-dominant mesh



(c) An all-hex mesh of the mechanical part
(42,156 hexes)



(d) A cross-section of the all-hex mesh

Figure 26 A hex-dominant mesh and an all-hex mesh of a mechanical part