Hex-Dominant Mesh Generation with Directionality Control via Packing Rectangular Solid Cells

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

A new computational method that creates a hex-dominant mesh of an arbitrary 3D geometric domain is presented. The proposed method generates a high-quality hex-dominant mesh by: (1) controlling the directionality of the output hex-dominant mesh; and (2) avoiding ill-shaped elements induced by nodes located too closely to each other. The proposed method takes a 3D geometric domain as input and creates a hex-dominant mesh that consists of mostly hexahedral elements with additional prism elements and tetrahedral elements. The proposed method packs rectangular solid cells on the boundary of and inside the input domain to obtain ideal node locations for a hex-dominant mesh. Each cell has a potential energy field that mimics a body centered cubic (BCC) structure, and the cells are moved to stable positions by a physically-based simulation. The simulation mimics the formation of a crystal pattern so that the centers of the cells give ideal node locations for a hex-dominant mesh. The domain is then meshed into a tetrahedral mesh by the advancing front method, and finally the tetrahedral mesh is converted to a hex-dominant mesh by merging some tetrahedrons.

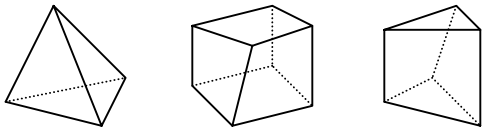
1 Introduction

A new computational method that creates a high-quality hex-dominant mesh of an arbitrary 3D geometric domain is presented. Although several hex-dominant meshing techniques are presented [1, 2], they typically have some limitations such as: the ratio of hexahedral elements is low, or the result heavily depends on the input tetrahedral mesh. The proposed method overcomes these limitations by: (1) controlling the directionality of the output hex-dominant mesh; and (2) avoiding ill-shaped elements induced by nodes located too closely to each other.

The proposed method creates a hex-dominant mesh that consists of hexahedral elements, prism elements and tetrahedral elements as shown in Figure 1, and hexahedral elements and prism elements yield more accurate solution than tetrahedral elements in non-linear finite element analyses since these two types of elements have higher order terms in the shape functions than a tetrahedral element [3]. In the rest of the paper, we simply write *hexes*, *prisms* and *tets* to refer to these three types of elements. Among these three types of elements, a hex has the highest order term, and thus an all-hex mesh, which consists of exclusively hexes, is in great demand in industry. Creating an all-hex mesh of an arbitrary domain is, however, a challenging problem, and no perfect solution is presented. Instead, a hex-dominant mesh is an alternative to an all-hex mesh, and it generally gives a better solution than a tet mesh [4]. A method that converts a hex-dominant mesh to an all-hex mesh is developed [5], and the conversion method requires a high quality hex-dominant mesh to create a high quality all-hex mesh. Thus, it is important to develop a high quality hex-dominant meshing technique.

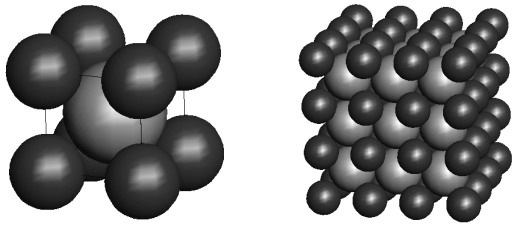
To create a hex-dominant mesh of well-shaped elements, the proposed method obtains node locations by the physical simulation of body centered cubic (BCC) structured particles. The BCC structure is seen in the crystal pattern of some natural substances, such as NaCl, illustrated in Figure 2 (a). If BCC structured cells are packed tightly in the domain, the cells will form a crystal pattern as shown in Figure 2 (b), and the centers of the cells will give ideal node locations for a hex-dominant mesh.

* 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



Tetrahedron Hexahedron Prism

Figure 1 Three types of elements included in the hex-dominant mesh



(a) A BCC structured cell (b) A crystal pattern of BCC structure

Figure 2 BCC structure

The proposed method takes a geometric domain, desired directionality and desired edge length as input and creates a hex-dominant mesh in three steps. First, rectangular solid cells are packed on the boundary of and inside the target geometric domain, and the nodes are created at the centers of the cells. Next, a tetrahedral mesh is created by connecting the nodes obtained in the previous step. Last, some tets are merged to create as many hexes and prisms as possible.

The original idea of this approach is presented by Shimada et al. [6] and Itoh et al. [7]. Their method packs square cells inside a two-dimensional geometric domain to obtain ideal node locations for a quad-dominant mesh. Then the nodes are connected by the Delaunay triangulation method, and a triangular mesh is created. Some pairs of triangles are then merged and converted to quadrilaterals. The method presented in this paper is an expansion of their method to the hex-dominant mesh generation.

The organization of the paper is as follows. Section 2 reviews previous work of hex-dominant mesh generation. Section 3 shows overview of the proposed method. Section 4 explains the representation of desired directionality and edge length. Section 5 explains the algorithm that packs rectangular solid cells in the input domain. Section 6 shows the algorithm that creates a tet mesh. The algorithm that converts a tet mesh to a hex-dominant mesh is presented in Section 7, followed by results in Section 8 and conclusions in Section 9.

2 Previous Work

Owen and Sagal present an algorithm called H-Morph [2], which converts a tet mesh to a hex-dominant mesh by creating hex elements one by one starting from domain boundaries and moving inward. The method always maintains a valid hex-tet mixed mesh during the process. The method, however, extensively modifies the tetrahedrons during the process and tends to yield ill-shaped tetrahedrons.

Meshkat and Talmor present an algorithm that converts a tet mesh into a hex-dominant mesh based on the graph theory [1]. Their method takes a tet mesh as input and creates a graph that represents the topology of the tet mesh. Their method then searches for a pattern that can be converted to a hex or a prism in the graph, and when a pattern is found, a new hex or a new prism is created, and the graph is updated accordingly. The result of their method significantly depends on the input tet mesh.

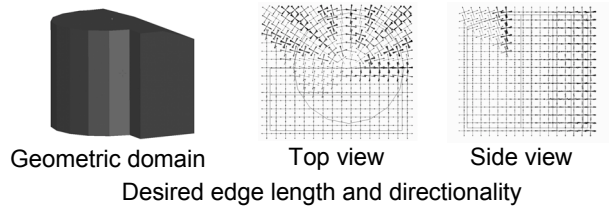
The results of the two methods heavily depend on the input tetrahedral mesh. However, the papers [1, 2] do not present any method that creates an appropriate input tet mesh. And, neither method can control the directionality of the hexahedrons.

3 Overview of the Proposed Method

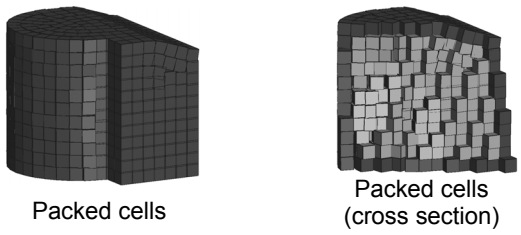
The proposed method takes a 3D geometric domain Ω, a desired directionality and a desired edge length as input and outputs a hex-dominant mesh in three steps as illustrated in Figure 3. In the first step, rectangular solid cells are packed on the boundary of and inside domain Ω. Since the cells form a crystal pattern shown in Figure 2 (b), the centers of the cells give ideal node locations for a hex-dominant mesh. Hex elements are, however, not easy to create directly from the set of nodes obtained in the second step. Instead of creating hex elements directly, the proposed method creates a tet mesh in the second step by the advancing front method [8] and the local transformation method [9], and the tet mesh is then transformed to a hex-dominant mesh by merging tets in the third step.

4 Representation of a Desired Directionality and a Desired Edge Length

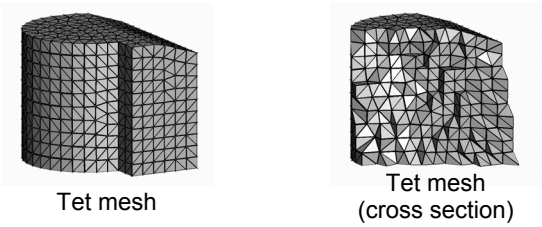
The proposed method takes a desired directionality and a desired edge length as input, as well as the target geometric domain. The desired directionality gives an ideal orientation of a hex at a point on the boundary of and inside the geometric domain, and a 3x3 matrix is sufficient to represent the directionality and the edge length.



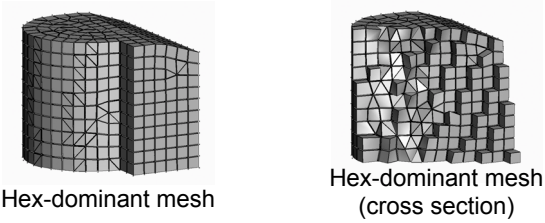
Input geometric domain, desired edge length and directionality



Step 1 Packing rectangular solid cells to obtain ideal node locations of a hex-dominant mesh



Step 2 Creating a tet mesh



Step 3 Converting a tet mesh into a hex-dominant mesh

Figure 3 Overview of the proposed method

An orientation is defined by three orthogonal unit vectors, and a 3x3 rotational matrix is obtained by combining the three vectors. Three orthogonal unit vectors are denoted as $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$. Then, the rotational matrix $\mathbf{R}$ can be written as:

$$\mathbf{R} = \begin{pmatrix} \mathbf{u} & \mathbf{v} & \mathbf{w} \end{pmatrix}, \quad \mathbf{u} = \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}, \quad \mathbf{v} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}, \quad \mathbf{w} = \begin{pmatrix} w_x \\ w_y \\ w_z \end{pmatrix}.$$

Scalar value d represents a desired edge length, and a scaling matrix is constructed as:

$$\mathbf{S} = \begin{pmatrix} d & 0 & 0 \\ 0 & d & 0 \\ 0 & 0 & d \end{pmatrix}.$$

By combining $\mathbf{R}$ and $\mathbf{S}$, a 3x3 matrix, $\mathbf{M} = \mathbf{RS}$, is obtained that represents the directionality and the edge length. Since the directionality and the edge length may vary over the domain, $\mathbf{M}$ is a function of position and can be written as:

$$\mathbf{M(x)} = \mathbf{R(x)S(x)}$$

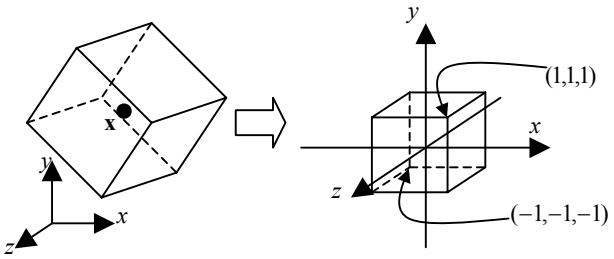
An ideal hex is transformed into a unit cube that aligns with the x, y and z axes and fills the region from (-0.5,-0.5,-0.5) to (0.5,0.5,0.5), by the following transformation:

$$\mathbf{p}_i' = \mathbf{M}^{-1} \cdot (\mathbf{p}_i - \mathbf{x}),$$

where $\mathbf{p}_i$ is a node position of the original hex, and $\mathbf{p}_i'$ is a node position of the unit cube, also illustrated in Figure 4.

In a special case, where a boundary-aligned hex-dominant mesh is required, the directionality is computed based on the normal vector of the boundary surface and the tangential vector of the boundary edges. Since hexes must be aligned with the boundary in this special case, principal vectors are constrained by two conditions as follows.

- On the boundary edges, one of three principal vectors must be parallel to the tangential direction of the edge.
- On the boundary surfaces, one of three principal vectors must be parallel to the normal direction of the surface.



(a) An ideal hexahedron (b) A unit cube

Figure 4 Transformation from an ideal hexahedron to a unit cube

To satisfy these two conditions, the first principal vector and the second principal vector at point $\mathbf{x}$, $\mathbf{u(x)}$ and $\mathbf{v(x)}$, are defined as follows:

(1) $\mathbf{u(x)}$ is equal to the unit normal vector at the point on the boundary surface that is closest to $\mathbf{x}$.

(2) $\mathbf{v(x)}$ is computed based on the tangential vector of a boundary edge. Let $\mathbf{x}_e$ be a point on the boundary edge that satisfies $\cos(3\pi/4) < \mathbf{t}_e(\mathbf{x}_e) \cdot \mathbf{u}(\mathbf{x}) < \cos(\pi/4)$, where $\mathbf{t}_e(\mathbf{x}_e)$ is the unit tangential vector of the boundary edge at $\mathbf{x}_e$. Hence, the tangential vector at $\mathbf{x}_e$ always makes more than 45 degrees and less than 135 degrees with $\mathbf{u(x)}$. Now $\mathbf{x}_v$ is chosen from all

possible $\mathbf{x}_e$ so that the distance between $\mathbf{x}_v$ and $\mathbf{x}$ is minimum. Then, $\mathbf{v}(\mathbf{x})$ is computed as:

$$\mathbf{v}(\mathbf{x}) = \frac{(\mathbf{u}(\mathbf{x}) \times (\mathbf{t}_e(\mathbf{x}_v)) \times \mathbf{u}(\mathbf{x})}{\left\| (\mathbf{u}(\mathbf{x}) \times (\mathbf{t}_e(\mathbf{x}_v)) \times \mathbf{u}(\mathbf{x}) \right\|}$$

The third principal vector $\mathbf{w}(\mathbf{x})$ is simply computed by taking cross-product of $\mathbf{u}(\mathbf{x})$ and $\mathbf{v}(\mathbf{x})$. Although there is apparent ambiguity in the choice of the first principal vector on the medial surface of the boundary because the distances from more than one surface to the point on the medial surface are equal, the ambiguity only affects the region near the medial surface of the volume, and it does not much affect the overall outcome of the whole meshing process.

5 Packing Rectangular Solid Cells on the Boundary of and Inside the Domain

In the first step of the proposed method, rectangular solid cells are packed on the boundary of and inside the geometric domain. During the process, rectangular solid cells are created in the domain and moved to stable positions by physically-based particle simulation. If appropriate number of cells with respect to the given directionality and edge length $\mathbf{M}(\mathbf{x})$ are closely packed in the domain, the cells form a crystal pattern as shown in Figure 2 (b). The centers of the cells will give ideal node locations for a hex-dominant mesh because they mimic the Voronoi polyhedra of an ideal hex-dominant mesh.

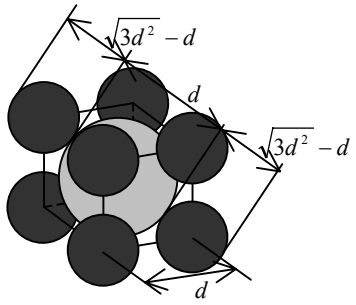


Figure 5 Diameters of the bubbles

To run the simulation, an equation of motion that governs the motion of the cells must be derived. The equation of motion is written as:

$$m\ddot{\mathbf{x}} = \sum \mathbf{f} ,$$

where m is a mass of the cell, $\mathbf{x}$ is a position of the cell, and $\sum \mathbf{f}$ is total force acting on the cell. A cell receives two types of forces, a force based on the proximity of the cells called inter-bubble force and a damping force.

Inter-bubble force is computed based on the proximity between spheres, or bubbles, that are making a body centered cubic (BCC) structure located at the center and at the eight corners of rectangular solid cells. A bubble at the center of the cell is called a main bubble, and bubbles at the eight corners of the cell are called sub-bubbles. A sub-bubble produces forces against main bubbles, but never produces a force against other sub-bubbles. Also, sub-bubbles never receive a force. If the length of an edge of a rectangular solid cell is d, the main bubble has diameter of d, and thus the main bubble touches each face at the centers of the faces. A sub-bubble has diameter of $\sqrt{3d^2} - d$, and thus it touches the main bubble at one point, also illustrated in Figure 5.

An inter-bubble forces acts between two adjacent bubbles, and the inter-bubble force acting on a rectangular solid cell is computed by summing up inter-bubble forces acting on the main bubble of the cell. If main bubble A located at $\mathbf{x}_A$ and bubble B located at $\mathbf{x}_B$ are adjacent to each other, and if r_A and r_B are the radii of A and B, the magnitude of inter-bubble force acting between two bubbles is calculated by a cubic function as:

$$f(w) = \begin{cases} k(1.25w^3 - 2.375w^2 + 1.125) & \text{if } 0 \le w \le 1.5 \\ 0 & \text{otherwise} \end{cases},$$

where $w = \dfrac{\left\| \mathbf{x}_B - \mathbf{x}_A \right\|}{r_A + r_B}$, $\|\cdot\|$ is the Euclidian norm, and k is a spring constant. The plot of $f(w)$ is also shown in Figure 6. And the direction of the inter-bubble force acting on A is given as a unit vector:

$$\mathbf{f}_{BA} = \frac{\mathbf{x}_A - \mathbf{x}_B}{\left\| \mathbf{x}_A - \mathbf{x}_B \right\|} .$$

As a result, the inter-bubble force acting on A by the interaction between A and B is calculated as:

$$\mathbf{F}_{BA} = f(w)\mathbf{f}_{BA} .$$

Notice that $f(w)$ can be either negative or positive. If $f(w)$ is positive, two bubbles repel each other, or if it is negative, two bubbles attract each other. Since more than one bubble can be adjacent to bubble A, the total inter-bubble force acting on A is computed as follows:

$$\mathbf{F} = \sum_i \mathbf{F}_{iA} ,$$

where i is ith bubble that is adjacent to A.

By defining a mass of a rectangular solid cell m and damping force $-c\dot{\mathbf{x}}$, the equation of motion of the cell is written as:

$$m\ddot{\mathbf{x}} = \mathbf{F} - c\dot{\mathbf{x}} ,$$

where $\mathbf{x}$ is the position of the main bubble, which is equal to the position of the rectangular solid cell. The equation is

solved by an iterative numerical integration scheme such as Euler's method or the 4th order Runge-Kutta method. In each iteration, the orientations of the rectangular solid cells are updated based on the vector field generated in the first step of the process.

It should be noted that the locations of the cells give good node locations for a hex-dominant mesh only if the appropriate number of cells are packed in the domain, and thus some cells should be added or deleted in the domain during the simulation to adjust the number of cells. Since the precise number of cells is often impossible to compute when the input domain is complex, the number of cells must be adjusted adaptively during the simulation based on the population density of the cells. In the current implementation, the program finds regions where cells are too sparse or too crowded, based on the distances between adjacent cells, and adds some cells to sparse regions and deletes some cells from crowded region.

Because this simulation mimics the formation of a BCC crystal pattern shown in Figure 2 (b), the centers of the rectangular solid cells tend to form a structured orthogonal grid pattern, which is appropriate for a hex-dominant mesh. Although the pattern will not be well structured in a region where $\mathbf{M}(\mathbf{x})$ changes drastically, those cases are usually local, and most node locations are well structured.

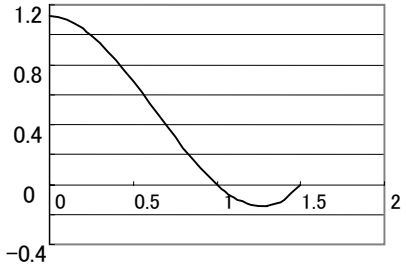


Figure 6 Plot of function $f(w)$

The simulation result also depends on the initial locations of the cells. In the current implementation, the cells are initially placed on the points of a grid that covers the bounding box of the target domain. Although the current initial cell placement method is satisfactory, developing a further improved initial placement method is one of the future research issues.

6 Generating a Tet Mesh

After the nodes are created by the rectangular solid cell packing method, the nodes are connected to form a tet mesh. The purpose of creating a tet mesh instead of creating a hex-dominant mesh directly is to avoid potential numerical instability caused by the computation of intersections between the bi-linear surfaces that are included in hex elements and prism elements. No two elements should intersect during the mesh generation process because any intersection between elements will make the mesh invalid. However, checking the intersection between elements that include quadrilateral faces potentially makes the computation unstable. The geometry of a quadrilateral face is a bi-linear surface, and a non-linear equation-solving scheme, such as Newton's method, is needed to find an intersection between bi-linear surfaces. Such a non-linear equation-solving scheme may become unstable, making the result of the intersection check inaccurate. In the worst case, two elements may intersect, and this makes the entire mesh invalid.

On the other hand, a tet element has only triangular faces. Since the geometries of the triangular faces are planar, an intersection between tet elements can be detected easily by solving linear equations. Hence, creating tet elements is easier than creating hex elements and prism elements in terms of robustness. In addition, if no two tet elements are intersecting, merging some tet elements to create hex elements and prism elements will not create a new intersection. It is thus reasonable to create a hex-dominant mesh in two steps: (1) creating a tet mesh, and (2) merging some tet elements to create hex elements and prism elements.

A tet mesh is created by the advancing front method presented in the reference [8]. Unlike typical advancing front method [10, 11], no new nodes are added inside the domain during this process because nodes are already created by the rectangular cell packing method.

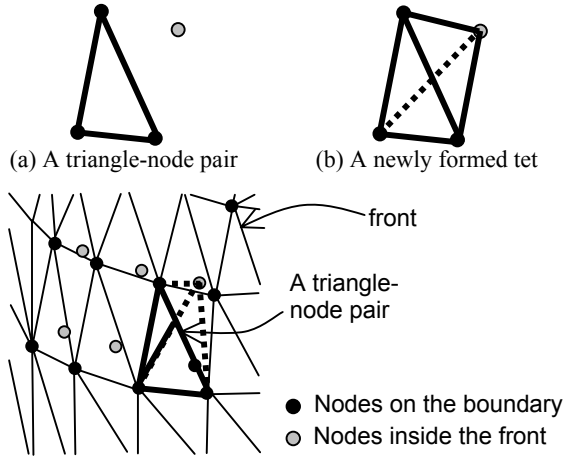


Figure 7 A front and a triangle-node pair

To run the advancing front method, the boundary of the domain is first meshed into a triangular mesh called a front. The program then searches a triangle-node pair that yields a tetrahedron that satisfies the Delaunay criterion, or circumsphere criterion [12, 13], as shown in Figure 7 (a). When a triangle-node pair is found, a tetrahedron is created by connecting the node to the triangle as shown in Figure 7

(b). Also the front is updated so the newly created tetrahedron is excluded by the front as shown in Figure 7 (c). Since the volume of the newly created tetrahedron is taken off from the volume enclosed by the front, the front shrinks every time a tetrahedron is created. Repeating this process until the front disappears, the program meshes the domain into a tet mesh.

After the tet mesh is created, the mesh quality is improved by a method called local transformation [9]. This post-process is required because the Delaunay criterion alone cannot avoid one particular type of ill-shaped tetrahedrons known as slivers. The local transformation method eliminates most slivers.

Another possible option of creating a tet mesh is to apply the Delaunay triangulation method [14, 15]. The Delaunay triangulation method is widely used to create a tetrahedral mesh. However, edges in the tetrahedral mesh created by the Delaunay triangulation method often intersect with the original boundary unless the geometry is convex, and those intersections must be removed in a boundary recovery post-process, which often adds new nodes. Since the nodes created in the rectangular solid cell packing are ideally located, adding a new node can worsen the quality of the output hex-dominant mesh, and the advancing front method is thus suitable for the work presented in this paper.

7 Converting a Tetrahedral Mesh to a Hex-Dominant Mesh

After the tet mesh is created, the proposed method merges some tets to create hexes and prisms. The method first creates a list of node combinations that form hexes, and the combinations are then sorted by the shape quality of the hexes so that the node combinations that yield the superior quality hex are listed first. Tets are merged and converted to hexes one by one based on the list. After exhausting all the possible hexes, the proposed method creates a list of node combinations that makes prisms. The combinations are then sorted by the quality of the prisms so that the node combination that yields the best quality prism comes first and the worst quality last. Some tets are then merged and converted to prisms based on the list.

7.1 Definition of Scaled Jacobian

During the conversion process, the proposed method measures the quality of an element by a scaled Jacobian [16]. If element e is a tet, a prism or a hex, and if nodes $\mathbf{n}$, $\mathbf{m}_{e1}$, $\mathbf{m}_{e2}$ and $\mathbf{m}_{e3}$ are four nodes of element e as shown in Figure 8, the scaled Jacobian of element e at node $\mathbf{n}$, $Js_e(\mathbf{n})$, is defined as:

$$Js_e(\mathbf{n}) = \frac{(\mathbf{m}_{e1}-\mathbf{n})\cdot(\mathbf{m}_{e2}-\mathbf{n})\times(\mathbf{m}_{e3}-\mathbf{n})}{\|\mathbf{m}_{e1}-\mathbf{n}\|\,\|\mathbf{m}_{e2}-\mathbf{n}\|\,\|\mathbf{m}_{e3}-\mathbf{n}\|}$$

Nodes $\mathbf{m}_{e1}$, $\mathbf{m}_{e2}$ and $\mathbf{m}_{e3}$ are ordered so that $Js_e(\mathbf{n}) = 1$ when element e is a hex and its geometry is a cube. The value of $Js_e(\mathbf{n})$ takes the maximum, 1.0, only when the element satisfies two conditions: (1) edges $\mathbf{n}-\mathbf{m}_{e1}$, $\mathbf{n}-\mathbf{m}_{e2}$ and $\mathbf{n}-\mathbf{m}_{e3}$ are perpendicular to each other, and (2) the element is not inverted at node $\mathbf{n}$. Since all angles of the quadrilateral faces of a hex must be as close to 90 degrees as possible, $Js_e(\mathbf{n})$ of a hex must be as large as possible. If $Js_e(\mathbf{n})$ is close to zero, it indicates that element e is flat at $\mathbf{n}$, and if $Js_e(\mathbf{n})$ is negative, element e is inverted at node $\mathbf{n}$. Note that $Js_e(\mathbf{n})$ represents quality at only one point and is not sufficient to measure the quality of the element. A scaled Jacobian is thus measured at every node of the element, and the minimum one, called a minimum scaled Jacobian, is used as a quality measure of the element. The value of a minimum Scaled Jacobian ranges from -1.0 to 1.0 for a hex, from –0.866 to 0.866 for a prism, and from –0.707 to 0.707 for a tet.

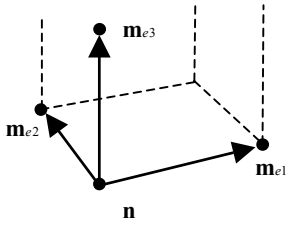


Figure 8 Three vectors used in computing the scaled Jacobian at node n

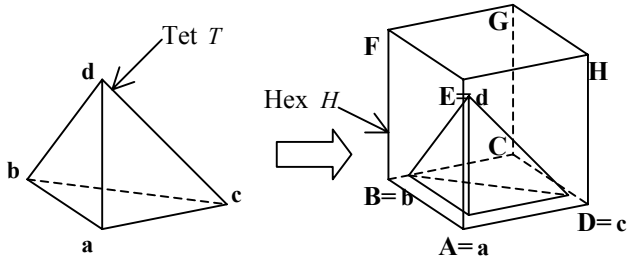
7.2 Finding Node Combinations that Make Hexes

The proposed method makes a list of possible node combinations that can each be converted to a hex. Each entry of the list stores eight nodes that will become eight nodes of a hex. To facilitate the explanation, a hex to be created is denoted as H, and the eight corners of hex H are denoted as **A** through **H**, and **ABCD**, **ADHE**, **BAEF**, **CBFG**, **DCGH** and **FEHG** are the six quadrilateral faces of hex H (see Figure 9). Nodes in the tet mesh are assigned to **A** through **H** to make a node combination.

The proposed method searches node combinations based on the two possible tet configurations: (Pattern 1) three edges of a tet become three edges of a hex as shown in Figure 9 (b), and (Pattern 2) six tets are sharing an edge that will become a diagonal of the hex as shown in Figure 10 (b).

The proposed method searches node combinations of Pattern 1 based on the assumption that four nodes of a tet become **A**, **B**, **D** and **E** of hex H, and nodes to be assigned

for **C**, **F**, **G** and **H** will be found by the edges of the tet mesh. Let T be a tet, and **a**, **b**, **c** and **d** are the four nodes of tet T satisfying $J_{ST}(\mathbf{a}) \geq J_{ST}(\mathbf{b}), J_{ST}(\mathbf{c}), J_{ST}(\mathbf{d})$ and $(\mathbf{c}-\mathbf{a})\times(\mathbf{b}-\mathbf{a})\cdot(\mathbf{d}-\mathbf{a})>0$ as shown in Figure 9 (a). The proposed method assumes that three edges of T, **a - b**, **a - c** and **a - d**, will become three edges of hex H as shown in Figure 9 (b). If node **a** is assigned to **A**, nodes **b**, **c** and **d** can be assigned to **B**, **D** and **E**, and edges **a - b**, **a - c** and **a - d** will become **AB**, **AD** and **AE**. Although (**b**, **c**, **d**) can also be assigned to (**D**, **E**, **B**) or (**E**, **B**, **D**), the result will be equivalent. The candidate nodes for **F** are then limited to the nodes directly connected to **b** and **d**, and the candidate nodes for **H** are limited to the nodes directly connected to **c** and **d**, and the candidate nodes for **C** are limited to the nodes directly connected to **b** and **c**. One of the candidates for **F** is denoted as **p**, for **C** is denoted as **q**, and for **H** is denoted as **r**. For each choice of **p**, **q** and **r**, the candidates for **G** are the nodes directly connected to nodes **p**, **q** and **r**, and one of the candidates for **G** is denoted as **s**. To avoid degeneracy, it is important to assure that no two nodes of **a**, **b**, **c**, **d**, **p**, **q**, **r**, and **s** are identical. The proposed method checks every possible choices of **p**, **q**, **r**, and **s**, and if node combination (**A**, **B**, **C**, **D**, **E**, **F**, **G**, **H**)=(**a**, **b**, **q**, **c**, **d**, **p**, **s**, **r**) makes a minimum scaled Jacobian of H positive, the node combination is added to the node combination list. It must be noticed that a scaled Jacobian $J_{ST}(\mathbf{a})$ is inherited to $J_{SH}(\mathbf{A})$, and since node **a** is chosen so that it satisfies $J_{ST}(\mathbf{a}) \geq J_{ST}(\mathbf{b}), J_{ST}(\mathbf{c}), J_{ST}(\mathbf{d})$, the largest possible scaled Jacobian of tet T, $J_{ST}(\mathbf{a})$, is inherited to hex H. The proposed method searches node combinations of Pattern 1 for every tet, and all node combinations found by the search are added in the node combination list.



(a) A tet in the tetrahedral mesh

(b) Assigning the four nodes of the tet

Figure 9 Finding node combinations based on Pattern 1

Although the program searches node combinations exhaustively for each tet, the computational time to search combinations for a tet is nearly constant because the proposed method only visits nodes that are less-than-three edges away from the tet, and the number of the nodes that are within less-than-three edges away from the tet is almost constant. The computational complexity for whole tet mesh is thus order of n, where n is number of tets included in the tet mesh.

The proposed method searches node combinations of Pattern 2 based on the assumption that the longest edge of a tet becomes a diagonal of a hex, and the assumption will narrow down the possible node combinations into two, and one of the two node combinations that yields better minimum scaled Jacobian will be added to the node combination list. Let T be a tet, and **a**, **b**, **c** and **d** be nodes of tet T satisfying $\|\mathbf{b}-\mathbf{d}\| \geq \|\mathbf{a}-\mathbf{b}\|, \|\mathbf{a}-\mathbf{c}\|, \|\mathbf{a}-\mathbf{d}\|, \|\mathbf{b}-\mathbf{c}\|, \|\mathbf{c}-\mathbf{d}\|$ and $(\mathbf{c}-\mathbf{a})\times(\mathbf{b}-\mathbf{a})\cdot(\mathbf{d}-\mathbf{a})>0$ as shown in Figure 10 (a). The proposed method assumes that edge **b - d** will become a diagonal of the hex, because the diagonal of a well-shaped hex is always longer than any edges of the hex and any diagonals of the faces of the hex. If the number of tets sharing edge **b - d** is not six, the method moves on to next T and starts over again. If the number of tets shareing edge **b - d** is six, the proposed method then makes a list of six tets sharing edge **b - d** as shown in Figure 10 (b). The edges included in the six tets and not connected to nodes **b** and **d** make a loop as shown in Figure 10 (c), and four nodes as well as nodes **a** and **c** are included in the loop and denoted as **p**, **q**, **r** and **s**, and the order of the nodes in the loop is (**a**, **c**, **p**, **q**, **r**, **s**). At this point, there are still two possible node combinations. One is (**A**, **B**, **C**, **D**, **E**, **F**, **G**, **H**)=(**a**, **s**, **b**, **c**, **d**, **r**, **q**, **p**) as shown in Figure 10 (d), and the other is (**A**, **B**, **C**, **D**, **E**, **F**, **G**, **H**)=(**a**, **b**, **p**, **c**, **s**, **r**, **q**, **d**) as shown in Figure 10 (e). Both combinations are topologically correct, and hence the geometric quality must be taken into account to choose one of the two possible combinations. The proposed method computes a minimum scaled Jacobian of the hexes created by both node combinations, and the one that yields a larger minimum scaled Jacobian is added to the node combination list. In the case shown in Figure 10, the combination shown in Figure 10 (d) clearly yields better quality and thus will be taken. The proposed method searches node combinations of Pattern 2 for every tet, and all the node combinations found by the search are added in the node combination list.

7.3 Creating Hexes Based on the Node Combination List

After the node combination list is created, the entries of the list are sorted by the minimum scaled Jacobian of hexes to be created so that the entry of the largest minimum scaled Jacobian comes first, and the entry of the smallest minimum Jacobian last. The proposed method then

attempts to create hexes based on the sorted node combination list.

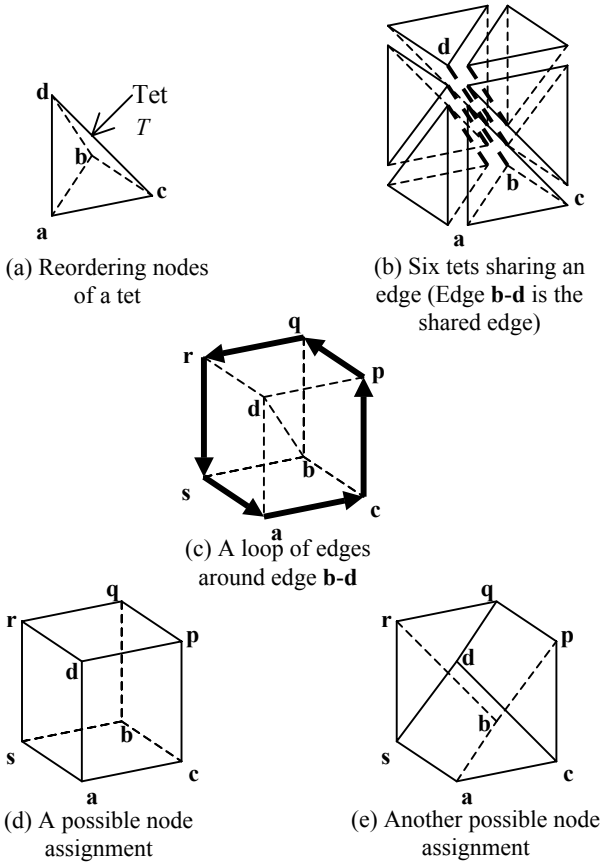


Figure 10 Finding node combinations based on Pattern 2

For each node combination, consisting of eight nodes denoted as (**A**, **B**, **C**, **D**, **E**, **F**, **G**, **H**), the proposed method attempts to create hex H, which consists of the eight nodes, by merging tets that each consists of four of the eight nodes. The node combination, however, is discarded if it does not satisfy following two conditions: (1) creating hex H does not induce gaps or overlaps, and (2) quadrilateral faces of H are compatible with quadrilateral faces of adjacent hexes.

Condition (1) is not satisfied when some tets that were using four of the eight nodes no longer exist because they already became a part of another hex. To enforce condition (1), the proposed method creates a polyhedron by merging the tets that each consists of four of the eight nodes as shown in Figure 11. If the number of triangles included in the polyhedron is not 12 or if every quadrilateral face of hex H does not correspond to a unique pair of adjacent triangles of the polyhedron, the node combination is discarded.

Condition (2) is not satisfied when a quadrilateral face of hex H shares three of four nodes of a quadrilateral face of an already-created hex as shown in Figure 12. To enforce condition (2), the proposed method checks six quadrilateral faces of hex H, **ABCD**, **ADHE**, **BAEF**, **CBFG**, **DCGH** and **FEHG**. If a quadrilateral face of hex H shares three of four nodes of a quadrilateral face of the adjacent hex, the node combination is discarded.

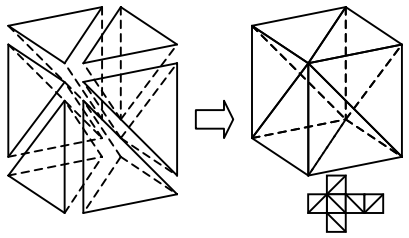


Figure 11 A polyhedron that will become a hex

If the two conditions are satisfied, the proposed method deletes the tets that each consists of four of the eight nodes of the node combination, and hex H is added to the mesh.

It must be noticed that if an ill-shaped tetrahedron, known as a sliver, consists of four nodes of a node combination as shown in Figure 13, the sliver will be deleted when a hex is created from the node combination. Although slivers are mostly eliminated by local transformation [9], some slivers may remain even after the local transformation, and a remaining one typically consists of four nodes that will become a quadrilateral face of a hex or a prism. The conversion process thus eliminates many of remaining slivers, and the quality of the mesh is improved.

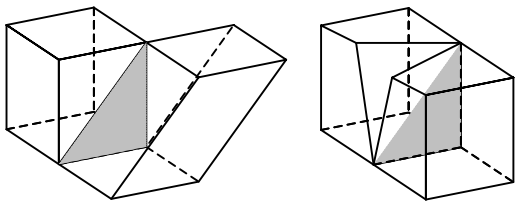


Figure 12 Two quadrilaterals sharing three nodes

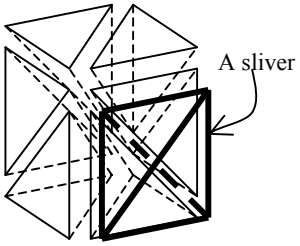
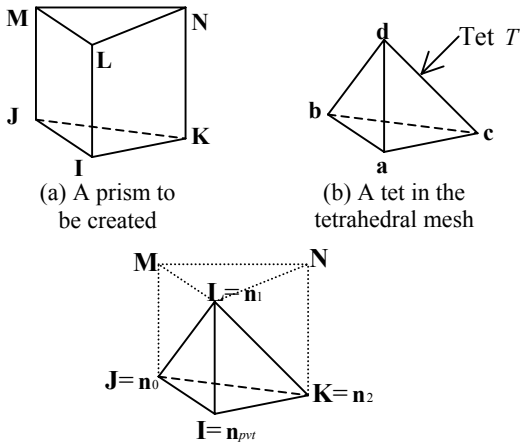


Figure 13 A sliver included in a node combination

7.4 Finding Node Combinations for Prisms

After creating hexes, the proposed method makes a list of node combinations that can each be a prism. Each entry has six nodes that will become nodes of a prism. To facilitate the explanation, the prism to be created is denoted as P, and nodes of P are denoted as (**I**, **J**, **K**, **L**, **M**, **N**). **IJK** and **MLN** are the two triangles of the prism, and **ILMJ**, **IKNL** and **IMNK** are the three quadrilaterals of the prism as shown in Figure 14 (a). Since some tets are already converted to hexes, the mesh that used to be a tet mesh is no longer a tet mesh, but it is already a hex-dominant mesh with no prism.



(a) A prism to be created

(b) A tet in the tetrahedral mesh

(c) Node assignments for **I**, **J**, **K** and **L**

Figure 14 Finding node combinations for a prism

The proposed method assumes that four nodes of a tet will become **I**, **J**, **K** and **L**, and candidate nodes to be assigned to **M** and **N** are searched by the edges in the mesh. Let T be a remaining tet, and **a**, **b**, **c** and **d** be four nodes of tet T as shown in Figure 14 (b). And node $\mathbf{n}_{pvt}$ is one of **a**, **b**, **c** and **d**, and $\mathbf{n}_0$, $\mathbf{n}_1$ and $\mathbf{n}_2$ are the other three nodes of the tet satisfying $(\mathbf{n}_1-\mathbf{n}_0)\times(\mathbf{n}_2-\mathbf{n}_0)\cdot(\mathbf{n}_{pvt}-\mathbf{n}_0)>0$. There are four possible choices of $\mathbf{n}_{pvt}$, and for each choice of $\mathbf{n}_{pvt}$ there are three different choices of $\mathbf{n}_0$, $\mathbf{n}_1$ and $\mathbf{n}_2$. For example, if $\mathbf{n}_{pvt}=\mathbf{a}$, ($\mathbf{n}_0$, $\mathbf{n}_1$, $\mathbf{n}_2$) can be (**b**, **c**, **d**), or (**c**, **d**, **b**), or (**d**, **b**, **c**). For each choice of $\mathbf{n}_{pvt}$, $\mathbf{n}_0$, $\mathbf{n}_1$ and $\mathbf{n}_2$, the proposed method assumes **I**= $\mathbf{n}_{pvt}$, **J**= $\mathbf{n}_0$, **K**= $\mathbf{n}_1$, and **L**= $\mathbf{n}_2$ as shown in Figure 14 (c), and nodes to be assigned to **M** and **N** are searched by the edges connected to $\mathbf{n}_0$, $\mathbf{n}_1$ and $\mathbf{n}_2$. The candidate nodes for **M** is limited to the nodes directly connected to $\mathbf{n}_0$ and $\mathbf{n}_1$, and the candidate nodes for **N** are the nodes directly connected to $\mathbf{n}_1$ and $\mathbf{n}_2$. One of the candidate nodes for **M** is denoted as **p**, and one of the candidate nodes for **N** is denoted as **q**. For every pair of **p** and **q** that is directly connected to each other, the proposed method computes the minimum scaled Jacobian of the prism made by node combination (**I**, **J**, **K**, **L**, **M**, **N**)=($\mathbf{n}_{pvt}$, $\mathbf{n}_0$, $\mathbf{n}_1$, $\mathbf{n}_2$, **p**, **q**). If the minimum scaled Jacobian is positive, the node combination is added to the node combination list. The proposed method searches node combinations for every remaining tet, and all node combinations found by the search are added to the node combination list.

7.5 Creating Prisms Based on the Node Combination List

After the node combination list is created, the entries of the list are sorted by the minimum scaled Jacobian of prisms to be created so that the entry of the largest minimum scaled Jacobian comes first, and the entry of the smallest minimum Jacobian last. The proposed method then attempts to create prisms based on the sorted node combination list.

For each node combination, consisting of six nodes denoted as (**I**, **J**, **K**, **L**, **M**, **N**), the proposed method attempts to create prism P, which consists of the six nodes, by merging tets that each consists of four of the six nodes. The node combination, however, is discarded if it does not satisfy the following two conditions: (1) creating hex P does not induce gaps or overlaps, and (2) quadrilateral faces of P are compatible with quadrilateral faces of adjacent hexes and prisms.

Condition (1) is not satisfied when some tets that were using four of the six nodes no longer exist because they are already became a part of another hex or prism. To enforce condition (1), the proposed method creates a polyhedron by merging the tets that each consists of four of the six nodes as shown in Figure 15. If the number of triangles included in the polyhedron is not 8, or if every quadrilateral face of prism P does not correspond to a unique pair of adjacent triangles of the polyhedron, the node combination is discarded.

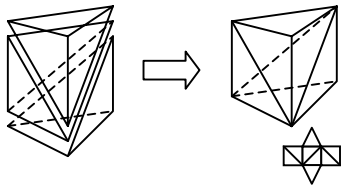


Figure 15 A polyhedron that will become a prism

Condition (2) is not satisfied when a quadrilateral face of prism P shares three of four nodes of a quadrilateral face of an already-created hex or prism as shown in Figure 12. To enforce condition (2), the proposed method checks three quadrilateral faces of prism P, **JILM**, **KJMN** and **IKNL**. If a quadrilateral face of prism P shares three of four nodes of a quadrilateral face of the adjacent hex or a prism, the node combination is discarded.

If the two conditions are satisfied, the proposed method deletes the tets that each consists of four of the six node of the node combination, and prism P is added to the mesh.

8 Results

This section presents some results of the proposed method. For each result, the ratio of the number of hexes, prisms and tets against the total number of elements, ratio of the volume of hexes, prisms and tets against the total volume of all elements, and quality of the elements are presented. Quality of hexes and prisms are presented by the histogram of the minimum scaled Jacobian, and the quality of tets is presented by the histogram of radius-ratio, which is the radius of circumscribed sphere divided by the radius of inscribed sphere. For an equilateral tetrahedron, which is a perfectly well-shaped tetrahedron, the radius-ratio becomes 3.0, and it becomes larger as the shape of the element becomes worse.

Figure 16 (d) and Figure 16 (e) show a hex-dominant mesh of a mechanical part with a cylindrical feature, and Figure 16 (a) shows the input geometric domain and the directionality. The packing of rectangular solid cells is shown in Figure 16 (b) and Figure 16 (c). The picture clearly shows that hexes on the boundary are aligned with the given directionality. The histogram of the radius-ratio of tets, the scaled Jacobian of hexes and the scaled Jacobian of prisms are shown in Figure 16 (f), Figure 16 (g), and Figure 16 (h). These histograms indicate that tets are well-shaped because most of them have radius-ratios of 3.0 to 4.0. Hexes and prisms are also well-shaped because the peaks of the histograms of hexes and prisms are located near the maximum value that the scaled Jacobian can take. Although the number of tets is taking 50% of the total number of elements, hexes and prisms are taking more than 89% of the total volume, i.e., major portion of the volume is filled with hexes and prisms.

Figure 17 (d) and Figure 17 (e) shows a hex-dominant mesh of a L-bracket, and Figure 17 (a) shows the input geometric domain and the directionality. The packing of rectangular solid cells is shown in Figure 17 (b) and Figure 17 (c). The histogram of radius-ratio of tets, the scaled Jacobian of hexes and the scaled Jacobian of prisms are shown in Figure 17 (f), Figure 17 (g) and Figure 17 (h). Again, the picture shows that hexes on the boundary are aligned with the given directionality. Hexes, prisms and tets are well-shaped, and the major portion of the volume is filled with hexes and prisms.

9 Conclusion

This paper has presented a new method that creates a hex-dominant mesh that consists of hexes, prisms and tets. The method takes as input a 3D geometric domain, desired edge length and mesh directionality, and creates a hex-dominant mesh in three steps: (1) packing rectangular solid cells on the boundary and the interior of the domain to obtain ideal node locations for a hex-dominant mesh, (2) creating a tet mesh based on the nodes created in the second step, and (3) converting a tet mesh to a hex-dominant mesh. The proposed method avoids ill-shaped elements induced by the nodes located too close to each other and creates well-shaped elements in an output hex-dominant mesh. Several experimental results show that the proposed method creates hex-dominant mesh of well-shaped elements.

Acknowledgement

This material is based in part on work supported under a NSF CAREER Award (No. 9985288).

References

[1] S. Meshkat and D. Talmor, "Generating a Mixed Mesh of Hexahedra, Pentahedra and Tetrahedra from an Underlying Tetrahedral Mesh," *International Journal for Numerical Methods in Engineering*, vol. 49, pp. 17-30, 2000.

[2] 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.

[3] E. B. Becker, G. F. Carey, and J. T. Oden, *Finite Elements: an Introduction*, vol. 1: Prentice Hall, 1981.

[4] S. J. Owen, "Non-Simplicical Unstructured Mesh Generation, Ph.D Thesis," in *The Department of Civil and Environmental Engineering*: Carnegie Mellon University, 1999, pp. 218.

[5] S. Yamakawa and K. Shimada, "HEXHOOP: Modular Templates for Converting a Hex-dominant Mesh to an All-hex Mesh," Proceedings of 10th International Meshing Roundtable, pp. 235-246, 2001.

[6] K. Shimada, J.-H. Liao, and T. Itoh, "Quadrilateral Meshing with Directionality Control through the Packing of Square Cells," Proceedings of 7th International Meshing Roundtable, pp. 61-75, 1998.

[7] T. Itoh, K. Shimada, K. Inoue, A. Yamada, and T. Furuhata, "Automated Conversion of 2D Triangular Mesh into Quadrilateral Mesh with Directionality Control," Proceedings of 7th International Meshing Roundtable, pp. 77-86, 1998.

[8] S. Yamakawa and K. Shimada, "High Quality Anisotropic Tetrahedral Mesh Generation via Ellipsoidal Bubble Packing," Proceedings of 9th International Meshing Roundtable, pp. 263-273, 2000.

[9] B. Joe, "Construction of Three-dimensional Improved-quality Triangulations Using Local Transformations," *SIAM Journal on Scientific Computing*, vol. 16, pp. 1292-1307, 1995.

[10] R. Löhner and J. R. Cebral, "Parallel Advancing Front Grid Generation," Proceedings of 8th International Meshing Roundtable, pp. 67-74, 1999.

[11] T. D. Blacker and M. B. Stephenson, "Paving : A New Approach to Automated Quadrilateral Mesh Generation," *International Journal for Numerical Methods in Engineering*, vol. 32, pp. 811-847, 1991.

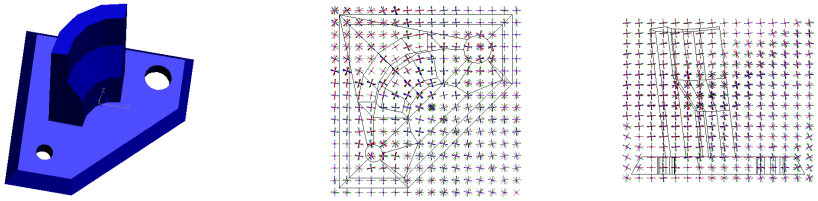
[12] D. F. Watson, "Computing the N-dimensional Delaunay Tessellation with Application to Voronoi Polytopes," *The Computer Journal*, vol. 24, pp. 167-172, 1981.

[13] A. Bower, "Computing Dirichlet tessellations," *The Computer Journal*, vol. 24, pp. 162-166, 1981.

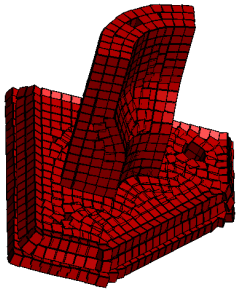
[14] P. L. George, F. Hecht, and E. Saltel, "Automatic Mesh Generator with Specified Boundary," *Computer Methods in Applied Mechanics and Engineering*, vol. 92, pp. 269-288, 1991.

[15] P. L. George, "Tet Meshing: Construction, Optimization and Adaptation," Proceedings of 8th International Meshing Roundtable, pp. 133-141, 1999.

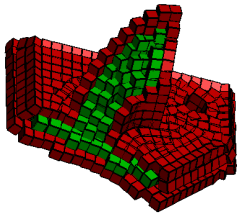
[16] P. M. Knupp, "Achieving Finite Element Mesh Quality via Optimization of the Jacobian Matrix Norm and Associated Quantities. Part II - A Framework for Volume Mesh Optimization and the Condition Number of the Jacobian Matrix," *International Journal for Numerical Methods in Engineering*, vol. 48, pp. pp. 1165-1185, 2000.



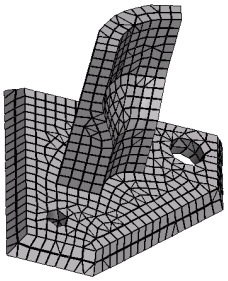
(a) Input geometric domain and directionality



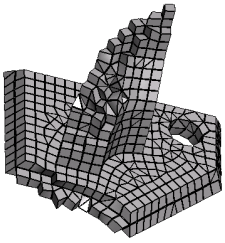
(b) Packing of rectangular solid cells



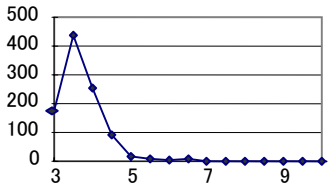
(c) Packing of rectangular solid cells (cross-section)



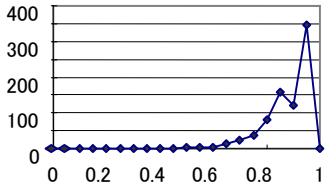
(d) Hex-dominant mesh



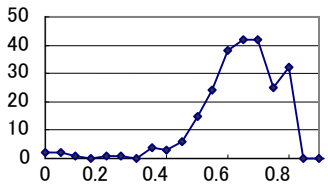
(e) Hex-dominant mesh (cross-section)



(f) Histogram of radius-ratio of tets



(g) Histogram of scaled Jacobian of hex



(h) Histogram of scaled Jacobian of prism

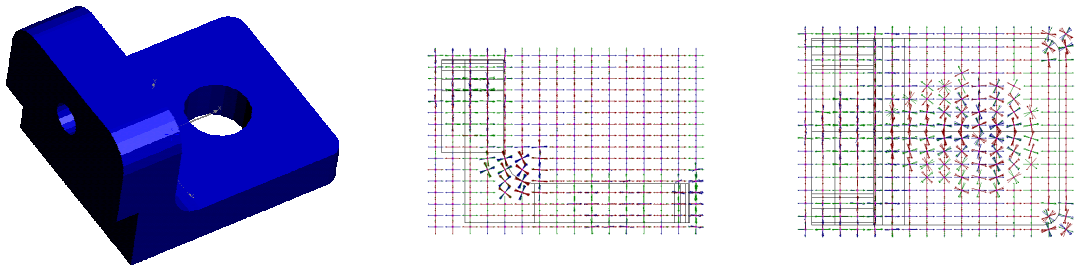
HEX	794 (39%)	TET	996 (49%)
PRISM	238 (12%)	TOTAL	2028

(i) Number of element ratio

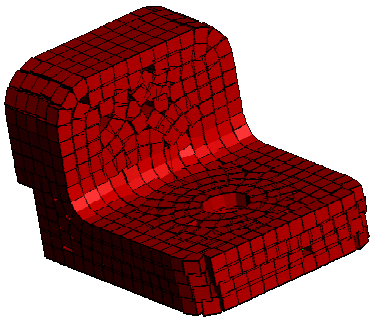
HEX	2486.7 (74.6%)	TET	362.4 (10.9%)
PRISM	482.7 (14.5%)	TOTAL	3331.8

(j) Volume ratio

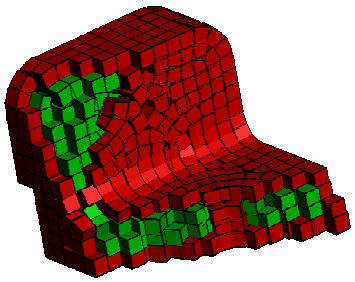
Figure 16 A hex-dominant mesh of a mechanical part with a cylindrical feature



(a) Input geometric domain and directionality

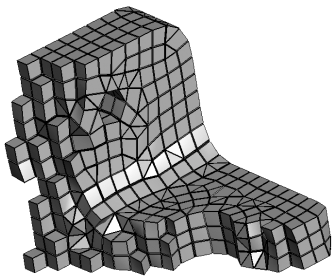


(b) Packing of rectangular solid cells

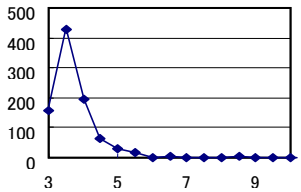


(c) Packing of rectangular solid cells (cross-section)

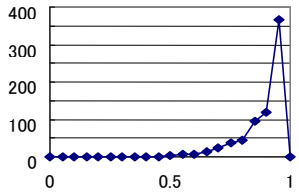
(d) Hex-dominant mesh



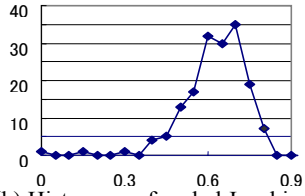
(e) Hex-dominant mesh (cross-section)



(f) Histogram of radius-ratio of tets



(g) Histogram of scaled Jacobian of hexes



(h) Histogram of scaled Jacobian of prisms

HEX	712 (40%)	TET	902 (51%)
PRISM	165 (9%)	TOTAL	1779

(i) Number of elements ratio

HEX	569.11 (76.3%)	TET	115.84 (15.5%)
PRISM	60.84 (8.2%)	TOTAL	745.79

(j) Volume ratio

Figure 17 A hex-dominant mesh of a L-bracket