INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2003; **57**:2099–2129 (DOI: 10.1002/nme.754)

Fully-automated hex-dominant mesh generation with directionality control via packing rectangular solid cells

Soji Yamakawa and Kenji Shimada*,†

Carnegie Mellon University, Pittsburgh, PA, U.S.A.

SUMMARY

A new fully automatic hex-dominant mesh generation technique of an arbitrary 3D geometric domain is presented herein. 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 consisting mostly of hexahedral elements, with additional prism and tetrahedral elements. Rectangular solid cells are packed 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-centred cubic (BCC) structure (seen in natural substances such as NaCl) 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 centres of the cells provide ideal node locations for a hex-dominant mesh. Via the advancing front method, the centres of the packed cells are then connected to form a tetrahedral mesh, and this is converted to a hex-dominant mesh by merging some of the tetrahedrons.

KEY WORDS: hex-dominant mesh; directionality; packing

1. INTRODUCTION

A new automatic hex-dominant mesh generation technique of an arbitrary 3D geometric domain is presented here. Although several hex-dominant meshing techniques are presented [1–3], they typically have some limitations; the ratio of hexahedral elements is low, or the result depends heavily on the input tetrahedral mesh; element directionality is not controlled, or the grading of element size is not precisely controlled. The proposed method overcomes these limitations by: (1) packing rectangular cells of various sizes; (2) avoiding ill-shaped elements induced by nodes located too closely together; and (3) controlling the directionality of the output hex-dominant mesh by aligning the packed rectangular solid cells.

*Correspondence to: Kenji Shimada, Department of Mechanical Engineering, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213-3890, U.S.A.
†E-mail: shimada@cmu.edu

Contract/grant sponsor: NSF; contract/grant number: 9985288

Received 6 May 2002
Revised 24 September 2002
Accepted 26 November 2002

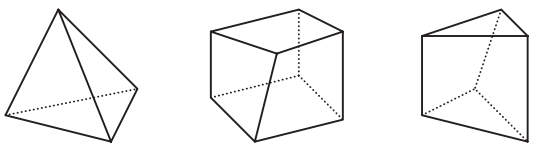


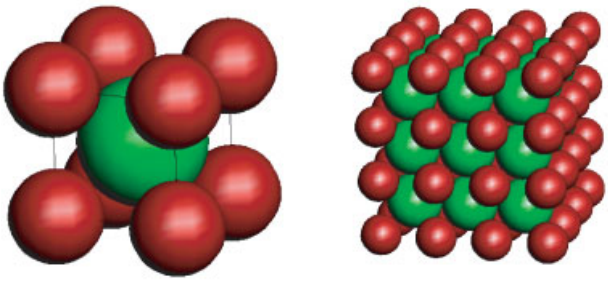
Figure 1. Three types of elements included in a hex-dominant mesh.

A hex-dominant mesh combines the good aspects of a tetrahedral mesh and an all-hex mesh. A tetrahedral mesh is widely used in industry, and it has three major advantages: (1) it can be created automatically, (2) several automatic tetrahedral mesh generation techniques give reasonably high-quality tetrahedral mesh, and (3) it is relatively easy to control the grading of element size precisely. However, a tetrahedral mesh generally needs more elements than an all-hex mesh to gain same accuracy in the finite element analysis. Although an all-hex mesh provides a more accurate solution with fewer elements than a tetrahedral mesh, an all-hex mesh is more difficult to create, and no available technique can automatically create a high-quality all-hex mesh. It is also difficult to control the grading of element size precisely in all-hex mesh. On the other hand, a hex-dominant mesh needs fewer elements than a tetrahedral mesh to attain the same accuracy in the finite element analysis, and the grading of the element size in a hex-dominant mesh can be controlled more easily than in an all-hex mesh.

The proposed method creates a hex-dominant mesh consisting of hexahedrons, prisms and tetrahedrons, as shown in Figure 1; and hexahedrons and prisms yield more accurate solutions than tetrahedrons in non-linear finite element analyses, since these two types of elements have higher order terms in the shape functions than a tetrahedron [4]. Among these three types of elements, a hexahedron has the highest order term, so an all-hex mesh, consisting exclusively of hexahedrons, 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, the hex-dominant mesh is an alternative to an all-hex mesh, and it generally gives a better solution than a tetrahedral mesh [5]. Section 10 also shows some results of finite element analyses, which indicate that a hex-dominant mesh outperforms a tetrahedral mesh. A method that converts a hex-dominant mesh to an all-hex mesh is also available [6], and the conversion method requires a high quality hex-dominant mesh to create a high quality all-hex mesh. Therefore, 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 centred 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 centres of the cells will provide ideal node locations for a hex-dominant mesh.

The proposed method takes a geometric domain, desired directionality and 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



(a) (b)

Figure 2. BCC structure, which can be seen in some natural substances such as a molecule of NaCl: Closely packed BCC structured cells yield a structured grid pattern: (a) a BCC structured cell; and (b) a crystal pattern of BCC structure.

centres of the cells. Next, a tetrahedral mesh is created by connecting the nodes obtained in the previous step. Last, some tetrahedrons are merged to create as many hexahedrons and prisms as possible.

The original idea of this approach is presented by Shimada *et al.* [7], and Itoh *et al.* [8]. Their method packs square cells inside a two-dimensional geometric domain to obtain ideal node locations for a quad-dominant mesh. The nodes are then 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 this paper is as follows: Section 2 discusses the interface-conformity conditions of the hex-dominant mesh, which must be taken into account while creating a hex-dominant mesh. Section 3 reviews previous attempts at hex-dominant mesh generation. Section 4 shows overview of the proposed method. Section 5 explains the representation of the desired directionality and edge length. Section 6 explains the algorithm that packs rectangular solid cells in the input domain. Section 7 shows the algorithm that creates a tetrahedral mesh. The algorithm that converts a tetrahedral mesh to a hex-dominant mesh is presented in Section 8. Section 9 shows some results of the proposed method followed by some results of experimental finite element analyses in Section 10, and conclusions in Section 12.

2. INTERFACE-CONFORMITY CONDITIONS OF THE HEX-DOMINANT MESH

In general mesh must satisfy interface-conformity conditions: (1) every interface between elements is shared by only two elements, and (2) no interface between two elements is exposed to the exterior of the target domain or to the third element. A two-dimensional mesh easily satisfies interface conformity because there is only one type of interface—an edge—and interface conformity is satisfied if every edge is shared by two elements. However, such conditions make hex-dominant mesh very difficult to create because there are two types of interface, a triangular face and a quadrilateral face. If the interface-conformity conditions are

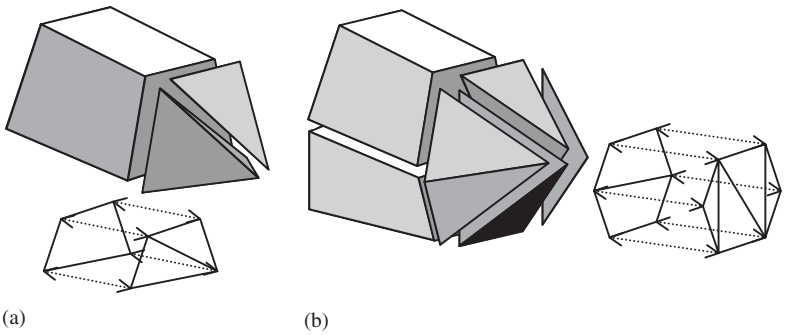


Figure 3. Valid transition and invalid transition between hexahedrons and tetrahedrons: (a) valid transition; and (b) invalid transition.

strictly enforced, no triangular face of a tetrahedron, a prism or a pyramid can be directly connected to a quadrilateral face of a hexahedron, a prism or a pyramid, because a triangle can hide only half of a quadrilateral; the other half of the quadrilateral is exposed either to the exterior of the target domain or to the third element, and the condition (2) is violated. Thus the interface-conformity conditions do not permit connection of a tetrahedron directly to a hexahedron, because a tetrahedron consists only of triangular faces, and a hexahedron consists exclusively of quadrilateral faces.

Interface-conformity conditions significantly limit the type of geometry that can be meshed into a hex-dominant mesh. Typically, only a limited volume of the target geometric domain can be meshed into hexahedrons. Although the remaining volume can be meshed into tetrahedrons, regions filled with hexahedrons and regions filled with tetrahedrons cannot be connected to each other because of these conditions.

One solution to the interface-conformity problem of hex-dominant mesh is to relax interface-conformity conditions and allow transition from a hexahedron or a prism to two tetrahedrons through a quadrilateral face, as shown in Figure 3(a). Such a quadrilateral face, connected to two tetrahedrons, is called a non-conforming quadrilateral. Allowing non-conforming quadrilaterals eases the difficulty of creating a hex-dominant mesh. However, non-conforming quadrilaterals yield considerable error in the finite element analysis because of the difference in the geometry of a quadrilateral and the geometry of a triangle, small gaps or overlaps exists around a non-conforming quadrilateral. The geometry of a quadrilateral is a bi-linear surface, while the counterpart of a triangle is a plane. Thus, unless four nodes on the non-conforming quadrilateral are co-planer, gaps or overlaps are inevitable around the non-conforming quadrilateral. For example, the cross-section of the non-conforming quadrilateral, shown in Figure 4, clearly visualizes the gaps.

Several methods that reduce errors induced by non-conforming quadrilaterals are presented. Dewhirst *et al.* present a method that reduces the error by applying multi point constraints (MPCs) [9, 10]. Their method is applicable when tetrahedrons included in the hex-dominant mesh are 10-node tetrahedrons, which have nodes at the four corners and at the centres of the six edges of each tetrahedron. Their method also constrains a node at the centre of an edge lying on the diagonal of a non-conforming quadrilateral by the function of the four

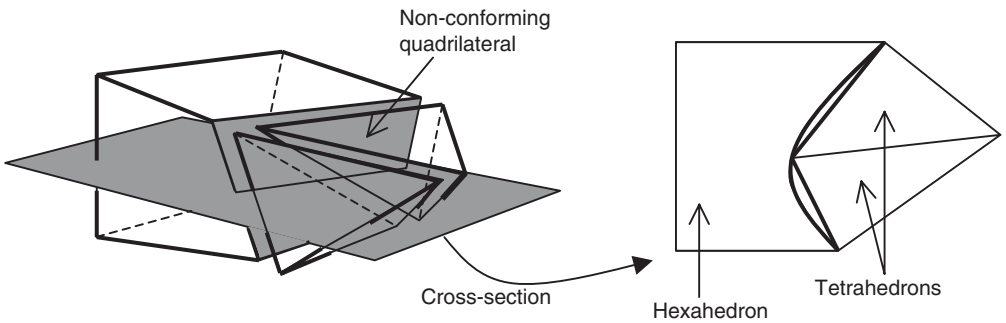


Figure 4. Gaps at the non-conforming quadrilateral.

nodes of the non-conforming quadrilateral, and the reductions of the errors are reported in their paper. If tetrahedrons included in the hex-dominant mesh are 4-node tetrahedrons, they can be converted to 10-node tetrahedrons simply by adding nodes at the centres of the tetrahedrons' edges. Dohrmann *et al.* present a method that allows connection between dissimilar surfaces of the three-dimensional meshes [11]. Although the primary purpose of their method is to increase the accuracy of the contact problem, their method can also be applied to reduce errors of non-conforming quadrilaterals.

Another solution to the interface-conformity problem of the hex-dominant mesh is to introduce pyramids. A pyramid consists of one quadrilateral face and four triangular faces, and it acts as a bridge between a quadrilateral face of a hexahedron or a prism and four tetrahedrons by connecting a quadrilateral face of the pyramid to a quadrilateral face of a hexahedron or of a prism and connecting the four triangular faces of the pyramid to four tetrahedrons. Owen *et al.* [12] present a method that places pyramids on non-conforming quadrilaterals, resolving the non-conformity. Their method takes a hex-dominant mesh that includes non-conforming quadrilaterals as input and converts it to a strictly conformed hex-dominant mesh. This method places a pyramid on every non-conforming quadrilateral, and tetrahedrons connected to the non-conforming quadrilateral are transformed so that they conform the triangular faces of the pyramid. Since pyramids eliminate all non-conforming quadrilaterals, errors induced by gaps and overlaps around the non-conforming quadrilaterals are also eliminated. However, this method is applicable only when the finite element solver accepts pyramids.

It should be emphasized that relaxed interface-conformity conditions do not allow arbitrary transition from a hexahedron or a prism to tetrahedrons through a quadrilateral face, and a non-conforming quadrilateral must be connected to two tetrahedrons. The hex-dominant mesh is invalid if a quadrilateral face is connected to more than two tetrahedrons. For example, if two hexahedrons are connected to four tetrahedrons, as shown in Figure 3(b), the connection is invalid because a quadrilateral face of the top hexahedron is connected to all four tetrahedrons, as is a quadrilateral face of the bottom hexahedron. If such an invalid transition exists, neither Dewhirst *et al.*'s error reduction technique, nor Owen *et al.*'s pyramid placing technique can be applied. Although Dohrmann's error reduction technique can still be applied, the effectiveness of the technique is reduced.

In summary, once a hex-dominant mesh with non-conforming quadrilaterals is created, (1) error reduction techniques such those of Dewhirst *et al.*'s or Dohrmann's can be applied when the solver does not accept pyramids, or (2) Owen *et al.*'s method can eliminate all non-conforming quadrilaterals by introducing pyramids if the solver accepts pyramids. Thus, the method presented in this paper focuses on creating a hex-dominant mesh with non-conforming quadrilaterals.

3. PREVIOUS WORK

Owen and Saigal present an algorithm called *H-Morph* [1], which converts a tetrahedral mesh to a hex-dominant mesh by creating individual hexahedrons starting from domain boundaries and moving inward. The method always maintains a valid hexahedron–tetrahedron 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 tetrahedral mesh into a hex-dominant mesh based on the graph theory [3]. Their method takes a tetrahedral mesh as input and creates a graph that represents the topology of the tetrahedral mesh. Their method then searches for a pattern that can be converted to a hexahedron or a prism in the graph, and when a pattern is found, a new hexahedron or a new prism is created, and the graph is updated accordingly. The result of their method significantly depends on the input tetrahedral mesh.

The results of the two methods depend heavily on the input tetrahedral mesh. However, the papers [1, 3] do not present any method that creates an appropriate input tetrahedral mesh. And neither method can control the directionality of the hexahedrons.

Meyers *et al.* [2] and Tuchinsky and Clark [13] present a method that creates a hex-dominant mesh by expanding the *plastering method*, which is presented by Blacker and Meyers [14]. Their method creates hexahedrons from the boundary to inward by the plastering method, and if the plastering method stops, and if some volume remains unmeshed, their method fills the unmeshed volume with tetrahedrons. Their method usually creates high-quality hexahedrons near the boundary; however, it cannot control the directionality of the hexahedrons, and ill shaped tetrahedrons are created if the remaining volume is very thin.

4. 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 creates a hex-dominant mesh in three steps, as illustrated in Figure 5. 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 centres of the cells provide ideal node locations for a hex-dominant mesh. However, hexahedrons are not easy to create directly from the set of nodes obtained in the second step. Instead of creating hexahedrons directly, the proposed method creates a tetrahedral mesh in the second step by the advancing front method [15] and the local transformation method [16]. The tetrahedral mesh is then transformed to a hex-dominant mesh by merging tetrahedrons in the third step.

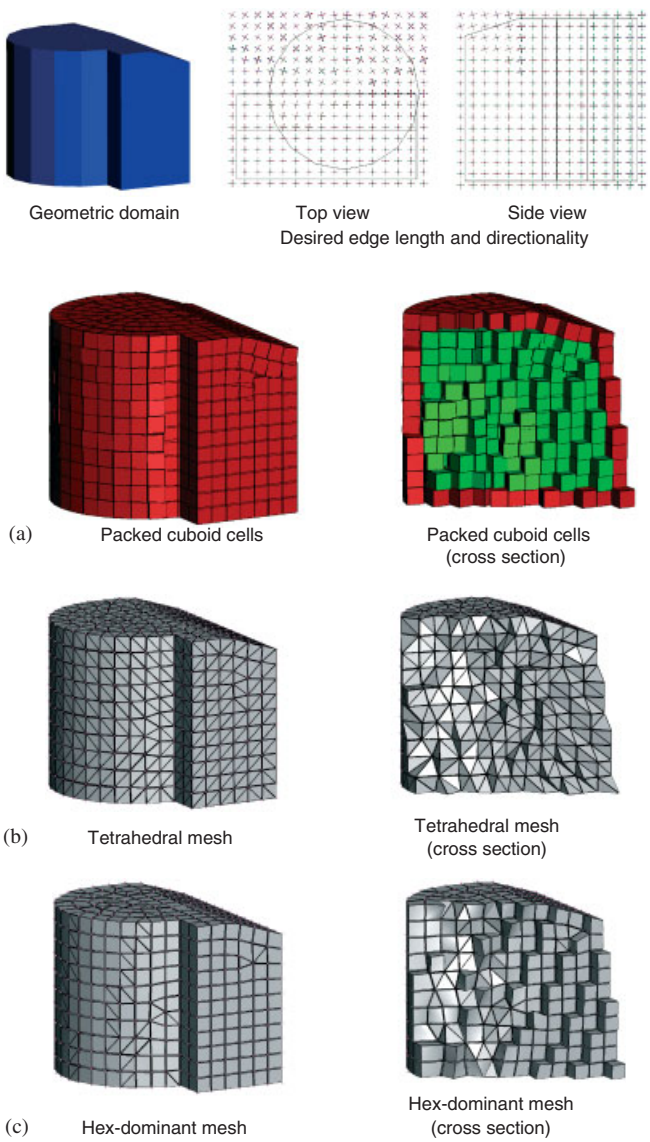


Figure 5. Overview of the proposed method. Input geometric domain, desired edge length and directionality: (a) step 1 Packing rectangular solid cells to obtain ideal node locations of a hex-dominant mesh; (b) step 2 Creating a tetrahedral mesh; and (c) step 3 Converting a tetrahedral mesh into a hex-dominant mesh.

5. REPRESENTATION OF DESIRED DIRECTIONALITY AND DESIRED EDGE LENGTH

The proposed method takes a desired directionality and edge length as input, as well as the target geometric domain. The desired directionality gives an ideal orientation of a hexahedron

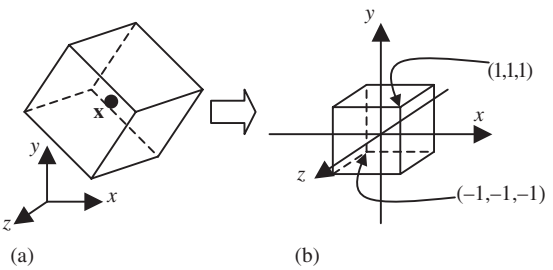


Figure 6. Transformation from an ideal hexahedron to a unit cube: (a) an ideal hexahedron; and (b) a unit cube.

at a point on the boundary of and inside the geometric domain, and a 3×3 matrix is sufficient to represent the directionality and the edge length.

An orientation is defined by three orthogonal unit vectors—or principal vectors, and a 3×3 rotational matrix is obtained by combining the principal vectors. Three principal vectors are denoted as **u**, **v** and **w**. The rotational matrix **R** can then be written as

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

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

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

An ideal hexahedron is transformed into a unit cube aligning with the x, y and z axes and filling 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 hexahedron, and $\mathbf{p}_i'$ is a node position of the unit cube, also illustrated in Figure 6.

In a special case, where a boundary-aligned hex-dominant mesh is required, directionality is computed based on the normal vector of the boundary surface and the tangential vector of the boundary edges. Since hexahedrons 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.

An appropriate interpolation scheme must be applied to compute principal vectors for the interior of the domain, and the experiments performed in this research suggest that zeroth order interpolation gives sufficiently good results.

The following definitions of the first and the second principal vectors $\mathbf{u}(\mathbf{x})$ and $\mathbf{v}(\mathbf{x})$ satisfy the above two conditions and compute $\mathbf{u}(\mathbf{x})$ and $\mathbf{v}(\mathbf{x})$ by the zeroth order interpolation for the interior of the domain, where $\mathbf{x}$ is a point on the boundary or a point inside the domain.

(1) $\mathbf{u}(\mathbf{x})$ is equal to the unit normal vector at the point on the boundary surface that is closest to $\mathbf{x}$.
(2) Let $\mathbf{x}_e$ be a point on a boundary edge, and $\mathbf{t}_e(\mathbf{x}_e)$ be a unit tangential vector of the boundary edge at $\mathbf{x}_e$. Point $\mathbf{x}_{e0}$, $\mathbf{x}_{e0} \in \mathbf{x}_e$, is defined such that it satisfies $\cos(3\pi/4) < \mathbf{t}_e(\mathbf{x}_{e0}) \cdot \mathbf{u}(\mathbf{x}) < \cos(\pi/4)$, i.e., $\mathbf{t}_e(\mathbf{x}_{e0})$ always makes more than 45° and less than 135° angle with $\mathbf{u}(\mathbf{x})$. Now assume that point $\mathbf{x}_v$, $\mathbf{x}_v \in \mathbf{x}_{e0}$, minimizes the distance between $\mathbf{x}_v$ and $\mathbf{x}$, and that vector $\mathbf{t}_v$ is defined as $\mathbf{t}_v = (\mathbf{u}(\mathbf{x}) \times \mathbf{t}_e(\mathbf{x}_v)) \times \mathbf{u}(\mathbf{x})$. $\mathbf{v}(\mathbf{x})$ is then computed as $\mathbf{v}(\mathbf{x}) = \mathbf{t}_v / \|\mathbf{t}_v\|$.

The third principal vector $\mathbf{w}(\mathbf{x})$ is simply computed by taking the 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; it does not much affect the overall outcome of the whole meshing process.

6. 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 an appropriate number of cells (with respect to the given directionality and edge length, $\mathbf{M}(\mathbf{x})$) are closely packed in the domain, they form a crystal pattern as shown in Figure 2(b), and the centres of the cells will become ideal node locations for a hex-dominant mesh.

To run the simulation, an equation of motion, governing 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 a 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.

The inter-bubble force is computed based on the proximity between spheres, or bubbles, creating a body-centred cubic (BCC) structure located at the centre and at the eight corners

of rectangular solid cells. A bubble at the centre of the cell is the main bubble; bubbles at the eight corners of the cell are 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 a diameter of d, and thus the main bubble touches each face at its centres. A sub-bubble has a diameter of $\sqrt{3d^2} - d$, and thus it touches the main bubble at one point, also illustrated in Figure 7.

An inter-bubble force 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

$$f(w) = \begin{cases} k(1.25w3 - 2.375w2 + 1.125) & \text{if } 0 \leqslant w \leqslant 1.5 \\ 0 & \text{otherwise} \end{cases}$$

where $w = \|\mathbf{x}_B - \mathbf{x}_A\|/r_A + r_B$, $\|\cdot\|$ is the Euclidian norm, and k is a spring constant. And the direction of the inter-bubble force acting on A is given as a unit vector:

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

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

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

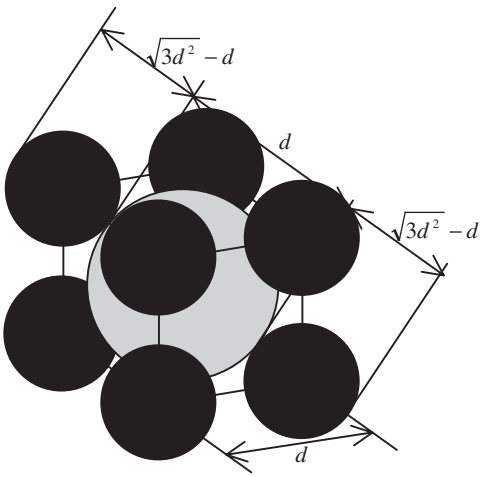


Figure 7. Diameters of the bubbles.

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

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

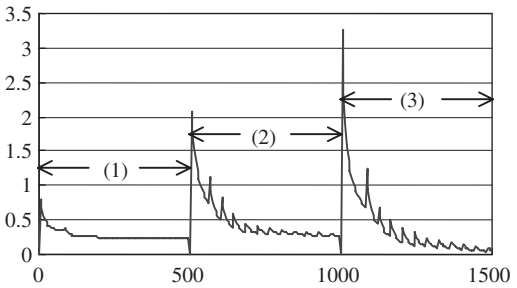
where $\mathbf{x}$ is the position of the main bubble, which is equal to the position of the rectangular solid cell. The equation is solved by an iterative numerical integration scheme such as Euler's method or the fourth order Runge–Kutta method. In each iteration, the orientations of the rectangular 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 in order 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 regions.

Because this simulation mimics the formation of a BCC crystal pattern, shown in Figure 2(b), the centres 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.

The experiments performed in this research show that rectangular solid cells packed in the target geometric domain converge to a stable configuration in less than 1500 iterations in general, and the number of iterations required for convergence does not depend significantly on the number of the cells. The cells packed on the edges of the geometric domain usually converge to a stable configuration in less than 500 iterations. Likewise, another 500 iterations or less are required to stabilize the cells packed on the faces of the geometric domain. Finally, another 500 iterations or less are required for the cells packed in the interior of the domain to reach stability. The plot of the average speed of the cells against the number of iterations is shown in Figure 8 to show the convergence. The average speed is measured while packing cells on the boundary of and inside the geometric domain of a mechanical part with a cylindrical feature shown in Figure 18(a). The cells move quickly when they are first packed on the edges, and when they are first packed on the faces, and when they are first packed in the interior of the domain. Nevertheless, the average speed of the cells quickly drops after some iterations.

Although the average speed of the cells does not drop to zero, the cells do not move significantly after the average speed drops to a certain level. The system of the cells is a second-order system because the motions of the cells are governed by a second-order



(1) Packing cells on the edges of the domain
(2) Packing cells on the faces of the domain
(3) Packing cells interior of the domain

Figure 8. Plot of the average speed of the rectangular solid cells against the number of iterations.

ordinary differential equation; therefore, the speed of the bubbles decays exponentially, but small vibrations of the cells remain indefinitely. As a result, a small amount of average cell speed is observed even after the cells stop showing significant movement. However, since the vibration magnitude is small, the configuration of the cells is stable and gives good node locations for a hex-dominant mesh.

However, there is no theoretical guarantee that this method is stable in every case. In fact, when the input domain has a small feature, and the specified cell size is much larger than the size of the feature, the cells often show slow convergence. If the input domain has a small feature, a reasonable cell size must be chosen with respect to the size of the feature.

7. GENERATING A TETRAHEDRAL MESH

After the nodes are created by the rectangular solid cell packing method, the nodes are connected to form a tetrahedral mesh. The purpose of creating a tetrahedral 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 hexahedrons and prisms. 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 makes the computation potentially 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 tetrahedron has only triangular faces. Since the geometries of the triangular faces are planer, an intersection between tetrahedrons can be detected easily by solving linear equations. Hence, creating tetrahedrons is easier than creating hexahedrons and prisms in terms of robustness. In addition, if no two tetrahedrons are intersecting, merging some tetrahedrons to create hexahedrons and prisms will not create a new intersection. It is

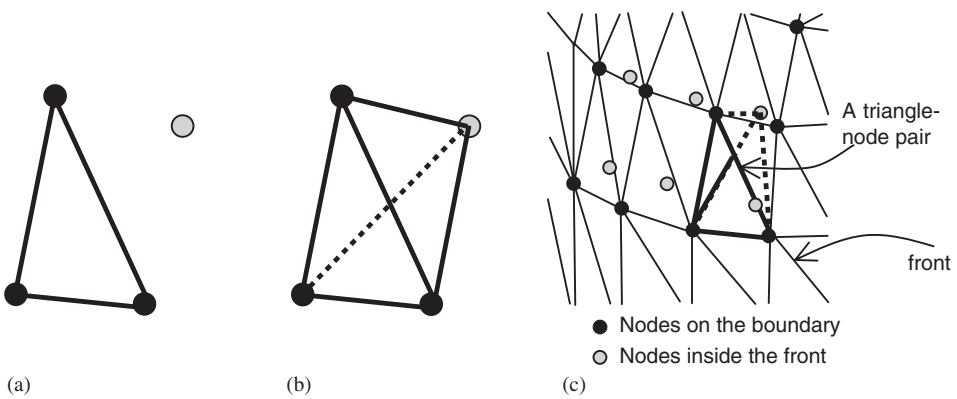


Figure 9. A front and a triangle-node pair: (a) a triangle-node pair; (b) a newly formed tetrahedron; and (c) updating the front.

thus reasonable to create a hex-dominant mesh in two steps: (1) creating a tetrahedral mesh and (2) merging some tetrahedrons to create hexahedrons and prisms.

A tetrahedral mesh is created by the advancing front method, presented in Reference [15]. Unlike the typical advancing front method [17, 18], no new nodes are added inside the domain during this process because nodes are already created by the rectangular solid cell packing method.

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 [19, 20], as shown in Figure 9(a). When a triangle-node pair is found, a tetrahedron is created by connecting the node to the triangle, as shown in Figure 9(b). The front is also updated so the newly created tetrahedron is excluded by the front, as shown in Figure 9(c). Since the volume of the newly created tetrahedron is removed from the volume enclosed by the front, the front shrinks every time a tetrahedron is created. By repeating this process until the front disappears, the program meshes the domain into a tetrahedral mesh. Unfortunately, the advancing front method does not guarantee success, and if it stops before it finishes the volume, some exception handling is necessary to make it more likely to succeed. A more detailed explanation of this step is found in Reference [15].

After the tetrahedral mesh is created, the mesh quality is improved by a method called local transformation [16]. 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 for creating a tetrahedral mesh is to apply the Delaunay triangulation method [21, 22]. 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. The advancing front method is thus suitable for the work presented in this paper.

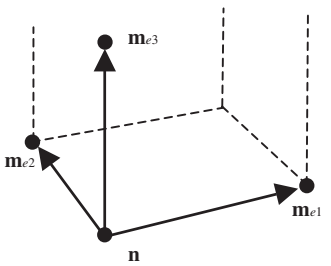


Figure 10. Three vectors used in computing the scaled Jacobian at node *n*.

8. CONVERTING A TETRAHEDRAL MESH TO A HEX-DOMINANT MESH

After the tetrahedral mesh is created, the proposed method merges some tetrahedrons to create hexahedrons and prisms. This method first creates a list of node combinations that form hexahedrons; the combinations are then sorted by the shape quality of the hexahedrons so that the node combinations yielding the superior quality hexahedrons are listed first. Tetrahedrons are merged and converted to hexahedrons individually based on the list. After exhausting all the possible hexahedrons, the proposed method creates a list of node combinations that create prisms. The combinations are then sorted by the quality of the prisms so that the combination that yields the best quality prism comes first and the worst quality last. Some tetrahedrons are then merged and converted to prisms based on the list.

8.1. Definition of scaled Jacobian

During the conversion process, the proposed method measures the quality of an element by a scaled Jacobian [23]. If element e is a tetrahedron, a prism or a hexahedron, 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 10, 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 hexahedron 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 hexahedron must be as close to 90° as possible, $Js_e(\mathbf{n})$ of a hexahedron 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, referred to as 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 hexahedron, from -0.866 to 0.866 for a prism, and from -0.707 to 0.707 for a tetrahedron.

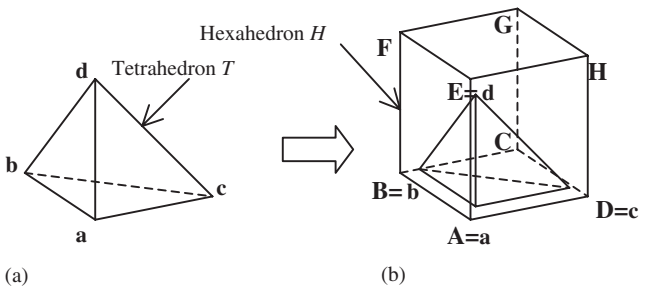


Figure 11. Finding node combinations based on Pattern 1: (a) a tetrahedron in the tetrahedral mesh; and (b) assigning the four nodes of the tetrahedron.

8.2. Finding node combinations that create hexahedrons

The proposed method makes a list of possible node combinations that can each be converted to a hexahedron. Each entry of the list stores eight nodes that will become eight nodes of a hexahedron. To facilitate the explanation, a hexahedron to be created is denoted as H, and the eight corners of hexahedron H are denoted as **A** through **H**, and **ABCD**, **ADHE**, **BAEF**, **CBFG**, **DCGH** and **FEHG** are the six quadrilateral faces of hexahedron H (see Figure 11). Nodes in the tetrahedral mesh are assigned to **A** through **H** to make a node combination.

The proposed method searches node combinations based on the two possible tetrahedron configurations: (Pattern 1) three edges of a tetrahedron become three edges of a hexahedron, as shown in Figure 11(b), and (Pattern 2) six tetrahedrons share an edge that will become a diagonal of the hexahedron, as shown in Figure 12(b).

The proposed method searches node combinations of Pattern 1 based on the assumption that four nodes of a tetrahedron become **A**, **B**, **D** and **E** of hexahedron H, and nodes to be assigned for **C**, **F**, **G** and **H** will be found by the edges of the tetrahedral mesh. Let T be a tetrahedron, and **a**, **b**, **c** and **d** are the four nodes of tetrahedron T, satisfying $Js_T(\mathbf{a}) \geqslant Js_T(\mathbf{b}), Js_T(\mathbf{c}), Js_T(\mathbf{d})$ and $(\mathbf{c}-\mathbf{a})\times(\mathbf{b}-\mathbf{a})\cdot(\mathbf{d}-\mathbf{a})>0$, as shown in Figure 11(a). The proposed method assumes that three edges of T, $\mathbf{a}-\mathbf{b}$, $\mathbf{a}-\mathbf{c}$ and $\mathbf{a}-\mathbf{d}$, will become three edges of hexahedron H, as shown in Figure 11(b). If node **a** is assigned to **A**, nodes **b**, **c** and **d** can be assigned to **B**, **D** and **E**, and edges $\mathbf{a}-\mathbf{b}$, $\mathbf{a}-\mathbf{c}$ and $\mathbf{a}-\mathbf{d}$ will become **AB**, **AD** and **AE**. Although $(\mathbf{b},\mathbf{c},\mathbf{d})$ can also be assigned to $(\mathbf{D},\mathbf{E},\mathbf{B})$ or $(\mathbf{E},\mathbf{B},\mathbf{D})$, the result will be equivalent. The candidate nodes for **F** are then limited to the nodes directly connected to **b** and **d**; 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 choice of **p**, **q**, **r**, and **s**, and if node combination $(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D},\mathbf{E},\mathbf{F},\mathbf{G},\mathbf{H})=(\mathbf{a},\mathbf{b},\mathbf{q},\mathbf{c},\mathbf{d},\mathbf{p},\mathbf{s},\mathbf{r})$ makes a minimum scaled Jacobian of H positive, the node combination is added to the node combination list. It must be noted that a scaled Jacobian $Js_T(\mathbf{a})$ is inherited to $Js_H(\mathbf{A})$, and since node **a** is chosen so that it satisfies $Js_T(\mathbf{a}) \geqslant Js_T(\mathbf{b}), Js_T(\mathbf{c}), Js_T(\mathbf{d})$, the largest possible

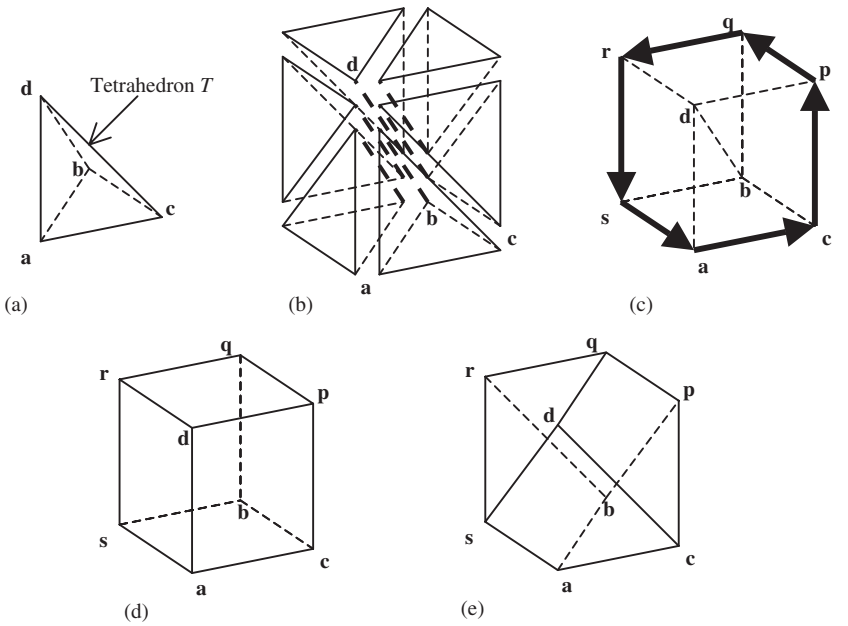


Figure 12. Finding node combinations based on Pattern 2: (a) reordering nodes of a tetrahedron; (b) six tetrahedrons sharing an edge; (c) a loop of edges; (d) a possible node assignment; and (e) another possible node assignment.

scaled Jacobian of tetrahedron T, $Js_T(\mathbf{a})$, is inherited to hexahedron H. The proposed method searches node combinations of Pattern 1 for every tetrahedron, and all node combinations found by the search are added to the node combination list.

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

The proposed method searches node combinations of Pattern 2 based on the assumption that the longest edge of a tetrahedron becomes a diagonal of a hexahedron, and the assumption will narrow the possible node combinations to two, and the combination yielding a better minimum scaled Jacobian will be added to the node combination list. Let T be a tetrahedron, and $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$ be nodes of tetrahedron T, satisfying $\|\mathbf{b}-\mathbf{d}\| \geqslant \|\mathbf{a}-\mathbf{b}\|, \|\mathbf{a}-\mathbf{c}\|, \|\mathbf{a}-\mathbf{d}\|, \|\mathbf{b}-\mathbf{c}\|, \|\mathbf{c}-\mathbf{d}\|$ and $(\mathbf{c}-\mathbf{a})\times(\mathbf{b}-\mathbf{a})\cdot(\mathbf{d}-\mathbf{a})>0$, as shown in Figure 12(a). The proposed method assumes that edge $\mathbf{b}-\mathbf{d}$ will become a diagonal of the hexahedron, because the diagonal of a well-shaped hexahedron is always longer than any edges of the hexahedron and any diagonals of the faces of the hexahedron. If the number of tetrahedrons sharing edge $\mathbf{b}-\mathbf{d}$ is not six, the method moves on to the next T and begins again. If the number of tetrahedrons sharing edge $\mathbf{b}-\mathbf{d}$ is six, the proposed method then makes a list of six tetrahedrons sharing edge $\mathbf{b}-\mathbf{d}$, as shown in Figure 12(b). The edges included in the six tetrahedrons and not

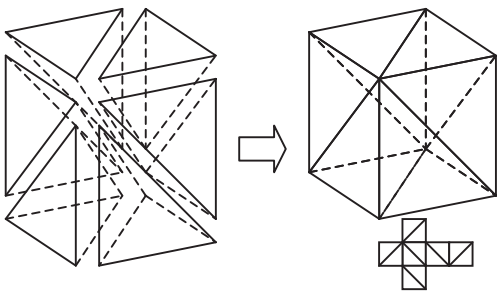


Figure 13. A set of tetrahedrons to be merged to form a hex, and a polyhedron made of the exterior of the set of the tetrahedrons.

connected to nodes **b** and **d** make a loop as shown in Figure 12(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 $(\mathbf{a},\mathbf{c},\mathbf{p},\mathbf{q},\mathbf{r},\mathbf{s})$. At this point, there are still two possible node combinations. One is $(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D},\mathbf{E},\mathbf{F},\mathbf{G},\mathbf{H})=(\mathbf{a},\mathbf{s},\mathbf{b},\mathbf{c},\mathbf{d},\mathbf{r},\mathbf{q},\mathbf{p})$ as shown in Figure 12(d), and the other is $(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D},\mathbf{E},\mathbf{F},\mathbf{G},\mathbf{H})=(\mathbf{a},\mathbf{b},\mathbf{p},\mathbf{c},\mathbf{s},\mathbf{r},\mathbf{q},\mathbf{d})$ as shown in Figure 12(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 hexahedrons 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 12, the combination shown in Figure 12(d) clearly yields better quality, and thus it will be taken. The proposed method searches node combinations of Pattern 2 for every tetrahedron, and all the node combinations found by the search are added in the node combination list.

8.3. Creating hexahedrons 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 hexahedrons 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 hexahedrons based on the sorted node combination list.

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

Condition (1) is not satisfied when some tetrahedrons that were using four of the eight nodes no longer exist because they already became a part of another hexahedron. To enforce condition (1), the proposed method creates a polyhedron by merging tetrahedrons, each consisting of four of the eight nodes, as shown in Figure 13. If the number of triangles included in the polyhedron is not 12, or if every quadrilateral face of hexahedron H does not correspond to a unique pair of adjacent triangles of the polyhedron, the node combination

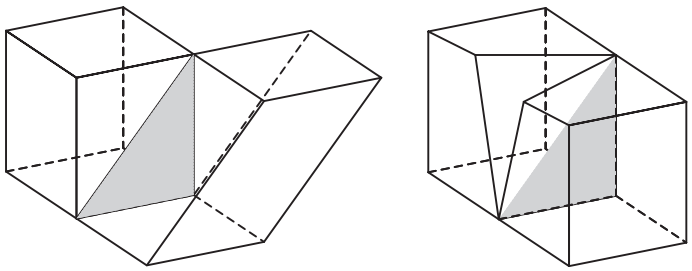


Figure 14. Examples of two quadrilaterals sharing three nodes, which violate the interface-conformity conditions.

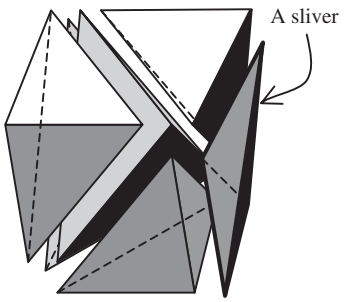


Figure 15. A sliver (very flat tetrahedron) included in a node combination: The sliver will be deleted when the tetrahedrons are merged and converted to a hexahedron.

is discarded. Since no geometric computation is carried out to enforce Condition (1), it is not necessary to check intersections between bi-linear surfaces, which potentially makes the computation unstable.

Condition (2) is not satisfied when a quadrilateral face of hexahedron H shares three of four nodes of a quadrilateral face of a hexahedron already-created, as shown in Figure 14. To enforce condition (2), the proposed method checks six quadrilateral faces of hexahedron H, **ABCD**, **ADHE**, **BAEF**, **CBFG**, **DCGH** and **FEHG**. If a quadrilateral face of hexahedron H shares three of four nodes of a quadrilateral face of the adjacent hexahedron, the node combination is discarded.

The two conditions also prevent the creation of an invalid transition between a hexahedron and tetrahedrons, such as shown in Figure 3(b), because a quadrilateral face of hexahedron H is connected either to two triangles, to a quadrilateral, or to the exterior of the domain, if the two conditions are satisfied. If the two conditions are satisfied, the proposed method deletes the tetrahedrons that each consist of four of the eight nodes of the node combination, and hexahedron H is added to the mesh.

It must be noted that if an ill-shaped tetrahedron—known as a sliver—consists of four nodes of a node combination, as shown in Figure 15, the sliver will be deleted when a hexahedron is created from the node combination. Although slivers are mostly eliminated by

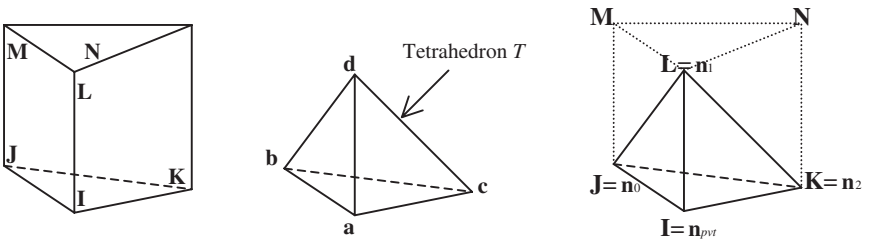


Figure 16. Finding node combinations for a prism: (a) a prism to be created; (b) a tetrahedron in the tetrahedral mesh; and (c) node assignments for $\mathbf{I}, \mathbf{J}, \mathbf{K}$ and $\mathbf{L}$.

local transformation [16], some slivers may remain even after the local transformation, and a remaining sliver typically consists of four nodes that will become the quadrilateral face of a hexahedron or a prism. The conversion process thus eliminates many of the remaining slivers, and the quality of the mesh is improved.

8.4. Finding node combinations for prisms

After creating hexahedrons, the proposed method makes a list of node combinations that can each be a prism; each entry contains six nodes capable of becoming nodes of a prism. To elucidate: the prism to be created is denoted as P, and nodes of P are denoted as $(\mathbf{I}, \mathbf{J}, \mathbf{K}, \mathbf{L}, \mathbf{M}, \mathbf{N})$. **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 16(a). Since some tetrahedrons are already converted to hexahedrons, the mesh that was a tetrahedral mesh is no longer a tetrahedral mesh, but it is already a hex-dominant mesh with no prism.

The proposed method assumes that four nodes of a tetrahedron 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 tetrahedron, and **a**, **b**, **c** and **d** be four nodes of tetrahedron T, as shown in Figure 16(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 tetrahedron, 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 $(\mathbf{b}, \mathbf{c}, \mathbf{d})$, or $(\mathbf{c}, \mathbf{d}, \mathbf{b})$, or $(\mathbf{d}, \mathbf{b}, \mathbf{c})$. For each choice of $\mathbf{n}_{pvt}$, $\mathbf{n}_0$, $\mathbf{n}_1$ and $\mathbf{n}_2$, the proposed method assumes $\mathbf{I} = \mathbf{n}_{pvt}, \mathbf{J} = \mathbf{n}_0, \mathbf{K} = \mathbf{n}_1$, and $\mathbf{L} = \mathbf{n}_2$ as shown in Figure 16(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** are 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 $(\mathbf{I}, \mathbf{J}, \mathbf{K}, \mathbf{L}, \mathbf{M}, \mathbf{N}) = (\mathbf{n}_{pvt}, \mathbf{n}_0, \mathbf{n}_1, \mathbf{n}_2, \mathbf{p}, \mathbf{q})$. 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 tetrahedron, and all node combinations found by the search are added to the node combination list.

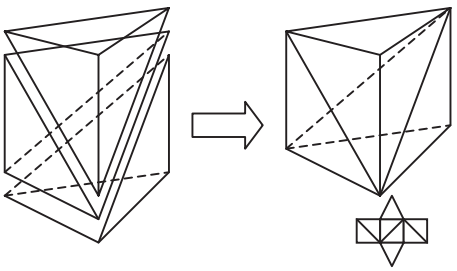


Figure 17. A set of tetrahedrons to be merged to form a prism, and a polyhedron made of the exterior of the set of the tetrahedrons.

8.5. 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 $(\mathbf{I}, \mathbf{J}, \mathbf{K}, \mathbf{L}, \mathbf{M}, \mathbf{N})$, the proposed method attempts to create prism P, which consists of the six nodes, by merging tetrahedrons each consisting of four of the six nodes. The node combination, however, is discarded if it does not satisfy the following two conditions: (1) the creation of prism P does not induce gaps or overlaps, and (2) the quadrilateral faces of P are compatible with quadrilateral faces of adjacent hexahedrons and prisms.

Condition (1) is not satisfied when some tetrahedrons that were using four of the six nodes no longer exist because they are already part of another hexahedron or prism. To enforce condition (1), the proposed method creates a polyhedron by merging the tetrahedrons that each consist of four of the six nodes as shown in Figure 17. 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.

Condition (2) is not satisfied when a quadrilateral face of prism p shares three of four nodes of a quadrilateral face of hexahedron or prism that has already been created, as shown in Figure 14. 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 hexahedron or a prism, the node combination is discarded.

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

9. RESULTS

This section presents some results of applying the proposed method. The ratios of hexahedrons, prisms and tetrahedrons are presented for each example in terms of the number of elements, and of volumes. The quality of the hexahedrons and prisms are presented by the histogram of

the minimum scaled Jacobian, and the quality of the tetrahedrons is presented by the histogram of radius-ratio, which is the radius of circumscribed sphere divided by the radius of the inscribed sphere. For an equilateral tetrahedron, which is a perfectly well-shaped tetrahedron, the radius-ratio becomes 3.0, and it grows as the shape of the element worsens.

Figures 18(d) and (e) show a hex-dominant mesh of a mechanical part with a cylindrical feature, and Figure 18(a) shows the input geometric domain and the directionality. The directionality shown in Figure 18(a) is computed from the boundary of the geometry by the method presented in Section 5. The packing of rectangular solid cells is shown in Figures 18(b) and (c). The picture clearly shows that hexahedrons on the boundary are aligned with the given directionality. The histogram of the radius-ratio of tetrahedrons, the scaled Jacobian of hexahedrons and the scaled Jacobian of prisms are shown in Figures 18(f)–(h). These histograms indicate that tetrahedrons are well shaped because most of them have radius-ratios of between 3.0 and 4.0. Hexahedrons and prisms are also well shaped because the peaks of the histograms of hexahedrons and prisms are located near the maximum value that the scaled Jacobian can take.

Figure 19 shows a hex-dominant mesh of the same geometry as that of the previous example with different directionality. For this example, the directionality is specified so that the hexahedrons are aligned with the cylindrical feature of the mechanical part, but the other portion of the mechanical part is not taken into account by the directionality. Figure 19(a) shows the given directionality, and Figure 19(b) shows the packing of rectangular solid cells. Figures 19(c) and (d) show the output hex-dominant mesh. The histogram of the radius-ratio of tetrahedrons, the scaled Jacobian of hexahedrons and prisms are shown in Figures 19(e)–(g). These histograms indicate that elements in the hex-dominant mesh are well shaped.

Figure 20 shows a graded hex-dominant mesh of the same geometry as the geometry of the previous example. In this example, the desired edge length, specified at the bottom of the cylindrical feature, is a half of the length specified at the top of the cylindrical feature. The specified directionality is identical to the Figure 18(a). As can be seen from Figure 20(a), the packed rectangular solid cells at the bottom of the cylindrical feature are small, as are the hexahedrons at the bottom of the cylindrical feature. The quality of the elements are almost same as the uniform hex-dominant mesh shown in Figure 18, therefore the mesh is of high quality.

Figures 21(d) and (e) show a hex-dominant mesh of an L-bracket, and Figure 21(a) shows the input geometric domain and directionality. The packing of rectangular solid cells is shown in Figures 21(b) and (c). The histogram of radius-ratio of tetrahedrons, the scaled Jacobian of hexahedrons and the scaled Jacobian of prisms are shown in Figures 21(d)–(f). The picture shows that hexahedrons are aligned with the given directionality, and hexahedrons, prisms and, tetrahedrons are well shaped.

Table I summarizes the statistics of the ratios of hexahedrons, prisms and tetrahedrons in the meshes in terms of the number of elements and volumes. The hex-dominant mesh of the mechanical part with a cylindrical feature, meshed with the boundary-aligned directionality, has hexahedrons filling almost three quarters of the entire volume, although the number of hexahedrons is less than 40% of the total number of elements. The volume ratio and the ratio of the number of elements are so different because a hexahedrons is created by merging five to six tetrahedrons, and an average hexahedron has about five to six times larger volume than an average tetrahedron.

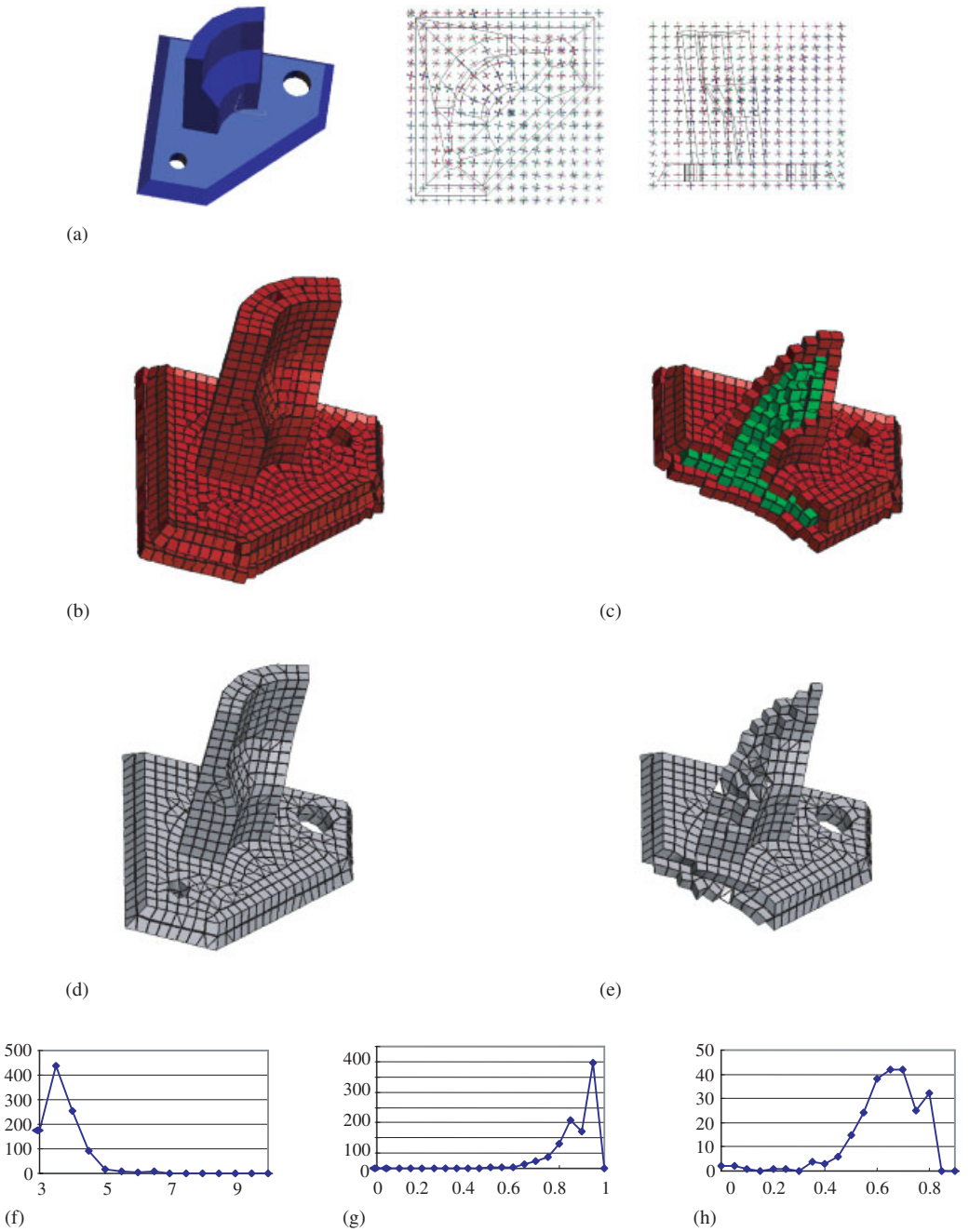


Figure 18. A hex-dominant mesh of a mechanical part with a cylindrical feature (meshed with boundary-aligned directionality): (a) input geometric domain and directionality; (b) packing of rectangular solid cells; (c) packing of rectangular solid cells (cross-section); (d) hex-dominant mesh; (e) hex-dominant mesh (cross-section); (f) histogram of radius-ratio of tetrahedrons; (g) histogram of scaled Jacobian of hexahedrons; and (h) histogram of scaled Jacobian of prisms.

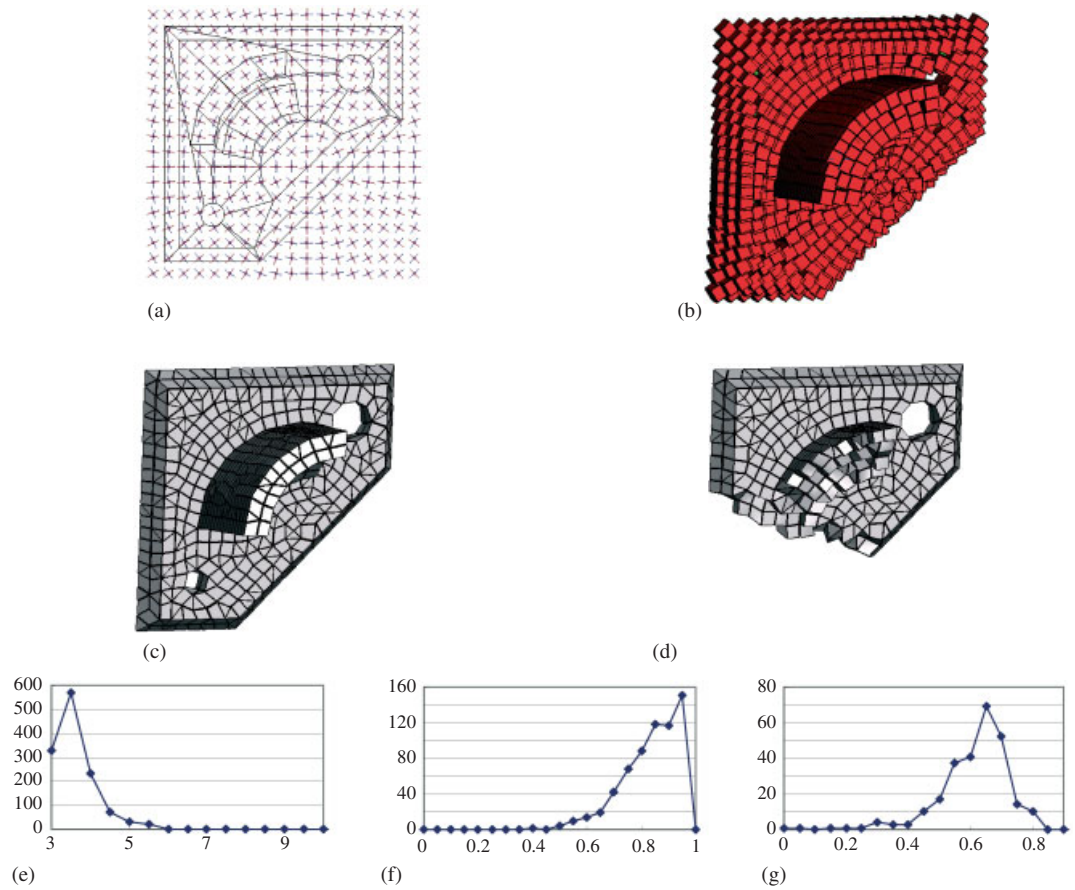


Figure 19. A hex-dominant mesh of a mechanical part with a cylindrical feature (meshed with boundary-unaligned directionality): (a) input directionality; (b) packed rectangular solid cells; (c) hex-dominant mesh; (d) hex-dominant mesh (cross-section); (e) histogram of radius-ratio of tetrahedrons; (f) histogram of scaled Jacobian of hexahedrons; and (g) histogram of scaled Jacobian of prisms.

The hex-dominant mesh of the mechanical part with a cylindrical feature, meshed with the boundary-unaligned directionality, indicates less volume ratio of hexahedrons than the part meshed with boundary-aligned directionality. The reason the volume ratio of hexahedrons lowers with boundary-unaligned directionality is because many non-hex elements are created near the boundary. If the given directionality near the boundary is not aligned with the boundary, i.e. if none of three principal vectors is parallel to the normal of the boundary surface, hexahedrons gap near the boundary, and this gap is filled with non-hex elements. As a result the volume ratio of non-hex elements increases. However, if the boundary-unaligned directionality is given, creating fewer hexahedrons is the right solution for the input.

The graded hex-dominant mesh of the mechanical part also indicates less volume ratio of hexahedrons than the uniform hex-dominant mesh of the mechanical part. When element size

(a) (b) (c)

Figure 20. A graded hex-dominant mesh of a mechanical part with a cylindrical feature: (a) packed rectangular solid cells (cross-section); (b) graded hex-dominant mesh; and (c) graded hex-dominant mesh (cross-section).

varies over the domain, some non-hex elements must be packed between a large hexahedron and a small hexahedron; as a result the ratio of hexahedrons decreases. The possible solution to this problem is to specify that the gradient of the element size in the critical region is small. If the gradient of the element size is small in the critical region, many hexahedrons are created in the critical region, and the result of the finite element analysis in and near the critical region is expected to be more accurate even though the overall hexahedron ratio is low.

The hex-dominant mesh of the L-bracket has hexahedrons filling more than three quarters of the total volume, and the number of hexahedrons is 40% of the total number of elements. The hex-dominant mesh of the L-bracket is created with the boundary-aligned directionality; the result confirms that the boundary-aligned directionality increases the total number of hexahedrons.

In general, when the specified element size is smaller and more uniform, more hex elements are created. However, it increases the total number of nodes, and thus it contradicts the purpose of a hex-dominant mesh—obtaining a more accurate finite element solution with smaller number of nodes. The ratio of hex elements to other elements also increases when the thickness of the most part of the input domain matches an integer multiple of the specified element size. However, finding an element size that maximizes the ratio of hex elements to other elements requires a sophisticated feature recognition technique, and thus it is one of the future research topics.

10. EXPERIMENTAL FINITE ELEMENT ANALYSIS

Some structural finite element analyses are performed in order to show the performance of hex-dominant meshes. The experiments are conducted by using ANSYS [24], using the geometry of a mechanical part with a cylindrical feature, shown in Figure 18(a). Five different tetrahedral meshes and three different hex-dominant meshes of various resolutions are fed to ANSYS, and ANSYS solves a structural analysis with the same loading conditions as: (1) the bottom face of the mechanical part is cantilevered; and (2) the force is applied on the top surface of the cylindrical feature to the centripetal direction of the feature. The tetrahedral meshes are created by the method presented in Reference [15], and none of the tetrahedrons in the meshes shows

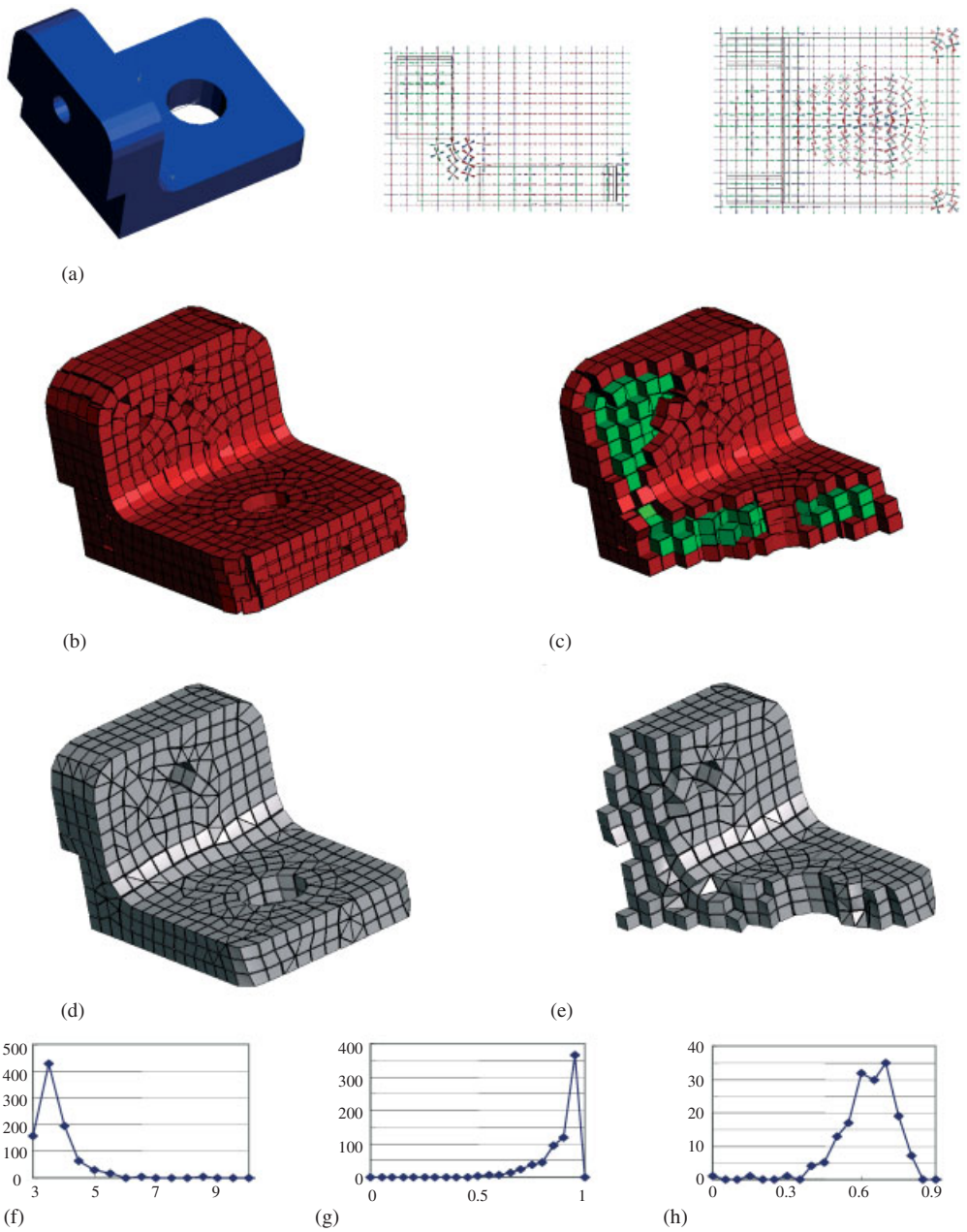


Figure 21. A hex-dominant mesh of a L-bracket: (a) input geometric domain and directionality; (b) packing of rectangular solid cells; (c) packing of rectangular solid cells (cross-section); (d) hex-dominant mesh; (e) hex-dominant mesh (cross-section); (f) histogram of radius-ratio of tetrahedrons; (g) histogram of scaled Jacobian of hexahedrons; and (h) histogram of scaled Jacobian of prisms.

Table I. Statistics of the example hex-dominant meshes.

	Volume				Number of elements			
	Hex	Prism	Tet	Total	Hex	Prism	Tet	Total
Mechanical part with a cylindrical feature meshed with the boundary-aligned directionality	2486.7 (75.6%)	482.7 (14.5%)	362.4 (10.9%)	3331.8	794 (39%)	238 (11%)	996 (49%)	2028
Mechanical part with a cylindrical feature meshed with the boundary-unaligned directionality	2163.9 (64.9%)	459.9 (13.8%)	710.9 (21.3%)	3334.7	630 (29%)	265 (12%)	1258 (58%)	2153
Mechanical part with a cylindrical feature meshed with the boundary-aligned directionality and graded element size	2145.6 (64.4%)	424.8 (12.8%)	760.8 (22.8%)	3331.3	2428 (29%)	762 (9%)	4976 (60%)	8166
L-bracket	569.1 (76.3%)	60.8 (8.2%)	115.8 (15.5%)	745.79	712 (40%)	165 (9%)	902 (50%)	1779

Table II. Number of elements and total displacements of the four corner nodes of the top surface of the cylindrical feature.

Tetrahedral mesh	Number of elements	12 063	17 249	26 986	46 240	89 726
	Displacement	1.89	1.95	1.99	2.04	2.08
	% error against 2.1	10%	7%	5%	3%	1%
Hex-dominant mesh	Number of elements	1677	3608	7692	11 996	21 185
	Displacement	1.68	1.97	2.03	2.08	2.09
	% error against 2.1	20%	6%	3%	1%	0.5%
Graded hex-dominant mesh	Number of elements			16 146		
	Displacement			2.04		
	% error against 2.1			3%		

the radius-ratio higher than 7.0; the mesh quality is thus very high. Hex-dominant meshes are created by the method presented in this paper, and since ANSYS can accept pyramids, Owen *et al.*'s method [12] is applied, and pyramids are placed on the non-conforming quadrilaterals to make the mesh conform fully.

To compare the accuracy of the results, total displacement of the four corner nodes of the top surface of the cylindrical feature, obtained with tetrahedral meshes and hex-dominant meshes, are tabulated in Table II and plotted in Figure 22. The results show that the hex-dominant meshes outperform the tetrahedral meshes. The plots in Figure 22 clearly show that total displacement converges to near 2.1 with finer mesh sizes. To facilitate the comparison, let 2.1 be the acceptably accurate solution to the total displacement of the four nodes, and it is taken as a reference value to compute per cent error of the solution. The tetrahedral mesh requires 89 726 elements to obtain a solution with 1% error, while the hex-dominant mesh needs only 21 185 elements to obtain a solution with 0.5% error. The tetrahedral mesh with 17 249 elements shows 7% error, and it is almost same as the error obtained with the

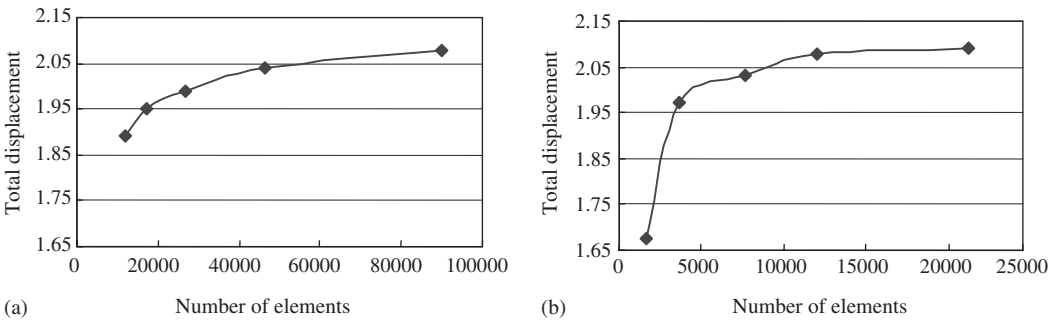


Figure 22. Plot of the total displacement of the four corner nodes of the top surface of the cylindrical feature: (a) total displacements obtained with tetrahedral meshes; and (b) total displacements obtained with hex-dominant meshes.

hex-dominant mesh with 3608 elements. Again, the hex-dominant mesh needs less than one fourth the number of elements than the tetrahedral mesh to achieve the same accuracy.

Figures 23(a) and (b) plot distributions of the first principal stress, obtained with the tetrahedral mesh with 89 726 elements, and the hex-dominant mesh with 10 561 elements, respectively. In both plots, stress is concentrated at the bottom of the cylindrical feature, and there is another dim peak at the middle of the cylindrical feature. Although the plot obtained with the hex-dominant mesh has some jaggy contours, it captures well the contour around the most important region—the region near the stress concentration. Again, the hex-dominant mesh needs less than one-eighth as many elements as the tetrahedral mesh to give a stress distribution of the same quality.

Figure 23(c) plots the distribution of the 1st principal stress obtained with a graded hex-dominant mesh with 16 146 elements. Although the plot has some jaggy contours, in general it agrees with the plot obtained with the tetrahedral mesh with 89 726 elements. The graded hex-dominant mesh tends to include more non-hex elements than a uniform hex-dominant mesh because some non-hex elements must fill the gaps between a large hexahedron and a small hexahedron, and the non-hex elements yield jagged contours. The future development of the error reduction technique for those non-hex elements may reduce those jagged contours, and the graded hex-dominant mesh will give more accurate results with fewer elements if such a technique is developed.

These experimental results show that the hex-dominant mesh with fewer elements gives a more accurate solution than the tetrahedral mesh. Thus, the hex-dominant mesh contributes the reduction in the computational time of the finite element analysis.

11. DISCUSSION OF THE COMPUTATIONAL COMPLEXITY

The total computational complexity of the proposed method is $O(n \log n)$, where n is the number of bubbles, which is equal to the number of nodes of the output mesh. The major portion of the computational time comes from the packing process and the advancing front process, and the computational complexity of each of the two processes is $O(n \log n)$. The

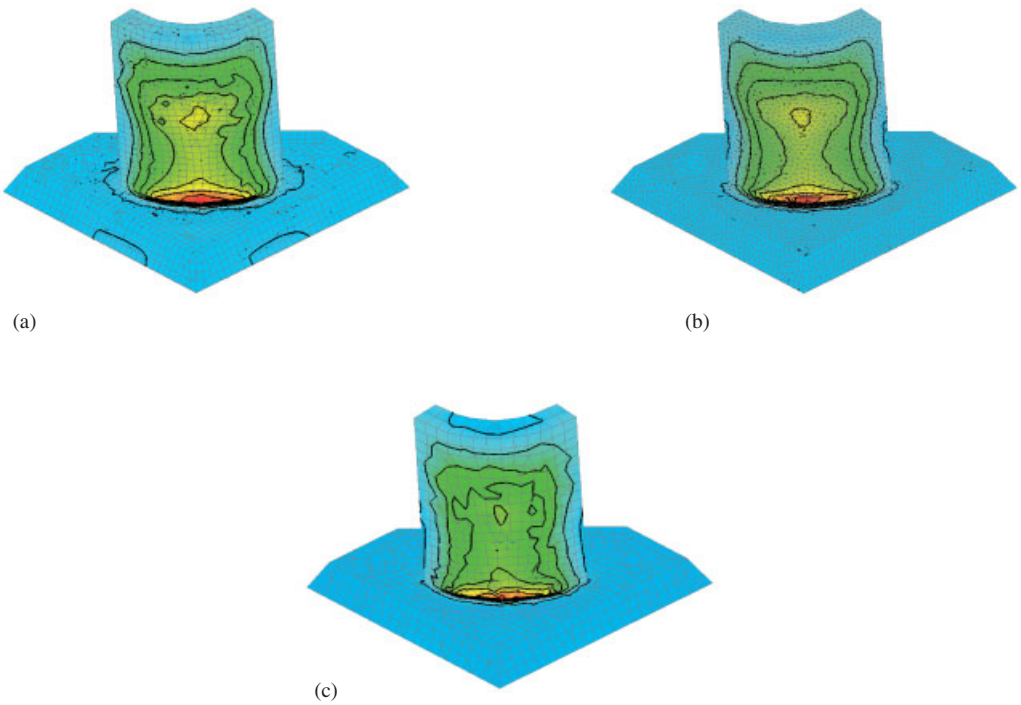


Figure 23. Stress distribution obtained in the experimental finite element analysis: (a) stress distribution obtained by the hex-dominant mesh with 21 185 elements; (b) stress distribution obtained by the dense tetrahedral mesh with 89 726 elements; and (c) stress distribution obtained by the graded hex-dominant mesh with 16 146 elements: Edge lengths of the elements at the bottom of the cylindrical feature are half the edge lengths of the elements at the top of the cylindrical feature.

computational cost for converting tetrahedrons to hexahedrons and prisms is $O(n)$, as discussed in Section 8, and it does not dominate the complexity.

The computational complexity of the bubble packing process becomes $O(n^2)$ if inter-bubble forces are computed for all pairs of cells exhaustively, and the proposed method applies a range-searching algorithm called kD-tree [25] to reduce the computational complexity to $O(n \log n)$. Since two too-distantly-separated cells do not interact with each other, the computation of the inter-bubble force between two non-interacting cells must be excluded from the computation of the inter-bubble forces to improve the computational efficiency. When creating a uniform mesh, the computational complexity can be reduced to $O(n)$ by applying a rectangular grid [26]. However, a rectangular grid becomes inefficient for a graded mesh. In fact, the experiments performed in this research showed that a kD-tree performs faster than a rectangular grid for graded meshes. A kD-tree is similar to a binary search tree, but a kD-tree is designed to search nodes distributed in a k-dimensional space while a binary search tree is appropriate for only one-dimensional space. A kD-tree can find neighbour cells of a cell in $\log n$ steps, where n is the number of cells, and inter-bubble forces are computed only between neighbouring bubbles. Since the number of neighbour bubbles of each bubble is almost constant, and there are n bubbles, and $\log n$ steps are required for each bubble to

Table III. Computational time for creating meshes with various resolution.

Number of nodes	Packing	Advancing front	Tet mesh to hex-dominant mesh conversion	Total
2367	428.6	23.7	8.9	461.2
2782	503.0	30.0	11.0	544.0
3176	575.0	33.0	13.8	621.8
4703	873.8	71.2	21.1	966.1
7316	1405.6	117.4	35.5	1558.5

find its neighbours, total computational complexity is $O(n \log n)$. The computational cost of building a kD-tree is also $O(n \log n)$.

Finding a triangle-node pair that yields a valid tetrahedron during the advancing front process is also a combinatorial problem, and its computational complexity easily becomes $O(n^2)$ without a sophisticated range-searching algorithm, so the proposed method again makes use of the kD-tree method to reduce the computational complexity to $O(n \log n)$. For each triangle included in the front, only a limited number of nodes can be connected to the triangle to create a valid tetrahedron, and those candidate nodes are located within a certain distance from the triangle. Thus, the proposed method makes a list of candidate nodes located within a certain distance from a triangle, and only triangle-node pairs consisting of the triangle and a node within the candidate list are tested. If the nodes are well spaced, then the node to be connected to the triangle is located in the domain bounded by a sphere about the triangle. The size of the sphere is set so that its radius is $2\sqrt{3}$ times larger than the desired edge length given by $\mathbf{M}$ at the triangle. A kD-tree efficiently finds all nodes located within the bounding box of such sphere. Since the number of the candidate nodes is almost constant for every triangle, and a candidate node list is made every time one tetrahedron is created, and $\log n$ steps are required to create a candidate node list, the total computational complexity becomes $O(m \log n)$ where m is the product of the number of tetrahedrons and the average number of candidate nodes. However, since m is nearly proportional to the number of nodes, n, $O(m \log n) = O(n \log n)$.

Some tests are performed to verify the theoretical computational complexity of the proposed method; hex-dominant meshes with various resolutions are created with the proposed method, and the computational time required for creating each mesh is measured. Table III shows the computational time required for creating a mesh and the number of nodes. The required computational time is also plotted against the number of nodes in Figure 24. The test is performed on a Pentium III 1 GHz PC with 1 GB RAM.

The plot indicates that the actual computational time agrees with the theoretical computational complexity of $O(n \log n)$. The plot of $y = n \log n$ almost becomes a straight line for large n because the curvature of the curve, or the second derivative of the curve, is $1/n$ and goes to zero for large n. Since the plot shows almost a straight line, it is reasonable to assume that the actual computational time agrees with the theoretical computational complexity of $O(n \log n)$.

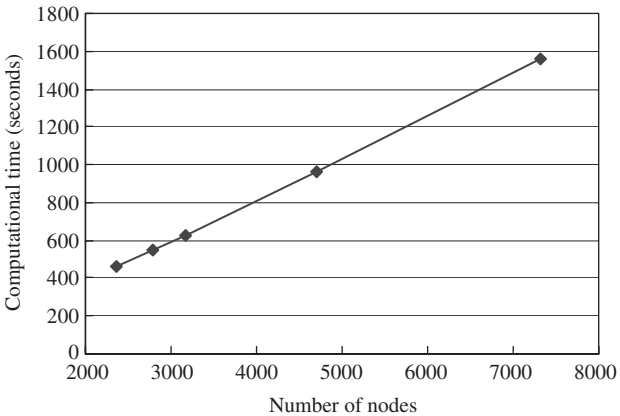


Figure 24. Plot of computational time for creating a mesh with different resolution.

12. CONCLUSION

This paper has presented a new method for creating a hex-dominant mesh consisting of hexahedrons, prisms and tetrahedrons. The process is fully automatic, and the output hex-dominant mesh show high quality. A hex-dominant mesh combines good aspects of a tetrahedral mesh and an all-hex mesh: the grading of element size can be controlled precisely as in a tetrahedral mesh, and a hex-dominant mesh gives a more accurate solution with fewer elements than a tetrahedral mesh.

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 tetrahedral mesh based on the nodes created in the second step, and (3) converting a tetrahedral mesh to a hex-dominant mesh. The proposed method avoids ill-shaped elements induced by nodes located too close together 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.

Although a hex-dominant mesh has some non-conforming quadrilaterals, which induce errors in the finite element analysis, several solutions are available; (1) reducing errors by MPCs as presented by Dewhirst *et al.* [9, 10]; (2) reducing errors by the technique allowing connection between dissimilar surface meshes as presented by Dohrmann *et al.* [11]; or (3) eliminating non-conforming quadrilaterals by introducing pyramids as presented by Owen *et al.* [12]. However, these techniques that cope with non-conforming quadrilaterals are not yet developed for some analyses such as fluid mechanics simulations. The future expansion of those techniques to more general analyses will increase the applicability and value of the proposed hex-dominant mesh generation techniques.

ACKNOWLEDGEMENT

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

REFERENCES

1. Owen SJ, Saigal S. H-Morph: an indirect approach to advancing front hex meshing. *International Journal for Numerical Methods in Engineering* 2000; **49**:289–312.
2. Meyers RJ, Tautges TJ, Tuchinsky PM. The 'Hex-Tet' Hex-dominant meshing algorithm as implemented in CUBIT. *Proceedings of 7th International Meshing Roundtable*, Dearborn, Michigan, 1998; 151–158.
3. Meshkat S, Talmor D. Generating a mixed mesh of hexahedra, pentahedra and tetrahedra from an underlying tetrahedral mesh. *International Journal for Numerical Methods in Engineering* 2000; **49**:17–30.
4. Becker EB, Carey GF, Oden JT. *Finite Elements*: *An Introduction*, vol. 1. Prentice Hall: Englewood Cliffs, 1981.
5. Owen SJ. Non-simplicial unstructured mesh generation. *Ph.D. Thesis*, The Department of Civil and Environmental Engineering, Carnegie Mellon University, 1999.
6. Yamakawa S, Shimada K. HEXHOOP: modular templates for converting a Hex-dominant mesh to an all-hex mesh. *Proceedings of 10th International Meshing Roundtable* 2001; 235–246.
7. Shimada K, Liao J-H, Itoh T. Quadrilateral meshing with directionality control through the packing of square cells. *Proceedings of 7th International Meshing Roundtable*, Dearborn, Michigan, 1998; 61–75.
8. Itoh T, Shimada K, Inoue K, Yamada A, Furuhata T. Automated conversion of 2D triangular mesh into quadrilateral mesh with directionality control. *Proceedings of 7th International Meshing Roundtable*, Dearborn, Michigan, 1998; 77–86.
9. Dewhirst DL, Grinsell PM. Joining tetrahedra to hexahedra. *Proceedings of MSC World Users' Conference* 1993 (http://www.mscsoftware.com/support/library/conf/).
10. Dewhirst DL, Vangavolu S, Wattrick H. The combination of hexahedral and tetrahedral meshing algorithms. *Proceedings of 4th International Meshing Roundtable*, Albuquerque, New Mexico, 1995; 291–304.
11. Dohrmann CR, Key SW, Heinstein MW. Methods for connecting dissimilar three-dimensional finite element meshes. *International Journal for Numerical Methods in Engineering* 2000; **47**:1057–1080.
12. Owen SJ, Canann SA, Saigal S. Pyramid elements for maintaining tetrahedra to hexahedra conformability. In *Trend in Unstructured Mesh Generation*, AMD-vol. 220. ASME: New York, 1997; 123–129.
13. Tuchinsky PM, Clark BW. The "HexTet" Hex-dominant automesher: an interim progress report. *Proceedings of 6th International Meshing Roundtable*, Park City, Utah, 1997; 183–193.
14. Blacker TD, Meyers RJ. Seams and wedges in plastering: a 3-D hexahedral mesh generation algorithm. *Engineering with Computers* 1993; **2**:83–93.
15. Yamakawa S, Shimada K. High quality anisotropic tetrahedral mesh generation via ellipsoidal bubble packing. *Proceedings of 9th International Meshing Roundtable*, New Orleans, Louisiana, 2000; 263–273.
16. Joe B. Construction of three-dimensional improved-quality triangulations using local transformations. *SIAM Journal on Scientific Computing* 1995; **16**:1292–1307.
17. Löhner R, Cebral JR. Parallel advancing front grid generation. *Proceedings of 8th International Meshing Roundtable*, Lake Tahoe, California, 1999; 67–74.
18. Blacker TD, Stephenson MB. Paving: a new approach to automated quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering* 1991; **32**:811–847.
19. Watson DF. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *The Computer Journal* 1981; **24**:167–172.
20. Bower A. Computing Dirichlet tessellations. *The Computer Journal* 1981; **24**:162–166.
21. George PL, Hecht F, Saltel E. Automatic mesh generator with specified boundary. *Computer Methods in Applied Mechanics and Engineering* 1991; **92**:269–288.
22. George PL. Tet meshing: construction, optimization and adaptation. *Proceedings of 8th International Meshing Roundtable*, Lake Tahoe, California, 1999; 133–141.
23. Knupp PM. 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* 2000; **48**:1165–1185.
24. *ANSYS*: ANSYS Inc. (http://www.ansys.com).
25. Sedgewick R. Range searching. In *Algorithms in C++*. Addison-Wesley: Reading, MA, 1992;373–388.
26. Heckbert P. Fast surface particle repulsion. *CMU Computer Science and Technology Report* CMU-CS-97-130, 1997, Carnegie Mellon University.

