Effective Adaptation Technique for Hexahedral Mesh

Yoshitaka Wada(1), Hiroshi Okuda(2)

(1) Department of Computational Earth Sciences, Research Organization for Information Science and Technology (RIST), 2-2-54 Nakameguro, Meguro-ku, Tokyo, Japan
E-mail: wada@tokyo.rist.or.jp Tel: +81-3-3712-5321 Fax: +81-3712-5552
(2) Department of Quantum Engineering & Systems Science, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan
E-mail: okuda@q.t.u-tokyo.ac.jp Tel: +81-5841-7426 Fax: +81-3-3818-3455

Abstract

This paper describes a simple and effective adaptive technique for mesh refinement in finite element method models. The proposed refinement method is based on a modified octree refinement approach, and can be applied to an arbitrary hexahedral unstructured mesh. The implementation, Hex-R, is used in conjunction with the finite element viewer GPPView, and applied to a complex model. The proposed method is demonstrated to be capability of easily and reliably generating an adaptive mesh.

Introduction

Adaptivity is one of the most important characteristics of effective solutions to finite element method (FEM) analysis tasks. Many studies on adaptive mesh generation and error estimation have been conducted to date, however the complex implementation of such systems remains a problem. This is further complicated by the fact that mesh generation in general is a problematic issue because there is no guarantee that mesh generation will succeed given an arbitrary shape and node distribution.

Adaptive mesh generation is essential in computational fluid dynamics (CFD), and development in this area has advanced significantly, particularly for aerospace applications. Structured meshes are widely used as non-adaptive solutions, and generate structured grids without fail. Tetrahedral unstructured meshes are perhaps the widest-used adaptive technique, but as with all adaptive systems, grid refinement is very complex and challenging. Hexahedral meshes are employed for mechanical analysis, particularly in contact and large deformation problems. However, the hexahedral mesh has inherent problems associated with the geometrical restriction, rendering it difficult to fill arbitrary domain. Hexahedral mesh generation is also very time consuming compared to structured and tetrahedral meshes. In general, as information on object geometry or geometric curved surfaces is required, the mesh generation process is time- and labor-intensive in terms of both development and execution.

The present study is conducted as a part of research on a parallel finite-element platform for solid earth simulation named GeoFEM [1]. One of the targets of GeoFEM is the development of FE code for earthquake generation cycles. In the analysis, the storage and release of stresses at plate boundaries are modeled. In order improve the accuracy of such analyses, adaptive or multigrid finite element methods are required. The complexity of this problem is further increased by the necessity to model the complex geometry of the plate system using a hexahedral mesh and complex geometries.

Here we propose a simple and effective hexahedral mesh refinement method, based on a modified octree-based hexahedral mesh generation technique [2]. The refinement method employs local mesh refinement and the creation of mesh connectivity. A method has already been proposed for fully automatic hexahedral mesh generation based on isomorphism [3]. In that proposal, octree-based generation is employed for filling the interior of the object with arbitrary density. In the present study, we find that octree-based generation can be used to control the density of elements and as such is very useful for adaptive and multigrid mesh generation. We present the effectiveness of the refinement method and the application of the octree-based method to an arbitrary hexahedral mesh. The proposed technique is implemented as a part of the GeoFEM software.

General Strategy

Figure 1 shows the processes involved in hexahedral refinement using a modified octree-based method. Figure 1(a) shows an initial cubic mesh, and Figures 1(b) and (c) show the first refinement instruction and result. Figures 1(d) to (i) show the second to fourth (last) refinement instruction and result. As shown, hexahedral mesh refinement can be applied to any number of elements. The realization of reliable refinement is strongly dependent on the appropriate refinement patterns.

The octree algorithm accurately represents geometric shape or features and can be used to optimize and reduce a computational search. The octree structure used here has been modified slightly: The primary octree structure has 27 sub-octrees, called an octant. This 27-tree octant is suitable for hexahedral mesh refinement because it can be easily connected to sub-octants using the patterns shown in Figure 2. In reference to the figure, pattern 4 (P4) is located at the center of the local refinement, and pattern 3 (P3) faces each face of P4 (i.e., 6 P3s surround one P4). Pattern 2 (P2) is generated adjacent to P3, and pattern 1 (P1) is generated adjacent to P2. The assignment of patterns is detailed in the next section.

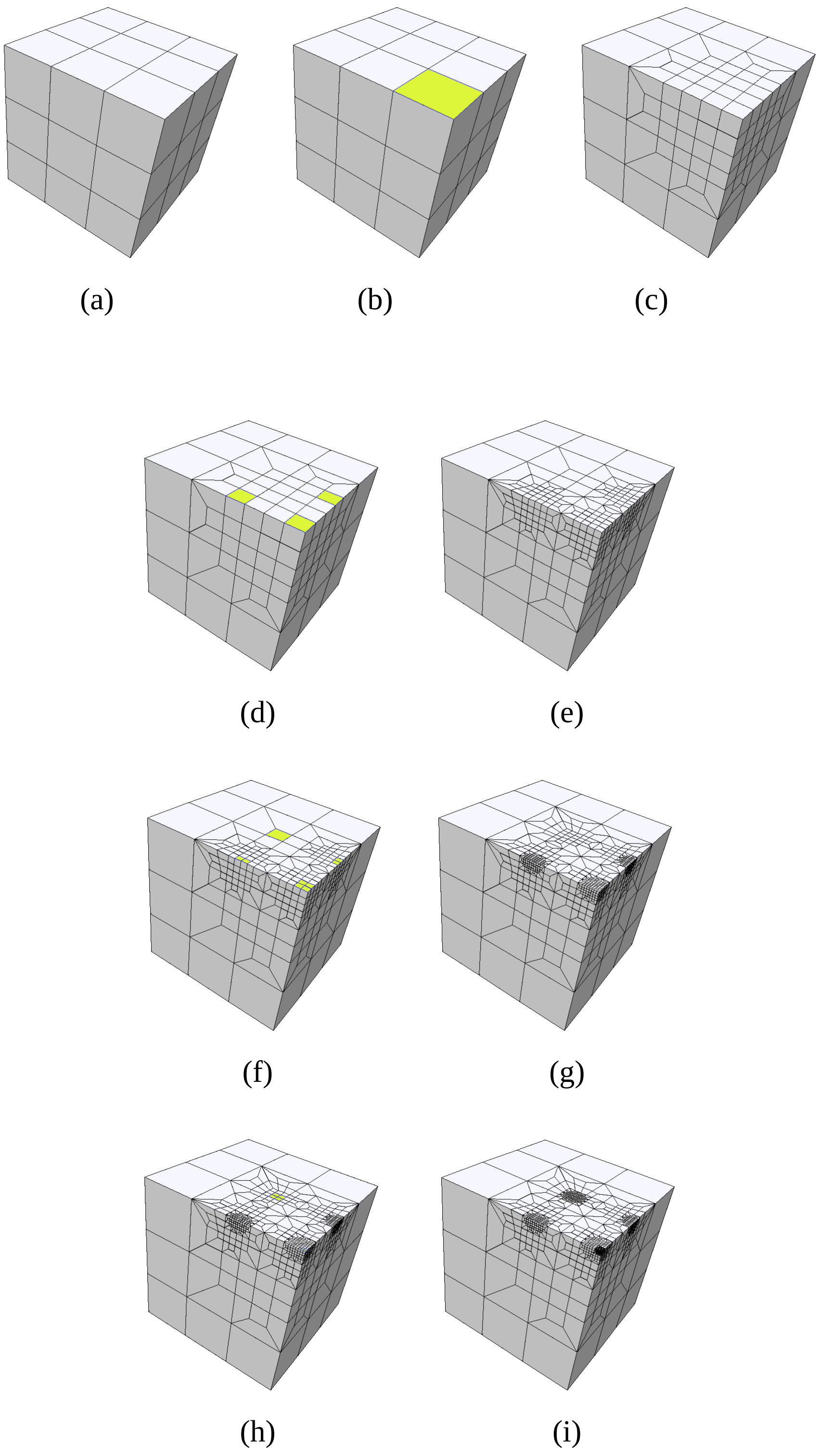


Figure 1. Recursive mesh refinement

The modified octree-based refinement process consists of 3 steps.

- An octree level is assigned to the elements to be refined; a higher octree level means a finer mesh is to be generated.
- The octree structure is examined in order to obtain sufficient information (pattern rotation and location) to apply mesh refinement patterns without fail.
- Refinement patterns are applied to the mesh.

We will describe the basic algorithm for 2D and 3D processes in the next section. The 3D process is essentially the same as the 2D process, except for the use of 4 patterns in 3D and 3 patterns in 2D.

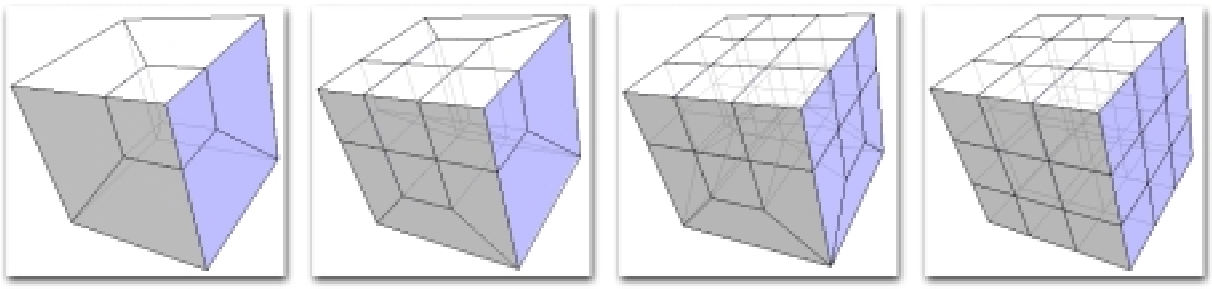
Modified Octree-Based Mesh Refinement

As mentioned in the previous section, the modified octant has 27 sub-octants. This 27-tree octant can be applied directly to arbitrary hexahedral meshes.

Basic 2D Processing

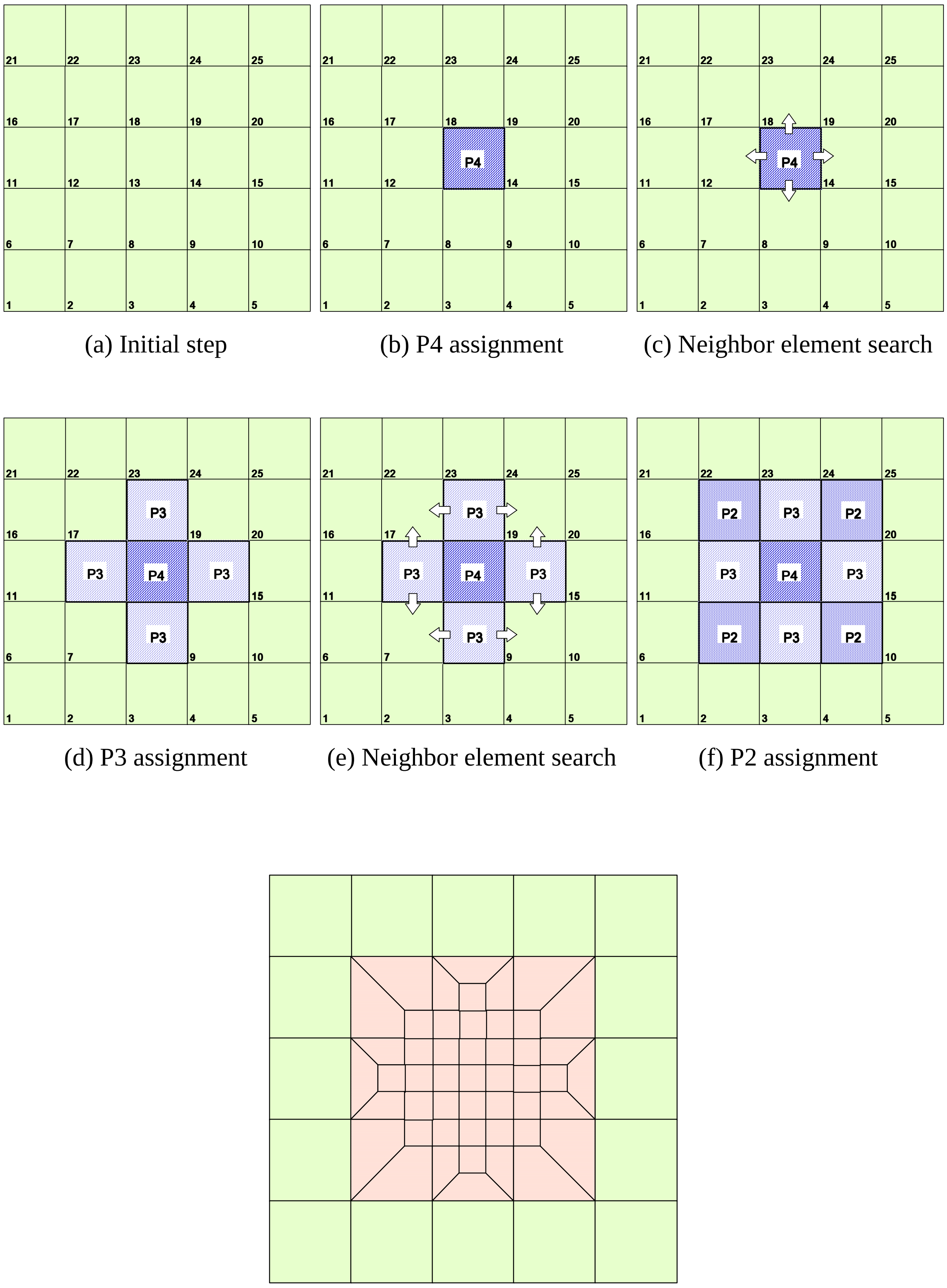
Here we assume an initial mesh as shown Figure 3(a). P4 is initially assigned to element 13 (Fig. 3(b)). Then, the 4 elements fully facing element 13, in this case, elements 12, 14, 8 and 18, are selected and assigned as P3. Elements adjacent to the P3 elements are then assigned as P2, as shown Figures 3(e) and (f). A final, regular pattern configuration is obtained. Figure 3(g) shows the 1st-level refined mesh.

Figure 4 shows an example for refinement of an actual unstructured mesh. P4 is assigned to element 13, and P3 is assigned to the 4 adjacent elements. Figure 4(d) illustrates the irregular pattern configuration. A complete mesh can only be achieved if element 12 is also assigned as P2. This is done by examining the number of P2 assignments to each element, as shown in Figure 4(e). In a regular 2D or 3D model, each element is assigned twice. However, in this case for example, elements 10 and 19 are only assigned once. We can therefore detect irregular configurations by monitoring the number of assignments to each element. If an irregular assignment is detected, 1 count adds to number of assignments at 9th and 18th element. In regular case as shown Figure 3, the amount of count should be 2 at adjacent element with P3, but in this case the number is 1 at element 10 and 19. Therefore irregular can be detected by count the number of the reference. P2 is assigned from element 10 and 19 once again in appropriate direction. Then the number becomes 2 at element 12. Finally P2 is assigned to 12th element from both 10th and 18th element. After this process, all elements are assigned as P2 twice, and a stable pattern is achieved. Figure 4(h) shows the 1st-level refined mesh.



(a) Pattern 1 (b) Pattern 2 (c) Pattern 3 (d) Pattern 4

Figure 2. Refinement Patterns

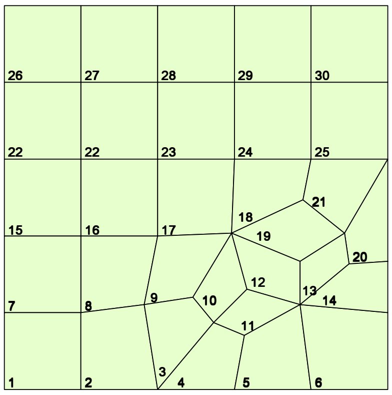


(a) Initial step (b) P4 assignment (c) Neighbor element search

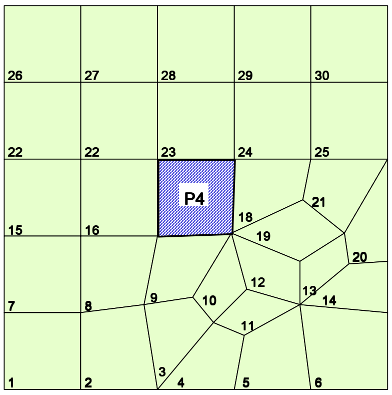
(d) P3 assignment (e) Neighbor element search (f) P2 assignment

(g) Refined mesh

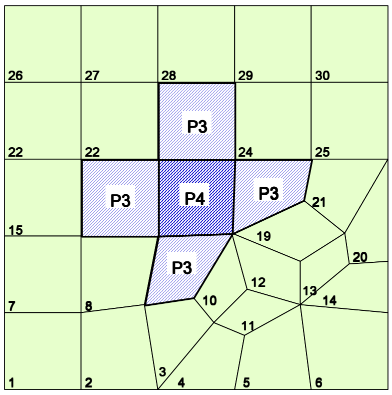
Figure 3. Basic pattern assignment



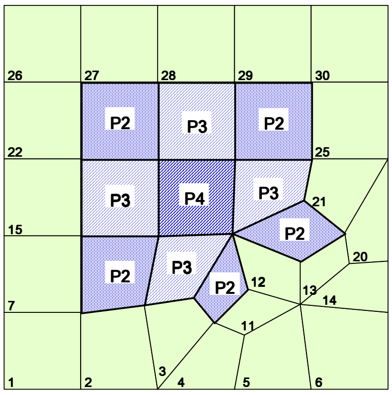
(a) Initial mesh



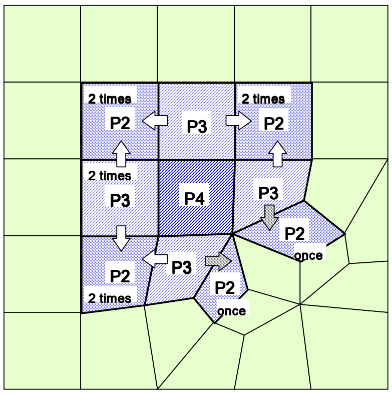
(b) P4 assignment



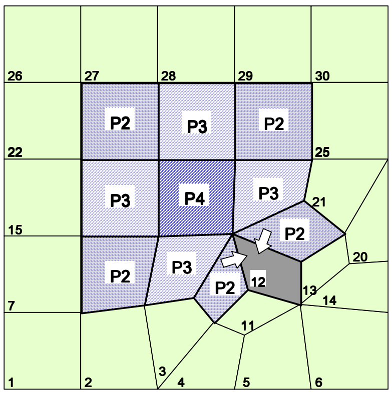
(c) P3 assignment



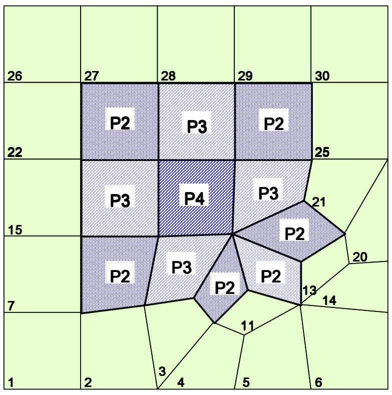
(d) P2 assignment



(e) Number of assignments



(f) P2 assignment



(g) Final pattern

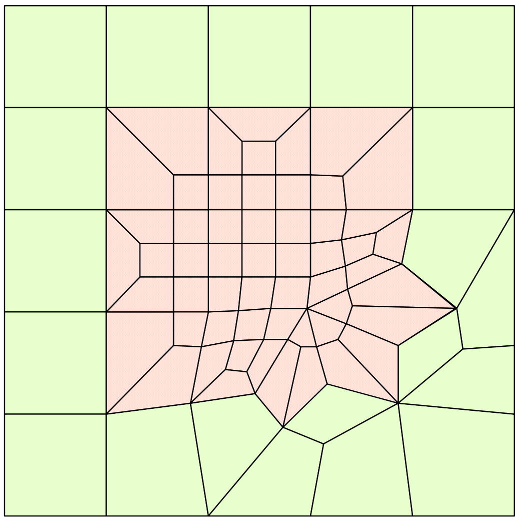
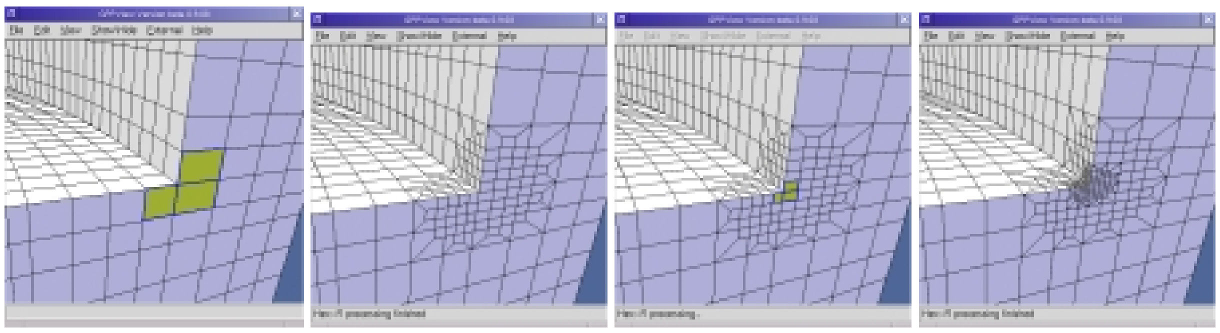

(h) Refined Mesh

Figure 4. Actual pattern assignment

Implementation of Modified Octree-Based Refinement

Figure 5 shows an example of recursive refinement using Hex-R [4]. Hex-R is an implementation of a modified octree-based hexahedral mesh refinement. Figure 5 shows a process of recursive mesh refinement on GPPView [5], which is a finite element viewer under development as part of the GeoFEM project. The first element region to be refined is selected (Fig. 5(a)) and a refined mesh is generated (Fig. 1(b)), followed by selection and refinement of a subsequent region (Figs. 5(c) and (d)).

GPPView is a viewer for finite element applications and has the ability to communicate with external applications such as Hex-R to process a mesh. This communication function allows mesh processing to be extended to most FE formats. GPPView has recently been released for free download from the GeoFEM website [5].


(a) Initial selection (b) First refinement (c) Second selection (d) Second refinement

sFigure 5. Recursive mesh refinement using Hex-R

Examples and Performance

Implementation and Testing Environment

Hex-R was implemented in C++ and was run on a UNIX-based computer (FreeBSD Relase 4.3) with 2 parallel Pentium III processors.

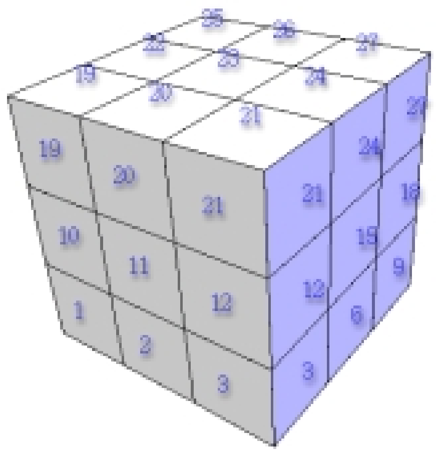
Simple Cube Model

As mentioned above, the 2D algorithm used to generate a stable pattern configuration can be extended directly to the 3D case by adding an additional iteration with pattern P1. Figure 6(a) shows the initial mesh and element numbers. The mesh consists of 27 cubic elements.

In the first step, P4 is assigned to element 14, which is located at the center of the mesh. In the second step, P3 is assigned to 6 elements with faces in contact with element 14. In the third step, P2 is assigned to the elements adjacent to the P3 elements. In final step, P1 is assigned to elements adjacent to P2 elements. This regular configuration results in the refined mesh shown in Figure 7. The refinement process produces 304 new nodes and 323 elements.

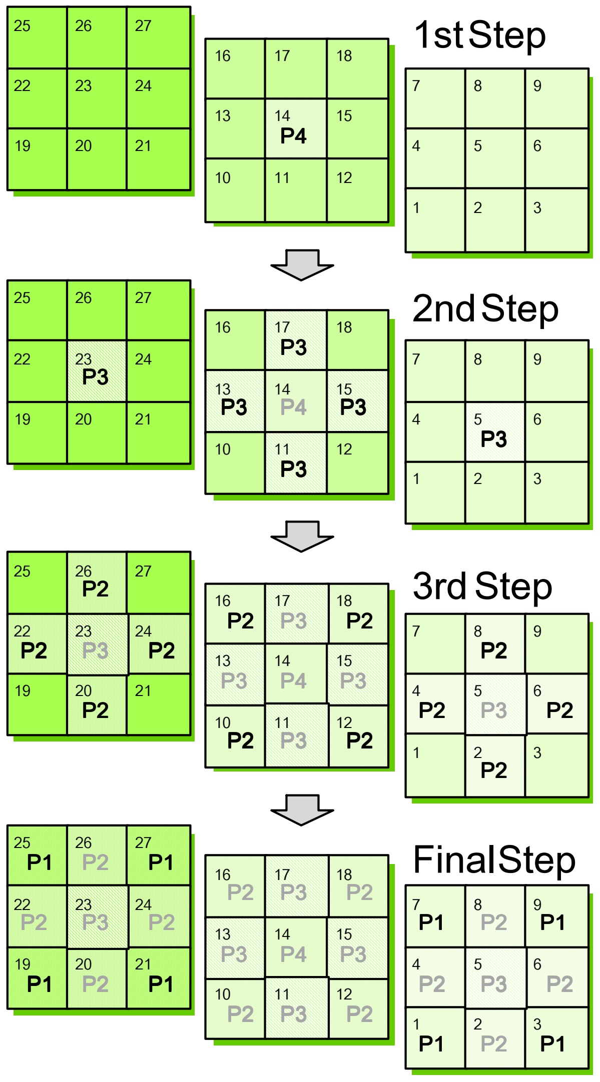
Tube Sheet Model

Figure 8 show the finite element model for a tube sheet as used in nuclear power plants. The model initially consists of 55,877 nodes and 45,624 elements. As heat is repeatedly generated in the vicinity of each hole, stress is concentrated between holes. In this case, the internal hole surfaces are selected for refinement (Fig. 8(b)). Figures 8(c) and (d) show the refined mesh. After 1st-level refinement, the mesh consists of 170,205 nodes and 197,904 elements.



(a) Initial mesh

Figure 6. Pattern assignment in 3D



(b) Consistent configuration of patterns in 3D

Figure 6. Pattern assignment in 3D

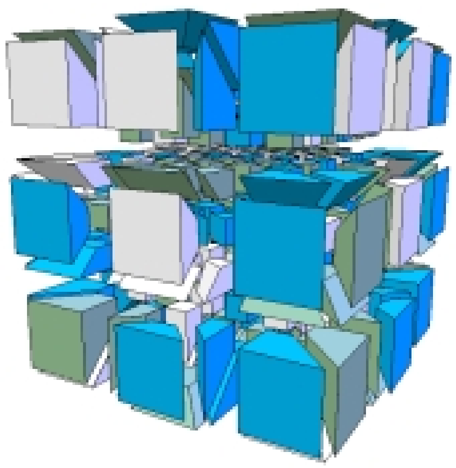
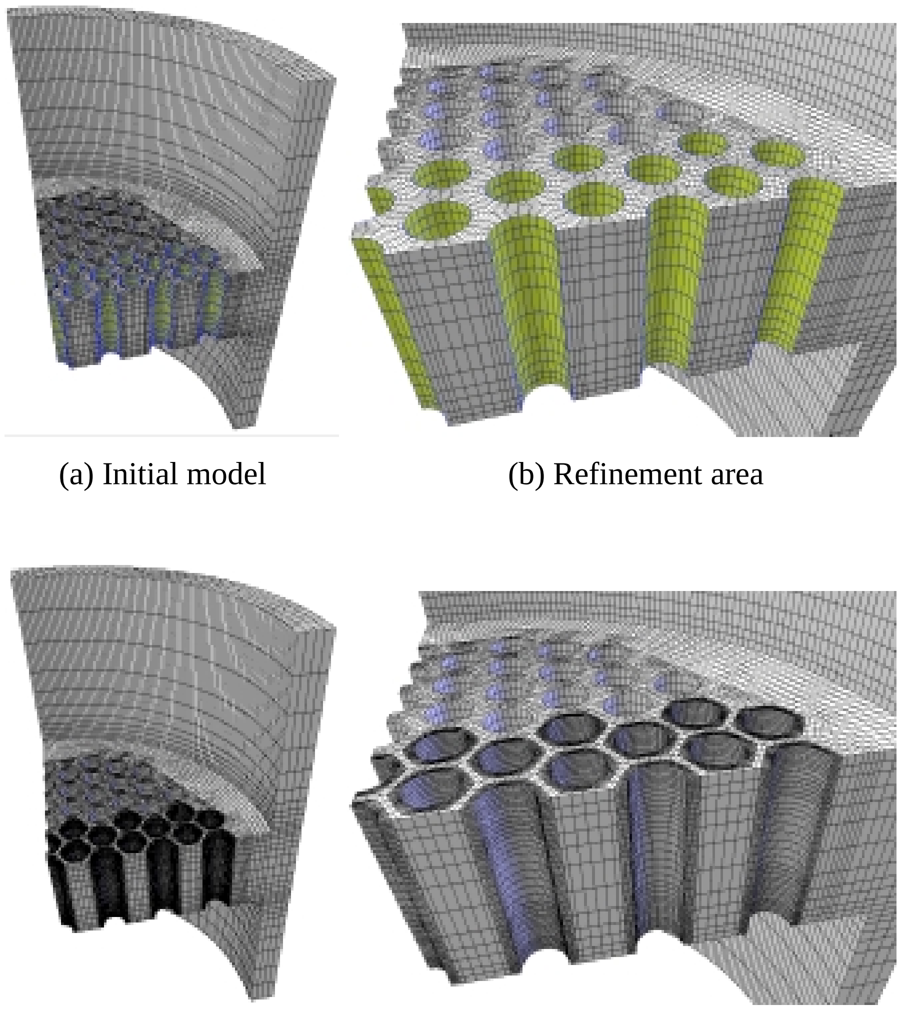


Figure 7. Structure of refined mesh



(a) Initial model

(b) Refinement area

(c) Refined model

(d) Refined holes

Figure 8. Refinement of tube sheet

Performance

Figure 9 shows the relationship between the number of starting elements and computation time. Computation time initially increases quite rapidly with the number of elements. However, refinement becomes computationally efficient when the number of starting elements exceeds approximately 1000.

Importantly, the proposed method is suitable for parallelization as long as the correct juxtaposition of partitioned meshes can be guaranteed. We are currently developing a parallel version of Hex-R, focusing on maintaining consistency between partitioned meshes. The parallel process for this algorithm is as follows.

1. Data for refinement, i.e., the set of element numbers and refinement level, is prepared.
2. The initial mesh is partitioned into a number of smaller meshes equal to the number of processors in the parallel environment. The program Metis [6] is suitable for mesh partitioning. Partitioned meshes should have local element IDs and parent element IDs corresponding to the ID in the initial mesh.
3. The data prepared in first step is input into Hex-R, and the refinement is generated on each processor in parallel. The final refined partition meshes should have pattern IDs.
4. All partitioned meshes are merged into a single mesh and inconsistencies/misalignments between partitioned meshes are resolved.

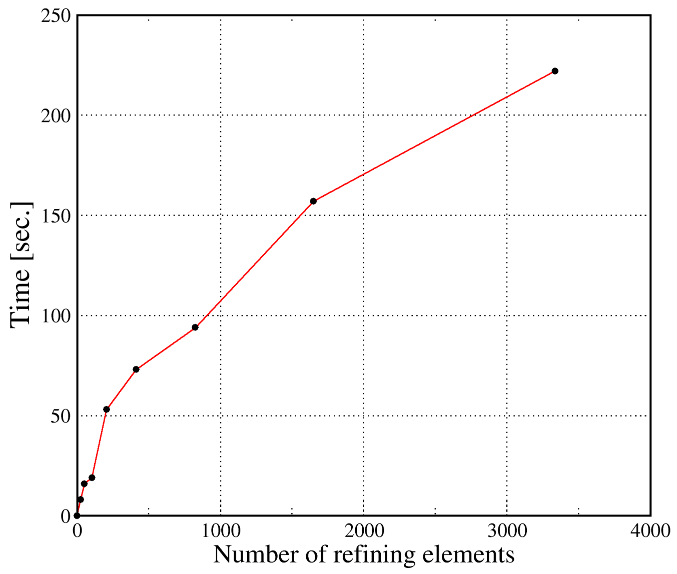


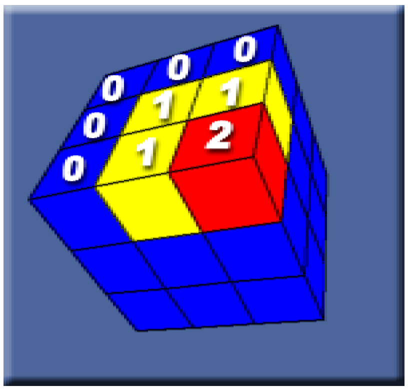
Figure 9. Comparison between initial number of elements and computation time

Mesh Adaptation using Hex-R

Hex-R can be used to easily create an adaptive mesh, with certain limitations. The method is a refining method and not a regeneration method, which means elements are only subdivided into fine elements. Secondly, the refinement area must be concave. If the region is not concave, the refinement will extend beyond the original boundaries, increasing the number of nodes.

These limitations, however, reduce complexity and ensure that mesh generation succeeds. Figure 10 illustrates simple case of adaptive data and a mesh. Figure 10(a) shows the adaptive information used for refinement, and Figure 10(b) shows a mesh refined based on the adaptive information.

In order to generate an adaptive mesh, error estimation or physical data needs to be converted to information that can be used for the subdivision of each element. There are 3 levels of subdivision in the information provided in Figure 10(a). The blue area (0) indicates that the area is not to be subdivided. Yellow (1) indicates that the element is to be divided into 27 elements. Red (2) indicates that the element is to be divided into 729 (27^2) elements. In this way, adaptive level information is converted into a consistent pattern configuration, and an adaptive mesh can be obtained.



(a) Adaptive information on initial mesh

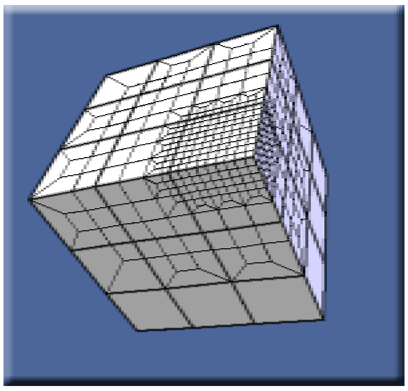

(b) Refined mesh for adaptation

Figure 10. Adaptive instruction and actual mesh

Conclusion

We propose an effective mesh refinement and adaptive mesh generation method that produce refined meshes without fail. In further research we will apply the method to several analyses and investigate the effect of increasing the number of nodes on the accuracy of FE analysis.

References

[1] GeoFEM Web site: http://geofem.tokyo.rist.or.jp/

[2] R. Schneiders, R. Sxhindler and F.Weiler, 1997, Octree-based Generation of Hexahedral Element Meshes, 6th Int. Meshing Round Table, http://www.andrew.cmu.edu/user/sowen/abstracts/Sc252.html

[3] R. Schneiders, 1995, Automatic generation of hexahedral finite element meshes, Proc. 4th Int. Meshing Round Table, pp. 103-114

[4] Y. Wada, 2000, Effective Adaptation Technique for Hexahedral Mesh, 2nd ACES Workshop Abstract, pp. 441-442

[5] Y. Wada, 2000, 2001, GeoFEM Pre/Post Viewer/Editor: GPPView, http://www.tokyo.rist.or.jp/GPPView/

[6] G. Karypis, Serial graph partitioning and sparse matrix ordering, http://www-users.cs.umn.edu/~karypis/metis/