Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 124 (2015) 31 – 43

Procedia Engineering

www.elsevier.com/locate/procedia

24th International Meshing Roundtable (IMR24)

A Template-Based Approach for Parallel Hexahedral Two-Refinement

Steven J. Owen[a,*], Ryan M. Shih[b]

[a]*Sandia National Laboratory, Albuquerque, New Mexico, USA.*
[b]*University of California, Berkeley, California, USA.*

Abstract

In this work we provide a template-based approach for generating locally refined all-hex meshes. We focus specifically on refinement of initially structured grids utilizing a 2-refinement approach where uniformly refined hexes are subdivided into eight child elements. The refinement algorithm consists of identifying marked nodes that are used as the basis for a set of four simple refinement templates. The target application for 2-refinement is a parallel grid-based all-hex meshing tool for high performance computing in a distributed environment. The result is a parallel consistent locally refined mesh requiring minimal communication and where minimum mesh quality is greater than scaled Jacobian 0.4 prior to smoothing.

Peer-review under responsibility of organizing committee of the 24th International Meshing Roundtable (IMR24)

Keywords: hexahedron; mesh refinement; 2-refinement; adaptivity; parallel;

1. Introduction

Massively parallel platforms, such as those deployed at the U.S. Department of Energy Laboratories, have enabled computational simulation of enormous complexity. For applications requiring hexahedral elements, traditional methods of mesh generation can require significant user interaction which will not easily scale for these problems. Fully automatic, scalable and embedded meshing methods are an increasingly important requirement for these next-generation computing platforms. Mesh generation based on an overlay grid procedure is an ideal candidate for high performance computing, however to be effective it must provide for geometry-sensitive mesh size adaptation.

Overlay grid procedures for generating all-hex meshes [1][2][3] usually rely on some form of refinement strategy to capture small features. Most of these methods begin with a regular three-dimensional Cartesian grid that is adaptively refined based on various geometric criteria to form an octree subdivided mesh. A Boundary representation (B-rep) of the geometry of interest is then super-imposed on the octree mesh, where nodes are snapped to the geometry and elements falling outside of the B-Rep are discarded.

* Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000

E-mail address: sjowen@sandia.gov

Peer-review under responsibility of organizing committee of the 24th International Meshing Roundtable (IMR24)
doi:10.1016/j.proeng.2015.10.120

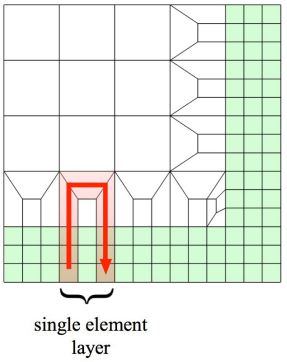

Fig. 1. Example 2D 3-refinement showing pillow loops accomplished within a single layer of elements.

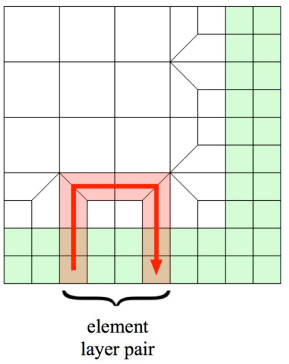

Fig. 2. Example 2D 2-refinement showing pillow loops requiring at least a pair of adjacent element layers.

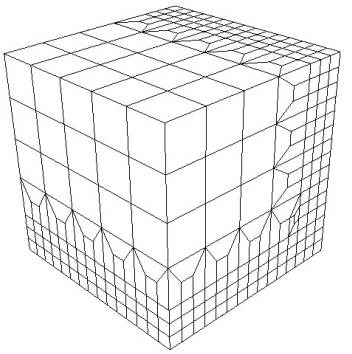

Fig. 3. Example 3D mesh using a 3-refinement procedure

In order to maintain continuity between refined and unrefined elements in the mesh, transition patterns are normally imposed. These patterns can utilize either a 3-refinement or a 2-refinement methodology. For 3-refinement, each edge of the uniformly refined grid is divided into three segments. On a three-dimensional hex element, this results in a 3x3x3 subdivision or 27 elements. 2-refinement, on the other hand, will split each edge in two resulting in a 2x2x2 subdivision with 8 elements.

Most refinement operations can be thought of as introducing a pillow layer of hexes surrounding a column of hexes. Figure 1 illustrates a 2D 3-refined mesh where the green elements were initially marked for uniform refinement. An example continuous pillow layer of quads (shown in red) is shown wrapping a single column of quads, noting that the same pattern is repeated throughout the mesh. Figure 2 illustrates a 2-refined mesh with a similar pillow layer surrounding two columns of quads. In 3-refinement, We note that the pillow layer can be accomplished within a single quad layer, whereas 2-refinement requires at least a pair of quad columns to accomplish the pillow. This problem extends to 3D (shown in figure 3) where sheets of hexes must be introduced to accomplish the refinement. For 3-refinement, since the pillowed sheet of hexes can be effected within a single column of hexes, each refinement transition can be performed independently and within a single element, making its implementation relatively straightforward. In contrast, 2-refinement must determine a consistent pairing of hex layers to effect the refinement transitions, making its implementation more challenging.

Although the 3-refinement strategy can produce far more elements resulting in high mesh size gradients in transition regions, it remains the most popular form of refinement because of its ease of implementation. The 3-refinement pattern, illustrated in figure 1, has well-defined templates for transition elements that can be introduced based on a set of marked nodes. First introduced by Schneiders [5] the number and pattern of marked nodes on a cell define the precise subdivision template to be used. This deterministic template-based approach to refinement, is relatively well-understood and easy to implement. Although beneficial for implementation, the change in element size and resulting mesh quality in the transition regions for 3-refinement can be problematic.

Although 2-refinement, shown in figure 2, by most measures is a more desirable approach, its implementation is more difficult, particularly for parallel distributed domains. Complications in 2-refinement can arise when the pairing patterns to form transition elements from nearby refinement zones do not match, or when the pairing of element layers must extend across processor boundaries. Several methods have been proposed which appear to present good results for serial applications, however implementation details for some of these methods are sparse and their application to distributed memory parallel is not addressed.

We address the need for hexahedral mesh refinement for distributed memory parallel environments. A refinement strategy that uses a deterministic algorithm that yields the same results regardless of the domain decomposition strategy is desirable. Because templates can provide local criteria for subdivision, they provide an attractive solution for parallel applications where very little inter-processor communication is necessary.