Parallel Dynamic Simplical Meshes in OpenFOAM®

S. Menon $^{1}$ and David P. Schmidt $^{*,†}$

$^{1,2}$University of Massachusetts Amherst, 160 Governors Drive, Amherst MA 01003

March 14, 2011

*Corresponding Author: David Schmidt (schmidt@ecs.umass.edu)

Keywords: Parallel adaptation, unstructured meshes, conservative remapping

Computational simulations in deforming domains often require complex mesh adaptation. The goal of the present work is to provide continuous and autonomous adaptation that does not require prior knowledge of the boundary motion. Even in cases where the boundary motions are pre-determined, this adaptation scheme can be helpful if the geometry or motion is very complex, since our scheme is innately unstructured. Our approach is limited to simplical meshes, namely triangles or tetrahedra, due to their convenient geometric properties. This abstract summarizes recent progress in parallelizing the algorithms for use in distributed calculations.

Mesh motion is the preferred way of maintaining mesh quality, since it avoids interpolation errors and can be naturally included into the governing equations using relative fluxes. Instead of employing the existing OpenFOAM mesh motion techniques, we created an interface to the MESQUITE library for mesh motion [1]. The MESQUITE library treats mesh motion as an optimization problem and offers the user many options, such as a choice of mesh quality measure. Since MESQUITE is best-suited for the interior mesh motion, we use a parallelized Laplacian smoother based on the spring-analogy for the surface.

However, there are occasions where changes in the connectivity of the mesh are absolutely essential. If two vertices move far enough apart, their connecting edge must be split. Alternatively, edges can collapse to eliminate points that are nearly redundant. Cells can be locally reconnected using dynamic programming, as explained by Dai and Schmidt [2]. These types of changes typically involve only a few cells in a given time step.

These changes in connectivity cause new cells and faces to form, requiring the initialization of fields at these new locations. The simplest methods rely on interpolation or weighted averages of the data from the old mesh, prior to reconnection. We have derived and implemented a method that is both second-order accurate and conservative [3].

A two-dimensional demonstration is shown in Fig. 1. This calculation shows how, as a cylinder moves through the domain, the mesh adapts automatically. The scheme handles the large motion of the cylinder without crushing or distorting cells. The shapes of the sub-domains
Figure 1: A sequence of snapshots showing parallel mesh adaptation around a cylinder moving from left to right in the domain. The colors indicate the subdomain to which the cells belong.

Figure 2: A Gasoline Direct Injection (GDI) engine, colored by cell subdomain, showing halo meshes in white.

tend to become distorted, increasing surface area and potentially affecting parallel efficiency. The proposed solution to this issue is cell migration using existing OpenFOAM utilities.

OpenFOAM typically uses a halo-free domain decomposition paradigm for parallel computation. The parallelization of mesh adaptation has required some departures from the standard OpenFOAM approach. Changes in cell connectivity near processor boundaries may involve any number of processor subdomains. For example, an edge may be shared by six cells, each belonging to a different sub-domain. In order to handle these changes, the current work uses a halo, as shown in white in Fig. 2. Testing of the new algorithms using this Gasoline Direct Engine case are underway.

References

