Computers & Fluids xxx (2013) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Computers & Fluids

journal homepage: www.elsevier.com/locate/compfluid

Application of surface conformed linear mesh and data subdivision technique to a spinning projectile

Erdal Yilmaz *, Shahrouz Aliabadi

Northrop Grumman Center for HPC, Jackson State University, 1230 Raymond Rd., Jackson, MS 39204, United States

ARTICLE INFO

Article history:
Received 3 March 2013
Received in revised form 5 June 2013
Accepted 10 June 2013
Available online xxxx

Keywords:
Mesh multiplication
Mesh subdivision
Parallel computing
Unstructured mesh
Finite element
Finite volume
Spinning projectile

ABSTRACT

In this paper, we report a spinning projectile application of a parallel CFD program featuring surface-conformed isotropic mesh subdivision. The subdivision technique generates finer computational mesh without user interaction using a 3-D hybrid unstructured coarse base. The main motivation behind the subdivision technique is to overcome the bottleneck in generating and processing of the computational meshes with billions of elements. First, we generate a coarse mesh using any unstructured mesh generator. Then, we subdivide the coarse mesh to the level of resolution needed for the simulations. Finally, we conform mesh nodes on solid surfaces to the original geometry since linear subdivision ignores surface curvatures. We use K-D tree search algorithm in the surface mapping. To deform interior mesh nodes due to the surface correction, we use the spring analogy method since deformations are very small. Surface correction is implemented in parallel using the Message Passing Interface. The new mesh obtained from the isotropic subdivision preserves mesh density distribution of the original coarse mesh. The mesh subdivision with surface correction is integral part of our Variable Intensity Computational Environment. A projectile test case with and without spinning at different angles of attack are used to demonstrate the applicability of this method. Flow solutions are obtained using our compressible Navier–Stokes flow solver, CaMEL Aero, with the Detached Eddy Simulation turbulence model. Flow solutions and mesh subdivisions are performed in our parallel cluster at the Jackson State University.

Published by Elsevier Ltd.

1. Introduction

As engineering and scientific computational simulations reach tens of billions of computational mesh elements, not only fast computation and efficient communication among computational subdomains, but also efficient data structure to handle storage and processing of the data becomes very important. Generating such big mesh data would even be more challenging and time consuming than solving flow equations. Future large-scale problems are set to exploit planned deployment of the exascale parallel computing platforms (10^{18} operations per second) by 2018 in USA [1]. Storing a full-range of simulation results for later analysis becomes impossible due to technology trends. New approaches are needed "*In order to achieve scientific understanding from exascale simulations, visualization and data analysis techniques must reduce the amount of data to a smaller understandable representation. Advanced data reduction techniques including statistical sampling, compression, multi-resolution and science-based feature extraction approaches are needed in order to distill exascale simulation results.*" [1].

Most of the existing Computational Fluid Dynamics (CFD) flow solvers are second-order accurate in both space and time. Very fine meshes are required to resolve the complex flow features (vorticities, boundary layers, turbulent eddies, etc.) around complex geometries. Meshes with tens of millions of elements or even billions of elements are becoming more and more indispensable in practical high-fidelity flow simulations. With the availability of parallel and vector computers, CFD codes themselves are quite ready for this level of large-scale simulations. However, the mesh generation is still a bottleneck for very large problems. Most of the existing unstructured mesh generators run sequentially. Therefore, the size of a mesh generated by unstructured mesh generators is limited by the available memory on a single machine. To overcome this bottleneck, one can use a parallel mesh generator, which is still a challenging research topic and the technology is not mature yet. Though attempts for parallel mesh generation have resulted in significant improvements in time and total size of the mesh generated [2–4], this problem persists and scalability is sacrificed as the mesh size gets bigger and bigger [5]. Visualization of the mesh and flow solution data requires more memory and disk space than initially required in the mesh generation phase especially for unsteady simulations. Post-processing of very large data sets requires usually special research tools rather than most widely available commercial tools [6].

* Corresponding author. Tel.: +1 601 979-1826; fax: +1 (601) 979 1831.
E-mail address: erdal.yilmaz@jsums.edu (E. Yilmaz).

0045-7930/$ - see front matter Published by Elsevier Ltd.
http://dx.doi.org/10.1016/j.compfluid.2013.06.006

In extremely data-centric world, traditional ways of handling large data sets needs to be changed for various reasons. Perhaps the most practical and efficient way is to carry out the simulations on extremely fine resolution mesh for the sake of solution accuracy, which would result in tera-size data if not peta, while giving comfort to the users at the pre- and post-processing phase with affordable smaller mesh size.

Recently, we developed and demonstrated our Variable Intensity Computational Environment (VICE) approach [7,8], which is based on our earlier studies [9]. The VICE technology accommodates simulations of engineering and scientific problems with automatically generated hundreds of billions of mesh elements for increased accuracy while facilitating users to interact with at least two-to-three orders of magnitude smaller mesh size. For the sake of convenience, the mesh that users interact with is called *base-mesh* and the mesh that the parallel system interacts with for the flow simulations is called actual *computational mesh*. Successive mesh subdivisions, which lie behind the VICE technology, starts from the *base mesh* and goes up to final *computational mesh* by consecutive mesh subdivision or mesh multiplication. Our VICE environment involves correction of solid boundaries. Surface mesh on the solid boundaries is mapped to the original geometry after mesh-multiplication process. Since mesh-multiplication linearly subdivides mesh elements everywhere in the domain, curvatures on the solid surfaces were ignored between subdivision levels. The VICE technology can be employed in the exascale environment as a data reduction and in-situ visualization and data analysis tool as well as for any large scale scientific computations to avoid the bottleneck of dealing with huge computational data. More details of our VICE environment can be in our recent study [7].

In this paper, we apply our mesh subdivision technique with surface correction feature to study a spinning projectile. We covered supersonic flow at two different angles of attack and two different Mach numbers. Flow field solutions and lift and drag coefficients are presented and force coefficients are compared with an experiment as appropriate. Following sections of this paper will discuss the details of our methodology and results for a spinning projectile application.

2. Mesh and data format

The unstructured mesh considered in this paper must be conforming (without hanging nodes) and may contain elements of mixed types of tetrahedron, pyramid, prism, and hexahedron cells as illustrated in Fig. 1. The current mesh format has been widely used in the scientific computing community. The format contains one ASCII file, `mesh.info`, and three binary files; `mxyz`, `mien` and `mrng`, to describe a hybrid unstructured mesh.

`mesh.info`: This is an ASCII file containing three lines of each are `nn`, `ne`, and `nen`, where "`nn`" is the number of nodes, "`ne`" is the number of elements and "`nen`" is the number of nodes constituting an element. For hybrid meshes, `nen` will take the maximum value for all types of elements. For example, `nen` = 6 for hybrid tetrahedron-prism meshes.

`mxyz`: It is a binary floating-value array `xn(nsd,nn)` containing the coordinates of each node where "`nsd`" is the number of space dimensions (=2 for 2D; =3 for 3D).

`mien`: This binary file contains the element-node connectivity, integer array `ien(nen,ne)`. The values of the `ien` array components can take positive integers or just "−1". If the array component is a positive integer, the integer is the index of the node. If the array component has the value of "−1", it means that the node is non-existent for this element, which is a possible situation for hybrid meshes.

`mrng`: This binary file contains the dual information of elements, a binary integer array `rng(nef,ne)`, where "`nef`" is the number of faces constituting an element. For hybrid meshes, `nef` will take the maximum value for all types of elements. For example, `nef` = 5 for hybrid tetrahedron-prism meshes. The values of the `rng` array components can take positive, negative or zero integers. If the array component is a positive integer, then the face is on the boundary with the boundary number as that positive integer. If this array is a negative integer, then the face is an interior face with its adjacent element index as the absolute value of that negative integer. If the array component has the value of zero, it means this face is non-existent for this element, which is a possible situation for hybrid meshes.

`data`: This binary file contains flow solutions at nodes, a real (double) array `snout(ndf,nn)` where `ndf` is the number of degrees of freedom. For compressible flow, `ndf` is equal to 8, being density, *x*-, *y*-, and *z*-velocity components, temperature, turbulence, pressure, and Mach number while, for incompressible flow, `ndf` is equal to 7, being *x*-, *y*-, and *z*-velocity components, pressure, turbulence, free surface, and temperature.

3. Mesh subdivision algorithm

The goal of the mesh multiplication program is to subdivide each element into several child elements. As a result, finer mesh can be obtained. There are numerous ways to subdivide an element. However, the subdividing scheme shown in Fig. 1, which is referred to as the isotropic subdivision, has the advantage of preserving the mesh quality of the original coarse mesh. Table 1 summarizes subdivision outcome for each type of elements. As can be seen in Fig. 1, the majority of new nodes will appear at the centers of edges of the original coarse elements. However, new nodes may also appear at the centers of quadrilateral faces and hexahedral element centers. The new nodes at the face centers or element centers can be relatively easily handled since we already have the element and face data structure, i.e. `IEN`, `IEF` and `IFE` arrays, available. To insert new nodes on each edge, it is necessary to extract independent edge data structure first. In our approach, we use the singly linked list data structure to represent the edge list connected to each node. Further details of the mesh subdivision can be found in our past studies [7].

Along with the mesh subdivision the flow solutions are interpolated without any cost of searching as the parent–child connectivity is already known. In the current version of the mesh multiplication program, only solution values at nodes are interpolated from previous solution. The CaMEL flow solver read this restart file along with the new mesh data. The mesh multiplier is implemented in such a way that newly generated nodes are appended to the original nodal-array. Therefore, the element connectivity of the original *base mesh* and the new mesh both seamlessly point to the updated nodal-array without additional post-processing.

4. Boundary surface correction

Our strategy to map linearly-subdivided mesh nodes of solid boundaries on to the original solid geometry is based on a mesh deformation algorithm by using a dense master surface mesh nodes as the original geometry. The master surface mesh with tri- or quad-elements and associated nodes can be extracted from a given fine volume mesh in the CaMEL mesh format or can be separately generated using the same geometry file and grid generation software. This fine volume mesh has nothing to do with the fine CFD mesh. It is for geometry extraction purposes only. The master mesh does not have to be dense in the same sense as in the CFD

Please cite this article in press as: Yilmaz E, Aliabadi S. Application of surface conformed linear mesh and data subdivision technique to a spinning projectile. Comput Fluids (2013), http://dx.doi.org/10.1016/j.compfluid.2013.06.006