Parallel Mesh Generation and Adaptation using MAdLib

T. K. Sheel
MEMA, Universite Catholique de Louvain
Batiment Euler, Louvain-La-Neuve, BELGIUM
Email: *tarun.sheel@uclouvain.be*

1 Past Research and Achievements

Title: Development of a Fast Vortex Method using Special-Purpose Computers

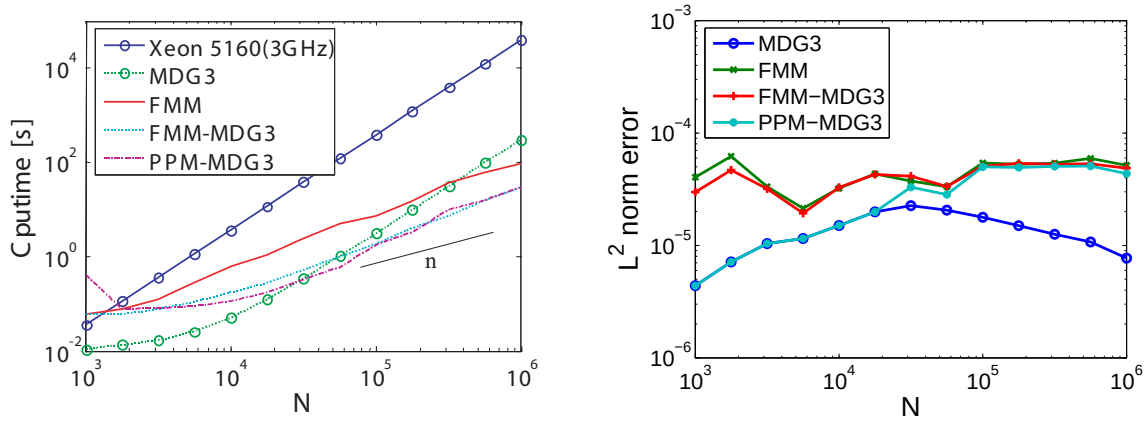
Vortex dynamics is of fundamental importance in a wide context of theory, applications and numerical simulation of fluid flows in a wide variety of contexts. Instability and interaction of vortices holds a key to an understanding of the mechanism for transferring energy from larger to smaller scales. The so-called vortex method keeps track of individual particles with vorticity frozen on them. Calculation of collisions of vortex rings, based on a vortex method, requires a million of vortex particles which consumed a very high computational cost to produce a secondary vortex rings. Since the pioneering work of Leonard in 1980s for the three-dimensional vortex method, several hybrid methods have been devised (vortex-in-cell and particle-in-cell methods by Christiansen, 1973, PPM library by Sbalzarini, 2006) to accelerate the vortex method calculation; those are semi-Lagrangian methods and mixed Eulerian methods using parallel computers. In our past research, we developed a number of novels numerical techniques for fully mesh less vortex methods that accelerate the calculation significantly, retaining accuracy at an acceptable level [1].

First, we adapted the special-purpose computers, MDGRAPEs, to develop a fast vortex method [1, 2]. The MDGRAPEs were exclusively designed for an efficient calculation of molecular dynamics simulations. The key feature of our contribution is adaptation of three-dimensional vortex method, whereby acceleration of vortex method is achieved without using the PC cluster. This is the first application of MDGRAPEs to 3D vortex motion, and serves as an interface with molecular dynamics calculations. The three main issues were addressed regarding the implementation of MDGRAPEs on vortex methods, namely the efficient calculation of the Biot-Savart law and vorticity-stretching equation, the optimization of the table domain, and the round-off error caused by the partially single precision calculation in MDGRAPEs.

Second, we moved on to MDGRAPE-2. a special-purpose computer MDGRAPE-2. A mathematical formulation for the 3D vortex method was developed for using the special-purpose computer MDGRAPE-2 for the 3D vortex method. Implementation of rigorous assessment of this hardware had been limited to a couple of problems. we made an assessment of the performance of MDGRAPE-2 by comparing the performance of calculation with and without it. It became clear that generation of appropriate function tables, which are used to call libraries, embedded in MDGRAPE-2 is crucial to gain high accuracy. The error arising from the approximation was estimated by calculating three pairs of vortex rings impinging to themselves [1, 2, 4]. Consequently, acceleration of about 100 times was achieved by MDGRAPE-2 while the error in the statistical quantities such as kinetic energy and enstrophy remained negligible.

Then we turned to MDGRAPE-3, a successor of MDGRAPE-2, and, repeating the same calculations, achieved improvement in calculation speed, for $N = 10^6$ vortex particles, by1000 times faster than using the host PC and by 25 times faster than using the MDGRAPE-2. Some issues regarding the comparative study between MDGRAPE-2 and MDGRAPE-3 were investigated carefully.

Finally, we amended the original formulation of the FMM to be compatible with these special-purpose computers for further development of the 3D vortex method. The simultaneous use of the FMM with MDGRAPE-2 and MDGRAPE-3 was devised to our previous calculations to seek the possibility of further accelerations. The various forms of the FMM and their performance on MDGRAPE-2 and MDGRAPE-3 were investigated. With the help of these acceleration techniques, the computation time for the dynamics of two colliding vortex rings was reduced by a factor of 2000 compared with a direct calculation on a standard PC. A dramatic improvement in both accuracy and speed was attained and the numerical results exhibited an excellent agreement with the previous work. The reconnection of the vortex rings was clearly observed, and the discretization error became nearly negligible for the calculation using 10^7 elements [1, 3, 4, 5].



(a) CPU Time

(b) Accuracy

Figure 1: CPU Time and Accuracy of Different Methods

The CPU-time for all the methods is plotted in the figure 1(a). Xeon 5160(3GHz), MDG3, FMM, FMM-MDG3, and PPM-MDG3 represent the calculation without either of FMM or MDGRAPE-3, with MDGRAPE-3, with FMM, with FMM and MDGRAPE-3, and with the pseudo-particle method and MDGRAPE-3. The direct summation is over 200 times faster when calculated on the MDGRAPE-3. The FMM is about 13 times faster and the PPM-MDG3 is about 48 times faster when the MDGRAPE-3 is used. There are still some rooms to improve the acceleration rates.

The L^2 norm error from the direct calculation without MDGRAPE-3 is shown for all other methods in figure 1(b). The MDGRAPE-3 contains errors of its own, which stem from the partially single precision calculation, and use of interpolation for the calculation of the cut-off function. The MDGRAPE-3 calculation exhibits better agreement compared with others similar calculation and standard for entire vortex method calculation [4, 5].

2 Ongoing Research

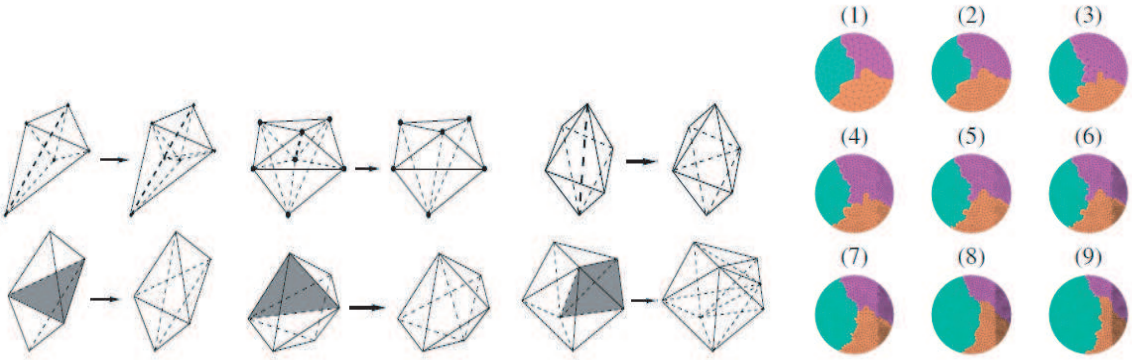
The advancements of parallel mesh adaptation tools are important in current CFD research. In fact there are lots of problems related to mesh generation and adaptivity remain open. Finite element and finite volume methods could largely benefit by proposing solutions to the remaining issues. Among the challenges that are still to be taken up, we mention the adaptation for curvilinear meshes, the anisotropic mesh adaptation with high aspect ratios, the gradation control on highly anisotropic meshes, the automatic generation and adaptation of boundary layer meshses,

the handling of complex CAD models, issues in mesh optimization like fully robust sliver elimination, and adaptivity for very large meshes. We address some of these issues, in particular those related to the mesh adaptation with large domain deformations and CPU memory issues.

Parallel Mesh Adaptation: The main difficulty of parallel mesh adaptation is the management of the mesh entities that are presents on several processors. For example, if several processors share a long edge and if we want either to split, swap or collapse this edge we have to modify the mesh consistently in several processors.

Some communications have therefore to be performed between the processors. Every basic mesh operation has to be parallelized: this includes point insertion, edge swap, edge collapse, edge split, face collapse and sliver eliminates(Fig. 2(a)). This represents a lot of effort and often leads to bad performance to the parallel mesher. The other way around is to leave the entities that are on inter-processor unchanged and to treat them later, i.e., at the next re-meshing iteration. In this case, some kind of mesh migration procedure has to be added to the algorithm. The advantage of the approach is that mesh migration is something orthogonal to mesh adaptation, i.e., those two building blocks are completely separated in their implementation.

The principle of our algorithm is based on a remeshing process and consists to not touch interface elements during the meshing process and to move them later to avoid their modifications. The processes begin with a mesh and then proceeds iteratively. The mesh is initially splitted into Np partitions with METIS. The parallel meshing algorithm is composed by three parts: (i) a call to the serial re-meshing algorithm, (ii) the migration step where inter-processor interfaces are moved, and (iii) the calculation of a criterion for stopping the procedure.



(a) Mesh Adaptation (b) Adaptation Algorithm

Figure 2: Mesh adaptation and parallel mesh adaptation algorithm

The above procedure is illustrated in figure 2(b). The initial mesh is partitioned into 3 partitions (Figure 2(b), image 1). In the first iteration, most of the edges of the mesh are split, except the ones at inter-processor boundaries (image 2). Then, interfaces are migrated (image 3). The second iteration of the algorithm is depicted the same way in Figure 2(b), images 4, 5 and 6. The algorithm usually converges in less than 10 iterations in serial. Images 7, 8, and 9 of figure 2(b) correspond to iterations 3, 5 and 7 of the algorithm. At iteration 7, the algorithm has converged. The migration of interfaces is the only additional cost of the parallel extension. For all the examples we have introduced, the serial code converge in the same number of iterations as the parallel code. However, due to the edges/faces constraints, the mesher is often a little less slow.

Mesh Adaptation Platform: The mesh adaptation field is very technical in the sense that the resulting meshes depend on the algorithms and implementations. Some algorithms are based

on heuristics, like for instance the sliver elimination procedures. Furthermore, the efficiency of an algorithm can be dramatically altered if some considerations about its implementation are not taken into account, like in the research of the optimal edge swap configuration to give an example. A contribution of the present work is the development of an open source parallel mesh adaptation platform: MAdLib (**M**esh **Ad**aptation **Lib**rary).

Memory Issue: In the present research, we described a mesh database that results from a compromise between memory requirements and time consumption for the most common operations. The memory of one million nodes is about 5 Gb (64 bit CPU) for tetrahedral meshes. In addition, 10% for parallel information, $10 - 15\%$ for adaptation procedure and virtual memory is required for parallel calculation. Compared to mesh generation software's, this amount is relatively large.

3 Proposed Research

Large, Periodic and BL Meshes with Deformation: We will do experiments about large deformations, geometry handling and BL mesh adapation for 'academic' test cases. Then we will do adaptation and generation of large meshes (>50 M nodes) for landing gear (only volume modifications). Introduction of the periodic transformations and the evaluation of the anisotropic unstructured meshes for turbulent boundary layers will be performed. Finally we will turn to do adaptation of meshes for aeroelastic computation for 3D wing with 1st bending mode along with BL mesh. In addition, the present algorithm can be applied to adaptive Boundary Layer mesh, Ocean Modelling and Fluid Structure Interaction problems. All computations are performed in parallel computers with Open MPI and parallel MAdLib(http://sites.uclouvain.be/madlib).

Memory Minimization The usual bottleneck of the mesh generation techniques for large meshes are the memory requirements in serial and the design of appropriate algorithms in parallel. Parallel mesh adaptivity for fixed domains opens a perspective for parallel mesh generation, since an initial coarse mesh can be generated in serial, partitioned by a classical partitioning procedure, and adapted by the present method. Therefore, the remaining question is about the required memory in parallel when dealing with very large meshes (>50 millions nodes). This issue is common for both mesh generation and adaptation.

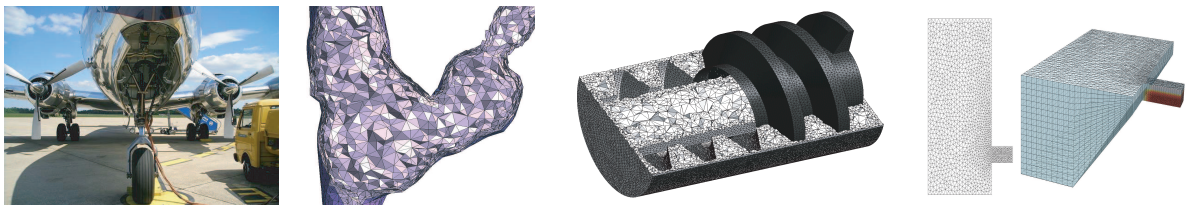
MAdLib with clusters GPUs: The advent of multicore CPUs and manycore GPUs means that mainstream processor chips are now parallel systems. The challenge is to develop application software that transparently scales its parallelism to leverage the increasing number of processor cores, such as 3D graphics applications transparently scale their parallelism to manycore GPUs with widely varying numbers of cores. We will do large scale calculation using this new technique combined with our parallel MAdLib. The simultaneouse technique will be a unique and one step algorithm for parallel mesh generation and adaptation for huge meshes ($>$ 50 millions nodes) in near future.

3.1 Benchmarks

We will apply our proposed algorithm to solve for different geometrical model as the following benchmarks problems (Figure 3). Our target is to get high efficiency with minimum CPU time and memory.

3.2 Objectives

Our main goal is to remesh highly complex geometry to produce large meshes ever been made using clusters PC. We will make subclassifications of our objectives as follows.



(a) Landing Gear (b) BL Mesh (c) Domain Deformation (d) Dome Mesh

Figure 3: Benchmarks of the proposed algorithm

- Analysis of high-order, unstructured, finite-element methods
- Analysis of high-order nodal Discontinuous Galerkin method using clusters GPUs
- Parallel Mesh Generation for Complex Geometries (e.g. Landing gear of aircraft)
- Calculate for several millions of nodes ($\sim$ 50 M)
- Parallel mesh adaptation with preserved BL mesh
- Parallel mesh adaptation for periodic boundary with transformation
- Domain of ocean modeling and FSI
- Bio-mechanical problems with BL Mesh
- Parallel computing using Open MPI and Open CL with CUDA, PyCUDA, Parallel MAdLib with GPUs
- Simplification and introduction of these techniques to academic and industrial environments
- Collaboration research with different universities and industries to share our new method

References

[1] **T. K. Sheel**, Development of a Fast Vortex Method for Fluid Flow Simulation using Special-Purpose Computers, PhD Thesis, Keio University, **March 2008.**

[2] **T. K. Sheel**, K. Yasuoka, S. Obi, Fast vortex method calculation using a special-purpose computer, Computers and Fluids, **2007**;36: 1319-26.

[3] R. Yokota, **T. K. Sheel**, S. Obi, Calculation of Isotropic Turbulence Using Pure Lagrangian Vortex Method, Journal of Computational Physics, **2007**;226:1589-1606.

[4] **T. K. Sheel**, R. Yokota, K. Yasuoka, S. Obi, The study of colliding vortex rings using a special-purpose computer and FMM, Transactions of the Japan Society for Computational Engineering and Science, **2008**: 20080003.

[5] **T. K. Sheel**, S. Obi, High Performance Computing Techniques for vortex method calculations, International Journal of Theoretical and Computational Fluid Dynamics, DOI 10.1007/s00162-009-0149-y, **2009**.