Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 00 (2015) 000–000

Procedia Engineering

www.elsevier.com/locate/procedia

24th International Meshing Roundtable (IMR24)

Parallel local remeshing for moving body applications

Jianjun Chen*, Shaolei Li, Jianjing Zheng, Yao Zheng

Center for Engineering and Scientific Computation, Zhejiang University, Hangzhou 310027, China

Abstract

This paper presents a parallel local remeshing approach on distributed parallel computers based on the message passing interfaces (MPI). In comparison to those existing approaches that require combining a single hole initially spanning multiple processors into one processor so that the sequential results can be reproduced, the proposed approach avoids this limit by decomposing the initially distributed holes into many sub-holes and distributing them evenly on the involved computer cores. The employed domain decomposition approach can ensure the generation of a well-shaped inter-hole boundary always. Therefore, the subsequent remeshing step can fix the inter-hole boundary without compromising the quality of elements around this boundary. In the mean time, because the remeshing procedure involves no communications, its parallel efficiency is very high. An application of the remeshing algorithms in a benchmark store separation simulation demonstrates the capabilities of the approach.

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

Keywords: mesh generation; local remeshing; parallel algorithm; unsteady flow; moving boundary

1. Introduction

In the community of computational aerodynamics, simulating flows around geometries that change their shape and/or position with time is commonly occurred in many important applications. Among abundant literatures reported on this issue, the local remeshing based approach prevails since it represents a good compromise between general applicability, efficiency and mesh quality **[1]**. This approach adopts a single consistent unstructured grid in every time step. To accommodate the geometry change between different time steps, the grid nodes are moved while

* Corresponding author. Tel.: +0-086-571-87951883; +0-086-571-87953167.
E-mail address: chenjj@zju.edu.cn

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

the node connectivity can be kept unchanged initially. However, this grid deformation method may sometimes result in badly shaped or even inverted elements when large movements are involved during the simulations. To continue the solution process, a suitable strategy is to adjust only a small subset of an existent grid of poor quality by cutting some holes in regions where the local element quality is deteriorated, and then remeshing the holes. After that, the solution on the new grid that fills in the holes is interpolated from the solution on the old grid and the solution process is advanced on the new grid.

Presently, the flow solution is usually computed in parallel in a moving body simulation, whereas the remeshing step runs in sequential. If the mesh magnitude is not too large and only a few number of computer cores is involved in the parallel flow solution, the performance bottleneck induced by the sequential remeshing step is negligible because the CPU requirements of local remeshing are orders of magnitude less than those of flow solutions. However, the moving body simulations of engineering interest involve a mesh consisting of millions of elements or more. To get the result in a reasonable time, more than hundreds of computer cores are employed. In this type of simulation, a sequential remeshing step becomes unfeasible due to the following facts. Firstly, the machine on which the sequential remeshing step runs must contain enough memory to fill in the entire mesh. Secondly, the CPU requirements of local remeshing become prominent when the flow solutions have been speeded up for hundreds of times. Thirdly, this approach need combine distributed submeshes into one single machine before each remeshing step, and distribute the modified mesh from the single machine to available computing nodes after each remeshing step. Not surprisingly, the mesh combination and distribution procedures need transfer data between different computing nodes intensively and can become a performance bottleneck of the entire simulation process, depending on the frequency of the remeshing callings and the data transfer efficiency between different computing nodes.

Parallelisation is a feasible way to overcome the performance bottleneck induced by a sequential local remeshing step. In **[1]**, Tremel *et al.* proposed an approach towards the parallel local remeshing of unstructured tetrahedral grids on distributed memory parallel computers based on the message passing paradigm (MPI). This approach requires combining a single hole initially spanning multiple processors into one processor so that the sequential results can be reproduced. Therefore, if the remeshed hole is too large, this approach may encounter performance bottlenecks in terms of both memory usage and computing time. By contrast, the new parallel remeshing approach proposed in this study can avoid this limit by decomposing the initially distributed holes into many *sub-holes* and distributing them evenly on the involved computer cores for loading balance. The domain decomposition approach employed in this study is an extension of that suggested in **[2]**, which ensures the generation of a well-shaped inter-hole boundary always. Therefore, the subsequent remeshing step can fix the inter-hole boundary without compromising the quality of elements around this boundary. In the mean time, because the remeshing procedure involves no communications, its parallel efficiency is very high.

The parallel local remeshing approach described above has been composed as an integral part of an in-house system for simulating unsteady complex flow problems with moving boundary components. The other parts of this system include a parallel spring-analogy grid deformation module, a solver for solving the six degrees of freedom (6-DOF) equations of body motions and a parallel finite volume solver. Due to the limit of the length upon this research notes, we will first describe the adopted domain decomposition approach in brief, and then detail the main steps involved in the parallel local remeshing approach. Finally, the computational accuracy and efficiency of the developed system will be analysed by considering a benchmark store separation simulation.

2. The domain decomposition approach

2.1. Review on the sequential domain decomposition approach **[2]**

Domain decomposition methods have been vastly demonstrated for solving numerically partial differential equations (PDEs) on parallel computers. In this context, the dual goals of load balancing and minimisation of communications are imposed on the partitioned meshes. In this study, the focus is on the domain decomposition approach that can be applied for parallel generation and improvement of unstructured meshes; thus, it will be beneficial if no artificial geometrical features are created in the inter-domain boundary. To accomplish this additional goal, a novel domain decomposition approach is proposed in **[2]**, which analyses the dual graphs of a mesh to identify and remove the undesirable mesh sides (edges in 2D or faces in 3D). The simplified mesh instead of the original mesh is partitioned

using a generic graph partitioner, which yields many subdomains without small dihedral angles and/or poorly shaped sides attached on the inter-domain boundary.

Two dual graphs of a mesh need be analysed in the domain decomposition approach, namely *the side dual graph* (SDG) and *the element dual graph* (EDG). In the SDG of a tetrahedral mesh, a graph node refers to a mesh face, and a graph edge exists between two mesh faces meeting at one mesh edge. A node of a SDG is weighted by the shape quality of the mesh face, e.g., the minimal interior angle of the face, and the edge of a SDG is weighted by the dihedral angle formed by two neighbouring faces. In the EDG of a tetrahedral mesh, a graph node refers to a tetrahedral element, and a graph edge exists between two elements meeting at one face. The weighting strategy of an EDG depends on the application purpose of the domain decomposition approach.

Figure 1 presents the basic flowchart of the proposed domain decomposition approach, where the frames, solid arrows and dotted arrows represent the data, algorithms and inputs of the algorithms, respectively. The input is a mesh composed of any type of polyhedral or polygonal elements. The output is the partitioned result of the input mesh, where no subdomains (submeshes) contain artificial small angles, and optionally, poorly shaped faces in their boundaries for 3D problems. If the input mesh is appropriate, the dual goals of load balancing and minimisation of communications are obtained as well. The intermediate steps that connect the input and output are as follows.

(1) Build the (SDG) and (EDG) of the input mesh.
(2) Identify some mesh sides with undesirable shapes as *removable*, and simplify the SDG by deleting the graph nodes dual to the removable sides.
(3) Simplify the SDG by deleting some graph nodes, and identify the mesh sides they correspond to as *removable*, to ensure that the dihedral angles contained in the mesh (apart from those formed by two initial boundary faces) are all larger than a predefined threshold.
(4) Simplify the EDG by contracting all of the edges that are identified as *removable* in Steps 2 and 3.
(5) Decompose the simplified EDG using a graph-partitioning tool, aimed at the dual goals of load balancing and minimisation of communications.
(6) Decompose the input mesh into subdomains according to the partitioning result of the simplified EDG.

Step 2 deserves more explanations because its implementation depends on the application purpose of the proposed domain decomposition approach. In this study, mesh faces that contain small interior angles or disrespect a prescribed size map are not favoured by the inter-domain boundary because badly shaped volume elements are usually incident to these faces. Therefore, such faces will be identified as *removable* in Step 2.

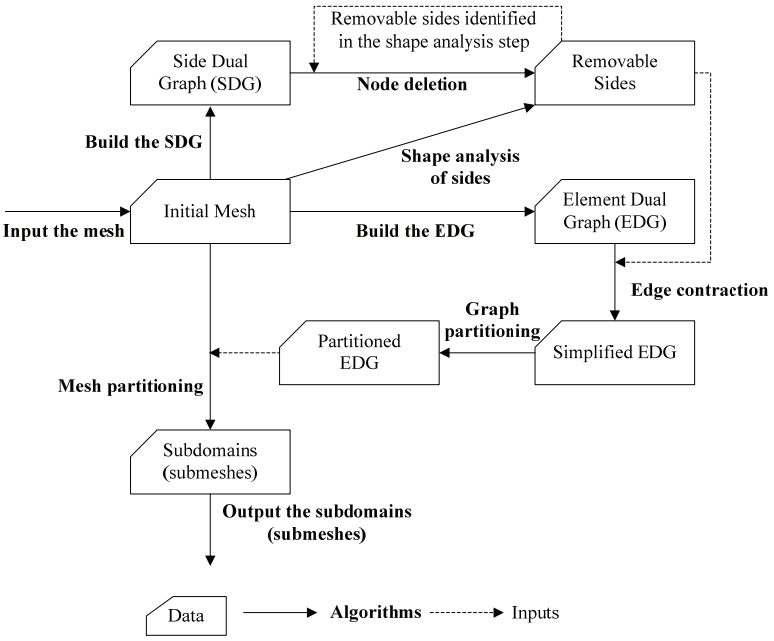


Figure 1. The flowchart of the proposed domain decomposition approach.

2.2. The parallel implementation of the domain decomposition approach

In this study, the input of the domain decomposition step is a mesh filling in the hole region. Since the flow solution is parallelised, this mesh may be distributed on multiple computing nodes. An option is to first assimilate the submeshes and then partition the assimilated mesh by using the sequential domain decomposer. However, this approach is unsuitable, if storing the entire mesh is beyond the accessible memory size of a single computer node or the computational scalability is a primary focus. Instead, a parallel domain decomposer is necessary in this circumstance.

Step 3 (the node-deletion step) and Step 4 (the edge-contraction step) discussed in Sections 2.1 need to access mesh data adjacent to one mesh side. Therefore, if the sides they access are inter-domain sides, inter-processor operations are necessary to ensure a qualified output. To avoid implementing these time-consuming inter-processor operations, an alternative scheme is proposed below:

(1) Simplify the EDG of each submesh concurrently by labelling inter-domain sides as boundary sides.
(2) For those inter-domain sides with undesirable shapes or those bounding small dihedral angles, penalise their dual EDG edges with large weights.
(3) Repartition the EDG by ParMetis.
(4) Redistribute the submeshes to comply with the graph partitioning result.

3. The parallel local remeshing approach

3.1. Overview of the sequential remeshing approach

To accommodate the geometry change between different time steps, the grid nodes are moved while the node connectivity can be kept unchanged initially. In the case of large boundary movements, the grid deformation algorithm may fail in producing a quality grid for the solver. Following each failure of the grid deformation step, a local remeshing algorithm is employed to improve the robustness and the quality of grid generation, so that the solution can be advanced further to the next steps. In general, the remeshing algorithm takes the following steps:

(1) *Determination of the region to be remeshed*. Analyse the quality of elements after grid deformation. If the grid has satisfactory quality for flow computation, then advance the solution to the next time step; otherwise, determine the regions to be remeshed and extract their surface boundaries.
(2) *Volume remeshing*. Input the surface boundaries to a constrained Delaunay mesher **[3]** to generate a *new* quality grid that fills in the holes, and then replace the old grid of the holes by this new grid.
(3) *Solution reconstruction*. Reconstruct the solution by interpolation most recent results and continue with the flow computation.

3.2. The parallel implementation of the remeshing approach

Given a distributed mesh, the parallel local remeshing approach inserts two additional domain decomposition steps in the sequential flowchart mentioned in Section 3.1 and then parallelises all of these steps as follow:

(1) Run the step of the determination of the region to be remeshed in parallel.
(2) *Domain decomposition I*. Repartition the initial mesh filling the hole region by the proposed parallel domain decomposition approach mentioned in Section 2.2. As a result, the holes with a high-quality inter-domain boundary in terms of geometric shape are distributed on involved processors.
(3) Remeshing the distributed holes in parallel with the inter-domain boundary preserved.
(4) Reconstruct the solution in parallel.
(5) *Domain decomposition II*. Combine the old and new mesh as a whole, and repartition this mesh by ParMetis to ensure the dual goals of load balancing and minimisation of communications.

4. Numerical results

The three store ripple release case is presented here to demonstrate the capabilities of the proposed parallel remeshing approach. This case was first presented in **[4]** and later revisited in **[5, 6]**. The geometric configuration is

composed of three generic stores in a triple ejector rack (TER) configuration under a generic pylon attached to a clipped delta wing. To ensure the separation trajectory safe, ejectors are acting on each store in the initial stage. The stores are released in a bottom-outboard-inboard order with a 0.04s delay between each release. The ejectors are applied after release for duration of 0.045s. In the simulation, ejectors are fixed on the stores so that the ejectors will not create rolling moments. See **[4-6]** for more details about the solution parameters of this simulation.

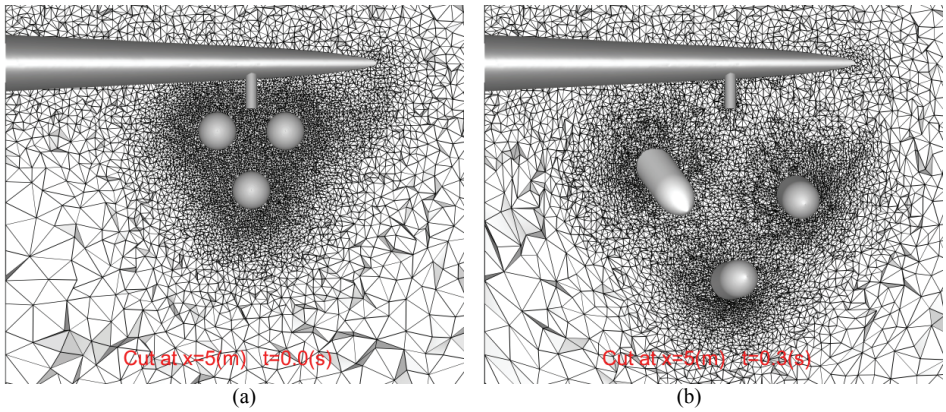
The inviscid flow is simulated in this study by using a tetrahedral mesh initially composed of 4,233,472 elements. The flow solution adopts the dual time stepping method, and the physical time interval is set to be 0.0005s; thus totally 800 steps are computed in the simulation since the overall simulated physical time is 0.4s. Figures 2a and 2b present the cut views of the initial volume mesh and the volume mesh at t_s=0.3s (t_s refers to the separation time), respectively. Figure 3 compares the simulated results with those published in **[6]**. The comparison reveals that the simulated trajectories of the bottom and outboard stores agree well with the referred data, while those of the inboard store diverge after 0.2s in x direction and 0.3s in y direction, respectively. The possible reason is that the geometric positions of the three stores may be inaccurate because these data are not provided in the referred literature **[6]**.

To demonstrate the benefit of the parallelised local remeshing step, Table 1 presents the timing performance of the simulations based on the sequential and parallel local remeshing approaches, respectively. Both simulations started from the same initial configuration and are executed on the same computing nodes (totally 256 computer cores are employed in both simulations). The simulation employing the parallel remeshing algorithm consumes about 3hr53min totally, 10.6min of which are consumed by the 18 callings of the remeshing algorithm. As a comparison, if the remeshing algorithm runs in sequential, the overall simulation time increases about 29%, up to about 5hr. The increased time consumption mainly sources from the remeshing step. It is now called for 16 times and takes about 1h15min. Compared to the sequential approach, the parallel approach reduces the simulation time by more than one hour (1h7min) at a cost of two additional remeshing callings.

Nominally, the efficiency of the parallelised remeshing step seems very low since it only achieves a 7.1 times speedup on 256 computer cores. Two facts mainly account for this result. Firstly, only those computer cores owning the mesh elements filling in the hole region take part in the parallel run in each remeshing step. Secondly, although the steps of parallel local volume remeshing and solution interpolation are well balanced, the loading balance performance of the steps of parallel determination of the regions to be remeshed and parallel domain decomposition highly depends on the initial distribution of the elements filling in the hole region. It is possible to achieve a higher parallel efficiency by improving the implementation of the algorithm. Nevertheless, this effort is not meaningful for this simulation since the remeshing time has been reduced to less than 5% of the total simulation time.

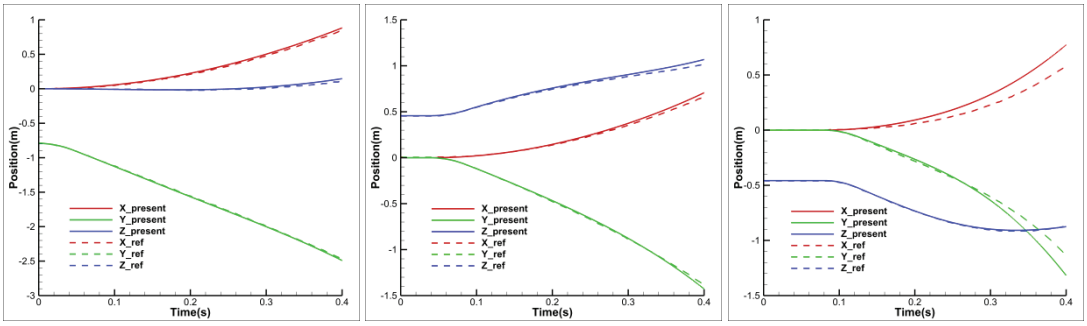
Table 1 Breakdown of the total time spent in the simulations (Unit: second).

Approaches	#Cores.	Total time	Mesh deformation	Flow solution	Remeshing	Other
Parallel	256	13997.97	488.94	12227.9	637.46	643.67
Sequential	256	17997.71	472.05	12333.96	4499.53	692.17



(a) (b)

Figure 2 the cut views of the initial volume mesh and the volume meshes at t_s=0.3s.



(a) (b) (c)

Figure 3 Comparison of the simulated trajectories of three stores with those published in [6]. (a)~(c) render the data for the bottom, outboard and inboard stores, respectively.

Acknowledgements

The authors appreciate the joint support for this project by the National Natural Science Foundation of China (Grant No. 11172267, 10872182 and 11432013) and Zhejiang Provincial Natural Science Foundation (Grant No. Y1110038). The first author acknowledges the joint support from Zhejiang University and China Scholarship Council for his visiting research at Swansea University, UK.

References

[1] U. Tremel, K.A. Sørensen, S. Hitzel, H. Rieger, O. Hassan, N.P. Weatherill, Parallel remeshing of unstructured volume grids for CFD applications, Int J Numer Meth Fluids. 53 (2007) 1361-1379.
[2] D. Zhao, J. Chen, Y. Zheng, Z. Huang, J. Zheng, Fine-grained parallel algorithm for unstructured surface mesh generation. Comput Struct, 154 (2015):177-191.
[3] J. Chen, D. Zhao, Z. Huang, Y. Zheng, S. Gao, Three-dimensional constrained boundary recovery with an enhanced Steiner point suppression procedure, Comput Struct, 89 (2011): 455-466.
[4] R.D. Thomas, J.K. Jordan, Investigation of multiple body trajectory prediction using time accurate computational fluid dynamics. AIAA paper 95-1970, 1995.
[5] N.C. Prewitt, D.M. Belk, R.C. Maple, Multiple-body trajectory calculations using the beggar code. J Aircraft. 36 (1999) 802-808.
[6] N.C. Prewitt, D.M. Belk, W. Shyy, Parallel computing of overset grids for aerodynamic problems with moving objects. Prog Aerosp Sci. 36 (2000) 117-172.