ParCFD'2020
32^{nd} International Conference on Parallel Computational Fluid Dynamics
May-11-13 2020, Nice FRANCE

AN EFFICIENT SLIDING-MESH METHOD FOR PARALLEL CFD SIMULATIONS IN 3D UNSTRUCTURED MESHES

J. MUELA[*,†], J. VERA[†] AND A. OLIVA[†]

[*]Termo Fluids S.L., Av. Jacquard 97-E, 08222, Terrassa (Barcelona), Spain.
Web page: http://www.termofluids.com/

[†]Heat and Mass Transfer Technological Center (CTTC), Universitat Politècnica de Catalunya-BarcelonaTech(UPC), *ESEIAAT, Colom 11, 08222, Terrassa (Barcelona), Spain.*
Web page: http://www.cttc.upc.edu/

Key words: Sliding-Mesh, ALE, Dynamic Mesh

Abstract. An efficient and accurate new methodology for parallel CFD simulations using sliding-meshes technique is presented. This methodology allows to simulate domains composed by different meshes non-overlapped but with shared boundaries and relative velocity between them. The developed algorithm is able to reconstruct the topology of the whole computational domain at each iteration connecting dynamically all the independent meshes. This reconstruction is done via geometrical intersection of the shared boundaries.

1 INTRODUCTION

Usually, in Computational Fluid Dynamics (CFD) the governing equations are discretized and solved in a static mesh representing the studied domain. Nonetheless, when aiming to study moving bodies this approach is not valid anymore. In some cases, when all the flow domain undergoes the same motion, these problems can be studied using non-inertial reference frameworks or moving the entire domain and solving the Navier-Stokes equations using the Arbitrary Lagrangian-Eulerian (ALE) technique [1]. Nonetheless, these approaches cannot be employed when the case of study involves various solids moving at different speeds. This occurs in multiple applications like wind turbines, fans, propellers, turbo-machinery, etc. For these cases it can be of interest to employ different meshes with relative movement between them. For example, in the case of wind turbines the domain would be composed by two meshes: one rotating mesh containing the rotors and another static one including tower, nacelle and ground. Then, some advanced technique must be employed to dynamically connect both meshes during simulation time. The two most common techniques are the chimera (overset) method [2] and the sliding-mesh technique [3]. Both methods were studied and compared by Francois et al. [4], showing that both methods are able to provide reliable results with a similar level of accuracy. However, the sliding-mesh technique was more efficient in terms of computational cost and much less memory consuming.

The main idea of the sliding-mesh technique is to *stitch* two or more mesh domains Ω_i ($\sum \Omega_i = \Omega$). These domain are not overlapped ($\Omega_i \cap \Omega_j = 0$) but they are adjacent

and share some or all boundaries $\partial\Omega_{i_b} = \partial\Omega_{j_b}$. These boundaries can be non-conformal. Within the sliding-mesh technique different approaches have been proposed. There are the *flux-based* methods like the *General Grid Interface* (GGI) method [5], techniques based on Topological changes [6, 7] or the *Shear-slip mesh update* method [8].All these methods require the projection and intersection of the sliding boundaries at each iteration. Another possibility are the methods based on *halo* nodes and/or boundary nodes interpolation, which allow to avoid the projection-intersection step (e.g.,[9, 10]). In the work of Ramirez et al. [11] is presented a comparison of both possibilities in high-order finite volume schemes.

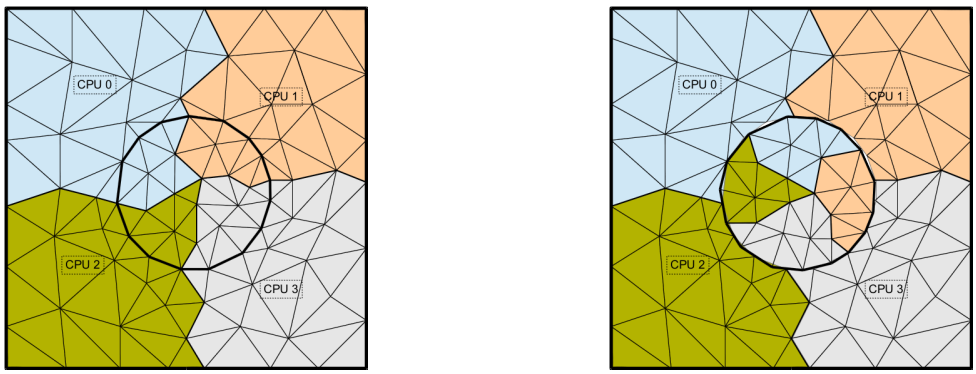
This work presents an improved version of the methodology previously developed in Muela et al. [12]. In the latter, the presented sliding-mesh technique was based on *halo* nodes (named *mirror* nodes in the work), while the sliding-mesh algorithm of this work is *flux-based*. This new methodology aims to enhance the conservation properties of the previous method although trying to keep the *essence* of the algorithm, allowing to have both good parallel performance and computing efficiency.

2 SLIDING-MESH ALGORITHM

There are three main aspects that will influence the performance of a sliding-mesh method in parallel computing: i) the computational cost associated to the displacement of the moving parts; ii) the cost of boundaries projection-intersection step or boundary nodes interpolation; iii) the parallelization strategy.

In the presented method, which is *flux-based*, it is important to accelerate as much as possible the projection-intersection step of the boundaries. Intersecting the edges and faces has a cost of $O(n^2)$ if no optimizing strategy is applied. In the present work the intersection is done using the Sutherland-Hodgman algorithm [13]. Another complexity is the parallelization strategy. As shown in Fig. 1, the initial partitioning of a mesh will change during simulation time. Hence, the dynamic evolution of the topological connections will involve new neighbourhoods of cells of different processors during simulation time. In order to overcome these issues and implement a highly-efficient sliding-mesh algorithm in parallel computing the strategy summarized below has been adopted. The algorithm is detailed for a domain composed by two meshes: one static, named the *master*, and a second rotary domain named the *slave*. This is because the algorithm is based on a *master-slave* strategy.

The strategy is as follows: initially, each face in the sliding boundary of the *slave* domain injects an auxiliary particle on the *master* domain. These auxiliary particles will move jointly with the rotary domain. Besides, each face in the sliding boundary of the *master* domain will pre-compute and store a *mini-mesh*. This *mini-mesh* is a face-mesh containing the geometrical info of the neighbouring faces, including faces belonging to other processors. This allows to avoid a large amount of communications during simulation time at a very low memory-cost. The size of these *mini-meshes* stored at each *master-face* is determined by the minimum length which assures that any possible face received from the *slave* domain will be fully intersected (i.e, the larger face).



(a) Initial mesh partitioning. (b) Mesh partitioning after N iterations.

Figure 1: Example of a rotating mesh partitioned in 4 CPUs.

Then, at each iteration the following steps are performed: i) first, all the auxiliary particles are projected towards their parent faces. This trajectory will intersect a face on the *master* domain; ii) following, each particle will communicate to his parent face which face in the *master* domain has been intersected. From this info, the *slave* face will send its geometrical data to the *master* face that has been intersected by the auxiliary particle; iii) then, the *master* face will perform the projection-intersection step between the geometrical info received by the *slave* face and its *mini-mesh*. This guarantees that the whole surface of the *slave* face is intersected. iv) once this projection-intersection step is done, the new surfaces are obtained. These new surfaces and the associated data is sent to the corresponding processors; v) finally, these new geometrical info is employed to reconstruct the new topology and the operators.

The algorithm summarized above is robust, fast and highly-scable. It optimizes the projection-intersection step and reduce the overall number of communications. In the present work are studied the conservation and accuracy properties of the method, as well as its parallel speed-up. Moreover, it is compared with the previous implementation presented in Muela et al. [12].

ACKNOWLEDGMENTS

This work has been financially supported by the Ministerio de Educación y Ciencia (MEC), Spain, within the project *"Algorimtos numéricos avanzados para la mejora de la eficiencia energética en los sectores eólico y solar-termico: Desarrollo/adaptación a nuevas arquitecturas computacionales"* (ref. ENE2017-88697-R).

REFERENCES

[1] Y. Bazilevs and T.J.R. Hughes. Nurbs-based isogeometric analysis for the computation of flows about rotating components. *Computational Mechanics*, 43(1):143–150, 2008.

[2] J.L. Steger, F.C. Dougherty, and J.A. Benek. A chimera grid scheme. In *Advances*

in grid generation; Proceedings of the Applied Mechanics, Bioengineering, and Fluids Engineering Conference, pages 59–69, Houston, TX, June 1983. American Society of Mechanical Engineers.

[3] M.M. Rai. A conservative treatment of zonal boundaries for euler equation calculations. *Journal of Computational Physics*, 62(2):472–503, 1986.

[4] B. Francois, M. Costes, and G. Dufour. Comparison of chimera and sliding mesh techniques for unsteady simulations of counter rotating open-rotors. In *20th ISABE Conference*. CERFACS, Toulouse.

[5] M. Beaudoin and H. Jasak. Development of a generalized grid interface for turbomachinery simulations with openfoam. In *Open source CFD International conference*, volume 2, 2008.

[6] T. Lucchini, G. D'Errico, H. Jasak, and Z. Tukovic. Automatic mesh motion with topological changes for engine simulation. Technical report, SAE Technical Paper, 2007.

[7] O. Petit, M. Page, M. Beaudoin, and H. Nilsson. The ercoftac centrifugal pump openfoam case-study. In *Proceedings of the 3rd IAHR International Meeting of the Workgroup on Cavitation and Dynamic Problem in Hydraulic Machinery and Systems, Brno, Czech Republic*, pages 14–16, 2009.

[8] M. Behr and T. Tezduyar. Shear-slip mesh update in 3d computation of complex flow problems with rotating mechanical components. *Computer Methods in Applied Mechanics and Engineering*, 190(24-25):3189–3200, 2001.

[9] E.L. Blades and D.L. Marcum. A sliding interface method for unsteady unstructured flow simulations. *International Journal for Numerical Methods in Fluids*, 53(3):507–529, 2007.

[10] J. McNaughton, I. Afgan, D.D. Apsley, S. Rolfo, T. Stallard, and P.K. Stansby. A simple sliding-mesh interface procedure and its application to the cfd simulation of a tidal-stream turbine. *International journal for numerical methods in fluids*, 74(4):250–269, 2014.

[11] L. Ramírez, C. Foulquié, X. Nogueira, S. Khelladi, J.C. Chassaing, and I. Colominas. New high-resolution-preserving sliding mesh techniques for higher-order finite volume schemes. *Computers & Fluids*, 118:114–130, 2015.

[12] J. Muela, D. Martínez, O. Lehmkuhl, C.D. Pérez-Segarra, and A. Oliva. New parallel method for adjacent disconnected unstructured 3d meshes. *International Journal of Computational Fluid Dynamics*, 30(6):388–394, 2016.

[13] I.E. Sutherland and G.W. Hodgman. Reentrant polygon clipping. *Communications of the ACM*, 17(1):32–42, 1974.