ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep. 02-04, 2024, Bonn, Germany

Parallel Mesh Developments to Prepare for Biosimulations

Phoebe Cullen[a,b], Charles Moulinec[a,*], James Gebbie-Rayet[a], Jérôme Bonelle[c] and Yvan Fournier[c]

[a] *UKRI STFC Daresbury Laboratory, Scientific Computing, Sci-Tech Daresbury, WA4 4AD, UK*
[b] *University of Leeds, Woodhouse, Leeds, LS2 9JT, UK*
[c] *EDF R&D, MFEE, 6 quai Watier, 78400, FR*

ARTICLE INFO[†]

Keywords:
Mesh Optimization;
Compatible Discrete Operator

ABSTRACT

A parallel mesh procedure is developed to create a single mesh from 2 or more meshes, in case proteins would merge together. The various steps of the algorithm are presented in details, before some preliminary results are shown using the Compatible Discrete Operator method.

1. Introduction

Biomolecular simulations are critical to understanding the highly dynamic environments within biological cells. Biosimulations work in conjunction with imaging techniques to relate static experimental structures to functionality. For large time and length scales, continuum mechanics has been proposed for use in a coarse-grained approach. Fluctuating Finite Element Analysis (FFEA) [1] is such a method. It uses a mesh-based approach in place of a particle-based one, i.e., computing bulk quantities as opposed to atomic positions. Protein interactions are fundamental to biological mechanisms that underpin life. They can interact in a number of different ways, for instance by sticking to each other, where mesh 'subsurfaces' temporarily glue together, or by merging together. The latter is a permanent interaction, where 2 meshes join together by a 'bridge'. Sticking is the more frequent phenomenon in biology, but merging is more challenging in terms of algorithm, and as such is considered here.

This work presents the parallel algorithm used to merge 2 or more meshes (see Section 2). Some preliminary results are shown for a steady 3-D vector-valued diffusion equation computed using the Compatible Discrete Operator (CDO) method [2] implemented in code_saturne[1] [3] (see Section 3), before some conclusions are drawn (see Section 4).

2. Principle of the Method

The merging algorithm is split into several operations. Figure 1 shows the steps to build a 'bridge' between 2 neighboring meshes, before collapsing into a single body. The principle of this method is to use a plane (in red in Fig. 1), equidistant from 2 meshes' centers of gravity, as an interface for the projection, extrusion and finally merging of each pair of meshes. All the steps following the computation of the plane orientation are local to each mesh, until the 2 projected 'subsurface' footprints are glued together at the plane. The footprint boundary faces are then changed into inner faces, in the final step of the method. The method is

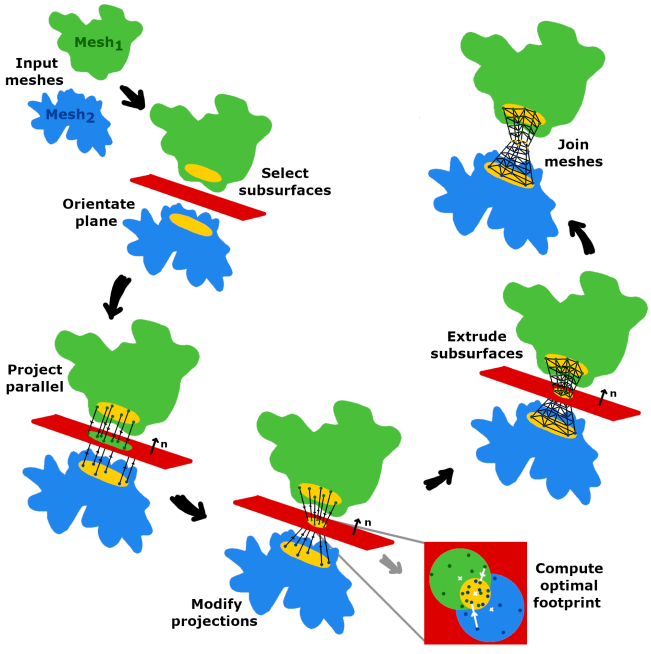

Figure 1: Merging procedure.

[†] This paper is part of the ParCFD 2024 Proceedings. A recording of the presentation is available on YouTube. The DOI of this document is 10.34734/FZJ-2025-02475 and of the Proceedings 10.34734/FZJ-2025-02175.

*Corresponding author
cm20pic@leeds.ac.uk (P. Cullen); charles.moulinec@stfc.ac.uk (C. Moulinec); james.gebbie@stfc.ac.uk (J. Gebbie-Rayet); jerome.bonelle@edf.fr (J. Bonelle); yvan.fournier@edf.fr (Y. Fournier)
ORCID(s): 0009-0005-1781-5323 (P. Cullen); 0009-0003-7011-7327 (C. Moulinec); 0000-0001-8271-3431 (J. Gebbie-Rayet); 0000-0002-8164-4646 (J. Bonelle); 0000-0001-5271-007X (Y. Fournier)

[1] https://www.code-saturne.org/cms/web

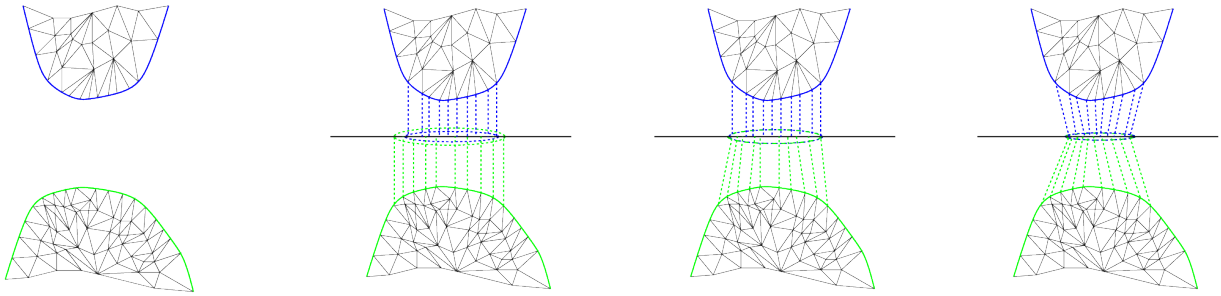

Figure 2: Sketch of the combined various steps of the projection procedure.

easily generalized to many concurrent merging occurrences between 2 individual meshes, and fully parallel.

2.1. Projection procedure

The projection procedure is the pivotal step of this algorithm (see Fig. 2). Its steps are enumerated here:

1. Construct/import the meshes to be merged
2. Select the local boundary 'subsurfaces' to be projected
3. Build the plane from the distance between the meshes
4. Perform initial projections onto the plane, parallel to the plane normal
5. Find the optimal projection footprints, as the 2 footprints might not be conformal
6. Define vectors between 'subsurfaces' and average footprint vertices
7. Re-project the 'subsurface' vertices, along these new vectors

2.2. Extrusion and interface gluing

For each mesh, an extrusion is computed between the 'subsurface' and the average footprint, creating layers of prisms. The average footprint is generally non-conformal, and the code_saturne embedded gluing algorithm is activated to get it conformal, creating new faces on both sides of the plane. This results in a layer of polyhedral cells on both sides of the plane, the average footprint faces being made inner faces.

2.3. Comments on the current implementation and potential improvements

- In this work, the location of 2 or more meshes is set a priori, and the distance between them computed from the gravity centers of the interacting objects. This is not optimal, in general, but an iterative process could be implemented to compute the minimum 'physical' distance, starting from the existing technique, based on the centers of gravity of the individual meshes.
  In the future, information could be obtained from Molecular Dynamics (MD), when the distance between 2 proteins is below a given threshold, depending on the studied case. MD could then inform whether the proteins are attracting or repulsing each other. In the former case, the merging algorithm would be activated, otherwise, the whole simulation would continue to compute the protein evolution.
- The position of the subsurface center depends on the 2 end points of the segment linked to the distance. A cylinder, which axis' center is one of these ends and direction's is based on the segment's, is used to select the subsurface elements, its radius being approximated from the shape of the physical boundaries of the protein.
- In order to get a workable number of layers to be used by each extrusion, information is gathered from the volumic elements the subsurface belongs to, focusing on their edge size.

3. Preliminary Results

The projection procedure has successfully been implemented into code_saturne, including the 'subsurface' extrusions, though some final developments are required to get the full algorithm in place. Figure 3 (left) shows the example of a 'bridge' merging between 2 cubes, and Fig. 3 (right) of 2 'bridge' merging between 3 ellipsoids. The newly created

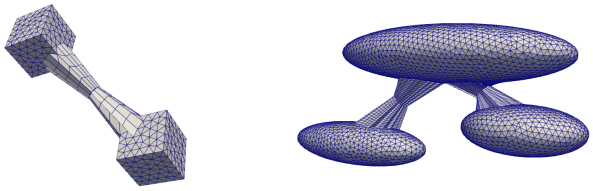

Figure 3: Examples of merging for 2 cubes (right) and 3 ellipsoids (left).

meshes will be used as supports to simulate 3-D vector-valued diffusion equations using the Compatible Discrete Operator approach. To demonstrate the performance of the

method, a simulation is carried out in a cubic box, with zero-Dirichlet boundary conditions in all 3 directions. The mesh is made of a mix of tetrahedral (inner mesh) and prismatic (at the wall) elements to assess the ability of the CDO approach for mixed element meshes. Figure 4 (top)

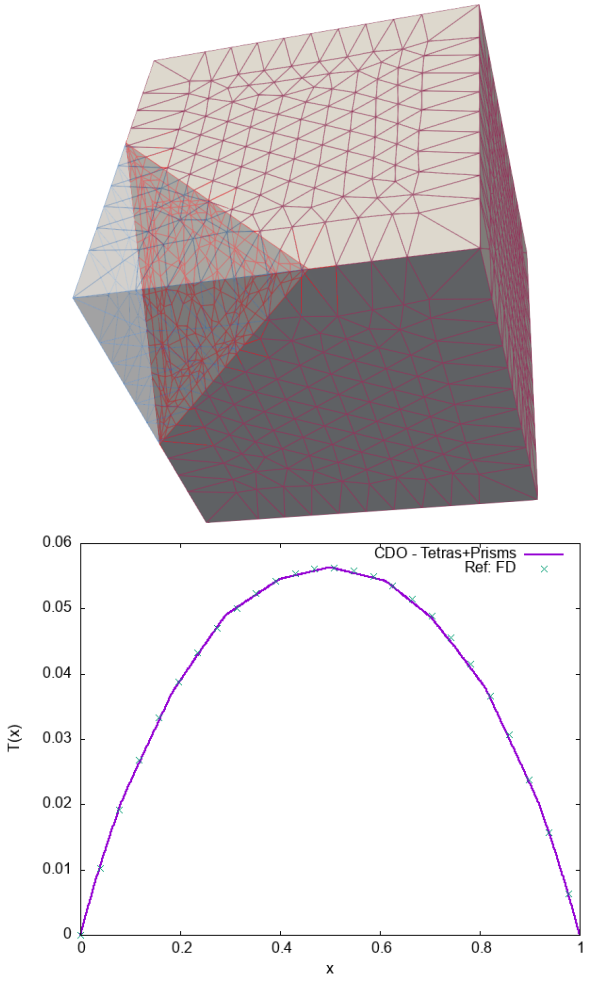

Figure 4: Top: Sketch of the cubic mesh with tetrahedral cells in the inner domain and prism layers at each wall. Bottom: CDO solution against a reference produced by a Finite Difference code.

shows a clip of the mesh, with tetrahedral and prismatic elements, and Fig. 4 (bottom) the profile of the solution in 1 direction (the test is such that all 3 directions show the same solution), obtained using the CDO method, and compared to a reference produced by a Finite Difference (FD) code ran on a refined homogeneous mesh. The CDO solution is very closed to the FD reference, despite the use of a very coarse grid. Figure 5 shows a clip of a 3-D array of 256 ellipsoids. For each of them, a steady vector-valued diffusion equation is computed by CDO in 1 single instance of code_saturne. The ellipsoidal shape is used to make the geometry more

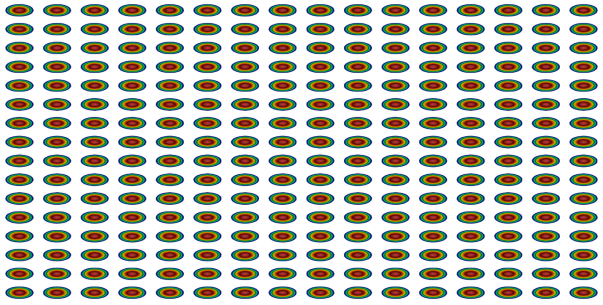

Figure 5: Simulation using 256 separate ellipsoids.

general and concurrent 'bridges' will be computed between neighboring ellipsoids. The same equation will be solved on the resulting single mesh.

4. Conclusions

A merging algorithm between 2 or more meshes has been presented, and its projection and extrusion procedure have been implemented into code_saturne. A 3-D steady vector-valued diffusion equation has been computed as a reference case to show the accuracy of the Compatible Discrete Operator method when using tetrahedral and prismatic elements. Furthermore, it has been shown that one instance of code_saturne's solver is able to compute 256 independent ellipsoidal meshes. More results will be presented for the full merging algorithm applied to 256 ellipsoids. Parallelization will also be explained in detail and timing of the merging procedure will be given, as well as results on the performance of the algorithm.

References

[1] A. Solernou, B. Hanson, R. Richardson, R. Welch, D. Read, O. Harlen, S. Harris, Fluctuating Finite Element Analysis (FFEA): A continuum mechanics software tool for mesoscale simulation of biomolecules, PLoS 14(3) (2018). doi:10.1371/journal.pcbi.1005897.

[2] J. Bonelle, Y. Fournier, C. Moulinec, New polyhedral discretisation methods applied to the Richards equation: CDO schemes in Code_Saturne, Computers & Fluids 173 (2018) 93–102. doi:10.1016/j.compfluid.2018.03.026.

[3] Y. Fournier, J. Bonelle, C. Moulinec, Z. Shang, A. Sunderland, J. Uribe, Optimizing Code_Saturne computations on Petascale systems, Computers & Fluids 45 (2011) 103–108. doi:10.1016/j.compfluid.2011.01.028.