Intersection of Distributed Meshes for Multi-Tracer Transport Schemes

Iulian Grindeanu[†], Kara Peterson[β], Timothy J. Tautges[†], Mark A. Taylor[β], and Patrick H. Worley[§]

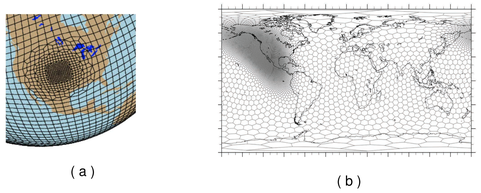
[†]Argonne National Laboratory, [§]Oak Ridge National Laboratory, and [β]Sandia National Laboratories

Introduction

- One of the important goals of ACES4BGC project is to implement and optimize new computationally efficient advection algorithms for large number of tracer species
- The resulting upgrades to the Community Earth System Model (CESM) will deliver new scientific capabilities, offer unprecedented accuracy in representing biogeochemical interactions, and yield *improved predictive skill and computational performance.*
- Significant computational challenges must be overcome to meet the scientific requirements for supporting large numbers of reactive tracers in CESM. The most significant challenge is presented by the advection of many tracers in the atmosphere and ocean component models.

Advection / Transport Problem in Climate Codes

- The transport problem can be cast in Eulerian, Lagrangian or Arbitrary Lagrangian-Eulerian form. One popular variant is semi-Lagrangian method, run in Lagrangian mode for one time step and then interpolate back to the Eulerian mesh (Staniforth and Cote, 1991)
- At every time-step, the semi-Lagrangian approach involves interpolating quantities from a distorted Lagrangian mesh to a regular Eulerian mesh or vice versa, depending on the trajectories. The transport problem is reduced to a regridding problem if the Lagrangian mesh movement is prescribed (Lauritzen et al., 2010)
- Conservative regridding (remapping) has been made practical by the application of Gauss-Green's theorem over the intersection mesh, which converts area-integrals into line-integrals. This approach has been applied for up to second-order static grid-to-grid remapping in (Jones, 1999) and the method was extended for regular lat-lon grids and cubed-sphere grids (Ulrich et al., 2009)
- For unstructured meshes that arise in either CAM-SE or MPAS formulations (figure 1) a more general framework for advection/transport is needed.



(a) (b)

Figure 1: *Unstructured meshes of interest*

Transport Equations

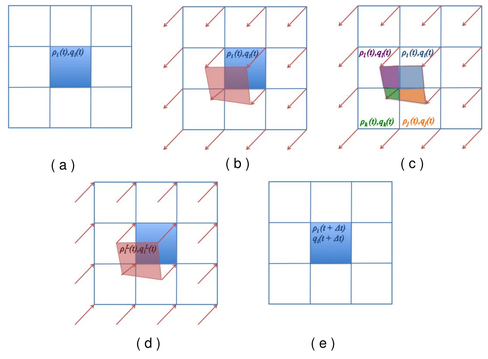
A tracer, represented by its mixing ratio q and mass ρq, is transported in the flow with velocity $\mathbf{u}$

$$\left.\begin{aligned}\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{u} = 0 \\ \frac{\partial \rho q}{\partial t} + \nabla \cdot \rho q \mathbf{u} = 0\end{aligned}\right\} \rightarrow \frac{Dq}{Dt} = 0$$

New methods should satisfy

- Local conservation of ρq
- Monotonicity or shape preservation of q
- Free stream preserving (consistency) when $q = 1$

Semi-Lagrangian Remap Algorithm for Transport



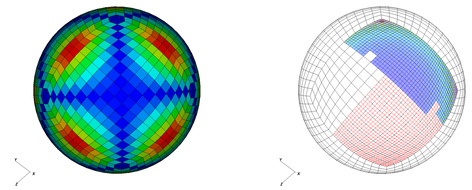
(a) (b) (c) (d) (e)

- Consider a cell c_i with cell-averaged density ρ_i and tracer q_i at time t
- Compute Lagrangian departure grid
- Compute tracer reconstructions on Eulerian (arrival) grid
- Find intersections between departure grid and arrival grid
- Integrate tracer reconstructions over cell intersections
- Update tracer on arrival grid $q_i(t + \Delta t) = q_i^L(t)$

By carefully choosing reconstruction function the method is guaranteed to be tracer mass conserving and free stream preserving. The method is stable for large time steps and in practice can allow for CFL ~ 5 in the atmosphere. It also is efficient for large numbers of tracers because the cell intersections are computed once and used for all tracers.

Coupling with CAM-SE

The MOAB grid intersection algorithm is now being coupled with CAM-SE. The algorithm for tracer transport begins with the initialization of the CAM-SE Eulerian tracer grid in MOAB, distributed on each processor (CAM-SE coarse grid). The CSLAM fine grid is instantiated on each processor, and the parallel infrastructure of MOAB is leveraged, so the ghosting and communication with neighboring processors is relatively easy to do.



(a) (b)

Figure 3: *Initial CAM-SE mesh (a) and CSLAM mesh (refined) on 2 out of 16 processors (b)*

Figure 3 shows the initial CAM-SE coarse mesh, for ne=15, and the distribution of the refined CSLAM mesh on the first 2 processors, for a small run, on 16 processors. The initial partition of mesh is done using an HSFC method implemented in HOMME, and the instantiation of the MOAB mesh data structures is done locally, on each task/processor. Global identificators are used in MOAB to resolve sharing and ghosting of local data with the neighbors.

For each time step, departure point locations computed in CAM-SE are passed to MOAB. MOAB will then compute intersections between departure (Lagrangian) grid and arrival (Eulerian) grids and return the intersection polygons to HOMME, which will continue with remapping.

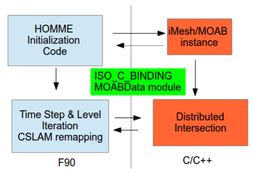


Figure 4: *Software link between HOMME/CESM and MOAB*

Curently, the interface between MOAB and HOMME is done using a new module in HOMME, data is flowing between HOMME and MOAB for initialization stage. A special library is developed in MOAB (mbcslam) , which contains all intersection related code and initialization.

Linear Complexity Algorithm for Intersection of Overlapping Meshes

- The intersection algorithm used in this paper follows the ideas from (Gander and Japhet, 2008), in which two meshes are covering the same domain. At the core is an advancing front method; For simplicity, label the two meshes red and blue, as they are covering about the same domain.
- First, 2 convex cells from the red and blue meshes that are intersecting are identified, by using a search tree. This will constitute the seed of the front. Advancing in both meshes, using adjacency information, we incrementally compute all possible intersections. Important for the algorithm is also a robust scheme, in which 2 intersecting cells from the different meshes are overlapped and resolved. Every edge on one of the meshes maintains a list of intersection points, and geometric tolerances are used to eliminate duplicates and ensure consistency.
- The intersection of cells from different meshes is simplified and more robust if they are convex. If in the initial meshes there exist concave polygons, they are decomposed in simpler convex polygons, by simply splitting along interior diagonals. The intersection between 2 convex cells will be then a convex polygon. The final result of meshes intersection will be a set of convex polygons, that cover the common domain.
- All edges of cells on the sphere are considered to be great circle arcs. Using gnomonic projection, on one of the six planes, each arc becomes a straight line in the gnomonic plane, and a robust intersection is much easier to achieve.

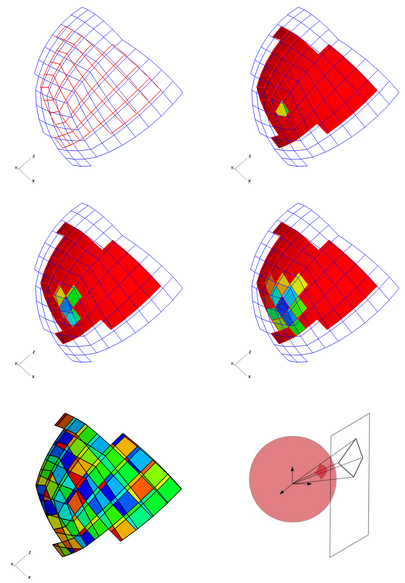
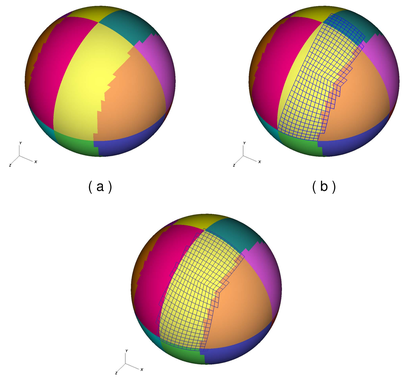


Figure 5: *Advancing front for local intersection*

Figure 5 shows how the algorithm advances; Each red cell is resolved, by building a local queue of blue cells that intersect the red cell. Blue cells are added to the local queue incrementally, using adjacency. At the same time, a global queue with seeds is formed, and it contains pairs of red/blue cells that have the potential to intersect. When there are no blue cells in the local queue, algorithm advances to the next seed from the global queue

Parallel Implementation Considerations

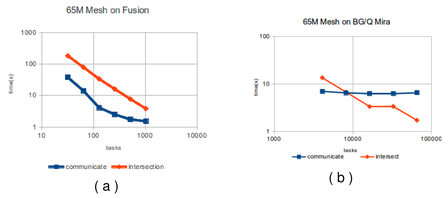
- Use parallel infrastructure from MOAB (Tautges et al., 2004)
- The two meshes to be intersected will be distributed; the arrival mesh is fixed, and can be distributed in advance (use Zoltan (Devine et al., 2002))
- The algorithm works with any distribution of the arrival mesh, and CAM-SE code use space filling curve method.
- The departure mesh needs to be distributed at run time
  - compute boundary boxes for each arrival processor mesh
  - departure points position stored in a MOAB tag, on each arrival node
  - departure cell bounded box computed locally, for each arrival cell
  - sent to processors whose bounding boxes are within tolerance



(a) (b) (c)

Figure 6: *(a) initial distributed arrival grid. (b) departure grid on one of the processors (blue mesh) (c) covering departure mesh after communication*

Figure 6(a) shows initial arrival mesh, distributed over processors. Each vertex knows its departure position, and the algoritm computes departure cells, blue mesh (b). Departure cells that intersect bounding boxes of other processors are sent there. Each arrival grid will be covered by departure mesh (c).



(a) (b)

Figure 7: *(a) results on fusion.lcrc.anl.gov (b) results on mira.alcf.anl.gov*

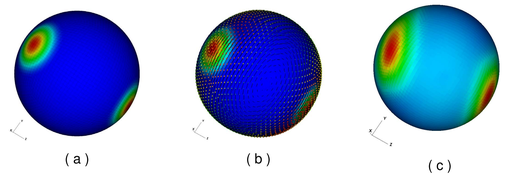
Figure 7 shows the scalability of the algorithm up to 64k tasks, on an MPAS model with 65M cells. Arrival mesh is distributed using Zoltan when loading the model; Recursive Coordinate Bisection is used, which ensures a balanced distribution, and that the bounding boxes of partitions have a low overlap. Intersection is almost embarassingly parallel, as long as the mesh is communicated between processors, and it scales well. It is important to note that this intersection needs to be carried out at each time step and at each level.

At high processor count we notice that the communication cost for departure mesh is not decreasing anymore. Also, these two machines have different networks, and this explains the differences in results. We may need to change the algorithm, or the methodology. One method would be to ghost sufficient layers in the beginning, on arrival mesh. Considering that the target CFL number would be at most 5, 5 ghost layers would be enough. The departure mesh would lay then in arrival mesh and the ghosted region on each processor, so no communication would be needed for the covering mesh.

Tracer Transport Example in MOAB

A simplified transport/advection method is implemented in MOAB, and it is exercising the intersection algorithms explained above. Quantities of interest are discretized on the arrival mesh at each time step.

Higher order approximations should be employed, but in this study, first order approximations were used, i.e., quantities of interest were averaged over each cell. *The original CSLAM type implementation uses higher order approximations.* So for this MOAB implementation, only the area of the intersection polygons determines the resulting tracers concentrations. Still, as some departure polygons will end up on different processors, their tracer mass concentrations need to be communicated to the corresponding arrival cell. So for each time step in the approximation, two communication steps are involved, one to distribute the departure mesh to processors, and second to communicate back the tracer concentrations from the intersection polygons. For a higher order approximations of the tracers on the arrival/departure mesh, complete information about polygons will be needed.



(a) (b) (c)

Figure 8: *(a) initial tracer field (b) velocity field (c) final tracer field*

Figure 8 shows the results of a transport simulation as in (Nair and Lauritzen, 2010), the cosine bell example and non-divergent flow. The transported scalar field follows complex trajectories and undergoes severe deformation during the simulation, but it reverses its course halfway through and returns to the initial position by the end of the simulation. If there would be no errors in the simulation, the initial and final plot should be the same. The pictures show that the constant tracer approximation is not sufficient, higher polynomial order is mandatory.

Conclusions

An efficient intersection algorithm for distributed unstructured meshes has been developed and is currently being integrated into CAM-SE. This intersection algorithm will provide CAM-SE with the ability to perform efficient tracer transport on fully unstructured quadrilateral meshes on the sphere. The infrastructure and methodology will be leveraged for MPAS framework, too.

References

K. Devine, E. Boman, R. Heaphy, B. Hendrickson, and C. Vaughan. Zoltan data management services for parallel dynamic applications. *Computing in Science and Engineering*, 4(2):90–97, 2002.

M. J. Gander and C. Japhet. An algorithm with optimal complexity for non-matching grid projections. In *Proceedings of 18th International Conference on Domain Decomposition Methods*, Springer Lecture Notes in Computer Science, 2008.

P. Jones. First- and second-order conservative remapping schemes for grids in spherical coordinates. *Monthly Weather Review*, 127:2204 – 2210, 1999.

P. H. Lauritzen, R. D. Nair, and P. A. Ullrich. A conservative semi-Lagrangian multi-tracer transport scheme (CSLAM) on the cubed-sphere grid. *J. Comput. Phys.*, 229(5):1401–1424, Mar. 2010. doi: 10.1016/j.jcp.2009.10.036.

R. D. Nair and P. H. Lauritzen. A class of deformational flow test cases for linear transport problems on the sphere. *Journal of Computational Physics*, 229(23):8868 – 8887, 2010. ISSN 0021-9991. doi: http://dx.doi.org/10.1016/j.jcp.2010.08.014. URL `http://www.sciencedirect.com/science/article/pii/S0021999110004511`.

A. Staniforth and J. Cote. Semi-lagrangian integration schemes for atmospheric models a review. *Monthly Weather Review*, 119(9):2206 – 2223, 1991. doi: DOI: 10.1175/1520-0493.

T. J. Tautges, R. Meyers, K. Merkley, C. Stimpson, and C. Ernst. MOAB: a mesh-oriented database. SAND2004-1592, Sandia National Laboratories, Apr. 2004. Report.

P. A. Ulrich, P. H. Lauritzen, and C. Jablonowski. Geometrically exact conservative remapping (gecore): Regular latitude lon-gitude and cubed - sphere grids. *Monthly Weather Review*, 137:1721 – 1741, 2009. doi: DOI: 10.1175/2008MWR2817.1.

2014 SciDAC PI Meeting (July 30 - Aug 1, 2014), Omni Shoreham Hotel, Washington, DC .

ACKNOWLEDGEMENTS: Funding provided through ACES4BGC, FASTMath and SUPER. This research used resources of the ALCF at ANL, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357