Second-order conservative remapping between unstructured spherical meshes

E. Kritsikis, Y. Meurdesoif, T. Dubos

September 25, 2012

Plan

Potential uses of conservative remapping:

- postprocessing
- solver coupling
- adaptive mesh refinement . . .

If you have a remapping step, the order of the dynamics is bounded by the order of the remapping!

Requirements:

- conservation
- high order
- positivity
- efficiency

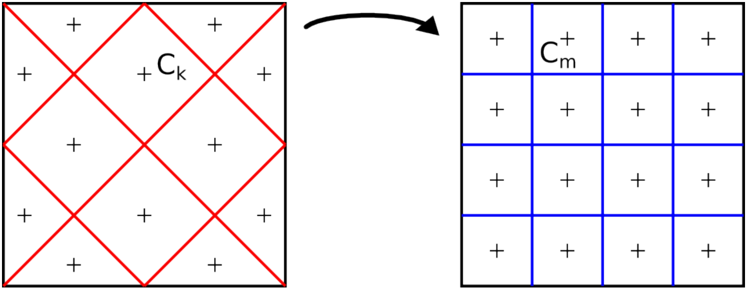
The problem

Given

- 2 meshes S_k, T_m on the sphere
- $f = (f_1 \ldots f_N) \in R^N$: a function's values at centres C_k of S_k

find $f' = (f'_1 \; \ldots \; f'_{N'})$: the function values at C_m, such that

$$\text{“} \int_S f = \int_T f' \text{”} \quad \text{i.e.} \quad \sum_k f_k A_k = \sum_m f_m A_m$$



$\bar{f}_k := f_k A_k$ elementary mass

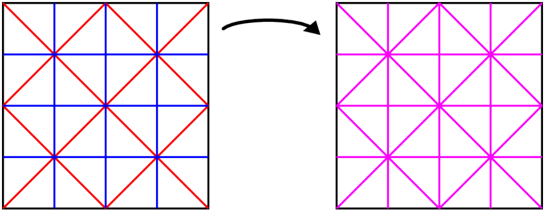
Local conservation: for any U that is both

- union of elements of S
- union of elements of T,

$$\int_U f = \int_U f' \quad \text{i.e.} \quad \sum_{S_k \in U} \bar{f}_k = \sum_{T_m \in U} \bar{f}_m \qquad (1)$$

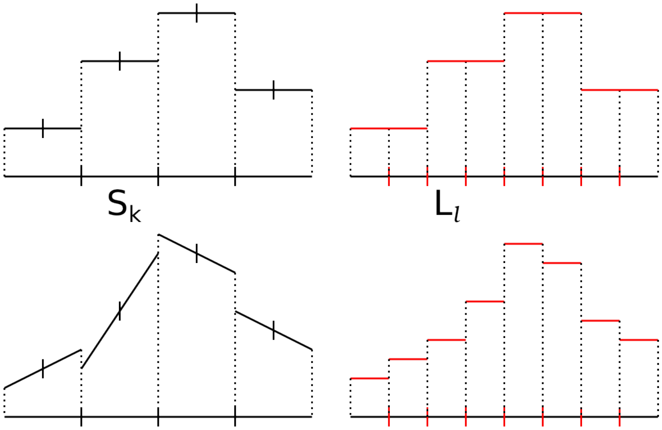
If there is an underlying decomposition L = mesh of intersections or "supermesh", $(S_k \cap T_m)_{k,m}$ s.t.

$$\forall k, \bar{f}_k = \sum_{L_\ell \in S_k} f_\ell A_\ell, \quad \text{then} \quad \forall m, \bar{f}_m := \sum_{L_\ell \in T_m} f_\ell A_\ell$$ then (1) holds.



$$\bar{f}_k = \sum_{L_\ell \in S_k} f_\ell A_\ell$$

1. compute the weights A_ℓ
2. find suitable f_ℓ : reconstruct f on the supermesh
   → 1st, or 2nd order (compute gradients on S)

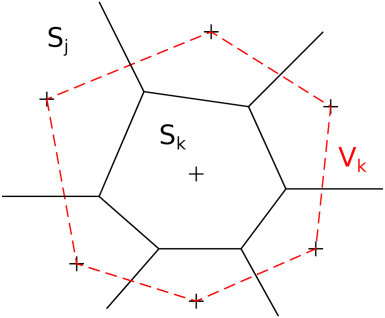


In 2D gradients are computed by the Gauss formula:

$$\int_{V_k} \nabla f \, da = \int_{\partial V_k} (f - f_k) \, n \, ds$$

or at the discrete level

$$g_k = \frac{1}{A(V_k)} \sum_{j \sharp k} \left(\frac{f^j + f^{j+1}}{2} - f^k \right) n^j$$



So in 1st order, $\forall \ell$ s.t. $L_\ell \in S_k$, $\quad f_\ell = f_k$
Whereas in 2nd order, $\quad f_\ell = f_k + g_k \cdot (C_\ell - C_k)$

Let's check that $\displaystyle\sum_{L_\ell \in S_k} f_\ell A_\ell = \bar{f}_k$.

1st order:

$$\sum_{L_\ell \in S_k} f_\ell A_\ell \quad = \quad f_k \sum_{L_\ell \in S_k} A_\ell \quad = \quad f_k A_k \quad = \quad \bar{f}_k$$

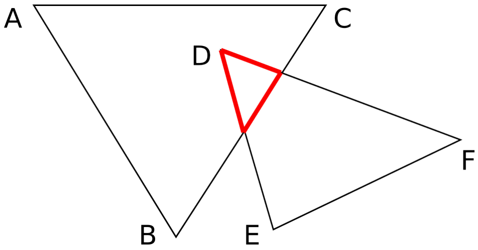
2nd order:

$$\sum_{L_\ell \in S_k} f_\ell A_\ell \quad = \quad f_k \sum_{L_\ell \in S_k} A_\ell + \sum_{L_\ell \in S_k} g_k \cdot (C_\ell - C_k) A_\ell$$

- $g_k \cdot C_k = 0$ it should be; if necessary $g_k := g_k - (g_k \cdot C_k) C_k$
- $g_k \cdot \sum A_\ell C_\ell = 0$ true if $A_k = \sum A_\ell C_\ell$:
  source centroids must be barycenters
  of underlying supermesh

How to compute the weights A_ℓ?

Find the intersection of two (spherical) polygons.



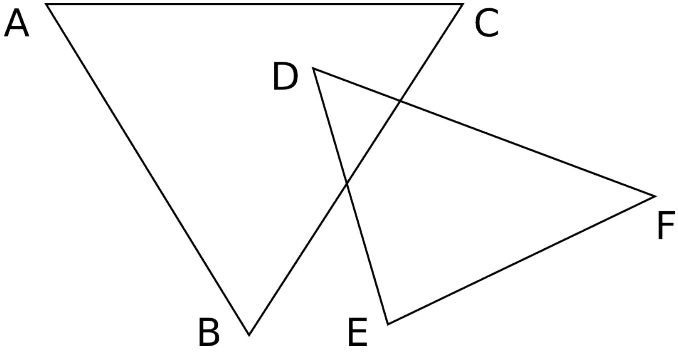
Then the area can be computed by the defect formula:

$$A = \sum \alpha - (n-2)\pi,$$

also called Girard's (1595-1632) theorem.

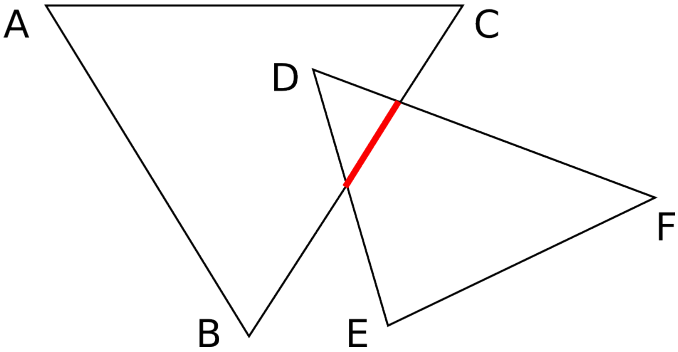
Loop over the sides of both polygons.

Side AB: A, B both exterior, no intersections.
→ dump that side.



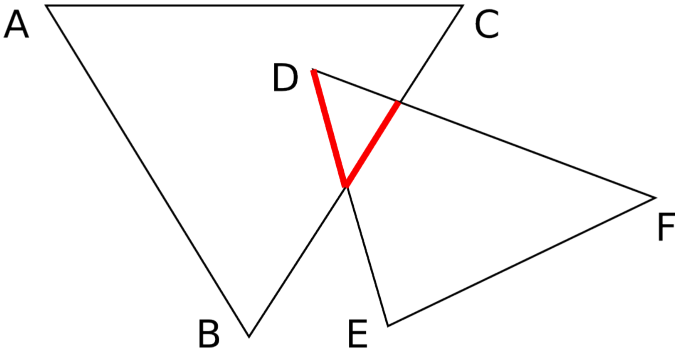
Side BC: B, C both exterior, 2 intersections B’, C’.
→ keep the B’C’ bit.

Side CA: C, A both exterior, no intersections.



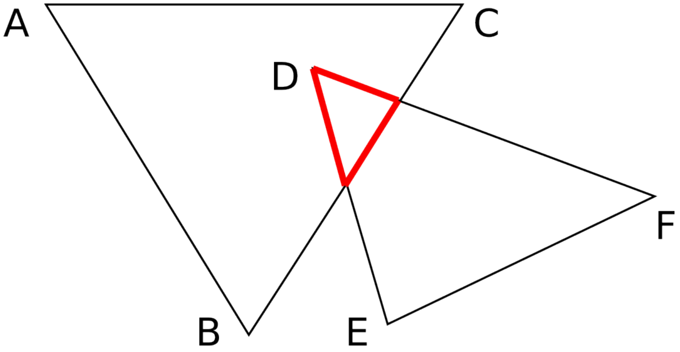
Side DE: D interior, E exterior, 1 intersection B'.
→ keep DB'.

Side EF: E, F both exterior, no intersections.



Side FD: F exterior, D interior, 1 intersection C’ → keep DC’.

Common sides count once.
Then sides are arranged in natural order.



OK for two polygons . . .
but with two sets of O(N) polygons?

Testing for all k, m whether S_k and T_m intersect is a
N^2 problem.

→ intersection computations should be local.

Need for a fast search algorithm for potential intersectors.

The same algorithm can be used to find

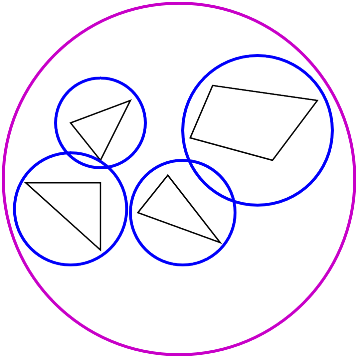
- intersections with another mesh
- neighbours of the same mesh (share exactly one side).

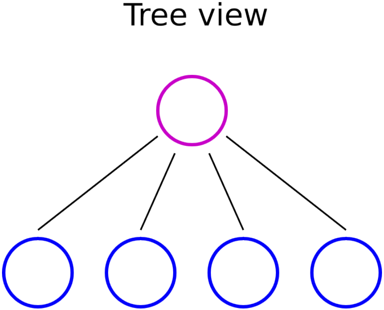
Hierarchical division of space

The elements S_k are wrapped into their circumcircles $\mathcal{C}(S_k)$ called nodes.

Nodes are enclosed into higher-order nodes (each containing a bounded number of nodes).

And so on.





Recursive search tree

Search algorithm

Level 0: Test if $\mathcal{C}(T_m)$ intersects the root node.
Level n: if $\mathcal{C}(T_m)$ and node P are intersecting, test for intersections between $\mathcal{C}(T_m)$ and the children of P.

Too many intersections slow down the search.

Tree construction is an important step (preprocessing).

The tree is built bottom-up: begins with an empty root of level 1. The $\mathcal{C}(S_k)$ are inserted into R_1:

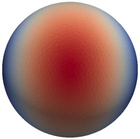
- the centre of R moves to the barycenter of $\mathcal{C}(S_k)$
- the radius of R is updated to enclose the $\mathcal{C}(S_k)$

When the number of children reaches a threshold value, R is split in two, and a new root R_2 is created.

Insert into a higher-order node = insert into its closest child.

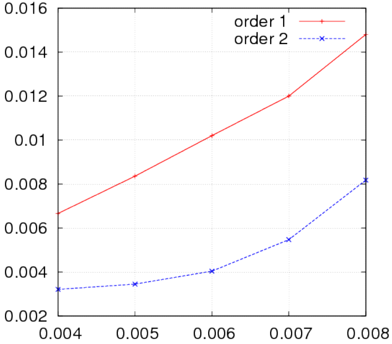
Test-case: spherical harmonic

$$f(\theta, \phi) = 2 + \cos^2(\theta) * \cos(2\phi)$$

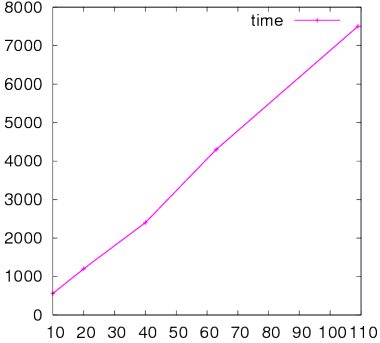


Conservation error $\approx 10^{-12}$.

Error $\|\frac{f_{remap} - f}{f}\|_\infty$



Computation time



Conclusions

- conservative remapper (almost) up to machine precision
- second-order interpolation
- logarithmic complexity

Ongoing work

- parallelisation
- conservation of vectors (what does it mean?)