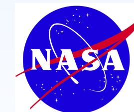
Spherical Barycentric Coordinates: Implications for Metrics and Validation

A. Pembroke[1], B. Curtis[2], L. Rastaetter[1]

1. Goddard Space Flight Center 2. George Mason University



Motivation: Metrics depend heavily on interpolation, but how do we handle the "hard-to-reach" places?

When this matters

Conjugate mapping through MHD models

M-I coupling

DST approximation

Work-arounds

dipole approximation

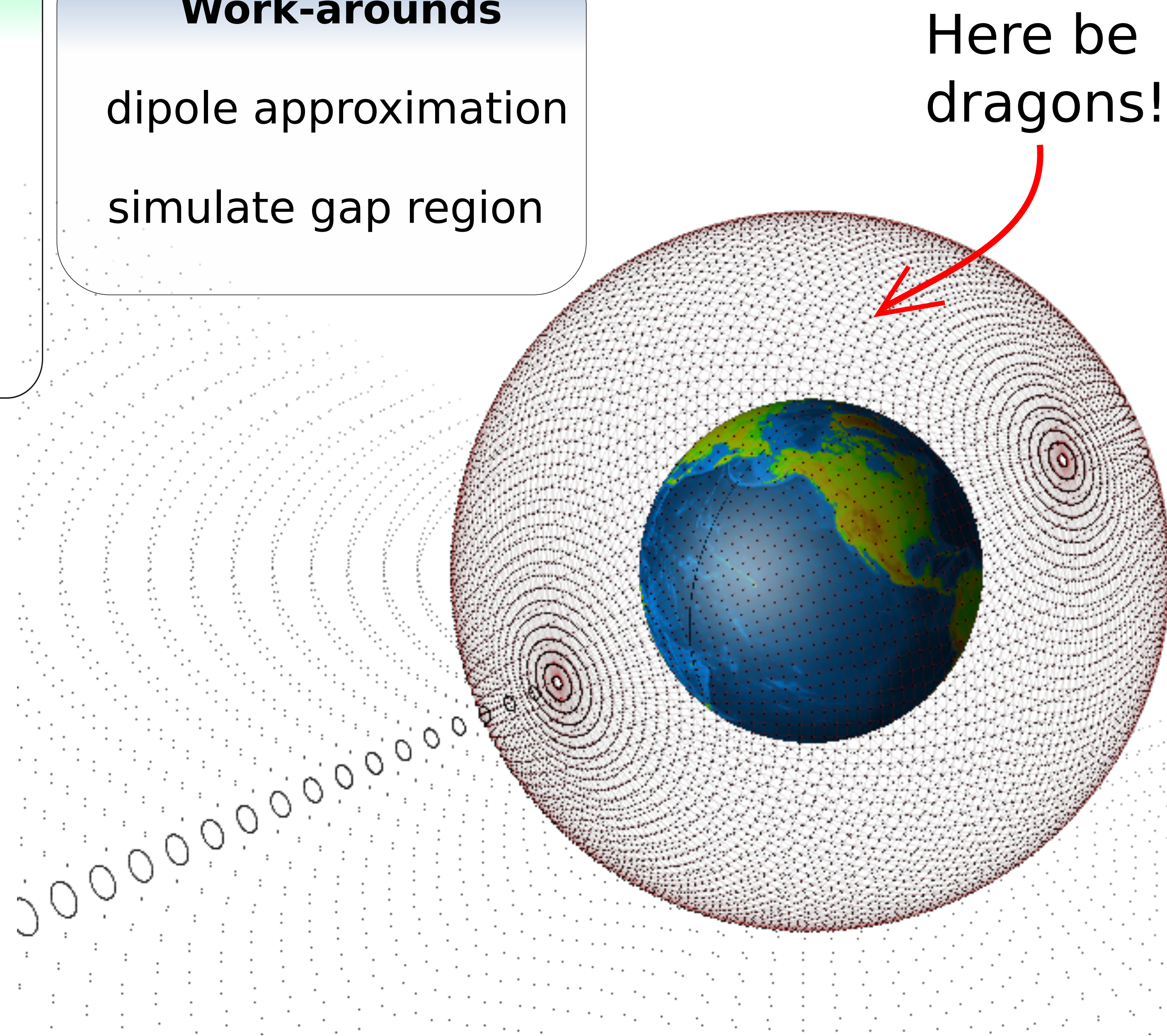
simulate gap region

Problem areas

Discontinuities

Complexity

Information loss obsfuscation



Spherical Barycentric Coordinates

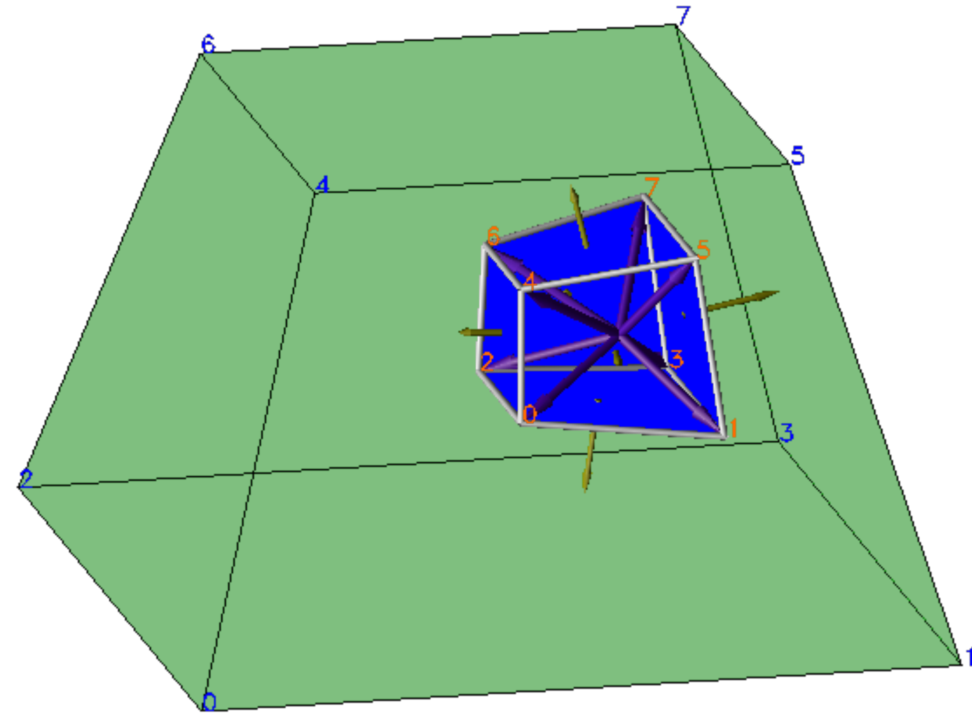
LANGER T., BELYAEV A., SEIDEL H.-P.: Spherical barycentric coordinates. In *Siggraph/ Eurographics Sympos. Geom. Processing* (2006) , pp. 81088.

Goal: Given a point **p**, compute weights w_i of polyhedron vertices $\mathbf{v}_i$, such that:

$$\Sigma_i\, w_i(\mathbf{p})\, \mathbf{v}_i = \mathbf{p} \quad \text{linear precision}$$

$$w_i(\mathbf{p}) > 0 \quad \text{positivity}$$

$$\Sigma_i\, w_i(\mathbf{p}) = 1 \quad \text{partition of unity}$$



step1: Map vertices to sphere centered at **p**.

$$\mathbf{u}_i = \text{norm}(\mathbf{v}_i - \mathbf{p})$$

step2: Compute face vectors, s.t. $\Sigma\, \mathbf{v}_F = \mathbf{0}$

$$\mathbf{v}_F = \Sigma_i\, \varphi_{i,i+1}\ \text{norm}(\mathbf{u}_i \times \mathbf{u}_{i+1})$$

$$\varphi_{i,i+1} = \text{angle}(\mathbf{u}_i, \mathbf{u}_{i+1})$$

step3: For each vertex incident on face F:

$$\lambda_i(\mathbf{v}_F, F) = \frac{|\mathbf{v}_F|}{|\mathbf{v}_i - \mathbf{p}|} \bullet \frac{\tan(\alpha_i/2) + \tan(\alpha_{i-1}/2)]/\sin\theta_i}{\Sigma_j \cot\theta_i\, (\tan(\alpha_j/2) + \tan(\alpha_j/2))}$$

$$\theta_i = \text{angle}(\mathbf{v}_F, \mathbf{u}_i)$$

$$\alpha_i = \text{angle}(\mathbf{v}_F \times \mathbf{v}_i, \mathbf{v}_F \times \mathbf{v}_{i+1})$$

step4: Final weights given by

$$w_i = \omega_i(\mathbf{p}) / \Sigma_j \omega_j(\mathbf{p})$$

$$\omega_i(\mathbf{p}) = \Sigma_{F(i)}\, \lambda_i(\mathbf{v}_F, F)$$

Where F(i) represents all faces incident on vertex i

Spherical Barycentric Coordinates

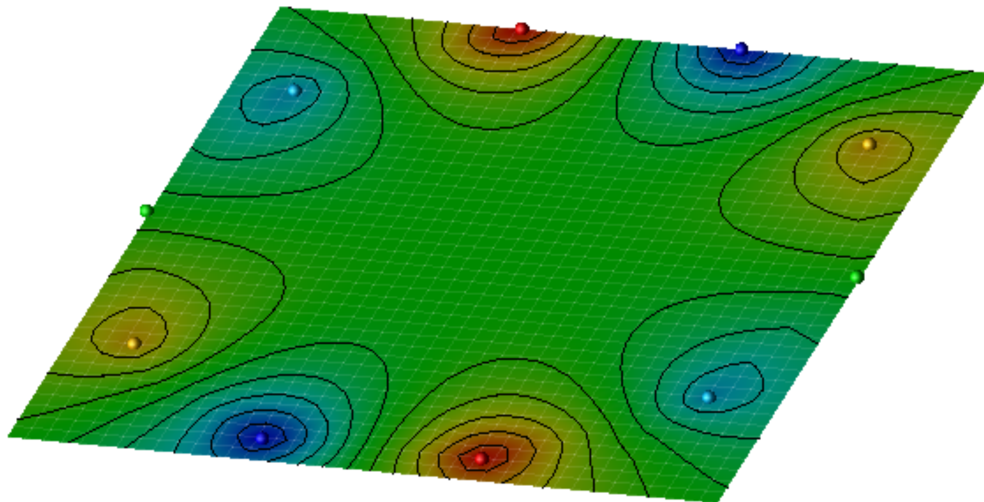
Limiting Behavior on faces: Floater Coordinatees

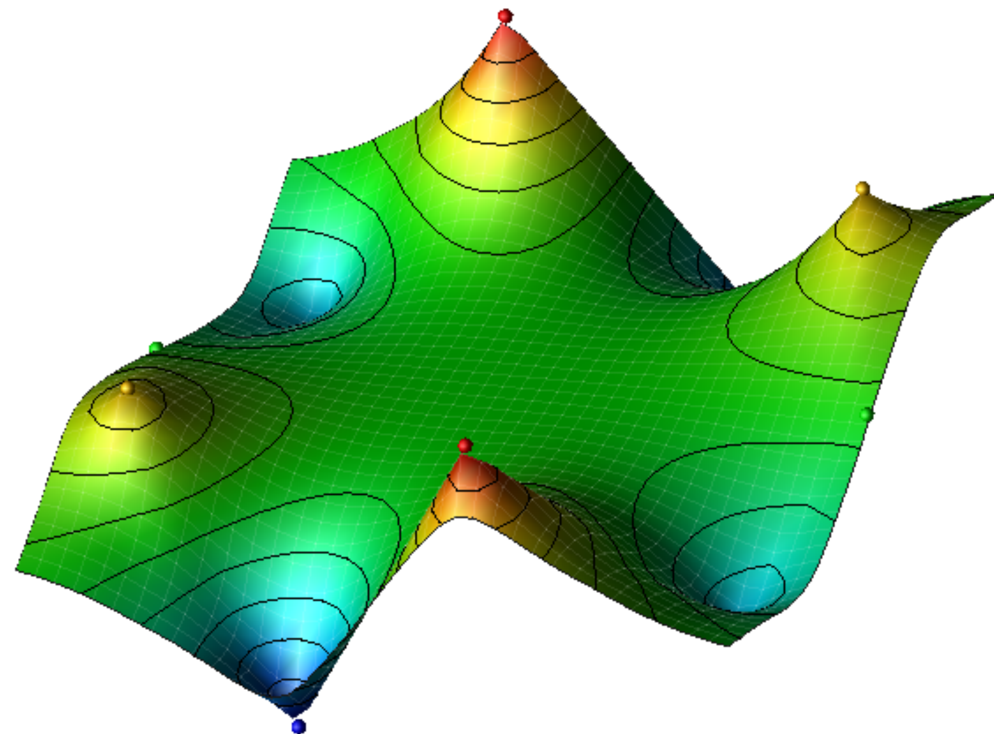
Floater, Michael S. "Mean value coordinates." *Computer aided geometric design* 20.1 (2003): 19-27.

i.c.b.s.t. as **p** approaches face *F*, θ -> pi/2 and weights approach Floater's coordinates:

$$w_i = \begin{cases} \omega_i(p)/\Sigma_j\omega_j(p) & \text{if } i \text{ incident on } F \\ 0 & \text{otherwise} \end{cases}$$

$$\omega_i(\mathbf{p}) = \frac{\tan(\alpha_i/2) + \tan(\alpha_i\text{-}1/2)}{|\mathbf{v}_i - \mathbf{p}|}$$



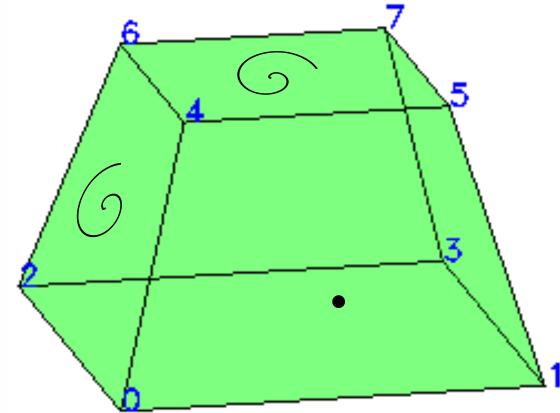


Kameleon-plus Implementation

CCMC's Access and Interpolation Library

https://code.google.com/p/ccmc-software/

ccmc::Polyhedron<T>



Public Variables:

weights: {0, 0, 0, ...}
vertices: {**v0**, **v1**, **v2**...**v7**}
loops: {0, 4, 6, 2,   4,5,7,6,   1,3,7,5 ....} //index into vertices (normal oriented by right hand rule)
faces: {0, 4, 8, 16, 20, 24} //index into loops

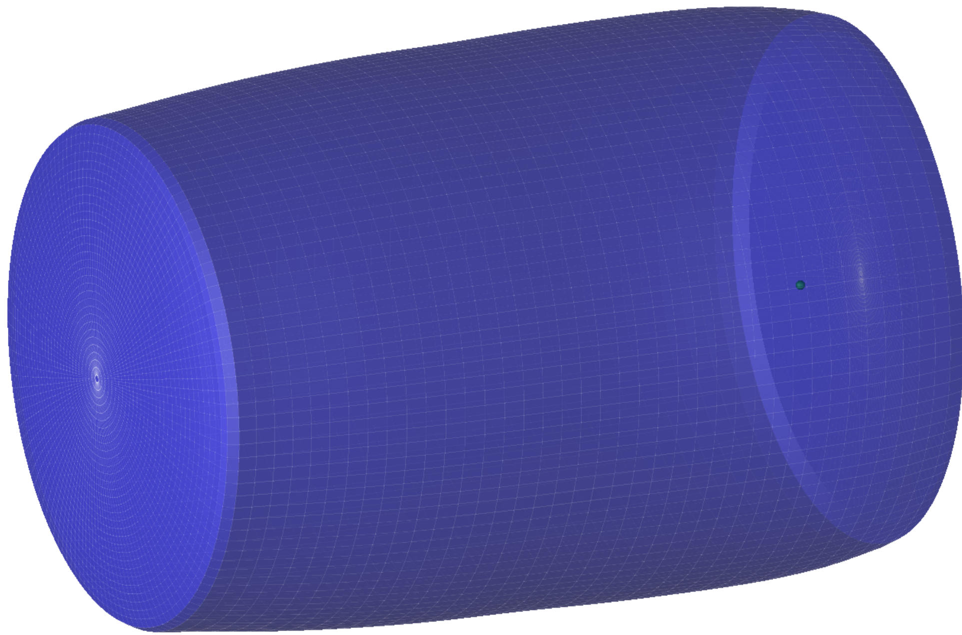
Public Methods:

void **setPositions**(std::vector<Vector<T>> positions)
void **setLoopsFaces**(std::vector<int> loops, std::vector<int> faces)
void **setBarycentricCoordinates**(Vector<T> point)
bool **isPointInside**(Vector<T> point)
int **getClosestFace**(Vector<T> point)
void **saveAsDXObject**(string filename)
Vector<T> **testLinearity**(Vector<T> point)

Virtual Methods:

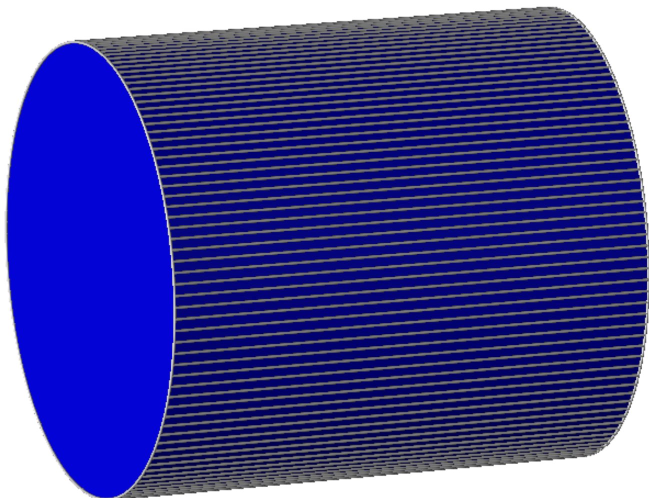
int getType()
Polyhedron<T>* **getNextPolyhedron**()

Application: Analaysis on LFM Mesh

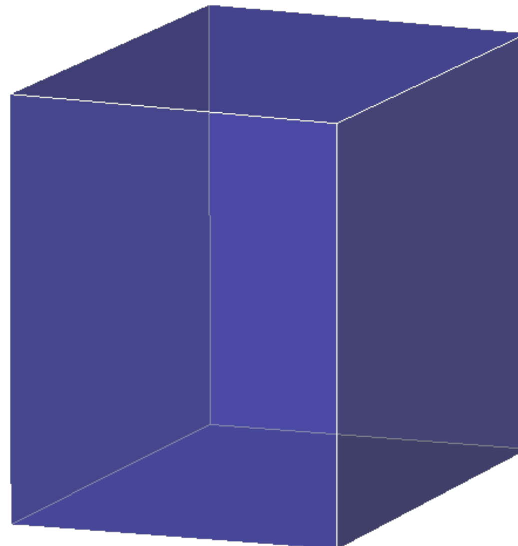


LFM Mesh (outer boundary)
Bow-shock weighted equatorial plane, rotated around earth-sun line.

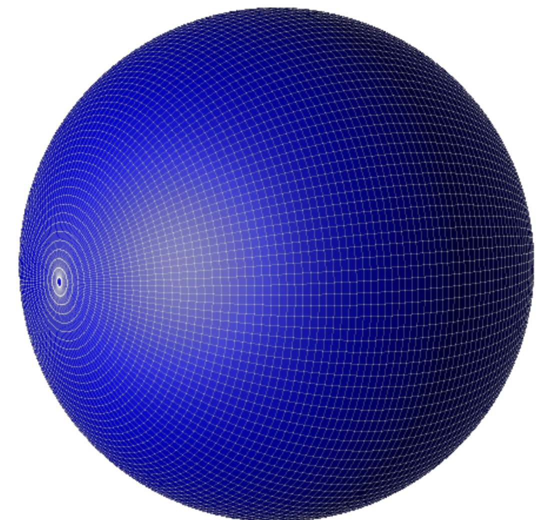
2 x ni-1 axis cells
nk+2 faces each



ni*nj*nk
hexahedra

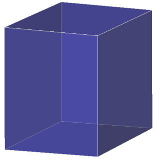


1 Inner Boundary

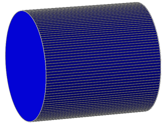


Application: Interpolation on LFM Mesh

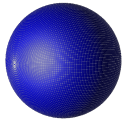
"You must construct additional Pylons!"



class GridPolyhedron: public Polyhedron<T>



class AxisPolyhedron: public Polyhedron<T>



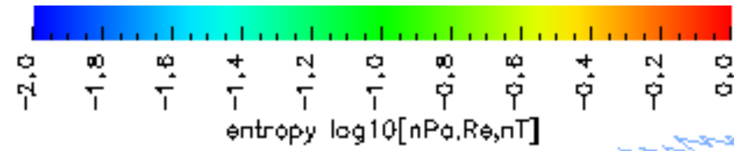
class IPoly: public Polyhedron<T>

LFMInterpolator::**interpolate**(variable, point)

```
getCell(point){
   searchCell = kd-tree(point) //gets cell with closest centroid
   while(!searchCell.isInside){ //insures that point is actually inside the output cell
       searchCell = getNextCell(searchCell) //uses closest face to search locally
       searchCell.setBarycentricCoordinates(point)
   }
}
searchPoly = getCell(point)
searchPoly.setBarycentricCoordinates(point)
for each v_i in searchPoly{
   result += variable(globalIndex(v_i))*w_i
}
return result
```

Results:
Field line Entropy Analysis

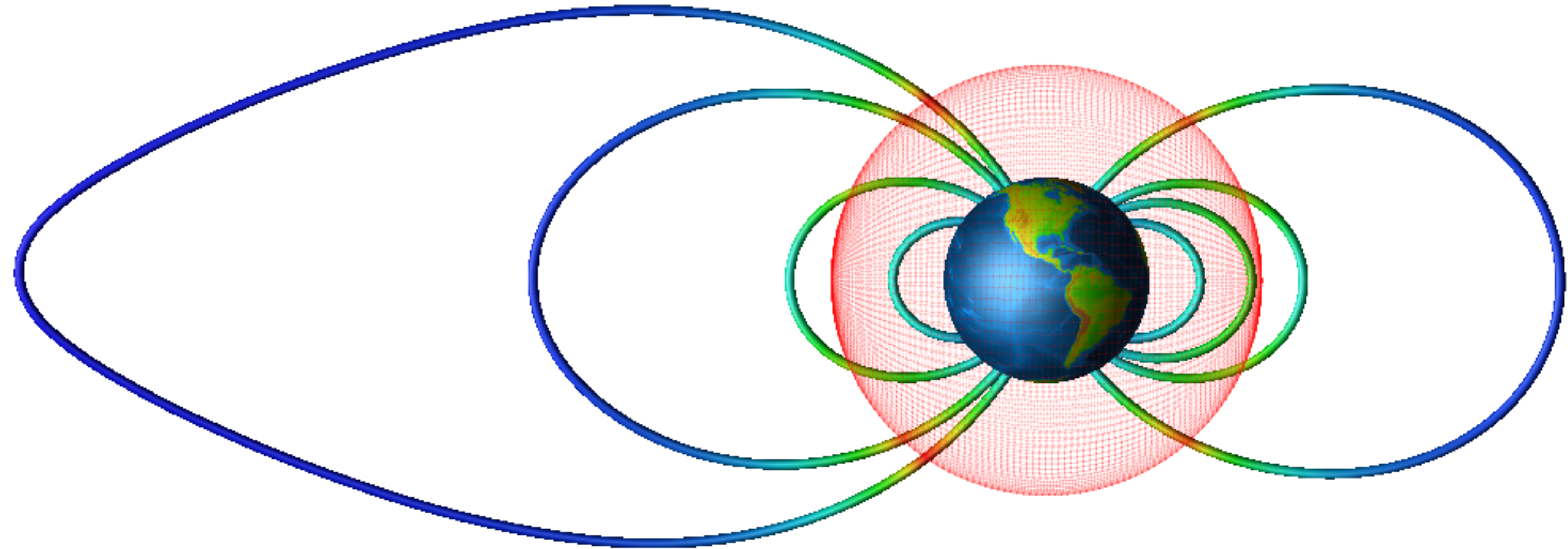
$PV^{5/3} = [\text{int } P^{3/5}\, ds/B]^{5/3}$



LFM-RCM coupled run at quad Resolution

Results:
Low-latitude mapping

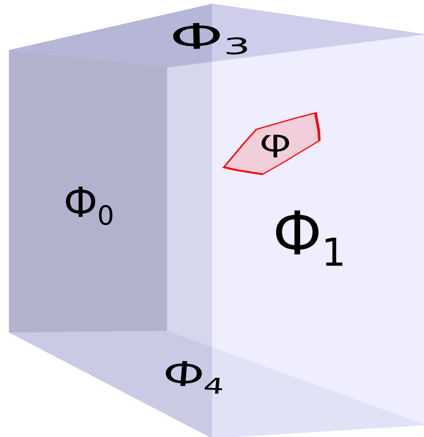
Fieldlines mapped continuously through inner boundary to earth



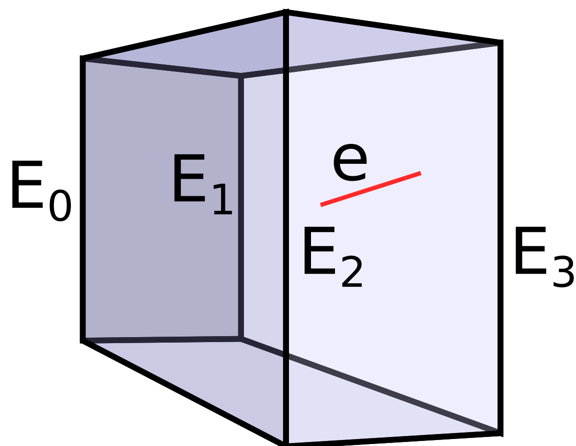
Behavior near earth center:
field direction consistent with dipole but magnitude -> 0 !
This is because we are interpolating vectors, not fluxes!

Future Work:
DST approximation, current interpolation, particle tracing

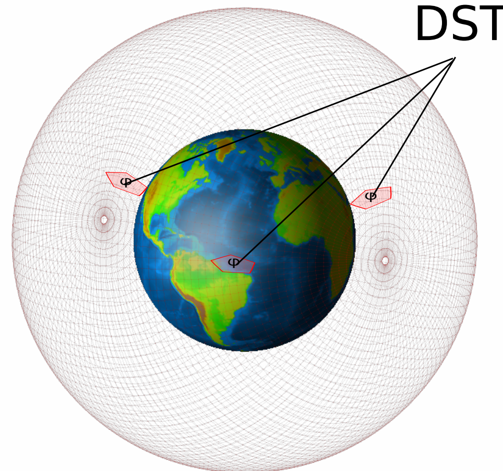
What is the flux through an arbitray polygon?

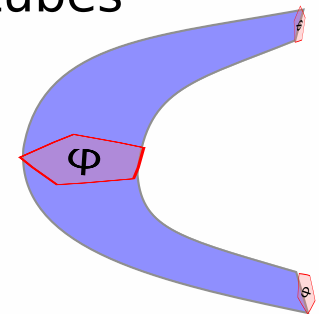


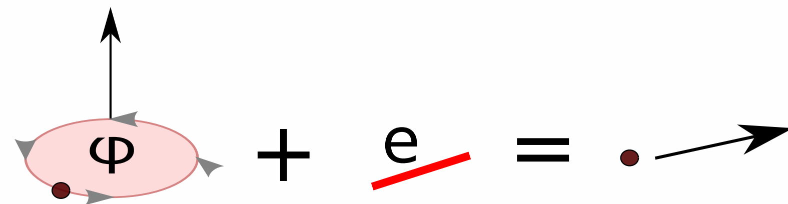
What about edge-based E-fields? Currents?



Applications





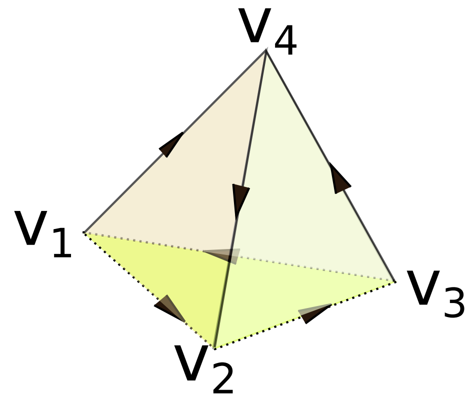


Adiabatic particle tracer

Future Work: Whitney Forms!

Hirani, A. N. *Discrete exterior calculus (dissertation)*. Technical report, California Institute of Technology, 2003.

Luckily, all of the parameters we are interested in have a natural framework for interpolation using the so-called "Whitney Forms"



oriented simplex

Scalar potential	0-form
Electric Field	1-form
Magnetic Flux, Particle Flux	2-form
Density	3-Form

The whitney forms are constructed from combinations of **barycentric coordinates** in the vertices. For a 2-form, each face gets a weight determined by the barycentric weights of incident vertices:

$$W([v_1,v_2,v_3])=2(\omega_1\mathbf{d}\omega_2\wedge\mathbf{d}\omega_3 - \omega_2\mathbf{d}\omega_1\wedge\mathbf{d}\omega_3 + \omega_3\mathbf{d}\omega_1\wedge\mathbf{d}\omega_2)$$

d is called the **discrete exterior derivative**, a fully generalized version of grad, curl, div., while ∧ is the discrete "wedge product"

Francesca Rapetti, Weights computation for simplicial Whitney forms of degree one *Comptes Rendus Mathematique*, Volume 341, Issue 8, Pages 519-523