

Scalable Vector Calculus for Geoscientific Analysis on Unstructured Grids uxarray

Esther Gallmeier

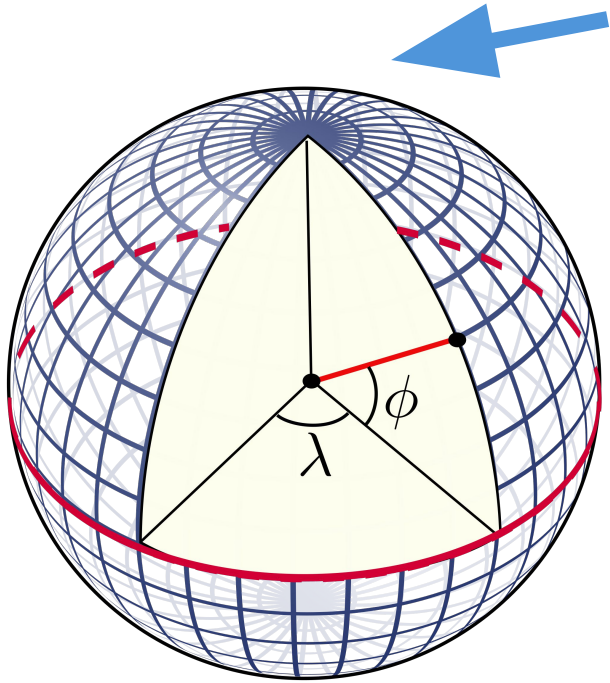
July 29th, 2025

Philip Chmielowiec, Orhan Eroglu, Katelyn Fitzgerald, Rajeev Jain



Structured Grids

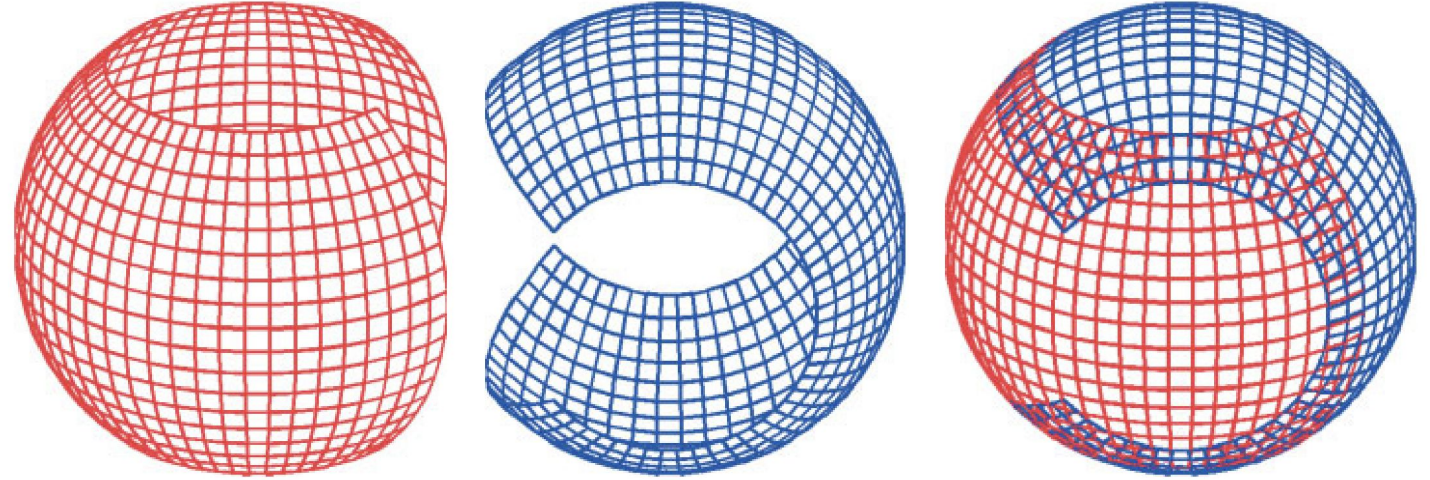
“Full” Lat-Lon Grid



Clustering at poles leads to **scalability issues** for massively parallel computations

Unstructured Grids

“Modified” Lat-Lon Grids



Other Polygonal Grids



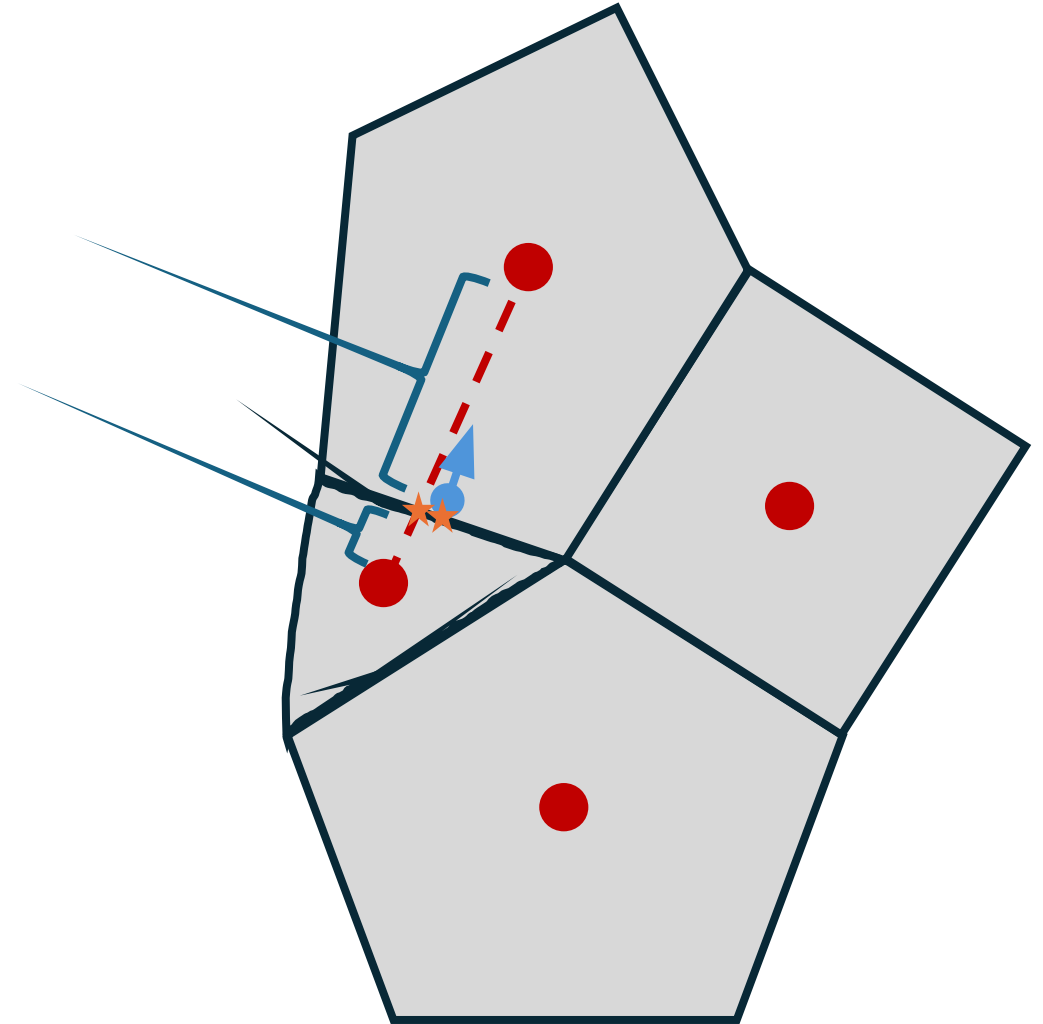
ICON

CAM

MPAS 2/17

Unstructured Grids - *challenges*

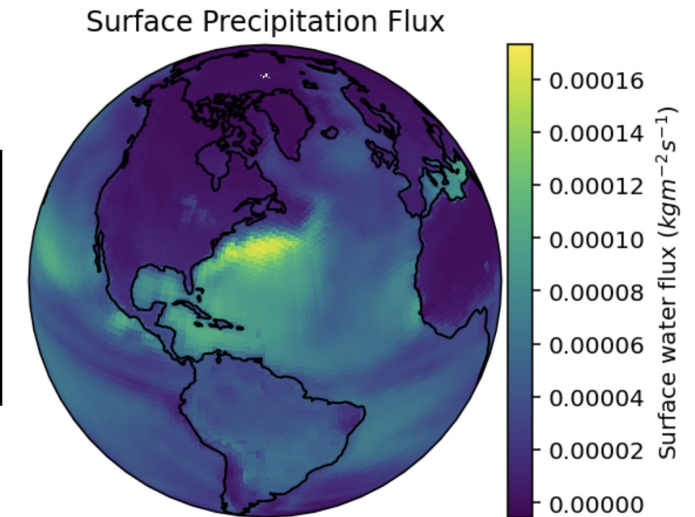
- Arbitrary polygons (not same shape or size)
 - Unevenness
 - Non-orthogonality
 - Skewness
- **No widely-used convention** of file formats for storing arbitrary unstructured grids



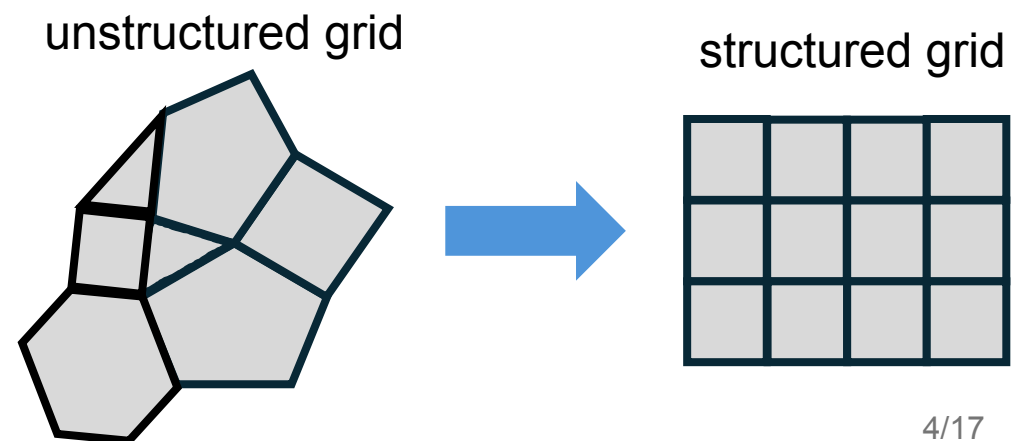
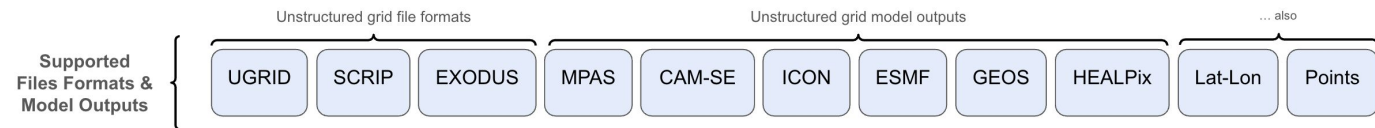
uxarray - *what is it?*

- Open-source Python package for geoscientific **data analysis & visualization** on unstructured grids

DOE's global climate model E3SMv2 output on a **cubed sphere grid** with 21600 faces



- **Supports** wide range of unstructured grid formats/files
- **Avoids** regridding





uxarray

- adding vector calculus

- Need scalable & robust vector calculus operations:

- **Gradient**

Measures magnitude & direction of steepest change of a scalar field

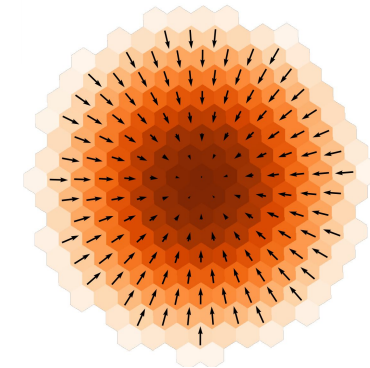
- **Curl**

Measures circulation of a vector field

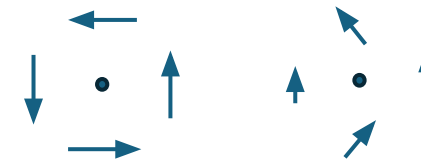
- **Divergence**

Measures how a vector field acts like a “sink” or a “source”

∇ Gaussian Field



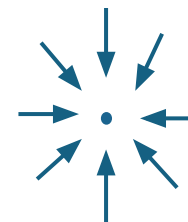
$$(\nabla \times \vec{v}) \cdot \vec{k} > 0$$



counter-clockwise rotation

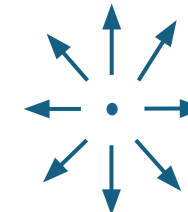
$$\nabla \cdot \vec{v} < 0$$

sink



$$\nabla \cdot \vec{v} > 0$$

source



$$\nabla \cdot \vec{v} = 0$$

neither



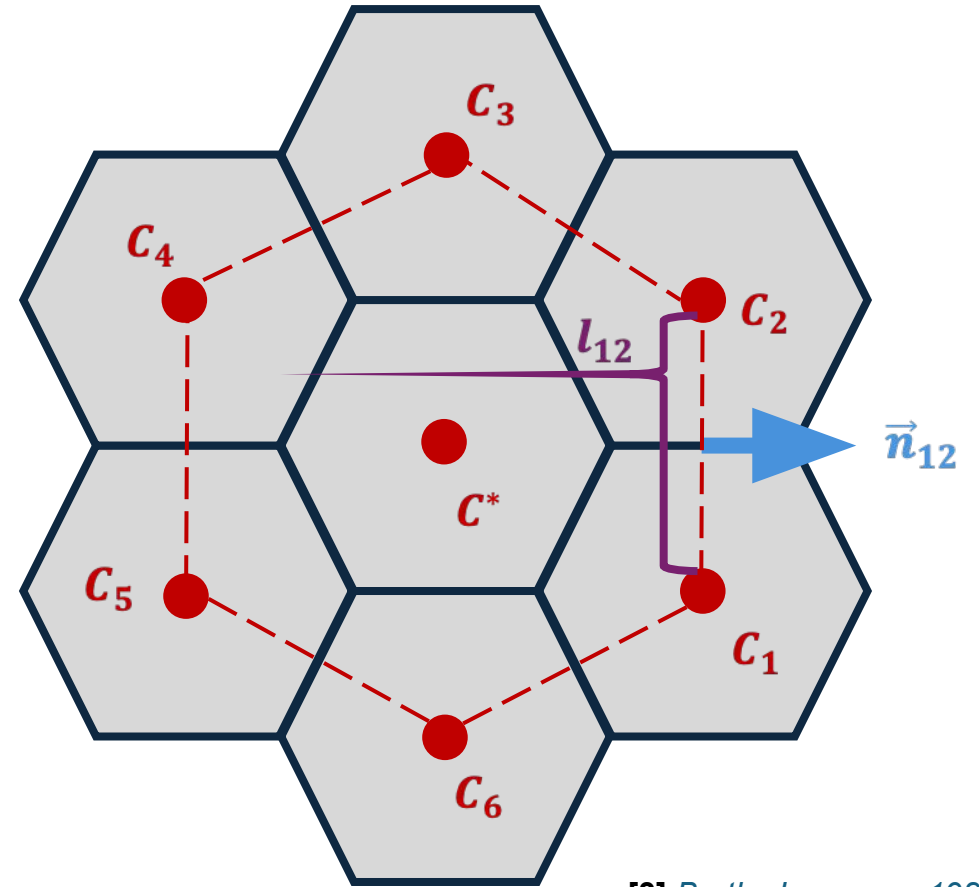
Gradient Implementation (*behind the scenes*)

Input: Scalar field ϕ $\xrightarrow{\phi.\text{gradient}()}$ Output: Vector field $\nabla\phi$

Finite volume discretization of Green-Gauss theorem:

$$\int_V \nabla\phi \, dV = \oint_{\partial V} \phi \, dS$$

$$\nabla\phi(\mathbf{C}^*) \approx \frac{1}{\text{Vol}(\mathbf{C}^*)} \sum_{i,j} \frac{\phi(\mathbf{C}_i) + \phi(\mathbf{C}_j)}{2} l_{ij} \vec{n}_{ij}$$



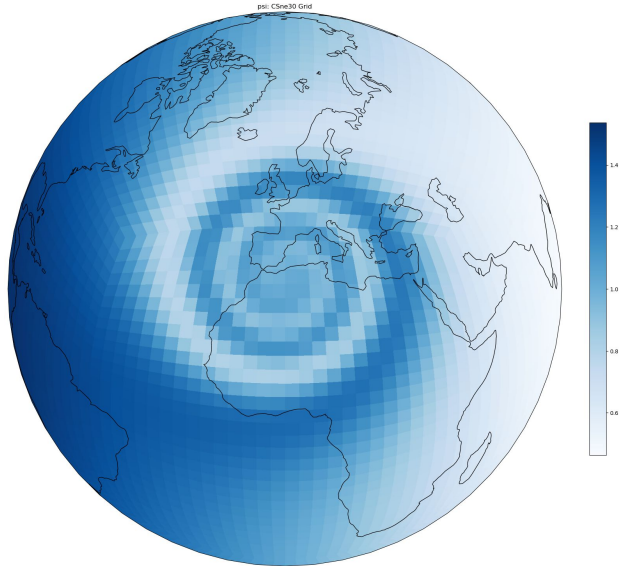
- [3] Barth, Jespersen. 1989.
- [4] Tomita, Hirofumi, et al. 2001.
- [5] Kritsikis, Evaggelo. 2017.

Gradient Implementation

Input:
Scalar field ϕ

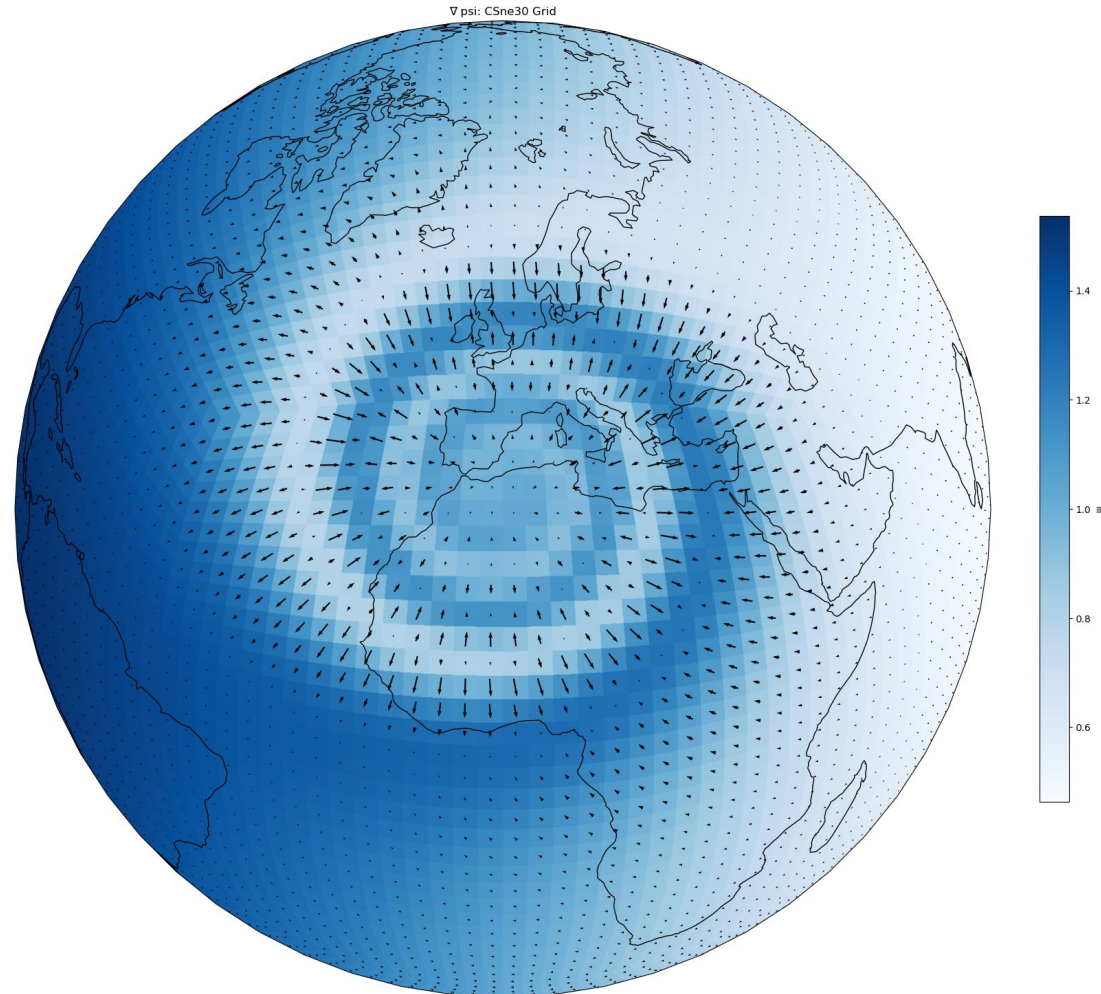
$\phi.\text{gradient}()$

Output:
Vector field $\nabla\phi$



Synthetic face-centered “psi” data on a **Cubed Sphere grid** with 5400 faces, 5402 nodes & 10800 edges

**magnitude & direction of
steepest change**



Gradient Implementation

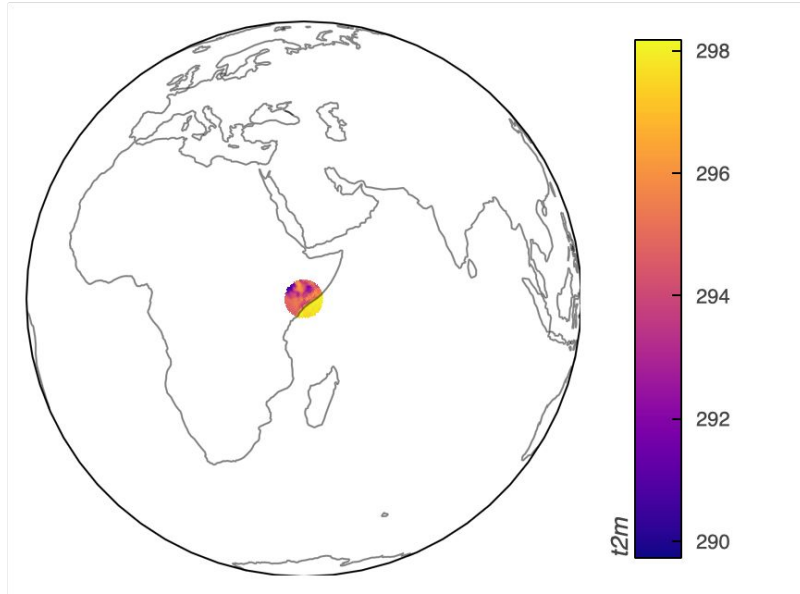
Input:

Scalar field ϕ

$\phi.\text{gradient}()$

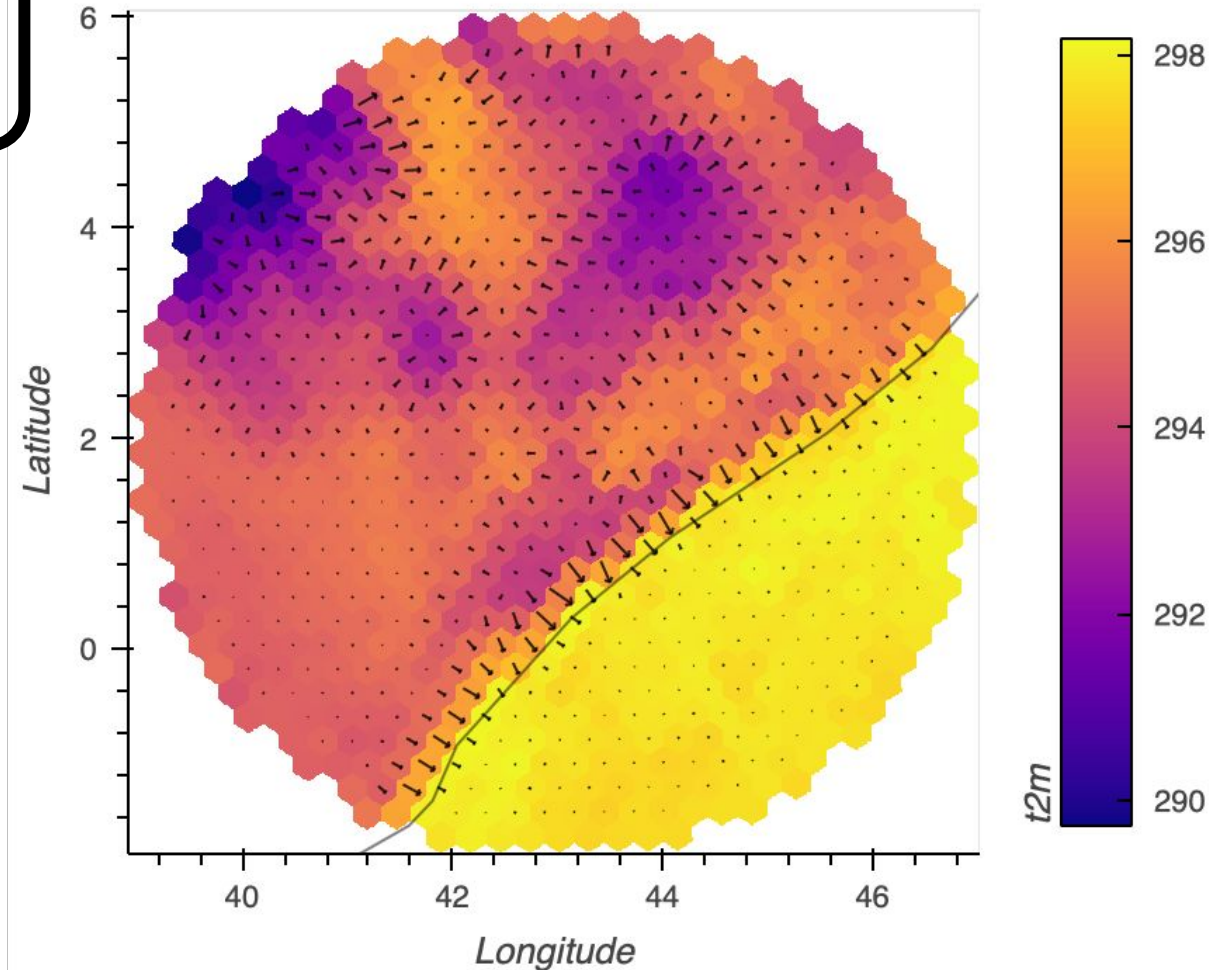
Output:

Vector field $\nabla\phi$



Face-centered data of 2 meter temperature on a **30 km MPAS mesh** with 655,362 faces, 1,310,720 nodes & 1,966,080 edges


magnitude & direction of
steepest change



Gradient Performance (optimize using)

MPAS Atmosphere Mesh	30 km	3.75 km
# of Faces	655,362	41,943,042
# of Edges	1,966,080	125,829,120
# of Nodes	1,310,720	83,886,080
Grid Storage	~0.38 GB	~18.45 GB

*Grids can be **big**
(& the data too)*

- What is  ?
 - **Easy to use!** (needs a decorator)
 - Translates Python functions to **optimized machine code** at runtime
 - Can **approach** the speeds of C or FORTRAN

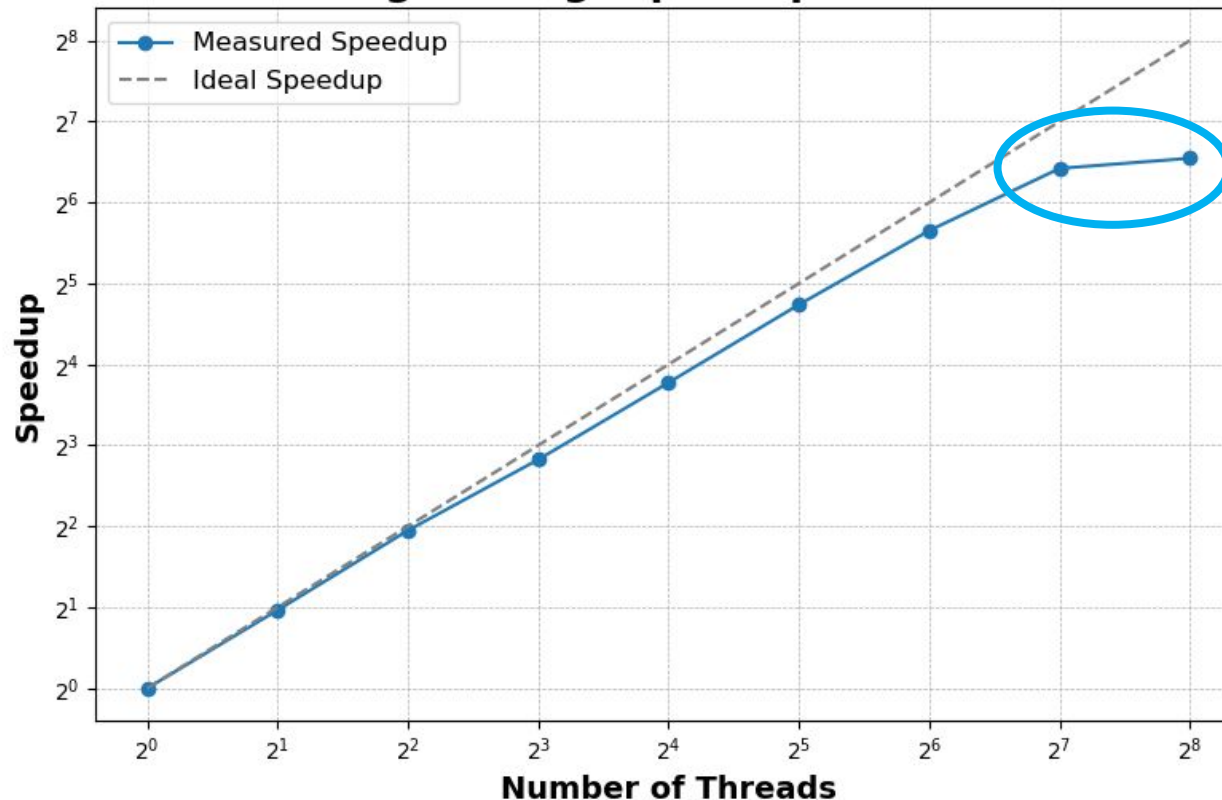
```
1 from numba import njit, prange
2 @njit(cache=True, parallel = True)
3 def gradient(data: np.ndarray) -> np.ndarray:
4     for ii in prange(data.shape):
5         ...
6     return gradient(data)
```

Gradient Performance (*strong scaling*)

Constant Grid Resolution while Thread Counts Increase
(reduced workload per thread)

MPAS Dyamond (3.75 km)

Strong Scaling: Speedup vs Threads



fastest: **6.57 s** for
3.75 km with 256
threads

Dual-socket CPU node:
64 cores / socket
256 GB memory / node
2 threads / core

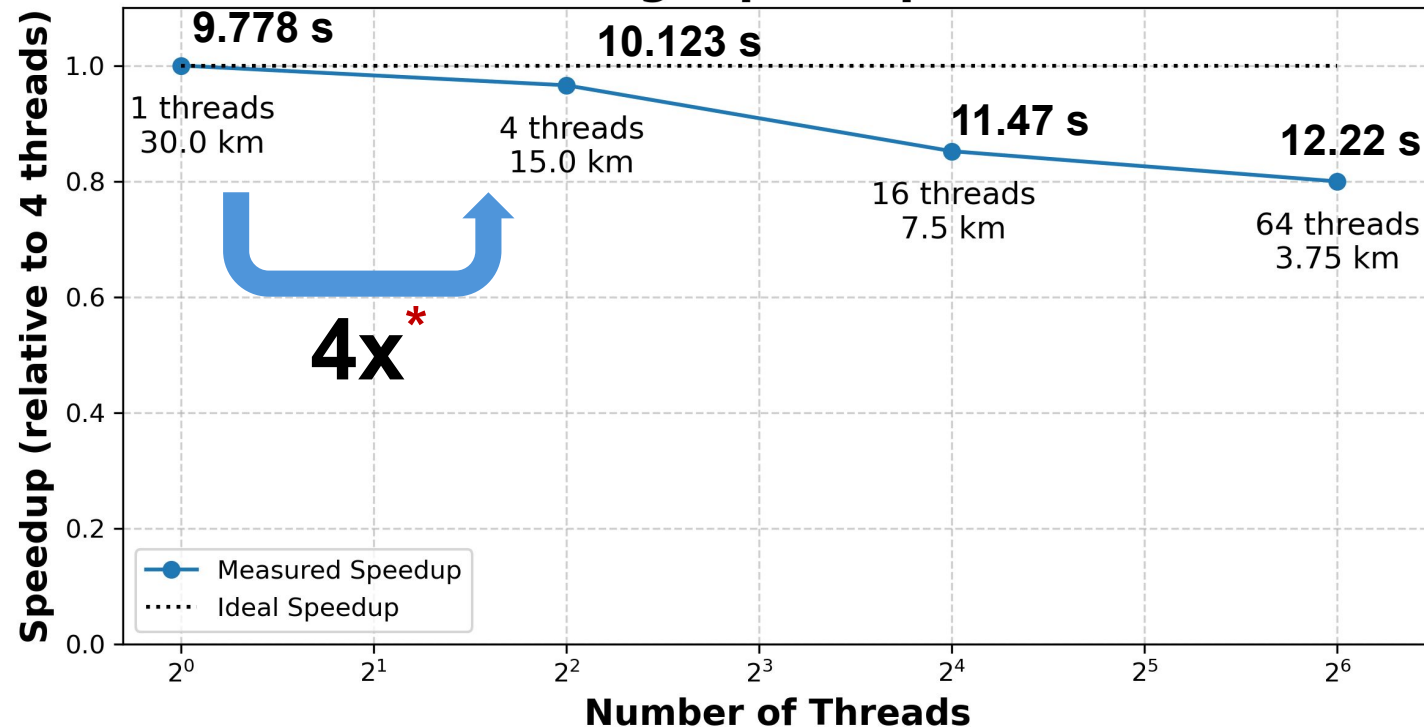
Gradient Performance (*weak scaling*)

Grid Resolution **Increase** & Thread Counts
Increase

(*constant* workload per thread)

MPAS Dyamond: 30 km, 15 km, 7.5 km, 3.75 km

Weak Scaling: Speedup vs Threads

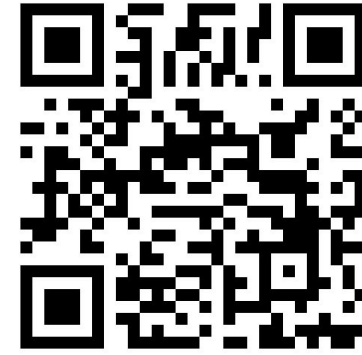


Dual-socket CPU node:
64 cores / socket
256 GB memory / node
2 threads / core

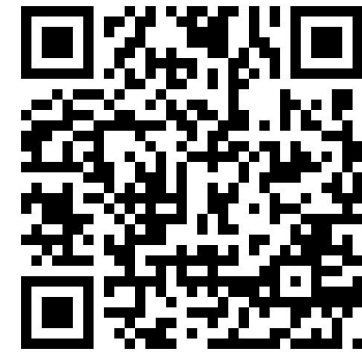
**Halving the resolution size (i.e., 30 km to 15 km) roughly quadruples the workload.*

Going forward

- Robust **implementations** for:
 - Curl, Divergence, Laplacian
- Detailed **documentation** for the user community
 - Add vector field visualizations
- **Feedback** from users



<https://uxarray.readthedocs.io>



<https://github.com/UXARRAY/uxarray>



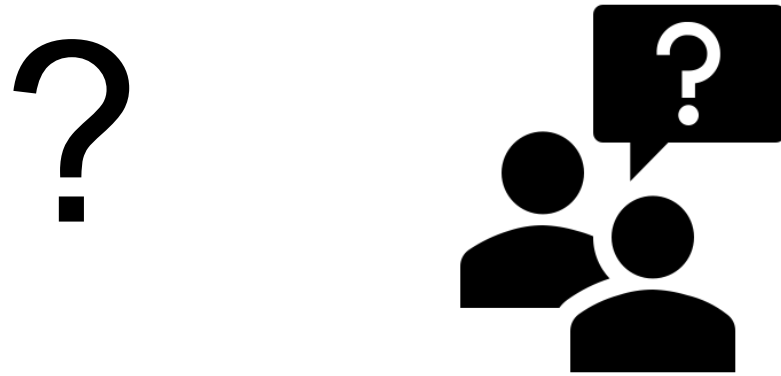
Acknowledgements

- Audience
- Philip Chmielowiec, Orhan Ergolu, Katelyn Fitzgerald, Rajeev Jain
- The Interns
- Virginia & Jessica
- Allison Baker & Peter Lauritzen
- And people I met at lunch!

References

- [1] Staniforth, Andrew, and John Thuburn. "Horizontal grids for global weather and climate prediction models: a review." *Quarterly Journal of the Royal Meteorological Society* 138.662 (2012): 1-26.
- [2] Syrakos, Alexandros, et al. "A critical analysis of some popular methods for the discretisation of the gradient operator in finite volume methods." *Physics of Fluids* 29.12 (2017).
- [3] Barth, Timothy, and Dennis Jespersen. "The design and application of upwind schemes on unstructured meshes." *27th Aerospace sciences meeting*. 1989.
- [4] Tomita, Hirofumi, et al. "Shallow water model on a modified icosahedral geodesic grid by using spring dynamics." *Journal of Computational Physics* 174.2 (2001): 579-613.
- [5] Kritsikis, Evaggelos, et al. "Conservative interpolation between general spherical meshes." *Geoscientific Model Development* 10.1 (2017): 425-431.

Questions

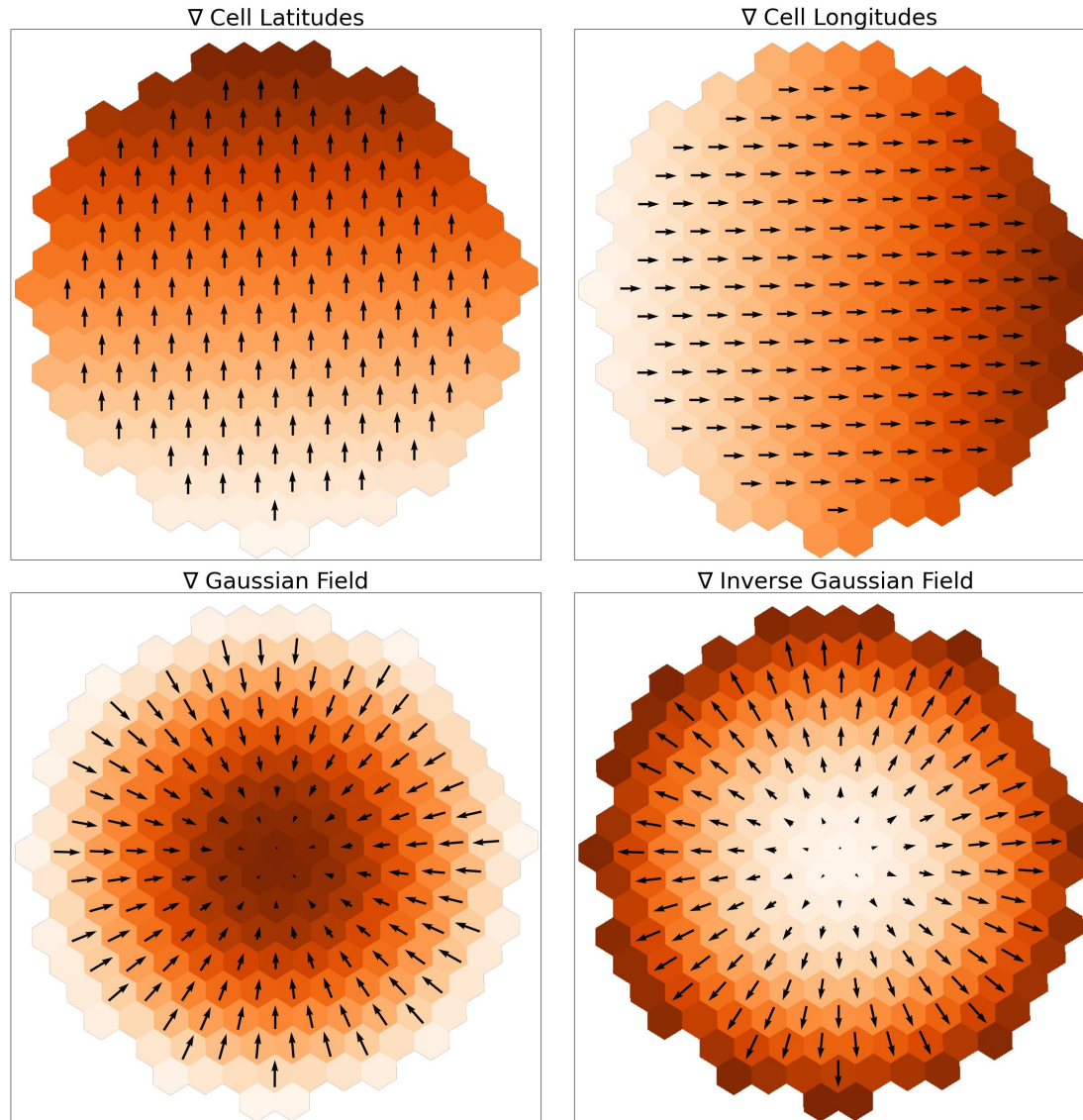


eg542@cornell.edu

GitHub: [egallmeier](#)

“This material is based upon work supported by the U.S. National Science Foundation National Center for Atmospheric Research, which is a major facility sponsored by the U.S. National Science Foundation under Cooperative Agreement No. 1755088. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. National Science Foundation.”

Gradient Visualization (*more!*)



MPAS Mesh Info

MPAS Atmosphere Mesh	30 km	15 km	7.5 km	3.75 km
# of Faces	655,362	2,621,442	10,485,762	41,943,042
# of Edges	1,966,080	7,864,320	31,457,280	125,829,120
# of Nodes	1,310,720	5,242,880	20,971,520	83,886,080
Grid Storage	~0.38 GB	~1.50 GB	~4.61 GB	~18.45 GB

Halve the resolution size
=> **quadruples** the grid

4x

*Grids can be
big (& the data
too)*