uxarray

Streamlining the Analysis of CESM Model Output on Native Grids

Philip Chmielowiec[1], Orhan Eroglu[1], John Clyne[1], Brian Medeiros[2], Colin Zarzycki[3], Robert Jacob[4], Paul Ullrich[5], Rajeev Jain[4], Robert Jacob[4], Aaron Zedwick[4], Hongyu Chen[5], Cecile Hannay[2], Lantao Sun[6]

[1] NSF NCAR, CISL, [2] NSF NCAR, CGD, [3] The Pennsylvania State University, [4] Argonne National Laboratory, [5] UC Davis. [6] Colorado State University

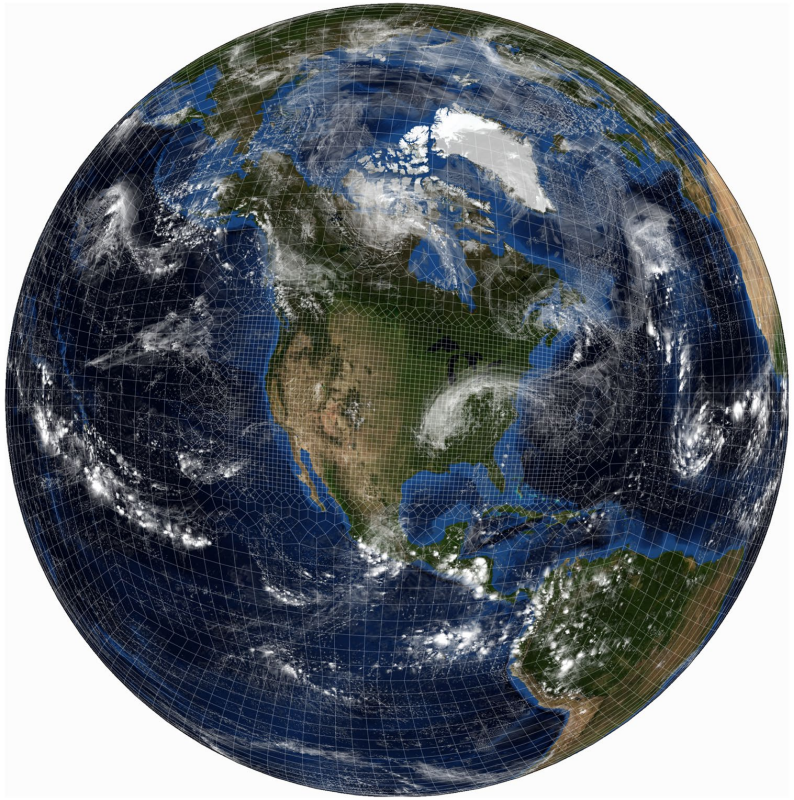
This material is based upon work supported by the National Science Foundation under Grants No.2126458 and 2126459

June 10, 2025

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsor ed by the National Science Foundation under Cooperative Agreement No. 1852977.

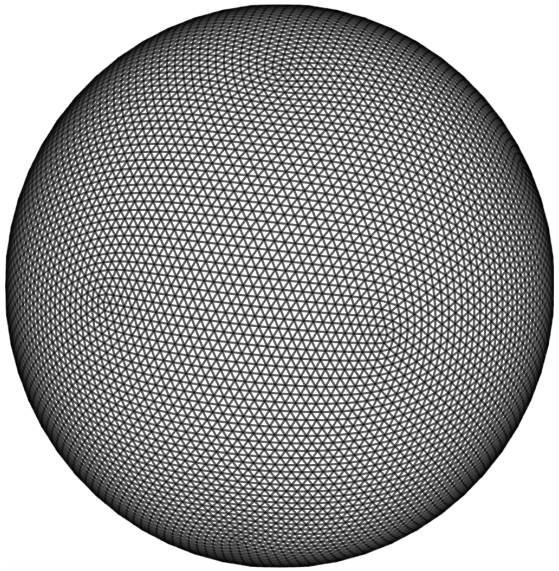
What is UXarray?

- An open-source Python package that provides data analysis and visualization functionality for unstructured geoscience data
  - pip install uxarray
  - conda install -c conda-forge uxarray
- Extends the extremely popular **Xarray** package to provide unstructured-grid aware functionality while maintaining similar workflows to structured grids
- Designed using the **Dask**, **Numba**, and **Datashader** packages to enable the analysis of kilometer-scale grids and **HoloViews** for interactive plotting

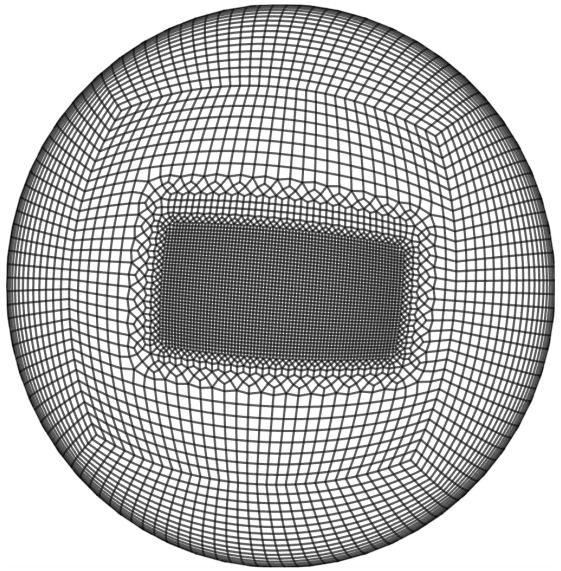


UGRID Conventions

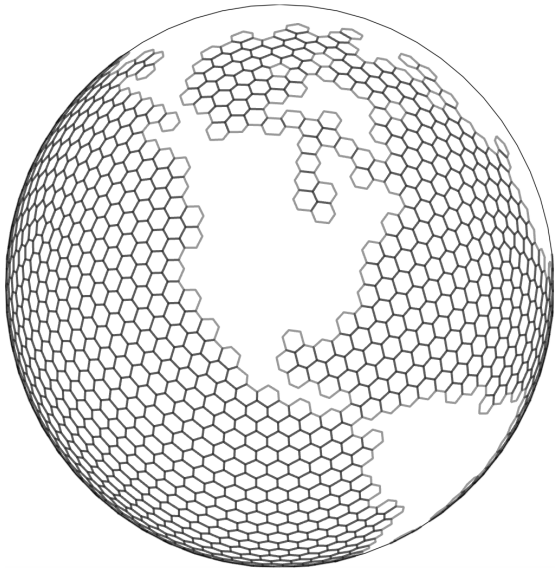
Grid A



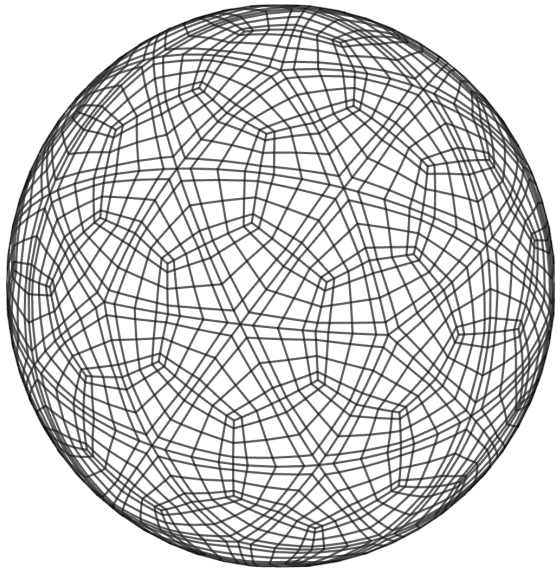
Grid B



Grid C



Grid D



- The UGRID conventions are a set of standardized definitions for representing unstructured grids
- UXarray only cares about the original grid format (i.e. MPAS, ICON, etc.) when initially loading the grid
- Each supported grid format is converted to the UGRID conventions for internal representation
  - This means that functionality can be written for a single grid format, instead of having separate functionality for each different grid format

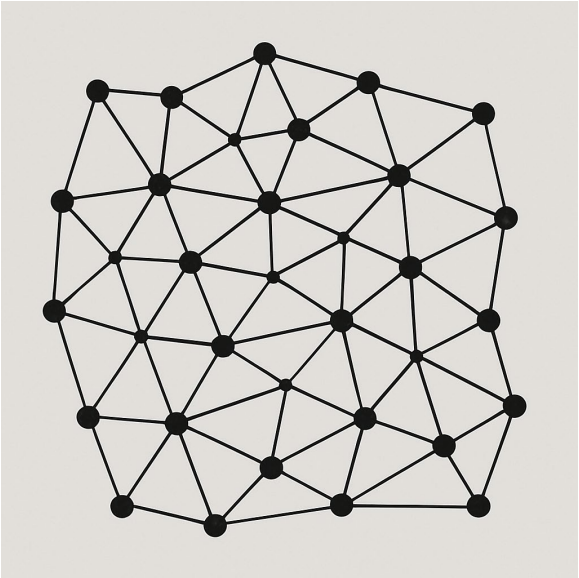
UGRID Conventions

Supported Files Formats & Model Outputs

- Unstructured grid file formats: UGRID, SCRIP, EXODUS
- Unstructured grid model outputs: MPAS, CAM-SE, ICON, ESMF, GEOS, HEALPix
- … also: Lat-Lon, Points

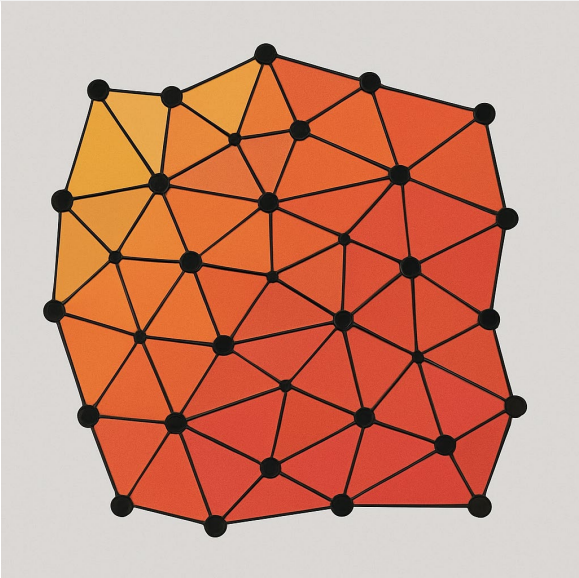
By representing all grid formats in the UGRID conventions, notebooks or scripts can be written for one format and be directly applied to others, reducing the need to duplicate code

Design Overview



ux.Grid()

Represents an arbitrary unstructured grid, storing coordinates, connectivity, and other information

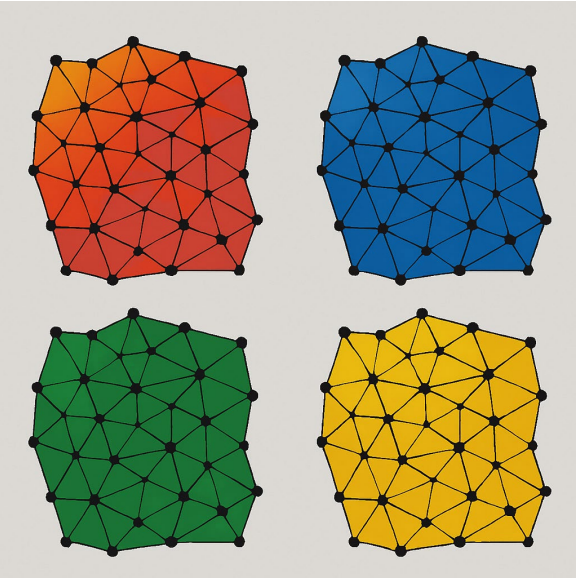


ux.UxDataArray()

Represents a single data variable that resides on an unstructured grid

Linked to a **ux.Grid()**

Inherits from **xarray.DataArray()**



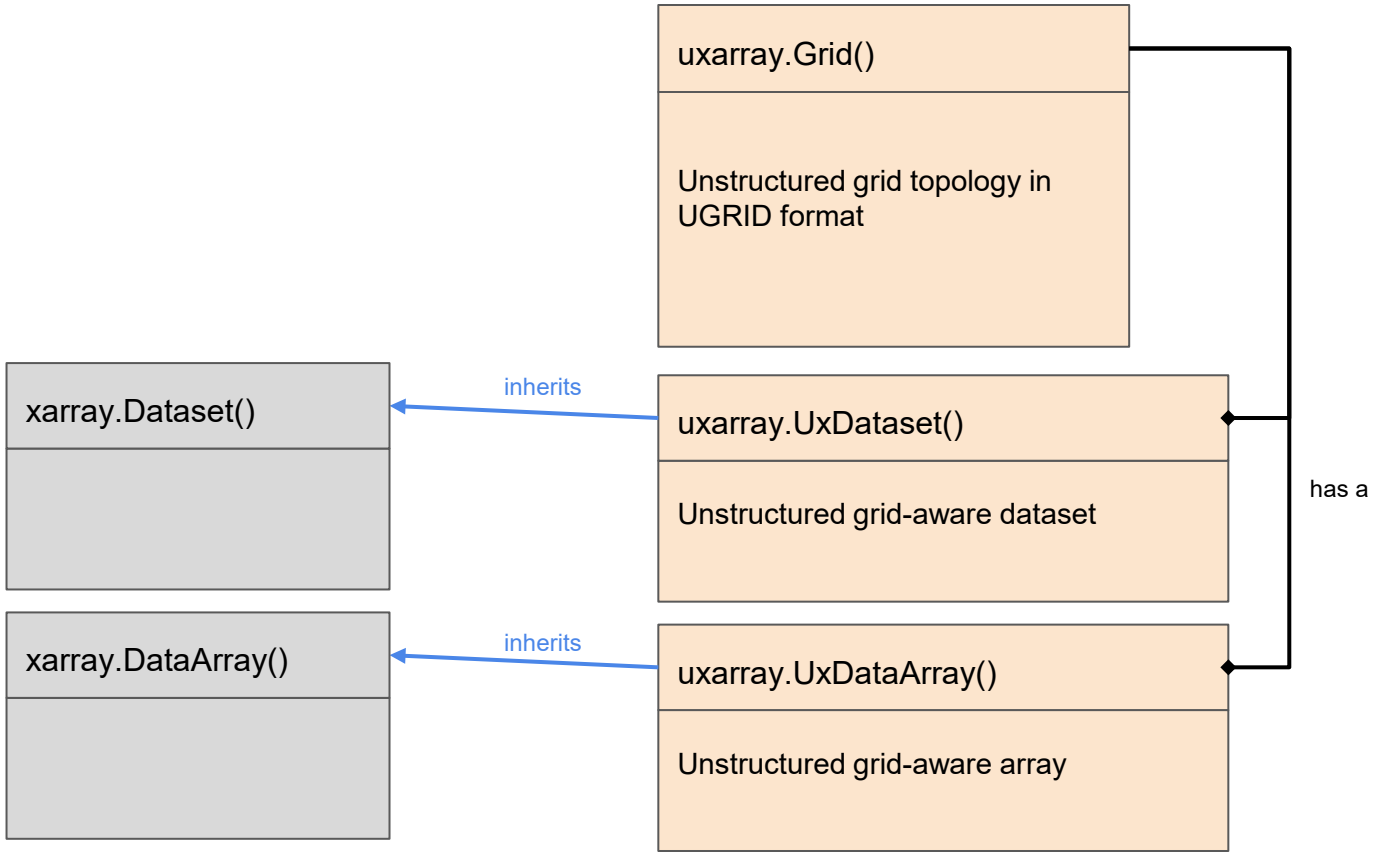
ux.UxDataset()

Represents a dataset of variables that reside on an unstructured grid

Linked to a **ux.Grid()**

Inherits from **xarray.Dataset()**

Grid-Informed Array and Dataset



uxarray.UxDataset

- Dimensions: (n_face: 4)
- Coordinates: (0)
- Data variables:

t2m	(n_face)	float32	...

- Indexes: (0)
- Attributes: (0)

▼ Show Grid Information

uxarray.UxDataset.uxgrid

- Dimensions: (n_node: 16, n_face: 4, n_edge: 19, n_max_face_nodes: 6)
- Spherical Coordinates:

node_lon	(n_node)	float32	-0.03841 -0.1864 ... 0.2352 0.2439
node_lat	(n_node)	float32	...
edge_lon	(n_edge)	float32	-0.1149 0.02647 ... 0.03119 0.1723
edge_lat	(n_edge)	float32	...
face_lon	(n_face)	float32	-0.04297 0.1006 -0.1818 0.2396
face_lat	(n_face)	float32	...

- Cartesian Coordinates: (0)
- Connectivity:

face_node_co...	(n_face, n_max_face_nodes)	int64	0 1 2 3 4 5 15 ... 4 14 11 10 15 5

- Descriptors:

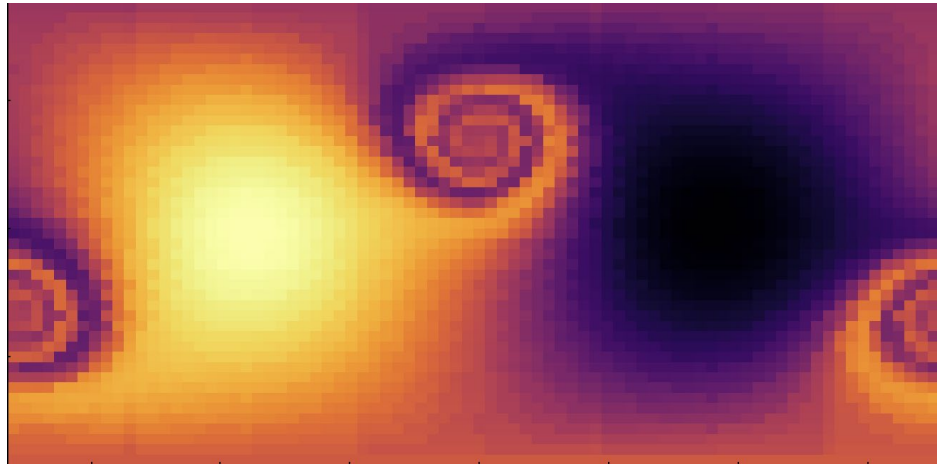
n_nodes_per_f...	(n_face)	int64	...

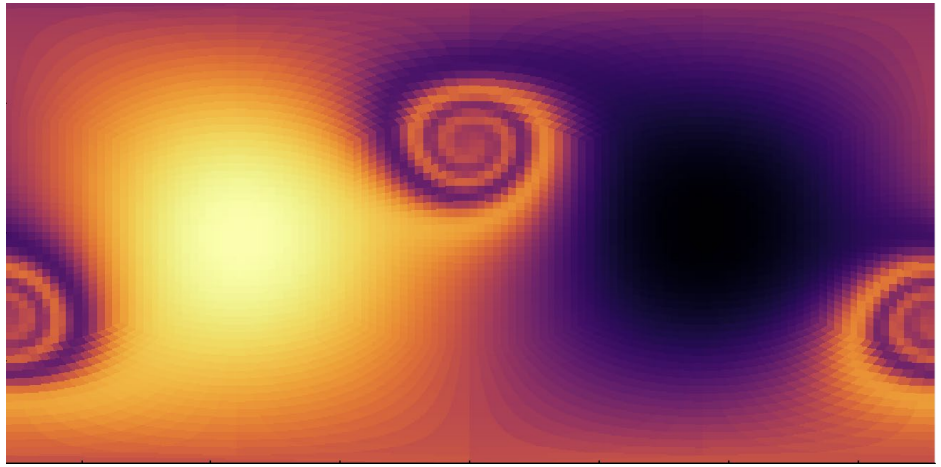
- Attributes: (55)

Basic Example - Xarray vs UXarray

```python
import xarray as xr

data_path = "/path/to/data.nc"

ds = xr.open_dataset(data_path)

ds.PSI.plot()
```

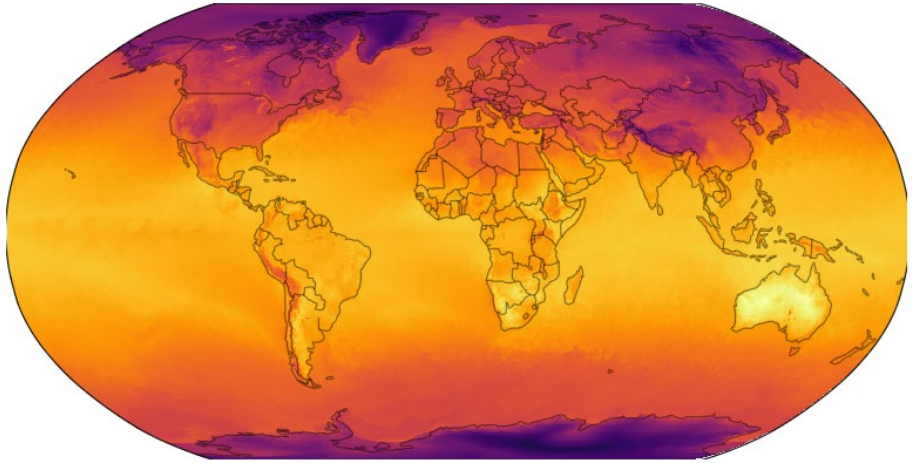
```python
import uxarray as ux

grid_path = "/path/to/grid.nc"
data_path = "/path/to/data.nc"

uxds = ux.open_dataset(grid_path, data_path)

uxds.PSI.plot()
```



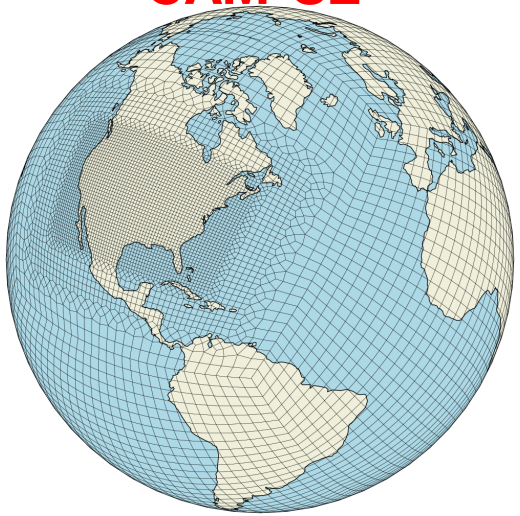


Visualization

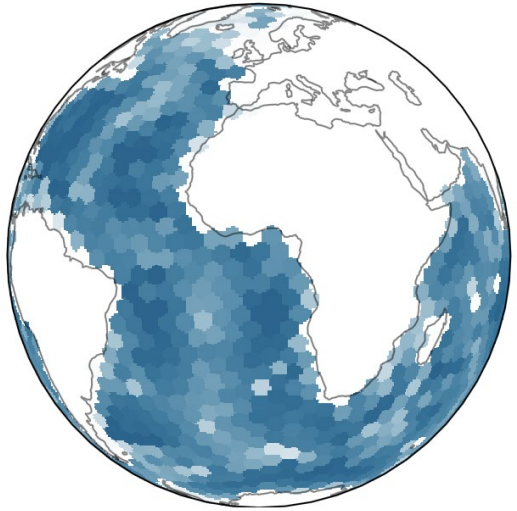
- UXarray provides an extensive visualization toolkit, including both **vector** and **raster** based grid topology and data visualization functions
- Written around **HoloViews** and **Datashader** , with support for native **Matplotlib** workflows currently in development



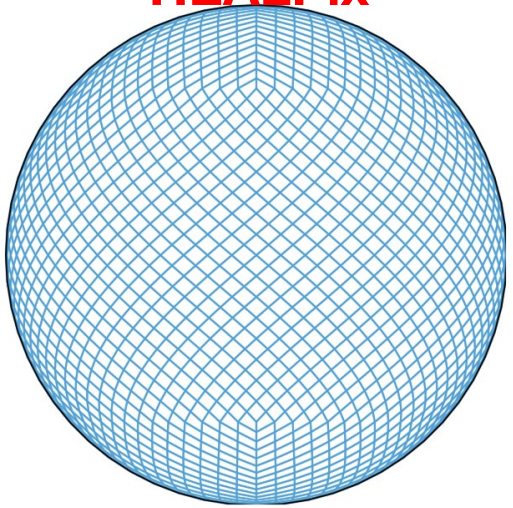
CAM-SE

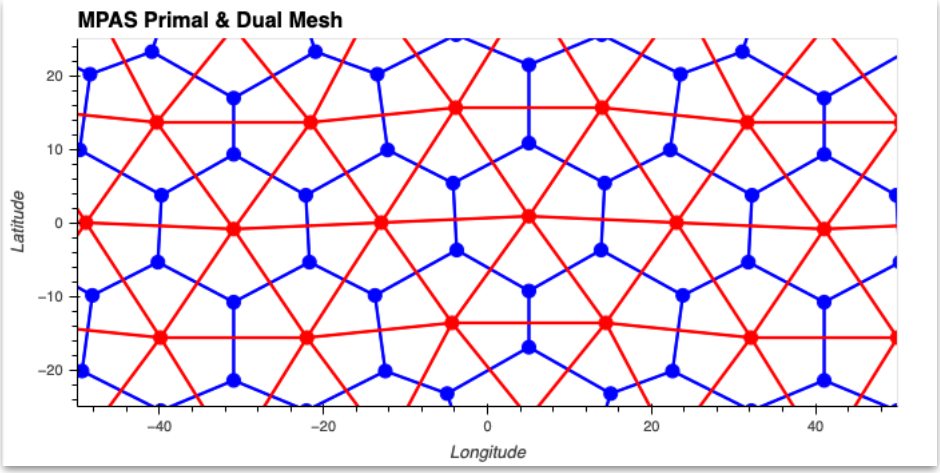


MPAS

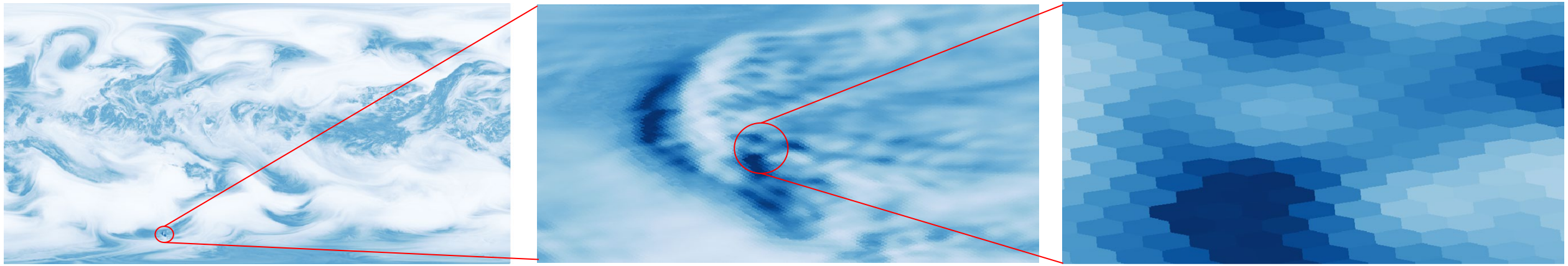


HEALPix





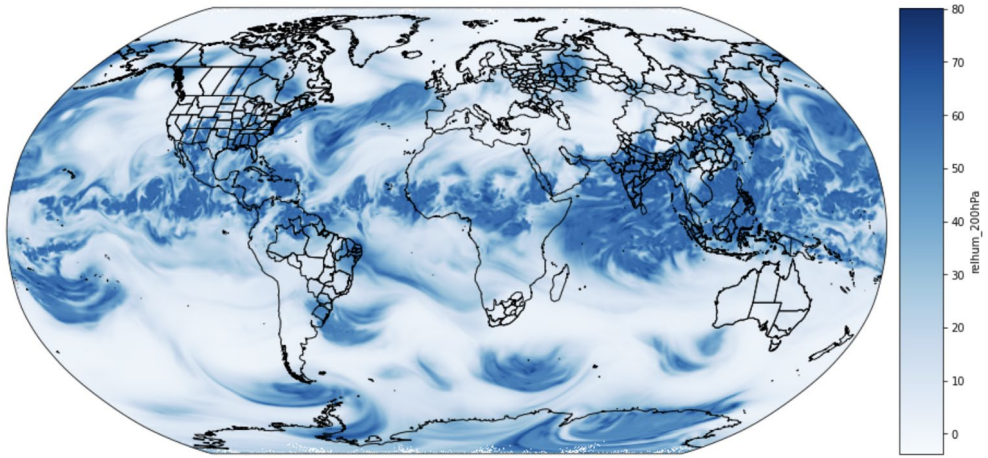
Data Visualization Example - 3.75km MPAS Model Output



Global Plot
~42 million individual cellls

Zoomed
Begin to observe the underlying grid topology (hexagons)

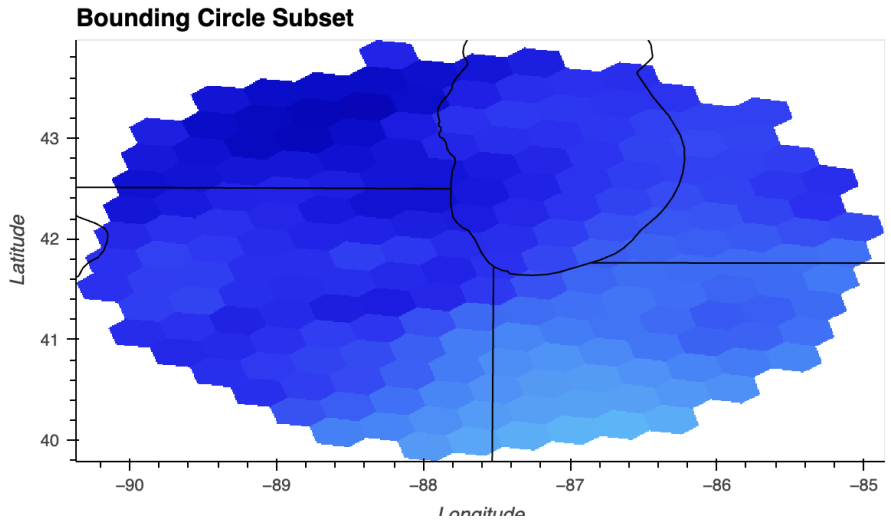
Further Zoomed Raster
After zooming far enough, the original hexagon grid becomes extremely clear

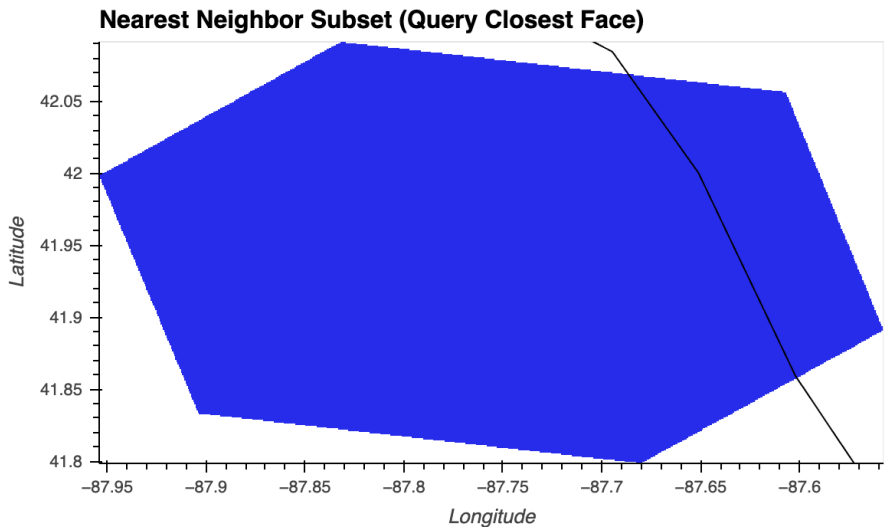


Subsetting

- Subsetting functionality can be used to restrict the domain of a grid to an area of interest
  - Bounding Circle
  - Bounding Box
  - Nearest Neighbor

```
uxda = ux.UxDataArray(...)

# Bounding Circle
sub_bc = uxda.subset.bounding_circle((-87, 41), r=2)

# Bounding Box
sub_bb = uxda.subset.bounding_box((-89, -85), (39, 43))

# Nearest Neighbor
sub_nn = uxda.subset.nearest_neighbor((-87, 41), k=1)
```

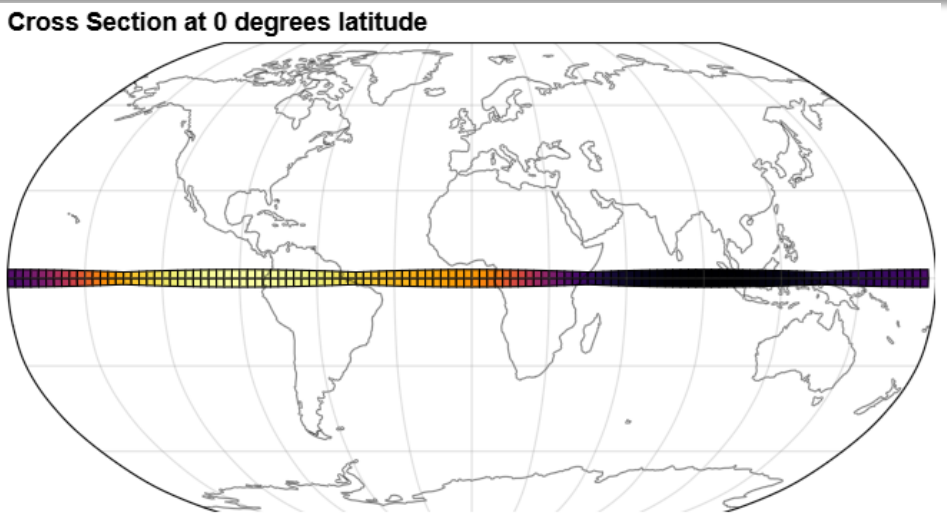


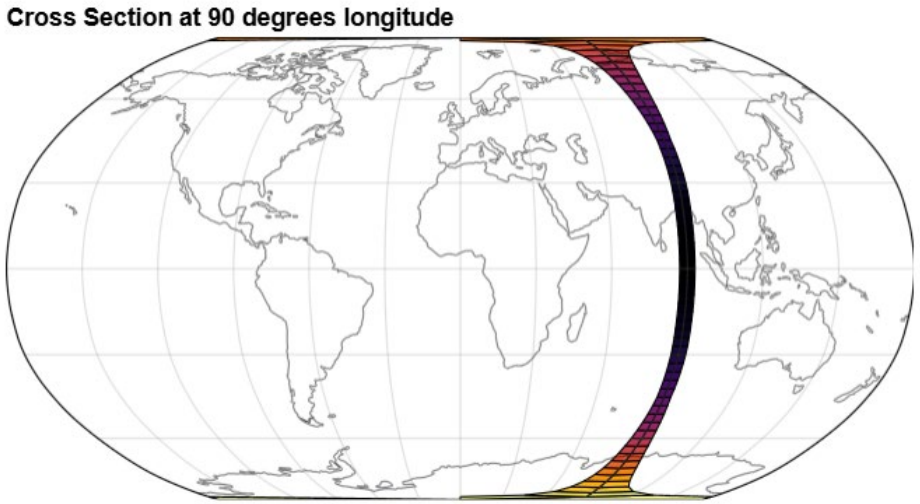


Cross-Sections

- Similar to subsetting, the cross-section functionality can be used to take slices of data along specified regions
  - Constant Latitude
  - Constant Longitude
  - Latitude Interval
  - Longitude Interval

```
uxda = ux.UxDataArray(...)

# Data intersecting a line of constant latitude
clat = uxda.cross_section.constant_latitude(0)

# Data intersecting a line of constant longitude
clon = uxda.cross_section.constant_longitude(45)

# Data between two lines of constant latitude
blat = uxda.cross_section.constant_latitude_interval(-10, 10)

# Data between two lines of constant longitude
blon = uxda.cross_section.constant_longitude_interval(-10, 10)
```

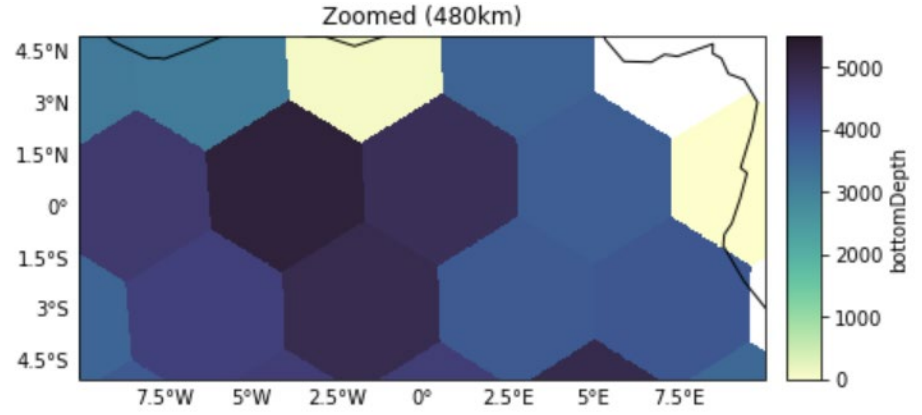


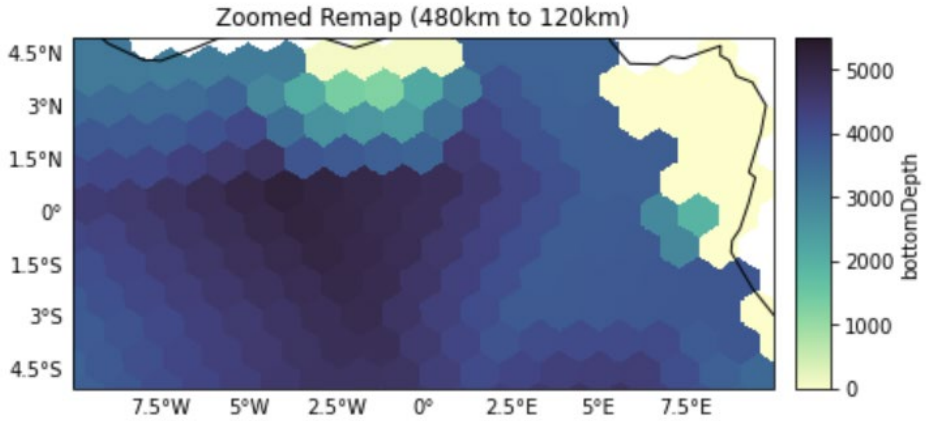


Remapping

- Lightweight remapping (regridding) functionality is provided to quickly convert data from one grid to another
  - Nearest Neighbor
  - Inverse Distance Weighted
  - Bilinear

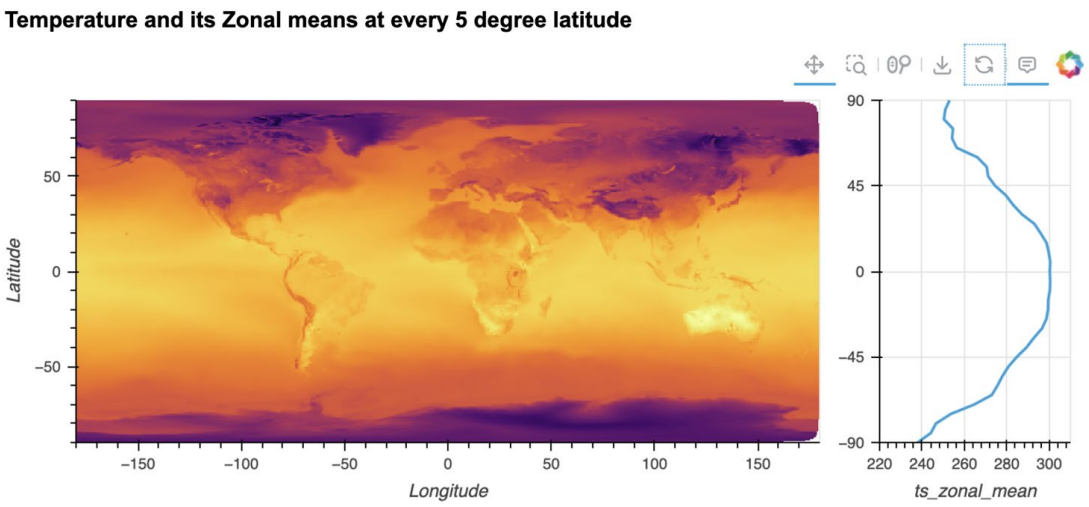
```
# Source Data and Destination Grid
source_uxda = ux.UxDataArray(...)
destination_grid = ux.Grid(...)

# Nearest Neighbor
uxda_nn = source_uxda.remap.nearest_neighbor(destination_grid)

# Inverse Distance Weighted
uxda_idw = source_uxda.remap.inverse_distance_weighted(destination_grid)

# Bilinear
uxda_bl = source_uxda.remap.bilinear(destination_grid)
```



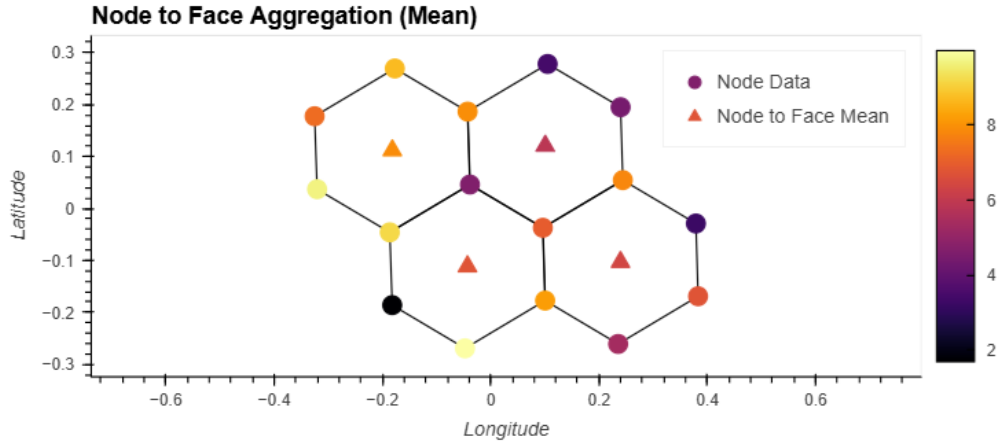


Analysis Operators

- UXarray supports many operations to enable the analysis of unstructured grid model output
  - Zonal Averaging
  - Unweighted and Weighted Averaging
  - Topological Aggregations
  - Point in Cell Containment



- Currently working with our SIParCS intern to develop operators for vector calculus operations
  - Gradient
  - Divergence
  - Curl



UXarray Resources - GitHub Repo and Homepage

README | Code of conduct | Apache-2.0 license

uxarray

Xarray extension for unstructured climate and global weather data

CI passing | CI Upstream passing | pre-commit.ci passed | codecov 86% | docs passing | benchmarked by asv
tag v2025.05.0 | conda-forge v2025.05.0 | pypi v2025.5.0 | license Apache-2.0 | DOI 10.5281/zenodo.15339820

UXarray aims to address the geoscience community's need for tools that enable standard data analysis techniques to operate directly on unstructured grid data. UXarray provides Xarray-styled functionality to better read in and use unstructured grid datasets that follow standard conventions, including UGRID, MPAS, ICON, SCRIP, ESMF, FESOM2, and Exodus grid formats. This effort is a result of the collaboration between Project Raijin (NSF NCAR and Pennsylvania State University) and the SEATS Project (Argonne National Laboratory, UC Davis, and Lawrence Livermore National Laboratory). The UXarray team welcomes community members to become part of this collaboration at any level of contribution.

https://github.com/UXARRAY/uxarray

uxarray

Search

For users

- Getting Started
- User Guide
- Gallery
- API Reference
- Tutorials and Videos
- Cite Uxarray

Community

- GitHub Discussions
- GitHub Issues
- UGRID Conventions

UXarray Documentation

Xarray extension for unstructured climate and global weather data analysis and visualization written around the UGRID conventions.

Getting Started
New to UXarray? Check out the getting started guides, which provide an introduction to UXarray's main motivation and functionality.

User Guide
The user guide provides in-depth information on the key concepts of UXarray with usefully definitions, background information and explanations.

Gallery
Collection of visualization and workflow examples showcasing UXarray in action.

API Reference
The reference guide contains a detailed description of the UXarray API. The reference describes how the methods work and which parameters can be used.

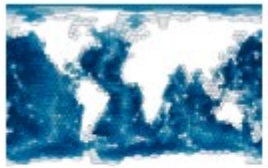
Installation
The installation guide provides step-by-step instructions for installing UXarray.

Contributor's Guide
Interested in contributing to UXarray? The contributors guide outlines the process of making contributions.

https://uxarray.readthedocs.io

UXarray Resources - Pythia Cookbooks

Advanced workflow demonstrations through Project Pythia



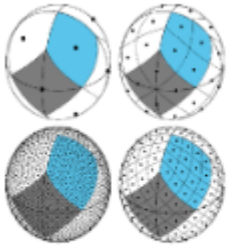
Unstructured Grids Visualization Cookbook

Author: Orhan Eroglu, Philip Chmielowiec, Rajeev Jain, Ian Franda, Unstructured Grids Visualization Cookbook contributors

This Cookbook is a comprehensive showcase of workflows

Cartopy | Datashader | Holoviews | Hvplot | Visualization

nightly-build failing | launch binder | DOI 10.5281/zenodo.10403389



HEALPix Cookbook

Author: Orhan Eroglu, Philip Chmielowiec, Andrew Gettelman, John Clyne, HEALPix Cookbook contributors

An introduction to HEALPix and utilization of easy.gems and UXarray to run data analysis and visualization functionality on HEALPix data sets.

Cartopy | Climate-Modeling | Easygems | Spatial-Analysis | Unstructured-Grids | Uxarray | Visualization

nightly-build passing | launch binder | DOI 10.5281/zenodo.15346818

PROJECT PYTHIA

projectpythia.org

Summary

UXarray is an extensible, scalable, open source Python package for analysis and visualization of unstructured grid data

Builds on top of a software ecosystem widely used by the Earth System Science community; i.e. extends the **Xarray** package, accelerated with **Numba** and **Dask**, documents with **Jupyter** , etc.

uxarray

https://uxarray.readthedocs.io