XDGGS: A community-developed Xarray package to support planetary DGGS data cube computations

Alexander Kmoch[1], Benoit Bovy[2], Justus Magin[3], Ryan Abernathey[4,5],
Alejandro Coca-Castro[6], Peter Strobl[7], Anne Fouilloux[8], Daniel Loos[9], Evelyn Uuemaa[1],
Wai Tik Chan[1], Jean-Marc Delouis[3], Tina Odaka[3]

[1] University of Tartu, Institute of Ecology and Earth Sciences, Landscape Geoinformatics Lab,
Tartu, Estonia - (alexander.kmoch, evelyn.uuemaa, wai.tik.chan)@ut.ee
[2] Georode, Liege, Belgium - benbovy@gmail.com
[3] LOPS - Laboratoire d'Oceanographie Physique et Spatiale UMR 6523 CNRS-IFREMER-IRD-Univ.Brest-IUEM,
Plouzane, France - (justus.magin, jean.marc.delouis, tina.odaka)@ifremer.fr,
[4] Earthmover PBC, New York, NY, USA – ryan@earthmover.io
[5] Lamont Doherty Earth Observatory of Columbia University, Palisades, NY, USA
[6] The Alan Turing Institute, London, UK - acoca@turing.ac.uk
[7] European Commission Joint Research Centre, Ispra, Italy - peter.strobl@ec.europa.eu
[8] Simula, Oslo, Norway - annef@simula.no
[9] Max Planck Institute for Biogeochemistry, Jena, Germany - dloos@bgc-jena.mpg.de

Keywords: Discrete Global Grid Systems, Xarray, Geospatial Data Cube, Earth Observation, open-source collaboration

Abstract

Traditional map projections introduce distortions, especially for global data. Discrete Global Grid Systems (DGGS) offer an alternative by dividing the Earth into equal-area grid cells at different resolutions. This paper describes xdggs, a new Xarray extension that simplifies working with DGGS. Xdggs provides a unified API for various DGGS libraries and integrates seamlessly with the Pangeo ecosystem through extending the widely used Xarray library to use the DGGS-specific cell identifiers as an index. This development makes DGGS more accessible and will lead to facilitating data analysis on a planetary scale.
Xdggs aims to provide a user-friendly API that hides the implementation complexities of different DGGS libraries. And because it integrates seamlessly with Xarray, a popular tool for geospatial data analysis, xdggs promotes FAIR data practices by simplifying data access and interoperability and can become a valuable tool for geospatial scientists and application developers working with global datasets.

1. Introduction

Traditional maps use projections to represent geospatial data in a 2-dimensional plane. This is both convenient and computationally efficient. However, it also introduces distortions in terms of area, distances, and angles, especially for global data sets (de Sousa, Luis M. and Poggio, Laura and Kempen, Bas, 2019). Several global grid system approaches like Equi7Grid or UTM aim to reduce the distortions by dividing the surface of the earth into many zones and using an optimized projection for each zone to minimize distortions. However, it introduces analysis discontinuities at the zone boundaries and makes it difficult to combine data sets of varying overlapping extents (Bauer-Marschallinger et al., 2014, Bauer-Marschallinger and Falkner, 2023). Discrete Global Grid Systems (DGGS) provide a new approach by introducing a hierarchy of global grids that tesselate the Earth's surface evenly into grid cells of similar size and shape around the globe at different spatial resolutions, and providing a unique indexing system (Sahr et al., 2004). DGGS are now also defined in the joint ISO and OGC DGGS Abstract Specification Topic 21 (ISO 19170-1:2021). DGGS can serve as spatial reference systems facilitating data cube construction, enabling integration and aggregation of multi-resolution data sources. Various cell geometries such as hexagons, quadrangles, and triangles and different tessellation strategies cater to different needs - equal area, optimal neighborhoods, congruent parent-child relationships, ease of use, or vector field representation in modeling flows. Purss et al. (2019) have explained the idea of combining DGGS and data cubes and underlined the compatibility of these two concepts. Thus, DGGS are a promising way to harmonize, store, and analyze spatial data on a planetary scale (Purss et al., 2019). DGGSs are traditionally often used in tabular format, where each cell is represented in a row and the cell id is a column serving as spatial identifier. In addition, usually datasets also have other parameters which can be considered as dimensions, such as time, altitude, ensemble member, etc. For these, Xarray (Hoyer and Hamman, 2017), one of the core packages in the Pangeo ecosystem, is an ideal container for multidimensional DGGS data. At the joint OSGeo and Pangeo codesprint at the ESA BiDS'23 conference (6.-9. November, 2023, Vienna), members from both communities came together and embarked on implementing support for DGGS in the Xarray Python package, which is at the core of many geospatial big data processing workflows. The result of the codesprint is a prototype Xarray extension named xdggs (https://github.com/xarray-contrib/xdggs), which we describe in this article.

2. Data and methods

2.1 DGGS and open-source software

There are several open-source libraries that make it possible to work with DGGS:

1. Uber H3: https://h3geo.org/
2. HEALPIX: https://healpy.readthedocs.io/
3. rHEALPix: https://github.com/manaakiwhenua/rhealpixdggs-py
4. DGGRID: https://github.com/sahrk/DGGRID
5. Google S2Geometry: https://s2geometry.io/
6. OpenEAGGR: https://github.com/riskaware-ltd/open-eaggr/

For these open-source libraries Python bindings are available. However, they come with their very own not easy-to-use APIs, different assumptions, and functionalities (Kmoch et al., 2022a). This makes it difficult for users to explore the wider possibilities that DGGS can offer and compare different DGGS for the same workflow.

The aim of xdggs is to provide a unified, high-level, and user-friendly API that simplifies working with various DGGS types and their respective backend libraries, seamlessly integrating with Xarray and the Pangeo open-source geospatial computing ecosystem. Executable notebooks demonstrating the use of the xdggs package are also being developed to showcase its capabilities.

Except for the Hierarchical Equal Area isoLatitude Pixelization (HEALPix), the above-mentioned DGGS open-source implementations have been evaluated with a focus on area and shape distortions (Kmoch et al., 2022b). The equal-area aspect is becoming an increasingly important criterion to consider when designing large-scale or even global geospatial data cubes and digital twins. Destination Earth (DestinE) is a European flagship initiative aiming to create a highly accurate digital Earth twin to model and simulate natural and human activities for climate adaptation and disaster mitigation (Hoffmann et al., 2023). A key component is the Climate Change Adaptation Digital Twin, which integrates data from various models and multi-decadal climate projections using the HEALPix grid system (Górski et al., 2005). Originally designed for cosmological applications, the HEALPix is an equal-area DGGS that offers versatile properties for Earth science modeling. In order to complement the statistics with an overview of HEALPix, we also reproduce Kmoch et al. (2022) original study for HEALPix in this article.

2.2 XDGGS design and roadmap

The xdggs community contributors set out with a set of guidelines and common DGGS features that xdggs should provide or facilitate, to make DGGS semantics and operations accessible to use via the user-friendly Xarray API of working with labelled arrays:

- import/export between DGGS and traditional 2D geospatial formats (e.g., rasters, latitude/longitude or UTM rectilinear grids, triangular irregular networks (TINs), vector gemoetries, or unstructured meshes and pointclouds)
- convert between different cell id representations of same type DGGSs (e.g., uint64 vs. string)
- select data on a DGGS by cell ids or by geometries (spatial indexing)
- aggregate/disaggregate data between different resolution levels of a DGGS, down and upsampling, respectively.
- operations between similar DGGS (with auto-alignment) re-organize cell ids (e.g., spatial shuffling/partitioning)
- and plotting.

To ensure seamless integration within Xarray's framework, xdggs exploits the Xarray extension mechanisms, like accessors, to link DGGS-specific functionalities. Efficiency needs to be considered by utilizing existing Python libraries with vectorized DGGS functions to work with large datasets.

While adhering to established DGGS standards, xdggs acknowledges practical considerations. Deviations from the standards might be necessary to ensure smooth integration with popular DGGS libraries. Scalability is a priority for both high-resolution DGGS (billions of cells) and diverse applications across GIS and Earth-System communities. Vertical scaling can be achieved through optimized DGGS implementations in backend libraries, while horizontal scaling will leverage Xarray's interoperability with Dask. Some operations may require focusing on vertical optimization first before exploring horizontal solutions.

The design explicitly also considered 'non-goals', implementation details that should not be included into xdggs directly, but be complemented through other libraries. xdggs relies on Xarray's extensive capabilities for data manipulation, especially regarding temporal coordinates as an orthogonal data dimension. Second, xdggs integrates with existing Python libraries that implement various DGGS. This eliminates the need for xdggs to replicate core functionalities. Similarly, xdggs concentrates on providing only essential DGGS operations and common resampling methods. More specialized functionalities, especially around re-gridding, should be implemented by combining core operations with complementary tools and libraries.

3. Results

3.1 XDGGS implementation details and examples

The goal of the joint code sprint was to implement these into an extension for the Xarray package. The library and the examples are publicly available at: https://github.com/xarray-contrib/xdggs

Xdggs represents a DGGS as an Xarray Dataset or DataArray with a 1-dimensional coordinate using the DGGS cell ids as labels. This coordinate is indexed using a custom, Xarray-compatible DGGSIndex, which needs to be instantiated with customizable DGGS-specific parameters like grid name, resolution, and additional attributes but does currently not support cell ids of mixed-resolutions in the same coordinate axis. xdggs also only supports one DGGS for a Dataset or DataArray but can index with multiple coordinates, like time, together with one DGGSIndex.

xdggs.DGGSIndex is the base class for all Xarray DGGS-aware indexes. It inherits from xarray.indexes.Index and uses an xarray.indexes.PandasIndex built from cell ids so that selection and alignment by cell id is possible. For each DGGS-type that xdggs shall support, a subclass of DGGSIndex is being implemented. Currently, the following DGGSs are usable but still considered experimental:

- HealpixIndex, for Healpix, uses the Healpy library
- H3Index for H3, uses the h3ronpy library
- ISEAIndex, uses the dggrid4py library

A DGGSIndex can be set directly from a cell ids coordinate using the Xarray API:

```
import xarray as xr
import xdggs

ds = xr.Dataset(
    coords={"cell": ("cell",
           [...],
           {"grid_name": "h3", "resolution": 3})}
)

# auto-detect grid system and parameters
ds.set_xindex("cell", xdggs.DGGSIndex)

# set the grid system and parameters manually
ds.set_xindex("cell", xdggs.H3Index, resolution=3)
```

The DGGSIndex can be set automatically when converting a gridded or vector dataset to a DGGS dataset. DGGS data creation involves multiple methods, including re-gridding from a latitude/longitude rectilinear grid, re-gridding from an unstructured grid, re-gridding and reprojecting from a raster in a local projection, aggregating from vector point data, and filling from polygon data. Conversely, DGGS data can be converted into various forms, such as re-gridding onto a latitude/longitude rectilinear grid, rasterizing through resampling or projection, converting to vector point data representing cell centroids, and converting to vector polygon data delineating cell boundaries.

```
# select by cell ids
ds.sel(cell_ids=[11320973, 11320975])
>>> xarray.Dataset ...

# select by geographic coordinates
# (DGGS library converts and queries for
# corresponding cell ids
ds.dggs.sel_latlon([48.0, 48.1], -5.0)
>>> xarray.Dataset ...

# recreate geometries from cell ids if needed
cell_boundaries = ds.dggs.cell_boundaries
>>> [POLYGON(... ) ...]
```

In the API, the ”dggs” accessor name functions as a prefix. Most DGGS-specific methods are invoked from an existing xarray Dataset or DataArray and typically return another Dataset or DataArray. Xarray natively supports regular grids. For raster data, methods could return objects compatible with rioxarray. For vector data, methods could return objects compatible with xvec, such as coordinates of shapely objects.

This development represents a significant step forward. With xdggs, DGGS become more accessible and actionable for data users. Like traditional cartographic projections, a user does not need to be an expert on the peculiarities of various grids and libraries to work with DGGS and can continue working in the well-known Xarray workflow. One of the aims of xdggs is making DGGS data access and conversion user-friendly, while dealing with the coordinates, tesselations, and indexing under the hood. Figure 1 illustrates the conversion of an orginally rectangularly latitude-longitude gridded dataset into hexagonal DGGS cells draped over the globe, using the Xarray tutorial weather data.

DGGS-indexed data can be stored in an appropriate format like Zarr or (Geo)Parquet, with corresponding metadata to identify

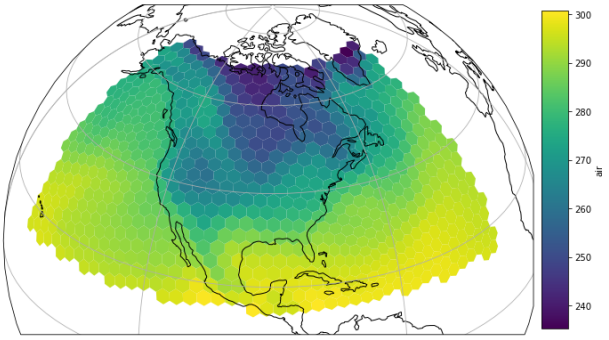

Figure 1. The classic air temperature data set from the toy weather data tutorial in Xarray, converted and plotted via a hexagonal DGGS.

which DGGS (and potentially under which specific configuration) is needed to address the grid cell indices correctly. An open-source interactive tutorial i.e. Jupyter notebooks on Pangeo-Forge, is being developed to demonstrate to users how effectively utilizing these storage formats, thereby facilitating knowledge transfer of the best practices in data storage within the geospatial open-source community.

3.2 HEALPix DGGS equal-area assessment

The HEALPix grid was originally developed to process, analyze, and create discretized spherical maps from very large volumes of astronomical data from cosmic microwave background experiments (Górski et al., 2005). It is increasingly considered in the Earth Sciences, e.g. by the World Meteorological Organization (WMO) as an indexing scheme for GRIB2 data files, or for the ICON climate model. This could signify a critical advancement for data representation in the climate sciences, where Xarray is also widely used.

We calculated the area and shape distortion for the HEALPix grid and updated the code and data supplements for the open-source DGGS cell geometry comparisons. Figure 2 shows the strong equal-area properties of HEALPIx.

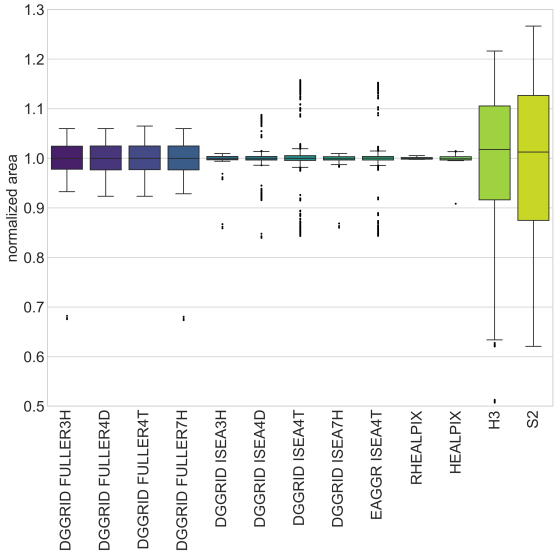

Figure 2. Updated summary boxplots of normalized area values for the cells of open-source DGGSs., including HEALPix.

Figure 3 illustrates the boxplots for compactness values for the compared open-source DGGSs, including HEALPix. Furthermore, we provide the corresponding histograms for the normalized area and compactness visualizations (Figure 4, Figure 5).

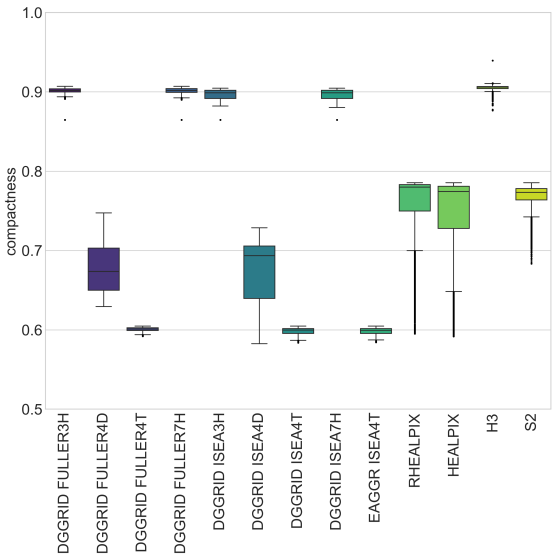

Figure 3. Updated summary boxplots of compactness values for the cells of open-source DGGSs, including HEALPix.

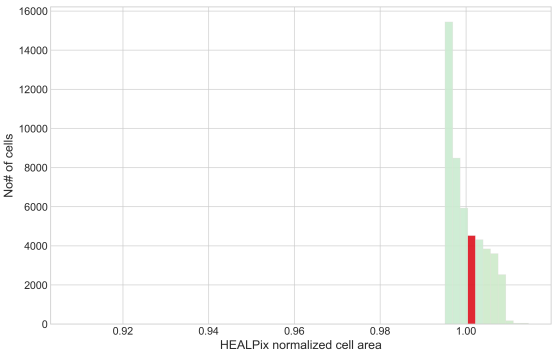

Figure 4. Histogram of normalized area values for HEALPix cells.

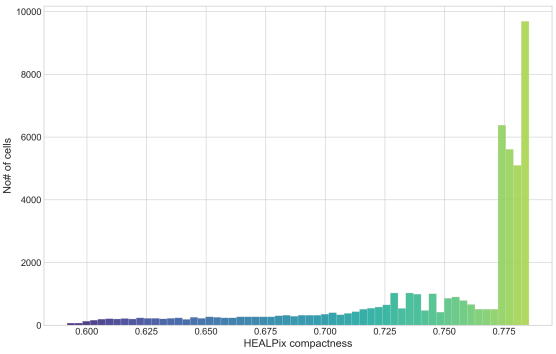

Figure 5. Histogram of compactness values for HEALPix cells.

4. Discussion and conclusion

4.1 DGGS as unified grid for EO data

Xdggs demonstrates that we can apply DGGS in place of the traditional rectangular coordinate systems typically used for geospatial raster and earth-system model data, side-by-side with other orthogonal dimensions such as time, vertical level, etc. This suggests that DGGS can potentially solve some of the core interoperability challenges that plague Earth System science—specifically, adopting standard DGGS offers a clear path forward for fusing datasets with different resolutions, such as Earth-observing satellites and global climate models (Liang et al., 2024). Rather than distributing EO datasets in the ubiquitous Military Grid Reference System with its discontinuous UTM-zone projections, data providers could potentially harmonize their raw data to a suitable DGGS instead (Salgues et al., 2023).

4.2 Plotting XDGGS data

Existing visualization and spatial analysis tools are primarily designed for rectangular pixels. Adapting these tools to work efficiently with the unique properties of DGGS (like variable resolutions and non-rectangular cells) requires significant development effort. This includes modifying the tools to interpret the hierarchical structure of DGGS and to perform spatial operations accurately.

Xdggs does currently not provide integrated visualization facilities. Several approaches, which are not mutually exclusive, can be used to plot Discrete Global Grid System (DGGS) data directly through the PyViz ecosystem (Signell and Pothina, 2019):

- Convert cell data into gridded or raster data, selecting the grid or raster resolution based on the resolution of the rendered figure, and then utilize existing Python plotting libraries such as matplotlib, cartopy, and holoviews via the xarray plotting API. For this approach, using datashader to set both the resolution and raster extent dynamically may be beneficial.
- Convert cell data into vector data and plot it using advanced geospatial Python libraries like geoviews, xvec, or geopandas. This approach may involve dynamically downgrading the DGGS resolution and aggregating the data before converting it to vector form to enable interactive plotting of large DGGS datasets. Figure 6 shows an orthographic plot of HEALPix at resolution 5 (ca 12.000 cells) with the GeoViews package.

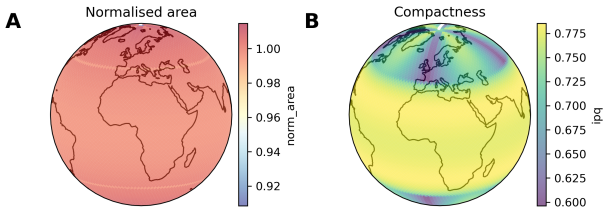

Figure 6. Globe view of normalized area and compactness values for HEALPix cells.

- Use libraries that support plotting DGGS data directly. For example, HEALPix provides direct plotting capabilities

from a cell id-based index to several global projections, including Mollweide and the Gnomonic projection. Another alternative are high-performance browser-based libraries, such as lonboard, which enables interactive plotting in Jupyter notebooks via deck.gl, which directly supports H3 and S2 cell data. This method is efficient for plotting large DGGS data as it only requires transferring cell IDs (tokens) and cell data, allowing deck.gl to render the cells efficiently in the web browser using the GPU.

4.3 Enabling DGGS hierarchies

DGGS are grid systems with grids of the same topology but varying spatial resolutions, maintaining a hierarchical relationship among grids of different resolutions. Currently, the coordinate of one grid in the DGGS of a Dataset or DataArray is restricted to cell ids of the same resolution. However, xdggs provides functionality to handle the hierarchical nature of DGGSs. This includes selecting by parent cell ids and utility methods to change the grid resolution explicitly. Additionally, the xarray datatree functionality, which is under development, might soon enable the inclusion of DGGS DataArrays for different resolutions in one active Dataset, enhancing the management of multi-resolution data. The underlying DGGS implementation supplies methods to navigate cell IDs across the DGGS hierarchy, facilitating advanced data operations and analysis.

4.4 Further directions

Nevertheless, continuous efforts are necessary to broaden the accessibility of DGGS for scientific and operational applications, especially in handling gridded data such as global climate and ocean modeling, satellite imagery, raster data, and maps. This would require, for example, an agreement ideally with entities such as the OGC for DGGS reference systems' registry (similar to the epsg/crs/proj database) in order to describe and identify unique DGGS types and their configurations. These are important metadata about the specifics of the used DGGS in form, e.g., an identifier, label, or link to a detailed definition. This would likely need to include the main type e.g. H3, S2, HEALPix, rHEALPix, ISEA3H, but also required parameters like for HEALPix (indexing: nested or ring), for rHEALPix (ellipsoid, orientation/rotation), icosahedron-based DGGS types (topology, refinement ratio, orientation, potentially mixed apertures, etc.). It would be good to have synergies here and not reinvent the wheel.

1. The implementation of DGGS on Xarray should be improved further with a more user-friendly API for gridding existing data into DGGS grids, similar to or as part of the GDAL suite.

2. DGGS indexed reference datasets could be validated and also used to highlight case studies and instructional material can be used in academic courses and workshops, focusing on the practical applications of data fusion, quick addressing of equal-area cell grids, artificial intelligence, infrastructure i.e. navigation systems of scientific polar ship navigation, AI, socio-economic and environmental studies. Especially the emerging property of selecting cell-ranges from different data sources to join and integrate only based on cell ids could make partial data access and sharing more dynamic and easy.

3. Continue to improve on the interoperability with STAC catalogs for satellite, modeled, and in-situ data - case studies and reference materials should include a workshop for practitioners to understand the integration and use of STAC and DGGS, emphasizing the importance of open-source tools and the open source software community.

4. Training materials and Pangeo community sessions should be conducted to demonstrate the use of DGGS in Xarray, aimed at enhancing the skillset of practitioners and researchers in geospatial data handling, spatial data analysis, and professional and academic institutions.

To address these challenges, the geospatial community needs to focus on developing open-source libraries that support DGGS, enhancing GIS software to handle non-rectangular grids, and creating efficient algorithms for data conversion and spatial analysis. There is also a need for extensive testing and validation to ensure that DGGS implementations can handle real-world datasets without significant loss of information, functionality, or excessive computing resources.

In conclusion, while DGGS offers a promising approach to unifying and managing earth observation data on a global scale, it also introduces a new set of challenges that require considerable innovation and cooperation among researchers, developers, and industry professionals to overcome. With xdggs we are making a significant step.

Acknowledgements

This work was funded by the Estonian Research Agency (grant number PRG1764, PSG841), Estonian Ministry of Education and Research, Centre of Excellence for Sustainable Land Use (TK232) and by the European Union (ERC, WaterSmartLand, 101125476, Interreg-BSR, HyTruck, #C031), and by CNES (PANGEO IAOCEA, contract R&T R-S23/DU-0002-025-01). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them. The authors are thankful for technical support from the High Performance Computing Center of the University of Tartu. The authors also acknowledge the constructive communication with the OGC DGGS domain and standards working groups.

References

Bauer-Marschallinger, B., Falkner, K., 2023. Wasting petabytes: A survey of the Sentinel-2 UTM tiling grid and its spatial overhead. *ISPRS Journal of Photogrammetry and Remote Sensing*, 202, 682–690. https://doi.org/10.1016/j.isprsjprs.2023.07.015.

Bauer-Marschallinger, B., Sabel, D., Wagner, W., 2014. Optimisation of global grids for high-resolution remote sensing data. *Computers and Geosciences*.

de Sousa, Luis M. and Poggio, Laura and Kempen, Bas, 2019. Comparison of FOSS4G Supported Equal-Area Projections Using Discrete Distortion Indicatrices. *ISPRS International Journal of Geo-Information*, 8(8), 351.

Górski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M., Bartelmann, M., 2005. HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere. *The Astrophysical Journal*, 622(2), 759. https://dx.doi.org/10.1086/427976.

Hoffmann, J., Bauer, P., Sandu, I., Wedi, N., Geenen, T., Thiemert, D., 2023. Destination Earth – A digital twin in support of climate services. *Climate Services*, 30, 100394. https://doi.org/10.1016/j.cliser.2023.100394.

Hoyer, S., Hamman, J., 2017. xarray: N-D labeled Arrays and Datasets in Python. *Journal of Open Research Software*, 5.

Kmoch, A., Matsibora, O., Vasilyev, I., Uuemaa, E., 2022a. Applied open-source Discrete Global Grid Systems. *AGILE: GIScience Series*, 3, 1–6. https://agile-giss.copernicus.org/articles/3/41/2022/.

Kmoch, A., Vasilyev, I., Virro, H., Uuemaa, E., 2022b. Area and Shape Distortions in Open-Source Discrete Global Grid Systems. *Big Earth Data*. https://doi.org/10.1080/20964471.2022.2094926.

Liang, Q., Zhou, J., Ben, J., Chen, Y., Huang, X., Ding, J., Dai, J., 2024. Precise hexagonal pixel modeling and an easy-sharing storage scheme for remote sensing images based on discrete global grid system. *International Journal of Digital Earth*, 17(1), 2328824. https://doi.org/10.1080/17538947.2024.2328824.

Purss, M. B. J., Peterson, P. R., Strobl, P., Dow, C., Sabeur, Z. A., Gibb, R. G., Ben, J., 2019. Datacubes: A Discrete Global Grid Systems Perspective. *Cartographica*, 54(1), 63–71.

Sahr, K., White, D., Kimerling, A. J., 2004. Geodesic Discrete Global Grid Systems. *Cartography and Geographic Information Science*.

Salgues, G., Cadau, E. G., Pessiot, L., Gaudissart, V., Enache, S., Gascon, F., Boccia, V., Strobl, P., 2023. A candidate dggs (discrete global grid system) for sentinel-2: First outcomes. *IGARSS 2023 - 2023 IEEE International Geoscience and Remote Sensing Symposium*, 4978–4981.

Signell, R. P., Pothina, D., 2019. Analysis and Visualization of Coastal Ocean Model Data in the Cloud. *Journal of Marine Science and Engineering*, 7(4). https://www.mdpi.com/2077-1312/7/4/110.

Appendix

The xdggs library and the examples are publicly available at: https://github.com/xarray-contrib/xdggs. The GitHub repository maintains the base implementation of xdggs and collates examples, and the issues are open to public participation.

Supplemental code for the area and compactness distortion assessment can be accessed online, openly available on Zenodo at https://zenodo.org/doi/10.5281/zenodo.5905935, version v4, and includes the updates for the HEALPix DGGS.

Supplemental data for the area and compactness distortion assessment can be accessed online, openly available on Zenodo at https://doi.org/10.5281/zenodo.11125478, version v4, and includes the updates for the HEALPix DGGS.