The Astronomical Journal, 163:259 (6pp), 2022 June https://doi.org/10.3847/1538-3881/ac6260

Multiresolution HEALPix Maps for Multiwavelength and Multimessenger Astronomy

I. Martinez-Castellanos[1,2,3], Leo P. Singer[1], E. Burns[4], D. Tak[5], Alyson Joens[1,6], Judith L. Racusin[1], and Jeremy S. Perkins[1]
[1] Astrophysics Science Division, NASA Goddard Space Flight Center, Greenbelt, MD 20771, USA; imc@umd.edu
[2] Department of Astronomy, University of Maryland, College Park, MD 20742, USA
[3] Center for Research and Exploration in Space Science and Technology, NASA/GSFC, Greenbelt, MD 20771, USA
[4] Department of Physics & Astronomy, Louisiana State University, Baton Rouge, LA 70803, USA
[5] Deutsches Elektronen-Synchrotron (DESY), Platanenallee 6, Zeuthen, D-15738, Germany
[6] George Washington University, 2121 I St NW, Washington, DC 20052, USA
Received 2021 November 22; revised 2022 March 28; accepted 2022 March 28; published 2022 May 11

Abstract

HEALPix—the Hierarchical Equal Area isoLatitude Pixelization—has become a standard in high-energy and gravitational wave astronomy. Originally developed to improve the efficiency of all-sky Fourier analyses, it is now also utilized to share sky localization information. When used for this purpose the need for a homogeneous all-sky grid represents a limitation that hinders a broader community adoption. This work presents `mhealpy`, a Python library able to create, handle and analyze multiresolution maps, a solution to this problem. It supports efficient pixel querying, arithmetic operations between maps, adaptive mesh refinement, plotting, and serialization into FITS—Flexible Image Transport System—files. This HEALPix extension makes it suitable to represent highly resolved region, resulting in a convenient common format to share spatial information for joint multiwavelength and multimessenger analyses.

Unified Astronomy Thesaurus concepts: Open source software (1866); Astronomy software (1855)

1. Introduction

HEALPix—the Hierarchical Equal Area isoLatitude Pixelization (Górski et al. 2005)[7]—is a scheme to pixelate the sphere, originally developed to facilitate the analysis of the cosmic microwave background (CMB) anisotropy. Its structural and geometrical properties allow for an efficient all-sky numerical Fourier decomposition and the convolution of local and global kernels. HEALPix has been implemented in multiple programming languages (C, C++, Fortran, IDL/GDL, Java, Python) and made available to the community since 1997.

HEALPix has grown beyond CMB analysis. Multiple astronomical fields have adopted it thanks to its convenient properties and accessible software libraries; for example, gamma-ray astronomy (Abdollahi et al. 2020; von Kienlin et al. 2020; Abeysekara et al. 2017), high-energy cosmic rays (Aab et al. 2014), neutrino astronomy (Aartsen et al. 2017), and gravitational waves (Singer & Price 2016). HEALPix is clearly established as a key tool for multimessenger astronomy.

Having the astronomical data from various sources in the same format is enormously useful to communicate findings among the community and to perform joint analyses. A clear example is to report the sky localization of a source, especially when the uncertainty region is not well-described by a simple geometrical shape—e.g., a circle or ellipse. Wide-spread adoption of a single format would not only make it easier to combine the localization information from various instruments, but it would also prompt collaborative efforts to develop generic code—rather than instrument specific—with the accompanying community support. Despite this, current limitations of the HEALPix standard and implementation have prevented a wider adoption.

The main HEALPix limitation is that the data structure holds a full-sky map, at the same resolution everywhere, even when only a small portion is relevant (see Section 2). While this was not a problem for its original use case, it severely restricts its use for multiwavelength studies. For example, instruments with a high-precision localization usually find the amount of computer resources needed impractical or even unfeasible.

The hierarchical structure of HEALPix provides a clear solution to this problem. Different levels of resolution can be realized for different regions by subdividing only certain pixels appropriately, as discussed in Section 3. This approach has been exploited either to generate a map efficiently, as a compression algorithm for storage or for faster querying (Fernique et al. 2015; Singer & Price 2016; Youngren & Petty 2017). It has also been utilized by the multiorder coverage (MOC) map International Virtual Observatory Alliance (IVOA) standard (Fernique et al. 2019) to specify arbitrary sky regions, which has been shown useful for the planning of multimessenger observations (Greco et al. 2022). The MOC standard was recently generalized by associating a value to each pixel of a full-coverage map, and is now used to distribute LIGO-Virgo gravitational wave sky localizations.[8]

Despite the fact that multiresolution HEALPix maps are already being used by some collaborations, they are yet to have a broader adoption. This can be attributed to the lack of tools to perform the tasks frequently needed in multiwavelength and multimessenger Astronomy. For example, the simple multiplication between two maps typically requires the rasterization of multiresolution maps into a common single-resolution grid, defeating their purpose.

[7] http://healpix.sourceforge.net

[8] LIGO/Virgo Public Alerts User Guide—https://emfollow.docs.ligo.org/userguide/tutorial/multiorder_skymaps.html.

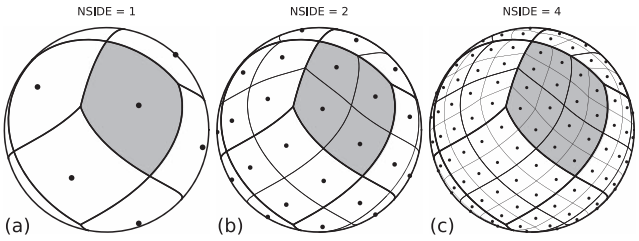

Figure 1. Pixelization of the sphere for various NSIDE values. The center of the pixels, along isolatitude lines, are represented with dots. A single base pixel is highlighted.

This work presents a new software package to generate and process muti-resolution HEALPix maps. It can create an adaptive multiorder mesh; query pixels and locations; efficiently perform binary operations between single or multi-resolution maps; and plot full or partial maps. Called `mhealpy`, it is an object-oriented extension of `healpy` (Zonca et al. 2019), a Python wrapper for HEALPix C++. The code is available to the community and open source.[9]

In this document we first review the current HEALPix standard in Section 2, followed by its generalization for multiresolution maps in Section 3. Automatic ways to construct them are discussed in Section 4, and the algorithms to query and operate on them in Section 5. Finally, some details of the implementation are presented on Section 6.

2. Single-resolution HEALPix Maps

In the HEALPix standard the pixelization of the sphere begins by dividing the sphere into 12 base pixels of equal area, as shown in Figures 1(a) and 2(a). There are four base pixels around each pole and four around the equator. Each of these base pixels can be subdivided into four child pixels, and subsequently, each child pixel can be subdivided into four pixels of a lower hierarchy level. This process can continue until the desired resolution is achieved.

The number of subdivisions along a side of a base pixel is called NSIDE—i.e., each base pixel is subdivided into $\texttt{NSIDE}^2$ pixels. This is illustrated in Figure 1. Alternatively, the pixel size can be specified using the resolution level k, also known as the map *order*. This is an integer such that such that $\texttt{NSIDE} = 2^k$. The maximum order supported on a 64-bit system is $k = 29$, which corresponds to a pixel size of $\sim$0.4 mas.

The HEALPix standard specifies two different pixel numbering schemes. The RING scheme starts from zero at the North pole and increments from west to east and from north to south, along isolatitude rings as shown in Figure 2(b). In the NESTED scheme the base pixels are labeled the same as in the RING scheme, but maps of higher NSIDE exploit the hierarchical nature of HEALPix to assign pixel numbers. As shown in Figure 2(c) the pixels are labeled such that the pixel p in a map of order k fully contains the pixels $(4p, 4(p+1))$ of a map of order $k+1$. These four child pixels are to be labeled in the following order: south, east, west and north.

[9] https://mhealpy.readthedocs.io (https://doi.org/10.5281/zenodo.5706525).

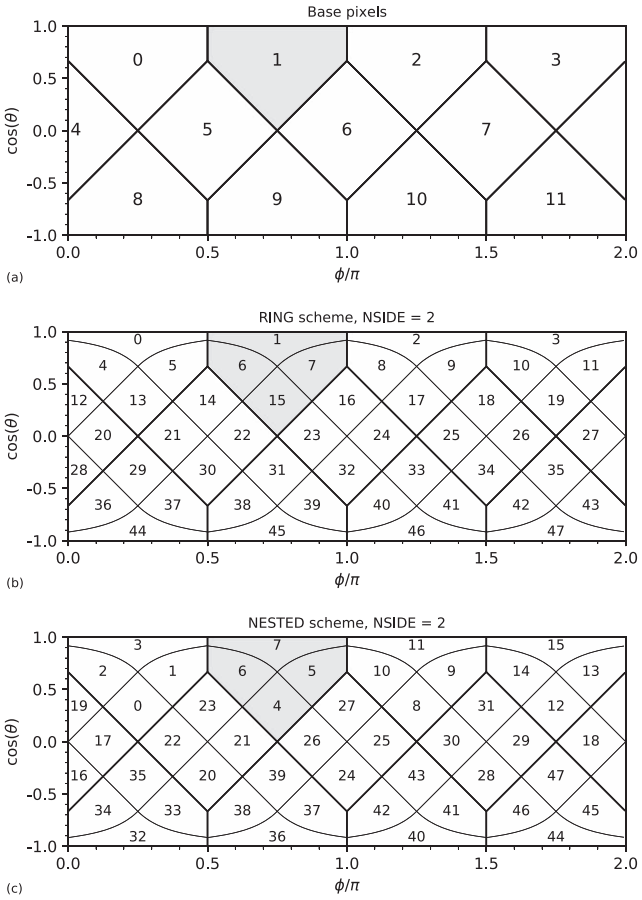

Figure 2. (a) Numbering of base HEALPix pixels (NSIDE = 1). (b), (c) Pixel numbers for a map with NSIDE = 2 under the RING and NESTED schemes, respectively. A single base pixel is highlighted.

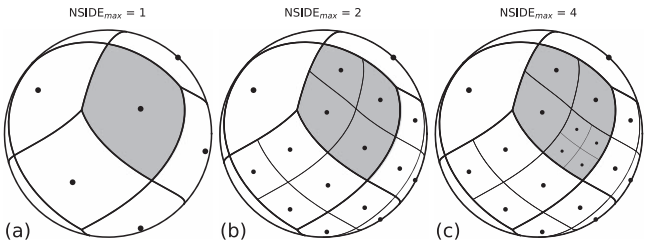

Figure 3. Simple multiresolution map. A single base pixel is highlighted (compare with Figure 1). A cylindrical projection of (c) is shown in Figure 4.

3. Geometry and Labeling of Multiresolution Maps

Thanks to its hierarchical nature obtaining different resolution levels for different regions using HEALPix is straightforward. As shown in Figure 3 only a subset of the pixels can be subdivided with each iteration. Appropriately chosen, all pixels in an arbitrary region can be equal in size, or smaller, to the pixels in a map of an arbitrary NSIDE.

In this work a multiresolution map is treated as nothing more than a compression technique. The NSIDE of a map is then defined as the equivalent NSIDE for the smallest pixel it contains. Larger pixels are simply considered to represent a group of pixels that can be sufficiently well characterized by a single value. Consequently, user code can be mesh-agnostic; that is, no changes are needed to handle single or multi-resolutions maps.

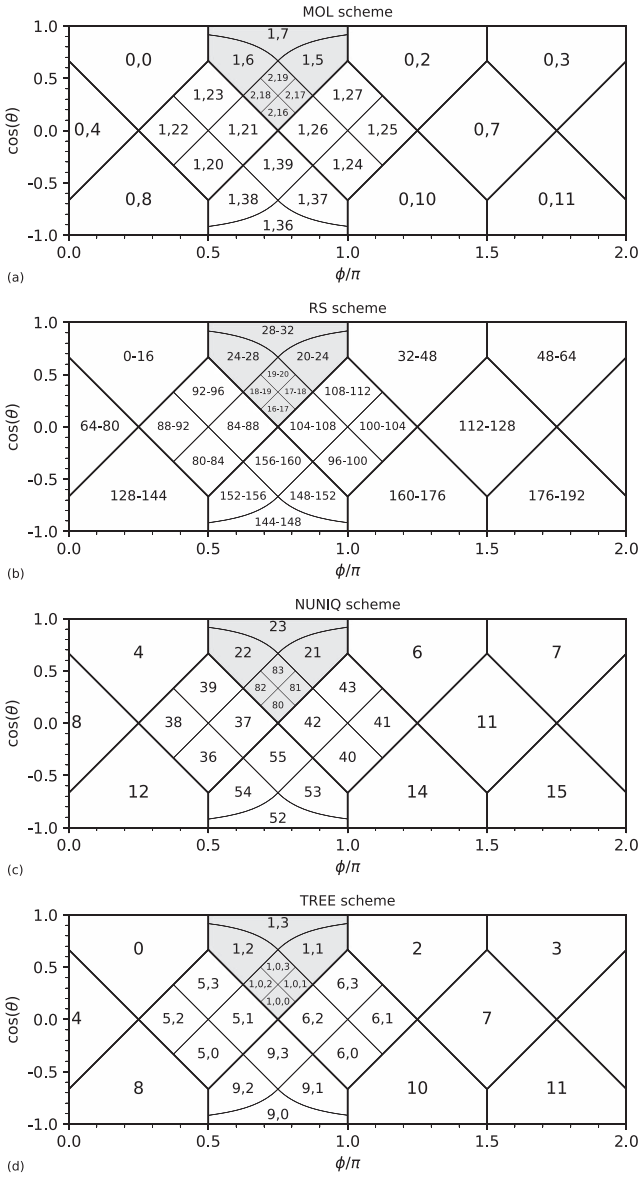

Figure 4. Comparison of various labeling schemes for multiresolution maps. The figures are cylindrical projections of Figure 3(c). A single base pixel is highlighted. (a) The `MOL` scheme labels each pixel by their equivalent order and pixel index in single-resolution `NESTED` maps. (b) The `RS` scheme uses the beginning (inclusive) and end (exclusive) indices of a equivalent `NESTED` map with the highest resolution level. (c) The `NUNIQ` encodes the equivalent order and nested pixel index into a single unambiguous number (explained in text). (d) The `TREE` scheme stores the corresponding base pixel and the node indices in a quadtree.

While there is consensus on how to construct a multiresolution HEALPix map, there exists multiple possible ways to label the pixels. One possibility is to label each pixel using the number p that would be assigned if it were a part of a single-resolution map. The `NESTED` numbering scheme is the natural choice for this purpose due to its hierarchical nature. In addition, the equivalent order k needs to be specified since different pixels from maps of different order share the same index. When arranged as one list of pixels per order this is called a multiorder list, or `MOL` scheme (see Figure 4(a)). While this provides a compact encoding it is not well-suited for various algorithms—e.g., arithmetic operations between maps, finding the pixel containing a given coordinate.

As suggested by Reinecke & Hivon (2015), efficient algorithms can be developed if a given pixel is described by a range of equivalent pixels at the highest resolution level. Using an effective `NESTED` scheme, pixels are then labeled by their beginning (inclusive) and end (exclusive) index; that is, converting from the `MOL` scheme:

$$rs_{\rm start} = 4^{k_{\rm max}-k}p$$
$$rs_{\rm stop} = 4^{k_{\rm max}-k}(p+1). \quad (1)$$

This is referred as the range set (`RS`) scheme, and it is illustrated in Figure 4(b). Using the `RS` scheme it is also straightforward to validate that a map is well-formed. The condition for this is that every location in the sphere is contained by one and only one pixel. This is satisfied if, after sorting, the beginning of the first pixel equals 0, the beginning and end of all subsequent pairs match, and the end of the last pixel equals $12\,\texttt{NSIDE}^2$.

For storage we adopt the nested unique (`NUNIQ`), proposed by Reinecke & Hivon (2015) and part of the IVOA recommendation for multiorder coverage maps (Fernique et al. 2019). In this scheme the equivalent order and nested pixel number is encoded into a single unique number, suitable for serialization into a binary format. The `NUNIQ` numbering scheme (Figure 4(c)) is defined as:

$$\texttt{UNIQ} = 4^{k+1} + p. \quad (2)$$

Since $0 \leqslant p < 3 \cdot 4^{k+1}$ this single number identifies unambiguously any pixel in a map of any given `NSIDE`. The inverse operation is:

$$k = \lfloor \log_2(\texttt{UNIQ}/4)/2 \rfloor$$
$$p = \texttt{UNIQ} - 4^{k+1}. \quad (3)$$

Finally, during the generation of a multiresolution HEALPix map, it is useful to label a pixel based on a series of indices n_i that show the position of the pixel represented as a node in a tree structure, where i runs from 0 to k (inclusive). The index n_0 is the base pixel number and the indices $n_{i>0}$ are within the range [0, 4] as seen Figure 4(d). We refer to this as the `TREE` scheme, as it explicitly recognizes each base pixels as a *quadtree*. The equivalent pixel number in a `MOL` scheme is computed as:

$$p = \sum_{i=0}^{k} 4^{k-i} n_i. \quad (4)$$

4. Mesh Generation

Different techniques can be used to efficiently generate a map with progressive refinements exploiting the hierarchical nature HEALPix—e.g., Singer & Price (2016). The output of these algorithms is a native multiresolution map, which does not need to be rasterized into single-resolution maps in order to be distributed and analyzed. Multiresolution maps, however, can be advantageous even if these methods are not used to compute a map.

An adaptive mesh refinement implementation is provided. This allows the user to automatically generate a mesh by increasing the resolution level of a given pixel using custom criteria. The algorithm accepts an arbitrary function $f(k, p)$ that decides whether a pixel should be split or added to the map as is. If split, the same evaluation is performed for the four pixels of order $k+1$ and indices $(4p,\ 4(p\ +1))$. This recursion

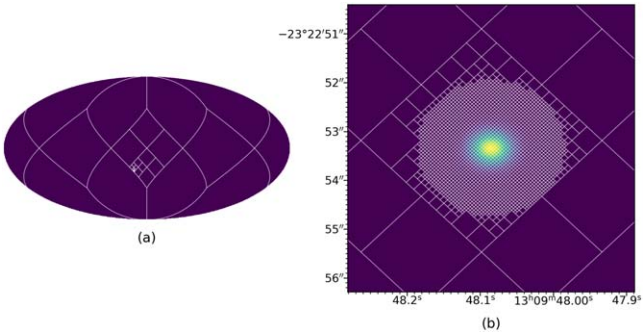

Figure 5. A multiresolution map representing the localization of SSS17a/AT 2017gfo by the Swope Telescope, both all-sky (a) and zoomed in around the source (b). The underlying mesh is overlaid. Sky maps for such well-localized sources are impractical or unfeasible using standard single-resolution maps.

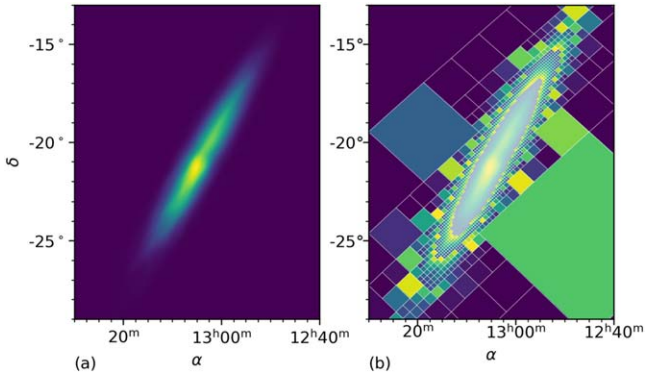

Figure 6. (a) GW170817 sky localization probability density map by LIGO and Virgo (Abbott et al. 2017) (b) Corresponding probability map integrated on the mesh resulting from an adaptive refinement process.

continues until there are no more pixels to split. This is implemented using the `TREE` scheme in order to easily guarantee the resulting map has a valid mesh.

Derived methods for common tasks are also provided. One of these occurs when we know a priori the approximate region of the sky that requires high resolution and the rest of the sky simply needs to be filled appropriately. This is exemplified in Figure 5 with a map representing the localization of the optical transient SSS17a/AT 2017gfo by the Swope Telescope (Coulter et al. 2017). A set of pixels $\{p'\}$ contained in a circular region with a radius significantly larger than the localization error $\Delta\theta \approx 0.''2$ were selected from a single-resolution map of an appropriate order k', such that the size of the pixels were a few times smaller than $\Delta\theta$. In order to have a valid map the rest of the pixels are selected by using the general adaptive mesh refinement method and choosing a splitting function $f(k, p)$ whose condition is whether there is any pixel number p' within the range $(4^{k'-k}p, 4^{k'-k}(p+1))$. The SSS17a map of effective order $k = 22$ would be impractical or unfeasible to work with if it were a standard single-resolution map, but it is straightforward to handle using this approach.

Another common task is to convert an existing single-resolution map into a multiresolution map based on a maximum value threshold. In this case the pixels act as buckets with a maximum capacity and are split if this value is exceeded. For example, in Figure 6 the GW170817 sky localization probability map by LIGO and Virgo (Abbott et al. 2017) is compressed based on the condition that no pixel must contain a greater probability that the maximum value of the original single-resolution map of order k'. That is, the criterion for the splitting function $f(k, p)$ is whether the sum of all pixels p_i within the range $(4^{k'-k}p, 4^{k'-k}(p+1))$ is greater than the threshold value. In the output multiresolution map all pixels carry a comparable weight, and the sampling resolution is proportional to the integrated probability contained in a given region. This algorithm is a form of quadtree decomposition frequently used for image compression (Shusterman & Feder 1994).

5. Map Operations

Map operations can be either related to the mesh itself, in the following referred to as "pixelization operations," or binary arithmetic operations between maps—e.g., addition, multiplication. The following section explains how these are implemented for multiresolution maps.

5.1. Pixelization Operations

Common pixel-related operations are pixel to coordinates conversion (i.e., computing the center and boundaries of a pixel), digitization (i.e., obtaining the pixel a given coordinate belongs to), and pixel querying (i.e., finding the pixels contained in or overlapping with a given region; for example, a disk). Most of these were generalized to multiresolution maps using routines written for standard single-resolution maps by choosing an appropriate labeling scheme. In particular, converting from a pixel to a coordinate is trivial in the `MOL` scheme.

The digitization is performed efficiently using the `RS` scheme. First the equivalent pixel number in a `NESTED` map of the same order is obtained, followed by a search for the actual pixel whose range contains this number. Presorting pixels based on their `RS` representation makes this and other procedures more efficient.

A similar strategy was used for pixel querying. All equivalent `NESTED` pixels overlapping a given region are found and then a search in a sorted `RS` list is performed. This works well when querying for overlapping pixels, since any pixels with at least one child pixel overlapping a region will also be itself an overlapping pixel. However, sometimes the user queries all pixels whose centers lie within a region. The previous strategy fails here since the center of a child pixel might be contained inside the region of interest while the center of the parent lies outside, even it they partially overlap. In this case, a robust albeit less efficient solution is to use the `MOL` representation and search order by order.

5.2. Binary Arithmetic Operations

Operations between standard single-resolution maps are performed pixel by pixel. If both operands have the same `NSIDE` and numbering scheme this is a straightforward and unambiguous operation. When that's not the case, it is left to the user to either upgrade or downgrade the resolution of one of the maps, and swap the scheme if needed.

Various interpolation algorithm can be used to increase the resolution of a map. At zeroth order however, and for the purpose of a binary operation, it is enough to divide each pixel of the coarser map into $4^{\Delta k}$ child pixels of equal value. This value depends on whether the quantity in a map depends on the solid angle area covered by a pixel—e.g., counts, probability or any extensive property—or not—e.g., temperature, probability density or any intensive property. That is, the child pixels in a

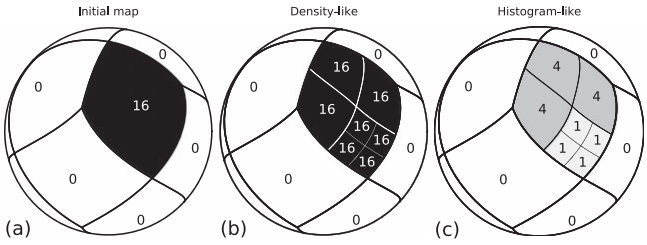

Figure 7. This figure exemplifies how pixels are automatically split during arithmetic operations depending on whether the user defines a map as density-like (b) or histogram-like (c). We use integers for simplicity, however both density-like and histogram-like maps can contain floating point values.

density-like map are assigned the same value as their parent, while in a histogram-like map the contents of the parent pixel are split equal ways among the children. The inverse process occurs when downgrading the resolution. This is exemplified in Figure 7.

Binary operations between multiresolution maps follow this same approach. During this process the algorithm combines or split pixels as needed, assigning appropriate values to the child pixels depending on whether the map is histogram-like or density-like as defined by a flag set by the user. Compare Figures 8(c) and (d) to Figures 8(e) and (f).

Any operation between two density-like maps results in a density-like map. Addition or subtraction between histogram-like maps results also in a histogram-like map. The product of a density-like map with a histogram-like map is histogram-like. The ratio between two histogram-like maps is a density-like map. The result from all other combinations defaults to the same type as the leftmost operand. Note however that these are in general ill-defined for physical quantities and can result in mesh-dependent results. Operations with scalars leave the map type unchanged.

Binary operations can be performed efficiently by having both maps in a sorted `RS` representation scheme, with the pixel ranges in both lists corresponding to the same `NSIDE`, the largest of the two maps. The algorithm then proceeds in order, splitting or combining pixels, as needed in order to match the mesh between the two operands. This is a general approach that can accept any two maps as input, whether they are single or multiresolutions maps, and whether they do or do not have the same resolution or numbering scheme.

We distinguish between standard and in-place operations. During standard binary operations, the pixel of the coarser map within a given equivalent pixel range is always split to match the mesh of the map with the highest resolution in that region. This ensures that there is no loss of information, as shown in Figures 8(c) and (e).

During an in-place operation the pixels of the second operand are either split or combined appropriately to match the pixels on the first map. The mesh of this map remains the same and the pixel values are updated on the go. No additional memory needs to be allocated, at the expense of potential information loss. This is exemplified in Figures 8(d) and (f). In-place operations can also be used to rasterize a multiresolution map by taking the product with a single-resolution map whose pixels have all been initialized to one.

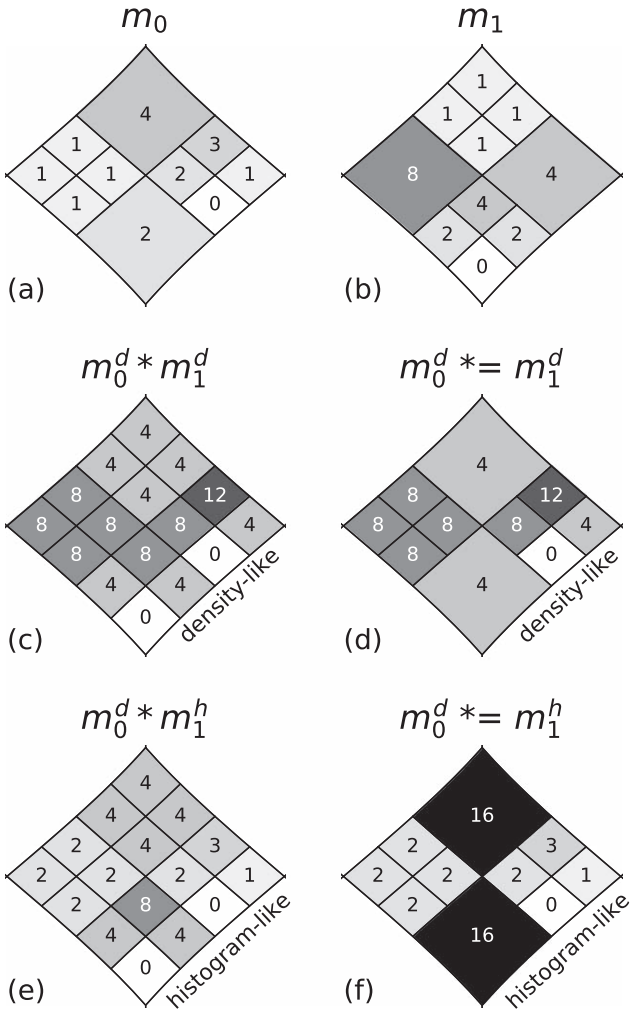

Figure 8. Example of the result of taking the product between two maps, (a) and (b), depending on various conditions. A single base pixel is shown. Figures (c) and (d) assume both maps are density-like, resulting also in a density-like map. Figures (e) and (f) consider m_0 to be density-like and m_1 histogram-like, the output therefore being histogram-like. Standard operations, with no information loss, are shown in (c) and (e), while in-place operations that keep the m_0 mesh unchanged result in (d) and (f).

6. Implementation

The ideas presented in Sections 4 and 5 were implemented in the Python library `mhealpy`. The only direct dependency is the `healpy` library, which in turns is a wrapper to the HEALPix C++ library. Similar nomenclature makes it easy to refactor code using `healpy` and generalize it to handle multiresolution maps.

The `mhealpy` library uses an object-oriented design. The user does not need to keep track of the various map properties. Similar to the C++ implementation, most of the code is either accessed through the `HealpixBase` or the `HealpixMap` classes. The former stores the mesh and numbering scheme, and contains all the pixelization operations which do not involve the map contents. `HealpixMap` is a derived class of `HealpixBase`, which in addition contains the map contents and whether it is a histogram-like or density-like map.

Most operations available in `healpy` were generalized in `mhealpy` for their use with multiresolution maps without rasterizing them into single-resolution maps first, which saves computer resources. This includes plotting. Spherical harmonic transforms are an exception, they were not implemented for multiresolution maps since it is more efficient to work with standard HEALPix maps.

The map serialization is complaint with the IVOA standard (Fernique et al. 2019) for multiorder coverage maps. Maps are saved into FITS (Hanisch et al. 2001) tables with an explicit `NUNIQ` ordering. This is readily compatible with LIGO-Virgo skymaps which adopted the same format.

7. Summary

This work introduced `mhealpy`, an object-oriented wrapper of the `healpy` Python library that allows to create and handle multiresolution HEALPix maps. This extension to the current standard prevents the need for a homogeneous all-sky grid, making it a viable option as a common format to share spatial information for joint multiwavelength and multimessenger analyses, including for well-localized detections. We presented the definition of multiresolution maps, various pixel labeling schemes, and how the various functionalities were implemented: pixel querying, binary arithmetic operations, adaptive grid refinement, plotting, and serialization into FITS files.

We would like to thank Boyan A. Hristov (UAH) for the valuable comments that improved this work significantly. The material is based upon work supported by NASA under award number 80GSFC21M0002. Some of the results in this paper have been derived using the healpy and HEALPix package.

ORCID iDs

I. Martinez-Castellanos https://orcid.org/0000-0002-2471-8696
Leo P. Singer https://orcid.org/0000-0001-9898-5597
E. Burns https://orcid.org/0000-0002-2942-3379
D. Tak https://orcid.org/0000-0002-9852-2469
Alyson Joens https://orcid.org/0000-0001-5783-8590
Judith L. Racusin https://orcid.org/0000-0002-4744-9898
Jeremy S. Perkins https://orcid.org/0000-0001-9608-4023

References

Aab, A., Abreu, P., Aglietta, M., et al. 2014, ApJ, 789, 160
Aartsen, M. G., Abraham, K., Ackermann, M., et al. 2017, ApJ, 835, 151
Abbott, B. P., Abbott, R., Abbott, T. D., et al. 2017, PhRvL, 119, 161101
Abdollahi, S., Acero, F., Ackermann, M., et al. 2020, ApJS, 247, 33
Abeysekara, A. U., Albert, A., Alfaro, R., et al. 2017, ApJ, 843, 39
Coulter, D. A., Foley, R. J., Kilpatrick, C. D., et al. 2017, Sci, 358, 1556
Fernique, P., Allen, M. G., Boch, T., et al. 2015, A&A, 578, A114
Fernique, P., Boch, T., Donaldson, T., et al. 2019, MOC—HEALPix Multi-Order Coverage map, v1.1, IVOA Recommendation, doi:10.5479/ADS/bib/2014ivoa.spec.0602F
Górski, K. M., Hivon, E., Banday, A. J., et al. 2005, ApJ, 622, 759
Greco, G., Punturo, M., Allen, M., et al. 2022, A&C, 39, 100547
Hanisch, R. J., Farris, A., Greisen, E. W., et al. 2001, A&A, 376, 359
Reinecke, M., & Hivon, E. 2015, A&A, 580, A132
Shusterman, E., & Feder, M. 1994, ITIP, 3, 207
Singer, L. P., & Price, L. R. 2016, PhRvD, 93, 024013
von Kienlin, A., Meegan, C. A., Paciesas, W. S., et al. 2020, ApJ, 893, 46
Youngren, R. W., & Petty, M. D. 2017, Heliyon, 3, e00332
Zonca, A., Singer, L., Lenz, D., et al. 2019, JOSS, 4, 1298