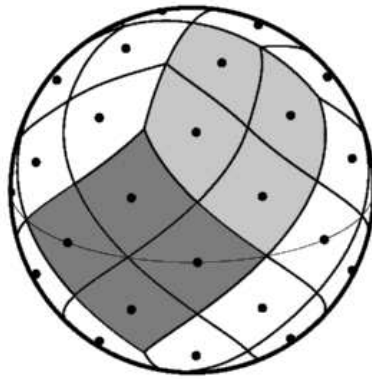


# HEALPix IDL Facilities Overview



Revision: Version 3.30; October 8, 2015

Prepared by: Eric Hivon, Anthony J. Banday, Benjamin D. Wandelt, Frode K. Hansen and Krzysztof M. Górski

Abstract: This document is an overview of the **HEALPix** IDL facilities.

<http://healpix.sf.net>

## TABLE OF CONTENTS

Using the <b>HEALPix</b> IDL facilities . . . . .	5
Using <b>HEALPix</b> IDL together with other IDL libraries . . . . .	5
Using GDL instead of IDL . . . . .	5
What is available? . . . . .	6
Maps related tools . . . . .	6
Pixels related tools . . . . .	6
Power spectrum, alm, beam and pixel window functions . . . . .	7
Other tools . . . . .	7
Changes between releases 3.20 and 3.30 . . . . .	8
Previous changes . . . . .	8
alm_i2t . . . . .	13
alm_t2i . . . . .	15
alm2fits . . . . .	17
ang2vec . . . . .	20
angulardistance . . . . .	22
azeqview . . . . .	24
beam2bl . . . . .	26
bin_llcl . . . . .	28
bl2beam . . . . .	30
bl2fits . . . . .	32
cartcursor . . . . .	34
cartview . . . . .	35
change_polconv . . . . .	37
cl2fits . . . . .	40
convert_oldhpx2cmbfast . . . . .	43
euler_matrix_new . . . . .	45
fits2alm . . . . .	47
fits2cl . . . . .	50
gaussbeam . . . . .	54
getdisc_ring . . . . .	56
getsize_fits . . . . .	57
gnomcursor . . . . .	60

gnomview	61
healpix_doc: PDF and HTML documentation	64
healpixwindow	66
hpx2dm	68
hpx2gs	71
ialteralm	74
ianafast	78
index2lm	82
init_healpix and !healpix system variable	84
iprocess_mask	86
ismoothing	89
isynfast	92
lm2index	95
median_filter	96
mollcursor	98
mollview	101
neighbours_nest	120
neighbours_ring	122
nest2uniq	124
npix2nside	126
nside2npix	128
nside2ntemplates	130
orthcursor	132
orthview	133
pix2xxx, ang2xxx, vec2xxx, nest2ring, ring2nest	135
planck_colors	138
query_disc	140
query_polygon	142
query_strip	144
query_triangle	146
read_fits_cut4	148
read_fits_map	150
read_fits_s	153
read_tqu	155

remove_dipole . . . . .	158
reorder . . . . .	161
rotate_coord . . . . .	163
same_shape_pixels_nest & same_shape_pixels_ring . . . . .	165
template_pixel_nest & template_pixel_ring . . . . .	168
ud_grade . . . . .	170
uniq2nest . . . . .	173
vec2ang . . . . .	175
write_fits_cut4 . . . . .	177
write_fits_map . . . . .	181
write_fits_sb . . . . .	184
write_tqu . . . . .	188

## Using the **HEALPix** IDL facilities

The current version of the **HEALPix** package provides an IDL startup file which defines various environment variables for your convenience, and adds the **HEALPix** IDL directory tree to your `IDL_PATH`. In order to utilise this feature, the user should invoke IDL using the commands `hidl` or `hidlde` which are aliases defined in the **HEALPix** profile created during the installation process for the package (see [the Installation Document](#)).

### Using **HEALPix** IDL together with other IDL libraries

Many users want to use **HEALPix** IDL routines at the same time as other (home made or third party) IDL routines. There are several ways to achieve this:

– *with* `hidl`:

before starting `hidl` or `hidlde`, (re)define the environment variable `$IDL_PATH` so that it looks like: “`+/path/to/my/idl/routines:+/path/to/other/idl/routines:<IDL_DEFAULT>`” (where `<IDL_DEFAULT>` should be typed literally and the `+/path` means that subdirectories of `path` will be searched recursively). For example, if young Albert types in Bourne shell:

```
export IDL_PATH="+/home/aeinstein/brownian:<IDL_DEFAULT>"
```

```
hidl
```

he will start an IDL session in which the **HEALPix** IDL routines are accessible, followed by all those located in subdirectories of `/home/aeinstein/brownian`, followed by the standard IDL routines. If `.pro` files of the same name are available at the different locations, the first one encountered will prevail;

– *without* `hidl`:

before starting IDL, the environment variables `$IDL_PATH` and `$IDL_STARTUP` must be defined. For instance, to emulate under (ba)sh the behavior of `hidl` shown above, the same (bolder) Albert will type:

```
export IDL_PATH="+${HEALPIX}/src/idl:+/home/aeinstein/brownian:<IDL_DEFAULT>"
```

```
export IDL_STARTUP="+${HEALPIX}/src/idl/HEALPix_startup"
```

```
idl
```

GDL users can do the same, replacing `hidl`, `IDL_PATH` and `IDL_STARTUP` with `hgdl`, `GDL_PATH` and `GDL_STARTUP` respectively, while `<IDL_DEFAULT>` must be replaced by the default value of `${GDL_PATH}`.

Of course, `${HEALPIX}/src/idl` (and all the `+` signs) remain unchanged.

### Using GDL instead of IDL

See the section “[Using GDL instead of IDL](#)” in [the Installation Document](#) for more information on this issue.

## What is available?

The **HEALPix**-IDL tools are mostly designed to generate, visualize, filter and analyze sky maps; identify, query and process **HEALPix** pixels; and deal with angular spectral objects (such as power spectra or Spherical Harmonics coefficients), as detailed below. The full documentation is available online in IDL via [healpix\\_doc](#)

### HEALPix maps related tools

- Visualization: gnomonic, Mollweide, Cartesian, orthographic and azimuthal equatorial projections [mollview](#), [gnomview](#), [cartview](#), [orthview](#), [azeqview](#) (with interactive cursor: [cartcursor](#), [mollcursor](#), [gnomcursor](#), [orthcursor](#))
- Color table creation [planck\\_colors](#)
- Production of **HEALPix** maps in Google Sky and Dome Master format [hpx2gs](#), [hpx2dm](#).
- Spherical Harmonics analysis and synthesis: [ianafast](#), [isynfast](#)
- Smoothing and filtering: [ismoothing](#), [median\\_filter](#), [remove\\_dipole](#)
- Pixel pro/down-grading and NESTED/RING pixel reordering: [ud\\_grade](#), [reorder](#)
- Mask processing: [iprocess\\_mask](#)
- Maps I/O: [read\\_fits\\_cut4](#), [read\\_fits\\_map](#), [read\\_fits\\_s](#), [read\\_tqu](#).  
[write\\_fits\\_cut4](#), [write\\_fits\\_map](#), [write\\_fits\\_sb](#), [write\\_tqu](#).  
[getsize\\_fits](#), [change\\_polconv](#)

### HEALPix pixels related tools

- Coordinate tools: [ang2vec](#), [angulardistance](#), [euler\\_matrix\\_new](#), [rotate\\_coord](#), [vec2ang](#)
- Coordinates to pixel transforms, and back [nside2npix](#), [npix2nside](#), [ang2pix\\_\\*](#), [pix2ang\\_\\*](#), [pix2vec\\_\\*](#), [vec2pix\\_\\*](#)
- RING/NESTED transforms [nest2ring](#), [ring2nest](#)
- Unique Identifier to NESTED index, and back [uniq2nest](#), [nest2uniq](#)
- Neighbouring pixels: [neighbours\\_nest](#), [neighbours\\_ring](#)
- Pixel query within a disc, polygon, strip or triangle: [query\\_disc](#), [query\\_polygon](#), [query\\_strip](#), [query\\_triangle](#).

- Template pixels: `nside2ntemplates`, `same_shape_pixels_ring`, `same_shape_pixels_nest`, `template_pixel_ring`, `template_pixel_nest`

## Power spectrum, $a_{lm}$ , beam and pixel window functions

- $B(l)$ ,  $B(\theta)$  and pixel WF generation: `gaussbeam`, `beam2bl`, `bl2beam`, `healpixwindow`
- $C(l)$  binning: `bin_llcl`
- $a_{lm}$  handling tools: `alm_i2t`, `alm_t2i`, `index2lm`, `lm2index`, `ialteralm`
- $C(l)$ ,  $B(l)$  and  $a_{lm}$  I/O: `fits2cl/cl2fits`, `bl2fits`, `fits2alm/alm2fits`,

## Other tools

- HEALPix variables and paths initialization: `init_healpix`
- online documentation: `healpix_doc`

## Changes between releases 3.20 and 3.30

- `azeqview`, `cartview`, `gnomview`, `mollview`, `orthview` visualization routines:
  - addition of `PDF` keyword for production of Adobe PDF outputs;
  - addition of `LATEX` keyword for genuine or emulated  $\text{\LaTeX}$  processing of character strings;
  - addition of `PFonts` keyword to select origin and type of character font;
  - the `CROP` keyword now has the same behavior for all output media (GIF, JPEG, PDF, PNG, PS, ... and X); the `NOBAR` keyword now removes the color bar *or* the polarization color wheel, as applicable; correct EQUINOX date in header of output `FITS` map; the double precision maps and those with constant value are now correctly handled.
- `fits2cl`: addition of `/PLANCK2` keyword to read best fit  $C(\ell)$  model to Planck 2015 data.
- new routines `nest2uniq` and `uniq2nest` for conversion of standard pixel index to/from Unique ID number. See "[The Unique Identifier scheme](#)" section in "[HEALPix Introduction Document](#)" for more details.
- HEALPix enabled GDL commands (`hgdl` and `hgdlde`) are defined during the [configuration process](#).
- update of the required `IDL-astron library` routines, and `Coyote` library routines (2015-09-23).

## Previous changes

### Changes between releases 3.11 and 3.20

- addition of `ialteralm` to modify Spherical Harmonics coefficients ( $a_{\ell m}$ ).
- addition of `planck.colors` to modify current color table to one used in Planck 2013 publications.
- `cartview`, `gnomview`, `mollview`, `orthview`:
  - addition of `BAD_COLOR`, `BG_COLOR` and `FG_COLOR` keywords to change the color of the missing pixels, background and foreground labels and lines.
  - support for `COLT='planck1'` and `COLT='planck2'` to use the Planck color tables defined in `planck.colors`
- Bugs correction in `bin_llcl`, `query_disc`.
- update of the required `IDL-astron library` routines, and their supporting `Coyote` routines (2014-11-10).



## Changes between releases 3.00 and 3.11

- Latest edition (version 3.11)
  - `ang2pix_ring` and `pix2ang_nest` routines now accept scalar arguments
- Previous edition (version 3.10)
  - bug corrections: `query_disc`: correct handling of empty disc; `bin_llcl`: correct handling of optional argument.
  - double precision of input now preserved in `gaussbeam` and `euler_matrix_new`.
  - `fits2cl`: addition of `/PLANCK1` keyword to read best fit  $C(\ell)$  model to Planck 2013 + external data.
  - it is now possible to read a specific FITS file extension identified by its (0-based) number or its case-insensitive EXTNAME value with the `Extension` keyword added to `fits2cl`, `getsize_fits`, `read_fits_map`, `read_fits_s` and `read_tqu`.
  - update of the required `IDL-astron library` routines, and their supporting `Coyote` routines (2013-02-08).

## Changes between releases 2.20 and 3.00

- Previous edition (version 3.0)
  - New routines to go from circular beam profile to transfer function (`beam2b1`), and back (`b12beam`); to go from indexed list of  $a_{lm}$  to a(l,m) 2D table (`alm_i2t`), and back (`alm_t2i`); and to compute the angular distance between pairs of vectors (`angulardistance`).
  - addition of `iprocess_mask` interface to F90 `process_mask` facility to compute the angular distance of valid pixels to the closest invalid pixels for a input binary mask.
  - creation of `hpx2dm` routine to generate DomeMaster images of **HEALPix** maps that can be projected on planetariums.
  - the pixel query routines `query_triangle`, `query_polygon`, and in particular `query_disc`, have been improved and will return fewer false positive pixels in the *inclusive* mode
  - improved accuracy of the co-latitude calculation in the vicinity of the poles for high resolution in `nest2ring`, `ring2nest`, `pix2ang_*`, `pix2vec_*`, ...
  - `cartview`, `gnomview`, `mollview`, `orthview`: the length and spacing of the headless vectors used to represent polarization is now user-controlled via `POLARIZATION` keyword. The `COLT` keyword now allows the use of an interactively modified color table.
  - `orthview` now accepts `STAGGER` keyword to overplot staggered spheres (with a twist) in order to detect periodic boundary conditions on the sky
  - `fits2cl`: addition of `WMAP7` keyword to read best fit  $C(l)$  model to WMAP 7yr data.
  - `read_fits_map` can now read  $N_{\text{side}}=8192$  **HEALPix** maps and is generally faster than previously for smaller maps
  - update of `astron` library routines (01-Feb-2012).

## Changes between release 2.0 and 2.20

Several routines have been added or improved since version 2.0, as listed below. Note that thanks to the newer IDL-`astron` library, FITS read/write routines in IDL-Healpix routines can now deal with **FITS files larger than 2GB** (on architectures supporting 64bit addressing).

Using 64 bit integers available since version 5.2 of IDL the maximum resolution parameter `Nside` supported has increased from  $2^{13} = 8192$  to  $2^{29} = 536870912$ , corresponding to  $3.46 \cdot 10^{18}$  pixels on the sphere.

- Recent edition (versions 2.20 and 2.20a)
  - `fits2cl`: addition of `WMAP1` and `WMAP5` keywords to read best fit  $C(l)$  model to WMAP 1st and 5yr data respectively,
  - `cartview`, `gnomview`, `mollview`, `orthview`: the `OUTLINE` option now accept symbols with `PSYM > 8`, using `symcat` symbols definition.
- Recent editions (versions 2.15 and 2.15a)
  - `cartview`, `gnomview`, `mollview`, `orthview`:
    - \* export of projected map into a FITS file (`FITS` keyword), or an IDL array (`MAP_OUT` option) now available with all viewing routines,
    - \* added `CHARTHICK` support; accept array of `OUTLINE` structures (if they have the same fields), and still support structure of structures,
    - \* correction of a bug (in `loaddata_healpix`) that was affecting the behavior of these viewing routines after consecutive calls with very partial cut-sky *and then* full-sky data sets [2.15a];
  - `remove_dipole` now outputs the monopole and dipole `covariance matrix`;
  - `write_fits_map`, `write_tqu`, `write_fits_sb`: `BAD_DATA` keyword added to FITS header;
  - update of `astron` library routines (24-May-2010) for improved WCS support.
- Previous edition (version 2.14a)
  - `cartview`, `gnomview`, `mollview`, `orthview`:
    - \* `OUTLINE=`, `GRATICULE=`, `IGRATICULE=` work again with virtual windows (`WINDOW < 0`)
    - \* `YPOS=` and `RETAIN=` keywords active again
    - \* `PS=` keyword fixed
  - `orthview`: fixed problems with `/SHADE` keyword, which now outputs 8-byte (instead of 16-byte) PNG files
  - `ianafast`, `ismoothing`: fixed problem with processing of polarized maps stored in memory.
  - `ud_grade`: improved handling of flagged pixels on Double Precision input maps
  - `remove_dipole`: `COORD_IN=` and `COORD_OUT=` now accept lower case values; `/SILENT` keyword added.
- Old edition (version 2.13)
  - new `healpix.doc` routine to browse HTML and PDF documentations
  - `cartview`, `gnomview`, `mollview`, `orthview`:
    - \* introduction of the `TRUECOLORS=` keyword to generate color image from 3 channel map
    - \* extended capability of the `TRANSPARENT=` keyword

- \* addition of `MAP_OUT=` to `gnomview`
  - improved compatibility with `GDL` (free IDL clone). See “[HEALPix Installation Document](#)” for current `GDL` limitations.
  - update of the `IDL-astron` library routines, which now require IDL 6.1 or more
  - `fits2alm`: new `LMAX=` and `LMIN=` keywords
  - `fits2cl`: new `LLFACTOR=` keyword
  - `init_healpix` defines substructure with complete path to **HEALPix** subdirectories (`test`, `data`, `bin`)
  - slightly faster `write_fits_cut4` and `write_fits_sb` routines.
  - `ianafast`, `ismoothing`: solved problem with `W8DIR=` keyword.
- Older editions (versions 2.11 and 2.12a)
    - `ianafast`, `ismoothing`, `isynfast`: the `TMPDIR` keyword now works properly, and `$IDL_TMPDIR` is used as the default temporary directory ; more stable behaviour of these routines
    - `ud_grade`:
      - \* correctly flags bad output pixels with `bad_data` value when upgrading maps
      - \* cut sky map: improved, faster routine, now works for `Nside > 8192`
    - `cartview`, `gnomview`, `mollview`, `orthview`:
      - \* using a virtual window (ie, setting `WINDOW` to a negative value) now allows faster generation of GIF and PNG files (especially useful over remote connections);
      - \* addition of `RETAIN=` keyword;
      - \* deals correctly with user provided `MIN` and `MAX` in `LOG` and `ASINH` modes
      - \* polarization norm map can be offset (`POLARIZATION=1` mode)
      - \* original color table and plot settings are restored when leaving these routines
    - `orthview`: addition of `/SHADED` keyword for 3D rendering
    - issues warning when non-integer pixel indexes are fed to `nest2ring`, `ring2nest`, `pix2ang-*`, `pix2vec-*`, ...
    - `ximview`:
      - \* fixed problem with cut-sky FITS files
      - \* color scale bar added to PNG output
      - \* version 0.6.2, fixed bug in pixel coordinates
    - cosmetic editions to `remove_dipole`
  - New routines in version 2.10 include
    - `ximview`: visualisation routine developed by J. P. Leahy intended for quick-look inspection of HEALPix images (as well as ordinary 2-D images) at the level of individual pixels. Features include panning, zooming, blinking, image statistics and peak finding.
    - `hpx2gs`: turns a healpix data set into a [Google Earth/Google Sky](#)-compatible image
    - `ianafast`: interface to (F90) `anafast` and (C++) `anafast_cxx` facilities
    - `isynfast`: interface to F90 `synfast` facility
    - `ismoothing`: interface to F90 `smoothing` facility

- `bin_llc1`:  $C(l)$  binning
- `bl2fits`: writes  $B(l)$  or  $W(l)$  window into FITS file
- `neighbours_nest`, `neighbours_ring`: find immediate neighbours of a given pixel
- `query_strip`: find pixels lying within a colatitude strip
- Routines with extended/improved user interface or new functionalities include
  - `mollview`, `gnomview`, `cartview`, `orthview`:
    - \* `ONLINE` keyword is now redundant,
    - \* introduction of `GLSIZE` and `IGLSIZE` to control automatic labeling of graticules, see Fig. 2 on page 117
    - \* addition of `SILENT` and `EXECUTE` keywords, see Fig. 2 on page 117
    - \* addition of `ASINH` keyword to allow better visualisation of highly contrasted maps; see Figure 3 on page 118,
    - \* under certain circumstances, can process high resolution cut sky data sets without creating full sky dummy maps,
    - \* accept gzip compressed FITS files,
    - \* accept polarized cut sky maps,
    - \* accept multi-dimensional online arrays,
    - \* more robust `OUTLINE` option.
  - `median_filter`: bugs correction
  - `ud_grade`: more robust user interface
  - `change_polconv`: new `/FORCE` keyword
  - `remove_dipole`: more accurate
  - `query_disc`: when the disc center is located at one of the poles, *only* the pixels overlapping with the disc are now returned.
- Miscellaneous
  - `mollcursor`, `gnomcursor...`: an X11 patch is given so that these routines work under Mac OS X 10.4 and 10.5.

# alm\_i2t

**Location in HEALPix directory tree: src/idl/misc/alm\_i2t.pro**

This IDL function turns an indexed list of alm (as generated by `fits2alm`) into a tabular (real or complex)  $a(l,m)$  array for easier manipulation

**FORMAT** IDL> alm\_table=alm\_i2t(**Index**, **Alm\_vec**,  
[**/COMPLEX**, **/HELP**, **LMAX=**, **MMAX=**])

## QUALIFIERS

Index	Integer vector of size ni containing the index $i$ of the of $a_{lm}$ coefficients, related to $\{l,m\}$ by $i = l^2 + l + m + 1$
Alm_vec	Array of $a_{lm}$ coefficients, with dimension (ni, nalm [,nsig]) where ni = number of $i$ indices nalm = 2 for real and imaginary parts of alm coefficients <i>or</i> 4 for above plus corresponding error values nsig = number of signals (usually 1 for any of T E B or 3 for T,E,B together)

## KEYWORDS

/COMPLEX	if set, the output array is complex with dimensions (lmax+1, mmax+1, [nalm/2 , nsig]), otherwise, the array is real with dimensions (lmax+1, mmax+1, nalm [, nsig]). lmax and mmax are determined from input Index values, unless set otherwise by user.
/HELP	if set, prints out the help header and exits
LMAX=	lmax to be used in output array, regardless of value found in input index
MMAX=	mmax to be used in output array, regardless of value found in input index

---

**DESCRIPTION** `alm_i2t` returns a real or complex array, containing the  $a_{lm}$  with  $0 \leq l \leq l_{\max}$  and  $0 \leq m \leq m_{\max}$ . The negative  $m$  are therefore ignored.

---

## RELATED ROUTINES

This section lists the routines related to **`alm_i2t`**.

<code>idl</code>	version 6.4 or more is necessary to run <code>alm_i2t</code> .
<code>alm_t2i</code>	turns tabular alm's such as those generated by <code>alm_i2t</code> into indexed lists than can written to FITS files with <code>alm2fits</code>
<code>alm2fits, fits2alm</code>	routines to read and write $a_{lm}$ indexed lists from and to FITS files.

---

## EXAMPLE:

```
fits2alm, i1, a1, 'alm1.fits'
ac1 = alm_i2t(i1, a1, /complex, lmax=100, mmax=100)

fits2alm, i2, a2, 'alm2.fits'
ac2 = alm_i2t(i2, a2, /complex, lmax=100, mmax=100)

ac = 0.9*ac1 + 0.1*ac2

alm_t2i, ac, i, a
alm2fits, i, a, 'almsum.fits'
```

The example above reads 2 sets of  $a_{lm}$  from FITS files, puts the alm's with  $(l,m) \leq 100$  in tabular arrays, and then make a weighted sum of the alm's. The resulting alm or put back into a indexed list in order to be written to FITS.

# alm\_t2i

Location in HEALPix directory tree: `src/idl/misc/alm_t2i.pro`

This IDL facility turns a tabular (real or complex)  $a(l,m)$  array into an indexed list of alm that can be written into a FITS file with `alm2fits`

---

**FORMAT** IDL> alm\_t2i, Alm\_table, Index, Alm\_vec, [/HELP, /MFIRST])

---

## QUALIFIERS

Alm_table	Input real or complex array, containing all the $a_{lm}^s$ for $l$ in $[0, l_{\max}]$ and $m$ in $[0, m_{\max}]$ (and $s$ in $[0, s_{\max}]$ if applicable) if REAL it has 3 (or 4) dimensions, if COMPLEX it has 2 (or 3) dimensions
Index	Output integer vector of size ni containing the index $i$ of the $a_{lm}$ coefficients, related to $\{l, m\}$ by $i = l^2 + l + m + 1$
Alm_vec	Output array of $a_{lm}$ coefficients, with dimension (ni, 2 $[, s_{\max} + 1]$ ) where ni = number of $i$ indices 2 for real and imaginary parts of alm coefficients $s_{\max} + 1$ = number of signals (usually 1 for any of T E B or 3 for T,E,B together)

---

## KEYWORDS

/HELP	if set, prints out the help header and exits
/MFIRST	if set, the input array is $a(m,l)$ instead of $a(l,m)$

---

**DESCRIPTION** alm\_t2i turns a real or complex tabular array of  $a(l,m)$  (or  $a(m,l)$  if MFIRST is set) into a real list of  $a_{lm}$  (with the real and imaginary parts separated) and its index  $i = l^2 + l + m + 1$ . The unphysical  $m > l$  elements of the input table are dropped from the output list.

---

## RELATED ROUTINES

This section lists the routines related to **alm\_t2i**.

idl	version 6.4 or more is necessary to run alm_t2i.
<b>alm_i2t</b>	this function is complementary to alm_t2i and turns an indexed list of alm (as generated by <b>fits2alm</b> ) into a tabular (real or complex) a(l,m) array for easier manipulation
<b>alm2fits, fits2alm</b>	routines to read and write $a_{lm}$ indexed lists from and to FITS files.

---

## EXAMPLE:

See **alm\_i2t example**



# alm2fits

---

**Location in HEALPix directory tree:** `src/idl/fits/alm2fits.pro`

This IDL routine provides a means to write spherical harmonic coefficients (and optional errors) and their index label to a FITS file. Each signal is written to a separate binary table extension. The routine also writes header information if required. The facility is primarily designed to allow the user to write a FITS files containing constraints for a constrained realisation performed by the **HEALPix** facility **synfast**.

---

**FORMAT** IDL> ALM2FITS, *index*, *alm\_array*, *fitsfile*,  
[HDR=, /HELP, XHDR=]

---

## QUALIFIERS

<i>index</i>	Long array containing the index for the corresponding array of alm coefficients (and erralm if required). The index $i$ is related to $l, m$ by the relation $i = \ell^2 + \ell + m + 1$
<i>alm_array</i>	Real array of alm coefficients written to the file. This has dimension (nl,nalm,nsig) – corresponding to nl = number of l,m indices nalm = 2 for real and imaginary parts of alm coefficients or 4 for above plus corresponding error values nsig = number of signals to be written (1 for any of T E B or 3 if ALL to be written). Each signal is stored in a separate extension.
<i>fitsfile</i>	String containing the name of the file to be written.

---

## KEYWORDS

HDR = String array containing the primary header to be written in the FITS file.

---

/HELP	If set, the routine documentation header is shown and the routine exits
XHDR =	String array containing the extension header. If ALL signals are required, then each extension table is given this header. NOTE: optional header strings should NOT include the header keywords explicitly written by this routine.

---

**DESCRIPTION** `alm2fits` writes the input alm coefficients (and associated errors if required) into a FITS file. Each signal type is written as a separate binary table extension. Optional headers conforming to the FITS convention can also be written to the output file. All required FITS header keywords are automatically generated by the routine and should NOT be duplicated in the optional header inputs. The keywords EXTNAME and TTYPE\* are now also automatically generated.

---

## RELATED ROUTINES

This section lists the routines related to **alm2fits**.

idl	version 6.4 or more is necessary to run <code>alm2fits</code> .
<code>fits2alm</code>	provides the complimentary routine to read in alm coefficients from a FITS file.
<code>alm_i2t</code> , <code>alm_t2i</code>	these facilities turn indexed lists of $a_{\ell m}$ into 2D $a(l,m)$ tables and back
<code>lm2index</code>	converts the $a_{\ell m}$ order and degree $(\ell, m)$ into the index $i = \ell^2 + \ell + m + 1$ required by <code>alm2fits</code> .
<code>cl2fits</code>	routine to write a power spectrum into a FITS file.
<code>fits2cl</code>	routine to read/compute $C(l)$ power spectra from a file containing $C(l)$ or $a_{\ell m}$ coefficients
<code>alteralm</code>	utilises the output file generated by <code>alm2fits</code> .
<code>synfast</code>	utilises the output file generated by <code>alm2fits</code> .

---

## EXAMPLE:

---

```
alm2fits, index, alm, 'alm.fits', HDR = hdr, XHDR = xhdr
```

alm2fits writes the coefficients stored in the variable `alm` to the output FITS file `alm.fits` with optional headers passed by the string variables `hdr` and `xhdr`.

## ang2vec

---

Location in HEALPix directory tree: `src/idl/toolkit/ang2vec.pro`

This IDL facility convert the position angles of points on the sphere into their 3D position vectors.

---

**FORMAT**            IDL> ANG2VEC , **Theta**, **Phi**, **Vector**[, **ASTRO=**]

---

## QUALIFIERS

Theta	input: scalar or vector, colatitude in radians measured southward from north pole (in $[0, \pi]$ ). If ASTRO is set, Theta is the latitude in degrees measured northward from the equator (in $[-90, 90]$ ).
Phi	input: scalar or vector of same size as Theta, longitude in radians measured eastward (in $[0, 2\pi]$ ). If ASTRO is set, it is the longitude in degree measured eastward (in $[0, 360]$ ).
Vector	output : array, three dimensional cartesian position vector $(x, y, z)$ normalised to unity. The north pole is $(0, 0, 1)$ . The coordinates are ordered as follows $x(0), \dots, x(n-1)$ , $y(0), \dots, y(n-1)$ , $z(0), \dots, z(n-1)$

---

## KEYWORDS

ASTRO=	if set Theta and Phi are the latitude and longitude in degrees instead of the colatitude and longitude in radians.
--------	--

---

**DESCRIPTION** ang2vec performs the geometrical transform from the position angles of points  $(\theta, \phi)$  into their position vectors  $(x, y, z)$ :  $x = \sin \theta \cos \phi$ ,  $y = \sin \theta \sin \phi$ ,  $z = \cos \theta$

---

## RELATED ROUTINES

This section lists the routines related to **ang2vec** .

idl	version 6.4 or more is necessary to run ang2vec .
<b>pix2xxx</b> , ...	conversion between vector or angles and pixel index
<b>vec2ang</b>	conversion from position vectors to angles

---

## EXAMPLE:

```
lat = -45 ; latitude in degrees
long = 120 ; longitude in degrees
ang2vec, lat, lon, /astro, vec
```

will return in **vec** the 3D cartesian position vector of the point of latitude -45 deg and longitude 120 deg

# angulardistance

Location in HEALPix directory tree: `src/idl/toolkit/angulardistance.pro`

This IDL facility computes the angular distance (in RADIANS) between pairs of vectors.

---

**FORMAT** IDL> distance=angulardistance(**V**, **W**,  
[/HELP])

---

## QUALIFIERS

V	3D-vector (of shape (3) or (1,3)) or list of n 3D-vectors (of shape (n,3))
W	3D-vector (of shape (3) or (1,3)) or list of n 3D-vectors (of shape (n,3))

It is **not** necessary for **V** and **W** vectors to be normalised to 1 upon calling the function

If **V** (and/or **W**) has the form (n,3,4) (like the pixel *corners* returned by `pix2vec_*`), it should be preprocessed with `V = reform( transpose(V, [0,2,1]), n_elements(V)/3, 3)` before being passed to `angulardistance`.

---

## KEYWORDS

/HELP	if set, prints out the help header and exits
-------	--

---

**DESCRIPTION** After renormalizing the vectors, `angulardistance` computes the angular distance using  $\cos^{-1}(\mathbf{V} \cdot \mathbf{W})$  in general, or  $2 \sin^{-1}(\|\mathbf{V} - \mathbf{W}\|/2)$  when **V** and **W** are almost aligned.

If **V** (resp. **W**) is a single vector, while **W** (resp. **V**) is a list of vectors, then the result is a list of distances  $d_i = \text{dist}(\mathbf{V}, \mathbf{W}_i)$  (resp.  $d_i = \text{dist}(\mathbf{V}_i, \mathbf{W})$ ).

If both **V** and **W** are lists of vector *of the same length*, then the result is a list of distances  $d_i = \text{dist}(\mathbf{V}_i, \mathbf{W}_i)$ .

---

## RELATED ROUTINES

This section lists the routines related to **angulardistance**.

idl            version 6.4 or more is necessary to run angulardis-  
tance.

---

## EXAMPLE:

```
nside=8  
pix2vec_ring, nside, lindgen(nside2npix(nside)), vpix  
mollview, angulardistance( vpix, [1,1,1])
```

will plot the angular distance between the Healpix pixels center  
for  $N_{\text{side}} = 8$  , and the vector  $(x, y, z) = (1, 1, 1)/\sqrt{3}$

## azeqview

---

Location in HEALPix directory tree: `src/idl/visu/azeqview.pro`

This IDL facility provides a means to visualise an azimuthal equidistant projection of **HEALPix** and COBE Quad-Cube maps in an IDL environment. It also offers the possibility to generate GIF, JPEG, PDF, PNG and Postscript color-coded images of the projected map. The projected (but not color-coded) data can also be output in FITS files and IDL arrays.

### FORMAT

```
IDL> AZEQVIEW,      File [, Select] [, AS-
    INH=,  BAD_COLOR=,  BG_COLOR=,  CHARSIZE=,
    CHARTHICK=, COLT=, COORD=, /CROP, EXECUTE=,
    FACTOR=, FG_COLOR=, FITS=, /FLIP, GAL_CUT=,
    GIF=, GLSIZE=, GRATICULE=, /HALF_SKY, HBOUND=,
    /HELP, /HIST_EQUAL, HXSIZE=, IGLSIZE=, IGRATIC-
    ULE=, JPEG=, LATEX=, /LOG, MAP_OUT=, MAX=,
    MIN=,  /NESTED,  /NO_DIPOLE,  /NO_MONOPOLE,
    /NOBAR, /NOLABELS, /NOPOSITION, OFFSET=, OUT-
    LINE=, PDF=, PFonts=, PNG=, POLARIZATION=,
    /PREVIEW, PS=, PXSIZ=, PYSIZ=, RESO_ARCMIN=,
    RETAIN=, ROT=, /SAVE, /SHADED, /SILENT, STAG-
    GER=, SUBTITLE=, TITLEPLOT=, TRANSPARENT=,
    TRUECOLORS=, UNITS=, WINDOW=, XPOS=, YPOS=]
```

### QUALIFIERS

For a full list of qualifiers see [mollview](#)

### KEYWORDS



For a full list of keywords see [mollview](#)

---

**DESCRIPTION**    azeqview reads in a **HEALPix** sky map in FITS format and generates an azimuthal equidistant projection of it, that can be visualized on the screen or exported in a GIF, JPEG, PNG, PDF or Postscript file. azeqview allows the selection of the coordinate system, map size, color table, color bar inclusion, linear, log, hybrid or histogram equalised color scaling, maximum and minimum range for the plot, plot-title *etc.* It also allows the representation of the polarization field.

---

## RELATED ROUTINES

This section lists the routines related to **azeqview**.

	see <a href="#">mollview</a>
<a href="#">hpx2dm</a>	turns Healpix maps into DomeMaster images using azeqview.

# beam2bl

---

**Location in HEALPix directory tree:** `src/idl/misc/beam2bl.pro`

This IDL facility computes a transfer (or window) function  $b(l)$  for a circular beam profile  $b(\theta)$ .

---

**FORMAT**            IDL> bl=beam2bl( beam, theta, lmax, [/ARCMIN , /DEGREES, /HELP, /RADIANS])

---

## QUALIFIERS

beam	input beam profile $b(\theta)$
theta	angles $\theta$ (in arcmin, degrees or radians) at which the input beam $b(\theta)$ is defined
lmax	maximum multipole on which the output $b(l)$ is to be computed

---

## KEYWORDS

/ARCMIN	if set, $\theta$ is in arcmin
/DEGREES	if set, $\theta$ is in degrees
/HELP	if set, prints out the help header and exits
/RADIANS	if set, $\theta$ is in radians

---

**DESCRIPTION** Since the SH Transform of an arbitrary beam is

$$b_{lm} = \int d\mathbf{r} \, b(\mathbf{r}) \, Y_{lm}^*(\mathbf{r}) \quad (1)$$

then, for a circular beam

$$\begin{aligned} b(l) &= b_{l0} \sqrt{\frac{4\pi}{2l+1}} \\ &= \int b(\theta) P_l(\theta) \sin(\theta) \, d\theta \, 2\pi \end{aligned} \quad (2)$$

where  $P_l$  is the Legendre Polynomial,  $b(l)$  is the beam window (or transfer) function returned by beam2bl and  $b(\theta)$  is the beam radial profile expected as input of beam2bl.

IDL's routine INT\_TABULATED is used to perform the integration.

---

## RELATED ROUTINES

This section lists the routines related to **beam2bl**.

idl	version 6.4 or more is necessary to run beam2bl.
<b>bl2beam</b>	facility to perform the inverse transform to beam2bl.
<b>bl2fits</b>	facility to write a $b(l)$ window function into a FITS file.
<b>fits2cl</b>	facility to read a $b(l)$ window function from a FITS file

---

## EXAMPLE:

```
bl = gaussbeam(15.d0, 4000, 1)
theta = dindgen(4000)/100.
beam = bl2beam(bl, theta, /arcmin)
bl1 = beam2bl(beam, theta, 4000, /arcmin)
plot, bl1-bl
```

the example above generates a beam window function (defined for all  $l$  in  $\{0, \dots, 4000\}$ ) for a 15arcmin-FWHM gaussian beam, computes the beam profile for angles in  $[0, 40]$  arcmin, computes back the beam window function from the beam profile and finally plots the difference between the beam window functions.

# bin\_llcl

Location in HEALPix directory tree: `src/idl/misc/bin_llcl.pro`

This IDL facility provides a means to bin an angular power spectrum into arbitrary bins.

---

**FORMAT** IDL> BIN\_LLCL, *Llcl\_in*, *Bin*, *L\_out*, *Llcl\_out*,  
*[Dllcl, DELTAL=, /FLATTEN, /HELP,*  
*/UNIFORM]*

---

## QUALIFIERS

<i>Llcl_in</i>	1D vector: <b>input</b> power spectrum (given for each $l$ starting at 0).
<i>Bin</i>	<b>input</b> : binning in $l$ to be applied, –either a scalar interpreted as the step size of a regular binning, the first bins are then $\{0, \text{bin} - 1\}, \{\text{bin}, 2\text{bin} - 1\}, \dots$ –or a 1D vector, interpreted as the lower bound of each bin, ie the first bins are $\{\text{bin}[0], \text{bin}[1] - 1\}, \{\text{bin}[1], \text{bin}[2] - 1\}, \dots$
<i>L_out</i>	contains on <b>output</b> the center of each bin $l_b$ .
<i>Llcl_out</i>	contains on <b>output</b> the binned power spectrum $C(b)$ , ie the (weighted) average of the input $C(l)$ over each bin.
<i>Dllcl</i>	<b>optional</b> , contains on <b>output</b> a rough estimate of the rms of the binned $C(l)$ for a full sky observation $C(b)\sqrt{2/((2l_b + 1)\Delta l_b)}$
<i>DELTAL=</i>	<b>optional</b> , contains on <b>output</b> the size of each bin $\Delta l(b)$

---

## KEYWORDS

*/FLATTEN* if set, the  $C(l)$  is internally multiplied by  $l(l + 1)/2\pi$  before being binned.  
 By default, the input *Llcl\_in* is binned as is.

---

/HELP	if set, an extended help is printed and the code exits.
/UNIFORM	if set, the $C(l)$ in each bin is given the same weight. By default a weight $\propto 2l + 1$ is used (inverse cosmic variance weighting). Note that this weighting affects <code>Llcl_out</code> but not <code>L_out</code> .

---

**DESCRIPTION** `bin_llcl` bins the input power spectrum (as is, or after flattening by a  $l(l+1)/2\pi$  factor) according to an arbitrary binning scheme defined by the user. Different weighting scheme (uniform or inverse variance) can be applied inside the bins.

---

## RELATED ROUTINES

This section lists the routines related to `bin_llcl`.

<code>idl</code>	version 6.4 or more is necessary to run <code>bin_llcl</code> .
<code>fits2cl</code>	facility to read a power spectrum from a FITS file.

---

## EXAMPLE:

```
init_healpix
fits2cl, cl, !healpix.directory+'/test/cl.fits', multipoles=1
f1 = 1*(l+1) / (2. * !pi)
bin_llcl, f1*cl[:,0], 10, lb, bbcb, /uniform
plot, l, f1*cl[:,0]
oplot, lb, bbcb, psym = 4
```

Read a power spectrum, bin it with a binsize of 10 and a uniform weighting, and overplot the input spectrum and its binned version.

Location in HEALPix directory tree: src/idl/misc/bl2beam.pro

```
FORMAT      IDL> beam=bl2beam( bl, theta, [/ARCMIN ,  
                                /DEGREES, /HELP, /RADIANS])
```

bl input  $b(l)$  window function of beam (defined for all integer multipoles  $l$  starting at 0)

theta angles  $\theta$  (in arcmin, degrees or radians) at which the output beam  $b(\theta)$  is to be computed.

/ARCMIN if set,  $\theta$  is in arcmin/DEGREES      if set,  $\theta$  is in degrees

/HELP if set, prints out the help header and exits

/RADIANS if set,  $\theta$  is in radians

**DESCRIPTION** Since an arbitrary beam is related to its SH Transform via

$$b(\mathbf{r}) = \sum_{lm} b_{lm} Y_{lm}(\mathbf{r}), \quad (3)$$

a circular beam has a radial profile (as returned by bl2beam)

$$b(\theta) = \sum_l b(l) P_l(\theta) \frac{2l+1}{4\pi}, \quad (4)$$

where  $P_l$  is Legendre Polynomial and

$$b(l) = b_{l0} \sqrt{\frac{4\pi}{2l+1}} \quad (5)$$

is the beam window (or transfer) function, expected as input to bl2beam.

---

## RELATED ROUTINES

This section lists the routines related to **bl2beam**.

idl	version 6.4 or more is necessary to run bl2beam.
<b>beam2bl</b>	facility to perform the inverse transform to bl2beam.
<b>bl2fits</b>	facility to write a $b(l)$ window function into a FITS file.
<b>fits2cl</b>	facility to read a $b(l)$ window function from a FITS file

---

## EXAMPLE:

```
bl = gaussbeam(15.d0, 4000, 1)
theta = dindgen(3000)/100.
beam = bl2beam(bl, theta, /arcmin)
plot, theta, beam
```

the example above generates a beam window function (defined for all  $l$  in  $\{0, \dots, 4000\}$ ) for a 15arcmin-FWHM gaussian beam, computes the beam profile for angles in  $[0, 30]$  arcmin and then plots it.

## bl2fits

---

Location in HEALPix directory tree: `src/idl/fits/bl2fits.pro`

This IDL facility provides a means to write into a FITS file as an ascii table extension a (beam) window function  $W(\ell)$  or  $W(\ell)$ . Adds additional headers if required. The facility is primarily intended to allow the user to write an arbitrary window function into a FITS file in the correct format to be ingested by the HEALPix simulation facility **synfast**.

---

**FORMAT** IDL> BL2FITS, **bl\_array**, **fitsfile**, [HDR = ,  
/HELP, XHDR =]

---

## QUALIFIERS

<b>bl_array</b>	real or double array of Bl coefficients to be written to file. This has dimension (lmax+1, <i>n</i> ) with $1 \leq n \leq 3$ , given in the sequence T E B.
<b>fitsfile</b>	String containing the name of the file to be written.

---

## KEYWORDS

<b>HDR=</b>	String array containing the (non-trivial) primary header for the FITS file.
<b>/HELP</b>	If set, a help message is printed out, no file is written
<b>XHDR=</b>	String array containing the (non-trivial) extension header for the FITS file.

---



---

**DESCRIPTION** bl2fits writes the input  $B(\ell)$  or  $W(\ell)$  coefficients into a FITS file containing an ascii table extension. Optional headers conforming to the FITS convention can also be written to the output file. All required FITS header keywords (like SIMPLE, BITPIX, ...) are automatically generated by the routine and should NOT be duplicated in the optional header inputs (they would be ignored anyway). The one/two/three column(s) are automatically named **TEMPERATURE**, **GRAD**, **CURL** respectively. If the window function is provided in a double precision array, the output format will automatically feature more decimal places.

---

## RELATED ROUTINES

This section lists the routines related to **bl2fits**.

idl	version 6.4 or more is necessary to run bl2fits.
<b>fits2cl</b>	provides the complimentary routine to read in a window function or power spectrum from a FITS file.
synfast	utilises the output file generated by bl2fits(option <b>beam_file</b> ).

---

## EXAMPLE:

```
beam1 = gaussbeam(10., 2000, 1)
beam2 = gaussbeam(15., 2000, 1)
beam = (beam1 + beam2) / 2.
bl2fits, beam, 'beam.fits'
```

bl2fits writes the beam window function stored in the variable **beam** (=Legendre transform of a circular beam) into the output FITS file **beam.fits**.

## cartcursor

---

Location in HEALPix directory tree: `src/idl/visu/cartcursor.pro`

This IDL facility provides a point-and-click interface for finding the astronomical location, value and pixel index of the pixels nearest to the pointed position on a cartesian projection of a **HEALPix** map.

---

**FORMAT**            IDL>    `CARTCURSOR,`    [`cursor_type=`,  
                         `file_out=`]

---

## QUALIFIERS

see [mollcursor](#)

---

**DESCRIPTION** `cartcursor` should be called immediately after `cartview`. It gives the longitude, latitude, map value and pixel number corresponding to the cursor position in the window containing the map generated by `orthview`. For more details, or in case of problems under **Mac OS X**, see [mollcursor](#).

---

## RELATED ROUTINES

This section lists the routines related to **cartcursor**.

see [mollcursor](#)

---

## EXAMPLE:

`cartcursor`

After `cartview` has read in a map and generated its cartesian projection, `cartcursor` is run to determine the position and flux of bright synchrotron sources, for example.

# cartview

Location in HEALPix directory tree: `src/idl/visu/cartview.pro`

This IDL facility provides a means to visualise a cartesian projection (where the longitude and latitude are treated as the cartesian abscissa and ordinate) of **HEALPix** and COBE Quad-Cube maps in an IDL environment. It also offers the possibility to generate GIF, JPEG, PDF, PNG and Postscript color-coded images of the projected map. The projected (but not color-coded) data can also be output in FITS files and IDL arrays.

## FORMAT

```
IDL> CARTVIEW, File [, Select] [, AS-
    INH=, BAD.COLOR=, BG.COLOR=, CHARSIZE=,
    CHARTHICK=, COLT=, COORD=, /CROP, EXECUTE=,
    FACTOR=, FG.COLOR=, FITS=, /FLIP, GAL_CUT=,
    GIF=, GLSIZE=, GRATICULE=, /HALF_SKY, HBOUND=,
    /HELP, /HIST_EQUAL, HXSIZE=, IGLSIZE=, IGRATIC-
    ULE=, JPEG=, LATEX=, /LOG, MAP_OUT=, MAX=,
    MIN=, /NESTED, /NO_DIPOLE, /NO_MONOPOLE,
    /NOBAR, /NOLABELS, /NOPOSITION, OFFSET=, OUT-
    LINE=, PDF=, PFonts=, PNG=, POLARIZATION=,
    /PREVIEW, PS=, PXSIZ=, PYSIZ=, RESO_ARCMIN=,
    RETAIN=, ROT=, /SAVE, /SHADED, /SILENT, STAG-
    GER=, SUBTITLE=, TITLEPLOT=, TRANSPARENT=,
    TRUECOLORS=, UNITS=, WINDOW=, XPOS=, YPOS=]
```

## QUALIFIERS

For a full list of qualifiers see [mollview](#)

---

## KEYWORDS

For a full list of keywords see [mollview](#)

---

## DESCRIPTION

`cartview` reads in a **HEALPix** sky map in FITS format and generates a cartesian projection of it, that can be visualized on the screen or exported in a GIF, JPEG, PNG, PDF or Postscript file. `cartview` allows the selection of the coordinate system, map size, color table, color bar inclusion, linear, log, hybrid or histogram equalised color scaling, maximum and minimum range for the plot, plot-title *etc.* It also allows the representation of the polarization field.

---

## RELATED ROUTINES

This section lists the routines related to **cartview**.

see [mollview](#)

---

## EXAMPLE:

```
map = findgen(48)
triangle= create_struct('coord','G','ra',[0,80,0],'dec',[40,45,65])
cartview,map,/online,res=45,graticule=[45,30],rot=[10,20,30],pysize=250,$
    title='Cartesian cylindrical (full sky)',subtitle='cartview', $
    outline=triangle
```

makes a cartesian cylindrical projection of map (see Figure [1a](#) on page [117](#)) after an arbitrary rotation, with a graticule grid (with a  $45^\circ$  step in longitude and  $30^\circ$  in latitude) and an arbitrary triangular outline

# change\_polconv

Location in HEALPix directory tree: `src/idl/fits/change_polconv.pro`

This IDL facility changes the coordinate convention in FITS file containing a polarised sky map. The main effect is to change the sign of the U Stokes parameter, and add/update the POLCCONV FITS header with either COSMO or IAU value.

---

**FORMAT**            IDL> `CHANGE_POLCCONV` , `File_In`,  
                          `File_Out` [, `/I2C`, `/C2I`, `/C2C`, `/I2I`, `/FORCE`]

---

## QUALIFIERS

<code>File_In</code>	name of a FITS file to be read
<code>File_Out</code>	name of a FITS file to be written, after modification of the polarisation coordinate convention, if applicable

---

## KEYWORDS

<code>/I2C</code>	changes from IAU to COSMO coordinate convention -if POLCCONV is not found or found with value 'IAU', it is added/replaced with value 'COSMO', and the sign of the U stokes parameter map is changed -if POLCCONV already has value 'COSMO', <code>File_In</code> is copied unchanged into <code>File_Out</code>
<code>/C2I</code>	changes from COSMO to IAU coordinate convention -if POLCCONV is not found or found with value 'COSMO', it is added/replaced with value 'IAU', and the sign of the U stokes parameter map is changed -if POLCCONV already has value 'IAU', <code>File_In</code> is copied unchanged into <code>File_Out</code>
<code>/C2C</code>	does NOT change coordinate system -if POLCCONV is found with value 'IAU', pro-

	gram will issue error message and no file is written
	-in all other case POLCCONV is set/added with value 'COSMO', but data is NOT changed
/I2I	does NOT change coordinate system
	-if POLCCONV is found with value 'COSMO', program will issue error message and no file is written
	-in all other case POLCCONV is set/added with value 'IAU', but data is NOT changed
/FORCE	if set, the value of POLCCONV read from the FITS header is ignored. The sign of U is swapped (if used with /C2I or /I2C), and the FITS keyword is updated accordingly.

---

**DESCRIPTION** This routine will change the sign of the  $U$  Stokes parameters (and related quantities, such as the  $TU$  and  $QU$  cross-correlations) and update the 'POLCCONV' FITS keyword where applicable. The recognised format are:

- standard Healpix full sky polarised format
- cut sky Healpix polarised format
- WMAP 2nd year polarised format

---

## RELATED ROUTINES

This section lists the routines related to **change\_polconv** .

idl	version 6.4 or more is necessary to run change_polconv
<b>write_fits_cut4</b>	This <b>HEALPix</b> IDL facility can be used to write a (polarised or unpolarised) cut sky map into a FITS file.
<b>read_fits_cut4</b>	This <b>HEALPix</b> IDL facility can be used to read a (polarised or unpolarised) cut sky map from a FITS file.
<b>write_tqu</b>	This <b>HEALPix</b> IDL facility can be used to write a polarised full sky map (with either the standard Healpix format or the WMAP 2nd year format) into a FITS file
<b>read_tqu</b>	This <b>HEALPix</b> IDL facility can be used to read

a polarised cut sky map from a FITS file

---

**EXAMPLE:**

```
change_polconv, 'map_cosmo.fits', 'map_iau.fits', /c2i
```

Modify the file 'map\_cosmo.fits', which was using the 'COSMO' convention for polarisation coordinate convention into 'map\_iau.fits' which uses the 'IAU' convention

## cl2fits

**Location in HEALPix directory tree:** `src/idl/fits/cl2fits.pro`

This IDL facility provides a means to write into a FITS file as an ascii table extension the power spectrum coefficients passed to the routine. Adds additional headers if required. The facility is primarily intended to allow the user to write a theoretical power spectrum into a FITS file in the correct format to be ingested by the **HEALPix** simulation facility **synfast**.

---

**FORMAT**            IDL> CL2FITS, `cl_array`, `fitsfile`, [`HDR=`,  
                         `/HELP`, `XHDR=`, `/CMBFAST`, `UNITS=`]

---

## QUALIFIERS

<code>cl_array</code>	real or double array of Cl coefficients to be written to file. This has dimension either (lmax+1,9) given in the sequence T E B TxE TxB ExB ExT BxT BxE <b>or</b> (lmax+1,6) given in the sequence T E B TxE TxB ExB <b>or</b> (lmax+1,4) given in the sequence T E B TxE <b>or</b> (lmax+1) for T alone. The convention for the power spectrum is that it is not normalised by the Harrison-Zeldovich (flat) spectrum.
<code>fitsfile</code>	String containing the name of the file to be written.

---

## KEYWORDS

<code>HDR=</code>	String array containing the (non-trivial) primary header for the FITS file.
<code>/HELP</code>	If set, a help message is printed out, no file is written
<code>XHDR=</code>	String array containing the (non-trivial) extension header for the FITS file.
<code>/CMBFAST</code>	if set, the routine will add the keyword 'POL-NORM = CMBFAST' in the FITS header, meaning that the polarization power spectra have the



same convention as CMBFAST (and Healpix 1.2). If this keyword is not present in the input FITS file, `synfast` will issue a warning when simulating a polarization map from that power spectrum, but no attempt to renormalize the power spectra will be made. To actually perform the renormalization, see `convert_oldhpx2cmbfast`

**UNITS=** String scalar containing units of power spectrum (eg,  $\text{uK}^2$ ,  $\text{Kelvin}^{**2}$ , ...), to be put in keywords 'TUNIT\*' of the extension header. If provided, will override the values present in XHDR (if any).  
NOTE: optional header strings should NOT include the header keywords explicitly written by this routine.

---

**DESCRIPTION** `cl2fits` writes the input power spectrum coefficients into a FITS file containing an ascii table extension. Optional headers conforming to the FITS convention can also be written to the output file. All required FITS header keywords (like SIMPLE, BITPIX, ...) are automatically generated by the routine and should NOT be duplicated in the optional header inputs (they would be ignored anyway). The one/four/six/nine column(s) are automatically named TEMPERATURE, GRAD, CURL, TG, TC, GC, GT, CT and CG respectively. If the power spectrum is provided in a double precision array, the output format will automatically feature more decimal places. The current implementation is much faster than the one available in Healpix 1.10 thanks to replacing an internal loop by vector operations.

---

## RELATED ROUTINES

This section lists the routines related to `cl2fits`.

<code>idl</code>	version 6.4 or more is necessary to run <code>cl2fits</code> .
<code>fits2cl</code>	provides the complimentary routine to read in a power spectrum from a FITS file.
<code>convert_oldhpx2cmbfast</code>	convert an existing power spectrum FITS file from the polarization convention used in Healpix 1.1 to the one used in Healpix 1.2 (and CMBFAST).

<code>bl2fits</code>	facility to write a window function into a FITS file.
<code>fits2alm</code> , <code>alm2fits</code>	routines to read and write $a_{lm}$ coefficients
<code>synfast</code>	utilises the output file generated by <code>cl2fits</code> .

---

**EXAMPLE:**

```
cl2fits, pwrsp, 'spectrum.fits', HDR = hdr, XHDR = xhdr
```

`cl2fits` writes the power spectrum stored in the variable `pwrsp` to the output FITS file `spectrum.fits` with optional headers passed by the string variables `hdr` and `xhdr`.

# convert\_oldhpx2cmbfast

**Location in HEALPix directory tree:** src/idl/fits/convert\_oldhpx2cmbfast.pro

This IDL facility provides a means to change the normalization of polarization power spectra in a FITS file from Healpix 1.1 convention to Healpix 1.2 (which is the same as CMBFAST).

---

**FORMAT** IDL> CONVERT\_OLDHPX2CMBFAST,  
file\_in, [file\_out, NO\_RENORM=]

---

## QUALIFIERS

file_in	String containing the name of the FITS file with the power spectra to be read.
file_out	(OPTIONAL) String containing the name of the file to be written after renormalization. If absent, file_in will be used for output

---

## KEYWORDS

NO_RENORM=	if set, the renormalization is not done. but the keyword POLNORM = CMBFAST is added to the FITS header (useful if the FITS file is already in CMBFAST format).
------------	--

---

**DESCRIPTION** convert\_oldhpx2cmbfast does the conversion from the polarization normalisation used in **HEALPix** 1.1 to the one used in **HEALPix** 1.2 (see the [Healpix primer document](#)). A keyword POLNORM = CMBFAST is added to the header to keep track of which files have been renormalized. If this keyword is not present in the input FITS file, **synfast** will issue a warning when simulating a polarization map from that power spectrum, but no attempt to renormalize the power spectra will be made.

---

## RELATED ROUTINES

This section lists the routines related to **convert\_oldhpx2cmbfast**.

idl	version 6.4 or more is necessary to run convert_oldhpx2cmbfast.
cl2fits	provides the a routine to write a power spectrum to a FITS file.
fits2cl	provides the complimentary routine to read in a power spectrum from a FITS file.
synfast	utilises the output file generated by convert_oldhpx2cmbfast.

---

### EXAMPLE:

```
convert_oldhpx2cmbfast, 'cl_flat.fits'
```

convert\_oldhpx2cmbfast will renormalize the polarization power spectra read from 'cl\_flat.fits', and write them in the same file.

# euler\_matrix\_new

Location in HEALPix directory tree: `src/idl/misc/euler_matrix_new.pro`

This IDL facility provides a means to generate a 3D rotation Euler matrix parametrized by three angles and three axes of rotation.

---

**FORMAT** IDL> `matrix = EULER_MATRIX_NEW(a1, a2, a3 [,DEG=, HELP=, X=, Y=, ZYX=])`

---

## QUALIFIERS

<code>matrix</code>	a 3x3 array containing the Euler matrix
<code>a1</code>	input, float scalar, angle of the first rotation, expressed in radians, unless DEG (see below) is set
<code>a2</code>	angle of the second rotation, same units as a1
<code>a3</code>	angle of the third rotation, same units as a1

---

## KEYWORDS

<code>DEG=</code>	if set, the angles are in degrees instead of radians
<code>HELP=</code>	if set, the routine prints its documentation header and exits
<code>X=</code>	if set, uses the classical mechanics convention (ZXZ): rotation a1 around original Z axis, rotation a2 around intermediate X axis, rotation a3 around final Z axis (see Goldstein for more details). ( <b>default:</b> this convention is used)
<code>Y=</code>	if set, uses the quantum mechanics convention (YZY): rotation a1 around original Z axis, rotation a2 around intermediate Y axis, rotation a3 around final Z axis.
<code>ZYX=</code>	if set, uses the aeronautics convention (ZYX): rotation a1 around original Z axis,

rotation a2 around intermediate Y axis,  
rotation a3 around final X axis.

---

**DESCRIPTION** `euler_matrix_new` allows the generation of a rotation Euler matrix. The user can choose the three Euler angles, and the three axes of rotation.  
If `vec` is an  $N \times 3$  array containing  $N$  3D vectors,  
`vecr = vec # euler_matrix_new(a1,a2,a3,/Y)`  
will be the rotated vectors

This routine supersedes `euler_matrix`, which had inconsistent angle definitions. The relation between the two routines is as follows :

$$\text{euler\_matrix\_new}(a,b,c,/X) = \text{euler\_matrix}(-a,-b,-c,/X) \\ = \text{Transpose}(\text{euler\_matrix}(c, b, a,/X))$$

$$\text{euler\_matrix\_new}(a,b,c,/Y) = \text{euler\_matrix}(-a, b,-c,/Y) \\ = \text{Transpose}(\text{euler\_matrix}(c,-b, a,/Y))$$

$$\text{euler\_matrix\_new}(a,b,c,/Z) = \text{euler\_matrix}(-a, b,-c,/Z)$$


---

## RELATED ROUTINES

This section lists the routines related to **`euler_matrix_new`**.

idl	version 6.4 or more is necessary to run <code>euler_matrix_new</code> .
<b>rotate_coord</b>	apply a rotation to a set of position vectors and polarization Stokes parameters.

# fits2alm

**Location in HEALPix directory tree: src/idl/fits/fits2alm.pro**

This IDL routine provides a means to read from a FITS file binary table extension(s) containing spherical harmonic coefficients  $a_{\ell m}$  (and optional errors) and their index. Reads header information if required. The facility is intended to enable the user to read the output from the **HEALPix** facilities **anafast** and **synfast**.

---

**FORMAT** IDL> FITS2ALM, index, alm\_array, fitsfile, [signal, /HELP, HDR=, LMAX=, LMIN=, XHDR= ]

---

## QUALIFIERS

index	Long array containing the index for the corresponding array of $a_{\ell m}$ coefficients (and errors if required). The index $i$ is related to $(l, m)$ by the relation $i = \ell^2 + \ell + m + 1.$ This has dimension nl (see below).
alm_array	Real or double array of alm coefficients read from the file. This has dimension (nl,nalm,nsig) – corresponding to nl = number of $(l, m)$ indices nalm = 2 for real and imaginary parts of alm coefficients or 4 for above plus corresponding error values nsig = number of signals to be written (1 for any of T E B or 3 if ALL to be written). Each signal is stored in a separate extension.
fitsfile	String containing the name of the file to be read.
signal	String defining the signal coefficients to read Valid options: 'T', 'E', 'B' or 'ALL' (default: 'T').

---

## KEYWORDS

HDR=	String array containing the primary header read from the FITS file.
/HELP	If set, the routine documentation header is shown and the routine exits
LMAX=	Largest $l$ multipole to be output
LMIN=	Smallest $l$ multipole to be output. If LMIN (resp. LMAX) is below (above) the range of $l$ 's present in the file, it will be silently ignored
XHDR=	String array containing the read extension header(s). If ALL signals are required, then the three extension headers are returned appended into one string array.

---

**DESCRIPTION** `fits2alm` reads binary table extension(s) which contain the  $a_{\ell m}$  coefficients (and associated errors if present) from a FITS file. FITS headers can also optionally be read from the input file.

---

## RELATED ROUTINES

This section lists the routines related to **fits2alm**.

idl	version 6.4 or more is necessary to run <code>fits2alm</code> .
<code>alm2fits</code>	provides the complimentary routine to write $a_{\ell m}$ coefficients into a FITS file.
<code>alm_i2t</code> , <code>alm_t2i</code>	these facilities turn indexed lists of $a_{\ell m}$ into 2D $a(l,m)$ tables and back
<code>index2lm</code>	converts the index $i = \ell^2 + \ell + m + 1$ returned by <code>fits2alm</code> into $\ell$ and $m$
<code>lm2index</code>	converts $(\ell, m)$ vectors into $i = \ell^2 + \ell + m + 1$
<code>fits2cl</code>	routine to read/compute $C(l)$ power spectra from a file containing $C(l)$ or $a_{\ell m}$ coefficients
<code>ianafast</code> , <code>isynfast</code>	IDL routine providing $a_{\ell m}$ coefficients file to be read by <code>fits2alm</code> .
<code>alteralm</code> , <code>anafast</code> , <code>synfast</code>	F90 facilities providing $a_{\ell m}$ coefficients file to be read by <code>fits2alm</code> .



---

**EXAMPLE:**

```
fits2alm, index, alm, 'alm.fits', HDR = hdr, XHDR = xhdr
```

fits2alm reads from the input FITS file `alm.fits` the  $a_{\ell m}$  coefficients into the variable `alm` with optional headers passed by the string variables `hdr` and `xhdr`. Upon return `index` will contain the value of  $\ell^2 + \ell + m + 1$  for each  $a_{\ell m}$  found in the file.

## fits2cl

**Location in HEALPix directory tree: src/idl/fits/fits2cl.pro**

This IDL facility provides a means to read from a FITS file an ascii or binary table extension containing power spectrum ( $C(l)$ ) or spherical harmonics ( $a_{lm}$ ) coefficients, and returns the corresponding power spectrum ( $C(l) = \sum_m a_{lm} a_{lm}^* / (2l + 1)$ ). Reads primary and extension headers if required. The facility is intended to enable the user to read the output from the HEALPix facility **anafast**.

---

**FORMAT** IDL> fits2cl, cl\_array, [fitsfile, EXTENSION=, HDR=, /HELP, /INTERACTIVE, LLFACTOR=, MULTIPOLES=, /PLANCK1=, /PLANCK2=, /RSHOW, /SHOW, /SILENT=, /WMAP1=, /WMAP5=, /WMAP7=, XHDR=]

---

## QUALIFIERS

cl_array	real array of $C_\ell$ coefficients read or computed from the file. The output dimension depends on the contents of the file. This has dimension either (lmax+1,9) given in the sequence T E B TxE TxB ExB ExT BxT BxE <b>or</b> (lmax+1,6) given in the sequence T E B TxE TxB ExB <b>or</b> (lmax+1,4) for T E B TxE <b>or</b> (lmax+1) for T alone. The convention for the power spectrum is that it is not normalised by the Harrison-Zeldovich (flat) spectrum.
fitsfile	String containing the name of the FITS file to be read. The file contains either $C(l)$ power spectra or $a_{lm}$ coefficients. In either cases, $C(l)$ is returned. If fitsfile is not set, then /PLANCK1, /PLANCK2, /WMAP1, /WMAP5 or /WMAP7 must be set.

---

## KEYWORDS

EXTENSION=	extension unit to be read from FITS file: either its 0-based ID number (ie, 0 for first extension <i>after</i> primary array) or the case-insensitive value of its EXTNAME keyword.
HDR =	String array containing on output the primary header read from the FITS file.
/HELP	If set, produces an extended help message (using the doc.library IDL command).
/INTERACTIVE	If set, the plots generated by <code>/SHOW</code> and <code>/RSHOW</code> options are produced using iPlot routine, allowing for interactive cropping, zooming and annotation of the plots. This requires IDL 6.4 or newer to work properly.
LLFACTOR =	vector containing on output the factor $l(l+1)/2\pi$ which is often applied to $C(l)$ to flatten it for plotting purposes
MULTIPOLES =	vector containing on output the multipoles $\ell$ for which the power spectra are provided. They are either <ul style="list-style-type: none"> <li>- read from the file (1st column in the Planck format),</li> <li>- or generated by the routine (assuming that all multipoles from 0 to lmax included are provided).</li> </ul>
/PLANCK1	If set, and <code>fitsfile</code> is not provided, then a Planck 2013+external data best fit model ( <code>!healpix.path.test+'planck2013ext_lcdm_cl_v1.fits'</code> which matches <code>!healpix.path.test+'cl_planck1.fits'</code> ) defined up to <code>lmax=4500</code> , is read. See <code>!healpix.path.test+'README'</code> for details
/PLANCK2	If set, and <code>fitsfile</code> is not provided, then a Planck 2015 data best fit model ( <code>!healpix.path.test+'planck2015_lcdm_cl_v2.fits'</code> which matches <code>!healpix.path.test+'cl_planck2.fits'</code> ) defined up to <code>lmax=4900</code> , is read. See <code>!healpix.path.test+'README'</code> for details
/RSHOW	If set, the raw power spectra $C(l)$ read from the file are plotted

/SHOW	If set, the rescaled power spectra $l(l+1)C(l)/2\pi$ are plotted
/SILENT	If set, no message is issued during normal execution
/WMAP1	If set, and <b>fitsfile</b> is not provided, then one WMAP-1yr best fit model (!healpix.path.test+'wmap_lcdm_pl_model_yr1_v1.fits' which currently matches !healpix.path.test+'cl.fits') defined up to $l_{\max}=3000$ , is read. See !healpix.path.test+'README' for details
/WMAP5	If set, and <b>fitsfile</b> is not provided, then one WMAP-5yr best fit model (!healpix.path.test+'wmap_lcdm_sz_lens_wmap5_cl_v3.fits' which matches !healpix.path.test+'cl_wmap5.fits') defined up to $l_{\max}=2000$ , is read. See !healpix.path.test+'README' for details
/WMAP7	If set, and <b>fitsfile</b> is not provided, then one WMAP-7yr best fit model (!healpix.path.test+'wmap_lcdm_sz_lens_wmap7_cl_v4.fits' which matches !healpix.path.test+'cl_wmap7.fits') defined up to $l_{\max}=3726$ , is read. <b>Note:</b> As opposed to the other WMAP spectra mentioned above, it includes a non-vanishing B (or CURL) power spectrum induced by lensing of E (or GRAD) polarization. See !healpix.path.test+'README' for details
XHDR =	String array containing on output the extension header read from the FITS file.

---

**DESCRIPTION** fits2cl reads the power spectrum coefficients from a FITS file containing an ascii table extension. Descriptive headers conforming to the FITS convention can also be read from the input file.

---

## RELATED ROUTINES

This section lists the routines related to **fits2cl**.

idl                      version 6.4 or more is necessary to run fits2cl.

---

<code>bin_llcl</code>	facility to bin a spectrum read with fits2cl.
<code>bl2fits</code>	facility to write a window function into a FITS file.
<code>cl2fits</code>	provides the complimentary routine to write a power spectrum to a FITS file.
<code>fits2alm</code> , <code>alm2fits</code>	routines to read and write $a_{lm}$ coefficients
<code>ianafast</code>	IDL routine computing $C(l)$ files that can be read by fits2cl.
<code>anafast</code>	F90 facility computing $C(l)$ files that can be read by fits2cl.

---

### EXAMPLE:

```
fits2cl, pwrsp, '$HEALPIX/test/cl.fits', $
        HDR=hdr, XHDR=xhdr, MULTI=1, LLFACT=fll
plot, l, powrsp[*,0]*fll
```

fits2cl reads a power spectrum  $C(l)$  from the input FITS file `$HEALPIX/test/cl.fits` into the variable `pwrsp`, with optional headers passed by the string variables `hdr` and `xhdr`. The multipoles  $l$  and factors  $l(l+1)/2\pi$  are read into `l` and `fll` respectively.  $l(l+1)C(l)/2\pi$  vs  $l$  is then plotted.

# gaussbeam

Location in HEALPix directory tree: `src/idl/misc/gaussbeam.pro`

This IDL facility provides the window function in  $\ell$  space for a gaussian axisymmetric beam of given FWHM.

---

**FORMAT** IDL> beam=GAUSSBEAM (Fwhm, Lmax[, Dim])

---

## QUALIFIERS

Fwhm	Full Width Half Maximum of the gaussian beam, in arcmin (scalar real)
Lmax	the window function is computed for the multipoles $\ell$ in $\{0, \dots, Lmax\}$
Dim	scalar integer, optional. If absent or set to 0 or 1, the output has size (Lmax+1) and is the temperature beam; if set to $2 \leq Dim \leq 4$ , the output has size (Lmax+1, Dim) and contains in that order : the TEMPERATURE beam, the GRAD/ELECTRIC polarization beam the CURL/MAGNETIC polarization beam the TEMPERATURE*GRAD beam

---

**DESCRIPTION** gaussbeam computes the  $\ell$  space window function of a gaussian beam of FWHM Fwhm. For a sky of underlying power spectrum  $C(\ell)$  observed with beam of given FWHM, the measured power spectrum will be  $C(\ell)_{meas} = C(\ell)B(\ell)^2$  where  $B(\ell)$  is given by gaussbeam(Fwhm, Lmax). The polarization beam is also provided (when Dim > 1) assuming a perfectly co-polarized beam (eg, Challinor et al 2000, astro-ph/0008228)

---

## RELATED ROUTINES

This section lists the routines related to **gaussbeam** .

---

idl	version 6.4 or more is necessary to run gaussbeam
healpixwindow	computes the $\ell$ space window function associated with a <b>HEALPix</b> pixel size
synfast	f90 code to generate CMB maps of given power spectrum convolved with a gaussian beam
smoothing	f90 code to smooth existing <b>HEALPix</b> maps with a gaussian beam
anafast	f90 code to compute the power spectrum of a <b>HEALPix</b> sky map

---

**EXAMPLE:**

```
beam = gaussbeam(5.,1200)
```

beam contains the window function in  $\{0,...,1200\}$  of a gaussian beam of fwhm 5 arcmin

## getdisc\_ring

---

Location in HEALPix directory tree: `src/idl/toolkit/getdisc_ring.pro`

This routine is obsolete. Use `query_disc` instead.



# getsize\_fits

**Location in HEALPix directory tree:** src/idl/fits/getsize\_fits.pro

This IDL function reads the number of maps and/or the pixel ordering of a FITS file containing a **HEALPix** map.

---

**FORMAT** IDL> var = GETSIZE\_FITS (File, [Nmaps=, Nside=, Mlpol=, Ordering=, Obs\_Npix=, Type=, Header=, Extension=, /Help])

---

## QUALIFIERS

File	name of a FITS file containing the <b>HEALPix</b> map(s).
var	contains on output the number of pixels stored in a map FITS file. Each pixel is counted only once (even if several information is stored on each of them, see nmaps). Depending on the data storage format, result may be : <ul style="list-style-type: none"> <li>– equal or smaller to the number Npix of Healpix pixels available over the sky for the given resolution (<math>N_{\text{pix}} = 12 * n_{\text{side}} * n_{\text{side}}</math>)</li> <li>– equal or larger to the number of non blank pixels (obs_npix)</li> </ul>
Nmaps=	contains on output the number of maps in the file
Nside=	contains on output the <b>HEALPix</b> resolution parameter $N_{\text{side}}$
Mlpol=	contains on output the maximum multipole used to generate the map
Ordering=	contains on output the pixel ordering scheme: either 'RING' or 'NESTED'
Obs_Npix=	contains on output the number of non blank pixels. It is set to -1 if it can not be determined from header
Type=	Healpix/FITS file type <0 : file not found, or not valid 0 : image only fits file, deprecated Healpix format ( $\text{var} = 12N_{\text{side}}^2$ ) 1 : ascii table, generally used for C(l) storage 2 : binary table : with implicit pixel indexing (full sky) ( $\text{var} = 12N_{\text{side}}^2$ ) 3 : binary table : with explicit pixel indexing (generally cut sky) ( $\text{var} \leq 12N_{\text{side}}^2$ ) 999 : unable to determine the type

Header=	contains on output the FITS extension header
Extension=	extension unit to be read from FITS file: either its 0-based ID number (ie, 0 for first extension <i>after</i> primary array) or the case-insensitive value of its EXTNAME keyword.

---

## KEYWORDS

HELP=	if set, an extensive help is displayed and no file is read
-------	--

---

**DESCRIPTION** `getsize_fits` gets the number of pixels in a FITS file. If the file follows the **HEALPix** standard, the routine can also get the resolution parameter `Nside`, the ordering scheme, ..., and can determine the type of data set contained in the file.

---

## RELATED ROUTINES

This section lists the routines related to `getsize_fits`.

idl	version 6.4 or more is necessary to run <code>getsize_fits</code>
<code>read_fits_map</code>	This <b>HEALPix</b> IDL facility can be used to read in maps written by <code>getsize_fits</code> .
<code>sxaddpar</code>	This IDL routine (included in <b>HEALPix</b> package) can be used to update or add FITS keywords to <code>Header</code>
<code>reorder</code>	This <b>HEALPix</b> IDL routine can be used to reorder a map from NESTED scheme to RING scheme and vice-versa.
<code>write_fits_sb</code>	routine to write multi-column binary FITS table

---

## EXAMPLE:

```
npix = getsize_fits(!healpix.directory+'/test/map.fits', nside=nside, $
    mlpol=lmax, type=filetype)
print, npix, nside, lmax, filetype
```

should produce something like

```
196608 128 256 2
```

meaning that the map contained in that file has 196608 pixels, the resolution parameter is `nside=128`, the maximum multipole was 256, and this a full sky map (type 2).

## gnomcursor

---

Location in HEALPix directory tree: `src/idl/visu/gnomcursor.pro`

This IDL facility provides a point-and-click interface for finding the astronomical location, value and pixel index of the pixels nearest to the pointed position on a gnomonic projection of a HEALPix map.

---

**FORMAT**            IDL>    GNOMCURSOR,    [`cursor_type=`,  
                         `file_out=`]

---

## QUALIFIERS

see `mollcursor`

---

**DESCRIPTION** `gnomcursor` should be called immediately after `gnomview`. It gives the longitude, latitude, map value and pixel number corresponding to the cursor position in the window containing the map generated by `gnomview`. For more details, or in case of problems under **Mac OS X**, see `mollcursor`.

---

## RELATED ROUTINES

This section lists the routines related to `gnomcursor`.

see `mollcursor`

---

## EXAMPLE:

`gnomcursor`

After `gnomview` has read in a map and generated its gnomonic projection, `gnomcursor` is run to determine the position and flux of bright synchrotron sources, for example.

# gnomview

Location in HEALPix directory tree: `src/idl/visu/gnomview.pro`

This IDL facility provides a means to visualise a Gnomonic projection (radial projection onto a tangent plane) of **HEALPix** and COBE Quad-Cube maps in an IDL environment. It also offers the possibility to generate GIF, JPEG, PDF, PNG and Postscript color-coded images of the projected map. The projected (but not color-coded) data can also be output in FITS files and IDL arrays.

## FORMAT

```
IDL> GNOMVIEW, File [, Select] [, AS-
    INH=, BAD.COLOR=, BG.COLOR=, CHARSIZE=,
    CHARTHICK=, COLT=, COORD=, /CROP, EXECUTE=,
    FACTOR=, FG.COLOR=, FITS=, /FLIP, GAL_CUT=,
    GIF=, GLSIZE=, GRATICULE=, /HALF_SKY, HBOUND=,
    /HELP, /HIST_EQUAL, HXSIZE=, IGLSIZE=, IGRATIC-
    ULE=, JPEG=, LATEX=, /LOG, MAP_OUT=, MAX=,
    MIN=, /NESTED, /NO_DIPOLE, /NO_MONOPOLE,
    /NOBAR, /NOLABELS, /NOPOSITION, OFFSET=, OUT-
    LINE=, PDF=, PFonts=, PNG=, POLARIZATION=,
    /PREVIEW, PS=, PXSIZ=, PYSIZ=, RESO_ARCMIN=,
    RETAIN=, ROT=, /SAVE, /SHADED, /SILENT, STAG-
    GER=, SUBTITLE=, TITLEPLOT=, TRANSPARENT=,
    TRUECOLORS=, UNITS=, WINDOW=, XPOS=, YPOS=]
```

## QUALIFIERS

For a full list of qualifiers see [mollview](#)

## KEYWORDS

For a full list of keywords see [mollview](#)

---

## DESCRIPTION

gnomview reads in a **HEALPix** sky map in FITS format and generates a gnomonic (or gnomonic) projection of it, that can be visualized on the screen or exported in a GIF, JPEG, PNG, PDF or Postscript file. gnomview allows the selection of the coordinate system, map size, color table, color bar inclusion, linear, log, hybrid or histogram equalised color scaling, maximum and minimum range for the plot, plot-title *etc.* It also allows the representation of the polarization field.

---

## RELATED ROUTINES

This section lists the routines related to **gnomview**.

see [mollview](#)

---

## EXAMPLES: #1

```
gnomview, 'planck100GHZ-LFI.fits', rot=[160,-30], reso_arcmin=2., $
  pxsize = 500., $
  title='Simulated Planck LFI Sky Map at 100GHz', $
  min=-100,max=100
```

gnomview reads in the map 'planck100GHZ-LFI.fits' and generates an output image of the size of 500×500 screen pixels, with a resolution of 2 arcmin/screen pixel at the center. The temperature scale has been set to lie between  $\pm 100$ , and the units will show as  $\mu\text{K}$ . The title 'Simulated Planck LFI Sky Map at 100GHz' has been appended to the image. The map is centered at ( $l = 160$ ,  $b = -30$ )

---

## EXAMPLES: #2

```
map = findgen(48)
triangle= create_struct('coord','G','ra',[0,80,0],'dec',[40,45,65])
gnomview,map,/online,res=25,graticule=[45,30],rot=[10,20,30],$
    title='Gnomic projection',subtitle='gnomview', $
    outline=triangle
```

makes a gnomonic projection of map (see Figure 1b on page 117) after an arbitrary rotation, with a graticule grid (with a  $45^\circ$  step in longitude and  $30^\circ$  in latitude) and an arbitrary triangular outline

# healpix\_doc

---

**Location in HEALPix directory tree:** `src/idl/misc/healpix_doc.pro`

This IDL facility displays HTML or PDF **HEALPix** documentation

---

**FORMAT**            IDL> healpix\_doc, [HTML= | PDF=] [,  
                              HELP=, WHOLE=]

---

## KEYWORDS

HELP=	if set, an extensive help on healpix_doc is displayed.
HTML=	if set, the <b>HEALPix</b> (IDL) HTML documentation is shown with a web browser. If the browser is already in use, a new tab is open.
PDF=	if set, the <b>HEALPix</b> (IDL) PDF documentation is shown with a pdf viewer. Either HTML or PDF must be set.
WHOLE=	if set, the whole <b>HEALPix</b> documentation is accessible, not just the IDL related part.

---

**DESCRIPTION** healpix\_doc calls `Online_help` to open either the HTML or PDF **HEALPix** documentation. The browser and viewer used are those found by the `$IDL_DIR/bin/online_help_html` and `$IDL_DIR/bin/online_help_pdf` scripts respectively. The content of the `!healpix` system variable is used to determine the documentation path.

---

## RELATED ROUTINES

This section lists the routines related to **healpix\_doc**.

idl	version 6.4 or more is necessary to run healpix_doc.
-----	--



**!HEALPIX**

IDL system variable used by healpix\_doc to locate the documentation.

---

## EXAMPLES: #1

```
healpix_doc, /html, /whole
```

will open the whole **HEALPix** HTML documentation in a web browser.

---

## EXAMPLES: #2

```
healpix_doc, /pdf
```

will open the IDL related **HEALPix** PDF documentation.

# healpixwindow

---

**Location in HEALPix directory tree:** `src/idl/misc/healpixwindow.pro`

This IDL facility provides the window function in  $\ell$  associated with the Healpix pixel of resolution Nside.

---

**FORMAT**            IDL> wpix=HEALPIXWINDOW (Nside [,   
Dim, Directory])

---

## QUALIFIERS

Nside	resolution parameter
Wpix	the pixel window function, computed for the multipoles $\ell$ in $\{0, \dots, 4N_{\text{side}}\}$
Dim	scalar integer, optional. If absent or set to 0 or 1, the output has size $(4N_{\text{side}}+1)$ and is the temperature window function; if set to $2 \leq \text{Dim} \leq 4$ , the output has size $(4N_{\text{side}}+1, \text{Dim})$ and contains in that order : the TEMPERATURE window function, the GRAD/ELECTRIC polarization one the CURL/MAGNETIC polarization one the TEMPERATURE*GRAD one.
Directory	directory in which the precomputed pixel window file is looked for. ( <b>default:</b> \$)HEALPIX/data/

---

**DESCRIPTION** `healpixwindow` computes the  $\ell$  space window function due to the finite size of the **HEALPix** pixels. The typical size of a pixel (square root of its uniform surface area) is  $\sqrt{3/\pi} \, 3600/N_{\text{side}}$  arcmin. If a unpixelated sky has a power spectrum  $C(\ell)$ , the same sky pixelated with a resolution parameter  $N_{\text{side}}$  will have the power spectrum  $C(\ell)_{\text{pix}} = C(\ell)W(\ell)^2$  where  $W(\ell)$  is given by `healpixwindow` ( $N_{\text{side}}$ ). The polarized pixel window function is also provided (when  $\text{Dim} > 1$ ). This routine reads some FITS files located in the subdirectory **data/** of the **HEALPix** distribution, unless the keyword **Directory** is set otherwise.

---

## RELATED ROUTINES

This section lists the routines related to **healpixwindow** .

idl	version 6.4 or more is necessary to run <code>healpixwindow</code>
<b>gaussbeam</b>	computes the $\ell$ space window function associated with a gaussian beam
synfast	f90 code to generate CMB maps of given power spectrum at a given resolution (=pixel size)
anafast	f90 code to compute the power spectrum of a <b>HEALPix</b> sky map

---

## EXAMPLE:

```
wpix = healpixwindow (256)
```

`wpix` contains the window function in  $\{0, \dots, 1024\}$  of the **HEALPix** pixel with resolution parameter 256 (pixel size of 13.7 arcmin)

## hpx2dm

---

Location in HEALPix directory tree: `src/idl/visu/hpx2dm.pro`

This IDL facility provides a means to turn a **HEALPix** data set into a DomeMaster compliant image (azimuthal equidistant projection of the half-sphere in a PNG or lossless JPEG file) that can be projected on a planetarium. See eg [http://fulldome.ryanwyatt.net/fulldome\\_domemasterSpec\\_v05.pdf](http://fulldome.ryanwyatt.net/fulldome_domemasterSpec_v05.pdf)

---

<b>FORMAT</b>	IDL> hpx2dm, <b>File</b> , [ <b>Select</b> , ] [ <b>/HELP</b> , <b>JPEG=</b> , <b>PNG=</b> , <b>PREVIEW=</b> , <b>PXSIZE=</b> , + most of azeqview keywords... ]
---------------	--

---

## QUALIFIERS

---

File	Required name of a FITS file containing the <b>HEALPix</b> map in an extension or in the image field, <i>or</i> name of an <i>online</i> variable (either array or structure) containing the <b>HEALPix</b> map (See note below); if Save is set : name of an IDL saveset file containing the <b>HEALPix</b> map stored under the variable <b>data</b> ( <b>default:</b> none)
Select	Optional column of the BIN FITS table to be plotted, can be either – a name : value given in TTYPEi of the FITS file NOT case sensitive and can be truncated, (only letters, digits and underscore are valid) – an integer : number i of the column containing the data, starting with 1 (also valid if <b>File</b> is an online array) ( <b>default:</b> 1 for full sky maps, 'SIGNAL' column for FITS files containing cut sky maps)

---

## KEYWORDS

---

JPEG=	name of the output <i>lossless</i> JPEG file
PNG=	name of the output PNG file
/PREVIEW	if set, the output JPEG or PNG file will be pre-viewed
/HELP	Prints out the documentation header
PXSIZE=	number of pixels in each dimension of the square output image
/ASINH,	
COLT=, COORD=, FACTOR=, /FLIP, HBOUND=,	
/HIST_EQUAL, /LOG, MAX=, MIN=, /NESTED, OFFSET=,	
/QUADCUBE, ROT=, SAVE=, /SILENT,	
TRUECOLORS=	those keywords have the same meaning as in <b>aze-view</b> and <b>mollview</b>

---

**DESCRIPTION** hpx2dm reads in a **HEALPix** sky map in FITS format or from a memory array and generates a PNG or JPEG file containing a DomeMaster compliant map (azimuthal equidistant projection of the half-sky).

---

## RELATED ROUTINES

This section lists the routines related to **hpx2dm**.

<b>azeqview</b>	performs Azimuthal Equidistant projection required by hpx2dm.
<b>hpx2gs</b>	turns Healpix maps into GoogleEarth or GoogleSky images

# hpx2gs

---

Location in HEALPix directory tree: `src/idl/visu/hpx2gs.pro`

This IDL facility provides a means to turn a **HEALPix** map into a image that can be visualized with [Google Earth](#) or [Google Sky](#).

---

<b>FORMAT</b>	IDL> hpx2gs, <b>File</b> , [ <b>Select</b> , ] [ <b>COORD_IN=</b> , <b>/HELP</b> , <b>KML=</b> , <b>PNG=</b> , <b>RESO_ARCMIN=</b> , <b>SUBTITLE=</b> , <b>TITLEPLOT=</b> ,+ most of cartview keywords... ]
---------------	--

---

## QUALIFIERS

---

File	<p>Required</p> <p>name of a FITS file containing the <b>HEALPix</b> map in an extension or in the image field,</p> <p><i>or</i> name of an <i>online</i> variable (either array or structure) containing the <b>HEALPix</b> map (See note below);</p> <p>if Save is set : name of an IDL saveset file containing the <b>HEALPix</b> map stored under the variable <b>data</b></p> <p>(<b>default:</b> none)</p>
Select	<p>Optional</p> <p>column of the BIN FITS table to be plotted, can be either</p> <ul style="list-style-type: none"> <li>– a name : value given in TTYPEi of the FITS file</li> </ul> <p>NOT case sensitive and can be truncated, (only letters, digits and underscore are valid)</p> <ul style="list-style-type: none"> <li>– an integer : number i of the column containing the data, starting with 1 (also valid if <b>File</b> is an online array)</li> </ul> <p>(<b>default:</b> 1 for full sky maps, 'SIGNAL' column for FITS files containing cut sky maps)</p>

---

## KEYWORDS

---

COORD.IN=	<p>1-character scalar, describing the input data coordinate system:</p> <p>either 'C' or 'Q' : Celestial2000 = eQuatorial, 'E' : Ecliptic, 'G' : Galactic.</p> <p>If set, it will over-ride the coordinates read from the FITS file header (when applicable). In absence of information, the input coordinates is assumed to be celestial.</p> <p>The data will be rotated so that the output coordinates are Celestial, as expected by Google Sky</p>
/HELP	Prints out the documentation header
KML=	<p>Name of the KML file to be created (if the <b>.kml</b> suffix is missing, it will be added automatically)</p> <p>(<b>default:</b> 'hpx2googlesky.kml')</p>
PNG=	<p>Name of the PNG overlay file to be created. Only to be used if you want the filename to be different from the default ((<b>default:</b> same as KML file, with a <b>.png</b> suffix instead of <b>.kml</b>))</p>
RESO_ARCMIN=	Pixel angular size in arcmin (at the equator) of



the cartesian map generated (**default:** 30)

SUBTITLE= information on the data, will appear in KML file  
GroundOverlay description field

TITLEPLOT= information on the data, will appear in KML file  
GroundOverlay name field

/ASINH,

COLT=, FACTOR=, /FLIP, GLSIZE=, GRATICULE=, HBOUND=,  
/HIST\_EQUAL, IGLSIZE=, IGRATICULE=, /LOG, MAX=, MIN=,  
/NESTED, OFFSET=,

OUTLINE=, POLARIZATION=, /PREVIEW,  
/QUADCUBE, SAVE=, /SILENT,

TRUECOLORS= those keywords have the same meaning as in  
cartview and mollview

---

**DESCRIPTION** hpx2gs reads in a **HEALPix** sky map in FITS format or from a memory array and generates a cartesian projection of it in a PNG file, as well as a Google Sky compatible **KML** file. Missing or unobserved pixels in the input data will be totally 'transparent' in the output file.

---

## RELATED ROUTINES

This section lists the routines related to **hpx2gs**.

hpx2dm see cartview  
turns Healpix maps into DomeMaster images

---

## EXAMPLE:

```
map = findgen(48)
hpx2gs, map, kml='my_map.kml', title='my map in Google'
```

produces in `my_map.kml` and in `my_map.png` an image of the input map that can be seen with Google Sky. To do so, start GoogleEarth or GoogleSky and open `my_map.kml`. Under MacOSX, simply type `open my_map.kml` on the command line.

# ialteralm

Location in HEALPix directory tree: `src/idl/interfaces/ialteralm.pro`

This IDL facility provides an interface to F90 'alteralm' facility. This program can be used to modify a set of  $a_{lm}$  spherical harmonics coefficients, as those extracted by `ianafast` or simulated by `isynfast`, before they are used as constraints on a `isynfast` run. Currently the alterations possible are

- rotation (using Wigner matrices) of the  $a_{lm}$  from the input coordinate system to any other standard astrophysical coordinate system. The resulting  $a_{lm}$  can be used with e.g. `synfast` to generate a map in the new coordinate system.
- removal of the pixel and beam window functions of the input  $a_{lm}$  (corresponding to the pixel size and beam shape of the map from which they were extracted) and implementation of an arbitrary pixel and beam window function.

$$a_{\ell m}^{\text{OUT}} = a_{\ell m}^{\text{IN}} \frac{B^{\text{OUT}}(\ell) P^{\text{OUT}}(\ell)}{B^{\text{IN}}(\ell) P^{\text{IN}}(\ell)}, \quad (6)$$

where  $P(\ell)$  is the pixel window function, and  $B(\ell)$  is the beam window function (assuming a circular beam) or any other  $\ell$  space filter (eg, Wiener filter). For an infinitely small pixel (or beam) one would have  $P(\ell) = 1$  (resp.  $B(\ell) = 1$ ) for any  $\ell$ .

## FORMAT

```
IDL> IALTERALM, alm_in, alm_out, [
    beam_file_in, beam_file_out, binpath=, coord_in, coord_out,
    epoch_in, epoch_out, fwhm_arcmin_in, fwhm_arcmin_out, /help,
    keep_tmp_files=, lmax_out, nlmax_out, nside_in, nside_out,
    nsmax_in, nsmax_out, /silent, tmpdir= ]
```

## QUALIFIERS

alm_in	required input: input $a_{lm}$ , must be a FITS file
alm_out	required output: output $a_{lm}$ , must be a FITS file

---

## KEYWORDS

binpath=	full path to back-end routine ( <b>default:</b> \$HEXE/alteralm, then \$HEALPIX/bin/alteralm) – a binpath starting with / (or \), or \$ is interpreted as absolute – a binpath starting with ./ is interpreted as relative to current directory – all other binpaths are relative to \$HEALPIX
beam_file_in=	Beam window function of input $a_{lm}$ , either a FITS file or an array. If present, will override <b>fwhm_arcmin_in</b> ( <b>default:</b> value of BEAM_LEG keyword read from <b>alm_in</b> )
beam_file_out=	Beam window function of output alm, either a FITS file or an array. If present and non-empty, will override <b>fwhm_arcmin_out</b> ( <b>default:</b> " (empty string, no beam window applied))
coord_in=	Astrophysical coordinates system used to compute input $a_{lm}$ . Case-insensitive single letter code. Valid choices are 'g','G' = Galactic, 'e','E' = Ecliptic, 'c','q','C','Q' = Celestial/eQuatorial. ( <b>default:</b> value of COORDSYS keyword read from <b>alm_in</b> )
coord_out=	Astrophysical coordinates system of output alm. ( <b>default:</b> <b>coord_in</b> )
epoch_in=	Astronomical epoch of input coordinates ( <b>coord_in</b> ) ( <b>default:</b> 2000.0)
epoch_out=	Astronomical epoch of output coordinates ( <b>coord_out</b> ) ( <b>default:</b> same as <b>epoch_in</b> )
fwhm_arcmin_in=	Full Width Half-Maximum in arcmin of Gaussian beam applied to map from which are obtained input $a_{lm}$ . ( <b>default:</b> value of FWHM keyword in <b>alm_in</b> )
fwhm_arcmin_out=	FWHM in arcmin to be applied to output alm. ( <b>default:</b> <b>fwhm_arcmin_in</b> )

/help	if set, prints extended help
/keep_tmp_files	if set, temporary files are not discarded at the end of the run
lmax_out=, nlmax_out=	maximum multipole of output alm
nside_in=, nsmax_in=	HEALPix resolution parameter of map from which were computed input $a_{lm}$ ( <b>default:</b> determined from <code>alm_in</code> )
nside_out=, nsmax_out=	HEALPix resolution parameter Nside whose window function will be applied to output alm. Could be set to 0 for infinitely small pixels (no window) ( <b>default:</b> same as input <code>nsmax_in</code> )
/silent	if set, works silently
tmpdir=	directory in which are written temporary files ( <b>default:</b> IDL_TMPDIR (see IDL documentation))

---

**DESCRIPTION** `ialteralm` is an interface to 'alteralm' F90 facility. It requires some disk space on which to write the parameter file and the other temporary files. Most data can be provided/generated as an external FITS file, or as a memory array.

---

## RELATED ROUTINES

This section lists the routines related to **ialteralm**.

idl	version 6.4 or more is necessary to run <code>ialteralm</code> .
alteralm	F90 facility called by <code>ialteralm</code> .
<code>ianafast</code>	IDL Interface to F90 <code>anafast</code> and C++ <code>anafast_cxx</code>
<code>iprocess_mask</code>	IDL Interface to F90 <code>process_mask</code>
<code>ismoothing</code>	IDL Interface to F90 <code>smoothing</code>
<code>isynfast</code>	IDL Interface to F90 <code>synfast</code>

---

## EXAMPLE:

```
ialteralm, !healpix.path.test+'alm.fits', '/tmp/alm_equat.fits', $  
coord_in='g',coord_out='q'  
isynfast, 0, alm_in='/tmp/alm_equat.fits', '/tmp/map_equat.fits'  
mollview, '/tmp/map_equat.fits',1  
mollview, '/tmp/map_equat.fits',2
```

This example script reads the test (polarised)  $a_{lm}$  located in `$HEALPIX/test/alm.fits` and rotates them from Galactic to Equatorial coordinates, it then synthesizes a map out of those, and finally plots its I and Q Stokes components (in Equatorial coordinates)

# ianafast

Location in HEALPix directory tree: `src/idl/interfaces/ianafast.pro`

This IDL facility provides an interface to 'anafast' F90 and 'anafast\_cxx' C++ facilities. It can be used to produce the Spherical Harmonics coefficients ( $a_{lm}$  of a **HEALPix** map (or pair of maps) and/or the resulting auto (or cross) power spectra  $C(l)$ .

## FORMAT

```
IDL> IANAFAST,  map1_in[,  cl_out,
alm1_out=,  alm2_out=,  binpath=,  cxx=,
double=,  help=,  healpix_data=,  iter_order=,
keep_tmp_files=,  map2_in=,  maskfile=,
nested=,  nlmax=,  nmmax=,  ordering=,
plmfile=,  polarisation=,  regression=,
ring=,  show_cl=,  simul_type=,  silent=,
theta_cut_deg=,  tmpdir=,  weighted=,  won=,
w8file=, w8dir=]
```

## QUALIFIERS

map1_in	required input: 1st input map, can be a FITS file, or a memory array containing the map to analyze
cl_out	optional output: auto or cross power spectrum $C(l)$ , can be a FITS file or a memory array

## KEYWORDS

alm1_out=	output alm of 1st map, must be a FITS file ( <b>default:</b> alm not kept)
alm2_out=	output alm of 2nd map (if any, must be a FITS file) ( <b>default:</b> alm not kept)
binpath=	full path to back-end routine ( <b>default:</b> \$HEXEXE/anafast,

	then <code>\$HEALPIX/bin/anafast</code> or <code>\$HEALPIX/src/cxx/\$HEALPIX_TARGET-</code> <code>/bin/anafast_cxx,</code> then <code>\$HEALPIX/src/cxx/generic_gcc/bin/anafast_cxx</code> if <code>cxx</code> is set)
	– a binpath starting with <code>/</code> (or <code>\</code> ), or <code>\$</code> is interpreted as absolute
	– a binpath starting with <code>./</code> is interpreted as relative to current directory
	– all other binpaths are relative to <code>\$HEALPIX</code>
<code>/cxx</code>	if set, the C++ back-end <code>anafast_cxx</code> is invoked instead of F90 <code>anafast</code> , AND the parameter file is written accordingly
<code>/double</code>	if set, I/O is done in double precision ( <b>default:</b> single precision I/O)
<code>/help</code>	if set, prints extended help
<code>healpix_data=</code>	directory with Healpix precomputed files (only for C++ back_end when <code>weighted=1</code> ) ( <b>default:</b> <code>\$HEALPIX/data</code> )
<code>iter_order=</code>	order of iteration in the analysis ( <b>default:</b> 0)
<code>/keep_tmp_files</code>	if set, temporary files are not discarded at the end of the run
<code>map2_in=</code>	2nd input map (FITS file or array), if provided, <code>Clout</code> will contain the cross power spectra of the 2 maps ( <b>default:</b> no 2nd map)
<code>maskfile=</code>	pixel mask (FITS file or array) ( <b>default:</b> no mask)
<code>/nested=</code>	if set, signals that <i>*all*</i> maps and mask read online are in NESTED scheme (does not apply to FITS file), see also <code>/ring</code> and <code>Ordering</code>
<code>nlmax=</code>	maximum multipole of analysis, <i>*required*</i> for C++ <code>anafast_cxx</code> , optional for F90 <code>anafast</code>
<code>nmmax=</code>	maximum degree <code>m</code> , only valid for C++ <code>anafast_cxx</code> ( <b>default:</b> <code>nlmax</code> )
<code>ordering=</code>	either 'RING' or 'NESTED', ordering of online maps and masks, see <code>/nested</code> and <code>/ring</code>
<code>plmfile=</code>	FITS file containing precomputed Spherical Harmonics (deprecated) ( <b>default:</b> no file)
<code>/polarisation</code>	if set analyze temperature + polarization (same as <code>simul_type = 2</code> )

---

regression=	0, 1 or 2, regress out best fit monopole and/or dipole before alm analysis ( <b>default:</b> 0, analyze raw map)
/ring	see /nested and ordering above
/show_cl	if set, and <code>cl_out</code> is defined, the produced $l(l+1)C(l)/2\pi$ will be plotted
simul_type=	1 or 2, analyze temperature only or temperature + polarization
/silent	if set, works silently
theta_cut_deg=	cut around the equatorial plane
tmpdir=	directory in which are written temporary files ( <b>default:</b> IDL_TMPDIR (see IDL documentation))
/weighted	same as won ( <b>default:</b> apply weighting)
/won	if set, a weighting scheme is used to improve the quadrature ( <b>default:</b> apply weighting)
w8file=	FITS file containing weights ( <b>default:</b> determined automatically by back-end routine). Do not set this keyword unless you really know what you are doing
w8dir=	directory where the weights are to be found ( <b>default:</b> determined automatically by back-end routine)

---

**DESCRIPTION** `ianafast` is an interface to '[anafast](#)' F90 and '`anafast_cxx`' C++ facilities. It requires some disk space on which to write the parameter file and the other temporary files. Most data can be provided/generated as an external FITS file, or as a memory array.

---

## RELATED ROUTINES

This section lists the routines related to **ianafast**.

idl	version 6.4 or more is necessary to run <code>ianafast</code> .
<a href="#">anafast</a>	F90 facility called by <code>ianafast</code> .
<code>anafast_cxx</code>	C++ called by <code>ianafast</code> .
<a href="#">ialteralm</a>	IDL Interface to F90 <a href="#">alteralm</a>



---

<code>iprocess_mask</code>	IDL Interface to F90 <a href="#">process_mask</a>
<code>ismoothing</code>	IDL Interface to F90 <a href="#">smoothing</a>
<code>isynfast</code>	IDL Interface to F90 <a href="#">synfast</a>

---

**EXAMPLE:**

```
whitenoise = randomn(seed, nside2npix(256))
ianafast, whitenoise, cl, /ring, /silent
plot, cl[*,0]
```

will plot the power spectrum of a white noise map

# index2lm

Location in HEALPix directory tree: `src/idl/misc/index2lm.pro`

This IDL routine provides a means to convert the  $a_{\ell m}$  index  $i = \ell^2 + \ell + m + 1$  (as returned by eg the `fits2alm` routine) into  $\ell$  and  $m$ .

**FORMAT** IDL> INDEX2LM, `index`, `l`, `m`

## QUALIFIERS

<code>index</code>	Long array containing on INPUT the index $i = \ell^2 + \ell + m + 1$ .
<code>l</code>	Long array containing on OUTPUT the order $\ell$ . It has the same size as <code>index</code> .
<code>m</code>	Long array containing on OUTPUT the degree $m$ . It has the same size as <code>index</code> .

**DESCRIPTION** `index2lm` converts  $i = \ell^2 + \ell + m + 1$  into  $(\ell, m)$ . Note that the index  $i$  is only defined for  $0 \leq |m| \leq \ell$ .

## RELATED ROUTINES

This section lists the routines related to **index2lm**.

<code>idl</code>	version 6.4 or more is necessary to run <code>index2lm</code> .
<code>fits2alm</code>	reads a FITS file containing $a_{\ell m}$ values.
<code>alm2fits</code>	writes $a_{\ell m}$ values into a FITS file.
<code>lm2index</code>	routine complementary to <code>index2lm</code> : converts $(\ell, m)$ into $i = \ell^2 + \ell + m + 1$ .

## EXAMPLE:

```
index2lm, index, l, m
```

---

will return in `l` and `m` the order  $\ell$  and degree  $m$  such that `index`  
 $= \ell^2 + \ell + m + 1$

# init\_healpix

---

**Location in HEALPix directory tree:** `src/idl/misc/init_healpix.pro`

This IDL facility creates an IDL system variable (!HEALPIX) containing various **HEALPix** related quantities

---

**FORMAT**            IDL> INIT\_HEALPIX [, **VERBOSE=**]

---

## KEYWORDS

VERBOSE=	if set, turn on the verbose mode, giving a short description of the variables just created.
----------	---

---

**DESCRIPTION** `init_healpix` defines the IDL system variable and structure !HEALPIX containing several quantities and character string necessary to **HEALPix**, eg : allowed resolution parameters Nside, full path to package directory, package version...

---

## RELATED ROUTINES

This section lists the routines related to **init\_healpix**.

idl	version 6.4 or more is necessary to run <code>init_healpix</code> .
!HEALPIX	IDL system variable defined by <code>init_healpix</code> .

---

## EXAMPLES: #1

```
init_healpix,/verbose
```

init\_healpix will create the system variable !Healpix, and give a short description of the tags available, as shown below

Initializing !HEALPIX system variable

This system variable contains some information on Healpix :

!HEALPIX.VERSION = current version number,

!HEALPIX.DATE = date of release,

!HEALPIX.DIRECTORY = directory containing Healpix package,

!HEALPIX.PATH = structure containing:

!HEALPIX.PATH.BIN = structure containing binary path :

!HEALPIX.PATH.BIN.CXX = C++

!HEALPIX.PATH.BIN.F90 = Fortran90

!HEALPIX.PATH.DATA = path to data subdirectory,

!HEALPIX.PATH.DOC = path to doc subdirectories (.html, .pdf),

!HEALPIX.PATH.SRC = path to src subdirectory,

!HEALPIX.PATH.TEST = path to test subdirectory,

!HEALPIX.NSIDE = list of all valid values of Nside parameter,

!HEALPIX.BAD\_VALUE = value of flag given to missing pixels in FITS files,

!HEALPIX.COMMENT = this description.

---

## EXAMPLES: #2

help, !healpix, /structure

will print the content of the !Healpix system structure.

## iprocess\_mask

Location in HEALPix directory tree: `src/idl/interfaces/iprocess_mask.pro`

This IDL facility provides an interface to F90 `'process_mask'` facility. For a given input binary mask, it can determine the angular distance in Radians of each valid (1 valued) pixel to the closest invalid (0 valued) pixel, with the option of ignoring small clusters of invalid pixels. The distance map can then be used to generate an apodized mask.

---

**FORMAT** IDL> IPROCESS\_MASK, `mask_in`, `distance_map`,[ `binpath=`, `filled_mask=`, `/help`, `hole_arcmin2=`, `hole_pixels=`, `keep_tmp_files=`, `/nested`, `ordering=`, `/ring`, `/silent`, `tmpdir=`]

---

## QUALIFIERS

<code>mask_in</code>	required input: input binary mask. It can be a FITS file, or a memory array containing the mask to process.
<code>distance_map</code>	optional output: double precision angular distance map in Radians. It can be a FITS file, or a memory array. It will have the same ordering as the input mask.

---

## KEYWORDS

<code>binpath=</code>	full path to back-end routine ( <b>default:</b> <code>\$HEXE/process_mask</code> , then <code>\$HEALPIX/bin/process_mask</code> ) – a binpath starting with <code>/</code> (or <code>\</code> ), or <code>\$</code> is interpreted as absolute – a binpath starting with <code>./</code> is interpreted as relative to current directory – all other binpaths are relative to <code>\$HEALPIX</code>
<code>filled_mask=</code>	optional output mask with holes smaller than <code>hole_arcmin2</code> or <code>hole_pixels</code> filled in. Will have

	the same ordering as the input mask
/help	if set, prints extended help
hole_arcmin2	Minimal size (in arcmin <sup>2</sup> ) of invalid regions to be kept (can be used together with <a href="#">hole_pixels</a> , the result will be the largest of the two). ( <b>default:</b> 0.0)
hole_pixels	Minimal size (in pixels) of invalid regions to be kept (can be used together with <a href="#">hole_arcmin2</a> , the result will be the largest of the two). ( <b>default:</b> 0)
/keep_tmp_files	if set, temporary files are not discarded at the end of the run
/nested	if set, signals that the mask read online is in NESTED scheme (does not apply to FITS file), see also <a href="#">/ring</a> and <a href="#">Ordering</a>
ordering=	either 'RING' or 'NESTED', ordering of online mask, see <a href="#">/ring</a> and <a href="#">/nested</a>
/ring	see <a href="#">/nested</a> and <a href="#">Ordering</a> above
/silent	if set, works silently
tmpdir=	directory in which are written temporary files ( <b>default:</b> IDL_TMPDIR (see IDL documentation))

---

**DESCRIPTION** `iprocess_mask` is an interface to '[process\\_mask](#)' F90 facility. It requires some disk space on which to write the parameter file and the other temporary files. Most data can be provided/generated as an external FITS file, or as a memory array.

---

## RELATED ROUTINES

This section lists the routines related to **`iprocess_mask`**.

idl	version 6.4 or more is necessary to run <code>iprocess_mask</code> .
<code>process_mask</code>	F90 facility called by <code>iprocess_mask</code> .
<a href="#">ialteralm</a>	IDL Interface to F90 <a href="#">alteralm</a>
<a href="#">ianafast</a>	IDL Interface to F90 <a href="#">anafast</a> and C++ <code>anafast_cxx</code>

<a href="#">ismoothing</a>	IDL Interface to F90 <a href="#">smoothing</a>
<a href="#">isynfast</a>	IDL Interface to F90 <a href="#">synfast</a>

---

**EXAMPLE:**

```
npix = nside2npix(256)
mask = replicate(1, npix) & mask[randomu(seed,100)*npix] = 0
iprocess_mask, mask, distance, /ring, /silent
mollview, distance
```

A binary mask in which 100 randomly located pixels are 0-valued (=invalid) is generated. Then the distance (in Radians) of the valid pixels to the closest invalid pixels is computed and plotted.



Location in HEALPix directory tree: `src/idl/interfaces/ismoothing.pro`

FORMAT	IDL>	ISMOOTHING,	map1_in,
	map2_out,[	beam_file=,	binpath=,
	/double,	fwhm_arcmin=,	/help,
	iter_order=,	keep_tmp_files=,	lmax=,
	nlmax=,	/nested,	ordering=,
	plmfile=,	regression=,	/ring,
	simul_type=,	/silent,	theta_cut_deg=,
	tmpdir=,	/won,	w8file=,
	w8dir=]		

map1_in	required input: input map, can be a FITS file, or a memory array containing the map to smooth
map2_out	required output: output smoothed map, can be a FITS file, or a memory array

beam\_file= beam window function, either a FITS file or an array

binpath= full path to back-end routine  
(**default:** \$HEXE/smoothing, then \$HEALPIX/bin/smoothing)

- a binpath starting with / (or \), or \$ is interpreted as absolute
- a binpath starting with ./ is interpreted as relative to current directory
- all other binpaths are relative to \$HEALPIX

---

<code>/double</code>	if set, I/O is done in double precision ( <b>default:</b> single precision I/O)
<code>fwhm_arcmin=</code>	gaussian beam Full Width Half Maximum in arc-minutes ( <b>default:</b> 0)
<code>/help</code>	if set, prints extended help
<code>iter_order=</code>	order of iteration in the analysis ( <b>default:</b> 0)
<code>/keep_tmp_files</code>	if set, temporary files are not discarded at the end of the run
<code>lmax=, nlmax=</code>	maximum multipole of smoothing ( <b>default:</b> determined by back-end routine (ie, smoothing))
<code>/nested</code>	if set, signals that *all* maps and mask read online are in NESTED scheme (does not apply to FITS file), <code>/ring</code> and <code>Ordering</code>
<code>ordering=</code>	either 'RING' or 'NESTED', ordering of online maps and masks, see <code>/ring</code> and <code>/nested</code>
<code>plmfile=</code>	FITS file containing precomputed Spherical Harmonics (deprecated) ( <b>default:</b> no file)
<code>regression=</code>	0, 1 or 2, regress out best fit monopole and/or dipole before alm analysis ( <b>default:</b> 0, analyze raw map)
<code>/ring</code>	see <code>/nested</code> and <code>Ordering</code> above
<code>simul_type=</code>	1 or 2, analyze temperature only or temperature + polarization
<code>/silent</code>	if set, works silently
<code>theta_cut_deg=</code>	cut around the equatorial plane
<code>tmpdir=</code>	directory in which are written temporary files ( <b>default:</b> IDL_TMPDIR (see IDL documentation))
<code>/won</code>	if set, a weighting scheme is used to improve the quadrature ( <b>default:</b> apply weighting)
<code>w8file=</code>	FITS file containing weights ( <b>default:</b> determined automatically by back-end routine). Do not set this keyword unless you really know what you are doing
<code>w8dir=</code>	directory where the weights are to be found ( <b>default:</b> determined automatically by back-end routine)

---

**DESCRIPTION** ismoothing is an interface to 'smoothing' F90 facility. It requires some disk space on which to write the parameter file and the other temporary files. Most data can be provided/generated as an external FITS file, or as a memory array.

---

## RELATED ROUTINES

This section lists the routines related to **ismoothing**.

idl	version 6.4 or more is necessary to run ismoothing.
smoothing	F90 facility called by ismoothing.
<b>beam2bl</b>	This IDL facility computes a transfer (or window) function $b(l)$ (such as the ones required by ismoothing) for a given circular beam profile $b(\theta)$
<b>ialteralm</b>	IDL Interface to F90 <a href="#">alteralm</a>
<b>ianafast</b>	IDL Interface to F90 <a href="#">anafast</a> and C++ <code>anafast.cxx</code>
<b>iprocess_mask</b>	IDL Interface to F90 <a href="#">process_mask</a>
<b>isynfast</b>	IDL Interface to F90 <a href="#">synfast</a>

---

## EXAMPLE:

```
whitenoise = randomn(seed, nside2npix(256))
ismoothing, whitenoise, rednoise, fwhm=120, /ring, simul=1,/silent
mollview, whitenoise, title='White noise'
mollview, rednoise, title='Smoothed white Noise'
```

will generate and plot a white noise map and its smoothed version

# isynfast

Location in HEALPix directory tree: `src/idl/interfaces/isynfast.pro`

This IDL facility provides an interface to F90 'synfast' facility. It can be used to generate sky maps and/or  $a_{lm}$  from power spectra ( $C(l)$ ), synthesize maps from  $a_{lm}$  or simulate maps from  $C(l)$  and constraining  $a_{lm}$ .

---

**FORMAT** IDL> ISYNFAST, `cl_in` [, `map_out`, `alm_in` =, `alm_out` =, `apply_windows` =, `beam_file` =, `bin_path` =, `double` =, `fwhm_arcmin` =, `help` =, `iseed` =, `keep_tmp_files` =, `lmax` =, `nlmax` =, `nside` =, `nsmax` =, `plmfile` =, `simul_type` =, `silent` =, `tmpdir` =, `windowfile` =, `winfiledir` =]

---

## QUALIFIERS

<code>cl_in</code>	input power spectrum, can be a FITS file, or a memory array containing the $C(l)$ , used to generate a map or a set of gaussian alm If empty quotes (") or a zero (0) are provided, it will be interpreted as "No input $C(l)$ ", in which case some input alm's ( <code>alm_in</code> ) are required.
<code>map_out</code>	optional output: <i>RING ordered</i> map synthetised from the power spectrum or from constraining alm

---

## KEYWORDS

<code>alm_in</code> =	optional input (constraining) alm (must be a FITS file) ( <b>default:</b> no alm)
<code>alm_out</code> =	contains on output the effective alm (must be a FITS file)
<code>/apply_windows</code>	if set, beam and pixel windows are applied to input <code>alm_in</code> (if any)
<code>beam_file</code> =	beam window function, either a FITS file or an array

binpath=	full path to back-end routine ( <b>default:</b> \$HEXE/synfast, then \$HEALPIX/bin/synfast) – a binpath starting with / (or \), or \$ is interpreted as absolute – a binpath starting with ./ is interpreted as relative to current directory – all other binpaths are relative to \$HEALPIX
/double	if set, I/O is done in double precision ( <b>default:</b> single precision I/O)
fwhm_arcmin=	gaussian beam FWHM in arcmin ( <b>default:</b> 0)
/help	if set, prints extended help
iseed=	integer seed of random sequence ( <b>default:</b> 0)
/keep_tmp_files	if set, temporary files are not discarded at the end of the run
lmax=, nlmax=	maximum multipole simulation ( <b>default:</b> $2*N_{\text{side}}$ )
nside=, nsmax=	Healpix resolution parameter $N_{\text{side}}$
plmfile=	FITS file containing precomputed Spherical Harmonics (deprecated) ( <b>default:</b> no file)
simul_type=	1) Temperature only 2) Temperature + polarisation 3) Temperature + 1st derivatives 4) Temperature + 1st & 2nd derivatives 5) T+P + 1st derivatives 6) T+P + 1st & 2nd derivatives ( <b>default:</b> 2: T+P)
/silent	if set, works silently
tmpdir=	directory in which are written temporary files ( <b>default:</b> IDL_TMPDIR (see IDL documentation))
windowfile=	FITS file containing pixel window ( <b>default:</b> determined automatically by back-end routine). Do not set this keyword unless you really know what you are doing
winfiledir=	directory where the pixel windows are to be found ( <b>default:</b> determined automatically by back-end routine). Do not set this keyword unless you really know what you are doing

**DESCRIPTION** `isynfast` is an interface to F90 [synfast](#) F90 facility. It requires some disk space on which to write the parameter file and the other temporary files. Most data can be provided/generated as an external FITS file, or as a memory array.

---

## RELATED ROUTINES

This section lists the routines related to **`isynfast`**.

<code>idl</code>	version 6.4 or more is necessary to run <code>isynfast</code> .
<code>synfast</code>	F90 facility called by <code>isynfast</code> .
<a href="#">ialteralm</a>	IDL Interface to F90 <a href="#">alteralm</a>
<a href="#">ianafast</a>	IDL Interface to F90 <a href="#">anafast</a> and C++ <code>anafast.cxx</code>
<a href="#">iprocess_mask</a>	IDL Interface to F90 <a href="#">process_mask</a>
<a href="#">ismoothing</a>	IDL Interface to F90 <a href="#">smoothing</a>

---

## EXAMPLE:

```
isynfast, '$HEALPIX/test/cl.fits', map, fwhm=30, nside=256, /silent
mollview, map, 1, title='I'
mollview, map, 2, title='Q'
```

will synthesize and plot I and Q maps consistent with WMAP-1yr best fit power spectrum and observed with a circular gaussian 30 arcmin beam.

# lm2index

Location in HEALPix directory tree: `src/idl/misc/lm2index.pro`

This IDL routine provides a means to convert the  $a_{\ell m}$  degree and order  $(\ell, m)$  into the index  $i = \ell^2 + \ell + m + 1$  (in order to be fed to alm2fits routine for instance)

---

**FORMAT** IDL> LM2INDEX, **l**, **m**, **index**

---

## QUALIFIERS

<b>l</b>	Long array containing on INPUT the order $\ell$ .
<b>m</b>	Long array containing on INPUT the degree $m$ .
<b>index</b>	Long array containing on OUTPUT the index $i = \ell^2 + \ell + m + 1$ .

---

**DESCRIPTION** lm2index converts  $(\ell, m)$  into  $i = \ell^2 + \ell + m + 1$ . Note that by definition  $0 \leq |m| \leq \ell$  (the routine does not check for this).

---

## RELATED ROUTINES

This section lists the routines related to **lm2index**.

<b>idl</b>	version 6.4 or more is necessary to run lm2index.
<b>fits2alm</b>	reads a FITS file containing $a_{\ell m}$ values.
<b>alm2fits</b>	writes $a_{\ell m}$ values into a FITS file.
<b>index2lm</b>	routine complementary to lm2index: converts $i = \ell^2 + \ell + m + 1$ into $(\ell, m)$ .

---

## EXAMPLE:

lm2index, **l**, **m**, **index**

will return in **index** in value  $\ell^2 + \ell + m + 1$

## median\_filter

Location in HEALPix directory tree: `src/idl/toolkit/median_filter.pro`

This IDL facility allows the median filtering of a Healpix map.

---

**FORMAT** IDL> MEDIAN\_FILTER (*InputMap*, *Radius*, *MedianMap* [, *ORDERING* =, /RING, /NESTED, /FILL\_HOLES, /DEGREES, /ARCMIN])

---

## QUALIFIERS

<i>InputMap</i>	(IN) either an IDL array containing a full sky Healpix map to filter ('online' usage), or the name of an external FITS file containing a full sky or cut sky map
<i>Radius</i>	(IN) radius of the disk on which the median is computed. It is in Radians, unless /DEGREES or /ARCMIN are set
<i>MedianMap</i>	(OUT) either an IDL variable containing on output the filtered map, or the name of an external FITS file to contain the map. Should be of same type of <i>InputMap</i> . Flagged pixels (ie, having the value <code>!healpix.bad_value</code> ) are left unchanged, unless /FILL_HOLES is set.

---

## KEYWORDS

/ARCMIN	If set, <i>Radius</i> is in arcmin rather than radians
/DEG	If set, <i>Radius</i> is in degrees rather than radians
/FILL_HOLES	If set, flagged pixels are replaced with the median of the valid pixels found within a distance <i>Radius</i> . If there are any.
/NESTED	Same as <i>ORDERING</i> ='NESTED'
<i>ORDERING</i> =	Healpix map ordering, should be either 'RING' or 'NESTED'. Only applies to 'online' usage.



/RING            Same as ORDERING='RING'

---

**DESCRIPTION** `median_filter` allows the median filtering of a Healpix map. Each pixel of the output map is the median value of the input map pixels found within a disc of given radius centered on that pixel. Flagged pixels can be either left unchanged or 'filled in' with that same scheme.

If the map is polarized, each of the three Stokes components is filtered separately.

The input and output can either be arrays or FITS files, but they to be both arrays or both FITS files.

---

## RELATED ROUTINES

This section lists the routines related to **median\_filter** .

idl            version 6.4 or more is necessary to run `median_filter`

---

## EXAMPLE:

```
median_filter ('map.fits', 10., /arcmin, 'med.fits')
```

Writes in 'med.fits' the median filtered map of 'map.fits' using a disc radius of 10 arcmin

---

## EXAMPLE:

```
map = randomn(seed, nside2npix(256))
median_filter (map, 0.5, /deg, med)
```

Returns in `med` the median filtered map of `map` using a disc radius of 0.5 degrees

## mollcursor

---

Location in HEALPix directory tree: `src/idl/visu/mollcursor.pro`

This IDL facility provides a point-and-click interface for finding the astronomical location, value and pixel index of the pixels nearest to the pointed position on a Mollweide projection of a **HEALPix** map.

---

**FORMAT**            IDL>    MOLLCURSOR,    [`cursor_type=`,  
                         `file_out=`]

---

## QUALIFIERS

`cursor_type=`            cursor type to be used  
                          (**default:** 34)

`file_out=`              file containing on output the list of point selected  
                          with the cursor.  
                          If set to 1, the file will take its default name: 'cur-  
cursor\_catalog.txt'.  
                          If set to a non-empty character string, the file  
                          name will be that string

---

**DESCRIPTION** mollcursor should be run immediately following mollview. It gives the longitude, latitude, map value and pixel number corresponding to the cursor position in the window containing the map generated by mollview. Mouse buttons are used to select the function :

left button = display the information relative to the current cursor position,

middle button = print out this information in the IDL command window

right button = quit mollcursor

**Note on Mac OS X, X11 and IDL cursor:** depending on the Mac OS X version<sup>a</sup> and most importantly on the X Window System being used,<sup>b</sup> the IDL function **cursor**, and therefore **HEALPix** mollcursor, gnomcursor, ... will not work properly under X11. To solve this problem, type the relevant line below at your X11 prompt and restart X11.

If you are using Apple's X11, type under Tiger (10.4):

```
defaults write com.apple.x11 wm_click_through -bool true
```

or, under Leopard (10.5), Snow Leopard (10.6), Lion (10.7):

```
defaults write org.x.x11 wm_click_through -bool true
```

If you are using Xquartz (default under Mountain Lion (10.8), Mavericks (10.9) and Yosemite (10.10)):

```
defaults write org.macosforge.xquartz.X11 wm_click_through -bool true
```

and if you are using MacPort's X11 (package xorg-server):

```
defaults write org.macports.X11 wm_click_through -bool true
```

(see [http://www.idlcoyote.com/misc\\_tips/maccursor.html](http://www.idlcoyote.com/misc_tips/maccursor.html) and "**HEALPix Installation Documentation**").

And finally, mollcursor obviously requires the '3 button mouse' to be enabled, which can be done in the X11 Preferences menu, or if Xquartz is used (see `man Xquartz`) via:

```
defaults write org.macosforge.xquartz.X11 enable_fake_buttons -bool true
```

<sup>a</sup>the command `sw_vers -productVersion` can be used to know the Mac OS X version being used

<sup>b</sup>the command `ls -lrt $HOME/Library/Preferences/*[xX]11.plist` can be used to determine the X implementation and its configuration file

---

## RELATED ROUTINES

This section lists the routines related to **mollcursor**.

idl	version 6.4 or more is necessary to run mollcursor
ghostview	ghostview or a similar facility is required to view the Postscript image generated by mollcursor.
xv	xv or a similar facility is required to view the GIF/PNG image generated by mollcursor(a browser can also be used).
synfast	This <b>HEALPix</b> facility will generate the FITS format sky map to be input to mollcursor.
cartview	IDL facility to generate a Cartesian projection of a <b>HEALPix</b> map.
cartcursor	interactive cursor to be used with cartview
gnomview	IDL facility to generate a gnomonic projection of a <b>HEALPix</b> map.
gnomcursor	interactive cursor to be used with gnomview
mollview	IDL facility to generate a Mollweide projection of a <b>HEALPix</b> map.
mollcursor	interactive cursor to be used with mollview
orthview	IDL facility to generate an orthographic projection of a <b>HEALPix</b> map.
orthcursor	interactive cursor to be used with orthview

---

**EXAMPLE:**

mollcursor

After mollview reads in a map and generates its mollweide projection, mollcursor is run to know the position and flux of bright synchrotron sources, for example.

# mollview

**Location in HEALPix directory tree:** `src/idl/visu/mollview.pro`

This IDL facility provides a means to visualise a full sky Mollweide projection of **HEALPix** and COBE Quad-Cube maps in an IDL environment. It also offers the possibility to generate GIF, JPEG, PDF, PNG and Postscript color-coded images of the projected map. The projected (but not color-coded) data can also be output in FITS files and IDL arrays.

## FORMAT

```
IDL> MOLLVIEW,      File [, Select] [, AS-
INH=,  BAD_COLOR=,  BG_COLOR=,  CHARSIZE=,
CHARTHICK=, COLT=, COORD=, /CROP, EXECUTE=,
FACTOR=, FG_COLOR=, FITS=, /FLIP, GAL_CUT=,
GIF=, GLSIZE=, GRATICULE=, /HALF_SKY, HBOUND=,
/HELP, /HIST_EQUAL, HXSIZE=, IGLSIZE=, IGRATIC-
ULE=, JPEG=, LATEX=, /LOG, MAP_OUT=, MAX=,
MIN=,  /NESTED,  /NO_DIPOLE,  /NO_MONOPOLE,
/NOBAR, /NOLABELS, /NOPOSITION, OFFSET=, OUT-
LINE=, PDF=, PFonts=, PNG=, POLARIZATION=,
/PREVIEW, PS=, PXSIZ=, PYSIZ=, RESO_ARCMIN=,
RETAIN=, ROT=, /SAVE, /SHADED, /SILENT, STAG-
GER=, SUBTITLE=, TITLEPLOT=, TRANSPARENT=,
TRUECOLORS=, UNITS=, WINDOW=, XPOS=, YPOS=]
```

Several visualization routines have a similar interface. Their **qualifiers** and **keywords** are all listed here, and the routines to which they apply are coded in the 'routine' column as: A: **azeqview**, C: **cartview**, G: **gnomview**, M: **mollview**, O: **orthview** and all: all of them

**Qualifiers** should appear in the order indicated. They can take a range of values, and some of them are optional.

**Keywords** are optional, and can appear in any order. They take the form **keyword=value** and can be abbreviated to a non ambiguous form (ie, **factor=10.0** can be replaced by **fac = 10.0**). They generally can take a range of values, but some of them (noted as **/keyword** below) are boolean switches: they are either present (or set to 1) or absent (or set to 0).

---

## QUALIFIERS

---

name	routines	description
File	<b>all</b>	<p>Required</p> <p>name of a (possibly gzip compressed) FITS file containing the <b>HEALPix</b> map in an extension or in the image field, <i>or</i> name of an <i>online</i> variable (either array or structure) containing the (RING or <b>NESTED</b> ordered) <b>HEALPix</b> map (See note below);</p> <p>if Save is set : name of an IDL saveset file containing the <b>HEALPix</b> map stored under the variable <b>data</b> (<b>default:</b> none)</p> <p><u>Note on online data:</u> in order to preserve the integrity of the input data, the content of the array or structure <b>File</b> is replicated before being possibly altered by the map making process. Therefore plotting online data will require more memory than reading the data from disc directly, and is not recommended to visualize data sets of size comparable to that of the computer memory.</p> <p><u>Note on high resolution cut sky data:</u> cut sky data (in which less than 50% of the sky is observed), can be processed with a minimal memory foot-print, by not allocating fake full map. In the current release, two restrictions apply: the input data set must be read from a FITS file in 'cut4' format, and the <b>POLARIZATION</b> IDL keyword (described below) must be 0 (default value). See <b>Example #4</b> on page 116.</p> <p>see also: <b>TrueColors</b>.</p>
Select	<b>all</b>	<p>Optional</p> <p>column of the BIN FITS table to be plotted, can be either</p> <ul style="list-style-type: none"> <li>– a name : value given in TTYPEi of the FITS file</li> </ul> <p>NOT case sensitive and can be truncated, (only letters, digits and underscore are valid)</p> <ul style="list-style-type: none"> <li>– an integer : number i of the column containing the data, starting with 1 (also valid if <b>File</b> is an online array)</li> </ul> <p>(<b>default:</b> 1 for full sky maps, 'SIGNAL' column for FITS files containing cut sky maps) (see the Examples below)</p>

---

## KEYWORDS

---

name	routines	description
ASINH=	all	<p>if set, the color table is altered to emulate a non-linear mapping of the input data enhancing the low contrast regions. If <b>asinh=1</b> the mapping is <math>y = \sinh^{-1}(x)</math>, such that <math>y \approx x</math> when <math>x \ll 1</math> and <math>y \approx \ln(2x)</math> when <math>x \gg 1</math>. If <b>asinh=2</b> the mapping is <math>y = \sinh^{-1}(x/2)/\ln(10)</math>, such that <math>y \approx 0.21x</math> when <math>x \ll 1</math> and <math>y \approx \log(x)</math> when <math>x \gg 1</math>. Here <math>x</math> is the input data, optionally altered by <b>Factor</b> and <b>Offset</b>.</p> <p>This option can <i>not</i> be used in conjunction with <b>/LOG</b> nor <b>/HIST_EQUAL</b>.</p>
BAD_COLOR=	all	<p>color given to missing pixels (having <b>!healpix.bad_value</b> (<math>= -1.6375 \cdot 10^{30}</math>) or NaN value on input). The color can be provided as either:</p> <ul style="list-style-type: none"> <li>– a single integer in <math>[0, 255]</math>, specifying the index to be used in the color table chosen via <b>COLT</b> (in which the indexes 0, 1 and 2 are reserved for black, white and grey respectively),</li> <li>– a 3 element vector, with each element in <math>[0, 255]</math>, specifying the amount of RED, GREEN and BLUE</li> <li>– a 7-character string, starting with '#', specifying the color in HTML Hexadecimal fashion (eg, '#ff0000' for red).</li> </ul> <p>(<b>default:</b> neutral grey (<math>=2</math>, <math>= [175, 175, 175]</math>, <math>= \text{'#afafaf'}</math>))</p> <p>see also: <b>BG_COLOR</b>, <b>FG_COLOR</b>, <b>TRANSPARENT</b></p>
BG_COLOR=	all	<p>color given to background pixels (outside the sphere). See <b>BAD_COLOR</b> for expected format. (<b>default:</b> white (<math>=1</math>, <math>= [255, 255, 255]</math>, <math>= \text{'#ffffff'}</math>))</p> <p>see also: <b>FG_COLOR</b>, <b>TRANSPARENT</b></p>
CHARSIZE=	all	<p>overall multiplicative factor applied to the size of all characters appearing on the plot (<b>default:</b> 1.0)</p>
CHARTHICK=	all	<p>character thickness (in <b>TITLE</b>, <b>SUBTITLE</b> and color bar labeling). Other characters thickness (such as <b>graticule labels</b>), can be controlled with <b>!P.CHARTHICK</b>. (<b>default:</b> 1)</p>

name	routines	description
COLT=	all	<p>color table index:</p> <ul style="list-style-type: none"> <li>– Indexes in [0,40] are reserved for standard IDL color tables, while [41,255] are used for user defined color tables read from disc (created and written to disc with <code>MODIFYCT</code>), if any.</li> <li>– Indexes 1001 (or <code>'planck1'</code>, case insensitive) and 1002 (or <code>'planck2'</code>) are reserved for Planck color tables 1 and 2 generated by <code>planck_colors</code>. See <a href="#">Example #6</a> on page 119.</li> <li>– If the index does not match any existing table, or if it is above 255, the current online table (modifiable with <code>TVLCT</code>, <code>XLOADCT</code>, <code>XPALETTE</code>, ... or eg, J. Davenport's <code>cubehelix.pro</code> implementation of D. Green's <code>cubehelix color scheme</code>) is used instead.</li> <li>– If <code>colt &lt; 0</code>, the IDL color table <code>ABS(colt)</code> is used, but the scale is reversed (ie a red to blue scale becomes a blue to red scale). Note: -0.1 can be used as negative 0.</li> </ul> <p>(default: 33 (Blue-Red))  see also: <a href="#">TrueColors</a></p>
COORD=	all	<p>vector with 1 or 2 elements describing the coordinate system of the map; either</p> <ul style="list-style-type: none"> <li>– 'C' or 'Q' : Celestial2000 = eQuatorial,</li> <li>– 'E' : Ecliptic,</li> <li>– 'G' : Galactic</li> </ul> <p>if <code>coord = ['x','y']</code> the map is rotated from system 'x' to system 'y'  if <code>coord = ['y']</code> the map is rotated to coordinate system 'y' (with the original system assumed to be Galactic unless indicated otherwise in the input file)  see also: <a href="#">Rot</a></p>
/CROP	all	<p>if set, the image produced (in GIF/JPEG/PDF/PNG/PS and on screen) only contains the projected map and no title, color bar, ...  see also: <a href="#">Gif</a>, <a href="#">Jpeg</a>, <a href="#">Pdf</a>, <a href="#">Png</a>, <a href="#">Ps</a></p>
EXECUTE=	all	<p>character string containing IDL command(s) to be executed in the plotting window. See <a href="#">Example #3</a> on page 117.</p>
FACTOR=	all	<p>scalar multiplicative factor to be applied to the valid data the data plotted is of the form <code>Factor*(data + Offset)</code>  This does not affect the flagged pixels  Can be used together with <code>ASINH</code> or <code>LOG</code>  When used with <code>TRUECOLORS</code>, <code>FACTOR</code> can be a 3-element vector.  see also: <a href="#">ASINH</a>, <a href="#">Offset</a>, <a href="#">LOG</a>, <a href="#">Truecolors</a>  (default: 1.0)</p>



name	routines	description
FG_COLOR=	all	<p>color of title and subtile characters, graticule lines and labels, units, outlines ...</p> <p>See <b>BAD_COLOR</b> for expected format.  <b>(default:</b> black (=0, =[0, 0, 0], ='#000000'))  see also: <b>BAD_COLOR</b>, <b>BG_COLOR</b></p>
FITS=	all	<p>string containing the name of an output FITS file with the projected map in the primary image</p> <ul style="list-style-type: none"> <li>– if set to 1 : output the plot in <i>plot_proj.fits</i>, where <i>proj</i> is either <i>cartesian</i>, <i>gnomic</i>, <i>mollweide</i>, or <i>orthographic</i> depending on the projection in use;</li> <li>– if set to a file name : output the plot in that file.</li> </ul> <p><b>(default:</b> 0: no .FITS done)</p> <p>In the case of Orthographic projection, <b>HALF_SKY</b> must be set.</p> <p>Except for the color mapping, all the keywords and options apply to the projected map, ie: its size is determined by <b>PX_SIZE</b> (and <b>PYSIZE</b> when applicable), its angular resolution by <b>RESO_ARCMIN</b> when applicable, its orientation and coordinates by <b>ROT</b> and <b>COORD</b> respectively, ...</p> <p>For compatibility with standard FITS viewers (including <b>STIFF</b>), unobserved pixels, and pixels outside the sphere, take the value NaN (ie <code>!values.f_nan</code> in IDL). The resulting FITS file can be read in IDL with eg. <code>map=readfits(filename)</code>.  see also: <b>Map_out</b></p>
/FLIP	all	<p>if set the longitude increases to the right, whereas by default (astronomical convention) it increases towards the left</p>
GAL_CUT=	—MO	<p>(positive float) specifies the symmetric galactic cut in degrees outside of which the monopole and/or dipole fitting is done  <b>(default:</b> 0: monopole and dipole fit done on the whole sky)  (see also: <b>No_dipole</b>, <b>No_monopole</b>)</p>
GIF=	all	<p>string containing the name of a .GIF output</p> <ul style="list-style-type: none"> <li>if set to 1 : outputs the plot in <i>plot_projection.gif</i>, where <i>projection</i> is either <i>azimequid</i>, <i>cartesian</i>, <i>gnomic</i>, <i>mollweide</i> or <i>orthographic</i>,</li> <li>if set to a file name : outputs the plot in that file</li> </ul> <p>Please note that the resulting GIF image might not always look as expected. The reason for this is a problem with 'backing store' in the IDL-routine TVRD. Please read the IDL documentation for more information.  <b>(default:</b> no .GIF done)</p> <p>see also: <b>Crop</b>, <b>Jpeg</b>, <b>Pdf</b>, <b>Png</b>, <b>Preview</b>, <b>Ps</b> and <b>Retain</b></p>

name	routines	description
GLSIZE=	CGMO	character size of the graticule labels in units of <b>Charsize</b> . ( <b>default:</b> 0: no labeling of graticules). see also: <b>Charsize</b> , <b>Graticule</b>
GRATICULE=	CGMO	if set, puts a graticule (ie, longitude and latitude grid) in the <i>output</i> astrophysical coordinates with <code>delta_long = delta_lat = gdef</code> degrees if set to a scalar $x > \text{gmin}$ then <code>delta_long = delta_lat = x</code> if set to <code>[x,y]</code> with $x, y > \text{gmin}$ then <code>delta_long = x</code> and <code>delta_lat = y</code> <code>cartview</code> : <code>gdef = 45, gmin = 0</code> <code>gnomview</code> : <code>gdef = 5, gmin = 0</code> <code>mollview</code> : <code>gdef = 45, gmin = 10</code> <code>orthview</code> : <code>gdef = 45, gmin = 10</code> Note that the graticule will rotate with the sphere if <b>Rot</b> is set. To outline only the equator set <code>graticule=[360,90]</code> . The automatic labeling of the graticule is controlled by <b>Glsiz</b> The graticule line thickness is controlled via <code>!P.THICK</code> . ( <b>default:</b> 0 [no graticule]) see also: <b>Igraticule</b> , <b>Rot</b> , <b>Coord</b> , <b>Glsiz</b>
/HALF_SKY	—O	if set, only shows only one half of the sky (centered on (0,0) or on the location parametrized by <b>Rot</b> ) instead of the full sky
HBOUND=	all	scalar or vector of up to 3 elements. If <code>Hbound[i]</code> is set to a valid $N_{\text{side}}$ , the routine will overplot the <b>HEALPix</b> pixel boundaries corresponding to that $N_{\text{side}}$ on top of the map. The first $N_{\text{side}}$ will be plotted with solid lines, the second one (if any) with dashes and the third one (if any) with dots. Obviously, better results are obtained for Hbounds elements in growing order. Since 0-valued boundaries are not plotted, but used for linestyle assignment, providing <code>Hbound=[0,4]</code> (or <code>[0,0,4]</code> ) will plot $N_{\text{side}} = 4$ boundaries with dashes (resp. dots), while <code>Hbound=4</code> would plot the same boundaries with solid lines.
/HELP	all	if set, the routine header is printed (by <code>doc_library</code> ) and nothing else is done
/HIST_EQUAL	all	if set, uses a histogram equalized color mapping (useful for non gaussian data field) ( <b>default:</b> uses linear color mapping and puts the level 0 in the middle of the color scale (ie, green for Blue-Red) unless <b>Min</b> and <b>Max</b> are not symmetric) see also: <b>Asinh</b> , <b>Log</b>

name	routines	description
HXSIZE=	all	horizontal dimension (in cm) of the <b>Postscript</b> printout ( <b>default:</b> 26 cm $\simeq$ 10 in) see also: <b>Pxsize</b>
IGLSIZE=	CGMO	character size of the input coordinates graticule labels in units of <b>Charsize</b> . ( <b>default:</b> 0: no labeling of graticules). see also: <b>Charsize</b> , <b>Igraticule</b>
IGRATICULE=	CGMO	if set, puts a graticule (ie, longitude and latitude grid) in the <i>input</i> astrophysical coordinates. See <b>Graticule</b> for conventions and details. If both <b>Graticule</b> and <b>Igraticule</b> are set, the latter will be represented with dashes. The automatic labeling of the graticule is controlled by <b>Iglsiz</b> e ( <b>default:</b> 0 [no graticule]) see also: <b>Graticule</b> , <b>Rot</b> , <b>Coord</b> , <b>Iglsiz</b> e
JPEG=	all	string containing the name of a <i>lossless</i> .JPEG output file if set to 1 : outputs the plot in <code>plot_projection.jpeg</code> , where <i>projection</i> is either <b>azimequid</b> , <b>cartesian</b> , <b>gnomic</b> , <b>mollweide</b> or <b>orthographic</b> , if set to a file name : output the plot in that file ( <b>default:</b> no .JPEG done) see also: <b>Crop</b> , <b>Fits</b> , <b>Gif</b> , <b>Map_out</b> , <b>Png</b> , <b>Preview</b> , <b>Pdf</b> , <b>Ps</b> , and <b>Retain</b>
LATEX=	all	if set to 1 or 2, enables L <sup>A</sup> T <sub>E</sub> X handling of character strings such as <b>Titleplot</b> , <b>Subtitle</b> and <b>Units</b> – if set to 2 <i>with</i> <b>PS</b> or <b>PDF</b> outputs, those strings (and the <b>graticule labels</b> ) will be processed by genuine L <sup>A</sup> T <sub>E</sub> X and inserted in the final PS or PDF file using <i>psfrag</i> package (requires the ubiquitous <b>latex</b> and its <i>color</i> , <i>geometry</i> , <i>graphicx</i> and <i>psfrag</i> packages as well as <b>dvips</b> ). In this case, the <b>Pfonts</b> settings will be ignored. Note that <b>cgPStoRASTER</b> , <b>ImageMagick convert</b> and/or <b>GraphicsMagick gm convert</b> can be used to turn a PS or PDF file into high resolution GIF, JPEG or PNG file. Beware that the option <b>latex=2</b> may not work properly under versions 0.9.5 and older of <b>gdl</b> . – if set to 1, with whatever output ( <b>GIF</b> , <b>JPEG</b> , <b>PDF</b> , <b>PNG</b> , <b>PS</b> , or <b>X</b> ) LaTeX is partially emulated with <b>TeXtoIDL</b> routines, which are now shipped with <b>HEALPix</b> (no extra requirements). In this case, <b>Pfonts</b> settings can be used. ( <b>default:</b> 0, no LaTeX handling)

name	routines	description
/LOG	all	display the log of map. This is intended for application to positive definite maps only, eg. Galactic foreground emission templates; for arbitrary maps, use /ASINH instead. see also: <a href="#">Asinh</a> , <a href="#">Factor</a> , <a href="#">Hist.Equal</a> , <a href="#">Offset</a>
MAP_OUT=	all	variable that will contain the projected map on output. Except for the color mapping, all the keywords and options apply to the projected map, ie: its size is determined by <a href="#">PX_SIZE</a> (and <a href="#">Pysize</a> when applicable), its angular resolution by <a href="#">RESO_ARCMIN</a> when applicable, its orientation and coordinates by <a href="#">ROT</a> and <a href="#">COORD</a> respectively, ... Unobserved pixels, and pixels outside the sphere, take value <a href="#">!healpix.bad_value</a> ( $= -1.6375 \cdot 10^{30}$ ). see also: <a href="#">Fits</a>
MAX=	all	Set the maximum value for the plotted signal ( <b>default:</b> is to use the actual signal maximum).
MIN=	all	Set the minimum value for the plotted signal ( <b>default:</b> is to use the actual signal minimum).
/NESTED	all	specify that the online data is ordered in the nested scheme
/NO_DIPOLE	—MO	if set (and <a href="#">Gal.cut</a> is not set) the best fit monopole *and* dipole over all valid pixels are removed; if <a href="#">Gal.cut</a> is set to $b > 0$ , the best monopole and dipole fit is performed on all valid pixels with $ \text{galactic latitude}  > b$ (in deg) and is removed from all valid pixels ( <b>default:</b> 0 (no monopole or dipole removal)) can NOT be used together with <a href="#">No_monopole</a> see also: <a href="#">Gal.cut</a> , <a href="#">No_monopole</a>
/NO_MONOPOLE	—MO	if set (and <a href="#">Gal.cut</a> is not set) the best fit monopole over all valid pixels is removed; if <a href="#">Gal.cut</a> is set to $b > 0$ , the best monopole fit is performed on all valid pixels with $ \text{galactic latitude}  > b$ (in deg) and is removed from all valid pixels ( <b>default:</b> 0 (no monopole removal)) can NOT be used together with <a href="#">No_dipole</a> see also: <a href="#">Gal.cut</a> , <a href="#">No_dipole</a>
/NOBAR	all	if set, the color bar (or the color wheel used when <a href="#">Polarization=2</a> ) is hidden
/NOLABELS	all	if set, color bar labels (min and max) are not present, ( <b>default:</b> labels are present)
/NOPOSITION	—G—	if set, the astronomical location of the map central point is not indicated

name	routines	description
OFFSET=	<b>all</b>	<p>scalar additive factor to be applied to the valid data the data plotted is of the form <math>\text{Factor} * (\text{data} + \text{Offset})</math></p> <p>This does not affect the flagged pixels</p> <p>can be used together with ASINH or LOG</p> <p>When used with TRUECOLORS, OFFSET can be a 3-element vector.</p> <p>see also: : <b>ASINH</b>, <b>Factor</b>, <b>LOG</b>, <b>TRUECOLORS</b> (<b>default:</b> 0.0)</p>
OUTLINE=	<b>CGMO</b>	<p>IDL structure, structure of structures, or array of structures, containing the description of one (or several) outline(s) to be overplotted on the final map.</p> <p>For each contour or point list, the corresponding (sub)structure should contain the following fields:</p> <ul style="list-style-type: none"> <li>– 'COORD': coordinate system (either 'C'/'Q', 'G' or 'E') of the contour (same meaning as in <b>Coord</b>)</li> <li>– 'RA': RA/longitude coordinates of the contour vertices (array or scalar)</li> <li>– 'DEC': Dec/latitude coordinates of the contour vertices (array or scalar)</li> </ul> <p>and can optionally contain the fields:</p> <ul style="list-style-type: none"> <li>– 'LINE[STYLE]': (optional, scalar) <b>+2</b>: black dashes, <b>+1</b>: black dots, <b>0</b>: black solid (default), <b>-1</b>: black dots on white background, <b>-2</b>: black dashes on white background</li> <li>– 'PSY[M]': (optional, scalar) symbol used to represent vertices (same meaning as standard PSYM in IDL. If <math>9 \leq  \text{psym}  \leq 46</math>, D. Fanning's <b>cgSYMCAT.PRO</b> symbols definition will be used; for example, <math>\text{psym}=9</math> is an open circle). If <math>\leq 0</math>, the vertices are represented with the chosen symbols, and connected by arcs of geodesics; if <math>&gt; 0</math>, only the vertices are shown (<b>default:</b> 0)</li> <li>– 'SYM[SIZE]': (optional, scalar) vertice symbol size (same meaning as SYMSIZE in IDL), (<b>default:</b> 1)</li> </ul> <p>The line and symbol thickness can be controlled (indirectly) via !P.THICK.</p> <p>Notes: when applicable, the vertices are connected by segments of geodesics. To obtain a better looking outline, increase the number of vertices provided. The outline does not have to be closed. The procedure will NOT attempt to close the outline. Several outlines can be overplotted at once by gathering the respective structures into one meta-structure or an array.</p> <p>see also: <b>Coord</b>, <b>Graticule</b></p>

name	routines	description
PDF=	all	<p>string containing the name of a .PDF output</p> <p>if set to 0 : no PDF output</p> <p>if set to 1 : outputs the plot in <code>plot_projection.pdf</code>, where <i>projection</i> is either <code>azimequid</code>, <code>cartesian</code>, <code>gnomic</code>, <code>mollweide</code> or <code>orthographic</code>,</p> <p>if set to a file name : outputs the plot in that file</p> <p>(default: 0)</p> <p>The PDF file is produced from a PostScript file using the script <code>epstopdf</code> now shipped with <b>HEALPix</b> . Note that <code>epstopdf</code> usually requires a fully functional implementation of the fairly widespread <code>gs</code>, aka <b>Ghostscript</b>, which may however not be available on the computation dedicated nodes of some computer clusters.</p> <p>see also: <a href="#">Preview</a>, <a href="#">Gif</a>, <a href="#">Jpeg</a>, <a href="#">Png</a>, <a href="#">Ps</a></p>
PFonts=	all	<p>2-element vector of integers <math>[p_0, p_1]</math> selecting the default IDL font of character strings such as the <a href="#">Subtitle</a>, <a href="#">Title</a> and <a href="#">Units</a>.</p> <p><math>p_0</math> must be in <math>\{-1, 0, 1\}</math> and selects the origin of the fonts among -1: <a href="#">Hershey Vector</a>, 0: <a href="#">Device Specific</a> and 1: <a href="#">True Type</a> Fonts.</p> <p><math>p_1</math> must be in <math>\{2, \dots, 20\}</math> and selects the starting font of the character strings as described <a href="#">here</a>.</p> <p>The font can be changed within each string with embedded formatting commands, as discussed on <a href="http://www.exelisvis.com/docs/Fonts_and_Colors.html">http://www.exelisvis.com/docs/Fonts_and_Colors.html</a>.</p> <p>(default: [-1,6], corresponding to the Hershey vector font of type 'Complex Roman', and is equivalent to typing <code>!p.font=-1</code> and prepending the <a href="#">Subtitle</a>, <a href="#">Title</a> and <a href="#">Units</a> strings with <code>'!6'</code>).</p> <p>Note that PFonts will be ignored if <a href="#">Latex</a>=2 and <a href="#">PDF</a> or <a href="#">PS</a> are set.</p>
PNG=	all	<p>string containing the name of a .PNG output</p> <p>if set to 1 : outputs the plot in <code>plot_projection.png</code>, where <i>projection</i> is either <code>azimequid</code>, <code>cartesian</code>, <code>gnomic</code>, <code>mollweide</code> or <code>orthographic</code>,</p> <p>if set to a file name : outputs the plot in that file</p> <p>Please note that the resulting PNG image might not always look as expected. The reason for this is problems with 'backing store' in the IDL-routine TVRD. Please read the IDL documentation for more information.</p> <p>(default: no .PNG done)</p> <p>see also: <a href="#">Crop</a>, <a href="#">Fits</a>, <a href="#">Gif</a>, <a href="#">Jpeg</a>, <a href="#">Map_out</a>, <a href="#">Preview</a>, <a href="#">Pdf</a>, <a href="#">Ps</a>, and <a href="#">Retain</a></p>

name	routines	description
POLARIZATION= <b>all</b>		<p>if set to</p> <p>0 no polarization information is plotted.</p> <p>1 the AMPLITUDE <math>P = \sqrt{(U^2 + Q^2)}</math> of the polarization is plotted (as long as the input data contains polarization information (ie, Stokes parameter Q and U for each pixel))</p> <p>2 the ANGLE <math>\phi = \tan^{-1}(U/Q)/2</math> of the polarization is plotted  Note: the angles are color coded with a fixed color table (independent of <b>Colt</b>)</p> <p>3 –the temperature is color coded (with a color table defined by <b>Colt</b>)  –and the polarization is overplotted as headless VECTORS  Polarization can then be a 3-element vector (the first element being 3). The second element controls the average length of vectors (<b>default:</b> 1), while the third one controls the distance between vectors (<b>default:</b> 1). Non-positive values are replaced by 1.</p> <p>(<b>default:</b> 0)</p> <p><b>Note:</b> The representation of the polarization direction (options 2 and 3 above), include the effects of the rotations and/or changes or astronomical coordinates (controlled by <b>ROT</b> and <b>COORD</b> respectively) but do not include the effects of the distortions induced by the projection from the sphere to the plan. Because the polarization usually has more power at small scales, it must generally be represented on maps of small patches of the sky to remain legible, in which case the projection-induced distortions are small.</p>

name	routines	description
/PREVIEW	all	if set, there is a 'ghostview' preview of the postscript file or a 'xv' preview of the gif file see also: <a href="#">Gif</a> , <a href="#">Jpeg</a> , <a href="#">Pdf</a> , <a href="#">Png</a> , and <a href="#">Ps</a>
PS=	all	if set to 0 : no PostScript output if set to 1 : outputs the plot <code>plot_projection.ps</code> , where <i>projection</i> is either <code>azimequid</code> , <code>cartesian</code> , <code>gnomic</code> , <code>mollweide</code> or <code>orthographic</code> , if set to a file name : outputs the plot in that file ( <b>default:</b> 0) see also: <a href="#">Preview</a> , <a href="#">Gif</a> , <a href="#">Jpeg</a> , <a href="#">Pdf</a> , <a href="#">Png</a>
PXSIZE=	all	set the number of horizontal screen_pixels or postscript_color_dots of the plot (useful for high definition color printer) or elements of the output map ( <b>default:</b> 800 (Mollview and full sky Orthview), 600 (half sky Orthview), 500 (Cartview and Gnomonic)) see also: <a href="#">FITS</a> , <a href="#">GIF</a> , <a href="#">JPEG</a> , <a href="#">MAP_OUT</a> , <a href="#">PDF</a> , <a href="#">PNG</a> , <a href="#">PS</a> .
PYSIZE=	ACG-	set the number of vertical screen_pixels or postscript_color_dots of the plot ( <b>default:</b> <a href="#">Pxsize</a> ).
RESO_ARCMIN=	ACG-	size of screen_pixels or postscript_color_dots in arcmin ( <b>default:</b> 1.5) see also: <a href="#">FITS</a> , <a href="#">GIF</a> , <a href="#">JPEG</a> , <a href="#">MAP_OUT</a> , <a href="#">PDF</a> , <a href="#">PNG</a> , <a href="#">PS</a> .
RETAIN=	all	specifies the type of backing store to use for direct graphics windows in {0,1,2}. ( <b>default:</b> 2). See IDL documentation for details.
ROT=	all	vector with 1, 2 or 3 elements specifying the rotation angles in DEGREES to apply to the map in the 'output' coordinate system (see <a href="#">Coord</a> ) = ( lon0, [lat0, rat0]) lon0 : longitude of the point to be put at the center of the plot the longitude increases Eastward, ie to the left of the plot ( <b>default:</b> 0) lat0 : latitude of the point to be put at the center of the plot ( <b>default:</b> 0) rot0 : anti clockwise rotation to apply to the sky around the center (lon0, lat0) before projecting ( <b>default:</b> 0)



name	routines	description
/SAVE	all	if set, assumes that <b>File</b> is in IDL saveset format, the variable saved should be DATA
/SHADED	—O	if set, the orthographic sphere is shaded, using a Phong model, to emulate 3D viewing. The sphere is illuminated by isotropic ambient light plus a single light source. Can NOT be used with <b>GIF</b> .
/SILENT	all	if set, the program runs silently
STAGGER=	—O	Scalar or 2 element vector: <ul style="list-style-type: none"> <li>– if <b>stagger</b>[0] is in ]0,2], three copies of the same sphere centered respectively at <b>[-stagger[0], 0, stagger[0]]</b> (expressed in radius units) along the plot horizontal axis are shown in ORTHOGRAPHIC projection</li> <li>– if set, <b>stagger</b>[1] defines the angle of rotation (in degrees) applied to the left and right partial spheres: the <i>lhs</i> sphere is rotated downward by the angle provided, while the <i>rhs</i> one is rotated upward. Rotations are swapped if <b>FLIP</b> is set.</li> </ul> Currently <b>can not</b> be used with <b>Graticule</b> nor <b>igraticule</b>
SUBTITLE=	all	String containing the subtitle to the plot see also: <b>Titleplot</b> , <b>Latex</b>
TITLEPLOT=	all	String containing the title of the plot, if not set the title will be File see also: <b>Subtitle</b> , <b>Latex</b>
TRANSPARENT=	all	If set to 1, the input data pixels with value <b>!healpix.bad_value</b> ( $= -1.6375 \cdot 10^{30}$ ) will appear totally transparent on the output PNG file (instead of the usual grey or <b>BAD_COLOR</b> ). If set to 2, the background pixels will be transparent (instead of the usual white or <b>BG_COLOR</b> ) If set to 3, both the grey and white pixels will look transparent. Active only in conjunction with <b>PNG</b>
TRUECOLORS=	all	if the input data is of the form [Npix,3], then the 3 fields are respectively understood as Red, Green, Blue True-Color channels, and the <b>color table</b> is ignored. <ul style="list-style-type: none"> <li>– If set to 1, the mapping field-intensity to color is done for the 3 channels at once. (see also: <b>Factor</b>, <b>Offset</b>)</li> <li>– If set to 2, that mapping is done for each channel separately (in that case, <b>MIN</b> and <b>MAX</b> keywords are ignored).</li> </ul>

name	routines	description
UNITS=	all	String containing the units, to be put on the right hand side of the color bar, overrides the value read from the input file, if any see also: <b>Nobar</b> , <b>Nolabels</b> , <b>Latex</b>
WINDOW=	all	IDL window index (integer) – if WINDOW < 0: virtual window: no visible window opened. Can be used with <b>PNG</b> , <b>JPEG</b> , or <b>GIF</b> , in particular if those files are larger than the screen. <b>Note:</b> The Z buffer will be used instead of the X server, allowing much faster production of the image over a slow network – if WINDOW in [0, 31]: the specified IDL window with index WINDOW is used (or reused). Can be used to have a sequence of images appear in the same window – if WINDOW > 31: a free (=unused) window with a random index > 31 will be created and used. ( <b>default:</b> 32, if X server properly set; -1, otherwise)
XPOS=	all	The X position on the screen of the lower left corner of the window, in device coordinate
YPOS=	all	The Y position on the screen of the lower left corner of the window, in device coordinate

## DESCRIPTION

mollview reads in a **HEALPix** sky map in FITS format and generates a Mollweide projection of it, that can be visualized on the screen or exported in a GIF, JPEG, PNG, PDF or Postscript file. mollview allows the selection of the coordinate system, map size, color table, color bar inclusion, linear, log, hybrid or histogram equalised color scaling, maximum and minimum range for the plot, plot-title *etc.* It also allows the representation of the polarization field.

---

## RELATED ROUTINES

This section lists the routines related to **mollview**.

idl	version 6.4 or more is necessary to run mollview
gv, ghostview	gv, ghostview or a similar facility is required to view the Postscript or PDF images generated by mollview.
display, xv	display, xv or a similar facility is required to view the GIF/JPEG/PNG image generated by mollview (a browser can also be used).
synfast, smoothing	These F90 <b>HEALPix</b> facilities will generate the FITS format sky maps to be input to mollview.
isynfast, ismoothing	These IDL routines will generate the FITS format sky maps to be input to mollview.
cartview	IDL facility to generate a Cartesian projection of a <b>HEALPix</b> map.
cartcursor	interactive cursor to be used with cartview
gnomview	IDL facility to generate a gnomonic projection of a <b>HEALPix</b> map.
gnomcursor	interactive cursor to be used with gnomview
mollview	IDL facility to generate a Mollweide projection of a <b>HEALPix</b> map.
mollcursor	interactive cursor to be used with mollview
orthview	IDL facility to generate an orthographic projection of a <b>HEALPix</b> map.
orthcursor	interactive cursor to be used with orthview
planck_colors	creates color tables used in Planck 2013 publications

---

## EXAMPLES: #1

```
mollview, 'planck100GHZ-LFI.fits', min=-100, max=100, /graticule, $
      title='Simulated Planck LFI Sky Map at 100GHz'
```

mollview reads in the map 'planck100GHZ-LFI.fits' and generates an output image in which the temperature scale has been set to lie between  $\pm 100$  ( $\mu\text{K}$ ), a **graticule** with a 45 degree step in longitude and latitude is drawn, and the **title** 'Simulated Planck LFI Sky Map at 100GHz' appended to the image.

---

## EXAMPLES: #2

```
map = findgen(48)
triangle= create_struct('coord','G','ra',[0,80,0],'dec',[40,45,65])
mollview,map, graticule=[45,30],rot=[10,20,30],$
    title='Mollweide projection',subtitle='mollview', $
    outline=triangle
```

makes a Mollweide projection of a pixel index map (see Figure 1c on page 117) after an arbitrary **rotation**, with a **graticule** grid (with a  $45^\circ$  step in longitude and  $30^\circ$  in latitude) and an arbitrary (triangular) **outline**

---

## EXAMPLES: #3

```
map = findgen(48)
mycommand = 'x=findgen(64)/10.  & ' + $
    'plot,x,sin(x),pos=[0.8,0.8,0.99,0.99],/noerase &' + $
    'xyouts,0.5,0.5,''Hello World !'',/normal,charsize=2,align=0.5'
mollview,map, execute=mycommand, png='plot_example_execute.png',$
    /preview,/graticule,/glsize
```

produces a PNG file containing a Mollweide projection of a pixel index map with labeled graticules, a simple sine wave in the upper right corner, and some greetings, as shown on Figure 2 on page 117

---

## EXAMPLES: #4

```
pixel = 164indgen(400000)
signal = pixel * 10.0
file = 'cutsky.fits'
write_fits_cut4, file, pixel+100000, signal, nside=32768, /ring
gnomview, file, rot=[0,90], grat=30, title='high res.  cut-sky map'
```

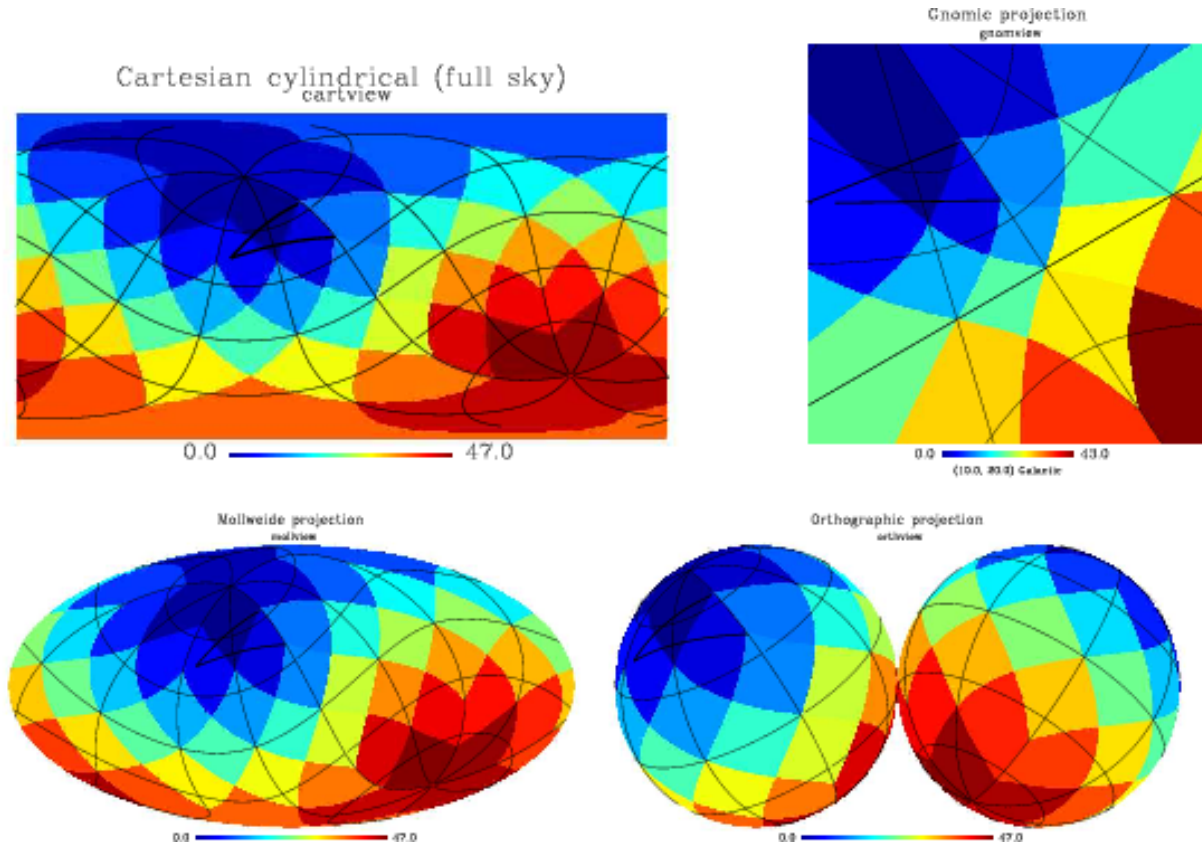


Figure 1: Figures produced by `cartview`, `gnomview`, `mollview` and `orthview`, see respective routine documentation for details.

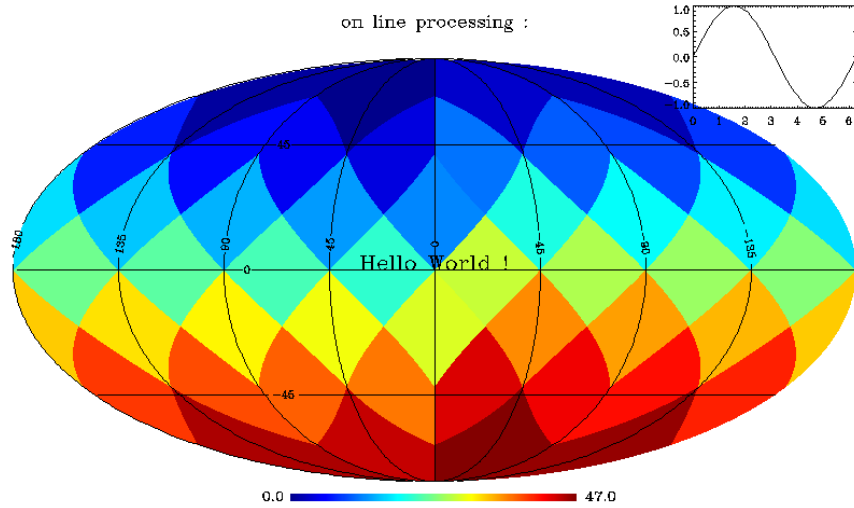


Figure 2: Figure produced by Example #3 .

produces and plots a high resolution map (6.4 arcsec/pixel), in which only a very small subset of pixels is observed

---

## EXAMPLES: #5

```
file = 'wmap_band_iqumap_r9_5yr_K_v3.fits'
mollview, file, title='Linear Color Scale', /silent
mollview, file,/asinh,title='Sinh!u-1!n Color Scale' , /silent
mollview, file,/hist, title='Histogram Equalized Color Scale', /silent
mollview, file,/log, title='Log Scale', /silent
```

produces Mollweide projections of the same map (here the WMAP-5yr K band) with various color scales: linear, Inverse Hyperbolic Sine, Histogram Equalized, and Log. See Figure 3 on page 118

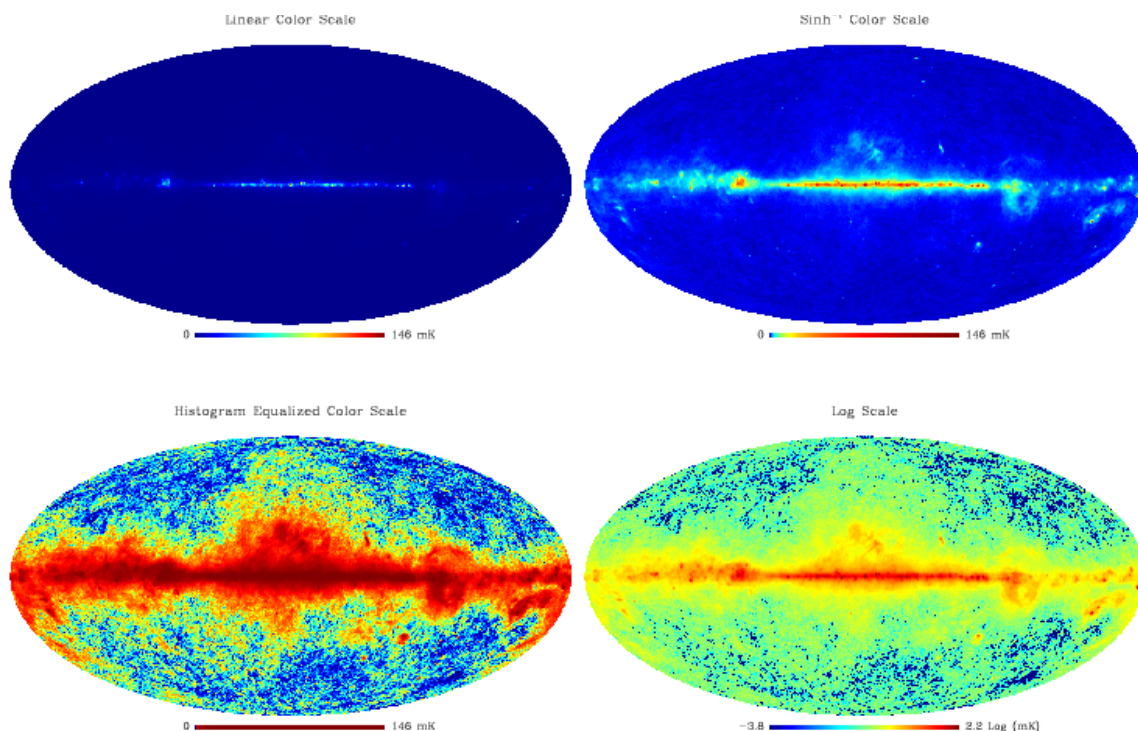


Figure 3: Illustration (generated by Example #5) of the various color scales available.

---

## EXAMPLES: #6



```
mollview, 'HFI_SkyMap_217_2048_R1.10_nominal.fits', $  
  colt='planck2', asinh=2, factor=1.e6, offset=-1.33e-4, $  
  min=-1.e3, max=1.e7, title='Planck @ 217GHz', charsize=2
```

Illustrates the application of the second color table created by `planck_colors` to the visualization of Planck data at 217GHz (see Fig. 4 on page 119)

Planck @ 217GHz

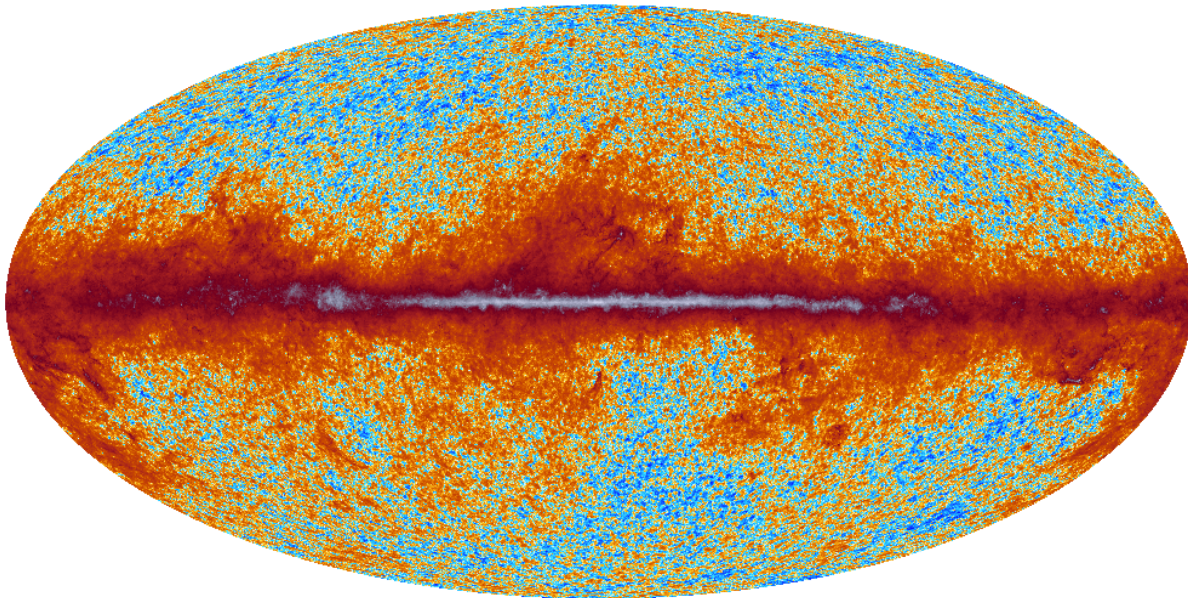


Figure 4: Illustration (generated by Example #6) of the application of Planck color table #2 to a Planck sky map.

# neighbours\_nest

**Location in HEALPix directory tree:** `src/idl/toolkit/neighbours_nest.pro`

This IDL facility returns the number and indices of the topological immediate neighbours of a central pixel. The pixels are ordered in a clockwise sense (when watching the sphere from the outside) about the central pixel with the southernmost pixel in first element. For the four pixels in the southern corners of the equatorial faces which have two equally southern neighbours the routine returns the southwestern pixel first and proceeds clockwise.

---

**FORMAT**            IDL> neighbours\_nest (Nside, Ipix0, Listpix[,Nneigh])

---

## QUALIFIERS

Nside	<b>HEALPix</b> resolution parameter (scalar integer), should be a valid Nside (power of 2)
Ipix0	NESTED-scheme index of central pixel in [0,12*Nside <sup>2</sup> -1]
Listpix	output: list of neighbouring pixel (NESTED scheme index) of size <b>Nneigh</b>
Nneigh	optional output: number of neighbours of pixel #Ipix0. Usually 8, sometimes 7 (for 8 particular pixels) or 6 (if Nside=1)

---

**DESCRIPTION** neighbours\_nest calls `pix2xy_nest` to find location of central pixel within the pixelation base-face, and then `xy2pix_nest` to find neighbouring pixels within the same face, or one of the bit manipulation routines if the neighbouring pixel is on a different base-face.

---

## RELATED ROUTINES

This section lists the routines related to **neighbours\_nest**.



	idl	version 6.4 or more is necessary to run neighbours_nest .
	neighbours_ring	returns topological immediate neighbouring pixels of a given central pixel, using RING indexing.
	query_disc, query_polygon, query_strip, query_triangle	render the list of pixels enclosed respectively in a given disc, polygon, latitude strip and triangle
	nest2ring, ring2nest	conversion between NESTED and RING indices

---

**EXAMPLE:**

```
neighbours_nest , 4, 1, list, nneigh
print,nneigh,list
```

will return:8        90 0 2 3 6 4 94 91, listing the NESTED-indexed 8 neighbors of pixel #1 for Nside=4

# neighbours\_ring

Location in HEALPix directory tree: `src/idl/toolkit/neighbours_ring.pro`

This IDL facility returns the number and indices of the topological immediate neighbours of a central pixel. The pixels are ordered in a clockwise sense (when watching the sphere from the outside) about the central pixel with the southernmost pixel in first element. For the four pixels in the southern corners of the equatorial faces which have two equally southern neighbours the routine returns the southwestern pixel first and proceeds clockwise.

---

**FORMAT** IDL> neighbours\_ring (Nside, Ipix0, Listpix[,Nneigh])

---

## QUALIFIERS

Nside	<b>HEALPix</b> resolution parameter (scalar integer), should be a valid Nside (power of 2)
Ipix0	RING-scheme index of central pixel in $[0, 12*Nside^2-1]$
Listpix	output: list of neighbouring pixel (RING scheme index) of size <b>Nneigh</b>
Nneigh	optional output: number of neighbours of pixel #Ipix0. Usually 8, sometimes 7 (for 8 particular pixels) or 6 (if Nside=1)

---

**DESCRIPTION** neighbours\_ring calls ring2nest, neighbours\_nest and nest2ring

---

## RELATED ROUTINES

This section lists the routines related to **neighbours\_ring**.

idl	version 6.4 or more is necessary to run neighbours_ring .
-----	---

<code>neighbours_nest</code>	returns topological immediate neighbouring pixels of a given central pixel, using NESTED indexing.
<code>query_disc, query_polygon, query_strip, query_triangle</code>	render the list of pixels enclosed respectively in a given disc, polygon, latitude strip and triangle
<code>nest2ring, ring2nest</code>	conversion between NESTED and RING indices

---

### EXAMPLE:

```
neighbours_ring , 4, 1, list, nneigh
print,nneigh,list
```

will return: 8      16 6 5 0 3 2 8 7 listing the RING-indexed 8  
neighbors of pixel #1 for Nside=4

# nest2uniq

Location in HEALPix directory tree: `src/idl/toolkit/nest2uniq.pro`

This IDL facility turns  $N_{\text{side}}$  and (NESTED) pixel index into the Unique Identifier .

---

**FORMAT** IDL> nest2uniq, **Nside**, **Pnest**, **Puniq** [, /HELP]

---

## QUALIFIERS

Nside	(IN, scalar or vector Integer) The <b>HEALPix</b> $N_{\text{side}}$ parameter(s)
Pnest	(IN, scalar or vector Integer) (NESTED scheme) pixel identification number(s) in the range $\{0, 12N_{\text{side}}^2 - 1\}$ . If <b>Nside</b> is a scalar, <b>Pnest</b> can be a scalar or a vector, if <b>Nside</b> is a vector, <b>Pnest</b> must be a vector of the same size
Puniq	(OUT, same size as <b>Pnest</b> ) The <b>HEALPix</b> Unique pixel identifier(s).

---

## KEYWORDS

/HELP	If set, a documentation header is printed out, and the routine exits
-------	--

---

**DESCRIPTION** nest2uniq turns the parameter  $N_{\text{side}}$  (a power of 2) and the pixel index  $p$  into the Unique ID number  $u = p + 4N_{\text{side}}^2$ . See ["The Unique Identifier scheme"](#) section in ["HEALPix Introduction Document"](#) for more details.

---

## EXAMPLE:

```
nest2uniq, [1, 2, 4], [0, 0, 0], puniq
print, puniq
```

returns

4        16        64

since the first pixels ( $p = 0$ ) at  $N_{\text{side}} = 1, 2$  and  $4$  are respectively the pixels with Unique ID numbers 4, 16 and 64.

---

## RELATED ROUTINES

This section lists the routines related to **nest2uniq**.

<b>uniq2nest</b>	Transforms Unique <b>HEALPix</b> pixel ID number into Nside and Nested pixel number
<b>pix2xxx,...</b>	to turn NESTED pixel index into sky coordinates and back

---

Location in HEALPix directory tree: `src/idl/toolkit/npix2nside.pro`

This IDL facility provides the **HEALPix** resolution parameter `Nside` corresponding to `Npix` pixels over the full sky.

## QUALIFIERS

Npix	number of pixels over the full sky (scalar integer), should be a valid Npix ( $N_{\text{pix}} = 12N_{\text{side}}^2$ with $N_{\text{side}}$ power of 2 in $\{1, \dots, 2^{29}\}$ )
Nside	on output: resolution parameter if Npix is valid, -1 otherwise

ERROR= error flag, set to 1 on output if Npix is NOT valid,  
or stays to 0 otherwise.

NAME	DESCRIPTION
<b>DESCRIPTION</b>	<code>npix2nside</code> checks that the given <code>Npix</code> is valid ( $N_{\text{pix}} = 12N_{\text{side}}^2$ with $N_{\text{side}}$ a power of 2 in $\{1, \dots, 2^{29}\}$ ) and then computes the corresponding resolution parameter $N_{\text{side}}$ .

This section lists the routines related to `npix2inside`.

idl	version 6.4 or more is necessary to run npix2nside
	.
nside2npix	computes Npix corresponding to Nside
pix2xxx, ang2xxx, vec2xxx, ...	conversion between vector or angles and pixel index and vice-versa

<code>vec2pix, pix2vec</code>	conversion between vector and pixel index
<code>nest2ring, ring2nest</code>	conversion between NESTED and RING indices

---

**EXAMPLE:**

```
Nside = npix2nside(49152, ERROR=error)
```

Nside will be 64 because 49152 is a valid pixel number ( $=12 \cdot 64^2$  and 64 is a power of 2), and error will be 0

---

**EXAMPLE:**

```
Nside = npix2nside(49151, ERROR=error)
```

Nside will be -1 and error: 1, because 49151 is not a valid number of **HEALPix** pixels over the full sky.

# nside2npix

Location in HEALPix directory tree: `src/idl/toolkit/nside2npix.pro`

This IDL facility provides the number of pixels `Npix` over the full sky corresponding to resolution parameter `Nside`.

---

**FORMAT** IDL> `Npix=NSIDE2NPIX (Nside[, ERROR=])`

---

## QUALIFIERS

<code>Nside</code>	<b>HEALPix</b> resolution parameter (scalar integer), should be a valid <code>Nside</code> (power of 2 $\leq 2^{29}$ )
<code>Npix</code>	number of pixels, $N_{\text{pix}} = 12 * N_{\text{side}}^2$ if <code>Nside</code> is a valid resolution parameter or -1 otherwise

---

## KEYWORDS

<code>ERROR=</code>	error flag, set to 1 on output if <code>Nside</code> is NOT valid, or stays to 0 otherwise.
---------------------	---

---

**DESCRIPTION** `nside2npix` checks that the given `Nside` is valid (power of 2 in  $\{1, \dots, 2^{29}\}$ ) and then computes the corresponding number of pixels  $N_{\text{pix}} = 12 N_{\text{side}}^2$ .

---

## RELATED ROUTINES

This section lists the routines related to **`nside2npix`**.

<code>idl</code>	version 6.4 or more is necessary to run <code>nside2npix</code> .
<code>npix2nside</code>	computes <code>Nside</code> corresponding to <code>Npix</code>
<code>pix2xxx</code> , <code>ang2xxx</code> , <code>vec2xxx</code> , ...	conversion between vector or angles and pixel index and vice-versa
<code>vec2pix</code> , <code>pix2vec</code>	conversion between vector and pixel index
<code>nest2ring</code> , <code>ring2nest</code>	conversion between NESTED and RING indices



---

**EXAMPLE:**

```
Npix = nside2npix(256, ERROR=error)
```

Npix will be 786432 the number of pixels over the full sky for the **HEALPix** resolution parameter 256 and error will be 0

---

**EXAMPLE:**

```
Npix = nside2npix(248, ERROR=error)
```

Npix will be -1 and error: 1, because 248 is not a valid value for a **HEALPix** resolution parameter

# nside2ntemplates

Location in HEALPix directory tree: `src/idl/toolkit/nside2ntemplates.pro`

This IDL facility provides the number of template pixels `Ntemplates` corresponding to resolution parameter `Nside`. Each template pixel has a different shape that *can not* be matched (by rotation or reflexion) to that of any of the other templates.

**FORMAT** IDL> `Ntemplates=NSIDE2NTEMPLATES`  
`(Nside[,ERROR=])`

## QUALIFIERS

<code>Nside</code>	<b>HEALPix</b> resolution parameter (scalar integer), should be a valid <code>Nside</code> (power of 2 in $\{1, \dots, 8192\}$ )
<code>Ntemplates</code>	number of templates

## KEYWORDS

<code>ERROR=</code>	error flag, set to 1 on output if <code>Nside</code> is NOT valid, or stays to 0 otherwise.
---------------------	---

**DESCRIPTION** `nside2ntemplates` outputs the number of template pixels

$$N_{\text{template}} = \frac{1 + N_{\text{side}}(N_{\text{side}} + 6)}{4}.$$

If the argument  $N_{\text{side}}$  is not valid, a warning is issued and the error flag is raised.

## RELATED ROUTINES

This section lists the routines related to `nside2ntemplates`.

<code>idl</code>	version 6.4 or more is necessary to run <code>nside2ntemplates</code> .
------------------	---

---

<code>template_pixel_ring</code>	
<code>template_pixel_nest</code>	return the template pixel associated with any <b>HEALPix</b> pixel
<code>same_shape_pixels_ring</code>	
<code>same_shape_pixels_nest</code>	return the ordered list of pixels having the same shape as a given pixel template

---

**EXAMPLE:**

```
Ntemplates = nside2ntemplates(256, ERROR=error)
```

Ntemplates will be 16768 the number of template pixels for the **HEALPix** resolution parameter 256 and error will be 0

## orthcursor

---

Location in HEALPix directory tree: `src/idl/visu/orthcursor.pro`

This IDL facility provides a point-and-click interface for finding the astronomical location, value and pixel index of the pixels nearest to the pointed position on a orthographic projection of a **HEALPix** map.

---

**FORMAT**            IDL>    ORTHCURSOR,    [`cursor_type=`,  
                         `file_out=`]

---

## QUALIFIERS

see `mollcursor`

---

**DESCRIPTION** `orthcursor` should be called immediately after `orthview`. It gives the longitude, latitude, map value and pixel number corresponding to the cursor position in the window containing the map generated by `orthview`. For more details, or in case of problems under **Mac OS X**, see `mollcursor`.

---

## RELATED ROUTINES

This section lists the routines related to **orthcursor**.

see `mollcursor`

---

## EXAMPLE:

`orthcursor`

After `orthview` has read in a map and generated its orthographic projection, `orthcursor` is run to determine the position and flux of bright synchrotron sources, for example.

# orthview

Location in HEALPix directory tree: `src/idl/visu/orthview.pro`

This IDL facility provides a means to visualise a full sky or half sky orthographic projection (projection onto a tangent plane from a point located at infinity) of **HEALPix** and COBE Quad-Cube maps in an IDL environment. It also offers the possibility to generate GIF, JPEG, PDF, PNG and Postscript color-coded images of the projected map. The projected (but not color-coded) data can also be output in FITS files and IDL arrays.

## FORMAT

```
IDL> ORTHVIEW, File [, Select] [, AS-
    INH=, BAD.COLOR=, BG.COLOR=, CHARSIZE=,
    CHARTHICK=, COLT=, COORD=, /CROP, EXECUTE=,
    FACTOR=, FG.COLOR=, FITS=, /FLIP, GAL_CUT=,
    GIF=, GLSIZE=, GRATICULE=, /HALF_SKY, HBOUND=,
    /HELP, /HIST_EQUAL, HXSIZE=, IGLSIZE=, IGRATIC-
    ULE=, JPEG=, LATEX=, /LOG, MAP_OUT=, MAX=,
    MIN=, /NESTED, /NO_DIPOLE, /NO_MONOPOLE,
    /NOBAR, /NOLABELS, /NOPOSITION, OFFSET=, OUT-
    LINE=, PDF=, PFonts=, PNG=, POLARIZATION=,
    /PREVIEW, PS=, PXSIZ=, PYSIZ=, RESO_ARCMIN=,
    RETAIN=, ROT=, /SAVE, /SHADED, /SILENT, STAG-
    GER=, SUBTITLE=, TITLEPLOT=, TRANSPARENT=,
    TRUECOLORS=, UNITS=, WINDOW=, XPOS=, YPOS=]
```

## QUALIFIERS

For a full list of qualifiers see [mollview](#)

---

## KEYWORDS

For a full list of keywords see [mollview](#)

---

## DESCRIPTION

`orthview` reads in a **HEALPix** sky map in FITS format and generates an orthographic projection of it, that can be visualized on the screen or exported in a GIF, JPEG, PNG, PDF or Postscript file. `orthview` allows the selection of the coordinate system, map size, color table, color bar inclusion, linear, log, hybrid or histogram equalised color scaling, maximum and minimum range for the plot, plot-title *etc.* It also allows the representation of the polarization field.

---

## RELATED ROUTINES

This section lists the routines related to **`orthview`**.

see [mollview](#)

---

## EXAMPLE:

```
map = findgen(48)
triangle= create_struct('coord','G','ra',[0,80,0],'dec',[40,45,65])
orthview,map,/online,graticule=[45,30],rot=[10,20,30],$
    title='Orthographic projection',subtitle='orthview' $
outline=triangle
```

makes an orthographic projection of map (see Figure 1d on page 117) after an arbitrary rotation, with a graticule grid (with a 45° step in longitude and 30° in latitude) and an arbitrary triangular outline

# **pix2xxx, ang2xxx, vec2xxx, nest2ring, ring2nest**

Location in HEALPix directory tree: `src/idl/toolkit/`

These routines provide conversion between pixel number in the **HEALPix** map and  $(\theta, \phi)$  or  $(x, y, z)$  coordinates on the sphere. Some of these routines are listed here.

## **QUALIFIERS**

name (dim.)	type	in/out	description
nside	scalar integer	IN	$N_{\text{side}}$ parameter for the <b>HEALPix</b> map.
ipnest(n)	vector integer	—	pixel identification number in NESTED scheme over the range $\{0, N_{\text{pix}} - 1\}$ .
ipring(n)	vector integer	—	pixel identification number in RING scheme over the range $\{0, N_{\text{pix}} - 1\}$ .
theta(n)	vector double	—	colatitude in radians measured southward from north pole in $\{0, \pi\}$
phi(n)	vector double	—	longitude in radians, measured eastward in $\{0, 2\pi\}$ .
vector(n,3)	array double	—	three dimensional cartesian position vector $(x, y, z)$ . The north pole is $(0, 0, 1)$ . An output vector is normalised to unity. The coordinates are ordered as follows $x(0), \dots, x(n-1)$ , $y(0), \dots, y(n-1)$ , $z(0), \dots, z(n-1)$
vertex(n,3,4)	array double	optional OUT	three dimensional cartesian position vector $(x, y, z)$ . Contains the location of the four vertices (=corners) of a pixel in the order North, West, South, East. The coordinates are ordered as follows $x_N(0), \dots, x_N(n-1)$ , $y_N(0), \dots, y_N(n-1)$ , $z_N(0), \dots, z_N(n-1)$ , $x_W(0), \dots, x_W(n-1)$ , $y_W(0), \dots, y_W(n-1)$ , $z_W(0), \dots, z_W(n-1)$ , and so on with South and East vertices

## **ROUTINES:**

`pix2ang_ring, nside, ipring, theta, phi`

renders `theta` and `phi` coordinates of the nominal pixel center given the pixel number `ipring` and a map resolution parameter `nside`.

`pix2vec_ring, nside, ipring, vector [,vertex]`

renders cartesian vector coordinates of the nominal pixel center given the pixel number `ipring` and a map resolution parameter `nside`. Optionally returns the location of the 4 vertices for the pixel(s) under consideration

`ang2pix_ring, nside, theta, phi, ipring`

renders the pixel number `ipring` for a pixel which, given the map resolution parameter `nside`, contains the point on the sphere at angular coordinates `theta` and `phi`.

`vec2pix_ring, nside, vector, ipring`

renders the pixel number `ipring` for a pixel which, given the map resolution parameter `nside`, contains the point on the sphere at cartesian coordinates `vector`.

`pix2ang_nest, nside, ipnest, theta, phi`

renders `theta` and `phi` coordinates of the nominal pixel center given the pixel number `ipnest` and a map resolution parameter `nside`.

`pix2vec_nest, nside, ipnest, vector [,vertex]`

renders cartesian vector coordinates of the nominal pixel center given the pixel number `ipnest` and a map resolution parameter `nside`. Optionally returns the location of the 4 vertices for the pixel(s) under consideration

`ang2pix_nest, nside, theta, phi, ipnest`

renders the pixel number `ipnest` for a pixel which, given the map resolution parameter `nside`, contains the point on the sphere at angular coordinates `theta` and `phi`.

`vec2pix_nest, nside, vector, ipnest`

renders the pixel number `ipnest` for a pixel which, given the map resolution parameter `nside`, contains the point on the sphere at cartesian coordinates `vector`.

`nest2ring, nside, ipnest, ipring`

performs conversion from NESTED to RING pixel number.

`ring2nest, nside, ipring, ipnest`

performs conversion from RING to NESTED pixel number.



---

## RELATED ROUTINES

This section lists the routines related to **pix2xxx**, **ang2xxx**, **vec2xxx**, **nest2ring**, **ring2nest**.

	idl	version 6.4 or more is necessary to run pix2xxx, ang2xxx,... .
	<b>npix2nside</b>	computes $N_{\text{side}}$ (resolution) corresponding to $N_{\text{pix}}$ (total pixel number)
	<b>nside2npix</b>	computes $N_{\text{pix}}$ corresponding to $N_{\text{side}}$
	<b>ang2vec</b> , <b>vec2ang</b>	geometrical conversion between position angles and position vector
	<b>nest2uniq</b> , <b>uniq2nest</b>	conversion of standard pixel index to/from Unique ID number

---

## EXAMPLE:

```
pix2ang_ring, 256, [17,1000], theta, phi
print,theta,phi
```

```
returns
0.0095683558      0.070182078
2.8797933        5.4620872
position of the two pixels #17 and 1000 in the RING scheme
with parameter  $N_{\text{side}}=256$ .
```

Location in HEALPix directory tree: `src/idl/visu/planck_colors.pro`

This IDL facility provides RGB color tables suitable for visualization of sky maps dominated by CMB or featuring foreground, and modify current color table. Those color tables can then be implemented in `cartview`, `gnomview`, `mollview` or `orthview` and were used in Planck 2013 publications

```

FORMAT          IDL>      PLANCK_COLORS,      option,
                    [GET=rgb, /HELP, /SHOW]

```

option required input for color table generation, must be either 1 or 2:

- 1: creates the 'parchment' Blue-red color table suitable for maps dominated by Gaussian signal (eg, CMB)
- 2: creates a Blue-red-white color table suitable for maps with high dynamic signal (eg, Galactic foreground)

GET=rgb	optional output, contains the newly created RGB color table in a [256, 3] array
/HELP	if set, prints extended help
/SHOW	if set, the chosen color table is shown in a new window

**DESCRIPTION** `planck.colors` creates a set of RGB color tables suitable for specific purpose, and modify the current IDL color table accordingly (using `TVLCT`). See below the example applications. The created color table can also be output as a 256\*3 array, or shown in a new window

---

## RELATED ROUTINES

This section lists the routines related to **planck\_colors**.

idl	version 6.4 or more is necessary to run planck_colors.
cartview, gnomview mollview, orthview	visualization routines that can make use of the color tables created in planck_colors (via keyword <b>colt</b> )
loadct	IDL routine to set current color table to one of the predefined IDL color tables (thus reverting the effect of planck_colors).

---

## EXAMPLE:

```
planck_colors, 1, /show
planck_colors, 2, /show
```

Create and show the two color tables (see Fig. 5 on page 139)

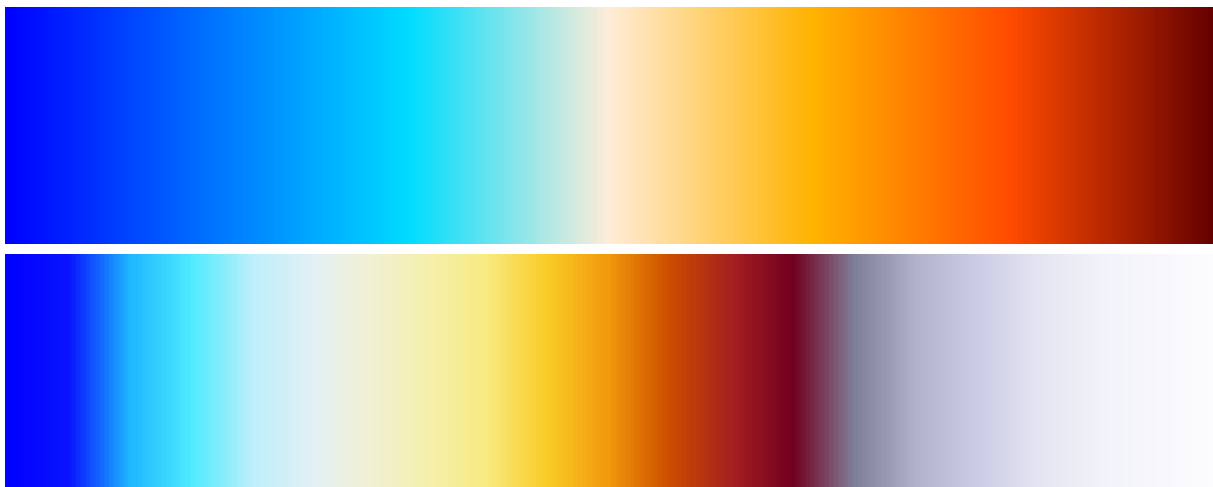


Figure 5: Illustration of the color tables created by planck\_colors.

## query\_disc

Location in HEALPix directory tree: `src/idl/toolkit/query_disc.pro`

This IDL facility provides a means to find the index of all pixels within an angular distance `Radius` from a defined center.

---

**FORMAT** IDL> query\_disc , `Nside`, `Vector0`, `Radius`, `Listpix`, [`Nlist`, `/DEG`, `/NESTED`, `/INCLUSIVE`]

---

## QUALIFIERS

<code>Nside</code>	<b>HEALPix</b> resolution parameter used to index the pixel list (scalar integer)
<code>Vector0</code>	position vector of the disc center (3 elements vector) NB : the norm of <code>Vector0</code> does not have to be one, what is consider is the intersection of the sphere with the line of direction <code>Vector0</code> .
<code>Radius</code>	radius of the disc (in radians, unless <b>DEG</b> is set), (scalar real)
<code>Listpix</code>	on output: list of ordered index for the pixels found within a radius <code>Radius</code> of the position defined by <code>vector0</code> . The RING numbering scheme is used unless the keyword <b>NESTED</b> is set. (= -1 if the radius is too small and no pixel is found)
<code>Nlist</code>	on output: number of pixels in <code>Listpix</code> (=0 if no pixel is found).

---

## KEYWORDS

<code>/DEG</code>	if set <b>Radius</b> is in degrees instead of radians
<code>/NESTED</code>	if set, the output list uses the NESTED numbering scheme instead of the default RING
<code>/INCLUSIVE</code>	if set, all the pixels overlapping (even partially) with the disc are listed, otherwise only those whose center lies within the disc are listed

---

**DESCRIPTION** query\_disc finds the pixels within the given disc in a selective way WITHOUT scanning all the sky pixels. The numbering scheme of the output list and the inclusiveness of the disc can be changed

---

## RELATED ROUTINES

This section lists the routines related to **query\_disc** .

idl	version 6.4 or more is necessary to run query_disc .
ang2pix, pix2ang	conversion between angles and pixel index
vec2pix, pix2vec	conversion between vector and pixel index
query_disc, query_polygon, query_strip, query_triangle	render the list of pixels enclosed respectively in a given disc, polygon, latitude strip and triangle

---

## EXAMPLE:

```
query_disc , 256L, [.5,.5,0.], 10., listpix, nlist, /Deg, /Nest
```

On return listpix contains the index of the (5982) pixels within 10 degrees from the point on the sphere having the direction [.5,.5,0.]. The pixel indices correspond to the Nested scheme with resolution 256.

## query\_polygon

---

Location in HEALPix directory tree: `src/idl/toolkit/query_polygon.pro`

This IDL facility provides a means to find the index of all pixels belonging to a spherical polygon defined by its vertices

---

**FORMAT** IDL> query\_polygon , Nside, Vlist, Listpix,  
[Nlist, HELP=, NESTED=, INCLUSIVE=]

---

## QUALIFIERS

Nside	<b>HEALPix</b> resolution parameter used to index the pixel list (scalar integer)
Vlist	3D cartesian position vector of the polygon vertices. Array of dimension (n,3) where n is the number of vertices
Listpix	on output: list of ordered index for the pixels found in the polygon. The RING numbering scheme is used unless the keyword <b>NESTED</b> is set. (= -1 if the polygon is too small and no pixel is found)
Nlist	on output: number of pixels in Listpix (=0 if no pixel is found).

---

## KEYWORDS

HELP=	if set, the documentation header is printed out and the routine exits
NESTED=	if set, the output list uses the <b>NESTED</b> numbering scheme instead of the default <b>RING</b>
INCLUSIVE=	if set, all the pixels overlapping (even partially) with the polygon are listed, otherwise only those whose center lies within the polygon are listed

---

**DESCRIPTION** query\_polygon finds the pixels within the given polygon in a selective way WITHOUT scanning all the sky pixels. The polygon should be convex, or have only one concave vertex. The edges should not intersect each other. The numbering scheme of the output list and the inclusiveness of the polygon can be changed

---

## RELATED ROUTINES

This section lists the routines related to **query\_polygon** .

idl	version 6.4 or more is necessary to run query_polygon .
ang2pix, pix2ang	conversion between angles and pixel index
vec2pix, pix2vec	conversion between vector and pixel index
query_disc, query_polygon, query_strip, query_triangle	render the list of pixels enclosed respectively in a given disc, polygon, latitude strip and triangle

---

## EXAMPLE:

```
query_polygon , 256L, [[0,1,1,0],[0,0,1,1],[1,0,-1,0]], listpix, nlist
```

On return listpix contains the index of the (131191) pixels contained in the polygon with vertices of cartesian coordinates (0,0,1), (1,0,0), (1,1,-1) and (0,1,0). The pixel indices correspond to the RING scheme with resolution 256.

## query\_strip

**Location in HEALPix directory tree:** `src/idl/toolkit/query_strip.pro`

This IDL facility provides a means to find the index of all pixels belonging to a latitude strip defined by its bounds

---

**FORMAT** IDL> query\_strip , Nside, Theta1, Theta2, Listpix, [Nlist, NESTED=, INCLUSIVE=, HELP=]

---

## QUALIFIERS

Nside	<b>HEALPix</b> resolution parameter used to index the pixel list (scalar integer)
Theta1	colatitude lower bound in radians measured from North Pole (between 0 and $\pi$ ).
Theta2	colatitude upper bound in radians measured from North Pole (between 0 and $\pi$ ). If <code>theta1 &lt; theta2</code> , the pixels lying in <code>[theta1, theta2]</code> are output, otherwise, the pixel lying in <code>[0, theta2]</code> and those lying in <code>[theta1, <math>\pi</math>]</code> are output.
Listpix	on output: list of ordered index for the pixels found in the strip. The RING numbering scheme is used unless the keyword <b>NESTED</b> is set. ( <code>=-1</code> if the strip is too small and no pixel is found)
Nlist	on output: number of pixels in Listpix ( <code>=0</code> if no pixel is found).

---

## KEYWORDS

NESTED=	if set, the output list uses the NESTED numbering scheme instead of the default RING
INCLUSIVE=	if set, all the pixels overlapping (even partially) with the strip are listed, otherwise only those whose center lies within the strip are listed



/HELP if set, the routine prints its documentation header and exits.

---

**DESCRIPTION** query\_strip finds the pixels within the given strip in a selective way WITHOUT scanning all the sky pixels. The numbering scheme of the output list and the inclusiveness of the strip can be changed

---

## RELATED ROUTINES

This section lists the routines related to **query\_strip** .

idl	version 6.4 or more is necessary to run query_strip .
ang2pix, pix2ang	conversion between angles and pixel index
vec2pix, pix2vec	conversion between vector and pixel index
query_disc, query_polygon, query_triangle	render the list of pixels enclosed respectively in a given disc, polygon and triangle

---

## EXAMPLE:

```
query_strip , 256, 0.75*!PI, !PI/5, listpix, nlist, /nest
```

Returns the NESTED pixel index of all pixels with colatitude in  $[0, \pi/5]$  and those with colatitude in  $[3\pi/4, \pi]$

# query\_triangle

Location in HEALPix directory tree: `src/idl/toolkit/query_triangle.pro`

This IDL facility provides a means to find the index of all pixels belonging to a spherical triangle defined by its vertices

---

**FORMAT** IDL> query\_triangle , Nside, Vector1, Vector2, Vector3, Listpix, [Nlist, NESTED=, INCLUSIVE=]

---

## QUALIFIERS

Nside	<b>HEALPix</b> resolution parameter used to index the pixel list (scalar integer)
Vector1	3D cartesian position vector of the triangle first vertex
Vector2	3D cartesian position vector of the triangle second vertex
Vector3	3D cartesian position vector of the triangle third vertex NB : the norm of Vector* does not have to be one, what is considered is the intersection of the sphere with the line of direction Vector*.
Listpix	on output: list of ordered index for the pixels found in the triangle. The RING numbering scheme is used unless the keyword <b>NESTED</b> is set. (= -1 if the triangle is too small and no pixel is found)
Nlist	on output: number of pixels in Listpix (=0 if no pixel is found).

---

## KEYWORDS

NESTED=	if set, the output list uses the <b>NESTED</b> numbering scheme instead of the default <b>RING</b>
INCLUSIVE=	if set, all the pixels overlapping (even partially) with the triangle are listed, otherwise only those whose center lies within the triangle are listed

---

**DESCRIPTION** query\_triangle finds the pixels within the given triangle in a selective way WITHOUT scanning all the sky pixels. The numbering scheme of the output list and the inclusiveness of the triangle can be changed

---

## RELATED ROUTINES

This section lists the routines related to **query\_triangle** .

idl	version 6.4 or more is necessary to run query_triangle .
ang2pix, pix2ang	conversion between angles and pixel index
vec2pix, pix2vec	conversion between vector and pixel index
query_disc, query_polygon, query_strip, query_triangle	render the list of pixels enclosed respectively in a given disc, polygon, latitude strip and triangle

---

## EXAMPLE:

```
query_triangle , 256L, [1,0,0],[0,1,0],[0,0,1], listpix, nlist
```

On return listpix contains the index of the (98560) pixels lying in the octant ( $x > 0, y > 0, z > 0$ ). The pixel indices correspond to the RING scheme with resolution 256.

## read\_fits\_cut4

Location in HEALPix directory tree: `src/idl/fits/read_fits_cut4.pro`

This IDL facility reads a cut sky **HEALPix** map from a FITS file according to the **HEALPix** convention. The format used for the FITS file follows the one used for Boomerang98 and is adapted from COBE/DMR. This routine can also be used to read polarized cut sky map, where each Stokes parameter is stored in a different extension of the same FITS file.

---

**FORMAT** IDL> READ\_FITS\_CUT4 , File, Pixel, Signal[, N\_Obs, Serror, EXTENSION=, HDR=, XHDR=, NSIDE=, ORDERING=, COORDSYS=]

## QUALIFIERS

File	name of a FITS file in which the map is to be written
Pixel	(OUT, LONG vector), index of observed (or valid) pixels
Signal	(OUT, FLOAT vector), value of signal in each observed pixel
N_Obs	(OUT, LONG or INT vector, Optional), number of observation per pixel
Serror	(OUT, FLOAT vector, Optional), <i>rms</i> of signal in pixel. For white noise, this is $\propto 1/\sqrt{n\_obs}$

## KEYWORDS

EXTENSION= (IN, optional),  
0 based number of extension to read. Extension 0 contains the temperature information, while extensions 1 and 2 contain respectively the Q and U Stokes parameters related information.  
(**default:** 0)

HDR=	(OUT, optional), String array containing the primary header.
XHDR=	(OUT, optional), String array containing the extension header.
NSIDE=	(OUT, optional), returns on output the <b>HEALPix</b> resolution parameter, as read from the FITS header. Set to -1 if not found
ORDERING=	(OUT, optional), returns on output the pixel ordering, as read from the FITS header. Either 'RING' or 'NESTED' or ' ' (if not found).
COORDSYS=	(OUT, optional), returns on output the astrophysical coordinate system used, as read from FITS header (value of keywords COORDSYS or SKYCOORD)

---

## DESCRIPTION

---

## RELATED ROUTINES

This section lists the routines related to **read\_fits\_cut4** .

idl	version 6.4 or more is necessary to run read_fits_cut4
<b>write_fits_cut4</b>	This <b>HEALPix</b> IDL facility can be used to generate the FITS format <i>cut-sky</i> maps compliant with <b>HEALPix</b> convention and readable by read_fits_cut4 .
<b>read_fits_cut4</b> , <b>read_fits_map</b> <b>read_tqu</b> , <b>read_fits_s</b>	<b>HEALPix</b> IDL routines to read cut-sky maps, full-sky maps, polarized full-sky maps and arbitrary data sets from FITS files
sxpar	This IDL routine (included in <b>HEALPix</b> package) can be used to extract FITS keywords from the header(s) HDR or XHDR read with read_fits_cut4 .

## read\_fits\_map

Location in HEALPix directory tree: `src/idl/fits/read_fits_map.pro`

This IDL facility reads in a **HEALPix** map from a FITS file.

---

**FORMAT**            IDL> READ\_FITS\_MAP , **File**, **T\_sky**, [**Hdr**,  
                          **Exthdr**, **PIXEL=**, **SILENT=**, **NSIDE=**, **OR-**  
                          **DERING=**, **COORDSYS=**, **EXTENSION=**,  
                          **HELP=**]

---

## QUALIFIERS

File	name of a FITS file containing the <b>HEALPix</b> map in an extension or in the image field
T_sky	variable containing on output the <b>HEALPix</b> map
Hdr	(optional), string variable containing on output the FITS primary header
Exthdr	(optional), string variable containing on output the FITS extension header
PIXEL=	(optional), pixel number to read from or pixel range to read (in the order of appearance in the file), starting from 0. if $\geq 0$ scalar : read from pixel to the end of the file if two elements array : reads from pixel[0] to pixel[1] (included) if absent : read the whole file
NSIDE=	(optional), returns on output the <b>HEALPix</b> resolution parameter, as read from the FITS header. Set to -1 if not found
ORDERING=	(optional), returns on output the pixel ordering, as read from

the FITS header. Either 'RING' or 'NESTED' or ' ' (if not found).

COORDSYS= (optional),  
returns on output the astrophysical coordinate system used, as read from FITS header (value of keywords COORDSYS or SKYCOORD)

Extension= (optional),  
extension unit to be read from FITS file: either its 0-based ID number (ie, 0 for first extension *after* primary array) or the case-insensitive value of its EXTNAME keyword. If absent, all available extensions are read.

---

## KEYWORDS

HELP= if set, an extensive help is displayed and no file is read

SILENT= if set, no message is issued during normal execution

---

**DESCRIPTION** read\_fits\_map reads in a **HEALPix** sky map from a FITS file, and outputs the variable `T_sky`, where the optional variables `Hdr` and `Exthdr` contain respectively the primary and extension headers. According to **HEALPix** convention, the map should be stored as a FITS file binary table extension. Note: the routine `read_tqu` which requires less memory is recommended when reading *large polarized* maps.

---

## RELATED ROUTINES

This section lists the routines related to `read_fits_map`.

idl version 6.4 or more is necessary to run read\_fits\_map

`read_fits_cut4`, `read_fits_map`  
`read_tqu`, `read_fits_s` **HEALPix** IDL routines to read cut-sky maps, full-sky maps, polarized full-sky maps and arbitrary data sets from FITS files

sxpar	This IDL routine (included in <b>HEALPix</b> package) can be used to extract FITS keywords from the header(s) <code>Hdr</code> or <code>Xhdr</code> read with <code>read_fits_map</code> .
synfast	This <b>HEALPix</b> facility will generate the FITS format sky map that can be read by <code>read_fits_map</code> .
<code>write_fits_map</code>	This <b>HEALPix</b> IDL facility can be used to generate the FITS format sky maps compliant with <b>HEALPix</b> convention and readable by <code>read_fits_map</code> .

---

**EXAMPLE:**

```
read_fits_map, 'planck100GHZ-LFI.fits', map, hdr, xhdr, /silent
```

`read_fits_map` reads in the file 'planck100GHZ-LFI.fits' and outputs the **HEALPix** map in `map`, the primary header in `hdr` and the extension header in `xhdr`.



# read\_fits\_s

Location in HEALPix directory tree: `src/idl/fits/read_fits_s.pro`

This IDL facility reads a FITS file into an IDL structure.

---

**FORMAT** IDL> READ\_FITS\_S , File, Prim\_stc,  
[Xten\_stc, COLUMNS=, EXTENSION=,  
/HELP, /MERGE]

---

## QUALIFIERS

File	name of a FITS file containing the healpix map(s) in an extension or in the image field
Prim_stc	variable containing on output an IDL structure with the following fields: - primary header (tag : 0, tag name : HDR) - primary image (if any, tag : 1, tag name : IMG)
Xten_stc	(optional), variable containing on output an IDL structure with the following fields: - extension header (tag : 0, tag name : HDR) - data column 1 (if any, tag : 1, tag name given by TTYPE1 (with all spaces removed and only letters, digits and underscore) - data column 2 (if any, tag : 2, tag name given by TTYPE2) ...
Columns=	(optional), list of columns to be read from a binary table can be a list of integer (1 based) indexing the columns positions or a list of names matching the TTYPE* of the columns by default, all columns are read
Extension=	(optional), extension unit to be read from FITS file: either its 0-based ID number (ie, 0 for first extension <i>after</i> primary array) or the case-insensitive value of its EXTNAME keyword. ( <b>default:</b> 0)

---

## KEYWORDS

/HELP	if set, an extensive help is displayed and no file is read
/MERGE	if set <b>Prim_stc</b> contains : - the concatenated primary and extension header (tag name : HDR) - primary image (if any, tag name : IMG) - data column 1 ... and <b>Exten_stc</b> is set to 0 ( <b>default:</b> :) not set (or set to 0)

---

**DESCRIPTION** `read_fits_s` reads in any type of FITS file (Image, Binary table or Ascii table) and outputs the data in IDL structures

---

## RELATED ROUTINES

This section lists the routines related to **read\_fits\_s** .

idl	version 6.4 or more is necessary to run <code>read_fits_s</code>
synfast	This <b>HEALPix</b> facility will generate the FITS format sky map that can be read by <code>read_fits_s</code> .
<code>read_fits_cut4</code> , <code>read_fits_map</code> <code>read_tqu</code> , <code>read_fits_s</code>	<b>HEALPix</b> IDL routines to read cut-sky maps, full-sky maps, polarized full-sky maps and arbitrary data sets from FITS files
<code>write_fits_sb</code>	This <b>HEALPix</b> IDL facility can be used to generate FITS format sky maps readable by <code>read_fits_s</code> .

---

## EXAMPLE:

```
read_fits_s , 'dmr_skymap_90a_4yr.fits', pdata, xdata
```

`read_fits_s` reads in the file 'dmr\_skymap\_90a\_4yr.fits'. On output, `pdata` contains the primary header and `xdata` is a structure whose first field is the extension header, and the other fields are vectors with respective tag names PIXEL, SIGNAL, N\_OBS, ERROR, ... (see `help,/struc,xdata`)

# read\_tqu

**Location in HEALPix directory tree:** `src/idl/fits/read_tqu.pro`

This IDL facility reads a temperature+polarization Healpix map (T,Q,U) from a binary table FITS file, with optionally the error (dT,dQ,dU) and correlation (dQU, dTU, dTQ) from separate extensions

---

**FORMAT** IDL> READ\_TQU , *File*, *TQU*, [*Extension=*,  
*Hdr=*, *Xhdr=*, */HELP*, *Nside=*, *Ordering=*,  
*Coordsys=*]

---

## QUALIFIERS

File	name of a FITS file from which the maps are to be read
TQU	: array of Healpix maps of size ( $N_{\text{pix}}, 3, \text{n\_ext}$ ) where $N_{\text{pix}}$ is the total number of Healpix pixels on the sky, and $\text{n\_ext} \leq 3$ is the number of extensions read Three maps are available in each extension of the FITS file : -the temperature+polarization Stokes parameters maps (T,Q,U) in extension 0 -the error maps (dT,dQ,dU) in extension 1 (if applicable) -the correlation maps (dQU, dTU, dTQ) in extension 2 (if applicable)
Extension=	(optional), extension unit to be read from FITS file: either its 0-based ID number (ie, 0 for first extension <i>after</i> primary array) or the case-insensitive value of its EXTNAME keyword. If absent, all available extensions are read.
Hdr=	(optional), string variable containing on output the contents of the primary header. (If already present, FITS reserved keywords will be automatically updated).

Xhdr=	(optional), string variable containing on output the contents of the extension header. If several extensions are read, then the extension headers are returned appended into one string array.
Nside=	(optional), returns on output the <b>HEALPix</b> resolution parameter, as read from the FITS header. Set to -1 if not found
Ordering=	(optional), returns on output the pixel ordering, as read from the FITS header. Either 'RING' or 'NESTED' or ' ' (if not found).
Coordsys=	(optional), returns on output the astrophysical coordinate system used, as read from FITS header (value of keywords COORDSYS or SKYCOORD)

---

## KEYWORDS

/HELP	if set, an extensive help is displayed and no file is read
-------	--

---

**DESCRIPTION** `read_tqu` reads out Stokes parameters (T,Q,U) maps for the whole sky into a FITS file. It is also possible to read the error per pixel for each map and the correlation between fields, as subsequent extensions of the same FITS file (see qualifiers above). Therefore the file may have up to three extensions with three maps in each. Extensions can be written together or one by one (in their physical order) using the Extension option

---

## RELATED ROUTINES

This section lists the routines related to **read\_tqu**.

idl	version 6.4 or more is necessary to run <code>read_tqu</code>
synfast	This <b>HEALPix</b> f90 facility can be used to generate temperature+polarization maps that can be read with <code>read_tqu</code>

<code>write_tqu</code>	This <b>HEALPix</b> IDL facility can be used to write out temperature+polarization that can be read by <code>read_tqu</code> .
<code>read_fits_cut4</code> , <code>read_fits_map</code> <code>read_tqu</code> , <code>read_fits_s</code>	<b>HEALPix</b> IDL routines to read cut-sky maps, full-sky maps, polarized full-sky maps and arbitrary data sets from FITS files
<code>read_fits_s</code>	This general purpose <b>HEALPix</b> IDL facility can be used to read into an IDL structure maps contained in binary table FITS files.
<code>sxpar</code>	This IDL routine (included in <b>HEALPix</b> package) can be used to extract FITS keywords from the header(s) HDR or XHDR read with <code>read_tqu</code> .

---

## EXAMPLE:

```
read_tqu, 'map_polarization.fits', TQU, xhdr=xhdr
```

Reads into `TQU` the polarization maps contained in the FITS file 'map\_polarization.fits'. The variable `xhdr` will contain the extension(s) header.

# remove\_dipole

Location in HEALPix directory tree: `src/idl/misc/remove_dipole.pro`

This IDL facility provides a means to fit and remove the dipole and monopole from a **HEALPix** map.

---

**FORMAT** IDL> REMOVE\_DIPOLE, Map [, Weight, BAD\_DATA=, GAL\_CUT=, COORD\_IN=, COORD\_OUT=, Covariance\_Matrix=, Dipole=, Monopole=, /NOREMOVE, NSIDE=, /ONLYMONOPOLE, ORDERING=, PIXEL=, /SILENT, UNITS=, /HELP]

---

## QUALIFIERS

Map	input and output, vector map from which monopole and dipole are to be removed (also used for output). Assumed to be a full sky data set, unless PIXEL is set and has the same size as map
Weight	input, vector, optional same size as map, describe weighting scheme to apply to each pixel for the fit ( <b>default:</b> uniform weight)
BAD_DATA =	scalar float, value given on input to bad pixels ( <b>default:</b> <code>!healpix.bad_value</code> $\equiv -1.6375 \times 10^{30}$ ).
GAL_CUT=	if set to a value larger than 0, the pixels with galactic latitude $ b  < \text{gal\_cut}$ degrees are not considered in the fit. <b>NB:</b> the cut is <i>really</i> done in Galactic coordinates. If the input coordinates are different (see <code>Coord_In</code> ), the map is rotated into galactic before applying the cut.
COORD_IN =	string, map coordinate system (either 'Q' or 'C': equatorial, 'G': galactic or 'E': ecliptic; upper/lower case accepted)

	( <b>default:</b> 'G' (galactic))
COORD_OUT =	string, coordinate system (see above) in which to output dipole vector in variable Dipole ( <b>default:</b> same as coord_in)
Covariance_Matrix =	OUTPUT, scalar (or symmetric 4x4 matrix), covariance of the statistical errors made on monopole (and dipole) determination
Dipole=	OUTPUT, 3d vector, coordinates of best fit dipole (done simultaneously with monopole), same units as input map
Monopole=	OUTPUT, scalar float, value found for the best fit monopole (done simultaneously with dipole), same units as input map
NSIDE=	scalar integer, healpix resolution parameter
ORDERING=	string, ordering scheme (either 'RING' or 'NESTED')
PIXEL=	input, vector, gives the Healpix index of the pixels whose temperature is actually given in map (for cut sky maps). If present, must match Map in size. If absent, it is assumed that the map covers the whole sky.
UNITS=	string, units of the input map

---

## KEYWORDS

/NOREMOVE	if set, the best fit dipole and monopole are computed but not removed (ie, Map is unchanged)
/ONLYMONOPOLE	if set, fit (and remove) only the monopole
/HELP	if set, only display documentation header
/SILENT	if set, the routine works silently

---

**DESCRIPTION** remove\_dipole makes a simultaneous least square fit of the monopole and dipole on all the valid pixels of Map (those with a value different from BAD\_DATA) with a galactic latitude larger in magnitude than GAL\_CUT (in degrees). The position of the pixels on the sky is reconstructed from NSIDE and ORDERING. If Map does not cover the full sky, the actual indices of the concerned pixels should be given in PIXEL

---

## RELATED ROUTINES

This section lists the routines related to **remove\_dipole**.

idl	version 6.4 or more is necessary to run remove_dipole.
-----	--



# reorder

**Location in HEALPix directory tree:** `src/idl/toolkit/reorder.pro`

This IDL facility allows the reordering of a full sky map from NESTED to RING scheme and vice-versa.

---

**FORMAT**            IDL> **Result** = REORDER (**Input\_map** [,  
                              /**HELP**, **In**=, **Out**=, /**N2R**, /**R2N**])

---

## QUALIFIERS

Result	variable containing on output the reordered map
Input_map	variable containing the input map

---

## KEYWORDS

/HELP	if set, the documentation header is printed out and the code exits
In=	specifies the input ordering, can be either 'RING' or 'NESTED'
Out=	specifies the output ordering, can be either 'RING' or 'NESTED'
/N2R	If set, does the NESTED to RING conversion, equivalent to In='NESTED' and Out='RING'
/R2N	If set, does the RING to NESTED conversion, equivalent to In='RING' and Out='NESTED'

---

**DESCRIPTION** reorder allows the reordering of a full sky map from NESTED to RING scheme and vice-versa

---

## RELATED ROUTINES

This section lists the routines related to **reorder** .

idl	version 6.4 or more is necessary to run reorder
-----	---

`ud_grade` downgrades or upgrades a full-sky or cut-sky  
**HEALPix** map.

---

**EXAMPLE:**

```
map_nest = reorder(map_ring, in='ring', out='nest')
```

The RING ordered map `map_ring` is converted to the NESTED map `map_nest`.

# rotate\_coord

---

Location in HEALPix directory tree: `src/idl/misc/rotate_coord.pro`

This IDL facility provides a means to rotate a set of 3D position vectors (and their Stokes parameters Q and U) between to astrophysical coordinate systems or by an arbitrary rotation.

---

**FORMAT** IDL> `Outvec = ROTATE_COORD(Invec`  
`[, /Help, Euler_Matrix=, Inco=, Outco=,`  
`Stokes_Parameters=] )`

## QUALIFIERS

---

Invec	input, array of size (n,3) : set of 3D position vectors
Outvec	output, array of size (n,3) : rotated 3D vectors
Euler_Matrix=	input, array of size (3,3). Euler Matrix describing the rotation to apply to vectors. ( <b>default:</b> unity : no rotation). Can not be used together with a change in coordinates.
Inco=	input, character string (either 'Q' or 'C': equatorial, 'G': galactic or 'E': ecliptic) describing the input coordinate system
Outco=	input, character string (see above) describing the output coordinate system. Can not be used together with Euler_Matrix
Stokes_Parameters=	input and output, array of size (n, 2) : values of the Q and U Stokes parameters on the sphere for each of the input position vector. Q and U are defined wrt the local parallel and meridian and are therefore transformed in a non trivial way in case of rotation

## KEYWORDS

---

/Help	if set, the documentation header is printed and the routine exits
-------	---

---

**DESCRIPTION** `rotate_coord` is a generalisation of the Astro library routine `skyconv`. It allows a rotation of 3D position vectors between two standard astronomic coordinates system but also an arbitrary rotation described by its Euler Matrix. It can also be applied to compute the effect of a rotation on the linear polarization Stokes parameters (Q and U) expressed in local coordinates system at the location of each of the input 3D vectors.

---

## RELATED ROUTINES

This section lists the routines related to **`rotate_coord`**.

<code>idl</code>	version 6.4 or more is necessary to run <code>rotate_coord</code> .
<code>euler_matrix_new</code>	constructs the Euler Matrix for a set of three angles and three axes of rotation

# same\_shape\_pixels\_nest & same\_shape\_pixels\_ring

Location in HEALPix directory tree: `src/idl/toolkit/same_shape_pixels_nest.pro`,  
`src/idl/toolkit/same_shape_pixels_ring.pro`

These IDL facilities provide the ordered list of all **HEALPix** pixels having the same shape as a given template, for a resolution parameter  $N_{\text{side}}$ .

---

**FORMAT** IDL> same\_shape\_pixels\_nest, *Nside*, *Template*, *List\_Pixels\_Nest* [, *Reflexion*, *NREPLICATIONS*=]

---

**FORMAT** IDL> same\_shape\_pixels\_ring, *Nside*, *Template*, *List\_Pixels\_Ring* [, *Reflexion*, *NREPLICATIONS*=]

---

## QUALIFIERS

<i>Nside</i>	(IN, scalar) the <b>HEALPix</b> $N_{\text{side}}$ parameter.
<i>Template</i>	(IN, scalar) identification number of the template (this number is independent of the numbering scheme considered).
<i>List_Pixels_Nest</i>	(OUT, vector) ordered list of NESTED scheme identification numbers for all pixels having the same shape as the template provided
<i>List_Pixels_Ring</i>	(OUT, vector) ordered list of RING scheme identification numbers for all pixels having the same shape as the template provided
<i>Reflexion</i>	(OUT, OPTIONAL, vector) in {0, 3} encodes the transformation(s) to apply to each of the returned pixels to match exactly in shape and position the template provided. 0: rotation around the polar axis only, 1: rotation + East-West swap (ie, reflexion around meridian), 2: rotation + North-South

swap (ie, reflexion around Equator), 3: rotation  
+ East-West and North-South swaps

---

## KEYWORDS

**NREPLICATIONS** (OUT, OPTIONAL, scalar) number of pixels having the same shape as the template. It is also the length of the vectors **List\_Pixel\_Nest**, **List\_Pixel\_Ring** and **Reflexion**. It is either 8, 16,  $4N_{\text{side}}$  or  $8N_{\text{side}}$ .

---

**DESCRIPTION** **same\_shape\_pixels\_nest** & **same\_shape\_pixels\_ring** provide the ordered list of all **HEALPix** pixels having the same shape as a given template, for a resolution parameter  $N_{\text{side}}$ . Depending on the template considered the number of such pixels is either 8, 16,  $4N_{\text{side}}$  or  $8N_{\text{side}}$ . The template pixels are all located in the Northern Hemisphere, or on the Equator. They are chosen to have their center located at

$$z = \cos(\theta) \geq 2/3, \quad 0 < \phi \leq \pi/2, \\ 2/3 > z \geq 0, \quad \phi = 0, \quad \text{or} \quad \phi = \frac{\pi}{4N_{\text{side}}}.$$

They are numbered continuously from 0, starting at the North Pole, with the index increasing in  $\phi$ , and then increasing for decreasing  $z$ .

---

## EXAMPLE:

`same_shape_pixels_ring, 256, 1234, list_pixels, reflexion, nrep=np`

Returns in **list\_pixels** the RING-scheme index of the all the pixels having the same shape as the template #1234 for  $N_{\text{side}} = 256$ . Upon return **reflexion** will contain the reflexions to apply to each pixel returned to match the template, and **np** will contain the number of pixels having that same shape (16 in that case).

---

## RELATED ROUTINES

This section lists the routines related to **same\_shape\_pixels\_nest** & **same\_shape\_pixels\_ring**.

<code>nside2templates</code>	returns the number of template pixel shapes available for a given $N_{\text{side}}$ .
<code>template_pixel_ring</code>	
<code>template_pixel_nest</code>	return the template shape matching the pixel provided

# template\_pixel\_nest & template\_pixel\_ring

Location in HEALPix directory tree: `src/idl/toolkit/template_pixel_nest.pro`,  
`src/idl/toolkit/template_pixel_ring.pro`

These IDL facilities provide the index of the template pixel associated with a given **HEALPix** pixel, for a resolution parameter  $N_{\text{side}}$ .

---

**FORMAT** IDL> `template_pixel_nest`, *Nside*, *Pixel\_Nest*,  
*Template*, *Reflexion*

---

**FORMAT** IDL> `template_pixel_ring`, *Nside*, *Pixel\_Ring*,  
*Template*, *Reflexion*

---

## QUALIFIERS

<i>Nside</i>	(IN, scalar) the <b>HEALPix</b> $N_{\text{side}}$ parameter.
<i>Pixel_Nest</i>	(IN, scalar or vector) NESTED scheme pixel identification number(s) over the range $\{0, 12N_{\text{side}}^2 - 1\}$ .
<i>Pixel_Ring</i>	(IN, scalar or vector) RING scheme pixel identification number(s) over the range $\{0, 12N_{\text{side}}^2 - 1\}$ .
<i>Template</i>	(OUT, scalar or vector) identification number(s) of the template matching in shape the pixel(s) provided (the numbering scheme of the pixel templates is the same for both routines).
<i>Reflexion</i>	(OUT, scalar or vector) in $\{0, 3\}$ encodes the transformation(s) to apply to each pixel provided to match exactly in shape and position its respective template. 0: rotation around the polar axis only, 1: rotation + East-West swap (ie, reflexion around meridian), 2: rotation + North-South swap (ie, reflexion around Equator), 3: rotation + East-West and North-South swaps



---

**DESCRIPTION** `template_pixel_nest` & `template_pixel_ring` provide the index of the template pixel associated with a given **HEALPix** pixel, for a resolution parameter  $N_{\text{side}}$ .

Any pixel can be *matched in shape* to a single of these templates by a combination of a rotation around the polar axis with reflexion(s) around a meridian and/or the equator.

The template pixels are all located in the Northern Hemisphere, or on the Equator. They are chosen to have their center located at

$$\begin{aligned} z = \cos(\theta) &\geq 2/3, & 0 < \phi &\leq \pi/2, \\ 2/3 > z &\geq 0, & \phi &= 0, \quad \text{or} \quad \phi = \frac{\pi}{4N_{\text{side}}}. \end{aligned}$$

They are numbered continuously from 0, starting at the North Pole, with the index increasing in  $\phi$ , and then increasing for decreasing  $z$ .

---

**EXAMPLE:**

`template_pixel_ring, 256, 500000, template, reflexion`

Returns in `template` the index of the template pixel (16663) whose shape matches that of the pixel #500000 for  $N_{\text{side}} = 256$ . Upon return `reflexion` will contain 2, meaning that the template must be reflected around a meridian and around the equator (and then rotated around the polar axis) in order to match the pixel.

---

**RELATED ROUTINES**

This section lists the routines related to `template_pixel_nest` & `template_pixel_ring`.

<code>inside2templates</code>	returns the number of template pixel shapes available for a given $N_{\text{side}}$ .
<code>same_shape_pixels_ring</code>	
<code>same_shape_pixels_nest</code>	return the ordered list of pixels having the same shape as a given pixel template

---

## ud\_grade

**Location in HEALPix directory tree:** `src/idl/toolkit/ud_grade.pro`

This IDL facility provides a means to upgrade/degrade or re-order a full sky or cut-sky **HEALPix** map contained in a FITS file or loaded in memory.

---

**FORMAT** IDL> UD\_GRADE , Map\_in, Map\_out [,  
BAD\_DATA=, HELP=, NSIDE\_OUT=, OR-  
DER\_IN=, ORDER\_OUT=, /PESSIMISTIC]

---

## QUALIFIERS

Map_in	input map: either a character string with the name of a FITS file containing a full-sky or cut-sky Healpix data set, or a memory vector (real, integer, ...) containing a <i>full sky</i> data set.
Map_out	reordered map: if map_in was a filename, map_out should be a filename, otherwise map_out should point to a memory array

---

## KEYWORDS

BAD_DATA =	flag value of missing pixels. ( <b>default:</b> <code>!healpix.bad_value</code> $\equiv -1.6375 \times 10^{30}$ ).
/HELP	if set, the documentation header is printed out and the code exits
NSIDE_OUT =	output resolution parameter, can be larger or smaller than the input one (scalar integer). ( <b>default:</b> same as input: map unchanged or simply reordered)
ORDER_IN =	input map ordering (either 'RING' or 'NESTED') ( <b>default:</b> same as the input FITS keyword ORDERING if applicable).
ORDER_OUT =	output map ordering (either 'RING' or 'NESTED') ( <b>default:</b> same as ORDER_IN).

/PESSIMISTIC      if set, during **degradation** each big pixel containing one bad or missing small pixel is also considered as bad,  
 if not set, each big pixel containing at least one good pixel is considered as good (optimistic) default = 0 (:not set)

---

**DESCRIPTION** ud\_grade can upgrade/degrade a **HEALPix** map using the hierarchical properties of **HEALPix** . It can also reorder a sky map (from NEST to RING and vice-versa). It operates on FITS files as well as on memory variables. Cut-sky operations are only accessible via FITS files. The degradation/upgradation is done assuming an intensive quantity (like temperature) that does not scale with surface area. In case of degradation a big pixel that contains at least one bad small pixel is considered as bad itself. When operating on FITS files, the header information from the input file that is not directly related the ordering/resolution is copied unchanged into the output file.

---

## RELATED ROUTINES

This section lists the routines related to **ud\_grade**.

idl	version 6.4 or more is necessary to run ud_grade .
reorder	reorder a full sky Healpix map.

---

## EXAMPLES: #1

```
ud_grade , 'map_512.fits', 'map_256.fits', nside_out = 256
```

ud\_grade reads the FITS file map\_512.fits (that allegedly contains a map with NSIDE=512), and write in the FITS file map\_256.fits a map degraded to resolution 256, with the same ordering.

---

## EXAMPLES: #2

```
ud_grade , 'map_512.fits', 'map_Nest256.fits', nside_out = 256, $
  order_out = 'NESTED'
```

ud\_grade reads the FITS file map\_512.fits (that allegedly contains a map with NSIDE=512), and writes in the FITS file map\_Nest256.fits a map degraded to resolution 256, with NESTED ordering.

---

## EXAMPLES: #3

```
read_fits_map, 'map_Nest256.fits', mymap  
ud_grade , mymap, mymap2, nside_out = 1024, order_in='NESTED', order_out='RING'
```

mymap is IDL variable containing a **HEALPix** NESTED-ordered map with resolution nside=256. ud\_grade upgrades this map to a resolution of 1024, reorder it to RING and write it in the IDL vector mymap2.

# uniq2nest

Location in HEALPix directory tree: `src/idl/toolkit/uniq2nest.pro`

This IDL facility turns the Unique Identifier into the corresponding  $N_{\text{side}}$  and (NESTED) pixel index.

**FORMAT** IDL> uniq2nest, **Puniq**, **Nside**, **Pnest** [,/HELP]

## QUALIFIERS

Puniq	(IN, scalar or vector Integer) The <b>HEALPix</b> Unique pixel identifier(s). Must be $\geq 4$ .
Nside	(OUT, same size as <b>Puniq</b> ) The <b>HEALPix</b> $N_{\text{side}}$ parameter(s)
Pnest	(OUT, same size as <b>Puniq</b> ) (NESTED scheme) pixel identification number(s) over the range $\{0, 12N_{\text{side}}^2 - 1\}$ .

## KEYWORDS

/HELP	If set, a documentation header is printed out, and the routine exits
-------	--

**DESCRIPTION** uniq2nest turns the Unique ID number  $u = p + 4N_{\text{side}}^2$ , into the parameter  $N_{\text{side}}$  (a power of 2) and the pixel index  $p$ . See ["The Unique Identifier scheme"](#) section in ["HEALPix Introduction Document"](#) for more details.

## EXAMPLE:

```
uniq2nest, [4,16, 64], nside, pnest
print, nside, pnest
```

```
returns
1      2      4
0      0      0
```

since the pixels with Unique ID numbers 4, 16 and 64 are the first pixels ( $p = 0$ ) at  $N_{\text{side}} = 1, 2$  and 4 respectively.

---

## RELATED ROUTINES

This section lists the routines related to **uniq2nest**.

<b>nest2uniq</b>	Transforms Nside and Nested pixel number into Unique <b>HEALPix</b> pixel ID number
<b>pix2xxx, ...</b>	to turn NESTED pixel index into sky coordinates and back

---

# vec2ang

---

Location in HEALPix directory tree: `src/idl/toolkit/vec2ang.pro`

This IDL facility convert the 3D position vectors of points into their angles on the sphere.

---

**FORMAT** IDL> VEC2ANG , **Vector**, **Theta**, **Phi**[, **ASTRO=**]

---

## QUALIFIERS

Vector	input, array, three dimensional cartesian position vector $(x, y, z)$ (not necessarily normalised). The north pole is $(0, 0, 1)$ . The coordinates are ordered as follows $x(0), \dots, x(n-1)$ , $y(0), \dots, y(n-1)$ , $z(0), \dots, z(n-1)$
Theta	output, vector, vector, colatitude in radians measured southward from north pole in $[0, \pi]$ (mathematical coordinates). If ASTRO is set, Theta is the latitude in degrees measured northward from the equator, in $[-90, 90]$ (astronomical coordinates).
Phi	output, vector, longitude in radians measured eastward, in $[0, 2\pi]$ (mathematical coordinates). If ASTRO is set, Phi is the longitude in degree measured eastward, in $[0, 360]$ (astronomical coordinates).

---

## KEYWORDS

ASTRO=	if set Theta and Phi are the latitude and longitude in degrees (astronomical coordinates) instead of the colatitude and longitude in radians (mathematical coordinates).
--------	--

---

**DESCRIPTION** `vec2ang` performs the geometrical transform from the 3D position vectors  $(x, y, z)$  of points into their angles  $(\theta, \phi)$  on the sphere:  $x = \sin \theta \cos \phi$ ,  $y = \sin \theta \sin \phi$ ,  $z = \cos \theta$

---

## RELATED ROUTINES

This section lists the routines related to **`vec2ang`** .

<code>idl</code>	version 6.4 or more is necessary to run <code>vec2ang</code> .
<code>pix2xxx</code> , ...	conversion between vector or angles and pixel index
<code>ang2vec</code>	conversion from angles to position vectors

---

## EXAMPLE:



# write\_fits\_cut4

**Location in HEALPix directory tree:** `src/idl/fits/write_fits_cut4.pro`

This IDL facility writes out a cut sky **HEALPix** map into a FITS file according to the **HEALPix** convention. The format used for the FITS file follows the one used for Boomerang98 and is adapted from COBE/DMR. This routine can be used to store polarized maps, where the information relative to the Stokes parameters I, Q and U are placed in extension 0, 1 and 2 respectively by successive invocation of the routine.

---

**FORMAT** IDL> WRITE\_FITS\_CUT4 , *File*, *Pixel*, *Signal* [, *N\_Obs*, *Serror*, *COORDSYS*=, *EXTENSION*=, *HDR*=, */NESTED*, *NSIDE*=, *ORDERING*=, */POLARISATION*, */RING*, *UNITS*=, *XHDR*=]

---

## QUALIFIERS

File	name of a FITS file in which the map is to be written
Pixel	(LONG or LONG64 vector), index of observed (or valid) pixels
Signal	(FLOAT or DOUBLE vector, same size as Pixel), value of signal in each observed pixel
N_Obs	(LONG or INT or LONG64 vector, Optional, same size as Pixel), number of observation per pixel. If absent, the field <code>N_OBS</code> will take a value of 1 in the output file. If set to a scalar constant, <code>N_OBS</code> will take this value in the output file
Serror	(FLOAT or DOUBLE vector, Optional, same size as Pixel) $rms$ of signal in pixel, for white noise, this is $\propto 1/\sqrt{n\_obs}$ If absent, the field <code>SERROR</code> will take a value of

0.0 in the output file. If set to a scalar constant, **SERROR** will take this value in the output file

---

## KEYWORDS

COORDSYS=	(optional), if set to either 'C', 'E' or 'G', specifies that the Healpix coordinate system is respectively Celestial=equatorial, Ecliptic or Galactic. (The relevant keyword is then added/updated in the extension header, but the map is NOT rotated)
EXTENSION=	(optional), (0 based) extension number in which to write data. ( <b>default:</b> 0). If set to 0 (or not set) <i>a new file is written from scratch</i> . If set to a value larger than 1, the corresponding extension is added or updated, as long as all previous extensions already exist. All extensions of the same file should use the same ORDERING, NSIDE and COORDSYS.
HDR=	(optional), String array containing the information to be put in the primary header.
/NESTED	if set, specifies that the map is in the NESTED ordering scheme see also: Ordering and Ring
NSIDE=	(optional), scalar integer, <b>HEALPix</b> resolution parameter of the data set. The resolution parameter should be made available to the FITS file, either thru this qualifier, or via the header (see XHDR).
ORDERING=	(optional), if set to either 'ring' or 'nested' (case un-sensitive), specifies that the map is respectively in RING or NESTED ordering scheme see also: Nested and Ring The ordering information should be made available to the FITS file, either thru a combination of Ordering/Ring/Nested, or via the header (see XHDR).
/POLARISATION	specifies that file will contain the I, Q and U polarisation Stokes parameter in extensions 0, 1 and

	2 respectively, and sets the FITS header keywords accordingly
/RING	if set, specifies that the map is in the RING ordering scheme see also: Ordering and Nested
UNITS=	(optional), string describing the physical units of the data set (only applies to Signal and Serror)
XHDR=	(optional), String array containing the information to be put in the extension header.

---

## DESCRIPTION

---

## RELATED ROUTINES

This section lists the routines related to **write\_fits\_cut4** .

idl	version 6.4 or more is necessary to run write_fits_cut4
<a href="#">read_fits_cut4</a>	This <b>HEALPix</b> IDL facility can be used to read in maps written by write_fits_cut4 .
<a href="#">write_fits_cut4</a> , <a href="#">write_fits_map</a> <a href="#">write_tqu</a> , <a href="#">write_fits_sb</a>	<b>HEALPix</b> IDL routines to write cut-sky maps, full-sky maps, polarized full-sky maps and arbitrary data sets into FITS files
sxaddpar	This IDL routine (included in <b>HEALPix</b> package) can be used to update or add FITS keywords to the header in HDR and XHDR

---

## EXAMPLES: #1

```
write_fits_cut4 , 'map_cut.fits', pixel, temperature, /ring, nside=32, /pol
```

writes in 'map\_cut.fits' a FITS file containing the temperature measured in a set of **HEALPix** pixel.

---

## EXAMPLES: #2

```
write_fits_cut4 , 'tqu_cut.fits', pixel, temperature, n_t, s_t, $  
    /ring, nside=32, /pol  
write_fits_cut4 , 'tqu_cut.fits', pixel, qstokes, n_q, s_q, $  
    /ring, nside=32, /pol, ext=1  
write_fits_cut4 , 'tqu_cut.fits', pixel, ustokes, n_u, s_u, $  
    /ring, nside=32, /pol, ext=2
```

writes in 'tqu\_cut.fits' a FITS file with three extensions, each of them containing information on the observed pixel, the measured signal, the number of observations and noise per pixel, for the three Stokes parameters I, Q and U respectively. The **HEALPix** ring ordered scheme and the resolution  $N_{\text{side}} = 32$  is assumed.

# write\_fits\_map

Location in HEALPix directory tree: `src/idl/fits/write_fits_map.pro`

This IDL facility writes out a **HEALPix** map into a FITS file according to the **HEALPix** convention

---

**FORMAT** IDL> WRITE\_FITS\_MAP , *File*, *T\_sky*,  
 [*Header*, *Coordsys*=, *Error*=, *Help*=, *Nested*=,  
*Ring*=, *Ordering*=, *Units*=]

---

## QUALIFIERS

File	name of a FITS file in which the map is to be written
T_sky	variable containing the <b>HEALPix</b> map
Header	(optional), string variable containing on input the information to be added to the extension header. (If already present, FITS reserved keywords will be automatically updated).
Coordsys=	(optional), if set to either 'C', 'E' or 'G', specifies that the Healpix coordinate system is respectively Celestial=equatorial, Ecliptic or Galactic. (The relevant keyword is then added/updated in the extension header, but the map is NOT rotated)
Error=	(optional output), will take value 1 if file can not be written
Ordering=	(optional), if set to either 'ring' or 'nested' (case un-sensitive), specifies that the map is respectively in RING or NESTED ordering scheme see also: Nested and Ring
Units=	(optional), string describing the physical units of the data set

---

## KEYWORDS

Help	if set, an extensive help is displayed and no file is written
Nested	if set, specifies that the map is in the NESTED ordering scheme see also: Ordering and Ring
Ring	if set, specifies that the map is in the RING ordering scheme see also: Ordering and Nested

---

**DESCRIPTION** `write_fits_map` writes out the full sky **HEALPix** map `T_sky` into the FITS file `File`. Extra information about the map can be given in `Header` according to the FITS header conventions. Coordinate systems can also be specified by `Coordsys`. Specifying the ordering scheme is compulsory and can be done either in `Header` or by setting `Ordering` or `Nested` or `Ring` to the correct value. If `Ordering` or `Nested` or `Ring` is set, its value overrides what is given in `Header`.

---

## RELATED ROUTINES

This section lists the routines related to `write_fits_map`.

idl	version 6.4 or more is necessary to run <code>write_fits_map</code>
<code>read_fits_map</code>	This <b>HEALPix</b> IDL facility can be used to read in maps written by <code>write_fits_map</code> .
<code>sxaddpar</code>	This IDL routine (included in <b>HEALPix</b> package) can be used to update or add FITS keywords to <code>Header</code>
<code>reorder</code>	This <b>HEALPix</b> IDL routine can be used to reorder a map from NESTED scheme to RING scheme and vice-versa.
<code>write_fits_cut4</code> , <code>write_fits_map</code> <code>write_tqu</code> , <code>write_fits_sb</code>	<b>HEALPix</b> IDL routines to write cut-sky maps, full-sky maps, polarized full-sky maps and arbitrary data sets into FITS files
<code>write_fits_sb</code>	routine to write multi-column binary FITS table

---

**EXAMPLE:**

```
write_fits_map, 'file.fits', map, coordsys='G', ordering='ring'
```

write\_fits\_map writes out the RING ordered map `map` in Galactic coordinates into the file `file.fits`.

## write\_fits\_sb

---

Location in HEALPix directory tree: `src/idl/fits/write_fits_sb.pro`

This IDL facility writes out a **HEALPix** map into a FITS file according to the **HEALPix** convention. It can also write an arbitray data set into a FITS binary table

---

**FORMAT** IDL> WRITE\_FITS\_SB , File, Prim\_Stc[, Xten\_stc, Coordsys=, /Nested, /Ring, Ordering=, /Partial, Nside=, Extension=, /Nohealpix]

## QUALIFIERS

---

File	name of a FITS file in which the map is to be written
Prim_stc	IDL structure containing the following fields: - primary header - primary image Set it to 0 to get an empty primary unit
Xten_stc	(optional), IDL structure containing the following fields: - extension header - data column 1 - data column 2 ... NB: because of some astron routines limitation, avoid using the single letters 'T' or 'F' as tagnames in the structures Prim_stc and Xten_stc.

## KEYWORDS

---

Coordsys=	(optional), if set to either 'C', 'E' or 'G', specifies that the Healpix coordinate system is respectively Celestial=equatorial, Ecliptic or Galactic. (The relevant keyword is then added/updated in the ex-
-----------	--



	tension header, but the map is NOT rotated)
Ordering=	(optional), if set to either 'ring' or 'nested' (case un-sensitive), specifies that the map is respectively in RING or NESTED ordering scheme see also: Nested and Ring
Nside=	(optional), scalar integer, <b>HEALPix</b> resolution parameter of the data set. Must be used when the data set does not cover the whole sky
Extension=	(optional), scalar integer, extension in which to write the data (0 based). ( <b>default:</b> 0)
/Nested	(optional), if set, specifies that the map is in the NESTED ordering scheme see also: Ordering and Ring
/Ring	(optional), if set, specifies that the map is in the RING or- dering scheme see also: Ordering and Nested
/Partial	(optional), if set, the data set does not cover the whole sky. In that case the information on the actual map reso- lution should be given by the qualifier Nside (see above), or included in the FITS header enclosed in the Xten_stc.
/Nohealpix	(optional), if set, the data set can be arbitrary, and the re- striction on the number of pixels do not apply. The keywords <b>Ordering</b> , <b>Nside</b> , <b>Nested</b> , <b>Ring</b> and <b>Partial</b> are ignored.

**DESCRIPTION** `write_fits_sb` writes out the information contained in `Prim_stc` and `Exten_stc` in the primary unit and extension of the FITS file `File` respectively. Coordinate systems can also be specified by `Coordsys`. Specifying the ordering scheme is compulsory for **HEALPix** data sets and can be done either in `Header` or by setting `Ordering` or `Nested` or `Ring` to the correct value. If `Ordering` or `Nested` or `Ring` is set, its value overrides what is given in `Header`.

The data is assumed to represent a full sky data set with the number of data points  $\text{npix} = 12 * \text{Nside} * \text{Nside}$  unless `Partial` is set *or* the input FITS header contains `OBJECT = 'PARTIAL'`

AND

the `Nside` qualifier is given a valid value *or* the FITS header contains a `NSIDE`.

In the **HEALPix** scheme, invalid or missing pixels should be given the value `!healpix.bad.value = -1.63750 1030`.

If `Nohealpix` is set, the restrictions on `Nside` are void.

---

## RELATED ROUTINES

This section lists the routines related to `write_fits_sb`.

idl	version 6.4 or more is necessary to run <code>write_fits_sb</code>
<code>read_fits_map</code>	This <b>HEALPix</b> IDL facility can be used to read in maps written by <code>write_fits_sb</code> .
<code>read_fits_s</code>	This <b>HEALPix</b> IDL facility can be used to read into an IDL structure maps written by <code>write_fits_sb</code> .
<code>sxaddpar</code>	This IDL routine (included in <b>HEALPix</b> package) can be used to update or add FITS keywords to the header in <code>Prim_stc</code> and <code>Exten_stc</code>
<code>write_fits_cut4</code> , <code>write_fits_map</code> <code>write_tqu</code> , <code>write_fits_sb</code>	<b>HEALPix</b> IDL routines to write cut-sky maps, full-sky maps, polarized full-sky maps and arbitrary data sets into FITS files
<code>write_tqu</code>	This <b>HEALPix</b> IDL facility based on

write\_fits\_sb is designed to write temperature+polarization (T,Q,U) maps

---

**EXAMPLE:**

```
npix = nside2npix(128)
f= randomn(seed,npix)
n= lindgen(npix)+3
map_FN = create_struct('HDR',[' '],'FLUX',f,'NUMBER',n)
write_fits_sb, 'map_fluxnumber.fits', 0, map_FN, coord='G', /ring
```

The structure map\_FN is defined to contain a fictitious Flux+number map, where one field is a float and the other an integer. write\_fits\_sb writes out the contents of map\_FN into the extension of the FITS file 'map\_fluxnumber.fits'.

## write\_tqu

Location in HEALPix directory tree: `src/idl/fits/write_tqu.pro`

This IDL facility writes a temperature+polarization Healpix map (T,Q,U) into a binary table FITS file, with optionally the error (dT,dQ,dU) and correlation (dQU, dTU, dTQ) in separate extensions

---

**FORMAT** IDL> WRITE\_TQU , *File*, *TQU*, [*Coordsys*=, *Nested*=, *Ring*=, *Ordering*=, *Error*=, *Extension*=, *Help*=, *Hdr*=, *Xhdr*=, *Units*=, *Help*=]

---

## QUALIFIERS

File	name of a FITS file in which the maps are to be written
TQU	<p>array of Healpix maps of size (<math>N_{\text{pix}}, 3, \text{n\_ext}</math>) where <math>N_{\text{pix}}</math> is the total number of Healpix pixels on the sky, and <math>\text{n\_ext} \leq 3</math>.</p> <p>Three maps are written in each extension of the FITS file :</p> <ul style="list-style-type: none"> <li>-the temperature+polarization Stokes parameters maps (T,Q,U) in extension 0</li> <li>-the error maps (dT,dQ,dU) (if <math>\text{n\_ext} \geq 2</math>) in extension 1</li> <li>-the correlation maps (dQU, dTU, dTQ) (if <math>\text{n\_ext} = 3</math>) in extension 2</li> </ul> <p>it is also possible to write 3 maps directly in a given extension (provided the preceding extension, if any, is already filled in) by setting <b>Extension</b> to the extension number in which to write (0 based) and if <math>\text{n\_ext} + \text{Extension} \leq 3</math></p>
Coordsys=	<p>(optional),</p> <p>if set to either 'C', 'E' or 'G', specifies that the Healpix coordinate system is respectively Celestial=equatorial, Ecliptic or Galactic. (The relevant keyword is then added/updated in the extension header, but the map is NOT rotated)</p>

Error=	(optional output), will take value 1 if file can not be written
Extension=	(optional), extension unit a which to put the data (0 based). The physical interpretation of the maps is determined by the extension in which they are written see also: TQU
Hdr=	(optional), string variable containing on input the information to be added to the primary header. (If already present, FITS reserved keywords will be automatically updated).
Ordering=	(optional), if set to either 'ring' or 'nested' (case un-sensitive), specifies that the map is respectively in RING or NESTED ordering scheme see also: Nested and Ring
Units=	(optional), string describing the physical units of the data set
Xhdr=	(optional), string variable containing on input the information to be added to the extension headerx. (If already present, FITS reserved keywords will be automatically updated). It will be repeated in each extension, except for TTYPE* and EXTNAME which are generated by the routine and depend on the extension

---

## KEYWORDS

Help	if set, an extensive help is displayed and no file is written
Nested	if set, specifies that the map is in the NESTED ordering scheme see also: Ordering and Ring
Ring	if set, specifies that the map is in the RING ordering scheme see also: Ordering and Nested

---

**DESCRIPTION** `write_tqu` writes out Stokes parameters (T,Q,U) maps for the whole sky into a FITS file. It is also possible to write the error per pixel for each map and the correlation between fields, as subsequent extensions of the same FITS file (see qualifiers above). Therefore the file may have up to three extensions with three maps in each. Extensions can be written together or one by one (in their physical order) using the Extension option

---

## RELATED ROUTINES

This section lists the routines related to **write\_tqu**.

idl	version 6.4 or more is necessary to run <code>write_tqu</code>
<code>read_tqu</code>	This <b>HEALPix</b> IDL facility can be used to read in maps written by <code>write_tqu</code> .
<code>read_fits_s</code>	This <b>HEALPix</b> IDL facility can be used to read into an IDL structure maps written by <code>write_tqu</code> .
<code>sxaddpar</code>	This IDL routine (included in <b>HEALPix</b> package) can be used to update or add FITS keywords to the header(s) HDR or XHDR
<code>write_fits_cut4</code> , <code>write_fits_map</code> <code>write_tqu</code> , <code>write_fits_sb</code>	<b>HEALPix</b> IDL routines to write cut-sky maps, full-sky maps, polarized full-sky maps and arbitrary data sets into FITS files

---

## EXAMPLE:

```
npix = nside2npix(64)
TQU = randomn(seed,npix,3)
write_tqu, 'map_polarization.fits', TQU, coord='G', /ring
```

The array TQU is defined to contain a fictitious polarisation map, with the 3 Stokes parameters T, Q and U. The map is assumed to be in Galactic coordinates, with a RING ordering of the pixels and  $N_{\text{side}} = 64$ . `write_tqu` writes out the contents of TQU into the extension of the FITS file 'map\_polarization.fits'.