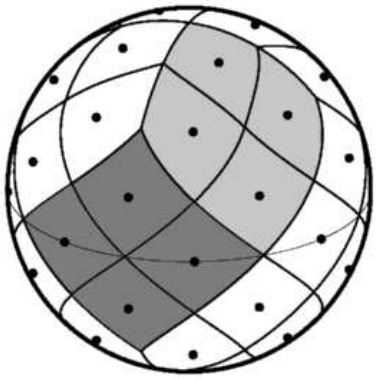
HEALPix Fortran90 Subroutines Overview



Revision: Version 3.83; November 13, 2024

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

Abstract: This document is an overview of the **HEALPix** Fortran90 subroutines.

https://healpix.sourceforge.io
http://healpix.sf.net

TABLE OF CONTENTS

Conventions

Here we list some conventions which are used in this document.

$*$	Fortran90 allows generic names which refer to several specific subroutines. Which one of the specific routines is called depends on the type and rank of the arguments supplied in the call. We tag generic names with a $*$ in this document.
$\mathbf{N}_{\mathrm{side}}$	**HEALPix** resolution parameter — see the **HEALPix** Primer.
map	We use the word "map" referring to a function, defined on the set of all **HEALPix** pixels.
θ	The polar angle or colatitude on the sphere, ranging from 0 at the North Pole to π at the South Pole.
ϕ	The azimuthal angle on the sphere, $\phi \in [0, 2\pi[$.

Changes between releases 3.80 and 3.83

- Bug corrections in `input_map` and `read_fits_partial`,
- added a workaround for a bug detected in Apple-ARM-chips implementation of gfortran 11.
- Note that cfitsio 4.1.0 or higher is required for Fortran codes running on Apple's ARM chips

Older Changes

Changes between releases 3.00 and 3.80

Version 3.80

- Improvement of `query_disc` routine in `inclusive` mode,
- the routines `alm2map_spin` and `map2alm_spin` now accept any (integer) spin values $|s| \geq 0$, but the scalar routines `alm2map` and `map2alm` are still recommended for vanishing spin ($s = 0$),
- correction of bugs preventing the compilation with versions 10.* of `gfortran`,

Version 3.70

- Addition of the subroutines `read_fits_partial` and `write_fits_partial` to read and write FITS files containing polarized or unpolarized maps defined on a fraction of the sky.

Version 3.60

- Faster Spherical Harmonics Transforms in `alm2map`, `alm2map_der`, `alm2map_spin`, `map2alm`, `map2alm_spin` thanks to the new `libsharp` library.
- The routines and function `getArgument`, `getEnvironment` and `nArguments` now calls F2003 extensions instead of external C routines.

Version 3.50

- correction of a bug in `map2alm_iterative`, when a `mask` is used in combination with `iter_order` > 0,
- addition of `zbounds` in `alm2map`, `alm2map_der`, `alm2map_spin` in order to simulate (faster) a signal on only a fraction of the sphere,
- introduction of `apply_mask` to apply an arbitrary mask and/or a latitude cut to a map,
- improved support for version 18 and more of Intel C and F90 compilers in `configure` script,
- edition to fitstools.F90 allowing a proper compilation with g95.

Version 3.40

- The facilities `anafast` and `smoothing` now support pixel-based quadrature weights. Introduction of the supporting `nside2npweights`, `unfold_weightsfile`, `get_healpix_weight_file`, `get_healpix_pixel_weight_file`.
- The subroutine `input_map` in its default mode test the value of the `POLCCONV` FITS keyword when reading a polarized map, and interpret the polarization accordingly, as described in the note on POLCCONV in The **HEALPix** Primer.
- `median` subroutine: faster by moving an internal array from heap to stack; does not crash anymore when dealing with empty data sets.

Version 3.31

- Bug correction in `input_map` routine for reading of polarized multi-HDU cut sky FITS files;
- Introduction of `winfiledir_*` and `windowfile_*` qualifiers in `alteralm` facility.

Version 3.30

- 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.
- `alm2cl` can now produces nine spectra (TT, EE, BB, TE, TB, EB, ET, BT and BE), instead of six previously, when called with two sets of polarized $a_{\ell m}$ and can also symmetrize the output $C(\ell)$ if requested
- the $a_{\ell m}$ generated by `create_alm` can now take into account non-zero (exotic) TB and EB cross-spectra (option `polar=2`) if the input FITS file contains the relevant information
- addition of `asym_cl` optional keyword in `write_minimal_header` routine
- addition of `extno` optional keyword in `write_asctab` routine to write in arbitrary HDU
- improved `repeat` behavior in `write_bintabh` routine
- edited `map2alm_iterative` routine to avoid a bug specific to Intel's Ifort 15.0.2
- CFITSIO version 3.20 (August 2009) or more now required

Version 3.20

- **HEALPix**-F90 routines and facilities can now also be compiled with the free Fortran95 compiler **g95** (www.g95.org)
- a separate `build` directory is used to store the objects, modules, ... produced during the compilation of the source codes
- bug correction in `query_disc` for some very small discs in standard mode
- improved handling of long FITS keywords, now producing FITS files fully compatible with the `PyFITS` and `Astropy` (https://www.astropy.org) `Python` libraries
- improved FITS file parsing in `generate_beam`, affecting the external $B(\ell)$ reading in the F90 facilities `alteralm`, `synfast`, `sky_ng_sim`, `smoothing`.

Version 3.11

- `libsharp` C routines used for Spherical Harmonics Transforms and introduced in **HEALPix** 3.10 can now be compiled with any `gcc` version.
- bug correction in `query_disc` routine in `inclusive` mode
- bug correction in `alm2map_spin` routine, which had its `spin` value set to 2

Version 3.10

- Support for `cfitsio` "Extended File Name Syntax", and usage of `libsharp` Spherical Harmonics Transform library. See "Fortran Facilities" for details.
- Faster Spherical Harmonics Transform routines thanks to `libsharp` C routines[1].

Changes up to release 3.00

Version 3.00

- all *input* FITS files can now be compressed (with a `.gz`, `.Z`, `.z`, or `.zip` extension) and/or remotely located (with a `ftp://` or `http://` prefix). Besides, the fits2cl routine, used to read external beam window functions from FITS files, supports (part of) the CFTISIO Extended File Name Syntax in order to read an arbitrary extension identified by its number or its name.
  Version 3.14 (March 2009) or newer of CFITSIO is required for ***HEALPix*** *3.00.*
- new code `process_mask` and new module `mask_tools` containing the routines dist2holes_nest, fill_holes_nest, maskborder_nest, size_holes_nest useful for mask apodization,
- improved accuracy of the co-latitude calculation in the vicinity of the poles at high resolution in `nest2ring, ring2nest, pix2ang_*, pix2vec_*, ...`,
- the pixel query routine `query_disc` has been improved and will return fewer false positive pixels in the inclusive mode.

Version 2.20

- Spherical Harmonics Transform routines now transparently call `libpsht` C routines, leading to a significant (2 to 4) speed-up factor. This concerns temperature and polarized transforms (alm2map, map2alm) *without precomputation* of the $P_{\ell m}$ as well as spin weighted (alm2map_spin, map2alm_spin) transforms for $0 < |s| \leq 100$, but *not* the generation of spatial derivatives (alm2map_der) which still uses the original F90 code. The compilation and linking to `libpsht`, now shipped with **HEALPix**, is done automatically, without any extra download or installation for the user[2].
- All routines for Spherical Harmonics Transforms and most routines for pixel manipulations (ang2xxx, pix2xxx, vec2xxx, ..., nside2npix, npix2nside, nside2ntemplates, ...) pixel queries (query_*, ...) and FITS I/O (input_map, output_map, read_bintab, write_bintab, ...) of sky maps now support resolution parameters $N_{\rm side} > 8192$. This means that the number of pixels and the pixel indexes can now be stored in either `integer(I4B)` or `integer(I8B)` variables (on systems supporting 64 bit variables).
  The reading and writing of $a_{\ell m}$ containing files remains limited to $\ell < 46340$, though. This restriction does not apply to $C(\ell)$ containing files.
- As a positive side effect of their upgrade, the F90 pixel/coordinate conversion routines are now up to 20% faster.
- Introduction of `long_count` and `long_size` functions.

Version 2.14

- In alm2map_der routine, a numerical bug affecting the accuracy of the Stokes parameter derivatives $\partial X/\partial\theta$, $\partial^2 X/(\partial\theta\partial\phi\sin\theta)$, $\partial^2 X/\partial\theta^2$, for $X = Q, U$ has been corrected. See "Fortran Facilities" Appendix for details.

[1] To *revert* to the original F90 implementation of these routines, the preprocessing variable `DONT_USE_SHARP` must be set during compilation.

[2] To *revert* to the original F90 implementation of all these routines, the preprocessing variable `DONT_USE_PSHT` must be set during compilation.

Versions 2.10 and 2.13

- New functions in version 2.13:
  - get_healpix_data_dir, get_healpix_main_dir, get_healpix_test_dir return full path to **HEALPix** directories.
- New routines in version 2.10:
  - alm2map_spin: synthesis of maps of arbitrary spin
  - map2alm_iterative: iterative analysis of map
  - map2alm_spin: analysis of maps of arbitrary spin
  - healpix_modules: meta-module
  - write_minimal_header: routine to write minimal FITS header
  - parse_check_unused: prints out parameters present in parameter file but not used by the code.
- Improved routines:
  - query_strip: the `inclusive` option now returns *all* (and only) the pixels overlapping, even partially, with the strip
  - query_disc: when the disc center is on one of the poles, *only* the pixels overlapping with the disc are now returned.
  - remove_dipole: can now deal with non-uniform pixel weights.
  - parse_init: silent mode
  - parse_string: can expand environment variables (${XXX}) and leading `~/`

Version 2.0

Some new features have been added

- Most routines dealing with maps and $a_{\ell m}$ (eg, create_alm, map2alm, alm2map, convert_inplace, convert_nest2ring, udgrade_nest, udgrade_ring) or inputting or outputting data (read_*, write_*) now accept both single and double precision arguments.
- The routines `map2alm` and `remove_dipole` can now deal with *non-symmetric* azimuthal cut sky. For backward compatibility, the former calling sequence is still accepted.
- most routines are now parallelized with OpenMP (for shared memory architecture), and some of them are also parallelized with MPI (for distributed memory architecture)

Some new routines have been introduced since version 1.2, as listed below.

- New routines in version 2.0
  - add_dipole, alm2cl, alm2map_der, fits2cl (replaces read_asctab), nside2ntemplates, plm_gen, rand_gauss, rand_init, rand_uni, same_shape_pixels_nest, same_shape_pixels_ring, template_pixel_nest, template_pixel_ring, write_plm (replaces write_dbintab).
- New modules or modules with new name
  - **misc_utils:** assert, assert_alloc, assert_directory_present, assert_not_present, assert_present, fatal_error, file_present, string, strupcase, strlowcase, upcase, lowcase, wall_clock_time, brag_openmp
  - **rngmod:** rand_gauss, rand_init, rand_uni
- The following routines are superseded.
  - read_asctab (replaced by fits2cl)
  - write_dbintab (replaced by write_plm)

Version 1.2

Some new routines have been introduced since version 1.1, as listed below.

- New routines in version 1.2
  - angdist, complex_fft, concatnl, del_card, get_card, getargument, getenvironment, input_tod*, nArguments, parse_double, parse_init, parse_int, parse_lgt, parse_long, parse_real, parse_string (see parse_xxx), query_disc (replaces getdisc_ring), query_polygon, query_strip, query_triangle, read_fits_cut4, real_fft, scan_directories, surface_triangle, vect_prod, write_bintabh, write_fits_cut4,
- New modules or modules with new name
  - the modules `extension` (C extensions), `healpix_fft` (FFT operations), `paramfile_io` (parameter parsing) have been introduced,
  - the module `wrap_fits` has been renamed `head_fits` to reflect its extended capabilities in manipulating FITS headers.
- The following routines are superseded. They have been moved to the `obsolete` module.
  - ask_inputmap, ask_outputmap, ask_lrange (initially in `fitstools` module)
  - setpar, getpar, anafast_parser, anafast_setpar, anafast_getpar, hotspots_parser, hotspots_setpar, hotspots_getpar, udgrade_parser, udgrade_setpar, udgrade_getpar, smoothing_parser, smoothing_setpar, smoothing_getpar (initially in `utilities` module).

add_card

Location in HEALPix directory tree: src/f90/mod/head_fits.F90

This routine writes a keyword of any kind into a FITS header. It is a wrapper to other routines that write keywords of different kinds.

FORMAT

call add_card(header, kwd, value[, *comment*, *update*])

Arguments appearing in *italic* are optional.

ARGUMENTS

name & dimensionality	kind	in/out	description
header(LEN=80) DIMENSION(:)	CHR	INOUT	The header to write the keyword to.
kwd(LEN=*)	CHR	IN	the FITS keyword to write. Should be shorter or equal to 8 characters.
value	any	IN	the value (double, real, integer, logical or character string) to give to the keyword. Note that long string values (more than 68 characters in length) are supported.
comment(LEN=)*	CHR	IN	comment to the keyword.

name & dimensionality	kind	in/out	description
update	LGT	IN	if set to `.true.`, the first occurence of the keyword `kwd` in `header` will be updated (and all other occurences removed); otherwise, the keyword will be appended at the end (and any previous occurence removed). If the keyword is either 'HISTORY' or 'COMMENT', `update` is ignored and the keyword is peacefully appended at the end of the header.

EXAMPLE:

```
character(len=80), dimension(1:120) ::  header
header = " !  very important
call add_card(header,'NSIDE',256,'the nside of the map')
```

Gives the keyword 'NSIDE' the value 256 in the given header-string. It is important to make sure that the `header` string array is empty before attempting to write anything in it.

MODULES & ROUTINES

This section lists the modules and routines used by **add_card**.

write_hl	more general routine for adding a keyword to a header.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

write_minimal_header	routine to write **HEALPix** compliant baseline FITS header

get_card	general purpose routine to read any keywords from a header in a FITS file.
del_card	routine to discard a keyword from a FITS header
read_par, number_of_alms	routines to read specific keywords from a header in a FITS file.
getsize_fits	function returning the size of the data set in a fits file and reading some other useful FITS keywords
merge_headers	routine to merge two FITS headers

add_dipole*

Location in HEALPix directory tree: **src/f90/mod/pix_tools.F90**

This routine provides a means to add a monopole and dipole to a **HEALPix** map.

FORMAT

call add_dipole*(nside, map, ordering, degree, multipoles[, *fmissval*])

Arguments appearing in *italic* are optional.

ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	value of $N_{\rm side}$ resolution parameter for input map
map(0:12*nside*nside-1)	SP/ DP	INOUT	**HEALPix** map to which the monopole and dipole will be added. Those are added to *all unflagged pixels*.
ordering	I4B	IN	**HEALPix** scheme 1:RING, 2: NESTED
degree	I4B	IN	multipoles to add. It is either 0 (nothing done), 1 (monopole only) or 2 (monopole and dipole)
multipoles(0:degree*degree-1)	DP	IN	values of monopole and dipole to add. The monopole is described as a scalar in the same units as the input map, the dipole as a 3D cartesian vector, in the same units.
fmissval	SP/ DP	IN	value used to flag bad pixel on input (**default:** -1.6375e30). Pixels with that value are left unchanged.

EXAMPLE:

```
call add_dipole*(128, map, 1, 2, (/ 10.0_dp, 0.0_dp, 1.2_dp, 0.0_dp /) )
```

`map` is a **HEALPix** map of resolution $N_{\rm side} = 128$, with the RING ordering scheme. A monopole of amplitude 10 and a dipole of amplitude 1.2 and directed along the y axis will be added to it.

MODULES & ROUTINES

This section lists the modules and routines used by **add_dipole***.

pix_tools module, containing:

RELATED ROUTINES

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

remove_dipole routine to remove the best fit monopole and monopole from a map.

alm2cl*

Location in HEALPix directory tree: **src/f90/mod/alm_tools.F90**

This routine computes the auto (or cross) power spectra of a one (or two) sets of spherical harmonics coefficients $a_{\ell m}$,

$$C_{12}^{XY}(\ell) = \frac{1}{2\ell+1} \sum_{m=-\ell}^{\ell} a_{1,\ell m}^{X} a_{2,\ell m}^{Y*}, \tag{1}$$

with X and Y belonging to T, E, B.
If requested, for $X \neq Y$, symmetrized power spectra

$$C_{\{12\}}^{\{XY\}}(\ell) \equiv \frac{C_{12}^{XY}(\ell) + C_{12}^{YX}(\ell)}{2} = \frac{C_{12}^{XY}(\ell) + C_{21}^{XY}(\ell)}{2} \tag{2}$$

are output.

FORMAT

call alm2cl*(nlmax, nmmax, alm1, *[alm2,]* cl, *[symmetric]*)

Arguments appearing in *italic* are optional.

ARGUMENTS

name & dimensionality	kind	in/out	description
nlmax	I4B	IN	the maximum ℓ value used for the $a_{\ell m}$.
nmmax	I4B	IN	the maximum m value used for the $a_{\ell m}$.
alm1(1:p, 0:nlmax, 0:nmmax)	SPC/ DPC	IN	First set of $a_{\ell m}$ values. p is 3 or 1 depending on wether polarisation is included or not. In the former case, the first index runs from 1 to 3 corresponding to (T,E,B).
alm2(1:p, 0:nlmax, 0:nmmax)	SPC/ DPC	IN	Second set of $a_{\ell m}$ values.
cl(0:nlmax,1:d)	SP/ DP	OUT	resulting auto or cross power spectra. If both `alm1` and `alm2` are present, `cl` will be their cross power spectrum. If only `alm1` is present,`cl` will be its power spectrum. If $d = 1$, only the temperature spectrum C_ℓ^{TT} will be output. If $d = 4$ and $p = 3$, the output will be C_ℓ^{TT}, C_ℓ^{EE}, C_ℓ^{BB} and C_ℓ^{TE}. If $d \geq 6$ and $p = 3$, C_ℓ^{TB} and C_ℓ^{EB} will also be output, and if $d \geq 9$, $p = 3$, and `symmetric` is not set, C_ℓ^{ET}, C_ℓ^{BT} and C_ℓ^{BE} will be included.
symmetric	LGT	IN	If set to `.true.` when $d \geq 4$, $p = 3$ and `alm2` is present then a symmetrized version of the cross spectra will be output in `cl`, namely C_ℓ^{TT}, C_ℓ^{EE}, C_ℓ^{BB}, $(C_\ell^{TE} + C_\ell^{ET})/2$, $(C_\ell^{TB} + C_\ell^{BT})/2$ and $(C_\ell^{EB} + C_\ell^{BE})/2$. (**default:** `.false.` (unsymmetrized output))

EXAMPLE:

```
lmax = 128 ; mmax = lmax
call alm2cl(lmax, mmax, alm1, cl_auto)
call alm2cl(lmax, mmax, alm1, alm2, cl_cross)
call alm2cl(lmax, mmax, alm1, alm2, cl_sym, symmetric=.true.)
```

`cl_auto` will contain the (auto) power spectrum of the $a_{\ell m}$ coefficients `alm1` up to $\ell = 128$, `cl_cross` will be the cross power spectra of the two sets of $a_{\ell m}$ coefficients `alm1` and `alm2`, while `cl_sym` will be a symmetrized version of `cl_cross`.

MODULES & ROUTINES

This section lists the modules and routines used by **alm2cl***.

none

RELATED ROUTINES

This section lists the routines related to **alm2cl***.

map2alm	routine extracting the $a_{\ell m}$ coefficients from a **HEALPix** map
create_alm	routine to generate randomly distributed $a_{\ell m}$ coefficients according to a given power spectrum

alm2map*

Location in HEALPix directory tree: src/f90/mod/alm_tools.F90

This routine is a wrapper to 10 other routines: alm2map_sc_X, alm2map_sc_pre_X, alm2map_pol_X, alm2map_pol_pre1_X, alm2map_pol_pre2_X, where X stands for either s or d. These routines synthesize a **HEALPix** *RING ordered* temperature map (and if specified, polarisation maps) from input $a^T_{\ell m}$ (and if specified $a^E_{\ell m}$ and $a^B_{\ell m}$) values. The different routines are called dependent on what parameters are passed. Some routines synthesize maps with or without precomputed harmonics (note that since **HEALPix** v2.20 precomputed harmonics most likely won't speed up computation) and some with or without polarisation. The routines accept both single and double precision arrays for alm_TGC and map_TQU. The precision of these arrays should match.

FORMAT

call alm2map*(nsmax, nlmax, nmmax, alm_TGC, map_TQU[, plm=| zbounds=])

ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	the $N_{\rm side}$ value of the map to synthesize.
nlmax	I4B	IN	the maximum ℓ value used for the $a_{\ell m}$.
nmmax	I4B	IN	the maximum m value used for the $a_{\ell m}$.
alm_TGC(1:p, 0:nlmax, 0:nmmax)	SPC or DPC	IN	The $a_{\ell m}$ values to make the map from. p is 3 or 1 depending on wether polarisation is respectively included or not. In the former case, the first index runs from 1 to 3 corresponding to (T,E,B).

map_TQU(0:12*nsmax**2-1)	SP or DP	OUT	if only a temperature map is to be synthesized, the map-array should be passed with this rank.
map_TQU(0:12*nsmax**2-1, 1:3)	SP or DP	OUT	if both temperature an polarisation maps are to be synthesized, the map array should have this rank, where the second index is (1,2,3) corresponding to (T,Q,U).
plm(0:n_plm-1), OPTIONAL	DP	IN	If this optional matrix is passed with this rank, precomputed $P_{\ell m}(\theta)$ are used instead of recursion. (n_plm = nsmax*(nmmax+1)*(2*nlmax-nmmax+2).
plm(0:n_plm-1,1:3), OPTIONAL	DP	IN	If this optional matrix is passed with this rank, precomputed $P_{\ell m}(\theta)$ AND precomputed tensor harmonics are used instead of recursion. (n_plm = nsmax*(nmmax+1)*(2*nlmax-nmmax+2).
zbounds(1:2), OPTIONAL	DP	IN	section of the sphere on which to perform the map synthesis, expressed in terms of z = sin(latitude) = cos(θ). If zbounds(1)<zbounds(2), it is performed *on* the strip zbounds(1)< z <zbounds(2); if not, it is performed *outside* the strip zbounds(2)≤ z ≤zbounds(1). If absent, the whole map is processed. Currently, zbounds and plm can not be used together.

EXAMPLE:

```
use healpix_types
use pix_tools, only :  nside2npix
use alm_tools, only :  alm2map
integer(I4B) ::  nside, lmax, mmax, npix
```

```
real(SP), dimension(:,:), allocatable ::  map
complex(SPC), dimension(:,:,:), allocatable ::  alm
real(DP), dimension(1:2) ::  zrange
...
nside=256 ; lmax=512 ; mmax=lmax
npix=nside2npix(nside)
allocate(alm(1:3,0:lmax,0:mmax))
allocate(map(0:npix-1,1:3))
...
zrange =(/ 0.0_dp, 0.5_dp /)
call alm2map(nside, lmax, mmax, alm, map, zbounds=zrange)
```

Make temperature and polarisation maps from the scalar and tensor $a_{\ell m}$ passed in alm. The maps have $N_{\rm side}$ of 256, and are constructed from $a_{\ell m}$ values up to 512 in ℓ and m. In order to save time, the maps are only generated on the range $0 < z < 0.5$ (leaving the other pixels to 0) even though the input $a_{\ell m}$ are those of a full sky map.

MODULES & ROUTINES

This section lists the modules and routines used by **alm2map***.

ring_synthesis	Performs FFT over m for synthesis of the rings.
compute_lam_mm, get_pixel_layout, gen_lamfac,gen_mfac, gen_normpol, gen_recfac, init_rescale, l_min_ylm	Ancillary routines used for $Y_{\ell m}$ recursion
misc_utils	module, containing:
assert_alloc	routine to print error message, when an array can not be allocated properly

Note: Starting with version 3.10, `libsharp` routines will be called when precomputed $P_{\ell m}$ are not provided.

RELATED ROUTINES

This section lists the routines related to **alm2map***.

alm2map_der	routine generating a map and its derivatives from its $a_{\ell m}$

alm2map_spin	routine generating maps of arbitrary spin from their ${}_sa_{\ell m}$
smoothing	executable using alm2map* to smooth maps
synfast	executable using alm2map* to synthesize maps.
map2alm	routine performing the inverse transform of alm2map*.
create_alm	routine to generate randomly distributed $a_{\ell m}$ coefficients according to a given power spectrum
pixel_window, generate_beam	return the ℓ-space **HEALPix** -pixel and beam window function respectively
alter_alm	modifies $a_{\ell m}$ to emulate effect of real space filtering

alm2map_der*

Location in HEALPix directory tree: src/f90/mod/alm_tools.F90

This routine is a wrapper to four other routines that synthesize a **HEALPix** temperature (and polarisation) map(s), its (their) first derivatives, and optionally its (their) second derivatives. The routines accept both single and double precision arrays for alm, map, der1 and der2. The precision of these arrays should match. All maps produced are RING ordered.

See "Fortran Facilities" Appendix for a note on a bug affecting the calculation of polarisation derivatives on past versions of this routine.

FORMAT

call alm2map_der*(nsmax, nlmax, nmmax, alm, map, der1[, der2=, zbounds=])

ARGUMENTS

name & dimensionality	**kind**	**in/out**	**description**
nsmax	I4B	IN	the $N_{\rm side}$ value of the map to synthesize.
nlmax	I4B	IN	the maximum ℓ value used for the $a_{\ell m}$.
nmmax	I4B	IN	the maximum m value used for the $a_{\ell m}$.
alm(1:p, 0:nlmax, 0:nmmax)	SPC/ DPC	IN	The $a_{\ell m}$ values to make the map from. p is either 1 (temperature only) or 3 (temperature+polarisation).
map(0:12*nsmax**2-1) *or* (0:12*nsmax**2-1,1:3)	SP/ DP	OUT	temperature map $T(p)$ or temperature + polarisation maps $T(p)$, $Q(p)$, $U(p)$ to be synthesized.
der1(0:12*nsmax**2-1, 1:2*p)	SP/ DP	OUT	contains on output the first derivatives of T: $(\partial T/\partial\theta, \partial T/\partial\phi/\sin\theta)$ or the interleaved derivatives of T, Q, and U: $(\partial T/\partial\theta, \partial Q/\partial\theta, \partial U/\partial\theta; \partial T/\partial\phi/\sin\theta, \ldots)$
der2(0:12*nsmax**2-1,1:3*p), OPTIONAL	SP/ DP	OUT	If this optional matrix is passed with this rank, it will contain on output the second derivatives $(\partial^2 T/\partial\theta^2, \partial^2 T/\partial\theta\partial\phi/\sin\theta, \partial^2 T/\partial\phi^2/\sin^2\theta)$ or $(\partial^2 T/\partial\theta^2, \partial^2 Q/\partial\theta^2, \partial^2 Q/\partial\theta^2, \ldots)$
zbounds(1:2), OPTIONAL	DP	IN	section of the sphere on which to perform the map synthesis, expressed in terms of z = sin(latitude) = cos(θ). If zbounds(1)<zbounds(2), it is performed *on* the strip zbounds(1)< z <zbounds(2); if not, it is performed *outside* the strip zbounds(2)≤ z ≤zbounds(1). If absent, the whole map is processed.

EXAMPLE:

```
use healpix_types
use pix_tools, only :  nside2npix
```

```
use alm_tools, only :  alm2map_der
integer(I4B) ::  nside, lmax, mmax, npix
real(SP), dimension(:), allocatable ::  map
real(SP), dimension(:,:), allocatable ::  der1, der2
complex(SPC), dimension(:,:,:), allocatable ::  alm
...
nside=256 ; lmax=512 ; mmax=lmax
npix=nside2npix(nside)
allocate(alm(1:1,0:lmax,0:mmax))
allocate(map(0:npix-1))
allocate(der1(0:npix-1,1:2), der2(0:npix-1,1:3))
...
call alm2map_der(nside, lmax, mmax, alm, map, der1, der2=der2)
```

Make temperature maps and its derivatives from the $a_{\ell m}$ passed in alm. The maps have $N_{\rm side}$ of 256, and are constructed from $a_{\ell m}$ values up to 512 in ℓ and m.

MODULES & ROUTINES

This section lists the modules and routines used by **alm2map_der***.

ring_synthesis	Performs FFT over m for synthesis of the rings.
compute_lam_mm, get_pixel_layout, gen_lamfac_der, gen_mfac, gen_recfac, init_rescale, l_min_ylm	Ancillary routines used for ${}_sY_{\ell m}$ recursion
misc_utils	module, containing:
assert_alloc	routine to print error message, when an array can not be allocated properly

RELATED ROUTINES

This section lists the routines related to **alm2map_der***.

alm2map	routine generating maps of temperature and polarisation from their $a_{\ell m}$
alm2map_spin	routine generating maps of arbitrary spin from their ${}_sa_{\ell m}$

synfast	executable using alm2map_der* to synthesize maps.
create_alm	routine to generate randomly distributed $a_{\ell m}$ coefficients according to a given power spectrum

alm2map_spin*

Location in HEALPix directory tree: src/f90/mod/alm_tools.F90

This routine produces the maps of arbitrary spin s and $-s$ given their alm coefficients. A (complex) map S of spin s is a linear combination of the spin weighted harmonics ${}_sY_{\ell m}$

$$ {}_sS(p) = \sum_{\ell m} {}_sa_{\ell m} \;\; {}_sY_{\ell m}(p) \tag{3}$$

for $\ell \geq |m|, \ell \geq |s|$, and is such that ${}_sS^* = {}_{-s}S$.
The usual phase convention for the spin weighted harmonics is ${}_sY^*_{\ell m} = (-1)^{s+m}{}_{-s}Y_{\ell -m}$ and therefore ${}_sa^*_{\ell m} = (-1)^{s+m}{}_{-s}a_{\ell -m}$.

alm2map_spin* expects the alm coefficients to be provided as

$$ {}_{|s|}a^+_{\ell m} = -({}_{|s|}a_{\ell m} + (-1)^s{}_{-|s|}a_{\ell m})/2, \tag{4}$$
$$ {}_{|s|}a^-_{\ell m} = -({}_{|s|}a_{\ell m} - (-1)^s{}_{-|s|}a_{\ell m})/(2i), \tag{5}$$

for $m \geq 0$, knowing that, just as for spin 0 maps, the coefficients for $m < 0$ are given by

$$ {}_{|s|}a^+_{\ell -m} = (-1)^m{}_{|s|}a^{+*}_{\ell m}, \tag{6}$$
$$ {}_{|s|}a^-_{\ell -m} = (-1)^m{}_{|s|}a^{-*}_{\ell m}. \tag{7}$$

The two (real) maps produced by alm2map_spin* are defined respectively as

$$ {}_{|s|}S^+ = ({}_{|s|}S + {}_{-|s|}S)/2, \tag{8}$$
$$ {}_{|s|}S^- = ({}_{|s|}S - {}_{-|s|}S)/(2i). \tag{9}$$

With these definitions, ${}_2a^+$, ${}_2a^-$, ${}_2S^+$ and ${}_2S^-$ match **HEALPix** polarization a^E, a^B, Q and U respectively. **However**, for $s = 0$, ${}_0a^+_{\ell m} = -a^T_{\ell m}$, ${}_0a^-_{\ell m} = 0$, ${}_0S^+ = T$, ${}_0S^- = 0$.

When dealing only with scalar quantities, like temperature or intensity maps, having a spin $s = 0$, it is highly recommended, and much more memory-efficient, to use directly the routine alm2map, rather then setting `spin`= 0 in alm2map_spin*.

FORMAT call alm2map_spin*(nsmax, nlmax, nmmax, spin, alm, map[, zbounds=])

ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	the $N_{\rm side}$ value of the map to synthesize.
nlmax	I4B	IN	the maximum ℓ value used for the $a_{\ell m}$.
nmmax	I4B	IN	the maximum m value used for the $a_{\ell m}$.
spin	I4B	IN	spin s of the maps to be generated (only its absolute value is relevant).
alm(1:2, 0:nlmax, 0:nmmax)	SPC/ DPC	IN	The ${}_{\|s\|}a^{+}_{\ell m}$ and ${}_{\|s\|}a^{-}_{\ell m}$ values to make the map from.
map(0:12*nsmax**2-1, 1:2)	SP/ DP	OUT	${}_{\|s\|}S^{+}$ and ${}_{\|s\|}S^{-}$ output maps
zbounds(1:2), OPTIONAL	DP	IN	section of the sphere on which to perform the map synthesis, expressed in terms of z = sin(latitude) = cos(θ). If zbounds(1)<zbounds(2), it is performed *on* the strip zbounds(1)< z <zbounds(2); if not, it is performed *outside* the strip zbounds(2)≤ z ≤zbounds(1). If absent, the whole map is processed.

EXAMPLE:

```
use healpix_types
use pix_tools, only :  nside2npix
use alm_tools, only :  alm2map_spin
integer(I4B) ::  nside, lmax, mmax, npix, spin
real(SP), dimension(:,:), allocatable ::  map
complex(SPC), dimension(:,:,:), allocatable ::  alm
...
nside=256 ; lmax=512 ; mmax=lmax ; spin=4
```

```
npix=nside2npix(nside)
allocate(alm(1:2,0:lmax,0:mmax))
allocate(map(0:npix-1,1:2))
...
call alm2map_spin(nside, lmax, mmax, spin, alm, map)
```

Make spin-4 maps from the $a_{\ell m}$ passed in alm. The maps have $N_{\rm side}$ of 256, and are constructed from $a_{\ell m}$ values up to 512 in ℓ and m.

MODULES & ROUTINES

This section lists the modules and routines used by **alm2map_spin***.

ring_synthesis	Performs FFT over m for synthesis of the rings.
compute_lam_mm, get_pixel_layout, gen_lamfac_der, gen_mfac, gen_mfac_spin, do_lam_lm_spin, gen_recfac, gen_recfac_spin, init_rescale, l_min_ylm	Ancillary routines used for $Y_{\ell m}$ recursion
misc_utils	module, containing:
assert_alloc	routine to print error message, when an array can not be allocated properly

Note: Starting with version 3.80, some `libsharp` routines will be called for any $|s|$ value.

RELATED ROUTINES

This section lists the routines related to **alm2map_spin***.

alm2map	routine generating maps of temperature and polarisation from their $a_{\ell m}$
alm2map_der	routine generating maps of temperature and polarisation, and their spatial derivatives, from their $a_{\ell m}$
map2alm_spin	routine performing the inverse transform of alm2map.
create_alm	routine to generate randomly distributed $a_{\ell m}$ coefficients according to a given power spectrum

alms2fits*

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This routine stores $a_{\ell m}$ values in a binary FITS file. Each FITS file extension created will contain one integer column with $index = \ell^2 + \ell + m + 1$, and 2 or 4 single (or double) precision columns with real/imaginary $a_{\ell m}$ values and real/imaginary standard deviation. One can store temperature $a_{\ell m}$ or temperature and polarisation, $a^T_{\ell m}$, $a^E_{\ell m}$ and $a^B_{\ell m}$. If temperature is specified, a FITS file with one extension is created. If polarisation is specified, a FITS file with 3 extensions one for each set of $a_{\ell m}$, $a^T_{\ell m}$, $a^E_{\ell m}$ and $a^B_{\ell m}$ is created.

FORMAT

call alms2fits*(filename, nalms, alms, ncl, header, nlheader, next)

ARGUMENTS

name & dimensionality	kind	in/out	description
filename(LEN=`filenamelen`)	CHR	IN	filename for the FITS file to store the $a_{\ell m}$ in.
nalms	I4B	IN	number of $a_{\ell m}$ to store.
ncl	I4B	IN	number of columns in the FITS file. If an standard deviation is given, this number is 5, otherwise it is 3.
next	I4B	IN	the number of extensions. 1 for temperature only, 3 for temperature and polarisation.

name & dimensionality	kind	in/out	description
alms(1:nalms,1:ncl+1,1:next)	SP/ DP	IN	the $a_{\ell m}$ to write to the file. alms(i,1,j) and alms(i,2,j) contain the ℓ and m values for the ith $a_{\ell m}$ (j=1,2,3 for (T,E,B)). alms(i,3,j) and alms(i,4,j) contain the real and imaginary value of the ith $a_{\ell m}$. Finally, the standard deviation for the ith $a_{\ell m}$ is contained in alms(i,5,j) (real) and alms(i,6,j) (imaginary).
nlheader	I4B	IN	number of header lines to write to the file.
header(LEN=80) (1:nlheader, 1:next)	CHR	IN	the header to the FITS file.

EXAMPLE:

```
call alms2fits ('alms.fits', 65*66/2, alms, 3, header, 80, 3)
```

Creates a FITS file with the $a^T_{\ell m}$, $a^E_{\ell m}$ and $a^B_{\ell m}$ values given in alms(1:65*66/2,1:4,1:3). The last index specifies (T,E,B). The second index gives l, m, real($a_{\ell m}$), imaginary($a_{\ell m}$) for each of the $a_{\ell m}$. The number 65*66/2 is the number of $a_{\ell m}$ values up to an ℓ value of 64. 80 lines from header(1:80,1:3) is written to each extension.

MODULES & ROUTINES

This section lists the modules and routines used by **alms2fits***.

write_alms	routine called by alms2fits* for each extension.
fitstools	module, containing:
printerror	routine for printing FITS error messages.
cfitsio	library for FITS file handling.

RELATED ROUTINES

This section lists the routines related to **alms2fits***.

fits2alms, read_conbintab	routines to read $a_{\ell m}$ from a FITS file
dump_alms	has the same function as alms2fits* but with parameters passed differently.

alter_alm*

Location in HEALPix directory tree: src/f90/mod/alm_tools.F90

This routine modifies scalar (and tensor) $a_{\ell m}$ by multiplying them by a beam window function described by a FWHM (in the case of a gaussian beam) or read from an external file (in the more general case of a circular beam) $a_{\ell m} \longrightarrow a_{\ell m} b(\ell)$. It can also be used to multiply the $a_{\ell m}$ by an arbitray function of ℓ.

FORMAT

call alter_alm*(nsmax, nlmax, nmmax, fwhm_arcmin, alm_TGC[, beam_file, window])

ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	$N_{\rm side}$ resolution parameter of the map associated with the $a_{\ell m}$ considered. Currently has no effect on the routine.
nlmax	I4B	IN	maximum ℓ value for the $a_{\ell m}$.
nmmax	I4B	IN	maximum m value for the $a_{\ell m}$.
fwhm_arcmin	SP/ DP	IN	fwhm size of the gaussian beam in arcminutes.
alm_TGC(1:p,0:nlmax,0:nmmax)	SPC/ DPC	INOUT	complex $a_{\ell m}$ values to be altered. The first index here runs from 1:1 for temperature only, and 1:3 for polarisation. In the latter case, 1=T, 2=E, 3=B.

beam_file(LEN=`filenamelen`) (OPTIONAL)	CHR	IN	name of the file containing the (non necessarily gaussian) window function B_ℓ of a circular beam. If present, it will override the argument `fwhm_arcmin`.
window(0:nlw,1:d) (OPTIONAL)	SP/ DP	IN	arbitrary window by which to multiply the $a_{\ell m}$. If present, it overrides both `fwhm_arcmin` and `beam_file`. If nlw $<$ nlmax, the $a_{\ell m}$ with $\ell \in$ {nlw+1,nlmax} are set to 0, and a warning is issued. If $d < p$ the window for temperature is replicated for polarisation.

EXAMPLE:

```
call alter_alm(64, 128, 128, 1, 5.0, alm_TGC)
```

Alters scalar and tensor $a_{\ell m}$ of a map with $N_{\rm side} = 64$, $\ell_{\rm max} = m_{\rm max} = 128$ by multiplying them by the beam window function of a gaussian beam with FWHM = 5 arcmin.

MODULES & ROUTINES

This section lists the modules and routines used by **alter_alm***.

alm_tools	module, containing:
generate_beam	routine to generate beam window function
pixel_window	routine to generate pixel window function

RELATED ROUTINES

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

create_alm	Routine to create $a_{\ell m}$ coefficients.
rotate_alm	Routine to rotate $a_{\ell m}$ coefficients between 2 different arbitrary coordinate systems.
map2alm	Routines to analyze a **HEALPix** sky map into its $a_{\ell m}$ coefficients.

alm2map	Routines to synthetize a **HEALPix** sky map from its $a_{\ell m}$ coefficients.
alms2fits, dump_alms	Routines to save a set of $a_{\ell m}$ in a FITS file.

ang2vec

Location in HEALPix directory tree: **src/f90/mod/pix_tools.F90**

Routine to convert the position angles (θ, ϕ) of a point on the sphere into its 3D position vector (x, y, z) with $x = \sin\theta\cos\phi$, $y = \sin\theta\sin\phi$, $z = \cos\theta$.

FORMAT

call ang2vec(theta, phi, vector)

ARGUMENTS

name & dimensionality	kind	in/out	description
theta	DP	IN	colatitude in radians measured southward from north pole (in $[0,\ \pi]$).
phi	DP	IN	longitude in radians measured eastward (in $[0,\ 2\pi]$).
vector(3)	DP	OUT	three dimensional cartesian position vector (x, y, z) normalised to unity. The north pole is $(0, 0, 1)$

RELATED ROUTINES

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

angdist	computes the angular distance between 2 vectors
vec2ang	converts the 3D position vector of point into its position angles on the sphere.
vect_prod	computes the vector product between two 3D vectors

angdist

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

Returns the angular distance in radians between two vectors. The input vectors do not have to be normalised. For almost colinear or anti-colinear vectors, renders numerically more accurate results than the $\cos^{-1}$ of the scalar product.

FORMAT

call angdist(v1, v2, dist)

ARGUMENTS

name & dimensionality	kind	in/out	description
v1(3)	DP	IN	cartesian vector.
v2(3)	DP	IN	cartesian vector.
dist	DP	OUT	angular distance in radians between the 2 vectors.

EXAMPLE:

```
use healpix_types
use pix_tools, only :  angdist
real(DP) ::  dist, one = 1.0_dp
call angdist((/1,2,3/)*one, (/1,2,4/)*one, dist)
print*, dist
```

Returns the angular distance between 2 vectors.

RELATED ROUTINES

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

ang2vec converts the position angles of a point on the sphere into its 3D position vector.

vec2ang	converts the 3D position vector of point into its position angles on the sphere.
vect_prod	computes the vector product between two 3D vectors

apply_mask

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

This routine multiplies a **HEALPix** map by an arbitrary pixel mask and/or sets to 0 pixels lying in or out a constant latitude strip

FORMAT

call apply_mask(map, ordering[, mask=, zbounds=])

ARGUMENTS

name & dimensionality	kind	in/out	description
map(0:npix-1,1:p)	SP/ DP	INOUT	**HEALPix** map(s) on which the mask(s) and/or cut will be applied. The map will obviously be modified on output if either mask or zbounds are provided.
ordering	I4B	IN	**HEALPix** scheme of the map(s) (and mask(s)) 1:RING, 2: NESTED (relevant for zbounds).
mask(0:npix-1,1:q) OPTIONAL	SP/ DP	IN	pixel mask(s). Should have the same ordering and number of pixels (npix) as map. If several maps are provided ($p > 1$), each map is multiplied by its respective mask, and if there are fewer masks than maps ($q < p$) the last mask is duplicated as many times as necessary to match the number of maps.
zbounds(1:2) OPTIONAL	DP	IN	section of the map to be set to 0, expressed in terms of $z = \sin(\text{latitude}) = \cos(\theta)$. If zbounds(1)<zbounds(2), pixels *outsise* the strip zbounds(1)< z <zbounds(2) are set to 0; if not, pixels *on* the strip zbounds(2)$\leq z \leq$zbounds(1) are set to 0. If absent, the map is unchanged.

EXAMPLE:

```
s = sin(15.0_dp * DEG2RAD)
call apply_mask(map, 1, zbounds=(/ s, -s /) )
```

Will set to 0 pixels of the input (RING ordered) map lying in the strip $|b| \leq 15^o$.

MODULES & ROUTINES

This section lists the modules and routines used by **apply_mask**.

pix_tools module, containing:

assert,assert_alloc, assert_directory_present, . . .

Location in HEALPix directory tree: src/f90/mod/misc_utils.F90

The Fortran90 module misc_utils contains a few routines to test an assertion and return an error message if it is false.

SUBROUTINES:

`call assert(test [, msg, errcode])`

if `test` is true, proceeds with normal code execution. If `test` is false, issues a standard error message (unless `msg` is provided) and stops the code execution with the status `errcode` (or 1 by default).

`call assert_alloc(status, code, array)`

if `status` is 0, proceeds with normal code execution. If not, issues an error message indicating a problem during memory allocation of `array` in program `code`, and stops the code execution.

`call assert_directory_present(directory)`

issues an error message and stops the code execution if the directory named `directory` can not be found

`call assert_not_present(filename)`

issues an error message and stops the code execution if a file with name `filename` already exists.

`call assert_present(filename)`

issues an error message and stops the code execution if the file named `filename` can not be found.

`call fatal_error([msg])`

`call fatal_error`

issue an (optional user defined) error message and stop the code execution.

ARGUMENTS

name & dimensionality		kind	in/out	description
test		LGT	IN	result of a logical test
msg	OPTIONAL	CHR	IN	character string describing nature of error
errorcode	OPTIONAL	I4B	IN	error status given to code interruption
status		I4B	IN	value of the `stat` flag returned by the F90 `allocate` command
code		CHR	IN	name of program or code in which allocation is made
array		CHR	IN	name of array allocated
directory		CHR	IN	directory name (contains a '/')
filename		CHR	IN	file name

EXAMPLE:

```
program my_code
use misc_utils
real, allocatable, dimension(:)  ::  vector
integer ::  status
real ::  a = -1.

allocate(vector(12345),stat=status)
call assert_alloc(status, 'my_code', 'vector')

call assert_directory_present('/home')

call assert(a > 0., 'a is NEGATIVE !!!')

end program my_code
```

Will issue a error message and stops the code if `vector` can not be allocated, will stop the code if '/home' is not found, and will stop the code and complain loudly about it because `a` is actually negative.

brag_openmp

Location in HEALPix directory tree: src/f90/mod/misc_utils.F90

If compiled with shared memory libraries (OpenMP), this routine prints out the number of CPUs used (controlled by the environment variable `OMP_NUM_THREADS`) and the number of CPUs available.

FORMAT

call brag_openmp()

EXAMPLE:

```
use misc_utils
call brag_openmp()
```

Will print out:

```
-------------------
Number of OpenMP threads in use:  2
Number of CPUs available:  2
-------------------
```

on a bi-pro (or dual core) computer

complex_fft

Location in HEALPix directory tree: **src/f90/mod/healpix_fft.F90**

This routine performs a forward or backward Fast Fourier Transformation on its argument `data`.

FORMAT

call complex_fft(data, backward)

ARGUMENTS

name&dimensionality	kind	in/out	description
data(:)	XXX	INOUT	array containing the input and output data. It can be of type real(sp), real(dp), complex(spc) or complex(dpc). If it is of type real, it is interpreted as an array of size(data)/2 complex variables.
backward	LGT	IN	Optional argument. If present and true, perform backward transformation, else forward

EXAMPLE:

```
use healpix_fft
call complex_fft (data, backward=.true.)
```

Performs a backward FFT on data.

RELATED ROUTINES

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

real_fft routine for FFT of real data

compute_statistics*

Location in HEALPix directory tree: src/f90/mod/statistics.f90

This routine computes the min, max, absolute deviation and first four order moment of a data set

FORMAT

call compute_statistics*(data,stats[, *badval*])

Arguments appearing in *italic* are optional.

ARGUMENTS

name & dimensionality	kind	in/out	description
data(:)	SP/ DP	IN	data set $\{x_i\}$
stats	tstats	OUT	structure containing the statistics of the data. The respective fields (stats%*field*) are:
ntot	I8B	–	total number of data points
nvalid	I8B	–	number n of valid data points
mind, maxd	DP	–	minimum and maximum valid data
average	DP	–	average of valid points $m = \sum_i x_i/n$
absdev	DP	–	absolute deviation $a = \sum_i \lvert x_i - m\rvert/n$
var	DP	–	variance $\sigma^2 = \sum(x_i - m)^2/(n-1)$
rms	DP	–	standard deviation σ
skew	DP	–	skewness factor $s = \sum(x_i - m)^3/(n\sigma^3)$
kurt	DP	–	kurtosis factor $k = \sum(x_i - m)^4/(n\sigma^4) - 3$
badval (OPTIONAL)	SP/ DP	IN	sentinel value given to bad data points. Data points with this value will be ignored during calculation of the statistics. If not set, all points will be considered. **Do not set to 0!**.

EXAMPLE:

```
use statistics, only:  compute_statistics, print_statistics, tstats
type(tstats) ::  stats
```

```
...
compute_statistics(map, stats)
print*,stats%average, stats%rms
print_statistics(stats)
```

Computes the statistics of `map`, prints its average and *rms* and prints the whole list of statistical measures.

RELATED ROUTINES

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

median	routine to compute median of a data set

concatnl

Location in HEALPix directory tree: src/f90/mod/paramfile_io.F90

Function to concatenate up to 10 subtrings interspaced with LineFeed character. Upon printing each subtring will be on a different line.

FORMAT

var=concatnl(string1[, string2, string3, ...])

ARGUMENTS

name & dimensionality	kind	in/out	description
string1	CHR	IN	the first substring to be concatenated.
string2	CHR	IN optional	the second substring (if any) to be concatenated.
string3	CHR	IN optional	... up to 10 substrings can be concatenated.
var	CHR	OUT	concatenation of the substrings interspaced with LineFeed character.

EXAMPLE:

```
use paramfile_io
print*,concatnl('a','bbbbbbbb','C 10 3')
```

Will return:

```
a
bbbbbbbb
C 10 3
```

RELATED ROUTINES

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

parse_xxx parse an ASCII file for parameters definition

convert_inplace*

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

Routine to convert a **HEALPix** map from NESTED to RING scheme or vice versa. The conversion is done in place, meaning that it doesn't require memory for a temporary map, like the *convert_nest2ring* or *convert_ring2nest* routines. But for that reason, this routine is slower and not parallelized. The routine is a wrapper for 6 different routines and can threfore process integer, single precision and double precision maps as well as mono or bi dimensional arrays.

FORMAT

call convert_inplace*(subcall, map)

ARGUMENTS

name & dimensionality	kind	in/out	description
subcall	—	IN	routine to be called by convert_inplace_real. Set this to ring2nest or nest2ring dependent on wether the conversion is RING to NESTED or vice versa.
map(0:npix-1)	I4B/ SP/ DP	INOUT	mono-dimensional full sky map to be converted, the routine finds the size itself.
map(0:npix-1,1:nd)	I4B/ SP/ DP	INOUT	bi-dimensional (nd> 0) full sky map to be converted, the routine finds both dimensions itself. Processing a bidimensional map with nd> 1 should be faster than each of the nd 1D-maps consecutively.

EXAMPLE:

```
call convert_inplace(ring2nest,map)
```

Converts an map from RING to NESTED scheme.

MODULES & ROUTINES

This section lists the modules and routines used by **convert_inplace***.

nest2ring	routine to convert a NESTED pixel index to RING pixel number.
ring2nest	routine to convert a RING pixel index to NESTED pixel number.

RELATED ROUTINES

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

convert_nest2ring	convert from NESTED to RING scheme using a temporary array. Requires more space then convert_inplace, but is faster.
convert_ring2nest	convert from RING to NESTED scheme using a temporary array. Requires more space then convert_inplace, but is faster.

convert_nest2ring*

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

Routine to convert a **HEALPix** map from NESTED to RING scheme.
The routine is a wrapper for 6 different routines and can threfore process integer, single precision and double precision maps as well as mono or bi dimensional arrays.
This routine is fast, and is parallelized for shared memory architecture, but requires extra memory to store a temporary map in.

FORMAT

call convert_nest2ring*(nside, map)

ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\rm side}$ parameter of the map to be converted.
map(0:12*nside**2-1)	I4B/ SP/ DP	INOUT	mono-dimensional full sky map to be converted to RING scheme.
map(0:12*nside**2-1,1:nd)	I4B/ SP/ DP	INOUT	bi-dimensional full sky map to be converted to RING scheme. The routine finds the second dimension (nd) by itself. Processing a bidimensional map with `nd`> 1 should be faster than each of the `nd` 1D-maps consecutively.

EXAMPLE:

```
call convert_nest2ring(256,map)
```

Converts an $N_{\rm side} = 256$ map given in array `map` from NESTED to RING scheme.

MODULES & ROUTINES

This section lists the modules and routines used by **convert_nest2ring***.

nest2ring	routine to convert a NESTED pixel index to RING pixel number.

RELATED ROUTINES

This section lists the routines related to **convert_nest2ring***.

convert_ring2nest	convert between RING and NESTED schemes.
convert_inplace	convert between NESTED and RING schemes inplace. This routine is slower than convert_nest2ring*, but doesn't require as much memory.

convert_ring2nest*

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

Routine to convert a **HEALPix** map from RING to NESTED scheme.
The routine is a wrapper for 6 different routines and can threfore process integer, single precision and double precision maps as well as mono or bi dimensional arrays.
This routine is fast, and is parallelized for shared memory architecture, but requires extra memory to store a temporary map in.

FORMAT

call convert_ring2nest*(nside, map)

ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\rm side}$ parameter of the map to be converted.
map(0:12*nside**2-1)	I4B/ SP/ DP	INOUT	mono-dimensional full sky map to be converted to RING scheme.
map(0:12*nside**2-1,1:nd)	I4B/ SP/ DP	INOUT	bi-dimensional full sky map to be converted to RING scheme. The routine finds the second dimension (nd) by itself. Processing a bidimensional map with `nd`> 1 should be faster than each of the `nd` 1D-maps consecutively.

EXAMPLE:

```
call convert_ring2nest(256,map)
```

Converts an $N_{\rm side} = 256$ map given in array `map` from RING to NESTED scheme.

MODULES & ROUTINES

This section lists the modules and routines used by **convert_ring2nest***.

ring2nest	routine to convert a RING pixel index to NESTED pixel number.

RELATED ROUTINES

This section lists the routines related to **convert_ring2nest***.

convert_nest2ring	convert between NESTED and RING schemes.
convert_inplace	convert between RING and NESTED schemes inplace. This routine is slower than convert_ring2nest*, but doesn't require as much memory.

coordsys2euler_zyz

Location in HEALPix directory tree: src/f90/mod/coord_v_convert.f90

This routine returns the three Euler angles ψ, θ, φ, corresponding to a rotation between standard astronomical coordinate systems. This angles can then be used in rotate_alm

FORMAT

call coordsys2euler_zyz(iepoch, oepoch, isys, osys, psi, theta, phi)

ARGUMENTS

name & dimensionality	kind	in/out	description
iepoch	DP	IN	epoch of the input astronomical coordinate system.
oepoch	DP	IN	epoch of the output astronomical coordinate system.
isys(len=*)	CHR	IN	input coordinate system, should be one of 'E'=Ecliptic, 'G'=Galactic, 'C'/'Q'=Celestial/eQuatorial.
osys(len=*)	CHR	IN	output coordinate system, same choice as above.
psi	DP	OUT	first Euler angle: rotation ψ about the z-axis.
theta	DP	OUT	second Euler angle: rotation θ about the original (unrotated) y-axis;
phi	DP	OUT	third Euler angle: rotation φ about the original (unrotated) z-axis;

EXAMPLE:

```
use coord_v_convert, only:  coordsys2euler_zyz
use alm_tools, only:  rotate_alm
...
call coordsys2euler_zyz(2000.0_dp, 2000.0_dp, 'E', 'G', psi, theta, phi)
```

```
call rotate_alm(64, alm_TGC, psi, theta, phi)
```

Rotate the $a_{\ell m}$ from Ecliptic to Galactic coordinates.

RELATED ROUTINES

This section lists the routines related to **coordsys2euler_zyz**.

rotate_alm	apply arbitrary sky rotation to a set of $a_{\ell m}$ coefficients.
xcc_v_convert	rotates a 3D coordinate vector from one astronomical coordinate system to another.

create_alm*

Location in HEALPix directory tree: src/f90/mod/alm_tools.F90

This routine generates scalar (and tensor) $a_{\ell m}$ for a temperature (and polarisation) power spectrum read from an input FITS file. The $a_{\ell m}$ are gaussian distributed with a zero mean, and their amplitude is multiplied with the ℓ-space window function of a gaussian beam characterized by its FWHM or an arbitrary circular beam and a pixel window read from an external file.

FORMAT

call create_alm*(nsmax, nlmax, nmmax, polar, filename, rng_handle, fwhm_arcmin, alm_TGC, header*[*, *windowfile*, *units*, *beam_file*]*)

Arguments appearing in *italic* are optional.

ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	$N_{\rm side}$ of the map to be synthetized from the $a_{\ell m}$ created by this routine.
nlmax	I4B	IN	maximum ℓ value to be considered (MAX=$4N_{\rm side}$ if `windowfile` is provided).
nmmax	I4B	IN	maximum m value for the $a_{\ell m}$.
polar	I4B	IN	**if set to 0**, only Temperature (scalar) $a_{\ell m}$ are generated using TT spectrum. **If set to 1**, 'conventional' polarization is added, based on EE, BB and TE spectra. **If set to 2**, and if the relevant information is in `filename`, polarization is generated assuming non-zero correlation of Curl (B) modes with Temperature (T) and Gradient (E) modes (TB and EB cross-spectra). Note that the `synfast` facility calls create_alm* with `polar`=0 *or* `polar`=1
filename(LEN=`filenamelen`)	CHR	IN	name of FITS file containing power spectra in the order TT, [EE, BB, TE, [TB, EB]] (terms in brackets are optional, see `polar`)
rng_handle	planck_rng	INOUT	structure containing information necessary to continue a random sequence initiated *previously* with the subroutine `rand_init`. Consecutive calls to create_alm* can be made after a single invocation to `rand_init`.
fwhm_arcmin	SP/ DP	IN	FWHM size of the gaussian beam in arcminutes.
alm_TGC(1:p,0:nlmax,0:nmmax)	SPC/ DPC	OUT	complex $a_{\ell m}$ values generated from the power spectrum in the FITS file. The first index here runs form 1:1 for temperature only, and 1:3 for polarisation. In the latter case, 1=T, 2=E, 3=B.

name & dimensionality	kind	in/out	description
header(LEN=80),dimension(60)	CHR	OUT	part of header which will be included in the FITS-file containing the map synthesised from the $a_{\ell m}$ which create_alm generates.
windowfile(LEN=`filenamelen`)	CHR	IN	full filename specification of the FITS file with the pixel window function (defined for $\ell \leq 4N_{\rm side}$)
units(LEN=80),dimension(1:)	CHR	OUT	physical units of the created $a_{\ell m}$ (square-root of the input power spectrum units).
beam_file(LEN=`filenamelen`)	CHR	IN	name of the file containing the (non necessarily gaussian) window function B_ℓ of a circular beam. If present, it will override the argument `fwhm_arcmin`.

EXAMPLE:

```
use alm_tools, only:  create_alm
use rngmod, only:  rand_init, planck_rng
type(planck_rng) ::  rng_handle

call rand_init(rng_handle, -1)
call create_alm(64, 128, 128, 1, 'cl.fits', rng_handle, 5.0, alm_TGC, &
& header, 'data/pixel_window_n0064.fits')
```

Creates scalar and tensor $a_{\ell m}$ from the power spectrum given in the file 'cl.fits'. The map to be created from these $a_{\ell m}$ is assumed to have $N_{\rm side} = 64$. C_ℓs from the power spectrum are used up to an ℓ value of 128. Corresponding $a_{\ell m}$ values up to l=128 and m=128 are created as gaussian distributed complex numbers. Their are drawn from a sequence of pseudo-random numbers initiated with a seed of -1. The produced $a_{\ell m}$ are convolved with a gaussian beam of FWHM 5 arcminutes and a pixel window read from 'data/pixel_window_n0064.fits'. It is assumed that after the return from this routine, a map is generated from the created $a_{\ell m}$. For this purpose, `header` is updated with FITS format information describing the origin and history of these $a_{\ell m}$.

MODULES & ROUTINES

This section lists the modules and routines used by **create_alm***.

alm_tools	module, containing:
pow2alm_units	routine to convert from power spectrum units to $a_{\ell m}$ units
generate_beam	routine to generate beam window function
pixel_window	routine to read in pixel window function
utilities	module, containing:
die_alloc	routine that prints an error message if there is not enough space for allocation of variables.
fitstools	module, containing:
fits2cl	routine to read a FITS file containing a power spectrum.
read_dbintab	routine to read a FITS-binary file containing the pixel window functions.
head_fits	module, containing:
add_card	routine to add a keyword to a FITS header.
get_card	routine to read a keyword value from FITS header.
merge_headers	routine to merge two FITS headers.
rngmod	module, containing:
rand_gauss	function which returns a gaussian distributed random number.

RELATED ROUTINES

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

rand_init	subroutine to initiate a random number sequence.
synfast	executable using create_alm* to synthesize CMB maps from a given power spectrum.
alm2map	Routine to transform a set of $a_{\ell m}$ created by create_alm* to a **HEALPix** map.
alms2fits, dump_alms	Routines to save a set of $a_{\ell m}$ in a FITS file.

del_card

Location in HEALPix directory tree: src/f90/mod/head_fits.F90

This routine removes one or several keywords from a FITS header.

FORMAT

call del_card(header, kwds)

ARGUMENTS

name & dimensionality	kind	in/out	description
header(LEN=80)(1:nlheader)	CHR	INOUT	The header to remove the keyword(s) from. The routines finds out the header size.
kwds(LEN=20)(1:nkws)	CHR	IN	list of FITS keywords to remove. The routine accepts either a vector a keywords or a single one in a scalar variable
kwds(LEN=20)	CHR	IN	the one FITS keyword to remove.

EXAMPLES: #1

```
call del_card(header,(/ 'NSIDE ','COORD ','ORDERING' /) )
```

Removes the keywords 'NSIDE', 'COORD' and 'ORDERING' from Header

EXAMPLES: #2

```
call del_card(header, 'ORDERING' )
```

Removes the keyword 'ORDERING' from Header

MODULES & ROUTINES

This section lists the modules and routines used by **del_card**.

write_hl	more general routine for adding a keyword to a header.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

add_card	general purpose routine to write any keywords into a FITS file header
get_card	general purpose routine to read any keywords from a header in a FITS file.
read_par, number_of_alms	routines to read specific keywords from a header in a FITS file.
getsize_fits	function returning the size of the data set in a fits file and reading some other useful FITS keywords
merge_headers	routine to merge two FITS headers

dist2holes_nest

Location in HEALPix directory tree: src/f90/mod/mask_tools.F90

For a input binary mask in NESTED ordering, dist2holes_nest returns the angular distance (in radians) from each *valid* (1-valued) pixel to the closest *invalid* (0-valued) pixel. Distances are measured between pixel centers.

FORMAT

call dist2holes_nest(nside, mask, distance)

ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\rm side}$ value of the input mask.
mask(0:Npix-1)	I4B	IN	Input NESTED-ordered mask. Npix = 12*nside*nside
distance(0:Npix-1)	DP	OUT	Output NESTED-ordered angular-distance map

EXAMPLE:

```
use healpix_types
use healpix_modules
...
call dist2holes_nest(nside, mask, distance)
```

???

MODULES & ROUTINES

This section lists the modules and routines used by **dist2holes_nest**.

mask_tools mask processing module (see related routines below)

RELATED ROUTINES

This section lists the routines related to **dist2holes_nest**.

dist2holes_nest	angular distance to closest invalid pixel of the given mask
fill_holes_nest	turn to *valid* all pixels located in 'holes' containing fewer pixels than the given threshold
maskborder_nest	identify inner boundary pixels of 'holes' for given mask
size_holes_nest	returns size (in pixels) of holes found in input mask

dump_alms*

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This routine stores $a_{\ell m}$ values in a binary FITS file. The FITS file created will contain one integer column with $index = \ell^2 + \ell + m + 1$ and 2 single precision columns with real/imaginary $a_{\ell m}$ values. One can store temperature $a_{\ell m}$ or polarisation, $a^E_{\ell m}$ or $a^B_{\ell m}$. If temperature is specified, a FITS file is created. If polarisation is specified, an old FITS file is opened and extra extensions is created.

FORMAT call dump_alms*(filename, alms, nlmax, header, nlheader, extno)

ARGUMENTS

name & dimensionality	kind	in/out	description
filename(LEN=`filenamelen`)	CHR	IN	filename for the FITS-file to store the $a_{\ell m}$ in.
nlmax	I4B	IN	maximum ℓ value to store.
alms(0:nlmax,0:nlmax)	SPC/ DPC	IN	array with $a_{\ell m}$, in the format used by eg. map2alm, so `alms(1,m)` corresponds to $a_{\ell m}$
extno	I4B	IN	extension number. If 0 is specified, a FITS file is created and $a_{\ell m}$ is stored in the first FITS extension as temperature $a_{\ell m}$. If 1 or 2 is specified, an already existing file is opened and a 2nd or 3rd extension is created, treating $a_{\ell m}$ as $a^E_{\ell m}$ or $a^B_{\ell m}$.
nlheader	I4B	IN	number of header lines to write to the file.
header(LEN=80) (1:nlheader)	CHR	IN	the header to the FITS-file.

EXAMPLE:

```
call dump_alms ('alms.fits', alms, 64, header, 100, 1)
```

Opens an already existing FITS file which contains temperature $a_{\ell m}$. An extra extension is added to the file where the $a_{\ell m}$ array are written in a three-column format as described above. 100 header lines are written to the file from the array header(1:80).

MODULES & ROUTINES

This section lists the modules and routines used by **dump_alms***.

fitstools	module, containing:
printerror	routine for printing FITS error messages.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

fits2alms, read_conbintab	routines to read $a_{\ell m}$ from a FITS-file
alms2fits	has the same function as dump_alms* but is more general.

fill_holes_nest

Location in HEALPix directory tree: src/f90/mod/mask_tools.F90

For a input binary mask in NESTED ordering, fill_holes_nest flip to 1 all pixels located in invalid regions with fewer pixels than the threshold provided.
Two pixels are adjacent (and belong to the same region or hole) if they have at least one point in common.

FORMAT

call fill_holes_nest(nside, new_min_size, mask_in, mask_out)

ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	The $N_{\rm side}$ value of the input mask.
new_min_size	I4B	IN	Minimal size of hole (in pixels) on output
mask_in(0:Npix-1)	I4B	IN	Input NESTED-ordered mask. Npix = 12*nside*nside
mask_out(0:Npix-1)	I4B	OUT	Output NESTED-ordered mask. Can be the same array as mask_in.

EXAMPLE:

```
use healpix_types
use healpix_modules
...
call fill_holes_nest(nside, new_min_size, mask_in, mask_in)
```

???

MODULES & ROUTINES

This section lists the modules and routines used by **fill_holes_nest**.

mask_tools	mask processing module (see related routines below)

RELATED ROUTINES

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

dist2holes_nest	angular distance to closest invalid pixel of the given mask
fill_holes_nest	turn to *valid* all pixels located in 'holes' containing fewer pixels than the given threshold
maskborder_nest	identify inner boundary pixels of 'holes' for given mask
size_holes_nest	returns size (in pixels) of holes found in input mask

fits2alms*

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This routine reads $a_{\ell m}$ values from a binary FITS file. Each FITS file extension is supposed to contain one integer column with $index = \ell^2+\ell+m+1$ and 2 or 4 single (or double) precision columns with real/imaginary $a_{\ell m}$ values and real/imaginary standard deviation. One can read temperature $a_{\ell m}$ or temperature and polarisation, $a^T_{\ell m}$, $a^E_{\ell m}$ and $a^B_{\ell m}$.

FORMAT

call fits2alms*(filename, nalms, alms, ncl, header, nlheader, next)

ARGUMENTS

name & dimensionality	kind	in/out	description
filename(LEN=`filenamelen`)	CHR	IN	filename of the FITS-file to read the $a_{\ell m}$ from.
nalms	I4B	IN	number of $a_{\ell m}$ to read.
ncl	I4B	IN	number of columns to read in the FITS file. If an standard deviation is to be read, this number is 5, otherwise it is 3.
next	I4B	IN	the number of extensions to read. 1 for temperature only, 3 for temperature and polarisation.

alms(1:nalms,1:(ncl+1),1:next)	SP/ DP	OUT	the $a_{\ell m}$ to read from the file. alms(i,1,j) and alms(i,2,j) contain the ℓ and m values for the ith $a_{\ell m}$ (j=1,2,3 for (T,E,B)). alms(i,3,j) and alms(i,4,j) contain the real and imaginary value of the ith $a_{\ell m}$. Finally, the standard deviation for the ith $a_{\ell m}$ is contained in alms(i,5,j) (real) and alms(i,6,j) (imaginary).
nlheader	I4B	IN	number of header lines to read from the file.
header(LEN=80) (1:nlheader, 1:next)	CHR	OUT	the header(s) read from the FITS-file.

EXAMPLE:

```
call fits2alms ('alms.fits', 65*66/2, alms, 3, header, 80, 3)
```

Reads a FITS file with the $a^T_{\ell m}$, $a^E_{\ell m}$ and $a^B_{\ell m}$ values read into alms(1:65*66/2,1:4,1:3). The last index specifies (T,E,B). The second index gives l, m, real($a_{\ell m}$), imaginary($a_{\ell m}$) for each of the $a_{\ell m}$. The number 65*66/2 is the number of $a_{\ell m}$ values up to an ℓ value of 64. 80 lines is read from the header in each extension and returned in header(1:80,1:3).

MODULES & ROUTINES

This section lists the modules and routines used by **fits2alms***.

read_alms	routine called by fits2alms* for each extension.
fitstools	module, containing:
printerror	routine for printing FITS error messages.
cfitsio	library for FITS file handling.

RELATED ROUTINES

This section lists the routines related to **fits2alms***.

alms2fits, dump_alms	routines to store $a_{\ell m}$ in a FITS-file
read_conbintab	has the same function as fits2alms* but with parameters passed differently.
number_of_alms, getsize_fits	can be used to find out the number of $a_{\ell m}$ available in the file.

fits2cl*

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This routine reads a power spectrum or beam window function from a FITS ASCII or binary table. The routine can read temperature coefficients C_ℓ^{TT} or both temperature and polarisation coefficients C_ℓ^{TT}, C_ℓ^{EE}, C_ℓ^{BB}, C_ℓ^{TE} (and C_ℓ^{TB}, C_ℓ^{EB}, C_ℓ^{ET}, C_ℓ^{BT}, C_ℓ^{BE} when applicable). If the keyword PDMTYPE is found in the header, fits2cl assumes the table to be in the special format used by *Planck* and will ignore the first data column. If the input FITS file contains several extensions or HDUs, the one to be read can be specified thanks to the CFITSIO Extended File Name Syntax, using its number (eg, `file.fits[2]` or `file.fits+2`) or its `EXTNAME` value (eg. `file.fits[beam_100x100]`). By default, only the first valid extension will be read.

FORMAT

call fits2cl*(filename, clin, lmax, ncl, header, [units])

ARGUMENTS

name & dimensionality	kind	in/out	description
filename(LEN=`filenamelen`)	CHR	IN	the FITS file containing the power spectrum.
lmax	I4B	IN	Maximum ℓ value to be read.
ncl	I4B	IN	1 for temperature coeffecients only, 4 for polarisation.
clin(0:lmax,1:ncl)	SP/ DP	OUT	the power spectrum read from the file.
header(LEN=80) (1:)	CHR	OUT	the header read from the FITS-file.
units(LEN=80) (1:)	CHR	OUT	the column units read from the FITS-file.

EXAMPLE:

```
use healpix_modules
real(SP), allocatable, dimension(:,:)  ::  cl
character(len=80), dimension(1:300) ::  header
character(len=80), dimension(1:100) ::  units
integer(I4B) ::  lmax, ncl, np
character(len=filenamelen) ::  fitsfile='cl.fits'
np = getsize_fits(fitsfile, nmaps=ncl, mlpol=lmax)
allocate(cl(0:lmax, 1:ncl))
call fits2cl(fitsfile, cl, lmax, ncl, header, units)
```

Reads a power spectrum from the FITS file `'cl.fits'` and stores the result in `cl(0:lmax,1:ncl)` which are the `ncl` C_ℓ coefficients up to ℓ =`lmax`. The FITS header is returned in `header`, the column units in `units`.

MODULES & ROUTINES

This section lists the modules and routines used by **fits2cl***.

fitstools	module, containing:
printerror	routine for printing FITS error messages.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

create_alm	Routine to create $a_{\ell m}$ values from an input power spectrum.
write_asctab	Routine to create an ascii FITS file containing a power spectrum.
getsize_fits	Routine to parse FITS file header, and determine the data storage features.
getnumext_fits	Routine to determine number of extensions of a FITS file.

gaussbeam

Location in HEALPix directory tree: src/f90/mod/alm_tools.F90

This routine generates the beam window function in multipole space of a gaussian beam parametrized by its FWHM. The polarization beam is also provided assuming a perfectly co-polarized beam (eg, Challinor et al 2000, astro-ph/0008228)

FORMAT

call gaussbeam(fwhm_arcmin, lmax, beam)

ARGUMENTS

name & dimensionality	kind	in/out	description
fwhm_arcmin	DP	IN	FWHM of the gaussian beam in arcminutes.
lmax	I4B	IN	maximum ℓ value of the window function.
beam(0:lmax,1:p)	DP	OUT	beam window function generated. The second index runs form 1:1 for temperature only, and 1:3 for polarisation. In the latter case, 1=T, 2=E, 3=B.

EXAMPLE:

```
call gaussbeam(5.0_dp, 1024, beam)
```

Generates the window function of a gaussian beam of FWHM = 5 arcmin, for $\ell \leq 1024$.

RELATED ROUTINES

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

generate_beam — Routine returning a beam window function.

pixel_window Routine returning a pixel window function.

generate_beam

Location in HEALPix directory tree: src/f90/mod/alm_tools.F90

This routine generates the beam window function in multipole space. It is either a gaussian parametrized by its FWHM in arcmin in real space, or it is read from an external file.

FORMAT

call generate_beam(fwhm_arcmin, lmax, beam[, *beam_file*])

Arguments appearing in *italic* are optional.

ARGUMENTS

name & dimensionality	kind	in/out	description
fwhm_arcmin	DP	IN	fwhm size of the gaussian beam in arcminutes.
lmax	I4B	IN	maximum ℓ value of the window function.
beam(0:lmax,1:p)	DP	OUT	beam window function generated. The second index runs form 1:1 for temperature only, and 1:3 for polarisation. In the latter case, 1=T, 2=E, 3=B.
beam_file(LEN=`filenamelen`) (OPTIONAL)	CHR	IN	name of the file containing the (non necessarily gaussian) window function B_ℓ of a circular beam. If present, it will override the argument `fwhm_arcmin`. If fewer columns than requested are found in the file, missing colums will duplicate the existing ones (based on the assumption that B_ℓ is the same in T, E and B). Supports the `fitsio` 'Extended Filename Syntax' (see examples below).

EXAMPLE:

```
use healpix_modules
real(dp), dimension(0:1024, 1:3) ::  gb0, b1, b2, b3
call generate_beam(5.0_dp, 1024, gb0)
call generate_beam(0_dp, 1024, b1, beam_file='file.fits')
call generate_beam(0_dp, 1024, b2, beam_file='file.fits[col 1]')
call generate_beam(0_dp, 1024, b3, beam_file='file.fits[col 1; 2=0; 3=0]')
```

`gb0` will contain the window function of a gaussian beam of FWHM = 5 arcmin, for $\ell \leq 1024$.
`b1` will contain the first 3 columns (if available) of `file.fits`. If the file contains only two columns, then `b1(:,3) = b1(:,2)`, and if it contains a single column, then `b1(:,3) = b1(:,2) = b1(:,1)`.
`b2` will be based on a virtual FITS file containg only the first column of `file.fits`, and we will have `b2(:,3) = b2(:,2) = b2(:,1)`.
Finally `b3` will read a virtual FITS file in which the first column is the same as in `file.fits`, while the columns 2 and 3 are set to 0. Therefore `b3(:,3) = b3(:,2) = 0`.

MODULES & ROUTINES

This section lists the modules and routines used by **generate_beam**.

alm_tools	module, containing:
gaussbeam	routine to generate a gaussian beam

RELATED ROUTINES

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

create_alm	Routine to create $a_{\ell m}$ coefficients using generate_beam.
alter_alm	Routine to alter $a_{\ell m}$ coefficients using generate_beam.
pixel_window	Routine returning a pixel window function.

get_card

Location in HEALPix directory tree: src/f90/mod/head_fits.F90

This routine reads a keyword of any kind from a FITS header. It is a wrapper to other routines that read keywords of different kinds.

FORMAT

call get_card(header, kwd, value, comment)

ARGUMENTS

name & dimensionality	kind	in/out	description
header(LEN=80) DIMENSION(:)	CHR	IN	The header to read the keyword from.
kwd(LEN=8)	CHR	IN	the FITS keyword to read (NOT case sensitive).
value	any	OUT	the value read for the keyword. The type of the fortran variable 'value' (double, real, integer, logical or character) should match the type under which the value is written in the FITS file, except if 'value' is a character string, in which case it can read any keyword value, or if 'value' if real or double, in which case it can read any numerical value. Note that long string values (more than 68 characters in length) are supported.
comment(LEN=*)	CHR	OUT	comment read for the keyword.

EXAMPLE:

```
call get_card(header,'NsIdE',nside,comment)
```

if `nside` is defined as an integer, it will contain on output the value of NSIDE (say 256) found in header

EXAMPLE:

```
call get_card(header,'ORDERING',ordering,comment)
```

if `ordering` is defined as an character string, it will contain on output the value of ORDERING (say 'RING') found in header

MODULES & ROUTINES

This section lists the modules and routines used by **get_card**.

cfitsio	library for FITS file handling.

RELATED ROUTINES

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

add_card	general purpose routine to write any keywords into a FITS file header
del_card	routine to discard a keyword from a FITS header
read_par, number_of_alms	routines to read specific keywords from a header in a FITS file.
getsize_fits	function returning the size of the data set in a fits file and reading some other useful FITS keywords
merge_headers	routine to merge two FITS headers

get_healpix_main_dir, . . .

Location in HEALPix directory tree: src/f90/mod/paramfile_io.F90

A few functions are available to return the full path to **HEALPix** main directory and its `data` and `test` subdirectories. This allow those paths to be controlled by preprocessing macros or environment variables in case of non-standard installation of the **HEALPix** directory structure.

FUNCTIONS:

`hmd = get_healpix_main_dir()`

returns the full path to the main **HEALPix** directory. It will be determined, in this order, from the value of the preprocessing macros `HEALPIX` and `HEALPIXDIR` if they are defined or the environment variable `$HEALPIX` otherwise.

`hdd = get_healpix_data_dir()`

returns the full path to **HEALPix** `data` subdirectory. It will be determined from the preprocessing macro `HEALPIXDATA` or the environment variable `$HEALPIXDATA`. If both fail, it will return the list of directories {`. ../data ./data .. $HEALPIX $HEALPIX/data $HEALPIX/../data $HEALPIX\data`} separated by LineFeed.

`htd = get_healpix_test_dir()`

returns the full path to **HEALPix** `test` subdirectory. It will be determined, in this order, from the preprocessing macro `HEALPIXTEST`, the environment variable `$HEALPIXTEST` or `$HEALPIX/test`.

get_healpix_weight_file, ...

Location in HEALPix directory tree: src/f90/mod/paramfile_io.F90

Functions are available to return the standardized name of pre-generated FITS files required by some **HEALPix** facilities and subroutines. Those files are available in the `data` subdirectory, whose full path is returned by companion functions such as `get_healpix_data_dir`.

FUNCTIONS:

`winfile = get_healpix_pixel_window_file(nside)`

returns, for the `nside` provided, the name of the FITS file containing the window function associated with the **HEALPix** pixel (of the form `pixel_window_n*.fits`).

`w8file = get_healpix_weight_file(nside, won)`

returns the name of the FITS file containing the ring-based or pixel-based weights for the given **HEALPix** resolution parameter `nside`. If `won` = 1, the output of `get_healpix_ring_weight_file(nside)` is returned, while if `won` = 2, it is the output of `get_healpix_pixel_weight_file(nside)`. `won`= 0 will return an empty string, while other choices of `won` will result in errors.

`w8rfile = get_healpix_ring_weight_file(nside)`

returns, for the `nside` provided, the name of the FITS file containing the ring-based weights (of the form `weight_ring_n*.fits`).

`w8pfile = get_healpix_pixel_weight_file(nside)`

returns, for the `nside` provided, the name of the FITS file containing the pixel-based weights (of the form `weight_pixel_n*.fits`). Some of them (for power of 2 `nside` in $[16, 2048]$) are located in the `data` directory, the other ones can be generated with the `compute_weights` C++ facility.

getArgument

Location in HEALPix directory tree: src/f90/mod/extension.F90

This subroutine emulates the C routine `getarg`, which returns the value of a given command line argument.
Starting with release 3.60, it calls the F2003 extension subroutine `get_command_argument`.

FORMAT

call getArgument(index, value)

ARGUMENTS

name & dimensionality	kind	in/out	description
index	I4B	IN	index of the command line argument (where the first argument has index 1)
value	CHR	OUT	value of the argument

RELATED ROUTINES

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

getEnvironment	returns value of environment variable
nArguments	returns number of command line arguments

getEnvironment

Location in HEALPix directory tree: src/f90/mod/extension.F90

This subroutine emulates the C routine `getenv`, which returns the value of an environment variable.
Starting with release 3.60, it calls the F2003 extension subroutine `get_environment_variable`.

FORMAT

call getEnvironment(name, value)

ARGUMENTS

name & dimensionality	kind	in/out	description
name	CHR	IN	name of the environment variable
value	CHR	OUT	value of the environment variable

EXAMPLE:

```
use extension
character(len=128) ::  healpixdir
call getEnvironment('HEALPIX', healpixdir)
print*,healpixdir
```

Will return the value of the `$HEALPIX` system variable (if it is defined)

RELATED ROUTINES

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

getArgument	returns list of command line arguments
nArguments	returns number of command line arguments

getdisc_ring

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

This routine is obsolete, use query_disc instead

getnumext_fits

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This routine returns the number of extensions present in a given FITS file.

FORMAT

var=getnumext_fits(filename)

ARGUMENTS

name & dimensionality	kind	in/out	description
var	I4B	OUT	number of extensions in the FITS file (excluding the primary unit). According to the current format, **HEALPix** files have at least one extension.
filename(LEN=`filenamelen`)	CHR	IN	filename of the FITS file.

EXAMPLE:

```
next = getnumext_fits('map.fits')
```

Returns in `next` the number of extensions present in the FITS file 'map.fits'.

MODULES & ROUTINES

This section lists the modules and routines used by **getnumext_fits**.

fitstools	module, containing:
printerror	routine for printing FITS error messages.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

getsize_fits	routine returning the number of data points in a FITS file, as well as much more information on the file.
input_map	routine to read a **HEALPix** FITS file

getsize_fits

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

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

FORMAT var=getsize_fits(filename[, *nmaps*, *ordering*, *obs_npix*, *nside*, *mlpol*, *type*, *polarisation*, *fwhm_arcmin*, *beam_leg*, *coordsys*, *polcconv*, *extno*])

Arguments appearing in *italic* are optional.

ARGUMENTS

name & dim.	kind	in/out	description
var	I8B	OUT	number of pixels or time samples in the chosen extension of the FITS file
filename(LEN=*)	CHR	IN	filename of the FITS-file containing **HEALPix** map(s).

name & dim.	kind	in/out	description
nmaps (OPTIONAL)	I4B	OUT	number of maps in the extension.
ordering (OPTIONAL)	I4B	OUT	pixel ordering, 0=unknown, 1=RING, 2=NESTED
obs_npix (OPTIONAL)	I4B	OUT	number of non blanck pixels. It is set to -1 if it can not be determined from header information alone
nside (OPTIONAL)	I4B	OUT	Healpix resolution parameter Nside. Returns a negative value if not found.
mlpol (OPTIONAL)	I4B	OUT	maximum multipole used to generate the map (for simulated map). Returns a negative value if not found.
type (OPTIONAL)	I4B	OUT	Healpix/FITS file type <0 : file not found, or not valid 0 : image only fits file, deprecated Healpix format (var = 12 * nside * nside) 1 : ascii table, generally used for C(l) storage 2 : binary table : with implicit pixel indexing (full sky) (var = 12 * nside * nside) 3 : binary table : with explicit pixel indexing (generally cut sky) (var $\leq$ 12 * nside * nside) 999 : unable to determine the type
polarisation (OPTIONAL)	I4B	OUT	presence of polarisation data in the file <0 : can not find out 0 : no polarisation 1 : contains polarisation (Q,U or G,C)
fwhm_arcmin (OPTIONAL)	DP	OUT	returns the beam FWHM read from FITS header, translated from Deg (hopefully) to arcmin. Returns a negative value if not found.
beam_leg(LEN=*) (OPTIONAL)	CHR	OUT	filename of beam or filtering window function applied to data (FITS keyword BEAM_LEG). Returns a empty string if not found.
coordsys(LEN=20) (OPTIONAL)	CHR	OUT	string describing the pixelation astrophysical coordinates. 'G' = Galactic, 'E' = ecliptic, 'C' = celestial = equatorial. Returns a empty string if not found.
polcconv (OPTIONAL)	I4B	OUT	polarisation coordinate convention (see Healpix primer for details) 0=unknown, 1=COSMO, 2=IAU, 3=neither COSMO nor IAU
extno (OPTIONAL)	I4B	IN	extension number (0 based) for which information is provided. Default = 0 (first extension).

EXAMPLE:

```
npix= getsize_fits('map.fits', nmaps=nmaps, ordering=ordering,
obs_npix=obs_npix, nside=nside, mlpol=mlpol, type=type,
polarisation=polarisation)
```

Returns 1 or 3 in nmaps, dependent on wether 'map.fits' contain only temperature or both temperature and polarisation maps. The pixel ordering number is found by reading the keyword ORDERING in the FITS file. If this keyword does not exist, 0 is returned.

MODULES & ROUTINES

This section lists the modules and routines used by **getsize_fits**.

fitstools	module, containing:
printerror	routine for printing FITS error messages.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

getnumext_fits	routine returning the number of extension in a FITS file
input_map	routine to read a **HEALPix** FITS file

healpix_modules

Location in HEALPix directory tree: src/f90/mod/healpix_modules.f90

This module is a meta module containing most of the **HEALPix** modules. It currently includes

- alm_tools,
- bit_manipulation,
- coord_v_convert,
- extension,
- fitstools,
- head_fits,
- healpix_fft,
- healpix_types,
- long_intrinsic,
- mask_tools,
- misc_utils,
- num_rec,
- obsolete,
- paramfile_io,
- pix_tools,
- ran_tools,
- rngmod,
- statistics,
- udgrade_nr,
- utilities.

Note that mpi_alm_tools is not included since it requires the MPI library for compilation.

EXAMPLE:

```
use healpix_modules
print*,' pi = ',PI
print*,' number of pixels in a Nside=64 map:',nside2npix(64)
```

Invoking healpix_modules gives access to all **HEALPix** routines and parameters.

healpix_types

Location in HEALPix directory tree: src/f90/mod/healpix_types.F90

This module defines a set of parameters used by most other **HEALPix** modules.

The parameters defined in healpix_types include

- 'kind' parameters, used when defining the type of a variable,

name	type	value[a]	definition
I1B	integer	1	number of bytes in the hardware-supported signed integers covering the range -99 to 99 with the least margin
I2B	integer	2	same as above for the range -9999 to 9999 (ie, 4 digits)
I4B	integer	4	same as above for 9 digits
I8B	integer	8	same as above for 16 digits[b]
SP	integer	4	number of bytes in the hardware-supported floating-point numbers covering the range 10^{-30} to 10^{30} with the least margin (hereafter single precision)
DP	integer	8	same as above for the range 10^{-200} to 10^{200} (double precision)
SPC	integer	4	number of bytes in real (*or* imaginary) part of single precision complex numbers
DPC	integer	8	same as above for double precision complex numbers
LGT	integer	4	number of bytes in logical variables

[a] actual value may depend on hardware or compiler

[b] may not be supported by some hardware or compiler; on those systems, the user should set the preprocessing variable `NO64BITS` to 1 during compilation to demote automatically `I8B` to `I4B`

- largest accessible numbers,

name	type or kind	value[a]	definition
MAX_I1B	integer	127	largest number accessible to integers of kind `I1B`
MAX_I2B	integer	32767	same as above for `I2B` integers
MAX_I4B	integer	$2^{31} - 1 \simeq 2.1\ 10^{9}$	same as above for `I4B` integers
MAX_I8B	I8B	$2^{63} - 1 \simeq 9.2\ 10^{18}$	same as above for `I8B` integers
MAX_SP	SP	$\simeq 3.40\ 10^{38}$	same as above for `SP` floating-point
MAX_DP	DP	$\simeq 1.80\ 10^{308}$	same as above for `DP` floating-point

[a] actual value may depend on hardware or compiler

- mathematical definitions,

name	kind	value	definition
QUARTPI	DP	$\pi/4$	
HALFPI	DP	$\pi/2$	
PI	DP	$\pi \simeq 3.14159\ldots$	
TWOPI	DP	2π	
FOURPI	DP	4π	
SQRT2	DP	$\sqrt{2}$	
EULER	DP	$\gamma \simeq 0.577\ldots$	Euler constant
SQ4PI_INV	DP	$1/\sqrt{4\pi}$	
TWOTHIRD	DP	$2/3$	
DEG2RAD	DP	$\pi/180$	Degrees to Radians conversion factor
RAD2DEG	DP	$180/\pi$	Radians to Degrees conversion factor

- and **HEALPix** specific definitions,

name	type or kind	value	definition
HPX_SBADVAL	SP	$-1.6375\ 10^{30}$	default sentinel value given to missing pixels in single precision data sets
HPX_DBADVAL	DP	$-1.6375\ 10^{30}$	same as above for double precision data sets
FILENAMELEN	integer	1024	default length in character of file names.
HEALPIX_VERSION	character	"3.83"	current **HEALPix** package version.

EXAMPLE:

```
use healpix_types
real(kind=DP) ::  dx
print*,' pi = ',PI
```

The value of `PI`, as well as all other healpix_types parameters are made known to the code

in_ring

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

Routine to find the pixel index of all pixels on a slice of a given ring. The output indices can be either in the RING or NESTED scheme, depending on the `nest` keyword.

FORMAT

call in_ring(nside, iz, phi0, dphi, listir, nir, nest)

ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\rm side}$ parameter of the map.
iz	I4B	IN	ring number, counted southwards from the north pole.
phi0	DP	IN	central ϕ position in the slice.
dphi	DP	IN	defines the size of the slice. The slice has length $2 \times dphi$ along the ring with center at $phi0$.
listir(0:4*nside-1)	I4B/ I8B	OUT	The pixel indexes in the slice.
nir	I4B	OUT	the number of pixels in the slice. `nir`$\leq 4N_{\rm side}$
nest (OPTIONAL)	I4B	IN	The pixel indexes are in the NESTED numbering scheme if `nest`=1, and in RING scheme otherwise.

EXAMPLE:

```
call in_ring(256, 10, 0, 0.1, listir, nir, nest=1)
```

Returns the NESTED pixel index of all pixels within 0.1 radians on each side of $\phi = 0$ on the 10th ring.

MODULES & ROUTINES

This section lists the modules and routines used by **in_ring**.

ring2nest	conversion from RING scheme pixel index to NESTED scheme pixel index
next_in_line_nest	returns NESTED index of pixel lying to the East of the current pixel and on the same ring

RELATED ROUTINES

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

pix2ang, ang2pix	convert between angle and pixel number.
pix2vec, vec2pix	convert between a cartesian vector and pixel number.
getdisc_ring	find all pixels within a certain radius.

input_map*

Location in HEALPix directory tree: **src/f90/mod/fitstools.F90**

This routine reads a **HEALPix** map from a FITS file. This can deal with full sky as well as cut sky maps, but always outputs a full sky map (with possibly many empty pixels).

FORMAT

call input_map*(filename, map, npixtot, nmaps*[*, *fmissval*, *header*, *units*, *extno*, *ignore_polcconv*]*)

Arguments appearing in *italic* are optional.

ARGUMENTS

name & dimensionality	kind	i/o	description
filename(len=`filenamelen`)	CHR	IN	FITS file to be read from, containing a full sky or cut sky map
map(0:npixtot-1,1:nmaps)	SP/ DP	OUT	full sky map(s) constructed from the data present in the file, missing pixels are filled with fmissval
npixtot	I4B/ I8B	IN	number of pixels in the full sky map
nmaps	I4B	IN	number of maps in the file
fmissval	SP/ DP	IN	value to be given to missing pixels, (**default:** 0)
header(LEN=80)(1:)	CHR	OUT	FITS extension header
units(LEN=20)(1:nmaps)	CHR	OUT	maps units
extno	I4B	IN	extension number to read the data from (0 based).(**default:** 0) (the first extension is read)
ignore_polcccconv	LGT	IN	by default (`ignore_polcconv=.false.`) the output of this routine depends on the value of the FITS keyword `POLCCONV` found in filename, as described in the note on POLCCONV in The **HEALPix** Primer. Setting `ignore_polcconv=.true.` will force input_map to ignore that keyword.

EXAMPLE:

```
use pix_tools, only:  nside2npix
use fitstools, only:  getsize_fits, input_map
...
npixtot = getsize_fits('map.fits',nmaps=nmaps, nside=nside)
npix = nside2npix(nside)
allocate(map(0:npix-1,1:nmaps))
call input_map('map.fits', map, npix, nmaps, fmissval=0.)
```

Reads into `map` the content of the FITS file 'map.fits'. If there are missing pixels in the input file (ie, having value `NaN` (Not of Number), ± `Infinity` or matching the FITS keyword `BAD_DATA`) they will take on output the value provided in optional `fmissval` (here 0, which also is its default value).

MODULES & ROUTINES

This section lists the modules and routines used by **input_map***.

fitstools	module, containing:
printerror	routine for printing FITS error messages.
read_bintab	routine to read a binary table from a FITS file
read_fits_cut4	routine to read cut sky map from a FITS file
read_fits_partial	routine to read a partial sky map from a FITS file
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

anafast	executable that reads a **HEALPix** map and analyses it.
synfast	executable that generate full sky **HEALPix** maps
getsize_fits	subroutine to know the size of a FITS file.
output_map	subroutine to write a FITS file from a **HEALPix** map
write_bintabh	subroutine to write a large array into a FITS file piece by piece
input_tod*	subroutine to read an arbitrary subsection of a large binary table

input_tod*

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This routine reads a large binary table (for instance a Time Ordered Data set) from a FITS file. The user can choose to read only a section of the table, starting from an arbitrary position. The data can be read into a single or double precision array.

FORMAT

call input_tod*(filename, tod, npix, ntods[, *header*, *firstpix*, *fmissval*])

Arguments appearing in *italic* are optional.

ARGUMENTS

name & dimensionality	kind	in/out	description
filename(LEN=`filenamelen`)	CHR	IN	FITS file to be read from
tod(0:npix-1,1:ntods)	SP/ DP	OUT	array constructed from the data present in the file (from the sample `firstpix` to `firstpix + npix - 1`. Missing pixels or time samples are filled with `fmissval`.
npix	I8B	IN	number of pixels or samples to be read. See Note below.
ntods	I4B	IN	number of columns to read
header(LEN=80)(1:)	CHR	OUT	FITS extension header
firstpix	I8B	IN	first pixel (or time sample) to read from (0 based). (**default:** 0). See Note below.
fmissval	SP/ DP	IN	value to be given to missing pixels, its default value is 0. Should be of the same type as `tod`.

Note : Indices and number of data elements larger than 2^{31} are only accessible in FITS files on computers with 64 bit enabled compilers and with some specific compilation options of cfitsio (see cfitsio documentation).

MODULES & ROUTINES

This section lists the modules and routines used by **input_tod***.

fitstools	module, containing:
printerror	routine for printing FITS error messages.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

anafast	executable that reads a **HEALPix** map and analyses it.
synfast	executable that generate full sky **HEALPix** maps
getsize_fits	subroutine to know the size of a FITS file.
write_bintabh	subroutine to write large arrays into FITS files
output_map	subroutine to write a FITS file from a **HEALPix** map
input_map	subroutine to read a **HEALPix** map (either full sky of cut sky) from a FITS file

long_count, long_size

Location in HEALPix directory tree: src/f90/mod/long_intrinsic.F90

The Fortran90 module `long_intrinsic` contains a subset of intrinsic functions (currently `count` and `size`) compiled so that they return I8B variables instead of the default integer (generally I4B), therefore allowing the handling of arrays with more than $2^{31} - 1$ elements.

FUNCTIONS:

```
cnt = long_count(mask1)
```

returns the I8B integer value that is the number of elements of the logical array `mask1` that have the value `true`.

```
sz = long_size(array1 [,dim])
sz = long_size(array2 [,dim])
```

returns the I8B integer value that is the size of the 1D array `array1` or 2D array `array2` or their extent along the dimension `dim` if the scalar integer `dim` is provided.

ARGUMENTS

name & dimensionality	kind	in/out	description
cnt	I8B	OUT	number of elements with value `true`
sz	I8B	OUT	size or extent of array
mask1(:)	LGT	IN	1D logical array
array1(:)	I4B/ I8B/ SP/ DP	IN	1D integer or real array
array2(:,:)	I4B/ I8B/ SP/ DP	IN	2D integer or real array
dim (OPTIONAL)	I4B	IN	dimension (starting at 1) along which the array extent is measured.

EXAMPLE:

```
use healpix_modules
real(SP), dimension(:,:), allocatable ::  bigarray
allocate(bigarray(2_i8b**31+5, 3))
print*, size(bigarray), size(bigarray,1), size(bigarray,dim=2)
print*, long_size(bigarray), long_size(bigarray,1),
long_size(bigarray,dim=2)
deallocate(bigarray)
```

Will return (with default compilation options)

-2147483633 -2147483643 3
6442450959 2147483653 3

meaning that `long_size` handles correctly this large array while by default `size` does not.

map2alm*

Location in HEALPix directory tree: src/f90/mod/alm_tools.F90

This routine is a wrapper to 5 internal routines:map2alm_sc, map2alm_sc_pre, map2alm_pol, map2alm_pol_pre1, map2alm_pol_pre2. These routines analyse a **HEALPix** *RING ordered* map and return $a^T_{\ell m}$ (and if specified $a^E_{\ell m}$ and $a^B_{\ell m}$) values up to the desired order in ℓ (maximum $3^*N_{\rm side}$). The different routines are called depending on what parameters are passed. Some routines analyse with or without precomputed harmonics and some with or without polarisation.

FORMAT call map2alm*(nsmax, nlmax, nmmax, map_TQU, alm_TGC[, zbounds=, w8ring_TQU=, plm=])

ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	the $N_{\rm side}$ value of the map to analyse.
nlmax	I4B	IN	the maximum ℓ value for the analysis.
nmmax	I4B	IN	the maximum m value for the analysis.
map_TQU(0:12*nsmax**2-1)	SP/ DP	IN	if only the temperature map is to be analyse, the map-array should be passed with this rank.
map_TQU(0:12*nsmax**2-1, 1:3)	SP/ DP	IN	if both temperature an polarisation maps are to be analysed, the map array should have this rank, where the second index is (1,2,3) corresponding to (T,Q,U).

alm_TGC(1:p, 0:nlmax, 0:nmmax)	SPC/ DPC	OUT	The $a_{\ell m}$ values output from the analysis. p is 1 or 3 dependent on wether polarisation is included or not. In the former case, the first index is (1,2,3) corresponding to (T,E,B).
zbounds(1:2), OPTIONAL	DP	IN	section of the map on which to perform the $a_{\ell m}$ analysis, expressed in terms of $z = \sin(\text{latitude}) = \cos(\theta)$. If zbounds(1)<zbounds(2), it is performed *on* the strip zbounds(1)< z <zbounds(2); if not, it is performed *outside* the strip zbounds(2)≤ z ≤zbounds(1). If absent, the whole map is processed.
w8ring_TQU(1:2*nsmax, 1:p), OPTIONAL	DP	IN	ring weights for quadrature corrections. If ring weights are not used, this array should be 1 everywhere. p is 1 for a temperature analysis and 3 for (T,Q,U).
plm(0:(nlmax+1)*(nlmax+2)*nsmax-1), OPTIONAL	DP	IN	If this optional matrix is passed with this rank, precomputed $P_{\ell m}(\theta)$ are used instead of recursion. Note that since version 2.20 this feature has become obsolete because of algorithm optimizations.
plm(0:(nlmax+1)*(nlmax+2)*nsmax-1, 1:3), OPTIONAL	DP	IN	If this optional matrix is passed with this rank, precomputed $P_{\ell m}(\theta)$ AND precomputed tensor harmonics are used instead of recursion.

EXAMPLE:

```
use healpix_modules
integer(i4b) ::  nside, lmax
real(dp), allocatable, dimension(:,:)  ::  dw8
real(dp), dimension(2) ::  z
real(sp), allocatable, dimension(:,:)  ::  map
complex(spc), allocatable, dimension(:,:,:)  ::  alm
nside = 256
lmax = 512
allocate(dw8(1:2*nside, 1:3))
```

```
allocate(map(0:nside2npix(nside)-1,1:3))
allocate(alm(1:3, 0:lmax, 0:lmax)
dw8 = 1.0_dp
z = sin(10.0_dp * DEG2RAD)
call map2alm(nside, lmax, lmax, map, alm, (/ z, -z /) , dw8)
```

Analyses temperature and polarisation maps passed in map. The map has an $N_{\rm side}$ of 256, and the analysis is performed up to 512 in ℓ and m. The resulting $a_{\ell m}$ coefficients for temperature and polarisation are returned in alm. A 10°cut on each side of the equator is applied. Uniform weights are used.

MODULES & ROUTINES

This section lists the modules and routines used by **map2alm***.

ring_analysis	Performs FFT for the ring analysis.
misc_util	module, containing:
assert_alloc	routine to print error message when an array is not properly allocated

Note: Starting with version 3.10, `libsharp` routines will be called when precomputed $P_{\ell m}$ are not provided.

RELATED ROUTINES

This section lists the routines related to **map2alm***.

anafast	executable using map2alm* to analyse maps.
alm2map	routine performing the inverse transform of map2alm*.
dump_alms	write $a_{\ell m}$ coefficients computed by map2alm* into a FITS file
map2alm_iterative	similar to map2alm* with iterative scheme.

map2alm_iterative

Location in HEALPix directory tree: src/f90/mod/alm_tools.F90

This routine covers and extends the functionalities of map2alm: it analyzes a (polarised) **HEALPix** *RING ordered* map and returns its $a_{\ell m}$ coefficients for temperature (and polarisation) up to a specified multipole, and use precomputed harmonics if those are provided, but it also can also perform an iterative (Jacobi) determination of the $a_{\ell m}$, and apply a pixel mask if one is provided.

Denoting $\mathbf{A}$ and $\mathbf{S}$ the analysis (map2alm) and synthesis (alm2map) operators and $\mathbf{a}, \mathbf{m}$ and $\mathbf{w}$, the $a_{\ell m}$, map and pixel mask vectors, the Jacobi iterative process reads

$$\mathbf{a}^{(n)} = \mathbf{a}^{(n-1)} + \mathbf{A}.\left(\mathbf{w}.\mathbf{m} - \mathbf{S}.\mathbf{a}^{(n-1)}\right), \tag{10}$$

with

$$\mathbf{a}^{(0)} = \mathbf{A}.\mathbf{w}.\mathbf{m}. \tag{11}$$

During the processing, the standard deviation of the input map ($\mathbf{w}.\mathbf{m}$) and the current residual map $\left(\mathbf{w}.\mathbf{m} - \mathbf{S}.\mathbf{a}^{(n-1)}\right)$ is printed out, with the latter expected to get smaller and smaller as n increases.

The standard deviation of map x has the usual definition $\sigma \equiv \sqrt{\sum_{p=1}^{N} \frac{(x(p)-\bar{x})^2}{N-1}}$, where the mean is $\bar{x} \equiv \sum_{p=1}^{N} \frac{x(p)}{N}$, and the index p runs over all pixels.

In version 3.50 a bug affecting previous versions of map2alm_iterative has been fixed. (It occured when `iter_order`> 0 was used in conjonction with a `mask` and/or a restrictive `zbounds`, with a magnitude that depended on each of those factors and was larger for non-boolean masks (ie, $\mathbf{w}^2 \neq \mathbf{w}$). To assess the impact of this bug on previous results, the old implementation remains available in `map2alm_iterative_old`). The result was correct when the mask (if any) was applied to the map prior to the map2alm_iterative calling, or when no iteration was requested.

FORMAT call map2alm_iterative(nsmax, nlmax, nmmax, iter_order, map_TQU, alm_TGC[, zbounds, w8ring_TQU, plm, mask])

ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	the $N_{\rm side}$ value of the map to analyse.
nlmax	I4B	IN	the maximum ℓ value ($\ell_{\rm max}$) for the analysis.
nmmax	I4B	IN	the maximum m value for the analysis.
iter_order	I4B	IN	the order of Jacobi iteration. Increasing that order improves the accuracy of the final $a_{\ell m}$ but increases the computation time $T_{\rm CPU} \propto 1 + 2\times$iter_order. iter_order $= 0$ is a straight analysis, while iter_order $= 3$ is usually a good compromise.
map_TQU(0:12*nsmax**2-1, 1:p)	SP/ DP	INOUT	input map. p is 1 or 3 depending if temperature (T) only or temperature and polarisation (T, Q, U) are to be analysed. It will be altered on output if a mask is provided and/or if iter_order> 0 and zbounds is provided.
alm_TGC(1:p, 0:nlmax, 0:nmmax)	SPC/ DPC	OUT	The $a_{\ell m}$ values output from the analysis. p is 1 or 3 depending on whether polarisation is included or not. In the former case, the first index is (1,2,3) corresponding to (T,E,B).
zbounds(1:2), OPTIONAL	DP	IN	section of the map on which to perform the $a_{\ell m}$ analysis, expressed in terms of $z = \sin(\text{latitude}) = \cos(\theta)$. If zbounds(1)<zbounds(2), it is performed *on* the strip zbounds(1)< z <zbounds(2); if not, it is performed *outside* the strip zbounds(2)≤ z ≤zbounds(1). If absent, the whole map is processed.

w8ring_TQU(1:2*nsmax,1:p), OPTIONAL	DP	IN	ring weights for quadrature corrections. p is 1 for a temperature analysis and 3 for (T,Q,U). If absent, the ring weights are all set to 1.
plm(0:,1:p), OPTIONAL	DP	IN	If this optional matrix is passed, precomputed scalar (and tensor) $P_{\ell m}(\theta)$ are used instead of recursion.
mask(0:12*nsmax**2-1,1:q), OPTIONAL	SP/ DP	IN	pixel mask, assumed to have the same resolution (and RING ordering) as the map. The map `map_TQU` is multiplied by that mask before being analyzed, and will therefore be altered on output. q should be in $\{1, 2, 3\}$. If $p = q = 3$, then each of the 3 masks is applied to the respective map. If $p = 3$ and $q = 2$, the first mask is applied to the first map, and the second mask to the second (Q) and third (U) map. If $p = 3$ and $q = 1$, the same mask is applied to the 3 maps. Note: the output $a_{\ell m}$ are computed directly on the masked map, and are *not* corrected for the loss of power, correlation or leakage created by the mask.

EXAMPLE:

```
use healpix_types
use alm_tools
use pix_tools
integer(i4b) ::  nside, lmax, npix, iter
real(sp), allocatable, dimension(:,:)  ::  map
real(sp), allocatable, dimension(:)  ::  mask
complex(spc), allocatable, dimension(:,:,:)  ::  alm

nside = 256
lmax = 512
iter = 2
npix = nside2npix(nside)
allocate(map(0:npix-1,1:3))
allocate(mask(0:npix-1))
```

```
mask(0:)  = 0.  !  set unvalid pixels to 0
mask(0:10000-1) = 1.  !  valid pixels
allocate(alm(1:3, 0:lmax, 0:lmax)
call map2alm_iterative(nside, lmax, lmax, iter, map, alm, mask=mask)
```

Analyses temperature and polarisation signals in the first 10000 pixels of `map` (as determined by `mask`). The map has an $N_{\rm side}$ of 256, and the analysis is supposed to be performed up to 512 in ℓ and m. The resulting $a_{\ell m}$ coefficients for temperature and polarisation are returned in `alm`. Uniform weights are assumed. In order to improve the $a_{\ell m}$ accuracy, 2 Jacobi iterations are performed.

MODULES & ROUTINES

This section lists the modules and routines used by **map2alm_iterative**.

map2alm	Performs the alm analysis
alm2map	Performs the map synthesis
misc_util	module, containing:
assert_alloc	routine to print error message when an array is not properly allocated

RELATED ROUTINES

This section lists the routines related to **map2alm_iterative**.

anafast	executable using map2alm_iterative to analyse maps.
alm2map	routine performing the inverse transform of map2alm_iterative.
alm2map_spin	synthesize spin weighted maps.
dump_alms	write $a_{\ell m}$ coefficients computed by map2alm_iterative into a FITS file
map2alm_spin	analyze spin weighted maps.

map2alm_spin*

Location in HEALPix directory tree: src/f90/mod/alm_tools.F90

This routine extracts the alm coefficients out of maps of spin s and $-s$. A (complex) map S of spin s is a linear combination of the spin weighted harmonics ${}_sY_{\ell m}$

$$ {}_sS(p) = \sum_{\ell m} {}_sa_{\ell m} \;\; {}_sY_{\ell m}(p) \tag{12}$$

for $\ell \geq |m|, \ell \geq |s|$, and is such that ${}_sS^* = {}_{-s}S$.
The usual phase convention for the spin weighted harmonics is ${}_sY^*_{\ell m} = (-1)^{s+m}{}_{-s}Y_{\ell -m}$ and therefore ${}_sa^*_{\ell m} = (-1)^{s+m}{}_{-s}a_{\ell -m}$. The two (real) input maps for map2alm_spin* are defined respectively as

$$ {}_{|s|}S^+ = ({}_{|s|}S + {}_{-|s|}S)/2, \tag{13}$$
$$ {}_{|s|}S^- = ({}_{|s|}S - {}_{-|s|}S)/(2i). \tag{14}$$

map2alm_spin* outputs the alm coefficients defined as

$$ {}_{|s|}a^+_{\ell m} = -({}_{|s|}a_{\ell m} + (-1)^s {}_{-|s|}a_{\ell m})/2 \tag{15}$$
$$ {}_{|s|}a^-_{\ell m} = -({}_{|s|}a_{\ell m} - (-1)^s {}_{-|s|}a_{\ell m})/(2i), \tag{16}$$

for $m \geq 0$, knowing that, just as for spin 0 maps, the coefficients for $m < 0$ are given by

$$ {}_{|s|}a^+_{\ell -m} = (-1)^m {}_{|s|}a^{+*}_{\ell m}, \tag{17}$$
$$ {}_{|s|}a^-_{\ell -m} = (-1)^m {}_{|s|}a^{-*}_{\ell m}. \tag{18}$$

With these definitions, ${}_2a^+$, ${}_2a^-$, ${}_2S^+$ and ${}_2S^-$ match **HEALPix** polarization a^E, a^B, Q and U respectively. **However**, for $s = 0$, ${}_0a^+_{\ell m} = -a^T_{\ell m}$, ${}_0a^-_{\ell m} = 0$, ${}_0S^+ = T$, ${}_0S^- = 0$.

When dealing only with scalar quantities, like temperature or intensity maps, having a spin $s = 0$, it is highly recommended, and much more memory-efficient, to use directly the routine map2alm, rather then setting `spin`= 0 in map2alm_spin*.

FORMAT call map2alm_spin*(nsmax, nlmax, nmmax, spin, map, alm[, zbounds=, w8ring_TQU=])

ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	the $N_{\rm side}$ value of the map to analyse.
nlmax	I4B	IN	the maximum ℓ value for the analysis.
nmmax	I4B	IN	the maximum m value for the analysis.
spin	I4B	IN	the spin s of the maps to be analysed (only its absolute value is relevant).
map(0:12*nsmax**2-1, 1:2)	SP/ DP	IN	${}_{\|s\|}S^{+}$ and ${}_{\|s\|}S^{-}$ input maps
alm(1:2, 0:nlmax, 0:nmmax)	SPC/ DPC	OUT	The ${}_{\|s\|}a^{+}_{\ell m}$ and ${}_{\|s\|}a^{-}_{\ell m}$ output values.
zbounds(1:2), OPTIONAL	DP	IN	section of the map on which to perform the $a_{\ell m}$ analysis, expressed in terms of $z = \sin(\text{latitude}) = \cos(\theta)$. If zbounds(1)<zbounds(2), it is performed *on* the strip zbounds(1)< z <zbounds(2); if not, it is performed *outside* the strip zbounds(2)≤ z ≤zbounds(1). If absent, the whole map is processed.
w8ring_TQU(1:2*nsmax,1:2), OPTIONAL	DP	IN	ring weights for quadrature corrections. If ring weights are not used, this array should be 1 everywhere.

EXAMPLE:

```
use healpix_types
use alm_tools
use pix_tools
integer(i4b) ::  nside, lmax, spin
real(sp), allocatable, dimension(:,:)  ::  map
complex(spc), allocatable, dimension(:,:,:)  ::  alm

nside = 256
lmax = 512
```

```
spin = 5
allocate(map(0:nside2npix(nside)-1,1:2))
allocate(alm(1:2, 0:lmax, 0:lmax)
...
call map2alm_spin(nside, lmax, lmax, spin, map, alm)
```

Analyses spin 5 and -5 maps. The maps have an $N_{\rm side}$ of 256, and the analysis is performed up to 512 in ℓ and m. The resulting $a_{\ell m}$ coefficients for are returned in alm.

MODULES & ROUTINES

This section lists the modules and routines used by **map2alm_spin***.

ring_analysis	Performs FFT for the ring analysis.
compute_lam_mm, get_pixel_layout, gen_lamfac_der, gen_mfac, gen_recfac, init_rescale, l_min_ylm	Ancillary routines used for ${}_sY_{\ell m}$ recursion
misc_util	module, containing:
assert_alloc	routine to print error message when an array is not properly allocated

Note: Starting with version 3.80, some `libsharp` routines will be called for any $|s|$ value.

RELATED ROUTINES

This section lists the routines related to **map2alm_spin***.

alm2map_spin	routine performing the inverse transform of map2alm_spin*.
map2alm	routine analyzing temperature and polarization maps

maskborder_nest

Location in HEALPix directory tree: src/f90/mod/mask_tools.F90

For a input binary mask in NESTED ordering, maskborder_nest identifies the pixels located on the inner boundary of *invalid* regions

FORMAT

call maskborder_nest(nside, mask_in, mask_out, nbordpix, *[border_value]*)

Arguments appearing in *italic* are optional.

ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	The $N_{\rm side}$ value of the input mask.
mask_in(0:Npix-1)	I4B	IN	Input binary NESTED-ordered mask. Npix = 12*nside*nside
mask_out(0:Npix-1)	I4B	OUT	Output NESTED-ordered mask, in which inner border pixels (defined as 0-valued and adjacent to 1-valued pixels) take the value 2 (or `border_value`). Can be the same array as mask_in.
nbordpix	I4B	OUT	Number of border pixels found
border_value	I4B	IN	value to be given to border pixels in output mask. (**default:** 2).

EXAMPLE:

```
use healpix_types
use healpix_modules
...
call maskborder_nest(nside, mask_in, mask_in, nbordpix)
```

For a binary input mask `mask_in`, it will look for border pixels and output their number in `nborpix`. In this example the `mask_in` will be modified so that border pixels take value 2 on output.

MODULES & ROUTINES

This section lists the modules and routines used by **maskborder_nest**.

mask_tools	mask processing module (see related routines below)

RELATED ROUTINES

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

dist2holes_nest	angular distance to closest invalid pixel of the given mask
fill_holes_nest	turn to *valid* all pixels located in 'holes' containing fewer pixels than the given threshold
maskborder_nest	identify inner boundary pixels of 'holes' for given mask
size_holes_nest	returns size (in pixels) of holes found in input mask

medfiltmap*

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

This routine performs the median filtering of a **HEALPix** full sky map for a given neighborhood radius

FORMAT

call medfiltmap*(in_map, radius, med_map [, nest, fmissval, fill_holes])

ARGUMENTS

name & dimensionality	kind	in/out	description
in_map(0:npix-1)	SP/ DP	IN	Full sky **HEALPix** map to filter. `npix` should be valid **HEALPix** pixel number.
radius	DP	IN	Radius in RADIANS of the disk on which the median is computed.
med_map(0:npix-1)	SP/ DP	OUT	Median filtered map: each pixel is the median of the input map valid neighboring pixels contained within a distance `radius`
nest OPTIONAL	I4B	IN	set to 1 if the map ordering is NESTED, set to 0 if it is RING.
fmissval OPTIONAL	SP/ DP	IN	sentinel value given to missing or non-valid pixels. Default: `HPX_SBADVAL` or `HPX_DBADVAL` $= -1.6375\ 10^{30}$
fill_holes OPTIONAL	LGT	IN	if set to `.true.` will replace non-valid pixels by median of neighbors; if set to `.false.` will leave non-valid pixels unchanged. Default: `.false.`

EXAMPLE:

```
use healpix_types
use pix_tools
...
call medfiltmap(map, 0.5*DEG2RAD, med)
```

Output in `med` the median filter of `map`, using a filter radius of 0.5 Deg

MODULES & ROUTINES

This section lists the modules and routines used by **medfiltmap***.

statistics	module, containing:
median	routine to compute the median of a data set
pix_tools	module, containing:
pix2vec_ring, pix2vec_nest	routines to find the location of a pixel on the sky
query_disc	routine to find pixels lying within a radius of a given point

median*

Location in HEALPix directory tree: src/f90/mod/statistics.f90

This function computes the median of a data set

FORMAT

var=median*(data[, badval, even])

ARGUMENTS

name & dimensionality	kind	in/out	description
var	SP/ DP	OUT	median of the data set, defined as the middle number (or the average of the 2 middle numbers) once the valid data points are sorted in monotonous order
data(:)	SP/ DP	IN	data set
badval (OPTIONAL)	SP/ DP	IN	sentinel value given to bad data points. Data points with this value will be ignored during calculation of the median. If not set, all points will be considered. **Do not set to 0!**.
even (OPTIONAL)	LGT	IN	if set to `.true.` and the number of valid data points is even, will output the average of the 2 middle points (which doubles the calculation time). If the number of points is odd, the single middle point is output and this keyword is ignored.

EXAMPLE:

```
use statistics, only:  median
...
med = median(map, even=.true.)
```

Outputs in `med` the median of `map`

MODULES & ROUTINES

This section lists the modules and routines used by **median***.

m_indmed	module of the Orderpack 2.0 package, written by: Michel Olagnon, http://www.fortran-2000.com/rank/, also available in src/f90/mod/indmed.f90
indmed	routine of `m_indmed` to output rank of median

RELATED ROUTINES

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

compute_statistics	routine min, max, absolute deviation, and first four order moments of a data set

merge_headers

Location in HEALPix directory tree: src/f90/mod/head_fits.F90

This routine merges two FITS headers.

FORMAT

call merge_headers(header1, header2)

ARGUMENTS

name&dimensionality	kind	in/out	description
header1(LEN=80) DIMENSION(:)	CHR	IN	First header.
header2(LEN=80) DIMENSION(:)	CHR	INOUT	Second header. On output, will contain the concatenation of (in that order) header2 and header1. If header2 is too short to allow the merging the output will be truncated

EXAMPLE:

```
call merge_headers(header1, header2)
```

On output header2 will contain the original header2, followed by the content of header1

MODULES & ROUTINES

This section lists the modules and routines used by **merge_headers**.

write_hl	more general routine for adding a keyword to a header.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

add_card	general purpose routine to write any keywords into a FITS file header
get_card	general purpose routine to read any keywords from a header in a FITS file.
del_card	routine to discard a keyword from a FITS header
read_par, number_of_alms	routines to read specific keywords from a header in a FITS file.
getsize_fits	function returning the size of the data set in a fits file and reading some other useful FITS keywords

mpi_alm_tools*

Location in HEALPix directory tree: src/f90/mod/mpi_alm_tools.f90

This module implements MPI parallelization of the alm2map and map2alm routines. It is not compiled by default during installation, but rather intended for users who need massive parallelization in their own programming. Typical applications are Monte Carlo simulations and Markov chain type analyses.

The routines can be called in two modes, either simple or advanced. The former mimics the interface of the standard routines, but with an additional MPI handle as a first argument, and is intended for applications which requires only one or a few transforms. The latter interface provides both more flexibility (in particular the option of pre-computation of the Legendre polynomials) and a simpler interface when multiple transforms are required. This interface is particularly well suited for Monte Carlo simulations and Markov chain type analyses.

EXAMPLE:

- Simple one-line interfaces:
  - mpi_map2alm_simple
  - mpi_alm2map_simple
- Three-step advanced interfaces:
  1. Initialization:
     mpi_initialize_alm_tools
  2. Execution of spherical harmonics transforms
     - mpi_map2alm (root processor)
     - mpi_alm2map (root processor)
     - mpi_map2alm_slave (slave processor)
     - mpi_alm2map_slave (slave processor)
  3. Finalizing:
     mpi_cleanup_alm_tools

mpi_alm2map*

Location in HEALPix directory tree: src/f90/mod/mpi_alm_tools.f90

This subroutine implements MPI parallelization of the serial alm2map routine. It supports both temperature and polarization inputs in both single and double precision. It must only be run by the root node of the MPI communicator.

FORMAT

call mpi_alm2map*(alms, map)

ARGUMENTS

name & dimensionality	kind	in/out	description
alms(1:nmaps,0:lmax,0:nmax)	SPC or DPC	IN	Input alms. If nmaps=1, only temperature information is included; if nmaps=3, polarization information is included
map(0:npix,1:nmaps)	SP or DP	OUT	Output map. nmaps must match that of the input alms array.

EXAMPLE:

```
call mpi_comm_rank(comm, myid, ierr)
if (myid == root) then
     call mpi_initialize_alm_tools(comm, nsmax, nlmax, nmmax,
               zbounds,polarization, precompute_plms)
     call mpi_alm2map(alms, map)
else
     call mpi_initialize_alm_tools(comm)
     call mpi_alm2map_slave
end
call mpi_cleanup_alm_tools
```

This example 1) initializes the mpi_alm_tools module (i.e., allocates internal arrays and defines required parameters), 2) executes a parallel alm2map operation, and 3) frees the previously allocated memory.

MODULES & ROUTINES

This section lists the modules and routines used by **mpi_alm2map***.

alm_tools	module

RELATED ROUTINES

This section lists the routines related to **mpi_alm2map***.

mpi_cleanup_alm_tools	Frees memory that is allocated by the current routine.
mpi_initialize_alm_tools	Allocates memory and defines variables for the mpi_alm_tools module.
mpi_alm2map_slave	Routine for executing a parallel inverse spherical harmonics transform (slave processor interface)
mpi_map2alm	Routine for executing a parallel spherical harmonics transform (root processor interface)
mpi_map2alm_slave	Routine for executing a parallel spherical harmonics transform (slave processor interface)
mpi_alm2map_simple	One-line interface to the parallel inverse spherical harmonics transform
mpi_map2alm_simple	One-line interface to the parallel spherical harmonics transform

mpi_alm2map_simple*

Location in HEALPix directory tree: src/f90/mod/mpi_alm_tools.f90

This subroutine provides a simplified (one-line) interface to the MPI version of alm2map. It supports both temperature and polarization inputs in both single and double precision. It must only be run by all nodes in the MPI communicator.

FORMAT

call mpi_alm2map_simple*(comm, alms, map)

ARGUMENTS

name & dimensionality	kind	in/out	description
comm	I4B	IN	MPI communicator.
alms(1:nmaps,0:lmax,0:nmax)	SPC or DPC	IN	Input alms. If nmaps=1, only temperature information is included; if nmaps=3, polarization information is included
map(0:npix,1:nmaps)	SP or DP	OUT	Output map. nmaps must match that of the input alms array.

EXAMPLE:

```
call mpi_alm2map_simple(comm, map, alms)
```

This example executes a parallel map2alm operation through the one-line interface. Although all processors must supply allocated arrays to the routine, only the root processor's information will be used as input, and only the root processor's alms will be complete after execution.

MODULES & ROUTINES

This section lists the modules and routines used by **mpi_alm2map_simple***.

alm_tools	module

RELATED ROUTINES

This section lists the routines related to **mpi_alm2map_simple***.

mpi_cleanup_alm_tools	Frees memory that is allocated by the current routine.
mpi_initialize_alm_tools	Allocates memory and defines variables for the mpi_alm_tools module.
mpi_alm2map	Routine for executing a parallel inverse spherical harmonics transform (root processor interface)
mpi_alm2map_slave	Routine for executing a parallel inverse spherical harmonics transform (slave processor interface)
mpi_map2alm	Routine for executing a parallel spherical harmonics transform (root processor interface)
mpi_map2alm_slave	Routine for executing a parallel spherical harmonics transform (slave processor interface)
mpi_map2alm_simple	One-line interface to the parallel spherical harmonics transform

mpi_alm2map_slave

Location in HEALPix directory tree: src/f90/mod/mpi_alm_tools.f90

This subroutine complements the master routine mpi_alm2map, and should be run by all slaves in the current MPI communicator. It is run without arguments, but after an appropriate call to initialize_mpi_alm_tools.

FORMAT

call mpi_alm2map_slave()

ARGUMENTS

None.

EXAMPLE:

```
call mpi_comm_rank(comm, myid, ierr)
if (myid == root) then
     call mpi_initialize_alm_tools(comm, nsmax, nlmax, nmmax,
               zbounds,polarization, precompute_plms)
     call mpi_alm2map(alms, map)
else
     call mpi_initialize_alm_tools(comm)
     call mpi_alm2map_slave
end
call mpi_cleanup_alm_tools
```

This example 1) initializes the mpi_alm_tools module (i.e., allocates internal arrays and defines required parameters), 2) executes a parallel alm2map operation, and 3) frees the previously allocated memory.

MODULES & ROUTINES

This section lists the modules and routines used by **mpi_alm2map_slave**.

alm_tools	module

RELATED ROUTINES

This section lists the routines related to **mpi_alm2map_slave**.

mpi_cleanup_alm_tools	Frees memory that is allocated by the current routine.
mpi_initialize_alm_tools	Allocates memory and defines variables for the mpi_alm_tools module.
mpi_alm2map	Routine for executing a parallel inverse spherical harmonics transform (root processor interface)
mpi_map2alm	Routine for executing a parallel spherical harmonics transform (root processor interface)
mpi_map2alm_slave	Routine for executing a parallel spherical harmonics transform (slave processor interface)
mpi_alm2map_simple	One-line interface to the parallel inverse spherical harmonics transform
mpi_map2alm_simple	One-line interface to the parallel spherical harmonics transform

mpi_cleanup_alm_tools

Location in HEALPix directory tree: src/f90/mod/mpi_alm_tools.f90

This subroutine deallocates any private arrays previously allocated in the mpi_alm_tools module. It should be run (without arguments) by all processors in the current communicator after the last call to any of the working routines.

FORMAT

call mpi_cleanup_alm_tools()

ARGUMENTS

None.

EXAMPLE:

```
call mpi_comm_rank(comm, myid, ierr)
if (myid == root) then
     call mpi_initialize_alm_tools(comm, nsmax, nlmax, nmmax,
               zbounds,polarization, precompute_plms)
     call mpi_map2alm(map, alms)
else
     call mpi_initialize_alm_tools(comm)
     call mpi_map2alm_slave
end
call mpi_cleanup_alm_tools
```

This example 1) initializes the mpi_alm_tools module (i.e., allocates internal arrays and defines required parameters), 2) executes a parallel map2alm operation, and 3) frees the previously allocated memory.

RELATED ROUTINES

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

mpi_initialize_alm_tools	Allocates memory and defines variables for the mpi_alm_tools module.
mpi_alm2map	Routine for executing a parallel inverse spherical harmonics transform (root processor interface)
mpi_alm2map_slave	Routine for executing a parallel inverse spherical harmonics transform (slave processor interface)
mpi_map2alm	Routine for executing a parallel spherical harmonics transform (root processor interface)
mpi_map2alm_slave	Routine for executing a parallel spherical harmonics transform (slave processor interface)
mpi_alm2map_simple	One-line interface to the parallel inverse spherical harmonics transform
mpi_map2alm_simple	One-line interface to the parallel spherical harmonics transform

mpi_initialize_alm_tools

Location in HEALPix directory tree: src/f90/mod/mpi_alm_tools.f90

This subroutine initializes the mpi_alm_tools module, and must be run prior to any of the advanced interface working routines by all processors in the MPI communicator. The root processor must supply all arguments, while it is optional for the slaves. However, the information is disregarded if they do.
A major advantage of MPI parallelization is large quantities of memory, allowing for pre-computation of the Legendre polynomials even with high $N_{\rm side}$ and $\ell_{\rm max}$, since each processor only needs a fraction $(1/N_{\rm procs})$ of the complete table. This feature is controlled by the "precompute_plms" parameter. In general, the CPU time can be expected to decrease by roughly 50% using pre-computed Legendre polynomials for temperature calculations, and by about 30% for polarization calculations.

FORMAT

call mpi_initialize_alm_tools(comm, [nsmax], [nlmax], [nmmax], [zbounds], [polarization], [precompute_plms], [w8ring])

ARGUMENTS

name & dimensionality	kind	in/out	description
comm	I4B	IN	MPI communicator.
nsmax	I4B	IN	the $N_{\rm side}$ value of the HEALPix map. (OPTIONAL)
nlmax	I4B	IN	the maximum ℓ value used for the $a_{\ell m}$. (OPTIONAL)
nmmax	I4B	IN	the maximum m value used for the $a_{\ell m}$. (OPTIONAL)

zbounds(1:2)	DP	IN	section of the map on which to perform the $a_{\ell m}$ analysis, expressed in terms of $z = \sin(\text{latitude}) = \cos(\theta)$. If zbounds(1)<zbounds(2), it is performed *on* the strip zbounds(1)< z <zbounds(2); if not, it is performed *outside* the strip zbounds(2)≤ z ≤zbounds(1). If absent, the whole map is processed. (OPTIONAL)
polarization	LGT	IN	if polarization is required, this should be set to true, else it should be set to false. (OPTIONAL)
precompute_plms	I4B	IN	0 = do not pre-compute any $P_{\ell m}$'s; 1 = pre-compute $P^{\rm T}_{\ell m}$; 2 = pre-compute $P^{\rm T}_{\ell m}$ and $P^{\rm P}_{\ell m}$. (OPTIONAL)
w8ring_TQU(1:2*nsmax, 1:p)	DP	IN	ring weights for quadrature corrections. If ring weights are not used, this array should be 1 everywhere. p is 1 for a temperature analysis and 3 for (T,Q,U). (OPTIONAL)

EXAMPLE:

```
call mpi_comm_rank(comm, myid, ierr)
if (myid == root) then
     call mpi_initialize_alm_tools(comm, nsmax, nlmax, nmmax,
               zbounds,polarization, precompute_plms)
     call mpi_map2alm(map, alms)
else
     call mpi_initialize_alm_tools(comm)
     call mpi_map2alm_slave
end
call mpi_cleanup_alm_tools
```

This example 1) initializes the mpi_alm_tools module (i.e., allocates internal arrays and defines required parameters), 2) executes a parallel map2alm operation, and 3) frees the previously allocated memory.

RELATED ROUTINES

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

mpi_cleanup_alm_tools	Frees memory that is allocated by the current routine.
mpi_alm2map	Routine for executing a parallel inverse spherical harmonics transform (root processor interface)
mpi_alm2map_slave	Routine for executing a parallel inverse spherical harmonics transform (slave processor interface)
mpi_map2alm	Routine for executing a parallel spherical harmonics transform (root processor interface)
mpi_map2alm_slave	Routine for executing a parallel spherical harmonics transform (slave processor interface)
mpi_alm2map_simple	One-line interface to the parallel inverse spherical harmonics transform
mpi_map2alm_simple	One-line interface to the parallel spherical harmonics transform

mpi_map2alm*

Location in HEALPix directory tree: src/f90/mod/mpi_alm_tools.f90

This subroutine implements MPI parallelization of the serial map2alm routine. It supports both temperature and polarization inputs in both single and double precision. It must only be run by the root node of the MPI communicator.

FORMAT

call mpi_map2alm*(map, alms)

ARGUMENTS

name & dimensionality	kind	in/out	description
map(0:npix,1:nmaps)	SP or DP	IN	map to analyse. If nmaps=1, only temperature information is included; if nmaps=3, polarization information is included
alms(1:nmaps,0:lmax,0:nmax)	SPC or DPC	OUT	output alms. nmaps must equal that of the input map

EXAMPLE:

```
call mpi_comm_rank(comm, myid, ierr)
if (myid == root) then
     call mpi_initialize_alm_tools(comm, nsmax, nlmax, nmmax,
               zbounds,polarization, precompute_plms)
     call mpi_map2alm(map, alms)
else
     call mpi_initialize_alm_tools(comm)
     call mpi_map2alm_slave
end
call mpi_cleanup_alm_tools
```

This example 1) initializes the mpi_alm_tools module (i.e., allocates internal arrays and defines required parameters), 2) executes a parallel map2alm operation, and 3) frees the previously allocated memory.

MODULES & ROUTINES

This section lists the modules and routines used by **mpi_map2alm***.

alm_tools	module

RELATED ROUTINES

This section lists the routines related to **mpi_map2alm***.

mpi_cleanup_alm_tools	Frees memory that is allocated by the current routine.
mpi_initialize_alm_tools	Allocates memory and defines variables for the mpi_alm_tools module.
mpi_alm2map	Routine for executing a parallel inverse spherical harmonics transform (root processor interface)
mpi_alm2map_slave	Routine for executing a parallel inverse spherical harmonics transform (slave processor interface)
mpi_map2alm_slave	Routine for executing a parallel spherical harmonics transform (slave processor interface)
mpi_alm2map_simple	One-line interface to the parallel inverse spherical harmonics transform
mpi_map2alm_simple	One-line interface to the parallel spherical harmonics transform

mpi_map2alm_simple*

Location in HEALPix directory tree: **src/f90/mod/mpi_alm_tools.f90**

This subroutine provides a simplified (one-line) interface to the MPI version of map2alm. It supports both temperature and polarization inputs in both single and double precision. It must only be run by all processors in the MPI communicator.

FORMAT

call mpi_map2alm_simple*(comm, map, alms, [zbounds], [w8ring])

ARGUMENTS

name & dimensionality	kind	in/out	description
comm	I4B	IN	MPI communicator.
map(0:npix-1,1:nmaps)	SP or DP	IN	input map. If nmaps=1, only temperature information is included; if nmaps=3, polarization information is included
alms(1:nmaps,0:lmax,0:nmax)	SPC or DPC	IN	output alms. nmaps must equal that of the input map
zbounds(1:2)	DP	IN	section of the map on which to perform the a_{lm} analysis, expressed in terms of $z = \sin(\text{latitude}) = \cos(\theta)$. If zbounds(1)<zbounds(2), it is performed *on* the strip zbounds(1)< z <zbounds(2); if not, it is performed *outside* the strip zbounds(2)≤ z ≤zbounds(1). If absent, the whole map is processed. (OPTIONAL)

w8ring_TQU(1:2*nsmax, 1:p)	DP	IN	ring weights for quadrature corrections. If ring weights are not used, this array should be 1 everywhere. p is 1 for a temperature analysis and 3 for (T,Q,U). (OPTIONAL)

EXAMPLE:

```
call mpi_map2alm_simple(comm, map, alms)
```

This example executes a parallel map2alm operation through the one-line interface. Although all processors must supply allocated arrays to the routine, only the root processor's information will be used as input, and only the root processor's alms will be complete after execution.

MODULES & ROUTINES

This section lists the modules and routines used by **mpi_map2alm_simple***.

alm_tools	module

RELATED ROUTINES

This section lists the routines related to **mpi_map2alm_simple***.

mpi_cleanup_alm_tools	Frees memory that is allocated by the current routine.
mpi_initialize_alm_tools	Allocates memory and defines variables for the mpi_alm_tools module.
mpi_alm2map	Routine for executing a parallel inverse spherical harmonics transform (root processor interface)
mpi_alm2map_slave	Routine for executing a parallel inverse spherical harmonics transform (slave processor interface)
mpi_map2alm	Routine for executing a parallel spherical harmonics transform (root processor interface)
mpi_map2alm_slave	Routine for executing a parallel spherical harmonics transform (slave processor interface)

mpi_alm2map_simple	One-line interface to the parallel inverse spherical harmonics transform

mpi_map2alm_slave

Location in HEALPix directory tree: src/f90/mod/mpi_alm_tools.f90

This subroutine complements the master routine mpi_map2alm, and should be run by all slaves in the current MPI communicator. It is run without arguments, but after an appropriate call to initialize_mpi_alm_tools.

FORMAT

call mpi_map2alm_slave()

ARGUMENTS

None.

EXAMPLE:

```
call mpi_comm_rank(comm, myid, ierr)
if (myid == root) then
     call mpi_initialize_alm_tools(comm, nsmax, nlmax, nmmax,
               zbounds,polarization, precompute_plms)
     call mpi_map2alm(map, alms)
else
     call mpi_initialize_alm_tools(comm)
     call mpi_map2alm_slave
end
call mpi_cleanup_alm_tools
```

This example 1) initializes the mpi_alm_tools module (i.e., allocates internal arrays and defines required parameters), 2) executes a parallel map2alm operation, and 3) frees the previously allocated memory.

MODULES & ROUTINES

This section lists the modules and routines used by **mpi_map2alm_slave**.

alm_tools	module

RELATED ROUTINES

This section lists the routines related to **mpi_map2alm_slave**.

mpi_cleanup_alm_tools	Frees memory that is allocated by the current routine.
mpi_initialize_alm_tools	Allocates memory and defines variables for the mpi_alm_tools module.
mpi_alm2map	Routine for executing a parallel inverse spherical harmonics transform (root processor interface)
mpi_alm2map_slave	Routine for executing a parallel inverse spherical harmonics transform (slave processor interface)
mpi_map2alm	Routine for executing a parallel spherical harmonics transform (root processor interface)
mpi_alm2map_simple	One-line interface to the parallel inverse spherical harmonics transform
mpi_map2alm_simple	One-line interface to the parallel spherical harmonics transform

nArguments

Location in HEALPix directory tree: src/f90/mod/extension.F90

This function emulates the C routine `iargc`, which returns the number of command line arguments provided.
Starting with release 3.60, it calls the F2003 extension function `command_argument_count`.

FORMAT

var=nArguments( )

ARGUMENTS

name&dimensionality	kind	in/out	description
var	I4B	OUT	number of command line arguments

RELATED ROUTINES

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

getEnvironment	returns value of environment variable
getArgument	returns list of command line arguments

neighbours_nest

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

This subroutine returns the number and locations (in terms of pixel numbers) of the topological neighbours of a central pixel. The pixels are ordered in a clockwise sense about the central pixel with the southernmost pixel in first element. For the 4 pixels in the southern corners of the equatorial faces which have two equally southern neighbours the routine returns the south-western pixel first and proceeds clockwise.

FORMAT

call neighbours_nest(nside, ipix, list, nneigh)

ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	The $N_{\rm side}$ parameter of the map.
ipix	I4B/ I8B	IN	The NESTED pixel index of the central pixel.
list(8)	I4B/ I8B	OUT	On exit, the vector of neighbouring pixels. This contains `nneigh` relevant elements.
nneigh	I4B	OUT	The number of neighbours (mostly 8, except at 8 sites, where there are only 7 neighbours).

EXAMPLE:

```
use pix_tools
integer ::  nneigh, list(1:8)
call neighbours_nest(4, 1, list, nneigh)
print*,nneigh
print*,list(1:nneigh)
```

This returns `nneigh`= 8 and a vector `list`, which contains the pixel numbers (90, 0, 2, 3, 6, 4, 94, 91).

MODULES & ROUTINES

This section lists the modules and routines used by **neighbours_nest**.

mk_xy2pix, mk_pix2xy	precomputing arrays for the conversion of NESTED pixel numbers to Cartesian coords in each face.
pix2xy_nest, xy2pix_nest	Conversion between NESTED pixel numbers to Cartesian coords in each face.
bit_manipulation	module, containing:
invMSB, invLSB,swapLSBMSB,invswapLSBMSB	functions which manipulate the bit vector which represents the NESTED pixel numbers. They correspond to NorthWest<->SouthEast, SouthWest<->NorthEast, East<->West and North-South flips of the diamond faces, respectively.

RELATED ROUTINES

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

pix2ang, ang2pix	convert between angle and pixel number.
pix2vec, vec2pix	convert between a cartesian vector and pixel number.

nest2uniq

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

This F90 facility turns the parameter $N_{\rm side}$ (a power of 2) and the pixel index p into the Unique ID number $u = p + 4N_{\rm side}^2$. See "The Unique Identifier scheme" section in "**HEALPix** Introduction Document" for more details.

FORMAT

call nest2uniq(nside, pnest, puniq)

ARGUMENTS

name	kind	in/out	description
nside	I4B	IN	The **HEALPix** $N_{\rm side}$ parameter.
pnest	I4B/I8B	IN	(NESTED scheme) pixel identification number over the range $\{0,12N_{\rm side}^2 - 1\}$.
puniq	I4B/I8B	OUT	The **HEALPix** Unique pixel identifier.

EXAMPLE:

```
use healpix_modules
integer(I4B) ::  puniq
call nest2uniq(1, 0, puniq)
print*,puniq
```

returns
4
since the first pixel ($p = 0$) at $N_{\rm side} = 1$ is the pixel with Unique ID number 4.

RELATED ROUTINES

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

uniq2nest] Transforms Unique **HEALPix** pixel ID number into Nside and Nested pixel number

pix2xxx, ...	to turn NESTED pixel index into sky coordinates and back

npix2nside

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

Function to provide the resolution parameter $N_{\rm side}$ correspoonding to $N_{\rm pix}$ pixels over the full sky.

FORMAT

var=npix2nside(npix)

ARGUMENTS

name & dimensionality	kind	in/out	description
npix	I4B/ I8B	IN	the number $N_{\rm pix}$ of pixels over the whole sky.
var	I4B	OUT	the parameter $N_{\rm side}$. If $N_{\rm pix}$ is valid (12 times a power of 2 in $\{1,\ldots,2^{28}\}$), $N_{\rm side} = \sqrt{N_{\rm pix}/12}$ is returned; if not, an error message is issued and -1 is returned.

EXAMPLE:

```
use healpix_modules
integer(I4B) ::  nside
nside= npix2nside(786432)
```

Returns the resolution parameter $N_{\rm side}$ (256) corresponding to 786432 pixels on the sky.

RELATED ROUTINES

This section lists the routines related to **npix2nside**.

nside2npix — returns the number of pixels $N_{\rm pix}$ correspondinng to resolution parameter $N_{\rm side}$

nside2npix

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

Function to provide the number of pixels $N_{\rm pix}$ over the full sky corresponding to resolution parameter $N_{\rm side}$.

FORMAT

var=nside2npix(nside)

ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\rm side}$ parameter of the map.
var	I8B	OUT	the number of pixels $N_{\rm pix}$ of the map. If $N_{\rm side}$ is valid (a power of 2 in $\{1, \ldots, 2^{28} = 268435456\}$), $N_{\rm pix} = 12N_{\rm side}^2$ is returned; if not, an error message is issued and -1 is returned.

EXAMPLE:

```
use healpix_modules
integer(I8B) ::  npix
npix= nside2npix(256)
```

Returns the number of **HEALPix** pixels (786432) for the resolution parameter 256.

RELATED ROUTINES

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

npix2nside	returns resolution parameter corresponding to the number of pixels.

nside2npweights

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

Function to return the number of pixel-based weights (in compact form) for a given Nside:

$$N_w = \frac{(N_{\rm side}+1)(3N_{\rm side}+1)}{4} \simeq \frac{N_{\rm pix}}{16}$$

FORMAT

var=nside2npweights(nside)

ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\rm side}$ parameter.
var	I8B	OUT	the number of template pixels N_w.

EXAMPLE:

```
use healpix_modules
integer(I8B) ::  nw8
nw8 = nside2npweights(256)
```

Returns in `nw8` the number of non-redundant **HEALPix** pixel-based weights (49408) for the resolution parameter 256.

RELATED ROUTINES

This section lists the routines related to **nside2npweights**.

unfold_weightsfile	reads of FITS file containing a list ring-based or pixel-based weights into a full sky map

nside2ntemplates

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

Function to provide the number of template pixels

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

corresponding to resolution parameter $N_{\rm side}$. 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

var=nside2ntemplates(nside)

ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\rm side}$ parameter.
var	I8B	OUT	the number of template pixels $N_{\rm template}$.

EXAMPLE:

```
use healpix_modules
integer(I8B) ::  ntpl
ntpl= nside2ntemplates(256)
```

Returns in `ntpl` the number of **HEALPix** template pixels (16768) for the resolution parameter 256.

RELATED ROUTINES

This section lists the routines related to **nside2ntemplates**.

template_pixel_ring

template_pixel_nest	return the template pixel associated with any **HEALPix** pixel
same_shape_pixels_ring	
same_shape_pixels_nest	return the ordered list of pixels having the same shape as a given pixel template

number_of_alms

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This function returns the number of $a_{\ell m}$ values stored in each FITS extension in a FITS file containing $a_{\ell m}$

FORMAT

var=number_of_alms(filename[, extnum])

ARGUMENTS

name & dimensionality	kind	in/out	description
filename(LEN=`filenamelen`)	CHR	IN	filename of the FITS-file containing $a_{\ell m}$.
extnum	I4B	OUT	number of extensions in the file

EXAMPLE:

```
print*,number_of_alms('alms.fits')
```

Prints the number of $a_{\ell m}$ stored in each extension of the file 'alms.fits'

MODULES & ROUTINES

This section lists the modules and routines used by **number_of_alms**.

fitstools	module, containing:
printerror	routine for printing FITS error messages.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

fits2alms, read_conbintab	routines that read $a_{\ell m}$ values from FITS files.

output_map*

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This routine writes a full sky **HEALPix** map into a FITS file. The map can be either single or double precision real. It *has* to be 2-dimensional.

FORMAT

call output_map*(map, header, filename[,*extno*])

ARGUMENTS

name & dimensionality	kind	in/out	description
map(0:,1:)	SP/ DP	IN	full sky map(s) to be output
header(LEN=80)(1:)	CHR	IN	string array containing the FITS header to be included in the file
filename(LEN=*)	CHR	IN	filename of the FITS-file to contain **HEALPix** map(s).
extno	I4B	IN	extension number in which to write the data (0 based). (**default:** 0)

EXAMPLE:

```
use healpix_types
use fits_tools, only :  output_map
real(sp), dimension(0:12*16**2-1, 1:1) ::  map
character(len=80), dimension(1:10) ::  header
header(:)  = "
map(:,:)  = 1.
call output_map(map, header, 'map.fits')
```

generates a simple map (made of 1s) and outputs it into the FITS file `map.fits`

MODULES & ROUTINES

This section lists the modules and routines used by **output_map***.

fitstools	module, containing:
printerror	routine for printing FITS error messages.
write_bintab	routine to write a binary table into a FITS file.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

anafast	executable that reads a **HEALPix** map from a FITS file and analyses it.
synfast	executable that generate full sky **HEALPix** maps
input_map	subroutine to read a **HEALPix** map from a a FITS file
write_bintabh	subroutine to write a large array into a FITS file piece by piece
input_tod*	subroutine to read an arbitrary subsection of a large binary table
write_minimal_header	routine to write minimal FITS header

parse__init, parse__int, ..., parse__finish

Location in HEALPix directory tree: src/f90/mod/paramfile_io.F90

The Fortran90 module paramfile_io contains functions to obtain parameters from parameter files or interactively

ARGUMENTS

name&dimensionality	**kind**	**in/out**	**description**
fname	CHR	IN	file containing the simulation parameters. If empty, parameters are obtained interactively.
handle	PMF	INOUT	Object of type (paramfile_handle) used to store parameter information
keyname	CHR	IN	name of the required parameter
default	XXX	IN	optional argument containing the default value for a given simulation parameter; must be of appropriate type.
vmin	XXX	IN	optional argument containing the minimum value for a given simulation parameter; must be of appropriate type.
vmax	XXX	IN	optional argument containing the maximum value for a given simulation parameter; must be of appropriate type.
descr	CHR	IN	optional argument containing a description of the required simulation parameter
filestatus	CHR	IN	optional argument. If present, the parameter must be a valid filename. If filestatus=='new', the file must not exist; if filestatus=='old', the file must exist.
code	CHR	IN	optional argument. Contains the name of the executable.
silent	LGT	IN	optional argument. If set to `.true.` the parsing routines will run silently in non-interactive mode (except for warning or error messages, which will always appear). This is mainly intended for MPI usage where many processors parse the same parameter file: `silent` can be set to `.true.` on all CPUs except one.

ROUTINES:

`handle = parse_init (fname [,silent])`

initializes the parser to work on the file fname, or interactively, if fname is empty

`intval = parse_int (handle, keyname [, default, vmin, vmax, descr])`

`longval = parse_long (handle, keyname [, default, vmin, vmax, descr])`

`realval = parse_real (handle, keyname [, default, vmin, vmax, descr])`

`doubleval = parse_double (handle, keyname [, default, vmin, vmax, descr])`

`stringval = parse_string (handle, keyname [, default, descr, filestatus])`

`logicval = parse_lgt (handle, keyname [, default, descr])`

These routines obtain integer(i4b), integer(i8b), real(sp), real(dp), character(len=*) and logical values, respectively.
Note: `parse_string` will expand all environment variables of the form ${XXX} (eg: ${HOME}). It will also replace a *leading* `~/` by the user's home directory.

`call parse_summarize (handle [, code])`

if the parameters were set interactively, this routine will print out a parameter file performing the same settings.

`call parse_check_unused (handle [, code])`

if a parameter file was read, this routine will print out all the parameters found in the file but not used by the code. Intended at detecting typos in parameter names.

`call parse_finish (handle)`

frees the memory

EXAMPLE:

```
program who_r_u
use healpix_types
use paramfile_io
use extension

implicit none
type(paramfile_handle) ::  handle
character(len=256) ::  parafile, name
real(DP) ::  age

parafile = ''
if (nArguments() == 1) call getArgument(1, parafile)
handle = parse_init(parafile)
name = parse_string(handle, 'name',descr='Enter your last name:  ')
age = parse_double(handle, 'age', descr='Enter your age in years:  ', &
```

```
& default=18.d0,vmin=0.d0)
call parse_summarize(handle, 'who_r_u')
end program who_r_u
```

If a file is provided as command line argument when running the executable `who_r_u`, that file will be parsed in search of the lines starting with 'name =' and 'age =', otherwise the same questions will be asked interactively.

RELATED ROUTINES

This section lists the routines related to **parse_init, parse_int, ..., parse_finish**.

concatnl	generates from a set of strings the multi-line description
nArguments	returns the number of command line arguments
getArgument	returns the list of command line arguments

pixel_window

Location in HEALPix directory tree: src/f90/mod/alm_tools.F90

This routine returns the *averaged* ℓ-space window function $w_{\rm pix}(\ell)$ (for temperature and polarisation) associated to **HEALPix** pixels of resolution parameter $N_{\rm side}$. Because of the integration of the signal over the pixel area, the $a_{\ell m}^{\rm (pix)}$ coefficients of a pixelated map are related to the unpixelated underlying $a_{\ell m}$ by $a_{\ell m}^{\rm (pix)} = a_{\ell m} w_{\rm pix}(\ell)$.
Unless specified otherwise, the $w_{\rm pix}(\ell)$ are read from the files $HEALPIX/data/pixel_window_n????.fits.

FORMAT

call pixel_window(pixlw, nside[, windowfile])

ARGUMENTS

name & dimensionality	kind	in/out	description
pixlw(0:lmax,1:p)	DP	OUT	pixel window function(s) $w_{\rm pix}(\ell)$ generated. The first index must be $\ell_{\rm max} \leq 4N_{\rm side}$. The second index runs from 1:1 for temperature only, and 1:3 for polarisation. In the latter case, 1=T, 2=E, 3=B.
nside	I4B	IN	**HEALPix** $N_{\rm side}$ resolution parameter. Unless `windowfile` is set, the file associated with $N_{\rm side}$ and shipped with the package is read by default. If `nside` = 0, the pixel is assumed infinitely small and `pixlw` is returned with value 1.
windowfile (OPTIONAL)	CHR	IN	FITS file containing the pixel window to be read instead of the default.

EXAMPLE:

```
call pixel_window(pixlw, 64)
```

returns in pixlw the pixel window function for $N_{\rm side} = 64$.

MODULES & ROUTINES

This section lists the modules and routines used by **pixel_window**.

misc_utils	module, containing:
assert, fatal_error	interrupt code in case of error
extension	module, containing:
getEnvironment	read environment variable
fitstools	module, containing:
read_dbintab	reads double precision binary table

RELATED ROUTINES

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

gaussbeam	routine to generate a gaussian beam window function
generate_beam	returns a beam window function
alter_alm, rotate_alm	modifies $a_{\ell m}$ to emulate effect of real space filtering and coordinate rotation respectively
alm2map	synthetize a **HEALPix** map from its $a_{\ell m}$ (or $a_{\ell m}^{\rm (pix)}$).
alm2map_der	synthetize a map and its derivatives from its $a_{\ell m}$ (or $a_{\ell m}^{\rm (pix)}$).

pix2xxx,ang2xxx,vec2xxx, nest2ring,ring2nest

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

The Fortran90 module pix_tools contains some subroutines to convert between pixel number in the **HEALPix** map and (θ, ϕ) or (x, y, z) coordinates on the sphere. Some of these routines are listed here.

ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	$N_{\rm side}$ parameter for the **HEALPix** map.
ipnest	I4B/ I8B	—	pixel identification number in NESTED scheme over the range $\{0,N_{\rm pix}-1\}$.
ipring	I4B/ I8B	—	pixel identification number in RING scheme over the range $\{0,N_{\rm pix}-1\}$.
theta	DP	—	colatitude in radians measured southward from north pole in $\{0,\pi\}$.
phi	DP	—	longitude in radians, measured eastward in $[0, 2\pi]$.
vector(3)	DP	—	three dimensional cartesian position vector (x, y, z). The north pole is $(0, 0, 1)$. An output vector is normalised to unity.
vertex(3,4) OPTIONAL	DP	OUT	three dimensional cartesian position vectors (x, y, z) (normalised to unity) pointing to the 4 vertices of a given pixel. The four vertices are listed in the order North, West, South, East.

ROUTINES:

`call 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`.

`call 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 renders cartesian vector coordinates of the considered pixel four vertices.

`call 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`.

`call 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`.

`call 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`.

`call 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 renders cartesian vector coordinates of the considered pixel four vertices.

`call 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`.

`call 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` .

`call nest2ring(nside, ipnest, ipring)`

performs conversion from NESTED to RING pixel number.

`call ring2nest(nside, ipring, ipnest)`

performs conversion from RING to NESTED pixel number.

MODULES & ROUTINES

This section lists the modules and routines used by **pix2xxx,ang2xxx,vec2xxx, nest2ring,ring2nest**.

mk_pix2xy, mk_xy2pix	routines used in the conversion between pixel values and "cartesian" coordinates on the Healpix face.

RELATED ROUTINES

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

neighbours_nest	find neighbouring pixels.
ang2vec	convert (θ, ϕ) spherical coordinates into (x, y, z) cartesian coordinates.
vec2ang	convert (x, y, z) cartesian coordinates into (θ, ϕ) spherical coordinates.
convert_inplace	in-place conversion between RING and NESTED for integer/real/double maps.
convert_nest2ring	convert from NESTED to RING scheme using a temporary array.
nest2uniq, uniq2nest	conversion of standard pixel index to/from Unique ID number

planck_rng

Location in HEALPix directory tree: src/f90/mod/rngmod.f90

The derived type planck_rng is used by the Random Number Generation routines rand_init, rand_uni, rand_gauss to describe fully the current RNG sequence.
Most users do not need to know about the planck_rng definition. It may be useful for those wanting to take a snapshot of the RNG sequence they are using (by eg, dumping the latest values of planck_rng structure on disk) so that the same sequence can be resumed later on from that same point.

The type planck_rng is a structure defined as

name	type	definition
x, y, z, w	I4B	internal variables of uniform RNG
gset	DP	internal variable for Gaussian RNG
empty	LGT	flag used by Gaussian RNG

RELATED ROUTINES

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

rand_gauss	function which returns a random normal deviate.
rand_uni	function which returns a random uniform deviate.
rand_init	subroutine to initiate a random number sequence.

plm_gen

Location in HEALPix directory tree: src/f90/mod/alm_tools.F90

This routine computes the latitude dependent part $\lambda_{\ell m}$ of the spherical harmonics ($Y_{\ell m}(\theta,\phi) = \lambda_{\ell m}(\theta)e^{im\phi}$) of spin 0 and 2 (see **HEALPix** primer) used to synthetize or analyze **HEALPix** maps of temperature and polarisation. Since version 2.20, which introduced optimized algorithms for spherical harmonic transforms, it has become obsolete and should no longer be used.

FORMAT

call plm_gen(nsmax, nlmax, nmmax, plm)

ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	The $N_{\rm side}$ value for which to compute the $\lambda_{\ell m}$.
nlmax	I4B	IN	The maximum multipole order ℓ of the generated $\lambda_{\ell m}$.
nmmax	I4B	IN	The maximum degree m of the generated $\lambda_{\ell m}$.
plm(0:n_plm-1, 1:p)	DP	OUT	The $\lambda_{\ell m}$ values, either for temperature only ($p = 1$) or temperature and polarisation ($p = 3$). The number of $\lambda_{\ell m}$ is n_plm = nsmax*(nmmax+1)*(2*nlmax-nmmax+2). They are stored in the order of increasing order ℓ, increasing degree m, for all the **HEALPix** ring colatitudes θ from North Pole to Equator, ie $\lambda_{00}(\theta_1)$, $\lambda_{10}(\theta_1)$, $\lambda_{20}(\theta_1)$, ..., $\lambda_{11}(\theta_1)$, $\lambda_{21}(\theta_1)$; ..., $\lambda_{00}(\theta_2)$ ...

EXAMPLE:

```
use healpix_types
use alm_tools, only :  plm_gen
integer(I4B) ::  nside, lmax, mmax, n_plm
real(DP), dimension(:,:), allocatable ::  plm
...
nside=256 ; lmax=512 ; mmax=lmax
npix=nside2npix(nside)
n_plm=nside*(mmax+1)*(2*lmax-mmax+2)
allocate(plm(0:n_plm-1,1:3))
...
call plm_gen(nside, lmax, mmax, plm)
```

Computes the spherical harmonics for temperature and polarisation for $N_{\rm side} = 256$, up to 512 in ℓ and m.

MODULES & ROUTINES

This section lists the modules and routines used by **plm_gen**.

compute_lam_mm, get_pixel_layout, gen_lamfac,gen_mfac, gen_normpol, gen_recfac, init_rescale, l_min_ylm	Ancillary routines used for $\lambda_{\ell m}$ recursion
misc_utils	module, containing:
assert_alloc	routine to print error message, when an array can not be allocated properly

RELATED ROUTINES

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

alm2map	routine generating maps of temperature and polarisation from their $a_{\ell m}$ that can use precomputed $\lambda_{\ell m}$ generated by plm_gen
map2alm	routine analysing maps of temperature and polarisation that can use precomputed $\lambda_{\ell m}$ generated by plm_gen

plmgen	executable using plm_gen to compute the $\lambda_{\ell m}$ and writting them on disc

query_disc

Location in HEALPix directory tree: **src/f90/mod/pix_tools.F90**

Routine to find the index of all pixels within an angular distance radius from a defined center. The output indices can be either in the RING or NESTED scheme

FORMAT

call query_disc(nside, vector0, radius, listpix, nlist[, nest, inclusive])

ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\rm side}$ parameter of the map.
vector0(3)	DP	IN	cartesian vector pointing at the disc center.
radius	DP	IN	disc radius in radians.
listpix(0:*)	I4B/ I8B	OUT	the index for all pixels within `radius`. Make sure that the size of the array is big enough to contain all pixels.
nlist	I4B/ I8B	OUT	The number of pixels listed in `listpix`.
nest (OPTIONAL)	I4B	IN	The pixel indices are in the NESTED numbering scheme if nest=1, and in RING scheme otherwise.
inclusive (OPTIONAL)	I4B	IN	If set to 1, all the pixels overlapping (even partially) with the disc are listed, otherwise only those whose center lies within the disc are listed.

EXAMPLE:

```
use healpix_modules
call query_disc(256,(/0,0,1/),pi/2,listpix,nlist,nest=1)
```

Returns the NESTED pixel index of all pixels north of the equatorial line in a $N_{\rm side} = 256$ map.

MODULES & ROUTINES

This section lists the modules and routines used by **query_disc**.

in_ring	routine to find the pixels in a certain slice of a given ring.
ring_num	function to return the ring number corresponding to the coordinate z

RELATED ROUTINES

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

pix2ang, ang2pix	convert between angle and pixel number.
pix2vec, vec2pix	convert between a cartesian vector and pixel number.
query_disc, query_polygon, query_strip, query_triangle	render the list of pixels enclosed respectively in a given disc, polygon, latitude strip and triangle
surface_triangle	computes the surface in steradians of a spherical triangle defined by 3 vertices

query_polygon

Location in HEALPix directory tree: **src/f90/mod/pix_tools.F90**

Routine to find the index of all pixels enclosed in a polygon. The polygon should be convex, or have only one concave vertex. The edges should not intersect each other. The output indices can be either in the RING or NESTED scheme

FORMAT

call query_polygon(nside, vlist, nv, listpix, nlist[, nest, inclusive])

ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\rm side}$ parameter of the map.
vlist(1:3,0:*)	DP	IN	cartesian vector pointing at polygon respective vertices.
nv	I4B	IN	number of vertices, should be equal to 3 or larger.
listpix(0:*)	I4B/ I8B	OUT	the index for all pixels enclosed in the triangle. Make sure that the size of the array is big enough to contain all pixels.
nlist	I4B/ I8B	OUT	The number of pixels listed in `listpix`.
nest (OPTIONAL)	I4B	IN	The pixel indices are in the NESTED numbering scheme if nest=1, and in RING scheme otherwise.
inclusive (OPTIONAL)	I4B	IN	If set to 1, all the pixels overlapping (even partially) with the polygon are listed, otherwise only those whose center lies within the polygon are listed.

EXAMPLE:

```
use healpix_modules
real(dp), dimension(1:3,0:9) ::  vertices
vertices(:,0) = (/0.,0.,1./) !  +z
vertices(:,1) = (/1.,0.,0./) !  +x
vertices(:,2) = (/1.,1.,-1./) !  x+y-z
vertices(:,3) = (/0.,1.,0./) !  +y

call query_polygon(256,vertices,4,listpix,nlist,nest=0)
```

Returns the RING pixel index of all pixels in the polygon with vertices of cartesian coordinates (0,0,1), (1,0,0), (1,1,-1) and (0,1,0) in a $N_{\rm side} = 256$ map.

MODULES & ROUTINES

This section lists the modules and routines used by **query_polygon**.

isort	routine to sort integer number
query_triangle	render the list of pixels enclosed in a given triangle
surface_triangle	computes the surface of a spherical triangle defined by 3 vertices
vect_prod	routine to compute the vectorial product of two 3D vectors

RELATED ROUTINES

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

pix2ang, ang2pix	convert between angle and pixel number.
pix2vec, vec2pix	convert between a cartesian vector and pixel number.
query_disc, query_polygon, query_strip, query_triangle	render the list of pixels enclosed respectively in a given disc, polygon, latitude strip and triangle
surface_triangle	computes the surface in steradians of a spherical triangle defined by 3 vertices

query_strip

Location in HEALPix directory tree: **src/f90/mod/pix_tools.F90**

Routine to find the index of all pixels enclosed in a latitude strip. The output indices can be either in the RING or NESTED scheme

FORMAT call query_strip(nside, theta1, theta2, listpix, nlist[, nest, inclusive])

ARGUMENTS

name&dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\rm side}$ parameter of the map.
theta1	DP	IN	colatitude lower bound in radians measured from North Pole (between 0 and π).
theta2	DP	IN	colatitude upper bound in radians measured from North Pole (between 0 and π). If theta1< theta2, the pixels lying in [theta1,theta2] are output, otherwise, the pixel lying in [0, theta2] and those lying in [theta1, π] are output.
listpix(0:*)	I4B/ I8B	OUT	the index for all pixels enclosed in the strip(s). Make sure that the size of the array is big enough to contain all pixels.
nlist	I4B/ I8B	OUT	The number of pixels listed in `listpix`.
nest (OPTIONAL)	I4B	IN	The pixel indices are in the NESTED numbering scheme if nest=1, and in RING scheme otherwise.
inclusive (OPTIONAL)	I4B	IN	If set to 1, all the pixels overlapping (even partially) with the strip are listed; otherwise only those whose center lies within the strip are listed.

EXAMPLE:

```
call query_strip(256,0.75*PI,0.2*PI,listpix,nlist,nest=1)
```

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

MODULES & ROUTINES

This section lists the modules and routines used by **query_strip**.

in_ring	routine to find the pixels in a certain slice of a given ring.
intrs_intrv	routine to compute the intersection of 2 intervals on a circle
ring_num	function to return the ring number corresponding to the coordinate z
vect_prod	routine to compute the vectorial product of two 3D vectors

RELATED ROUTINES

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

pix2ang, ang2pix	convert between angle and pixel number.
pix2vec, vec2pix	convert between a cartesian vector and pixel number.
query_disc, query_polygon, query_strip, query_triangle	render the list of pixels enclosed respectively in a given disc, polygon, latitude strip and triangle
surface_triangle	computes the surface in steradians of a spherical triangle defined by 3 vertices

query_triangle

Location in HEALPix directory tree: **src/f90/mod/pix_tools.F90**

Routine to find the index of all pixels enclosed in a spherical triangle described by its three vertices. The output indices can be either in the RING or NESTED scheme

FORMAT call query_triangle(nside, v1, v2, v3, listpix, nlist[, nest, inclusive])

ARGUMENTS

name&dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\rm side}$ parameter of the map.
v1(3)	DP	IN	cartesian vector pointing at the triangle first vertex.
v2(3)	DP	IN	cartesian vector pointing at the triangle second vertex.
v3(3)	DP	IN	cartesian vector pointing at the triangle third vertex.
listpix(0:*)	I4B/ I8B	OUT	the index for all pixels enclosed in the triangle. Make sure that the size of the array is big enough to contain all pixels.
nlist	I4B/ I8B	OUT	The number of pixels listed in `listpix`.
nest (OPTIONAL)	I4B	IN	The pixel indices are in the NESTED numbering scheme if nest=1, and in RING scheme otherwise.
inclusive (OPTIONAL)	I4B	IN	If set to 1, all the pixels overlapping (even partially) with the triangle are listed, otherwise only those whose center lies within the triangle are listed.

EXAMPLE:

```
call query_triangle(256,(/1,0,0/),(/0,1,0/),(/0,0,1/),listpix,nlist)
```

Returns the RING pixel index of the (98560) pixels in the octant $(x, y, z > 0)$ in a $N_{\rm side} = 256$ map.

MODULES & ROUTINES

This section lists the modules and routines used by **query_triangle**.

in_ring	routine to find the pixels in a certain slice of a given ring.
intrs_intrv	routine to compute the intersection of 2 intervals on a circle
ring_num	function to return the ring number corresponding to the coordinate z
vect_prod	routine to compute the vectorial product of two 3D vectors

RELATED ROUTINES

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

pix2ang, ang2pix	convert between angle and pixel number.
pix2vec, vec2pix	convert between a cartesian vector and pixel number.
query_disc, query_polygon, query_strip, query_triangle	render the list of pixels enclosed respectively in a given disc, polygon, latitude strip and triangle
surface_triangle	computes the surface in steradians of a spherical triangle defined by 3 vertices

rand_gauss

Location in HEALPix directory tree: src/f90/mod/rngmod.f90

This routine returns a number out of a pseudo-random normal deviate (ie, its probability distribution is a Gaussian of mean 0 and variance 1).

FORMAT

var=rand_gauss(rng_handle)

ARGUMENTS

name & dimensionality	kind	in/out	description
rng_handle	planck_rng	INOUT	structure of type `planck_rng` containing on all information necessary to continue same random sequence.
var	DP	OUT	number belonging to a pseudo-random normal deviate.

EXAMPLE:

```
use healpix_types
use rngmod
type(planck_rng) ::  rng_handle
real(dp) ::  gauss

call rand_init(rng_handle, 12345, 6789012)
gauss = rand_gauss(rng_handle)
```

initiates a random sequence with the pair of seeds (12345, 6789012), and generates one number out of the normal deviate.

RELATED ROUTINES

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

planck_rng	derived type describing RNG state
rand_uni	function which returns a random uniform deviate.
rand_init	subroutine to initiate a random number sequence.

rand_init

Location in HEALPix directory tree: src/f90/mod/rngmod.f90

This routine initializes, with up to 4 seeds, a randomn number sequence. The generator being primed is an F90 port of an xorshift generator described in Marsaglia, Journal of Statistical Software 2003, vol 8. It has a theoretical period of $2^{128} - 1 \approx 3.410^{38}$. Please refer to the "Comment on Random Number Generator" in the Fortran90 facilities guidelines.

FORMAT

call rand_init(rng_handle, [seed1, seed2, seed3, seed4])

ARGUMENTS

name & dimensionality	kind	in/out	description
rng_handle	planck_rng	OUT	structure of type `planck_rng` containing on output all information necessary to continue same random sequence.
seed1 (OPTIONAL)	I4B	IN	first seed of the random sequence. Can be of arbitray sign. If set to zero or not provided will be replaced internally by a non-zero hard coded value.
seed2 (OPTIONAL)	I4B	IN	second seed. Same properties as above
seed3 (OPTIONAL)	I4B	IN	third seed. Same as above.
seed4 (OPTIONAL)	I4B	IN	fourth seed. Same as above.

EXAMPLE:

```
use rngmod
type(planck_rng) ::  rng_handle
call rand_init(rng_handle, 12345, 6789012)
```

initiates a random sequence with the pair of seeds (12345, 6789012).

RELATED ROUTINES

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

planck_rng	derived type describing RNG state
rand_gauss	function which returns a random normal deviate.
rand_uni	function which returns a random uniform deviate.

rand_uni

Location in HEALPix directory tree: src/f90/mod/rngmod.f90

This routine returns a number out of a pseudo-random uniform deviate (ie, its probability distribution is uniform in the range]0,1[).

FORMAT

var=rand_uni(rng_handle)

ARGUMENTS

name & dimensionality	kind	in/out	description
rng_handle	planck_rng	INOUT	structure of type `planck_rng` containing on all information necessary to continue same random sequence.
var	DP	OUT	number belonging to a pseudo-random uniform deviate.

EXAMPLE:

```
use healpix_types
use rngmod
type(planck_rng) ::  rng_handle
real(dp) ::  uni

call rand_init(rng_handle, 12345, 6789012)
uni = rand_uni(rng_handle)
```

initiates a random sequence with the pair of seeds (12345, 6789012), and generates one number out of the uniform deviate.

RELATED ROUTINES

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

planck_rng	derived type describing RNG state
rand_gauss	function which returns a random normal deviate.
rand_init	subroutine to initiate a random number sequence.

read_asctab*

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This routine is obsolete, use fits2cl instead

read_bintab*

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This routine reads a **HEALPix** map from a binary FITS-file. The routine can read a temperature map or both temperature and polarisation maps (T,Q,U)

FORMAT call read_bintab*(filename, map, npixtot, nmaps, nullval, anynull*[,header, units, extno]*)

Arguments appearing in *italic* are optional.

ARGUMENTS

name &d imensionality	**kind**	**in/out**	**description**
filename(LEN=`filenamelen`)	CHR	IN	filename of FITS-file containing the map(s).
npixtot	I4B	IN	Number of pixels to be read from map.
nmaps	I4B	IN	number of maps to be read, 1 for temperature only, and 3 for (T,Q,U).
map(0:npixtot-1,1:nmaps)	SP/ DP	OUT	the map read from the FITS-file.
nullval	SP/ DP	OUT	value of missing pixels in the map.
anynull	LGT	OUT	`.TRUE.`, if there are missing pixels, and `.FALSE.` otherwise.
header(LEN=80)(1:) (OPTIONAL)	CHR	OUT	character string array containing the FITS header read from the file. Its dimension has to be defined prior to calling the routine
units(LEN=*)(1:nmaps)	CHR	OUT	character string array containing the physical units of each map read
extno	I4B	IN	extension number to read the data from (0 based).(**default:** 0) (the first extension is read)

EXAMPLE:

```
call read_bintab ('map.fits', map, 12*32**2, 1, nullval, anynull)
```

Reads a **HEALPix** temperature map from the file 'map.fits' to the array map(0:12*32**2-1,1:1). The pixel number 12*32**2 is the number of pixels in a $N_{\rm side} = 32$ **HEALPix** map. If there are missing pixels in the input file (with value `NaN` (Not a Number), ±`Infinity`, or matching the FITS keyword `BAD_DATA`) then `anynull` is `.TRUE.` and these pixels get the value returned in `nullval`.

MODULES & ROUTINES

This section lists the modules and routines used by **read_bintab***.

fitstools	module, containing:
printerror	routine for printing FITS error messages.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

input_map	Routine which reads a map using read_bintab* and fills missing pixels with a given value.
map2alm	Routine which analyse a map and returns the a_{lm} coefficients.
read_fits_cut4	Routine to read cut sky **HEALPix** FITS maps
write_plm, write_bintab	Routines to write **HEALPix** FITS maps

read_conbintab*

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This routine reads a FITS file containing $a_{\ell m}$ values for constained realisation. The FITS file is supposed to contain one integer column with $index = \ell^2 + \ell + m + 1$ and 2 or 4 single (or double) precision columns with real/imaginary $a_{\ell m}$ values and real/imaginary standard deviation on these $a_{\ell m}$. It is supposed to contain either 1 or 3 extension(s) containing respectively the $a_{\ell m}$ for T and if applicable E and B.

FORMAT

call read_conbintab*(filename, alms, nalms[, units, extno])

ARGUMENTS

name&dimensionality	kind	in/out	description
filename(LEN=`filenamelen`)	CHR	IN	filename of FITS file containing $a_{\ell m}$.
nalms	I4B	IN	Number of $a_{\ell m}$ values to read from the file.
alms(0:nalms-1,1:6)	SP/ DP	OUT	the $a_{\ell m}$ read from the file. alms(i,1) and alms(i,2) contain the ℓ and m values for the ith $a_{\ell m}$. alms(i,3) and alms(i,4) contain the real and imaginary value of the ith $a_{\ell m}$. Finally, the standard deviation for the ith $a_{\ell m}$ is contained in alms(i,5) (real) and alms(i,6) (imaginary).
units(len=20)(1:) (OPTIONAL)	CHR	OUT	character string containing the units of the $a_{\ell m}$
extno (OPTIONAL)	I4B	IN	extension (0 based) of the FITS file to be read

EXAMPLE:

```
call read_conbintab ('alms.fits',alms,65*66/2)
```

Read 65*66/2 (the number of $a_{\ell m}$ needed to fill the whole range from l=0 to l=64) $a_{\ell m}$ values from the file 'alms.fits' into the array alms(0:65*66/2-1,1:6).

MODULES & ROUTINES

This section lists the modules and routines used by **read_conbintab***.

fitstools	module, containing:
printerror	routine for printing FITS error messages.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

alms2fits, dump_alms	routines to write $a_{\ell m}$ to a FITS-file
fits2alms	has the same function as read_conbintab but is more general.
number_of_alms, getsize_fits	can be used to find out the number of $a_{\ell m}$ available in the file.

read_dbintab

Location in HEALPix directory tree: **src/f90/mod/fitstools.F90**

This routine reads a double precision binary array from a FITS-file. It is used by **HEALPix** to read precomputed $P_{\ell m}(\theta)$ values and pixel window functions.

FORMAT

call read_dbintab(filename, map, npixtot, nmaps, nullval, anynull, units)

ARGUMENTS

name & dimensionality	kind	in/out	description
filename(LEN=`filenamelen`)	CHR	IN	filename of FITS-file containing the double precision array.
npixtot	I4B	IN	Number of values to be read from the file.
nmaps	I4B	IN	number of 1-dim. arrays, 1 for scalar $P_{\ell m}$s and pixel windows, 3 for scalar and tensor $P_{\ell m}$s.
map(0:npixtot-1,1:nmaps)	DP	OUT	the array read from the FITS-file.
nullval	DP	OUT	value of missing pixels in the array.
anynull	LGT	OUT	TRUE, if there are missing pixels, and FALSE otherwise.
units(len=20)(1:nmaps)	CHR	OUT	respective physical units of the maps in the FITS file.

EXAMPLE:

```
call read_dbintab ('plm_32.fits',plm,65*66*32,1,nullval,anynull)
```

Reads precomputed scalar $P_{\ell m}(\theta)$ from the file 'plm_32.fits'. The values are returned in the array plm(0:65*66*32,1:1). The number of values 65*66*32 is the number of precomputed $P_{\ell m}(\theta)$ for a $N_{\rm side} = 32$, $\ell_{\rm max} = 64$ map. If there are missing values in the file, anynull is TRUE and nullval contains the values of these pixels.

MODULES & ROUTINES

This section lists the modules and routines used by **read_dbintab**.

fitstools	module, containing:
printerror	routine for printing FITS error messages.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

plmgen	Executable to create files with precomputed $P_{\ell m}(\theta)$.
write_plm	Routine to create a file to be read by read_dbintab.

read_fits_cut4

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This routine reads a cut sky **HEALPix** map from a FITS file. The format used for the FITS file follows the one used for Boomerang98 and is adapted from COBE/DMR

FORMAT

call read_fits_cut4(filename, np, pixel, *[signal, n_obs, serror, header, units, extno]*)

Arguments appearing in *italic* are optional.

ARGUMENTS

name&dimensionality	kind	in/out	description
filename(LEN=`filenamelen`)	CHR	IN	FITS file to be read from, containing a cut sky map
np	I4B	IN	number of pixels to be read from the file
pixel(0:np-1)	I4B	OUT	index of observed (or valid) pixels
signal(0:np-1) (OPTIONAL)	SP	OUT	value of signal in each observed pixel
n_obs(0:np-1)	I4B	OUT	number of observation per pixel
serror(0:np-1)	SP	OUT	*rms* of signal in pixel. (For white noise, this would be $\propto 1/\sqrt{\text{n\_obs}}$)
header(LEN=80)(1:)	CHR	OUT	FITS extension header
units(LEN=20)	CHR	OUT	maps units (applies only to Signal and Serror, which are assumed to have the same units)
extno	I4B	IN	extension number (0 based) for which map is read. Default = 0 (first extension).

MODULES & ROUTINES

This section lists the modules and routines used by **read_fits_cut4**.

fitstools	module, containing:
printerror	routine for printing FITS error messages.
cfitsio	library for FITS file handling.

RELATED ROUTINES

This section lists the routines related to **read_fits_cut4**.

anafast	executable that reads a **HEALPix** map and analyses it.
synfast	executable that generate full sky **HEALPix** maps
getsize_fits	routine to know the size of a FITS file and its type (eg, full sky vs cut sky)
input_map	all purpose routine to input a map of any kind from a FITS file
output_map	subroutine to write a FITS file from a **HEALPix** map
write_fits_cut4	subroutine to write a cut sky map into a FITS file

read_fits_partial

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This routine reads unpolarised or polarised partial sky **HEALPix** map from a FITS file.
For more information on the FITS file format supported in **HEALPix**, including the one implemented in read_fits_partial, see https://healpix.sourceforge.io/data/examples/healpix_fits_specs.pdf.

FORMAT

call read_fits_partial(filename, pixel, cutmap, *[header, extno]*)

Arguments appearing in *italic* are optional.

ARGUMENTS

name & dimensionality	kind	in/out	description
filename(LEN=filenamelen)	CHR	IN	FITS file to be read from, containing a partial sky map
pixel(0:np-1)	I4B/ I8B	OUT	index of observed (or valid) pixels
cutmap(0:np-1,1:nc)	SP/ DP	OUT	value of unpolarised or polarised map for each observed pixel
header(LEN=80)(1:)	CHR	OUT	FITS extension header
extno	I4B	IN	extension number (0 based) for which map is read. Default = 0 (first extension).

EXAMPLE:

```
use healpix_modules
character(len=FILENAMELEN) ::  file
integer(i4b) ::  nmaps, polarisation, npix, nside
integer(i4b), allocatable, dimension(:)  ::  pixel
real(SP), allocatable, dimension(:,:)  ::  data
character(len=80), dimension(1:100) ::  header=""
```

```
file="https://healpix.sourceforge.io/data/examples/partial_TQU.fits"
npix = getsize_fits(file, nmaps=nmaps, polarisation=polarisation)
print*, npix, nmaps, polarisation
allocate(pixel(0:npix-1))
allocate(data(0:npix-1,1:3))
call read_fits_partial(file, pixel, data, header=header)
print*,pixel(0), data(0,1:3)
```

reads a remote partial sky FITS file and prints the index and IQU values of the first pixel its contains.

MODULES & ROUTINES

This section lists the modules and routines used by **read_fits_partial**.

fitstools	module, containing:
printerror	routine for printing FITS error messages.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

anafast	executable that reads a **HEALPix** map and analyses it.
synfast	executable that generate full sky **HEALPix** maps
getsize_fits	routine to know the size of a FITS file and its type (eg, full sky vs cut sky)
input_map	all purpose routine to input a map of any kind from a FITS file
output_map	subroutine to write a FITS file from a full sky **HEALPix** map
write_fits_partial	subroutine to write a partial map into a FITS file which can be read by read_fits_partial and/or input_map.

read_par

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This routine reads the 'NSIDE', 'TFIELDS' , 'MAX-LPOL', and optionally 'MAX-MPOL' keywords from a FITS-file. These keywords desribe $N_{\rm side}$, number of datasets (maps) and maximum multipole ℓ (order) and m (degree) value for the file. If a keyword is not found in the FITS file, a value of -1 is returned instead. The file could eg. be a **HEALPix** map, or a set of $a_{\ell m}$ or precomputed $P_{\ell m}(\theta)$

FORMAT

call read_par(filename, nside, lmax, tfields[, *mmax*])

ARGUMENTS

name & dimensionality	kind	in/out	description
filename(LEN=`filenamelen`)	CHR	IN	filename of the FITS file.
nside	I4B	OUT	'NSIDE' keyword value from the FITS header.
lmax	I4B	OUT	'MAX-LPOL' keyword value from the FITS header.
tfields	I4B	OUT	'TFIELDS' keyword value from the FITS header.
mmax (OPTIONAL)	I4B	OUT	'MAX-MPOL' keyword value from the FITS header.

EXAMPLE:

```
call read_par('plm_128p.fits', nside, lmax, nhar)
```

Checks the $N_{\rm side}$ and maximum ℓ value used for the precomputed $P_{\ell m}(\theta)$ that are stored in the file 'plm_128p.fits'. If the file also contains tensor harmonics, nhar is returned with the value 3, otherwise it is 1.

MODULES & ROUTINES

This section lists the modules and routines used by **read_par**.

fitstools	module, containing:
printerror	routine for printing FITS error messages.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

synfast, plmgen	executables that produce FITS-files with keywords relevant to this routine.

real_fft

Location in HEALPix directory tree: src/f90/mod/healpix_fft.F90

This routine performs a forward or backward Fast Fourier Transformation on its argument `data`.

FORMAT

call real_fft(data, backward)

ARGUMENTS

name & dimensionality	kind	in/out	description
data(:)	XXX	INOUT	array containing the input and output data. Can be of type real(sp) or real(dp)
backward	LGT	IN	Optional argument. If present and true, perform backward transformation, else forward

EXAMPLE:

```
use healpix_fft
call real_fft (data, backward=.true.)
```

Performs a backward FFT on data.

RELATED ROUTINES

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

complex_fft routine for FFT of complex data

remove_dipole*

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

This routine provides a means to fit and remove the dipole and monopole from a **HEALPix** map. The fit is obtained by solving the linear system

$$\sum_{j=0}^{d^2-1} A_{ij}\ f_j = b_i \tag{19}$$

with, $d = 1$ or 2, and

$$b_i \equiv \sum_{p\in\mathcal{P}} s_i(p) w(p) m(p), \tag{20}$$

$$A_{ij} \equiv \sum_{p\in\mathcal{P}} s_i(p) w(p) s_j(p), \tag{21}$$

where $\mathcal{P}$ is the set of valid, unmasked pixels, m is the input map, w is pixel weighting, while $s_0(p) = 1$ and $s_1(p) = x$, $s_2(p) = y$, $s_3(p) = z$ are respectively the monopole and dipole templates. The output map is then

$$m'(p) = m(p) - \sum_{i=0}^{d^2-1} f_i s_i(p). \tag{22}$$

FORMAT call remove_dipole*(nside, map, ordering, degree, multipoles, zbounds[, fmissval, mask, weights])

ARGUMENTS

name & dimensionality	**kind**	**in/out**	**description**
nside	I4B	IN	value of $N_{\rm side}$ resolution parameter for input map
map(0:12*nside*nside-1)	SP/ DP	INOUT	**HEALPix** map from which the monopole and dipole will be removed. Those are removed from *all unflagged pixels*, even those excluded by the cut `zounds` or the `mask`.
ordering	I4B	IN	**HEALPix** scheme 1:RING, 2: NESTED
degree	I4B	IN	multipoles to fit and remove. It is either 0 (nothing done), 1 (monopole only) or 2 (monopole and dipole).
multipoles(0:degree*degree-1)	DP	OUT	values of best fit monopole and dipole. The monopole is described as a scalar in the same units as the input map, the dipole as a 3D cartesian vector, in the same units.
zbounds(1:2)	DP	IN	section of the map on which to perform the fit, expressed in terms of z = sin(latitude) = cos(θ). If zbounds(1)<zbounds(2), it is performed *on* the strip zbounds(1)< z <zbounds(2); if not, it is performed *outside* the strip zbounds(2)$\leq z \leq$zbounds(1). If absent, the whole map is processed.
fmissval (OPTIONAL)	SP/ DP	IN	value used to flag bad pixel on input (**default:** -1.6375e30). Pixels with that value are ignored during the fit, and left unchanged on output.
mask(0:12*nside*nside-1) (OPTIONAL)	SP/ DP	IN	mask of valid pixels. Pixels with $\|\text{mask}\| < 10^{-10}$ are not used for fit. Note: the map is *not* multiplied by the mask.
weights(0:12*nside*nside-1) (OPTIONAL)	SP/ DP	IN	weight to be given to each map pixel before doing the fit. By default pixels are given a uniform weight of 1. Note: the output map is *not* multiplied by the weights.

EXAMPLE:

```
s = sin(15.0_dp * DEG2RAD)
call remove_dipole*(128, map, 1, 2, multipoles, (/ s, -s /) )
```

Will compute and remove the best fit monopole and dipole from a map with $N_{\rm side} = 128$ in RING ordering scheme. The fit is performed on pixels with $|b| > 15^o$.

MODULES & ROUTINES

This section lists the modules and routines used by **remove_dipole***.

pix_tools module, containing:

RELATED ROUTINES

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

add_dipole routine to add a dipole and monopole to a map.

ring_analysis

Location in HEALPix directory tree: **src/f90/mod/alm_tools.F90**

This subroutine computes the Fast Fourier Transform of a single ring of pixels and extends the computed coefficients up to the maximum m of the transform.

FORMAT

call ring_analysis(nsmax, nlmax, nmmax, datain, nph, dataout, kphi0)

ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	$N_{\rm side}$ of the map.
nlmax	I4B	IN	Maximum ℓ of the analysis.
nmmax	I4B	IN	Maximum m of the analysis.
nph	I4B	IN	The number of points on the ring.
datain(0:nph-1)	DP	IN	Function values on the ring.
dataout(0:nmmax)	DPC	OUT	Fourier components, replicated to $Nmmax$.
kphi0	I4B	IN	0 if the first pixel on the ring is at $\phi = 0$; 1 otherwise.

EXAMPLE:

```
call ring_analysis(64,128,128,datain,8,dataout,0)
```

Returns the periodically extended complex Fourier Transform of datain in dataout. They are returned in the following order: 0 1 2 3 4 5 6 7 6 5 4 3 2 1 0 .... The value $kphi0 = 0$ specifies that no phase factor needed to be applied, because the ring starts at $\phi = 0$.

MODULES & ROUTINES

This section lists the modules and routines used by **ring_analysis**.

healpix_fft	module.

RELATED ROUTINES

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

ring_synthesis	Inverse transform (complex-to-real), used in alm2map, alm2map_der and synfast

ring_num

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

This function returns the ring number for a z coordinate.

FORMAT

var=ring_num(nside, z)

ARGUMENTS

name&dimensionality	kind	in/out	description
nside	I4B	IN	the $N_{\rm side}$ parameter of the map.
z	DP	IN	the z coordinate to find the ring number for.

EXAMPLE:

```
print*,ring_num(256, 0.5)
```

Prints the ring number of the ring at position $z = 0.5$.

MODULES & ROUTINES

This section lists the modules and routines used by **ring_num**.

None

RELATED ROUTINES

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

in_ring	Returns the pixels in a slice on a given ring.

ring_synthesis

Location in HEALPix directory tree: src/f90/mod/alm_tools.F90

FORMAT call ring_synthesis(nsmax, nlmax, nmmax, datain, nph, dataout, kphi0)

ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	$N_{\rm side}$ of the map.
nlmax	I4B	IN	Maximum ℓ of the analysis.
nmmax	I4B	IN	Maximum m of the analysis.
nph	I4B	IN	The number of points on the ring.
datain(0:nmmax)	DPC	IN	Fourier components as computed from the a_{lm}.
dataout(0:nph-1)	DP	OUT	Synthesized function values on the ring.
kphi0	I4B	IN	0 if the first pixel on the ring is at $\phi = 0$; 1 otherwise.

EXAMPLE:

```
call ring_synthesis(64,128,128,datain,8,dataout,1)
```

This computes the inverse (complex-to-real) Fast Fourier Transform for the second ring from the pole, containing 8 pixels, for a map resolution of $N_{\rm side} = 64$. 128 complex Fourier compoments contribute to these 8 pixels. The value $kphi0 = 1$ specifies that a phase factor needed to be applied to correctly rotate the ring into position on the **HEALPix** grid.

MODULES & ROUTINES

This section lists the modules and routines used by **ring_synthesis**.

healpix_fft	module.

RELATED ROUTINES

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

ring_analysis	Forward transform, used in map2alm and anafast

rotate_alm*

Location in HEALPix directory tree: src/f90/mod/alm_tools.F90

This routine transform the scalar (and tensor) $a_{\ell m}$ coefficients to emulate the effect of an arbitrary rotation of the underlying map. The rotation is done directly on the $a_{\ell m}$ using the Wigner rotation matrices, computed by recursion. To rotate the $a_{\ell m}$ for $\ell \leq \ell_{\rm max}$ the number of operations scales like $\ell^3_{\rm max}$.

FORMAT

call rotate_alm*(nlmax, alm_TGC, psi, theta, phi)

ARGUMENTS

name & dimensionality	kind	in/out	description
nlmax	I4B	IN	maximum ℓ value for the $a_{\ell m}$.
alm_TGC(1:p,0:nlmax,0:nlmax)	SPC/ DPC	INOUT	complex $a_{\ell m}$ values before and after rotation of the coordinate system. The first index here runs from 1:1 for temperature only, and 1:3 for polarisation. In the latter case, 1=T, 2=E, 3=B.
psi	DP	IN	first rotation: angle ψ about the z-axis. All angles are in radians and should lie in [-2π,2π], the rotations are active and the referential system is assumed to be right handed, the routine coordsys2euler_zyz can be used to generate the Euler angles ψ, θ, φ for rotation between standard astronomical coordinate systems;
theta	DP	IN	second rotation: angle θ about the original (unrotated) y-axis;
phi	DP	IN	third rotation: angle φ about the original (unrotated) z-axis;

EXAMPLE:

```
use alm_tools, only:  rotate_alm
...
call rotate_alm(64, alm_TGC, PI/3., 0.5_dp, 0.0_dp)
```

Transforms scalar and tensor $a_{\ell m}$ for $\ell_{\max} = m_{\max} = 64$ to emulate a rotation of the underlying map by $(\psi = \pi/3, \theta = 0.5, \varphi = 0)$.

EXAMPLE:

```
use coord_v_convert, only:  coordsys2euler_zyz
use alm_tools, only:  rotate_alm
...
call coordsys2euler_zyz(2000.0_dp, 2000.0_dp, 'E', 'G', psi, theta, phi)
call rotate_alm(64, alm_TGC, psi, theta, phi)
```

Rotate the $a_{\ell m}$ from Ecliptic to Galactic coordinates.

RELATED ROUTINES

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

coordsys2euler_zyz	can be used to generate the Euler angles ψ, θ, φ for rotation between standard astronomical coordinate systems
create_alm	Routine to create $a_{\ell m}$ coefficients.
alter_alm	Routine to modify $a_{\ell m}$ coefficients to apply or remove the effect of an instrumental beam.
map2alm	Routines to analyze a **HEALPix** sky map into its $a_{\ell m}$ coefficients.
alm2map	Routines to synthetize a **HEALPix** sky map from its $a_{\ell m}$ coefficients.
alms2fits, dump_alms	Routines to save a set of $a_{\ell m}$ in a FITS file.
xcc_v_convert	rotates a 3D coordinate vector from one astronomical coordinate system to another.

same_shape_pixels_nest, same_shape_pixels_ring

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

These routines provide the ordered list of all **HEALPix** pixels having the same shape as a given template, for a resolution parameter $N_{\rm side}$. Depending on the template considered the number of such pixels is either 8, 16, $4N_{\rm side}$ or $8N_{\rm side}$.

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

$$\begin{aligned} z = \cos(\theta) \geq 2/3, \qquad & 0 < \phi \leq \pi/2, \\ 2/3 > z \geq 0, \qquad & \phi = 0, \quad \text{or} \quad \phi = \frac{\pi}{4N_{\rm side}}. \end{aligned} \tag{23}$$

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

FORMAT call same_shape_pixels_nest(nside, template [, list, reflexion, nrep])

FORMAT call same_shape_pixels_ring(nside, template [, list, reflexion, nrep])

ARGUMENTS

name & dimensionality	**kind**	**in/out**	**description**
nside	I4B	IN	the **HEALPix** $N_{\rm side}$ parameter.
template	I4B/ I8B	IN	identification number of the template pixel (the numbering scheme of the pixel templates is the same for both routines).
list(0:nrep-1) OPTIONAL	I4B/ I8B	OUT	pointer containing the ordered list of NESTED/RING scheme identification numbers (in $\{0,12N_{\rm side}^2-1\}$) of all pixels having the same shape as the template provided. The routines will allocate the `list` array if it is not allocated upon calling.
reflexion(0:nrep-1) OPTIONAL	I4B	OUT	pointer containing the transformation(s) (in $\{0,\ 3\}$) to apply to each of the returned pixels 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. The routines will allocate the `list` array if it is not allocated upon calling.
nrep OPTIONAL	I4B/ I8B	OUT	number of pixels having the same template (either 8, 16, $4N_{\rm side}$ or $8N_{\rm side}$).

EXAMPLE:

```
use healpix_modules
integer, parameter ::  IXB = I4B ! for nside <= 8192
!integer, parameter ::  IXB = I8B ! for any valid nside
integer(I4B)::  nside
integer(IXB)::  template, nrep
integer(I4B), dimension(:), pointer ::  listref
integer(IXB), dimension(:), pointer ::  listpix

allocate(listref(0:0)) !  only the lower bound matters
allocate(listpix(0:0)) !  only the lower bound matters
nside = 256
template = 1234
```

```
call same_shape_pixels_ring(nside, template, list=listpix,
reflexion=listref, nrep=nrep)
print*,nrep
print*,listpix(0:nrep-1)
print*,listref(0:nrep-1)
```

Returns in `listpix` the RING-scheme index of the all the pixels having the same shape as the template #1234 for $N_{\rm side} = 256$. Upon return `listref` will contain the rotation/reflexions to apply to each pixel returned to match the template, and `nrep` will contain the number of pixels having that same shape (16 in that case). Note that some variables (corresponding to arguments `template`, `list` and `nrep`) must be of type `I8B` instead of `I4B` if $N_{\rm side} > 8192$ is to be used.

RELATED ROUTINES

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

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

scan_directories

Location in HEALPix directory tree: **src/f90/mod/paramfile_io.F90**

Function to scan a set of directories for a given file

FORMAT var=scan_directories(directories, filename, fullpath)

ARGUMENTS

name&dimensionality	kind	in/out	description
directories	CHR	IN	contains the set of directories (up to 20), separated by an ASCII character of value < 32 (see `concatnl`). During the search, it is assumed that the given directories and filename can be separated by nothing, a / (slash) or a \ (backslash)
filename	CHR	IN	the file to be found.
fullpath	CHR	OUT	returns the full path to the first occurrence of the file among the directories provided. Empty if the file is not found. The search is not recursive.
var	LGT	OUT	set to true if the file is found, to false otherwise.

EXAMPLE:

```
use paramfile_io
character(len=filenamelen) ::  dirs, full
logical(lgt) ::  found
dirs = concatnl('dir1','/dir2','/dir2/subdir1/') !build directories list
found = scan_directories(dirs, 'myfile', full) !  do the search
if (found) print*,trim(full)
```

Search for 'myfile' in the directories 'dir1', '/dir2', '/dir2/subdir1/'

RELATED ROUTINES

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

parse_xxx	parse an ASCII file for parameters definition
concatnl	concatenates a set of substrings into one string, interspaced with LineFeed character

size_holes_nest

Location in HEALPix directory tree: src/f90/mod/mask_tools.F90

For a input binary mask in NESTED ordering, size_holes_nest identifies the pixels located on the inner boundary of *invalid* regions

FORMAT

call size_holes_nest(nside, mask, nholes, nph, *[tags, sizeholes, listpix]*)

Arguments appearing in *italic* are optional.

ARGUMENTS

name & dim.	kind	in/out	description
nside	I4B	IN	The *nside* value of the input mask.
mask(0:Npix-1)	I4B	IN	Input binary NESTED-ordered mask. Npix = 12*nside*nside
nholes	I4B	OUT	Number of holes found
nph	I4B	OUT	Number of pixels in holes
tags(0:Npix-1) (OPTIONAL)	I4B	OUT	Pointer allocated by size_holes_nest, containing a sky map in which *invalid* pixels belonging to the largest hole have value -1, those belonging to the second largest hole have value -2, and so on, while valid pixels keep value +1.
sizeholes(0:nholes-1)	I4B	OUT	Pointer allocated by size_holes_nest, containing on output the respective size of each hole (in decreasing order). Eg, `sizeholes(0)` is the number of pixels in the largest hole (taking value -1 in `tags`).
listpix(0:nholes+nph)	I4B	OUT	Pointer allocated by size_holes_nest, containing on output the indexed list of pixels in each hole. Pixels located in the first (and largest) hole are given by `listpix(listpix(0):listpix(1)-1)`

EXAMPLE:

```
use healpix_types
use healpix_modules
...
call size_holes_nest(nside, mask, nholes, nph)
```

???

MODULES & ROUTINES

This section lists the modules and routines used by **size_holes_nest**.

mask_tools	mask processing module (see related routines below)

RELATED ROUTINES

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

dist2holes_nest	angular distance to closest invalid pixel of the given mask
fill_holes_nest	turn to *valid* all pixels located in 'holes' containing fewer pixels than the given threshold
maskborder_nest	identify inner boundary pixels of 'holes' for given mask
size_holes_nest	returns size (in pixels) of holes found in input mask

string, strlowcase, strupcase

Location in HEALPix directory tree: src/f90/mod/misc_utils.F90

The Fortran90 module misc_utils contains three functions to create or manipulate character strings.

ARGUMENTS

name & dimensionality	kind	in/out	description
number	LGT/ I4B/ SP/ DP	IN	number or boolean flag to be turned into a character string.
instring	CHR	IN	arbitrary character string.
outstring	CHR	—	output character string.
format OPTIONAL	CHR	IN	character string describing Fortran format of output.

FUNCTIONS:

`outstring = string(number [,format])`

returns in `outstring` its argument `number` converted to a character string. If `format` is provided it is used to format the output, if not, the fortran default format matching `number`'s type is used.

`outstring = strlowcase(instring)`

returns in `outstring` its argument `instring` converted to lowercase. ASCII characters in the [A-Z] range are mapped to [a-z], while all others remain unchanged.

`outstring = strupcase(instring)`

returns in `outstring` its argument `instring` converted to uppercase. ASCII characters in the [a-z] range are mapped to [A-Z], while all others remain unchanged.

EXAMPLE:

```
use misc_utils
character(len=24) ::  s1
s1 = string(123,'(i5.5)')
print*, trim(s1)
print*,trim(strupcase('*aBcD-123'))
print*,trim(strlowcase('*aBcD-123'))
```

Will printout `00123`, `*ABCD-123` and `*abcd-123`.

surface_triangle

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

Returns the surface in steradians of the spherical triangle described by its three vertices

FORMAT

call surface_triangle(v1, v2, v3, surface)

ARGUMENTS

name&dimensionality	kind	in/out	description
v1(3)	DP	IN	cartesian vector pointing at the triangle first vertex.
v2(3)	DP	IN	cartesian vector pointing at the triangle second vertex.
v3(3)	DP	IN	cartesian vector pointing at the triangle third vertex.
surface	DP	OUT	surface of the triangle in steradians.

EXAMPLE:

```
use healpix_types
use pix_tools, only :  surface_triangle
real(DP) ::  surface, one = 1.0_dp
call surface_triangle((/1,0,0/)*one, (/0,1,0/)*one, (/0,0,1/)*one,
surface)
print*, surface
```

Returns the surface in steradians of the triangle defined by the octant $(x, y, z > 0)$: 1.5707963267948966

RELATED ROUTINES

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

pix2ang, ang2pix	convert between angle and pixel number.
pix2vec, vec2pix	convert between a cartesian vector and pixel number.
query_disc, query_polygon, query_strip, query_triangle	render the list of pixels enclosed respectively in a given disc, polygon, latitude strip and triangle

template_pixel_nest, template_pixel_ring

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

Routines to provide the index of the template pixel associated with a given **HEALPix** pixel, for a resolution parameter $N_{\rm side}$.

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

$$\begin{aligned} z = \cos(\theta) \geq 2/3, \qquad & 0 < \phi \leq \pi/2, \\ 2/3 > z \geq 0, \qquad & \phi = 0, \quad \text{or} \quad \phi = \frac{\pi}{4N_{\rm side}}. \end{aligned} \tag{24}$$

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

FORMAT call template_pixel_nest(nside, pixel_nest, template, reflexion)

FORMAT call template_pixel_ring(nside, pixel_ring, template, reflexion)

ARGUMENTS

name & dimensionality	kind	in/out	description
nside	I4B	IN	the **HEALPix** $N_{\rm side}$ parameter.
pixel_nest	I4B/ I8B	IN	NESTED scheme pixel identification number over the range $\{0,12N_{\rm side}^2-1\}$.
pixel_ring	I4B/ I8B	IN	RING scheme pixel identification number over the range $\{0,12N_{\rm side}^2-1\}$.
template	I4B/ I8B	OUT	identification number of the template matching in shape the pixel provided (the numbering scheme of the pixel templates is the same for both routines).
reflexion	I4B	OUT	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

EXAMPLE:

```
use healpix_modules
integer(I4B)::  nside, reflexion
integer(I4B)::  pixel, template !  for nside <= 8192
!integer(I8B)::  pixel, template !  for any valid nside
nside = 256
pixel = 500000
call template_pixel_ring(nside, pixel, template, reflexion)
print*, template, reflexion
```

Returns in `template` the index of the template pixel (16663) whose shape matches that of the pixel #500000 for $N_{\rm side} = 256$. 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. Note that the variables `pixel` and `template` must be of type `I8B` instead of `I4B` if $N_{\rm side} > 8192$ is to be used.

RELATED ROUTINES

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

nside2templates	returns the number of template pixel shapes available for a given $N_{\rm side}$.
same_shape_pixels_ring	
same_shape_pixels_nest	return the ordered list of pixels having the same shape as a given pixel template

udgrade_nest*

Location in HEALPix directory tree: src/f90/mod/udgrade_nr.F90

Routine to degrade or prograde the pixel size of a **HEALPix** map indexed with the NESTED scheme. The degradation/progradation is done assuming an intensive quantity (like temperature) that does NOT scale with surface area.
In case of degradation, a big pixel that contains one or several bad pixels will take the average of the valid small pixels, unless a 'pessimistic' behavior is assumed in which case the big pixel will take the bad pixel sentinel value. In case of progradation, a bad pixel only spawns bad pixels.
The routine accepts both mono and bi-dimensional maps.

FORMAT

call udgrade_nest*(map_in, nside_in, map_out, nside_out*[*, *fmissval*, *pessimistic*]*)

Arguments appearing in *italic* are optional.

ARGUMENTS

name & dimensionality	kind	in/out	description
map_in(0:12*nside_in**2-1)	SP/DP	IN	mono-dimensional full sky map to be prograded or degraded.
map_in (0:12*nside_in**2-1,1:nd)	SP/DP	IN	bi-dimensional full sky map to be prograded or degraded. The routine finds the second dimension (nd) by itself.
nside_in	I4B	IN	the $N_{\rm side}$ resolution parameter of the input map. Must be a power of 2.
map_out(0:12*nside_out**2-1)	SP/DP	OUT	mono-dimensional full sky map after degradation or prograded.
map_out (0:12*nside_out**2-1,1:nd)	SP/DP	OUT	bi-dimensional full sky map after degradation or prograded. The second dimension (nd) should match that of the input map.
nside_out	I4B	IN	the $N_{\rm side}$ resolution parameter of the output map. Must be a power of 2. If nside_out > nside_in, the map is prograded (ie, more and smaller pixels) with each pixel having the same value as its parent; otherwise, the map in degraded (ie, fewer larger pixels), with each pixel being the average of its (nside_in/nside_out)^2 components.
fmissval	SP/DP	IN	sentinel value given to bad pixels in input and output maps.(**default:** HPX_SBADVAL or HPX_DBADVAL)
pessimistic	LGT	IN	if set to `.true.`, during a degradation, a big pixel containing at least a small bad pixel will be returned as bad as well, instead of taking the average of the remaing valid pixels. (**default:** `.false.`)

EXAMPLE:

```
use udgrade_nr
call udgrade_nest(map_hi, 256, map_low, 64)
```

Degrades a NESTED ordered map with $N_{\rm side} = 256$ into a NESTED map with $N_{\rm side} = 64$

RELATED ROUTINES

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

udgrade_ring	prograde or degrade a RING ordered map.

udgrade_ring*

Location in HEALPix directory tree: **src/f90/mod/udgrade_nr.F90**

Routine to degrade or prograde the pixel size of a **HEALPix** map indexed with the RING scheme. The degradation/progradation is done assuming an intensive quantity (like temperature) that does NOT scale with surface area.
In case of degradation, a big pixel that contains one or several bad pixels will take the average of the valid small pixels, unless a 'pessimistic' behavior is assumed in which case the big pixel will take the bad pixel sentinel value. In case of progradation, a bad pixel only spawns bad pixels.
The routine accepts both mono and bi-dimensional maps.

FORMAT

call udgrade_ring*(map_in, nside_in, map_out, nside_out*[*, *fmissval*, *pessimistic*]*)

Arguments appearing in *italic* are optional.

ARGUMENTS

name & dimensionality	kind	in/out	description
map_in(0:12*nside_in**2-1)	SP/DP	INOUT	mono-dimensional full sky map to be prograded or degraded. The routine finds the second dimension (nd) by itself. **Note that the map is modified on output (reordered into NESTED scheme).**
map_in (0:12*nside_in**2-1,1:nd)	SP/DP	INOUT	bi-dimensional full sky map to be prograded or degraded. **Note that the map is modified on output (reordered into NESTED scheme).**
nside_in	I4B	IN	the $N_{\rm side}$ resolution parameter of the input map. Must be a power of 2.
map_out(0:12*nside_out**2-1)	SP/DP	OUT	mono-dimensional full sky map after degradation or prograd ation.
map_out (0:12*nside_out**2-1,1:nd)	SP/DP	OUT	bi-dimensional full sky map after degradation or prograd ation. The second dimension (nd) should match that of the input map.
nside_out	I4B	IN	the $N_{\rm side}$ resolution parameter of the output map. Must be a power of 2. If nside_out > nside_in, the map is prograded (ie, more and smaller pixels) with each pixel having the same value as its parent; otherwise, the map in degraded (ie, fewer larger pixels), with each pixel being the average of its (nside_in/nside_out)^2 components.
fmissval	SP/DP	IN	sentinel value given to bad pixels in input and output maps.(**default:** HPX_SBADVAL or HPX_DBADVAL)
pessimistic	LGT	IN	if set to `.true.`, during a degradation, a big pixel containing at least a small bad pixel will be returned as bad as well, instead of taking the average of the remaing valid pixels. (**default:** `.false.`)

EXAMPLE:

```
use udgrade_nr
```

```
call udgrade_ring(map_hi, 256, map_low, 64)
```

Degrades a RING ordered map with $N_{\rm side} = 256$ into a RING map with $N_{\rm side} = 64$

RELATED ROUTINES

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

udgrade_nest	prograde or degrade a NESTED ordered map.

unfold_weightsfile

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This routine read a ring-based or pixel-based quadrature weight file and turn it into a full sky (ring-ordered) **HEALPix** map.

FORMAT

call unfold_weightsfile(w8file, w8map)

ARGUMENTS

name & dimensionality	kind	in/out	description
w8file(LEN=`filenamelen`)	CHR	IN	filename of FITS-file containing a list of ring-ordered or pixel-ordered quadrature weights for some Nside.
w8map(0:12*Nside**2-1)	SP/ DP	OUT	an array containing a full sky map of weights for the same Nside.

EXAMPLE:

```
use healpix_modules
real(DP), allocatable(:)  ::  w8map
character(len=FILENAMELEN) ::  dirname, filename
integer(I4B) ::  nside, won
won = 2
nside = 128
npix = nside2npix(nside)
allocate(w8map(0:npix-1))
dirname = get_healpix_data_dir()
filename = get_healpix_weights_file(nside, won)
filename = trim(dirname)//'/'//trim(filename)
call unfold_weightsfile(filename, w8map)
```

This code snippet looks for a pixel-based (`won=2`) weights file in standard location, with a standard name, for `nside=128`, reads it and unfolds it into a full sky ring-ordered map named `w8map`.

MODULES & ROUTINES

This section lists the modules and routines used by **unfold_weightsfile**.

pixtools	module, containing:
nside2npweights	function returning the number of pixel-based weights expected for a given Nside when stored in compact form
nside2npix	function returning the number of pixels on the full sky, for a given Nside
fitstools	module, containing:
printerror	routine for printing FITS error messages.
getsize_fits	function returning the size of a FITS file, and parsing its header.
input_map	routine to read FITS file.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

get_healpix_data_dir	routine returning actual path to `data` directory
get_healpix_weights_file	routine returning the standard name of precomputed weights file
anafast, smoothing	these two facilities use unfold_weightsfile to apply quadrature weighting to the maps they respectively analyze and smooth

uniq2nest

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

This F90 facility turns the Unique Identifier $u = p + 4N_{\rm side}^2$, into the parameter $N_{\rm side}$ (a power of 2) and the pixel index p. See "The Unique Identifier scheme" section in "**HEALPix** Introduction Document" for more details.

FORMAT

call uniq2nest(puniq, nside, pnest)

ARGUMENTS

name	kind	in/out	description
puniq	I4B/I8B	IN	The **HEALPix** Unique pixel identifier. Must be ≥ 4.
nside	I4B	OUT	The **HEALPix** $N_{\rm side}$ parameter.
pnest	I4B/I8B	OUT	(NESTED scheme) pixel identification number over the range $\{0,12N_{\rm side}^2 - 1\}$.

EXAMPLE:

```
use healpix_modules
integer(I4B) ::  nside, pnest
call uniq2nest(4, nside, pnest)
print*,nside,pnest
```

returns

1 0

since the pixel with Unique ID number 4 is the first pixel ($p = 0$) at $N_{\rm side} = 1$.

RELATED ROUTINES

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

nest2uniq — Transforms Nside and Nested pixel number into Unique **HEALPix** pixel ID number

pix2xxx, ...	to turn NESTED pixel index into sky coordinates and back

vec2ang

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

Routine to convert the 3D position vector (x, y, z) of point into its position angles (θ, ϕ) on the sphere with $x = \sin\theta\cos\phi$, $y = \sin\theta\sin\phi$, $z = \cos\theta$.

FORMAT

call vec2ang(vector, theta, phi)

ARGUMENTS

name&dimensionality	kind	in/out	description
vector(3)	DP	IN	three dimensional cartesian position vector (x, y, z). The north pole is $(0, 0, 1)$
theta	DP	OUT	colatitude in radians measured southward from north pole (in [0,π]).
phi	DP	OUT	longitude in radians measured eastward (in [0, 2π]).

RELATED ROUTINES

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

ang2vec	converts the position angles of a point on the sphere into its 3D position vector.
angdist	computes the angular distance between 2 vectors
vect_prod	computes the vector product between two 3D vectors

vect_prod

Location in HEALPix directory tree: src/f90/mod/pix_tools.F90

Returns the vectorial product of two vectors.

FORMAT

call vect_prod(v1, v2, v3)

ARGUMENTS

name & dimensionality	kind	in/out	description
v1(3)	DP	IN	cartesian vector $\mathbf{v}_1$.
v2(3)	DP	IN	cartesian vector $\mathbf{v}_2$.
v3(3)	DP	OUT	cartesian vector $\mathbf{v}_3 = \mathbf{v}_1 \times \mathbf{v}_2$

EXAMPLE:

```
use healpix_types
use pix_tools, only :  vect_prod
real(DP), dimension(3) ::  vec
real(DP) ::  one = 1.0_dp
call vect_prod((/2,0,0/)*one, (/0,1,0/)*one, vec)
print*, vec
```

will return : 0.00E+000 0.00E+000 2.00

RELATED ROUTINES

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

ang2vec	converts the position angles of a point on the sphere into its 3D position vector.
angdist	computes the angular distance between 2 vectors

vec2ang	converts the 3D position vector of point into its position angles on the sphere.

write_asctab*

Location in HEALPix directory tree: **src/f90/mod/fitstools.F90**

This routine stores a power spectrum in an ascii FITS-file. The routine can store temperature coeffecients C_ℓ^T or both temperature and polarisation coeffecients C_ℓ^T, C_ℓ^E, C_ℓ^B, $C_\ell^{T\times E}$.

FORMAT

call write_asctab*(clout, lmax, ncl, header, nlheader, filename*[, extno]*)

Arguments appearing in *italic* are optional.

ARGUMENTS

name & dimensionality	kind	in/out	description
filename(LEN=`filenamelen`)	CHR	IN	the FITS file to which the power spectrum is written.
lmax	I4B	IN	Maximum ℓ value to be written.
ncl	I4B	IN	1 for temperature coeffecients only, 4 for polarisation.
clout(0:lmax,1:ncl)	SP/ DP	IN	the powerspectrum to be saved in the file.
nlheader	I4B	IN	number of header lines to write to the file.
header(LEN=80) (1:nlheader)	CHR	IN	the header to the FITS-file.
extno	I4B	IN	extension number in which to write the data (0 based). (**default:** 0)

EXAMPLE:

```
use healpix_modules
real(SP), allocatable, dimension(:,:)  ::  cl
character(len=80), dimension(1:100) ::  header
allocate(cl(0:64,1:1))
call write_minimal_header(header,'cl',nlmax=64)
```

```
call write_asctab (cl,64,1,header,100,'cl.fits')
```

Writes a power spectrum in the array cl(0:64,1:1) to a FITS-file called 'cl.fits'. The cl array contains the temperature power spectrum C_ℓ^T up to an ℓ value of 64. 100 header lines are written to the file from the array header(1:100) which was previously filled the minimal required information for a power spectrum file.

MODULES & ROUTINES

This section lists the modules and routines used by **write_asctab***.

fitstools	module, containing:
printerror	routine for printing FITS error messages.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

alm2cl	Routine computing the power spectrum from spherical harmonics coefficients $a_{\ell m}$
fits2cl	Routine to read a FITS file created by write_asctab.
write_minimal_header	routine to write minimal FITS header

write_bintab*

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This routine creates a binary FITS-file from a **HEALPix** map. The routine can save a temperature map or both temperature and polarisation maps (T,Q,U) to the file.

FORMAT

call write_bintab*(map, npix, nmap, header, nlheader, filename*[*, extno*]*)

Arguments appearing in *italic* are optional.

ARGUMENTS

name & dimensionality	kind	in/out	description
map(0:npix-1,1:nmap)	SP/ DP	IN	the map to write to the FITS-file.
npix	I4B/ I8B	IN	Number of pixels in the map.
nmap	I4B	IN	number of maps to be written, 1 for temperature only, and 3 for (T,Q,U).
header(LEN=80) (1:nlheader)	CHR	IN	The header for the FITS-file.
nlheader	I4B	IN	number of header lines to write to the file.
filename(LEN=*)	CHR	IN	the map(s) is (are) written to a FITS-file with this filename.
extno	I4B	IN	extension number in which to write the data (0 based). (**default:** 0)

EXAMPLE:

```
call write_bintab (map,12*32**2,3,header,120,'map.fits')
```

Makes a binary FITS-file called 'map.fits' from the **HEALPix** maps (T,Q,U) in the array map(0:12*32**2-1,1:3). The number of pixels 12*32**2 corresponds to the number of pixels in a $N_{\rm side} = 32$ **HEALPix** map. The header for the FITS-file is given in the string array header and the number of lines in the header is 120.

MODULES & ROUTINES

This section lists the modules and routines used by **write_bintab***.

fitstools	module, containing:
printerror	routine for printing FITS error messages.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

input_map, read_bintab	routines which read a file created by write_bintab*.
map2alm	subroutine which analyse a map and returns the $a_{\ell m}$ coeffecients.
output_map	subroutine which calls write_bintab*
write_bintabh	subroutine to write a large array into a FITS file piece by piece
input_tod*	subroutine to read an arbitrary subsection of a large binary table
write_minimal_header	routine to write minimal FITS header

write_bintabh

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This routine is designed to write large (or huge) arrays into a binary table extension of a FITS file. The user can choose to write the array piece by piece. This is designed to deal with Time Ordered Data set (tod).

FORMAT

call write_bintabh(tod, npix, ntod, header, nlheader, filename, *[extno, firstpix, repeat]*)

Arguments appearing in *italic* are optional.

ARGUMENTS

name & dimensionality	kind	in/out	description
tod(0:npix-1,1:ntod)	SP/ DP	IN	The map or tod to write to the FITS file. It will be written in the file at the location corresponding to pixels (or time samples) `firstpix` to `firtpix + npix -1`.
npix	I8B	IN	Number of pixels or time samples in the map or TOD. See Note below.
ntod	I4B	IN	Number of maps or tods to be written. Each of them will be in a different column of the FITS binary table.
header(LEN=80) (1:nlheader)	CHR	IN	The header for the FITS file.
nlheader	I4B	IN	number of header lines to write to the file.
filename(LEN=`filenamelen`)	CHR	IN	The array is written into a FITS file with this filename.
extno	I4B	IN	extension number in which to write the data (0 based). (**default:** 0)
firstpix	I8B	IN	0 Location in the FITS file of the first pixel (or time sample) to be written (0 based). (**default:** 0). See Note below.
repeat	I4B	IN	Length of the element vector used in the binary table. (**default:** 1024 if `npix` $\propto$ 1024; 12000 if `npix` $>$ 12000 and 1 otherwise). Choosing a large `repeat` for multi-column tables (`ntod` $>$ 1) generally speeds up the I/O. It also helps bringing the number of rows of the table under 2^{31}, which is a hard limit of cfitsio. If the number of samples or pixels of each map or TOD is not a multiple of `repeat`, then the last element vector will be padded with sentinel values `HPX_SBADVAL` or `HPX_DBADVAL`.

Note : Indices and number of data elements larger than 2^{31} are only accessible in FITS files on computers with 64 bit enabled compilers and with some specific compilation options of cfitsio (see cfitsio documentation).

EXAMPLE:

```
use healpix_types
use fitstools, only :  write_bintabh
character(len=80), dimension(1:128) ::  hdr
real(SP), dimension(0:49,1) ::  tod
character(len=FILENAMELEN) ::  fname='tod.fits'
hdr(:)  = ' '
tod(:,1) = 1.
call write_bintabh(tod, 50_i8b, 1, hdr, 128, fname, firstpix=0_i8b,
repeat=10)
tod = tod * 3.
call write_bintabh(tod, 20_i8b, 1, hdr, 128, fname, firstpix=40_i8b)
```

Writes into the FITS file 'tod.fits' a 1 column binary table, where the first 40 data samples have the value 1. and the next 20 have the value 3. (Note that in this example the second call to write_bintabh overwrites some of the pixels written by the first call). The samples will be written in element vectors of length 10. The header for the FITS file is given in the string array `hdr` and its number of lines is 128.

MODULES & ROUTINES

This section lists the modules and routines used by **write_bintabh**.

fitstools	module, containing:
printerror	routine for printing FITS error messages.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

input_tod*	routine that reads a file created by write_bintabh.
input_map, read_bintab	routines to read **HEALPix** sky map,
write_minimal_header	routine to write minimal FITS header

write_dbintab

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This routine is obsolete.
To write P_{lm} polynoms into a FITS file, use write_plm instead.
To write a Healpix map into a FITS file, use write_bintab or output_map.

write_fits_cut4

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This routine writes a cut sky **HEALPix** map into a FITS file. 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

call write_fits_cut4(filename, np, pixel, signal, n_obs, serror[, *header*, *coord*, *nside*, *order*, *units*, *extno*, *polarisation*])

Arguments appearing in *italic* are optional.

ARGUMENTS

name&dimensionality	kind	in/out	description
filename(LEN=`filenamelen`)	CHR	IN	FITS file into which the cut sky map will be written
np	I4B	IN	number of pixels to be written in the file
pixel(0:np-1)	I4B	IN	index of observed (or valid) pixels
signal(0:np-1)	SP	IN	value of signal in each observed pixel
n_obs(0:np-1)	I4B	IN	number of observation per pixel
serror(0:np-1)	SP	IN	*rms* of signal in pixel, for white noise, this is $\propto 1/\sqrt{\text{n\_obs}}$.
header(LEN=80)(1:) (OPTIONAL)	CHR	IN	FITS extension header to be included in the FITS file
coord(LEN=1)	CHR	IN	astrophysical coordinates ('C' or 'Q' Celestial/eQuatorial, 'G' for Galactic, 'E' for Ecliptic)
nside	I4B	IN	**HEALPix** resolution parameter of data set
order	I4B	IN	**HEALPix** ordering scheme, 1: RING, 2: NESTED
units(LEN=20)	CHR	IN	maps units (applies only to Signal and Serror)
extno	I4B	IN	(0 based) extension number in which to write data. (**default:** 0). If set to 0 (or not set) *a new file is written from scratch*. 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 Nside, Order and Coord
polarisaton	I4B	IN	if set to a non zero value, 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. If not set, the keywords found in `header` will prevail. Note: the information relative to Nside, Order and Coord *has* to be given, either thru these keyword or via the FITS Header.

MODULES & ROUTINES

This section lists the modules and routines used by **write_fits_cut4**.

fitstools	module, containing:
printerror	routine for printing FITS error messages.
cfitsio	library for FITS file handling.

RELATED ROUTINES

This section lists the routines related to **write_fits_cut4**.

anafast	executable that reads a **HEALPix** map and analyses it.
synfast	executable that generate full sky **HEALPix** maps
getsize_fits	routine to know the size of a FITS file and its type (eg, full sky vs cut sky)
input_map	all purpose routine to input a map of any kind from a FITS file
output_map	subroutine to write a FITS file from a **HEALPix** map
read_fits_cut4	subroutine to read a **HEALPix** cut sky map from a FITS file
write_minimal_header	routine to write minimal FITS header

write_fits_partial

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This routine writes unpolarised or polarised partial sky **HEALPix** map into a FITS file.
For more information on the FITS file format supported in **HEALPix**, including the one implemented in write_fits_partial, see https://healpix.sourceforge.io/data/examples/healpix_fits_specs.pdf.

FORMAT

call write_fits_partial(filename, pixel, cutmap[, *header*, *coord*, *nside*, *order*, *units*, *extno*])

Arguments appearing in *italic* are optional.

ARGUMENTS

name & dimensionality	**kind**	**in/out**	**description**
filename(LEN=`filenamelen`)	CHR	IN	FITS file in which the partial sky map will be written
pixel(0:np-1)	I4B/ I8B	IN	index of observed (or valid) pixels
cutmap(0:np-1,1:nc)	SP/ DP	IN	value of polarised (if nc= 3) or unpolarised (if nc= 1) map value in each observed pixel
header(LEN=80)(1:) (OPTIONAL)	CHR	IN	FITS extension header to be included in the FITS file
coord(LEN=1)	CHR	IN	astrophysical coordinates ('C' or 'Q' Celestial/eQuatorial, 'G' for Galactic, 'E' for Ecliptic)
nside	I4B	IN	**HEALPix** resolution parameter of data set
order	I4B	IN	**HEALPix** ordering scheme, 1: RING, 2: NESTED
units(LEN=20)	CHR	IN	maps physical units (applies to all columns except PIXEL)
extno	I4B	IN	(0 based) extension number in which to write data. (**default:** 0). If set to 0 (or not set) *a new file is written from scratch*. 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 Nside, Order and Coord Note: the information relative to Nside, Order and Coord *has* to be given, either thru these keyword or via the FITS Header.

MODULES & ROUTINES

This section lists the modules and routines used by **write_fits_partial**.

fitstools	module, containing:
printerror	routine for printing FITS error messages.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

anafast	executable that reads a **HEALPix** map and analyses it.
synfast	executable that generate full sky **HEALPix** maps
getsize_fits	routine to know the size of a FITS file and its type (eg, full sky vs cut sky)
input_map	all purpose routine to input a map of any kind from a FITS file
output_map	subroutine to write a FITS file from a **HEALPix** map
read_fits_partial	subroutine to read a **HEALPix** partial sky map from a FITS file
write_minimal_header	routine to write minimal FITS header

write_minimal_header

Location in HEALPix directory tree: src/f90/mod/head_fits.F90

This routine writes the baseline FITS header for the most common **HEALPix** data sets: (cut sky or full sky) map, $C(\ell)$ power spectra and $a_{\ell m}$ coefficients.

FORMAT

call write_minimal_header(header, dtype, *[append, nside, order, ordering, coordsys, creator, version, randseed, beam_leg, fwhm_degree, units, nlmax, polar, nmmax, bcross, deriv, asym_cl]*)

Arguments appearing in *italic* are optional.

ARGUMENTS

name & dimensionality	kind	in/out	description
header(LEN=80) DIMENSION(:)	CHR	INOUT	The FITS header to fill in.
dtype(LEN=*)	CHR	IN	data to be put in the FITS file, must be one of 'ALM', 'CL', 'MAP', 'CUTMAP' (case un-sensitive).

name & dimensionality	kind	in/out	description
append	LGT	IN	if set to TRUE, the keywords will be appended to the content of `header` instead of written from scrath
nside	I4B	IN	map resolution parameter; required for dtype='MAP' and dtype='CUTMAP'
order	I4B	IN	map ordering, either 1 (=ring) or 2 (=nested); see `ordering`
ordering(LEN=)*	CHR	IN	map ordering, either 'RING' or 'NESTED' (case un-sensitive); either `order` or `ordering` is required for dtype='MAP' and dtype='CUTMAP'
coordsys(LEN=)*	CHR	IN	map coordinate system; Valid choices are 'G' = Galactic, 'E' = Ecliptic, 'C'/'Q' = Celestial = eQuatorial
creator(LEN=)*	CHR	IN	name of software generating the data set
version(LEN=)*	CHR	IN	version of `creator` software
randseed	I4B	IN	random number generator seed used to generate the data
beam_leg(LEN=)*	CHR	IN	File containing Legendre transform of symmetric beam
fwhm_degree	DP	IN	FWHM in degrees of gaussian symmetric beam (FITS keyword: `FWHM`)
units(LEN=)*	CHR	IN	physical units of the data set (FITS keyword: `TUNIT*`)
nlmax	I4B	IN	maximum multipole order l of the data set (FITS keyword: `MAX-LPOL`)
polar	LGT	IN	if set to `.TRUE.`, the file to be written contains polarized data
nmmax	I4B	IN	maximum degree m of data set (FITS keyword: `MAX-MPOL`)
bcross	LGT	IN	if set to `.TRUE.`, the magnetic cross terms power spectra (TB and EB) are included; only applies to dtype='CL'
deriv	I4B	IN	order of derivatives to included in FITS file (0, 1 or 2); only applies to dtype='MAP'
asym_cl	LGT	IN	if set to `.TRUE.`, the asymmetric power spectra (ET, BT and BE on top of TE, TB and EB) are included; only applies to dtype='CL'

EXAMPLE:

```
use healpix_types
use head_fits
character(len=80), dimension(1:60) ::  header
call write_minimal_header(header, 'MAP', nside=256, ordering='Nested')
call add_card(header, 'HISTORY', 'Dummy map')
```

Writes in `header` a **HEALPix** compliant FITS header for a $N_{\rm side} = 256$ map with NESTED ordering. Further HISTORY information is added with add_card

MODULES & ROUTINES

This section lists the modules and routines used by **write_minimal_header**.

write_hl	more general routine for adding a keyword to a header.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

add_card	general purpose routine to write/edit an arbitrary keyword into a FITS file header.
get_card	general purpose routine to read any keywords from a header in a FITS file.
del_card	routine to discard a keyword from a FITS header
read_par, number_of_alms	routines to read specific keywords from a header in a FITS file.
getsize_fits	function returning the size of the data set in a fits file and reading some other useful FITS keywords
merge_headers	routine to merge two FITS headers

write_plm

Location in HEALPix directory tree: src/f90/mod/fitstools.F90

This routine creates a double precision binary FITS-file from a given array. The routine is used by the **HEALPix** facility plmgen to store precomputed $P_{\ell m}(\theta)$.

FORMAT

call write_plm(plm, nplm, nhar, header, nlheader, filename, nsmax, nlmax)

ARGUMENTS

name&dimensionality	kind	in/out	description
plm(0:nplm-1,1:nhar)	DP	IN	the array with the precomputed $P_{\ell m}(\theta)$ values.
nplm	I4B	IN	Number of $P_{\ell m}$ values to store.
nhar	I4B	IN	1 for scalar $P_{\ell m}$ only and 3 for tensor harmonics.
header(LEN=80) (1:nlheader)	CHR	IN	The header for the FITS-file.
nlheader	I4B	IN	number of header lines to write to the file.
filename(LEN=`filenamelen`)	CHR	IN	the precomputed $P_{\ell m}(\theta)$ values are written to this file.
nsmax	I4B	IN	$N_{\rm side}$ for the precomputed $P_{\ell m}$s.
nlmax	I4B	IN	maximum ℓ value for the precomputed $P_{\ell m}$s.

EXAMPLE:

```
call write_plm (plm, 65*66*32, 1, header, 120, 'plm_32.fits', 32, 64)
```

Makes a double precision binary FITS-file called 'plm_32.fits' from the precomputed $P_{\ell m}(\theta)$ in the array plm(0:65*66*32-1,1:1). The number 65*66*32 corresponds to the number of precomputed $P_{\ell m}$s needed for a $N_{\rm side} = 32$ **HEALPix** map synthesis/analysis. The header for the FITS-file is given in the string array header and the number of lines in the header is 120.

MODULES & ROUTINES

This section lists the modules and routines used by **write_plm**.

fitstools	module, containing:
printerror	routine for printing FITS error messages.
cfitsio	library for FITS file handling.

RELATED ROUTINES

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

read_dbintab, read_bintab	routines which reads a file created by write_plm.
map2alm, alm2map	routines using precomputed $P_{\ell m}(\theta)$.

xcc_v_convert

Location in HEALPix directory tree: src/f90/mod/coord_v_convert.f90

This routine rotates a 3D coordinate vector from one astronomical coordinate system to another.

FORMAT

call xcc_v_convert(ivector, iepoch, oepoch, isys, osys, ovector)

ARGUMENTS

name & dimensionality	kind	in/out	description
ivector(1:3)	DP	IN	3D coordinate vector of one astronomical object, in the input coordinate system.
iepoch	DP	IN	epoch of the input astronomical coordinate system.
oepoch	DP	IN	epoch of the output astronomical coordinate system.
isys(len=*)	CHR	IN	input coordinate system, should be one of 'E'=Ecliptic, 'G'=Galactic, 'C'/'Q'=Celestial/eQuatorial.
osys(len=*)	CHR	IN	output coordinate system, same choice as above.
ovector(1:3)	DP	IN	3D coordinate vector of the same object, in the output coordinate system.

EXAMPLE:

```
use healpix_types
use coord_v_convert, only:  xcc_v_convert
real(dp) ::  vecin(1:3), vecout(1:3)
vecin = (/ 0_dp, 0_dp, 1_dp /)
call xcc_v_convert(vecin, 2000.0_dp, 2000.0_dp, 'g', 'c', vecout)
```

Will produce in `vecout` the location in Celestial coordinates (2000 epoch) of the North Galactic Pole (defined in `vecin`)

RELATED ROUTINES

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

coordsys2euler_zyz	produces the Euler angles ψ, θ, φ in (Z,Y,Z) convention for rotation between standard astronomical coordinate systems.
ang2vec, vec2ang	Routine to convert spherical coordinates (co-latitude and longitude) into 3D vector coordinates and vice-versa.