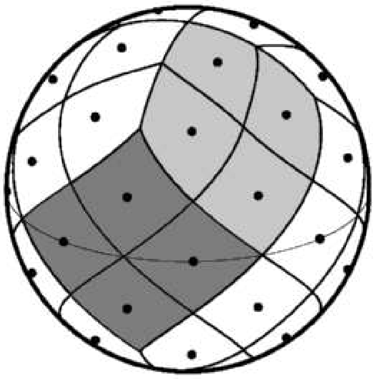
HEALPix Fortran90 Subroutines Overview



Revision: Version 2.00; August 31, 2005

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.

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.
$\mathbf{N_{side}}$	**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 release 1.2 and 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:** fatal_error, assert, assert_present, assert_not_present, assert_alloc, file_present, assert_directory_present, 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)

Changes between release 1.1 and 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).

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_{lm} 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_{lm} 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_{lm} in a FITS file.

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)

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 to give to the keyword.
comment(LEN=*)	CHR	IN	comment to the keyword.

EXAMPLE:

```
call add_card(header,'NSIDE',256,'the nside of the map')
```

Gives the keyword ‘NSIDE’ the value 256 in the given header-string.

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**.

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

name & dimensionality	kind	in/out	description
nside	I4B	IN	value of N_{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 (OPTIONAL)	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 $a_{\ell m}$. $C_{12}(\ell) = \sum_{m=-\ell}^{\ell} a_{1,\ell m} a^*_{2,\ell m} / (2\ell + 1)$

FORMAT

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

ARGUMENTS

name & dimensionality	kind	in/out	description
nlmax	I4B	IN	the maximum ℓ value used for the a_{lm}.
nmmax	I4B	IN	the maximum m value used for the a_{lm}.
alm1(1:p, 0:nlmax, 0:nmmax)	SPC/ DPC	IN	First set of a_{lm} 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) (OPTIONAL)	SPC/ DPC	IN	Second set of a_{lm} 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^T will be output. If $d = 4$ and $p = 3$, the output will be $C_l^T, C_l^E, C_l^B, C_l^{T\times E}$, and if $d \geq 6$ and $p = 3$, $C_l^{T\times B}$ $C_l^{E\times B}$ will also be output.

EXAMPLE:

```
lmax = 128 ; mmax = lmax
call alm2cl(lmax, mmax, alm1, cl_auto)
call alm2cl(lmax, mmax, alm1, alm2, cl_cross)
```

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

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** temperature map (and if specified, polarisation maps) from input a^T_{lm} (and if specified a^E_{lm} and a^B_{lm}) values. The different routines are called dependent on what parameters are passed. Some routines synthesize maps with or without precomputed harmonics 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])

ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	the N_{side} value of the map to synthesize.
nlmax	I4B	IN	the maximum ℓ value used for the a_{lm}.
nmmax	I4B	IN	the maximum m value used for the a_{lm}.
alm_TGC(1:p, 0:nlmax, 0:nmmax)	SPC or DPC	IN	The a_{lm} 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 $P_{lm}(\theta)$ 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 $P_{lm}(\theta)$ AND precomputed tensor harmonics are used instead of recursion. (n_plm = nsmax*(nmmax+1)*(2*nlmax-nmmax+2)

EXAMPLE:

```
use healpix_types
use pix_tools, only :  nside2npix
use alm_tools, only :  alm2map
integer(I4B) ::  nside, lmax, mmax, npix, n_plm
real(SP), dimension(:,:), allocatable ::  map
complex(SPC), dimension(:,:,:), allocatable ::  alm
real(DP), dimension(:,:), allocatable ::  plm
...
nside=256 ; lmax=512 ; mmax=lmax
npix=nside2npix(nside)
n_plm=nside*(mmax+1)*(2*lmax-mmax+2)
allocate(alm(1:3,0:lmax,0:mmax))
allocate(map(0:npix-1,1:3))
allocate(plm(0:n_plm-1,1:3))
...
call alm2map(nside, lmax, mmax, alm, map, plm)
```

Make temperature and polarisation maps from the scalar and tensor a_{lm} passed in alm. The maps have N_{side} of 256, and are constructed from a_{lm} values up to 512 in ℓ and m. Since the optional plm array is passed with both precomputed $P_{lm}(\theta)$ AND tensor harmonics, there will be no recursions in the routine and execution will be faster.

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

RELATED ROUTINES

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

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

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, der1 and der2. The precision of these arrays should match.

FORMAT

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

ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	the N_{side} value of the map to synthesize.
nlmax	I4B	IN	the maximum ℓ value used for the a_{lm}.
nmmax	I4B	IN	the maximum m value used for the a_{lm}.
alm(1:p, 0:nlmax, 0:nmmax)	SPC/ DPC	IN	The a_{lm} 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 $\left(\partial^2 T/\partial\theta^2, \partial^2 T/\partial\theta\partial\phi/\sin\theta, \partial^2 T/\partial\phi^2/\sin^2\theta\right)$ or $\left(\partial^2 T/\partial\theta^2, \partial^2 Q/\partial\theta^2, \partial^2 Q/\partial\theta^2, \ldots\right)$

EXAMPLE:

```
use healpix_types
use pix_tools, only :  nside2npix
use alm_tools, only :  alm2map_der
integer(I4B) ::  nside, lmax, mmax, npix, n_plm
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(1:3,0:lmax,0:mmax))
allocate(map(0:npix-1,1:3))
allocate(der1(0:npix-1,1:2), der2(0:npix-1,1:3))
...
call alm2map_der(nside, lmax, mmax, alm, map, der1, der2)
```

Make temperature maps and its derivatives from the a_{lm} passed in alm. The maps have N_{side} of 256, and are constructed from a_{lm} 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 $Y_{\ell m}$ 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 $a_{\ell m}$
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

ALMS2FITS*

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

This routine stores a_{lm} 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_{lm} values and real/imaginary standard deviation. One can store temperature a_{lm} or temperature and polarisation, a^T_{lm}, a^E_{lm} and a^B_{lm}. 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_{lm}, a^T_{lm}, a^E_{lm} and a^B_{lm} 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_{lm} in.
nalms	I4B	IN	number of a_{lm} 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_{lm} to write to the file. alms(i,1,j) and alms(i,2,j) contain the ℓ and m values for the ith a_{lm} (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_{lm}. Finally, the standard deviation for the ith a_{lm} 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_{lm}, a^E_{lm} and a^B_{lm} 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_{lm}), imaginary(a_{lm}) for each of the a_{lm}. The number 65*66/2 is the number of a_{lm} 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_{lm} from a FITS file
dump_alms	has the same function as alms2fits* but with parameters passed differently.

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**.

vec2ang	converts the 3D position vector of point into its position angles on the sphere.

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**.

vect_prod	computes the vector product between two 3D vectors

ASSERT, ASSERT_ALLOC, …

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.

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

FUNCTIONS:

`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_present(filename)`

issues an error message and stops the code execution if the file named `filename` 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.

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.

COMPLEX_FFT

Location in HEALPix directory tree: src/f90/mod/healpix_fft.F90 or src/f90/mod/healpix_fftw.F90 (module healpix_fft in either case)

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

name & dimensionality	kind	in/out	description
data(:)	SP/ DP	IN	data set
stats	tstats	OUT	structure containing the statistics of the data. The respective fields (stats%*field*) are:
ntot	I4B	–	total number of data points
nvalid	I4B	–	number n of valid data points
mind, maxd	DP	–	minimum and maximum valid data
average	DP	–	average of valid points $m = \sum x/n$
absdev	DP	–	absolute deviation $a = \sum \lvert x-m \rvert /n$
var	DP	–	variance $\sigma^2 = \sum (x-m)^2/(n-1)$
rms	DP	–	standard deviation σ
skew	DP	–	skewness factor $s = \sum (x-m)^3/(n\sigma^3)$
kurt	DP	–	kurtosis factor $k = \sum (x-m)^4/(n\sigma^4) - 3$
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 inplace, 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_{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 $N_{side} = 256$ 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_{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 $N_{side} = 256$ 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 in-place. 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_{lm} 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_{lm} coefficients.

CREATE_ALM*

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

This routine generates scalar (and tensor) a_{lm} for a temperature (and polarisation) power spectrum read from an input FITS file. The a_{lm} 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, iseed, fwhm_arcmin, alm_TGC, header [, windowfile, units, beam_file, rng_handle])

ARGUMENTS

name & dimensionality	kind	in/out	description
nsmax	I4B	IN	N_{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 ($MAX = 3 \times N_{side}$).
nmmax	I4B	IN	maximum m value for the $a_{\ell m}$.
polar	I4B	IN	equals 1 if polarisation is used, 0 otherwise.
filename(LEN=`filenamelen`)	CHR	IN	name of FITS file containing power spectrum.
rng_handle	planck_rng	IN	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 powerspectrum 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.
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`) (OPTIONAL)	CHR	IN	full filename specification of the FITS file with the pixel window function.
units(LEN=80),dimension(1:) (OPTIONAL)	CHR	OUT	physical units of the created $a_{\ell m}$ (square-root of the input power spectrum units).
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`.

EXAMPLE:

```
use alm_tools, only:  create_alm
use rngmod, only:  rand_init, rng_handle
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_{lm} from the power spectrum given in the file 'cl.fits'. The map to be created from these a_{lm} is assumed to have $N_{side} = 64$. C_ls from the power spectrum are used up to an ℓ value of 128. Corresponding a_{lm} 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_{lm} 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_{lm}. For this purpose, `header` is updated with FITS format information describing the origin and history of these a_{lm}.

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_{lm} created by create_alm* to a **HEALPix** map.
alms2fits, dump_alms	Routines to save a set of a_{lm} 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

DUMP_ALMS*

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

This routine stores a_{lm} 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_{lm} values. One can store temperature a_{lm} or polarisation, a^E_{lm} or a^B_{lm}. 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_{lm} in.
nlmax	I4B	IN	maximum ℓ value to store.
alms(0:nlmax,0:nlmax)	SP/ DP	IN	array with a_{lm}. alms(l,m) corresponds to a_{lm}
extno	I4B	IN	extension number. If 1 is specified, a FITS file is created and a_{lm} is stored in the first FITS extension as temperature a_{lm}. If 2 or 3 is specified, an already existing file is opened and a 2nd or 3rd extension is created, treating a_{lm} as a^E_{lm} or a^B_{lm}.
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, 80, 2)
```

Opens an already existing FITS file which contains temperature a_{lm}. An extra extension is added to the file where the a_{lm} array are written in a three-column format as described above. 80 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_{lm} from a FITS-file
alms2fits	has the same function as dump_alms* but is more general.

FITS2ALMS*

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

This routine reads a_{lm} 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_{lm} values and real/imaginary standard deviation. One can read temperature a_{lm} or temperature and polarisation, a^T_{lm}, a^E_{lm} and a^B_{lm}.

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_{lm} from.
nalms	I4B	IN	number of a_{lm} 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_{lm} to read from the file. alms(i,1,j) and alms(i,2,j) contain the ℓ and m values for the ith a_{lm} (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_{lm}. Finally, the standard deviation for the ith a_{lm} 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_{lm}, a^E_{lm} and a^B_{lm} 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_{lm}), imaginary(a_{lm}) for each of the a_{lm}. The number 65*66/2 is the number of a_{lm} 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_{lm} in a FITS-file
read_conbintab	has the same function as fits2alms* but with parameters passed differently.

FITS2CL*

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

This routine reads a power spectrum from a FITS ASCII or binary table. The routine can read temperature coeffecients C_l^T or both temperature and polarisation coeffecients C_l^T, C_l^E, C_l^B, $C_l^{T\times E}$. 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.

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:

```
call fits2cl ('cl.fits',cl,64,4,header,units)
```

Reads a power spectrum from the FITS file ‘cl.fits’ and stores the result in cl(0:64,1:4) which are the C_l coeffecients up to $l = 64$ for $(T, E, B, T\times E)$. 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.

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

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`.

EXAMPLE:

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

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

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
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

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.

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.

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

name & dimensionality	kind	in/out	description
var	I8B	OUT	number of pixels or time samples in the fits file
filename(LEN=`filenamelen`)	CHR	IN	filename of the FITS-file containing **HEALPix** map(s).
nmaps (OPTIONAL)	I4B	OUT	number of maps in the file.
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=`filenamelen`) (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 pixelisation 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
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_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; if so, demote it 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 `I4B` 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	π	
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.

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_{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	OUT	The pixel indexes in the slice.
nir	I4B	OUT	the number of pixels in the slice.
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

FORMAT

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

ARGUMENTS

name & dimensionality	kind	in/out	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	IN	number of pixels in the full sky map
nmaps	I4B	IN	number of maps in the file
fmissval (OPTIONAL)	SP/ DP	IN	value to be given to missing pixels, its default value is 0
header(LEN=80)(1:) (OPTIONAL)	CHR	OUT	FITS extension header
units(LEN=20)(1:nmaps) (OPTIONAL)	CHR	OUT	maps units
extno (OPTIONAL)	I4B	IN	extension number to read the data from (0 based).(**default:** 0) (the first extension is read)

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)
```

Reads into `map` the content of the FITS file 'map.fits'

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
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

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:) (OPTIONAL)	CHR	OUT	FITS extension header
firstpix (OPTIONAL)	I8B	IN	first pixel (or time sample) to read from (0 based). (**default:** 0). See Note below.
fmissval (OPTIONAL)	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

MAP2ALM*

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

This routine is a wrapper to 5 other routines:map2alm_sc, map2alm_sc_pre, map2alm_pol, map2alm_pol_pre1, map2alm_pol_pre2. These routines analyse a **HEALPix** map and return a^T_{lm} (and if specified a^E_{lm} and a^B_{lm}) values up to the desired order in ℓ (maximum 3*N_{side}). The different routines are called dependent 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_{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_{lm} 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)	DP	IN	section of the map on which to perform the a_{lm} analysis, expressed in terms of $z = \sin(\mathrm{latitude}) = \cos(\theta)$. If zbounds(1)<zbounds(2), the analysis is performed *on* the strip zbounds(1)< z <zbounds(2); if not, it is performed *outside* of the strip zbounds(2)< z <zbounds(1).
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).
plm(0:(nlmax+1)*(nlmax+2)*nsmax-1), OPTIONAL	DP	IN	If this optional matrix is passed with this rank, precomputed $P_{lm}(\theta)$ are used instead of recursion.
plm(0:(nlmax+1)*(nlmax+2)*nsmax-1, 1:3), OPTIONAL	DP	IN	If this optional matrix is passed with this rank, precomputed $P_{lm}(\theta)$ AND precomputed tensor harmonics are used instead of recursion.

EXAMPLE:

```
real(dp), allocatable, dimension(:,:)  ::  dw8
allocate(dw8(1:512,1:3))
dw8 = 1.0_dp
z = sin(10.0_dp * PI/180.0_dp)
call map2alm(256, 512, 512, map(0:12*256**2-1,1:3), alm(1:3,0:512,0:512),
(\ z, -z \) , dw8, plm(0:513*514*256-1)
```

Analyses temperature and polarisation maps passed in map. The map has an N_{side} of 256, an the analyses is supposed to be performed up to 512 in ℓ and m. The resulting a_{lm} coeffecients for temperature and polarisation are returned in alm. A 10° cut on each side of the equator is applied. Uniform weights are used. Since the optional plm array is passed, precomputed $P_{lm}(\theta)$ are used, but only scalar ones because of the rank of the array. The tensor harmonics are still computed with a recursion.

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

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*.

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/
indmed	routine 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_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_{side} and ℓ_{max}, since each processor only needs a fraction $(1/N_{\mathrm{procs}})$ 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_{side} value of the HEALPix map. (OPTIONAL)
nlmax	I4B	IN	the maximum ℓ value used for the a_{lm}. (OPTIONAL)
nmmax	I4B	IN	the maximum m value used for the a_{lm}. (OPTIONAL)

zbounds(1:2)	DP	IN	section of the map on which to perform the a_{lm} analysis, expressed in terms of $z = \sin(\mathrm{latitude}) = \cos(\theta)$. If zbounds(1)<zbounds(2), the analysis is performed *on* the strip zbounds(1)< z <zbounds(2); if not, it is performed *outside* of the strip zbounds(2)< z <zbounds(1). (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 $P^{\mathrm{T}}_{\ell m}$; 2 = pre-compute $P^{\mathrm{T}}_{\ell m}$ and $P^{\mathrm{P}}_{\ell m}$. (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), the analysis is performed *on* the strip zbounds(1)$<$ z $<$zbounds(2); if not, it is performed *outside* of the strip zbounds(2)$<$ z $<$zbounds(1). (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.

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 southwestern 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_{side} parameter of the map.
ipix	I4B	IN	The pixel number of the central pixel.
list(8)	I4B	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 ::  n, 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.

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	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 $\{1,\ldots,8192\}$), $N_{\rm side} = \sqrt{N_{\rm pix}/12}$ is returned; if not, an error message is issued and -1 is returned.

EXAMPLE:

```
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	I4B	OUT	the number of pixels $N_{\rm pix}$ of the map. If $N_{\rm side}$ is valid (a power of 2 in $\{1,\ldots,8192\}$), $N_{\rm pix} = 12{N_{\rm side}}^2$ is returned; if not, an error message is issued and -1 is returned.

EXAMPLE:

```
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.

NSIDE2NTEMPLATES

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

Function to provide the number of template pixels

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

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	I4B	OUT	the number of template pixels $N_{\rm templates}$.

EXAMPLE:

```
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_{lm} values stored in each FITS extension in a FITS file containing a_{lm}

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_{lm} 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_{lm} 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.

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=`filenamelen`)	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) ::  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 output in 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

PARSE_XXX

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	IN	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.

ROUTINES:

`handle = parse_init (fname)`

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

RELATED ROUTINES

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

concatnl	generates from a set of strings the multi-line description

PIXEL_WINDOW

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

This routine returns the ℓ space window function (for temperature and polarisation) associated to **HEALPix** pixels of resolution parameter $N_{\rm side}$. Unless specified otherwise, the files Healpix/data/pixel_window_n????.fits are used.

FORMAT

call pixel_window(pixlw, nside [, windowfile])

ARGUMENTS

name & dimensionality	kind	in/out	description
pixlw(0:lmax,1:p)	DP	OUT	pixel window function generated. The first index must be $\ell_{\rm max} \leq 4N_{\rm side}$. 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.
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

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	—	pixel identification number in NESTED scheme over the range $\{0,N_{\rm pix}-1\}$.
ipring	I4B	—	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.

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.

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 λ_{lm}.
nmmax	I4B	IN	The maximum degree m of the generated λ_{lm}.
plm(0:n_plm-1, 1:p)	DP	OUT	The λ_{lm} 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 $\lambda_{00}(\theta_1)$, $\lambda_{10}(\theta_1)$, $\lambda_{20}(\theta_1)$, $\ldots$, $\lambda_{11}(\theta_1)$, $\lambda_{21}(\theta_1)$; $\ldots$, $\lambda_{00}(\theta_2)$ $\ldots$

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 $N_{side} = 256$, 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_{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	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	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:

```
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 $N_{side} = 256$ 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

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_{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	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	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:

```
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 $N_{side} = 256$ 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

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_{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	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	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 disc 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 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_{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	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	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 $N_{side} = 256$ 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 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**.

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 $2^{128} - 1 \approx 3.4 10^{38}$. 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**.

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**.

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

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.
nmap	I4B	IN	number of maps to be read, 1 for temperature only, and 3 for (T,Q,U).
map(0:npixtot-1,1:nmap)	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) (OPTIONAL)	CHR	OUT	character string array containing the physical units of each map read
extno (OPTIONAL)	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 file, 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_{lm} 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_{lm} values and real/imaginary standard deviation on these a_{lm}. It is supposed to contain either 1 or 3 extension(s) containing respectively the a_{lm} 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_{lm}.
nalms	I4B	IN	Number of a_{lm} values to read from the file.
alms(0:nalms-1,1:6)	SP/ DP	OUT	the a_{lm} read from the file. alms(i,1) and alms(i,2) contain the ℓ and m values for the ith a_{lm} . alms(i,3) and alms(i,4) contain the real and imaginary value of the ith a_{lm} . Finally, the standard deviation for the ith a_{lm} 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_{lm} needed to fill the whole range from l=0 to l=64) a_{lm} 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_{lm} to a FITS-file
fits2alms	has the same function as read_conbintab but is more general.

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 $P_{lm}(\theta)$ 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_{lm}s and pixel windows, 3 for scalar and tensor P_{lm}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 $P_{lm}(\theta)$ 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 $P_{lm}(\theta)$ for a $N_{side} = 32$, $lmax = 64$ 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 $P_{lm}(\theta)$.
write_dbintab	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

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) (OPTIONAL)	I4B	OUT	number of observation per pixel
serror(0:np-1) (OPTIONAL)	SP	OUT	*rms* of signal in pixel. (For white noise, this would be $\propto 1/\sqrt{\text{n\_obs}}$)
header(LEN=80)(1:) (OPTIONAL)	CHR	OUT	FITS extension header
units(LEN=20) (OPTIONAL)	CHR	OUT	maps units (applies only to Signal and Serror, which are assumed to have the same units)
extno (OPTIONAL)	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_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_{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_{lm} or precomputed $P_{lm}(\theta)$

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_{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 or src/f90/mod/healpix_fftw.F90 (module healpix_fft in either case)

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.

FORMAT

call remove_dipole*(nside, map, ordering, degree, multipoles, zbounds [, fmissval, mask])

ARGUMENTS

name & dimensionality	**kind**	**in/out**	**description**
nside	I4B	IN	value of N_{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 $z = \sin(\mathrm{latitude}) = \cos(\theta)$. If zbounds(1)<zbounds(2), the fit is performed *on* the strip zbounds(1)$< z <$zbounds(2); if not, the fit is performed *outside* of the strip zbounds(2)$< z <$zbounds(1).
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 $\lvert\mathrm{mask}\rvert < 10^{-10}$ are not used for fit. Note: the map is *not* multiplied by the mask.

EXAMPLE:

```
s = sin(15.0_dp * PI / 180.0_dp)
call remove_dipole*(128, map, 1, 2, multipoles, (\ s, -s \) )
```

Will compute and remove the best fit monopole and dipole from a map with $N_{\mathrm{side}} = 128$ 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,dataou

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_{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,datao

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 $\ell \leq \ell_{\max}$ the number of operations scales like $\ell^3_{\max}$.

FORMAT

call rotate_alm*(lmax, 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_{lm} for $\ell_{\rm max} = m_{\rm max} = 64$ 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_{lm} 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_{lm} in a FITS file.

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

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

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	OUT	identification number(s) of the template matching in shape the pixel(s) provided (the numbering scheme of the pixel templates is the same for both routines).
list(0:nrep-1) OPTIONAL	I4B	OUT	pointer containing the ordered list of NESTED/RING scheme identification numbers (in $\{0,12{N_{\rm side}}^2-1\}$) 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	OUT	number of pixels having the same template (either 8, 16, $4N_{\rm side}$ or $8N_{\rm side}$).

EXAMPLE:

```
call same_shape_pixels_ring(256, 1234, list, reflexion, np)
```

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

RELATED ROUTINES

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

nside2templates	returns the number of template pixel shapes available

	for a given N_{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

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	I4B/ SP/ DP	IN	number 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

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

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	IN	NESTED scheme pixel identification number over the range $\{0,12{N_{\rm side}}^2-1\}$.
pixel_ring	I4B	IN	RING scheme pixel identification number over the range $\{0,12{N_{\rm side}}^2-1\}$.
template	I4B	OUT	identification number(s) of the template matching in shape the pixel(s) provided (the numbering scheme of the pixel templates is the same for both routines).
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:

```
call template_pixel_ring(256, 500000, template, reflexion)
```

Returns in `template` the index of the template pixel (16663) whose shape matches that of the pixel #500000 for $N_{\rm side} = 256$. 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.

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

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_{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 progradation.
map_out(0:12*nside_out**2-1,1:nd)	SP/ DP	OUT	bi-dimensional full sky map after degradation or progradation. The second dimension (nd) should match that of the input map.
nside_out	I4B	IN	the N_{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 $(\mathrm{nside\_in}/\mathrm{nside\_out})^2$ components.
fmissval (OPTIONAL)	SP/ DP	IN	sentinel value given to bad pixels in input and output maps. (**default:** $-1.6375\ 10^{30}$)
pessimistic (OPTIONAL)	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:

```
call udgrade_nest(map_hi, 256, map_low, 64)
```

Degrades a NESTED ordered map with $N_{side} = 256$ into a NESTED map with $N_{side} = 64$

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

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_{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 progradation.
map_out(0:12*nside_out**2-1,1:nd)	SP/ DP	OUT	bi-dimensional full sky map after degradation or progradation. The second dimension (nd) should match that of the input map.
nside_out	I4B	IN	the N_{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 $(\text{nside\_in/nside\_out})^2$ components.
fmissval (OPTIONAL)	SP/ DP	IN	sentinel value given to bad pixels in input and output maps. (**default:** $-1.6375\ 10^{30}$)
pessimistic (OPTIONAL)	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:

```
call udgrade_ring(map_hi, 256, map_low, 64)
```

Degrades a RING ordered map with $N_{side} = 256$ into a RING map with $N_{side} = 64$

RELATED ROUTINES

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

udgrade_nest	prograde or degrade a NESTED ordered map.

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.

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**.

angdist	computes the angular distance between 2 vectors

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, $C_l^{T\times E}$.

FORMAT

call write_asctab*(clout, lmax, ncl, header, nlheader, filename)

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.

EXAMPLE:

```
call write_asctab (cl,64,1,header,80,'cl.fits')
```

Writes a powerspectrum in the array cl(0:64,1:1) to a FITS-file called ‘cl.fits’. The cl array contains the temperature powerspectrum C_l^T up to an ℓ value of 64. 80 header lines are written to the file from the array header(1:80).

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_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

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	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=`filenamelen`)	CHR	IN	the map(s) is (are) written to a FITS-file with this filename.
extno OPTIONAL	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 $N_{side} = 32$ **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_{lm} 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_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

name & dimensionality	**kind**	**in/out**	**description**
tod(0:npix-1,1:ntod)	SP	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.
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 (OPTIONAL)	I4B	IN	extension number in which to write the data (0 based). (**default:** 0)
firstpix (OPTIONAL)	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 (OPTIONAL)	I4B	IN	length of the element vector used in the binary table. (**default:** 1)024 if `npix` $\propto$ 1024, 12000 is `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.

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
hdr(:)  = ' '
tod(:,1) = 1.
call write_bintabh (tod, 50_i8b, 1, header, 128, 'tod.fits',
firstpix=0_i8b, repeat=10)
tod = tod * 3.
call write_bintabh (tod, 20_i8b, 1, header, 128, 'tod.fits',
```

`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_bintabhoverwrites 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 the number of lines in the header 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_DBINTAB

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

This routine is obsolete. Use write_plm instead.

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

FORMAT

call write_fits_cut4(filename, np, pixel, signal , n_obs, serror[, header, coord, nside, order, units])

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 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 $\propto 1/\sqrt{\mathrm{n\_obs}}$.
header(LEN=80)(1:) (OPTIONAL)	CHR	IN	FITS extension header
coord(LEN=1) (OPTIONAL)	CHR	IN	astrophysical coordinates ('C' or 'Q' Celestial/eQuatorial, 'G' for Galactic, 'E' for Ecliptic)
nside (OPTIONAL)	I4B	IN	**HEALPix** resolution parameter of data set
order (OPTIONAL)	I4B	IN	**HEALPix** ordering scheme, 1: RING, 2: NESTED
header(LEN=80) (OPTIONAL)	CHR	IN	FITS header to be included in the FITS file
units(LEN=20) (OPTIONAL)	CHR	IN	maps units (applies only to Signal and Serror)
			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_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 $P_{lm}(\theta)$.

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 $P_{lm}(\theta)$ values.
nplm	I4B	IN	Number of P_{lm} values to store.
nhar	I4B	IN	1 for scalar P_{lm} 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 $P_{lm}(\theta)$ values are written to this file.
nsmax	I4B	IN	N_{side} for the precomputed P_{lm}s.
nlmax	I4B	IN	maximum ℓ value for the precomputed P_{lm}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 $P_{lm}(\theta)$ in the array plm(0:65*66*32-1,1:1). The number 65*66*32 corresponds to the number of precomputed P_{lm} s needed for a $N_{side} = 32$ **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 $P_{lm}(\theta)$.

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**.

ang2vec, vec2ang	Routine to convert spherical coordinates (co-latitude and longitude) into 3D vector coordinates and vice-versa.