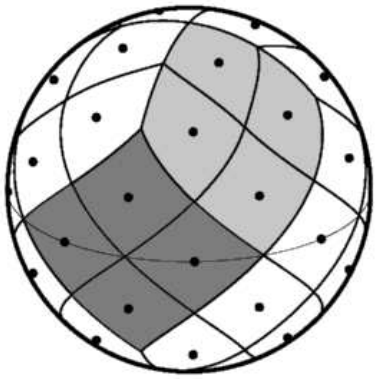
HEALPix Facility Installation Guidelines



Revision: Version 3.80; June 22, 2021

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

Abstract: This document describes the installation for the **HEALPix** facilities.

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

TABLE OF CONTENTS

1 Introduction

In this document the installation procedure for the **HEALPix** distribution is outlined. **HEALPix** comprises a suite of Fortran 90, C++, IDL, Java and Python routines providing both stand-alone facilities and callable subroutines as an alternative for those users who wish to build their own tools. A set of C subroutines and functions is also provided.

The distribution can be downloaded as a gzipped tarred file, *or* as a zipped file, which can respectively be unpacked by executing the commands[1]

```
% gunzip Healpix_3.80.tar.gz
% tar -xpf Healpix_3.80.tar
```

or

```
% tar -xzpf Healpix_3.80.tar.gz
```

or

```
% unzip Healpix_3.80.zip
```

creating a directory named `Healpix_3.80` whose structure is shown in Figure 1.

As with most freely available software, the distribution comes with caveats, the major one being that although we have attempted to automate the installation as much as possible, not all eventualities can ever be foreseen. We have tested the installation on the following platforms:
AIX, IRIX, IRIX64, Linux, SunOS, ALPHA and Darwin (MacOS)

There may be problems in the facility build due to the local system configuration which is beyond our control.

2 Installation Requirements

The major part of the **HEALPix** distribution is written in both **Fortran 90** and **C++** and so the appropriate compiler(s) must be present (Linux and Darwin users should look at Section 7.1 about free F90 compilers. Microsoft Windows users should look at Section 7.2). Many visualisation tools and map manipulation routines are provided in **IDL** (please note that at least version 6.4 is required), **Java** and **Python**. Some of the **HEALPix** routines are also available in **C**.
Starting with version 3.0, the healpy (HEALPix in **Python**) library has been integrated into **HEALPix** releases. Since it is, to a large extent, a wrapper to the C++ routines, installing it also requires a C++ compiler (on top of `python` and a few supporting Python libraries) but it will perform its own compilation of the current **HEALPix** C++ library.
Starting with version 3.10, all the (computation intensive) Spherical Harmonics operations required by the C++, Fortran and Python routines are performed with the highly optimized

[1] Microsoft Windows users can have a look at
https://sourceforge.net/p/healpix/wiki/Windows%20and%20peazip/

```
Healpix_3.80/
├── data/
├── doc/
│   ├── epub/
│   ├── html/
│   └── pdf/
├── src/
│   ├── C/
│   │   └── autotools/, subs/
│   ├── common_libraries/
│   │   └── libsharp/
│   ├── cxx/
│   │   └── Healpix_cxx/, autom4te.cache/, autotools/, cxxsupport/, m4/, test/
│   ├── f90/
│   │   └── alteralm/, anafast/, hotspot/, lib/, map2gif/, median_filter/, mod/,
│   │       ngsims_full_sky/, plmgen/, process_mask/, smoothing/, synfast/, ud_grade/
│   ├── healpy/
│   │   └── bin/, cfitsio/, doc/, healpixsubmodule/, healpy/, paper/
│   ├── idl/
│   │   └── examples/, fits/, interfaces/, misc/, toolkit/, visu/, zzz_external/
│   ├── java/
│   │   └── lib/, src/
│   └── python/
└── test/
```

Figure 1: The directory structure for the **HEALPix** distribution.

C-written `libsharp` library, also included in the package, and which was further optimized in version 3.60.

*This section and the next focus on the compilation and installation of the **C**, **C++**, **Fortran 90**, **IDL** and **Python** routines and **libsharp** library. For more information on the **Java** routines see table 1.*

The configure script is written in the Bourne shell. The script attempts to generate a `Makefile` which is tailored to one of the above Operating Systems (OS's) and using `Makefile.in` as a template for non system-specific statements. Only the basic UNIX make facility is required to build the software, although we do still recommend the GNU make facility (`https://ftp.gnu.org/gnu/make/`). In addition, several environment configuration files and an IDL/GDL/FL startup file are generated. These automatically establish various environment variables and aliases to make the use of the **HEALPix** package simpler.

The **HEALPix Fortran 90**, **C++**, **C** and **Python** distributions also require the publicly available CFITSIO library. Note that the **Fortran 90** routines require version 3.20 or more (post August 2009) of CFITSIO. Out of security concerns, the CFITSIO developers

Healpix Package	**Information on installation**	**Information on routines**
Fortran 90	This document	"Fortran Facilities" and "Fortran Subroutines" documents
IDL/GDL/FL	This document	"IDL Facilities"
C++	This document	"C++ Facilities and Subroutines" (HTML only)
C	This document, or `src/C/README`	"C Subroutines Overview"
Java	`src/java/README`	"Java Overview" (HTML only)
Python	This document, or `src/healpy/INSTALL`	"Healpy Documentation" (HTML only)

Table 1: Documentation on the installation and usage of the different packages

recommend using version 3.44 (April 2018) or more.

Software Package	**Source**
CFITSIO V 3.20 (3.44) or more	`https://heasarc.gsfc.nasa.gov/fitsio/`

The **IDL** visualization software is commercially available at

Software Package	**Source**
IDL V 6.4 or more	`https://www.harrisgeospatial.com/Software-Technology/IDL`

while the GNU Data Language **GDL**, a *free open* clone of IDL 7.1, or the freeware Fawlty Language **FL**, a *free closed* clone of IDL 8, can also be used (with some caveats, listed respectively in §7.9 and §7.10) and can be downloaded for free from

Software Package	Source
GDL 1.0.0rc3 or more	https://github.com/gnudatalanguage/gdl
FL 0.79.49 or more	https://www.flxpert.hu/fl or https://bitbucket.org/fawlty/fl/src/master

As it was already the case in version 1.20, users no longer need to acquire the **IDL** Astronomy User's Library (https://idlastro.gsfc.nasa.gov/homepage.html) or the COBE (IDL) Analysis Software (https://lambda.gsfc.nasa.gov/product/cobe/cgis.cfm), although we do recommend these packages to the user. The 100-odd routines required for version 3.80 are contained in the subdirectory `Healpix_3.80/src/idl/zzz_external`. These procedures are included in the **HEALPix** package unchanged and solely for the purpose of making it self contained. In this way, we remove the burden of installation of additional libraries from the end user.

The **Python** `healpy` package requires

Software Package	Source
Python 3.6 or more	https://www.python.org
Numpy 1.13 or more	https://numpy.scipy.org
Matplotlib 0.91.2 or more	https://matplotlib.sourceforge.net
Cython	https://cython.org
Astropy.io.fits	https://www.astropy.org
(PyFITS)	https://pypi.org/project/pyfits/3.3

While not required, the `IPython` (https://ipython.org) and `jupyter` (https://jupyter.org) interfaces can also be useful.

A parallel implementation (based on OpenMP, for shared memory architectures) of the Spherical Harmonics Transforms involved in **F90** `synfast, anafast, smoothing, plmgen, alteralm` and **C++** `synalm_cxx, alm2map_cxx, anafast_cxx, smoothing_cxx, rotalm_cxx` ... is now available by default and can be readily compiled and used with the standard installation script.

A set of routines with MPI parallelization (for distributed memory architectures) is also available for Spherical Harmonics Transform, thanks to the work of H.K. Eriksen (UIO) and Snorre Boasson (ITEA, NTNU). See the **F90** subroutines documentation for more information on how to use those routines in your code.

We found that it was remarkably difficult to find random number generators in the public domain which are simple yet powerful and easy to use. We are providing one (both in **C++** and **F90**) which is an adaptation of an xorshift generator described in Marsaglia (Journal of Statistical Software 2003, vol 8). It has a theoretical period of $2^{128} - 1 \approx 3.4\ 10^{38}$.

3 `healpix_doc`: an easy access to HEALPix documentation

The shell script `healpix_doc` now is available to provide easy access to the HTML, PDF and/or EPUB documentation of *all* Healpix packages. It will automatically open a web browser, a PDF viewer, or an EPUB viewer (among those found on the system) on the documentation available locally (at `$HEALPIX/doc`) or on remote web sites. To use it, simply type

```
$HEALPIX/healpix_doc -e          or
$HEALPIX/healpix_doc -h          or
$HEALPIX/healpix_doc -p
```

to access respectively the EPUB, HTML and PDF documentation. The default browser and viewers used by `healpix_doc` can *optionally* be set with the environment variables `$HEALPIX_HTML_BROWSER`, `$HEALPIX_EPUB_VIEWER` and `$HEALPIX_PDF_VIEWER`.
EPUB is an open e-book format whose dynamical layout allows the same document to be read comfortably on screens of any size. It is supported by many commercial and free e-readers, with sometimes unequal results. Among the *free* ones we tested, we can recommend

- `ebook-viewer`, included in the open source Calibre e-book manager package (`https://calibre-ebook.com`, used to generate the **HEALPix** EPUB files) available for many platforms, including Linux, MacOS and Windows,
- `Books` (formerly iBooks), pre-installed in MacOS and available for download for iOS and watchOS,
- the cost-free, ad-free `Lithium EPUB reader` for Android.

4 The Installation Procedure

If the user has one of the supported OS's, then installation proceeds utilizing the following commands. If your OS is not supported, the configuration step should be omitted, `Makefile.in` should be copied as `Makefile` and explicitly tailored to the user environment.

`% ./configure` *[-L] [--auto=<list>]*	uses `Makefile.in` as a template to build the correct Makefile (from user inputs as required), it will also configure the IDL/GDL/FL routines
`% make`	builds all the facilities
`% make test`	tests all the facility previously compiled
`% make clean`	removes object (`*.o`) files
`% make tidy`	removes object files, module files (`*.mod`), executables and libraries
`% make distclean`	same as above and restores the directories to the state of the original distribution

These different steps are detailed below.

4.1 ./configure [-L] [--auto=<list>] [-h] [-v]

The `./configure` script manages the configuration of the C, C++, Fortran90, IDL and Python suites of routines and facilities.

An online help is available with `./configure -h`, while `./configure -v` will return the **HEALPix** release number (currently 3.80) and exit.

The `-L` option can be used to write the **HEALPix** specific configuration files into the **HEALPix** directory itself rather than in installer's home directory (see §4.1.1). Using the `-L` option is recommended when doing a *project* or *system wide* installation of **HEALPix** to be accessed by several different users.

Two new features have been introduced in version 3.60.

- The `configure` script now supports the `--auto=`*<list>* option to perform an automated (batch) configuration of the various packages using default answers provided by the script (possibly customized with environment variables described below) where *<list>* is a comma separated list of items to be configured, to be chosen (and combined) among
  - `all`: (re-)configures everything;
    `--auto=all` is the same as `--auto=profile,sharp,c,cxx,f90,idl,healpy`;
  - `c`: configures C and the required items, same as `profile,c`;
  - `cxx`: configures C++ and the required items, same as `profile,cxx`,
    libsharp will also be configured, compiled and installed if it was not previously done;
  - `cpp`: same as `cxx` (see above);
  - `f90`: configures F90 and the required items, same as `profile,f90`,
    libsharp will also be configured, compiled and installed if it was not previously done;

  - `idl`: configures IDL and the required items, same as `profile,idl`;
  - `sharp`: forces the (re-)configuration, compilation and installation of libsharp (required by F90 and C++);
  - `profile`: configures user or system configuration files (required by C, C++, F90 and IDL);
  - `healpy`: configures healpy.

- If the environment variables `CC` (C compiler, relevant for C,C++,F90,healpy,sharp), `C_FITS`, `C_SHARED` (C), `CXX`, `CXXFLAGS`, `CXX_PARAL` (C++), `FC`, `F_DIRSUFF`, `F_OPT`, `F_PARAL`, `F_SHARED` (F90), `FITSDIR`, `FITSINC` (C,C++,F90), `PYTHON` (healpy), `PROFILE_EDIT` (profile), `SHARP_COPT`, `SHARP_PARAL` (sharp), and `papersize`, `ps_com`, `pdf_com`, `gif_com` (IDL) are defined prior to calling the `configure` script, they will change the default values proposed in the (interactive or automated) configuration process.
  The online help `./configure -h` will show the current value of these variables.
  As detailed further down, several of these variables are **boolean** in nature, with the values `1`, `y`, `Y`, `t` or `T` meaning 'true', and `0`, `n`, `N`, `f` or `F` meaning 'false'.

These two new features can of course be combined. For instance, in Bourne related shells (eg, `sh, bash, dash, ksh, zsh`), the command
`FC=ifort CC=icc ./configure --auto=f90`
and in C related shells (eg, `csh, tcsh`)
`setenv FC ifort ; setenv CC icc ; ./configure --auto=f90`
will automatically run the `configure` script for the `libsharp` (if not previously done) and `f90` package items, using Intel's Fortran and C compilers.

4.1.1 Configuration profile

A feature introduced in previous releases and enhanced since v2.10, is that the configure script creates a shell configuration file
(located in `${HOME}/.healpix/3_80_`⟨*OS_TYPE*⟩`/config` or in
`${HEALPIX}/confdir/3_80_`⟨*OS_TYPE*⟩`/config` if `./configure -L` was used) according to shell type in which various environment variables and aliases are defined for your convenience. If you agree upon prompting (or set the boolean environment variable `PROFILE_EDIT` to an affirmative value), it will also change your default system profile during installation to automatically source this profile. If you do not agree to this change (or set the boolean environment variable `PROFILE_EDIT` to a false value), you will need to explicitly source the configuration file above for any session in which you intend to run **HEALPix** facilities. **In particular, you will have to make sure that the `HEALPIX` system variable is correctly defined (as the full path to the HEALPix directory) before running the package**.

4.1.2 C configuration

The `./configure` script will ask for the C compiler and options to be used, and for the full path of an installed `cfitsio` library to (optionally) link to. By default, only a static library is created, but the user can also ask for a shared (Unix/Linux systems) or dynamic (Darwin) library.
The environment variables `CC`, `C_FITS` (boolean), `FITSDIR`, `FITSINC` and `C_SHARED` (boolean) can be used to control the script behavior.
After compilation (see `make` section) and linking, all libraries will be in `${HEALPIX}/lib/chealpix.*` .
See also §6 on `pkg-config`.

4.1.3 C++ configuration

Starting with version 3.60, the `./configure` script will be used to provide information (like the choice of C++ compiler and options) to an automated (`autoconf` generated) configure script, (located in `src/cxx/configure`), which will take care of the configuration.
The environment variables `CC`, `CXX`, `CXXFLAGS`, `FITSDIR` and `FITSINC` can be used to customize the whole process. If the latter two are not explicitely set, the autoconf configure script will look for a `cfitsio` installation on its own.
The configuration of `libsharp` will be also taken care of at this stage.
The boolean variable `CXX_PARAL`, introduced in version 3.80 and defaulting to 1 (=true), controls whether the code will be parallelized (with OpenMP) or not. To obtain a serial implementation of the code, set `CXX_PARAL=0` and make sure no OpenMP related flags appear in `CXXFLAGS`.
At odds with previous versions, the C++ binaries, libraries and header files will be installed in `${HEALPIX}/bin`, `${HEALPIX}/lib` and `${HEALPIX}/include` directories respectively.
If the **HEALPix** configuration file is sourced as described in §4.1.1, the full path to the C++ executables will be added to the environment PATH variable.
See also §6 on `pkg-config`.

4.1.4 Fortran 90 configuration

When you run `./configure` on a supported system you will be prompted to enter compiler optimisation flags. We have not attempted to provide the best optimisation flags for all operating systems. The configure script will have a guess at optimisation options for some systems, but it is up to the user to figure out an optimal set[2]. From our experience, we have not found significant accumulation of numerical error even when using the most aggressive

[2]In particular, the Intel Fortran Compiler, available for free for Linux systems with Intel-like processors, have specific tuning options for each Intel processor family and instructions set. Please consult the online help (`ifort -help`) or PDF documentation (`/opt/intel/composer*/Documentation/en_US/Release_NotesF.pdf`) or HTML documentation (`/opt/intel/composer*/Documentation/en_US/documentation_f.htm`) for further information.

optimisation level available.
The environment variable `FITSDIR`, `CC`, `FC`, `F_DIRSUFF`, `F_OPT`, `F_SHARED` (boolean) and `F_PARAL` (boolean) can be used to customize the configuration.
The configuration of `libsharp` will be also taken care of at this stage.
If the **HEALPix** configuration file is sourced as described in §4.1.1, the full path to the F90 executables will be added to the environment PATH variable.
See also §6 on `pkg-config`.

4.1.5 IDL/GDL/FL configuration

You will be asked for the external applications, such as `gv`, `xpdf`, `display`, ... you want to use to visualize the GIF, JPEG, PDF, Postscript and PNG files created by IDL, GDL or FL.
The environment variables `papersize`, `ps_com`, `pdf_com` and `gif_com` can be used to customize this configuration.
If the **HEALPix** configuration file is sourced as described in §4.1.1, the aliases `hidl` (`hidlde`), `hgdl` (`hgdlde`) and/or `hfl` (`hflde`) are also defined to give you access to **HEALPix** routines from respectively IDL, GDL and/or FL, with a command-line (or graphical) interface.
See the **HEALPix** IDL Document for more information on using **HEALPix** IDL/GDL/FL together with other IDL libraries, and §7.9,7.10 on using GDL or FL instead of IDL.

4.1.6 Java installation

The configuration and installation of the **HEALPix** Java package is currently handled separately. See table 1 for more information.

4.1.7 Python (healpy) installation

The `./configure` script will ask for the Python command you want to use (in case many coexist on your computer) and will check that its version number meets the `healpy` requirements (see above), as well as for C and C++ compilers.
The environment variables `PYTHON`, `C` and `CXX` can be used to customize the configuration process.
Note that during the compilation with `make` (see below), the `src/healpy/setup.py` Python script will be invoked to automatically prompt a *fresh* compilation of the `src/cxx/*` libraries, with all the options necessary to Python linkage, and can be done independently of the C++ installation described above.
Note also that the `healpy` compilation will very likely require an active network connection in order to download on the fly some of the required Python ancillary packages.

4.1.8 libsharp library

The `libsharp` C-written library for optimized Spherical Harmonics operations is used by the C++, Fortran, IDL and healpy routines and facilities. Starting with version 3.60, a new, faster, implementation is in use, and will be configured (only once) at the same time as any of the C++ or Fortran packages, but can also be configured on its own.
The environment variables `CC` and `SHARP_COPT` can be used to set respectively the C compiler and its options proposed during the interactive or automated configuration process. For optimal performance, the C compilation flags should include `-ffast-math` and `-march=native` (or your compiler's equivalent options), and may look like `SHARP_COPT='-O3 -ffast-math -march=native -fopenmp'` (multi-worded values must be enclosed in quotes).
If you are using `gcc` or `clang` (see below) and you want to produce a portable, high-performance library, and if you compiler and assembler support it, you can also replace `-march=native` by `-DMULTIARCH`.
If you are using `clang` to compile `libsharp`, make sure it supports OpenMP (as in version 3.7 or more), and that OpenMP is enabled explicitly among the compiler options (possibly requiring the flag `-fopemp=libomp` or `-fopemp=libiomp5` instead of the usual `-fopemp` or by specifying the location of the OpenMP libraries during compilation and at run time).
The boolean variable `SHARP_PARAL`, introduced in version 3.80 and defaulting to 1 (=true), controls whether the library will be parallelized (with OpenMP) or not. To obtain a serial implementation of `libsharp`, set `SHARP_PARAL=0` and make sure no OpenMP related flags appear in `SHARP_COPT`.
After compilation and installation (which, for `libsharp` only, are done as early as the `configure` step) the resulting library will be in `${HEALPIX}/lib/libsharp*` and the header files in `${HEALPIX}/include/libsharp/sharp*.h`.
See also §6 on pkg-config.

4.2 Compilation and installation

The

```
make
```

command will compile one or several of the C, C++, F90, libsharp and Python packages depending on what was configured with the `./configure` script. Specific packages can be compiled with the respective commands

```
make c-all
make cpp-all
make f90-all
make sharp-all
make healpy-all
```

To perform several compilation jobs simultaneously, the command `make -j` *[num-*

ber_of_jobs] can be used.

Please neglect any possible warnings at compile time. If you run into trouble please refer to the section **Troubleshooting and further information**.

After running `make`, the user must re-login to ensure that the new profiles built by the installation procedure are correctly sourced. Only then will the user have full access to the specific **HEALPix** environment variables etc.

4.3 Testing the installation

All installed libraries and executables can be tested with

```
make test
```

while specific tests of the C, C++ and Fortran products can be performed with, respectively

```
make c-test
make cpp-test
make f90-test
make sharp-test
make healpy-test
```

For `f90-test`, Table 2 lists the codes tested with the parameter files used, as well as the data files produced and the respective reference files.

code & parameter file	output data	reference data	output image	reference image
synfast syn.par	test_map.fits	map.fits	test_map.gif	map.gif
	test_alm.fits	alm.fits	*NA*	*NA*
smoothing smo.par	test_sm.fits	map_sm.fits	test_sm.gif	map_sm.gif
ud_grade udg.par	test_LOres.fits	map_LOres.fits	test_LOres.gif	map_LOres.gif
hotspot hot.par	test_ext.fits	map_ext.fits	test_ext.gif	map_ext.gif
	test_max.asc	max.asc	*NA*	*NA*
	test_min.asc	min.asc	*NA*	*NA*
anafast ana.par	test_cl.fits	cl_out.fits	*NA*	*NA*
— ana2maps.par	test_cl2maps.fits	cl2maps.fits	*NA*	*NA*
— ana_w2.par	test_cl_w2.fits	cl_w2.fits	*NA*	*NA*
alteralm alt.par	test_almdec.fits	almdec.fits	*NA*	*NA*
median_filter med.par	test_mf.fits	map_mf.fits	test_mf.gif	map_mf.gif
sky_ng_sim ngfs.par	test_ngfs.fits	map_ngfs.fits	test_ngfs.gif	map_ngfs.gif
process_mask prmask.par	test_distmask.fits	distmask.fits	test_distmask.gif	distmask.gif

Table 2: Data files and images produced by the Fortran codes during the tests, and the respective reference files to which they can be compared. All the files listed are located or produced in the `Healpix_3.80/test` directory. The GIF images of full sky maps were produced using `map2gif`. *NA*: No image available, because the data set is not a sky map

Notes:

- the input power spectrum (in `Healpix_3.80/test/cl.fits`) used to generate the Fortran90 test maps is currently the WMAP 1yr best fit, in $(\mu\mathrm{K})^2$, and is therefore different from the one included in releases 1.* (that can still be found in `cl_old.fits`).
- Other input fiducial $C(\ell)$, all in $(\mu\mathrm{K})^2$ and defined on the multipole range $[0, \ell_{\max}]$, have been included for convenience in `Healpix_3.80/test/` in a **HEALPix** compatible format.

File name	alias	Origin	$\ell_{\max}$
wmap_lcdm_pl_model_yr1_v1.fits	cl.fits	WMAP-1yr (2005)	3000
wmap_lcdm_sz_lens_wmap5_cl_v3.fits	cl_wmap5.fits	WMAP-5yr (2008)	2000
wmap_lcdm_sz_lens_wmap7_cl_v4.fits	cl_wmap7.fits	WMAP-7yr (2011)	3726
planck2013ext_lcdm_cl_v1.fits	cl_planck1.fits	Planck 2013	4500
planck2015_lcdm_cl_v2.fits	cl_planck2.fits	Planck 2015	4900
planck2018_lcdm_cl_v3.fits	cl_planck3.fits	Planck 2018	5000

For more information on their respective origin and underlying model see their FITS header, or `Healpix_3.80/test/README`.

In order to test the new **HEALPix** profile set-up one can then attempt to run any C++ or F90 facility from any directory on your system. Similarly, IDL, GDL or FL can be tested by invoking respectively `hidl` (or `hidlde`), `hgdl` (or `hgdlde`), or `hfl` (or `hflde`).

4.4 Cleaning up

Three levels of cleaning are available:

```
make clean
```

will remove the intermediate files created during compilation, such as object files, (Fortran) modules files, ... found in the source or build directories;

```
make tidy
```

same as above, and will also remove the **HEALPix** executables, libraries and module and/or include files;

```
make distclean
```

will return the **HEALPix** directory to its original 'distribution' state by discarding the same files as above, as well as the executable and library directories and the top level Makefile.

As a consequence, `make clean` can be used after a successful compilation and installation in order to remove now useless intermediate files while keeping the codes functional, while `make tidy` should be used between consecutive (failed) attempts with different compilers, compiler versions or compiler options, to avoid any conflict between new and pre-existing files.

4.5 Linking a code with HEALPix library

Third party or user-developed codes may require **HEALPix** as an external library. An easy way to achieve this linking is to use the `pkg-config` facility (now available on many systems, including Linux, Unix*, MacOS and MS Windows), following the procedure described in §6 on `pkg-config`.

5 A Note on *Re*-installation

As a result of the line added to your shell profile which explicitly sources the **HEALPix** profile, care must be taken if the package is reinstalled in a different directory. If such reinstallation is desired, the included line must be removed from your system profile, allowing the corrected version to be added.

6 Pkg-config files

Starting with **HEALPix** 3.12, `pkg-config` (`.pc`) files are generated during the configuration of the libsharp, C, C++ and F90 packages, and are initially located respectively in `${HEALPIX}/lib/pkgconfig/libsharp.pc`, `${HEALPIX}/lib/pkgconfig/chealpix.pc`, `${HEALPIX}/lib/pkgconfig/healpix_cxx.pc`, and `${HEALPIX}/lib`*suffix*`/pkgconfig/healpix.pc`.

If the `pkg-config` software is available on your system (see `https://www.freedesktop.org/wiki/Software/pkg-config/` to download, install and use it) and if the location of the **HEALPix** `pkg-config` files above are known to it (either by moving/copying them to one of the standard locations returned by
`pkg-config --variable=pc_path pkg-config`
or by customizing the environment variable `PKG_CONFIG_PATH`[3]), then linking your own or third-party code with the C, C++, F90 **HEALPix** library simply becomes

```
cc `pkg-config --cflags --libs chealpix` mycode.c -o mycode
c++ `pkg-config --cflags --libs healpix_cxx` mycode.cpp -o mycode
FC  `pkg-config --cflags --libs healpix` mycode.f90 -o mycode
```

(where *FC* has to be replaced by the Fortran compiler used to generate the **HEALPix** library).

[3] a third option is provide the location of the `.pc` file in full at each `pkg-config` invocation : *eg* `pkg-config --cflags --libs` *full_path*`/healpix.pc`

7 Troubleshooting and further information

This section contains a list of difficulties which we have dealt with. It is by no means exhaustive. In case of problems, see https://healpix.sourceforge.io/support.php or contact *healpix-support* `at` *lists.sourceforge.net*

7.1 Free Fortran90/95 Compilers

Some **free** Fortran90/95 compilers that can be used to compile **HEALPix** are listed below. They all support the few Fortran 2003 features used in **HEALPix**.

- **Intel Fortran** compiler (`ifort`) for Linux based computers (versions 11.* to 19.*[4])
  https://software.intel.com/en-us/fortran-compilers
- **GNU Fortran 95** compiler (`gfortran`) included in GNU Compiler Collection *GCC* version 4.0.0 and up and available for Linux, Mac OSX, Windows, Sun ... platforms
  https://www.gnu.org/software/gcc/fortran/.
  GFortran binaries for all platforms can also be downloaded from
  https://gcc.gnu.org/wiki/GFortranBinaries.
  Please note that only recent versions of `gfortran` (Aug 2005 and later) compile **HEALPix** correctly, and v4.2.1 and more have given satisfying results so far, including native OpenMP support.
- **Nvidia's LLVM-based Fortran** compiler (`flang`) available as pre-compiled executables and libraries for Linux
  https://www.scivision.co/flang-compiler-build-tips
  and as source files for all platforms
  https://github.com/flang-compiler/flang/wiki/Building-Flang.
- **Nvidia's PGI Fortran** (formerly Portland Group) compilers (`pgf90`) available as freemium (without support) or commercially for Linux, Mac OSX and Windows from
  https://www.pgroup.com/index.htm.
- **G95** compiler available for Linux, Mac OSX, Windows, Sun and HP platforms with 32 and 64 bit architectures (eg, x86 and x86-64). In the latter case, the '32bit default integer' (32bit DI) version of `g95` *must* be used. Note that this compiler was last released in 2013, and it generally generates slower codes than the compilers listed above.
  http://www.g95.org

See http://fortranwiki.org/fortran/show/Compilers for an extended list of free, freemium and commercial Fortran compilers.

[4] problems have been reported with one of the code (`sky_ng_sim`) when compiled with `ifort 14.0.1.106`

7.2 Installation under Microsoft Windows

Detailed instructions to install **HEALPix** on Windows 7 using Cygwin, kindly provided by John Arballo, are available in §7.2.1, while other configurations are discussed in §7.2.2.

7.2.1 Installation on Windows 7 with Cygwin

The three steps (installation of Cygwin, cfitsio and **HEALPix** respectively) must be done in that order.

A: Install Cygwin

1. Go to https://www.cygwin.com/ and click on 'Install Cygwin' in the menu on the left.
2. Click on `setup-x86.exe` (for 32-bit installation) or `setup-x86_64.exe` (for 64-bit installation) and then 'Save File' when prompted.
3. Go to your Downloads folder (or wherever you saved setup-x86*.exe) and double-click on the `setup-x86*.exe` file to run it.
4. Accept all defaults, except:
   (a) You have to 'Choose A Download Site'. (eg: https://ftp.gtlib.gatech.edu).
   (b) When prompted to 'Select Packages', expand 'Default' (if you see a '+' to the left of it), expand 'Devel', then find and add the following packages (click on 'Skip' for each of them so it changes to the version number and a checkbox appears in the 'Bin' column):
   ```
   gcc-core
   gcc-fortran
   gcc-g++
   make
   ```

The installation will take a few minutes.

B: Install CFITSIO Library

1. Get the latest source code package from NASA's HEASARC website (https://heasarc.gsfc.nasa.gov/FTP/software/fitsio/c/cfitsio_latest.tar.gz). When prompted to save the file, in the Save dialog window, navigate to `C:\cygwin64\usr\local` (assuming you accepted the defaults when installing Cygwin), click on 'New folder' and name it 'src', go into that folder and 'Save'.
2. Open a Cygwin terminal (via the new Desktop icon or through your Start menu).
3. Enter the following commands at the '**$**' prompt:
   ```
   $ cd /usr/local/src
   $ tar zxvf cfitsio_latest.tar.gz
   ```

```
$ cd cfitsio
$ ./configure --prefix=/usr
$ make
$ make install
$ cd ../
```

4. Leave the Cygwin terminal open.

C: Install HEALPix

1. Get the latest version of HEALPix from SourceForge (https://sourceforge.net/projects/healpix/files/latest/download). When prompted to save the file, save it in `C:\cygwin64\usr\local\src`.

2. In Windows Explorer, navigate to C:\cygwin64\usr\local\src, right-click on `Healpix_3.80_*.zip` and 'Extract all...'. Accept the default location.

3. In the Cygwin terminal, type the following commands at the '**$**' prompt (use the names of the Healpix directories for the version you installed):
   ```
   $ cd Healpix_3.80_*
   $ cd Healpix_3.80
   $ ./configure
   ```
   Select an option from the menu (e.g., '2' for the C package) and accept all of the defaults except that the first time you run configure, you'll be prompted at the end to modify your home shell profile ('.profile'). Enter 'y' at this prompt.
   ```
   $ make
   $ make test
   $ make tidy
   ```

7.2.2 Other Windows configurations

Installation on Windows versions other than 7 should be very similar to the one detailed above.
In step A above, replacing Cygwin with **MinGW** (http://www.mingw.org/) together with the MSYS collection of GNU utilities (see http://www.mingw.org/wiki/msys and https://sourceforge.net/projects/mingw/files) is also possible. The Unix/Linux tools required include `sh, make, awk, grep, sed, ls, wc, cat, more, nm, ar`, as well as C, C++ and Fortran compilers.
The latest `gfortran` binaries for Cygwin and/or MinGW can be found at, eg https://cygwin.com/cgi-bin2/package-grep.cgi?grep=gcc-fortran&arch=x86_64, following the tips found at https://gcc.gnu.org/wiki/GFortranBinaries.

7.3 Problems with CFITSIO

Compilation of CFITSIO Fortran wrappers

The most common problem with the Fortran **HEALPix** compilation will produce messages like:

```
ld: Undefined symbols:
 _ftbnfm_
 _ftclos_
 _ftcrhd_
 _ftdkey_
 ...
```

or

```
fitstools.f90: undefined reference to ‘ftdkey_’
fitstools.f90: undefined reference to ‘ftbnfm_’
fitstools.f90: undefined reference to ‘ftclos_’
...
```

or

```
Undefined symbols:
 "_ftghbn_", referenced from:
     ___fitstools_MOD_read_fits_cut4.clone.2 in libhealpix.a(fitstools.o)
     ___fitstools_MOD_getsize_fits.clone.1 in libhealpix.a(fitstools.o)
     ___fitstools_MOD_getsize_fits in libhealpix.a(fitstools.o)
  ...
ld: symbol(s) not found
collect2: ld returned 1 exit status
```

and occurs when the CFITSIO installation script could not find a valid fortran compiler. To solve this problem

1. Go into the CFITSIO directory.
   Assuming that **ifort** is available on your system (it can be replaced below by **gfortran**, **g95**, **f77**, **f2c**, ...) type:
   `./configure FC=ifort`
   `make`
   `make install` (optional).
2. Then go back into the **HEALPix** directory and do
   `./configure` (making sure that you are using the

newly created `libcfitsio.a` library)

```
make
make test
```

See also the note below on 64 bit architectures.

CFITSIO problems on systems with 64 bit architecture

1. **Linux, Mac OS X**

   If the **HEALPix** codes are compiled in 64 bits, and the GNU C Compiler (`gcc`) is used to compiled CFITSIO, then issue the following commands in the CFITSIO directory:

   ```
   ./configure FC='gcc -m64'
   make
   ```

   You can then force compilation to the same binary format by entering `-m64` when asked for the optimisation options in the **HEALPix** configure script.

2. **IRIX64**

   On a 64-bit architecture such as IRIX64, CFITSIO will have to be compiled in the same binary format as the **HEALPix** codes. This can be achieved by typing the following on the command line in the CFITSIO directory:

   ```
   rm config.cache
   setenv CC 'cc -n32'
   ./configure
   make
   ```

   Alternatively you can replace the `-n32` with `-64`. You can then force compilation to the same binary format by entering either `-n32` or `-64` when asked for the optimisation options in the **HEALPix** configure script.

CFITSIO linking problems

A particular problem encountered with the CFITSIO Version 2.0 release relates to the inclusion of various libraries within the system release for a given machine. This led to some modifications to the Makefile to include the specific library links `-lm -lnsl -lsocket` on SunOS, but only `-lm` for IRIX64. If your OS is not completely supported by the distribution, you may find this as one source of errors. The CFITSIO developers recommend compilation of the `testprog` routine. Inspection of the libraries linked after executing the `make testprog` statement will reveal those you need to include in the Makefile.

CFITSIO and Debian/Linux

Some problems have been reported on Debian/Linux systems during the linking to the CFITSIO library shipped with Linux. If these problems occur, try to recompile the CFITSIO library from scratch before linking to **HEALPix**.

CFITSIO and `libcurl`

Starting with version 3.42, CFITSIO is by default linked with the curl library (`https://curl.haxx.se/libcurl`, used to read remote FITS files via https) whenever it is available. This shared or dynamic library is pretty standard on modern systems, and often located in `/usr/lib` or `/usr/lib64`, and the command `curl-config` can be used to determine its location. In this case, when executing the **HEALPix** code, the system must know where to find this library at runtime as explained for instance here for Linux/Unix or there for MacOSX.

CFITSIO from Heasoft

The Heasoft suite of software packages for High Energy Astrophysics, also hosted at HEASARC and available as source files or precompiled binaries, includes a cfitsio library and its header files. However, trying to link **HEALPix** to that installation of cfitsio will generally fail, because

- the *precompiled* cfitsio library may not be properly detected during the configuration of **HEALPix** (in C, C++ and F90),
- the Heasoft header files `rotmatrix.h` and `pointing.h` found in `${HEADAS}/include` (like `fitsio.h`) will conflict with the ones provided in `${HEALPIX}/src/cxx/cxxsupport`, preventing the compilation of **HEALPix** C++ routines.

It is therefore recommended to link **HEALPix** to a cfitsio library compiled locally and *not* included in Heasoft.
If Heasoft's cfitsio is to be used, Heasoft must have been compiled locally from source files, and the paths provided during the **HEALPix** configuration must be `FITSDIR=${HEADAS}/../heacore/`PLATFORM`/lib` and `FITSINC=${HEADAS}/../heacore/`PLATFORM`/include` (*instead* of the expected `FITSDIR=${HEADAS}/lib` and `FITSINC=${HEADAS}/include`) where PLATFORM depends on your computer and operating system and may look like `x86_64-pc-linux-gnu-libc2.29`.

7.4 diff shows that the test files are different from the supplied files

This by itself is no cause for concern. When comparing using a `diff` on the test files will most likely report a difference even when the installation has been successful. This may be due to the fact that different installations have different floating point representations. Also, the FITS files carry date information.

7.5 Try unlimit

If you have unforeseen problems at runtime, try `unlimit` (under csh or tcsh) or `ulimit` (under sh or bash), in order to increase the heap and stack memory size. It sometimes helps.

7.6 hidl usage

We have found that in very rare cases the alias `hidl` is not recognised by the user's system. Usually, this is related to the local system's IDL script. A quick-fix is achieved by setting the environment variable `IDL_STARTUP` to be equal to the **HEALPix** startup file `HEALPix_startup` **including** the directory path to the file. This enables the user to access the **HEALPix** IDL procedures simply by invoking IDL. For example, in the typical installation documented above for a user running the tcsh shell, the command
`setenv IDL_STARTUP /disk1/user1/HEALPix_3.80/src/idl/HEALPix_startup`
should be issued (or added to the user's shell profile).

If the user already has an IDL startup file, then this should be merged with `HEALPix_startup`. This temporary solution does mean that the **HEALPix** IDL procedures are available in the `IDL_PATH` at all times, which may lead to conflicts with user-defined procedures. The `hidl` invocation was intended to circumvent these issues, allowing **HEALPix** IDL procedures to be available only when desired.

A proper fix requires the user to ask the local system administrator to adjust the local IDL script.

7.7 Using HEALPix IDL together with other IDL libraries

See the homonymous section in the "IDL Facilities Overview"

7.8 Mac OS X, X11 and IDL cursor

If the IDL cursor does not work correctly on X11 windows under Mac OS X, and the 2nd and 3rd button clicks are ineffective, type

- with Apple's X11:
  - under Tiger (10.4.*):
    `defaults write com.apple.x11 wm_click_through -bool true`
  - under Leopard (10.5.*), Snow Leopard (10.6.*) and Lion (10.7.*):
    `defaults write org.x.x11 wm_click_through -bool true`
- with Xquartz (default under Moutain Lion (10.8.*), Mavericks (10.9.*) and Yosemite (10.10.*), available for download for El Capitan (10.11.*), Sierra (10.12.*), High Sierra (10.13.*), Mojave (10.14.*), Catalina (10.15.*) and Big Sur (11.0.*)):
  `defaults write org.macosforge.xquartz.X11 wm_click_through -bool true`
- with MacPort's X11 (package xorg-server):
  `defaults write org.macports.X11 wm_click_through -bool true`

at your X11 prompt and restart X11.
Note that the command `ls -lrt $HOME/Library/Preferences/*[xX]11*plist` can be used to determine the X window system installed on your Mac. See also http://www.idlcoyote.com/misc_tips/maccursor.html and mollcursor documentation in "IDL Facilities").

7.9 Using GDL instead of IDL

GNU Data Language (GDL), is a *free* clone of IDL 7.1, with support for some IDL 8.0 features (for more information see https://github.com/gnudatalanguage/gdl). Both the source code and precompiled executables for various platforms are available.

When used to run IDL-Healpix routines, GDL 1.0.0rc3 or more gives very satisfactory results[5]. The calculations agree with those done under IDL, with comparable computation times, but a few minor features, mostly related to the font selection, are missing.

GDL+HEALPix specific requirements

To fully enjoy GDL capabilities

- **HEALPix** 3.80 or more must be installed
- Besides the mandatory requirements (plplot, gsl, readline and zlib) GDL must also have been (pre-)compiled with links to

[5] All the caveats listed below have been noticed in GDL 0.9.7 (released in Jan 2017), 0.9.8 (March 2018), 0.9.9 (Nov 2018) and 1.0.0rc3 (June 2020) and may be solved in subsequent versions. Please send all your questions *on* GDL and/or its installation directly to GDL developpers at https://github.com/gnudatalanguage/gdl/issues.

- ImageMagick (or GraphicsMagick) to produce GIF, JPEG and PNG output files, and
- pslib (recommended, but not required) to produce PostScript and PDF files (in the latter case, a recent version of ghostscript, i.e. 9.07 or more, is also recommended).

Impact of GDL limitations on HEALPix

- When run under GDL 1.0.0rc3, and if the requirements stated above are met, the visualization routines azeqview, cartview, gnomview, mollview and orthview will produce correct screen (X) outputs and PS, PDF, PNG, GIF, and JPEG images, with the following caveat(s):
  - the pfonts keyword will not allow the selection of other fonts than Hershey vectorial fonts (pfonts[0]=-1).

  All other features work properly, including the Latex keyword.

7.10 Using FL instead of IDL

Fawlty Language (FL) is a black-box implementation of IDL 8.0, for which precompiled self-contained packages are available for Linux, Windows, MacOSX and more from https://www.flxpert.hu/fl.

Most of the IDL routines and features have been implemented, with a few exceptions (like `xloadct`) and the restrictions listed below.

FL+HEALPix specific requirements

To fully enjoy FL capabilities

- **HEALPix** 3.80 or more must be installed,
- the version 0.79.49 or more of FL must be used,
- it is recommended to set the environment variable `FL_DIR` to the FL top directory (ie *path*`/fl/fl_0.79.49` in Linux and Windows, and *path*`/fl.app` in MacOSX) in order for the **HEALPix** enabled FL tools (`hfl` and `hflde`) to be defined properly during the IDL/GDL/FL configuration.
- to produce PDF files a recent version of ghostscript, i.e. 9.07 or more, is recommended.

Impact of FL limitations on HEALPix

- In FL, the `!p.font` selection is ignored in the 'X' device. In 'PS' device, the Hershey Fonts (`!p.font=-1`) and Device Fonts (`!p.font=0`) look respectively slightly and noticeably different from their IDL counterparts, while the TrueType Fonts (`!p.font=1`) are not fully implemented yet.
  As a consequence, the graphical outputs of `azeqview`, `cartview`, `gnomview`, `mollview` and `orthview` will look slightly different in FL and IDL, while in those routines the option `PFONTS` will not work fully as expected. However, the `Latex` keyword will work properly in those routines.

8 Appendix I: Recent Changes and New Features

8.1 Bug corrections and Improvements in Version 3.80

8.1.1 General

- addition of SHARP_PARAL and CXX_PARAL to control the parallel implementation of the `libsharp` library and C++ library and codes;
- PYTHON now defaults to `python3`

8.1.2 C++

- the line-integral convolution interface is now accessible not only from the command line, but also via C++ calls, to allow calling from `healpy`;
- some internals were restructured to allow easier integration with SWIG

8.1.3 Fortran 90 facilities and subroutines

- Improvement of `query_disc` routine in `inclusive` mode,
- the routines `alm2map_spin` and `map2alm_spin` now accept any (integer) spin values $|s| \geq 0$, but the scalar routines `alm2map` and `map2alm` are still recommended for vanishing spin ($s = 0$),
- correction of bugs preventing the compilation with versions 10.* of `gfortran`,
- fixed bug affecting `map2gif` when compiled with versions 10.* of `gfortran` and `gcc`.

8.1.4 IDL

- Improvement of `query_disc` routine in `inclusive` mode;
- update of the required IDL-astron library routines, and Coyote library routines (2021-04-08).

8.1.5 Python

- Switch to `healpy 1.15.0` (CHANGELOG)
  - `write_map` keeps dtype of input map array instead of float32; `read_map` keeps dtype of FITS file instead of upcasting to float64; `write_cl` uses dtype of input cl instead of float64

- Changed all warnings to using the `logging` module, deprecated all `verbose` keywords
- Flip sign for spin-0 `alm2map_spin` and `map2alm_spin`; fixed `map2alm_spin` bug for masked input
- Support transparency in plotting with the `alpha` parameter
- Experimental `projview` function to plot maps using projections from `matplotlib`
- Removed the note that we will change order of cl in `synfast` and `synalm`, we will leave `new=False` default
- Added convenience functions `order2npix` and `npix2order`
- Support nested maps `hp.smoothing`; fixed indexing issue in `bl2beam`
- Allow OBJECT FITS header not to be a string
- Drop support for Python 2.7-3.5; Improvements of the build system; Automatically build wheels for Linux/MacOS on Github actions
- and other minor bug fixes ...

9 Appendix II: Older changes (versions 3.00 to 3.70)

Bug corrections and Improvements in Version 3.70 (2020-07)

General

- Fixed several bugs in the configure script
- Documentation now available in EPUB format

Fortran 90 facilities and subroutines

- Addition of the subroutines read_fits_partial and write_fits_partial to read and write FITS files containing polarized or unpolarized maps defined on a fraction of the sky (see https://healpix.sourceforge.io/data/examples/healpix_fits_specs.pdf).

IDL

- Addition of read_fits_partial and write_fits_partial to read and write FITS files containing polarized or unpolarized maps defined on a fraction of the sky (see https://healpix.sourceforge.io/data/examples/healpix_fits_specs.pdf).
- Update of the required IDL-astron library routines, and Coyote library routines (2020-07-15).

Python

- Switch to healpy 1.14.0 (CHANGELOG)
  - Line Integral Convolution plots to plot polarization,
  - fixed FITS files that were left open,
  - increased precision in coordinate transforms,
  - fix propagation on mmax in smoothing,
  - reworked verbose,
  - and many other improvements and bugs fixes ...

Bug corrections and Improvements in Version 3.60 (2019-12)

General

- The computation time of a map synthesis or analysis has been reduced (for instance, by at least 30% at $N_{\rm side} = 2048$ and $\ell_{\rm max} = 4096$), with the same memory footprint and numerical accuracy as previously, thanks to
  - major performance increase for Spherical Harmonics Transforms in the libsharp C-written library called by the C++, F90, IDL and python routines and facilities, thanks to ideas of Keiichi Ishioka (https://doi.org/10.2151/jmsj.2018-019 and personal communication);
  - the possibility of building the libsharp library with simultaneous support for different x86 CPU features (SSE2, AVX, AVX2, FMA3, FMA4, AVX512F); the appropriate set of subroutines being selected automatically at runtime.
- The configure script will ensure a single and seamless configuration, compilation and installation of the libsharp library, even if several language implementations of **HEALPix** are compiled.
- The configure script now supports an automated mode beside the usual interactive mode, and some environment variables can be used to customize its behavior in both modes (eg, choice of compilers and their options).

C++

- Link to the new and faster libsharp library
- Simpler configuration with the systematic use of autotools
- The C++ binaries, libraries and header files now installed in ${HEALPIX}/bin, ${HEALPIX}/lib and ${HEALPIX}/include directories respectively.
- Added documentation for the module needlet_tool.

Fortran 90 facilities and subroutines

- Link to the new and faster libsharp library
- Some external C routines replaced by Fortran 2003 extensions.

IDL

- Faster isynfast, ianafast, ismoothing routines
- addition of outline_earth to create a structure outlining Earth features such as coastlines, rivers, country boundaries, ...
- azeqview, cartview, gnomview, mollview, orthview visualization routines: support for color and thickness in outline keyword
- Update of the required IDL-astron library routines, and Coyote library routines (2019-10-30).

Python

- Switch to healpy 1.13.0 (CHANGELOG)
  - different handling of default dtype in read_cl, write_cl and read_map
  - implemented dist2holes, distance from pixel center to closest invalid pixel
  - allow not-power-of-2 $N_{\rm side}$ for RING

Bug corrections and Improvements in Version 3.50 (2018-11)

Fortran 90 facilities and subroutines

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

C++

- C implementation of fftpack replaced with pocketfft
- online documentation for Line Integral Convolution code `alice3`

IDL

- `fits2cl`: addition of `/PLANCK3` keyword to read the fiducial Λ-CDM $C(\ell)$ model which best fits the 2018 Planck data analysis (available from `Healpix/test/planck2018_lcdm_cl_v3.fits`);
- `rotate_coord`: addition of optional variable `Delta_Psi` containing rotation of polarization on output, and of keyword `Free_Norm` to deal with un-normalized input coordinate vectors;
- minor bugs correction in `azeqview, cartview, gnomview, mollview, orthview` (when `polarization=3`) and `alm2fits` (user provided header now correctly processed).
- Update of the required `IDL-astron` library routines, and `Coyote` library routines (2018-09-27).

Python

- Switch to `healpy 1.12.8` (CHANGELOG)

Bug corrections and Improvements in Version 3.40 (2018-06)

General

- A new set of (pixel-based) quadrature weights has been introduced, besides the older ring-based ones, to improve the accuracy of the Spherical Harmonics calculation. For maps containing a signal that is band-limited at $\ell_{\rm max} = 1.5N_{\rm side}$, this allows recovery of the $a_{\ell m}$ at almost machine precision. They are supported by map-analysis routines in C++, Fortran, IDL and Python. The weights for power-of-2 values of $N_{\rm side}$ in $\{16, \ldots, 2048\}$ are precomputed and shipped in `Healpix/data/weight_pixel_n?????.fits`, and the missing ones can be computed for any value of $N_{\rm side}$ with the `compute_weights` C++ facility.

C++

- IMPORTANT: the syntax for specifying ring weights and pixel windows has changed! This affects the facilities `anafast_cxx`, `smoothing_cxx`, `udgrade_harmonic_cxx`, `alm2map_cxx`, `mult_alm_cxx`. Pixel window files have to be specified (with path) using the parameter `windowfile`; `ringweights` is used for ring weight files, and `pixelweights` for pixel weight files.
- Full pixel quadrature weights are now supported in map analysis facilities such as `anafast_cxx`, `smoothing_cxx` and `udgrade_harmonic_cxx` using the `pixelweights` parameter.
- Experimental `needlet_tool` code for needlet analysis

Fortran 90 facilities and subroutines

- The facilities `anafast` and `smoothing` now support pixel-based quadrature weights. Introduction of the supporting `nside2npweights`, `unfold_weightsfile`, `get_healpix_weight_file`, `get_healpix_pixel_weight_file`.
- The subroutine `input_map` in its default mode and the facilities `anafast`, `median_filter`, `smoothing`, and `ud_grade` test the value of the `POLCCONV` FITS keyword when reading a polarized map, and interpret the polarization accordingly, as described in the note on POLCCONV in The **HEALPix** Primer.
- `median_filter` facility and `median` subroutine: faster by moving an internal array from heap to stack; do not crash anymore when dealing with empty data sets; slightly different output of `median_filter`: when median is computed over an even number of pixels $n \geq 100$, sorted in $[1, n]$, the output is $d(n/2)$ instead of $(d(n/2) + d(n/2 + 1))/2$ previously. Result remains $d(n/2 + 1)$ for odd n.

IDL

- The routines `ianafast` and `ismoothing` can now use pixel-based quadrature weights. Addition of the supporting functions `nside2npweights` and `unfold_weights`.
- `ianafast` and `ismoothing`: see note on `POLCCONV` in F90 facilities above.
- `change_polcconv` has been improved to allow the change of polarization convention (by changing the sign of U Stokes parameter and updating `POLCCONV` value) in FITS files containing polarized maps generated by standard **HEALPix** tools, as well as for specific formats brewed by the WMAP and Planck projects throughout the years.
- New `help_st` to get information on a structure and its sub-structures
- `azeqview, cartview, gnomview, mollview, orthview` visualization routines:
  - addition of the keywords `CUSTOMIZE` and `DEFAULT_SETTINGS` for extensize customization of the figures produced
  - `GLSIZE` and `IGLSIZE` can now be 2-element vectors to control separately the size (and presence) of labels on the parallel and meridian graticules
  - fine control of polarisation rods thickness with `POLARIZATION`
  - addition of the `SILHOUETTE` keyword to add a tunable silhouette around the projected map (`mollview` and `orthview` only)
- Improved support for GDL and FL (Fawlty Language).
- Update of the required `IDL-astron` library routines, and `Coyote` library routines (2018-05-15).

Python

- Switch to healpy 1.12.0 (CHANGELOG)
- Addition of the python facility src/python/change_polcconv.py to change the polarization convention (see IDL's change_polcconv.pro above).

Bug corrections and Improvements in Version 3.31 (2016-08)

General

- Detailed HOWTO for installation under Windows;
- Interactive configure script now supports MINGW environment (for Windows), and better detects gcc and python versions;
- Improved cross-document linking in PDF documentation.

C++

- Removal of C++11 features inadvertently introduced in Version 3.30 (see https://sourceforge.net/p/healpix/bugs/72)

Fortran 90 facilities and subroutines

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

IDL

- Improved support for GDL;
- update of the required IDL-astron library routines, and Coyote library routines (2016-08-19).

Python

- Switch to healpy 1.9.1 (CHANGELOG)
  - Removed C++ 11 features
  - Streamlined setup.py
  - Plotting fixes for Python 3
  - Numpy 1.10 fix

Bug corrections and Improvements in Version 3.30 (2015-10)

C++

- support for multi-order coverages (MOC);
- allow generation of $a_{\ell m}$ from 6-component power spectra;
- moved from alice2 to alice3, which produces FITS **HEALPix** maps as output. These can be visualized more flexibly with external tools.
- switch from custom xcomplex class to std::complex;
- rangeset class has been redesigned.

Fortran 90 facilities and subroutines

- anafast facility now produces nine spectra (TT, EE, BB, TE, TB, EB, ET, BT and BE), instead of six previously, when analyzing two polarized maps;
- alm2cl subroutine can now produces nine spectra (TT, EE, BB, TE, TB, EB, ET, BT and BE), instead of six previously, when called with two sets of polarized $a_{\ell m}$ and can also symmetrize the output $C(\ell)$ if requested;
- the $a_{\ell m}$ generated by create_alm subroutine can now take into account non-zero (exotic) TB and EB cross-spectra (option polar=2) if the input FITS file contains the relevant information
- new routines nest2uniq and uniq2nest for conversion of standard pixel index to/from Unique ID number. See "The Unique Identifier scheme" section in "**HEALPix** Introduction Document" for more details.
- improved repeat behavior in write_bintabh routine
- edited map2alm_iterative routine to avoid a bug specific to Intel's Ifort 15.0.2
- CFITSIO version 3.20 (August 2009) or more now required;

IDL

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

Java

- deprecated parts of the library have been removed;
- MOC support (see http://ivoa.net/documents/MOC/ for high-level description);
- queries for arbitrary polygons (using MOC);
- new targets in `build.xml` which allow compilation without external JARs.

Python

- switch to `healpy 1.9.0`
  - same C++ source code as **HEALPix** 3.30
  - drop support for Python 2.6
  - support for `astropy.fits`
  - improvements to `read_map` and `write_map`
  - renamed `get_neighbours` to `get_interp_weights`
  - several bug fixes in build and installation processes

Bug corrections and Improvements in Version 3.20 (2014-12)

General

- Generation of `pkg-config` files during the configuration of the C, C++ and F90 packages. See Section 6 of "**HEALPix** Installation" for details.

C

- Top `configure` script now proposes compilation with *or* without CFITSIO-related functions
- Improved `autotools` support

C++

- automatic workaround for bugs in older versions of GNU `g++` compiler (bug reports 37, 45, 48, 51)
- workaround for possible bug in Intel `icc` 14.0 compiler
- bug fix in Mollweide projection in `map2tga` when not looking at (0,0)
- `autotools` updates
- deprecation warnings in `alice2`, soon to be replaced

Fortran 90 facilities and subroutines

- **HEALPix**-F90 routines and facilities can now also be compiled with the free Fortran95 compiler **g95** (http://www.g95.org/). See Section 7.1 of "**HEALPix** Installation" for details.
- A separate `build` directory is used to store the objects, modules, ... produced during the compilation of the source codes
- improved handling of long FITS keywords, now producing FITS files fully compatible with the PyFITS and `Astropy` (https://www.astropy.org/) `Python` libraries
- improved FITS file parsing in `generate_beam`, affecting the external $B(l)$ reading in the F90 facilities `alteralm`, `synfast`, `sky_ng_sim`, `smoothing`.

IDL

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

Java

- explicit deprecation warnings in the source codes

Python

- switch to `healpy 1.8.1`
  - fixes bugs in monopole removal,
  - adds orthographic projection,
  - easier install on MacOSX

Bug corrections and Improvements in Version 3.11 (2013-04)

General

- `libsharp` C library used for Spherical Harmonics Transforms in Fortran and C++ since **HEALPix** 3.10 can now be compiled with *any* `gcc` version.

C++

- See General section above

Fortran 90 facilities and subroutines

- bug correction in `query_disc` routine in `inclusive` mode
- bug correction in `alm2map_spin` routine, which had its `spin` value set to 2
- See General section above

IDL

- `ang2pix_ring` and `pix2ang_nest` routines now accept scalar arguments

Bug corrections and Improvements in Version 3.10 (2013-03)

General

N/A

C

- experimental GNU autotools support (undocumented); the standard configuration script remains available

C++

- Spherical Harmonics Transform library `libpsht` replaced by `libsharp` (Reinecke & Seljebotn, 2013).
  Note that some `gcc` *versions (4.4.1 to 4.4.6) crash with an internal compiler error during compilation of* `libsharp`. The problem has been fixed in `gcc` 4.4.7, 4.5.*, 4.6.*, 4.7.* and newer versions and was not present in versions 4.2.* and 4.3.*.
- added `boundaries()` method to `T_Healpix_Base`
- experimental GNU autotools support (undocumented); the standard configuration script remains available

Fortran 90 facilities and subroutines

- all Fortran facilities now support most of `cfitsio`'s "Extended File Name Syntax" features, allowing the reading and processing of an arbitrary HDU and table column out of remote, compressed FITS files. For example, setting
  `infile = ftp://`*url/file.fits*`.gz[`*extn*`][col` *colname*`]`
  in `anafast` will download the FITS file *file.fits.gz* from *url*, uncompress it, open the HDU (extension) featuring keyword `EXTNAME=`*extn*, or the one with 1-based rank number *extn*, read the table column with `TTYPE*=`*colname* out of it and will analyze it.
  It is also possible to perform a remote `anafast` analysis of a Planck Legacy Archive (PLA) sky map named *map.fits* via the PLA AIO Subsystem by simply setting `infile=https://pla.esac.esa.int/pla/aio/product-action?MAP.MAP_ID=`*map.fits* as input map file.
- yet faster `synfast`, `anafast`, `smoothing` thanks to `libsharp` routines (see warning on gcc releases above).

IDL

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

Java

N/A

Python

- switch to `healpy 1.5.0`: addition of `gauss_beam` to generate Gaussian beam window function.

Bug corrections and Improvements in Version 3.0 (2012-11)

General

Introduction of the script `healpix_doc` for easy access to the **HEALPix** PDF and HTML documentation.

C

- Interface has remained unchanged, but the code has been replaced by a C port of the relevant Healpix C++ functions, resulting in significant speedups.
- Additional functions are provided which support Nside values up to 2^{29}. They have the same name as the traditional functions, with a "64" suffix appended.

C++

- Query routines: query_polygon() and query_polygon_inclusive() added. Query routines now return lists of pixel ranges instead of lists of pixels, which is much more economic. Inclusive query routines: tradeoff between performance and number of false positives is tuneable. Queries now work natively in both NESTED and RING schemes. Operations on the NESTED scheme are typically slower than in RING, but still much faster than computing the query in RING and converting all pixel numbers to NESTED afterwards.
- Healpix_Base: Healpix_Base and Healpix_Base2 have been merged into the templated class T_Healpix_Base; functionality is still available under the old names. Various performance improvements to T_Healpix_Base functionality
- User-friendliness: module parameters can now optionally be passed on the command line instead of using a parameter file. For example:
  `anafast_cxx nlmax=500 infile=test.fits iter_order=3` ⟨...⟩
  Facilities now check input maps for undefined pixels before calling map2alm(). If undefined pixels are found, a warning is printed, and the pixels are set to zero. udgrade_cxx refuses downgrading of polarised maps (which would produce unphysical results)
- Bug fixes: accuracy of pix2ang near the poles at high resolutions has been improved.
- Configuration: optional autoconf support
- Interface changes:
  - Healpix_Base::query_*(): new interface
  - cxxutils.h has been split up into announce.h (dealing with module banners), share_utils.h (dealing with subdividing tasks between multiple workers) and string_utils.h (dealing with string manipulation and file parsing)
  - psht.h: interface to alm_info changed in order to add MPI support
  - ylmgen_c.h: Ylmgen_init() interface has changed
  - bluestein.h: bluestein_i() interface changed

Fortran 90 facilities and subroutines

- Compressed and/or remote (ftp or http) FITS files can now be read. CFITSIO 3.14 or more is now required;
- introduction of the process_mask facility to compute the angular distance of valid pixels to the closest invalid pixels for a input binary mask, and of the supporting routines dist2holes_nest, fill_holes_nest, maskborder_nest, size_holes_nest;
- the pixel query routine query_disc has been improved and will return fewer false positive pixels in the inclusive mode;
- improved accuracy of the co-latitude calculation in the vicinity of the poles at high resolution in nest2ring, ring2nest, pix2ang_*, pix2vec_*, ...;
- sky_ng_sim now allows the computation of the spatial derivatives of the non Gaussian map being produced, and the output of the $a_{\ell m}$ coefficients of that map;
- anafast now allows the pro/down-grading of the input mask to match the resolution of the map(s) being analyzed;
- the median filter routine medfiltmap, used by the facility median_filter is now parallelized.

IDL

- New routines to go from circular beam profile to transfer function (beam2bl), and back (bl2beam); to go from indexed list of $a_{\ell m}$ to a(l,m) 2D table (alm_i2t), and back (alm_t2i); and to compute the angular distance between pairs of vectors (angulardistance).
- addition of iprocess_mask interface to F90 `process_mask` facility to compute the angular distance of valid pixels to the closest invalid pixels for a input binary mask.
- creation of hpx2dm routine to generate DomeMaster images of **HEALPix** maps that can be projected on planetariums.
- the pixel query routines query_triangle, query_polygon, and in particular query_disc, have been improved and will return fewer false positive pixels in the *inclusive* mode
- improved accuracy of the co-latitude calculation in the vicinity of the poles at high resolution in nest2ring, ring2nest, pix2ang_*, pix2vec_*, ...
- cartview, gnomview, mollview, orthview: the length and spacing of the headless vectors used to represent polarization is now user-controlled via `POLARIZATION` keyword. The `COLT` keyword now allows the use of an interactively modified color table.
- orthview now accepts `STAGGER` keyword to overplot staggered spheres (with a twist) in order to detect periodic boundary conditions on the sky
- fits2cl: addition of `WMAP7` keyword to read best fit $C(l)$ model to WMAP 7yr data.
- read_fits_map can now read $N_{\rm side}$=8192 **HEALPix** maps and is generally faster than previously for smaller maps
- update of `astron` library routines (01-Feb-2012).

Java

- Core functionality has been reimplemented from scratch in the form of the "healpix.essentials" package. It is strongly recommended to use this package directly in future projects making use of Java **HEALPix**. "healpix.essentials" is a port of the Healpix C++ library and presents a very similar interface.

  The "healpix.core" package is still provided. It uses "healpix.essentials" internally, and its interface has been kept stable as much as possible. Some adaptations in user code will still be necessary, however. Please note that using "healpix.core" will result in slightly lower performance than calling "healpix.essentials" methods directly, because of the necessary data conversion.
- New features and improvements introduced with the HealpixBase class, compared to the HealpixIndex, Healpix and PixTools classes:
  - close similarities with Healpix_Base_T class from Healpix C++, which allows simultaneous development and bug fixes for both.
  - support for arbitrary positive Nside values in RING scheme; no longer limited to powers of 2
  - maximum supported Nside value: 2^{29}
  - significant performance improvements: most methods have been accelerated by integral factors, some by more than an order of magnitude.
  - re-implementation of queryDisc and queryPolygon, with same new features as the C++ implementation (see above).
  - the HealpixProc class offers a procedural (instead of object-oriented) interface to the HealpixBase functionality, which simplifies transition for users of the "Healpix" and "PixTools" classes. NOTE: this only works for Nside parameters which are powers of 2
  - many bug fixes
  - no external library dependencies, except for "nom.tam.fits" if FITS I/O is required

Python

- the healpy package (C. Rosset, A. Zonca et al.) is now part of **HEALPix**