SHTOOLS - Tools for working with spherical harmonics

Version 4.1

Last generated: December 26, 2017

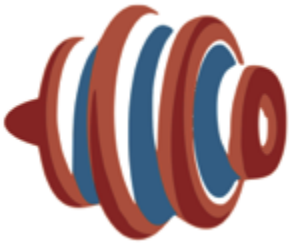


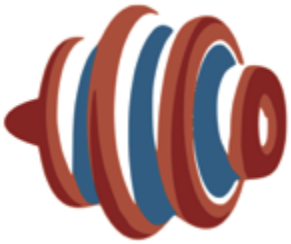
Table of Contents

Support

Release Notes

SHTOOLS - Tools for working with spherical harmonics

Summary: SHTOOLS is an archive of Python and Fortran 95 software that can be used to perform spherical harmonic transforms and reconstructions, rotations of data expressed in spherical harmonics, and multitaper spectral analyses on the sphere.



Follow @pyshtools

Features

SHTOOLS is extremely versatile:

- All standard normalizations of the spherical harmonic functions are supported (4π normalized, Schmidt semi-normalized, orthonormalized, and unnormalized).
- Both real and complex spherical harmonics are supported.
- Spherical harmonic transforms are calculated by exact quadrature rules using either the sampling theorem of *Driscoll and Healy* (1994) or Gauss-Legendre quadrature.
- One can choose to use or exclude the Condon-Shortley phase factor of $(-1)^m$ with the associated Legendre functions.
- Localized multitaper spectral analyses are easily performed.
- Routines are included for performing standard gravity and magnetic field calculations.
- The Fortran routines are OpenMP compatible and OpenMP thread-safe.
- The spherical harmonic transforms are accurate to approximately degree 2800.

- The routines are fast. Spherical harmonic transforms and reconstructions take on the order of 1 second for bandwidths close to 800 and about 30 seconds for bandwidths close to 2600.

Installation

The Python components of SHTOOLS can be installed using the Python package manager `pip`. Binaries are pre-built for linux, macOS, and windows architectures, and you need only to execute the following command in a unix terminal:

```
pip install pyshtools
```

To install the Fortran 95 components for use in your Fortran programs, installation can be as simple as executing the following command in the SHTOOLS directory

```
make fortran
make fortran-mp  # (for OpenMP)
```

or by using the brew package manager (macOS)

```
brew tap shtools/shtools
brew install shtools
```

Using

SHTOOLS can be invoked in any Fortran 95 or Python program. The core software is written in Fortran 95, and Python wrappers allow simple access to the fortran-compiled routines. A variety of Python notebooks and example files are included that demonstrate the major features of the library.

Acknowledgments

SHTOOLS is open source software (3-clause BSD license) and makes use of the freely available packages FFTW , LAPACK and BLAS .

Installing SHTOOLS

Summary: SHTOOLS can be installed in several ways. If you will be using only the Python components, you should use the pip package manager. If you will be writing and compiling Fortran 95 code, you should use either the brew package manager (on macOS) or compile manually using the Makefile.

Python package installer (pip)

The easiest way to install the Python components of SHTOOLS (pyshtools) is to use `pip`. On Linux, macOS and windows machines, the binary wheels can be installed by executing the following command:

```
pip install pyshtools
```

If you wish to instead compile the archive yourself, first make sure that you have the necessary dependencies installed. On most linux distributions, this can be accomplished using

```
sudo apt-get install libblas-dev liblapack-dev g++ gfortran lib
fftw3-dev tcsh
```

or on macOS using brew

```
brew install fftw --with-fortran
```

Then clone the SHTOOLS repo

```
git clone https://github.com/SHTOOLS/SHTOOLS.git
```

To install pyshtools in the active Python environment lib folder, execute this command in the main directory:

```
pip install .
```

To instead install the files in the current working directory and link them to the system Python directory, use

```
pip install -e .
```

To uninstall pyshtools from your system directory, use the command

```
pip uninstall pyshtools
```

Note that these commands will install only the Python version that corresponds to the version of `pip` being used. On some systems, it may be necessary to specify `pip2.7` or `pip3.x`.

Fortran 95 library using brew (macOS)

If brew is already installed, it is only necessary to enter the following commands in the terminal:

```
brew tap shtools/shtools
brew install shtools
```

To also install both the standard Fortran 95 library along with the OpenMP components, add the option `--with-openmp` to the last command:

```
brew install shtools --with-openmp
```

Fortran 95 library using the Makefile

Before installing the Fortran 95 components of SHTOOLS, it will be necessary to have a Fortran 95 compiler and the FFTW, LAPACK, and BLAS libraries installed on your computer. After this is done, the Fortran 95 components of SHTOOLS can be compiled in most cases by executing the following command in a unix shell in the main directory:

```
make
```

To compile the Fortran 95 components with OpenMP use

```
make fortran-mp
```

To compile and run the Fortran 95 test suites, use

```
make fortran-tests
```

To delete the compiled archive, module files, object files, and test suite files, use

```
make clean
```

By default, the `Makefile` will use the `gfortran` compiler. Different compilers and options can be specified by passing optional arguments to the `Makefile` using the syntax

```
make F95 = "MyCompiler" F95FLAGS = "MyCompilerFlags"
```

where "`MyCompiler`" and "`MyCompilerFlags`" are to be replaced by the path of the compiler name and options, respectively. Supported options include:

```
F95 = "Name (including path) of the f95 compiler"
F95FLAGS = "F95 compiler options"
FFTW = Name and path of the FFTW3 library of the form "-Lpath
-lfftw3"
LAPACK = Name and path of the LAPACK library of the form "-Lpat
h -llapack"
BLAS = Name and path of the BLAS library of the form "-Lpath -l
blas"
```

Successful compilation will create the library file `libSHTOOLS.a` (and `libSHTOOLS-mp.a` when compiling with OpenMP) in the directory `lib`, and will place a few compiled module files in the directory `modules`. If you need to

recompile SHTOOLS a second time using a different set of compiler flags, it will be necessary to first remove all the previously compiled object files by using `make clean`.

To make all files available at a system level, execute

```
make install
```

This will move the compiled SHTOOLS files and documentation to

```
SYSLIBPATH  # libSHTOOLS.a, libSHTOOLS-mp.a
SYSMODPATH  # fftw3.mod, planetsconstants.mod, shtools.mod
SYSSHAREPATH/shtools/examples  # example files
SYSSHAREPATH/man/man1  # man pages
SYSDOCPATH/shtools  # index.html, web documentation
```

The locations of the above directories can be set as optional arguments passed to the `Makefile`, and the default locations are

```
SYSLIBPATH = /usr/local/lib
SYSMODPATH = /usr/local/include
SYSSHAREPATH = /usr/local/share
SYSDOCPATH = /usr/local/share/doc
```

To remove all installed SHTOOLS files, use

```
make uninstall
```

To access the unix man pages, it will be necessary to add `SYSSHAREPATH/man` to your man path.

Fortran 95 compiler specific flags and optimizations

Default compiler options are specified in the main `Makefile` for a few common compilers (`gfortran`, Absoft `f95`, `g95`, and `ifort`). If it is necessary to change these, consider the following guidelines.

One should always use some form of optimization when compiling SHTOOLS, such as by specifying the option

```
-O3
```

Performance will likely be increased by 10s of percent by specifying the compiler to target the host architecture

```
-march=host  # f95
-march=native  # gfortran
```

and to use fast math

```
-speed_math=11  # f95
-ffast-math  # gfortran
```

The biggest difficulty in compiling SHTOOLS is setting the compiler flags so that the external subroutine names are in a format that is compatible with the already compiled FFTW and LAPACK libraries. In general, it is necessary to ensure that the SHTOOLS subroutine names are in lower case and have the right number of underscores appended to them.

For Absoft ProFortran, this is achieved by setting

```
-YEXT_NAMES=LCS -YEXT_SFX=_
```

For `g95`, it will be necessary to use one of the following:

```
-fno-second-underscore # most likely
-fno-underscoring
```

For `gfortran`, it is generally not necessary to use any special flags, though it could arise that one of the following might be necessary:

```
-fno-underscoring
-fsecond-underscore
```

For the Intel Fortran compiler `ifort`, it will be necessary to use

```
-free -Tf
```

in order that the compiler recognizes files with the extension `.f95` as Fortran 95 files. In this case, the f95 file should come directly after the option `-Tf`.

Setting the right compiler flags is more complicated when the FFTW and LAPACK libraries have different naming and underscoring conventions. In order to accommodate this case, underscores can be added explicitly to either the LAPACK or FFTW subroutine names in the SHTOOLS source code by specifying the optional `make` arguments when building the archive:

```
FFTW_UNDERSCORE = 1  # to add an extra underscore to the FFTW r
outine names
LAPACK_UNDERSCORE = 1  # to add an extra underscore to the LAPA
CK routine names
```

For both cases, compiler flags should probably be set so that underscores are not appended to routine names. See Fortran 95 problems (page 71) for further information.

To generate 64 bit code, use the compiler option

```
-m64
```

For this case, it will be necessary to use 64-bit compiled FFTW and LAPACK libraries.

Using SHTOOLS with Fortran 95

Summary: SHTOOLS subroutines and functions can be accessed easily from any Fortran 95 program. It is only necessary to use the SHTOOLS module and link to the compiled archive.

Calling SHTOOLS routines

To access the SHTOOLS functions and subroutines in your code, it is necessary to place the statement

```
use SHTOOLS
```

directly after the program, subroutine, or function declaration (i.e., before an `implicit none` statement). The SHTOOLS module contains an interface block that declares the subroutines and functions used in this archive and allows for the use of implicitly shaped arrays. It should be noted that all arrays passed to a subroutine or function can be *larger* than that specified in the documentation.

Linking to the SHTOOLS library

When compiling a code that references SHTOOLS, it will be necessary to link to the SHTOOLS library and module files. For this purpose, it may be useful to define the environment variables `SHTOOLSMODPATH` and `SHTOOLSLIBPATH` using the default file locations. In a C-shell, this is accomplished using

```
setenv SHTOOLSLIBPATH = "/usr/local/lib"
setenv SHTOOLSMODPATH = "/usr/local/include"
```

whereas for a bash shell the syntax is

```
export SHTOOLSLIBPATH = "/usr/local/lib"
export SHTOOLSMODPATH = "/usr/local/include"
```

In addition, most routines require linking to the fast Fourier transform package FFTW , and the linear algebra packages LAPACK and BLAS . Typical examples of compiling and linking a program `MyProgram.f95` to the necessary library and module files are given below for several common compilers.

gfortran

```
gfortran MyProgram.f95 -I$SHTOOLSMODPATH -L$SHTOOLSLIBPATH -lSH
TOOLS -lfftw3 -lm -llapack -lblas -O3 -m64 -o MyProgram
```

Absoft Pro Fortran (f95)

```
f95 MyProgram.f95 -p $SHTOOLSMODPATH -L$SHTOOLSLIBPATH -YEXT_NA
MES=LCS -YEXT_SFX=_ -lSHTOOLS -lfftw3 -lm -llapack -lblas -O3
-m64 -o MyProgram
```

g95

```
g95 MyProgram.f95 -I$SHTOOLSMODPATH -L$SHTOOLSLIBPATH -fno-seco
nd-underscore -lSHTOOLS -lfftw3 -lm -llapack -lblas -O3 -m64
-o MyProgram
```

Intel Fortran (ifort)

```
ifort -fpp -free -I$SHTOOLSMODPATH -L$SHTOOLSLIBPATH -lSHTOOLS
-lfftw3 -lm -llapack -lblas -O3 -m64 -Tf MyProgram.f95 -o MyPro
gram
```

Note that the position of the source file in the above examples might be important for some compilers. It may be necessary to modify some options in order to properly link to both the LAPACK and FFTW libraries (see installing SHTOOLS (page 5) for more details). If the library ATLAS exists on your system, this could be used instead of BLAS.

OpenMP

The Fortran 95 routines in the OpenMP version of SHTOOLS are OpenMP compatible and OpenMP thread-safe. To use these routines in a program that uses OpenMP, it is necessary to link to the library `libSHTOOLS-mp.a` and to add one of the following compiler options:

```
-fopenmp  # gfortran
-openmp  # Absoft Pro Fortran
-qopenmp  # ifort
```

Fortran array indices

Fortran arrays usually start with an index of 1. However, with spherical harmonic functions and coefficients a degree-0 term exists. In order to deal with this, all arrays that are a function of spherical harmonic degree `l` and order `m` have 1 added to each of their indices. For instance, the real Fortran array `clm` is related to the spherical harmonic coefficients by

In this notation, the positive and negative angular orders correspond to the cosine and sine coefficients, respectively (see the page real spherical harmonics (page 18) for more information).

Using optional parameters

Many of the subroutines and functions in this archive can accept one or more optional parameters. To specify these parameters, it is only necessary to use the syntax

```
call SHRead (FILENAME, CILM, LMAX, SKIP=1, ERROR=errorcoef)
```

where `SKIP` and `ERROR` are the names of two optional parameters, and the constant `1` and variable `errorcoef` are their respective arguments.

Documentation

Documentation for the Fortran 95 subroutines and functions in SHTOOLS can be accessed by their unix man pages, using all lower case letters. As an example, to access the `MakeGridDH` man page, use

```
man makegriddh
```

Alternatively, the man pages can be accessed from the *Fortran 95 Routines* menu on this web site.

Using SHTOOLS in Python

Summary: To use SHTOOLS in Python, it is only necessary to import the pyshtools module.

The pyshtools module

To use SHTOOLS in Python, it is only necessary to execute this statement in the Python environment

```
import pyshtools
```

This will load the following three classes and subpackages into the `pyshtools` namespace:

Class	Description
SHCoeffs (page 0)	A high-level class for spherical harmonic coefficients
SHGrid (page 0)	A high level-class for global grids
SHWindow (page 0)	A high-level class for localization windows

Subpackage	Description
shclasses (page 48)	All pyshtools classes and subclasses
shtools	All Python wrapped Fortran 95 routines
legendre (page 49)	Legendre functions
expand (page 51)	Spherical harmonic expansion routines
shio (page 53)	Spherical harmonic I/O, storage, and conversion routines

Subpackage	Description
spectralanalysis (page 55)	Global and localized spectral analysis routines
rotate (page 58)	Spherical harmonic rotation routines
gravmag (page 59)	Gravity and magnetics routines
constant (page 62)	pyshtools constants
utils (page 61)	Utilities

If you are using iPython , which adds improved functionality to Python, the available `pyshtools` routines can be explored by typing

```
pyshtools.[tab]
```

where `[tab]` is the tab key.

Documentation

To read the documentation of a routine in iPython, such as `MakeGridDH` , enter

```
pyshtools.expand.MakeGridDH?
```

To read the info string of an SHTOOLS constant, such as `a_mars` , enter

```
pyshtools.constant.a_mars.info()
```

Documentation for the Python functions used in SHTOOLS can also be accessed by their unix man pages, appending `py` to the name and using all lower case letters. As an example, to access the python `MakeGridDH` man page, use

```
man pymakegriddh
```

Alternatively, the man pages can be accessed from the *Python components* menu item on this web site.

Real spherical harmonics

Summary: SHTOOLS uses by default 4π-normalized spherical harmonic functions that exclude the Condon-Shortley phase factor. Schmidt semi-normalized, orthonormalized, and unnormalized harmonics can be employed in most routines by specifying optional parameters.

Definitions: Real -normalized harmonics

Any real square-integrable function can be expressed as a series of spherical harmonic functions

\begin{equation} f\left(\theta,\phi\right) = \sum_{l=0}^{\infty} \sum_{m=-l}^l f_{lm} \, Y_{lm}\left(\theta,\phi \right), \label{eq:f} \end{equation}

where is the spherical harmonic coefficient, is the corresponding spherical harmonic function, is co-latitude, is longitude, and and are the spherical harmonic degree and order, respectively. The real spherical harmonics are defined as

where the normalized associated Legendre functions for use with the -normalized spherical harmonic functions are given by

and where is the Kronecker delta function. The unnormalized associated Legendre functions are derived from the standard Legendre polynomials using the relations \begin{equation} P_{lm}(\mu) = \left(1-\mu^2\right)^{m/2} \frac{d^m}{d\mu^m} P_l(\mu) \end{equation} and

\begin{equation} P_l(\mu) = \frac{1}{2^l l!}\frac{d^l}{d\mu^l}\left(\mu^2-1\right)^l. \end{equation}

The normalized associated Legendre functions are orthogonal for a given value of

,

and the spherical harmonic functions are orthogonal for all degrees and orders

\begin{equation} \int_\Omega Y_{lm}(\theta,\phi) \, Y_{l'm'}(\theta,\phi) \, d\Omega = 4 \pi \, \delta_{ll'}\, \delta_{mm'}, \end{equation} where is the differential surface area on the unit sphere, . By multiplying equation \eqref{eq:f} by and integrating over all space, it is straightforward to show that the spherical harmonic coefficients of a function can be calculated by the integral \begin{equation} f_{lm} = \frac{1}{4\pi} \int_\Omega f(\theta,\phi) \, Y_{lm}(\theta,\phi) \, d\Omega. \end{equation}

Power spectrum

Parseval's theorem in Cartesian geometry relates the integral of a function squared to the sum of the squares of the function's Fourier coefficients. This relation is easily extended to spherical geometry using the orthogonality properties of the spherical harmonic functions. Defining *power* to be the integral of the function squared divided by the area it spans, the total power of a function is equal to a sum over its power spectrum \begin{equation} \frac{1}{4\pi} \int_\Omega f^2(\theta,\phi) \, d\Omega = \sum_{l=0}^{\infty} S_{ff}(l), \end{equation} where the power spectrum is related to the spherical harmonic coefficients by

Similarly, the cross power of two functions and is given by \begin{equation} \frac{1}{4\pi} \int_\Omega f(\theta,\phi) \, g(\theta,\phi)\, d\Omega = \sum_{l=0}^{\infty} S_{fg}(l), \end{equation} with

The power spectrum is unmodified by a rotation of the coordinate system. Furthermore, the numerical values of the power spectrum are independent of the normalization convention used for the spherical harmonic functions (though the mathematical formulae will be different, as given below (page 20)). If the functions and have a zero mean, and represent the contribution to the variance and covariance, respectively, as a function of degree .

is the total power of the function at spherical harmonic degree , which in SHTOOLS is called the *power per degree* . Alternatively, one can calculate the average power per coefficient at spherical harmonic degree , which in SHTOOLS is referred to as the *power per* . Since there are spherical harmonic coefficients at degree , this is \begin{equation} \mbox{power per lm} = \frac{S(l)}{(2l+1)}. \end{equation} One can also calculate the power from all angular orders over an infinitesimal logarithmic spherical harmonic degree band , where is the logarithmic base. In SHTOOLS, this is referred to as the *power per* , which is given by \begin{equation} \mbox{power per $d\log_a l$} = S(l)\, l \, \ln a. \end{equation} Finally, SHTOOLS defines the *energy* of a function as the integral of its square. The energy spectrum is thus equal to the power spectrum multiplied by .

Condon-Shortley phase factor

The above definitions of the Legendre functions and spherical harmonic functions do not include the Condon-Shortley phase factor of that is often employed in the physics and seismology communities [Varshalovich et al. 1988, Dahlen and Tromp 1998]. Nevertheless, this phase can be included in most SHTOOLS routines by specifying the optional parameter

- `csphase = 0` : exclude the Condon-Shortley phase factor (default)
- `csphase = 1` : append the Condon-Shortley phase factor to the Legendre functions.

The choice of the Condon-Shortley phase factor does not affect the numerical value of the power spectrum.

Supported normalizations

SHTOOLS supports the use of -normalized, Schmidt semi-normalized, orthonormalized, and unnormalized spherical harmonic functions. To specify which normalization should be used, it is only necessary to specify the optional parameter `norm` in the Fortran 95 routines or `normalization` in the Python routines:

- `norm = 1`, `normalization = '4pi'`: normalized (default, unless stated otherwise)
- `norm = 2`, `normalization = 'schmidt'`: Schmidt semi-normalized
- `norm = 3`, `normalization = 'unnorm'`: Unnormalized
- `norm = 4`, `normalization = 'ortho'`: Orthonormalized.

Each of these normalizations has slightly different definitions for the normalized Legendre functions, the orthogonality conditions of the Legendre functions and spherical harmonic functions, and the power spectrum. These equations are provided below.

normalized

Schmidt semi-normalized

Orthonormalized

Unnormalized

References

- Dahlen, F. A. and J. Tromp, “Theoretical Global Seismology,” *Princeton University Press*, Princeton, New Jersey, 1025 pp., 1998.
- Varshalovich, D. A., A. N. Moskalev, and V. K. Khersonskii, “Quantum theory of angular momentum,” *World Scientific*, Singapore, 1988.

Complex spherical harmonics

Summary: SHTOOLS uses by default 4π-normalized spherical harmonic functions that exclude the Condon-Shortley phase factor. Schmidt semi-normalized, orthonormalized, and unnormalized harmonics can be employed in most routines by specifying optional parameters.

Definitions: Complex -normalized harmonics

Any real square-integrable function can be expressed as a series of spherical harmonic functions

\begin{equation} f\left(\theta,\phi\right) = \sum_{l=0}^{\infty} \sum_{m=-l}^l f_l^m \, Y_l^m\left(\theta,\phi \right), \label{eq:f-complex} \end{equation}

where is the complex spherical harmonic coefficient, is the corresponding complex spherical harmonic function, is co-latitude, is longitude, and and are the spherical harmonic degree and order, respectively. The complex spherical harmonics are defined as \begin{equation} Y_l^m(\theta,\phi) = \bar{P}_l^m(\cos \theta) \, e^{im\phi}, \end{equation}

where the normalized associated Legendre functions for use with the complex -normalized spherical harmonic functions are given by

and where is the Kronecker delta function. The unnormalized associated Legendre functions are derived from the standard Legendre polynomials using the relations \begin{equation} P_{lm}(\mu) = \left(1-\mu^2\right)^{m/2} \frac{d^m}{d\mu^m} P_l(\mu) \end{equation} and

\begin{equation} P_l(\mu) = \frac{1}{2^l l!}\frac{d^l}{d\mu^l}\left(\mu^2-1\right)^l. \end{equation}

The normalized associated Legendre functions are orthogonal for a given value of ,

The complex spherical harmonic functions possess the symmetry relationship for positive and negative angular orders

where the asterisk denotes complex conjugation, and they satisfy the orthogonality relationship \begin{equation} \int_\Omega Y_l^m(\theta,\phi) \, Y_{l'}^{m'}(\theta,\phi) \, d\Omega = 4 \pi \, \delta_{ll'} \, \delta_{mm'}, \end{equation} where is the differential surface area on the unit sphere, . By multiplying the equation \eqref{eq:f-complex} by and integrating over all space, it

is straightforward to show that the spherical harmonic coefficients of a function can be calculated by the integral \begin{equation} f_l^m = \frac{1}{4\pi} \int_\Omega f(\theta,\phi) \, Y_l^{*m}(\theta,\phi) \, d\Omega. \end{equation}

Finally, it is noted that if a function defined on the sphere is entirely real, then the real and complex spherical harmonic coefficients are related by

Power spectrum

Parseval's theorem in Cartesian geometry relates the integral of a function squared to the sum of the squares of the function's Fourier coefficients. This relation is easily extended to spherical geometry using the orthogonality properties of the spherical harmonic functions. Defining *power* to be the integral of the function squared divided by the area it spans, the total power of a function is equal to a sum over its power spectrum \begin{equation} \frac{1}{4\pi} \int_\Omega |f|^2(\theta,\phi) \, d\Omega = \sum_{l=0}^{\infty} S_{ff}(l), \end{equation} where the power spectrum is related to the spherical harmonic coefficients by

Similarly, the cross power of two functions and is given by \begin{equation} \frac{1}{4\pi} \int_\Omega f(\theta,\phi) \, g^*(\theta,\phi)\, d\Omega = \sum_{l=0}^{\infty} S_{fg}(l), \end{equation} with

The power spectrum is unmodified by a rotation of the coordinate system. Furthermore, the numerical values of the power spectrum are independent of the normalization convention used for the spherical harmonic functions (though the mathematical formulae will be different, as given below (page 24)). If the functions and have a zero mean, and represent the contribution to the variance and covariance, respectively, as a function of degree . It should be noted that while the power spectrum of a function is inherently real, the cross power of two functions may be a complex quantity.

is the total power of the function at spherical harmonic degree , which in SHTOOLS is called the *power per degree* . Alternatively, one can calculate the average power per coefficient at spherical harmonic degree , which in SHTOOLS is referred to as the *power per* . Since there are spherical harmonic coefficients at degree , this is \begin{equation} \mbox{power per lm} = \frac{S(l)}{(2l+1)}. \end{equation} One can also calculate the power from all angular orders over an infinitesimal logarithmic spherical harmonic degree band , where is the logarithmic base. In SHTOOLS, this is referred to as the *power per* , which is given by \begin{equation} \mbox{power per $d\log_a l$} = S(l)\, l \, \ln a. \end{equation} Finally, SHTOOLS defines the *energy* of a function as the integral of its square. The energy spectrum is thus equal to the power spectrum multiplied by .

Condon-Shortley phase factor

The above definitions of the Legendre functions and spherical harmonic functions do not include the Condon-Shortley phase factor of that is often employed in the physics and seismology communities [Varshalovich et al. 1988, Dahlen and Tromp 1998]. Nevertheless, this phase can be included in most SHTOOLS routines by specifying the optional parameter

- `csphase = 0` : exclude the Condon-Shortley phase factor (default)
- `csphase = 1` : append the Condon-Shortley phase factor to the Legendre functions.

The choice of the Condon-Shortley phase factor does not affect the numerical value of the power spectrum.

Supported normalizations

SHTOOLS supports the use of -normalized, Schmidt semi-normalized, orthonormalized, and unnormalized spherical harmonic functions. To specify which normalization should be used, it is only necessary to specify the optional parameter `norm` in the Fortran 95 routines or `normalization` in the Python routines:

- `norm = 1`, `normalization = '4pi'`: normalized (default, unless stated otherwise)
- `norm = 2`, `normalization = 'schmidt'`: Schmidt semi-normalized
- `norm = 3`, `normalization = 'unnorm'`: Unnormalized
- `norm = 4`, `normalization = 'ortho'`: Orthonormalized.

Each of these normalizations has slightly different definitions for the normalized Legendre functions, the orthogonality conditions of the Legendre functions and spherical harmonic functions, and the power spectrum. These equations are provided below.

normalized

Schmidt semi-normalized

Orthonormalized

Unnormalized

References

- Dahlen, F. A. and J. Tromp, “Theoretical Global Seismology,” *Princeton University Press*, Princeton, New Jersey, 1025 pp., 1998.
- Varshalovich, D. A., A. N. Moskalev, and V. K. Khersonskii, “Quantum theory of angular momentum,” *World Scientific*, Singapore, 1988.

Implementation details

Summary: The spherical harmonic transforms in SHTOOLS make use of integrations over longitude that involve fast Fourier transforms and integrations over latitude that utilize either Gauss-Legendre quadrature or exact quadrature rules for regularly spaced grids. The transforms and reconstructions are accurate to about degree 2800.

Numerical computations

Spherical harmonic reconstructions

If the spherical harmonic coefficients of a function were known up to a maximum degree , the function could be evaluated at arbitrary colatitude and longitude using the equation

\begin{equation} f\left(\theta,\phi\right) = \sum_{l=0}^{L} \sum_{m=-l}^l f_{lm} \, Y_{lm}\left(\theta,\phi \right). \end{equation}

Using the separate variables and for the cosine and sine spherical harmonic coefficients, respectively, and after interchanging the order of summations over and , the function can be expressed equivalently as

Defining the two component vector

The function can be written more simply as

For a given latitude band, the function can thus be evaluated on a series of grid nodes all at the same time using an inverse fast Fourier transform. For this operation, SHTOOLS makes use of the highly optimized software package FFTW [Frigo and Johnson 2005] that supports grids of arbitrary length and both real and complex data. To adequately sample the function in longitude, a minimum of data points should be employed.

The coefficients and depend upon the associated Legendre functions, which are calculated efficiently using standard three-term recursion relations over adjacent spherical harmonic degrees. For a given colatitude, the sectoral term is first calculated using an analytic equation, and then is calculated for all values of . To circumvent numerical underflows near the poles for large values of , the associated Legendre functions are calculated using the approach of Holmes and Featherstone [2002], where the sectoral terms are multiplied by prior to performing the recursions, and then appropriately unscaled at the end of the recursion. This ensures that the Legendre functions are accurate to about degree 2800.

Spherical harmonic transforms

The spherical harmonic coefficients of a function can be calculated from the relation

Defining the two intermediary variables and

it is seen that for a given colatitude and degree , all of the angular orders can be calculated at once by making use of a fast Fourier transform of the function . Replacing the integral over latitude with a numerical quadrature rule, the spherical harmonic coefficients can be calculated as

Here, is the latitudinal weight, and is the number of latitudinal points over which the integration is performed.

Supported grid formats

It is possible to choose the weights and the locations of the latitudinal sampling points such that the quadrature in equation \eqref{eq:quadrature} is exact. SHTOOLS makes use of two grid formats that accommodate exact quadrature using Gauss-Legendre quadrature [e.g., Press et al. 1992], and exact quadrature using regular grids sampled according to the *Driscoll and Healy* (1994) theorem. In both techniques, the quadrature is exact only when the function being integrated is a terminating polynomial. The functions and can be approximated as polynomials of maximum degree , and when multiplied by the associated Legendre function, the integrand is approximately a polynomial of maximum degree . The following table summarizes the properties of the different types of grids supported by SHTOOLS.

Acronym	Name	Shape ()
DH	Driscoll and Healy	
DH2	Driscoll and Healy	
GLQ	Gauss-Legendre Quadrature	

Gauss-Legendre Quadrature

For the case of Gauss-Legendre quadrature (`GLQ`), the quadrature is exact when the function is sampled in latitude at the zeros of the Legendre Polynomial of degree . Since the function also needs to be sampled on equally space grid nodes for the Fourier transforms in longitude, the function is sampled on a grid of size .

Driscoll and Healy [1994]

The second type of grid is for data that are sampled on regular grids. As shown by *Driscoll and Healy* [1994], an exact quadrature exists when the function is sampled at equally spaced nodes in latitude and equally spaced nodes in longitude. For this sampling (`DH`), the grids make use of the longitude band at 90 N, but not 90 S, and the number of samples is , which is always even. Given that the sampling in latitude was imposed a priori, these grids contain almost twice as many samples in latitude as the grids used with Gauss-Legendre quadrature.

For geographic data, it is common to work with grids that are equally spaced in degrees latitude and longitude. SHTOOLS provides the option of using grids of size , and when performing the Fourier transforms for this case (`DH2`), the coefficients and with are discarded.

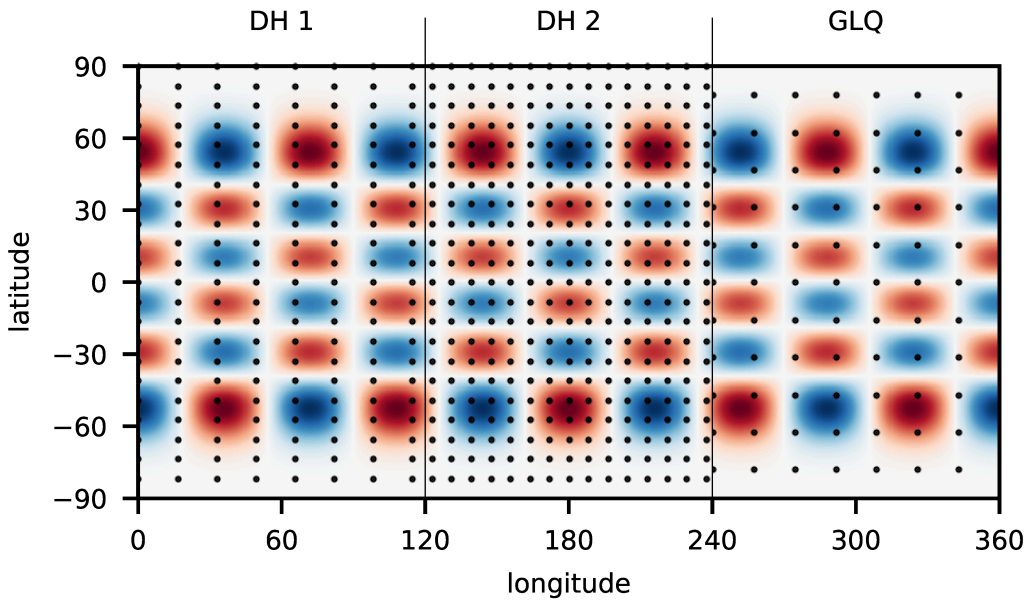


Figure 1. Schematic diagram illustrating the properties of the grids used with the Gauss-Legendre quadrature and Driscoll and Healy routines in SHTOOLS.

Accuracy

To test the accuracy of the spherical harmonic transforms and reconstructions, two sets of synthetic spherical harmonic coefficients were created. Each coefficient was chosen to be a random Gaussian distributed number with unit variance, and the coefficients were then scaled such that the power spectrum was

proportional to either or . For a given spherical harmonic bandwidth , the function was reconstructed in the space domain using the Gauss-Legendre quadrature implementation and then re-expanded into spherical harmonics. The maximum and root-mean square (rms) relative errors between the initial and final set of coefficients were then computed as a function of spherical harmonic degree, as plotted in the figure below.

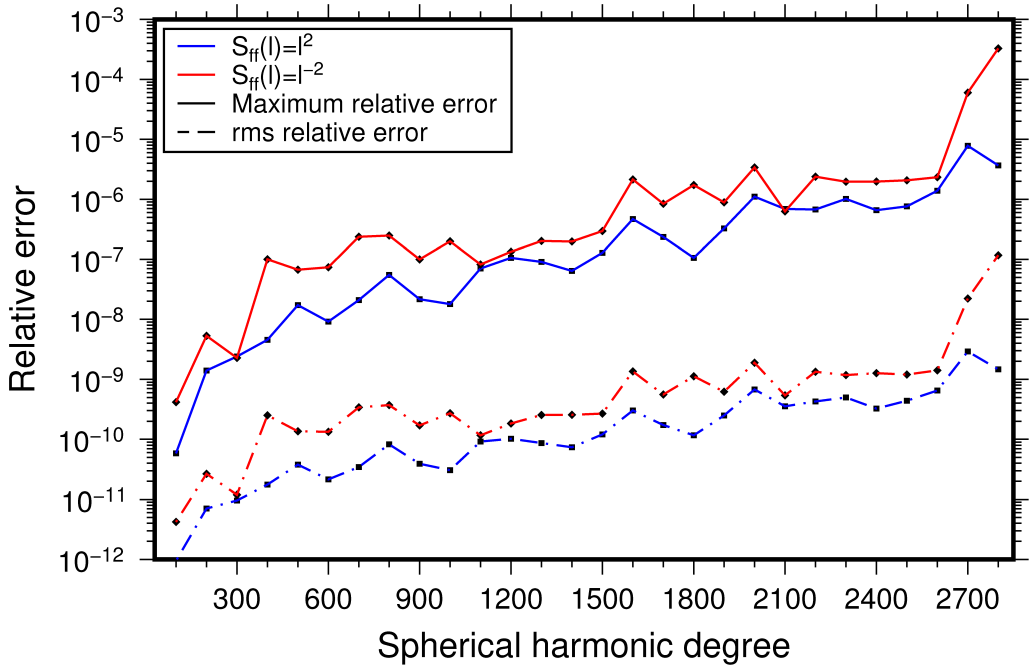


Figure 2. Maximum and rms relative errors of the spherical harmonic coefficients as a function of spherical harmonic bandwidth following a reconstruction of the function in the space domain and a subsequent spherical harmonic transform.

The errors associated with the transform and inverse pair increase in an quasi-exponential manner, with the maximum relative error being approximately 1 part in a billion for degrees close to 400, and about 1 part in a million for degrees close to 2600. Though the errors are negligible to about degree 2600, they then grow somewhat between degrees 2700 and 2800. The rms errors for the coefficients as a function of the bandwidth are typically three orders of magnitude smaller than the maximum relative errors.

The relative errors are nearly the same for the two tested power spectra, implying that the form of the data do not strongly affect the accuracy of the routines. Relative errors using the *Driscoll and Healy* [1994] sampled grids are nearly identical to those using Gauss-Legendre quadrature. The errors associated with the routines for complex data are lower by a few orders of magnitude.

Timing tests

The speed of the spherical harmonic reconstructions and transforms were tested for both real and complex data using the Gauss-Legendre and *Driscoll and Healy* [1994] quadrature implementations. The amount of time in seconds required to perform these operations is plotted in the figure below as a function of the spherical harmonic bandwidth of the function. These calculations were performed on a modern Mac Pro 2.7 GHz 12-Core Intel Xeon E5 using 64 bit executables and level 3 optimizations.

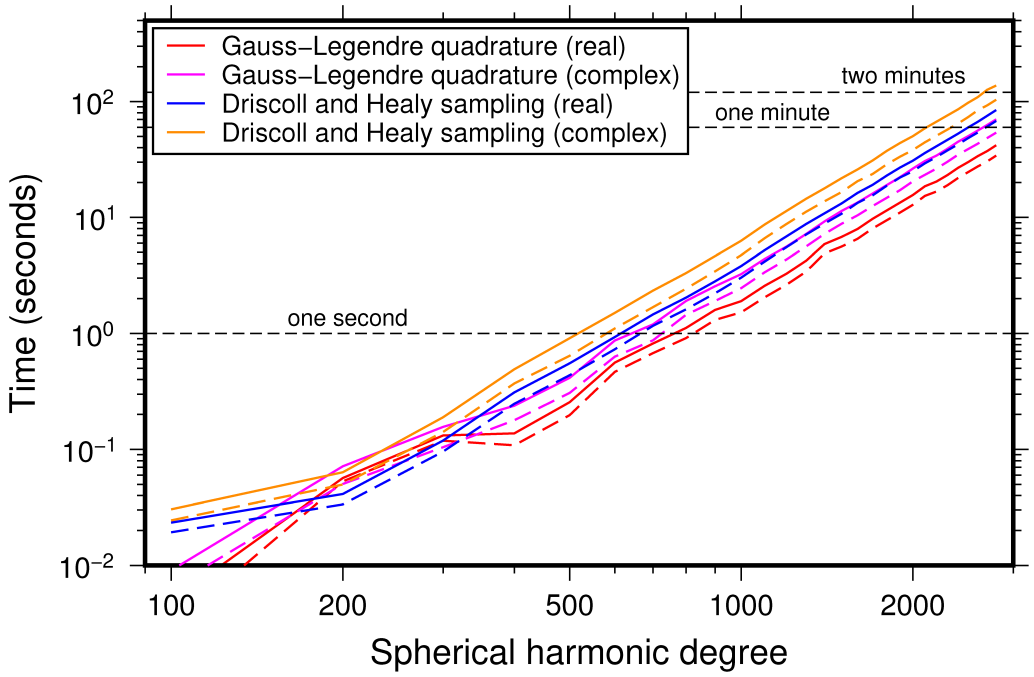


Figure 3. Time to perform the reconstruction of a function from its spherical harmonic coefficients (solid lines) and the spherical harmonic transform of the function (dashed lines).

For the real Gauss-Legendre quadrature routines, the transform time is on the order of one second for degrees close to 800 and about 30 seconds for degree 2600. For the real *Driscoll and Healy* [1994] sampled grids, the transform time is close to a second for degree 600 and about 1 minute for degrees close to 2600. The complex routines are slower by a factor of about 1.4.

References

- Driscoll, J. R. and D. M. Healy, Computing Fourier transforms and convolutions on the 2-sphere, Adv. Appl. Math., 15, 202-250, doi:10.1006/aama.1994.1008 , 1994.
- Frigo, M., and S. G. Johnson, The design and implementation of FFTW3, Proc. IEEE, 93, 216–231, doi:10.1109/JPROC.2004.840301 , 2005.
- Holmes, S. A., and W. E. Featherstone, A unified approach to the Clenshaw summation and the recursive computation of very high degree and order normalised associated Legendre functions, J. Geodesy, 76, 279- 299, doi:10.1007/s00190-002-0216-2 , 2002.
- Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, “Numerical Recipes in FORTRAN: The Art of Scientific Computing,” 2nd ed., Cambridge Univ. Press, Cambridge, UK, 1992.

Modules

Module name	Description
SHTOOLS	Module containing the SHTOOLS function and subroutine declarations.
PlanetsConstants (page 0)	Module containing planetary constants.

Legendre functions

normalized

Routine name	Description
PlmBar (page 0)	Compute all the 4π-normalized associated Legendre functions.
PlmBar_d1 (page 0)	Compute all the 4π-normalized associated Legendre functions and first derivatives.
PlBar (page 0)	Compute all the 4π-normalized Legendre polynomials.
PlBar_d1 (page 0)	Compute all the 4π-normalized Legendre Polynomials and first derivatives.

Orthonormalized

Routine name	Description
PlmON (page 0)	Compute all the orthonormalized associated Legendre functions.
PlmON_d1 (page 0)	Compute all the orthonormalized associated Legendre functions and first derivatives.
PlON (page 0)	Compute all the orthonormalized Legendre polynomials.
PlON_d1 (page 0)	Compute all the orthonormalized Legendre polynomials and first derivatives.

Schmidt semi-normalized

Routine name	Description
PlmSchmidt (page 0)	Compute all the Schmidt-normalized associated Legendre functions.

Routine name	Description
PlmSchmidt_d1 (page 0)	Compute all the Schmidt-normalized associated Legendre functions and first derivatives.
PlSchmidt (page 0)	Compute all the Schmidt-normalized Legendre polynomials.
PlSchmidt_d1 (page 0)	Compute all the Schmidt-normalized Legendre polynomials and first derivatives.

Unnormalized

Routine name	Description
PLegendreA (page 0)	Compute all the unnormalized associated Legendre functions.
PLegendreA_d1 (page 0)	Compute all the unnormalized associated Legendre functions and first derivatives.
PLegendre (page 0)	Compute all the unnormalized Legendre polynomials.
PLegendre_d1 (page 0)	Compute all the unnormalized Legendre polynomials and first derivatives.

Utilities

Routine name	Description
PlmIndex (page 0)	Compute the index of an array of Legendre functions corresponding to degree l and angular order m.

Spherical harmonic transforms

Equally sampled (N×N) and equally spaced (N×2N) grids

Routine name	Description
SHExpandDH (page 0)	Expand an equally sampled or equally spaced map into spherical harmonics using *Driscoll and Healy*'s (1994) sampling theorem.
MakeGridDH (page 0)	Create a 2D map from a set of spherical harmonic coefficients that conforms with *Driscoll and Healy*'s (1994) sampling theorem.
SHExpandDHC (page 0)	Expand an equally sampled or equally spaced complex map into complex spherical harmonics using *Driscoll and Healy*'s (1994) sampling theorem.
MakeGridDHC (page 0)	Create a 2D complex map from a set of complex spherical harmonic coefficients that conforms with *Driscoll and Healy*'s (1994) sampling theorem.

Gauss-Legendre quadrature grids

Routine name	Description
SHGLQ (page 0)	Precompute weights, nodes, and associated Legendre functions used in the GLQ-based spherical harmonics routines.
SHExpandGLQ (page 0)	Expand a 2D map sampled on the Gauss-Legendre quadrature nodes into spherical harmonics.
MakeGridGLQ (page 0)	Create a 2D map from a set of spherical harmonic coefficients sampled on a the Gauss-Legendre quadrature nodes.

Routine name	Description
SHExpandGLQC (page 0)	Expand a 2D complex map sampled on the Gauss-Legendre quadrature nodes into complex spherical harmonics.
MakeGridGLQC (page 0)	Create a 2D complex map from a set of complex spherical harmonic coefficients sampled on a the Gauss-Legendre quadrature nodes.
GLQGridCoord (page 0)	Compute the latitude and longitude coordinates used in Gauss-Legendre quadrature grids.

Other routines

Routine name	Description
SHExpandLSQ (page 0)	Expand a set of irregularly sampled data points into spherical harmonics using a least squares inversion.
MakeGrid2D (page 0)	Create a 2D cylindrical map with arbitrary grid spacing from a set of spherical harmonic coefficients.
MakeGridPoint (page 0)	Evaluate a real function expressed in real spherical harmonics at a single point.
MakeGridPointC (page 0)	Evaluate a complex function expressed in complex spherical harmonics at a single point.
SHMultiply (page 0)	Multiply two functions and determine the spherical harmonic coefficients of the resulting function.

Spherical harmonic I/O, storage, and conversions

Spherical harmonic I/O

Routine name	Description
SHRead (page 0)	Read spherical harmonic coefficients from an ascii-formatted file.
SHRead2 (page 0)	Read spherical harmonic coefficients from a CHAMP or GRACE-like ASCII file.
SHReadJPL (page 0)	Read spherical harmonic coefficients from a JPL ASCII file.

Spherical harmonic storage

Routine name	Description
SHCilmToCindex (page 0)	Convert a three-dimensional array of complex spherical harmonic coefficients to a two-dimensional indexed array.
SHCindexToCilm (page 0)	Convert a two-dimensional indexed array of complex spherical harmonic coefficients to a three-dimensional array.
SHCilmToVector (page 0)	Convert a 3-dimensional array of real spherical harmonic coefficients to a 1-dimensional ordered array.
SHVectorToCilm (page 0)	Convert a 1-dimensional indexed vector of real spherical harmonic coefficients to a 3-dimensional array.
YilmIndexVector (page 0)	Determine the index of a 1D ordered vector of spherical harmonic coefficients corresponding to i, l, and m.

Spherical harmonic conversions

Routine name	Description
SHrtoc (page 0)	Convert real spherical harmonics to complex form.
SHctor (page 0)	Convert complex spherical harmonics to real form.

Global spectral analysis

Real spectral analysis

Routine name	Description
SHPowerL (page 0)	Compute the power of a real function for a single spherical harmonic degree.
SHPowerDensityL (page 0)	Compute the power spectral density of a function for a single spherical harmonic degree.
SHCrossPowerL (page 0)	Compute the cross-power of two functions for a single spherical harmonic degree.
SHCrossPowerDensityL (page 0)	Compute the cross-power spectral density of two functions for a single spherical harmonic degree.
SHPowerSpectrum (page 0)	Compute the power spectrum of a function.
SHPowerSpectrumDensity (page 0)	Compute the power spectral density of a function.
SHCrossPowerSpectrum (page 0)	Compute the cross-power spectrum of two functions.
SHCrossPowerSpectrumDensity (page 0)	Compute the cross-power spectral density of two functions.
SHAdmitCorr (page 0)	Calculate the admittance and correlation spectra of two functions.
SHConfidence (page 0)	Compute the probability that two sets of spherical harmonic coefficients are correlated at a given degree and for a given correlation coefficient.

Complex spectral analysis

Routine name	Description
SHPowerLC (page 0)	Compute the power of a complex function for a single spherical harmonic degree.
SHPowerDensityLC (page 0)	Compute the power spectral density of a complex function for a single spherical harmonic degree.
SHCrossPowerLC (page 0)	Compute the cross-power of two complex functions for a single spherical harmonic degree.
SHCrossPowerDensityLC (page 0)	Compute the cross-power spectral density of two complex functions for a single spherical harmonic degree.
SHPowerSpectrumC (page 0)	Compute the power spectrum of a complex function.
SHPowerSpectrumDensityC (page 0)	Compute the power spectral density of a complex function.
SHCrossPowerSpectrumC (page 0)	Compute the cross-power spectrum of two complex functions.
SHCrossPowerSpectrumDensityC (page 0)	Compute the cross-power spectral density of two complex functions.

Localized spectral analysis

Multitaper spectral estimation (spherical cap domain)

Routine name	Description
SHMultiTaperSE (page 0)	Perform a localized multitaper spectral analysis.
SHMultiTaperCSE (page 0)	Perform a localized multitaper cross-spectral analysis.
SHLocalizedAdmitCorr (page 0)	Calculate the localized admittance and correlation spectra of two functions at a given location.
SHReturnTapers (page 0)	Calculate the eigenfunctions of the spherical-cap concentration problem.
SHReturnTapersM (page 0)	Calculate the eigenfunctions of the spherical-cap concentration problem for a single angular order.
ComputeDm (page 0)	Compute the space-concentration kernel of a spherical cap.
ComputeDG82 (page 0)	Compute the tridiagonal matrix of *Grünbaum et al.* (1982) that commutes with the space-concentration kernel of a spherical cap.
SHFindLWin (page 0)	Determine the spherical-harmonic bandwidth that is necessary to achieve a certain concentration factor.
SHBiasK (page 0)	Calculate the multitaper (cross-)power spectrum expectation of a windowed function.
SHMTCouplingMatrix (page 0)	Calculate the multitaper coupling matrix for a given set of localization windows.
SHBiasAdmitCorr (page 0)	Calculate the expected multitaper admittance and correlation spectra associated with the input global cross-power spectra of two functions.
SHMTDebias (page 0)	Invert for the global power spectrum given a localized multitaper spectrum estimate.

Routine name	Description
SHMTVarOpt (page 0)	Calculate the minimum variance and corresponding optimal weights of a localized multitaper spectral estimate.
SHSjkPG (page 0)	Calculate the expectation of the product of two functions, each multiplied by a different data taper, for a given spherical harmonic degree and two different angular orders.

Localization windows (arbitrary domain)

Routine name	Description
SHReturnTapersMap (page 0)	Calculate the eigenfunctions of the concentration problem for an arbitrary concentration region.
SHMultiTaperMaskSE (page 0)	Perform a localized multitaper spectral analysis using arbitrary windows.
SHMultiTaperMaskCSE (page 0)	Perform a localized multitaper cross-spectral analysis using arbitrary windows.
SHBiasKMask (page 0)	Calculate the multitaper (cross-)power spectrum expectation of a function localized by arbitrary windows derived from a mask.
ComputeDMap (page 0)	Compute the space-concentration kernel of a mask defined on the sphere.
Curve2Mask (page 0)	Given a set of latitude and longitude coordinates representing a closed curve, output a gridded mask.

Localization bias (general)

Routine name	Description
SHBias (page 0)	Calculate the (cross-)power spectrum expectation of a windowed function.

Other routines

Routine name	Description
SphericalCapCoef (page 0)	Calculate the spherical harmonic coefficients of a spherical cap.

References

- Grünbaum, F.A., L. Longhi, and M. Perlstadt, Differential operators commuting with finite convolution integral operators: some non-abelian examples, SIAM J. Appl. Math., 42, 941-955, doi:10.1137/0142067 , 1982.

Spherical harmonic rotations

Module name	Description
djpi2 (page 0)	Compute the rotation matrix d(π/2) used in rotating data expressed in spherical harmonics.
SHRotateCoef (page 0)	Determine the spherical harmonic coefficients of a complex function rotated by three Euler angles.
SHRotateRealCoef (page 0)	Determine the spherical harmonic coefficients of a real function rotated by three Euler angles.

Gravity and Magnetics routines

Gravity routines

Routine name	Description
MakeGravGridDH (page 0)	Create 2D cylindrical maps on a flattened and rotating ellipsoid of all three components of the gravity field, the gravity disturbance, and the gravitational potential.
MakeGravGradGridDH (page 0)	Calculate the components of the gravity "gradient" tensor on a flattened ellipsoid.
MakeGeoidGrid (page 0)	Create a global map of the geoid.
CilmPlus (page 0)	Calculate the gravitational potential exterior to relief along a spherical interface using the finite-amplitude algorithm of *Wieczorek and Phillips* (1998).
CilmMinus (page 0)	Calculate the gravitational potential interior to relief along a spherical interface using the finite-amplitude algorithm of *Wieczorek and Phillips* (1998).
CilmPlusRhoH (page 0)	Calculate the gravitational potential exterior to relief along a spherical interface with laterally varying density using the finite amplitude algorithm of *Wieczorek* (2007).
CilmMinusRhoH (page 0)	Calculate the gravitational potential interior to relief along a spherical interface with laterally varying density using the finite amplitude algorithm of *Wieczorek* (2007).
BAtoHilm (page 0)	Calculate iteratively the relief along an interface with constant density contrast that corresponds to a given Bouguer anomaly using the algorithm of *Wieczorek and Phillips* (1998).

Routine name	Description
BAtoHilmRhoH (page 0)	Iteratively calculate the relief along an interface with laterally varying density contrast that corresponds to a given Bouguer anomaly using the algorithm of *Wieczorek and Phillips* (1998).
DownContFilterMA (page 0)	Compute the minimum-amplitude downward continuation filter of *Wieczorek and Phillips* (1998).
DownContFilterMC (page 0)	Calculate a minimum-curvature downward continuation filter for a given spherical harmonic degree.
NormalGravity (page 0)	Calculate the normal gravity on a flattened ellipsoid using the formula of Somigliana.

Magnetics routines

Routine name	Description
MakeMagGridDH (page 0)	Create 2D cylindrical maps on a flattened ellipsoid of all three vector components of the magnetic field, the magnitude of the magnetic field, and the magnetic potential.
SHMagPowerSpectrum (page 0)	Compute the power spectrum of the magnetic field given the Schmidt semi-normalized magnetic potential spherical harmonic coefficients.
SHMagPowerL (page 0)	Compute the power of the magnetic field for a single degree L given the Schmidt semi-normalized magnetic potential spherical harmonic coefficients.

References

- Wieczorek, M. A. and R. J. Phillips, Potential anomalies on a sphere: applications to the thickness of the lunar crust, J. Geophys. Res., 103, 1715-1724, doi:10.1029/97JE03136 , 1998.
- Wieczorek, M. A. Gravity and topography of the terrestrial planets, Treatise on Geophysics, 10, 165-206, doi:10.1016/B978-044452748-6/00156-5 , 2007.

Miscellaneous routines

Routine name	Description
MakeCircleCoord (page 0)	Compute coordinates of a circle placed at a given latitude and longitude.
MakeEllipseCoord (page 0)	Compute coordinates of an ellipse placed at a given latitude and longitude.
RandomN (page 0)	Return a pseudo uniform random deviate between 0 and 1 using the algorithm of Park and Miller with a Marsaglia shift sequence.
RandomGaussian (page 0)	Return a pseudo Gaussian deviate of zero mean and unit variance.
PreGLQ (page 0)	Calculate the weights and nodes used in integrating a function by Gauss-Legendre quadrature.
EigValVecSym (page 0)	Compute the eigenvalues and eigenvectors of a real symmetric matrix.
EigValVecSymTri (page 0)	Compute the eigenvalues and eigenvectors of a real symmetric tridiagonal matrix.
EigValSym (page 0)	Compute the eigenvalues of a real symmetric matrix.
Wigner3j (page 0)	Compute the Wigner-3j symbols for all allowable values of j.
DHaj (page 0)	Compute the latitudinal weights used in the *Driscoll and Healy* (1994) spherical harmonic transform.

Python classes

Class name	Description
SHCoeffs (page 0)	Class for working with spherical harmonic coefficients.
SHGrid (page 0)	Class for working with global gridded data.
SHWindow (page 0)	Class for working with localization windows.

Legendre functions

4π normalized

Function name	Description
PlmBar (page 0)	Compute all the geodesy-normalized associated Legendre functions.
PlmBar_d1 (page 0)	Compute all the geodesy-normalized associated Legendre functions and first derivatives.
PlBar (page 0)	Compute all the geodesy-normalized Legendre polynomials.
PlBar_d1 (page 0)	Compute all the geodesy-normalized Legendre Polynomials and first derivatives.

Orthonormalized

Function name	Description
PlmON (page 0)	Compute all the orthonormalized associated Legendre functions.
PlmON_d1 (page 0)	Compute all the orthonormalized associated Legendre functions and first derivatives.
PlON (page 0)	Compute all the orthonormalized Legendre polynomials.
PlON_d1 (page 0)	Compute all the orthonormalized Legendre polynomials and first derivatives.

Schmidt normalized

Function name	Description
PlmSchmidt (page 0)	Compute all the Schmidt-normalized associated Legendre functions.

Function name	Description
PlmSchmidt_d1 (page 0)	Compute all the Schmidt-normalized associated Legendre functions and first derivatives.
PlSchmidt (page 0)	Compute all the Schmidt-normalized Legendre polynomials.
PlSchmidt_d1 (page 0)	Compute all the Schmidt-normalized Legendre polynomials and first derivatives.

Unnormalized

Function name	Description
PLegendreA (page 0)	Compute all the unnormalized associated Legendre functions.
PLegendreA_d1 (page 0)	Compute all the unnormalized associated Legendre functions and first derivatives.
PLegendre (page 0)	Compute all the unnormalized Legendre polynomials.
PLegendre_d1 (page 0)	Compute all the unnormalized Legendre polynomials and first derivatives.

Utilities

Function name	Description
PlmIndex (page 0)	Compute the index of an array of Legendre function corresponding to degree *l* and angular order *m*.

Spherical harmonic transforms

Equally sampled (N×N) and equally spaced (N×2N) grids

Function name	Description
SHExpandDH (page 0)	Expand an equally sampled or equally spaced map into spherical harmonics using *Driscoll and Healy*'s (1994) sampling theorem.
MakeGridDH (page 0)	Create a 2D map from a set of spherical harmonic coefficients that conforms with *Driscoll and Healy*'s (1994) sampling theorem.
SHExpandDHC (page 0)	Expand an equally sampled or equally spaced complex map into complex spherical harmonics using *Driscoll and Healy*'s (1994) sampling theorem.
MakeGridDHC (page 0)	Create a 2D complex map from a set of complex spherical harmonic coefficients that conforms with *Driscoll and Healy*'s (1994) sampling theorem.

Gauss-Legendre quadrature grids

Function name	Description
SHGLQ (page 0)	Precompute the weights and nodes used in the GLQ-based spherical harmonics routines.
SHExpandGLQ (page 0)	Expand a 2D map sampled on the Gauss-Legendre quadrature nodes into spherical harmonics.
MakeGridGLQ (page 0)	Create a 2D map from a set of spherical harmonic coefficients sampled on a the Gauss-Legendre quadrature nodes.
SHExpandGLQC (page 0)	Expand a 2D complex map sampled on the Gauss-Legendre quadrature nodes into complex spherical harmonics.

Function name	Description
MakeGridGLQC (page 0)	Create a 2D complex map from a set of complex spherical harmonic coefficients sampled on a the Gauss-Legendre quadrature nodes.
GLQGridCoord (page 0)	Compute the latitude and longitude coordinates used in Gauss-Legendre quadrature grids.

Other routines

Function name	Description
SHExpandLSQ (page 0)	Expand a set of irregularly sampled data points into spherical harmonics using a least squares inversion.
MakeGrid2D (page 0)	Create a 2D cylindrical map with arbitrary grid spacing from a set of spherical harmonic coefficients.
MakeGridPoint (page 0)	Evaluate a real function expressed in real spherical harmonics at a single point.
MakeGridPointC (page 0)	Evaluate a complex function expressed in complex spherical harmonics at a single point.
SHMultiply (page 0)	Multiply two functions and determine the spherical harmonic coefficients of the resulting function.

Spherical harmonic I/O, storage, and conversions

Spherical harmonic I/O

Function name	Description
SHRead (page 0)	Read spherical harmonic coefficients from an ascii-formatted file.
SHReadH (page 0)	Read spherical harmonic coefficients from an ascii-formatted file with a header line.
SHReadError (page 0)	Read spherical harmonic coefficients and associated errors from an ascii-formatted file.
SHReadErrorH (page 0)	Read spherical harmonic coefficients and associated errors from an ascii-formatted file with a header line.
SHRead2 (page 0)	Read spherical harmonic coefficients from a CHAMP or GRACE-like ascii-formatted file.
SHRead2Error (page 0)	Read spherical harmonic coefficients and associated errors from a CHAMP or GRACE-like ascii-formatted file.
SHReadJPL (page 0)	Read spherical harmonic coefficients from a JPL ascii-formatted file.
SHReadJPLError (page 0)	Read spherical harmonic coefficients and associated errors from a JPL ascii-formatted file.
read_icgem_gfc (page 0)	Read spherical harmonic coefficients from an ICGEM GFC ascii-formatted file.

Spherical harmonic storage

Function name	Description
SHCilmToCindex (page 0)	Convert a three-dimensional array of complex spherical harmonic coefficients to a two-dimensional indexed array.

Function name	Description
SHCindexToCilm (page 0)	Convert a two-dimensional indexed array of complex spherical harmonic coefficients to a three-dimensional array.
SHCilmToVector (page 0)	Convert a 3-dimensional array of real spherical harmonic coefficients to a 1-dimensional ordered array.
SHVectorToCilm (page 0)	Convert a 1-dimensional indexed vector of real spherical harmonic coefficients to a 3-dimensional array.
YilmIndexVector (page 0)	Determine the index of a 1-dimensional ordered vector of spherical harmonic coefficients corresponding to *i*, *l*, and *m*.

Spherical harmonic conversions

Function name	Description
SHrtoc (page 0)	Convert real spherical harmonics to complex form.
SHctor (page 0)	Convert complex spherical harmonics to real form.

Global and localized spectral analysis

Global spectral analysis

Function name	Description
spectrum (page 0)	Calculate the spectrum of a real or complex function.
cross_spectrum (page 0)	Calculate the cross-spectrum of a real or complex function.
SHAdmitCorr (page 0)	Calculate the admittance and correlation spectra of two functions.
SHConfidence (page 0)	Compute the probability that two sets of spherical harmonic coefficients are correlated at a given degree and for a given correlation coefficient.

Multitaper spectral estimation (spherical cap domain)

Function name	Description
SHMultiTaperSE (page 0)	Perform a localized multitaper spectral analysis.
SHMultiTaperCSE (page 0)	Perform a localized multitaper cross-spectral analysis.
SHLocalizedAdmitCorr (page 0)	Calculate the localized admittance and correlation spectra of two functions at a given location.
SHReturnTapers (page 0)	Calculate the eigenfunctions of the spherical-cap concentration problem.
SHReturnTapersM (page 0)	Calculate the eigenfunctions of the spherical-cap concentration problem for a single angular order.
ComputeDm (page 0)	Compute the space-concentration kernel of a spherical cap.

Function name	Description
ComputeDG82 (page 0)	Compute the tridiagonal matrix of *Grünbaum et al.* (1982) that commutes with the space-concentration kernel of a spherical cap.
SHFindLWin (page 0)	Determine the spherical-harmonic bandwidth that is necessary to achieve a certain concentration factor.
SHBiasK (page 0)	Calculate the multitaper (cross-)power spectrum expectation of a windowed function.
SHMTCouplingMatrix (page 0)	Calculate the multitaper coupling matrix for a given set of localization windows.
SHBiasAdmitCorr (page 0)	Calculate the expected multitaper admittance and correlation spectra associated with the input global cross-power spectra of two functions.
SHMTDebias (page 0)	Invert for the global power spectrum given a localized multitaper spectrum estimate.
SHMTVarOpt (page 0)	Calculate the minimum variance and corresponding optimal weights of a localized multitaper spectral estimate.
SHSjkPG (page 0)	Calculate the expectation of the product of two functions, each multiplied by a different data taper, for a given spherical harmonic degree and two different angular orders.

Localization windows (arbitrary domain)

Function name	Description
SHReturnTapersMap (page 0)	Calculate the eigenfunctions of the concentration problem for an arbitrary concentration region.
SHBiasKMask (page 0)	Calculate the multitaper (cross-)power spectrum expectation of a function localized by arbitrary windows derived from a mask.

Function name	Description
SHMultiTaperMaskSE (page 0)	Perform a localized multitaper spectral analysis using arbitrary windows.
SHMultiTaperMaskCSE (page 0)	Perform a localized multitaper cross-spectral analysis using arbitrary windows.
ComputeDMap (page 0)	Compute the space-concentration kernel of a mask defined on the sphere.
Curve2Mask (page 0)	Given a set of latitude and longitude coordinates representing a closed curve, output a gridded mask.

Localization bias (general)

Function name	Description
SHBias (page 0)	Calculate the (cross-)power spectrum expectation of a windowed function.

Other routines

Function name	Description
SphericalCapCoef (page 0)	Calculate the spherical harmonic coefficients of a spherical cap.

References

- Grünbaum, F. A., L. Longhi, and M. Perlstadt, Differential operators commuting with finite convolution integral operators: some non-abelian examples, SIAM J. Appl. Math., 42, 941-955, doi:10.1137/0142067 , 1982.

Spherical harmonic rotations

Function name	Description
djpi2 (page 0)	Compute the rotation matrix d(π/2) used in rotating data expressed in spherical harmonics.
SHRotateCoef (page 0)	Determine the spherical harmonic coefficients of a complex function rotated by three Euler angles.
SHRotateRealCoef (page 0)	Determine the spherical harmonic coefficients of a real function rotated by three Euler angles.

Gravity and magnetics routines

Gravity routines

Function name	Description
MakeGravGridDH (page 0)	Create 2D cylindrical maps on a flattened and rotating ellipsoid of all three components of the gravity field, the gravity disturbance, and the gravitational potential.
MakeGravGradGridDH (page 0)	Calculate the components of the gravity "gradient" tensor on a flattened ellipsoid.
MakeGeoidGridDH (page 0)	Create a global map of the geoid.
CilmPlusDH (page 0)	Calculate the gravitational potential exterior to relief along a spherical interface using the finite-amplitude algorithm of *Wieczorek and Phillips* (1998) on a *Driscoll and Healy* (1994) grid.
CilmMinusDH (page 0)	Calculate the gravitational potential interior to relief along to a spherical interface using the finite-amplitude algorithm of *Wieczorek and Phillips* (1998) on a *Driscoll and Healy* (1994) grid.
CilmPlusRhoHDH (page 0)	Calculate the gravitational potential exterior to relief along a spherical interface with laterally varying density using the finite amplitude algorithm of *Wieczorek* (2007) on a *Driscoll and Healy* (1994) grid.
CilmMinusRhoHDH (page 0)	Calculate the gravitational potential interior to relief along a spherical interface with laterally varying density using the finite amplitude algorithm of *Wieczorek* (2007) on a *Driscoll and Healy* (1994) grid.
BAtoHilmDH (page 0)	Calculate iteratively the relief along an interface with constant density contrast that corresponds to a given Bouguer anomaly using the algorithm of *Wieczorek and Phillips* (1998).

Function name	Description
BAtoHilmRhoHDH (page 0)	Iteratively calculate the relief along an interface with laterally varying density contrast that corresponds to a given Bouguer anomaly using the algorithm of *Wieczorek and Phillips* (1998).
DownContFilterMA (page 0)	Compute the minimum-amplitude downward continuation filter of *Wieczorek and Phillips* (1998).
DownContFilterMC (page 0)	Calculate a minimum-curvature downward continuation filter for a given spherical harmonic degree.
NormalGravity (page 0)	Calculate the normal gravity on a flattened ellipsoid using the formula of Somigliana.

Magnetics routines

Function name	Description
MakeMagGridDH (page 0)	Create 2D cylindrical maps on a flattened ellipsoid of all three vector components of the magnetic field, the magnitude of the magnetic field, and the magnetic potential.
SHMagPowerSpectrum (page 0)	Compute the power spectrum of the magnetic field given the Schmidt semi-normalized magnetic potential spherical harmonic coefficients.
SHMagPowerL (page 0)	Compute the power of the magnetic field for a single degree L given the Schmidt semi-normalized magnetic potential spherical harmonic coefficients.

References

- Wieczorek, M. A. and R. J. Phillips, Potential anomalies on a sphere: applications to the thickness of the lunar crust, J. Geophys. Res., 103, 1715-1724, doi:10.1029/97JE03136 , 1998.
- Wieczorek, M. A. Gravity and topography of the terrestrial planets, Treatise on Geophysics, 10, 165-206, doi:10.1016/B978-044452748-6/00156-5 , 2007.

SHTOOLS utilities

Function name	Description
MakeCircleCoord (page 0)	Compute coordinates of a circle placed at a given latitude and longitude.
MakeEllipseCoord (page 0)	Compute coordinates of an ellipse placed at a given latitude and longitude.
Wigner3j (page 0)	Compute the Wigner-3j symbols for all allowable values of *j*.
DHaj (page 0)	Compute the latitudinal weights used in the Driscoll and Healy (1994) spherical harmonic transform.

Constants

The `constant` subpackage defines several constants that are useful when working with gravity and topography data of the terrestrial planets. All units are SI. Confer with the constant's `info()` method for exact values and references.

Fundamental constants

Constant	Description
`grav_constant`	Gravitational Constant
`pi_constant`	Pi
`mu0_constant`	Magnetic constant

Mercury

Constant	Description
`r_mercury`	Average radius of Mercury
`gm_mercury`	Gravitational constant times the mass of Mercury
`mass_mercury`	Mass of Mercury
`r0_pot_mercury`	Reference radius used for the spherical harmonic gravity solutions
`omega_mercury_orbit`	Angular rotation rate of Mercury about the Sun
`omega_mercury_spin`	Angular rotation rate of Mercury
`rho_bar_mercury`	Average density of Mercury
`g0_mercury`	Gravitational acceleration at `r_mercury`, not including rotation

Venus

Constant	Description
r_venus	Average radius of Venus
r0_pot_venus	Reference radius of the gravitational-potential models of Venus
gm_venus	Gravitational constant times the mass of Venus
mass_venus	Mass of Venus
omega_venus	Angular rotation rate of Venus
rho_bar_venus	Average density of Venus
g0_venus	Gravitational acceleration at R_Venus, not including rotation

Earth

Constant	Description
gm_earth	Gravitational constant times the mass of Earth
r0_pot_earth	Reference radius of the terrestrial gravitational-potential models
mass_earth	The mass of Earth
wgs84_a	The semi-major axis of the WGS84 ellipsoid
wgs84_b	The semi-minor axis of the WGS84 ellipsoid
wgs84_r3	The radius of a sphere of Earth's volume
wgs84_f	The flattening of the WGS84 ellipsoid
wgs84_gm	The adopted GM of the WGS84 model, which includes the atmosphere

Constant	Description
wgs84_gma	The GM of the atmosphere adopted by the WGS84 model
wgs84_omega	The adopted angular rotation rate of Earth of the WGS84 model
wgs84_u0	The theoretical normal potential associated with the WGS84 model

The Moon

Constant	Description
r_moon	Mean radius of the Moon
gm_moon	Gravitational constant times the mass of the Moon
mass_moon	Mass of the Moon
rho_bar_moon	Average density of the Moon
a_moon	Semi-major axis of the lunar orbit
g0_moon	Mean gravitational acceleration of the Moon at the mean surface radius (r_moon), not including rotation
r0_pot_moon	Reference radius of the gravitational-potential models of the Moon
omega_moon	Angular rotation rate of the Moon
c_moi_moon	Polar moment of inertia of the Moon, using R=1738 km
i_moi_moon	Average moment of inertia of the Moon, using R=1738 km
gamma_moi_moon	Libration parameter of the Moon, (B-A)/C
beta_moi_moon	Libration parameter of the Moon, (C-A)/B

Mars

Constant	Description
r_mars	Average radius of Mars
gm_mars	Gravitational constant times the mass of Mars
mass_mars	Mass of Mars
rho_bar_mars	Average density of Mars
g0_mars	Gravitational acceleration of Mars at r_mars , not including rotation
r0_pot_mars	Reference radius of the gravitational-potential models of Mars
omega_mars	Angular rotation rate of Mars
f_mars	Topographic flattening of Mars
a_mars	Semi-major axis radius of Mars
b_mars	Semi-minor axis radius of Mars
w0_mars	Reference potential of Mars

Python examples

Notebooks

Notebook name	Description
Introduction 1 (page 0)	Grids and spherical harmonic coefficients.
Introduction 2 (page 0)	Localization windows and spectral analysis.
Tutorial 1 (page 0)	Simple spherical harmonic analyses.
Tutorial 2 (page 0)	Localized spectral analysis on the sphere.
Tutorial 3 (page 0)	The SHTOOLS class interface.
Tutorial 4 (page 0)	Spherical harmonic normalizations and Parseval's theorem.
Tutorial 5 (page 0)	Multitaper spectral analysis class interface.
Tutorial 6 (page 0)	3D plots of gridded data.

Test programs

A variety of test programs can be found in the folders in `examples/python`.

Folder directory	Description
`ClassInterface/`	Test the python class interfaces.
`TestLegendre/`	Test and plot the Legendre functions.
`IOStorageConversions/`	Read coefficients from a file and test conversions between real and complex coefficients.
`GlobalSpectralAnalysis/`	Test functions to compute different power spectra from real and complex coefficients.
`LocalizedSpectralAnalysis/`	Test the coupling matrix, localized spectral analysis, and bias routines.

Folder directory	Description
`GravMag/`	Test the gravity and magnetics routines, and compute the crustal thickness of Mars.
`TimingAccuracy/`	Perform timing and accuracy tests using real and complex coefficients, with *Driscoll and Healy* (1994) and Gauss-Legendre quadrature grids.
`Other/`	Test a variety of other routines.

Fortran examples

A variety of test programs can be found in the folders in `examples/fortran`.

Folder directory	Description
`SHCilmPlus/`	Demonstration of how to expand spherical harmonic files into gridded maps using the GLQ routines, and how to compute the gravity field resulting from finite amplitude surface relief.
`SHExpandDH/`	Demonstration of how to expand a grid that is equally sampled in latitude and longitude into spherical harmonics using the sampling theorem of *Driscoll and Healy* (1994).
`SHExpandLSQ/`	Demonstration of how to expand a set of irregularly sampled data points in latitude and longitude into spherical harmonics by use of a least squares inversion.
`SHMag/`	Demonstration of how to expand scalar magnetic potential spherical harmonic coefficients into their three vector components and total field.
`MarsCrustalThickness` /	Demonstration of how to compute a crustal thickness map of Mars.
`SHRotate/`	Demonstration of how to determine the spherical harmonic coefficients for a body that is rotated with respect to its initial configuration.
`SHLocalizedAdmitCorr/`	Demonstration of how to calculate localized admittance and correlation spectra for a given set of gravity and topography spherical harmonic coefficients.
`TimingAccuracy/`	Test programs that calculate the time required to perform the GLQ and DH spherical harmonic transforms and reconstructions and the accuracy of these operations.

Frequently asked questions

I have a question

Before contacting us, first

- read the documentation on this web site,
- read the rest of this FAQ,
- ensure that you are using the current version of SHTOOLS,
- uninstall SHTOOLS using `make clean` and `pip uninstall pyshtools`, and then recompile/install SHTOOLS,
- consult the SHTOOLS issues at GitHub

If at this point your problem is not resolved, you can

- open an issue on GitHub ,
- ask your question on the SHTOOLS gitter chat forum , or
- contact us by email.

I have a suggestion

We often make improvements based on user suggestions. Nevertheless, please realize that the developers are very busy and that we are not paid to develop or maintain this archive. We suggest that you open an issue on GitHub describing your suggestion, and we will then flag the issue as a future enhancement.

I would like to contribute

Please see this page (page 79).

Can I use the SHTOOLS library with C, F77, and Matlab?

Probably, but we have not yet implemented this. If you get this to work, let us know how you did it and we will add the instructions and source files to the distribution.

How do I cite SHTOOLS in a publication?

Each SHTOOLS release has a DOI (digital object identifier) at Zenodo . The suggested citation will always be provided in the release notes of each release.

Where can I find more information about spherical harmonics?

Two online resources are:

- Mathworld - Spherical Harmonic
- Wikipedia - Spherical harmonics

Will you help me with my homework?

No.

I don't understand Fortran and Python. Will you explain how to modify the example codes?

No.

How do I make images with the output from SHTOOLS?

If you are using the Fortran version of SHTOOLS, the output is typically in the form of an ASCII raster file. These can be read by any standard graphics package, such as the free unix-based command line software GMT .

If you are using the Python version of SHTOOLS, the output can be visualized by use of the `matplotlib` package.

Who maintains SHTOOLS?

This software package was created initially in 2004 by Mark Wieczorek who is the lead developer. Matthias Meschede is responsible for the initial Python implementation. A list of all contributors can be found here (page 90).

Fortran 95 problems

SHTOOLS won't compile on my computer

If the command `make fortran` (or `make fortran-mp`) results in compilation errors:

- verify that you have the latest release of SHTOOLS,
- perform a `make clean` and try again,
- go to the SHTOOLS issues at GitHub to see if this problem has already been documented,
- open an issue on the GitHub SHTOOLS project page and include the **entire** output generated by the Makefile, along with information on the operating system and compilers that you are using.

If you are having problems linking an already compiled SHTOOLS library to your code, see the "linking problems" below. If you are instead receiving compilation errors in *your* code, only contact us after you have thoroughly debugged your program. If you are receiving non fatal warnings, please let us know so that we can clean up the relevant code.

Can I compile SHTOOLS with Fortran 77?

No, you must use a Fortran 90/95 compiler. Two free Fortran 90/95 compilers are gfortran and g95. If you will be using the Python code, it will be easiest to use the `gfortran` compiler.

Linking fails because of "Undefined symbols"

Linking your program to the SHTOOLS and LAPACK libraries might result in an error similar to the following:

```
ld: Undefined symbols:
_dgels
../../lib/libSHTOOLS.a(SHExpandLSQ.o) reference to undefined _d
gels
link failed.
```

If the linker is correctly finding the LAPACK and FFTW libraries, the most likely cause of this is that the subroutine names in the SHTOOLS, LAPACK, and FFTW libraries are not exactly the same. If SHTOOLS is able to link to the FFTW library correctly, but not LAPACK, try rebuilding SHTOOLS using

```
make clean
make fortran LAPACK_UNDERSCORE=1
```

If a similar problem is arising with the FFTW libraries, use

```
make clean
make fortran FFTW_UNDERSCORE=1
```

These commands will compile SHTOOLS source files that append explicitly underscores to either the LAPACK of FFTW subroutine names. If you get similar link errors, but with an added underscore, this probably means that the linker can't find the LAPACK library.

Linking fails because e_wsfe, z_abs, c_sqrt, s_cmp, etc., are undefined

Linking your program to the SHTOOLS, LAPACK, BLAS, and FFTW libraries might result in link errors similar to:

```
/usr/lib///liblapack.so: undefined reference to `e_wsfe'
/usr/lib///liblapack.so: undefined reference to `z_abs'
/usr/lib///liblapack.so: undefined reference to `c_sqrt'
/usr/lib///liblapack.so: undefined reference to `s_cmp'
/usr/lib///liblapack.so: undefined reference to `z_exp'
/usr/lib///liblapack.so: undefined reference to `c_exp'
/usr/lib///liblapack.so: undefined reference to `do_fio'
/usr/lib///liblapack.so: undefined reference to `z_sqrt'
/usr/lib///liblapack.so: undefined reference to `s_cat'
/usr/lib///liblapack.so: undefined reference to `s_stop'
/usr/lib///liblapack.so: undefined reference to `c_abs'
/usr/lib///liblapack.so: undefined reference to `s_wsfe'
/usr/lib///liblapack.so: undefined reference to `s_copy'
```

This can arise when the offending libraries were built using the g77 compiler. In order to rectify this, it should only be necessary to link to the additional library `libg2c.a` (i.e., g77 to c). Assuming that this library can be found by the linker, just append `-lg2c` to the list of libraries passed to the linker. If the `g2c` library can not be found by the linker, an easy way to find its location is by using either the "locate" or "find" shell commands

```
locate libg2c.a
find /usr -name libg2c.a
```

where the find command searches the directory `/usr`. This pathname can then be added to the linker's search path by using the option `-Ldirname`, by, for example,

```
-L/usr/lib/gcc/i586-mandrake-linux-gnu/3.4.3 -lg2c
```

Linking fails because of undefined references to SHTOOLS routines

For some compilers, the location of the source file following the compiler name is important. When this is the case, if the source file is not in its correct position, you could receive link errors that resemble the following:

```
gfortran -L../lib -lSHTOOLS TimingAccuracy/TimingAccuracyGLQC.f
95 -I../modules/ -L../lib -lfftw3 -lm -m64 -O3 -o TimingAccurac
y/TimingAccuracyGLQC
/tmp/cchgdOpg.o: In function `MAIN__':
TimingAccuracyGLQC.f95:(.text+0x5b3): undefined reference to `r
andomgaussian_'
TimingAccuracyGLQC.f95:(.text+0xb9b): undefined reference to `s
hglq_'
TimingAccuracyGLQC.f95:(.text+0xd39): undefined reference to `m
akegridglqc_'
TimingAccuracyGLQC.f95:(.text+0xedd): undefined reference to `s
hexpandglqc_'
collect2: error: ld returned 1 exit status
```

For this example, successful compilation can be achieved by placing the source file before the library calls:

```
gfortran  TimingAccuracy/TimingAccuracyGLQC.f95 -L../lib -lSHTO
OLS -I../modules/ -L../lib -lfftw3 -lm -m64 -O3 -o TimingAccura
cy/TimingAccuracyGLQC
```

The linker can't seem to find either the LAPACK, BLAS, or FFTW libraries

The linker `ld` generally searches for libraries in the directories `/lib`, `/usr/lib` and `/usr/local/lib`. When passing `-lname` to the linker, it will search for the filename `libname.a` in these directories. Additional directories can be added to the search path by passing one or more `-Ldirname` to the linker. For instance, to link to the library `/Users/Me/Lapack/liblapack.a` you would pass the following to the linker

```
-L/Users/Me/Lapack -llapack
```

If you do not know the location of a given library, it can be easily found using either the "locate" or "find" shell commands

```
locate lapack.a
find /usr -name lapack.a
```

where the find command searches the directory `/usr`.

Note that the LAPACK and BLAS library names might be somewhat different than assumed in this documentation. For instance, the LAPACK and BLAS libraries might be invoked by passing `-llapack3`, `-lf77BLAS`, or `-lcblas` to the linker. Note that if the library ATLAS is present, it can be substituted for the BLAS library. If you are certain that the library exists, and that the linker is attempting to link to it, then see How do I know if I need to append underscores to external function and subroutine names? (page 74)

How do I know if I need to append underscores to external function and subroutine names?

When making libraries such as SHTOOLS, LAPACK and FFTW, some compilers by default add trailing underscores to external function and subroutine names. If the function and subroutine names in the SHTOOLS library do not correspond exactly to what is in the LAPACK and FFTW libraries, the linker will not be able to

resolve all of the symbols, and you will receive a linker “undefined symbols” error. Most compilers have options that allows you to “modify” the name of the subroutine and function name by specifying arguments such as “fold external function names to lower case”, “do not append trailing underscores to external function names”, and so on.

If you are certain that you have installed correctly the FFTW, LAPACK and BLAS libraries, and that the linker can indeed find them, then it is quite possible that a linking problem is related to a trailing underscore being present in the external library, but not in the function and subroutine names in the SHTOOLS library (or vice versa). It is simple to inspect these libraries to determine the underscoring convections. For instance, the unix command “display name list”

```
nm -j libSHTOOLS.a | grep makegridglq
```

yields

```
_makegridglq_
_makegridglq_
_makegridglq_
_makegridglq_
_makegridglqc_
```

Indicating that underscores were added to the SHTOOLS subroutine names.

The command

```
nm -j /Applications/Absoft13.0/lib/liblapack.a | grep dgels
```

yields

```
/Applications/Absoft/lib/liblapack.a(dgels.o):
_dgels_
/Applications/Absoft/lib/liblapack.a(dgelsd.o):
_dgelsd_
/Applications/Absoft/lib/liblapack.a(dgelss.o):
_dgelss_
/Applications/Absoft/lib/liblapack.a(dgelsx.o):
_dgelsx_
/Applications/Absoft/lib/liblapack.a(dgelsy.o):
_dgelsy_
```

indicating that trailing underscores where also appended to the LAPACK subroutine names. However, the command

```
nm -j /usr/local/lib/libfftw3.a | grep dfftw_plan_dft_r2c_1d
```

might yield

```
nm: no name list
_dfftw_plan_dft_r2c_1d
```

indicating that trailing underscores were not appended to the FFTW subroutine names. It is this discrepancy between the FFTW and LAPACK libraries that could make it difficult to link to both simultaneously. To resolve this issue, see Linking fails because of “Undefined symbols” (page 73).

Linking fails because SHTOOLS routines are undefined

Linking your program to the SHTOOLS library might result in errors such as

```
TestSHRotate.f95:(.text+0xdb6): undefined reference to `makegri
d2d'
```

The most likely source of this error is that the SHTOOLS library was compiled with a different set of options than used when compiling your program. In particular, it is necessary to use the same compiler options that relate to the naming of external names, such as “fold to lower case” and the appending of underscores.

I get a “segmentation fault” when running a program that links to the SHTOOLS library

If the error is related to SHTOOLS, and not your code, then there are a few possible ways to fix this. One possibility is that this problem is related to memory allocation and stack overflows. First, try recompiling your code with the option `-s`. This will try to allocate dynamic arrays in the dataspace, as opposed to the stack. Second, try increasing the stacksize by typing

```
limit stacksize unlimited
```

at the unix prompt. If this works, consider adding this to your default shell initialization file. Finally, try increasing the datasize by typing

```
limit datasize unlimited
```

My program crashes when calling SHTOOLS routines

If your program still crashes after making the above modifications, then try the following:

- Read the version history to make certain that you are aware of any recent modifications.
- Verify that you have enough memory to perform the calculations. Be aware that some routines dynamically allocate arrays.
- Verify that the documentation describing the routine is correct by (1) comparing the man pages with the intent and argument ordering in the interface block of the file `SHTOOLS.f95`, and (2) reading the documentation in the actual source code.
- Try recompiling the SHTOOLS library *without* using optimization, i.e., by removing the fortran flag `-O3`.
- Use a debugger to find out where the error is occurring.

If this all fails, then let us know which routine is involved, as well as the error message you encounter. Any debugging that is done on your part will help greatly in finding the problem. Be aware that we are not being paid to solve your problems.

How do I specify “optional” parameters in functions and subroutines?

In Fortran 95, this can be specified in one of two ways. First, if there is only one possible optional parameter, it is only necessary to include (or exclude) it at the end of the list of arguments. In contrast, if more than one optional parameter is allowed, it is necessary to use the syntax `OPTIONALVARIABLENAME=variable` at the end of the list of calling arguments. It is strongly suggested that you always directly specify the name of the optional parameter, because it is quite possible that additional optional parameters will be added to the routine at a later date.

The LAPACK routine DSTEGR crashes with an arithmetic exception

The LAPACK routine DSTEGR is used to obtain the eigenvalues of a symmetric real tridiagonal matrix. Normal execution of this routine may generate infinities and NANs that are properly treated by ieee-754 floating point standards. If this routine crashes with the message

```
*** Arithmetic exception:  - aborting
Aborted
```

the LAPACK library was most likely not compiled using full ieee-754 standards. Try recompiling LAPACK with a compiler option such as `-ieee=full`.

How to contribute

Summary: SHTOOLS is an open source project whose success depends upon contributions from people like you. Even if you can not contribute code, you can report issues at GitHub and help us improve the documentation.

I would like to contribute to the code base

The easiest way to contribute to developing SHTOOLS is by GitHub.

- Start by forking the SHTOOLS repo into your GitHub account.
- Clone this repo on your local computer.
- Commit changes to your personal repo. We work on the `develop` branch, so please base your commits on the most recent commit on this branch.
- When you think that your changes are ready to be incorporated into the main repo, make a pull request, describing what you changed and why.
- The administrators will verify your work and may ask for changes. After they are happy with your contribution, they will merge your pull request into the main repo.

If you are not comfortable with using `git`, please just contact us by email. We will be happy to help.

I would like to report a bug, or suggest a change

- If you find a bug, please open an issue on GitHub.
- If you would like to suggest a change, or suggest a new feature that should be implemented in SHTOOLS, please open an issue at the same place. Before doing so, please look at the SHTOOLS milestones to see if this is already on the development roadmap.

I would like to fix some typos in the documentation

- The easiest way to suggest a change to the documentation is to find the file on GitHub and modify it by clicking on the edit button. Then just follow the instructions for opening a pull request. This is really very easy, and it is impossible to mess anything up.

SHTOOLS release notes: Version 4

Version 4.1

This version adds improved functionality to SHTOOLS and fixes a couple of minor bugs. In addition, this release will be the first where pre-built wheels for unix/macOS/windows will be distributed via PYPI.

Change log:

- Added an optional argument `lmax` to `SHCoeffs.from_array()`.
- Coefficients are zero-padded when `lmax` is greater than the maximum degree in `SHCoeffs.to_array()`.
- The method `pad()` was added to the `SHCoeffs` class that zero pads or truncates the coefficients to a different `lmax`.
- Fixed the method `SHCoeffs.from_file()` such that the maximum spherical harmonic degree of the class is the maximum spherical harmonic degree of the coeffs (and not `lmaxin` as before).
- Fixed formatting issues with error messages in `SHCoeffs`.
- Removed print statements from the fortran code in `BAtoHilm` and `BAtoHilmRohH` that served no purpose.
- Fixed a bug in the argument order of the python wrappers of `CilmPlusRhoDH` and `BAtoHilmRhoDH`.
- Fixed the makefile to remove the `dist` directory during clean.
- Fixed a bug in the python routine `cross_spectrum()`, where the numpy arange function was incorrectly called.
- Fixed the `SHWindow` plotting methods to work when the number of rows is equal to 1.
- Conditional tests in the routine `Wigner3j` were reordered to avoid a division by zero.
- Numpy's auto-configuration is now used to detect the LAPACK libraries.
- Many minor updates to the python documentation and unix man pages.

Citation:

M. A. Wieczorek, M. Meschede, E. Sales de Andrade, I. Oshchepkov, B. Xu, and A. Walker (2017). SHTOOLS: Version 4.1, Zenodo, doi:10.5281/zenodo.1067108

Version 4.0

This is a major update that fixes bugs, adds new functionality, and improves Python error handing. All users are requested to upgrade to 4.0.

Change log:

- Instead of executing a Fortran STOP, which kills the Python kernel, the Fortran subroutines now return an `exitstatus` that allows Python to raise an exception. This technique does not work with the few Fortran functions that pyshtools calls, but these functions are relatively benign, and will soon be phased out for Python native functions.
- The Fortran `powerspectrum` routines have been removed from pyshtools, and have been replaced with Python native routines `spectrum` and `cross_spectrum`. The Python routines allow to specify the normalization, whether the output should be power, energy or l2norm, and whether the spectrum is per degree, per coefficient, or per log bandwidth.
- The method `plot_spectrum2d()` was added to the class `SHCoeffs` to plot the power as a function of degree and order.
- All pyshtools modules have been converted into proper Python subpackages. The subpackage `localizedpsectralanalysis` has been merged into `spectralanalysis`, and the subpackage `other` has been renamed `utils`.
- The Python class method `SHCoeffs.expand()` now can evaluate the function either on an SHGrid or for a list of latitude and longitude points. As part of this change, a new fortran function `MakeGridPointC` was created for complex coefficients.
- The majority of the methods for the classes `SHCoeffs`, `SHGrid` and `SHWindow` have been rename for consistency (see documentation!). Also, the classes now give the option of reading or saving to files as numpy arrays.
- Added new Python function `read_icgen_gfc` for reading ICGEM-format gravity coefficient files.
- The operator `pow` was added to the class `SHCoeffs`.
- All methods in the pyshtools classes now return copies by default, which can be modified by the optional argument `copy`.
- Added `pot` as a mandatory return argument for the Python routine

`MakeGravGridDH`.

- Several minor modifications and bug fixes were made to the makefiles to improve compatibility and to allow the use of `make -j`.
- The routines `other.EigValSym`, `other.EigValVecSym`, `other.EigValVecSymTri`, `other.RandomGaussian`, `other.RandomN` and `other.PreGLQ` were removed from pyshtools, as these can be found in other scipy packages.
- The SHTOOLS routine `DHaj` was added to the pyshtools subpackage `utils`.
- Python docstrings have been streamlined and standardized.
- …plus, many minor changes and optimizations…

Citation:

M. A. Wieczorek, M. Meschede, I. Oshchepkov, E. Sales de Andrade, and heroxbd (2016). SHTOOLS: Version 4.0. Zenodo. doi:10.5281/zenodo.206114

SHTOOLS release notes: Version 3

Version 3.4

This release adds missing functionality to the SHGrids, SHCoeffs, and SHWindow classes, and adds support for PyPI.

Change log:

- Add pyshtools to PyPI repository. Can now be installed using `pip install pyshtools`.
- Add new function `SHBiasKMask` which is the arbitrary window counterpart to the spherical cap window `SHBiasK`.
- Add `get_biasedpowerspectrum()` method to `SHWindows` for arbitrary windows.
- Add `copy()` method to all classes, which returns a deep copy of the instance.
- Add `__sub__`, `__add__`, `__rsub__`, `__radd__`, `__mul__`, `__div__`, `__truediv__`, and `__pow__` operators for two sets of coefficient or grid classes, or one coefficient or grid class and a scalar.
- Add `nwinrot` option when rotating spherical cap windows in `SHWindow` that will rotate only the first `nwinrot` windows.
- Add degrees option to `get_lats()` and `get_lons()` methods.
- Add the constructor `from_file()` to initialize an `SHGrid` with a numpy formatted data file. Add option to read coeffs and grids from a numpy formatted binary file. Add `tofile()` methods to output raw grid and coefficient data as either text or binary formatted files.
- Update Intro 1 notebook and add example to Intro notebook 2 showing how to use arbitrary localization windows.
- Convert notebooks to html and add links to web documentation.
- Add option `fixed_power` to `SHCoeffs.from_random()` method to generate random coefficients that fit exactly the expected power spectrum.
- Add Earth topography coefficients referenced to mean sea level to the example files, expanded to degree 300: `srtmp300.msl`.

Citation:

M. A. Wieczorek, M. Meschede, I. Oshchepkov, E. Sales de Andrade (2016). SHTOOLS: Version 3.4. Zenodo. doi:10.5281/zenodo.61180

Version 3.3

This is a major upgrade to SHTOOLS. Full support for Python 3 has been added, a `setup.py` file has been added for easy installation, Python notebook tutorials have been created, and full support for three major Python classes has been provided (`SHCoeffs`, `SHGrid` and `SHWindow`). One bug in the Fortran code has been fixed, as well as several minor issues with the Python wrapper functions.

Change log:

- Added full support for Python 3.
- Fixed a critical, but rare, bug in `MakeGridDH`, `MakeGravGridDH`, `MakeMagGridDH`, and `MakeGravGradGridDH`. In these routines, the rows of the output grid are calculated by inverse Fourier transforming a vector that depends upon the spherical harmonic coefficients. This vector, when using the complex-to-real FFTW routines, includes one element that corresponds to the coefficients with `m=lmax+1`. This element of the array was not properly initialized to zero in the Fortran code, and if the Fortran compiler did not explicitly zero all new arrays, this could have resulted in incorrect output. Given that this term is 1 index above the Nyquist frequency (`m=lmax`), if this element were not zero, each column of the grid would contain a component that oscillates from -1 to +1 (scaled by the magnitude of the element). Even when this element was not initialized, a subsequent spherical harmonic expansion of the grid would usually give correct results.
- Added a `setup.py` file for easy installation.
- Makefiles were extensively modified to simplify the Fortran and Python builds: `make all2` and `make all3` were removed and replaced by flags that make use of the precompiler to resolve underscore problems; make install now places compiled module files in /usr/local/includes.
- Makefiles were improved to minimize problems when installing both fortran and fortran-mp components. When making the latter, all object and module files in the directory `src` are first deleted.
- The namespace of pyshtools was reorganized to list the routines by submodule name. A list of all routines is given in the submodule shtools.
- Add full support for the Python classes `SHCoeffs`, `SHGrid`, and `SHWindow`.
- Added an `info()` method to shtools constants that prints an info string.

- Changed the Python wrapper so that the output arrays in the SHExpand routines correspond to the optional input variable `lmax_calc`. Modified the dimensions by 1 for the output power spectrum in the wrapper functions for `SHBias` and `SHBiasK`.
- Changed the primary input parameter of `SHMTCouplingMatrix` to be a matrix of power spectra of the localization windows instead of a matrix of spherical-harmonic coefficients of spherical-cap localization windows. Furthermore, the output dimensions of the matrix have been switched.
- Added two fortran routines for performing multitaper spectral analyses when using windows generated from a mask, `SHMultiTaperMaskSE` and `SHMultiTaperCSE`.
- Minor modifications to the Python example scripts.
- Minor bug fix to the scripts that create the unix man documentation.
- Added Python notebook tutorials.
- Created an SHTOOLS development fund, funded by bitcoin donations.

Citation:

Mark Wieczorek et al. (2016). SHTOOLS: Version 3.3. Zenodo. doi:10.5281/zenodo.60010

Version 3.2

Change log:

- Added the optional argument `centralmeridian` to `Curve2Mask` that accounts for cases where the curve makes a complete circle in longitude about the planet.
- Fixed in bug in the python implementation where the outputs `error` and `corr` of `SHAdmitCorr` were switched.
- Added OpenMP support. When compiling with `make fortran-mp`, `make fortran2-mp`, or `make fortran3-mp`, saved variables in the subroutines are defined as being `threadprivate`.
- Optimized performance of the routines `SHMultiTaperSE`, `SHMultiTaperCSE`, and `SHLocalizedAdmitCorr`.
- Minor documentation fixes.

Citation:

Mark Wieczorek et al.. (2016). SHTOOLS: Version 3.2. Zenodo. doi:10.5281/zenodo.55790

Version 3.1

This release of SHTOOLS adds improved documentation for all Fortran 95 and Python routines, fixes several bugs, adds new functionalities, and adds additional example scripts.

Change log:

- Added OSX installation support via `brew`.
- Added `make install` that copies files to `/usr/local`.
- Reformatted all Fortran documentation files and rewrote the Python documentation and man pages.
- Added the routines `CilmMinus` and `CilmMinusRhoH`, which are the counterparts to `CilmPlus` and `CilmPlusRhoH`.
- Removed the following routines from the fortran documentation and Python wrappers: `DhAj`, `NGLQ`, `NGLQSH`, `NGLQSHN`.
- Removed the following redundant routines from SHTOOLS: `ComputeD0`, `SHMTVarOpt0`, `SHSjkPG0`.
- Renamed the routine `PreCompute` to `SHGLQ`.
- Renamed the routine `YilmIndex` to `YilmIndexVector`.
- Renamed the routine `Hilm` to `BAtoHilm`, and `HilmRhoH` to `BAtoHilmRhoH`.
- Renamed the routine `Wl` to `DownContFilterMA`, and `WlCurv` to `DownContFilterMC`.
- Added minimal support for three python classes: `SHGrid`, `SHCoeffs`, and `SHWindow`. These will be expanded upon in the next release.
- Added the routine `SHMTCouplingMatrix`.

Citation:

Mark A. Wieczorek, Matthias Meschede and Ilya Oshchepkov (2015). SHTOOLS - Tools for working with spherical harmonics (v3.1), ZENODO, doi:10.5281/zenodo.20920.

Version 3.0

This is a major release of SHTOOLS that adds full support for Python. In addition to Python support, this release contains minor bug fixes and improved documentation.

Change log:

- Added full python compatibility. This includes python wrappers for all SHTOOLS functions, python documentation for all functions, and a simple test suite to ensure that the SHTOOLS functions are working correctly.
- Moved the project from Sourceforge to GitHub. The Sourceforge project will no longer be maintained.
- Fixed bugs in `SHrtoc` and `SHctor` when the optional parameter `CONVENTION` was set to its default values, causing zeros to be returned.
- Modified `ComputeDMap` and `SJHReturnTapersMap` such that the parameter `SAMPLING` is optional.
- Fixed a bug in `Curve2Mask` that could give rise to vertical lines when the longitudes of the profile points were decreasing.
- Removed the routines `Import_Wisdom_From_File` and `Export_Wisdom_From_File`.

Citation:

Mark A. Wieczorek and Matthias Meschede (2015). SHTOOLS - Tools for working with spherical harmonics (v3.0), ZENODO, doi:10.5281/zenodo.15967 .

License

Summary: SHTOOLS is open source software and can be distributed in accordance with the 3-clause BSD license.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the author nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

This software is provided by the copyright holders and contributors “as is” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the copyright owner or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

Contributors

- Mark Wieczorek / MarkWieczorek (admin)
- Matthias Meschede / MMesch (admin)
- Elliott Sales de Andrade / QuLogic
- Ilya Oshchepkov / ioshchepkov
- xoviat
- Benda Xu / heroxbd
- Andrew Walker / andreww