AGU100 ADVANCING EARTH AND SPACE SCIENCE

Geochemistry, Geophysics, Geosystems

RESEARCH ARTICLE
10.1029/2018GC007529

SHTools: Tools for Working with Spherical Harmonics

Mark A. Wieczorek[1] and Matthias Meschede[2]

[1]Université Côte d'Azur, Observatoire de la Côte d'Azur, CNRS, Laboratoire Lagrange, Nice, France, [2]Institut de Physique du Globe de Paris, Sorbonne Paris Cité, Université Paris Diderot, UMR 7154 CNRS, Paris, France

Key Points:
- SHTools is an archive of Fortran 95 and Python software for spherical harmonic analyses
- The routines are fast, accurate, and support all normalization conventions used in the geosciences
- SHTools is open source, versioned by git, and is released with an unrestrictive license

Correspondence to:
M. A. Wieczorek,
mark.wieczorek@oca.eu

Citation:
Wieczorek, M. A., & Meschede, M. (2018). SHTools: Tools for working with spherical harmonics. *Geochemistry, Geophysics, Geosystems*, *19*, 2574–2592. https://doi.org/10.1029/2018GC007529

Received 6 MAR 2018
Accepted 8 MAY 2018
Accepted article online 12 MAY 2018
Published online 17 AUG 2018

Abstract Geophysical analyses are often performed in spherical geometry and require the use of spherical harmonic functions to express observables or physical quantities. When expanded to high degree, the accuracy and speed of the spherical harmonic transforms and reconstructions are of paramount importance. SHTools is a time and user-tested open-source archive of both Fortran 95 and Python routines for performing spherical harmonic analyses. The routines support all spherical-harmonic normalization conventions used in the geosciences, including 4π-normalized, Schmidt seminormalized, orthonormalized, and unnormalized harmonics, along with the option of employing the Condon-Shortley phase factor of $(-1)^m$. Data on the sphere can be sampled on a variety of grid formats, including equally spaced cylindrical grids and grids appropriate for integration by Gauss-Legendre quadrature. The spherical-harmonic transforms are proven to be fast and accurate for spherical harmonic degrees up to 2800. Several tools are provided for the geoscientist, including routines for performing localized spectral analyses and basic operations related to global gravity and magnetic fields. In the Python environment, operations are very simple to perform as a result of three class structures that encompass all operations on grids, spherical harmonic coefficients, and spatiospectral localization windows. SHTools is released under the unrestrictive BSD 3-clause license.

1. Introduction

The planets in our solar system are fundamentally spherical in nature, which often demands the use of spherical geometry when studying global geological phenomena. Thus, just as Fourier transforms are ubiquitous in Cartesian geometry, it is common to express observables and physical quantities as series of spherical harmonic functions on a planetary scale. Spherical harmonics are the natural basis functions for describing how a quantity varies across the surface of a sphere. As solutions to Laplace's equation, it is natural to express the radial and angular dependence of both gravitational and magnetic fields as a series involving spherical harmonics functions (e.g., Blakely, 1995; Jekeli, 2015; Wieczorek, 2015). As a result of simplified relations involving derivatives, it is convenient to express displacements in spherical harmonics when studying seismic wave propagation (e.g., Dahlen & Tromp, 1998), postglacial rebound (e.g., Peltier, 1974), and elastic flexure (e.g., Beuthe, 2008; Turcotte et al., 1981). Since any physical quantity that varies along a spherical interface can be expressed in spherical harmonics, the spectral properties of the function can be investigated by making use of its associated spherical harmonic coefficients.

Expanding a function into a series of spherical harmonic functions and reconstructing the function from the spherical harmonic coefficients are two of the most basic operations employed when working with data on the sphere. Whereas highly accurate transforms are an obvious requirement for any implementation of these operations, computational speed is also an important consideration given that many data sets are now routinely expanded to spherical harmonic degrees beyond 1000. Over the past two decades, a number of highly accurate and optimized software packages have been written to meet these needs. These include the C-based codes SHTns (Schaeffer, 2013), Libsharp (Reinecke & Seljebotn, 2013), and SpharmonicKit (http://www.cs.dartmouth.edu/geelong/sphere/), the Fortran-based code harmonic_synth (Holmes & Pavlis, 2006), and the MATLAB-based code GrafLab (Bucha & Janák, 2013). In addition to these generic codes, more specialized software packages exist such as the HEALpix package (Górski et al., 2005) that can be used with high-resolution pixelated data, the Fortran-based SPHEREPACK (Adams & Swarztrauber, 1999) that can be used with differential equations involving spherical harmonic functions, and a suite of MATLAB-based code that can be used for localized spectral analyses (Harig et al., 2015). Though these software packages have many similarities, they each differ in their ease of use, their graphical visualization capabilities, the

completness of the documentation, the manner that they are maintained, the programming languages they support, the conventions used for the gridded data and spherical harmonic functions, and the number of additional tools they provide for specialized applications.

SHTools is an archive of Fortran 95 and Python 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. SHTools is extremely versatile and easy to use. The software is open source, versioned by git, and released with an unrestrictive BSD 3-clause license. Though many of the routines are generic and could be used in any of the physical sciences, this package is tailored to the specific needs of the geosciences community. As examples, it can accommodate any standard normalization of the spherical harmonic functions, including 4π-normalized, Schmidt seminormalized, orthonormalized, and unnormalized harmonics. One can choose to use or exclude the Condon-Shortley phase factor of $(-1)^m$ with the associated Legendre functions, and both real and complex spherical harmonics are supported.

Besides providing basic routines for working with data expressed in spherical harmonics, several specialized tools are provided by SHTools for common geophysical problems. As examples, routines are provided for performing multitaper spectral analyses localized to arbitrarily shaped domains. Routines are provided for computing the three vector components of the gravitational and magnetic field on a flattened ellipsoid from their respective potential coefficients. For gravitational fields, one can also calculate the geoid, the gravity "gradient" tensor, and the gravitational potential associated with finite-amplitude surface relief to arbitrary precision. Extensive documentation is provided for all routines, either as unix man pages or within the Python environment.

The spherical harmonic transforms in SHTools are calculated using exact quadrature rules. Several grid types are available, including unevenly sampled grids in latitude that are appropriate for integration by Gauss-Legendre quadrature, or regularly sampled grids that conform to the sampling theorem of Driscoll and Healy (1994). This latter grid type is appropriate for cylindrically projected geographical data, where the grid nodes are equally spaced in degrees latitude and longitude. By the use of a simple scaling when calculating the associate Legendre functions (Holmes & Featherstone, 2002), the transforms are accurate to spherical harmonic degree 2800, which corresponds to a spatial resolution of less than 4 arc minutes. By using fast Fourier transforms when integrating and expanding over latitude bands, the routines are fast: on a modern desktop computer, spherical harmonic transforms and reconstructions take on the order of 1 s for bandwidths close to 800 and about 30 s for bandwidths close to 2,600. The core software is written in Fortran 95, and Python wrappers and class structures allow simple access to the Fortran-compiled routines. The Fourier transforms are computed by the software FFTW (Frigo & Johnson, 2005), and the Fortran routines are OpenMP thread-safe allowing for their use in parallelized programs.

In this document, we provide all information that is necessary to start using the SHTools software package. In section 2, we start by providing the definitions of the spherical harmonic functions for both real and complex fields. This includes descriptions of all the normalization conventions that can be found in the geosciences. Next, in section 3, we provide some of the implementation details regarding how the spherical harmonic expansions and reconstructions are performed. In sections 4 and 5, the Fortran 95 and Python components are described separately. This includes simple installation instructions and the definitions of common variables. For the Python environment, we provide a detailed description of the three main class structures that encompass most operations on grids, spherical harmonic coefficients, and spatiospectral localization windows. In section 6, we provide some simple examples of SHTools in the Python environment. Last, we end by discussing the software development roadmap, which includes better support for map projections, class structures for gravitational and magnetic field data, ultrahigh spherical harmonic transforms, and routines for reading and saving spherical harmonic coefficients in common formats.

2. Definitions

Many different conventions exist for both the associated Legendre and spherical harmonic functions, and this is often a source of confusion, even for those who are well versed in the topic. In this section, we provide all necessary definitions for both the real and complex 4π-normalized spherical harmonics that are commonly used in the fields of geodesy and spectral analysis. Though SHTools uses this normalization as the default for most routines, all other normalizations can be handled readily by specifying optional parameters. In Tables 1 and 2, the equivalent definitions are provided for the Schmidt seminormalized harmonics

Table 1
Normalization Conventions for Real Associated Legendre and Spherical Harmonic Functions

4π normalized	*Schmidt seminormalized*
$\bar{P}_{lm}(\mu)=\sqrt{(2-\delta_{m0})(2l+1)\frac{(l-m)!}{(l+m)!}}P_{lm}(\mu)$	$\bar{P}_{lm}(\mu)=\sqrt{(2-\delta_{m0})\frac{(l-m)!}{(l+m)!}}P_{lm}(\mu)$
$\int_{-1}^{1}\bar{P}_{lm}(\mu)\,\bar{P}_{l'm}(\mu)=2(2-\delta_{0m})\,\delta_{ll'}$	$\int_{-1}^{1}\bar{P}_{lm}(\mu)\,\bar{P}_{l'm}(\mu)=\frac{2(2-\delta_{0m})}{(2l+1)}\,\delta_{ll'}$
$\int_{\Omega}Y_{lm}(\theta,\phi)\,Y_{l'm'}(\theta,\phi)\,d\Omega=4\pi\,\delta_{ll'}\,\delta_{mm'}$	$\int_{\Omega}Y_{lm}(\theta,\phi)\,Y_{l'm'}(\theta,\phi)\,d\Omega=\frac{4\pi}{(2l+1)}\,\delta_{ll'}\,\delta_{mm'}$
$S_{fg}(l)=\sum_{m=-l}^{l}f_{lm}\,g_{lm}$	$S_{fg}(l)=\frac{1}{(2l+1)}\sum_{m=-l}^{l}f_{lm}\,g_{lm}$
Orthonormalized	*Unnormalized*
$\bar{P}_{lm}(\mu)=\sqrt{\frac{(2-\delta_{0m})(2l+1)}{4\pi}\frac{(l-m)!}{(l+m)!}}P_{lm}(\mu)$	$\bar{P}_{lm}(\mu)=P_{lm}(\mu)$
$\int_{-1}^{1}\bar{P}_{lm}(\mu)\,\bar{P}_{l'm}(\mu)=\frac{(2-\delta_{0m})}{2\pi}\,\delta_{ll'}$	$\int_{-1}^{1}\bar{P}_{lm}(\mu)\,\bar{P}_{l'm}(\mu)=\frac{2}{(2l+1)}\frac{(l+m)!}{(l-m)!}\,\delta_{ll'}$
$\int_{\Omega}Y_{lm}(\theta,\phi)\,Y_{l'm'}(\theta,\phi)\,d\Omega=\delta_{ll'}\,\delta_{mm'}$	$\int_{\Omega}Y_{lm}(\theta,\phi)\,Y_{l'm'}(\theta,\phi)\,d\Omega=\frac{4\pi\,(l+m)!}{(2-\delta_{0m})(2l+1)(l-m)!}\,\delta_{ll'}\,\delta_{mm'}$
$S_{fg}(l)=\frac{1}{4\pi}\sum_{m=-l}^{l}f_{lm}\,g_{lm}$	$S_{fg}(l)=\sum_{m=-l}^{l}\frac{(l+m)!}{(2-\delta_{0m})(2l+1)(l-m)!}f_{lm}\,g_{lm}$

that are used in the geomagnetism community, the orthonormalized harmonics that are used commonly in the seismology community, and the unnormalized harmonics that are widely used when only the lowest few degrees are of importance.

2.1. Real Spherical Harmonics

Spherical harmonics are the natural set of basis functions on the sphere, and any real square-integrable function can be expressed as a series of these functions as

$$f(\theta,\phi)=\sum_{l=0}^{\infty}\sum_{m=-l}^{l}f_{lm}\,Y_{lm}(\theta,\phi), \tag{1}$$

where f_{lm} is the spherical harmonic coefficient, Y_{lm} is the corresponding spherical harmonic function, θ is colatitude, ϕ is longitude, and l and m are the spherical harmonic degree and order, respectively. The real spherical harmonics are defined as

Table 2
Normalization Conventions for Complex Associated Legendre and Spherical Harmonic Functions

4π normalized	*Schmidt seminormalized*
$\bar{P}_{l}^{m}(\mu)=\sqrt{(2l+1)\frac{(l-m)!}{(l+m)!}}P_{lm}(\mu)$	$\bar{P}_{l}^{m}(\mu)=\sqrt{\frac{(l-m)!}{(l+m)!}}P_{lm}(\mu)$
$\int_{-1}^{1}\bar{P}_{l}^{m}(\mu)\,\bar{P}_{l'}^{m}(\mu)\,d\mu=2\,\delta_{ll'}$	$\int_{-1}^{1}\bar{P}_{l}^{m}(\mu)\,\bar{P}_{l'}^{m}(\mu)\,d\mu=\frac{2}{(2l+1)}\,\delta_{ll'}$
$\int_{\Omega}Y_{l}^{m*}(\theta,\phi)\,Y_{l'}^{m'}(\theta,\phi)\,d\Omega=4\pi\,\delta_{ll'}\,\delta_{mm'}$	$\int_{\Omega}Y_{l}^{m*}(\theta,\phi)\,Y_{l'}^{m'}(\theta,\phi)\,d\Omega=\frac{4\pi}{(2l+1)}\,\delta_{ll'}\,\delta_{mm'}$
$S_{fg}(l)=\sum_{m=-l}^{l}f_{l}^{m}\,g_{l}^{m*}$	$S_{fg}(l)=\frac{1}{(2l+1)}\sum_{m=-l}^{l}f_{l}^{m}\,g_{l}^{m*}$
Orthonormalized	*Unnormalized*
$\bar{P}_{l}^{m}(\mu)=\sqrt{\frac{(2l+1)}{4\pi}\frac{(l-m)!}{(l+m)!}}P_{lm}(\mu)$	$\bar{P}_{l}^{m}(\mu)=P_{lm}(\mu)$
$\int_{-1}^{1}\bar{P}_{l}^{m}(\mu)\,\bar{P}_{l'}^{m}(\mu)\,d\mu=\frac{1}{2\pi}\,\delta_{ll'}$	$\int_{-1}^{1}\bar{P}_{l}^{m}(\mu)\,\bar{P}_{l'}^{m}(\mu)=\frac{2}{(2l+1)}\frac{(l+m)!}{(l-m)!}\,\delta_{ll'}$
$\int_{\Omega}Y_{l}^{m*}(\theta,\phi)\,Y_{l'}^{m'}(\theta,\phi)\,d\Omega=\delta_{ll'}\,\delta_{mm'}$	$\int_{\Omega}Y_{l}^{m*}(\theta,\phi)\,Y_{l'}^{m'}(\theta,\phi)\,d\Omega=\frac{4\pi}{(2l+1)}\frac{(l+m)!}{(l-m)!}\,\delta_{ll'}\,\delta_{mm'}$
$S_{fg}(l)=\frac{1}{4\pi}\sum_{m=-l}^{l}f_{l}^{m}\,g_{l}^{m*}$	$S_{fg}(l)=\sum_{m=-l}^{l}\frac{(l+m)!}{(2l+1)(l-m)!}f_{l}^{m}\,g_{l}^{m*}$

$$Y_{lm}(\theta,\phi)=\begin{cases}\bar{P}_{lm}(\cos\theta)\cos m\phi & \text{if } m\geq 0\\ \bar{P}_{l|m|}(\cos\theta)\sin|m|\phi & \text{if } m<0,\end{cases} \tag{2}$$

where the normalized associated Legendre functions are given by

$$\bar{P}_{lm}(\mu)=\sqrt{(2-\delta_{m0})(2l+1)\frac{(l-m)!}{(l+m)!}}P_{lm}(\mu), \tag{3}$$

and where δ_{ij} is the Kronecker delta function. The unnormalized associated Legendre functions are derived from the standard Legendre polynomials using the relations

$$P_{lm}(\mu)=\left(1-\mu^2\right)^{m/2}\frac{d^m}{d\mu^m}P_l(\mu) \tag{4}$$

and

$$P_l(\mu)=\frac{1}{2^l l!}\frac{d^l}{d\mu^l}\left(\mu^2-1\right)^l. \tag{5}$$

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

$$\int_{-1}^{1}\bar{P}_{lm}(\mu)\,\bar{P}_{l'm}(\mu)=2(2-\delta_{0m})\,\delta_{ll'}, \tag{6}$$

and the spherical harmonics are orthogonal over both degree l and order m according to

$$\int_{\Omega}Y_{lm}(\theta,\phi)\,Y_{l'm'}(\theta,\phi)\,d\Omega=4\pi\,\delta_{ll'}\,\delta_{mm'}, \tag{7}$$

where $d\Omega$ is the differential surface area on the unit sphere $\sin\theta\,d\theta\,d\phi$. By multiplying equation (1) by $Y_{l'm'}$ and integrating over all space, it is straightforward to show that the spherical harmonic coefficients of a function can be calculated by the integral

$$f_{lm}=\frac{1}{4\pi}\int_{\Omega}f(\theta,\phi)\,Y_{lm}(\theta,\phi)\,d\Omega. \tag{8}$$

The normalized Legendre functions are efficiently calculated using standard three-term recursion relations (e.g., Holmes & Featherstone, 2002), and the calculation of the integral in equation (8) will be discussed further in section 3. It is important to note that the above definition of the Legendre functions does not include the Condon-Shortley phase factor of $(-1)^m$ that is often employed in the physics and seismology communities (e.g., Dahlen & Tromp, 1998; Varshalovich et al., 1988). Nevertheless, this phase can be included in most SHTools routines by specifying an optional parameter.

A few simple properties allow to visualize the spherical harmonic functions: A harmonic possesses $2|m|$ zero crossings in the longitudinal direction, and $l-|m|$ zero crossings in latitude. When the angular order m is zero, the harmonics are called zonal and oscillate only in the latitudinal direction. When $l=|m|$ the harmonics are called sectoral and oscillate only in the longitudinal direction. For other values of m the harmonics are called tesseral and oscillate in both the longitudinal and latitudinal directions (see Figure 1). Negative and positive values of a given m correspond to the cosine and sine components in equation (2) and determine the longitudinal phase of the harmonic. The equivalent Cartesian wavelength for a spherical harmonic function of degree l is approximately $\lambda=2\pi R/\sqrt{l(l+1)}$, where R is the radius of the sphere, a result known as the Jeans relation. In many fields, it is common to refer to the positive order spherical harmonic coefficient as C_{lm}, and the negative order coefficient as S_{lm}, in reference to the cosine and sine components of equation (2). In the geomagnetism community, these are instead referred to as g_{lm} and h_{lm}. Furthermore, in some fields, one refers to the zonal C_{l0} coefficients as $-J_l$.

It is straightforward to generalize Parseval's theorem from Cartesian geometry 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 can be shown to be equal to a sum over its power spectrum

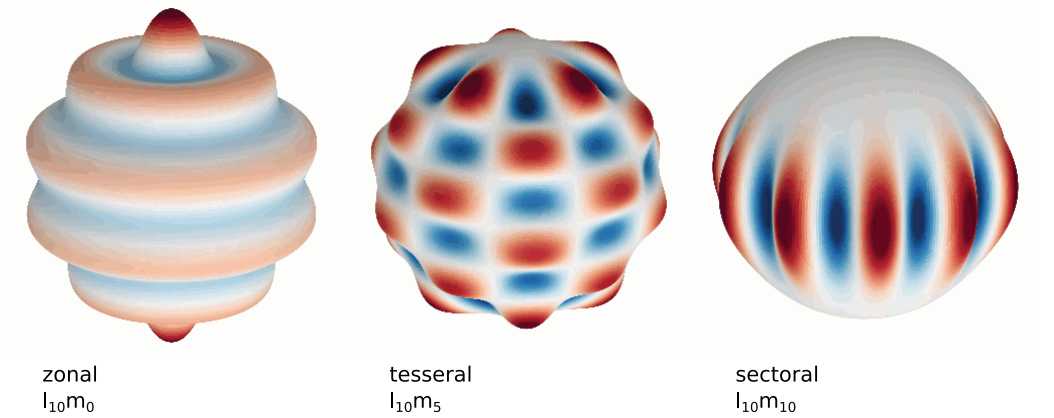

Figure 1. 3-D illustrations of zonal, tesseral, and sectoral spherical harmonic functions. These images were generated in Python using the method `plot3d()` of the class `SHGrid`.

$$\frac{1}{4\pi}\int_{\Omega} f^2(\theta,\phi)\, d\Omega = \sum_{l=0}^{\infty} S_{ff}(l), \tag{9}$$

where the power spectrum S is related to the spherical harmonic coefficients by

$$S_{ff}(l) = \sum_{m=-l}^{l} f_{lm}^2. \tag{10}$$

It can be shown that the power spectrum is unmodified by a rotation of the coordinate system. Furthermore, it should be noted that the numerical value of the power spectrum is independent of the normalization convention used for the spherical harmonic functions (though the mathematical form of S will be different). Similarly, the cross power of two functions f and g is given by

$$\frac{1}{4\pi}\int_{\Omega} f(\theta,\phi)\, g(\theta,\phi)\, d\Omega = \sum_{l=0}^{\infty} S_{fg}(l), \tag{11}$$

with

$$S_{fg}(l) = \sum_{m=-l}^{l} f_{lm}\, g_{lm}. \tag{12}$$

If the functions f and g have a zero mean, then S_{ff} and S_{fg} represent the contribution to the variance and covariance, respectively, as a function of degree l.

The term "power spectrum" is often used ambiguously, with some definitions differing by factors of 4π and $(2l+1)$. Here, S refers to the total power of the function at spherical harmonic degree l, which we will call the *power per degree*. Alternatively, one can calculate the average power per coefficient at spherical harmonic degree l, which we will refer to as the *power per lm*. Since there are $(2l+1)$ spherical harmonic coefficients at degree l, this is simply

$$\text{power per } lm = \frac{S(l)}{(2l+1)}. \tag{13}$$

One can calculate the power from all angular orders over an infinitesimal logarithmic spherical harmonic degree band $d\log_a l$, where a is the logarithmic base. We refer to this as the *power per* $d\log_a l$, which is given by

$$\text{power per } d\,\log_a l = S(l)\, l \ln a. \tag{14}$$

Finally, we define the energy of a function as the integral of its square. The energy spectrum is thus equal to the power spectrum multiplied by 4π.

2.2. Complex Spherical Harmonics

Complex functions can also be expressed as a spherical harmonics series with complex harmonics and coefficients. To differentiate real from complex coefficients and functions, we will use mixed subscript-superscripts for the complex quantities, and only subscripts for their real counterparts. With this convention, any complex square-integrable function f can be expressed in spherical harmonics as

$$f(\theta,\phi)=\sum_{l=0}^{\infty}\sum_{m=-l}^{l} f_l^m\, Y_l^m(\theta,\phi). \tag{15}$$

The complex spherical harmonics are defined as

$$Y_l^m(\theta,\phi)=\bar{P}_l^m(\cos\theta)\, e^{im\phi}, \tag{16}$$

where the normalized associated Legendre functions are given by

$$\bar{P}_l^m(\mu)=\sqrt{(2l+1)\frac{(l-m)!}{(l+m)!}}P_{lm}(\mu), \tag{17}$$

and where the unnormalized Legendre functions are as given before by equations (4) and (5). The normalized Legendre functions are orthogonal for each order m as given by

$$\int_{-1}^{1}\bar{P}_l^m(\mu)\,\bar{P}_{l'}^m(\mu)\,d\mu=2\,\delta_{ll'}. \tag{18}$$

The complex spherical harmonics possess a symmetry relationship for positive and negative angular orders

$$Y_l^{m*}(\theta,\phi)=(-1)^m\, Y_l^{-m}(\theta,\phi), \tag{19}$$

where the asterisk denotes complex conjugation, and satisfy the orthogonality relationship

$$\int_{\Omega} Y_l^{m*}(\theta,\phi)\, Y_{l'}^{m'}(\theta,\phi)\,d\Omega=4\pi\,\delta_{ll'}\,\delta_{mm'}. \tag{20}$$

Using this relationship, it is straightforward to show that the spherical harmonic coefficients of a complex function can be calculated by the integral

$$f_l^m=\frac{1}{4\pi}\int_{\Omega} f(\theta,\phi)\, Y_l^{m*}(\theta,\phi)\,d\Omega. \tag{21}$$

The generalized Parseval's theorem for complex functions is given by

$$\frac{1}{4\pi}\int_{\Omega} f(\theta,\phi)\, f^*(\theta,\phi)\,d\Omega=\sum_{l=0}^{\infty} S_{ff}(l), \tag{22}$$

where the power spectrum is

$$S_{ff}(l)=\sum_{m=-l}^{l} f_l^m\, f_l^{m*}. \tag{23}$$

Similarly, the cross-power of two functions f and g is given by

$$\frac{1}{4\pi}\int_{\Omega} f(\theta,\phi)\, g^*(\theta,\phi)\,d\Omega=\sum_{l=0}^{\infty} S_{fg}(l), \tag{24}$$

where the cross-power spectrum is

$$S_{fg}(l)=\sum_{m=-l}^{l} f_l^m\, g_l^{m*}. \tag{25}$$

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. 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

$$f_l^m = \begin{cases} (f_{lm} - i f_{l-m})/\sqrt{2} & \text{if } m > 0 \\ f_{l0} & \text{if } m = 0 \\ (-1)^m f_l^{-m*} & \text{if } m < 0. \end{cases} \tag{26}$$

3. Implementation Details

The majority of the routines in SHTools make use of spherical harmonic transforms and reconstructions, and we describe the main details of how these operations are performed in this section. As with most other software packages, these operations make use of fast Fourier transforms over latitude bands, which decrease the computational time dramatically with respect to a more naive implementation (e.g., Sneeuw, 1994). Furthermore, symmetry relations about the equator are used to halve the amount computational effort when calculating the associated Legendre functions. As we demonstrate, the transforms are both accurate and fast up to about spherical harmonic degree 2800.

We start with the reconstruction of a function from its spherical harmonic coefficients. Using the separate variables C_{lm} and S_{lm} for the cosine and sine coefficients, respectively, and after interchanging the order of summations over l and m, equation (1) can be written as

$$f(\theta, \phi) = \sum_{m=0}^{L} \sum_{l=m}^{L} (C_{lm} \bar{P}_{lm}(\cos\theta) \cos m\phi + S_{lm} \bar{P}_{lm}(\cos\theta) \sin m\phi), \tag{27}$$

where the summation is truncated at a maximum spherical harmonic degree L that is appropriate for the analysis. Making use of two component vectors

$$(a_m(\theta), b_m(\theta)) = \sum_{l=m}^{L} (C_{lm}, S_{lm}) \bar{P}_{lm}(\cos\theta), \tag{28}$$

Equation (27) can be written more simply as

$$f(\theta, \phi) = \sum_{m=0}^{L} (a_m(\theta) \cos m\phi + b_m(\theta) \sin m\phi). \tag{29}$$

For a given latitude band, the function f 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 & Johnson, 2005) that supports grids of arbitrary length and both real and complex data. To adequately sample the function in longitude, a minimum of $2L+1$ data points should be employed.

The slowest part of reconstructing the function f involves the computation of the coefficients a_m and b_m. These coefficients depend upon the associated Legendre functions, which are calculated efficiently using standard three-term recursion relations over adjacent spherical harmonic degrees (e.g., Holmes & Featherstone, 2002). For a given colatitude, the sectoral term $\bar{P}_{mm}$ is first calculated using an analytic equation, and then $\bar{P}_{lm}$ is calculated for all values of $l > m$. It is well known, however, that the standard formulas can lead to numerical underflows near the poles for large values of m, even when using double precision floating point numbers. To circumvent this problem, the associated Legendre functions are here calculated using the approach of Holmes and Featherstone (2002), where the sectoral $\bar{P}_{mm}$ terms are multiplied by $10^{280} \sin^m\theta$ prior to performing the recursions, and then appropriately unscaled at the end of the recursion.

For the spherical harmonic transform, we start by writing equation (8) in a two-component vector notation, where the two elements are for the cosine and sine spherical harmonic coefficients, respectively:

$$(C_{lm}, S_{lm}) = \frac{1}{4\pi} \int_0^{2\pi} \int_0^{\pi} f(\theta, \phi) \bar{P}_{lm}(\theta) (\cos m\phi, \sin m\phi) \sin\theta \, d\theta \, d\phi. \tag{30}$$

Defining the two intermediary variables

$$\left(c_{lm}^{(i)}, s_{lm}^{(i)}\right) = \int_0^{2\pi} f(\theta_i, \phi) (\cos m\phi, \sin m\phi) d\phi, \tag{31}$$

it is seen that for a given colatitude θ_i and degree l, all of the angular orders can be calculated at once by making use of a fast Fourier transform of the function f. By replacing the integral over latitude in equation (30) with a numerical quadrature rule, the spherical harmonic coefficients can be calculated as

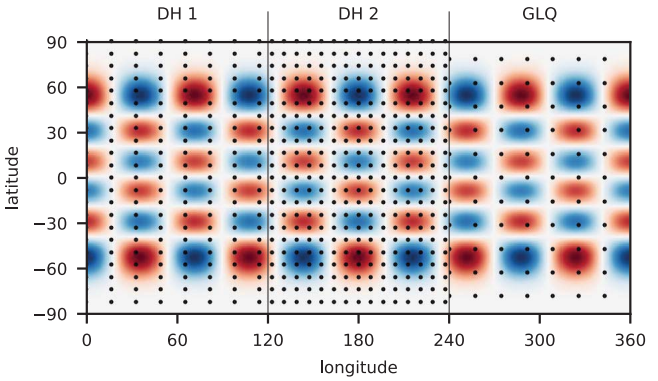

Figure 2. Schematic diagram illustrating the properties of the grids used with the Driscoll and Healy and Gauss-Legendre quadrature routines in SHTools.

$$(C_{lm}, S_{lm}) = \frac{1}{4\pi} \sum_{i=1}^{N} w_i \bar{P}_{lm}(\cos\theta_i) \left(c_{lm}^{(i)}, s_{lm}^{(i)}\right). \quad (32)$$

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

It is possible to choose the weights w_i and the locations of the latitudinal sampling points θ_i such that the quadrature in equation (32) is exact. SHTools implements two such quadrature rules, one based on Gauss-Legendre quadrature (e.g., Press et al., 1992) and the other based on the sampling theorem of Driscoll and Healy (1994). In both techniques, the quadrature is exact only when the function being integrated is a terminating polynomial, i.e., when the spherical harmonic degree is limited to a maximum value L. As seen in equation (32), the functions c_{lm} and s_{lm} can be approximated as polynomials of maximum degree L, and when multiplied by the associated Legendre function, the integrand is approximately a polynomial of maximum degree $2L$. For the case of Gauss-Legendre quadrature, the quadrature is exact when the function f is sampled in latitude at the $(L+1)$ zeros of the Legendre Polynomial of degree $(L+1)$. Since the function also needs to be sampled on $(2L+1)$ equally space grid nodes for the Fourier transforms in longitude, the function f is sampled on a grid of size $(L+1)\times(2L+1)$.

The second type of quadrature used in SHTools is appropriate for data that are sampled on regular grids. We make use of the work of Driscoll and Healy (1994) who showed that an exact quadrature exists when the function f is sampled at N equally spaced nodes in latitude and N equally spaced nodes in longitude. For this sampling, the grids make use of the latitude band at 90°N, but not 90°S, and the number of samples is $2(L+1)$, which is always even. Given that the sampling in latitude was imposed a priori, it should not be a surprise that these grids contain almost twice as many samples in latitude as the grids used with Gauss-Legendre quadrature. The convenience of using a regularly spaced grid comes with the drawback of being computationally slower by about a factor of two. In the geosciences, it is common to work with grids that are equally spaced in degrees latitude and longitude, such as with data sets expressed in cylindrical projections, and SHTools provides the option of using grids of size $N\times 2N$. For this case, when performing the Fourier transform in equation (31), the coefficients c_{lm} and s_{lm} with $m > L$ are simply discarded. A graphical summary of the Gauss-Legendre quadrature and Driscoll and Healy (1994) grids is shown in Figure 2, and the properties of these grids are summarized in Table 3.

The accuracy of the spherical harmonic transforms depends upon a number of factors, including the accuracy of the numerical quadrature, the accuracy of the Legendre functions, the accumulation of roundoff errors, and the nature of the function being transformed. Concerning the numerical quadrature, it is noted that this will be exact when the integrand is a terminating polynomial. Though the zonal Legendre functions P_{l0} are polynomials of degree l, the Legendre functions for $m > 0$ are not. Nevertheless, after replacing the function f by its spherical harmonic series, the integrand in equation (32) is seen to consist of a series of products of two Legendre functions P_{lm} and $P_{l'm'}$. It can be verified that when $m+m'$ is even, this product is a polynomial of order $l+l'$. The present algorithm makes the assumption that this product is also a polynomial when $m+m'$ is odd. Tests using more samples in latitude than required by this assumption do not give rise to any noticeable improvement, indicating that this assumption is valid for all practical purposes.

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 l^2 or l^{-2}. It is noted that a power spectrum proportional to l^{-2} is representative of

Table 3
Properties of the Three Grid Types Used in SHTools

	DH 1	DH 2	GLQ
Name	Driscoll-Healy	Driscoll-Healy	Gauss-Legendre Quadrature
Shape ($N_{lat}\times N_{lon}$)	$N \times N$	$N\times 2N$	$N\times(2N-1)$
L	$N/2-1$	$N/2-1$	$N-1$
N	$2L+2$	$2L+2$	$L+1$
$\Delta\theta$	$180^\circ/N$	$180^\circ/N$	Variable
$\Delta\phi$	$360^\circ/N$	$180^\circ/N$	$360^\circ/(2N-1)$

Note. $\Delta\theta$ and $\Delta\phi$ are the latitudinal and longitudinal grid spacings in degrees, respectively.

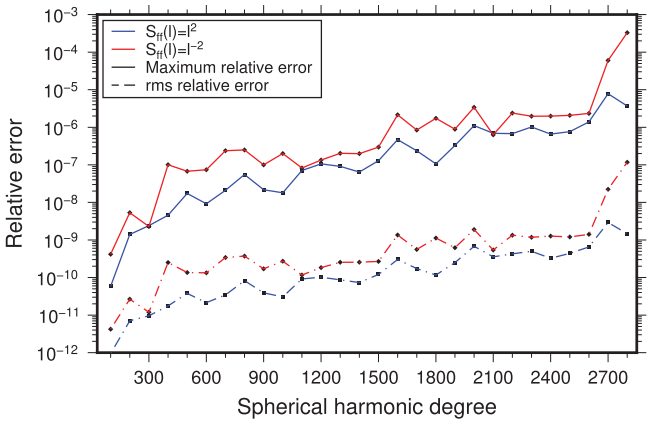

Figure 3. Maximum relative errors (solid lines) and rms relative errors (dashed lines) of the spherical harmonic coefficients as a function of spherical harmonic bandwidth. The function was first reconstructed on a grid appropriate for Gauss-Legendre quadrature, expanded into spherical harmonics, and then compared with the initial coefficients.

many geophysical observables, such as gravity and topography. For a given spherical harmonic bandwidth L, 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.

Figure 3 plots these two errors as a function of the bandwidth of the initial function. 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 L are typically 3 orders of magnitude smaller than the maximum relative errors. While the errors are slightly larger for the set of coefficients that possessed a power spectrum proportional to l^2, the difference with respect to the coefficients with the l^{-2} power spectrum is modest, implying that the form of the data does 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.

Finally, we tested the speed of the spherical harmonic reconstructions and transforms 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 Figure 4 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. For the real Gauss-Legendre quadrature routines, the transform time is seen to be on the order of one second for degrees close to 800 and about 30 s 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 min for degrees close to 2600. The complex routines are slower by a factor of about 1.4.

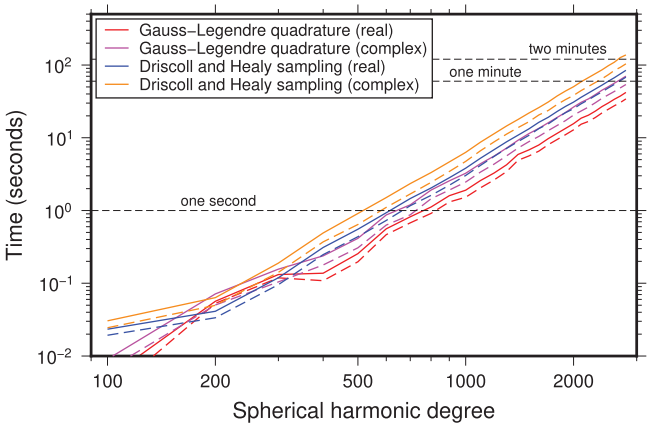

Figure 4. Time to perform the reconstruction of a function from its spherical harmonic coefficients (solid lines) and the spherical harmonic transform of the function (dashed lines). Plotted are timing results as a function of spherical harmonic bandwidth using the real and complex Gauss-Legendre and Driscoll and Healy (1994) quadrature implementations. Calculations were performed on a Mac Pro 2.7 GHz 12 Core Intel Xeon E5 using 64 bit executables and level 3 optimizations.

4. SHTools for Fortran 95

Installation of the Fortran 95 components of SHTools is straightforward and has been tested on multiple platforms using multiple compilers. Before attempting an installation, it will first be necessary to install both the FFTW (Frigo & Johnson, 2005) and LAPACK software packages. After having done this, the most generic way to install the SHTools archive is by use of the unix command-line utility `make`. To build both the standard library, as well as the thread-safe library that is OpenMP compatible, it is only necessary to execute

```
make fortran fortran-mp
```

in a unix terminal. By default, the archive will be built using the `gfortran` compiler, but alternative compilers and compiler flags can be used by specifying optional arguments to the `make` variables `F95` and `F95FLAGS`. Tested compilers include Absoft ProFortran (`f95`), Intel fortran (`ifort`) and `g95`. A series of tests programs can be built and executed using

```
make fortran-tests
```

and the compiled library, modules, man pages, and web documentation can be made available on a system level by executing

```
make install-fortran
```

Alternatively, if using the macOS operating system, the Fortran 95 components can be installed using the `brew` package manager by executing

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

Additional installation instructions can be found in the web documentation.

Almost all capabilities of SHTools can be accessed directly from within any Fortran 95 program. To call an SHTools routine, it is necessary to make use of the `shtools` and/or `planetsconstants` modules by including the commands

```
use shtools
use planetsconstants
```

in the source file. These should be placed after the program, subroutine, or function declaration, but before any statements such as `implicit none`. The second use statement is only necessary if one needs to access the constants defined by SHTools, such as the mean radii and *GM* of the planets. When compiling the program, it is necessary to ensure that the SHTools library file `libSHTOOLS.a` and the directory containing the module files are accessible to the compiler. For a standard installation, these should be found `in/usr/local/lib and/usr/local/include`, respectively.

The SHTools package contains a large number of routines that can be accessed by Fortran 95 programs. Each of these can be found in the web documentation, and each has an associated man page that can be accessed from a unix terminal. The routines are organized into seven main themes, which include: (1) Legendre functions, (2) spherical harmonic transforms and reconstructions, (3) spherical harmonic input/output, storage, and conversions, (4) global spectral analyses, (5) localized spectral analyses, (6) spherical harmonic rotations, and (7) specialized routines for working with gravity and magnetic fields. In general, most of these routines use 4π-normalized harmonics that exclude the Condon-Shortley phase factor as the default. (An exception to this rule are the magnetics routines that employ Schmidt seminormalized harmonics.) Regardless, the normalization convention can be modified in most routines by specifying the optional parameters `norm` and `csphase`, of which `norm` accepts the following values:

- `1`: 4π normalized harmonics
- `2`: Schmidt semi-normalized harmonics
- `3`: unnormalized harmonics
- `4` orthonormalized harmonics.

By setting `csphase` $= -1$, the factor $(-1)^m$ will be appended to either the associated Legendre functions or the spherical harmonic functions.

5. SHTools for Python

The Python package *pyshtools* provides access to the Fortran-95 SHTOOLS library by use of Python wrapper functions. This package requires the FFTW and LAPACK libraries to be installed, as well as the Python packages *numpy*, *scipy*, and *matplotlib*. With these dependencies installed, it is only necessary to execute the following command in a unix terminal to install *pyshtools*:

```
pip install pyshtools
```

In the Python environment, the command

```
import pyshtools
```

will load several packages into the *pyshtools* namespace. These packages are very similar to the seven high-level groupings of functions in the SHTOOLS library, and include

- `shclasses`: All *pyshtools* classes and subclasses,
- `shtools`: All Fortran-95 wrapped SHTools routines,
- `legendre`: Legendre functions,
- `expand`: Spherical harmonic expansion routines,
- `shio`: Spherical harmonic input/output, storage, and conversion routines,
- `spectralanalysis`: Global and local spectral analysis routines,
- `rotate`: Spherical harmonic rotation routines,
- `gravmag`: Gravity and magnetics routines,
- `constant`: Planetary constants,
- `utils`: Utilities.

In addition to loading these packages, three high-level class structures will be loaded into the primary namespace that provided easy access to most operations on spherical harmonic coefficients, grids, and localization windows:

- `SHCoeffs`: A high-level class for spherical harmonic coefficients,
- `SHGrid`: A high-level class for global grids,
- `SHWindow`: A high-level class for localization windows.

The vast majority of the *pyshtools* functionality can be accessed via the three high-level classes. Using these classes, operations are exceedingly simple to perform, as all required metadata are stored as class attributes. These attributes include the spherical harmonic normalization, Condon-Shortley phase convention, maximum spherical harmonic degree, data type, grid type, and grid size. Once the classes are initialized, it is only necessary to execute one of its methods, usually without any arguments, to perform basic operations. As an example, if `clm` is an `SHCoeffs` instance, then reconstructing these coefficients on a grid, and re-expanding this grid into spherical harmonics is as easy as

```
grid = clm.expand()
clm2 = grid.expand()
```

Tables (4–6) provide all the attributes and methods that are implemented for these three classes.

When using the package IPython, which adds improved interactive functionality to Python, the available *pyshtools* routines can be explored by typing

```
pyshtools.[ tab]
```

where `[ tab]` is the tab key. To read the documentation of a class, method, or routine in IPython, such as `SHCoeffs`, it is only necessary to enter

```
pyshtools.SHCoeffs?
```

To access an SHTools constant and its associated info string, such as the mass of Mars, one would use the commands

```
pyshtools.constant.mass_mars
pyshtools.constant.mass_mars.info()
```

Documentation for the Python functions used in SHTools can also be accessed by their unix man pages, by appending `py` to the name and using all lower case letters.

6. Examples

The Python *pyshtools* package provides three classes for interacting with spherical harmonic coefficients, grids, and localization windows: `SHCoeffs`, `SHGrid`, and `SHWindow`. These classes provide simple methods for initializing class instances, visualizing spectra and grids, basic data transformations, and output to arrays and files. In this section, we first provide simple demonstrations of basic operations involving

Table 4
Attributes and Methods of the pyshtools *Class* `SHCoeffs`

Attributes	
`lmax`	The maximum spherical harmonic degree of the coefficients.
`coeffs`	The raw coefficients with the specified normalization and csphase conventions.
`normalization`	The normalization of the coefficients: `4pi`, `ortho`, `schmidt`, or `unnorm`.
`csphase`	Defines whether the Condon-Shortley phase is used (1) or not (−1).
`mask`	A boolean mask that is True for the permissible values of degree *l* and order *m*.
`kind`	The coefficient data type: either `complex` or `real`.
Methods	
`to_array()`	Return an array of spherical harmonic coefficients, optionally with a different normalization convention.
`to_file()`	Save raw spherical harmonic coefficients to a file.
`degrees()`	Return an array listing the spherical harmonic degrees from 0 to lmax.
`spectrum()`	Return the spectrum of the function as a function of spherical harmonic degree.
`set_coeffs()`	Set coefficients in-place to specified values.
`rotate()`	Rotate the coordinate system used to express the spherical harmonic coefficients and return a new class instance.
`convert()`	Return a new class instance using a different normalization convention.
`pad()`	Return a new class instance where the coefficients are zero padded or truncated to a different lmax.
`expand()`	Evaluate the coefficients either on a spherical grid and return an SHGrid class instance, or for a list of latitude and longitude coordinates.
`copy()`	Return a copy of the class instance.
`plot_spectrum()`	Plot the spectrum as a function of spherical harmonic degree.
`plot_spectrum2d()`	Plot the 2-D spectrum of all spherical harmonic coefficients.
`info()`	Print a summary of the data stored in the `SHCoeffs` instance.

spherical harmonic coefficients and grids. Following this, the basic techniques for performing localized spectral analyses are demonstrated using both spherical cap and arbitrarily shaped localization windows.

6.1. `SHCoeffs` Class

The `SHCoeffs` class provides all methods for working with spherical harmonic coefficients. Each class instance saves as attributes the properties of the coefficients, including the normalization, the Condon-Shortley phase convention, the spherical harmonic bandwidth, and whether the coefficients are real or

Table 5
Attributes and Methods of the pyshtools *Class SHGrid*

Attributes	
`data`	Gridded array of the data.
`nlat, nlon`	The number of latitude and longitude bands in the grid.
`lmax`	The maximum spherical harmonic degree that can be resolved by the grid sampling.
`sampling`	For Driscoll and Healy grids, the longitudinal sampling of the grid. Either `1` for `nlong= nlat` or `2` for `nlong= 2*nlat`.
`kind`	Either `complex` or `real` for the data type.
`grid`	Either `DH` or `GLQ` for Driscoll and Healy grids or Gauss-Legendre quadrature grids.
`zeros`	The cos(colatitude) nodes used with Gauss-Legendre quadrature grids. Default is `None`.
`weights`	The latitudinal weights used with Gauss-Legendre quadrature grids. Default is `None`.
Methods	
`to_array()`	Return the raw gridded data as a numpy array.
`to_file()`	Save gridded data to a text or binary file.
`lats()`	Return a vector containing the latitudes of each row of the gridded data.
`lons()`	Return a vector containing the longitudes of each column of the gridded data.
`expand()`	Expand the grid into spherical harmonics.
`copy()`	Return a copy of the class instance.
`plot()`	Plot the raw data using a simple cylindrical projection.
`plot3d()`	Plot the raw data on a 3d sphere.
`info()`	Print a summary of the data stored in the `SHGrid` instance.

Table 6
Attributes and Methods of the pyshtools *Class SHWindow*

Attributes	
`kind`	Either `cap` or `mask`.
`tapers`	Matrix containing the spherical harmonic coefficients (in packed form) of either the unrotated spherical cap localization windows or the localization windows corresponding to the input mask.
`coeffs`	Array of spherical harmonic coefficients of the rotated spherical cap localization windows. These are `4pi` normalized and do not use the Condon-Shortley phase factor.
`eigenvalues`	Concentration factors of the localization windows.
`orders`	The angular orders for each of the spherical cap localization windows.
`weights`	Taper weights used with the multitaper spectral analyses. Default is `None`.
`lwin`	Spherical harmonic bandwidth of the localization windows.
`theta`	Angular radius of the spherical cap localization domain (default in degrees).
`theta_degrees`	`True` (default) if `theta` is in degrees.
`nwin`	Number of localization windows. Default is $(\texttt{lwin}+1)^2$.
`nwinrot`	The number of best concentrated windows that were rotated and whose coefficients are stored in `coeffs`.
`clat, clon`	Latitude and longitude of the center of the rotated spherical cap localization windows (default in degrees).
`coord_degrees`	`True` (default) if `clat` and `clon` are in degrees.
Methods	
`to_array()`	Return an array of the spherical harmonic coefficients for taper i, where i=0 is the best concentrated, optionally using a different normalization convention.
`to_shcoeffs()`	Return the spherical harmonic coefficients of taper i, where i=0 is the best concentrated, as a new `SHCoeffs` class instance, optionally using a different normalization convention.
`to_shgrid()`	Return as a new SHGrid instance a grid of taper i, where i=0 is the best concentrated window.
`number_concentrated()`	Return the number of windows that have concentration factors greater or equal to a specified value.
`degrees()`	Return an array containing the spherical harmonic degrees of the localization windows, from `0` to `lwin`.
`spectra()`	Return the spectra of one or more localization windows.
`rotate()`	Rotate the spherical cap tapers, originally located at the north pole, to `clat` and `clon` and save the spherical harmonic coefficients in the attribute `coeffs`.
`coupling_matrix()`	Return the coupling matrix of the first `nwin` localization windows.
`biased_spectrum()`	Calculate the multitaper (cross-) spectrum expectation of a localized function.
`multitaper_spectrum()`	Return the multitaper power spectrum estimate and uncertainty for the input `SHCoeffs` class instance.
`multitaper_cross_spectrum()`	Return the multitaper cross-power spectrum estimate and uncertainty for two input `SHCoeffs` class instances.
`copy()`	Return a copy of the class instance.
`plot_windows()`	Plot the best concentrated localization windows using a simple cylindrical projection.
`plot_spectra()`	Plot the spectra of the best-concentrated localization windows.
`plot_coupling_matrix()`	Plot the multitaper coupling matrix.
`info()`	Print a summary of the data stored in the `SHWindow` instance.

complex. Once these attributes are set, it is not necessary to specify these parameters when calling most of the class methods. Most methods do not require the use of optional calling parameters, but we use them here to demonstrate the full capabilities of the package.

Initialization of a new `SHCoeffs` class instance can be done in several ways. Random coefficients can be generated from a specified input power spectrum using `from_random()`, the coefficients can be read from a file using `from_file()`, the coefficients can be set to zero using `from_zeros()`, and the coefficients can be read from a *numpy* array using `from_array()`. By default, these methods assume that the coefficients are real 4π-normalized excluding the Condon-Shortley phase factor, but this convention can be modified by specifying optional parameters as demonstrated in these initialization examples:

```
clm = pyshtools.SHCoeffs.from_random(power, normalization= '4pi' )
clm = pyshtools.SHCoeffs.from_file(filename, format= 'shtools,' lmax = 50)
clm = pyshtools.SHCoeffs.from_zeros(lmax, kind= 'complex' )
clm = pyshtools.SHCoeffs.from_array(array, csphase = -1)
```

Here, `power` is an array containing the input power spectrum, `lmax` is the maximum spherical harmonic degree to read from the file `filename`, `normalization` specifies the spherical harmonic normalization (`4pi`, `schmidt`, `ortho`, or `unnorm`), `kind` specifies whether the coefficients are `real` or `complex`, and `csphase` determines whether the Condon-Shortley phase factor is used (`-1`) or not (`1`). If the raw data ever need to be accessed, they are stored in the attribute `coeffs`.

The spectrum of the coefficients, which is independent of the chosen normalization, can be output as a *numpy* array using the `spectrum()` method. By default, the output spectrum corresponds to the power per degree of the function with `unit = 'per_l.'` The energy spectrum can be output by specifying `convention= 'energy,'` and the `unit` of the spectrum can be set alternatively to either `per_lm` or `per_dlogl`. Two methods are provided for visualizing the spectrum graphically: `plot_spectrum()` plots the spectrum as a function of spherical harmonic degree, and `plot_spectrum2d()` plots the spectrum as a function of degree and angular order. The following two lines of code demonstrate how to calculate and plot the power per coefficient spectrum:

```
power = clm.spectrum(convention= 'power,' unit= 'per_lm' )
fig, ax = clm.plot_spectrum(unit= 'per_lm' )
```

The spherical harmonic coefficients of a given `SHCoeffs` class instance can be converted easily to a different normalization using the `convert()` method. To rotate either the physical body or coordinate system by the three Euler angles `alpha`, `beta`, and `gamma`, one would use the method `rotate()`. The value of the function at a specified latitude and longitude can be determined using the method `expand()` (expansions on grids will be discussed in the following section). Finally, if you ever need to find out the normalization conventions used with a specific class instance, the `info()` method will print an info string that summarizes all the class attributes. Typical uses of these methods might include the following:

```
clm_ortho = clm.convert(normalization= 'ortho,' csphase = -1)
clm_rot = clm.rotate(alpha, beta, gamma)
value = clm.expand(lat=[ 10.] , lon=[ 275.] )
```

The spherical harmonic coefficients in an `SHCoeffs` class instance can be edited and output to either an array or file. The method `set_coeffs()` allows one to modify the value of one or more degrees and orders by providing a list of values for the degrees `ls` and orders `ms`. To obtain a *numpy* array of the coefficients, with the option of using a different normalization, one would use the method `to_array()`. Finally, the method `to_file()` outputs the coefficients either to a *numpy* formatted file or to an `shtools` ascii-formatted list of degrees, orders, and cosine and sine coefficients. The following example lines of code show how to set the degree 2 coefficients to zero and to output the spherical-harmonic coefficients to an array and file:

```
clm.set_coeffs(values = 0, ls = 2, ms =[ -2, -1, 0, 1, 2] )
coeffs = clm.to_array()
clm.to_file(filename, lmax = 40)
```

It is further noted that one can apply arithmetic operations to `SHCoeffs` class instances, including addition, subtraction, multiplication, division, and powers.

6.2. `SHGrid` Class

The class `SHGrid` is the counterpart to `SHCoeffs`. Each class instance contains as attributes all information about the gridded data that are necessary for computing its associated spherical harmonic coefficients. This includes the grid type, the number of latitude and longitude samples, and whether the data type is real or complex. The grid type is specified by the attribute `grid` which can take values of `GLQ` for Gauss-Legendre quadrature grids and `DH` for grids that conform to the Driscoll and Healy (1994) sampling

theorem. The shape of the input Driscoll and Healy grids determines whether they are $N \times N$ (with attribute `sampling = 1`) or $N \times 2N$ (with `sampling = 2`).

Two methods exist for initializing an `SHGrid` class instance using preexisting arrays. The method `from_array()` initializes the class instance using a *numpy* array, whereas the method `from_file()` uses a grid stored in either an ascii-formatted or binary npy file. These two examples demonstrate the use of typical optional parameters employed during initialization:

```
grid = pyshtools.from_array(array, grid= 'GLQ' )
grid = pyshtools.from_file(filename, binary= True)
```

The latitude and longitude coordinates for each row and column of the gridded data can be obtained using the `lats()` and `lons()` methods. If the raw gridded data ever need to be accessed, they are stored in the data attribute.

In addition to initializing an `SHGrid` class instance with data from an array, one can also initialize the class instance by performing a spherical harmonic reconstruction using a preexisting `SHCoeffs` class instance. In this case, one would make use of the `SHCoeffs` method `expand()`, while specifying `grid` as either `GLQ` for Gauss-Legendre quadrature grids, `DH` for $N \times N$ Driscoll and Healy sampled grids, or `DH2` for $N \times 2N$ Driscoll and Healy sampled grids. The following example creates an `SHGrid` class instance for use with

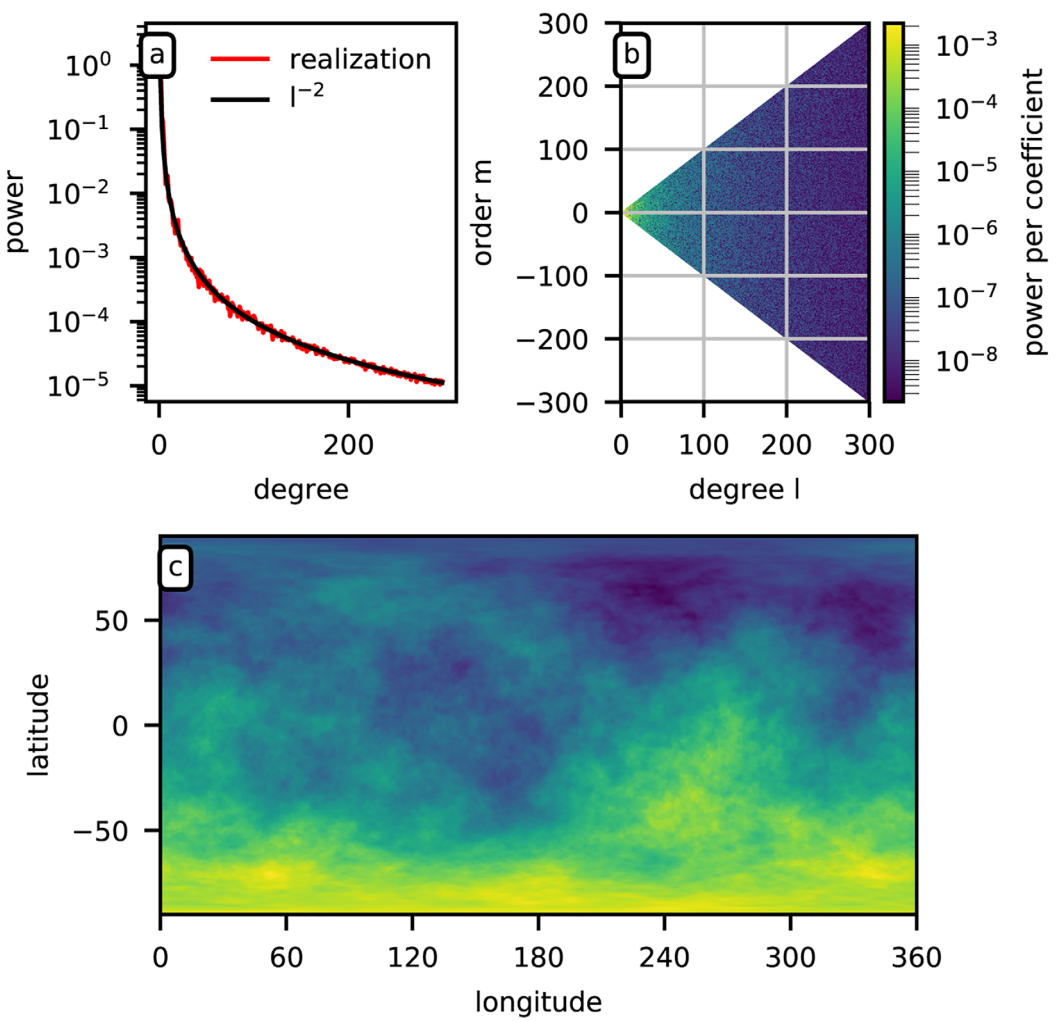

Figure 5. Realization of a random process with an expected power spectrum equal to l^{-2}. (a) The power spectrum of a single realization of the process along with the process expectation, (b) the contribution to the power from each coefficient, and (c) a global map of the realization.

Gauss-Legendre quadrature using only the first 30 spherical harmonic degrees, and then plots the grid using a cylindrical projection with the method `plot()`:

```
grid = clm.expand(grid= 'GLQ,' lmax = 30)
fig, ax = grid.plot()
```

To obtain an `SHCoeffs` class instance containing the spherical harmonic coefficients associated with an `SHGrid` class instance, it is only necessary to use the `expand()` method:

```
clm = grid.expand()
```

In this case, it is not necessary to specify the grid type, as this is already saved in the attributes of the `SHGrid` class instance. Like the `SHCoeffs` class, basic arithmetic operations can be used with `SHGrid` class instances, including addition, subtraction, multiplication, division, and powers.

Figure 5 demonstrates some of the graphical output capabilities of the *pyshtools* package. In this example, the `SHCoeffs` method `from_random()` was used to generate a single realization of a random process whose power spectrum is l^{-2}. The `from_random()` method treats each coefficient as an independent Gaussian random variable, whose variance is the input power at degree l divided by the number of coefficients at this degree. The power spectrum of this individual realization, along with the expected value are plotted in Figure 5a by making use of the method `plot_spectrum()`. The contribution to this power spectrum from each coefficient is plotted in Figure 5b by making use of the method `plot_spectrum2d()`. Finally, the data are expanded on a grid using the method `expand()` and plotted on a global map using the `SHGrid` method `plot()`.

6.3. `SHWindow` Class

The class `SHWindow` provides all methods that are necessary to perform a localized spectral analysis. The theoretical foundations of this analysis technique can be found in Wieczorek and Simons (2005), Simons et al. (2006), and Wieczorek and Simons (2007), and *pyshtools* implements the use of localization windows concentrated within either a spherical cap or an arbitrarily shaped region. The procedure for obtaining a multitaper spectrum estimate is straight forward. After the localization windows are constructed, each is multiplied by the data, the result is expanded in spherical harmonics and the power spectrum is computed. The multitaper spectrum estimate is simply the average of the power spectra associated with each of the localization windows.

`SHWindow` class instances can be initialized either with the `from_cap()` or `from_mask()` methods:

```
win = pyshtools.SHWindow.from_cap(theta = 30., lwin = 30)
win = pyshtools.SHWindow.from_mask(array, lwin = 30, nwin = 10)
```

For spherical cap windows, `theta` specifies the angular radius of the window in degrees and `lwin` specifies the spherical harmonic bandwidth. For arbitrarily shaped windows, instead of providing a value for `theta` one only needs to provide a Driscoll and Healy sampled binary mask as a *numpy* array. With these input parameters, the total number of windows whose fraction of power in the concentration region is greater than α can be determined using the method `number_concentrated(alpha)`. In practice, for a given concentration domain, the analyst chooses `lwin` to obtain the number of desired well-localized windows.

The above initializations compute the spherical harmonic coefficients and concentration factors of all $(\text{lwin}+1)^2$ windows and saves them in the attributes `tapers` and `eigenvalues`, respectively. For spherical cap localization windows, the angular orders of the windows are stored in the attribute `orders`. The poorly localized windows are in general not used, and to compute only a subset of the localization windows, one can specify the number of best-concentrated windows to retain with the optional parameter `nwin`.

When using spherical cap localization windows, the windows are by default centered at the north pole. The `rotate()` method is used to rotate these to an arbitrary latitude and longitude, as specified by `clat` and `clon` in degrees, respectively. Given that most windows will be poorly concentrated and not utilized in a localized spectral analysis, and in order to speed up the calculations, one can choose to rotate only the first `nwinrot` best-concentrated windows, as demonstrated in this example:

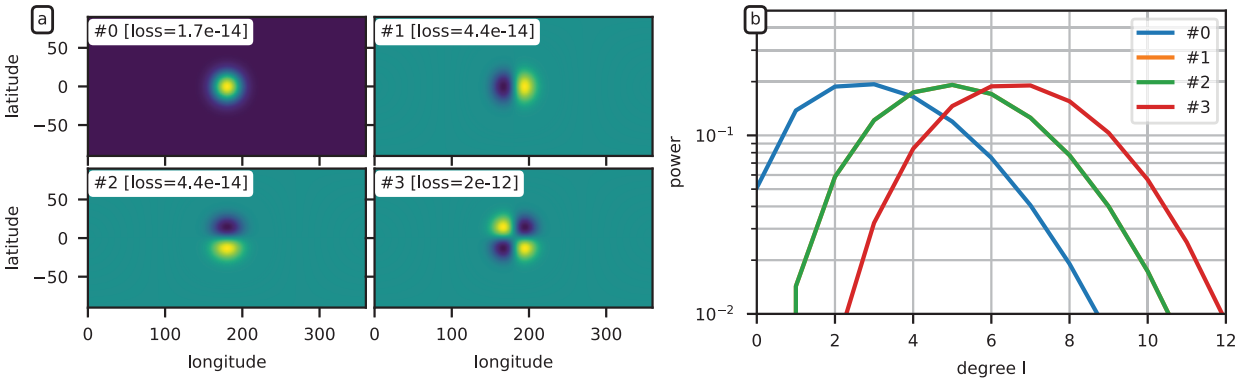

Figure 6. Spherical cap localization windows with `theta` equal to 60° and lwin equal to 18. (a) The four best concentrated windows after rotation to the equator, where *loss* is 1 minus the concentration factor. (b) The corresponding power spectra of the four best concentrated localization windows. Note that tapers 1 and 2 have the same power spectrum.

```
win.rotate(clat = 45., clon = 90., nwinrot = 50)
```

As part of this operation, the rotated coefficients are saved in the attribute `coeffs`. If you ever need to determine the details of how the windows were constructed and rotated, the `info()` method outputs an info string containing the window properties.

The localization windows, their spectra, and their corresponding coupling matrix can be easily visualized using the methods `plot_windows()`, `plot_spectra()`, and `plot_coupling_matrix()`, respectively. The coupling matrix is the transformation matrix between the global input spectrum and the expectation of the localized spectrum. In the following example code, the first four windows and their spectra are plotted, and the coupling matrix is plotted for a data spectral bandwidth of `ldata`,

```
fig1, axes1 = win.plot_windows(4)
fig2, axes2 = win.plot_spectra(4)
fig3, axes3 = win.plot_coupling_matrix(ldata)
```

If a specific window is needed for use elsewhere, it can be output as an `SHCoeffs` or `SHGrid` class instance using the methods `to_shcoeffs()` and `to_shgrid()`, respectively:

```
bestwin_lm = win.to_shcoeffs(0, normalization= 'ortho')
bestwin = win.to_shgrid(0, grid= 'GLQ')
```

As an example of the graphical capabilities of *pyshtools*, we calculate spherical cap localization windows with `theta` equal to 60° and `lwin` equal to 18, and then rotate them to the equator at 180°E longitude. In plot a of Figure 6, the best four concentrated windows are plotted in the space domain using the method `plot_windows()`, and in Figure 6b their corresponding power spectra are plotted using the method `plot_spectra()`.

Once the localization windows are constructed, it is trivial to calculate the multitaper power spectrum estimate and its uncertainty using the method `multitaper_spectrum()`. For this method, one only needs to provide an `SHCoeffs` class instance of the function's spherical harmonic coefficients, the number of tapers `k` to use, and for spherical cap localization windows, the latitude and longitude of the analysis as specified by `clat` and `clon`. When performing a cross-power spectrum analysis, one would use the corresponding method `multitaper_cross_spectrum()`. The following lines of code demonstrated how to obtain the localized power and cross-power spectrum using all tapers with concentration factors greater than 0.99:

```
k = win.number_concentrated(0.99)
mtse, se = win.multitaper_spectrum(clm, k, clat = -55., clon = 90.)
mtse, se = win.multitaper_cross_spectrum(clm, slm, k)
```

where the output parameters are vectors of the multitaper spectrum estimate and standard error, respectively. Optionally, one could provide a vector `taper_wt` that provides the weights when calculating the

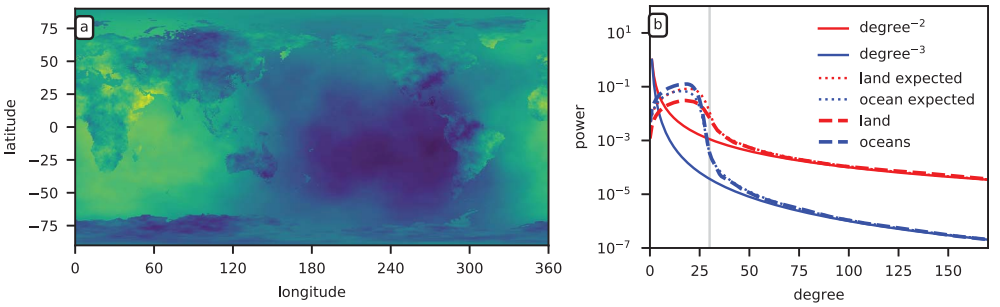

Figure 7. Localized spectral analysis over the land mass and oceans of Earth. (a) A realization of a random process where the power spectrum over the land mass and oceans are l^{-2} and l^{-3}, respectively. (b) The global power spectra of the process over the land and oceans (solid lines), the localized spectra over the land (red dashed line) and oceans (blue dashed line), and the statistical expectation of the two localized power spectra (dotted lines).

multitaper average. In practice, these could be either the eigenvalues of the employed localization windows, or they could be specially constructed to minimize the estimation variance using the function `SHMTVarOpt` that is provided in the `spectralanalysis` subpackage (see Wieczorek & Simons, 2007).

When a function is multiplied by a window, the power spectrum of the localized function is modified. Because of this, when comparing a model with the observed multitaper power spectrum, it is necessary to bias the model in the same way as the data. When the model is statistical in nature and described by a global power spectrum, the biased power spectrum can be obtained using the method `biased_spectrum()`:

```
power_biased = win.biased_spectrum(power, k)
```

It is noted that the method `multitaper_spectrum()` returns a power spectrum whose maximum degree is `ldata-lwin`. This is because localized degrees beyond this limit depend upon coefficients of the global data beyond `ldata`, and these are not known. The method `biased_spectrum()`, in contrast, assumes that the model spectrum `power` is zero beyond its maximum degree, and returns a spectrum with a maximum degree `ldata + lwin`.

An example of two spectral analyses localized over the land mass and oceans of Earth is provided in Figure 7 (a similar example using the lithospheric magnetic field can be found in Beggan et al., 2013). In this example, two realizations of a random process with global power spectra of l^{-2} and l^{-3} were synthesized. The two maps were then masked for the continents and oceans, respectively, and combined as shown in Figure 7a. Localization windows were next constructed that were concentrated over both the land mass and oceans using the method `from_mask()`. With a spectral bandwidth L equal to 30, a total of 83 and 433 windows are obtained that concentrate more than 99% of their power over the land and ocean localization domains, respectively. The localized spectral analysis using these windows is shown in the right plot of Figure 7, where the dashed lines represent the global power spectrum of the process over the land and oceans, the red and blue lines represent the localized power spectrum over the land and oceans, and the dotted lines represent the statistical expectation of the localized spectrum using the method `biased_spectrum()`. As is seen, the localized spectra are very close to their statistical expectations, demonstrating that their is little leakage of signal between the two localization domains. The localized spectra are seen to be biased significantly away from the global spectrum for degrees close to the spectral bandwidth of the localization windows, which is a direct result of the signal from low degrees leaking into the high degrees over the spectral bandwidth of the window. Importantly, this bias is quantifiable, as given by the method `biased_spectrum()`.

7. Development Roadmap

The SHTools software package started development in 2004. It is time and user-tested, and new features have been continuously implemented based on real-life needs of scientists working in the geosciences. Initially started as a Fortran 95 project, full Python capabilities were added in 2015, and the *pyshtools* package is now being actively developed and expanded upon.

In its current state (version 4.2), *pyshtools* supports all basic operations involving spherical harmonic coefficients, grids, and localization windows. In coming releases, four different development activities will be emphasized. First, even though basic plotting routines are provided for inspecting `SHCoeffs`, `SHGrid`, and `SHWindow` class instances, these graphical routines will be improved upon by adding map projection capabilities from the *generic mapping tools (GMT)* package (Wessel & Smith, 1991). A Python implementation of *GMT* is currently being developed based on *GMT* version 6 (http://www.gmtpython.xyz/), and this will allow for the production of publication quality images. Second, high-level classes will be developed for working with both magnetic and gravity field data. This will allow easy access to all of the specialized routines available in SHTools, and would provide for standard geodetic operations. Third, support for ultrahigh spherical-harmonic transforms and reconstructions will be implemented using the technique of extended range arithmetic (Fukushima, 2012; Rexer & Hirt, 2015). Finally, improved input/output routines will be provided for interacting with standard geographical and spherical harmonic coefficient data formats.

Acknowledgments
SHTools is an open-source software project, and we are indebted to all those who have contributed, including Elliott Sales de Andrade, Ilya Oshchepkov, Andrew Walker, Benda X, and xoviat. The most recent release of SHTools can always be downloaded from GitHub (https://github.com/SHTOOLS/SHTOOLS/releases), and the web documentation can be accessed at https://shtools.github.io/SHTOOLS/. This work was supported by CNES.

References

Adams, J. C., & Swarztrauber, P. N. (1999). SPHEREPACK 3.0: A model development facility. *American Meteorological Society*, *127*(8), 1872–1878. https://doi.org/10.1175/1520-0493(1999)127<1872:SAMDF>2.0.CO;2

Beggan, C. D., Saarimäki, J., Whaler, K. A., & Simons, F. J. (2013). Spectral and spatial decomposition of lithospheric magnetic field models using spherical slepian functions. *Geophysical Journal International*, *193*(1), 136–148. https://doi.org/10.1093/gji/ggs122

Beuthe, M. (2008). Thin elastic shells with variable thickness for lithospheric flexure of one-plate planets. *Geophysical Journal International*, *172*(2), 817–841. https://doi.org/10.1111/j.1365-246X.2007.03671.x

Blakely, R. J. (1995). *Potential theory in gravity and magnetic applications*. New York, NY: Cambridge University Press.

Bucha, B., & Janák, J. (2013). A MATLAB-based graphical user interface program for computing functionals of the geopotential up to ultra-high degrees and orders. *Computers & Geosciences*, *56*, 186–196. https://doi.org/10.1016/j.cageo.2013.03.012

Dahlen, F. A., & Tromp, J. (1998). *Theoretical global seismology*. Princeton, NJ: Princeton University Press.

Driscoll, J. R., & Healy, D. M. (1994). Computing Fourier transforms and convolutions on the 2-sphere. *Advances in Applied Mathematics*, *15*(2), 202–250. https://doi.org/10.1006/aama.1994.1008

Frigo, M., & Johnson, S. G. (2005). The design and implementation of FFTW3. *Proceedings of the IEEE*, *93*(2), 216–231. special issue on "Program Generation, Optimization, and Platform Adaptation".

Fukushima, T. (2012). Numerical computation of spherical harmonics of arbitrary degree and order by extending exponent of floating point numbers. *Journal of Geodesy*, *86*(4), 271–285. https://doi.org/10.1007/s00190-011-0519-2

Górski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M., et al. (2005). HEALPix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere. *The Astronomical Journal*, *622*(2), 759–771. https://doi.org/10.1086/427976

Harig, C., Lewis, K. W., Plattner, A., & Simons, F. J. (2015). A suite of software analyzes data on the sphere. *Eos Transactions, American Geophysical Union*, *96*(6), 18–22. https://doi.org/10.1029/2015EO025851

Holmes, S. A., & Featherstone, W. E. (2002). A unified approach to the Clenshaw summation and the recursive computation of very high degree and order normalised associated Legendre functions. *Journal of Geodesy*, *76*(5), 279–299.

Holmes, S. A., & Pavlis, N. K. (2006). *A Fortran program for very-high-degree harmonic synthesis: Harmonic_synthx*. Available at http://earth-info.nga.mil/GandG/wgs84/gravitymod/new_egm/README.txt

Jekeli, C. (2015). Potential theory and static gravity field of the earth. In G. Schubert (Ed.), *Treatise on geophysics* (2nd ed., Vol. 3, pp. 9–35).. Amsterdam, the Netherlands: Elsevier. https://doi.org/10.1016/B978-0-444-53802-4.00056-7

Peltier, W. R. (1974). The impulse response of a Maxwell Earth. *Reviews of Geophysics*, *12*(4), 649–669. https://doi.org/10.1029/RG012i004p00649

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical recipes in FORTRAN: The art of scientific computing* (2nd ed.). Cambridge, UK: Cambridge University Press.

Reinecke, M., & Seljebotn, D. S. (2013). Libsharp: Spherical harmonic transforms revisited. *Astronomy and Astrophysics*, *554*, A112. https://doi.org/10.1051/0004-6361/201321494

Rexer, M., & Hirt, C. (2015). Ultra-high degree surface spherical harmonic analysis using the gauss-legendre and the driscoll/healy quadrature theorem and application to planetary topography models of earth, mars and moon. *Surveys in Geophysics*, *36*(6), 803–830. https://doi.org/10.1007/s10712-015-9345-z

Schaeffer, N. (2013). Efficient spherical harmonic transforms aimed at pseudospectral numerical simulations. *Geochemistry, Geophysics, Geosystems*, *14*, 751–758. https://doi.org/10.1002/ggge.20071

Simons, F. J., Dahlen, F. A., & Wieczorek, M. A. (2006). Spatiospectral concentration on a sphere. *SIAM Review*, *48*(3), 504–536. https://doi.org/10.1137/S0036144504445765

Sneeuw, N. (1994). Global spherical harmonic-analysis by least-squares and numerical quadrature methods in historical perspective. *Geophysical Journal International*, *118*(3), 707–716.

Turcotte, D. L., Willemann, R. J., Haxby, W. F., & Norberry, J. (1981). Role of membrane stresses in the support of planetary topography. *Journal of Geophysical Research*, *86*(B5), 3951–3959. https://doi.org/10.1029/JB086iB05p03951

Varshalovich, D. A., Moskalev, A. N., & Khersonskii, V. K. (1988). *Quantum theory of angular momentum*. Singapore: World Scientific.

Wessel, P., & Smith, W. H. F. (1991). Free software helps map and display data. *Eos Transactions, American Geophysical Union*, *72*(41), 441–446.

Wieczorek, M. A. (2015). Gravity and topography of the terrestrial planets. In T. Spohn & G. Schubert (Eds.), *Treatise on geophysics* (2nd ed., Vol. 10, pp. 153–193). Oxford, UK: Elsevier-Pergamon. https://doi.org/10.1016/B978-0-444-53802-4.00169-X

Wieczorek, M. A., & Simons, F. J. (2005). Localized spectral analysis on the sphere. *Geophysical Journal International*, *162*(3), 655–675. https://doi.org/10.1111/j.1365-246X.2005.02687.x

Wieczorek, M. A., & Simons, F. J. (2007). Minimum-variance multitaper spectral estimation on the sphere. *The Journal of Fourier Analysis and Applications*, *13*(6), 665–692. https://doi.org/10.1007/s00041-006-6904-1