

# Differentiable and accelerated spherical harmonic and Wigner transforms

Matthew A. Price<sup>a,\*</sup>, Jason D. McEwen<sup>a,b,\*</sup>

<sup>a</sup>Mullard Space Science Laboratory, University College London, Dorking, RH5 6NT, UK.

<sup>b</sup>Alan Turing Institute, London, NW1 2DB, UK.

## ARTICLE INFO

### Article history:

Received —

Received in final form —

Accepted —

Available online —

Communicated by —

2000 MSC: 43A90, 44A15, 65T50,  
65D17, 68W01, 85A04, 86A04

## ABSTRACT

Many areas of science and engineering encounter data defined on spherical manifolds. Modelling and analysis of spherical data often necessitates spherical harmonic transforms, at high degrees, and increasingly requires efficient computation of gradients for machine learning or other differentiable programming tasks. We develop novel algorithmic structures for accelerated and differentiable computation of generalised Fourier transforms on the sphere  $\mathbb{S}^2$  and rotation group  $SO(3)$ , *i.e.* spherical harmonic and Wigner transforms, respectively. We present a recursive algorithm for the calculation of Wigner  $d$ -functions that is both stable to high harmonic degrees and extremely parallelisable. By tightly coupling this with separable spherical transforms, we obtain algorithms that exhibit an extremely parallelisable structure that is well-suited for the high throughput computing of modern hardware accelerators (*e.g.* GPUs). We also develop a hybrid automatic and manual differentiation approach so that gradients can be computed efficiently. Our algorithms are implemented within the JAX differentiable programming framework in the S2FFT software code. Numerous samplings of the sphere are supported, including equiangular and HEALPix sampling. Computational errors are at the order of machine precision for spherical samplings that admit a sampling theorem. When benchmarked against alternative C codes we observe up to a 400-fold acceleration. Furthermore, when distributing over multiple GPUs we achieve very close to optimal linear scaling with increasing number of GPUs due to the highly parallelised and balanced nature of our algorithms. Provided access to sufficiently many GPUs our transforms thus exhibit an unprecedented effective linear time complexity.

© 2023 Elsevier Inc. All rights reserved.

## 1. Introduction


Research efforts in many fields of science and engineering, both in industrial and academic settings, are increasingly considering the analysis of data that live on spherical manifolds, or variants thereof. The diversity in applications

\*Correspondence e-mail: {m.price.17, jason.mcewen}@ucl.ac.uk

is remarkable, ranging from quantum chemistry [1, 2], molecular modelling [3, 4], biomedical imaging [5, 6, 7, 8], geophysical inverse problems [9, 10, 11, 12], general relativity and gravitational waves [13, 14, 15], to the wider cosmos [16, 17, 18, 19] and beyond. Reflecting the growth of scientific interest, consortiums within these fields have grown in both size and scope, with international missions including, *e.g.*, ESA’s Planck [20] and Euclid [21] satellite missions, the Dark Energy Survey [DES; 22], the Rubin Observatory Legacy Survey of Space and Time [LSST; 23] and soon the Laser Interferometer Space Antenna [LISA; 24]. As such, a substantial global community of researchers have an interest in the development and use of foundational technology for the analysis and modelling of spherical data.

For many of these applications a harmonic analysis of data can be insightful, in some cases critical. Many physical models are best described in harmonic space (often due to the independent physical evolution of different harmonic modes) necessitating spherical harmonic transforms, for example in cosmology [25] and seismology [26]. Furthermore, many analyses either rely on spherical harmonic transforms for a direct analysis or as an integral component of other analysis techniques, for example wavelets on the sphere [*e.g.* 27, 28, 29] or geometric machine learning on the sphere [*e.g.* 30, 4, 31, 32, 33].

Widespread uses of the harmonic analysis of spherical data, combined with increasingly large data volumes, heightens the need for scalable and efficient algorithms, with readily accessible and easy to use software, to compute generalised fast Fourier transforms on both the sphere  $\mathbb{S}^2$  and rotation group  $\text{SO}(3)$ . Modern hardware accelerators, such as GPUs and TPUs, provide a great opportunity to leverage high throughput computing to address the increasing computational challenge. With the growth of differentiable programming opening up many new types of analysis, many applications also require spherical transforms that are differentiable. Machine learning models on the sphere require differentiable transforms so that the models may be trained by gradient-based optimisation algorithms. Emerging physics-enhanced machine learning approaches [34] also require differentiable physical models, which in many cases themselves require differentiable spherical transforms, for hybrid data-driven and model-based approaches [*e.g.* 35, 36, 37, 38, 39]. While a number of software codes are available to compute spherical harmonic transforms, these do not typically run on hardware accelerators (*e.g.* GPUs) or support differentiable transforms.

In this work we address precisely this need by developing spherical transforms that are both differentiable and accelerated. Specifically, we develop algorithms to compute generalised Fourier transforms on the sphere and also on the rotation group, also referred to as spherical harmonic and Wigner transforms, respectively. Our algorithmic structures are designed to be highly parallelised in order to effectively exploit very high parallel throughput on hardware accelerators like GPUs. We pay careful attention to the computation of gradients through a hybrid automatic and manual differentiation approach, avoiding the memory overhead of full automatic differentiation while also avoiding the complexities of full manual differentiation. We implement these algorithms in a new open-source software code called S2FFT , which is developed in the JAX differentiable programming framework. Our algorithms and code support a number of different approaches to sample the sphere: including various equiangular samplings that admit sampling theorems [40, 41, 42] and thus theoretically exact spherical harmonic transforms; and also the popular HEALPix scheme [43], which, although it does not support exact harmonic transforms, has the practical advantage of pixels of equal area. Various sampling schemes are illustrated in Fig. 1.

The remainder of this article is structured as follows. In Sec. 2 we review the mathematical background that underpins themes throughout this article. Following this, in Sec. 4 we present a recursive algorithm suitable for the parallel calculation of the Wigner  $d$ -functions. Algorithms to compute the spherical harmonic transform for arbitrary spin and the Wigner transform are presented in Sec. 4. In Sec. 5 we present various approaches to the efficient computational of the gradients of our spherical transforms. Penultimately, we provide an overview of our software implementation with benchmarking results in Sec. 6, before drawing concluding remarks in Sec. 7.

## 2. Fourier transforms on the sphere and rotation group

We concisely review the generalised Fourier transform on the two-sphere  $\mathbb{S}^2$  and three-dimensional rotation group  $\text{SO}(3)$ . We first review the scalar spherical harmonic transform, after which we generalise to spin functions. We describe the generalised Fourier transform on the rotation group, the so-called Wigner transform, and draw connections to spin spherical harmonic transforms. Finally, we discuss sampling spherical signals for the computation of spherical transforms, highlighting the distinction between sampling schemes and sampling theorems (the latter of which provide exact quadrature and exact harmonic transforms), and discuss a number of common spherical samplings that our algorithms support.

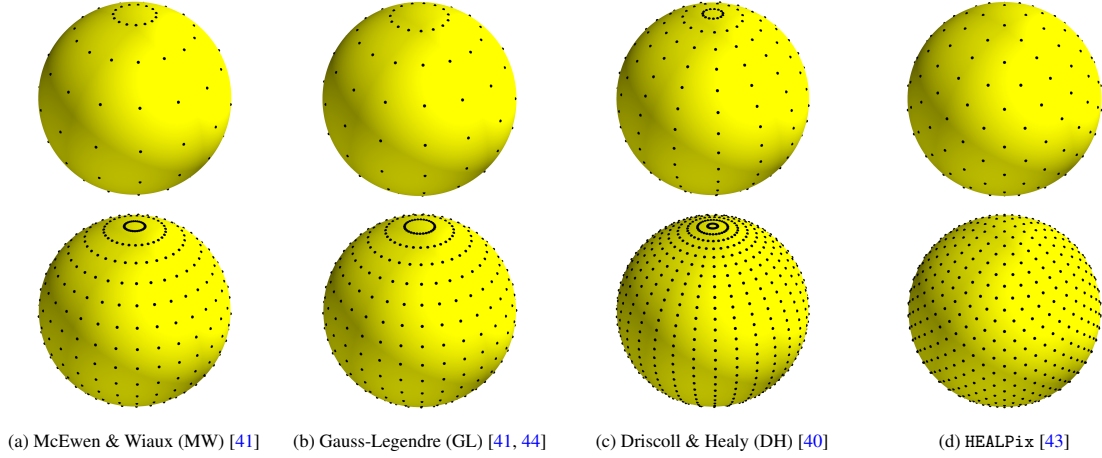


Fig. 1: Visualisation of sample positions on the sphere for various sampling schemes. The Driscoll & Healy [DH; 40], McEwen & Wiaux [MW; 41], and Gauss-Legendre [GL; 41, 44] samplings all admit sampling theorems with exact spherical harmonic transforms, although HEALPix does not. Both MW and GL sampling require  $\sim 2L^2$  samples (although the MW sampling is slightly more efficient), whereas DH [40] sampling requires  $\sim 4L^2$ , and HEALPix considers  $\sim kL^2$  samples, where  $k$  typically ranges from 3 to 12. The HEALPix scheme has the practical advantage of equal-area pixels and is widely adopted in cosmology. Modern computer vision, machine learning and geophysical applications typically adopt equiangular sampling and thus map most efficiently and straightforwardly onto the MW sampling.

### 2.1. Spherical harmonic transforms for scalar functions

The scalar spherical harmonic functions  $Y_{\ell m} : \mathbb{S}^2 \rightarrow \mathbb{C}$  form the canonical orthogonal basis for the space of square integrable scalar functions on the sphere and are defined by

$$Y_{\ell m}(\vartheta, \varphi) = \sqrt{\frac{2\ell + 1}{4\pi} \frac{(\ell - m)!}{(\ell + m)!}} P_{\ell}^m(\cos \vartheta) e^{im\varphi}, \quad (1)$$

for spherical coordinates with colatitude  $\vartheta \in [0, \pi]$  and longitude  $\varphi \in [0, 2\pi)$ , natural order  $\ell \in \mathbb{N}$  and integer degree  $m \in \mathbb{Z}$ ,  $|m| \leq \ell$ , and where  $P_{\ell}^m(\cdot)$  are the associated Legendre functions. Throughout this article we adopt the Condon-Shortley phase convention such that the conjugate symmetry relation  $Y_{\ell m}^*(\vartheta, \varphi) = (-1)^m Y_{\ell, -m}(\vartheta, \varphi)$  holds. The orthogonality and completeness relations for the spherical harmonics read

$$\langle Y_{\ell m}, Y_{\ell' m'} \rangle = \int_{\mathbb{S}^2} d\mu(\vartheta, \varphi) Y_{\ell m}(\vartheta, \varphi) Y_{\ell' m'}^*(\vartheta, \varphi) = \delta_{\ell \ell'} \delta_{m m'} \quad (2)$$

and

$$\sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} Y_{\ell m}(\vartheta, \varphi) Y_{\ell m}^*(\vartheta', \varphi') = (\sin \vartheta)^{-1} \delta(\vartheta - \vartheta') \delta(\varphi - \varphi'), \quad (3)$$

respectively, where  $\langle \cdot, \cdot \rangle$  denotes the inner product,  $d\mu(\vartheta, \varphi) = \sin \vartheta d\vartheta d\varphi$  is the rotationally invariant Haar measure on the sphere,  $\delta_{ij}$  is the Kronecker delta symbol and  $\delta(\cdot)$  is the Dirac delta function.

By the orthogonality and completeness of the spherical harmonic functions, any square integrable function  $f \in L^2(\mathbb{S}^2)$  may be represented by its spherical harmonic expansion

$$f(\vartheta, \varphi) = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} \hat{f}_{\ell m} Y_{\ell m}(\vartheta, \varphi), \quad (4)$$

where the square summable spherical harmonic coefficients  $\hat{f} \in l^2$  are given by the projection

$$\hat{f}_{\ell m} = \langle f, Y_{\ell m} \rangle = \int_{\mathbb{S}^2} d\mu(\vartheta, \varphi) f(\vartheta, \varphi) Y_{\ell m}^*(\vartheta, \varphi). \quad (5)$$

We denote the forward spherical harmonic transform by  $\mathcal{F} : L^2(\mathbb{S}^2) \rightarrow l^2$ , which maps functions on the sphere to their spherical harmonic coefficients  $\mathcal{F} : f \mapsto \hat{f}$ . Similarly, we denote the inverse spherical harmonic transform by  $\mathcal{F}^{-1} : l^2 \rightarrow L^2(\mathbb{S}^2)$ , which maps spherical harmonic coefficients to the corresponding function on the sphere  $\mathcal{F}^{-1} : \hat{f} \mapsto f$ . For real signals, note that the Hermitian symmetry relation  $\hat{f}_{\ell m}^* = (-1)^m \hat{f}_{\ell, -m}$  holds.

## 2.2. Spherical harmonic transforms for spin functions

The (scalar) spherical harmonic functions are sufficient to characterise the harmonic representation of scalar functions on the sphere, however they do not fully capture the symmetries associated with spin functions. Square integrable spin- $s$  functions on the sphere  ${}_sf \in L^2(\mathbb{S}^2)$  exhibit an additional symmetry such that they transform by  ${}_sf' = e^{-s\chi} {}_sf$  under right-handed rotations  $\chi$  in the local tangent plane. The spin spherical harmonics  ${}_sY_{\ell m} : \mathbb{S}^2 \rightarrow \mathbb{C}$  form the canonical orthogonal basis for  $L^2(\mathbb{S}^2)$  spin- $s$  functions on the sphere.

Spin spherical harmonics may be defined through actions of spin raising operator  $\tilde{\partial}$  and lowering operator  $\tilde{\partial}^\dagger$  on the scalar spherical harmonics, which are given by [45, 46, 47]

$$\tilde{\partial} = -\sin^s \vartheta \left( \partial_\vartheta + \frac{i\partial_\varphi}{\sin \vartheta} \right) \sin^{-s} \vartheta \quad \text{and} \quad \tilde{\partial}^\dagger = -\sin^{-s} \vartheta \left( \partial_\vartheta - \frac{i\partial_\varphi}{\sin \vartheta} \right) \sin^s \vartheta. \quad (6)$$

The action of the spin raising and lowering operators on the spin spherical harmonics  ${}_sY_{\ell m}$  is given, respectively, by

$$\tilde{\partial} {}_sY_{\ell m} = \sqrt{(\ell-s)(\ell+s+1)} {}_{s+1}Y_{\ell m} \quad \text{and} \quad \tilde{\partial}^\dagger {}_sY_{\ell m} = -\sqrt{(\ell+s)(\ell-s+1)} {}_{s-1}Y_{\ell m}. \quad (7)$$

It therefore follows that any spin spherical harmonic basis function can be generated from  $s$  repeated applications of the spin raising or lowering operators on the scalar (spin  $s = 0$ ) spherical harmonics:

$${}_sY_{\ell m} = \begin{cases} (-1)^s \sqrt{\frac{(\ell+s)!}{(\ell-s)!}} \tilde{\partial}^{-s} Y_{\ell m} & \text{for } -\ell \leq s \leq 0 \\ \sqrt{\frac{(\ell-s)!}{(\ell+s)!}} \tilde{\partial}^s Y_{\ell m} & \text{for } 0 \leq s \leq \ell \end{cases}. \quad (8)$$

The spin spherical harmonics, for a given spin, satisfy identical orthogonality and completeness relations as the scalar spherical harmonics, *cf.* Eq. 2 and Eq. 3, respectively. Consequently, the forward and inverse spin spherical harmonic transforms are also similar to the scalar transforms, *cf.* Eq. 5 and Eq. 4 respectively, but with the scalar spherical harmonics replaced by the spin spherical harmonics. We denote forward and inverse spin spherical harmonic transforms by  ${}_s\mathcal{F}$  and  ${}_s\mathcal{F}^{-1}$ , respectively, often dropping the explicit reference to spin  $s$  for notational brevity. The Hermitian symmetry relation of the spin spherical harmonic coefficients is given by  ${}_sf_{\ell m}^* = (-1)^{s+m} {}_{-s}f_{\ell, -m}$  for a function satisfying  ${}_sf^* = {}_{-s}f$  (which for a spin  $s = 0$  function equates to the usual reality condition).

## 2.3. Wigner transforms for functions on the rotation group

In many settings we encounter functions defined over the space of three-dimensional rotations. The Wigner  $D$ -functions  $D_{mn}^\ell : \text{SO}(3) \rightarrow \mathbb{C}$  form an irreducible unitary representation of  $\text{SO}(3)$ , the group of three-dimensional rotations, for natural  $\ell \in \mathbb{N}$  and integers  $m, n \in \mathbb{Z}$  such that  $|m|, |n| \leq \ell$  [48]. The Wigner  $D$ -functions are related to the spin spherical harmonics by

$$D_{mn}^\ell(\alpha, \beta, \gamma) = (-1)^n \sqrt{\frac{4\pi}{2\ell+1}} {}_{-n}Y_{\ell m}^*(\beta, \alpha) e^{-iny}, \quad (9)$$

where  $(\alpha, \beta, \gamma)$  are the Euler angles parameterising  $\text{SO}(3)$ , with  $\alpha \in [0, 2\pi)$ ,  $\beta \in [0, \pi]$  and  $\gamma \in [0, 2\pi)$ . We adopt the  $zyz$  Euler convention corresponding to the rotation of a physical body in a *fixed* co-ordinate system about the  $z$ ,  $y$  and  $z$  axes by  $\gamma$ ,  $\beta$  and  $\alpha$  respectively. The Wigner  $D$ -functions may be decomposed in terms of the real Wigner  $d$ -functions by [48]

$$D_{mn}^\ell(\alpha, \beta, \gamma) = e^{-im\alpha} d_{mn}^\ell(\beta) e^{-iny}. \quad (10)$$

The orthogonality and completeness relations for the Wigner  $D$ -functions read, respectively,

$$\langle D_{mn}^\ell, D_{m'n'}^{\ell'} \rangle = \int_{\text{SO}(3)} d\mu(\alpha, \beta, \gamma) D_{mn}^\ell(\alpha, \beta, \gamma) D_{m'n'}^{\ell'*}(\alpha, \beta, \gamma) = \frac{8\pi^2}{2\ell+1} \delta_{\ell\ell'} \delta_{mm'} \delta_{nn'}, \quad (11)$$

and

$$\sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} \sum_{n=-\ell}^{\ell} D_{mn}^\ell(\alpha, \beta, \gamma) D_{mn}^{\ell'*}(\alpha', \beta', \gamma') = \delta(\alpha - \alpha') \delta(\cos\beta - \cos\beta') \delta(\gamma - \gamma'), \quad (12)$$

where we overload the notation  $\langle \cdot, \cdot \rangle$  to denote inner products on both the sphere and rotation group (note that the case considered can be easily inferred from the context) and  $d\mu(\alpha, \beta, \gamma) = \sin\beta d\alpha d\beta d\gamma$  is the rotationally invariant Haar measure on the rotation group.

The space of square integrable functions  $f \in L^2(\text{SO}(3))$  admits the Wigner expansion

$$f(\alpha, \beta, \gamma) = \sum_{\ell=0}^{\infty} \frac{2\ell+1}{8\pi^2} \sum_{m=-\ell}^{\ell} \sum_{n=-\ell}^{\ell} \hat{f}_{mn}^{\ell} D_{mn}^{\ell*}(\alpha, \beta, \gamma), \quad (13)$$

where the square summable Wigner coefficients  $\hat{f} \in l^2$  are given by the projection

$$\hat{f}_{mn}^{\ell} = \langle f, D_{mn}^{\ell*} \rangle = \int_{\text{SO}(3)} d\mu(\alpha, \beta, \gamma) f(\alpha, \beta, \gamma) D_{mn}^{\ell}(\alpha, \beta, \gamma). \quad (14)$$

As before, we introduce an operator notation with forward Wigner transform  $\mathcal{W} : L^2(\text{SO}(3)) \rightarrow l^2$ , which maps functions on the rotation group to their harmonic coefficients  $\mathcal{W} : f \mapsto \hat{f}$ . Similarly, we denote the inverse Wigner transform by  $\mathcal{W}^{-1} : l^2 \rightarrow L^2(\text{SO}(3))$ , which maps Wigner harmonic coefficients to the corresponding function on the rotation group  $\mathcal{W}^{-1} : \hat{f} \mapsto f$ . Note that we use  $f$  to denote functions on both the sphere and rotation group, with corresponding harmonic coefficients  $\hat{f}$ . For real signals, note that the Hermitian symmetry relation  $\hat{f}_{mn}^{\ell*} = (-1)^{m+n} \hat{f}_{-m, -n}^{\ell}$  holds.

Due to the relationship between the Wigner  $D$ -functions and the spin spherical harmonics (Eq. 9), Wigner transforms can be expressed as a series of spin spherical harmonic transforms and Fourier transforms [42]. For completeness, the forward Wigner transform may be expressed as

$$\hat{f}_{mn}^{\ell} = (-1)^n \sqrt{\frac{4\pi}{2\ell+1}} \int_{\mathbb{S}^2} d\mu(\beta, \alpha) f_n(\alpha, \beta) {}_{-n}Y_{\ell m}^*(\beta, \alpha), \quad (15)$$

where

$$f_n(\alpha, \beta) = \int_0^{2\pi} d\gamma f(\alpha, \beta, \gamma) e^{-in\gamma}. \quad (16)$$

Note that Eq. 15 corresponds to a (scaled) spin spherical harmonic transform with spin  $-n$ . The inverse Wigner transform may be expressed as

$$f(\alpha, \beta, \gamma) = \sum_{n=-\ell}^{\ell} f_n(\alpha, \beta) e^{in\gamma}, \quad (17)$$

where

$$f_n(\alpha, \beta) = (-1)^n \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} \sqrt{\frac{2\ell+1}{16\pi^3}} \hat{f}_{mn}^{\ell} {}_{-n}Y_{\ell m}(\beta, \alpha). \quad (18)$$

Note that Eq. 18 corresponds to an inverse spin spherical harmonic transform (of scaled harmonic coefficients) with spin  $-n$ . These expressions may be leveraged to compute Wigner transforms simply through spin spherical harmonic transforms and standard 1D Fourier transforms [42].

#### 2.4. Sampling schemes versus sampling theorems on the sphere and rotation group

Up until this point we have considered the continuous form of generalised Fourier transforms on the sphere and rotation group. In practice, however, one recovers sampled values of the signal of interest at a finite number of positions. Consequently, sampling schemes must be adopted to pixelise the sphere and rotation group. Generalised Fourier transforms must then be computed from these sampled values.

Often we consider the space of bandlimited signals  $\mathcal{B}_L(\Omega)$ , where  $\Omega$  represents either the sphere or rotation group, i.e.  $\Omega = \{\mathbb{S}^2, \text{SO}(3)\}$ . Bandlimited signals are those whose harmonic coefficients are zero for degrees  $\ell$  greater than or equal to the bandlimit  $L$ . A bandlimited signal on the sphere has  $\hat{f}_{\ell m} = 0$  for all  $\ell \geq L$ , while a bandlimited signal on the rotation group has  $\hat{f}_{mn}^{\ell} = 0$  for all  $\ell \geq L$ . Consequently, harmonic coefficients for bandlimited signals on the sphere and rotation group live, respectively, in  $\mathbb{C}^{L^2}$  and  $\mathbb{C}^{(4L^3-L)/3}$  [42, 49]. For bandlimited signals, summations over degree  $\ell$  (cf. Eq. 4 and Eq. 13) may be truncated to  $L-1$ . In many settings azimuthal bandlimits also arise, such that harmonic coefficients are zero for  $m \geq M$  and/or  $n \geq N$  (summations over  $m$  and  $n$  may then also be truncated). Real-world signals may be approximated accurately by bandlimited signals for a suitable bandlimit.

For the spatial signal representation, we distinguish between sampling *schemes* and *theorems*. For *sampling schemes*, the sphere or rotation group may be sampled in any arbitrary manner and harmonic coefficients computed

by approximate quadrature. This provides a great deal of flexibility in how sample positions are set. However, this flexibility comes at a cost: for an arbitrary sampling, harmonic transforms can only be computed approximately and in some cases can be quite inaccurate. Furthermore, for an arbitrary sampling the computation of the spherical harmonic transform scales as  $O(L^4)$ . By adopting an isolatitude sampling, where  $\varphi$  samples are collected in isolatitudinal rings for each colatitude  $\vartheta$ , one may apply a separation of variables to reduce computational complexity to  $O(L^3)$  [e.g. 43, 41, 42, 50]. For this reason isolatitudinal sampling is typically adopted; hence, we focus on isolatitudinal sampling in the remainder of this article. A *sampling theorem*, in contrast, provides a way to capture all of the information content of a bandlimited signal in a finite set of sampled values. Since all information content of the underlying continuous signal is captured, a sampling theorem implicitly provides a quadrature rule to compute harmonic coefficients exactly. Sampling theorems are thus tied to particular structured samplings of the sphere.

For both arbitrary sampling schemes and sampling theorems, the forward spin spherical and Wigner transforms of Eq. 5 and Eq. 14, respectively, may be discretised by

$${}_sf_{\ell m} \simeq \sum_{\theta, \phi} q(\theta, \phi) {}_sf(\theta, \phi) {}_sY_{\ell m}^*(\theta, \phi) \quad (19)$$

and

$$\hat{f}_{mn}^\ell \simeq \sum_{\alpha, \beta, \gamma} q(\alpha, \beta, \gamma) f(\alpha, \beta, \gamma) D_{mn}^\ell(\alpha, \beta, \gamma), \quad (20)$$

where we have introduced an upright notation to denote sampled angles, *i.e.*  $(\vartheta, \varphi) \mapsto (\theta, \phi)$  and  $(\alpha, \beta, \gamma) \mapsto (\alpha, \beta, \gamma)$ . We adopt the symbol  $\simeq$  to denote that the computation is approximate for arbitrary sampling schemes but is exact for sampling theorems. Inverse spherical and Wigner transforms of Eq. 4 and Eq. 13, respectively, involve summations only and so can be computed directly (up to a particular degree  $\ell$ ).

We again adopt an operator notation overloading the previous notation to now consider bandlimited signals with forward spin spherical harmonic transform  $\mathcal{F} : \mathcal{B}_L(\mathbb{S}^2) \rightarrow \mathbb{C}^{L^2}$  and inverse  $\mathcal{F}^{-1} : \mathbb{C}^{L^2} \rightarrow \mathcal{B}_L(\mathbb{S}^2)$ . Similarly, the forward Wigner transform is denoted  $\mathcal{W} : \mathcal{B}_L(\text{SO}(3)) \rightarrow \mathbb{C}^{(4L^3-L)/3}$ , with inverse  $\mathcal{W}^{-1} : \mathbb{C}^{(4L^3-L)/3} \rightarrow \mathcal{B}_L(\text{SO}(3))$ .

### 2.5. Specific isolatitudinal sampling schemes and theorems on the sphere and rotation group

To complete this section, let us briefly discuss the sampling schemes and theorems considered throughout this article and in the associated software implementation. One of the most common spherical sampling schemes is the Hierarchical Equal Area isoLatitude Pixelisation of the sphere, abbreviated HEALPix [43].<sup>1</sup> HEALPix has numerous practical advantages, such as a hierarchical representation and equal area pixels. Consequently, it is widely used in many application domains [e.g. 20]. However, since HEALPix does not exhibit a sampling theorem, spherical harmonic transforms are necessarily approximate. In fact, harmonic transforms can be quite inaccurate with absolute errors of  $\sim 10^{-1}$  in some settings [51]. For this reason, iterations are usually performed to increase accuracy.

The canonical sampling theorems on the sphere for equiangular samplings are those of Driscoll & Healy [DH; 40] and McEwen & Wiaux [MW; 41]. Spherical sampling theorems differ significantly to the Euclidean setting. The Nyquist-Shannon sampling theorem for Euclidean spaces results in an identical number of samples in both the spatial and harmonic domains. In contrast, no sampling theorem on the sphere achieves the optimal dimensionality of bandlimited spherical signals in harmonic space, given by  $L^2$ . The MW sampling theorem provides the most efficient equiangular sampling, with  $\sim 2L^2$  spatial samples, reducing the Nyquist rate on the sphere by a factor of two compared to the DH sampling scheme, which requires  $\sim 4L^2$  spatial samples.<sup>2</sup> Fast algorithms for both the DH and MW sampling theorems have been developed. While a number of algorithms have been introduced for the DH sampling theorem [40, 53], some of which have computational complexity of  $O(L^2 \log L)$ , the only variant is that is universally stable is the so-called semi-naïve algorithm, with complexity of  $O(L^3)$ . Fast algorithms for the MW sampling theorem also achieve complexity of  $O(L^3)$  [41]. While equiangular samplings are the most common structured sampling, for example matching the acquisition format of 360 panoramic photos and videos, spherical sampling theorems based on Gauss-Legendre (GL) quadrature have also been proposed [e.g. 54, 41], with samples

<sup>1</sup><https://healpix.sourceforge.io/>

<sup>2</sup>Note that an equiangular sampling *scheme* that achieves the optimal number of spatial samples has been developed [52]. A sampling theorem with exact quadrature is not recovered but high accuracy is nevertheless obtained.

defined on isolatitudinal rings with  $\theta$  specified by the roots of the Legendre polynomials of order  $L$ . GL sampling exhibits a similar, although slightly greater, number of samples as the MW sampling theorem.

Both the DH and MW sampling theorems have been extended to the rotation group  $\text{SO}(3)$  [55, 42]. Both exhibit fast algorithms with complexity  $\mathcal{O}(L^4)$ , while the MW sampling theorem again achieves a reduction in the Nyquist rate on  $\text{SO}(3)$  by a factor of two, requiring  $\sim 4L^3$  spatial samples [42] rather than  $\sim 8L^3$  samples [55]. Note that in many practical settings low azimuthal bandlimits arise, reducing a factor of  $L$  in both number of samples and complexity to  $N \ll L$ . The Gauss-Legendre sampling theorem has also been extended to  $\text{SO}(3)$  [49], requiring a similar (although slightly larger) number of samples as the MW sampling theorem on  $\text{SO}(3)$ .

Throughout the remainder of this article we develop general spherical transforms that can be applied to any isolatitudinal sampling that exhibits an explicit quadrature (either exact or approximate). In particular, our software implementation supports the HEALPix sampling scheme, due to its widespread use and practical advantages, and the DH and MW sampling theorems, due to the exact spherical transforms afforded.

### 3. Parallelised and differentiable Wigner computation

To compute spin spherical harmonic and Wigner transforms it is necessary to compute the Wigner  $D$ -functions, or equivalently the real Wigner  $d$ -functions (Eq. 10). Since we target accelerated computation on modern hardware accelerators, such as GPUs and TPUs that can execute a very large number of threads in parallel, it is desirable to compute Wigner functions in a highly parallelised manner. Furthermore, since we also target differentiable transforms, we must compute Wigner functions in a manner that lends itself to efficient automatic differentiation. Finally, we are interested in use cases that reach very high harmonic degrees  $\ell$ , hence careful attention must be given to ensure stable computation up to very high  $\ell$ . In this section we first define explicit expressions for the Wigner functions, before discussing in greater detail the specific requirements that must be met to achieve a high degree of parallel computation and efficient automatic differentiation. We then present a recursive algorithm to compute Wigner functions that meets these requirements, discussing specific considerations to ensure our approach is suitable for differentiable programming frameworks such as JAX [56].

#### 3.1. Explicit Wigner representations

We relate the Wigner  $D$ -functions to the spin spherical harmonics above (Eq. 9), and also to the Wigner  $d$ -functions (Eq. 10). However, we of course require an explicit representation for computational purposes. We leverage the decomposition of the Wigner  $D$ -functions in terms of the  $d$ -functions and thus focus on the computation of  $d_{mn}^\ell(\beta)$ .

A number of explicit representations of the  $d$ -functions can be considered, such as the differential representation [48, Sec. 4.3.2, p. 77, Eq. 7]

$$d_{mn}^\ell(\beta) = \frac{(-1)^{\ell-n}}{2^\ell} \sqrt{\frac{(\ell+m)!}{(\ell-m)!(\ell+n)!(\ell-n)!}} (1 - \cos\beta)^{-\frac{(m-n)}{2}} (1 + \cos\beta)^{-\frac{(m+n)}{2}} \frac{d^{\ell-m}}{(d \cos\beta)^{\ell-m}} [(1 - \cos\beta)^{\ell-n} (1 + \cos\beta)^{\ell+n}], \quad (21)$$

or the expression in terms of Jacobi polynomials  $P_s^{\mu,\nu}(\cdot)$  given by [48, Sec. 4.3.4, p. 78, Eq. 13–15]

$$d_{mn}^\ell(\beta) = \zeta_{mn} \sqrt{\frac{s!(s+\mu+\nu)!}{(s+\mu)!(s-\nu)!}} \left(\sin \frac{\beta}{2}\right)^\mu \left(\cos \frac{\beta}{2}\right)^\nu P_s^{\mu,\nu}(\cos\beta), \quad (22)$$

where  $\mu = |m - n|$ ,  $\nu = |m + n|$ ,  $s = \ell - (\mu + \nu)/2$  and

$$\zeta_{mn} = \begin{cases} 1 & \text{if } n \geq m \\ (-1)^{n-m} & \text{if } n < m \end{cases}. \quad (23)$$

However, these expressions do not lend themselves to convenient computation (indeed the Jacobi representation simply shifts the task to the computation of the Jacobi polynomials). Instead, it is typical to compute Wigner  $d$ -functions by recursion.

### 3.2. Requirements for parallel computation and automatic differentiation

While the Wigner  $d$ -functions can be computed by recursion, our aim is to deploy our algorithms efficiently on hardware accelerators that provide very high levels of parallelisation and to ensure they are stable to very high harmonic degrees  $\ell$ . Careful attention therefore must be given to the recursion used to ensure these requirements are met.

A number of recursions, while stable, recurse over both harmonic degree  $\ell$  and order  $m$  [e.g. 57, 58]. This limits the degree of parallelisation that can be achieved. Three-term recurrences typically recurse over a single index only, which is well-suited for high levels of parallelisation, although they can be unstable with errors compounding and resulting in highly inaccurate calculation [59]. Nevertheless, careful attention can be given to ensure stability can be achieved [e.g. 60, 61].

For spin spherical harmonic transforms we need to compute  $d_{mn}^\ell(\beta)$  for all  $\ell$ ,  $m$ , and  $\beta$ , but only for a single  $n$  (the spin number of interest). We can compute Wigner transforms in terms of a series of spin spherical harmonic transforms (as discussed in Sec. 2.3). We therefore target a recursion over  $m$  only that we can compute for arbitrary  $\ell$ ,  $n$ , and  $\beta$ . Such a recursion supports high degrees of parallelisation, where computations for  $\ell$ ,  $n$ , and  $\beta$  are independent and so can be parallelised across the very large number of threads available on modern hardware accelerators, such as GPUs and TPUs. Furthermore, this avoids the need for any recursion to compute transforms for all spins  $\leq n$ , providing an efficiency saving over alternative approaches that often compute spin transforms through  $n$  repeated application of spin lowering/raising operations. We pay careful attention in the next section (Sec. 3.3) to how such a recursion can be computed in a stable manner up to very high degrees  $\ell$ .

Another key aim of our approach is to provide differentiable transforms. A final requirement is that our recursive algorithm must therefore be structured in such a way as to provide computationally efficient automatic differentiation. We pay careful attention below (Sec. 3.4) to how the recursion we consider can be computed efficiently and stably in a differentiable programming framework such as JAX. Further details regarding automatic differentiation of not only the computation of the Wigner functions but the full spherical transforms are discussed in Sec. 5.

### 3.3. Recursion

Our algorithm is built upon the three-term Wigner  $d$ -function recursion [48, Sec. 4.8.3, p. 93, Eq. 17]

$$\frac{n - m \cos \beta}{\sin \beta} d_{mn}^\ell(\beta) = \frac{1}{2} \sqrt{(\ell + m)(\ell - m + 1)} d_{m-1,n}^\ell(\beta) + \frac{1}{2} \sqrt{(\ell - m)(\ell + m + 1)} d_{m+1,n}^\ell(\beta). \quad (24)$$

This recursion iterates across order  $m$  independently of  $\ell$ ,  $n$ , and  $\beta$ , thereby supporting an algorithmic structure that can be highly parallelised. While computing this recursion for increasing  $m$  is unstable, it can be computed stably for decreasing  $m$  (as discussed in detail in Prézeau and Reinecke [60], although they consider recursion over  $n$ ). One may straightforwardly rearrange Eq. 24 to recover

$$d_{m-1,n}^\ell(\beta) = \lambda_m a_{m-1} d_{mn}^\ell(\beta) - \frac{a_{m-1}}{a_m} d_{m+1,n}^\ell(\beta), \quad (25)$$

where, for notational brevity, the recursion coefficients are concisely refactored into

$$\lambda_m = \frac{n - m \cos \beta}{\sin \beta} \quad \text{and} \quad a_m = \frac{2}{\sqrt{(\ell - m)(\ell + m + 1)}}. \quad (26)$$

The recursion needs to be initialised at  $d_{\ell n}^\ell(\beta)$  so that we can recurse down in  $m$ , starting from  $m = \ell$ . We therefore initialise the algorithm using the relation

$$d_{\ell n}^\ell(\beta) = \sqrt{\frac{(2\ell)!}{(\ell + n)!(\ell - n)!}} \left( -\sin \frac{\beta}{2} \right)^{\ell-n} \left( \cos \frac{\beta}{2} \right)^{\ell+n}, \quad (27)$$

which follows by the expression for the Wigner  $d$ -functions in terms of Jacobi polynomials of Eq. 22. To sketch how one reaches this result, consider the case in which the spin number  $n = \ell$ , the Jacobi polynomial term then reduces to unity, and the symmetry relation  $d_{mn}^\ell(\beta) = (-1)^{n-m} d_{nm}^\ell(\beta)$  [48, Sec. 4.4, p. 79, Eq. 1] can then be applied to reorder the  $m$  and  $n$  indices, following which the trigonometric expressions and prefactor follow straightforwardly. We calculate

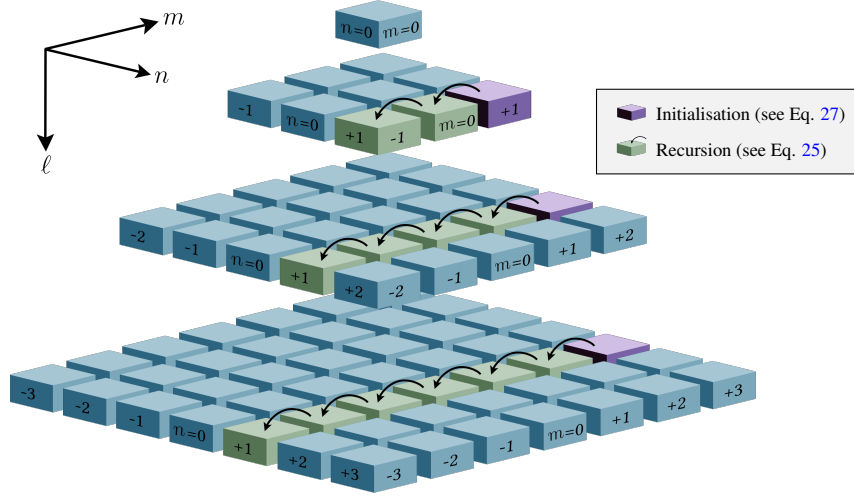


Fig. 2: Overview of the recursive strategy that we adopt to calculate the Wigner  $d$ -functions  $d_{mn}^\ell(\beta)$ , which underpin both the spin spherical harmonic and Wigner transforms. The recursion iterates across order  $m$  independently of  $\ell$ ,  $n$ , and  $\beta$ , in the direction of reducing  $m$  (to ensure stability). Such a recursion supports high degrees of parallelisation, where computations for  $\ell$ ,  $n$ , and  $\beta$  are independent and so can be parallelised across the very large number of threads available on modern hardware accelerators, such as GPUs and TPUs. Furthermore, this approach avoids the need for any recursion to compute transforms for different spins  $n$ , providing an efficiency saving over alternative approaches that often compute spin transforms through repeated application of spin lowering/raising operations. The algorithm is initialised for  $d_{\ell n}^\ell(\beta)$ . We also consider a non-conditional renormalisation approach to avoid under- and over-flow in a manner that is applicable to differentiable programming frameworks such as JAX. The resulting recursive algorithm is stable to at least  $\ell \sim 10,000$  and likely well beyond.

these terms in log-space to mitigate numerical issues inherent in evaluating large factorials. Also note that  $d_{\ell+1,n}^\ell(\beta)$  is trivially zero as  $d_{mn}^\ell(\beta) = 0$  for all  $m > \ell$ .

While some spherical samplings avoid samples at the north and south poles, others do have samples on the poles. Since we aim to support a number of different spherical samplings (as discussed in Sec. 2.5), including those with samples defined on the poles, we must be able to evaluate Wigner  $d$ -functions on the poles. The initialisation and recursion presented above are not suitable on the poles ( $\beta = \{0, \pi\}$ ) since the recursion coefficient  $\lambda_m$  is undefined due to the  $\sin\beta$  term in the denominator. Fortunately, simple forms for these elements of the Wigner  $d$ -functions are available, which we implement explicitly. Specifically, at the north pole [48, Sec. 4.16, p. 112, Eq. 1]

$$d_{mn}^\ell(0) = \delta_{mn} \quad (28)$$

and at the south pole

$$d_{mn}^\ell(\pi) = (-1)^{\ell+m} d_{m,-n}^\ell(0) = (-1)^{\ell+m} \delta_{m,-n}, \quad (29)$$

which follows from Varshalovich et al. [48, Sec. 4.4, p. 79, Eq. 1]. Our recursive algorithm to compute the Wigner  $d$ -functions in a highly parallelised and stable manner, and its initialisation, is outlined graphically in Fig. 2 for clarity.

Additionally, note that the Wigner  $d$ -functions exhibit an extreme dynamic range. For example, the magnitude of these coefficients  $d_{mn}^\ell(\beta)$  often ranges in excess of many hundreds of orders of magnitude. Hence, not only must one recurse along stable paths but one must also develop protocols to avoid under- and over-flowing, which can be equally catastrophic. A common approach to address this issue is to implement a renormalisation scheme, which conditionally compresses the dynamic range during the recursion to an acceptable interval. However, while this may solve the dynamic range problem in theory, algorithms with this conditional structure do not permit efficient automatic differentiation. In fact, due to branching trees, such an algorithm can become extremely expensive to differentiate. Alternative methods exist, such as continued fractions [e.g. 59], however, as we discuss below in Sec. 3.4, with some careful software engineering it is possible to develop non-conditional algorithms for renormalisation that are not only feasible, but exhibit preferable complexity scaling.

### 3.4. Stable JAX recursion

Beginning from the initial values given by Eq. 27 sequentially evaluating the recursion presented in Eq. 25 can rapidly result in iterates  $d_{mn}^\ell(\beta)$  that under- or over-flow 64-bit representations. One may diligently check the iterates magnitude at recursive step  $j$  against some large scaling factor  $C$  and, if larger, update a running normalisation constant  $C^{\ell\beta} \rightarrow C^{\ell\beta}/C$  before retrospectively renormalising all  $d_{\ell-k,n}^\ell \rightarrow d_{\ell-k,n}^\ell/C$  for  $k \leq j$  therefore avoiding problematic numerical overflows. This approach is simple and effective for serial evaluation, however it creates two issues for accelerated differentiable programming.

Firstly, suppose whilst recursing the iterate magnitude becomes larger than  $C$  after each  $j$  iterations, at which point the above renormalisation scheme is executed. For a given harmonic degree  $\ell$  one would expect to encounter  $\sim \ell/j$  renormalisation nested loops. The average number of iterates that must be renormalised within each of these loops is straightforwardly  $\sim \ell/2$ . Compounding these factors it is clear that this approach to renormalisation changes the overall complexity of the recursion from  $O(\ell)$  to  $O(\ell^2)$ . Adopting such an approach for a spherical harmonic transform with bandlimit  $L$  results in  $O(L^4)$  complexity which is prohibitive for high  $L$ .

Secondly, although control flow primitives acting on tracer objects, such as checking the magnitude of iterates, are supported by JAX, they are inefficient in this case. Both branches of the conditional flow must be compiled and executed at runtime, whether the condition is met or not. The renormalisation strategy involves two such primitives within every iteration, one to determine whether the renormalisation condition is met and another to determine which elements to which to apply this renormalisation. Not only do such nested conditional flow primitives require exponential memory allocation, they also require CPU scheduling incurring further communication overhead. Similar issues are prevalent when evaluating the gradients of such primitives.

We propose, perhaps counter-intuitively, to instead update the normalisation after every recursive step, in effect performing a running renormalisation by default, rather than dynamically renormalising when under- or over-flow conditions are met. In this manner we avoid passing back over terms many times to renormalise, thus avoiding the associated computation cost and conditional branching that is otherwise required. Explicitly, we begin by normalising  $d_{\ell,n}^\ell$  to unity and store this value as our initial normalisation. We then iterate once in descending  $m$  calculating the normalised version of  $d_{\ell-1,n}^\ell$ , before undoing the normalisation and reading off  $d_{\ell-1,n}^\ell$ . At this point the absolute value of the normalised  $d_{\ell-1,n}^\ell$  is computed and aggregated into the normalisation factor. Hence, each recursive step begins normalised to unity with an appropriate running normalisation. Updating the normalisation in this way is computationally inexpensive and is a fair price to pay to retain  $O(\ell)$  complexity. Moreover, as we automatically renormalise after each iteration, no control flow primitives are needed, and thus issues pertaining to automatic differentiation and nested branching trees are avoided entirely. Throughout we compute and store terms in log-space to achieve a larger dynamic range, further mitigating the risk of under- and over-flow.

Through the approaches presented, we are able to compute Wigner  $d$ -functions in a manner that can be efficiently deployed on modern hardware accelerators exhibiting high degrees of parallelisation, that provides efficient automatic differentiation, and that are stable to very high degrees  $\ell$ , to at least  $\ell \sim 10,000$  and likely well beyond. In the following section we leverage these approaches to compute spin spherical harmonic and Wigner transforms.

## 4. Fast Fourier transforms on the sphere and rotation group

We present in this section algorithms for the fast computation of Fourier transforms on the two-sphere  $\mathbb{S}^2$  and three-dimensional rotation group  $\text{SO}(3)$ : namely, spherical harmonic transforms (for any arbitrary spin) and Wigner transforms, respectively. We consider a simple general algorithmic structure that is well suited to many different sampling approaches and for deployment on modern hardware accelerators, such as GPUs and TPUs, leveraging the parallelised and differentiable Wigner  $d$ -function computation presented in Sec. 3. We discuss in Sec. 5 the gradient computation for the spherical transforms presented in this section.

In much the same way that the standard fast Fourier transform [FFT; 62] is reliant on uniform sampling, fast generalised Fourier transforms on the sphere and rotation group are reliant on isolatitudinal sampling. Consequently, we focus our attention solely to isolatitudinal sampling (as discussed in Sec. 2.4), which includes the majority of commonly adopted spherical sampling approaches (as discussed in Sec. 2.5). The key advantage of isolatitudinal samplings is that a separation of variables may be performed to reduce computational complexity [e.g. 43, 41, 42, 50].

Since our aim is to support many different spherical sampling approaches (e.g. DH, MW, GL, HEALPix), we adopt a general algorithmic structure for the computation of spherical transforms that can be applied to any isolatitudinal

sampling that exhibits an explicit quadrature for the spherical transform. Not all sampling approaches necessarily exhibit an explicit quadrature and we discuss some tricks to address this.

We focus first on the general algorithmic form of the spherical harmonic transform and then the specifics of different sampling schemes. We then discuss a precomputation approach that is highly efficient for moderate bandlimits, before presenting an alternative optimisation applicable for high bandlimits that is achieved by interleaving the Wigner  $d$ -function recursion with the computation of the harmonic transform. Finally, we discuss the application of the techniques presented to also compute Wigner transforms.

#### 4.1. Spin spherical harmonic transforms

We consider the computation of the spin spherical harmonic transform of bandlimited signals  $f \in \mathcal{B}_L(\mathbb{S}^2)$  denoted by  $\mathcal{F} : \mathcal{B}_L(\mathbb{S}^2) \rightarrow \mathbb{C}^{L^2}$  and the inverse denoted by  $\mathcal{F}^{-1} : \mathbb{C}^{L^2} \rightarrow \mathcal{B}_L(\mathbb{S}^2)$ . Since we consider isolatitudinal samplings where the transform can be expressed through an explicit quadrature, a separation of variables can be applied to the discretised representation of the forward transform given by Eq. 19, yielding

$${}_s\hat{f}_{\ell m} \simeq (-1)^s \sqrt{\frac{2\ell+1}{4\pi}} \sum_{\theta} q_{\Theta}(\theta) d_{m,-s}^{\ell}(\theta) {}_s\tilde{f}_m(\theta), \quad (30)$$

with

$${}_s\tilde{f}_m(\theta) = \sum_{\phi} q_{\Phi}(\phi) {}_sf(\theta, \phi) e^{-im\phi}. \quad (31)$$

Without loss of generality we have expressed the quadrature weights in the separable form  $q(\theta, \phi) = q_{\Theta}(\theta) q_{\Phi}(\phi)$  (merely for notational convenience for comparison with the inverse transform subsequently). Recall that we adopt the symbol  $\simeq$  to denote that the computation is approximate for arbitrary sampling schemes (*e.g.* HEALPix) but is exact for sampling theorems (*e.g.* DH, MW, GL).

Note that Eq. 30 is a projection onto the Wigner  $d$ -functions that can be computed with complexity  $O(L^3)$ , while Eq. 31 is simply a one-dimensional Euclidean Fourier transform that can be computed by an FFT for each isolatitudinal ring at  $\theta$  with overall complexity  $O(L^2 \log L)$ . The separation of variables therefore results in an overall complexity of  $O(L^3 + L^2 \log L) = O(L^3)$ , in contrast with the  $O(L^4)$  complexity of direct computation by Eq. 19. Since Eq. 31 is simply computed by FFTs, the main computational challenge is the efficient computation of Eq. 30, which is discussed in greater detail in Sec. 4.3 and Sec. 4.4.

A similar approach through a separation of variables may also be applied to compute the inverse transform of Eq. 4, with a further reordering of summations, yielding

$${}_sf(\theta, \phi) = \sum_m {}_s\tilde{f}_m(\theta) e^{im\phi}, \quad (32)$$

with

$${}_s\tilde{f}_m(\theta) = \sum_{\ell} (-1)^s \sqrt{\frac{2\ell+1}{4\pi}} {}_s\hat{f}_{\ell m} d_{m,-s}^{\ell}(\theta). \quad (33)$$

Again, Eq. 32 may be computed by FFTs with overall complexity  $O(L^2 \log L)$  and Eq. 33 may be computed with complexity  $O(L^3)$ , resulting in an overall complexity of  $O(L^3 + L^2 \log L) = O(L^3)$ , in contrast with the  $O(L^4)$  complexity of direct computation by Eq. 4. The efficient computation of Eq. 33 is discussed further in Sec. 4.3 and Sec. 4.4.

For now we simply note that since we develop methods to compute the Wigner  $d$ -functions that recurse over  $m$  only (see Sec. 3) that we can compute independently for all  $\ell$  and  $\theta$ , and typically fixed  $n$ , Eq. 30 and Eq. 33 can be computed in a highly parallelised manner on hardware accelerators, *i.e.* GPUs and TPUs, before reducing over  $\theta$  or  $\ell$ .

Next we discuss how to map various sampling approaches onto the general algorithmic structure for the forward and inverse spherical harmonic transform presented here. In many cases this is straightforward, although in some cases it is non-trivial.

#### 4.2. Specifics of different sampling approaches

We summarise the details required to support a number of different sampling approaches, including the DH, MW, and GL sampling theorems, which admit theoretically exact spherical transforms, and the HEALPix sampling scheme, which admits only approximate spherical transforms but exhibits the practical advantage of equal-area pixels. All sampling approaches are mapped onto the general algorithmic structure presented in Sec. 4.1 for the computation of spherical harmonic transforms through a separation of variables with an explicit quadrature for the spherical transform.

#### 4.2.1. Driscoll & Healy (DH) sampling theorem

The DH [Driscoll & Healy; 40] sampling theorem adopts equiangular sampling position on the sphere defined by

$$\theta_t = \frac{\pi t}{2L} \quad \text{for } t \in \{0, 1, \dots, 2L-1\} \quad (34)$$

and

$$\phi_p = \frac{2\pi p}{2L} \quad \text{for } p \in \{0, 1, \dots, 2L-1\}, \quad (35)$$

resulting in  $\sim 4L^2$  samples on the sphere. These sample locations afford a sampling theorem with explicit quadrature weights for spherical transforms given by [40, 63]

$$q(\theta, \phi) = \frac{2\pi}{L^2} \sin \theta \sum_{k=0}^{L-1} \frac{\sin((2k+1)\theta)}{2k+1}. \quad (36)$$

Spherical harmonic transforms can thus be computed straightforwardly for the DH sampling theorem, following the algorithmic structure outlined above in Sec. 4.1.

#### 4.2.2. McEwen & Wiaux (MW) sampling theorem

The MW [McEwen & Wiaux; 41] sampling theorem provides an alternative equiangular sampling theorem requiring only  $\sim 2L^2$  samples on the sphere, providing a reduction in the Nyquist rate on the sphere by a factor of two compared to the DH sampling theorem.

There are two variants of the MW sampling scheme: the original scheme which provides the minimal Nyquist sampling on the sphere [41]; and a variant which slightly over samples (but yet still with  $\sim 2L^2$  samples) to yield sample locations with antipodal symmetry that is useful for many practical applications [6, 64], denoted MWSS (MW with symmetric sampling). The MW sample positions are given by

$$\theta_t = \frac{(2t+1)\pi}{2L-1} \quad \text{for } t \in \{0, 1, \dots, L-1\} \quad (37)$$

and

$$\phi_p = \frac{2\pi p}{2L-1} \quad \text{for } p \in \{0, 1, \dots, 2L-2\}, \quad (38)$$

whereas the MWSS sample positions are given by

$$\theta_t = \frac{2\pi t}{2L} \quad \text{for } t \in \{0, 1, \dots, L\} \quad (39)$$

and

$$\phi_p = \frac{2\pi p}{2L} \quad \text{for } p \in \{0, 1, \dots, 2L-1\}. \quad (40)$$

Quadrature weights have been presented for both the MW [41] and MWSS [64] cases, which we unify into a single expression. The MW sampling theorem is based on an extension of the sphere to the torus through careful period extensions of the function of interest to the  $\theta$  domain  $[0, 2\pi]$ ; recall that on the sphere  $\theta \in [0, \pi]$ . Consider the bandlimited representation of the function defined by  $\sin \theta$  on  $[0, \pi]$  and zero on  $(\pi, 2\pi)$ , given by

$$w(\theta_t) = \sum_m \hat{w}(m') e^{im'\theta_t}, \quad (41)$$

with Fourier coefficients

$$\hat{w}(m') = \int_0^\pi \sin(\theta) e^{im'\theta} d\theta = \begin{cases} \pm i\pi/2 & m' = \pm 1 \\ 0 & m' \text{ odd}, m' \neq \pm 1 \\ 2/(1-m'^2) & m' \text{ even} \end{cases}. \quad (42)$$

The unified quadrature weights are then given by folding back contributions from  $(\pi, 2\pi)$  onto  $(0, \pi)$  and read

$$q(\theta_t, \phi) = \frac{2\pi}{T^2} (w(\theta_t) + \zeta_t(-1)^s w(\theta_t^{\text{reflect}})), \quad (43)$$

where

$$T = \begin{cases} 2L-1 & \text{for MW} \\ 2L & \text{for MWSS} \end{cases}; \quad \theta_t^{\text{reflect}} = \begin{cases} \theta_{2L-2-t} & \text{for MW} \\ \theta_{2L-t} & \text{for MWSS} \end{cases}; \quad \zeta_t = \begin{cases} (1 - \delta_{t,L-1}) & \text{for MW} \\ (1 - \delta_{t0})(1 - \delta_{tL}) & \text{for MWSS} \end{cases}. \quad (44)$$

Note, however, that these quadrature weights are for the integration of a function bandlimited at  $L$  and not for the computation of the spherical harmonic transform of Eq. 5, the integrand of which is bandlimited at  $2L$ . While one could derive an explicit quadrature weight associated with the spherical transforms, we simply upsample signals on the sphere by a factor of two and consider quadrature weights appropriate for the integration of a function bandlimited at  $2L$ . Note that the signal on the sphere is simply upsampled and the resulting spherical harmonic transform is still performed at bandlimit  $L$ .

Signals sampled following either of the MW sampling approaches can first be extended to the torus by a reflection in  $\theta$ , combined with the introduction on the extended domain of a shift of  $\pi$  in  $\phi$  and a  $(-1)^s$  sign shift [41]. While this extension can be computed in harmonic space for either MW or MWSS sampling theorems, it can also be straightforwardly computed in the spatial domain for MWSS due to the symmetric nature of the sample positions. Once the extended function on the torus is computed it can be upsampling simply by zero padding in the Fourier domain and then truncated back to the spherical domain  $\theta \in [0, \pi]$ . Following this approach the algorithmic structure of the spherical harmonic transform algorithms presented in Sec. 4.1 can be followed directly.

#### 4.2.3. Gauss-Legendre (GL) sampling theorem

An alternative sampling theorem on the sphere can be built on Gauss-Legendre quadrature, where sample positions in  $\theta$  are defined by the roots of the Legendre polynomials of order  $L$  [e.g. 54, 41]. Samples positions in  $\phi$  may be defined by

$$\phi_p = \frac{2\pi p}{2L-1} \quad \text{for } p \in \{0, 1, \dots, 2L-2\}. \quad (45)$$

While equiangular sampling theorems are usually preferred for practical purposes due to the regular nature of sample positions in  $\theta$ , the GL sampling theorem on the sphere recovers the same asymptotic number of samples on the sphere as the MW sampling theorem ( $\sim 2L^2$ ), although in practice slightly more samples are required by the GL sampling compared to MW. Explicit quadrature weights for the computation of spherical transforms are given by [65]

$$q(\theta, \phi) = \frac{4\pi \sin^2 \theta}{(2L-1)(P_L'(\cos \theta))^2}, \quad (46)$$

where above the prime denotes the derivative of the Legendre polynomial. Spherical harmonic transforms can thus be computed straightforwardly for the GL sampling theorem, following the algorithmic structure outlined above in Sec. 4.1.

#### 4.2.4. HEALPix sampling scheme

HEALPix sampling is designed to provide pixels of equal area, which can be a very useful practical property. We refer the reader to Górski et al. [43] for full details regarding HEALPix, including the precise definition of sample positions, and focus here on the implications for computing spherical harmonic transforms. Since pixels have equal area, approximate quadrature weights are simply given by  $q(\theta, \phi) = 4\pi/N_{\text{pix}}$  where  $N_{\text{pix}}$  denotes the total number of samples over the sphere. Before proceeding, note that for HEALPix it is customary to compute transforms to a maximum degree denoted by  $\ell_{\text{max}}$ , which is related to our definition of bandlimit  $L$  by  $\ell_{\text{max}} = L - 1$ .

By construction HEALPix samples are defined on isolatitudinal *rings* with two additional attributes to support a hierarchical tessellation of the sphere: (i) every other ring is shifted by a fixed longitudinal offset; and (ii) the total number of samples per ring is not constant. The ring offsets are managed straightforwardly by a phase correction term which is very cheap and easily accounted for within the Fourier transform by modification of the exponential term.

For the forward spherical harmonic transform a 1D Fourier transform of  $f(\theta, \phi)$  over  $\phi$  for a ring of constant  $\theta$  is performed to compute the intermediate functions  $\hat{f}_m(\theta)$  of Eq. 31. In theory, each ring contributes information to all  $|m| \leq \ell$  for  $\ell \leq L$ , and when in the equatorial region (the region near the equator) HEALPix collects sufficient samples to capture this information. In the polar regions the number of longitudinal samples per ring is dynamic and (potentially much) less than  $m$ , hence information is lost. This is in fact precisely why HEALPix provides only approximate spherical harmonic transforms.

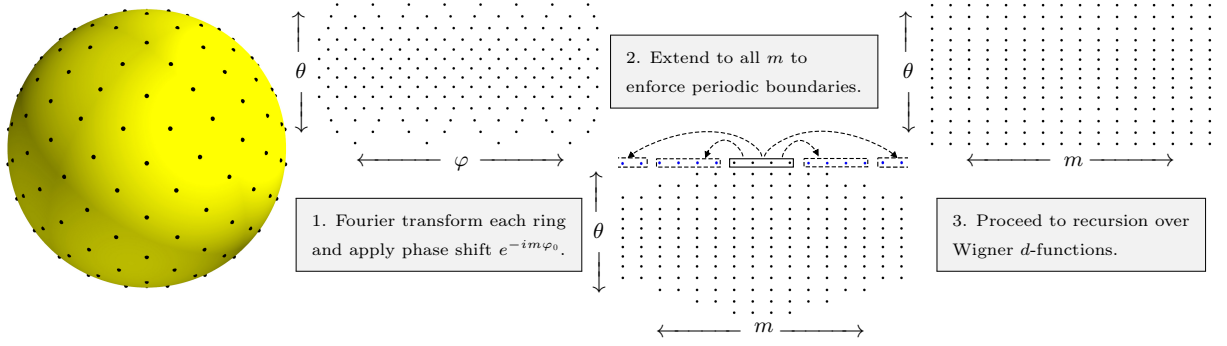


Fig. 3: Overview of the additional steps required for HEALPix sampling [43] when computing Fourier transforms over  $\phi$ . For HEALPix one must: (i) compute the Fourier transform each ring at varying resolution, determined by  $n_\phi$  per ring; then (ii) generate synthetic orders  $m'$  by extending periodically available orders  $|m| \leq |m'|$ . For the inverse spherical harmonic transform the order is reversed and step (ii) is simply replaced by aliasing higher orders  $m'$   $|m'| > |m|$  into lower orders  $m$ . This is in fact precisely why HEALPix provides only approximate spherical harmonic transforms. For equirectangular sampling schemes this entire step can be computed simply by an FFT for each  $\phi$  for all rings in a single line of code.

To mitigate this effect in the polar regions it is customary to first perform a fast Fourier transform over  $\phi$  before repeating the intermediate function  $\hat{f}_{m' \leq m}^{\text{polar}}(\theta)$ , extending the coefficients periodically to fill out the Fourier  $m$  line. Correspondingly, during the inverse transform one *folds* the full order  $\hat{f}_m(\theta)$  onto  $\hat{f}_{m' \leq m}^{\text{polar}}(\theta)$ , aliasing high frequency information into the low frequency components. As this nuance is perhaps not obvious, we provide an illustration in Fig. 3 for clarity.

From a software engineering perspective, this raises some further complications. Ideally, the FFTs of Eq. 31 and Eq. 32 would be implemented as a single function call to `jax.numpy.fft` across a regular array. However, `jax.numpy.fft` does not support ragged arrays and therefore each ring within the polar region must be computed by individual function calls within a scheduled loop. As the arrays are dynamic this cannot be replaced by, *e.g.*, a `lax` scan, and therefore suffers from compile time bloat due to unrolled loops in XLA compilation. Moreover, such serial evaluation of fast Fourier transforms for each isolatitudinal ring does not fully exploit the distributed compute of GPU devices. Nevertheless, the compute time of the overall transform is dominated by the Wigner  $d$ -functions summations and therefore this inefficiency of HEALPix is relatively minor, and may be solved by future developments of core JAX primitives. In summary, HEALPix just-in-time compilation is therefore slow, although this only needs to be performed on the first function call (further optimisations to improve this will be considered in future). Once HEALPix transforms are just-in-time compiled, execution is a little slower than for equiangular samplings but nevertheless still relatively fast.

#### 4.3. Full precomputation

We now consider how best to evaluate the summations involving the Wigner  $d$ -functions presented in Eq. 30 and Eq. 33. For moderate bandlimits ( $L \lesssim 2048$ ) the Wigner  $d$ -functions may be precomputed, yielding highly efficient harmonic transforms, although at a cost of increased memory usage. One may precompute and store all required elements  $d_{m,-s}^\ell(\theta)$ , the weighted sum of which can then be implemented extremely straightforwardly. Explicitly, for a forward spherical harmonic transform, given the precomputation and storage of the Wigner  $d$ -functions  $D = d_{m,-s}^\ell(\theta)$  for all  $\ell, m, \theta$  at a fixed spin  $s$ , the transform is simply given by a series of 1D FFTs over  $\phi$  for each  $\theta$ , followed by multiplication by the quadrature weights, followed by multiplication by matrix  $D$ , followed by an  $\ell$ -dependent scaling. The core implementation of the spherical harmonic transform therefore collapses effectively to a single line of code `jnp.einsum('t1m, tm -> lm', D, jnp.fft(f))`. The inverse transform follows analogously, with minor changes (different ordering, forward 1D FFTs replaced by inverse FFTs, and different scalings). While this precompute approach is highly computationally efficient, it requires  $O(L^3)$  memory, which is not feasible at high bandlimits  $L$ . We next consider an approach that avoids fully precomputing Wigner  $d$ -functions, eliminating  $O(L^3)$  memory requirements, and thus is scalable to high degrees  $L$ . In this approach we do consider an optional precomputation of some other internal terms, although in a manner that limits memory requirements to  $O(L^2)$ .

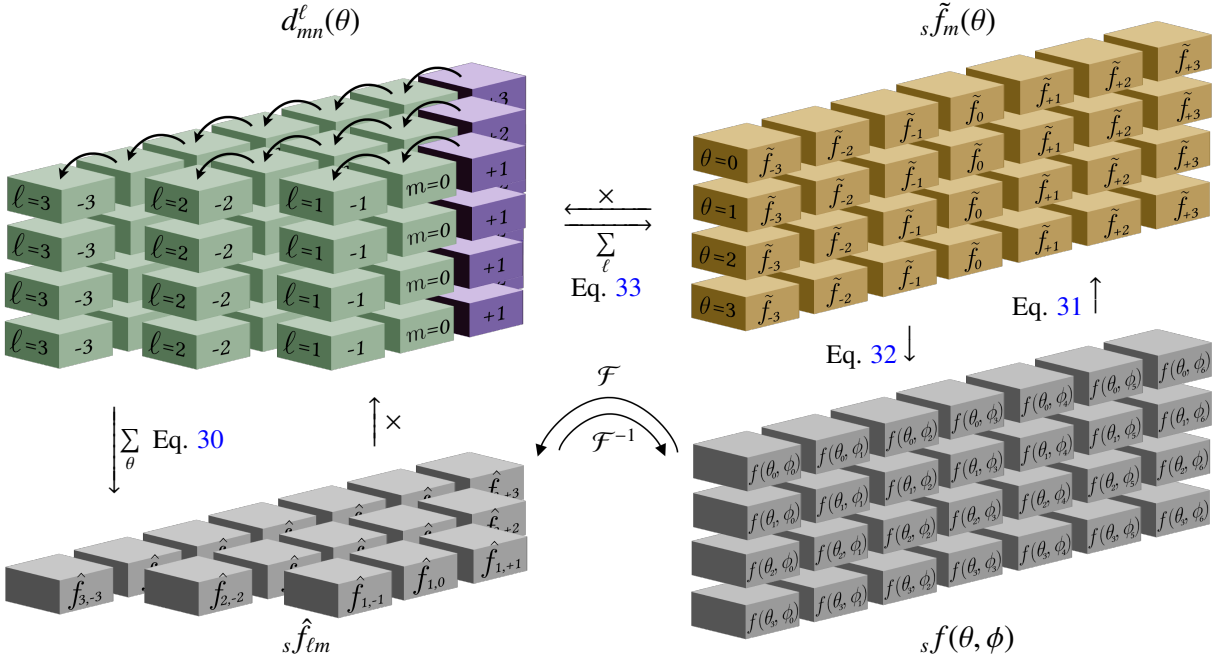


Fig. 4: Diagrammatic overview of our forward and inverse spin spherical harmonic transforms. The forward transform  $\mathcal{F}$  begins from a bandlimited signal on the sphere  $s f(\theta, \phi)$  (bottom right) from which an intermediate function  $\tilde{s}\tilde{f}_m(\theta)$  (top right) is calculated by performing a 1D FFT over  $\phi$  for each  $\theta$ . This intermediate function  $\tilde{s}\tilde{f}_m(\theta)$  is subsequently projected onto  $d_{mn}^\ell(\theta)$  (top left) before being summed over  $\theta$  to produce harmonic coefficients  $\hat{s}\hat{f}_{\ell m}$  (bottom left). The inverse spin spherical harmonic  $\mathcal{F}^{-1}$  follows straightforwardly in the reverse direction as illustrated. We provide support for two operational modalities: precompute and on-the-fly. Our precompute transform explicitly calculates and stores  $d_{mn}^\ell(\theta)$  for a given spin number  $n$ , which produces dramatically accelerated transforms at runtime at the cost of  $O(L^3)$  memory overhead. Our on-the-fly transform evaluates  $d_{mn}^\ell(\theta)$  for a given spin number  $n$  and single harmonic mode  $m$  (see Sec. 3) at each recursive step (curved arrow) projecting onto harmonic space by  $\sum_\theta$  or the intermediate representation by  $\sum_\ell$ .

#### 4.4. Interleaving recursion and computation for on-the-fly transforms

To scale to high resolutions (from  $L \gtrsim 1024$ , to  $L \sim 10,000$  and beyond), the Wigner  $d$ -function components  $d_{m,-s}^\ell(\theta)$  must instead be computed, used, and discarded recursively on-the-fly. Suppose one adopts the recursive algorithm defined in Eq. 25, then evaluation of Eq. 30 is best solved by interleaving the summation over  $\theta$  with the recursive step over order  $m$ . In the following we adopt the shorthand  $D_j^{\ell\theta}$  for the  $j^{\text{th}}$  iterant during our recursive computation of  $d_{m,-s}^\ell(\theta)$ . Within a recursive step  $j$  we: (i) evaluate  $D_{j+1}^{\ell\theta}$  from  $D_j^{\ell\theta}$  and  $D_{j-1}^{\ell\theta}$  after which these arrays are incremented; (ii) renormalise  $D_{j+1}^{\ell\theta}$  as discussed in Sec. 3; (iii) project onto  $D_{j+1}^{\ell\theta}$ ; and (iv) sum over all co-latitudes  $\theta$ . In this way our peak memory overhead is capped with complexity  $O(L^2)$  which is the memory required to store a single spherical image. Furthermore, we optionally precompute some additional internal terms to save computational time, while keeping memory overhead to  $O(L^2)$ . The inverse transform is completely analogous. This resulting program is outlined in Fig. 4.

As noted above, the Wigner summation terms of the forward and inverse transforms, given by Eq. 30 and Eq. 33, can be computed in a highly parallelised manner on hardware accelerators, *i.e.* GPUs and TPUs, before reducing over  $\theta$  or  $\ell$ . We recover  $O(L^2)$  independent threads, which for high bandlimits  $L \sim 10^4$  can reach  $L \sim 10^8$ . We are therefore able to make good use of the very large number of threads available on modern hardware accelerators. Furthermore, we can trivially distribute computations over multiple accelerators, *e.g.* multiple GPU devices, through single program multiple data (SPMD) computation, which is handled seamlessly in JAX. Asymptotically, such distributed computation recovers a computational saving given by the number of GPU devices available, and so for large clusters the acceleration can be extreme (indeed, in practice we find very close to a linear computational saving; see Sec. 6). Due to the  $O(L^2)$  distribution and parallelisation of our algorithms, given sufficient computational resources our transforms exhibit  $O(L)$  time complexity.

#### 4.5. Wigner transforms

To conclude this section we consider the computation of the forward Wigner transform of bandlimited signals  $f \in \mathcal{B}_L(\text{SO}(3))$  denoted by  $\mathcal{W} : \mathcal{B}_L(\text{SO}(3)) \rightarrow \mathbb{C}^{(4L^3-L)/3}$  and the inverse denoted by  $\mathcal{W}^{-1} : \mathbb{C}^{(4L^3-L)/3} \rightarrow \mathcal{B}_L(\text{SO}(3))$ . As discussed in Sec. 2.3 [42] by relating the Wigner  $D$ -functions to the spin spherical harmonic basis functions, one can compute the Wigner transform through a series of spin spherical harmonic transforms and Fourier transforms (see Eq. 15–Eq. 18). We can therefore make use of the approach previously discussed to compute forward and inverse spherical harmonic transforms for arbitrary spin, making the association  $\alpha \leftrightarrow \phi$  and  $\beta \leftrightarrow \theta$ , and simply include an additional 1D FFT with respect to  $\gamma$ .

By selecting an equiangular sampling in  $\gamma$  we can appeal to the usual Nyquist sampling theorem for periodic functions on the circle  $\mathbb{S}^1$ . The full quadrature weights can then be written as  $q(\alpha, \beta, \gamma) = q_\phi(\alpha) q_\theta(\beta) q_\gamma(\gamma)$ , where  $q_\phi(\alpha) q_\theta(\beta)$  vary depending on the spherical sampling theorem or scheme adopted (as discussed in Sec. 4.2). Since for all spherical samplings we consider equiangular sampling in  $\gamma$ , the corresponding quadrature weights read  $q_\gamma(\gamma) = 2\pi/(2N - 1)$  for azimuthal bandlimit  $N$ . Whether or not we recover a sampling theorem on  $\text{SO}(3)$  with exact Wigner transforms simply depends on whether we adopt a sampling scheme (*e.g.* HEALPix) or theorem (*e.g.* DW, MW, GL) on the sphere.

This formulation of the Wigner transform straightforwardly inherits the advantages of our JAX spin spherical harmonic implementation and provides further avenues for distribution. The internal forward or inverse spherical harmonic transform considered must be evaluated for each spin number  $n$ , however each of these operations is independent and can be parallelised and/or distributed. Therefore for our Wigner transforms one may choose to distribute over not only  $\ell$  and  $\beta$  but also  $n$ .

### 5. Gradients of spherical transforms

One of the primary goals of this work is not only to provide computational approaches for spherical transforms that can be deployed on hardware accelerators but that also facilitate the efficient computation of gradients. Gradients are essential for differentiable programming applications. For example, machine learning models on the sphere [66, 4, 32, 33] often require spherical transforms that are differentiable so that the models may be trained by gradient-based optimisation algorithms. Differentiable physical models are also required for hybrid data-driven and model-based approaches; in many cases an underlying component of such physical models includes spherical transforms, for example in cosmology [25] and seismology [26].

In this section we present techniques to efficiently compute gradients associated with forward and inverse spherical transforms. We discuss both automatic and manual differentiation and their respective merits, and derive explicit expressions for Jacobian-vector and vector-Jacobian products. Finally we present a hybrid approach, where we adopt manual differentiation for some components and automatic differentiation for others. Our hybrid approach avoids the large memory overhead of (reverse-mode) automatic differentiation, while also avoiding the cumbersome bespoke implementation of full manual differentiation.

#### 5.1. Automatic differentiation (AD) modes

Modern automatic differentiation (AD) predominately relies on one of two gradient propagation schemes: forward- and reverse-mode. Forward-mode AD applies the chain rule sequentially to each primitive operation in a forward (primal) trace of a function, accumulating the primal and derivative (tangent) variables during a single forward pass [67]. Conversely, in reverse-mode AD forward and reverse passes are performed. During the forward pass primal variables are computed and propagated, with intermediate values stored in memory for use in the following reverse pass. In the reverse pass the derivatives (cotangents) with respect to intermediate values are propagated backwards [68]. Reverse mode AD applied to training neural networks with a scalar cost function is commonly referred to as back-propagation [69]. Forward- and reverse-mode AD provide efficient ways to compute, respectively, Jacobian-vector products and vector-Jacobian (transposed Jacobian-vector) products for the traced function.

The choice of which mode is most efficient is determined by the problem at hand. Consider the case in which we wish to compute the derivative of a function  $\mathcal{F} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ . A full column of the Jacobian can be computed in a single pass of forward-mode AD, whereas a full row of the Jacobian can be computed in a single pass of reverse-mode AD [68]. Consequently, when  $n \ll m$  forward-mode AD is most efficient, whereas when  $m \ll n$  reverse-mode is most efficient. However, note that intermediate values must be stored for reverse-mode AD, increasing memory requirements, which is not the case for forward-mode AD.

In machine learning scenarios one typically wishes to optimise a scalar loss function that depends on a very large number of input parameters (e.g., millions or billions of weights of a neural network). To train a model using a gradient-based optimiser it is therefore necessary to compute gradients of a function  $\mathcal{F} : \mathbb{R}^n \rightarrow \mathbb{R}$ , where the Jacobian is identified with the gradient vector  $\nabla \mathcal{F} \in \mathbb{R}^n$ . Reverse-mode AD is thus typically the preferred approach for training machine learning models. Nevertheless, we support both forward- and reverse-mode AD.

### 5.1.1. Jacobian-vector product (JVP)

Consider the general setting of a function  $\mathcal{F} : \mathbb{R}^n \rightarrow \mathbb{R}^m$  and its Jacobian  $\partial \mathcal{F}(x)$  evaluated at  $x \in \mathbb{R}^n$ . The Jacobian provides a linear map between the tangent space of the domain of  $\mathcal{F}$  at  $x$  (isomorphic to  $\mathbb{R}^n$ ) to the tangent space of the codomain of  $\mathcal{F}$  at  $\mathcal{F}(x)$  (isomorphic to  $\mathbb{R}^m$ ):  $\partial \mathcal{F}(x) : \mathbb{R}^n \rightarrow \mathbb{R}^m$ . Computationally, the Jacobian-vector product (JVP) of the Jacobian at  $x$  and a tangent vector  $v \in \mathbb{R}^n$  is given by the functional mapping  $(x, v) \mapsto (\mathcal{F}(x), \partial \mathcal{F}(x)v)$ . When composing multiple functions both the primal and tangent values are computed and propagated. Since values are used as they are computed in a single forward pass, intermediate values do not need to be stored.

To compute the Jacobian explicitly, forward-mode AD is initialised by setting a single element of  $v$  to unity, corresponding to the input variable of interest, and the remainder to zero (i.e. one hot encoding). It is therefore clear that a full column of the Jacobian can be computed in a single pass of forward-mode AD and hence it is most efficient for  $m \gg n$ , i.e. for tall Jacobians, as commented above.

### 5.1.2. Vector-Jacobian product (VJP)

Consider again the function  $\mathcal{F} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ . Now, however, consider an element  $v \in \mathbb{R}^m$  of the cotangent space (i.e. the tangent space of the codomain of  $\mathcal{F}$ ). Computationally, the vector-Jacobian product (VJP) is given by the functional mapping  $(x, v) \mapsto (\mathcal{F}(x), v^T \partial \mathcal{F}(x))$ . One may alternatively consider the linear mapping of a VJP as the transpose of the linear mapping of a JVP:  $(x, v) \mapsto (\mathcal{F}(x), \partial \mathcal{F}(x)^T v)$ . The map  $\partial \mathcal{F}(x)^T : \mathbb{R}^m \rightarrow \mathbb{R}^n$  provides a linear mapping on cotangent spaces. Since the mapping is in the reverse direction to the application of the function  $\mathcal{F}$ , cotangents need to be propagated in reverse.

When composing multiple functions, for reverse-mode AD an initial forward pass is performed to compute and propagate the primals  $\mathcal{F}(x)$ , storing intermediate values of  $x$ . On the reverse pass the cotangents are computed by  $\partial \mathcal{F}(x)^T v$ , making use of the intermediate values of  $x$  to compute  $\partial \mathcal{F}(x)^T$ , and then accumulated in reverse.

To compute the Jacobian explicitly, reverse-mode AD is initialised by setting a single element of  $v$  to unity, corresponding to the output variable of interest, and the remainder to zero. It is therefore clear that a full row of the Jacobian can be computed in a single pass of reverse-mode AD and hence it is most efficient for  $m \ll n$ , i.e. for wide Jacobians, as commented above.

## 5.2. Automatic versus manual differentiation

By leveraging an AD framework, such as JAX, one can compute JVPs and VJPs simply through the framework. This approach avoids the need for any gradient-related implementation, thus eliminating complexity and possibilities for errors. However, as discussed above, reverse-mode AD induces a memory overhead that in the worst case can grow in proportion to the number of operations of the traced function [68]. Furthermore, for machine learning applications, reverse-mode AD is the mode of interest since it is most efficient for gradient-based optimisation of scalar loss functions that depend on a very large number of input parameters. For the spherical transforms considered in this article, we indeed observed that the computation of VJPs purely by reverse-mode AD exhibited a very high memory overhead (as discussed further below).

An alternative to automatic differentiation is manual differentiation, where explicit expressions for JVPs and VJPs are determined analytically and then implemented directly. For linear transforms, such as the spherical transforms considered herein, JVPs and VJPs take relatively straightforward forms, where the JVP is simply the transform itself and the VJP is closely related to the adjoint transform. Algorithms to compute adjoints of spherical transforms efficiently have been constructed previously [70, 71, 72]. Such a direct implementation of manual derivatives requires bespoke implementation, which can be cumbersome (especially to support the subtleties of different spherical samplings) and thus can be prone to errors. Nevertheless, a forward pass to compute VJPs is not required, eliminating the memory overhead that is otherwise necessary.

### 5.3. Manual JVPs and VJP

We derive analytic expressions for the JVP and VJP (and adjoint) of spherical transforms. For simplicity, but without loss of generality, we focus on the forward and inverse spherical harmonic transforms, although the expressions presented can be trivially extended to Wigner transforms. For notational brevity we drop the explicit spin subscript in what follows, although the results presented directly hold for spin functions on the sphere.

#### 5.3.1. Forward spin spherical harmonic transform

Consider the forward spin spherical harmonic transform  $\mathcal{F} : \mathcal{B}_L(\mathbb{S}^2) \rightarrow \mathbb{C}^{L^2}$ . For computational purposes we must of course consider discretised representations of signals, which for sampling theorems corresponds to the space of bandlimited signals  $f \in B_L(\mathbb{S}^2)$ . When adopting a sampling scheme that does not exhibit a formal sampling theorem (e.g. HEALPix), the space of signals is not formally  $B_L(\mathbb{S}^2)$ . Nevertheless, we avoid making an explicit distinction to avoid complicating notation. Furthermore, while for Eq. 19 we distinguished between the exactness of sampling theorems and the approximation of sampling schemes, we consider the computed harmonic coefficients here  $\hat{f} \in \mathbb{C}^{L^2}$  (rather than underlying true coefficients of a bandlimited signal) and so expressions that follow are computationally equal, rather than approximate.

The elements of the Jacobian of the forward transform  $\mathcal{F}$  follow from Eq. 19 and read

$$\left(\partial\mathcal{F}(f)\right)_{\ell m}^{\theta\phi} = \frac{\partial\hat{f}_{\ell m}}{\partial f(\theta, \phi)} = q(\theta, \phi) {}_sY_{\ell m}^*(\theta, \phi). \quad (47)$$

Critically, notice that the Jacobian is independent of the signal  $f$ , i.e.  $\partial\mathcal{F}(f) = \partial\mathcal{F}$ . Consequently, intermediate primal values do not need to be stored in reverse-mode AD (nevertheless, other memory overheads do arise, as discussed below). The JVP of a tangent  $v \in T_f\mathcal{B}_L(\mathbb{S}^2) \cong \mathcal{B}_L(\mathbb{S}^2)$ , where  $T_f$  denotes the tangent space evaluated at  $f$ , is then given by

$$\left(\partial\mathcal{F}v\right)_{\ell m} = \sum_{\theta\phi} q(\theta, \phi) v(\theta, \phi) {}_sY_{\ell m}^*(\theta, \phi), \quad (48)$$

which as expected is nothing more than the transform itself, i.e.

$$\partial\mathcal{F}v = \mathcal{F}v. \quad (49)$$

The VJP of a cotangent  $\hat{v} \in T_{\mathcal{F}(f)}\mathbb{C}^{L^2} \cong \mathbb{C}^{L^2}$ , where  $T_{\mathcal{F}(f)}$  denotes the cotangent space evaluated at  $\mathcal{F}(f)$ , is given by

$$\left(\partial\mathcal{F}^T\hat{v}\right)_{\theta\phi} = \sum_{\ell m} q(\theta, \phi) \hat{v}_{\ell m} {}_sY_{\ell m}^*(\theta, \phi), \quad (50)$$

which may be expressed in terms of an inverse transform  $\mathcal{F}^{-1}$  by

$$\partial\mathcal{F}^T\hat{v} = (Q\mathcal{F}^{-1}\hat{v}^*)^*, \quad (51)$$

where  $Q$  is the diagonal operator acting on spatial functions that corresponds to multiplication by the quadrature weights  $q(\theta, \phi)$ . It is apparent that a VJP can be computed by an inverse spherical harmonic transform of a complex conjugated cotangent vector, followed by the application of quadrature weights and subsequent conjugation.

Adjoint operators of transforms are commonly required (e.g. when solving inverse problems posed as variational regularisation problems by convex optimisation techniques; [70, 71, 72]). The VJP is closely related to the adjoint operator of the transform up to complex conjugation. For completeness we note the adjoint of the forward spherical harmonic transforms  $\mathcal{F}$  can be represented as

$$\mathcal{F}^H = Q\mathcal{F}^{-1}. \quad (52)$$

While we have focused on the forward spin spherical harmonic transform  $\mathcal{F}$  the above results extend trivially to the forward Wigner transform. It is simply necessary to replace the spherical harmonic transform  $\mathcal{F}$  with the Wigner transform for discretised signals  $\mathcal{W} : \mathcal{B}_L(\text{SO}(3)) \rightarrow \mathbb{C}^{(4L^3-L)/3}$  and to let  $Q$  represent application of the quadrature weights  $q(\alpha, \beta, \gamma)$  on  $\text{SO}(3)$ .

### 5.3.2. Inverse spin spherical harmonic transform

Let us now consider the inverse spin spherical harmonic transform  $\mathcal{F}^{-1} : \mathbb{C}^{L^2} \rightarrow \mathcal{B}_L(\mathbb{S}^2)$ . Again, for computational purposes we must of course consider discretised representations of signals.

The elements of the Jacobian of the inverse transform  $\mathcal{F}^{-1}$  follow from Eq. 4 and simply read

$$\left(\partial\mathcal{F}^{-1}(\hat{f})\right)_{\theta\phi}^{\ell m} = \frac{\partial f(\theta, \phi)}{\partial \hat{f}_{\ell m}} = {}_s Y_{\ell m}(\theta, \phi). \quad (53)$$

The Jacobian of the inverse transform differs to the Jacobian of the forward transforms only due to complex conjugation and the absence of quadrature weights (*cf.* Eq. 47). Notice again that the Jacobian is independent of the signal  $\hat{f}$ , i.e.  $\partial\mathcal{F}^{-1}(\hat{f}) = \partial\mathcal{F}^{-1}$ . Consequently, intermediate primal values again do not need to be stored in reverse-mode AD (but again other memory overheads do arise). The JVP of a tangent  $\hat{v} \in T_{\hat{f}}\mathbb{C}^{L^2} \cong \mathbb{C}^{L^2}$  is then given by

$$\left(\partial\mathcal{F}^{-1}\hat{v}\right)_{\theta\phi} = \sum_{\ell m} \hat{v}_{\ell m} {}_s Y_{\ell m}(\theta, \phi), \quad (54)$$

which again as expected is nothing more than the transform itself, i.e.

$$\partial\mathcal{F}^{-1}\hat{v} = \mathcal{F}^{-1}\hat{v}. \quad (55)$$

The VJP of a cotangent  $v \in T_{\mathcal{F}^{-1}(\hat{f})}\mathcal{B}_L(\mathbb{S}^2) \cong \mathcal{B}_L(\mathbb{S}^2)$  is given by

$$\left(\partial\mathcal{F}^{-T}v\right)_{\ell m} = \sum_{\theta\phi} v(\theta, \phi) {}_s Y_{\ell m}(\theta, \phi), \quad (56)$$

where we have adopted the notation  $\mathcal{F}^{-T} = (\mathcal{F}^{-1})^T$ . The VJP may be expressed in terms of a forward transform  $\mathcal{F}$  by

$$\partial\mathcal{F}^{-T}v = (\mathcal{F}Q^{-1}v^*)^*. \quad (57)$$

It is apparent that a VJP can be computed by the application of quadrature weights to a complex conjugated cotangent vector, followed by a forward spherical harmonic transform and subsequent conjugation.

For completeness, the adjoint of the inverse transform is closely related to the VJP and can be represented as

$$\mathcal{F}^{-H} = \mathcal{F}Q^{-1}. \quad (58)$$

Again, the above results extend trivially to the inverse Wigner transforms simply by replacing  $\mathcal{F}^{-1}$  by  $\mathcal{W}^{-1} : \mathbb{C}^{(4L^3-L)/3} \rightarrow \mathcal{B}_L(\text{SO}(3))$  and let  $Q$  represent application of the quadrature weights  $q(\alpha, \beta, \gamma)$  on  $\text{SO}(3)$ . A commutative diagram is provided in Figure 5 to more clearly summarise our formulation of the JVPs and VJP associated with the various spherical harmonic transforms.

### 5.4. Hybrid automatic and manual differentiable spherical transforms

In Sec. 5.3 we derive analytic expressions for JVP and VJP of spherical harmonic and Wigner transforms, which can be expressed simply in terms of forward or inverse transforms and application of quadrature weights (with appropriate complex conjugation). Due to the linear nature of the generalised Fourier transforms considered, we also show that intermediate primal values do not need to be stored for reverse-mode AD, substantially alleviating memory overheads.

In principle AD would seem like the preferred approach since it avoids the complexity and implementation overhead of manual differentiation. However, in practice the recursive nature of the computation of the Wigner  $d$ -functions requires a scan operation. The scan carry requires  $O(L^2)$  memory and the total number of iterations over which one must scan is  $O(L)$ . Hence, in JAX for reverse-mode AD  $O(L^3)$  memory is required, which is prohibitive at high bandlimits.

It would therefore seem that manual differentiation is the preferred approach to avoid significant memory overhead. However, the simple nature of the explicit expressions for the JVPs and VJP given in Sec. 5.3 hides the subtlety surrounding the details of different spherical samplings (e.g. the specific details discussed in Sec. 4.2, particularly for HEALPix and MW samplings).

We therefore consider a hybrid approach. We adopt manual differentiation for the core internal components of spherical transforms related to recursions but adopt AD for surrounding components of the transforms that deal with the subtleties of spherical samplings. In this manner we avoid the prohibitive memory overhead of a pure AD approach, while also avoiding the subtleties of cumbersome direct implementations of a manual differentiation approach.

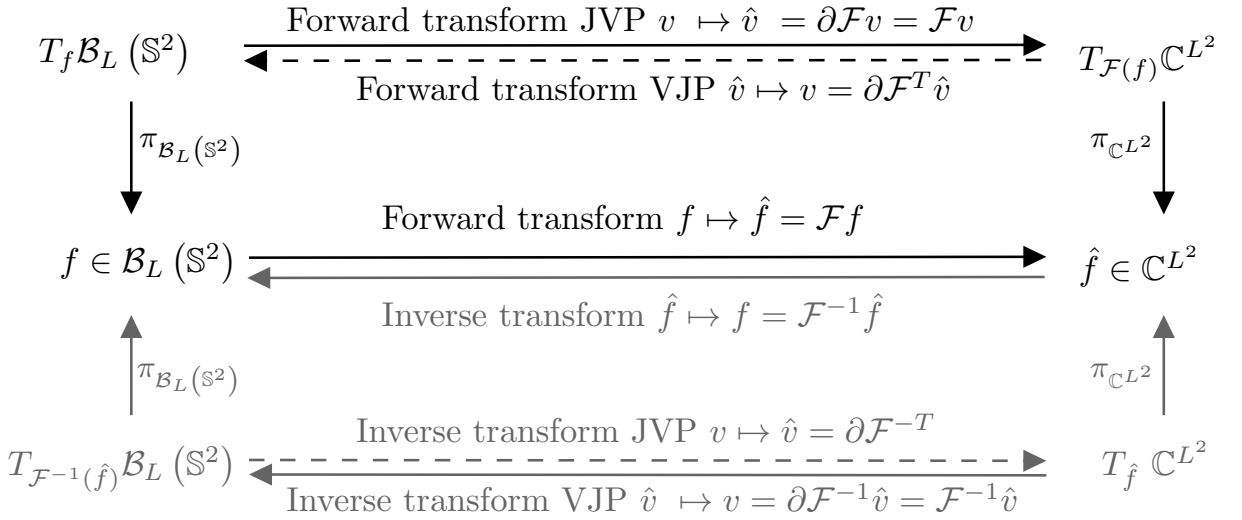


Fig. 5: Commutative diagram describing JVPs and VJP for both the forward (top loop in black) and inverse (bottom loop in grey) spherical transforms. These relations hold for both scalar and spin spherical harmonic transforms and also for Wigner transforms. Note that in this diagram  $\pi_X$  denotes the generalised projection of a function onto  $X$ . In each case the tangent (cotangent) space is isomorphic with the target space.

#### 5.4.1. Forward spin spherical harmonic transforms

Consider the forward transform  $\mathcal{F} = \mathcal{F}_\Theta \mathcal{F}_\Phi : f \mapsto \hat{f}$  in terms of the separate transforms with respect to  $\phi$  and  $\theta$ , respectively  $\mathcal{F}_\Phi : f \mapsto \tilde{f}$  (Eq. 31) and  $\mathcal{F}_\Theta : \tilde{f} \mapsto \hat{f}$  (Eq. 30). Since the computation of  $\mathcal{F}_\Theta$  requires evaluation of the Wigner  $d$ -functions by recursion, we compute gradients of this transform by explicit manual gradients, avoiding the memory overheads of AD. Remaining computations, which effectively capture the computation of  $\mathcal{F}_\Phi$  through multiple weighted 1D FFTs, are computed by AD, avoiding the subtleties of specific sampling schemes. In the following we therefore focus on the computation of manual gradients for  $\mathcal{F}_\Theta$ .

The elements of the Jacobian of the forward transform  $\mathcal{F}_\Theta$  read

$$\left(\partial \mathcal{F}_\Theta(\tilde{f})\right)_\ell^\theta = \frac{\partial \hat{f}_{\ell m}}{\partial \tilde{f}_m(\theta)} = (-1)^s \sqrt{\frac{2\ell+1}{4\pi}} q_\Theta(\theta) d_{m,-s}^\ell(\theta). \quad (59)$$

Again, notice that the Jacobian is independent of the signal  $\tilde{f}$ , i.e.  $\partial \mathcal{F}_\Theta(\tilde{f}) = \partial \mathcal{F}_\Theta$ . The JVP is then given by

$$\left(\partial \mathcal{F}_\Theta \tilde{v}\right)_\ell = \sum_\theta (-1)^s \sqrt{\frac{2\ell+1}{4\pi}} q_\Theta(\theta) \tilde{v}_m(\theta) d_{m,-s}^\ell(\theta), \quad (60)$$

which, as expected due to linearity, is simply given by the transform itself, i.e.

$$\partial \mathcal{F}_\Theta = \mathcal{F}_\Theta. \quad (61)$$

The VJP is given by

$$\left(\partial \mathcal{F}_\Theta^T \hat{v}\right)_\theta = \sum_\ell (-1)^s \sqrt{\frac{2\ell+1}{4\pi}} q_\Theta(\theta) \hat{v}_{\ell m} d_{m,-s}^\ell(\theta), \quad (62)$$

which may be expressed by

$$\partial \mathcal{F}_\Theta^T = Q_\Theta \mathcal{F}_\Theta^{-1} = \mathcal{F}_\Theta^H, \quad (63)$$

where  $\mathcal{F}_\Theta^{-1}$  is given by Eq. 33,  $Q_\Theta$  is the diagonal operator corresponding to multiplication by the quadrature weights  $q_\Theta(\theta)$ , and  $\mathcal{F}_\Theta^H$  denotes the corresponding adjoint operator. Note the similarity to Eq. 51 and Eq. 52, although here complex conjugations are not required due to the reality of the transform (which follows from the reality of the Wigner  $d$ -functions).

#### 5.4.2. Inverse spin spherical harmonic transforms

Consider the inverse transform  $\mathcal{F}^{-1} = \mathcal{F}_{\Phi}^{-1} \mathcal{F}_{\Theta}^{-1} : \hat{f} \mapsto f$  in terms of the separate transforms with respect to  $\theta$  and  $\phi$ , respectively  $\mathcal{F}_{\Theta}^{-1} : \hat{f} \mapsto \tilde{f}$  (Eq. 33) and  $\mathcal{F}_{\Phi}^{-1} : \tilde{f} \mapsto f$  (Eq. 32). For identical reasons to the forward transform, we compute gradients of  $\mathcal{F}_{\Theta}^{-1}$  manually and gradients of  $\mathcal{F}_{\Phi}^{-1}$  by AD. In the following we therefore focus on the computation of the manual gradients of  $\mathcal{F}_{\Theta}^{-1}$ .

The elements of the Jacobian of the inverse transform  $\mathcal{F}_{\Theta}^{-1}$  read

$$\left(\partial \mathcal{F}_{\Theta}^{-1}(\tilde{f})\right)_{\theta}^{\ell} = \frac{\partial \tilde{f}_m(\theta)}{\partial \hat{f}_{\ell m}} = (-1)^s \sqrt{\frac{2\ell+1}{4\pi}} d_{m,-s}^{\ell}(\theta), \quad (64)$$

which is independent of the signal itself, i.e.  $\partial \mathcal{F}_{\Theta}^{-1}(\tilde{f}) = \partial \mathcal{F}_{\Theta}^{-1}$ . The JVP is then given by

$$\left(\partial \mathcal{F}_{\Theta}^{-1} \hat{v}\right)_{\theta} = \sum_{\ell} (-1)^s \sqrt{\frac{2\ell+1}{4\pi}} \hat{v}_{\ell m} d_{m,-s}^{\ell}(\theta), \quad (65)$$

which, as expected, is simply given by the transform itself, i.e.

$$\partial \mathcal{F}_{\Theta}^{-1} = \mathcal{F}_{\Theta}^{-1}. \quad (66)$$

The VJP is given by

$$\left(\partial \mathcal{F}_{\Theta}^{-T} \tilde{v}\right)_{\ell} = \sum_{\theta} (-1)^s \sqrt{\frac{2\ell+1}{4\pi}} \tilde{v}_m(\theta) d_{m,-s}^{\ell}(\theta), \quad (67)$$

which may be expressed by

$$\partial \mathcal{F}_{\Theta}^{-T} = \mathcal{F}_{\Theta} Q_{\Theta}^{-1} = \mathcal{F}_{\Theta}^{-H}, \quad (68)$$

where recall  $\mathcal{F}_{\Theta}$  is given by Eq. 30 and we have adopted the notation  $\mathcal{F}^{-H} = (\mathcal{F}^{-1})^H$ .

#### 5.4.3. Wigner transforms


The extension to compute gradients for Wigner transforms follows straightforwardly. As discussed in Sec. 2.3, Wigner transforms can be computed via a series of spherical harmonic transforms of varying spin, combined with a Fourier transform for  $\gamma$ . Following the hybrid differentiable approach discussed directly above, we compute explicit manual gradients for transforms with respect to  $\beta$  and adopt AD for transforms with respect to both  $\alpha$  and  $\gamma$ . This yields the best of both approaches, avoiding memory overheads associated with AD of the recursions of Wigner  $d$ -functions, while also avoiding the computational complexities associated with manual differentiation for different sampling schemes.

## 6. Software and evaluation

A key aim of this work is the development of a modern open-source Python library for the accelerated computation of generalised Fourier transforms on the sphere and rotation group and their derivatives. We target deployment on one or more modern hardware accelerators, such as GPUs and TPUs. By leveraging the parallelised computation of Wigner functions presented in Sec. 3 and the general algorithmic framework presented in Sec. 4, we ensure our algorithms are able to well-utilise hardware accelerators. Moreover, our algorithmic framework ensures we are largely agnostic to the choice of approach used to sample the sphere. Critically, to support the increasing interest surrounding differentiable programming, gradients must be able to be computed efficiently. By paying careful attention to the computation of gradients through a hybrid automatic and manual differentiation approach as described in Sec. 5, we avoid the memory overhead of full automatic differentiation while also avoiding the complexities of full manual differentiation. Finally, as with all good software packages, our library must be simple to use and easy to pick up.

A variety of differentiable programming and machine learning ecosystems exist in Python that could be considered for the implementation of our software, *e.g.* TensorFlow and PyTorch, both of which exhibit a mature collection of Python APIs. However, over time these libraries have become somewhat rigid due to the nature of heavily templated object oriented software focused on machine learning and hence the development of other bespoke functionality is hindered by, for example, significant boilerplate code. Consequently, we choose to develop our software in JAX, which

is a functional differentiable programming Python framework recently developed by Google [56]. JAX is particularly well-suited to scientific software development, where flexibility and rapid development are highly desirable.

With the above in mind we introduce S2FFT<sup>3</sup> , which is an open-source JAX package implementing the parallelised Wigner recursions presented in Sec. 3 and generalised Fourier transforms on the sphere and rotation group presented in Sec. 4 and their derivatives as presented in Sec. 5. The software code has been developed following software engineering best practices, exhibits an extensive test suite and is well documented. In the remainder of this section we briefly discuss the spherical sampling schemes supported and JAX hardware acceleration, before presenting benchmarking results in terms of precision and computational speed.

### 6.1. Sampling agnostic transforms

As discussed, our software is largely agnostic to the choice of spherical sampling considered. We simply require an isolatitudinal sampling, which most popular spherical samplings follow due to the considerable computational savings that it affords through a separation of variables, and corresponding quadrature weights. As a starting point Driscoll & Healy [40], McEwen & Wiaux (both MW and MWSS) [41, 6, 64], and HEALPix [43] samplings are supported. To simplify further open-sourced contributions, our code is designed with integration of future sampling schemes in mind, should the need arise. It is worth noting that once sample locations and associated quadrature weights are defined, all downstream functionality should work seamlessly.

### 6.2. Hardware acceleration

By design JAX functions can straightforwardly be deployed on hardware accelerators, *e.g.* GPU and TPUs. In fact, the ease with which existing Python codebases, particularly those which make frequent use of numpy, may be abstracted for GPU deployment is one of the primary advantages of JAX. In addition, we integrate the single program multiple data (SPMD) functionality accessible in JAX by the pmapi API. Specifically, for core transforms we provide an spmd flag that when set distributes expensive operations evenly across available devices. As we show when benchmarking (Sec. 6.3) we recover approximately linear acceleration in the number of devices considered. This distribution is straightforwardly engineered to balance the compute load across devices (see Sec. 4). Of course, for very low bandlimits the communication overhead incurred during distribution is significant. However, for even moderate to low bandlimits this communication is sub-dominant and significant computational savings are realised. Currently, the codebase provides tested and documented support for intra-node distribution (*i.e.* multiple GPUs on a single node), with inter-node distribution to be added in future (*i.e.* multiple nodes each with multiple GPUs).

### 6.3. Benchmarking

We present comprehensive benchmarking of the core transforms provided by our S2FFT software. First, we quantify the precision of our transforms by computing the round-trip error and find errors to be at the level of numerical precision for the sampling theorems considered. Second, we compute the average wall-clock computational speed of transforms, in both single and multiple GPU scenarios, for both our on-the-fly algorithms (computing Wigner  $d$ -functions on-the-fly) and precompute algorithm (computing and storing Wigner  $d$ -functions *a priori*). We compare computational time to the SSHT<sup>4</sup> [41] and SO3<sup>5</sup> [42] codes that compute spherical harmonic and Wigner transforms, respectively, for which underlying algorithms are implemented in C. An extensive comparison against a wide variety of existing spherical transform software codes is beyond the scope of this study (furthermore, while there are numerous spherical harmonic transforms codes there are relatively few codes capable of also computing Wigner transforms). In any case, comparison against other codes can be inferred from contemporary papers. In addition, all algorithms implemented in S2FFT are validated through extensive unit and regression tests.

<sup>3</sup><https://github.com/astro-informatics/s2fft>

<sup>4</sup><https://github.com/astro-informatics/ssht>

<sup>5</sup><https://github.com/astro-informatics/so3>

Spherical harmonic transform round-trip error						
L	On-the-fly transform			Precompute transform		
	MW	MWSS	DH	MW	MWSS	DH
8	$3.6 \times 10^{-16}$	$1.7 \times 10^{-16}$	$5.1 \times 10^{-16}$	$4.6 \times 10^{-16}$	$4.3 \times 10^{-16}$	$4.3 \times 10^{-16}$
16	$3.7 \times 10^{-16}$	$2.7 \times 10^{-16}$	$6.3 \times 10^{-16}$	$5.4 \times 10^{-16}$	$4.5 \times 10^{-16}$	$4.5 \times 10^{-16}$
32	$7.5 \times 10^{-16}$	$6.3 \times 10^{-16}$	$3.5 \times 10^{-16}$	$7.3 \times 10^{-16}$	$7.2 \times 10^{-16}$	$7.2 \times 10^{-16}$
64	$1.2 \times 10^{-15}$	$1.1 \times 10^{-15}$	$6.7 \times 10^{-16}$	$1.2 \times 10^{-15}$	$1.2 \times 10^{-15}$	$1.2 \times 10^{-15}$
128	$2.3 \times 10^{-15}$	$2.3 \times 10^{-15}$	$1.3 \times 10^{-15}$	$2.5 \times 10^{-15}$	$2.4 \times 10^{-15}$	$2.4 \times 10^{-15}$
256	$4.7 \times 10^{-15}$	$5.0 \times 10^{-15}$	$2.6 \times 10^{-15}$	$4.7 \times 10^{-15}$	$4.7 \times 10^{-15}$	$4.7 \times 10^{-15}$
512	$1.0 \times 10^{-14}$	$9.8 \times 10^{-15}$	$4.6 \times 10^{-15}$	$9.8 \times 10^{-15}$	$9.7 \times 10^{-15}$	$8.9 \times 10^{-15}$
1024	$1.9 \times 10^{-14}$	$1.9 \times 10^{-14}$	$9.3 \times 10^{-15}$	$1.7 \times 10^{-14}$	$1.5 \times 10^{-14}$	$1.2 \times 10^{-14}$
2048	$3.7 \times 10^{-14}$	$3.8 \times 10^{-14}$	$1.9 \times 10^{-14}$	—	—	—
4096	$7.5 \times 10^{-14}$	$7.7 \times 10^{-14}$	$3.8 \times 10^{-14}$	—	—	—
8192	$1.5 \times 10^{-13}$	$1.5 \times 10^{-13}$	$8.3 \times 10^{-14}$	—	—	—

Table 1: Round-trip error for scalar spherical harmonic transforms, *i.e.*  $\mathbb{E}[\|\hat{f} - \mathcal{F}\mathcal{F}^{-1}\hat{f}\|_2]$ . Random bandlimited functions are generated, which are passed sequentially through an inverse and then forward spherical harmonic transform. Errors are averaged across 10 realisations. We consider samplings of the sphere that exhibit a sampling theorem with exact spherical harmonic transforms. As expected, accuracy is therefore at the level of machine precision.

### 6.3.1. Precision

Natively, our software supports both 32-bit and 64-bit floating point precision, however due to the unstable nature of three term recursions [59] as outlined in Sec. 3 it is strongly recommended to adopt 64-bit precision. To mimic the most likely use-case, in our benchmarking we default to 64-bit precision, which is provided by JAX as an extended data type. Both Driscoll & Healy sampling [40] and both McEwen & Wiaux sampling schemes [41] afford sampling theorems on the sphere and therefore transforms based on such samplings are theoretically exact to machine precision.

The protocol for assessing accuracy in practice is to first generate a random set of bandlimited generalised Fourier coefficients  $\hat{f}$  from which a round-trip error metric  $\mathbb{E}[\|\hat{f} - \mathcal{F}\mathcal{F}^{-1}\hat{f}\|_2]$  is evaluated, where the expectation is computed across 10 randomised experiments. Note that for a Wigner transform one clearly should replace  $\mathcal{F}$  with  $\mathcal{W}$ . When considering Wigner transforms we enforced an azimuthal bandlimit of  $N = 5$ , corresponding to 9 equally spaced directions within each tangent plane, which is a typical use case. The results of these experiments are presented in Table 1 and Table 2 and also summarised in the bottom panels of Fig. 6 and Fig. 7. In all cases our transforms are exact to 64-bit machine precision. Note also that the gradient algorithms derived herein, and implemented in S2FFT, are validated against finite differences using the `check_grads` functionality provided within JAX.

In the era of high precision science, access to theoretically exact transforms is of great importance [73, 20, 21, 24]. The best way to perform high precision science is to avoid introducing numerical errors in the first place. For other applications one may wish to adopt 32-bit floating point precision, which can further accelerate compute and will half memory overhead. Such applications may include geometric machine learning on the sphere [74, 30, 4, 32], where 32-bit precision (or lower) is often adopted. When adopting 32-bit precision one may expect to recover commensurate errors, *i.e.* errors beginning at approximately  $\mathcal{O}(10^{-8})$  rather than  $\mathcal{O}(10^{-16})$ .

### 6.3.2. Computational speed

To benchmark computational speed our transforms are run over both a single and three NVIDIA A100 GPUs, each with 40GB on board memory. For comparison the existing SSHT and S03 C codes are run on a Xeon(R) E5-2650L v3 multithreaded CPU with a dedicated machine.

A summary of computational times are presented in Fig. 6 for a complex scalar spherical harmonic transform, with raw numerical results tabulated in Table 3. We provide results for both our on-the-fly and precompute transforms; for the latter we also quote memory requirements for the precompute. Our software handles functions of arbitrary spin, and therefore performance does not degrade with spin number, in contrast to other approaches. Results for our implementation of the Wigner transform are presented in Fig. 7, with raw numerical results provided in Table 4.

Unsurprisingly, for very low  $L$  we find that when running on GPUs our transforms are slightly slower than those provided by SSHT, which is primarily due to scheduling and memory communication overheads that can throttle exe-

Wigner transform round-trip error				
L	On-the-fly transform		Precompute transform	
	MW	MWSS	MW	MWSS
8	$1.6 \times 10^{-15}$	$1.3 \times 10^{-15}$	$1.3 \times 10^{-15}$	$1.2 \times 10^{-15}$
16	$1.2 \times 10^{-15}$	$1.0 \times 10^{-15}$	$1.1 \times 10^{-15}$	$1.1 \times 10^{-15}$
32	$1.3 \times 10^{-15}$	$1.2 \times 10^{-15}$	$1.5 \times 10^{-15}$	$1.2 \times 10^{-15}$
64	$1.5 \times 10^{-15}$	$1.4 \times 10^{-15}$	$1.8 \times 10^{-15}$	$1.4 \times 10^{-15}$
128	$2.2 \times 10^{-15}$	$2.0 \times 10^{-15}$	$3.6 \times 10^{-15}$	$2.0 \times 10^{-15}$
256	$2.9 \times 10^{-15}$	$2.8 \times 10^{-15}$	$4.2 \times 10^{-15}$	$2.9 \times 10^{-15}$
512	$4.2 \times 10^{-15}$	$4.3 \times 10^{-15}$	$6.4 \times 10^{-15}$	$4.2 \times 10^{-15}$
1024	$5.7 \times 10^{-15}$	$5.9 \times 10^{-15}$	—	—
2048	$8.1 \times 10^{-15}$	$8.2 \times 10^{-15}$	—	—
4096	$1.2 \times 10^{-14}$	$1.3 \times 10^{-14}$	—	—

Table 2: Round-trip error for Wigner transforms with azimuthal bandlimit  $N = 5$ , i.e.  $\mathbb{E}[\|\hat{f} - \mathcal{W}^* \mathcal{W}^{-1} \hat{f}\|_2]$ . Random bandlimited functions are generated, which are passed sequentially through an inverse and then forward Wigner transform. Errors are averaged across 10 realisations. We consider samplings of the rotation group that exhibit a sampling theorem with exact Wigner transforms. As expected, accuracy is therefore at the level of machine precision.

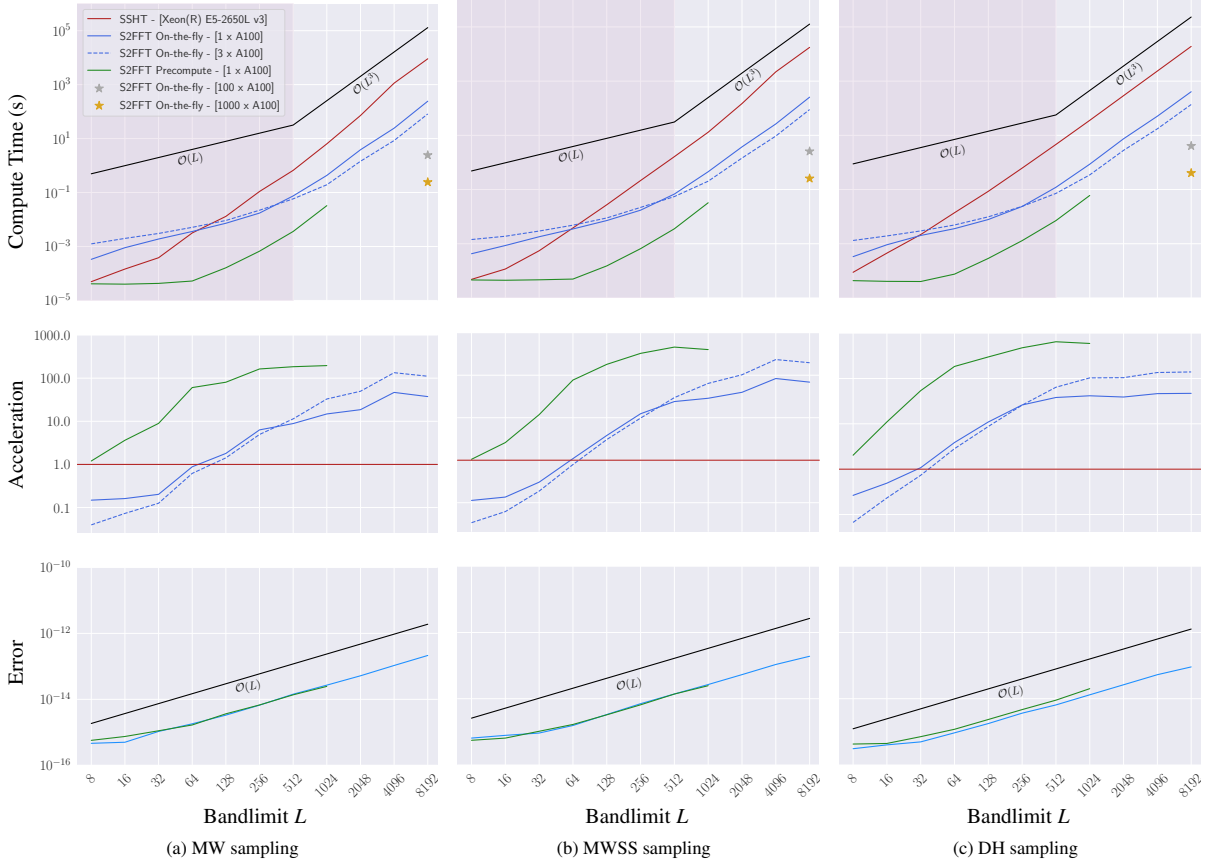


Fig. 6: Numerical benchmarking results for the spherical harmonic transforms implemented in S2FFT compared to the C implementation of SSHT. We consider samplings of the sphere that afford a sampling theorem and hence exact spherical harmonic transforms. S2FFT provides multiple operational modes: a precompute approach with faster compute at  $O(L^3)$  memory overhead and an on-the-fly transform with negligible memory overhead (see Sec. 4). **Top:** Average compute time required for a spherical harmonic transform for each operational modality. **Middle:** Relative acceleration of a given modality with respect to their SSHT counterpart. **Bottom:** Averaged round-trip numerical error. In each case we recover machine precision. We also empirically observe  $O(L)$  error scaling (in line with previous work [41]). The silver and gold stars represent the performance one may expect to realise should one distribute our on-the-fly algorithms across 100 and 1000 GPUs respectively.

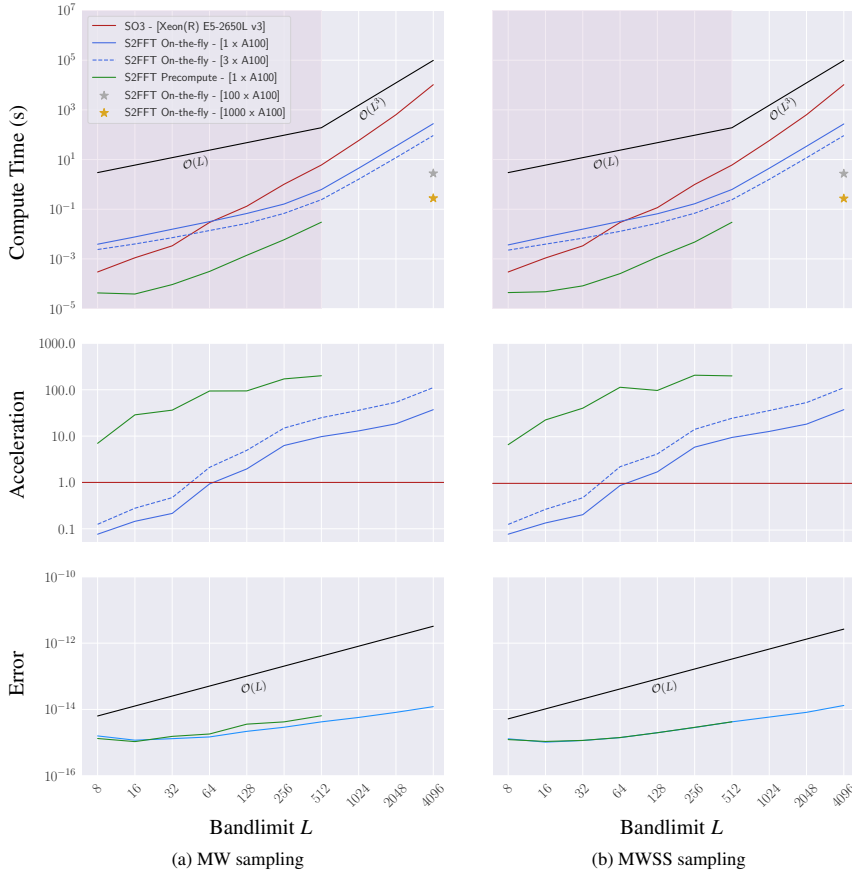


Fig. 7: Numerical benchmarking results for the Wigner transforms implemented S2FFT compared to the C implementation of S03. For this analysis we consider MW and MWSS [41] samplings of the rotation group. As these samplings afford sampling theorems on the rotation group, their corresponding Wigner transforms are exact. In these comparisons we fix the azimuthal bandlimit to  $N = 5$ , a typical use case corresponding to 9  $\gamma$  rotation angles. S2FFT provides multiple operational modes: a precompute approach with faster compute at  $O(NL^3)$  memory overhead and an on-the-fly transform with negligible memory overhead (see Sec. 4). **Top:** Average compute time required for a Wigner transform for each operational modality. **Middle:** Relative acceleration of a given modality with respect to their S03 counterpart. **Bottom:** Averaged round-trip numerical error. In each case we recover machine precision. We also empirically observe at most  $O(L)$  error scaling (in line with previous work [42]). The silver and gold stars represent the performance one may expect to realise should one distribute our on-the-fly algorithms across 100 and 1000 GPUs respectively.

cution on GPUs. However, for  $L \geq 64$  our transforms begin to achieve superior speed, reaching as much as a 234-fold and 459-fold acceleration for the on-the-fly and precompute transforms respectively. When running across multiple GPUs we see degraded performance at very low  $L$ , due to the additional inter-device communication overhead, however we asymptotically recover a further acceleration by a factor of the number of available devices. In practice for high bandlimits we achieve very close to optimal linear scaling with increasing number of GPUs due to the highly parallelised and balanced nature of our algorithms. For example, for spherical harmonic transforms at  $L = 8192$  when moving from one to three GPUs we achieve an additional acceleration of 2.96, 2.97, 2.87 for DH, MW and MWSS sampling respectively. For the Wigner transform at  $L = 4096$  and  $N = 5$  we achieve an additional acceleration of 2.99. With this in mind, provided access to sufficiently many GPUs our transforms exhibit an effective linear time complexity.

## 7. Conclusions

We have developed differentiable and accelerated algorithms to compute generalised Fourier transforms on the sphere and rotation group and their gradients. We present a recursive algorithm for the calculation of Wigner  $d$ -functions that is both stable to high harmonic degrees and extremely parallelisable. Leveraging this recursion we


Spherical harmonic transform computational time (ms)								
	$L$	Time SSHT	On-the-fly transform			Precompute transform		
			Time S2FFT	Speed-Up 1 $\times$ GPU	Speed-Up 3 $\times$ GPU	3 $\times$ GPU Ratio	Time S2FFT	Speed-Up 1 $\times$ GPU
DH Sampling	8	$8.8 \times 10^{-2}$	$3.3 \times 10^{-1}$	0.26	0.07	0.27	$4.4 \times 10^{-2}$	2.03
	16	$4.5 \times 10^{-1}$	$9.2 \times 10^{-1}$	0.49	0.23	0.47	$4.1 \times 10^{-2}$	11.0
	32	$2.2 \times 10^0$	$2.0 \times 10^0$	1.08	0.73	0.68	$4.0 \times 10^{-2}$	53.9
	64	$1.4 \times 10^1$	$3.6 \times 10^0$	3.87	2.83	0.73	$7.6 \times 10^{-2}$	184
	128	$8.6 \times 10^1$	$7.8 \times 10^0$	11.0	8.74	0.80	$2.9 \times 10^{-1}$	299
	256	$6.2 \times 10^2$	$2.4 \times 10^1$	26.1	25.8	0.99	$1.3 \times 10^0$	475
	512	$4.6 \times 10^3$	$1.2 \times 10^2$	38.1	64.1	1.68	$7.1 \times 10^0$	646
	1024	$3.6 \times 10^4$	$8.6 \times 10^2$	41.5	103	2.48	$6.0 \times 10^1$	592
	2048	$2.9 \times 10^5$	$7.5 \times 10^3$	39.0	104	2.67	—	—
	4096	$2.4 \times 10^6$	$5.1 \times 10^4$	46.4	135	2.91	—	—
8192	$1.9 \times 10^7$	$4.1 \times 10^5$	47.0	139	2.96	—	—	
MW Sampling	8	$5.0 \times 10^{-2}$	$3.3 \times 10^{-1}$	0.15	0.04	0.27	$4.1 \times 10^{-2}$	1.20
	16	$1.5 \times 10^{-1}$	$9.0 \times 10^{-1}$	0.16	0.07	0.44	$4.0 \times 10^{-2}$	3.63
	32	$3.8 \times 10^{-1}$	$1.9 \times 10^0$	0.20	0.13	0.65	$4.3 \times 10^{-2}$	8.94
	64	$3.2 \times 10^0$	$3.6 \times 10^0$	0.88	0.62	0.70	$5.2 \times 10^{-2}$	60.5
	128	$1.3 \times 10^1$	$7.3 \times 10^0$	1.80	1.42	0.79	$1.6 \times 10^{-1}$	80.6
	256	$1.1 \times 10^2$	$1.7 \times 10^1$	6.32	4.95	0.78	$6.7 \times 10^{-1}$	163
	512	$6.6 \times 10^2$	$7.5 \times 10^1$	8.85	11.4	1.29	$3.6 \times 10^0$	184
	1024	$6.4 \times 10^3$	$4.3 \times 10^2$	14.8	32.9	2.22	$3.3 \times 10^1$	196
	2048	$7.2 \times 10^4$	$3.9 \times 10^3$	18.6	49.7	2.67	—	—
	4096	$1.1 \times 10^6$	$2.4 \times 10^4$	46.7	134	2.87	—	—
8192	$9.1 \times 10^6$	$2.4 \times 10^5$	37.4	111	2.97	—	—	
MW/SS Sampling	8	$4.8 \times 10^{-2}$	$4.3 \times 10^{-1}$	0.11	0.03	0.27	$4.6 \times 10^{-2}$	1.05
	16	$1.2 \times 10^{-1}$	$8.6 \times 10^{-1}$	0.14	0.06	0.43	$4.5 \times 10^{-2}$	2.60
	32	$5.5 \times 10^{-1}$	$1.8 \times 10^0$	0.30	0.19	0.63	$4.7 \times 10^{-2}$	11.8
	64	$3.8 \times 10^0$	$3.5 \times 10^0$	1.09	0.79	0.72	$5.0 \times 10^{-2}$	76.9
	128	$2.7 \times 10^1$	$7.2 \times 10^0$	3.82	3.06	0.80	$1.5 \times 10^{-1}$	180
	256	$2.1 \times 10^2$	$1.7 \times 10^1$	12.4	9.80	0.79	$6.6 \times 10^{-1}$	327
	512	$1.6 \times 10^3$	$6.8 \times 10^1$	24.0	29.7	1.24	$3.6 \times 10^0$	459
	1024	$1.3 \times 10^4$	$4.5 \times 10^2$	28.7	64.2	2.24	$3.3 \times 10^1$	400
	2048	$1.5 \times 10^5$	$3.8 \times 10^3$	39.6	102	2.58	—	—
	4096	$2.2 \times 10^6$	$2.6 \times 10^4$	83.6	234	2.80	—	—
8192	$1.8 \times 10^7$	$2.6 \times 10^5$	68.7	197	2.87	—	—	

Table 3: Computational time of the S2FFT scalar spherical harmonic transform when distributed over a single and three NVIDIA A100 GPUs compared to SSHT running on a multithreaded Xeon(R) E5-2650L v3 CPU. We provide raw round-trip wall-clock time benchmarks and quote the relative acceleration provided by our algorithms. Note that the 3  $\times$  GPU ratio corresponds to the ratio of the acceleration realised when running S2FFT over three compared to one GPU. This ratio approaches the ideal ratio of three that corresponds to optimal linear scaling with increasing number of GPUs due to the highly parallelised and balanced nature of our algorithms.

Wigner transform computational time (ms)							
$L$	Time S03	On-the-fly transform				Precompute transform	
		Time S2FFT	Speed-Up 1 $\times$ GPU	Speed-Up 3 $\times$ GPU	3 $\times$ GPU Ratio	Time S2FFT	Speed-Up 1 $\times$ GPU
8	$3.0 \times 10^{-1}$	$3.9 \times 10^0$	0.08	0.12	1.5	$4.3 \times 10^{-2}$	6.98
16	$1.1 \times 10^0$	$7.7 \times 10^0$	0.14	0.28	2.0	$3.9 \times 10^{-2}$	28.7
32	$3.4 \times 10^0$	$1.6 \times 10^1$	0.21	0.47	2.24	$9.2 \times 10^{-2}$	36.3
64	$2.9 \times 10^1$	$3.2 \times 10^1$	0.93	2.11	2.27	$3.1 \times 10^{-1}$	93.9
128	$1.3 \times 10^2$	$6.8 \times 10^1$	1.96	4.93	2.52	$1.4 \times 10^0$	94.4
256	$1.0 \times 10^3$	$1.6 \times 10^2$	6.26	14.9	2.38	$5.9 \times 10^0$	171
512	$6.0 \times 10^3$	$6.2 \times 10^2$	9.73	24.9	2.56	$3.0 \times 10^1$	200
1024	$5.8 \times 10^4$	$4.5 \times 10^3$	12.9	36.1	2.80	–	–
2048	$6.4 \times 10^5$	$3.5 \times 10^4$	18.4	53.6	2.91	–	–
4096	$1.0 \times 10^7$	$2.8 \times 10^5$	37.1	111	2.99	–	–

Table 4: Computational time of the S2FFT Wigner transform with azimuthal bandlimit  $N = 5$  when distributed over a single and three NVIDIA A100 GPUs compared to S03 running on a multithreaded Xeon(R) E5-2650L v3 CPU. We provide raw round-trip wall-clock time benchmarks and quote the relative acceleration provided by our algorithms. Note that the 3  $\times$  GPU ratio corresponds to the ratio of the acceleration realised when running S2FFT over three compared to one GPU. This ratio approaches the ideal ratio of three that corresponds to optimal linear scaling with increasing number of GPUs due to the highly parallelised and balanced nature of our algorithms.

develop fast and exact spin spherical harmonic and Wigner transforms that are well-suited for the high throughput computing provided by modern hardware accelerators. We develop a hybrid differentiation approach so that gradients can be computed efficiently, avoiding the memory overhead of full automatic differentiation while also avoiding the complexities of full manual differentiation. Both forward- and reverse-mode differentiation is supported.

To support the above advances, we develop and release S2FFT , an open-source software library implemented in the JAX differentiable programming framework. Our transforms provide accelerated computation over one or multiple hardware accelerators (*i.e.* GPUs and TPUs), accelerated and memory-efficient gradient computation, and are largely agnostic to the spherical sampling considered, with support already for DH [Driscoll & Healy; 40], MW [McEwen & Wiaux; 41, 42], MWSS [McEwen & Wiaux symmetric sampling; 41, 6, 64], and HEALPix [43] sampling, with plans to add others in the near future. S2FFT supports both real and complex functions of arbitrary spin within a single easy to use API. As such, and in line with our design ethos, S2FFT is lightweight, straightforward to install, well tested and documented, and can be integrated into existing differentiable pipelines easily. For moderate resolution application we also provide a precompute mode, with enhanced computational speed at the cost of added memory requirements.

S2FFT is comprehensively benchmarked up to bandlimit  $L = 8192$  (higher bandlimits are certainly feasible but have not been considered to date due to the computational burden of further benchmarking). The transforms and gradient computations implemented are validated, where for sampling theorems that admit exact spherical transforms the error of a round-trip transform is observed to be of order of machine precision. In addition we benchmark the accelerated computation time of S2FFT against alternative C codes implementing spherical harmonic [41] and Wigner [42] transforms. Our transforms provide up to a 234-fold and 400-fold acceleration, respectively, for the on-the-fly and precompute transforms. Moreover, our spin spherical harmonic transforms are engineered to distribute compute across many accelerators. For high bandlimits we achieve very close to optimal linear scaling with increasing number of GPUs due to the highly parallelised and balanced nature of our algorithms, *i.e.* when considering 3 GPUs we realise a 3 $\times$  acceleration compared to running on a single GPU. Provided access to sufficiently many GPUs our transforms thus exhibit an unprecedented effective linear time complexity.

With researchers becoming increasingly interested in differentiable programming for scientific applications, we hope these foundational tools will be of great use in coming years. As just one example of potential applications, for global spherical modelling or analysis of the Earth’s climate or geophysical properties through data-driven or hybrid methods, sub-kilometre resolution and below is now feasible.

## CRediT authorship contribution statement

Author contributions are specified below, following the Contributor Roles Taxonomy (CRediT<sup>6</sup>).

**Matthew A. Price:** Conceptualisation, Methodology, Software, Investigation, Validation, Writing (Original Draft, Review & Editing)

**Jason D. McEwen:** Conceptualisation, Methodology, Software, Investigation, Validation, Writing (Original Draft, Review & Editing).

## Declaration of competing interest

The authors declare they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

All algorithms and transforms presented in this work are collectively released to the public in a professionally developed open-source software package S2FFT<sup>7</sup>. All results presented in this paper are entirely and transparently reproducible, provided access to appropriate computing hardware. This software is an ongoing development project and we strongly encourage community involvement going forward.

## Acknowledgments

The authors would like to thank Martin Reinecke for discussions relating to HEALPix spherical harmonic transforms and to François Lanusse for discussions relating to JAX development. We developed the software presented in this article in collaboration with Advanced Research Computing (ARC) at UCL through an open source software sustainability initiative. This work is supported by EPSRC (grant number EP/W007673/1).

## References

- [1] D. W. Ritchie, G. J. L. Kemp, Fast computation, rotation and comparison of low resolution spherical harmonic molecular surfaces, *J. Comput. Chem.* 20 (1999) 383–395.
- [2] C. H. Choi, J. Ivancic, M. S. Gordon, K. Ruedenberg, Rapid and stable determination of rotation matrices between spherical harmonics by direct recursion, *J. Chem. Phys.* 111 (1999) 8825–8831.
- [3] W. Boomsma, J. Frellsen, Spherical convolutions and their application in molecular modelling, *Advances in neural information processing systems* 30 (2017).
- [4] R. Kondor, Z. Lin, S. Trivedi, Clebsch–gordan nets: a fully fourier space spherical convolutional neural network, *Advances in Neural Information Processing Systems* 31 (2018).
- [5] D. S. Tuch, Q-ball imaging 52 (2004) 1358–1372.
- [6] A. Daducci, J. D. McEwen, D. V. D. Ville, J.-P. Thiran, Y. Wiaux, Harmonic analysis of spherical sampling in diffusion MRI, in: 19th Annual Meeting of the International Society for Magnetic Resonance in Medicine, 2011. [arXiv:arXiv:1106.0269](https://arxiv.org/abs/1106.0269).
- [7] J. D. McEwen, M. A. Price, Scale-discretised ridgelet transform on the sphere, in: 27th European Signal Processing Conference (EUSIPCO), 2019. doi:[10.23919/EUSIPCO.2019.8903034](https://doi.org/10.23919/EUSIPCO.2019.8903034). [arXiv:arXiv:1510.01595](https://arxiv.org/abs/1510.01595).
- [8] T. Goodwin-Allcock, J. D. McEwen, H. Z. R. Gray, P. Nachev, How can spherical CNNs benefit ML-based diffusion MRI parameter estimation?, in: *Computational Diffusion MRI*, 2022. doi:[10.1007/978-3-031-21206-2\\_9](https://doi.org/10.1007/978-3-031-21206-2_9). [arXiv:arXiv:2207.00572](https://arxiv.org/abs/2207.00572).
- [9] P. Audet, Directional wavelet analysis on the sphere: Application to gravity and topography of the terrestrial planets 116 (2011).
- [10] F. J. Simons, I. Loris, G. Nolet, I. C. Daubechies, S. Voronin, J. S. Judd, P. A. Vetter, J. Charléty, C. Vonesch, Solving or resolving global tomographic models with spherical wavelets, and the scale and sparsity of seismic heterogeneity 187 (2011) 969–988.
- [11] F. J. Simons, I. Loris, E. Brevdo, I. C. Daubechies, Wavelets and wavelet-like transforms on the sphere and their application to geophysical data inversion, in: *SPIE Wavelets and Sparsity XIV*, 2011. doi:[10.1117/12.892285](https://doi.org/10.1117/12.892285).
- [12] A. Marignier, J. D. McEwen, A. M. G. Ferreira, T. D. Kitching, Posterior sampling for inverse imaging problems on the sphere in seismology and cosmology, *Roy. Astron. Soc. Tech. & Instrum.* 2 (2022) 20–32.
- [13] K. S. Thorne, Multipole expansions of gravitational radiation, *Reviews of Modern Physics* 52 (1980) 299.
- [14] F. Beyer, B. Daszuta, J. Frauendiener, B. Whale, Numerical evolutions of fields on the 2-sphere using a spectral method based on spin-weighted spherical harmonics, *Classical and Quantum Gravity* 31 (2014) 075019.

<sup>6</sup><https://www.elsevier.com/authors/policies-and-guidelines/credit-author-statment>

<sup>7</sup><https://github.com/astro-informatics/s2fft>

- [15] M. Boyle, Transformations of asymptotic gravitational-wave data, *Physical Review D* 93 (2016) 084031.
- [16] C. G. R. Wallis, M. A. Price, J. D. McEwen, T. D. Kitching, B. Leistedt, A. Plouviez, Mapping dark matter on the celestial sphere with weak gravitational lensing, *Mon. Not. Roy. Astron. Soc.* 509 (2022) 4480–4497.
- [17] M. A. Price, J. D. McEwen, L. Pratley, T. D. Kitching, Sparse Bayesian mass-mapping with uncertainties: full-sky observations on the celestial sphere, *Mon. Not. Roy. Astron. Soc.* 500 (2021) 5436–5452.
- [18] A. Loureiro, L. Whiteway, E. Sellentin, J. S. Lafaerie, A. H. Jaffe, A. F. Heavens, Almanac: Weak lensing power spectra and map inference on the masked sphere, *arXiv preprint arXiv:2210.13260* (2022).
- [19] Z. Atkins, A. J. Duivenvoorden, W. R. Coulton, F. J. Qu, S. Aiola, E. Calabrese, G. E. Chesmore, S. K. Choi, M. J. Devlin, J. Dunkley, et al., The atacama cosmology telescope: Map-based noise simulations for dr6, *arXiv preprint arXiv:2303.04180* (2023).
- [20] Planck Collaboration I, Planck 2018 results. I. Overview, and the cosmological legacy of Planck, *Astron. & Astrophys.* 641 (2020).
- [21] S. Pires, V. Vandenbussche, V. Kansal, R. Bender, L. Blot, D. Bonino, A. Boucaud, J. Brinchmann, V. Capobianco, J. Carretero, et al., Euclid: Reconstruction of weak-lensing mass maps for non-gaussianity studies, *Astronomy & Astrophysics* 638 (2020) A141.
- [22] N. Jeffrey, M. Gatti, C. Chang, L. Whiteway, U. Demirbozan, A. Kovács, G. Pollina, D. Bacon, N. Hamaus, T. Kacprzak, et al., Dark energy survey year 3 results: Curved-sky weak lensing mass map reconstruction, *Monthly Notices of the Royal Astronomical Society* 505 (2021) 4626–4645.
- [23] Ž. Ivezić, S. M. Kahn, J. A. Tyson, B. Abel, E. Acosta, R. Allsman, D. Alonso, Y. AlSayyad, S. F. Anderson, J. Andrew, et al., Lsst: from science drivers to reference design and anticipated data products, *The Astrophysical Journal* 873 (2019) 111.
- [24] P. Amaro-Seoane, H. Audley, S. Babak, J. Baker, E. Barausse, P. Bender, E. Berti, P. Binetruy, M. Born, D. Bortoluzzi, et al., Laser interferometer space antenna, *arXiv preprint arXiv:1702.00786* (2017).
- [25] S. Dodelson, *Modern Cosmology*, Academic Press, Academic Press, 2003.
- [26] F. Dahlen, J. Tromp, *Theoretical Global Seismology*, Princeton University Press, 1998.
- [27] J. D. McEwen, M. P. Hobson, D. J. Mortlock, A. N. Lasenby, Fast directional continuous spherical wavelet transform algorithms, *IEEE Trans. Sig. Proc.* 55 (2007) 520–529.
- [28] J. D. McEwen, C. Durastanti, Y. Wiaux, Localisation of directional scale-discretised wavelets on the sphere, *Applied Comput. Harm. Anal.* 44 (2018) 59–88.
- [29] J. D. McEwen, B. Leistedt, M. Büttner, H. V. Peiris, Y. Wiaux, Directional spin wavelets on the sphere, *IEEE Trans. Sig. Proc.*, submitted (2015).
- [30] T. S. Cohen, M. Geiger, J. Köhler, M. Welling, Spherical cnns, *arXiv preprint arXiv:1801.10130* (2018).
- [31] C. Esteves, C. Allen-Blanchette, A. Makadia, K. Daniilidis, Learning SO(3) equivariant representations with spherical CNNs, in: *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. URL: <https://arxiv.org/abs/1711.06721>.
- [32] O. J. Cobb, C. G. R. Wallis, A. N. Mavor-Parker, A. Marignier, M. Price, M. d’Avezac, J. D. McEwen, Efficient generalized spherical CNNs, in: *International Conference on Learning Representations (ICLR)*, 2021. [arXiv:arXiv:2010.11661](https://arxiv.org/abs/2010.11661).
- [33] J. D. McEwen, C. G. R. Wallis, A. N. Mavor-Parker, Scattering networks on the sphere for scalable and rotationally equivariant spherical CNNs, in: *International Conference on Learning Representations (ICLR)*, 2022. [arXiv:arXiv:2102.02828](https://arxiv.org/abs/2102.02828).
- [34] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, *Nature Reviews Physics* 3 (2021) 422–440.
- [35] T. Liaudat, J.-L. Starck, M. Kilbinger, P.-A. Frugier, Rethinking the modeling of the instrumental response of telescopes with a differentiable optical model, in: *NeurIPS 2021 Machine Learning for Physical sciences workshop*, 2021. URL: <http://arxiv.org/abs/2111.12541>. [arXiv:2111.12541](https://arxiv.org/abs/2111.12541).
- [36] M. Mars, M. M. Betcke, J. D. McEwen, Learned interferometric imaging for the SPIDER instrument, *Roy. Astron. Soc. Tech. & Instrum.*, submitted (2023).
- [37] J. D. McEwen, T. I. Liaudat, M. A. Price, X. Cai, M. Pereyra, Proximal nested sampling with data-driven priors for physical scientists, in: *International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering*, 2023. [arXiv:arXiv:2307.00056](https://arxiv.org/abs/2307.00056).
- [38] J.-E. Campagne, F. Lanusse, J. Zuntz, A. Boucaud, S. Casas, M. Karamanis, D. Kirkby, D. Lanzieri, Y. Li, A. Peel, Jax-cosmo: An end-to-end differentiable and gpu accelerated cosmology library, *arXiv preprint arXiv:2302.05163* (2023).
- [39] D. Piras, A. S. Mancini, Cosmopower-jax: high-dimensional bayesian inference with differentiable cosmological emulators, *arXiv preprint arXiv:2305.06347* (2023).
- [40] J. R. Driscoll, D. M. J. Healy, Computing Fourier transforms and convolutions on the sphere, *Adv. Appl. Math.* 15 (1994) 202–250.
- [41] J. D. McEwen, Y. Wiaux, A novel sampling theorem on the sphere, *IEEE Trans. Sig. Proc.* 59 (2011) 5876–5887.
- [42] J. D. McEwen, M. Büttner, B. Leistedt, H. V. Peiris, Y. Wiaux, A novel sampling theorem on the rotation group, *IEEE Sig. Proc. Let.* 22 (2015) 2425–2429.
- [43] K. M. Górski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, M. Bartelmann, Healpix – a framework for high resolution discretization and fast analysis of data distributed on the sphere 622 (2005) 759–771.
- [44] N. Schaeffer, Efficient spherical harmonic transforms aimed at pseudospectral numerical simulations, *Geochemistry, Geophysics, Geosystems* 14 (2013) 751–758.
- [45] E. T. Newman, R. Penrose, Note on the Bondi-Metzner-Sachs group 7 (1966) 863–870.
- [46] J. N. Goldberg, A. J. Macfarlane, E. T. Newman, F. Rohrlich, E. C. G. Sudarshan, Spin- $s$  spherical harmonics and  $\bar{0}$  8 (1967) 2155–2161.
- [47] M. Kamionkowski, A. Kosowsky, A. Stebbins, Statistics of cosmic microwave background polarization D55 (1997) 7368–7388.
- [48] D. A. Varshalovich, A. N. Moskalev, V. K. Khersonskii, *Quantum theory of angular momentum*, World Scientific, Singapore, 1989.
- [49] Z. Khalid, S. Durrani, R. A. Kennedy, Y. Wiaux, J. D. McEwen, Gauss-legendre sampling on the rotation group, *IEEE Sig. Proc. Let.* 23 (2016) 207–211.
- [50] J. D. McEwen, P. Vanderghenst, Y. Wiaux, On the computation of directional scale-discretized wavelet transforms on the sphere, in: *Wavelets and Sparsity XV, SPIE international symposium on optics and photonics*, invited contribution, volume 8858, 2013. doi:[10.1117/12.2022889](https://doi.org/10.1117/12.2022889). [arXiv:arXiv:1308.5706](https://arxiv.org/abs/1308.5706).
- [51] B. Leistedt, J. D. McEwen, P. Vanderghenst, Y. Wiaux, S2LET: A code to perform fast wavelet analysis on the sphere, *Astron. & Astrophys.* 558 (2013) 1–9.

- [52] Z. Khalid, R. A. Kennedy, J. D. McEwen, An optimal-dimensionality sampling scheme on the sphere with fast spherical harmonic transforms, *IEEE Trans. Sig. Proc.* 62 (2014) 4597–4610.
- [53] D. M. J. Healy, D. Rockmore, P. J. Kostelec, S. S. B. Moore, FFTs for the 2-sphere – improvements and variations 9 (2003) 341–385.
- [54] W. Skukowsky, A quadrature formula over the sphere with application to high resolution spherical harmonic analysis 60 (1986) 1–14. 10.1007/BF02519350.
- [55] P. Kostelec, D. Rockmore, FFTs on the rotation group 14 (2008) 145–179.
- [56] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, JAX: composable transformations of Python+NumPy programs, 2018. URL: <http://github.com/google/jax>.
- [57] T. Risbo, Fourier transform summation of Legendre series and  $D$ -functions 70 (1996) 383–396.
- [58] S. Trapani, J. Navaza, Calculation of spherical harmonics and Wigner  $d$  functions by FFT. Applications to fast rotational matching in molecular replacement and implementation into *AMoRe*, *Acta Crystallographica Section A* 62 (2006) 262–269.
- [59] W. Gautschi, Computational aspects of three-term recurrence relations, *SIAM review* 9 (1967) 24–82.
- [60] G. Prézeau, M. Reinecke, Algorithm for the evaluation of reduced wigner matrices, *The Astrophysical Journal Supplement Series* 190 (2010) 267.
- [61] M. Reinecke, D. S. Seljebotn, Libsharp–spherical harmonic transforms revisited, *Astronomy & Astrophysics* 554 (2013) A112.
- [62] J. W. Cooley, J. W. Tukey, An algorithm for the machine calculation of complex fourier series 19 (1965) 297–301.
- [63] J. D. McEwen, G. Puy, J.-P. Thiran, P. Vandergheynst, D. V. D. Ville, Y. Wiaux, Sampling theorems and compressive sensing on the sphere, in: *Wavelets and Sparsity XIV, SPIE international symposium on optics and photonics*, invited contribution, volume 8138, 2011. doi:10.1117/12.893481. [arXiv:arXiv:1110.6297](https://arxiv.org/abs/1110.6297).
- [64] J. Ocampo, M. A. Price, J. D. McEwen, Scalable and equivariant spherical CNNs by discrete-continuous (DISCO) convolutions, in: *International Conference on Learning Representations (ICLR)*, 2023. [arXiv:arXiv:2209.13603](https://arxiv.org/abs/2209.13603).
- [65] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Fannery, *Numerical recipes in Fortran 77*, 2nd ed., Cambridge University Press, Cambridge, 1992.
- [66] T. S. Cohen, M. Geiger, J. Koehler, M. Welling, Convolutional networks for spherical signals, in: *ICML*, Springer, 2018. [arXiv:arXiv:1709.04893](https://arxiv.org/abs/1709.04893).
- [67] R. E. Wengert, A simple automatic derivative evaluation program, *Communications of the ACM* 7 (1964) 463–464.
- [68] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, Automatic differentiation in machine learning: a survey, *Journal of Machine Learning Research* 18 (2018) 1–43.
- [69] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, *nature* 323 (1986) 533–536.
- [70] J. D. McEwen, G. Puy, J.-P. Thiran, P. Vandergheynst, D. V. D. Ville, Y. Wiaux, Sparse image reconstruction on the sphere: implications of a new sampling theorem, *IEEE Trans. Image Proc.* 22 (2013) 2275–2285.
- [71] C. G. R. Wallis, Y. Wiaux, J. D. McEwen, Sparse image reconstruction on the sphere: analysis vs synthesis, *IEEE Trans. Image Proc.* 26 (2017) 5176–5187.
- [72] M. A. Price, L. Pratley, J. D. McEwen, Sparse image reconstruction on the sphere: a general approach with uncertainty quantification, *IEEE Trans. Image Proc.*, submitted (2021).
- [73] BICEP2 Collaboration, Detection of B-Mode Polarization at Degree Angular Scales by BICEP2 112 (2014) 241101.
- [74] M. M. Bronstein, J. Bruna, T. Cohen, P. Veličković, Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, *arXiv preprint arXiv:2104.13478* (2021).