The Astrophysical Journal Supplement Series, 222:17 (10pp), 2016 February doi:10.3847/0067-0049/222/2/17

SYMPIX: A SPHERICAL GRID FOR EFFICIENT SAMPLING OF ROTATIONALLY INVARIANT OPERATORS

D. S. Seljebotn and H. K. Eriksen
Institute of Theoretical Astrophysics, University of Oslo, P.O. Box 1029 Blindern, NO-0315 Oslo, Norway; d.s.seljebotn@astro.uio.no
Received 2015 April 20; accepted 2015 December 1; published 2016 February 10

ABSTRACT

We present SymPix, a special-purpose spherical grid optimized for efficiently sampling rotationally invariant linear operators. This grid is conceptually similar to the Gauss–Legendre (GL) grid, aligning sample points with iso-latitude rings located on Legendre polynomial zeros. Unlike the GL grid, however, the number of grid points per ring varies as a function of latitude, avoiding expensive oversampling near the poles and ensuring nearly equal sky area per grid point. The ratio between the number of grid points in two neighboring rings is required to be a low-order rational number (3, 2, 1, 4/3, 5/4, or 6/5) to maintain a high degree of symmetries. Our main motivation for this grid is to solve linear systems using multi-grid methods, and to construct efficient preconditioners through pixel-space sampling of the linear operator in question. As a benchmark and representative example, we compute a preconditioner for a linear system that involves the operator $\widehat{\boldsymbol{D}} + \widehat{\boldsymbol{B}}^T \boldsymbol{N}^{-1} \widehat{\boldsymbol{B}}$, where $\widehat{\boldsymbol{B}}$ and $\widehat{\boldsymbol{D}}$ may be described as both local and rotationally invariant operators, and $\boldsymbol{N}$ is diagonal in the pixel domain. For a bandwidth limit of $\ell_{\rm max} = 3000$, we find that our new SymPix implementation yields average speed-ups of 360 and 23 for $\widehat{\boldsymbol{B}}^T \boldsymbol{N}^{-1} \widehat{\boldsymbol{B}}$ and $\widehat{\boldsymbol{D}}$, respectively, compared with the previous state-of-the-art implementation.

Key words: cosmic background radiation – cosmology: miscellaneous – methods: data analysis – methods: numerical – methods: statistical

1. INTRODUCTION

Unlike the plane, it is impossible to construct a regular discretization of the sphere. Instead, every conceivable spherical grid comes with its own set of trade-offs, emphasizing one or more features at the cost of others. Thus, there is no such thing as a perfect spherical grid, but the optimal grid instead depends sensitively on the application under consideration.

In this paper, we will restrict our attention to high-resolution grids designed for fast and accurate spherical harmonic transforms (SHTs). In such cases, the primary consideration is that the grid must allow for efficient $\mathcal{O}(\ell_{\rm max}^3)$ SHTs, where $\ell_{\rm max}$ denotes the upper harmonic space bandwidth limit of the field in question, as opposed to the $\mathcal{O}(\ell_{\rm max}^4)$ scaling resulting from naive brute-force summation. This requires the use of FastFourier Transforms (FFTs) in the longitudinal direction, which in turn implies that (i) sample points must be placed on a set of iso-latitude rings, and (ii) sample points within each ring must be equidistant. However, there is still flexibility in choosing the latitude of each ring ($\theta_j \in [0, \pi]$), the number of grid points along each ring (n_j), and the initial offset of each ring ($\phi_{0,j}$).

Three popular spherical grids are the equiangular grid, the Gauss–Legendre (GL) grid (e.g., Doroshkevich et al. 2005), and HEALPix[1] (Górski et al. 2005). Of these, the equiangular grid is the most straightforward, simply defined by evenly spaced grid points (θ_i, ϕ_i) in both directions. This grid is typically used for geographical maps, and it is therefore also called a geographical grid.

Similarly, the standard GL grid has a constant number of grid points per ring. However, the ring latitudes θ_j are defined such that $P_{N_{\rm rings}}(\cos\theta_j) = 0$, where P_n is the Legendre polynomial of degree n. This simple modification allows efficient spherical harmonic analysis to machine precision, and the grid is thus optimized for spherical harmonics transforms.

Both of these grids suffer from a massive oversampling of the polar regions (θ close to 0 or π) compared to the equatorial region ($\theta \approx \pi/2$), and this renders them sub-optimal, and sometimes even useless, for certain practical applications. An important example is the solution of discretized and bandwidth-limited linear systems. If there is a large number of sample points within the correlation length implied by $\ell_{\rm max}$, the system becomes degenerate and numerically unstable. Grids with nearly constant pixel areas perform much better than grids with strongly varying pixel areas for these types of applications.

One example of such grids is HEALPix, which is short for "Hierarchical Equal Area and Latitude Pixelization." By construction, this grid has both constant area pixel area per pixel and grid points located on iso-latitude, so it is a good general-purpose grid. However, this generality comes at the cost of spherical harmonics precision, as well as a low level of internal pixel symmetries.

The latter point is particularly important for our applications. Consider a function of two grid points, $\hat{n}_1$ and $\hat{n}_2$, that is both localized and rotationally invariant,

$$f(\hat{n}_1, \hat{n}_2) = \begin{cases} f(\hat{n}_1 \cdot \hat{n}_2) & \text{if } \arccos(\hat{n}_1 \cdot \hat{n}_2) < k\Delta \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where Δ denotes the average distance between two neighboring grid points. Thus, f is assumed to be identically zero if the two grid points are separated by more than k grid units. In our applications, which employ multi-grid and/or preconditioning methods, we need to evaluate f for all relevant pairs ($\hat{n}_1$, $\hat{n}_2$). Furthermore, because f typically is computationally expensive, it is important to minimize the total number of function evaluations, and large speed-ups can be gained by exploiting symmetries and caching.

For HEALPix, f needs to be evaluated $\mathcal{O}(k^2 N_{\rm pix})$ times, because the angular distances between neighboring grid points are all different, up to a handful of overall symmetries. In contrast, for the equiangular and GL grids only $\mathcal{O}(k^2\sqrt{N_{\rm pix}})$

[1] http://healpix.sourceforge.net

evaluations are needed. Since the number of grid points is constant for every ring, we only need to evaluate f for the first grid point on every ring, accounting for all its neighbors, after which all function evaluations along the same ring will be given by symmetry.

In this paper, we construct a novel spherical grid called SymPix that combines the spherical harmonics transform precision of the GL grid with the nearly uniform sample point distances of HEALPix, while at the same time maintaining a high degree of symmetries within each ring, ensuring that fully sampling $f(\hat{n}_1 \cdot \hat{n}_2)$ scales as $\mathcal{O}(k^2\sqrt{N_{\rm pix}})$.

2. THE SYMPIX GRID

2.1. Ring Layout Basics

The main role of the SymPix grid is that of a supporting grid in internal multi-grid and/or preconditioning calculations, so maintaining high numerical precision is therefore essential. For this reason, we adopt the GL latitudinal ring layout as the basis of our grid. This provides support for both spherical harmonic *synthesis* (i.e., transforming from harmonic coefficients to pixel space) and *analysis* (transforming from pixel space to harmonic coefficients) to machine precision, by virtue of having an exact quadrature rule on the form

$$a_{\ell m} = \int_\Omega Y^*(\hat{n}) f(\hat{n}) d\Omega = \sum_i Y^*(\hat{n}_i) f(\hat{n}_i) w_i, \qquad (2)$$

where w_i is a set of quadrature weights. By placing rings exclusively on the zeros of the $\ell_{\rm max}$'th polynomial, one is guaranteed that $P_{\ell_{\rm max}+1}(\cos\theta_i) = 0$, and the discretized field is algebraically bandwidth-limited to harmonic modes with $\ell \leqslant \ell_{\rm max}$.

Next, we need to include enough sample points along each ring to fully resolve all spherical harmonic modes with $\ell \leqslant \ell_{\rm max}$. Formally speaking, this requires $2N_{\rm rings}$ grid points per ring. However, this requirement is somewhat counter-intuitive because it suggests massive oversampling of the polar regions compared to the equatorial region. And indeed, our intuition is correct: the spherical harmonic modes $Y_{\ell m}(\theta, \phi)$ are very close to zero in the polar regions for high ℓ and m, and these are the only modes that can cause high-frequency variation in the longitudinal direction. For this reason, the `libsharp` SHT package (Reinecke & Seljebotn 2013; Reinecke 2011) omits $Y_{\ell m}(\theta, \phi)$ whenever

$$\sqrt{m^2 - 2\,m\cos\theta} - \ell_{\rm max}\sin\theta > \max(100, 0.01\ell_{\rm max}), \qquad (3)$$

exploiting the fact that contributions from higher-ordered harmonics are numerically irrelevant. An explicit bound on the number of pixels required for machine precision was derived by Prézeau & Reinecke (2010), and Reinecke & Seljebotn (2013) used this to construct the *reduced GL grid*. Explicitly, for a given ring located at some latitude θ, Equation (3) defines the maximum m such that $Y_{\ell m}(\theta, \phi)$ does not vanish. The minimum number of pixels on that ring is then given by $2m + 1$, resulting in a longitudinal sample frequency that exceeds the Nyquist frequency.

2.2. Tiling

As discussed in Section 1, our primary usecase is evaluating a function $f(\hat{n}_1, \hat{n}_2)$ for all possible pairs $(\hat{n}_1, \hat{n}_2)$, but with the restriction that f is zero unless $\hat{n}_1$ and $\hat{n}_2$ are close together. To avoid unnecessary searches over vanishing pairs, we therefore partition our grid into a set of $k \times k$-sized tiles, where k is chosen such that $f(\hat{n}_1, \hat{n}_2) = 0$ unless $\hat{n}_1$ and $\hat{n}_2$ are either in the same tile or in two neighboring tiles. Thus, finding all relevant partner points for a given grid point simply amounts to a closest neighbor tile look-up. However, this also requires that the number of rings is divisible by k (letting $N_{\rm rings} > \ell_{\rm max} + 1$ if necessary), and that a set of k consecutive rings must have the same number of sample points. We will refer to each such set of k rings as a *band*.

2.3. Enforcing Symmetries

The main remaining step is to define the number of tiles per band. On the one hand, it must satisfy the minimum number of pixels given by Equation (3). On the other hand, it may be beneficial to increase it beyond this, in order to increase symmetries within and across bands. For instance, if we sample f from Equation (1) for all point pairs within a tile, the result can obviously be reused for all tiles in that band, since all between-point angular distances are conserved between tiles. Similarly, we can reuse results between neighboring tiles within the same band due to longitudinal symmetry.

In addition, we exploit the additional degrees of freedom in choosing the number of tiles to ensure symmetries with respect to latitudinally neighboring tiles. Specifically, we require that the number of tiles can increase from one band to the next only by a factor of exactly 3, 2, 1, 4/3, 5/4, or 6/5. Additionally, at least two bands in a row must have the same number of tiles, except for the polar bands. Finally, in order to avoid special cases we allow no equatorial ring (i.e., we insist that $N_{\rm rings}$ is an even number), and, purely conventionally, the location of the first grid point in a given ring is chosen to be half the pixel distance within that same ring. Together, these requirements ensure that the pattern of neighboring tiles repeats itself with a short period, and the total number of different cases evaluates scales as $\mathcal{O}(N_{\rm ring})$ rather than $\mathcal{O}(N_{\rm pix})$. We employ a dynamic programming algorithm to find the optimal number of tiles per band, subject to the constraints defined above, as detailed in Section 2.5. An example grid corresponding to $k = 2$ tiling is illustrated in Figure 1.

2.4. Memory Layout and Pixel Ordering

While the above constraints fully define the geometric properties of the SymPix grid, they do not imply a canonical memory layout or "pixel ordering." To fix this, we adopt two additional rules, both designed to maximize memory access efficiency and programming convenience.

First, the northern and southern hemispheres are band-wise interleaved. That is, we first list the northernmost polar band, followed by the southernmost polar band, followed by the second northern band and so on. The main advantage of this organization lies in convenient distributed programming across multiple computing nodes; interleaving the two hemispheres ensures that the same node can readily exploit north–south symmetries.

Second, grid points are latitudinally major-ordered within a given tile, i.e., the pixel ordering increases most rapidly along the θ direction. While the order within each tile could have been in any direction, this choice implies that pixel ordering is

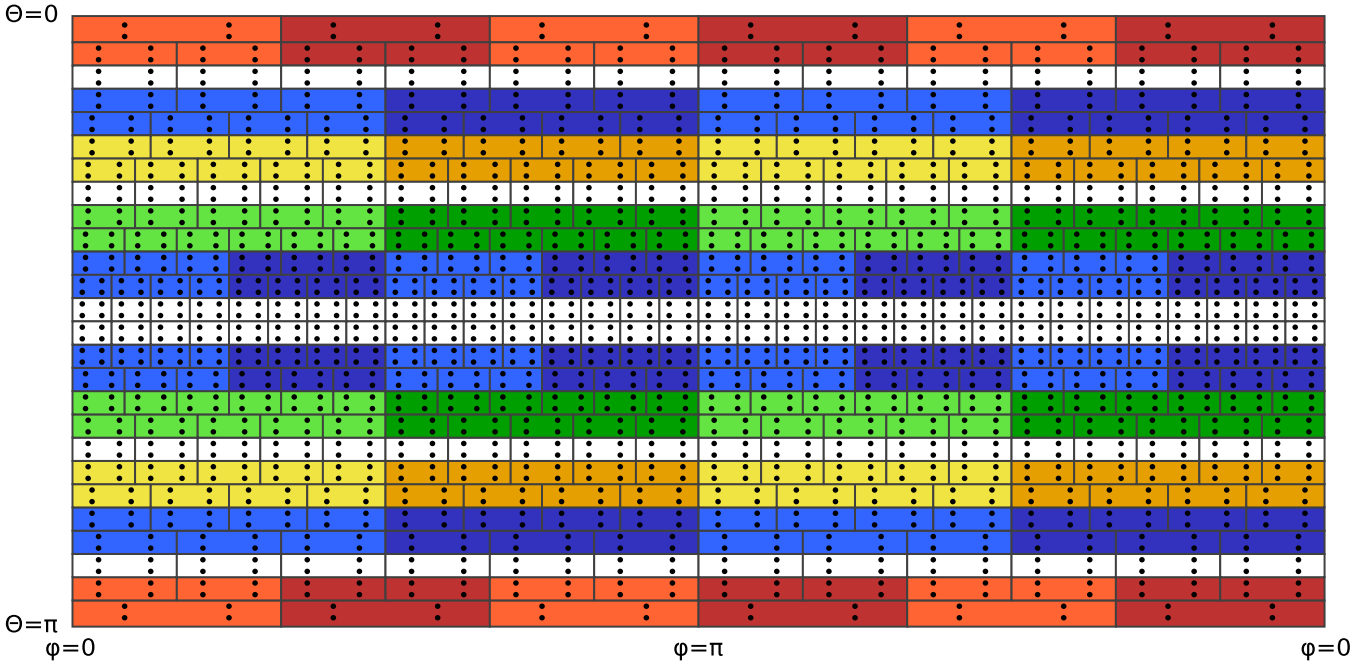

Figure 1. Geometric layout of SymPix sample points, implementing a cylindrical projection of the sphere. Each rectangle indicates a *tile* of (in this case) 2×2 sample points. For white tile-bands, the bands above and below have the same number of tiles, and angular distances between sample points in a given tile and sample points in the neighboring tiles are therefore constant throughout the band. Function evaluations depending only on angular distances may therefore be cached and reused. Colored tile-bands increment the number of tiles by a factor of 2 (red), $4/3$ (blue), $5/4$ (yellow), $6/5$ (green), and $4/3$ again (blue) toward the equator. For these bands, the neighboring tile relationship repeats itself (as indicated by shading), and there are still only a few cases that need to be computed and cached for each band.

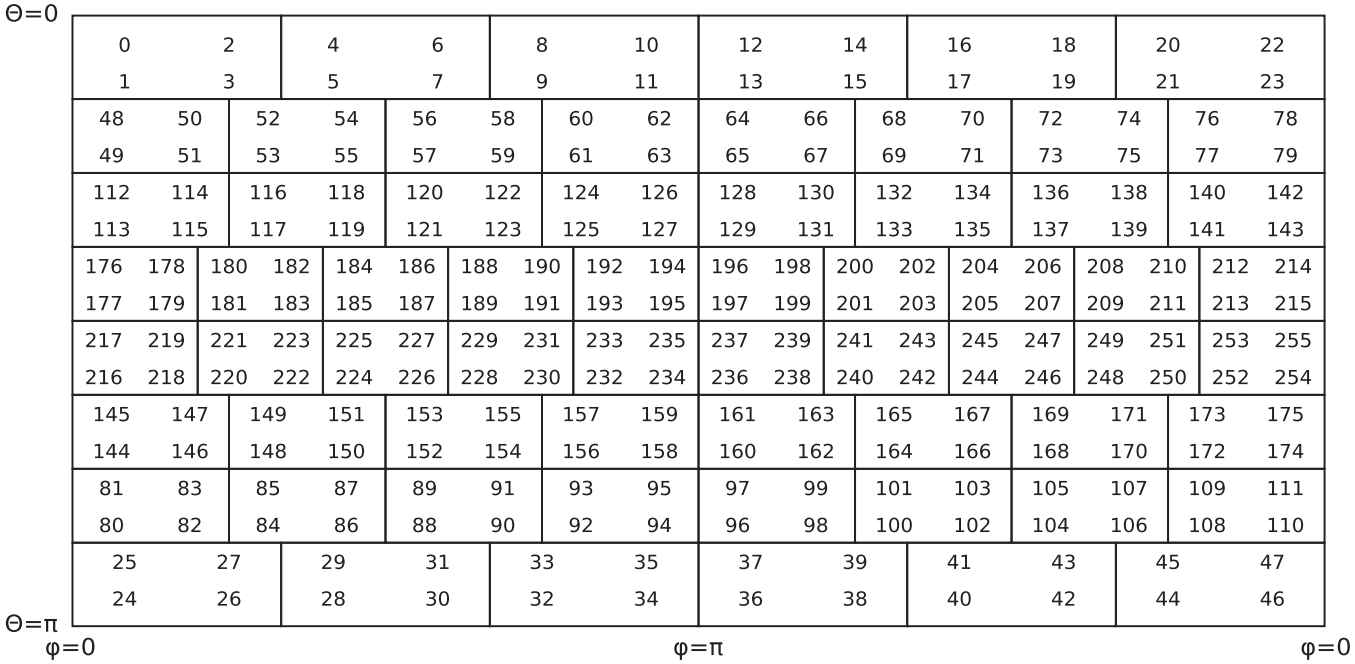

Figure 2. Memory ordering of SymPix sample points. Note that the resolution is lower than in Figure 1. Within each band the pixel order increases first latitudinally, i.e., along the θ direction. This ensures that access within the same tile is local in memory, and there are no discontinuities along each ring, which is convenient for SHTs. Additionally, to support efficient distributed programming, we interleave northern and southern bands, such that they naturally are assigned to the same node without explicit additional bookkeeping.

continuous across longitudinal tile borders, which is particularly convenient for SHTs.

Figure 2 provides an example of the resulting pixel ordering. Note that the resolution is lower than the corresponding illustration in Figure 1.

2.5. Grid Optimization

We end this section by describing the algorithm used to optimize the number of of tiles in each band, subject to the constraints defined in Section 2.3. We will only discuss the northern hemisphere, as the southern hemisphere is given directly by symmetry.

To initialize the algorithm, the user must provide a tile size k and a total number of rings $N_{\rm rings}$, where $N_{\rm rings}$ must be divisible by both 2 and k. The grid will be able to accurately represent fields that are band-limited at $\ell_{\rm max} = N_{\rm rings} - 1$. Together, these parameters specify the angular resolution of the grid, and correspond in principle to the HEALPix $N_{\rm side}$ parameter. We then number the bands by $i = 0, \ldots N_{\rm bands} - 1 \equiv N_{\rm rings}/(2k) - 1$, such that each band consists of k rings. We also define α_i to be the minimum number of tiles in each band subject to the constraint that the southmost ring within the band fulfills Equation (3).

Deriving the optimal SymPix grid is now equivalent to determining the number of tiles, T_i, for each band. For this optimization process we adopt the following cost function,

$$c(T_0, \ldots, T_{N_{\rm bands}-1}) \equiv \sum_i c_i(T_i) \equiv \sum_i (T_i - \alpha_i)^2, \quad (4)$$

which must be minimized, subject to

$$\frac{T_{i+1}}{T_i} \in \left\{ \frac{6}{5}, \frac{5}{4}, \frac{4}{3}, 1, 2, 3 \right\}. \quad (5)$$

Additionally, we initialize the recursion by defining T_0 as the smallest number larger than α_0 that is only a product of the factors 2, 3, and 5, and for computational speed we add the heuristic (or modification to the cost function) that $T_i < 3\alpha_i$, i.e., that no band should be over-pixelized by more than three times the Nyquist frequency.

The actual calculation is then a simple exercise in dynamic programming, as described in any standard text on algorithms (e.g., Cormen et al. 1989). Our implementation is summarized in Figure 3, which has a worst-case computational complexity of $\mathcal{O}(n\alpha_n) = \mathcal{O}(N_{\rm rings}^2) = \mathcal{O}(N_{\rm pix})$, and the same worst-case memory use. Due to the low computational complexity and the fact that the optimization only needs to be performed once per grid resolution, we do not present benchmarks for this operation; its computational cost is negligibly small for our purposes.

Optimal-SymPix-Grid:

Inputs:
- $\ell_{\rm max}$ – Band-limit of field to represent
- k – Tile size

Output:
- n – number of bands
- T_i – number of tiles in each band

Auxiliary:
- α_i – minimum number of tiles for band i
- $C_{i,t}$ – the cost of the best partial solution for bands 0 to i when assuming $T_i = t$
- $P_{i,t}$ – "previous-pointers"; when assuming $T_i = t$, the solution for bands 0 to i has $T_{i-1} = P_{i,t}$

Treat unassigned $C_{i,t}$ as ∞ and unassigned P_{i,T_0} as -1
$N_{\rm rings} \leftarrow \ell_{\rm max} + 1$ rounded up to next multiple of $2k$
$n \leftarrow N_{\rm rings}/2k$
Find θ_j for each ring j as for Gauss-Legendre grid
$T \leftarrow \max(100, \ell_{\rm max}/100)$
for each $i \in \{0, \ldots, n-1\}$:
  Find minimum m that satisfies Equation (3) for θ_{ik}
  $\alpha_i \leftarrow \lceil (2m+1)/k \rceil$
$T_0 \leftarrow \min(\{2^i 3^j 5^k \,|\, 2^i 3^j 5^k \geq \alpha_0, i \in \mathcal{N}, j \in \mathcal{N}, k \in \mathcal{N}\})$
$C_{i,T_0} \leftarrow (T_0 - \alpha_0)^2$
for each i **from** 1 **to** $n-1$:
  for each $t_{\rm prev}$ such that $C_{i-1,t_{\rm prev}} < \infty$:
    for each $x \in \{3, 2, 1, 4/3, 5/4, 6/5\}$ such that $xt_{\rm prev} \in \mathcal{N}$:
      $t \leftarrow xt_{\rm prev}$
      if $C_{i-1,t_{\rm prev}} + (\alpha_i - t)^2 < C_{i,t}$
          and $\alpha_i \leq t \leq 3\alpha_i$
          and ($P_{i-1,t_{\rm prev}} = t_{\rm prev}$ or $x = 1$):
        $C_{i,t} \leftarrow c + (\alpha_i - t)^2$
        $P_{i,t} \leftarrow t_{\rm prev}$
$T_{n-1} \leftarrow \mathrm{argmin}_t(C_{n,t})$
for each i **from** $n-2$ **to** 1:
  $T_i \leftarrow P_{i+1,T_{i+1}}$

Figure 3. Dynamic programming algorithm for optimizing the SymPix grid layout. In summary, the algorithm considers all possible solutions, and employs look-up tables of partial solutions for bands 0 to $i-1$ when considering band i. The condition $P_{i-1,t_{\rm prev}} = t_{\rm prev}$ ensures that at least two bands in a row have the same number of tiles, except (possibly) for the first two rows, $T_1 \neq T_0$.

3. BENCHMARKS AND COMPARISONS

Before considering specific applications, we first characterize the basic performance of the SymPix grid in terms of computational efficiency and numerical accuracy.

3.1. Geometric Efficiency

We start by quantifying the geometric efficiency of our grid, as characterized by the overall number of grid points and the pixel area uniformity. For these tests, we consider an example grid with $\ell_{\rm max} = 2000$ and $k = 4$, sufficient to discretize a spherical field with an angular resolution of 15′ FWHM. Running the algorithm summarized in Figure 3 with these input parameters yields a SymPix grid with 5.6×10^6 grid points.

In Figure 4 we compare the number of SymPix grid points per ring with the optimal number of points per ring used by the reduced GL grid (Reinecke & Seljebotn 2013). The ratio between the solid and dashed lines thus indicates the amount of longitudinal oversampling implied by the SymPix grid. Except for very close to the poles, where there are very few points in terms of absolute numbers, this ratio is never larger than 1.35.

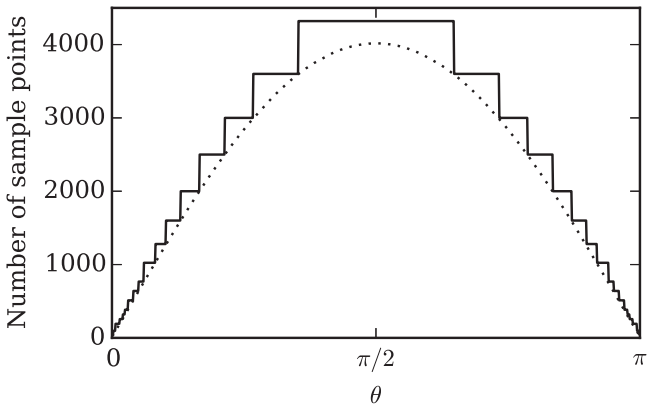

Figure 4. Number of SymPix grid points per ring as a function of latitude (solid line). The dotted line shows α_i, i.e., the same quantity for the reduced Gauss–Legendre grid (Reinecke & Seljebotn 2013).

A similar illustration is provided in Figure 5, where we plot the pixel area as a function of latitude, defining pixel borders strictly along longitudes and latitudes. The pixel area is given in units of the pixel area averaged over the full-sky, i.e., $4\pi/N_{\rm pix}$, such that a perfectly uniform pixelization, like HEALPix, corresponds to a constant value of unity. Overall, we see that the effective pixel areas vary at most by 20% relative to the average, except near the poles, where the normalized area may be as low as 0.1.

Figure 6 shows a histogram of normalized pixel areas, and we see that the vast majority of grid points have a normalized

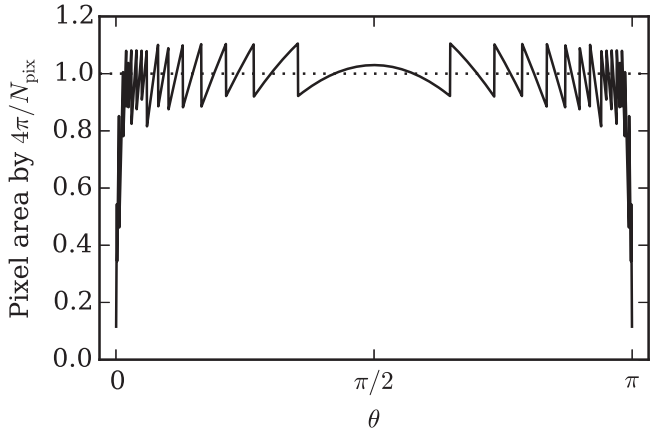

Figure 5. SymPix pixel area as a function of latitude in units of $4\pi/N_{\rm pix}$ (solid line). For the HEALPix grid, pixel areas are perfectly uniform (dotted line), while significant oversampling occurs close to the poles for the SymPix grid.

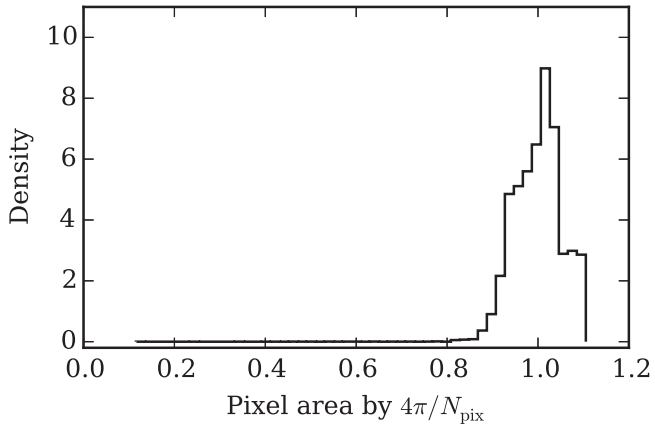

Figure 6. Histogram of normalized SymPix pixel areas. The tail extending below 0.8 corresponds to polar oversampling, and contains about 0.4% of the total number of pixels for this particular grid setup.

area between 0.9 and 1.1. The tail below 0.8 corresponds to the over-pixelized polar caps, and these contain only 0.4% of the total number of grid points for this particular example. Overall, the SymPix grid implies an oversampling of about 11% compared to the reduced GL grid, which is acceptable for our purposes.

3.2. Accuracy of Spherical Harmonic Quadrature

Next, we compare the numerical accuracy of spherical harmonics transforms as implemented on the SymPix, HEALPix, and reduced GL grids. This test is carried out through the following experiment.

1. We draw a fiducial signal $\boldsymbol{a} = \{a_{\ell m}\}$ in a spherical harmonic domain, band-limited by some $\ell_{\rm max}$. The spherical harmonics coefficients are drawn such that they correspond to an random isotropic and Gaussian field with a power spectrum (coefficient variance) $C_\ell \equiv 1/(\ell(\ell+1))$ for $\ell > 0$ and $C_0 \equiv 1$, i.e., the same overall properties as signals of interest for cosmic microwave background (CMB) analysis.
2. We project this signal onto the respective grid sample points by spherical harmonic synthesis.
3. We convert the real-space signal back to harmonic space through spherical harmonic analysis, including multipoles up to $\ell_{\rm max}$, to recover $\widehat{\boldsymbol{a}}$.
4. We repeat this procedure $N_{\rm sim}$ times, and summarize the results in terms of relative round-trip errors, $e_{\ell m}^{(i)} \equiv (\widehat{a}_{\ell m}^{(i)} - a_{\ell m}^{(i)})/\sqrt{C_\ell}$.

Before presenting the results, we note that no fundamental band-limit and/or resolution parameter $N_{\rm side}$ exist for HEALPix for a given angular resolution. For instance, changing the band-limit $\ell_{\rm max}$ will add/reduce aliasing for *all* scales. A quantitative head-to-head comparison at a given resolution is therefore difficult, as additional parameter tuning can affect the results. With this caveat in mind, in Table 1 we present results for three different band-limits, $\ell_{\rm max} = \{2.0, 2.5, 3.0\}N_{\rm side}$, with $N_{\rm side} = 256$, quoting both the maximum and mean errors as evaluated over all error coefficients $e_{\ell m}^{(i)}$. Each case includes $N_{\rm sim} = 100$ simulations, and the SymPix tile size is fixed at $k = 8$.

Starting with the highest-bandwidth case, $\ell_{\rm max} = 3N_{\rm side}$, we first note that the regular GL grid is the only grid that achieves overall machine precision, with a mean error of $\mathcal{O}(10^{-14})$ and a maximum error of $\mathcal{O}(10^{-12})$. For comparison, the corresponding mean and maximum SymPix errors are $\mathcal{O}(10^{-6})$ and $\mathcal{O}(10^{-2})$, respectively, while HEALPix achieves $\mathcal{O}(10^{-1})$ and $\mathcal{O}(1)$ for this high-bandwidth case. Reducing the band-limit to $\ell_{\rm max} = 2N_{\rm side}$ improves the latter by about two orders of magnitude. As already noted by Górski et al. (2005), the large difference between the average and maximum error is largely driven by the $m = 0$ modes, which integrate poorly on iso-latitude rings; both types of errors can, however, be reduced by iterative quadrature.

The statistics listed in Table 1 provide only a very coarse comparison point, because the round-trip errors are highly scale-dependent. In Figure 7 we therefore plot the error as a function of multipole, ℓ, choosing the SymPix and HEALPix band-limits such that the corresponding grids roughly match a HEALPix $N_{\rm side} = 256$ grid in terms of the total number of sample points. For SymPix, this corresponds to $\ell_{\rm max} = 735$, and for the GL grid it is $\ell_{\rm max} = 628$.

Starting with the GL grid (blue lines), we see that the error reaches machine precision up to the bandwidth limit; at higher multipoles no information is carried by the grid. In contrast, the SymPix grid reaches machine precision up to $\ell \approx 0.5\ell_{\rm max}$, while the error increases more smoothly at higher multipoles. However, even though the high-ℓ error increase is smooth, it is still exponential, and the mean and maximum statistics listed in Table 1 are therefore strongly dominated by the small-scale errors. Thus, by virtue of deriving its main geometric grid layout from the GL grid, we see that the numerical performance of the SymPix grid is excellent on large and intermediate angular scales, and the cost of its superior symmetry properties primarily comes in the form of sub-optimal small-scale residuals. For comparison, the HEALPix errors are roughly constant at $\mathcal{O}(10^{-4})$–$\mathcal{O}(10^{-2})$, and vary only weakly with angular scale. Note that in all cases the errors can be reduced by iteration techniques, essentially using least squares minimization to find the spherical harmonic signal with the least power that projects exactly to the map, and employing the result of spherical harmonic analysis as a preconditioner.

The large errors seen for the GL grid above $\ell_{\rm max}$ are due to undersampling, or equivalently, aliasing. In Figure 8 we study this effect directly by varying the spherical harmonics

Table 1
Comparison of Different Grids in Terms of the Number of Pixels and the Accuracy of Spherical Harmonic Analysis

$\ell_{\max}$	Grid	Parameter	$N_{\rm pix}$	$N_{\rm pix}/N_{\rm pix}^{\rm HEALPix}$	Max. Error	Mean Error	CPU Time for SHT (ms)
511	HEALPix	$N_{\rm side} = 256$	786 432	1.00	$2.1 \cdot 10^{-2}$	$2.9 \cdot 10^{-5}$	160
	SymPix	$\ell_{\max} = 511$	390 656	0.50	$7.8 \cdot 10^{-3}$	$8.1 \cdot 10^{-7}$	67
	Gauss–Legendre	$\ell_{\max} = 511$	524 288	0.67	$7.5 \cdot 10^{-13}$	$2.8 \cdot 10^{-14}$	66
639	HEALPix	$N_{\rm side} = 256$	786 432	1.00	$2.2 \cdot 10^{-1}$	$1.3 \cdot 10^{-3}$	219
	SymPix	$\ell_{\max} = 639$	591 232	0.75	$7.2 \cdot 10^{-3}$	$1.1 \cdot 10^{-6}$	118
	Gauss–Legendre	$\ell_{\max} = 639$	819 200	1.04	$1.2 \cdot 10^{-12}$	$3.2 \cdot 10^{-14}$	118
767	HEALPix	$N_{\rm side} = 256$	786 432	1.00	$1.6 \cdot 10^{0}$	$6.8 \cdot 10^{-2}$	287
	SymPix	$\ell_{\max} = 767$	838 656	1.07	$4.0 \cdot 10^{-2}$	$4.8 \cdot 10^{-6}$	188
	Gauss–Legendre	$\ell_{\max} = 767$	1 179 648	1.50	$1.0 \cdot 10^{-12}$	$3.8 \cdot 10^{-14}$	188

Note. The HEALPix resolution is kept constant at $N_{\rm side} = 256$, while the spherical harmonic band-limit varies over $\ell_{\max} = \{2.0, 2.5, 3.0\} N_{\rm side}$. The SymPix and Gauss–Legendre band-limits are identical to the spherical harmonic band-limit.

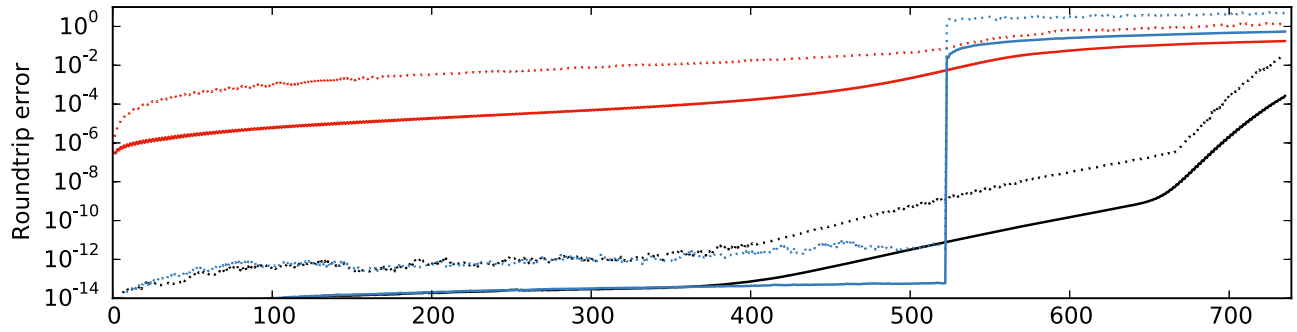

Figure 7. Spherical harmonic round-trip error as a function of multipole, summarized in terms of maximum (dotted lines) and mean (solid lines) errors, averaged over both harmonic quantum number m and $N_{\rm sim} = 100$ simulations. Black lines show results for a SymPix grid with $\ell_{\max} = 735$ and tile size 8; red lines show results for a HEALPix grid with $N_{\rm side} = 256$ and $\ell_{\max} = 735$; and blue lines show results for a regular Gauss–Legendre grid with $\ell_{\max} = 628$. All grids have roughly the same number of grid points, $N_{\rm pix} \approx 780,000$.

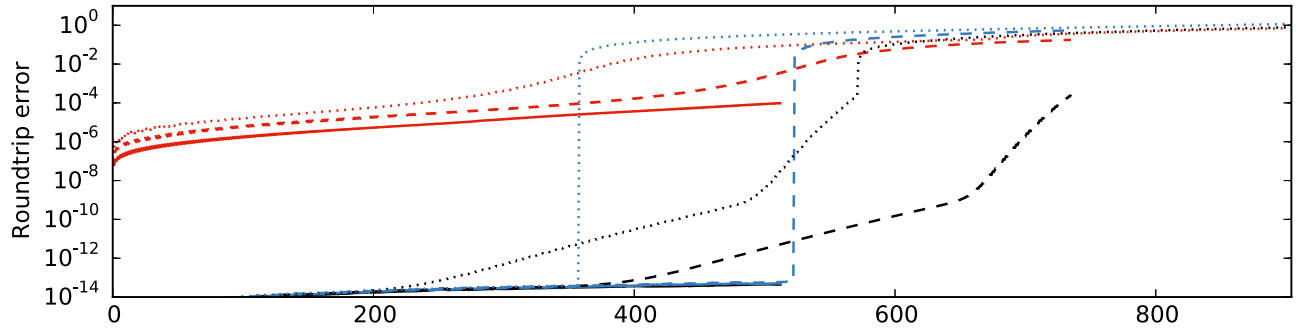

Figure 8. Error induced by undersampling (aliasing) as a function of multipole in terms of average errors, averaged over both harmonic quantum number m and $N_{\rm sim} = 100$ simulations. The experimental setup is the same as in Figure 7, but the spherical harmonic bandwidth limit varies between $\ell_{\max} = 512$ (solid), $\ell_{\max} = 735$ (dashed), and $\ell_{\max} = 900$ (dotted).

bandwidth limit between $\ell_{\max}^{\rm SH} = 512$, 735 and 900; note, however, that the actual grid resolution parameters are kept fixed at the above values, and the higher resolutions enforced here therefore no longer match the respective grid properties. Considering first the GL grid with a SHT band-limit of $\ell_{\max} = 512$, we see, as expected, that the errors reach machine precision at all scales. However, for the higher band-limits, $\ell_{\max} = 735$ and 900, both of which are higher than the grid resolution of $\ell_{\max}^{\rm grid} = 628$, the errors saturate at a multipole *below* the grid resolution. To be specific, the critical multipole is $2\ell_{\max}^{\rm grid} - \ell_{\max}^{\rm SH}$, corresponding to the well-known aliasing limit from standard Fourier theory. However, at lower multipoles no aliasing is observed for the GL grid, which implies that it is fully robust with respect to undersampling, given a known band-limit.

In comparison, the corresponding HEALPix errors are non-local, in the sense that increasing the spherical harmonics band-limit increases the errors at *all* angular scales: the dotted line ($\ell_{\max} = 900$) lies consistently higher than the dashed line ($\ell_{\max} = 735$), which in turn lies consistently higher than the solid line ($\ell_{\max} = 512$). The HEALPix grid is thus not robust against undersampling, and it is very important to choose a grid resolution appropriate for the bandwidth of the signal under consideration, which in several applications may imply oversampling the signal.

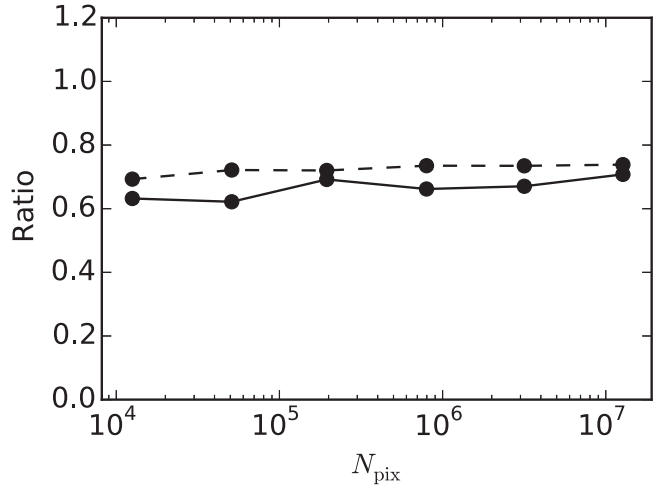

Figure 9. Comparison between spherical harmonic transforms cost as performed with SymPix and HEALPix as a function of $N_{\rm pix}$, plotted in terms of their ratios (black solid line). The dashed line shows the ratio between the number of grid point rings.

The SymPix grid performance lies, as expected, between those of GL and HEALPix. On large angular scales, it achieves numerical precision, while on small scales the aliasing increases exponentially with multipole, and eventually reaches similar levels as HEALPix.

3.3. Computational Speed of SHTs

Before ending this section, we compare the performance of the SymPix, HEALPix, and GL grids in terms of computational speed. The rightmost column in Table 1 lists the CPU time for each of the cases considered above in units of wall-clock milliseconds, while Figure 9 presents a head-to-head comparison of the SymPix and HEALPix grid performance as a function of $N_{\rm pix}$. All benchmarks were performed using `libsharp` on a single Intel Core i7 Q840 at 1.87 GHz (SSE2); for full details including CPU times in absolute numbers, we refer the interested reader to Reinecke & Seljebotn (2013).

Overall, SymPix perform similarly to the GL grid, and both execute about 30% faster than HEALPix. This latter difference may be explained by the fact that the HEALPix grid points form a zig-zag pattern in which every other ring is longitudinally shifted by half a pixel width. This implies a grid point organization that comprises about 30% more rings than GL and SymPix grids, which exhibit more regular longitudinal pixel organizations. This is relevant, because the computational complexity of SHTs scales as

$$\begin{aligned} C_{\rm SHT} &= \mathcal{O}(N_{\rm ring}\ell_{\rm max}^2) + \mathcal{O}\left(N_{\rm pix}\log\frac{N_{\rm pix}}{N_{\rm ring}}\right) \\ &= \mathcal{O}(\ell_{\rm max}^3) + \mathcal{O}(\ell_{\rm max}^2\log\ell_{\rm max}). \end{aligned} \tag{6}$$

The first term represents the cost of computing the associated Legendre polynomials for each ring, and dominates the second term, which accounts for evaluating FFTs along each ring. Thus, the number of grid points per ring is not critical for the overall speed of SHTs, while the total number of rings is.

In addition, by construction, SymPix grids have rings with pixel numbers that are only products of 2, 3, and/or 5, which ensures efficient FFTs. In contrast, many HEALPix rings have pixel numbers that includes large primes, and therefore the Bluestein algorithm must be employed for these. This effect is more important for lower-resolution grids, for which the cost of FFTs is relatively higher.

4. APPLICATIONS

We now turn our attention to practical applications, and in particular to the construction of efficient preconditioners. Before doing that, however, we consider a simpler application, namely real-space convolution, in order to build up intuition regarding the relevant operations. We emphasize that the purpose of this preliminary discussion is not to provide a real-world alternative to SHTs, or the methods presented by Elsner & Wandelt (2011) and Sutter et al. (2012) for such convolutions, but simply to quantify the computational efficiency of the SymPix grid on a simple and intuitive application.

4.1. Spherical Convolution

The convolution of a spherical image f with a kernel b is given by the spherical surface integral

$$g(\hat{n}) = \int_{4\pi} b(\hat{n},\hat{m}) f(\hat{m}) d\Omega_{\hat{m}}. \tag{7}$$

In our case we assume an azimuthally symmetric kernel, and $b(\hat{n},\hat{m})$ therefore depends only on the distance between $\hat{n}$ and $\hat{m}$, such that

$$g(\hat{n}) = \int_{4\pi} b(\hat{n}\cdot\hat{m}) f(\hat{m}) d\Omega_{\hat{m}}. \tag{8}$$

This integral is most commonly performed in a spherical harmonic domain, turning full-sky convolution into coefficient-wise multiplication with a corresponding transfer function, b_ℓ, which is given by the Legendre transform of $b(\hat{n}\cdot\hat{m})$. These computations are dominated by the SHTs, and therefore have a computational scaling of $\mathcal{O}(N_{\rm pix}^{3/2}) = \mathcal{O}(\ell_{\rm max}^3)$.

If b is spatially narrow compared to the required pixelization, as is usually the case, one could instead consider the pixel-domain convolution by evaluating

$$g(\hat{n}_i) = \sum_{j=1}^{N_{\rm pix}} b(\hat{n}_i\cdot\hat{n}_j) f(\hat{n}_j), \tag{9}$$

where the convolution kernel reads

$$b(x) = \sum_{\ell=0}^{\ell_{\rm max}} \frac{2\ell+1}{4\pi} b_\ell P_\ell(x). \tag{10}$$

One would then make the approximation that $b(\hat{n}_i\cdot\hat{n}_j) = 0$ whenever sample points i and j are more than k sample point distances apart, as discussed in Section 1.

For HEALPix, almost all sample point distances are different, and b must therefore be evaluated $\mathcal{O}(N_{\rm pix}\,k^2)$ times. The computational complexity of pixel-domain convolution on the HEALPix grid therefore scales as $\mathcal{O}(N_{\rm pix}\,k^2\,\ell_{\rm max}) = \mathcal{O}(k^2\,\ell_{\rm max}^3)$, which is clearly inferior to the harmonic approach both in terms of speed and accuracy. With SymPix, however, the large number of symmetries allows us to reduce the computational complexity to $\mathcal{O}(k^2\,N_{\rm pix} + \sqrt{N_{\rm pix}}\,\ell_{\rm max}) = \mathcal{O}(k^2\,N_{\rm pix})$: one simply needs to choose a tile size k such that only sample point pairs within a

Table 2
CPU Time and Theoretical Speed-up for Evaluating $b(\hat{n}\cdot\hat{m})$

$\ell_{\max}$	CPU Time (s)	Speed-up (factor)
3000	9.8	732
1500	3.6	335
750	1.4	149
375	0.74	70
188	0.50	26
100	0.31	14

Note. We have approximated $b(\hat{n}\cdot\hat{m}) = 0$ whenever $\hat{n}$ and $\hat{m}$ are not in neighboring tiles. The third column shows the number of non-zero $b(\hat{n}\cdot\hat{m})$, which scales as $O(k^2 N_{\rm pix})$, divided by the number of elements we had to compute when making use of the SymPix symmetries, which scales as $O(k^2\sqrt{N_{\rm pix}})$. In this example we have chosen $k = 8$.

tile and between neighboring tiles must be considered. Then for, each band of k rings, $b(\hat{n}\cdot\hat{m})$ only needs to be evaluated for the first few tiles of the band, as other distances within the same band will be identical within the remainder of the band.

The speed-ups for evaluating all necessary $b(\hat{n}\cdot\hat{m})$, when approximating $b(\hat{n}\cdot\hat{m}) = 0$ whenever $\hat{n}$ and $\hat{m}$ are not in neighboring tiles, are given in Table 2. In addition to scaling better than the $\mathcal{O}(N_{\rm pix}^{3/2})$ SHTs, this approach should also be easier to parallelize and implement efficiently on a GPU.

Note that yet another method for spherical convolution with a symmetric kernel has been implemented in the ARKCoS code (Elsner & Wandelt 2011; Sutter et al. 2012), with a computational scaling of $\mathcal{O}(k\,\ell_{\max}^2\,\log\ell_{\max}) = \mathcal{O}(k\,N_{\rm pix}\,\log N_{\rm pix})$. Whether a SymPix-based convolution would improve relative to their work for relevant resolution parameters and accuracy requirements remains to be explored.

4.2. Preconditioner Construction for Linear Systems

Finally, we are in the position to discuss the application of the SymPix grid to our main usecase, namely for solving linear systems involving rotationally invariant operators in a pixel domain, either through multi-grid methods or constructing efficient preconditioners. The simplest example of such a system is

$$\boldsymbol{Y}\boldsymbol{B}\boldsymbol{Y}^T\boldsymbol{x} = \boldsymbol{b}, \tag{11}$$

where $\boldsymbol{Y}$, as usual, is the matrix corresponding to spherical harmonic synthesis and $\boldsymbol{B}$ is a diagonal matrix in a spherical harmonic domain, $B_{\ell m,\ell' m'} = b_\ell\,\delta_{\ell,\ell'}\,\delta_{mm'}$. The product $\boldsymbol{Y}\boldsymbol{B}\boldsymbol{Y}^T$ is a pixel-domain operator with strong spatial couplings within the correlation length implied by b. Of course, this particular system could have been trivially solved by converting to a spherical harmonic domain, which would diagonalize the coefficient matrix. However, if there are more terms in the operator, this is no longer possible, and iterative solvers like Conjugate Gradients or multi-level algorithms are needed. In these cases SymPix is useful to construct preconditioners or smoothers.

Our own main interest lies in drawing constrained Gaussian realizations of the CMB sky by using a multi-level solver (Seljebotn et al. 2014). This maybe performed by solving the following linear system (Eriksen et al. 2004; Jewell et al. 2004; Wandelt et al. 2004),

$$\boldsymbol{Y}_1(\boldsymbol{D} + \boldsymbol{B}\boldsymbol{Y}_{\rm obs}\boldsymbol{N}^{-1}\boldsymbol{Y}_{\rm obs}^T\boldsymbol{B})\boldsymbol{Y}_1^T\boldsymbol{x} = \boldsymbol{r}, \tag{12}$$

where $\boldsymbol{D}$ and $\boldsymbol{B}$ are diagonal matrices in a spherical harmonic domain, characterized by transfer functions d_ℓ and b_ℓ, $\boldsymbol{N}^{-1}$ is a diagonal (inverse noise covariance) matrix in a pixel domain, pixelized on some external grid θ_i, and $\boldsymbol{r}$ is a stochastic term that depends on the data set in question.

Two different spherical grids are involved in a system. First, the outermost spherical harmonics transform, $\boldsymbol{Y}_1$, denotes synthesis to a grid of our own choosing. We will use a SymPix grid of resolution $\ell_{\max}$ for this operator in the following. The inner transform, $\boldsymbol{Y}_{\rm obs}$, is determined by some external experiment, and is thus not flexible. Here we will assume that this operator is defined on a full-sky HEALPix grid of $N_{\rm side} = 2048$, typical for the CMB maps published by the *Planck* experiment (Planck Collaboration 2015).

Of course, from the viewpoint of the overall linear system, the details of any individual operator are irrelevant, and the only crucial point is that the combined operator remains the same. In order to speed up the calculations through the use of symmetries, we therefore substitute the innermost HEALPix-based noise covariance matrix product with a corresponding SymPix-based product,

$$\boldsymbol{Y}_{\rm obs}\boldsymbol{N}^{-1}\boldsymbol{Y}_{\rm obs}^T = \boldsymbol{Y}_2\boldsymbol{N}_2^{-1}\boldsymbol{Y}_2^T, \tag{13}$$

where $\boldsymbol{Y}_2$ denotes an auxiliary SymPix grid; note that this does not need to be the same as $\boldsymbol{Y}_1$, but its resolution can be adjusted to trade numerical precision for computational speed. As shown by Seljebotn et al. (2014), Equation (13) holds true if $\boldsymbol{N}_2$ is constructed from

$$\boldsymbol{\theta}_2 = \boldsymbol{W}_2\boldsymbol{Y}_2\boldsymbol{Y}_{\rm obs}^T\boldsymbol{\theta}, \tag{14}$$

in the same way as $\boldsymbol{N}$ is constructed from θ. In this latter expression, $\boldsymbol{W}_2$ is a diagonal matrix containing the quadrature weights used in the spherical harmonic analysis of the target grid, while $\boldsymbol{Y}_{\rm obs}^T$ lacks the ring weights one normally uses in spherical harmonic analysis. Note that this operation is in fact the opposite procedure compared to naive resampling, which would be written as $\boldsymbol{Y}_2\boldsymbol{Y}_{\rm obs}^T\boldsymbol{W}_{\rm obs}$ in our notation. For full details, we refer the interested reader to Seljebotn et al. (2014).

The precision of Equation (13) depends on the relative band-limits of $\boldsymbol{Y}_1$, $\boldsymbol{Y}_2$, and $\boldsymbol{Y}_{\rm obs}$. For instance, choosing $\ell_{\max}$ for $\boldsymbol{Y}_2$ and $\boldsymbol{Y}_{\rm obs}$ to be twice that of $\boldsymbol{Y}_1$ yields a numerical precision of $\mathcal{O}(10^{-10})$. Increasing these to four times that of $\boldsymbol{Y}_1$ results in an accuracy of $\mathcal{O}(10^{-14})$, whereas reducing it to only one, such that $\boldsymbol{Y}_1 = \boldsymbol{Y}_2$, gives an accuracy of $\mathcal{O}(10^{-2})$. Even the latter may be acceptable for preconditioning purposes.

In order to derive an approximation to the full coefficient matrix defined by Equation (12), we first rewrite the system as

$$\widehat{\boldsymbol{D}} + \widehat{\boldsymbol{B}}^T\boldsymbol{N}_2^{-1}\widehat{\boldsymbol{B}}\boldsymbol{x} = \boldsymbol{r}, \tag{15}$$

where

$$\widehat{\boldsymbol{D}} = \boldsymbol{Y}_1\boldsymbol{D}\boldsymbol{Y}_1^T \quad \text{and} \quad \widehat{\boldsymbol{B}} = \boldsymbol{Y}_2\boldsymbol{B}\boldsymbol{Y}_1^T. \tag{16}$$

We now introduce the approximation that $\widehat{D}_{ij} = 0$ and $\widehat{B}_{ij} = 0$ whenever two sample points i and j are not in the same or neighboring tiles, as per the SymPix organization. The non-zero elements (i.e., the "local" part) of $\widehat{\boldsymbol{D}}$ and $\widehat{\boldsymbol{B}}$ are evaluated by Equation (10), at a cost of $\mathcal{O}(\ell_{\max})$ operations per matrix

Table 3
CPU Time for Constructing Preconditioner

$\ell_{\max}$	Naive (CPU min)	SymPix (CPU min)	Speed-up
Evalution of $\widehat{\boldsymbol{B}}^T \boldsymbol{N}^{-1}\widehat{\boldsymbol{B}}$			
3000	727	5.4	130
1500	509	1.4	360
750	340	0.37	920
375	230	0.11	2 100
188	452	0.035	13 000
100	363	0.027	13 000
Sum	2 621	7.3	360
Evalution of $\widehat{\boldsymbol{D}}$			
3000	85	3.3	26
1500	15	0.83	18
750	2.4	0.22	11
375	0.36	0.07	5
188	0.05	0.02	3
100	0.01	0.01	1
Sum	103	4.5	23

Note. The top section lists the CPU time for preconditioner calculations that depend only on data geometry (mask, beam, noise characterization), while the bottom section lists the corresponding CPU time for calculations that depend on d_ℓ, which in CMB applications typically corresponds to an angular power spectrum, C_ℓ. The second column is copied directly from Seljebotn et al. (2014), which does not employ any symmetries. The third row shows similar results using SymPix, while the fourth column shows the ratio between the two.

element. However, as discussed in Section 4.1, evaluating all required elements for a SymPix grid scales as $\mathcal{O}(k^2\sqrt{N_{\rm pix}})$, as opposed to $\mathcal{O}(k^2 N_{\rm pix})$ for less symmetric grids.

These calculations constitute essential components of the pre-computation step of the multi-grid solver presented by Seljebotn et al. (2014). In that paper, all evaluations were performed without employing any symmetries, with a computational scaling of $\mathcal{O}(\ell_{\max} k^2 N_{\rm pix})$ as discussed above. Their Table 2 summarizes the resulting computational costs in units of CPU minutes. Here we repeat those calculations, adopting the exact same overall parameters, facilitating a one-to-one comparison, but we employ SymPix for intermediate calculations. The results are summarized in Table 3, in which the second column is copied directly from Seljebotn et al. (2014), and the third column shows the new SymPix results. The fourth column shows the ratio between the two.

Clearly, the net gains achieved by the SymPix grid vary with resolution. For the high-resolution levels the speed-up is driven by symmetries drastically reducing the time taken to evaluate $\widehat{\boldsymbol{B}}$. The theoretical speed-up of 732 times for evaluating $\widehat{\boldsymbol{B}}$ at $\ell_{\max} = 3000$, found in Table 2, is reduced to 130 and 26 for $\widehat{\boldsymbol{B}}^T\boldsymbol{N}^{-1}\widehat{\boldsymbol{B}}$ and $\widehat{\boldsymbol{D}}$, respectively. This is due to work that was previously unimportant now dominating the computation.

As already noted, the HEALPix grid also exhibits a handful of internal symmetries that could have been exploited in a similar manner to reduce the overall computing time. The benchmarks presented here therefore do not represent a head-to-head comparison of grids, but rather a comparison of specific implementations. To be explicit, the implementation presented by Seljebotn et al. (2014) may in theory be sped up by a factor of 24 if exploiting all HEALPix symmetries, and this factor should be compared to the results presented in the third column of Table 2. However, at lower resolutions the speed-ups seen in in Table 3 are almost entirely due to being able to use the operator resampling given in Equation (13). This degradation procedure is not as straightforward when using the HEALPix grid, as the approximation is significantly less accurate. Our previous code therefore used a resolution of $N_{\rm side} = 2048$ along columns on all the levels, leading to very long computation times.

To summarize, the SymPix grid reduces what used to be overnight jobs with our previous implementation to essentially interactive tasks.

5. CONCLUSION

We have presented SymPix, a novel spherical grid for efficiently sampling rotationally invariant operators. This grid derives many of its properties from the GL grid, ensuring overall excellent spherical harmonics transform performance. The main difference between the two grids is that SymPix sacrifices proper Nyquist sampling in the longitudinal direction in order to increase pixel symmetries, such that all grid pair distances repeat perfectly along constant-latitude rings. This decreases the computational scaling of evaluating rotationally invariant operators from $\mathcal{O}(N_{\rm pix})$ to $\mathcal{O}(\sqrt{N_{\rm pix}})$.

The intended primary application of the SymPix grid is efficient construction of preconditioners (or smoothers) for iterative linear solvers. In this paper we considered the specific example of drawing constrained Gaussian realizations using a multi-grid solver, which is an important problem in current CMB analysis. Comparing with previous state-of-the-art results (Seljebotn et al. 2014), we achieve average speed-ups of 360 and 23 for the two most important pre-computation steps when using SymPix for internal calculations.

However, we emphasize that SymPix is a special-purpose grid designed for precisely such tasks; it is not intended to provide a general-purpose spherical pixelization that is suitable for, say, mapmaking. HEALPix is clearly preferred for such purposes due to its uniform pixel areas, regular pixel window, and hierarchical pixel structure. Likewise, if machine precision spherical harmonics transforms are required, the GL grid is the obvious choice. However, for those particular applications that can benefit from efficient pixel-space sampling of linear operators, such as ours, SymPix holds a clear edge over existing alternatives.

D.S.S. and H.K.E. are supported by European Research Council grant StG2010-257080.

APPENDIX
CODE

The SymPix code has been developed as part of the Commmander project, and does not yet have its own library. For the benefit of the reader, however, we have copied the source files relevant to this paper to their own repository at http://github.com/dagss/sympix. Please consult the accompanying README file for further details. This repository will be updated if the code does eventually develop into a stand-alone package.

The SHTs are all done using `libsharp` (Reinecke & Seljebotn 2013), which, at the time of writing, is available at http://sourceforge.net/projects/libsharp/. We then construct the grid geometry in our Python code and feed it to

`libsharp`. In the future we may port our Python code to C and make it available directly in `libsharp`.

REFERENCES

Cormen, T. H., Leiserson, C. E., & Rivest, R. L. 1989, Introduction to Algorithms (Cambridge, MA: MIT Press)

Doroshkevich, A. G., Naselsky, P. D., Verkhodanov, O. V., et al. 2005, IJMPD, 14, 275

Elsner, F., & Wandelt, B. D. 2011, A&A, 532, A35

Eriksen, H. K., O'Dwyer, I. J., Jewell, J., et al. 2004, ApJS, 155, 227

Górski, K. M., Hivon, E., Banday, A. J., et al. 2005, ApJ, 622, 2

Jewell, J., Levin, S., & Anderson, C. H. 2004, ApJ, 609, 1

Planck Collaboration 2015, A&A, submitted, (arXiv:1502.01582)

Prézeau, G., & Reinecke, M. 2010, ApJS, 190, 267

Reinecke, M. 2011, A&A, 526, A108

Reinecke, M., & Seljebotn, D. S. 2013, A&A, 554, A112

Seljebotn, D. S., Mardal, K.-A., Jewell, J. B., Eriksen, H. K., & Bull, P. 2014, ApJS, 210, 24

Sutter, P., Wandelt, B. D., & Elsner, F. 2012, in Proc. Big Bang, Big Data, Big Computers (Big3), 2012

Wandelt, B. D., Larson, D. L., & Lakshminarayanan, A. 2004, PhRvD, 70, 08351