The Astrophysical Journal Supplement Series, 210:24 (13pp), 2014 February doi:10.1088/0067-0049/210/2/24

A MULTI-LEVEL SOLVER FOR GAUSSIAN CONSTRAINED COSMIC MICROWAVE BACKGROUND REALIZATIONS

D. S. Seljebotn[1], K.-A. Mardal[2,3], J. B. Jewell[4], H. K. Eriksen[1], and P. Bull[1]
[1] Institute of Theoretical Astrophysics, University of Oslo, P.O. Box 1029 Blindern, NO-0315 Oslo, Norway; d.s.seljebotn@astro.uio.no
[2] Department of Informatics, University of Oslo, P.O. Box 1080 Blindern, NO-0316 Oslo, Norway
[3] Centre for Biomedical Computing, Simula Research Laboratory, P.O. Box 134, NO-1325 Lysaker, Norway
[4] Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109, USA
Received 2013 August 25; accepted 2013 November 29; published 2014 January 23

ABSTRACT

We present a multi-level solver for drawing constrained Gaussian realizations or finding the maximum likelihood estimate of the cosmic microwave background sky, given noisy sky maps with partial sky coverage. The method converges substantially faster than existing Conjugate Gradient (CG) methods for the same problem. For instance, for the 143 GHz Planck frequency channel, only three multi-level W-cycles result in an absolute error smaller than 1 μK in any pixel. Using 16 CPU cores, this translates to a computational expense of 6 minutes wall time per realization, plus 8 minutes wall time for a power-spectrum-dependent precomputation. Each additional W-cycle reduces the error by more than an order of magnitude, at an additional computational cost of 2 minutes. For comparison, we have never been able to achieve similar absolute convergence with conventional CG methods for this high signal-to-noise data set, even after thousands of CG iterations and employing expensive preconditioners. The solver is part of the Commander 2 code, which is available with an open source license at http://commander.bitbucket.org/.

Key words: cosmic background radiation – methods: numerical – methods: statistical

Online-only material: color figures

1. INTRODUCTION

Apart from a substantial kinematical dipole, the cosmic microwave background (CMB) radiation is observed to be isotropic to around one part in 10^4. Below this level, there are random fluctuations over a wide range of angular scales. The prevailing "concordance" cosmological model explains these anisotropies as the imprints of Gaussian-distributed, statistically isotropic perturbations of spacetime that were generated during an inflationary epoch in the early universe. Correlations between the fluctuations provide a wealth of information about inflation and the subsequent growth of structure, and so being able to accurately measure and characterize them is of paramount importance to modern cosmology.

As detector technology has improved, it has become possible to probe smaller and smaller angular scales with ever-increasing noise sensitivities. The resulting improvement in resolution and signal-to-noise ratio presents a formidable computational challenge, as one must now reliably reconstruct the CMB sky to high accuracy over tens of millions of pixels, while simultaneously taking into account complexities of the data such as inhomogeneous noise, foreground contamination, and regions of missing/masked data.

Consider an observed map of the CMB, for instance similar to those provided by the *Wilkinson Microwave Anisotropy Probe* (*WMAP*; Bennett et al. 2013) and Planck (Planck Collaboration 2013a) experiments. The ideal CMB map would consist of an error-free value at every single position on the sky. In reality this is of course not possible, because of instrumental imperfections (such as noise and beam-smoothing) and strong foreground contamination from astrophysical sources; there will always be uncertainties in a real CMB map. Therefore, rather than aiming to extract "a single true CMB sky map," a more realistic solution is to compute an ensemble of many possible CMB skies, each of which is both noise-free, full-sky, and *statistically consistent* with the observed data. This idea has already been implemented for CMB analysis purposes in terms of a Gibbs sampling framework, as described by Jewell et al. (2004), Wandelt et al. (2004), and Eriksen et al. (2004, 2008b).

An underlying assumption in this line of work is that both the CMB sky and instrumental noise are random Gaussian fields with covariance matrices **S** and **N**, respectively. In most applications—following the basic inflationary prediction—one additionally assumes that the CMB field is isotropic, so that the CMB covariance matrix can be specified in terms of a simple angular power spectrum, C_ℓ. Of course, this power spectrum is not known a priori, but must instead be estimated from the data, and indeed, this is usually the main goal for most CMB experiments.

The Gibbs sampling framework provides a well-structured mathematical solution to this power spectrum estimation problem, by establishing the full joint Bayesian posterior distribution of the CMB sky and CMB power spectrum. This is found by iteratively sampling from the (more tractable) conditional distributions according to a simple algorithm: (1) make an arbitrary initial "guess" for the CMB power spectrum, (2) draw a CMB sky map compatible with the data and the assumed power spectrum, (3) draw a power spectrum compatible with the sky sample that was just drawn, and (4) iterate. The resulting set of sky and power spectrum samples will (after some burn-in period) converge to the true joint posterior distribution.

Although simple to write down, this algorithm is also computationally rather expensive due to step (2), which essentially amounts to solving a large linear system with one or more random terms on the right-hand side, corresponding to different realizations. We will refer to this system as the *constrained realization* (CR) system. The same linear system can also be solved for the maximum likelihood CMB sky map estimate, which is sometimes referred to as the *Wiener-filtered map*. Since the degrees of freedom of the CR system scale with the number of pixels, brute force solutions are out of bounds except for very low-resolution data sets. However, it is computationally feasible to multiply an arbitrary vector with the system matrix by repeatedly changing basis functions (i.e., spherical harmonic

transforms, SHTs), so that the system can be solved using iterative linear equation solvers. The main problem is to optimize the convergence rate of these solvers to produce a solution in a timely manner.

Commander (Eriksen et al. 2004), the CMB Gibbs sampler mentioned above, solves the CR system through the Conjugate Gradient (CG) method, using a combination of a block preconditioner on large angular scales and a diagonal preconditioner on small angular scales. While this approach was successful for analyzing *WMAP* observations (O'Dwyer et al. 2004; Eriksen et al. 2007a, 2007b, 2008a), the higher signal-to-noise level of data from more recent experiments like Planck effectively halts convergence of the solver. Indeed, as we will see in Section 2.4, the number of CG iterations intrinsically scales with the signal-to-noise ratio of a given data set, limiting the utility of CG for data sets such as these. To produce the low-ℓ power spectrum likelihood for the Planck mission, for example, the data had to be downgraded to low angular resolution and a substantial amount of regularization noise added (Planck Collaboration 2013b). Even then, several thousands of CG iterations were required for convergence. To go to full angular resolution with this scheme is simply not computationally feasible.

A somewhat better approach was described by Smith et al. (2007), who applied the CG method recursively, such that a CG solution on a coarse grid was used as the preconditioner for CG on a finer grid. We are not aware of any head-to-head comparisons of this method versus the one described by Eriksen et al. (2004), but our understanding is that, although it is faster, it still scales with the signal-to-noise ratio of the data set, and therefore does not inherently fix the fundamental convergence problems for high-sensitivity, high-resolution analysis.

More recently, Elsner & Wandelt (2012, 2013) introduced a stationary iterative method for solving the CR equation. They did not quote the usual statistics for convergence, such as total reduction in residual and error, however. Not knowing the accuracy of their solution, we are unable to compare the efficiency of their method directly to ours. While they do quote the change in the χ^2 statistic of the posterior probability density *between successive iterations*, iterative methods (and stationary methods in particular) are vulnerable to breaking down in terms of convergence rate well before reaching true convergence. Also, the χ^2 explicitly ignores large scales under the mask. While there certainly are applications where this is acceptable, CMB Gibbs sampling is not one of them, since it explicitly iterates between considering the CMB signal a sample from the posterior, which mostly ignores the masked area, and a sample from the prior, which gives equal weight to the masked area.

In this paper we present a new solver for the CR system that is radically different from the CG approach, and instead builds on the multi-level (or multi-grid) framework. These algorithms are best known in the astrophysics community as solvers for elliptical partial differential equations (PDEs), although they are in fact more generally applicable to solving many types of linear systems (Brandt 2001). We apply multi-level theory to the CR equation (although the algorithm is not entirely traditional), and show that the resulting algorithm converges to the exact solution with only a handful of iterations even for the most sensitive Planck channel. Most importantly, and contrary to the CG solver, the convergence rate is nearly independent of the signal-to-noise ratio of the data set.

Multi-level methods have been explored before in the CMB community for the purposes of map-making. Doré et al. (2001) described a standard multi-grid method for map-making, although it was eventually unable to compete with standard CG and approximate map-makers. Grigori et al. (2012) also presented a promising two-level CG preconditioner for map-making based on the domain-decomposition method in Havé et al. (2013). The map-making equation is different from CR equation, however, in that one does not solve for the CMB signal under a mask. As we will see in Section 2.4, it is this feature in particular that makes convergence difficult to achieve on the CR system.

2. EXPLORING THE CR LINEAR SYSTEM

2.1. Matrix Notation for Spherical Harmonic Transforms

The details of changing between pixel domain and spherical harmonic (SH) domain are usually glossed over in the literature. Since we will be solving a large linear system that couples signals on all scales—from individual pixels to the full-sky—it is of the utmost importance to be precise about how these conversions are performed. If implemented incorrectly, even small pixel-scale errors can lead to overall divergence of the entire method.

There is no perfect grid on the sphere, and in choosing a particular one, a number of trade-offs must be considered. In our current implementation we adopt both the HEALPix[5] pixelization (Górski et al. 2005) and the Gauss–Legendre spherical grid (Reinecke 2011, and references therein). The HEALPix software package contains routines that are useful for our pixel-domain computations, while the latter is required for accurate evaluation of Equation (2) below.

Given such a grid on the sphere (by which we mean a set of positions $\hat{n}_i$ on the sky), we can use *SH synthesis* to transform a field expressed in spherical harmonic basis, with coefficients $s_{\ell m}$, to a field sampled on the sphere,

$$\hat{s}(\hat{n}_i) = \sum_{\ell=0}^{\ell_{\max}} \sum_{m=-\ell}^{\ell} s_{\ell m} Y_{\ell m}(\hat{n}_i). \quad (1)$$

We will write this operation in matrix form as $\widehat{\mathbf{s}} = \mathbf{Y}\mathbf{s}$, where $\mathbf{Y}$ encodes the value of the SHs evaluated at each $\hat{n}_i$ of the chosen grid. Note that $\mathbf{Y}$ is not a square matrix, as spherical grids need to over-sample the signal to faithfully represent it up to some bandlimit $\ell_{\max}$. In typical applications there are between 30% and 100% more pixels along the rows of $\mathbf{Y}$ than there are SH coefficients along the columns. For the purposes of our method, it will turn out that we need to under-pixelize the signal instead, so there will be more columns than rows in $\mathbf{Y}$.

The opposite action of converting from pixel basis to harmonic basis is *SH analysis*, which generally takes the quadrature form

$$s_{\ell m} = \int_{4\pi} Y^*_{\ell m}(\hat{n})\hat{s}(\hat{n})d\Omega \approx \sum_{i=1}^{N_{\rm pix}} Y^*_{\ell m}(\hat{n}_i) w_i \hat{s}(\hat{n}_i), \quad (2)$$

where w_i combines quadrature weights and pixel area. Similar to the synthesis case, this operation can be written in matrix form as $\mathbf{s} = \mathbf{Y}^T\mathbf{W}\widehat{\mathbf{s}}$, where $W_{ij} = w_i\delta_{ij}$. A crucial feature of our method is the ability to (for the most part) avoid SH analysis, however. Instead, we will rely on *adjoint SH synthesis*, $\mathbf{Y}^T$, which simply appears algebraically as the transpose of $\mathbf{Y}$.

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

Note that, unlike in the case of the more famous discrete Fourier transform, $\mathbf{Y}$ is not a square orthogonal matrix, and synthesis and analysis differ by more than transposition and a scale factor. One may in some situations have that $\mathbf{Y}^T\mathbf{W}\mathbf{Y} = \mathbf{I}$, but this depends on both $\ell_{\max}$, $N_{\rm pix}$ and the spherical grid.

The action of applying $\mathbf{Y}$, $\mathbf{Y}^T$, $\mathbf{Y}^T\mathbf{W}$ or $\mathbf{W}\mathbf{Y}$ to a vector is in general referred to as a SHT. Carefully optimized libraries are available that perform SHTs in $O(\ell_{\max} N_{\rm pix})$ time; we use the `libsharp` library (Reinecke & Seljebotn 2013).

2.2. Data Model

We now define our data model, and assume from the beginning that the CMB is Gaussian and isotropic (e.g., Planck Collaboration 2013c). Following the notation of Eriksen et al. (2004), it is convenient to define the CMB signal to be a vector $\mathbf{s}$ of SH coefficients, in which case the associated covariance matrix $\mathbf{S}$ is given by

$$S_{\ell m,\ell' m'} = \delta_{\ell\ell'}\delta_{mm'}C_\ell,$$

where C_ℓ is the CMB power spectrum.

Using the notation of the previous section, the model for the observed sky map pixel vector, $\mathbf{d}$, is

$$\mathbf{d} = \mathbf{Y}_{\rm obs}\mathbf{B}\mathbf{s} + \mathbf{n}, \tag{3}$$

where $\mathbf{B}$ denotes beam-smoothing and the pixel window function, $\mathbf{n}$ is Gaussian instrumental noise, and the subscript of $\mathbf{Y}_{\rm obs}$ indicates projection to the pixelization of the map $\mathbf{d}$.

We assume a symmetric instrumental beam, so that the beam matrix $\mathbf{B}$ is a diagonal matrix given by $B_{\ell m,\ell' m'} = b_\ell p_\ell \delta_{\ell\ell'}\delta_{mm'}$, where b_ℓ is the instrumental beam and p_ℓ the pixel window function of the observed grid. We also assume white instrumental noise, such that the noise covariance matrix, $\mathbf{N}$, is diagonal. We discuss the likely impact of asymmetric beams and correlated noise in Section 5.

Discretization of the model is done simply by picking some $\ell_{\max}$ for the $\mathbf{s}$ vector. The noise vector $\mathbf{n}$ is related to the map-making process, averaging the noise of time-ordered data that fall within the same pixel, and so is inherently discrete rather than being a discretization of any underlying field. As already mentioned above, no SH analysis of $\mathbf{d}$ (and therefore $\mathbf{n}$) is required when solving the CR system; rather, one solves for the projected $\mathbf{s}$, and so the noise treatment is always perfectly consistent with the assumed model.

2.3. The CR Linear System

Given the data model above, we are interested in exploring the Bayesian posterior distribution $p(\mathbf{s}|\mathbf{d}, C_\ell)$, the CMB signal given the data and CMB power spectrum. Let us first define

$$\mathbf{A} \equiv \mathbf{S}^{-1} + \mathbf{B}\mathbf{Y}_{\rm obs}^T\mathbf{N}^{-1}\mathbf{Y}_{\rm obs}\mathbf{B}, \tag{4}$$

where in what follows we will refer to the first term as the *prior term*, and the second as the *inverse-noise term*. It can be shown that if we now solve the *CR system*

$$\mathbf{A}\mathbf{x} = \mathbf{B}\mathbf{Y}_{\rm obs}^T\mathbf{N}^{-1}\mathbf{d}, \tag{5}$$

the solution $\mathbf{x}$ will be the maximum likelihood estimate of $\mathbf{s}$. Alternatively, if particular random fluctuation terms are added to the right-hand side of Equation (5), the solution $\mathbf{x}$ will instead be samples from the posterior (Jewell et al. 2004; Wandelt et al. 2004). Since $b_\ell \to 0$ as ℓ increases, the diagonal prior term will at some point dominate the dense inverse-noise term, so that truncation at sufficiently high $\ell_{\max}$ does not affect the solution of the system.

As stressed in Section 2.1, $\mathbf{Y}_{\rm obs}^T$ denotes SH *adjoint synthesis*, and not SH analysis. Pixels that are masked out, typically due to strong foreground contamination, are simply missing from the data vector $\mathbf{d}$, and so the corresponding rows are not present in $\mathbf{Y}_{\rm obs}$. This means $\mathbf{Y}_{\rm obs}$ is not an orthogonal matrix, but that is not a concern since we never perform SH analysis of pixels on the observation grid. The solution $\mathbf{x}$ is still well-defined everywhere on the sky due to the prior term $\mathbf{S}^{-1}$. This is typically implemented by introducing zeroes in $\mathbf{N}^{-1}$ rather than removing rows of $\mathbf{Y}_{\rm obs}$, which has the statistical interpretation of giving those pixels infinite variance. The two interpretations are algebraically equivalent.

2.4. Eigenspectrum and CG Performance

The CR system in Equation (4) is symmetric and positive definite, which suggests the use of the CG algorithm. For the behavior of CG and other Krylov methods, we are primarily interested in the eigenspectrum after preconditioning (Shewchuk 1994, and references therein), i.e., the eigenspectrum of $\mathbf{M}\mathbf{A}$, where $\mathbf{M} \approx \mathbf{A}^{-1}$. To illustrate the fundamental problem with the CG algorithm for the application considered here, we show in Figure 1 the eigenspectrum of a low-resolution setup, using a diagonal preconditioner. This case corresponds to a simulation of the 143 GHz Planck frequency map (Planck Collaboration 2013a), downgraded to an angular resolution of 5.°4, bandwidth-limited at $\ell_{\max} = 95$, and with a mask applied that removes 40% of the sky. The overall shape of the spectrum appears to be mostly independent of the resolution, with a significant fraction of degrees of freedom found in the tails. This behavior is representative of that found in real-world cases.

The problematic feature is the exponential drop in the eigenvalues seen to the left of the figure. Theoretical results indicate that the CG search needs at least one iteration per eigenvalue located in exponentially increasing parts of the eigenspectrum (Axelsson & Lindskog 1986a, 1986b). This leads to extreme degradation of CG performance, which is indeed what has been observed with Commander on high-resolution, high-sensitivity data.

The exponential spectral feature is due to large-scale modes under the mask. For all but the smallest angular scales, the $\mathbf{N}^{-1}$ term dominates by many orders of magnitude, so that the $\mathbf{S}^{-1}$ term is hardly seen at all. However, vectors that only build-up signal under the mask after beam-smoothing will only see the $\mathbf{S}^{-1}$ term of the matrix, as the $\mathbf{N}^{-1}$ term vanishes in that case. The eigenvectors corresponding to the smallest eigenvalues are therefore characterized by having large scales localized within the mask. Moreover, the solution under the mask is constrained by the values at the mask edge, meaning the $\mathbf{N}^{-1}$ term takes effect, and this constraint is harder closer to the edges. The result is an exponentially falling eigenspectrum, rather than separated clusters of eigenvalues that CG could more easily deal with.

Phrased differently, for data having a high signal-to-noise ratio, the pixels near the edge of the mask carry a large predictive power on the signal inside the mask—a signal that must be reconstructed by the CG algorithm by navigating through a nearly degenerate system. In total, the CG convergence rate is determined by a combination of the overall signal-to-noise ratio and the size and shape of the mask. We have been unable to achieve proper convergence with this method for the

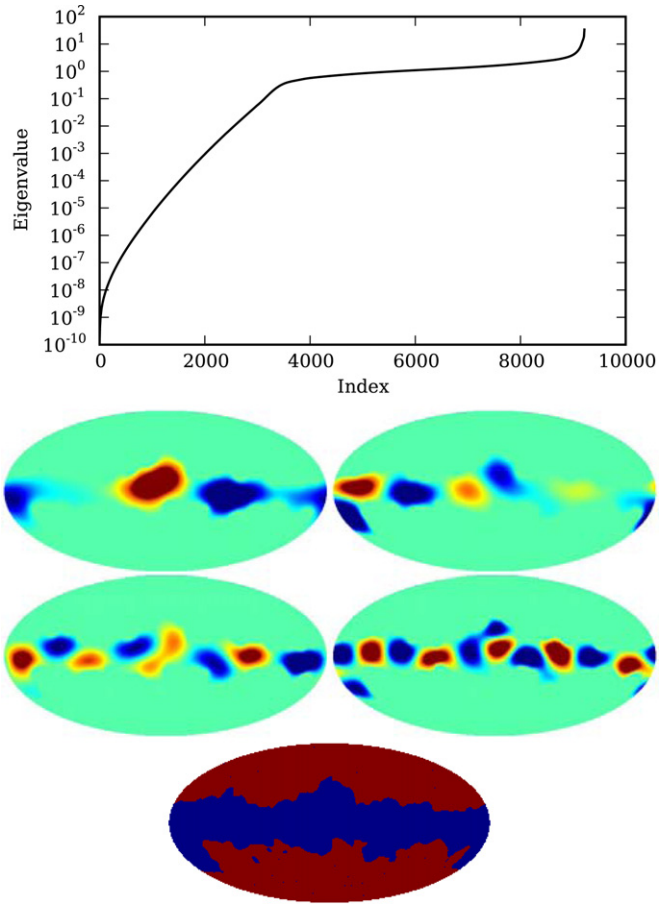

Figure 1. Eigendecomposition of the CR system using a diagonal preconditioner. Top panel: the eigenvalues of $\mathrm{diag}(\mathbf{A})^{-1}\mathbf{A}$ for the 143 GHz Planck channel with a mask covering 40% of the sky, smoothed with a 5.′6 FWHM beam and truncated at $\ell_{\max} = 95$. Bottom panel: a selection of eigenvectors corresponding to very low eigenvalues. The structure of the mask (bottom) is clearly visible in the eigenvectors.

(A color version of this figure is available in the online journal.)

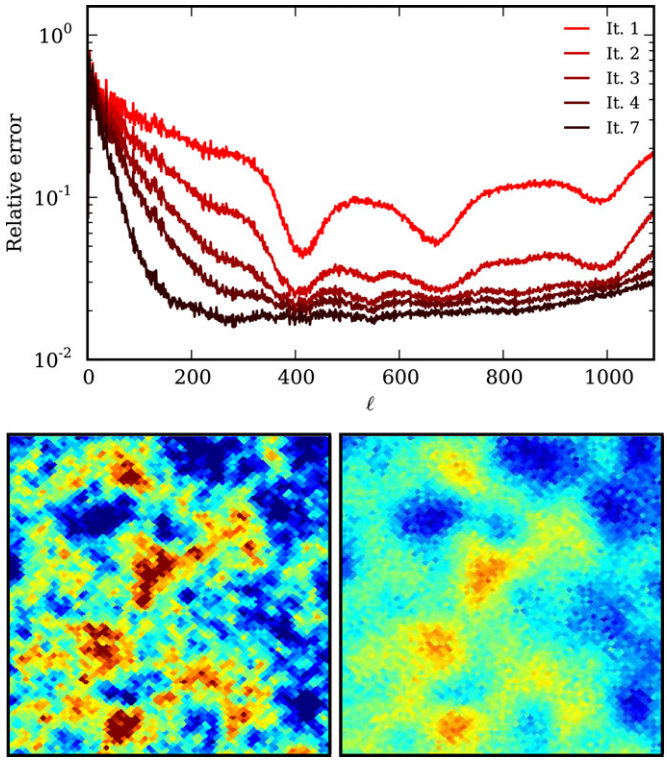

Figure 2. Effect of the error smoother/approximate inverse $\widehat{\mathbf{M}}$. Top: relative error $\|\mathbf{x}_\ell - \mathbf{x}_{\mathrm{true},\ell}\|/\|\mathbf{x}_{\mathrm{true},\ell}\|$. For each iteration, the error smoother developed in Section 3.5 is applied on a HEALPix $N_{\mathrm{side}} = 512$ grid. The error smoother is only able to get closer to the solution for some part of the frequency spectrum, and quickly stagnates since no improvement is made to the larger or smaller scales. Bottom: the left patch shows the initial error when starting at $\mathbf{x} = \mathbf{0}$, while the right patch shows the error after the first iteration. The remaining large-scale errors can be represented on a coarser grid. This observation leads to the multi-level algorithm.

(A color version of this figure is available in the online journal.)

signal-to-noise ratio of a Planck-like experiment, for example, independent of preconditioners or number of iterations; downgrading and adding regularization noise is required to produce robust results.

3. THE MULTI-LEVEL SOLVER

3.1. Motivation for a Multi-level Method

The matrix $\mathbf{A}$ of Equation (4) is defined in SH domain, and describes the coupling strength between pairs of (ℓ, m) and (ℓ', m'). Except in unrealistic scenarios with very simple instrumental noise and mask, we have found no pattern in the magnitudes of the matrix coefficients $A_{\ell m,\ell' m'}$ that is consistent enough to be exploited in a solver.

By moving to pixel domain, however, we can *create* such an exploitable pattern in the magnitudes of the matrix coefficients. In Section 3.3 we will construct a corresponding pixel-domain matrix $\widehat{\mathbf{A}}$ that is *localized*, in the sense that $\widehat{A}_{ij}$ has small magnitude (less than 1% of $\widehat{A}_{ii}$) unless pixels i and j are very close together on the sphere.

It is no surprise that the $\mathbf{N}^{-1}$ term of Equation (4) enjoys this property, since we have assumed that instrumental noise is uncorrelated between pixels. When it comes to the $\mathbf{S}^{-1}$ term, we note that $1/C_\ell$ is roughly proportional to $\ell(\ell+1)$, at least for $\ell \lesssim 1000$. These are the eigenvalues of the Laplacian on the sphere, with $\mathbf{Y}$ being the corresponding eigenbasis. Therefore we can hope that a projection of $\mathbf{S}^{-1}$ to pixel domain should be close to a Laplacian. The Laplacian is often approximated with a matrix where $\widehat{A}_{ij} = 0$ unless pixel i and j are neighbors or $i = j$. While our case will be less perfect, it still suggests that multi-level methods can be very efficient, since those are highly successful for PDEs involving the Laplacian.

In Section 3.5, we exploit the localization properties in pixel domain to develop an approximate inverse $\widehat{\mathbf{M}} \approx \widehat{\mathbf{A}}^{-1}$. Figure 2 demonstrates the use of this approximate solver as part of a simple stationary method

$$\mathbf{x} \leftarrow \mathbf{x} + \widehat{\mathbf{M}}(\mathbf{b} - \widehat{\mathbf{A}}\mathbf{x}), \tag{6}$$

where we initialize $\mathbf{x} \leftarrow \mathbf{0}$ and then iteratively update the solution. Note that if we replace $\widehat{\mathbf{M}}$ with $\mathrm{diag}(\mathbf{A})^{-1}$, Equation (6) represents what are known as Jacobi iterations.

The problem that is evident from Figure 2 is that $\widehat{\mathbf{M}}$ will only make improvements to one part of the frequency spectrum—namely, the highest frequencies that can be represented on the grid used. This is the typical case when multi-level methods are applied; iterations of the form of Equation (6) are usually only efficient at resolving the relations between pixels/elements that are strongly coupled, which, when $\widehat{\mathbf{A}}$ is localized, translates to resolving the solution at highest frequencies. Little or no improvement is made between pixels that are weakly or indirectly

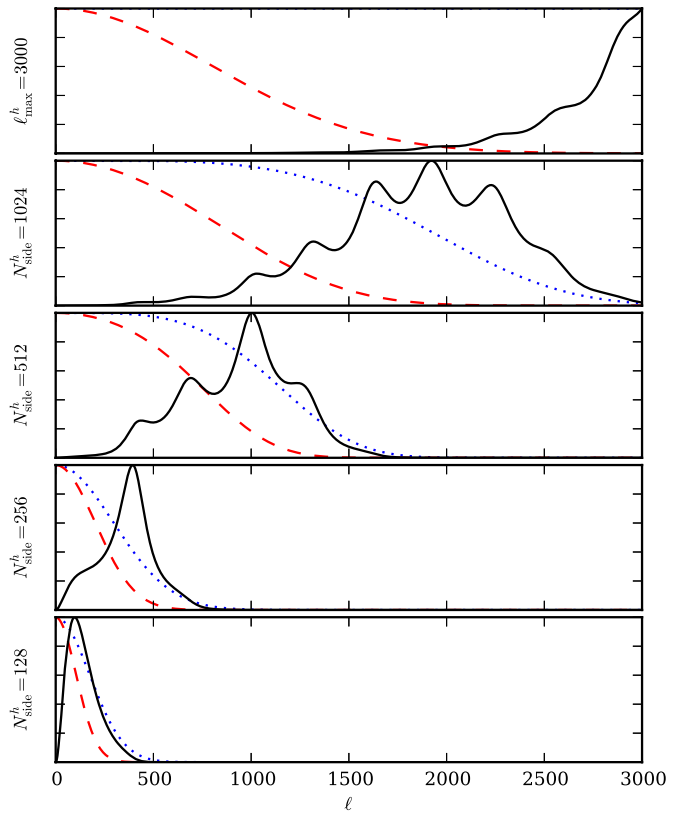

Figure 3. Effect of filters in harmonic domain for the top five levels. For each level H, starting from the original system of Equation (4) at the top, we plot the transfer filter $f^H_{h,\ell}$ (dotted blue), the filtered prior $(\tilde{f}^H_\ell)^2/C_\ell$ (solid black), and an approximation to the diagonal of the inverse-noise term (dashed red). Functions are normalized to an arbitrary scale (see Figure 4 for the absolute scale). Note how the prior term on the pixel levels looks superficially similar to wavelets/needlets in harmonic domain (Scodeller et al. 2011, and references therein). The real-space transform is also similar to wavelets/needlets (not plotted).

(A color version of this figure is available in the online journal.)

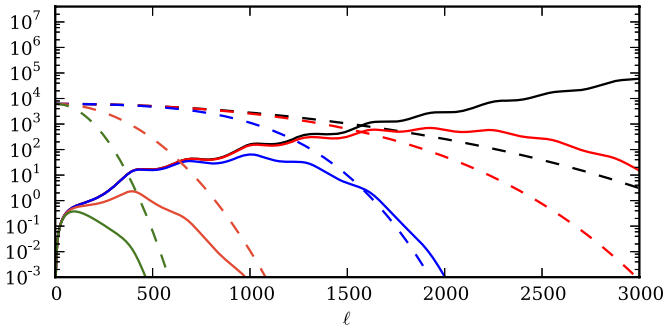

Figure 4. Same as Figure 3, but all levels plotted together with a logarithmic scale and with absolute normalization. We plot the filtered prior $(\tilde{f}^H_\ell)^2/C_\ell$ (solid), and the diagonal of the inverse-noise term for $26\,\mu$K constant rms and no mask (dashed). This noise level corresponds to the average of the rms map of the 143 GHz Planck band. The levels are: the original system (black), $N^h_{\rm side}=1024$ (red), $N^h_{\rm side}=512$ (blue), $N^h_{\rm side}=256$ (orange), and $N^h_{\rm side}=128$ (green). Note the effect of the filters on the signal-to-noise ratio; harmonic scales go from being data-dominated to noise-dominated at the point where the solid and dashed lines intersect.

(A color version of this figure is available in the online journal.)

coupled in $\widehat{\mathbf{A}}$, so that no improvement is made to the coarser scales. Put another way, the error, $\mathbf{e}\equiv\mathbf{x}-\mathbf{x}_{\rm true}$, has its high-frequency components reduced, while the low frequencies are left relatively unaffected. The approximate inverse $\widehat{\mathbf{M}}$ is therefore dubbed a *smoother* in multi-level terminology. We will use the term *error smoother* to distinguish it from the act of applying a low-pass filter (which is instead called *restriction* in this context).

The key is now to project the matrix $\mathbf{A}$ to pixel grids at different resolutions, producing a set of matrices $\widehat{\mathbf{A}}_h$, where h is a level indicator. For each $\widehat{\mathbf{A}}_h$ we construct a corresponding error smoother $\widehat{\mathbf{M}}_h\approx\widehat{\mathbf{A}}_h^{-1}$ that resolves the errors in one region of the frequency spectrum only. Using these levels together, we arrive at a method that converges very well over the entire frequency spectrum.

3.2. The Multi-level Algorithm

In this section we give a brief overview of multi-level theory, together with the specification of our algorithm. For a more detailed introduction to multi-grid methods, consult one of the number of standard texts (e.g., Hackbush 1985). Ingredients of multi-level algorithms are:

1. A set of bases to project the linear system into in order to work on different parts of the solution. Usually these form a hierarchy of levels from finest to coarsest, so that each level solves for different frequencies of the solution. It is customary to label levels relatively, using h for the current level and H for the coarser level.
2. A way to transfer vectors between the different levels. The *restriction* operator, $\mathbf{I}^H_h$, takes a vector from a finer level to a coarser level, while the *interpolation* operator $\mathbf{I}^h_H$ works in the opposite direction. For symmetric systems, one often takes $\mathbf{I}^h_H=(\mathbf{I}^H_h)^T$.
3. One linear operator (left-hand side matrix) for each level. For the case where interpolation is chosen to be transposed restriction, these are often defined recursively as
$$\mathbf{A}_H=\mathbf{I}^h_H\mathbf{A}_h\left(\mathbf{I}^h_H\right)^T \tag{7}$$
for the projection of a fine matrix $\mathbf{A}_h$ to a coarser matrix $\mathbf{A}_H$.
4. An *error smoother* $\mathbf{M}_h$ for each $\mathbf{A}_h$ that removes the higher frequencies of the error on level h, as discussed in the previous section.

Multi-level algorithms are often implemented on a grid or a tessellation in real space, with a sparse linear operator, and using averages of neighboring points as the restriction operator $\mathbf{I}^H_h$. In our case, $\widehat{\mathbf{A}}_h$ on each level is not sparse, and, at least without approximations, multiplying $\widehat{\mathbf{A}}_H=\mathbf{I}^h_H\widehat{\mathbf{A}}_h\mathbf{I}^H_h$ with a vector would be computationally very expensive on the coarser levels as it would require interpolating back to the highest-resolution grid.

To avoid this cost, we instead define our levels in SH domain. Let $\tilde{f}^h_\ell$ be a SH low-pass filter that emphasizes one part of the frequency spectrum, and define $\mathbf{F}_h$ to be a diagonal matrix with elements $\tilde{f}^h_\ell$. We then define

$$\mathbf{A}_h\equiv\mathbf{F}_h\mathbf{A}\mathbf{F}^T_h\equiv\mathbf{D}_h+\mathbf{B}_h\mathbf{Y}^T_{\rm obs}\mathbf{N}^{-1}\mathbf{Y}_{\rm obs}\mathbf{B}_h, \tag{8}$$

where the prior term $\mathbf{D}_h$ is diagonal with entries given by $(\tilde{f}^h_\ell)^2/C_\ell$ and the modified beam matrix $\mathbf{B}_h$ is diagonal with elements given by $\tilde{f}^h_\ell b_\ell p_\ell$. In this case, the system is bandlimited by some $\ell^h_{\rm max}\leqslant\ell_{\rm max}$, above which $\tilde{f}^h_\ell=0$. Figures 3 and 4 show the filters used in our setup; we discuss the choice of filters further in Section 3.3.

With this choice, we can clearly satisfy the multi-level hierarchy of Equation (7) by choosing the restriction operator $\mathbf{I}_h^H$ as an $(\ell_{\max}^H+1)^2$-by-$(\ell_{\max}^h+1)^2$ block matrix, where the block for $\ell \leqslant \ell_{\max}^H$ is diagonal with entries

$$f_{h,\ell}^H \equiv \frac{\tilde{f}_\ell^H}{\tilde{f}_\ell^h}, \tag{9}$$

and the block for $\ell_{\max}^H < \ell \leqslant \ell_{\max}^h$ is zero.

As already mentioned in Section 3.1, the error smoother that we have available, $\widehat{\mathbf{M}}_h$, is defined in pixel domain. For every SH level we therefore tag on a corresponding sibling pixel level with matching HEALPix resolution N_{side}^h. The result is the following level structure:

$$\begin{array}{ccc}
\text{SH at } \ell_{\max}^h = 3000 & \longleftrightarrow & \text{Pixels at } N_{\text{side}}^h = 1024 \\
\updownarrow & & \\
\text{SH at } \ell_{\max}^h = 2048 & \longleftrightarrow & \text{Pixels at } N_{\text{side}}^h = 512 \\
\updownarrow & & \\
\text{SH at } \ell_{\max}^h = 1280 & \longleftrightarrow & \text{Pixels at } N_{\text{side}}^h = 256 \\
\updownarrow & & \\
\vdots & & .
\end{array}$$

The arrows indicate that transfers between different scales happen only through the SH levels. As emphasized in Section 2.1, no SH analysis operations are performed in each conversion (only synthesis and adjoint synthesis operations), and the implied under-pixelization in the above scheme is therefore numerically unproblematic.

The full details of how to properly move between the levels to obtain a solution is given in pseudo-code in Figure 5. We highlight some aspects in what follows.

Assume that we are currently on some SH level h (where the original equation is simply the top level), with corresponding system

$$\mathbf{A}_h \mathbf{x}_{\text{true},h} = \mathbf{b}_h. \tag{10}$$

We start with some search vector $\mathbf{x}_h$ (initialized to zero), and want to improve it to get closer to the true value $\mathbf{x}_{\text{true},h}$. In order to make use of $\widehat{\mathbf{M}}$, we must now move to the corresponding pixel level. Our chosen restriction operator, denoted $\mathbf{Y}_h$, is SH synthesis to a HEALPix grid[6] of resolution N_{side}^h. The key to efficient multi-level solvers is to transfer the residual vector $\mathbf{r}_h$, and *not* the search vector $\mathbf{x}_h$;

$$\mathbf{r}_h \leftarrow \mathbf{b}_h - \mathbf{A}_h \mathbf{x}_h \tag{11}$$

$$\widehat{\mathbf{r}}_h \leftarrow \mathbf{Y}\mathbf{r}_h, \tag{12}$$

where $\widehat{\mathbf{r}}_h$ is the pixel-domain projection of $\mathbf{r}_h$. Then, we approximately solve the projected system for a *correction vector* $\widehat{\mathbf{c}}_h$,

$$\widehat{\mathbf{c}}_h \leftarrow \widehat{\mathbf{M}}\widehat{\mathbf{r}}_h \approx \left(\mathbf{Y}_h \mathbf{A}_h \mathbf{Y}_h^T\right)^{-1} \widehat{\mathbf{r}}_h, \tag{13}$$

where the computation of $\widehat{\mathbf{M}}\widehat{\mathbf{r}}_h$ is further described in Section 3.5. The approximation is better for small scales than for large scales. Finally, we let the interpolation operator be the transpose of restriction, $\mathbf{Y}^T$, so that the correction is brought over to the SH search vector by adjoint SH synthesis,

$$\mathbf{x}_h \leftarrow \mathbf{x}_h + \mathbf{Y}^T \widehat{\mathbf{c}}_h. \tag{14}$$

Together, these steps act as the error smoothing of a SH level, labeled pre- and post-smoothing in Figure 5. Here we have motivated the procedure as arising from moving between levels, but the idea of solving for a correction in a projected system arises in many settings (Tang et al. 2009), and other variations on this theme may prove fruitful in the future.

The vertical movement between coarser and finer levels follows the same pattern, but uses the restriction operator $\mathbf{I}_h^H$ defined in Equation (9) instead of pixel projection $\mathbf{Y}$. First, a fine residual is computed and restricted (i.e., low-pass filtered) to the coarser level,

$$\mathbf{r}_h \leftarrow \mathbf{b}_h - \mathbf{A}_h \mathbf{x}_h, \tag{15}$$

$$\mathbf{r}_H \leftarrow \mathbf{I}_h^H \mathbf{r}_h. \tag{16}$$

Then, a coarse correction $\mathbf{c}_H$ is sought that approximates the solution of the coarse system

$$\mathbf{I}_h^H \mathbf{A}_h \left(\mathbf{I}_h^H\right)^T \mathbf{c}_H = \mathbf{A}_H \mathbf{c}_H = \mathbf{r}_H. \tag{17}$$

CR-Cycle($h, \mathbf{x}, \mathbf{b}$):
Inputs:
h – The current level
$\mathbf{x}$ – Starting vector
$\mathbf{b}$ – Right-hand side

H denotes the coarser level relative to h.
Output:
Improved solution vector $\mathbf{x}$

if h is bottom level:
  $\mathbf{x} \leftarrow \mathbf{A}_h^{-1}\mathbf{b}$ *By dense Cholesky*
else:
  $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{Y}_h^T \widehat{\mathbf{M}}_h \mathbf{Y}_h (\mathbf{b} - \mathbf{A}_h \mathbf{x})$ *Pre-smoothing*
  $\mathbf{r}_H \leftarrow \mathbf{I}_h^H (\mathbf{b} - \mathbf{A}_h \mathbf{x})$ *Restricted residual*
  $\mathbf{c}_H \leftarrow \mathbf{0}$ *Coarse correction*
  repeat n_{rec}^h times:
    $\mathbf{c}_H \leftarrow$ CR-Cycle($H, \mathbf{c}_H, \mathbf{r}_H$) *Recurse*
  $\mathbf{x} \leftarrow \mathbf{x} + (\mathbf{I}_h^H)^T \mathbf{c}_H$ *Apply correction*
  $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{Y}_h^T \widehat{\mathbf{M}}_h \mathbf{Y}_h (\mathbf{b} - \mathbf{A}_h \mathbf{x})$ *Post-smoothing*
return $\mathbf{x}$

CR-Solve($\mathbf{b}, \epsilon$):
Inputs:
$\mathbf{b}$ – Right-hand side
ϵ – Requested improvement in residual
Output:
Approximate solution $\mathbf{x}$

$\mathbf{x} \leftarrow \mathbf{0}$
repeat:
  $\mathbf{x} \leftarrow$ CR-Cycle(1st, $\mathbf{x}, \mathbf{b}$)
  $\mathbf{r} \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x}$ *Reused in next CR-Cycle*
  if $\mathbf{r}^T \mathbf{S}^{-1} \mathbf{r} < \epsilon \mathbf{b}^T \mathbf{S}^{-1} \mathbf{b}$: *Improvement relative to* C_ℓ
    return $\mathbf{x}$

Figure 5. Multi-level CR solver. The matrices involved are defined in the main text. In place of the simple iteration scheme of CR-Solve, one can use CR-Cycle as a preconditioner within another solver, such as CG. By varying the n_{rec}^h parameter, a variety of solver cycles can be constructed, such as a V-cycle ($n_{\text{rec}}^h = 1$) or W-cycle ($n_{\text{rec}}^h = 2$). Note that, for simplicity, the top-level diagonal error correction is omitted; see the main text.

[6] The pixel level is actually coarser than the SH level, because $\ell_{\max}^h$ must be chosen so high that the grid cannot resolve all the scales of the projected field. See Section 3.4.

Except for at the bottom level, this happens by initializing a search vector $\mathbf{c}_H$ to zero and recursively applying the algorithm. Finally, the correction is interpolated and applied to our current search vector,

$$\mathbf{x}_h \leftarrow \mathbf{x}_h + \mathbf{I}_H^h \mathbf{c}_H. \tag{18}$$

Using this idea of transferring residuals and corrections between levels with different bases, one can form a variety of multi-level *cycles*, moving between the levels in different patterns. Our choice in the end is a W-cycle on the coarser levels and a V-cycle on the finer levels, as described in Section 4.1 and the pseudo-code.

In addition to the pixel levels described above, the top and bottom levels are special. The smallest scales ($\ell \gtrsim 2200$ in our experimental setup) are strongly noise-dominated, making the SH domain matrix $\mathbf{A}$ nearly diagonal. As a result, we do not project to a pixel grid, but simply use $\mathrm{diag}(\mathbf{A})^{-1}$ as the error smoother. Note, however, that this process would destroy the solution on scales that are not entirely noise-dominated, and so we first apply a high-pass filter to the correction vector before applying it to the solution search vector. For the largest scales ($\ell \leqslant 40$), we do not project to pixel domain either, but simply solve $\mathbf{Ax} = \mathbf{b}$ restricted to $\ell \leqslant 40$ by explicitly computing the matrix entries and using a simple Cholesky solver.

For the top solver level, we need to compute the diagonal of $\mathbf{Y}_{\mathrm{obs}}^T \mathbf{N}^{-1} \mathbf{Y}_{\mathrm{obs}}$ in SH domain, and for the bottom solver level we similarly need all entries of $\mathbf{Y}_{\mathrm{obs}}^T \mathbf{N}^{-1} \mathbf{Y}_{\mathrm{obs}}$ for ℓ up to some ℓ_{dense}. While such entries can be computed using Wigner 3j-symbols (Hivon et al. 2002; Eriksen et al. 2004), the following procedure has some significant advantages. First, while the computational scaling is the same, it is much faster in practice, in particular due to the optimized code for associated Legendre polynomials $P_{\ell m}$ available in `libpsht` (Reinecke 2011). Second, it is accurate to almost machine precision for any grid, whereas the method relying on Wigner 3j-symbols relies on approximation by evaluation of an integral, and is therefore inaccurate for low-resolution HEALPix grids.

Let ξ_{kj} be the jth of J_k pixels on ring k in the masked inverse-noise map. One can then evaluate

$$\begin{aligned}
&\left(\mathbf{Y}_{\mathrm{obs}}^T \mathbf{N}^{-1} \mathbf{Y}_{\mathrm{obs}}\right)_{\ell_1 m_1, \ell_2 m_2} \\
&\quad = \sum_k \sum_{j=1}^{J_k} \xi_{kj} Y_{\ell_1 m_1}(\theta_j, \phi_{kj}) Y^*_{\ell_2 m_2}(\theta_j, \phi_{kj}) \\
&\quad = \sum_k \widetilde{P}_{\ell_1 m_1}(\cos\theta_k) \widetilde{P}_{\ell_2 m_2}(\cos\theta_k) \sum_{j=1}^{J_k} \xi_{kj} e^{i(m_1 - m_2)\phi_{kj}},
\end{aligned}$$

where the normalized associated Legendre function is

$$\widetilde{P}_{\ell m}(\cos\theta) = \sqrt{\frac{2\ell+1}{4\pi} \frac{(\ell-m)!}{(\ell+m)!}} P_{\ell m}(\cos\theta). \tag{19}$$

The inner sum can be precomputed for each ring k and every $(m_1 - m_2)$ by discrete Fourier transforms, allowing the evaluation of matrix elements in $O(N_{\mathrm{ring}}) = O(\ell_{\mathrm{max}})$ time. In the case that we want a dense low-ℓ block, the procedure to downgrade the inverse-noise operator from Section 3.4 should be applied first, to reduce the computational cost from $O(\ell_{\mathrm{dense}}^2 \ell_{\mathrm{max}})$ to $O(\ell_{\mathrm{dense}}^3)$.

3.3. Filter Selection and Pixel-domain Localization

So far we have not specified the exact form of the low-pass filters $\tilde{f}_\ell^h$ required for every level. It turns out that careful selection of these filters is essential to ensure that the pixel projection of $\mathbf{A}_h$ is localized, and hence that the construction of an efficient error smoother is possible.

As indicated in Equation (13), the SH system $\mathbf{A}_h$ on each level h is projected to pixel domain with

$$\widehat{\mathbf{A}}_h \equiv \mathbf{Y}_h \mathbf{A}_h \mathbf{Y}_h^T = \widehat{\mathbf{D}}_h + \widehat{\mathbf{B}}_h^T \mathbf{N}^{-1} \widehat{\mathbf{B}}_h, \tag{20}$$

where the prior and pixelized beam terms are this time given by (respectively)

$$\widehat{\mathbf{D}}_h = \mathbf{Y}_h \mathbf{F}_h \mathbf{S}^{-1} \mathbf{F}_h \mathbf{Y}_h^T \tag{21}$$

$$\widehat{\mathbf{B}}_h = \mathbf{Y}_{\mathrm{obs}} \mathbf{B}\, \mathbf{F}_h \mathbf{Y}_h^T. \tag{22}$$

Note that the pixelization along the rows of $\widehat{\mathbf{B}}_h$ is the observational grid, while the pixelization down the columns is that of the current level.

The matrices $\widehat{\mathbf{D}}_h$ and $\widehat{\mathbf{B}}_h$ are both rotationally invariant. By the addition theorem of SHs, the coupling strength between two points on the sphere separated by angular distance θ is given by

$$g(\theta) = \sum_\ell \frac{2\ell+1}{4\pi} g_\ell P_\ell(\cos\theta), \tag{23}$$

where we insert $g_\ell = (\tilde{f}_\ell^h)^2 / C_\ell$ for $\widehat{\mathbf{D}}_h$ and $g_\ell = \tilde{f}_\ell^h b_\ell p_\ell$ for $\widehat{\mathbf{B}}_h$. The pixel-domain localization of such matrices depends entirely on g_ℓ. In our experience, the g_ℓ that lead to localized matrices in pixel domain tend to be flat or polynomially increasing before an exponential drop. Since b_ℓ already describes a localized beam, and $1/C_\ell$ increases non-exponentially, crafting a localized system $\widehat{\mathbf{A}}_h$ at each level is indeed possible.

Selecting the filters $f_{h,\ell}^H$, whose products form $\tilde{f}_\ell^h$ and $\mathbf{F}_h$ for each level, is a non-trivial matter. The main characteristic the filters must have is that each $\tilde{f}_\ell^h$ falls off quickly enough in real space to avoid strong couplings between the edge of the mask and the interior. Figure 6 shows what happens if this is not controlled correctly—the long-range couplings make the construction of an error smoother $\widehat{\mathbf{M}}$ impossible. In contrast, Figure 7 shows the behavior of the operators in the well-tuned case.

A filter that we found to work very well is given by squaring the exponent of a Gaussian,

$$q_\ell = \exp(-\ell^2(\ell+1)^2\lambda). \tag{24}$$

The scale parameter λ is simply chosen from the scale behavior that we want. In our test runs, we chose the constraints $q_{2570} = 0.1$ at the $N_{\mathrm{side}}^h = 1024$ level and $q_{1536} = 0.1$ at the $N_{\mathrm{side}}^h = 512$ level.

This filter has the following advantages over a simple Gaussian:

1. It decays much more quickly in ℓ, while in real space it decays almost as quickly in the tails as the Gaussian. This allows us to avoid increasing the bandlimit of the original system beyond $\ell_{\mathrm{max}} = 3000$.
2. The rapid decay with ℓ is also beneficial to counter the behavior of $1/C_\ell$. In the range $2000 < \ell < 3000$, $1/C_\ell$ follows a rather steep trajectory (between $\sim\ell^7$ and ℓ^8) which, when only countered by a Gaussian, causes some ringing and less locality.

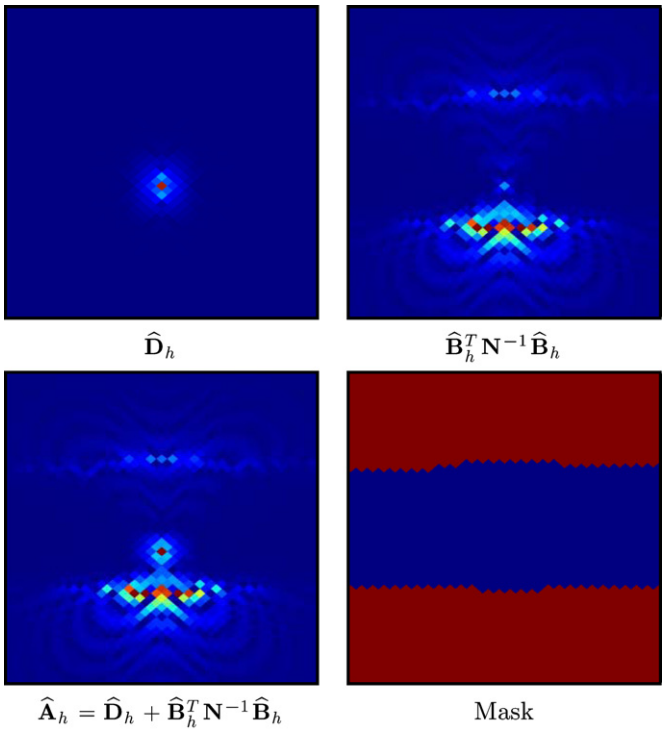

Figure 6. Effect of a poor choice of filter $\tilde{f}_\ell^h$. Each panel shows the couplings between a single pixel and its neighboring region, corresponding to a row/column of $\widehat{\mathbf{A}}_h$. In this case we used a low-pass filter based on modifying a standard needlet (Scodeller et al. 2011, and references therein). While the harmonic properties of this filter were very attractive, the tails do not decay quickly enough in real space. The resulting strong, long-range couplings are fatal to our algorithm.

(A color version of this figure is available in the online journal.)

3. Using Gaussian filters shapes the $\mathbf{N}^{-1}$ term so that couplings around a given pixel are similar to a Gaussian with FWHM of 4 pixels. That is, the couplings between neighboring pixels are rather strong. The filter defined above produces much weaker couplings between neighbors. This is not currently an advantage, because we let every pixel "see" a radius of $k = 8$ pixels around itself anyway in the error smoother. However, it could become an advantage in the future if k is chosen adaptively for each pixel.

Despite these features, the simple Gaussian filter behaved better at the coarser levels with very high signal-to-noise, as can be seen by comparing the second panel of Figure 7 with the first panel of Figure 8. In our tests we chose a Gaussian filter $f_{h,\ell}^H$ for levels $N_{\rm side}^h \leqslant 256$, tuned so that the cumulative filter $\tilde{f}_\ell^h$ on each level roughly corresponds to a Gaussian with FWHM of 2 pixels.

3.4. Band-limitation and Coarsening $\boldsymbol{Y}^T \boldsymbol{N}^{-1} \boldsymbol{Y}$

Figure 9 shows the effect of choosing the bandlimit $\ell_{\rm max}^h$ too low. On the coarser levels, ringing from the inverse-noise term causes strong non-local couplings unless the bandlimit is set as high as $6N_{\rm side}^h$. This limit depends on the signal-to-noise ratio, and $\ell_{\rm max} = 4N_{\rm side}^h$ is sufficient on the $N_{\rm side} = 512$ level.

The HEALPix grid can only represent a field accurately up to $\ell_{\rm max}^h \sim 2N_{\rm side}^h$, and will in fact see different scales on different parts of the sphere, due to the necessary irregularities in the pixelization. This is the primary reason for the non-traditional level traversal structure chosen in Section 3.2. The pixel projection operator $\mathbf{Y}_h$ removes some parts of the projected field that the grid cannot represent, but this is after all how a multi-level restriction normally works, and so poses no problems.

The filter $\tilde{f}_\ell^h$ allows us to set $\ell_{\rm max}^h$ much lower than the full $\ell_{\rm max}$. The two SHTs involved in $\mathbf{Y}_{\rm obs}^T \mathbf{N}^{-1} \mathbf{Y}_{\rm obs}$ still involve an $N_{\rm side} = 2048$ grid, however, so the coarsest levels are still almost as computationally expensive as the finest levels.

To work around this, the key is to note that the operator $\mathbf{Y}_{\rm obs}^T \mathbf{N}^{-1} \mathbf{Y}_{\rm obs}$ does not "see" scales in the inverse-noise map beyond $2\ell_{\rm max}^h$. This follows from an expansion into Wigner 3j-symbols (Hivon et al. 2002; Eriksen et al. 2004). Simply degrading the inverse-noise map to a coarser resolution HEALPix grid was found to be far too inaccurate, so more care is needed.

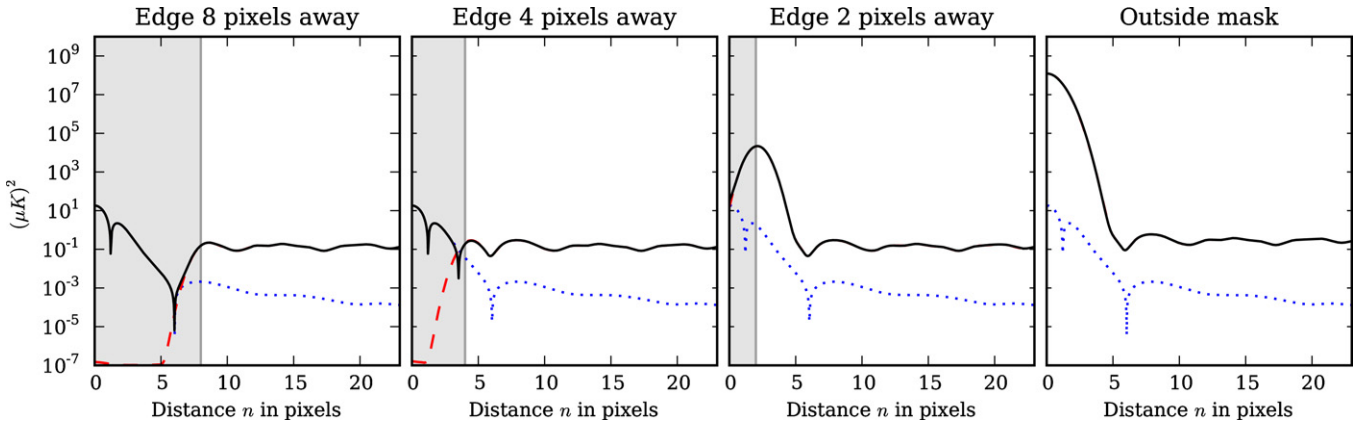

Figure 7. Effect of the mask on $\widehat{\mathbf{A}}_h$. Each panel shows the coupling strength in absolute value in the $\widehat{\mathbf{A}}_h$ operator, between a sample point at (θ, ϕ) (plotted at the origin), and another sample point n pixels away at $(\theta, \phi + n\Delta)$, where Δ is the angular size of one pixel. The couplings of $\widehat{\mathbf{A}}_h$ (black) are a sum of the prior term $\widehat{\mathbf{D}}_h$ (dotted blue) and the inverse-noise term $\widehat{\mathbf{B}}_h^T \mathbf{N}^{-1} \widehat{\mathbf{B}}_h$ (dashed red). For each panel, we vary the position of (θ, ϕ) relative to the mask (gray band), so that the origin is in each case a value on the diagonal of $\widehat{\mathbf{A}}_h$. Displayed here is our $N_{\rm side}^h = 32$ level in the case of 1.9 μK constant rms noise (the minimum rms level of the Planck 143 GHz band). The filter $\tilde{f}_\ell$ is a product of all the inter-level filters $f_{H,\ell}^h$ (as described in the text), but corresponds roughly to a Gaussian with FWHM of 2 pixels divided by the pixel window p_ℓ. The "floor" at 10^{-1} is caused by the non-Gaussian features of the instrumental beam, b_ℓ. For comparison, a perfect Gaussian instrumental beam is used in Figure 9.

(A color version of this figure is available in the online journal.)

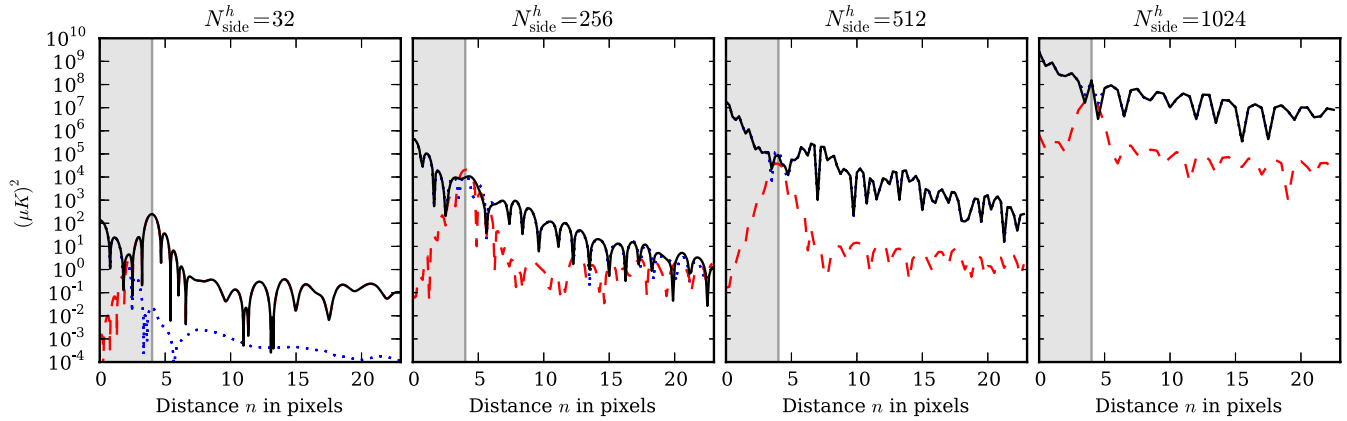

Figure 8. Effect of resolution on $\widehat{\mathbf{A}}_h$. See Figure 7 for legend and experimental setup. In this figure, we also show the effect of the filter q_ℓ of Equation (24), with λ appropriately tuned for the resolution in each case. As the resolution is increased, the signal-to-noise ratio decreases, making the influence of the edge of the mask less important.

(A color version of this figure is available in the online journal.)

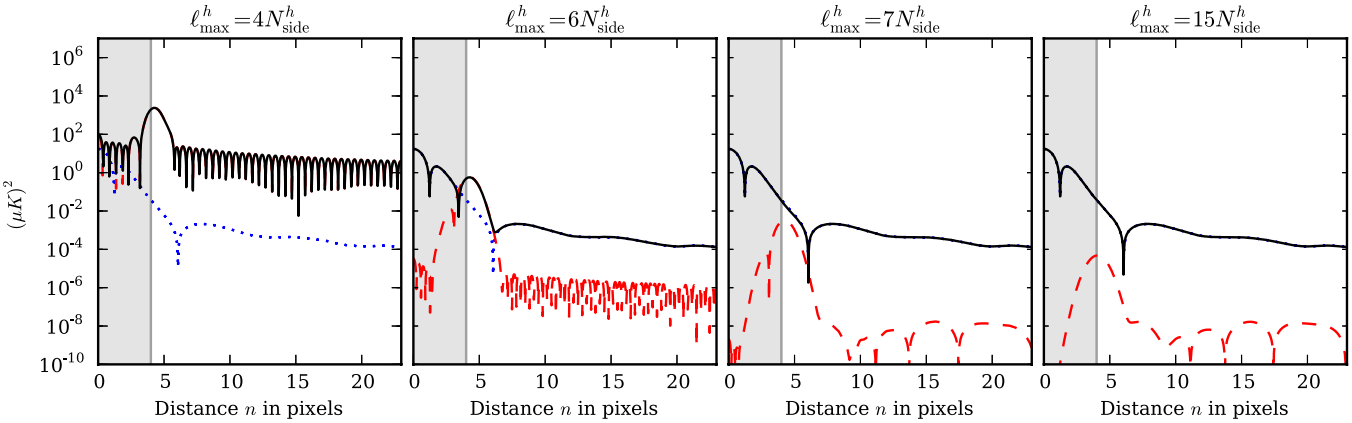

Figure 9. Effect of the band-limit $\ell^h_{\rm max}$ on $\widehat{\mathbf{A}}_h$. See Figure 7 for legend and experimental setup. The settings for each panel are the same except for varying $\ell^h_{\rm max}$. Here, the product $\tilde{f}^h_\ell b_\ell p_\ell$ is a pure Gaussian with FWHM of 2 pixels. Since the instrumental beam is in this case taken to be a perfect Gaussian, there is also no "floor" at 10^{-1} (compare with Figure 7 for the effect of a non-Gaussian beam).

(A color version of this figure is available in the online journal.)

First, we rewrite the operator as

$$\mathbf{Y}^T_{\rm obs}\mathbf{N}^{-1}\mathbf{Y}_{\rm obs} = \mathbf{Y}^T_{\rm obs}\mathbf{W}_{\rm obs}\left(\mathbf{W}^{-1}_{\rm obs}\mathbf{N}^{-1}\right)\mathbf{Y}_{\rm obs}, \tag{25}$$

where $\mathbf{W}_{\rm obs}$ denotes the quadrature weights of the HEALPix $N_{\rm side} = 2048$ grid, so that $\mathbf{Y}^T_{\rm obs}\mathbf{W}_{\rm obs}$ corresponds to SH analysis, as described in Section 2.1. Then, we write ξ_i for the pixels on the diagonal of $\mathbf{W}^{-1}\mathbf{N}^{-1}$, and $\xi_{\ell m}$ for the same map expanded into SHs. Since the operator of Equation (25) does not see coefficients beyond $2\ell^h_{\rm max}$, we can truncate $\xi_{\ell m}$ and project it onto a Gauss–Legendre grid of the same order, which (unlike HEALPix grids) allows SH analysis that is accurate to almost machine precision. Using this re-weighted and downgraded inverse-noise map as the diagonal of a new inverse-noise matrix $\tilde{\mathbf{N}}^{-1}_h$, we have that

$$\mathbf{Y}^T_{\rm obs}\mathbf{N}^{-1}\mathbf{Y}_{\rm obs} = \tilde{\mathbf{Y}}^T_h\tilde{\mathbf{W}}_h\tilde{\mathbf{N}}^{-1}_h\tilde{\mathbf{Y}}_h, \tag{26}$$

where $\tilde{\mathbf{Y}}_h$ and $\tilde{\mathbf{Y}}^T_h\tilde{\mathbf{W}}_h$ indicate SH synthesis and analysis on the Gauss–Legendre grid.

3.5. Error Smoother Construction for the CR System

A simple diagonal error smoother does not converge in our setup, primarily because pixels on the edge of the mask can have a very strong influence on the solution in the interior of the mask, as seen in Figure 7. Also, when applying Gaussian filters, the couplings between neighboring pixels are rather strong, preventing the use of a diagonal error smoother even far from the mask.

The basic strategy for our error smoother is to make sure that every pixel "sees" neighboring pixels in some radius k around it. In our case we let $k = 8$ on all levels, although improvements on this may be possible, especially in cases with lower signal-to-noise than ours.

We start by dividing the sphere into tiles of size k-by-k. Then, we include the couplings between pixels in the same and neighboring tiles while ignoring any couplings between pixels further apart, so that couplings are included in a radius of at least k pixels around every pixel. The result is a block sparse matrix, as shown in Figure 10.

Next, we explicitly compute the parts of $\widehat{\mathbf{D}}_h$ (Equation (21)) and $\widehat{\mathbf{B}}_h$ (Equation (22)) that fall within the sparsity pattern by evaluating the sum over Legendre polynomials from Equation (23). After preparing the block sparse matrix approximations, we use matrix multiplication without fill-in to compute $\widehat{\mathbf{B}}^T\mathbf{N}^{-1}\widehat{\mathbf{B}}$—that is, we neglect resulting blocks outside of the same sparsity pattern. The approximant for $\widehat{\mathbf{D}}_h$ can then be added directly. Finally, we perform a zero-fill-in incomplete Cholesky

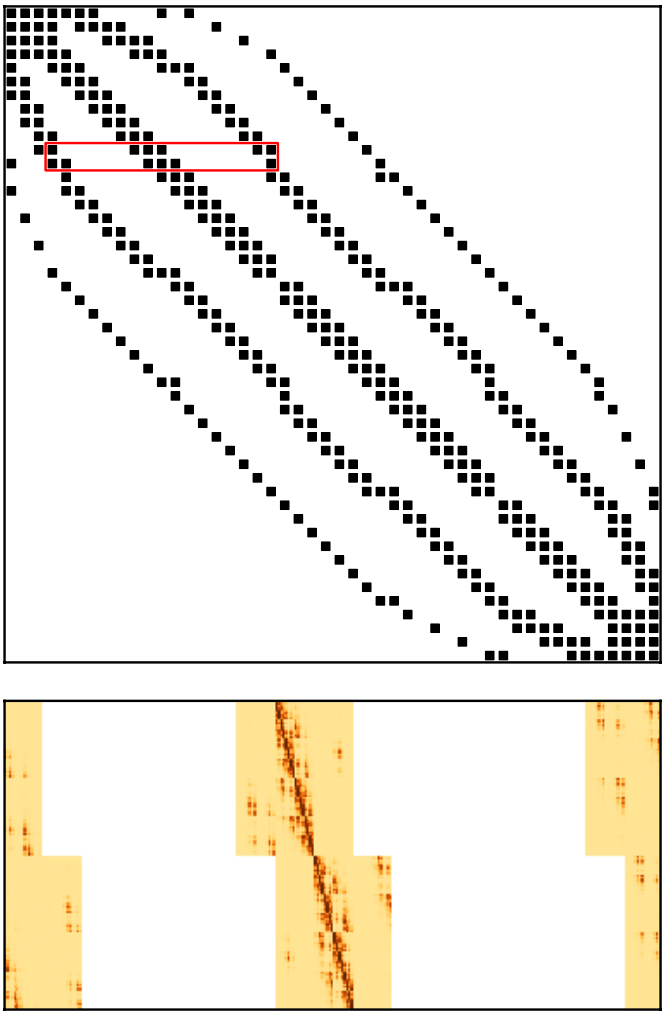

Figure 10. Structure of the block sparse matrices used in the error smoothers. Top panel: the sparsity pattern when every tile is coupled to its eight neighboring tiles. In this case, the pattern of tiles is an $N_{\rm side} = 2$ HEALPix grid in ring-ordering. Bottom panel: the blocks of $\widehat{\mathbf{B}}_h = \mathbf{Y}_{\rm obs}\mathbf{B}\mathbf{Y}_h^T$ corresponding to the red rectangle in the top panel. The blocks on the diagonal contain within-tile couplings, while off-diagonal blocks are couplings between pixels in neighboring tiles. Each block is rectangular because $\mathbf{Y}_{\rm obs}$ samples on a grid with 4× more pixels than the grid sampled by $\mathbf{Y}_h$.

(A color version of this figure is available in the online journal.)

factorization (ICC), i.e., we perform in-place Cholesky factorization of the block sparse approximant as usual, but ignore any element updates outside of the sparsity pattern during the factorization process.

Without modification, the factorization process usually fails, either due to the sparse approximant of the full dense matrix ending up non-positive-definite, or because of elements dropped during the ICC. When this happens, we do a binary search for the lowest ridge adjustment α that, when added to the diagonal, makes the factorization procedure succeed, and scale this α by a factor of 1.5 for the final factorization. Typical ridge values α are in the range 10^{-2} to 10^{-4} times the maximum element of $\mathbf{A}_h$.

After factorization, applying the smoother is simply a matter of doing the usual triangular solve. This is an inherently sequential process, and the smoother therefore currently runs on a single CPU core. Since an error smoother only needs to work locally, we expect to be able to apply domain decomposition techniques, partitioning the sphere into large domains that overlap by k or $2k$ pixels, and applying one error smoother on each domain. Proper parallelization of the error smoother is left for future work, however. Also note that the process described above is the very simplest incomplete factorization algorithm, and more sophisticated incomplete factorization algorithms are standard in the literature.

In Section 4.1, we quote numbers for the execution time and memory usage of the smoother. One possibility for reducing memory consumption in the future is to let k be adaptive, as it can be made smaller away from the edges of the mask. All error smoother computations are done in single precision. In the current implementation, computing $\widehat{\mathbf{B}}$ is very expensive, as we sample it directly on the $N_{\rm side} = 2048$ grid. This is not a fundamental scaling problem, but rather an issue of implementation, as the degraded inverse-noise map on the Gauss–Legendre grid described in Section 3.4 could also be used in this setting.

4. IMPLEMENTATION AND RESULTS

4.1. Numerical Results and Performance

The basic assumptions for our experimental setup have already been laid out in Section 2.2. We choose for our example the rms map and symmetric beam approximation of the 143 GHz channel of Planck, as provided in the Planck 2013 data release (Planck Collaboration 2013a).

We tried running both with the 40%-sky, 80%-sky, and 97%-sky masks used in the Planck analysis, in all cases together with the 143 GHz point source mask. The mask has some impact on speed of convergence, but not enough to warrant attention, and we therefore only present the results from the 80%-sky mask, which was the *slowest* to converge.

For the power spectrum, C_ℓ, we use the standard best-fit Planck+WP+high-ℓ six-parameter ΛCDM spectrum (Planck Collaboration 2013d), but set C_0 and C_1 to the value of C_2 as a wide prior for any residual monopole or dipole component. Statistically, the prior for the monopole and dipole is of little relevance, since the data so strongly constrain these components. Note that the present algorithm will not let us condition on a given monopole and dipole (i.e., set $C_0 = C_1 = 0$), at least without modifications.

To produce the right-hand side, $\mathbf{b}$, corresponding to a random test realization, we draw a simulated $\mathbf{x}_{\rm true}$ from the prior $p(\mathbf{s}|C_\ell)$, and multiply it with $\mathbf{A}$ of Equation (4). This synthetic setup allows us to track the true error, $\mathbf{e} = \mathbf{x}_{\rm true} - \mathbf{x}$. In a real setting the right-hand side is of course generated from observed data, and in this case one can only track the residual, $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$.

The error smoothers are least efficient on the largest scales. At the same time, these are much cheaper to process than the small-scale smoothers due to the $O(\ell_{\rm max}^3)$ scaling of the SHTs. We therefore choose a partial W-cycle, where the levels for $N_{\rm side}^h \leqslant 1024$ participate in a W-cycle ($n_{\rm rec}^h = 2$ in Figure 5), but the very expensive error smoother of the $N_{\rm side}^h = 1024$ level, as well as SHTs at $\ell_{\rm max} = 3000$, are only run once on the way down and once on the way up (a V-cycle).

In Figure 11 we plot the resulting convergence, in terms of absolute error as a function of W-cycle iteration count. Here we see that the error falls exponentially with cycle count, at the rate of roughly one order of magnitude per iteration. The largest error anywhere on the sky is smaller than 1 μK after only three W-cycles, and approaches the numerical precision limit after eight cycles.

Table 1
Structure and Computational Cost of a W-cycle

Level	$\ell_{\max}$	No. of Visits	Time in $\mathbf{Y}_h$ (Wall s)	Time in $\mathbf{Y}_{\rm obs}$ (Wall s)	Time in $\widehat{\mathbf{M}}_h$ or $\mathbf{A}_h^{-1}$ (Wall s)	Total Time (Wall s)
$\ell_{\max} = 3000$	3000	1	···	6×2.8	···	16.8
$N_{\rm side} = 1024$	3000	1	4×1.2	4×2.8	2×11	38.0
$N_{\rm side} = 512$	2048	2	8×0.3	10×1.3	4×2.3	24.6
$N_{\rm side} = 256$	1280	4	16×0.07	20×0.40	8×0.57	13.7
$N_{\rm side} = 128$	768	8	32×0.016	40×0.10	16×0.14	6.75
$N_{\rm side} = 64$	384	16	64×0.004	80×0.03	32×0.035	3.78
$N_{\rm side} = 32$	224	32	128×0.002	160×0.008	64×0.009	2.11
$\ell_{\max} = 40$	40	32	···	···	32×0.028	0.90
Other work						8
Full W-cycle						114

Notes. All times are given in wall time seconds using 16 CPU cores. The total number of operations of each kind for the W-cycle is indicated in each case; this number is not a multiple of the number of visits because the input vector $\mathbf{x}$ is zero on the first visit (except on the first level). Ignoring this aspect, each pixel level requires: (1) two-level-transfer spherical harmonic transforms ($\mathbf{Y}_h$), (2) three multiplications with $\mathbf{A}_h$, each with two inverse-noise spherical harmonic transforms ($\mathbf{Y}_{\rm obs}$), and (3) two applications of the error smoother $\widehat{\mathbf{M}}$. The top spherical harmonic level also requires two applications of $\mathbf{A}_h$, while the smoother application time is negligible. The bottom spherical harmonic level consists only of dense triangular solves.

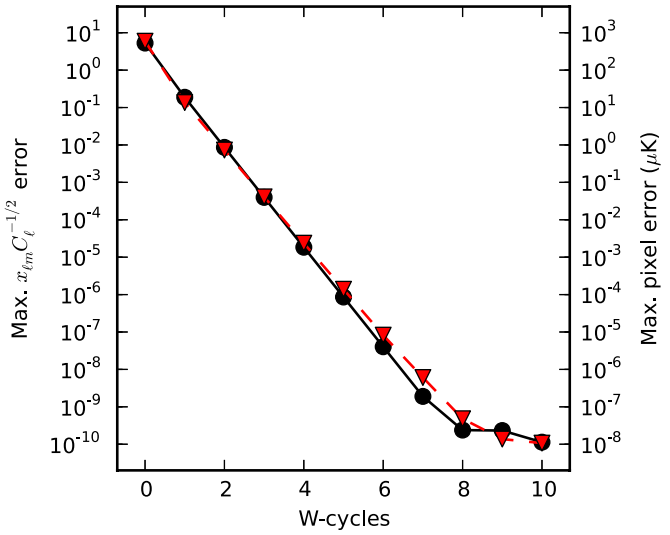

Figure 11. Absolute errors as a function of W-cycle count. For every iteration we plot the maximum error over all $C_\ell^{-1/2} x_{\ell m}$ (black circles, left axis), as well as the largest error across all pixels (red triangles, right axis).
(A color version of this figure is available in the online journal.)

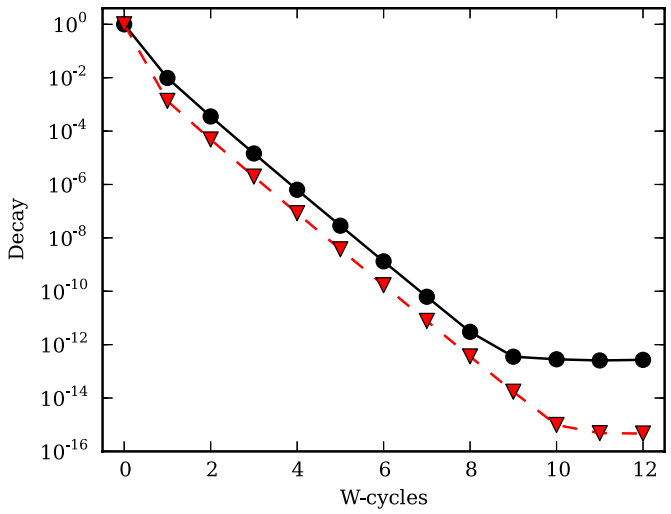

Figure 12. Comparison of absolute errors relative to C_ℓ, $(\mathbf{x}-\mathbf{x}_{\rm true})^T \mathbf{S}^{-1} (\mathbf{x}-\mathbf{x}_{\rm true})$ (black circles), and similarly scaled residuals, $(\mathbf{b} - \mathbf{A}\mathbf{x})^T \mathbf{S}^{-1} (\mathbf{b} - \mathbf{A}\mathbf{x})$ (red triangles). Both are normalized with respect to the initial error/residual. The two quantities behave very similarly, implying that the residual is an excellent proxy for the true error.
(A color version of this figure is available in the online journal.)

As mentioned above, since we know what the true solution is for the simulated data, we are also able to trace the absolute error, $\mathbf{e} = \mathbf{x}_{\rm true} - \mathbf{x}$, although only the residual, $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$, is available in real-world applications. Figure 12 shows that these have qualitatively very similar behavior as a function of W-cycle count, which implies that the residual can be used as a robust proxy for the actual error for the multi-level algorithm. The same is not true for the CG method, for which the error can flatten earlier than the residual due to the presence of the nearly singular modes in $\mathbf{A}$.

Finally, in Figure 13 we show the relative error as a function of multipole moment and W-cycle count. This plot highlights the problematic angular scales, and is therefore particularly useful during the debugging and tuning phase of the analysis; for example, the use of a V-cycle rather than a W-cycle would make the large scales noticeably lag behind in convergence on this plot. Another example is that, if the filters $\tilde{f}_h$ are poorly tuned (potentially causing the method to diverge), the responsible level can often be picked out on this plot.

The total run-time for this setup was 114 s wall time per W-cycle on 16 CPU cores (AMD 6282 running at 2.6 GHz). Table 1 breaks this cost down further to the individual levels and actions. The bulk of the memory use is by the error smoothers, which consume about 20 GiB of memory (see Table 2). The total process footprint was around 30 GiB, although unnecessary temporary arrays abound in the current implementation.

Table 2 presents the cost of the necessary precomputations. For every new combination of instrumental beam, noise map and mask, or for a new choice of multi-level filters $f_{h,\ell}^H$, one needs to precompute an approximation to $\widehat{\mathbf{B}}_h^T \mathbf{N}^{-1} \widehat{\mathbf{B}}_h$ for every solver level. These precomputations required a total of 44 CPU hours in our tests, but are trivially parallel. We also expect that one will usually load the results from disk. The approximation

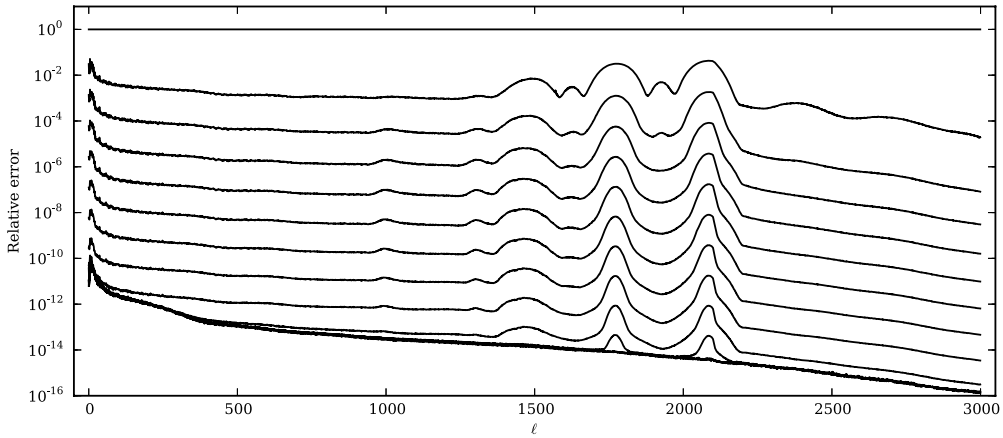

Figure 13. Relative error as a function of angular scale. Starting from the top, each line shows the error for a given multi-level W-cycle. Specifically, we plot $\|\mathbf{x}_{\mathrm{true},\ell} - \mathbf{x}_\ell\|/\|\mathbf{x}_{\mathrm{true},\ell}\|$, where $\mathbf{x}_\ell$ denotes a vector with the coefficients for a given ℓ only. This plot is especially useful during development and tuning of the code, as one can immediately see which error smoothers do not perform well.

Table 2
Error Smoother Precomputation Cost/Memory Use per Solver Level

N_{side}	Time Obs. (CPU minutes)	Time C_ℓ (CPU minutes)	Time ICC (CPU minutes)	Mem. Use (GiB)
1024	727	85	1.15	15
512	509	15	0.35	3.7
256	340	2.4	0.10	0.93
128	230	0.36	0.02	0.23
64	452	0.05	0.007	0.058
32	363	0.01	0.002	0.014
Total	2621	103	1.6	20

Notes. All times are given in CPU minutes (wall time times the number of CPU cores used). Precomputations can be divided into the part that must be performed whenever the observational setup (beam/mask/noise map) changes and the part that must be performed whenever the prior (C_ℓ) changes. If any part changes, the non-parallel incomplete Cholesky factorization (ICC) must also be performed again.

for $\widehat{\mathbf{D}}_h$ must be recomputed every time C_ℓ changes, which in the case of Gibbs sampling means every time one wants to run the solver. Fortunately, this computation is much cheaper and only requires around 100 CPU minutes of trivially parallel work, plus 2 minutes of non-parallel work. We argue in Section 5 that it should be possible to greatly decrease precomputation time in future.

The main weakness in the current implementation is the lack of parallelization in the error smoothers. Not only does the code need to be run on a single node, but the 40 s spent on error smoothing runs on a single CPU core, with the 15 other cores idling. Parallelization of the smoother would bring the wall time much closer to 80 s, as well as allowing the distribution of the 20 GiB of smoother data among several cluster nodes.

4.2. Notes on Implementation and Dependencies

The CR solver is part of Commander 2, which is made available as open source software under the BSD license (core code) and the GNU General Public License (GPL) (full software when including dependencies). For more information, see http://commander.bitbucket.org/.

Commander 2 is implemented in a mixture of Python (using NumPy and SciPy), Cython (Behnel et al. 2011), Fortran 90, and C. For SHTs we use `libsharp` (Reinecke & Seljebotn 2013). For our benchmarks we have used OpenBLAS (Goto & van de Geijn 2008; Xianyi et al. 2012) for linear algebra.

The main computation time is spent in `libsharp` or OpenBLAS, and as such is already highly optimized. The computation of Equation (23) benefited greatly from being structured as described in the appendix of Seljebotn (2012). In addition to what is mentioned there, we made use of the AVX and FMA4 instruction sets. Also, note that all the computations for the error smoother could be performed in single precision.

5. DISCUSSION

We have presented a new algorithm for solving the Gaussian CR system for high-resolution CMB data. This method is based on ideas from multi-grid (or multi-level) theory, and is fundamentally different from the CG methods traditionally used for this problem. Being only weakly dependent on the signal-to-noise ratio of the data set under consideration, our new method converges exponentially to numerical precision when properly tuned, and is capable of producing CRs for the full resolution of a Planck-like data set within minutes. For comparison, we have yet to achieve robust full-sky convergence with CG methods for the same data set. Indeed, this particular issue was the single most important obstacle preventing a full-resolution analysis of the Planck 2013 data release with the Commander code.

The ultimate goal of this line of work is to perform an exact global Bayesian analysis of the high-resolution, high-sensitivity observations now being produced by CMB experiments, including component separation as described by Eriksen et al. (2008b). For this to be successful, multi-frequency and multi-component analysis must be added to the algorithm. Other complications, such as the possible asymmetry of the CMB on large scales (e.g., Planck Collaboration 2013c), will also need to be taken into account. As such, the present paper represents only the first step toward a complete solution. We also emphasize that the algorithm as presented here is only the first implementation of a more general framework, and we expect that many improvements with respect to computational speed, application to more general cases, overall robustness and stability, and even user interfaces, will be introduced in the near future. Before concluding this paper, we will mention a few relevant ideas, but leave all details for future publications.

Firstly, as is evident from Figure 7, our method is quite sensitive to the behavior of the tails of the instrumental beams

extending as far out as the 10^{-5} level, as these formally constrain the solution inside the mask. These tails are not realistically known to such high accuracy, and so this issue is therefore a modeling problem as well as a numerical problem. In practice, it seems that in the absence of other options, one should just choose a form for the tails that falls quickly enough to not have an effect on the solution, and that allows a small computational bandlimit, $\ell_{\rm max}$. In short, optimally tuning the tails of the beam profile may render a more stable solution at a lower computational cost.

For an exact analysis of data from current and forthcoming CMB experiments, one would ideally like to account for the effect of asymmetric beams. With the above in mind, we envision two solutions for this. One option is to modify the algorithm so that the beams are defined in pixel space, as is done in FEBECop (Mitra et al. 2011) for instance, and then carry the FEBECop beams through to the computation of the smoother. The main challenge in this scenario is how to avoid very expensive matrix–vector multiplications at the coarse levels. Alternatively, and perhaps more simply, one could use the multi-level solver for perfect symmetric beams described here as a preconditioner for a CG search, which then accounts for the beam asymmetries in its own internal matrix multiplications.

Correlated noise is another significant complication for current CMB observations. While these correlations have a complicated morphology in pixel space, being convolved with the scanning strategy of the experiment, they are simple to describe in the time-domain. With the vastly improved convergence rate of the multi-level method presented here—requiring only a handful of iterations to reach sub-μK errors—it may for the first time be realistic to define the CR system in time-domain, rather than map-domain. As for asymmetric beams, this can either be done by defining the multi-level scheme directly in time-domain, or, if that does not succeed, by using the multi-level solver for uncorrelated noise as a preconditioner for a time-domain CG search. Going to time-domain also provides a direct route to handling beam asymmetries and optical sidelobes by full-sky convolution (Wandelt & Górski 2001).

The current computational bottleneck in our implementation is the time needed to precompute the error smoothers. The time is spent almost exclusively on sampling rotationally invariant operators at every position on the sphere by brute force evaluation of Equation (23). While the code for this computation is already highly optimized, as mentioned above, we do not exploit any symmetries from pixel to pixel. The grid used within the multi-level process is arbitrary, and not necessarily related to the grid of the inverse-noise map, **N**, or data vector, **d**. A future implementation of the algorithm will therefore employ a different grid with greater symmetry than the HEALPix grid, which will only require evaluation of the smoother blocks 3–7 times per pixel ring, thus reducing the computational scaling from $O(k^2\ell_{\rm max}N_{\rm pix})$ to $O(k^2\ell_{\rm max}\sqrt{N_{\rm pix}})$.

Finally, the error smoother evaluation is currently not parallelized, and only executes on a single CPU core. As the error smoothers only need to work well for the local couplings, we expect to be able to partition the sphere into multiple partially overlapping domains, and apply an error smoother on each domain in parallel, at the cost of some extra computation on the domain borders. Assuming that this approach is successful, the SHTs will once again become the bottleneck of the overall algorithm.

We thank Mikolaj Szydlarski and Martin Reinecke for useful discussions. D.S.S., H.K.E., and P.B. are supported by European Research Council grant StG2010-257080. K.A.M. is supported by the Research Council of Norway through a Centre of Excellence grant to the Centre for Biomedical Computing at Simula Research Laboratory.

REFERENCES

Axelsson, O., & Lindskog, G. 1986a, NuMat, 48, 479
Axelsson, O., & Lindskog, G. 1986b, NuMat, 48, 499
Behnel, S., Bradshaw, R., Citro, C., et al. 2011, CSE, 13, 2
Bennett, C. L., Larson, D., Weiland, J. L., et al. 2013, ApJS, 208, 20
Brandt, A. 2011, in Multiscale and Multiresolution Methods, ed. T. J. Barth et al. (Berlin: Springer), 1
Doré, O., Teyssier, R., Bouchet, F. R., Vibert, D., & Prunet, S. 2001, A&A, 374, 358
Elsner, F., & Wandelt, B. D. 2012, in Proc. Big Bang, Big Data, Big Computers (arXiv:1211.0585)
Elsner, F., & Wandelt, B. D. 2013, A&A, 549, A111
Eriksen, H. K., Dickinson, C., Jewell, J. B., et al. 2008a, ApJL, 672, L87
Eriksen, H. K., Huey, G., Banday, A. J., et al. 2007a, ApJL, 665, L1
Eriksen, H. K., Huey, G., Saha, R., et al. 2007b, ApJ, 656, 641
Eriksen, H. K., Jewell, J. B., Dickinson, C., et al. 2008b, ApJ, 676, 10
Eriksen, H. K., O'Dwyer, I. J., Jewell, J. B., et al. 2004, ApJS, 155, 227
Górski, K. M., Hivon, E., Banday, A. J., et al. 2005, ApJ, 622, 759
Goto, K., & van de Geijn, R. 2008, ACM Trans. Math. Softw., 34, 3
Grigori, L., Stompor, R., & Szydlarski, M. 2012, in Proc. of the Int. Conf. on High Performance Computing, Networking, Storage and Analysis, ed. J. K. Hollingsworth (Los Alamitos, CA: IEEE Computer Society Press), 91
Hackbush, W. 1985, Multi-grid Methods and Applications (Berlin: Springer)
Havé, P., Masson, R., Nataf, F., et al. 2013, SIAM J. Sci. Comput., 35, 3
Hivon, E., Górski, K. M., Netterfield, C. B., et al. 2002, ApJ, 567, 2
Jewell, J., Levin, S., & Anderson, C. H. 2004, ApJ, 609, 1
Mitra, S., Rocha, G., Górski, K. M., et al. 2011, ApJS, 193, 5
O'Dwyer, I. J., Eriksen, H. K., Wandelt, B. D., et al. 2004, ApJL, 617, L99
Planck Collaboration , Ade, P. A. R., & Aghanim, N. 2013a, A&A, submitted (arXiv:1303.5062)
Planck Collaboration , Ade, P. A. R., & Aghanim, N. 2013b, A&A, submitted (arXiv:1303.5075)
Planck Collaboration , Ade, P. A. R., & Aghanim, N. 2013c, A&A, submitted (arXiv:1303.5083)
Planck Collaboration , Ade, P. A. R., & Aghanim, N. 2013d, A&A, submitted (arXiv:1303.5076)
Reinecke, M. 2011, A&A, 526, A108
Reinecke, M., & Seljebotn, D. S. 2013, A&A, 554, A112
Scodeller, S., Rudjord, Ø., Hansen, F. K., et al. 2011, ApJ, 733, 121
Seljebotn, D. S. 2012, ApJS, 199, 5
Shewchuk, J. R. 1994, http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.ps
Smith, K. M., Zahn, O., & Doré, O. 2007, PhRvD, 76, 043510
Tang, J. M., Nabben, R., Vuik, C., & Erlangga, Y. A. 2009, JSCom, 39, 3
Wandelt, B. D., & Górski, K. M. 2001, PhRvD, 63, 123002
Wandelt, B. D., Larson, D. L., & Lakshminarayanan, A. 2004, PhRvD, 70, 083511
Xianyi, Z., Qian, W., & Yunquan, Z. 2012, in IEEE 18th International Conference on Parallel and Distributed Systems (ICPADS), Model-driven Level 3 BLAS Performance Optimization on Loongson 3A Processor, ed. W. Cai, R. Siow Mong Goh, & M. Snir (Los Alamitos, CA: IEEE Computer Society Press), 684