All-sky map-making with Single-Dish Radio Telescopes

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF MASTER OF SCIENCE BY RESEARCH
IN THE FACULTY OF SCIENCE AND ENGINEERING

2020

By
Aishwarya Selvaraj
School of Physics and Astronomy

Contents

List of Tables

List of Figures

Abstract

Radio sky surveys using single-dish radio telescopes are conducted at frequencies ranging from 300 MHz to 857 GHz. Examples include the WMAP and *Planck* cosmic microwave background satellites, C-BASS, and the GMIMS polarization survey which uses Penticton in Canada. In all such surveys, the telescope measures the brightness distribution of the radio emission convolved with the radiation pattern of the telescope. For map-making, the sky is pixelated so that the measured data is distributed on the pixelated sky. However, the telescope scans the sky in a random pattern irrespective to the pixelated sky. So, in practice, the data measured is binned to the nearest pixel centre on the pixelated sky, where the pixel size is less than the telescope beam. This method, however, increases the noise in the resultant map and ignores the fact that the sky is a continuous function. In this thesis, we develop an interpolation-based map-making algorithm which takes into consideration the continuous sky function, so that a weighted intensity of the measured data is assigned to each pixel. The algorithm uses the ideal pixelization scheme *HEALPix* with a Gaussian distribution as the weight function. The all-sky map produced is less noisy than one made by binning but is computationally expensive. We report a standard deviation of 2.33×10^{-4} for the difference map and 2.78×10^{-3} for the smoothed binning map and a standard deviation of 2.23×10^{-4} for the difference map and 2.7×10^{-4} for the grid interpolation map. This dissertation is also extended to develop an elliptical interpolation function to approximately circularize an elliptical telescope beam.

Declaration

No portion of the work referred to in this dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Copyright

i. The author of this dissertation (including any appendices and/or schedules to this dissertation) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this dissertation, either in full or in extracts and whether in hard or electronic copy, may be made **only** in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the dissertation, for example graphs and tables ("Reproductions"), which may be described in this dissertation, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place

is available in the University IP Policy, in any relevant Thesis restriction declarations deposited in the University Library, The University Library's regulations and in The University's policy on Presentation of Theses.

Acknowledgements

Firstly, I would like to take this opportunity to thank my supervisor Dr John Patrick Leahy (Paddy) for providing me with the opportunity of pursuing my post-graduation at The University of Manchester under his supervision. It has been an absolute pleasure to work with him. Paddy has been a constant encouragement in conducting my research work throughout the year. His mentoring has helped me to improve my research skills and has motivated me towards pursuing a PhD degree.

Secondly, I would like to thank my PGR advisor Dr Neal Jackson for being a constant support and advisor during my course. I would also like to thank Roke Cepeda-Arroita, Adam Barr my group mates for welcoming me into the C-BASS team and providing me with all the support during the academic year. It wouldn't be fair if I don't mention my classmates for all the great time we had. In spite of the adverse situation, our constant interactions have been a real stress-buster.

Lastly and most importantly, I would like to thank my parents, sister, fiancée and friends for their constant support during my academic year. All this would have been impossible without each one of you.

Chapter 1

Introduction

1.1 Sky surveys

Surveying the sky is a fundamental activity in astronomy. In early times, astronomers carried out sky surveys by observing with naked eyes to identify and discover celestial objects. Charles Messier's catalogue from the middle of the 18th century, the Bonner Durchmusterung's star catalogue of the northern hemisphere and the Cordoba Durchmusterung of the southern hemisphere are a few among the many works in early observational sky surveys (Djorgovski et al. 2013). These sky surveys consisted of counting the stars and cataloguing its celestial features. However, with the advent of modern technology, the invention of imaging techniques revolutionized observational astronomy. The Smithsonian Astrophysical Observatory Catalogue, the Catalogue of Galaxies and of Clusters of Galaxies by Fritz Zwicky and his collaborators using an 18-inch Schmidt telescope, the first Palomar Observatory Sky Survey (POSS-I) and Second Palomar Observatory Sky Survey (POSS-II) are few of the many sky surveys in the photographic era.

In this dissertation, we are specifically interested in radio sky surveys. The history of radio astronomical observations began with the serendipitous dis-

covery of radio emission by Jansky (1932). A detailed description of the history of radio astronomy is given in Sullivan (2009). The first all-sky map, at 200 MHz was produced roughly 20 years later by Droge and Priester (Dröge & Priester 1956). The survey was made with the measurements from the northern hemisphere combined with the southern hemisphere data (Allen & Gum 1950) to produce the all-sky map. By the early 1950s various radio surveys in the frequency range of 100 to 400 MHz were published. This includes Landecker & Wielebinski (1970) at 150 MHz, Yates, Wielebinski & Landecker (1967) at 85 MHz, Turtle & Baldwin (1962) at 178 MHz etc. However, all of the early sky surveys had two serious limitations, low angular resolution and limited sensitivity. The 408 MHz all-sky radio survey was the first survey to reach a resolution of better than 1°. The successive survey at 408 MHz (Haslam, Quigley & Salter 1970) was done by the Mark I radio telescope at Jodrell Bank observing the Galactic anti-centre region in the northern hemisphere which was followed by Haslam et al. (1974). Max-Planck-Institute für Radioastronomie (MPIfR) observed the complementary northern region using Jodrell Bank and Haslam et al. (1975) covered the southern hemisphere using the Parkes telescope. The maps from these surveys were combined and published in Haslam et al. (1981, 1982). The 408 MHz surveys led to intense science study in radio frequencies (Phillipps et al. 1981; Beuermann, Kanbach & Berkhuijsen 1985), predicting the Galactic synchrotron contribution in Cosmic Microwave Background (CMB) studies. Therefore, the 408 MHz survey is considered as a milestone in radio sky survey. Sky surveys thereafter conducted includes 1.4 GHz northern sky survey with the Stockert 25-m telescope (Reich 1982; Reich & Reich 1986), 2.7 GHz map of Reich et al. (1984), 2.3 GHz southern survey (Jonas, De Jager & Baart 1985), 45 MHz survey (Alvarez et al. 1997; Maeda et al. 1999) etc. Such ground-based maps covered frequencies up to 2.4 GHz and satellites were used to cover higher frequencies.

1.2 CMB-based radio surveys

Satellite-based CMB mapping was first done by the COsmic Background Explorer (COBE) (Smoot et al. 1992). COBE, with a low-Earth orbit, carried three instruments: DIRBE (Diffuse InfraRed Background Experiment) (Hauser et al. 1998), DMR (Differential Microwave Radiometer) (Bennett et al. 1992) and FIRAS (Far InfraRed Absolute Spectrophotometer) (Mather, Fixsen & Shafer 1993) operating in the wavelength range of 1 μm to 1 cm. DMR observed at three wavelengths 3 mm, 6 mm, and 10 mm with an angular resolution of 7°. FIRAS measured the spectral distribution of the CMB in the range 0.1 – 10 mm with a 7° beam and DIRBE measured the infrared background radiation. The COBE satellite, with relatively close orbit to the earth successfully mapped the entire sky and detected intrinsic anisotropies in the CMB radiation of the order of $\sim 10^{-5}$. These anisotropies represent the fluctuations in the temperature from the epoch of recombination which would grow up to become galaxies and cluster of galaxies. The COBE mission was followed by Wilkinson Microwave Anisotropy Probe (WMAP) (Bennett et al. 2003) launched in 2001 by NASA. It made full-sky maps with an improved angular resolution of 0.2° covering frequencies centred at 22.8, 33.0, 40.7, 60.8 and 93.5 GHz. The cosmological results and analyses were improved with the data added every two years resulting in First-year (Bennett et al. 2003), Three-year (Spergel et al. 2007), Five-year (Hinshaw et al. 2009), Seven-year (Jarosik et al. 2011) and finally Nine-year results (Bennett et al. 2013). WMAP was followed by third generation CMB satellite *Planck* (Ade et al. 2011). It had two instruments designed to map the sky at nine frequencies with angular resolution from 33′ to 5′ which aimed at measuring temperature and polarized anisotropies. The Low Frequency Instrument (LFI) (Bersanelli et al. 2010; Mennella et al. 2011) covered frequencies centered at 30, 44, and 70 GHz. The detectors of the High Frequency Instrument (HFI) (Lamarre et al. 2010; Ade et al. 2011) covered frequencies centered at 100, 143,

217, 353, 545, and 857 GHz.

1.3 Single-Dish Radio Telescope

Sky surveys can be done by scanning the sky with a single-dish radio telescope. They range in size from few metre diameter up to few hundreds of metres. Examples include the 7.6-m C-BASS South telescope, the world's largest steerable dish Green Bank Telescope with diameter of 100 m, Arecibo with 305-m and 500-m Five hundred meter Aperture Spherical Telescope (FAST) non-steerable single dish. A typical single-dish radio telescope consists of a circular parabolic reflector antenna and a radio receiver such as a feed horn. The incoming radio waves in the frequency range of 10 MHz - 100 GHz from extraterrestrial sources are made to focus on the feed horn, where the signals are collected and are transferred to the radio receiver for amplification. The radiation pattern of an antenna describes the response of the circular antenna of diameter D when planar waves of wavelength λ is incident on it. Figure 1.1 shows an illustration of the antenna's radiation pattern. It defines the radiation pattern of an antenna as a function of its direction. The power pattern consists of a main lobe and multiple side lobes. The angular separation at which the intensity reduces to half of that at the centre of the main lobe is defined as the Full Width at Half Maximum, FWHM $\approx \lambda / D$, where λ is the wavelength and D is the diameter of the single dish telescope.

Unlike optical telescopes, radio-based telescopes are sensitive to the polarization states of the incoming radiation. At radio wavelengths, various process like synchrotron/cyclotron radiation, dust grains aligned with magnetic field, Thomson scattering etc. produces polarized radiation. The Stokes parameters I, Q, U, and V are used to give a complete description of the polarization states (Tinbergen 2005). Ng et al. (2005) and Robishaw & Heiles (2018) further describes the techniques used for the analysis of polarization properties

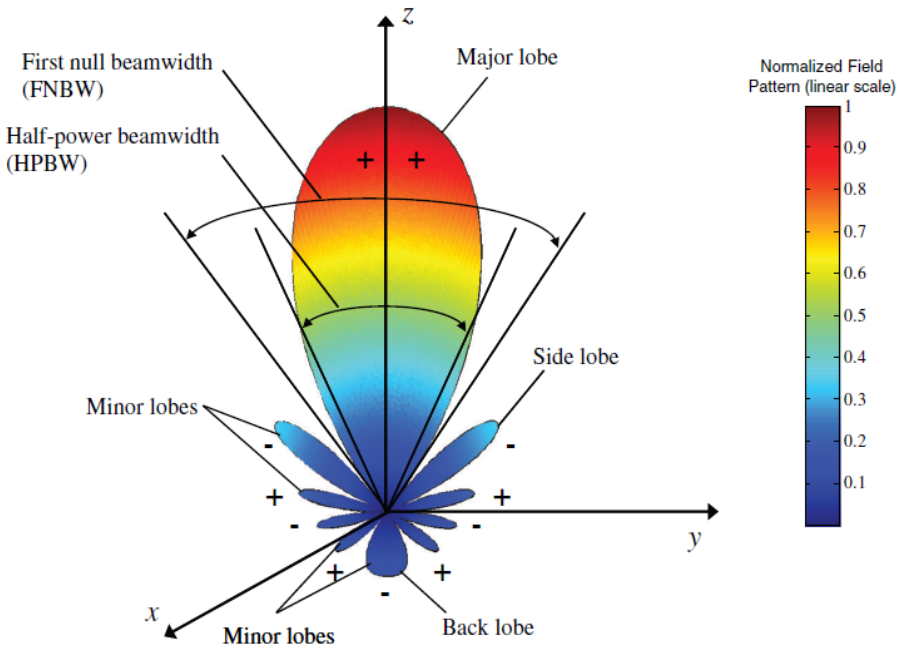


Figure 1.1: Power pattern of an antenna (Balanis 2016).

in radio astronomy. The telescope scans across the sky to measure the distribution of radio emission. The telescope sees the brightness distribution of the radio emission across the sky convolved with the point-spread function of the antenna. Thus, the measured flux density (S_ν) is given by (Burke & Graham-Smith 2002)

$$S_\nu = \iint_{source} B(\theta,\phi) P_n(\theta,\phi) d\Omega \quad \mathrm{W\,m^{-2}Hz^{-1}}, \tag{1.1}$$

where $B(\theta,\phi)$ is the brightness distribution of radio emission across the sky, $P_n(\theta,\phi)$ is the power pattern and $d\Omega$ is the solid angle.

1.4 Cosmic microwave background (CMB) observations

The discovery of the Cosmic Microwave Background (CMB) anisotropy was first reported in Penzias & Wilson (1965). The CMB is an almost isotropic black-body spectrum at an average temperature of 2.7 K. Small deviations or anisotropies from this average value encode a wealth of information about the Universe. The two important questions CMB measurements need to answer are about the earliest moments in the lifetime of the Universe and the distribution of matter in the Universe. Therefore the two main properties of the CMB measured are, its brightness temperature $T(\theta, \phi)$ (or intensity) which depends on the frequency, and its polarization states. The anisotropies in the CMB were first convincingly detected by the DMR instrument on the COBE satellite in 1992. The discovery of the CMB anisotropies was followed by various independent measurements by balloon-borne, rocket-borne and ground-based instruments over a broad range of angular scales. To avoid disturbance due to atmospheric absorption, the higher frequencies of the CMB must be observed from space rather than from ground-based experiments. The WMAP experiment confirmed the anisotropy results from COBE and also made measurements in temperature and polarization. CMB anisotropies are used to fix cosmological parameters which define the geometry, age, and composition of the Universe. The power spectrum of the temperature fluctuations (C_l), a function of angular wavenumber l calculated from the temperature anisotropies are used to estimate cosmological parameters. Thomson scattering of anisotropic CMB radiation generates linear polarization, thereby making CMB radiation polarized (Hu & White 1997). The polarization component has two modes, the gradient of a scalar field (the "E" mode) and the curl of a vector field (the "B" mode) (Zaldarriaga & Seljak 1997). The even parity E-mode polarization caused by electron scattering allow to the study the re-ionization history of

the Universe and about the formation of first stars and galaxies. Odd parity B-modes in the CMB polarization may have been induced by the gravitational waves during the inflation period. Thus, primordial B-modes enables a direct probe of inflation and conditions during the evolution of the Universe.

Temperature measurements (de Bernardis et al. 2000; Hanany et al. 2000) and polarization anisotropies measurements (Kovac et al. 2002; Readhead et al. 2004) of the CMB constrain the cosmological parameters. Thus, CMB studies are an important tool to study the origin of the Universe and cosmology. Typically, CMB data analysis is conducted in five stages. The CMB data must be devoid of unwanted signals before it is processed for map-making. Hence, before the map-making process as the first step the CMB data is flagged and calibrated. This will be discussed in detail in Section 1.6. The second step is the map-making process in which the raw time-ordered scan data is converted into an image of the sky. A detailed description about the map-making process is discussed in the upcoming Section 1.6. The third stage is the process of foreground removal. Foreground includes emission from the Galaxy and more distant galaxies, which are still in front of the CMB last scattering surface. Sky maps are produced at multiple frequencies and analysed to yield a map of the CMB, excluding foregrounds as much as possible. The fourth stage is the estimation of noise covariance matrices and angular power spectrum C_l from the maps obtained. Finally the cosmological parameters are estimated from the previously calculated values of C_l power spectrum. Having said that, some parts of foreground estimation can be done during parameter extraction, e.g. fitting the residual level of point source contamination in the power spectrum. With a clear understanding of the importance of the CMB, there are various projects aimed at studying the CMB. They include QUIJOTE (López-Caniego et al. 2014), QUIET (Araujo et al. 2012), PolarBEAR (Ade et al. 2014a) etc. Satellite-based CMB projects include the Cosmic Background Explorer (COBE) (Smoot et al. 1992), Wilkinson Microwave Anisotropy Probe (WMAP)

(Hinshaw et al. 2013) and *Planck* (Ade et al. 2014b). Among the three, *Planck* had better instrumentation with higher angular resolution and hence it produced maps with unprecedented precision and accuracy.

The *Planck* maps (Akrami et al. 2018) assumes a symmetric beam profile for the detector, where the data is assigned to the pixel where the beam centre falls without correcting for the beam shape. Thus, they reduce the complexity of the map-making method. The details of the symmetric beam profile of *Planck* is discussed in Ade et al. (2016). However, in practice an asymmetric beam profile needs to be considered to compensate for the telescope beam ellipticity. With an asymmetric beam, every pixel is smoothed with a different response, unlike in the case of symmetric beams where the resulting maps are smoothed by the same beam. The local ellipticity depends upon the orientation of the scans across the pixel and the beam orientation relative to the scan direction. With the future CMB projects aiming towards detecting feeble polarization signals it is necessary to account for the systematic errors due to asymmetric beams. Hence, deconvolution of the instrument beam needs to be carried out to obtain the true sky map. Deconvolution algorithm *ArtDeco* (Keihänen & Reinecke 2012) was used to deconvolve the asymmetric beam effects of LFI *Planck* data map and is reported in Keihänen et al. (2019). However, this algorithm which intends to increase the resolution inevitably amplifies the noise which is not ideal. Hence, in this dissertation we report a new map-making algorithm by interpolating with an elliptical kernel to circularise the beam. The algorithm with elliptical interpolation function is developed which could be used to approximately circularize an elliptical telescope beam. A detailed discussion about it is made in Chapter 5 of this Thesis.

1.5 C-BASS

The CMB anisotropies are used to estimate cosmological parameters which helps to study the primordial phases of the Universe. For example, from polarization measurements of the CMB, primordial B-modes could be detected to find evidence of primordial gravitational waves from the inflationary stage (Seljak & Zaldarriaga 1997; Frewin, Polnarev & Coles 1994; Polnarev 1985). However, since the initial results of CMB anisotropies from WMAP (Naselsky & Verrkhodanov 2008; Naselsky, Novikov & Chiang 2006; Chiang et al. 2007) foreground emission has been of great concern in analysing the results. These foregrounds include Galactic synchrotron emission at low microwave frequencies ($\lesssim$ 30 GHz), thermal dust emission at frequencies $\gtrsim$ 70 GHz and spinning dust or Anomalous Microwave Emission (AME) present in the frequency range 20-60 GHz (Ichiki 2014). Hence, for the accurate extraction of cosmological signals from observations, contamination by foreground emissions should be removed effectively from the observation maps. CMB experiments conducted in the frequency range of $70 - 100$ GHz where the foreground emissions are minimum also observe at multiple frequencies to mitigate foreground problems. For future B-mode CMB projects, it is essential to model and subtract foregrounds which dominate over different spectral and spatial ranges with high accuracy to detect the feeble B-mode primordial signals. For the frequency range below 30 GHz, the previously used foreground template maps for intensity include traditional Haslam maps at 0.408 GHz (Haslam et al. 1982), 1.420 GHz maps by Reich & Reich (1986); Testori et al. (2001); Bennett et al. (2013) map at 22.8 GHz and the *Planck* map (Ade et al. 2014b) at 28.4 GHz. Polarization template maps includes Carretti et al. (2013) at 2.3 GHz for southern hemisphere and the 5 GHz map with a FWHM of 9.5′ (Sun et al. 2007). However, there is a dearth of all-sky survey maps for Galactic synchrotron foreground removal in the frequency range of 1.420 GHz and

22.8 GHz (Irfan et al. 2015). The C-Band All-Sky Survey (C-BASS) is one such project, proposed in 2005 to map the sky at a low frequency of 5 GHz in both intensity and polarization. The observing frequency has been chosen as the one where Galactic synchrotron emission dominates and the measured polarization provides a good template for foreground removal at high-frequency observations (Jones et al. 2018).

Figure 1.2: Left: C-BASS North telescope at Owens Valley Radio Observatory. Right: C-BASS South telescope at Klerefontein in South Africa. (Taylor 2018)

C-BASS is an all-sky radio survey at a frequency of 5 GHz with an angular resolution of 45′ (Jones et al. 2018). The survey consists of two telescopes, one in the northern hemisphere and another in the southern hemisphere. The northern survey is conducted with 6.1-m telescope at the Owens Valley Radio Observatory (OVRO) in California and has completed its survey in 2015. The ongoing southern survey is carried out with a 7.6-m telescope at Klerefontein, the support base for MeerKAT and SKA South Africa. The observing telescopes of the C-BASS survey are shown in Figure 1.2. As the University of Manchester is a part of the C-BASS project, in this thesis we will be testing the algorithm developed for map-making on C-BASS pointing data.

1.6 Map making

All all-sky surveys at frequencies between 3 and 300 GHz has been done so far by scanning the sky using single-dish telescope and making a map of the sky. These single dish telescopes consists of detectors which used either coherent

amplifiers or bolometers. The data thus collected is a long time series called the time-ordered data (TOD). The correlation of this TOD with the locations on the sky is defined by the scan strategy of the survey. The C-BASS scan strategy is described in Jones et al. (2018).

Map-making, a three step process is the method of converting raw time-ordered scan data into an image of the sky. The first stage is the method of flagging and calibrating the time-ordered scan data, where flagging means excluding unwanted data, e.g. affected by RFI, solar interference etc. and calibration means correcting the gain scale and polarization leakage. The second step of the map-making process is destriping. The TOD collected consists of the wanted signal (e.g. CMB), foregrounds and instrumental noise. This noise is assumed to be Gaussian distributed stationary noise which has a $1/f$ component and a white noise component. If this $1/f$ spectrum noise is ignored then the signal drifts due to this noise and results in stripes in the final map. The amplitude of such stripes, being greater than the signal itself, compromises the scientific goals of the observation. This low frequency noise component $1/f$ could always be filtered out by using a high pass filter leaving the white noise in the TOD. However, such a procedure is not lossless. Thus, a destriping algorithm is introduced to provide an estimate of the low-frequency component of the instrument noise and return a TOD where the noise components are removed.

The third step is the map-making process itself. The TOD measured is approximated by (Tegmark 1997)

$$y = Ax + n, \tag{1.2}$$

where y is the data vector for a given scanning strategy at a given sampling rate, x refers to the pixelized sky map and n is random noise vector. A is a rectangular matrix called the pointing matrix of dimension (N_t, N_{pix}) describ-

ing the path of the detector across the sky, where N_t is the number of TOD samples and N_{pix} is the total number of pixels in the map. Assuming that the beam is symmetrical, x is the convolution of the sky with the beam pattern of the telescope and matrix A indicates the pixel at which the detector beam centre is pointing to and has one non-zero element per row. The linear map-making problem is given by (Tegmark 1997)

$$\tilde{x} = Wy, \tag{1.3}$$

where $\tilde{x}$ is the estimate of the map and W is the inversion matrix. Linear map-making methods needs to minimize the reconstruction error $\epsilon \equiv \tilde{x} - x$ to obtain the true map of the sky.

The optimal map-making solution for equation 1.3 includes a Generalized Least Squares (GLS) method for estimating $\tilde{x}$ as discussed in Natoli et al. (2001). Other linear methods includes the method used in COBE (Smoot et al. 1992) and the Weiner filter method (Wiener 1949). Dupac & Giard (2002) discusses a comparison between optimal map-making methods. Tegmark (1997) gives a complete comparison among various optimal map-making methods for CMB data. However, these optimal map-making methods are computationally expensive for very large TOD as expected in CMB community. The optimal methods need to know the full noise correlation matrix, i.e the covariance $< n_i n_j >$ for every pair of measurements (i, j). In general, this is a matrix with size N^2, where N is the number of measurements. For large TOD, where N could be of the order of 10^{10}, the noise matrix is too large to be stored on any computer. So in order to reduce such effects an approximation of noise is considered in the map-making process. Thus, in practice non-optimal map-making methods were developed called destriping method itself which are used for making maps for large TOD. Variations of such destriping algorithms have been implemented by various groups and are discussed in Burigana et al.

(1999); Delabrouille (1998); Keihänen et al. (2004); Maino et al. (1999); Sutton et al. (2009). Ashdown et al. (2007) compares the maps obtained from simulated *Planck* data using optimal map-making algorithms and destriping based map-maker algorithms. Destriping algorithms Springtide, Polar and MADAM gives results essentially as good as the optimal methods.

1.7 Pixelization scheme

In practice, to make maps, a pixelization scheme needs to be selected for optimal and non-optimal methods. Thus, a pixelized sky-map is an intermediate stage between the raw data acquired and the final stage where the physical parameters are estimated. Since the COBE maps with an angular resolution of 7°, the angular resolution has improved greatly, as with *Planck*. Hence more pixels are needed to construct such sky maps and careful attention must be given for efficient processing of such large data sets. Various tessellation schemes exist for the discretization and analysis of functions. For example, there is Hierarchical Triangular Mesh (HTM) in which the spherical surface is subdivided into triangles of nearly equal shape and size (Kunszt, Szalay & Thakar 2001; Szalay et al. 2007), rectangular grids as discussed in Driscoll & Healy (1994); Muciaccia, Natoli & Vittorio (1997), icosahedral triangulation of the sphere (Baumgardner & Frederickson 1985; Tegmark 1996) and igloo grids (Saff & Kuijlaars 1997; Crittenden & Turok 1998). The choice of a particular pixelization scheme depends upon the features of the problem to be solved. Tegmark (1996), discusses the features needed for an ideal pixelization scheme. Later, Gorski KM and his collaborates defined the ideal conditions for pixelization in Gorski et al. (2005). They are given by

1. Hierarchical structure of the database: The pixelated scheme should scale hierarchically with an increasing number of pixels, where the nearby data elements on the sphere should also be near in the hierarchical tree

structure. This increase should typically be a power of two, thereby making use of the binary nature of computational technology. Fig. 1.3 shows the hierarchical partition with quadrilateral structure.

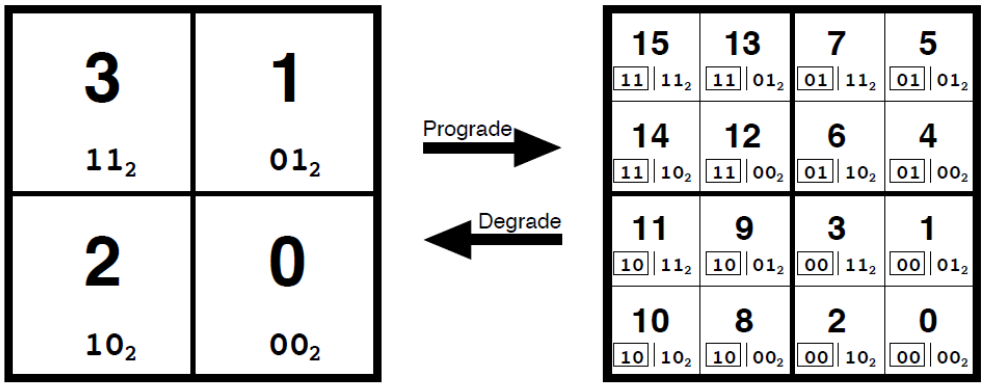


Figure 1.3: Quadrilateral tree pixel numbering. The pixelized coordinate patch on the left consists of four pixels. Two bits are used to label the pixels. Every pixel is split into four daughter pixels as shown on the right.(Gorski et al. 1999).

2. Equal areas of discrete elements of the partition: Pixels on the same latitude has the same shape but pixel shape can vary between latitudes. However, in all cases the pixel area remains the same. With equal area, white noise of the instrument is integrated into the white noise of the pixel (Gorski et al. 1999). Hence, the pixel size needs to be small enough compared to the resolution of the instrument. However, this has the drawback that the sky signal is integrated over the size of the instrument beam, but the noise is averaged only over the pixel which is much smaller than the beam, thereby reducing the signal-to-noise ratio.

3. Iso-latitude distribution of discrete area elements on the sphere: Pixels should be spaced equally on the latitude. This would affect the computational speed of spherical harmonics operations.

As specified in Gorski et al. (2005), all of the previously mentioned pixelization schemes fail to satisfy these three criteria, while all of the above conditions for

ideal pixelization are satisfied by Hierarchical Equal Area iso-Latitude Pixelation ($HEALPix$) pixelization scheme proposed by Gorski, Hivon & Wandelt (1998).

The base-level spherical pixelization is defined by two parameters (N_θ, N_ϕ). N_θ is the number of base-resolution pixel layers between the north and south poles and N_ϕ is the number of equatorial base-resolution pixels (Gorski et al. 2005). Then, the total number of pixels in the map is given by $N_{base\_pix} = N_\theta \times N_\phi$ (Gorski et al. 2005). $HEALPix$ is a versatile pixelization method which divides the sphere into quadrilaterals of equal area given by $\Omega_{base\_pix} = 4\pi/N_\theta N_\phi$ sr but of varying shape. By choosing $N_\theta = 3$ and $N_\phi = 4$, $HEALPix$ base-resolution consists of 12 curvilinear quadrilaterals. This choice was based on three basic requirements as given in Gorski et al. (2005): i) at the poles there shouldn't be more than four pixels in order to avoid acute angles, ii) simultaneous minimization of elongation of equatorial pixels, and iii) the 2^k multiplicity factor of pixels on equatorial zone rings should be retained for reasons pertaining to the fast harmonic transform.

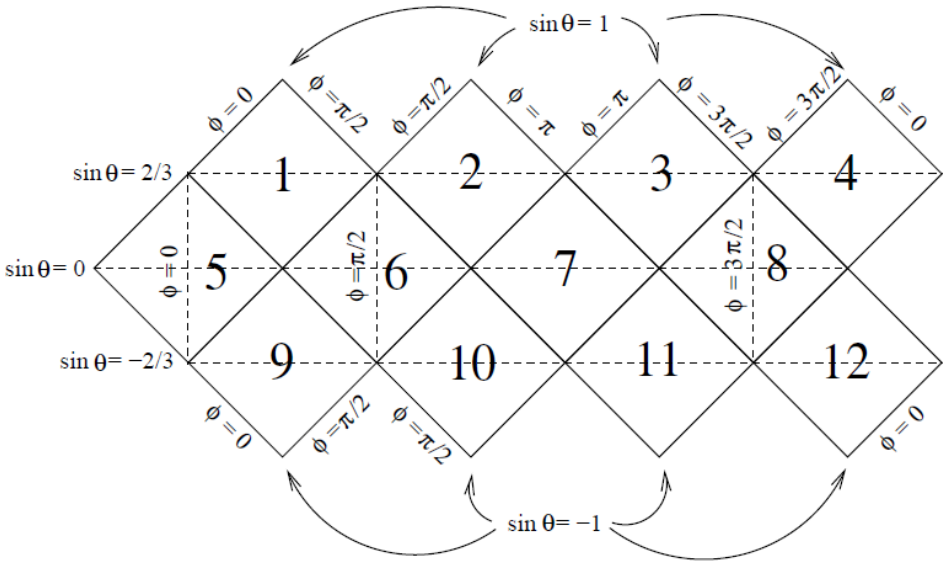


Figure 1.4: $HEALPix$ projection of 12 identical pixels covering a full sky of 4π sr (Roukema & Lew 2004).

As shown in Fig. 1.4, the base pixels numbered $1-4$ are the North facets, $5-8$ the equatorial and $9-12$ are the South facets. Each of the 12 quadrilateral pixels are recursively sub-divided into similar smaller pixels to create higher order mesh. Thus, the resolution of the mesh at a given order k is defined by N_{side}. It is defined by the number of sub-divisions along the side of a base-resolution pixel. Hence, a *HEALPix* map of $N_{side} = 2^k$ consists of a total number of $12 \times N_{side}^2$ pixels and each pixel has an area $\Omega_{pix} = \pi/3N_{side}^2$sr . *HEALPix* geometry supports two types of numbering scheme for the pixels called the *RING* and *NESTED* (Gorski et al. 1999). For a given N_{side}, each of the pixels are placed on rings of constant latitude called isolatitude rings. For a given N_{side}, there are $4N_{side} - 1$ such isolatitude rings. The locations of pixel centers on the sphere are defined by (θ, ϕ), where $\theta \in [-\pi/2, \pi/2]$ is the colatitude in radians measured from the north pole and $\phi \in [0, 2\pi]$ is the longitude in radians measured eastward (Gorski et al. 2005).

The *HEALPix* pixelization scheme corresponds to a map projection in which each quadrilateral pixel on the sky is projected to a square on the 2D plane, as discussed by Calabretta & Roukema (2007). In fact, in the usual orientation with latitude horizontal, the pixels are rotated by 45 degrees to give diamond shapes, since the poles correspond to the meeting point of four pixels. A feature of this projection is that the half-facets nearest the poles are "gores", that is, they are separated by regions in the 2D plane that do not correspond to regions on the sky, as shown in Fig. 1.4.

The *HEALPix* pixelization scheme as discussed has a mathematical software package. The software package is available in various platforms like *C*, *C++*, *Fortran90*, *IDL*, *Java* and *Python*. Each of these has extensive library of *HEALPix* specific tools for pixelization, hierarchical indexation, synthesis, analysis and visualization (HEALPix 2019). *HEALPy* is the *Python* based package bundled in *HEALPix*, *C++* and *cfitsio* (*C*). A detailed explanation regarding the installation of *HEALPy* is given in Cyrille Rosset (2019).

1.8 Science motivation for the work

The classic CMB map-making method discussed in Section 1.6 is only optimal in a restricted sense. In particular, it allows relatively easy error propagation while calculating spherical harmonics. But it is bad that it does not allow in any way for variation of emission within a pixel. In order to ameliorate the error propagation problem, the pixels are forced to be smaller than the telescope beam. Therefore, fewer data points are collected within each pixel. Hence, the maps has much higher noise per pixel than noise per beam. In aperture synthesis maps, the noise is correlated over the synthesized beam in exactly the same way as the sky signal and also the sky is effectively empty for isolated sources since interferometry filters out large-scale structure. Therefore, in contrast to the single-dish maps, the noise in the aperture synthesis maps are always the noise per beam. The optimal map-making method also has no easy way of addressing non-circular beams.

The practical implementation of map-making is done by choosing a pixelization scheme to decide the distribution of discretized map elements on the sky (Bond et al. 1999). Equation 1.2 should be an approximation since the sky is a continuous function and not made of finite discrete pixels. The simplest method of map-making is by averaging the data samples that falls to the same pixel on the sky, where the pixel size of the map is considered to be smaller than the beam size. To summarise the preceding two sections, equation 1.2, 1.3 are a compact way of expressing binning mathematically, provided that there is only one entry per row in the pointing matrix, which is the usual method. The binning approach is specific to the CMB community and is not used in traditional radio astronomy. The binning method ignores the continuous nature of the sky and hence the equation 1.2 is not exactly correct. On the contrary, interpolation-based map-making takes into consideration the continuous nature of the sky. Another advantage is that by using an interpolation function

varying smoothly with distance from the pointing position, more positional information is retained, whereas binning erases the information about the position of the sample on scales smaller than a pixel. This information can be recovered by fitting the interpolated map. A final advantage is that it allows the resolution to be circularized using a simple and linear algorithm if the intrinsic beam shapes are elliptical, as they were (approximately) for WMAP and the low-frequency *Planck* channels. There is one disadvantage compared to binning, which is that the random errors in the image are strongly correlated between pixels, which makes accurate error propagation more difficult especially when processing the entire image, as is done in spherical harmonic analysis for CMB studies.

Interpolation-based map-making was done in the Haslam and Reich surveys. However, they used a Cartesian grid which implies inefficient data storage. Therefore, this dissertation suggests an interpolation onto a *HEALPix* grid which would recognize the continuous feature of the sky which the binning approach described ignored. The *HEALPix* grid has the great advantage of equal-area pixels, and is becoming the de-facto standard for all-sky maps thanks to its adoption by WMAP and *Planck*. A minor advantage is that this allows fully-sampled maps to be stored using fewer pixels. A more significant advantage compared to the conventional approach of binning in theta,phi (-ARC or Plate Carrée projection) is that the poles can be properly treated as points rather than being spread out into a line, which often means there is no interpolation "over the top". With interpolation, it is meaningful to evaluate the sky at positions intermediate between the pixels. In 2D, this is exact with sinc interpolation provided that the sampling is adequate. On the *HEALPix* grid it is approximate, but it still should allow large pixels than in the binning approach. The maps thus created can be considered for general purpose. They can be used in analysing any astrophysical data such as large scale structure like Galactic emission. They can also be used to model the Galactic fore-

grounds at central frequency which is subtracted from CMB maps where the noise is uncorrelated.

Elliptical beams produce maps which are difficult to interpret when they contain extended emission, as nearly all all-sky radio maps do. This is because the apparent brightness of an elongated structure will vary depending on how its direction of elongation is oriented with respect to the elliptical beam. For instance a ring of emission, e.g. a supernova remnant with uniform brightness will appear to be modulated in brightness by $\sin(2\theta)$, where θ is the angle around the ring measured with respect to the beam orientation. In all-sky maps the orientation of an elliptical beam is fixed with respect to the scan direction which cannot be uniform over the sky, and indeed differs between different scans which cross a given point. This gives rise to a complicated variation of effective beam shape across the sky. As a result, spherical harmonic analysis cannot easily be corrected for resolution effects, a significant problem for both WMAP and *Planck*.

Non-linear deconvolution algorithm like ArtDeco has been tried but has the disadvantage that it amplifies noise whenever the resolution is increased, even if only in one direction. The usual tricks used to suppress such noise amplification when cleaning synthesis images fail for single-dish maps, partly because the sky is full of emission and so the "empty sky" prior used by Hogbom CLEAN is not appropriate, but mainly because the noise in single-dish maps is uncorrelated between samples, whereas the signal is correlated over the telescope beam. Therefore there are high spatial frequency components to the noise which are inevitably amplified by any attempt to improve the resolution, which involves amplifying such high-frequency components. Our approach, a linear convolution method is immune to this since it only reduces the resolution, specifically along the minor axis of the beam. This means that despite the negative impact on the error propagation, interpolation-circularized maps may be advantageous even for CMB analysis, especially as the technique

will be applied only to the low frequency maps whose purpose is to model the foreground contamination in the CMB-dominated channels.

In this dissertation, we develop an interpolation-based map-making algorithm on the *HEALPix* grid. Chapter 2 discusses the commonly used binning-based map-making algorithm and a simple but slow implementation of the interpolation map-making for the *HEALPix* that we call the standard interpolation method. Thereafter, in Chapter 3 we discuss in detail about the new interpolation-based map-making algorithm on the *HEALPix* grid also called grid interpolation. The grid interpolation algorithm is tested using C-BASS pointing data and *Planck* 857 GHz input map. All of the results obtained and reported in Chapter 4 are under the assumption that the beam has a circular profile. Additionally, we also extend the new map-maker algorithm by considering an elliptical beam profile. The details of the method and its results are discussed in Chapter 5. Finally, Chapter 6 presents the conclusion for the work and also discusses the future work.

Chapter 2

Motivation

This chapter discusses the commonly used binning-based map-making algorithm. We also describe a simple and exact but probably slow implementation of interpolation map-making for *HEALPix* that we call the standard interpolation method, which is described as the standard scheme for comparison with the approaches described in later chapters.

2.1 Binning-based map-maker algorithm

The single-dish telescope collects data by scanning the sky in a random pattern with no particular relation to the sky pixelization. The simplest way to treat such data is to bin each measurement into the nearest pixel. Let the radio telescope makes a measurement (I) by scanning the sky along a predefined scanning strategy. At each scan location, the corresponding intensity (I_i) is assigned to the nearest pixel bin which is defined by its boundary. The schematic representation of the binning procedure is shown in Fig. 2.1. Each pixel tile on the pixelated sphere has a location defined by coordinates longitude, latitude.

For implementation, we use *HEALPy* function $ang2pix$ to convert the angular coordinates of the data point to corresponding *HEALPix* pixel numbers. The intensity at each pixel is derived from the measurements falling into it.

Along with this, we also create a count map which counts the number of times each pixel bin is selected. Finally, for the entire data measurement length, the final intensity is the average of all the individual values derived by summing the values and then dividing by the hit count. The final map obtained is the averaged map. As discussed in section 1.8, this method of map-making is a sub-optimal method. However, this method is exceptionally fast with a single *HEALPy* function.

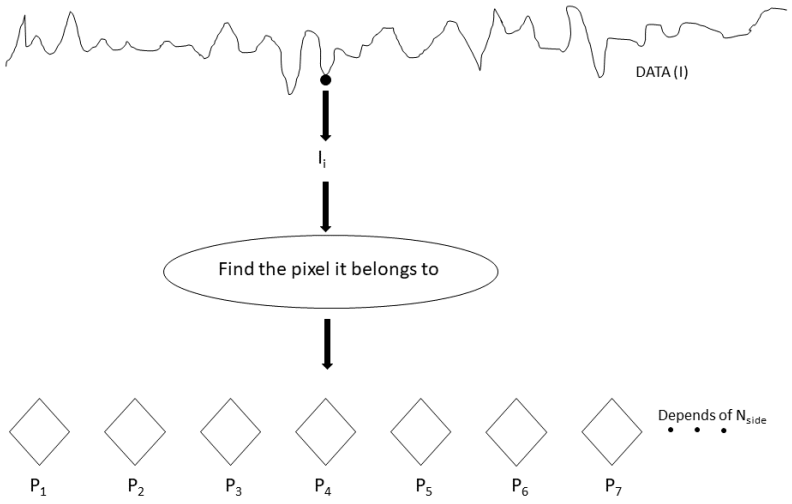


Figure 2.1: Schematic representation of binning algorithm.

2.2 Standard interpolation map-maker algorithm

The binning-based map-maker algorithm is a sub-optimal method for direct analysis of the image, including the visual inspection. With pixel size smaller than the beam, the noise per pixel is higher than noise per beam and thus in the binning method a choice needs to be made between resolution and noise-level. This demands a different approach e.g an interpolation-based map-maker algorithm which takes into consideration the continuous nature of the sky. The

algorithm is developed with the underlying idea that for each data point all the pixels within a circular radius around it are used to find the distance between them. Based on the proximity of the data point to each pixel, a weight is evaluated and a weighted data measurement is assigned for the pixels.

The schematic representation of interpolation-based method is as shown in Fig. 2.2. The square boxes shows a small portion of the pixelated sky with pixel centres represented by plus marks. In the context of interpolation, a pixel is defined by its nominal central position. The pixel is conceived as a delta-function sample of a smooth continuous function. The cross points indicate the data measurements made by the radio telescope along the scan direction. Unlike the binning-based map-maker algorithm, where intensity data values are assigned to the nearest pixel bins, in interpolation a weighted intensity is assigned to all the pixels within an interpolation patch, indicated by the red circle in Fig. 2.2.

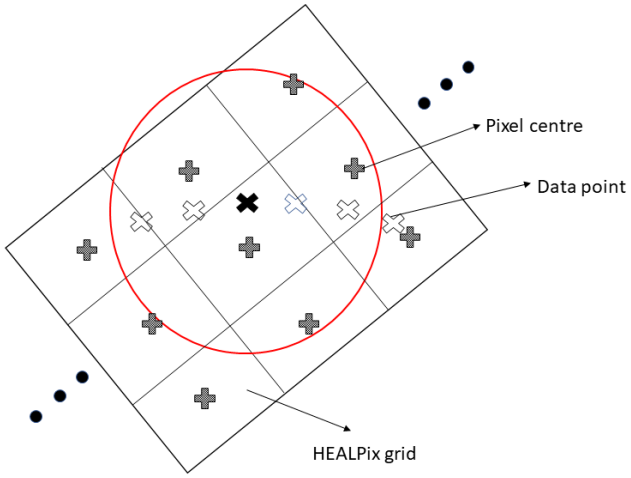


Figure 2.2: Schematic representation of the standard interpolation algorithm.

There are several weight functions available that could be considered. The most simple and straightforward weighing function is uniform weighting. In

uniform weighting, an equal and constant weight is assigned to the intensity measurement. If there are multiple measurements which has to be assigned for a given pixel, then the weight is down weighted by the count or number of such measurements. This weight assigned is inversely proportional to the sampling density function hence this method has highest resolution. However, this weight function increases the noise level. Clearly this isn't the right choice for the weighting function. In the context of interpolation, the choice of weighting function needs to satisfy certain specifications. The weighting function should fall off with the distance between data and pixel. It should have a scale related to the pixel separation by the Nyquist criterion and it should also be similar to or a bit narrower than the telescope beam to minimise the loss of resolution in the final image, but not too narrow to get the improved accuracy from averaging several data points. However, this is a trade-off and it depends upon the application intended for the map.

With this, sinc weighting function gives exactly the correct interpolation for a regularly sampled band-limited function in 2D. For a flat 2D grid the sinc function gives optimal interpolation. There is no such correct case for a grid of pixels on the sphere, although, the sinc function is approximately correct when the pixels are small and square, so that the pixel grid can be treated as locally flat. Since all-sky maps are not exactly band-limited and the sky is not a flat plane the rationale for sinc interpolation fails. A Gaussian distribution function which represents a probability density function for normal distribution is a good choice for the weight function. With a Gaussian distribution, the significant points closer to the centre are considered with increased weight compared to far away points. Its tail drops off rapidly as required and very little weight is assigned at the tail end yet it never becomes zero. With the non-zero tail end there is no sharp edge which might give ringing in the harmonic space, but this is actually a disadvantage as it would give infinite computational cost. So in practice, we choose a truncated Gaussian as the weighting function for

the interpolation based map-making. The standard deviation of the Gaussian distribution is calculated from the FWHM of the data under consideration.

For a given data point, as shown by the highlighted cross mark in Fig. 2.2, a circular interpolating window of radius (r) is considered. The size of this interpolation window decides the computational time of the algorithm and it is depended upon the FWHM of the data under consideration. However, ideally the pixel size is chosen according to the sampling theorem to be a fixed fraction of the beam width; e.g. a pixel width of about a third of a beam is a common choice. In that case, the number of pixels in the interpolation window would be approximately independent of the beam width, and the computational time would scale as the number of data points. In practice, for *HEALPix* the pixel width can only be adjusted in a factor of two steps, so this relation is approximate, but it will still hold as a rule of thumb. Thereby, for a given radius (r), the program needs to find all the pixels which fall within it to calculate the distance to the data point under consideration. This distance offset is then used to evaluate the corresponding Gaussian weight which needs to be multiplied to the data and added to the corresponding pixel.

For implementing using *HEALPy* functions, we proceed with a vectorization method, where the *ang*2*vec* function from *HEALPy* is used to convert angular coordinates (longitude, latitude) in radians to 3D position vector. Thus, this function would give the coordinate positions of each of the data points in vector notation as $\overrightarrow{v_n} = v_x\hat{i} + v_y\hat{j} + v_z\hat{k}$. For a given N_{side} and interpolation radius, *HEALPy* function *query_disc* returns pixels which lies within the circular disk defined for each position vector. All these pixel numbers are then converted to the corresponding position vectors. Dot product of these two vectors would give the cosine of the angle between them. Thus, the inverse of the

cosine gives the offset distance.

$$\overrightarrow{v_n} \cdot \overrightarrow{v_p} = |\overrightarrow{v_n}||\overrightarrow{v_p}| \cos\omega, \tag{2.1}$$

$$\omega = \cos^{-1}\left(\frac{\overrightarrow{v_n} \cdot \overrightarrow{v_p}}{|\overrightarrow{v_n}||\overrightarrow{v_p}|}\right), \tag{2.2}$$

where ω is the offset distance and $\overrightarrow{v_p}$ is the position vector. Depending upon the offset distance ω calculated using equation 2.1, a Gaussian weight is calculated and multiplied to the data point which thereby gives the weighted intensity map. Another map called the weight map which adds the Gaussian weight for the corresponding pixels are also created. The ratio of intensity weighted map to the weight map gives the final weighted map with Gaussian based interpolation.

2.3 Improvement of interpolation algorithm

A simple binning program and the standard interpolation program with *HEALPy* function *query_disc* were developed and tested. The test results and a detailed comparison between the two maps obtained is discussed in detail in Chapter 4 of this dissertation. The corresponding code for the standard interpolation map-making algorithm is described in Appendix C.

Before suggesting an interpolation onto the *HEALPix* grid map-maker algorithm, we decided to analyse the existing interpolation method for improvement. In terms of computational time, we identified two possible issues which could be analysed for improving the algorithm. The two problems identified are i) size of the interpolation patch, ii) the distance method. The size of the interpolation patch is defined from the beam size of the data set under consideration. The second possible improvement is by suggesting variations in the distance formula. In interpolation, there are two transcendental functions,

inverse cosine and exponential function. The inverse cosine is used for finding the distance between the data point and the pixel under consideration and the exponential function is present in the Gaussian distribution function. Having such transcendental functions in a looping structure makes it a computationally expensive operation. Hence, the aim is to replace these transcendental functions with a different method for finding the distance. With this intent we suggested two methods i) Polynomial fitting ii) Haver-sine Formula.

2.3.1 Polynomial fitting

In the interpolation-based method, an inverse cosine of the dot product between the position vectors of data point and pixels are used to find the distance between them. Based on this distance a Gaussian weight (w) is estimated.

$$x = \cos^{-1} y, \tag{2.3}$$

$$w = \exp(-x^2/2\sigma^2), \tag{2.4}$$

where y is the dot-product of the position vectors and w is the Gaussian weight. As given by equation 2.4, weight is estimated from the dot-product. Fig. 2.3 shows the plot of the dot-product against the Gaussian weight estimated for a given setup of data point position vectors. In the polynomial fitting method, we intend to fit a polynomial of optimal order n to the weight function (w). The graph of Fig 2.3 remains the same for any given dot-product which is usually in the range of (0.9998477, 0.9999995). Hence, polynomial coefficients for a given order n needs to be calculated only once and the weight is estimated without implementing the transcendental function. We consider a polynomial given by the equation,

$$\hat{w} = a_n + a_{n-1}y + a_{n-2}y^2 + a_{n-3}y^3 + ..., \tag{2.5}$$

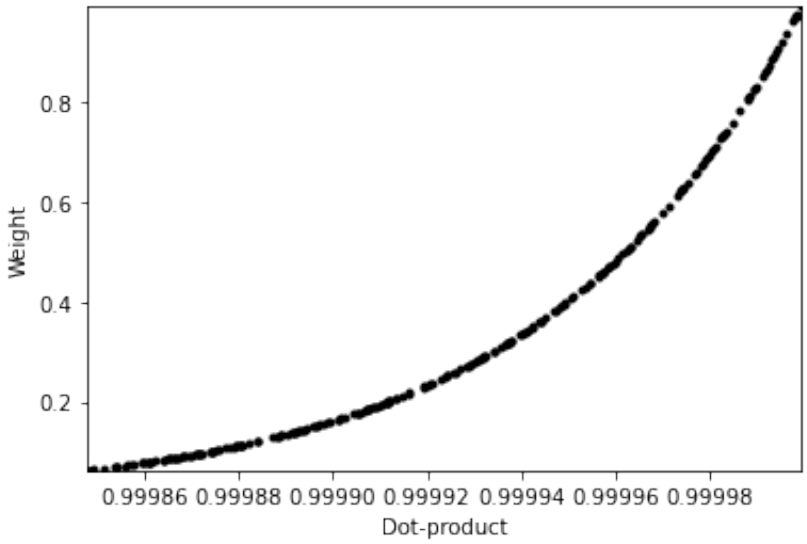
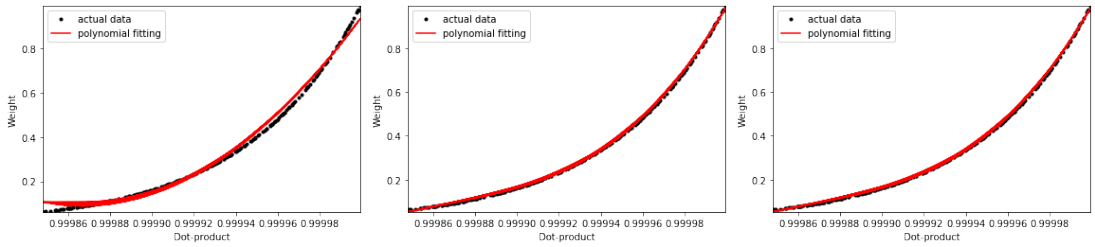


Figure 2.3: Plot of dot-product vs Gaussian weight for FWHM $= 1^{\circ}$.

where n is the order of the polynomial, $\hat{w}$ is the estimated weight and $a_n, a_{n-1}, a_{n-2}, ...$ are the polynomial co-efficients. It provides the best approximation between the independent variable y and dependent variable $\hat{w}$.



(a) Polynomial order 2 (b) Polynomial order 6 (c) Polynomial order 50

Figure 2.4: Polynomial curve fitting for Gaussian weight for varying order.

In this method the only unknown parameter is the optimal value for the order of the polynomial n. Fig. 2.4 shows the plot of dot-product vs Gaussian weight for n = 2, 6, 50. The quality of fitting is estimated by calculating the least square error between the polynomial fit ($\hat{w}$) and the original function (w). The least square error is given by,

$$e = ||w - \hat{w}||^2, \tag{2.6}$$

where e is the error calculated. It should be noticed that for lower order n the fitting is poor (high error value) while the fit increases (smaller error value) with order. The dot-product calculated for FWHM of 1° lies in the range of (0.9998477, 0.9999995). Therefore, a very high order is required to get a usable fit for which the computational time is extremely expensive. Hence, the polynomial method of distance is not ideal and is not considered for the map-making algorithm.

2.3.2 Haver-sine Formula

Another variation for distance calculation is by considering spherical geometry. Spherical geometry is used for measuring the distance between two points on a spherical surface like Earth. Haver-sine formula (Sinnott 1984), a trigonometric function is used to find the accurate distance between two points on the sphere given its location by latitude and longitude.

$$d = 2\sin^{-1}\left(\sqrt{\sin^2\left(\frac{\phi_2-\phi_1}{2}\right)+\cos\phi_1\cos\phi_2\sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right), \quad (2.7)$$

where (ϕ_1, ϕ_2) is the latitude and (λ_1, λ_2) is the longitude for two distinct positions on a sphere. However, this equation has multiple transcendental functions which makes it computationally expensive. Hence, there is no motivation in order to replace the distance function with Haver-sine formula.

2.4 Modified interpolation map-making algorithm

From the analysis done so far, there seems no possibility for improving the existing interpolation algorithm with any of the methods suggested. Hence, it is clear that we need a new interpolation based map-maker algorithm. With the ground work set so far, we suggest a variation of interpolation algorithm,

weighted interpolation onto the *HEALPix* grid. The details of the new map-maker algorithm would be discussed in detail in Chapter 3.

Chapter 3

Methodology

3.1 Programming language

For developing the grid interpolation map-making algorithm, we carefully considered several programming options such as *IDL*, *Matlab*, *Python*, *C/C++* etc. For high-speed applications, traditional C/C++ programming is usually the preferred choice. However, developing and maintaining a low-level C/C++ code is a highly time consuming and complex task. After careful consideration, we chose *Python* as the programming platform. *Python* open-source, powerful, dynamic language has become an increasingly popular language for scientific programming and astronomy is not an exception (Langtangen 2012). The *Python* has a plethora of powerful external libraries like *Numerical Python* (*NumPy*) which provides an extremely convenient, high-level interface to contiguous (and non-contiguous) arrays of data, graphical library *Matplotlib* which provides high-quality plotting capabilities, *Astropy* for handling Flexible image transport system (FITS) files etc. However, the major downside of *Python* is its speed. *Python* being dynamically typed rather than statically typed is the primary reason for its significantly lower speed when compared to languages like *C*, *Fortran*. However, this is not nearly as serious an obstacle.

Various frameworks has been developed to address this limitation. e.g *Ctypes*, *CFFI*, *SWIG* and *Cython*. Currently, the algorithm has been developed in pure *Python* and a serious consideration of these possibilities to improve the speed will be analysed at a later stage. For developing the algorithm we used *Python* of version 3.7.7 along with *HEALPy* 1.12.10.

3.2 Pre-requisites

Before we discuss in detail about the new map-making algorithm, it is important for us to understand the significance of certain concepts used for the development.

3.2.1 Gaussian Distribution

As discussed in Section 2.2 of Chapter 2, a Gaussian distribution is used as the weight function for the interpolation-based map-maker algorithm. For a defined circular radius centred at the data point under consideration, the pixels that fall within the circular patch are identified. Then, a Gaussian-weighted response of the data measurement is calculated for the pixels depending upon its proximity to the data point. The probability density function (pdf) for the Gaussian distribution (Miller & Childers 2012) is given by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{3.1}$$

where σ is the standard deviation and μ is the mean. We took FWHM to be 1° for the tests reported in this thesis. Considering equation 3.1, FWHM is obtained by finding the half-maximum points x_0 by solving the condition

$$f(x_0) = f(x_{max})/2. \tag{3.2}$$

The standard deviation for the Gaussian distribution occurs when $x_{max} = \mu$,

$$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x_0-\mu)^2}{2\sigma^2}} = \frac{1}{2}\frac{1}{\sigma\sqrt{2\pi}}, \tag{3.3}$$

$$\text{FWHM} = 2\sqrt{2\ln 2}\sigma. \tag{3.4}$$

3.2.2 Co-ordinate System

The basic requirement of astronomy is to be able to determine the location of celestial objects relative to each other in the sky. This is given by a coordinate system. Over the years, astronomers have developed various coordinate systems. They include the equatorial coordinate system, the horizon coordinate system, the ecliptic coordinate system, the galactic coordinate system etc. Each of this coordinate system considers an imaginary sphere called the celestial sphere with the observer at its centre (Green & Green 1985).

The coordinates of the most commonly used equatorial coordinate system is given by declination (DEC) and right ascension (RA) denoted by δ and α respectively. Declination measures the angle of an object from the celestial equator such that the North celestial pole is $+90°$. Right-ascension measures the angle of an object eastward from the Vernal Equinox (Bradt 2004).

The Galactic coordinate system is often used by Galactic astronomers and cosmologists, as the foregrounds that obscure the view of the distant Universe are along the Galactic plane. The coordinates are galactic longitude and galactic latitude. Galactic latitude denoted by (b) is similar to declination which measures the north and south poles from the galactic equator such that it is $+90°$ at the north galactic pole (NGP) and $-90°$ at the south galactic pole (SGP). Galactic longitude (l) is similar to right ascension and is measured along the galactic equator in the counter-clockwise direction (Karttunen et al. 2016).

The test data set from *Planck* is measured in Galactic coordinates. However, in our setup they have been used as RA and DEC. We use the colatitude

(the complementary angle of a given latitude) instead of the latitude, where colatitude $(\theta) = \pi/2 - DEC$.

3.2.3 Small angle distance approximation

In the new grid interpolation map-making algorithm, a Gaussian weight is applied to the intensity measurements based on the distance of the pixels from the data sample under consideration. For finding the distance between two points on a sphere we use spherical trigonometry. The two points on the sphere are a part of the spherical triangle, which is an intersection of great circles on a sphere. A great-circle arc on the sphere is equivalent to a straight line on planar geometry. Spherical trigonometry uses trigonometric functions to find the sides and angles of the spherical triangle. Fig. 3.1 shows a spherical triangle ABC, where the vertex A indicates the north pole and the vertices B, C are the two points under consideration given by the coordinates declination and right-ascension (δ_1, α_1), (δ_2, α_2). In our setup, we have considered points B, C as the data point and the pixel centre respectively. Assuming that they are located in close proximity, the coordinates of the points are given by $(\tilde{\delta} + \frac{\Delta\delta}{2}, \alpha_1)$ and $(\tilde{\delta} - \frac{\Delta\delta}{2}, \alpha_2)$, where $\tilde{\delta}$ is an approximation of the declination for the points under consideration and Δ is a small incremental difference. The sides of the spherical triangle u, v, w are the arcs of the great circle. In terms of spherical trigonometry, the sides are nothing but the angle subtended given by ρ, $(\tilde{\delta} - \Delta\delta/2)$, and $(\tilde{\delta} + \Delta\delta/2)$ respectively, where ρ is the distance from the pointing data sample to a particular pixel centre.

The spherical law of cosine rule (Smart, Smart & Green 1977) as given in equation 3.5 is used to find the distance ρ between the two points under consideration.

$$\cos u \quad = \quad \cos v \cos w + \sin v \sin w \cos A. \tag{3.5}$$

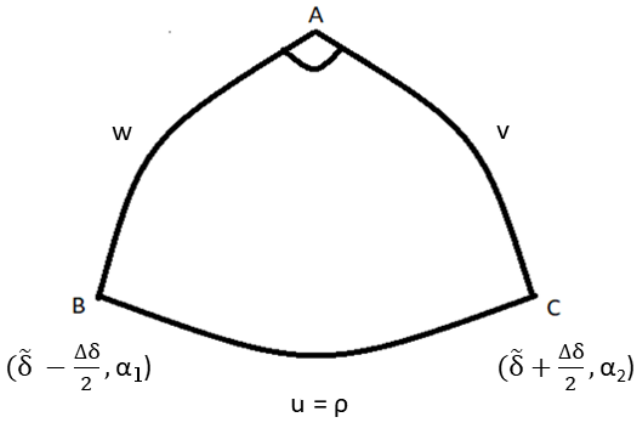


Figure 3.1: Spherical trigonometry geometry for the two points under consideration on the sky.

Considering that the points are very close to each other, we use second-order small angle approximation for a small angle ϵ.

$$\sin\epsilon \approx \epsilon, \tag{3.6}$$

$$\cos\epsilon \approx 1 - \frac{\epsilon^2}{2}. \tag{3.7}$$

In our frame of reference, we use colatitude (θ) instead of declination along with the above-mentioned approximations in equation 3.5, the distance is given by

$$\begin{aligned} \cos\rho &= \cos\left(\tilde{\theta} + \frac{\Delta\theta}{2}\right)\cos\left(\tilde{\theta} - \frac{\Delta\theta}{2}\right) + \\ &\quad \sin\left(\tilde{\theta} + \frac{\Delta\theta}{2}\right)\sin\left(\tilde{\theta} - \frac{\Delta\theta}{2}\right)\cos\Delta\alpha, \end{aligned} \tag{3.8}$$

$$\rho \approx \sqrt{\Delta\theta^2 + \Delta\alpha^2\sin^2\tilde{\theta}}, \tag{3.9}$$

where $\Delta\theta = \theta_{pixel} - \theta_{data}$ is the difference in the colatitude values between the pixel and data point under consideration and $\Delta\alpha = \alpha_{pixel} - \alpha_{data}$ is the difference in the right-ascension values between the pixel and data point under consideration. A detailed derivation of the distance formula is given in Appendix

A.

For nearby points, ρ, $\Delta\theta$ and $\Delta\alpha$ are all small except very close to the pole for $\Delta\alpha$. The value of $\tilde{\theta}$ in equation 3.9 is given by,

$$\tilde{\theta} = \frac{\theta_{data} + \theta_{pixel}}{2}. \tag{3.10}$$

For computational ease, we approximate θ_{data} to the nearest value of θ_{pixel}. This is done by making use of iso-latitude rings. As described in Section 1.7, for a given N_{side} there are $(4N_{side} - 1)$ such rings and every pixel centre on *HEALPix* geometry is centred on an iso-latitude ring. Thus, the pixel centre under consideration is placed on one such iso-latitude ring. So instead of using the corresponding colatitude for every data point in equation 3.10, we approximate the colatitude of the data to the latitude of the nearest iso-latitude ring. For every data point, the nearest iso-latitude ring is found by using a simple *minimum* function. Let j be the index of the corresponding iso-latitude ring. The θ_{pixel} is a value very close to the corresponding latitude of iso-latitude ring, so the approximated $\tilde{\theta}$ is given by

$$\tilde{\theta} \approx \frac{\theta_j + \theta_{pixel}}{2}, \tag{3.11}$$

$$\approx \theta_j, \tag{3.12}$$

where θ_j is the latitude value of the corresponding iso-latitude ring. With this approximation, for every input data file the values for $\sin^2\tilde{\theta}$ in equation 3.9 is calculated and is tabulated prior to the map-making process. Given that it is easy to work out the θ value of the data (= value for that iso-latitude ring), we realised towards the end of this work that there is no reason to use this approximation.

3.2.4 Interpolation window

In grid interpolation map-making algorithm, for every data sample under consideration the pixels within a circular radius centered at the data are needed to implement the method. The size of the interpolation patch can affect the computational time of the algorithm. Considering a window radius to be equal to the $FWHM$, the Gaussian distribution is at $1/16$ its maximum amplitude, so it is not too bad an approximation to neglect it even further away. In practice, we use a square interpolation window, just large enough to contain a circle of radius (r) equal to $FWHM$. For a given N_{side}, the pixel-size (κ) and radius (r), the width of the window in pixels, called the window length (τ), is calculated by

$$r = 1^{\circ}, \tag{3.13}$$

$$\kappa = \sqrt{\frac{4\pi}{12 \times N_{side}^2}}, \tag{3.14}$$

$$\tau = \text{ceil}(r/\kappa). \tag{3.15}$$

3.3 Interpolation onto the *HEALPix* grid map-maker algorithm

The basic *HEALPix* projection as illustrated in Chapter 1, Fig. 1.4, is rotated by 45° as shown in Fig. 3.2. The base-resolution diamond shaped pixels aligned with diamond symmetry are now projected as squares by the 45° rotation (Calabretta & Roukema 2007). On the *HEALPix* projection, data can be displayed without any artefacts and re-gridding. The butterfly projection discussed in Calabretta & Lowe (2013) is a polar variant of this $HEALPix$ projection. Thus, the new grid interpolation map created is a 2D grid map of size $5N_{side} \times 5N_{side}$, where N_{side} is the resolution of the *HEALPix* pixelization.

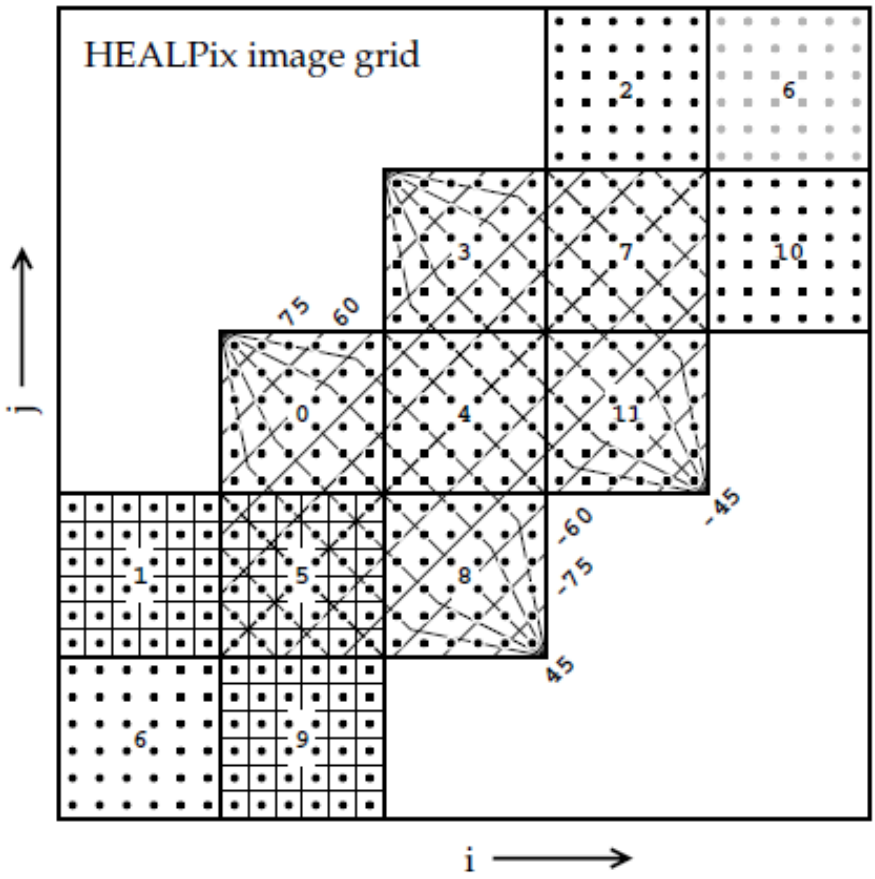


Figure 3.2: The *HEALPix* projection geometry obtained from 45° rotation of basic *HEALPix* geometry (Calabretta & Roukema 2007). The bottom left corner of the grid indexed at $(0, 0)$ is at $\delta = 0°$ and $\alpha = 225°$.

For developing the grid-based map, we need to focus on polar and equatorial regions separately. The separation between these two regions is defined by the transition latitude $\lambda_\times$, where both polar and equatorial pixels meet. For defined values of $N_\theta = 3$ and $N_\phi = 4$ as discussed in Section 1.7 of Chapter 1, it is defined that the north polar region covers only $(1/6)$ of the total area on the sphere. By equating the spherical cap area to the polar fraction of the total

area gives the boundary condition (Calabretta & Roukema 2007):

$$2\pi(1-\sin\lambda_\times) = 4\pi/6, \tag{3.16}$$

$$\lambda_\times = \sin^{-1}(2/3) \approx 41.8103^\circ. \tag{3.17}$$

1. Equatorial region: The equatorial region is defined by the condition, $|\delta| \leq \lambda_\times$. Projecting the spherical coordinates onto the Cartesian coordinate system as in Calabretta & Roukema (2007) gives

$$x = -(\alpha - 1.25\pi) \times k_1, \tag{3.18}$$

$$y = \frac{3\pi}{8}\sin\delta \times k_1, \tag{3.19}$$

   where k_1 is a scale factor discussed below.

2. Polar region: The polar region is defined by $\delta > \lambda_\times$ on the unit sphere. The projection equation is given by Calabretta & Roukema (2007),

$$x = -(\alpha_c + (\alpha - \alpha_c)\sigma^* - 1.25\pi) \times k_1, \tag{3.20}$$

$$y = \pm\frac{\pi}{4}(2-\sigma^*) \times k_1, \tag{3.21}$$

   where $-$ve sign in equation 3.21 is for south polar region and $+$ve sign is for north polar region. k_1 is a scale factor. The value of σ^*, α_c is given by

$$\sigma^* = \sqrt{3(1-|\sin\delta|)}, \tag{3.22}$$

$$\alpha_c = -\pi + \left(2\left\lfloor\frac{4\times(\alpha+\pi)}{2\pi}\right\rfloor + 1\right)\frac{\pi}{4}, \tag{3.23}$$

   where $\lfloor u \rfloor$ is the floor function and $\alpha_c = \pi/4, 3\pi/4, 5\pi/4, 7\pi/4, 9\pi/4$ for α in the range 0 to $\pi/2$ and $\pi/2$ to π etc.

In the projection equations for x in equations 3.18, 3.20 there is an offset of 1.25π. Inspection of Fig. 3.2 shows that the bottom left corner of the grid, that is $(i, j) = (0, 0)$ is at $\delta = 0^\circ$ and $\alpha = 225^\circ = 5\pi/4$ radians. So in projecting right-ascension (α) to obtain x, we need to subtract an offset = $5\pi/4$. The reason a minus sign is used in this conversion is just that astronomers measure right-ascension (and other longitudes) increasing from right to left across the sky as opposed to the usual convention with coordinates in mathematics. The diagonal of the *HEALPix* grid has a length of $5\sqrt{2}N_{side}$ and it corresponds to a total angle of 2.5π radians. There are five facets in the prime diagonal, but only four facets are needed to go around the whole sky. Therefore, to get to units of pixels from units of radians, we should multiply by a constant $k_1 = 5\sqrt{2}N_{side}/(2.5\pi) = 2\sqrt{2}N_{side}/\pi$.

To obtain the $HEALPix$ grid projection, the projection in Cartesian coordinate system, with RA & DEC horizontal and vertical, obtained from equations 3.18, 3.19 for equatorial regions and equations 3.20, 3.21 for polar regions is rotated by 45° in the anticlockwise direction using the rotation matrix as given below.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\omega & -\sin\omega \\ \sin\omega & \cos\omega \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix}, \tag{3.24}$$

where ω is the rotation angle. The x', y' shows the rotated projection of Cartesian coordinate locations of the data points along the scan track in the new $HEALPix$ projection grid. Consider the schematic representation of the setup for the algorithm in Figure 3.3. On the *HEALPix* projection, the blue dotted lines shows the scanning trajectory for a given C-BASS pointing input file. For creating the map, each of these data points which satisfies the condition for the polar and equatorial region needs to be considered with an interpolation window round it. Since the projection equations are different and the regions are separated by the condition given by equation 3.17, the data points that belong

to each region are treated separately with their corresponding conditions and equations.

Let the black point in Figure 3.3 be the data sample under consideration and the red dots within the square interpolation window shows the individual pixels. Let (i_c, j_c) be the coordinates of the central pixel in the square interpolation window. Based on the window length τ, the square interpolation window has

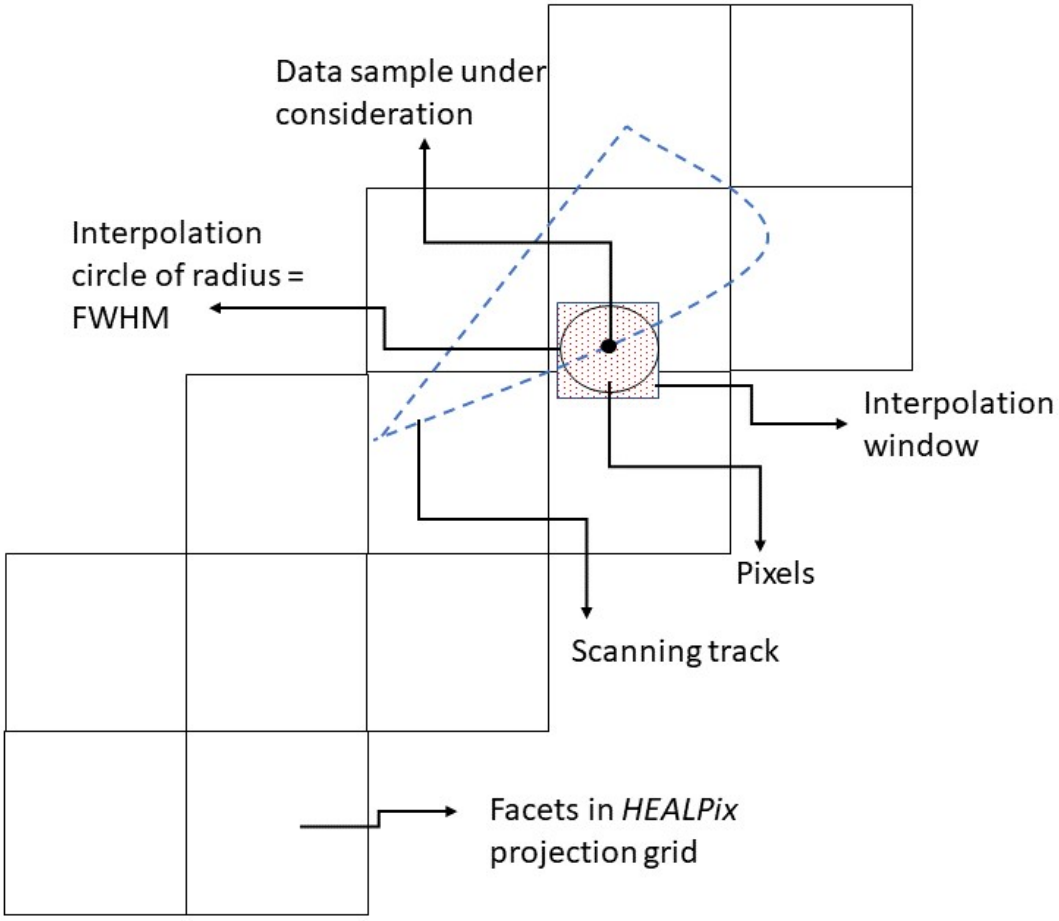


Figure 3.3: Schematic representation of the setup in interpolation on the *HEALPix* grid algorithm

a size given by $(2\tau + 1) \times (2\tau + 1)$. For the estimation of the weight we precompute the pixel offsets for all the pixels in the interpolation window relative to the central pixel.

To calculate the distance (ρ) for each pixel in the interpolation window re-

quires the corresponding sky positions. Thus, the 2D window with pixel offset values is rotated 45° in a clockwise direction to revert from the new *HEALPix* projection geometry. Let (x_{off}, y_{off}) be the rotated 2D arrays with offset values of the corresponding pixels in the interpolation window. This offset matrix in Cartesian coordinates with RA and DEC as horizontal and vertical is just the offset of the pixels relative to the central pixel in the window. To obtain the differential coordinates, the corresponding data coordinates given by (x,y) in equation 3.24, must be subtracted from the 2D array of pixel coordinates given by (x_{off}, y_{off}). Let the differential coordinate arrays be represented as $(\Delta x, \Delta y)$. Finally, to obtain the sky positions, the 2D arrays $(\Delta x, \Delta y)$ are converted into spherical to obtain the corresponding spherical coordinates of the interpolation patch under consideration. Let the inverse mapping of spherical coordinates be represented by $(\Delta\alpha, \Delta\theta)$ and is given by differentiating the projection equations. For the equatorial region, $(\Delta\alpha, \Delta\theta)$ is obtained by differentiating equation 3.18, 3.19 respectively (Calabretta & Roukema 2007).

$$\Delta\alpha = \Delta x/k_1, \tag{3.25}$$

$$\Delta\theta = \Delta y/\left(\frac{3\pi}{8} \times \sin\theta \times k_1\right). \tag{3.26}$$

For polar region, to obtain $(\Delta\alpha, \Delta\theta)$, we need to differentiate Equation 3.20, 3.21 respectively (Calabretta & Roukema 2007).

$$\Delta\alpha = \Delta x/\sigma^*, \tag{3.27}$$

$$\Delta\theta = \left(\Delta y \times \sigma^*\right)\Big/\left(\frac{3\pi}{8} \times \sin\theta \times k_1\right), \tag{3.28}$$

where θ in all these equations is the colatitude. Using these equations for equatorial and polar regions, the small angle approximated distance is estimated using the equation 3.9. This ρ gives the distance calculated with respect to each data point under consideration from the pixels around it in a radius r.

For a given ρ, the Gaussian distributed weight (w) is calculated using equation 3.1. The values of the weighted intensity (I) and the Gaussian weight are summed into the pixels of the $HEALPix$ grid . The ratio of these two gives the weighted map in the $HEALPix$ grid format.

$$\text{Weighted Map} \quad = \quad \frac{\sum_i I_i w_i}{\sum_i w_i}. \tag{3.29}$$

3.4 Post processing to obtain the final map

The weighted intensity map and the weight map obtained from the procedure discussed in Section 3.3 does not give the complete and final $HEALPix$ projection map. Before finding the final weighted map there are a few issues that need to be fixed for the map. The evaluation of facet number 6 as shown in Fig. 3.2 is incomplete. Facet number 6 which appears on either ends of the grid projection, lower-left and upper-right corner, is evaluated to only half of the facet. Hence, we need to fill in the other half of the facet 6. Another issue with the grid map is the boundary problem for the interpolation window for all the pixels along the polar edge. For every pixel along the polar edge, the interpolation window goes off-grid and it would return negative indices. The solution to this problem is to introduce guard rings around the whole grid. Guard ring, is a zero padded array added along the four edges of the grid map. The third issue is that at the polar vertical and horizontal edges of the gores, the interpolation contribution is not fully included within the interpolation window centered at that data point because the edge of the adjacent gore would also contribute to this. Hence, to obtain a complete map we need to address this polar edge problem by stitching together the corresponding neighbouring edges. We dedicate this section to discuss in detail the procedure undertaken to fix these problems with the weighted intensity and weight maps and to finally obtain the complete weighted map.

3.4.1 Guard rings

To solve the boundary issue related to the grid, we add a guard ring around the whole grid. The size of this border is decided by the interpolation window length τ calculated by equation 3.15. Therefore, by adding the guard ring, the weighted intensity and weight array runs from $-\tau$ to $5N_{side} + \tau - 1$ in both dimensions. Let the indices of this bigger array be (p, q), so that $p = i + \tau$, $q = j + \tau$. Basically, this means that every time we use the weighted intensity map and weight map, we need to add a $+\tau$ to both indices. With this we no longer have to check if the interpolation window goes off the grid because it is never centered at a pixel nearer to the edge than τ pixels.

3.4.2 Facet 6

The half split facet number 6 which exists at the lower-left and upper-right projection grid needs to be filled for the full facet size. Facet number 6 which appears on either end shows the continuation of the projected map on the sphere. Thus, for refilling the empty half of the facet, we find the average of the half split facet 6 on either ends of the grid. This resultant averaged map is now copied to the entire region of the facet 6 on either ends. This procedure is repeated separately for both the weighted intensity and weight array.

3.4.3 Internal edges

There are four facets along the northern polar region and they have been numbered as facet $0 - 3$. Each of this facet has two internal edges, a side-edge and top-edge. According to the geometry of the projection, all of these edges are stitched together to form the north polar cap. With this idea, consider facet 0 and 3 in Fig. 3.2; the top-edge of facet 0 needs to be stitched to the side-edge of facet 3. Thus, while considering the interpolation along these edges, there is contribution from the corresponding neighbouring edge along with the points

within the interpolation window length. For a given window length τ, the interpolation window size is $(2\tau + 1) \times (2\tau + 1)$. With the guard ring considered around the whole grid, for facet 0 the top-edge patch indices are given by $(N_{side} - \tau : 2N_{side}, 3N_{side} - \tau : 3N_{side} + \tau + 1)$. For facet 3 the side-edge indices are given by $(2N_{side} - \tau : 2N_{side} + \tau + 1, 3N_{side} : 4N_{side} + \tau)$. After completing the interpolation onto the grid, for the intensity-weighted array and the weight array the top-edge patch and side-edge strip are extracted. The average of these two edge strips is then calculated by rotating the side edge strip by 90° counter-clockwise. This averaged strip is now copied to the top-edge strip of facet 0 for the intensity-weighted and weight maps and the 90° clockwise rotated average strip is assigned to the side-edge strip of facet 3 likewise. This procedure needs to be repeated for each of the four edges in each of the two polar regions. However, only the edges between facets 0 & 3 and 8 & 11 are entirely on the grid, so before dealing with the rest we have to add the guard ring. For computational simplicity and efficiency, we have looped over all the edges instead of coding each edge individually.

3.4.4 North and south poles

With all the interpolation done so far, we are still not getting the poles right. This is because, at the poles, the interpolation zones of all four polar facets overlap. Suppose we start with the edge between facets 1 & 0. This takes data from near the pole in facet 1 and interpolates it near the pole in facet 0. Dealing with the gap between the facets 0 and 3 carries this over to facet 3 and then in the next stage it is carried to facet 2. Finally doing the edge between facet 2 and 1 will interpolate back into facet 1 which is unnecessary as this interpolation has already been done. This is one of the problems yet to be solved. We would be undertaking this in future work beyond this dissertation.

3.4.5 Cleaning up the edges

The guard ring is added round the whole grid just to avoid the usage of negative indices for the boundary pixels. Hence, once all the interpolation is complete, the guard ring needs to be removed. This is repeated for both the intensity weighted map and weight map. We also identify the bad pixels in the weight map where the value is zero. For such bad pixels, the weight map is assigned to one and the corresponding intensity weighted map is assigned to a constant -1.6375×10^{30} as defined in $HEALPy$ module. Finally, the ratio of the weighted intensity to the weight map is calculated using equation 3.29 to obtain the weighted map. However, we still have some interpolation data outside the edge of the weighted map, that is in pixels of the grid that are outside the 12 facets that covers the sky. In a sense this is unimportant as these pixels will be left out when we convert this grid projection to $HEALPix$, but for display purpose it would be tidy to set all these to blank values. In practice we create a $N_{side} \times N_{side}$ array filled with NaN values and assign it to the weighted map with indices $(0 : N_{side}, 2N_{side} : 3N_{side})$. This is repeated for the other five empty blocks adjacent to the polar facets. Thus, finally we obtain the complete interpolation onto the *HEALPix* grid map.

Chapter 4

Results and Discussions

4.1 Details of the test data

Interpolation onto the *HEALPix* grid was tested by simulating observations of the *Planck* 857 GHz all-sky map using C-BASS pointing data. For testing purpose, we need to know the true sky. Hence, we observed the *Planck* map using the C-BASS pointings, rather than using the actual C-BASS data. *Planck* intensity data is obtained from High Frequency Instrument (HFI) measurements at a frequency of 857 GHz with an angular resolution of 4.22′ (Aghanim et al. 2018). *Planck* uses the *HEALPix* pixelization scheme with a resolution $N_{side} = 2048$. For testing we have re-binned it to $N_{side} = 512$. Fig. 4.1 shows the full-sky Mollweide projection of the *Planck* data. The unit of the colour scale for all the intensity images in this thesis is Kelvin. The coordinates of the *Planck* map were interpreted as (RA, DEC) even though they were Galactic longitude and latitude. However, this makes no difference for the test purpose.

The C-BASS sky survey makes use of two telescopes for scanning the entire sky. The Northern C-BASS scans 360° in azimuth at a constant elevation of 37.2° and 47.2° (Jones et al. 2018) at a speed of $\approx 4°$ azimuth per second. We used data from the C-BASS North survey conducted by the Gregorian antenna.

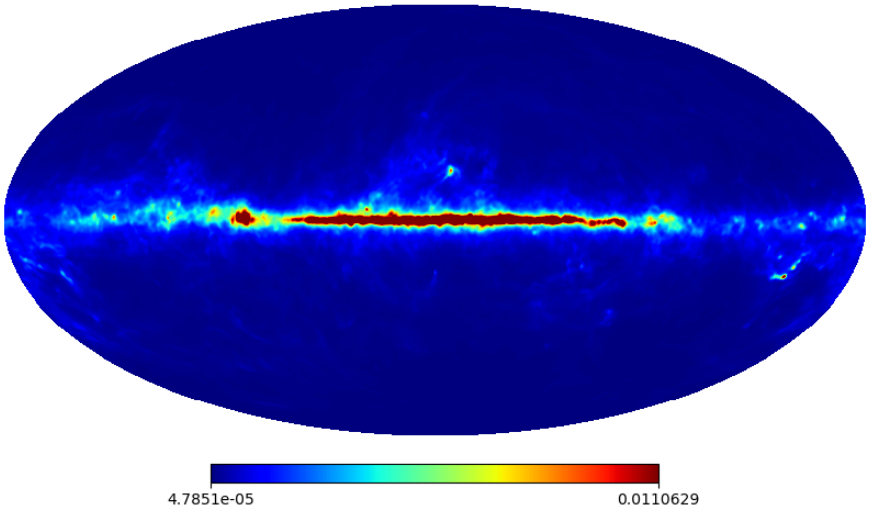


Figure 4.1: Full-sky Mollweide projection of *Planck* 857 GHz intensity map.

The C-BASS data used belongs to the observations conducted in September 2013 of the 3-year observing programme. This corresponds to 384 FITS files.

For each C-BASS FITS file, the *HEALPy* sub-routine *ang*2*pix* function was used to find out which sky pixel does each C-BASS pointing corresponds to. The intensity measurements of the corresponding pixels from the *Planck* 1D array data was used as the intensity values for the given C-BASS pointing FITS file. A hit map was created by counting the number of pixels covered in each scan. The density of observed points reveals the scan strategy for C-BASS and is shown in Fig. 4.2. As the telescope scans the north pole in each scan of the sky, the hit count map has weight piled up at such locations. In Fig. 4.2, the weights piles up at the northern and southern limits of each set of scans, corresponding to declination -16° and -6° for scans at 47 and 37 degrees elevation respectively. The mottling in the mid-latitude is due to the uneven distribution of sidereal start time for this particular month, which is one of about 15 months of accumulated observational data over the entire C-BASS north survey.

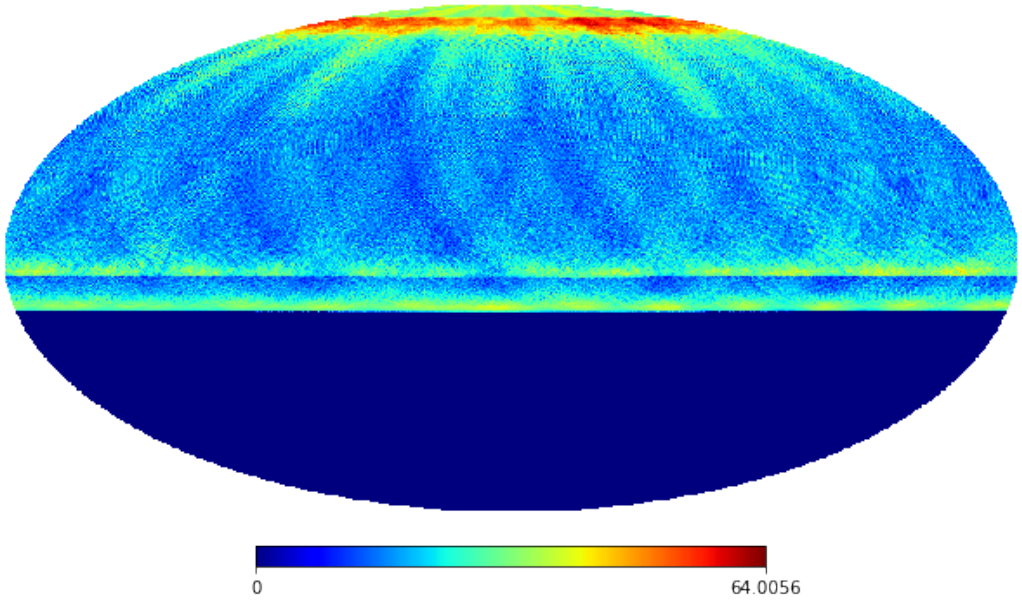


Figure 4.2: Hit count map for C-BASS. The unit of the colour scale is hits.

4.2 Results from interpolation onto the *HEALPix* grid-based map-maker algorithm

The 2D interpolation onto the $HEALPix$ grid projection map of size $5N_{side} \times 5N_{side}$, where $N_{side} = 512$ obtained from the grid interpolation map-making algorithm discussed in Section 3.3 is shown in Fig. 4.3. This is the weighted map obtained after the post processing and edge removal carried out as discussed in Sec. 3.4 of Chapter 3. It shows the equatorial and northern polar regions. The *Python* library *Matplotlib* is used for displaying the 2D map. However, this is not the usual representation scheme for maps. The 2D map was saved as a FITS file and is converted to a 1-D *HEALPix* array in *RING* ordering using the *IDL* program *HPX2HP_ARRAY* written by J.P. Leahy. Once in this standard form, the map could be displayed using standard *HEALPy* routines. The corresponding map in Mollweide projection produced by the function mollview of *HEALPy* is shown in Fig. 4.4. The pixels which don't have any data are assigned to grey values for display purpose.

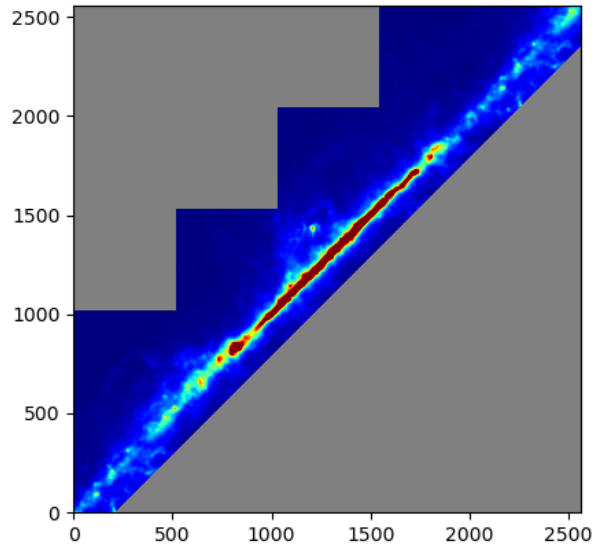


Figure 4.3: Interpolation onto the *HEALPix* grid projection map

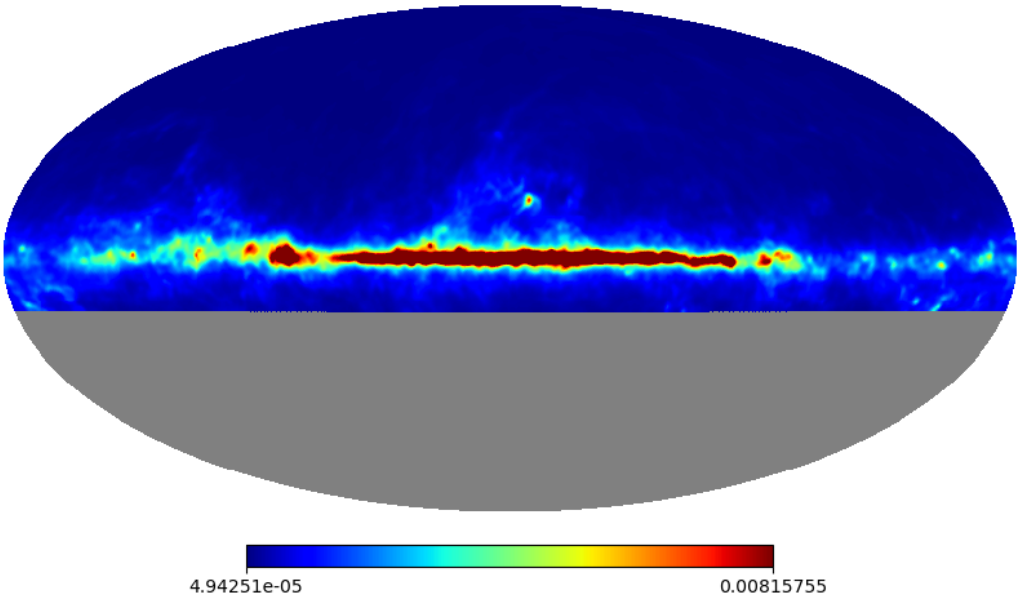


Figure 4.4: Mollweide projection of interpolation onto the *HEALPix* grid projection map.

4.3 Comparison of results

In Chapter 2 the commonly used binning map-making algorithm and the standard interpolation map-making algorithm are discussed. Interpolating onto the *HEALPix* grid is an approximation to the standard interpolation: although they use the same interpolation formula, there are subtle differences, especially in the truncation of the interpolation function, that will be explored in this chapter. Hence, we consider the standard interpolation algorithm to be the most precise method and therefore it is considered as a benchmark for other algorithms. In this section, we compare the standard interpolation map-making algorithm with the rest of the algorithms suggested in Chapter 2, 3.

4.3.1 Standard interpolation results

The map obtained from the standard interpolation algorithm as discussed in Section 2.2 of Chapter 2 using the data set mentioned in Section 4.1 is shown in Fig. 4.5. The interpolated map blurs the image by the width of the interpolation function. In practice, the interpolation width needs to be smaller than the telescope beam. With this, the true sky is blurred twice, once with the telescope beam and once with the interpolation function. Assuming that both the telescope beam and interpolation function are Gaussian, the net effect is the same as blurring with a single beam of FWHM = $\sqrt{FHWM_{tel}^2 + FHWM_w^2}$, the widths add in quadrature, where $FHWM_w$ is the FWHM of the interpolation window . But for the *Planck* 857 GHz data, the telescope FWHM ($FHWM_{tel}$) was 5 arcmin. After re-binning to $N_{side} = 512$, the pixel size is about 6.9 arcmin. Effectively, the map has been convolved with a pixel shape so the re-binned map has a resolution very roughly equal to 8.5 arcmin. Although the telescope beam is not quite Gaussian it is close enough for the effective FWHM to be roughly correct. For test purpose, we use FWHM = 1° for the interpolation function which is substantially larger than the resolution of the input map.

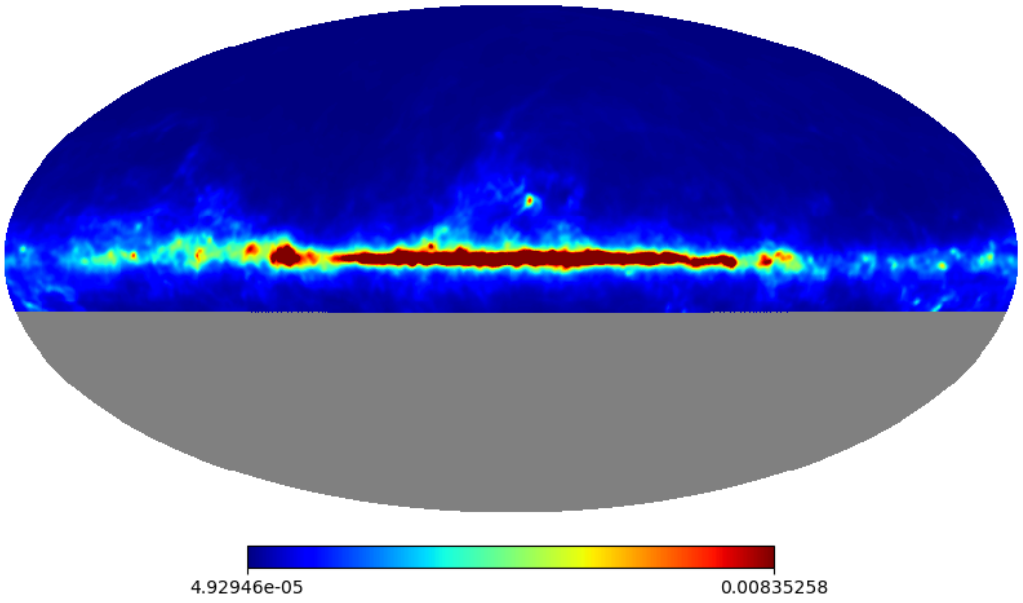


Figure 4.5: Standard interpolation all-sky map

4.3.2 Binning-based map-making algorithm results

The Fig. 4.6 shows the map obtained from the binning algorithm discussed in Sec. 2.1 of Chapter 2. It is the most computationally fast algorithm as it doesn't involve heavy computational elements. As expected, this map is exactly same as the input *Planck* data map. The standard interpolation map as discussed in Section 4.3.1 is convolved twice. Hence, for comparing it with the binning map the latter should be convolved with a Gaussian distribution of the same FWHM.

The *HEALPix* routine *smoothing* with FWHM = 1° is used for smoothing the binning map. In Fig. 4.7 we compare a small patch of resolution $0.5'$ on the sky with a bright source at $(-7, +17)$ degrees. Fig. 4.7a shows the bright source from the original binning map and Fig. 4.7b shows the same bright source after smoothing the binning map with the *smoothing* function. There is clear visual evidence of smoothing between the two sky images.

In the original binning map, the pixels at which the map returns zero value

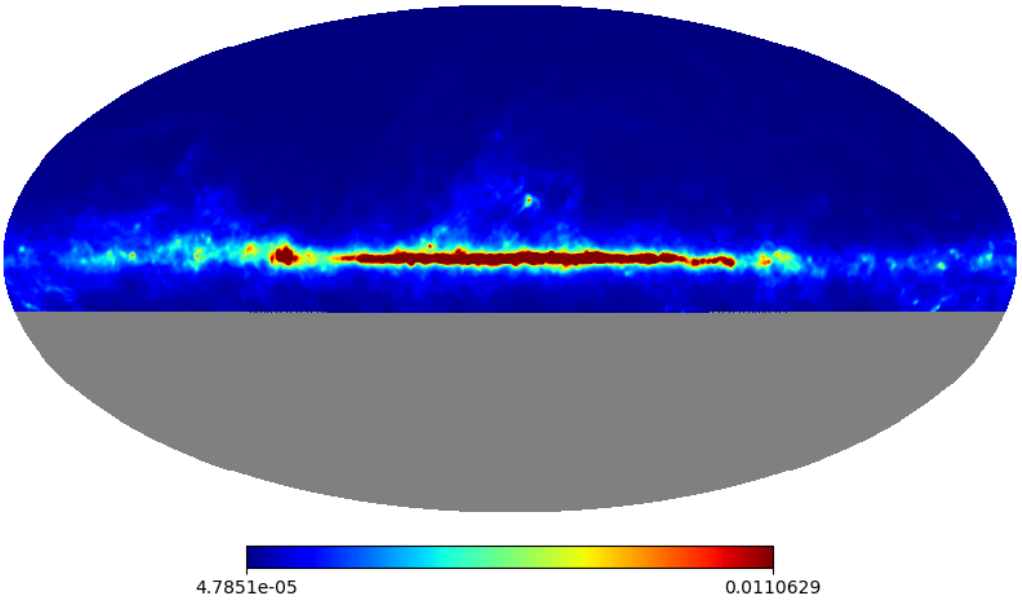
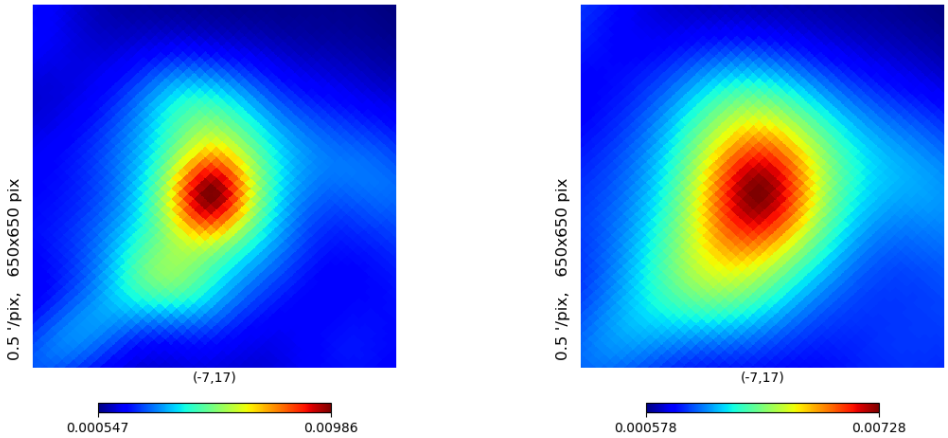


Figure 4.6: Binning based full-sky map



(a) Original binning based map.

(b) Binning map after smoothing with FWHM = 1°.

Figure 4.7: Comparison of a small patch of sky of resolution 0.5′ in binning map before and after the implementation of smoothing function with FWHM = 1°.

as there is no pointing data are considered as the bad pixels. These bad pixels are set to blank values in the difference and fractional difference maps throughout the analysis.

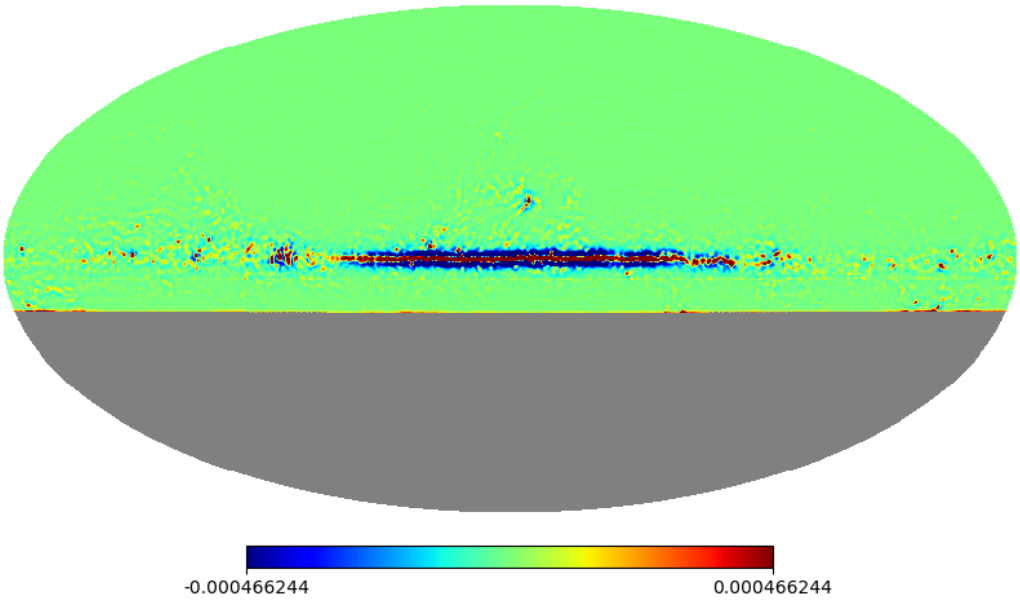


Figure 4.8: Difference map for binning method with the standard interpolation map

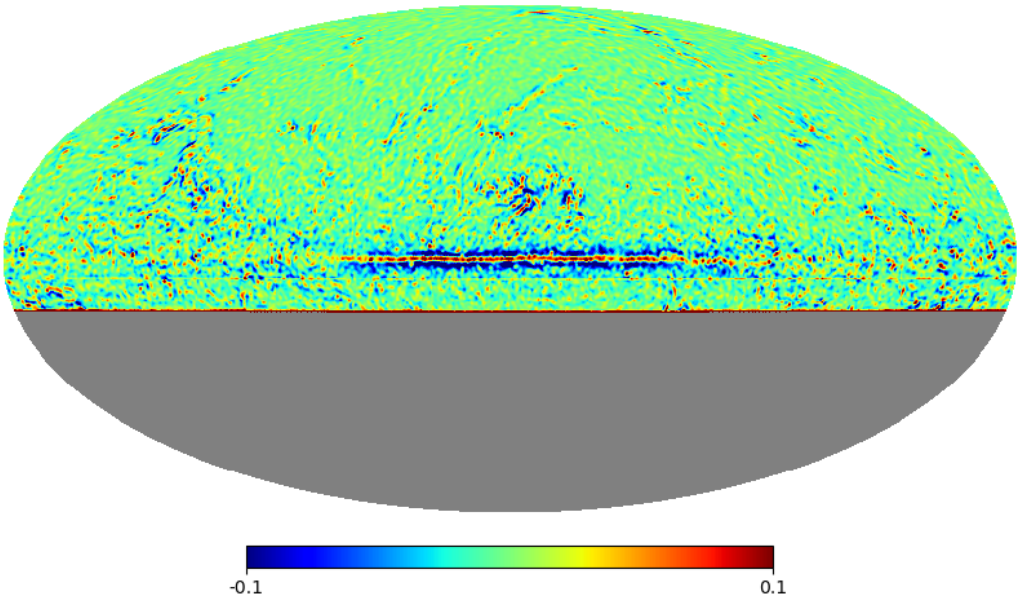
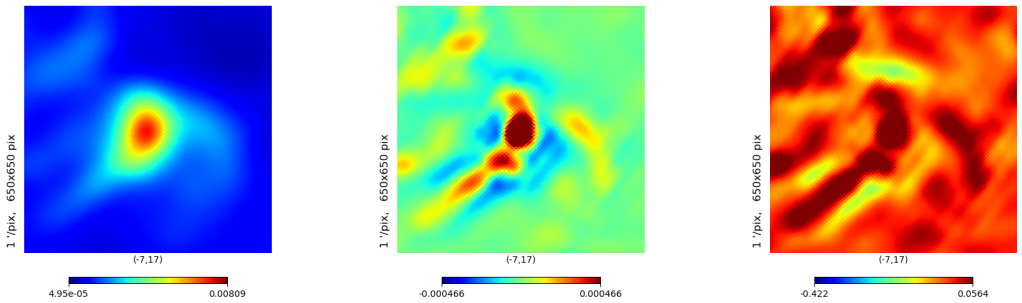


Figure 4.9: Fractional difference map for binning with the standard interpolation map.

The resultant smoothed binning map is then subtracted from the standard interpolation map and the resultant Mollweide projection of the difference

map is given in Fig. 4.8. A fractional difference map is created by dividing the difference map, obtained by subtracting the binning map from the standard map, with the standard interpolation map. The resultant fractional difference is given in Fig. 4.9. It highlights the regions where there are sharp features like the gradient. This might be because of the slightly different beam shapes considered.

For display, the limits of the range for the difference map is between $\pm 2\sigma_{diff}$, where σ_{diff} is the standard deviation of the difference map and for the fractional difference map the limits are set to ± 0.1. The all-sky Mollweide projection maps show that we are correctly dealing with the overall organization of the *HEALPix* grid. The remaining errors are on the scale of the interpolation function, which is too small to be seen on an all-sky image. Therefore, we use the *gnomview* function of *HEALPy* to take a closer look at various specific regions on the map.

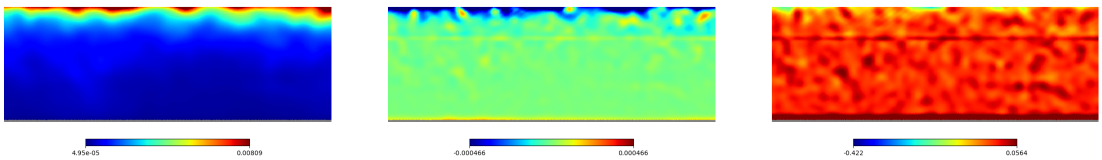


(a) Smoothed binning map. (b) Difference map. (c) Fractional difference map.

Figure 4.10: Comparison of bright source at $(-7°, 17°)$ extracted from the smooth binning and difference, fractional difference maps of binning method.

In all-sky maps of difference and fractional difference map, we are interested in analysing a bright region on the Galactic plane and a patch overlapping the two horizontal lines of abrupt changes in the hit count map, Fig. 4.2 of Section 4.1. For bright region, the bright spot at (longitude, latitude) = $(-7°, +17°)$ with 1 arcmin resolution is considered. This is the star-forming

region around the star rho Ophiucus. The corresponding regions are extracted from the smoothed binning and difference, fractional difference maps and is displayed in Fig. 4.10 using *gnomview* function. The limits of the range of the bright region image extracted from the fractional difference map is set between the minimum of the fractional difference binning map and 0.1 times the maximum of the fractional difference binning map. The bright source has a peak value of 0.184 in the fractional difference map. The differences are small due to the fact that the interpolation function is a truncated Gaussian and therefore slightly narrower than the full Gaussian produced by the smoothing program.



(a) Smoothed binning map. (b) Difference map. (c) Fractional difference map.

Figure 4.11: Comparison of a horizontal patch between declination of -16 and -6 degrees in the smoothed binning map and difference, fractional difference maps of binning method.

For comparing the horizontal jumps in the hit count density, *cartview* function of the *HEALPy* module is used. We analyse a horizontal strip with the latitude range of $[-16°, -2°]$ and longitude range of $[-20°, 20°]$. The corresponding regions from the smooth binning map, difference and fractional maps of binning are extracted and compared in Fig.4.11. In the fractional difference map, at a declination of -16 and -6 degrees there are two horizontal thin lines. They coincide with the southern declination limits of the scans at elevation 47 and 37 degrees. At declination of -16 degree there is also an edge effect as the interpolation window in the standard method is truncated to zero for the pixels beyond the edge while the smoothed binning map still has non-zero values.

With this method, we report a standard deviation of 2.33×10^{-4} for the difference map and 2.78×10^{-3} for the smoothed binning map.

4.3.3 Results for interpolation onto the *HEALPix* grid

In interpolation onto the *HEALPix* grid map, the query patch considered is inevitably different from the standard interpolation method. The query patch considered is a square window which encloses a circle of radius 1°. The Mollweide projection of the grid-based map given in Fig. 4.4 is subtracted from the standard interpolation map. The resultant difference map is given in Fig. 4.12. The corresponding fractional difference map is created by dividing the difference map, obtained by subtracting the grid map from the standard interpolation map, by the standard interpolation map. The resultant fractional difference map is given in Fig. 4.13.

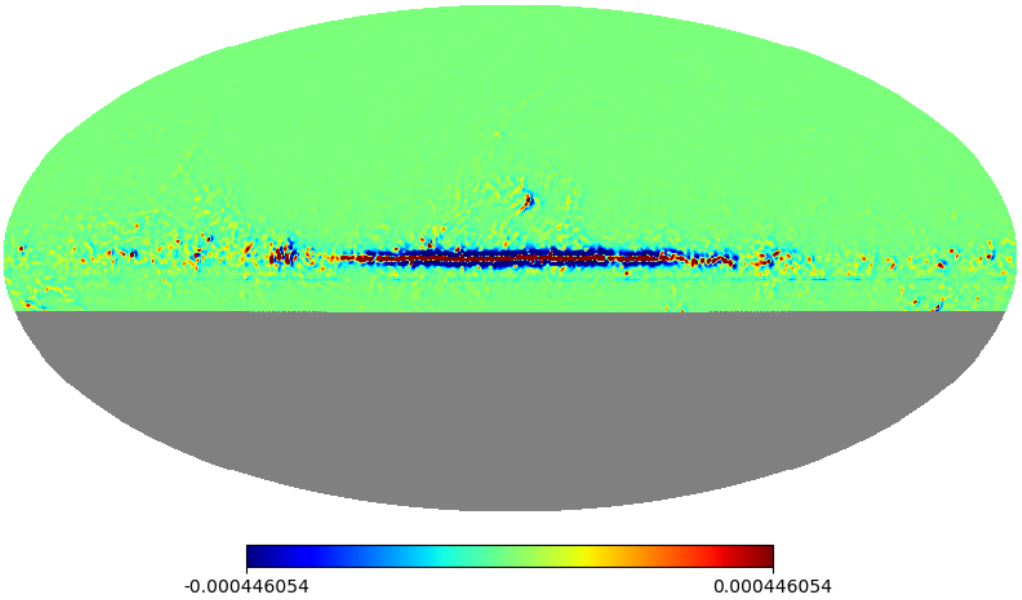


Figure 4.12: Difference map of grid interpolation from the standard interpolation map.

For display purposes, the range of the difference map is set to be between $\pm 2\sigma_{diff}$, where σ_{diff} is the standard deviation of the difference map and for

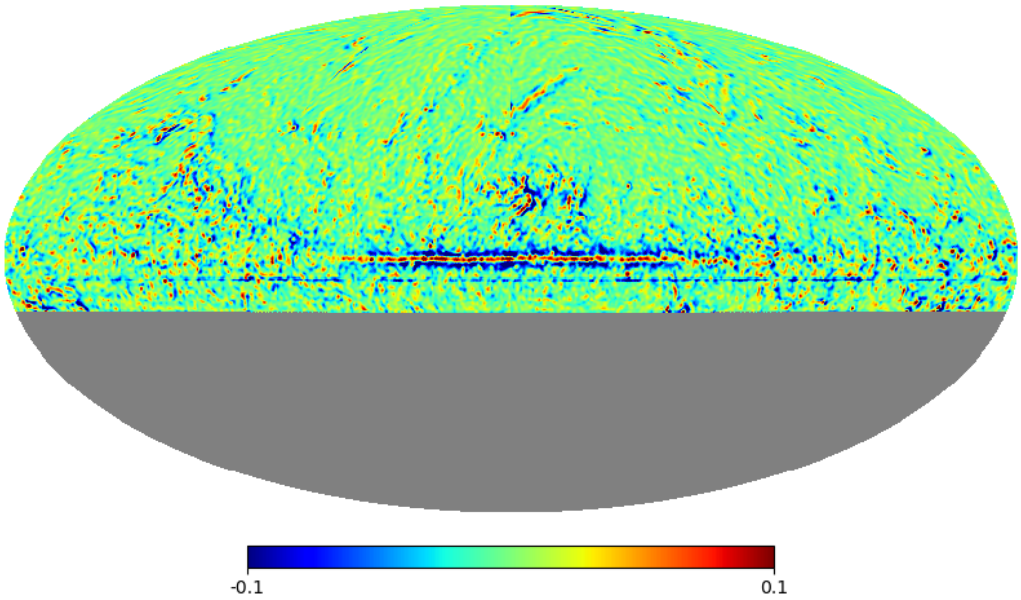
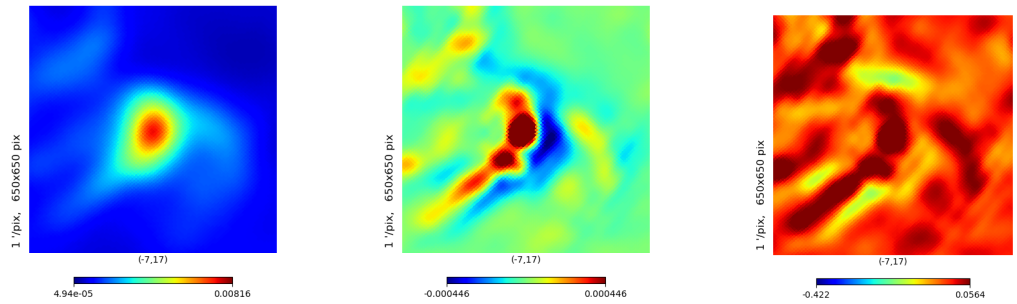


Figure 4.13: Fractional difference map of grid interpolation map with the standard interpolation map.

fractional difference map, the limits are set to ± 0.1 For both the maps, the bad pixels obtained from the original binning map where the map returns zero value are set to blank values.

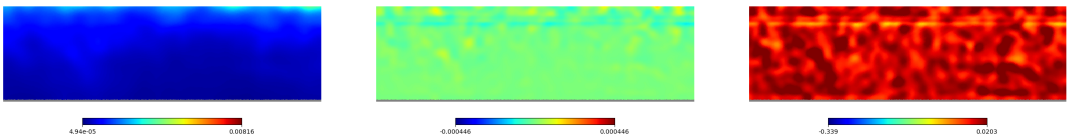
In all-sky maps the errors are on the scale of the interpolation function. Just as discussed for the binning based maps, we use *gnomview* function of *HEALPy* for analysing specific regions of the sky. In grid interpolation maps, the interested regions to analyse includes a bright source, polar regions, horizontal strip in the Galactic plane and north pole. For a bright region, the star-forming region around the star rho Ophiucus at (longitude, latitude) = $(-7^{\circ}, 17^{\circ})$ is considered. The corresponding regions were extracted from the grid interpolation, difference and fractional difference maps. The results are shown in Fig. 4.14. The limits of the range of the bright region image extracted from the fractional difference map is set between the minimum of the fractional difference map of grid interpolation map and 0.1 times the maximum of the fractional difference map of grid interpolation map. The bright source at

its peak in the fractional difference map has a value of 0.184. For analysing the horizontal patch overlapping the declination $(-6, -16)$ degrees, a horizontal strip with a latitude range of $[-16°, -2°]$ and longitude range of $[-20°, 20°]$ was considered. The corresponding horizontal stripes are extracted from the grid interpolation map along with difference and fractional difference maps. These are shown in Fig. 4.15. The horizontal line at declination of -6 degrees can be seen in the fractional difference map, but the edge effect at declination of -16 degrees is not very evident.



(a) Grid interpolation map. (b) Difference map. (c) Fractional difference map.

Figure 4.14: Comparison of bright sources at $(-7, 17)$ degrees in the interpolation grid map, difference map and fractional difference map of grid interpolation method.



(a) Grid interpolation map. (b) Difference map. (c) Fractional difference map.

Figure 4.15: Comparison of horizontal strip between declination -16 and -6 degrees in the interpolation grid and difference, fractional difference maps of grid interpolation method.

In the *HEALPix* grid geometry, the facets are square shaped. These facets in the polar region as discussed in 3.4.3 needs to be stitched together. This is a region where grid interpolation is likely to go wrong. For analysing these

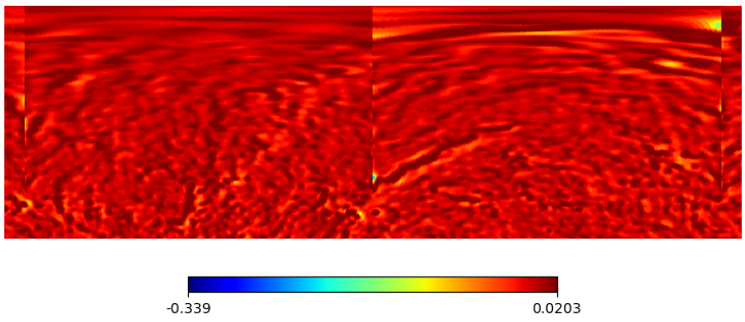


Figure 4.16: Fractional difference map of grid interpolation between latitude range of $(10°, 90°)$ and longitude range of $(-100°, 100°)$.

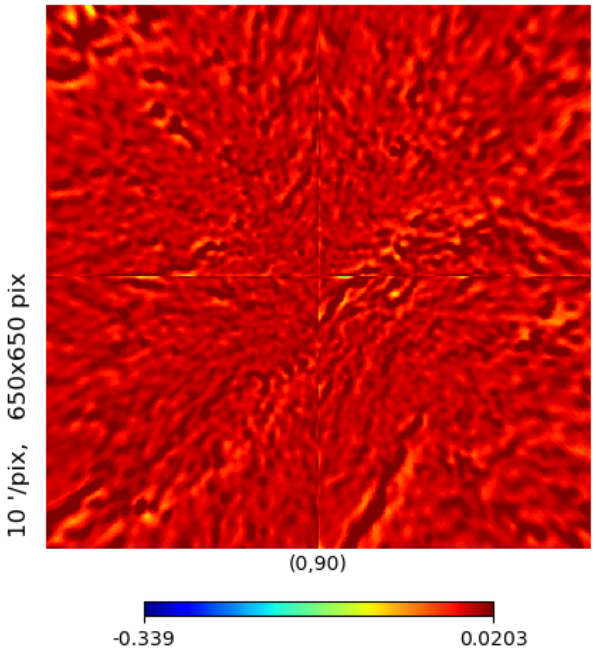
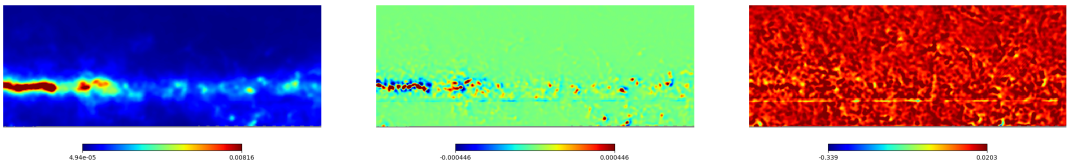


Figure 4.17: Fractional difference map for grid interpolation map at the north pole at longitude, latitude = $(0, 90)$ degrees.

polar facet edges, the region on the all-sky map corresponding to a latitude range of $[10°, 90°]$ and longitude range of $[-100°, 100°]$ is extracted from the

fractional difference map. The corresponding map is shown in Fig. 4.16. As expected at longitude = $0°, -90°, 90°$, the joins between two facets are evident. We also analysed the fractional difference map at the north pole cap given by longitude, latitude = $(0°, 90°)$ with a resolution of 10 arcmin. The corresponding map is given in Fig. 4.17. Clearly, the facet edges are visible. Additionally, there is also a problem in the interpolation at corners of the polar region gores in the grid at DEC = 41 degrees and at the poles, as discussed in Section 3.4.4 of Chapter 3. The demarcation lines at the facet edge maybe a bug in the program but unless the gore issue is fixed, the interpolation is not complete at the polar regions. Hence, further analysis and testing needs to be done for this region.



(a) Grid interpolation map. (b) Difference map. (c) Fractional difference map.

Figure 4.18: Comparison of facet 6 region between longitude range of $[-180, -60]$ degrees and latitude range of $[-16, 30]$ degrees in the interpolation grid and difference, fractional difference maps of grid interpolation method.

The half filled facet 6 in *HEALPix* geometry is dealt by a mirror averaging operation as discussed in Section 3.4.2 of Chapter 3. To analyse this region, we consider a longitude range of $[-180, -60]$ degrees and latitude range of $[-16, 30]$ degrees in the all-sky map. The corresponding window was extracted from the full-sky grid interpolation, difference and fractional difference maps and these are displayed using the *cartview* routine of *HEALPy*. The extracted region corresponding to the facet 6 is given in Fig. 4.18. There are no evident demarcation lines present. Hence, we conclude that the facet 6 operation implemented is a perfectly correct method.

Overall, we report a standard deviation of 2.23×10^{-4} for the difference

map and 2.7×10^{-4} for the grid interpolation map.

4.3.4 Comparison of computational time

The different map-making algorithms developed needs to be compared with respect to their computational time. The algorithms were tested with *Planck* 857 GHz data on C-BASS pointing data contained in 384 FITS files. They were tested on a workstation with an AMD Ryzen Threadripper 1950 X 16-Core / 32-Thread Processor. It has a base clock speed of 3.4 GHz and maximum single thread boost to 4.0 GHz. The internal RAM memory is of 128 GB with a Samsung 512 GB Solid State Drive and a 4 TB Hard Drive in RAID1 mirror configuration. The computational time of each algorithm is timed by using *Python* function *tqdm* which returns the time taken in HH:MM:SS clock format. The tabulated computational time for different map-making algorithms is given in Table 4.1.

Table 4.1: Comparison of computational time for map-making algorithms

Method for map-making algorithm	**Computational time** (HH:MM:SS)
Binning	00:00:20
Standard Interpolation	4:33:50
Polynomial map making	
Polynomial order 1	5:26:07
Polynomial order 2	5:51:09
Polynomial order 6	9:43:29
Polynomial order 10	12:17:35
Polynomial order 20	18:12:19
Polynomial order 50	41:06:32
Haver-sine based algorithm	142:14:46
Interpolation onto the *HEALPix* grid (grid interpolation)	9:53:58

As expected, binning is by far the fastest algorithm. Except, that the polynomial interpolation is surprisingly slow and the interpolation onto the *HEALPix*

grid method is much slower than expected. There is significant scope for optimisation and this will be analysed in detail beyond the scope of this dissertation.

Chapter 5

Elliptical beam compensation

This chapter focuses on Elliptical Beam Compensation, which is an extended piece of work done in addition to the primary problem statement of this dissertation.

5.1 Asymmetric beam profile and deconvolution

The map-making algorithms, discussed in Chapter 1, are most appropriate for a symmetric beam profile, which is used to reduce the complexity of the map-making method. In WMAP and *Planck* maps, data is assigned to the pixel where the beam centre falls without correcting for the beam shape. However, it is advantageous to use an elliptical interpolation function to compensate for the telescope beam ellipticity. With the future CMB projects aiming towards being able to detect feeble polarization signals, it is necessary to account for the systematic errors such as asymmetric beams. Without compensation, asymmetric beams can be a source of systematic errors in the map-making process which will reduce the accuracy of results. Therefore, it is important to address the case of asymmetric beams in the map-making process.

With an asymmetric detector beam, every pixel is smoothed with a different response, unlike in the case of symmetric beams where the resulting maps are

smoothed by the same beam. For an asymmetric beam, the effective smoothing varies with the position. The local ellipticity depends on both the orientation of the scan(s) crossing that pixel and the orientation of the beam relative to the scan direction. Considering an asymmetric beam shape, an immediate effect is that the image of a point source is no longer symmetric. Rather, it takes the shape of the sum of the detector beams rotated by the various different orientations of the scans passing over it. By deconvolution, a correction is made during the map-making pipeline to deconvolve this instrumental beam effect.

Beam deconvolution is usually carried out in either harmonic space or in pixelated space, where the latter is computationally expensive. For deconvolution, the effective beam depends upon two factors, one being the scanning strategy and the other being detector setup. The ArtDeco deconvolution algorithm (Keihänen & Reinecke 2012) is based on the Wigner function where no assumption is made about the scanning strategy. Tests were conducted on simulated temperature and polarization data. With deconvolution, effects of beam asymmetry from temperature maps are completely removed such that the circular shape of point sources are restored. While for polarization, with an asymmetric beam profile and in the absence of noise, deconvolution was able to remove the temperature leakage. But in the presence of noise, the leakage removal was less efficient (Keihänen et al. 2017). Keihänen et al. (2019) discusses the results when the ArtDeco deconvolution algorithm was tested on LFI *Planck* data. Though simulations show that the deconvolution can reduce the temperature leakage problem, real measurement results show that it amplifies noise; they also lists bandpass mismatch as another source for leakage. A detailed discussion of the noise effect on LFI data for ArtDeco deconvolution algorithm is given by Keihänen et al. (2016). Armitage & Wandelt (2004) discusses an iterative maximum likelihood-based deconvolution method to remove artefacts due to an arbitrary beam shape. This was tested on WMAP scan

strategy for temperature measurements and showed promising results. The polarization counterpart is discussed in Armitage-Caplan & Wandelt (2009), where deconvolution is done in harmonic space and thereby avoids artefacts due to pixelization. Other deconvolution algorithms include the work discussed in Harrison, van Leeuwen & Ashdown (2011); Wallis et al. (2014); Mitra et al. (2011).

It is interesting to contrast the relative failure of deconvolution approaches for single dish maps with the success of the CLEAN algorithm in synthesis imaging, which is often used to slightly increase and/or circularise the resolution. This succeeds at least up to a point due to two properties of synthesis images: firstly, the noise is correlated over the synthesized beam in exactly the same way as the signal, and secondly the sky is usually effectively empty except for isolated sources since interferometry filters out large-scale structure. In contrast, the noise in single-dish maps is uncorrelated between samples, whereas the signal is correlated over the telescope beam. Therefore there are high spatial frequency components to the noise which are inevitably amplified by any attempt to improve the resolution, which involves amplifying such high-frequency components. Of course, using CLEAN to super-resolve synthesis data is well known to degrade the image fidelity, but the onset of this problem is much less rapid than in the case of attempts to deconvolve single-dish maps.

The deconvolution algorithms like ArtDeco, which aim to increase the resolution by reducing the effective beam in just one direction will inevitably amplify the noise. We suggest an elliptical beam compensation method which circularizes the beam by widening it in the narrow direction instead of shrinking it in the long direction, which reduces the noise rather than increasing it. The drawback of this method is a slightly lower resolution, which is not critical for most cases. Overall in this chapter, an elliptical interpolation function is developed which can be used to approximately circularize an elliptical telescope

beam.

5.2 Elliptical beam profile compensation method

An elliptical beam compensation method is proposed in the pixel space. Fig.5.1 shows the scanning track of a radio telescope under consideration. The cross

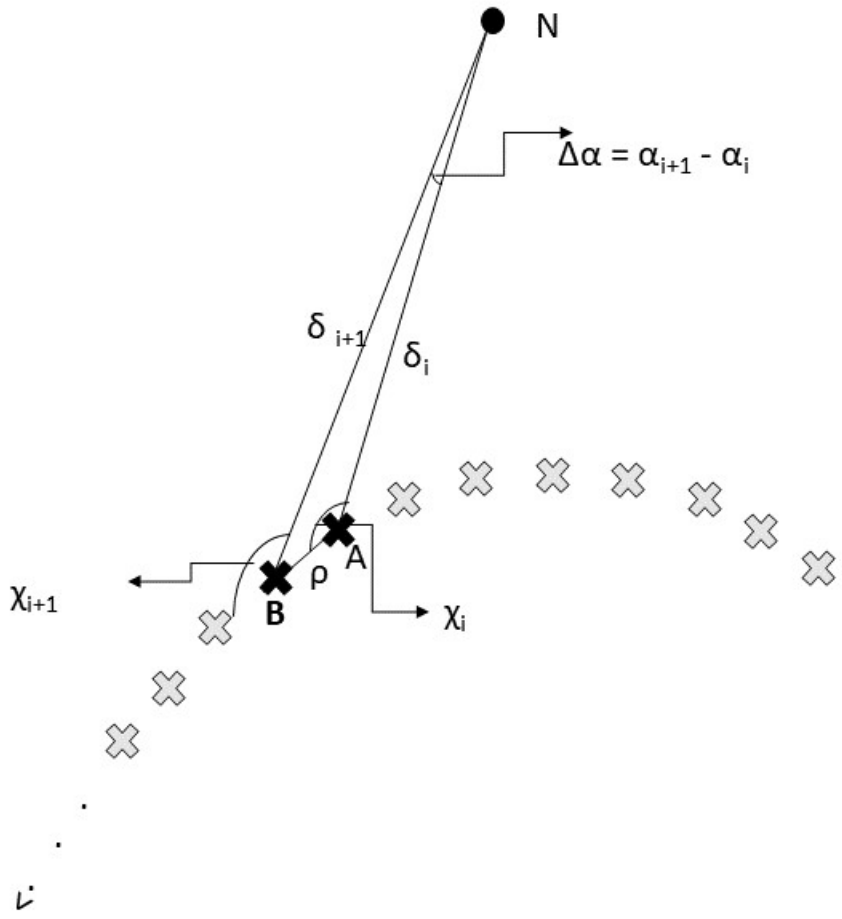


Figure 5.1: Beam scanning track.

marks in the Fig. 5.1 represents the data points measured along the scanning track while the highlighted points represent the two data points under consideration. The two consecutive points A, B are indexed at i and $i+1$. Let χ be the local position angle of the scan path measured anticlockwise from due north.

The aim is to derive the position angle χ_i of point A on the sky relative to the point B. For each of the data points on the scanning track, its location is defined by (δ, α) which represents declination and right-ascension respectively. The distance between any two successive data points is represented by ρ. By the cosine rule,

$$\cos\chi_i = \frac{\cos\rho\cos\delta_i - \cos\delta_{i+1}}{\sin\rho\sin\delta_i}. \tag{5.1}$$

By the sine rule,

$$\frac{\sin\chi_i}{\sin\delta_{i+1}} = \frac{\sin\Delta\alpha}{\sin\rho}. \tag{5.2}$$

From equation 5.1, 5.2, $\cos\chi_i$ and $\sin\chi_i$ gives an angle between 0 and π but does not resolve the ambiguity between positive and negative position angles. Hence, the position angle must be represented as a function of inverse tangent which allows to produce an angle in the complete range $-\pi \leq \chi \leq \pi$. Using four-part formula (Smart, Smart & Green 1977), the position angle is given by

$$\tan\chi_i = \frac{\sin\Delta\alpha}{\cos\delta_i\tan\delta_{i+1} - \sin\delta_i\cos\Delta\alpha}. \tag{5.3}$$

For data points very close to each other, we use second-order small angle approximation for $\sin\Delta\alpha$, $\cos\Delta\alpha$ and $\tan\delta_{i+1}$. Writing $\delta_i = \delta$ and $\delta_{i+1} = \delta + \Delta$, where Δ is assumed to be a small increment in δ, gives

$$\sin\Delta\alpha \approx \Delta\alpha, \tag{5.4}$$

$$\cos\Delta\alpha \approx 1 - \frac{\Delta\alpha^2}{2}, \tag{5.5}$$

$$\tan\delta_{i+1} = \tan(\delta+\Delta), \tag{5.6}$$

$$= \frac{\tan\delta + \tan\Delta}{1 - \tan\delta\tan\Delta}, \tag{5.7}$$

$$\approx \frac{\tan\delta + \Delta}{1 - \Delta\tan\delta}. \tag{5.8}$$

Substituting equation 5.4, 5.5 and 5.8 in equation 5.3 obtain,

$$\tan\chi_i = \frac{\Delta\alpha}{\cos\delta\Big(\frac{\tan\delta+\Delta}{1-\Delta\tan\delta}\Big) - \sin\delta\Big(1-\frac{\Delta\alpha^2}{2}\Big)}, \tag{5.9}$$

$$= \frac{\Delta\alpha(1-\Delta\tan\delta)\cos\delta}{(\sin\delta+\Delta\cos\delta)\cos\delta - \sin\delta\Big(1-\frac{\Delta\alpha^2}{2}\Big)\Big(\cos\delta-\Delta\sin\delta\Big)}, \tag{5.10}$$

$$= \frac{\Delta\alpha(1-\Delta\tan\delta)\cos\delta}{\Delta+\frac{\alpha^2}{2}\sin\delta(\cos\delta-\Delta\sin\delta)}, \tag{5.11}$$

$$\approx \frac{\Delta\alpha\cos\delta}{\Delta}. \tag{5.12}$$

A detailed derivation of the position angle χ_i is discussed in Appendix B. Equation 5.12 is used to calculate the position angle χ_i of the data points, where $i \in 1, 2....N-1$ and N is the length of the data.

To compensate for the elliptical beam, the interpolation function needs to be broad along the minor axis of the beam and narrow along the major axis, so the combination gives a nearly circular beam (exactly circular if the telescope beam and interpolation function are both elliptical Gaussian). Thus the correction term is given by

$$\gamma^2 = b^2 - a^2, \tag{5.13}$$

where a is the minor axis and b is the major axis of the ellipse which are the properties of the telescope. The elliptical interpolation function (w) is given by,

$$w = \exp\left[-0.5\left(\frac{x^2}{\sigma_i^2}+\frac{y^2}{\sigma_j^2}\right)\right], \tag{5.14}$$

where (x, y) are the coordinates of the pixel centre relative to the data point. The standard deviation σ_i, σ_j define the size and shape of the beam. The major axis of the interpolation function (d) should be bigger than the γ defined in

equation 5.13 and the minor axis of the interpolation function (c) needs to be small but finite. This gives $b^2 + c^2 = a^2 + d^2 = f^2$, where f is the FWHM of the effective beam which is now circular after the data have been interpolated. Thus, $d^2 = \gamma^2 + c^2$. The area of the interpolation function is ($1.1 \times c \times d$) and there needs to be several pixels of areas for the interpolation to work. Hence, c = 0 cannot be used. This gives the standard deviation of the beam as

$$\sigma_i^2 = d^2/k_2^2, \tag{5.15}$$

$$\sigma_j^2 = c^2/k_2^2, \tag{5.16}$$

where $k_2 = \sqrt{8 \ln 2}$.

Consider the elliptical beam geometry as shown in Fig. 5.2. The plus marks indicate the data points and cross mark indicate the pixel centre. An elliptical beam is centered at the data point and its major axis is aligned with the x-axis of the frame of reference. The dotted line running horizontally through the centre of the ellipse is the scanning axis of the telescope. The label N represents the north pole. The pixel centre is given by the coordinate (δ_P, α_P). The angle Ψ_i in this Figure is the position angle of the pixel centre relative to the data point and is measured in anti-clockwise from the data point. For the pixel centre with Cartesian coordinate (x, y) and included angle ϵ gives the projection

$$x = \rho \cos \epsilon, \tag{5.17}$$

$$y = \rho \sin \epsilon, \tag{5.18}$$

where ϵ is the angle between the major axis of the ellipse and the direction from the data point to the pixel centre. From the elliptical beam geometry of Fig. 5.2, considering the angles around the data point we get the condition

$$360^\circ = \chi + \Omega + \epsilon - \Psi, \tag{5.19}$$

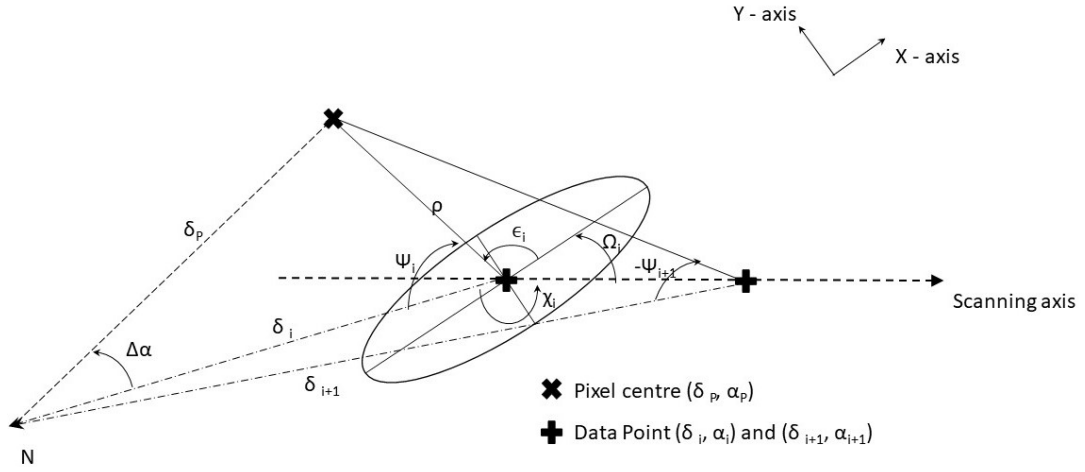


Figure 5.2: Elliptical beam geometry

where Ω is the orientation angle of the beam major axis relative to the scan direction and is pre-defined for the detectors in *Planck* and WMAP. For implementing the grid-based interpolation method with elliptical beam profile, the unknowns ϵ and Ψ needs to be estimated. Considering the sine rule for the elliptical beam geometry,

$$\frac{\sin\Psi_i}{\sin\delta_P} = \frac{\sin\Delta\alpha}{\sin\rho}, \tag{5.20}$$

where δ_P gives the declination of the pixel centre on the sky and $\Delta\alpha$ is the difference in the right-ascension between the data point and the pixel centre. The ρ in this equation is the small angle approximated distance between the data point and pixel centre. Using the cosine rule,

$$\cos\Psi_i = \frac{\cos\rho\cos\theta_i - \cos\delta_P}{\sin\rho\sin\theta_i}, \tag{5.21}$$

where θ_i is the colatitude of the data point under consideration. To get a complete sky coverage for position angle ($-\pi \leq \Psi_i \leq \pi$), it is defined as the

inverse of the tangent. Dividing equation 5.20 by 5.21 gives

$$\tan \Psi_i = \frac{\sin \Psi_i}{\cos \Psi_i}. \tag{5.22}$$

With this, we have all the equations required to estimate interpolation for an elliptical beam. Implementing the algorithm in pixel space, we expect it to be computationally expensive. However, right now we are interested only in developing a fully functional algorithm and not in optimizing for computational speed.

5.3 Results and Discussion

For testing this algorithm, a simulated point source map is created using IDL. A function *rcat* is written which generates a map with a desired number of point sources scattered randomly over the sky, e.g. 2000 point sources of FWHM = 15′ and $N_{side} = 1024$. The map was made with a finite source size to make sure that they were caught by the test scan strategy. Fig. 5.3 shows a small patch of resolution 1.5′ on the sky with point sources. This simulated point source map along with C-BASS pointing data are used as inputs for the elliptical beam compensation algorithm.

The scan strategy of the C-BASS pointing data is shown in Fig. 4.2 of Chapter 4. The problem with this scan data is that each point on the sky is covered by scans going in two or four directions depending on whether they are in the region covered by the elevation = 47° scans. This makes it impossible to see the simple elliptical shape of the interpolation function. To avoid this and to speed up the test, a very short stretch of C-BASS observations is used so that we could focus on a part of the sky covered only by one set of scans. But with a short set of data, the scans have large gaps between them. To ensure that the test sources are hit by the test scans, the point sources are made with a finite

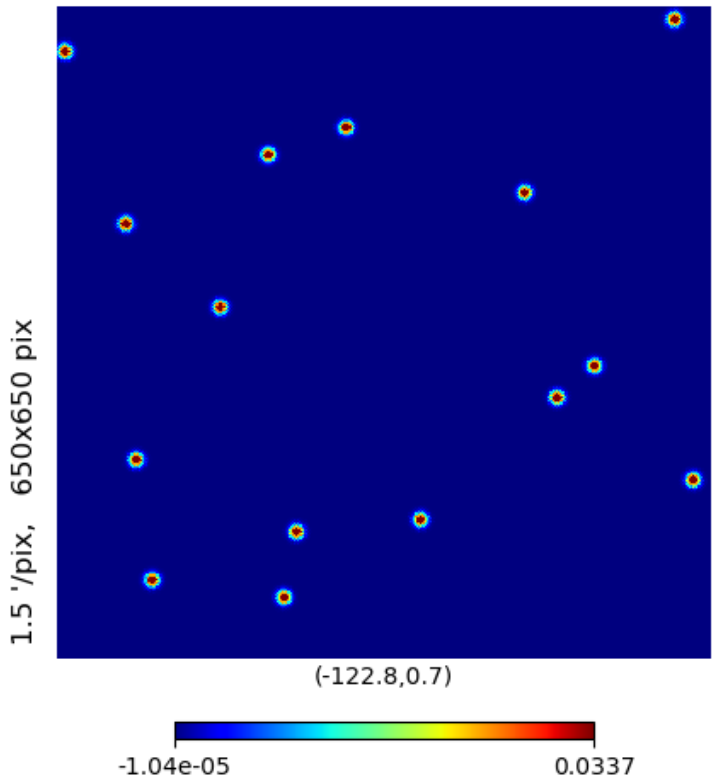


Figure 5.3: Simulated point source input map of size $1.5' \times 1.5'$.

size (15 arcmin). These hits are rarely through the peak of the source on the sky. The data sees only a fraction of the source intensity, so in the interpolated map the sources appear with different peak brightness even though the model sources are all having the same peak brightness.

For test purposes an interpolation function with major axis of 1.5° and minor axis of 0.5° is used. The beam orientation angle is assumed to be 90°. However, this could be any value and wouldn't affect the shape of the ellipse nor the computational time. For enclosing the modelled ellipse within the interpolation window, the window width is defined to be 2°.

The terms $\cos\delta_P$, $\sin\delta_P$ given in equation 5.20, 5.21 are 2D arrays and needs to be calculated and tabulated prior to the map-making process. These ma-

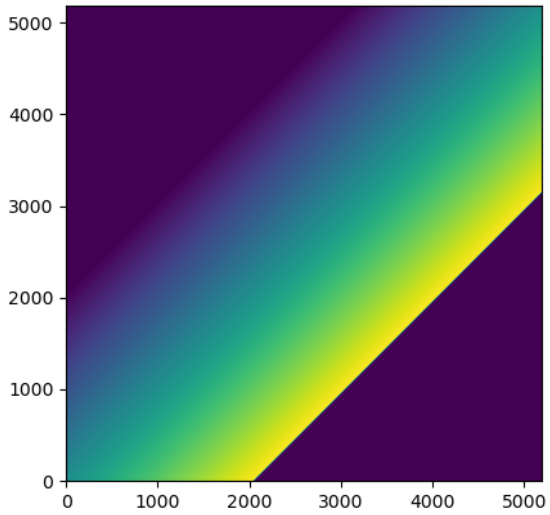


Figure 5.4: Diagonal index 2D array in *HEALPix* grid geometry.

trices corresponds to the sine and cosine values of the declination of the pixel centres in the *HEALPix* grid geometry. The declination values of the pixel centres correspond to the declination of the isolatitude rings. For a given N_{side}, *pix2ang HEALPy* function is used to find the latitude and longitude of pixel centres in radians for the HEALPix geometry. As discussed in Section 1.7 of Chapter 1, isolatitude rings are the constant circles around the sphere where the pixel centres are placed. Thus, the iso-latitude rings are the unique values of the latitude values. Let sin_{ring}, cos_{ring} corresponds to the sine and cosine of the isolatitude rings. The value of sin_{ring}, cos_{ring} 1D arrays needs to be converted to the *HEALPix* grid geometry and this is represented by $\sin\delta_P, \cos\delta_P$ respectively. In order to convert the 1D array to the *HEALPix* grid, we create an index 2D array which gives the index of the isolatitude rings on the rotated $HEALPix$ projection. To create this index array, a square grid of size $nsize + wsize$ is created using the function *meshgrid*, where $nsize = 5N_{side}$ and *wsize* is the window size of the interpolation function assuming guard rings are added as discussed in Section 3.4.1. It returns 2D arrays in the x, y-directions

which have variations along the x, y-axis respectively. Using this, the diagonal index array is created by adding the middle index of the isolatitude array to the difference of the (x, y) grid arrays. The resultant diagonal index array in *HEALPix* grid geometry is shown in Fig. 5.4 and is used to find $(\sin\delta_P, \cos\delta_P)$ from the corresponding sin_{ring}, cos_{ring} values. Thus, we have $(\sin\delta_P, \cos\delta_P)$ in the 2D $HEALPix$ grid format.

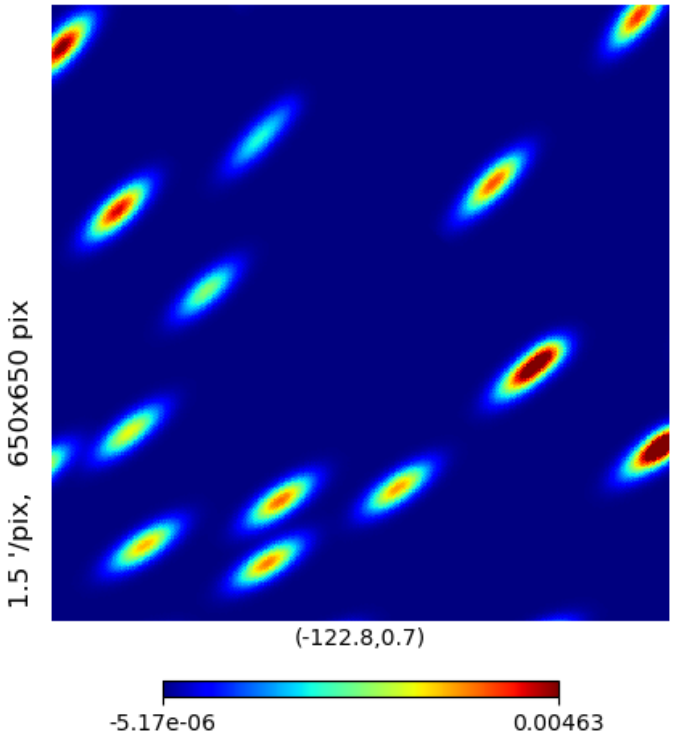
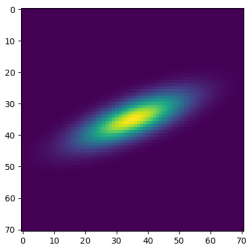


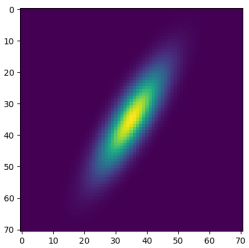
Figure 5.5: Elliptical beam interpolation map.

We developed an algorithm which produces sky-map by considering an elliptical beam. However, instead of displaying the full-sky map to demonstrate the elliptical interpolation function we display only a small portion of the sky corresponding to the image in Fig. 5.5. Each of the circularly modelled beams in the input sky-map now exists as an elliptical beam at a given orientation. The slow variation of position angle across this field follows the telescope scan

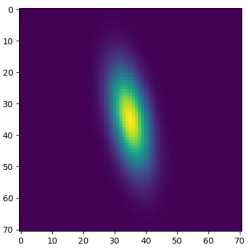
pattern, since the interpolation pattern has a fixed orientation relative to the scan direction. The input map with a single point source was also tested for varying orientation angle. The resultant map is shown in Fig. 5.6.



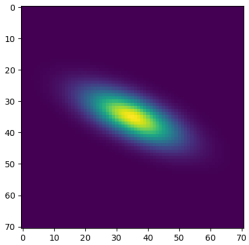
(a) $\Omega = 0^\circ$



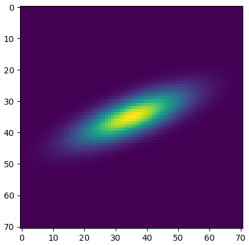
(b) $\Omega = 45^\circ$



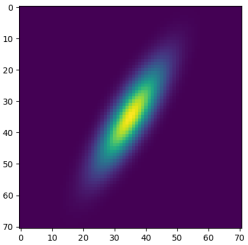
(c) $\Omega = 90^\circ$



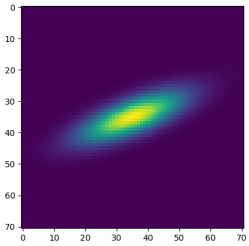
(d) $\Omega = 135^\circ$



(e) $\Omega = 180^\circ$



(f) $\Omega = 225^\circ$



(g) $\Omega = 360^\circ$

Figure 5.6: Elliptical beam interpolation map for a single point source with varying orientation angle.

This new elliptical beam interpolation method does shows promising results. However, more tests needs to be done for verifying its efficiency and improving the computational speed. This would be dealt with as a part of future work.

5.4 Conclusion

We have outlined an approach to compensate for elliptical beams by using an elliptical interpolation function to circularize the effective resolution, and have demonstrated a preliminary version of the code. The interpolation function needs to be broad along the minor axis of the beam and narrow along the major axis. The net result of this combination gives a nearly circular beam. It would be exactly circular if the telescope beam and the interpolation function are both elliptical Gaussian. The correction term is given by equation 5.13 which depends upon the properties of the telescope. The elliptical interpolation function is given by equation 5.14 and the standard deviation which define the size and shape of the beam is given in equation 5.15. The minor axis of the interpolation function needs to be small and finite while the major axis of the interpolation function should be greater than the correction term defined, so that

$$b^2 + c^2 = a^2 + d^2 = f^2, \tag{5.23}$$

where (b, a) are the major axis and minor axis of the ellipse and (d, c) are the major axis and minor axis of the interpolation function, and f is the FWHM of the effective beam. Thus, the effective beam is now circular after elliptical interpolation. Our approach is immune to noise amplification since it only reduces the resolution, mainly along the minor axis of the beam. This means that despite the negative impact on the error propagation, interpolation-circularised maps may be advantageous even for CMB analysis, especially as the technique will be applied only to the low frequency maps whose purpose is to model the foreground contamination in the CMB-dominated channels.

Chapter 6

Conclusion and Future Scope

In this dissertation, a detailed description of CMB map-making methods has been made. Map-making is a three-stage process consisting of flagging and calibration, destriping and map-making itself. The optimal map-making solutions obtained by minimising linear algebraic equations are computationally expensive for large TOD as expected in the CMB community. Hence, non-optimal map-making methods were developed called destriping itself. For each of these methods, most appropriate for circular beam, a pixelization scheme needs to be considered for making the maps. So in practice, pixelization schemes are used for the distribution of discretized map elements on the pixelated sky. Hence, the sky needs to be approximated, as it is a continuous function. The simplest and commonly used binning-based map-making algorithm, with only one non-zero entry in the pointing matrix, ignores the fact that the sky is continuous. Choosing pixels to be smaller than the telescope beam increases the noise per pixel in such maps. Therefore, interpolation-based map-making methods were suggested, which take into account the continuous nature of the sky. Interpolation schemes used to date interpolate data onto a rectangular grid which therefore has variable pixel size and inefficient data storage.

In this dissertation, we developed a simple but slow implementation of the

interpolation map-making for the *HEALPix* pixelization scheme that we call the standard interpolation method. In the interpolation-based map-making, a weighted data intensity is assigned to all the pixels within a radius centred at the data sample. However, the maps thus created were computationally expensive compared to binning maps. Attempts were made to improve the interpolation algorithm by suggesting polynomial curve-fitting methods to replace the transcendental functions in the algorithm, but they never showed any promising results. Thereafter, in this dissertation, we developed a new interpolation onto the *HEALPix* grid also called grid interpolation method. The grid interpolation algorithm, developed on the ideal pixelization scheme *HEALPix* with a truncated Gaussian distribution as the weight function, assumes a circular beam profile. The interpolation step that we have developed can be incorporated into any map-making scheme, whether optimal or non-optimal. The algorithm was tested on *Planck* 857 GHz data with C-BASS pointing information. With the future of CMB projects aiming towards detecting feeble B-mode signals, it is necessary to address the telescope beam ellipticity that is a potential source for systematic errors. Chapter 5 of this dissertation discusses an elliptical interpolation function to approximately circularize an elliptical telescope beam.

This dissertation has discussed in detail the methods and procedures undertaken to develop the interpolation onto the *HEALPix* grid map-making. The standard interpolation with the *query_disc* implementation is considered to be the most precise method and is used as a benchmark comparison for the grid interpolation algorithm. The resultant all-sky maps from the algorithm and its comparison with the standard interpolation map were discussed in detail in Chapter 4. The all-sky maps produced for C-BASS pointing data showed that we are correctly dealing with the overall organization of the *HEALPix* grid. In the difference maps, there were no evident errors in the equatorial zone of the grid apart from slight fractional differences due to the different

truncation of the interpolation function in the two algorithms. There weren't any noticeable problems in the region corresponding to facet 6 in the all-sky map indicating that the mirror operation carried out was correct. However, from the fractional difference map, there were evident demarcation lines at the edges of the polar facets. The fractional difference maps of the interpolation grid also showed evident thin horizontal lines at the limiting declinations of C-BASS scans corresponding to 37 and 47 degrees. At a declination of -16° (C-BASS boundary), there is also an edge effect as the interpolation window is truncated to zero for the pixels beyond the edge. Due to time constraints, no further analysis or possible solutions were suggested. Unexpectedly, the computational time of the new interpolation was found to be slower than the standard interpolation method, but there is still considerable scope for optimising the grid interpolation algorithm. With respect to the elliptical interpolation function algorithm, it was tested for a short stretch of C-BASS observations which focus on a part of the sky covered by only one set of scans. The method developed is independent of the beam shape although the idea is to tune the method parameters for a particular input beam. For an input data consisting of circular point sources at random sky locations, the interpolation function wides it in the minor axis instead of shrinking it in the major axis. Thus, the elliptical interpolation function can be used to approximately circularize an elliptical telescope beam.

There were few issues identified during the progress of the work with respect to the algorithms which should be addressed in the future scope for this dissertation. The standard interpolation algorithm is expensive when compared to the binning-based method. In Chapter 2, possible ways to speed it up by replacing the transcendental functions from the algorithm were suggested. However, the suggested methods showed no promising results. Hence, we need to re-evaluate the algorithm for optimization. We should also test whether the method works noticeably faster in one or other of the two *HEALPIx* order-

ings, i.e. *RING* and *NEST*. The binning maps are truncated to zero where there is no data while the smoothed binning maps are always non-zero. In the comparison of these maps, especially at the edge of the C-BASS boundary, pixels are forced to truncate to zero. This asymmetric distribution of pixels at -16° needs to be analysed further.

The grid interpolation map-making is not yet complete in terms of an algorithm. The interpolation window currently in use is a square window. For future work, it would be better to only use pixels within the circle enclosed to give a more consistent interpolation across the sky. This could be done efficiently by rejecting the pixels from the set selected in the square patch. The two parameters, radius of the interpolation window (r) and the FWHM of the Gaussian distribution are variable parameters which needs to be adjusted as per requirements. Choosing a large interpolation window can affect the computational time. The FWHM of the interpolation function controls the trade-off between noise suppression (better for larger FWHM) and the degradation of the resolution (better for small FWHM). In the tests conducted, the radius (r) and FWHM are considered to be 1° each. However, for future analyses variations of these parameters needs to be tested and analysed. In the all-sky maps discussed in Chapter 4, horizontal lines were present at the declination limits of the C-BASS scans. We expect that it is due to the extremely steep gradient in sample density at the scan limits, with the samples piling up at the declination limits so that pixels near the limit receive nearly all their weight from observations along the limit line. In principle it could be possible to correct for this asymmetric sampling but at considerable cost in computational time. In the fractional difference maps of grid interpolation, the demarcation lines at the edges of the polar facets were evident. This may be due to the usage of differential form of the projection equations, when the actual equations have a step change at the grid boundaries, caused by the *CEIL* function. One possible solution would be to simply create grids of the pixel centre (θ, α) and to use those

directly for the offset calculations. There is also a problem in the interpolation at the polar region gores in the grid i.e. at the poles and the point at DEC = 41 degrees. As discussed in Section 3.4.4 of Chapter 3, a better way to avoid the double counting at these places is required to solve the gore problem. Though analysis were done for checking the precision and efficiency of the algorithm, they are not sufficient. With the complete algorithm to be developed, more accuracy checks are necessary for understanding the efficiency of the algorithm. Alongside this, we also should be analysing the algorithm for optimization, with an intent to decrease the computational time. There is significant scope for optimisation, though, for instance the construction of the window arrays $(\Delta x, \Delta y)$ could be done outside the loop which should save quite a lot of time. With respect to the elliptical interpolation function, more tests need to be done for verifying its efficiency and accuracy. It also needs to be tested on data with a simulated elliptical beam. The current algorithm developed should also be optimized to reduce its computational time.

Appendix A

Small angle approximation distance formula

The spherical cosine formula gives,

$$\cos u \quad = \quad \cos v \cos w + \sin v \sin w \cos A. \tag{A.1}$$

In our frame of reference where we use colatitude (θ) instead of declination. The two data points under consideration are($\tilde{\theta} - \frac{\Delta\theta}{2}$), ($\tilde{\theta} + \frac{\Delta\theta}{2}$) and $\Delta\alpha$ is the difference in the right-ascension. The distance between the two data points under consideration is given by ρ.

$$\cos\rho \quad = \quad \cos\left(\tilde{\theta} + \frac{\Delta\theta}{2}\right)\cos\left(\tilde{\theta} - \frac{\Delta\theta}{2}\right) + \sin\left(\tilde{\theta} + \frac{\Delta\theta}{2}\right)\sin\left(\tilde{\theta} - \frac{\Delta\theta}{2}\right)\cos\Delta\alpha. \tag{A.2}$$

Expanding for cosine compound angle formula,

$$\cos(A + B) = \cos A \cos B - \sin A \sin B. \tag{A.3}$$

Substituting equation A.3 in A.2,

$$\begin{aligned}
\cos\rho &= \Big(\cos\tilde{\theta}\cos\frac{\Delta\theta}{2} - \sin\tilde{\theta}\sin\frac{\Delta\theta}{2}\Big)\Big(\cos\tilde{\theta}\cos\frac{\Delta\theta}{2} + \sin\tilde{\theta}\sin\frac{\Delta\theta}{2}\Big) \\
&+ \Big(\sin\tilde{\theta}\cos\frac{\Delta\theta}{2} + \cos\tilde{\theta}\sin\frac{\Delta\theta}{2}\Big)\Big(\sin\tilde{\theta}\cos\frac{\Delta\theta}{2} - \cos\tilde{\theta}\sin\frac{\Delta\theta}{2}\Big) \\
&\quad \Big(\cos\Delta\alpha\Big), \qquad \text{(A.4)} \\
&= \Big(\cos^2\tilde{\theta}\cos^2\frac{\Delta\theta}{2} + \cancel{\cos\tilde{\theta}\sin\tilde{\theta}\cos\frac{\Delta\theta}{2}\sin\frac{\Delta\theta}{2}} - \cancel{\cos\tilde{\theta}\sin\tilde{\theta}\cos\frac{\Delta\theta}{2}\sin\frac{\Delta\theta}{2}} \\
&- \sin^2\tilde{\theta}\sin^2\frac{\Delta\theta}{2}\Big) + \Big(\sin^2\tilde{\theta}\cos^2\frac{\Delta\theta}{2} - \cancel{\cos\tilde{\theta}\sin\tilde{\theta}\cos\frac{\Delta\theta}{2}\sin\frac{\Delta\theta}{2}} \\
&+ \cancel{\cos\tilde{\theta}\sin\tilde{\theta}\cos\frac{\Delta\theta}{2}\sin\frac{\Delta\theta}{2}} - \cos^2\tilde{\theta}\sin^2\frac{\Delta\theta}{2}\Big)\Big(\cos\Delta\alpha\Big). \qquad \text{(A.5)}
\end{aligned}$$

Using small angle approximation, where $\sin\Delta\theta \approx \Delta\theta$ and $\cos\Delta\alpha = 1 - \frac{\Delta\alpha^2}{2}$.

$$\begin{aligned}
1 - \frac{\rho^2}{2} &= \Big(\cos^2\tilde{\theta}\cos^2\frac{\Delta\theta}{2} - \sin^2\tilde{\theta}\sin^2\frac{\Delta\theta}{2}\Big) + \Big(\sin^2\tilde{\theta}\cos^2\frac{\Delta\theta}{2} - \\
&\quad \cos^2\tilde{\theta}\sin^2\frac{\Delta\theta}{2}\Big)\Big(1 - \frac{\Delta\alpha^2}{2}\Big), \qquad \text{(A.6)} \\
&= \Big(\cos^2\tilde{\theta}\cos^2\frac{\Delta\theta}{2} - \sin^2\tilde{\theta}\sin^2\frac{\Delta\theta}{2} + \sin^2\tilde{\theta}\cos^2\frac{\Delta\theta}{2} - \\
&\quad \cos^2\tilde{\theta}\sin^2\frac{\Delta\theta}{2}\Big) - \frac{\Delta\alpha^2}{2}\Big(\sin^2\tilde{\theta}\cos^2\frac{\Delta\theta}{2} - \cos^2\tilde{\theta}\sin^2\frac{\Delta\theta}{2}\Big), \qquad \text{(A.7)} \\
&= \Big(\cos^2\frac{\Delta\theta}{2} - \sin^2\frac{\Delta\theta}{2}\Big) - \frac{\Delta\alpha^2}{2}\Big(\sin^2\tilde{\theta}\cos^2\frac{\Delta\theta}{2} - \\
&\quad \cos^2\tilde{\theta}\sin^2\frac{\Delta\theta}{2}\Big), \qquad \text{(A.8)} \\
&= \Big(1 - \frac{(\Delta\theta/2)^2}{2}\Big)^2 - \Big(\Delta\theta/2\Big)^2 - \frac{\Delta\alpha^2}{2}\Bigg[\sin^2\tilde{\theta}\Big(1 - \frac{(\Delta\theta/2)^2}{2}\Big)^2 + \\
&\quad \cos^2\tilde{\theta}(\Delta\theta/2)^2\Bigg]. \qquad \text{(A.9)}
\end{aligned}$$

Approximating to fourth order,

$$1 - \frac{\rho^2}{2} \approx 1 - \frac{\Delta\theta^2}{2} + \frac{\Delta\theta^4}{64} - \frac{\Delta\alpha^2}{2}\left[\sin^2\tilde{\theta}\right], \tag{A.10}$$

$$\frac{\rho^2}{2} \approx \frac{\Delta\theta^2}{2} - \cancelto{0}{\frac{\Delta\theta^4}{64}} + \frac{\Delta\alpha^2}{2}\sin^2\tilde{\theta}, \tag{A.11}$$

$$\rho \approx \sqrt{\Delta\theta^2 + \Delta\alpha^2 \sin^2\tilde{\theta}}. \tag{A.12}$$

Equation A.12 gives the small angle distance formula for two points on the surface of a sphere.

Appendix B

Approximate formula for position angle

For position angle χ we have:

$$\tan\chi = \frac{\sin\Delta\alpha}{\cos\delta_1\tan\delta_2 - \sin\delta_1\cos\Delta\alpha}. \tag{B.1}$$

Writing $\delta_1 = \delta$ and $\delta_2 = \delta + \Delta$, where Δ is assumed small.

$$\text{denominator} = \cos\delta\frac{\sin\delta\cos\Delta + \cos\delta\sin\Delta}{\cos\delta\cos\Delta - \sin\delta\sin\Delta} - \sin\delta\cos\Delta\alpha, \tag{B.2}$$

$$\approx \cos\delta\frac{(1-\Delta^2/2)\sin\delta + \Delta\cos\delta}{(1-\Delta^2/2)\cos\delta - \Delta\sin\delta} - \left(1 - \frac{(\Delta\alpha)^2}{2}\right)\sin\delta, \tag{B.3}$$

$$= \sin\delta\left\{\frac{1-\Delta^2/2 + \Delta\cot\delta}{1-\Delta^2/2 - \Delta\tan\delta} - 1 + \frac{(\Delta\alpha)^2}{2}\right\}. \tag{B.4}$$

Note that all angles are in radians. Expanding the denominator of the first fraction as a binomial series.

$$\begin{aligned}(1-\Delta^2/2-\Delta\tan\delta)^{-1} &= 1-(-\Delta\tan\delta-\Delta^2/2)+ \\ &\quad \frac{(-1)(-2)}{2!}(\Delta\tan\delta-\Delta^2/2)^2+O(\Delta^3), \quad \text{(B.5)} \\ &= 1+\Delta\tan\delta+\Delta^2/2+\Delta^2\tan^2\delta+O(\Delta^3). \quad \text{(B.6)}\end{aligned}$$

Hence, to second order in the small angles Δ and $\Delta\alpha$,

$$\tan\chi \approx \frac{\Delta\alpha}{\sin\delta\{\Delta(\cot\delta+\tan\delta)+\Delta^2(\cot\delta\tan\delta+\tan^2\delta)+(\Delta\alpha)^2/2\}} \quad \text{(B.7)}$$

Using $\cot\delta+\tan\delta = 1/(\cos\delta\sin\delta)$ and $1+\tan^2\delta = 1/\cos^2\delta$ in equation B.7 gives

$$\begin{aligned}\tan\chi &\approx \frac{\Delta\alpha\cos\delta}{\Delta+\sin\delta(\Delta^2+(\Delta\alpha\cos\delta)^2/2)/\cos\delta}, \quad \text{(B.8)} \\ &= \frac{\Delta\alpha^*}{\Delta+\sin\delta(\Delta^2+\Delta\alpha^{*2}/2)/\cos\delta}. \quad \text{(B.9)}\end{aligned}$$

Here $\Delta\alpha^* = \Delta\alpha\cos\delta$ is the angular length corresponding to a change in RA of $\Delta\alpha$ and declination (δ).

Appendix C

Code developed for standard interpolation map-making

In this Section, the program developed for implementing the standard interpolation map-making algorithm is given.

```
#importing libraries
from astropy.io import fits
import matplotlib.pyplot as plt
import os
import healpy as hp
import numpy as np
from tqdm import tqdm
import math
import time
from matplotlib import cm

#The function defined for Gaussian distribution
def Gaussian_weight(offset):
  weight = np.exp(-( offset**2 /  two_sigma2 ) )
return weight

# Reading the input Planck fits file
dir_path = '/local/scratch/selvaraj/thesis/scripts'
file = open(dir_path+"/filelist.txt", "r")
file_list = (file.read().splitlines())
file_list = file_list[0]
file.close()
data_map = hp.read_map(file_list)
Nside = fits.open(file_list, memmap=False)[1].header['NSIDE']
pixels = 12*Nside**2 # defining for the total number of pixels
```

```
26
27 # Reading C-BASS pointing fits files
28 dir_path = os.getcwd()
29 file = open("file_list.txt", "r")
30 file_list = file.read().splitlines()
31 file.close()
32
33 # Parameters for query-disc
34 radius_degrees = 1
35 fwhm_degrees =  radius_degrees
36 radmax = np.radians(radius_degrees) #in degree  convert it to
    radians
37 fwhm = np.radians(fwhm_degrees)
38 sigma = (fwhm/np.sqrt(8*math.log(2)))
39 two_sigma2 = 2*sigma**2
40
41 # Defining the weighted intensity and weight map
42 m = np.zeros(pixels)
43 weight= np.zeros(pixels)
44
45
46 for file in tqdm(file_list):
47
48   infile = fits.open(file)[1]
49   RA = infile.data['RA']
50   DEC = infile.data['DEC']
51   colatitude =  ((np.pi/2) - DEC) #co-latitude
52   phi = RA
53
54   pixint = hp.ang2pix(Nside, colatitude, phi, nest = False,
     lonlat = False)
55   I = data_map[pixint]
56   veclist = hp.ang2vec(colatitude,phi)
57
58   nlength = len(I)
59
60   for ipx in range(nlength):
61     # get list of nearby healpy pixels to the position at ipx
62     ipx2 = hp.query_disc(Nside,veclist[ipx],radmax)
63
64     # converts al those pixels to vectors.
65     vec2list = hp.pix2vec(Nside, ipx2, nest = False)
66
67     # take dot product between the two position vectors
68     dot_product = np.matmul(veclist[ipx], vec2list)
69
70     # to get angular offset.
71     dot_product[np.where(dot_product > 1)] = 1
72     dot_product[np.where(dot_product <-1)] = -1
```

```
    offset = np.arccos(dot_product)

    # to calculate Gaussian weight  as a function of angular
    offset
    weight_new = Gaussian_weight(offset)

    m[ipx2]+= I[ipx]*weight_new # weighted intensity map
    weight[ipx2] += weight_new # weight map

# identifying the bad pixels
bad_values = np.where(weight == 0)
m[bad_values]=hp.pixelfunc.UNSEEN
weight[bad_values] = 1.
weighted_map = m/weight # standard interpolation weighted map

# display
hp.mollview(weighted_map, nest=False, title="Interpolation based
    map",min = -0.01, max = 0.5, cmap = cm.jet)
```

Listing C.1: Python code for standard interpolation map-making

Appendix D

Code developed for interpolation onto the *HEALPix* grid map-making

In this Section, the program developed for implementing interpolation onto the *HEALPix* grid map-making algorithm is given.

```
#importing libraries
from astropy.io import fits
import matplotlib
import matplotlib.pyplot as plt
from matplotlib import cm
import os
import healpy as hp
import numpy as np
from tqdm import tqdm
import math

#The function defined for Gaussian distribution
def Gaussian_weight(offset):
  weight = np.exp(-( offset**2 /  two_sigma2 ) )
return weight

# Reading the input Planck fits file
dir_path = '/local/scratch/selvaraj/thesis/scripts'
file = open(dir_path+"/filelist.txt", "r")
file_list = (file.read().splitlines())
file_list = file_list[0]
file.close()
data_map= hp.read_map(file_list)
Nside = fits.open(file_list, memmap=False)[1].header['NSIDE'] #
   reading the Nside from the input file.
```

```
25
26 # Reading C-BASS pointing fits files
27 dir_path = os.getcwd()
28 file = open("file_list.txt", "r")
29 file_list = (file.read().splitlines())
30 file.close()
31
32 pixels = 12*Nside**2 # defining for the total number of pixels
33 nsize = 5*Nside # defining the size of the grid
34
35 # Getting the coordinates of the pixel centres and defining
      isolatitude rings.
36 theta_pixel, phi_pixel = hp.pix2ang(Nside,np.arange(pixels),
      lonlat=False) # returns the healpixel coordinates (latitude
      and longitutde) in radians.
37 isolatitude_rings = np.unique(theta_pixel)
38
39 # Parameters for the interpolation function
40 radius_degrees = 1
41 fwhm_degrees =  radius_degrees
42 radmax = np.radians(radius_degrees)
43 fwhm = np.radians(fwhm_degrees)
44 sigma = (fwhm/np.sqrt(8*math.log(2))) # standard deviation for
      the Gaussian distribution
45 two_sigma2 = 2*sigma**2
46 pixelsize = np.sqrt(4*np.pi/pixels)
47 dpixel = np.int(np.ceil(radmax / pixelsize)) # interpolation
      window size
48 irange = np.arange(-dpixel, dpixel+1)
49 wsize = int(dpixel*2+1)
50
51 # Defining constants
52 cos45 = math.cos(math.radians(45))
53 sin45 = math.sin(math.radians(45))
54 cnst = (3*np.pi/8.)
55 k_1 = (2*np.sqrt(2)*Nside)/np.pi
56 sin_inv = np.arcsin(2/3.)
57
58 # Defining the matrices for grid interpolation
59 dx = np.zeros((wsize, wsize))
60 dy = np.zeros((wsize, wsize))
61 m = np.zeros((nsize+wsize, nsize+wsize)) # intensity map array
62 weight = np.zeros((nsize+wsize, nsize+wsize)) # weight array
63
64 # Looping over the entire input file list
65 for file in tqdm(file_list):
66   polar = 1 # binary condition to evaluate polar region
67   equator = 1 # binary condition to evaluate equatorial region
68   infile = fits.open(file)[1]
```

```
69  # reading declination and right-ascension from the C-BASS
     pointing files
70  RA = infile.data['RA']
71  DEC = infile.data['DEC']
72
73  colatitude =  ((np.pi/2) - DEC) #co-latitude
74  phi = RA
75
76  pixint = hp.ang2pix(Nside, colatitude, phi, nest = False,
     lonlat = False)
77  I = data_map[pixint]
78
79  nlength = len(RA)
80  colatitude_app = np.zeros(nlength)
81
82  # approximation of colatitude values to the nearest iso-
     latitude rings.
83  gap = abs(isolatitude_rings[0] - isolatitude_rings[1])
84  for k in range(len(isolatitude_rings)):
85    colatitude_app[(colatitude> (isolatitude_rings[k]-gap)) * (
     colatitude< (isolatitude_rings[k]+gap))] = isolatitude_rings[k
     ]
86
87  sin_colat = np.sin(colatitude_app)
88  sin2_colat = sin_colat**2
89
90  # Evaluating for polar region
91  if polar == 1:
92    pix_polar = np.where( DEC> sin_inv)[0] # condition for
     identifying pixels in polar region.
93
94    Sigma = np.sqrt(3*(1-np.abs(np.sin(DEC))))
95    phi_c = -np.pi + (2*np.floor((phi+np.pi)*4/(2*np.pi))+1)*np.
     pi/4.
96    # projection in cartesian coordinates
97    y = (np.pi/4)*(2-Sigma) *k_1
98    x = -(phi_c + (phi - phi_c)*Sigma - 1.25*np.pi) *k_1
99    # rotation by 45 degrees in anti-clockwise direction
100   jcoord = (x*cos45 - y*sin45)
101   icoord = (x*sin45 + y*cos45)
102
103   for idx in pix_polar:
104     # coordinates of the pixel centre.
105     ic = round(icoord[idx])
106     jc = round(jcoord[idx])
107     # creating a square window centered at the pixel centre.
108     ii = (irange + ic).astype(int)
109     jj = (irange + jc).astype(int)
110     ifloat = ii + (icoord[idx]-ic)
```

```
111       jfloat = jj + (jcoord[idx]-jc)
112
113       # rotation of the interpolatio window by 45 degree in
    clockwise direction.
114       dx1 = cos45*jfloat
115       dx2 = sin45*ifloat
116       dy1 = -sin45*jfloat
117       dy2 = cos45*ifloat
118
119       # creating the differential matrix
120       for j in range(wsize):
121         dx[:, j] = dx1 + dx2[j]
122         dy[:, j] = dy1 + dy2[j]
123
124       dx = dx-x[idx]
125       dy = dy-y[idx]
126       dphi = dx/(Sigma[idx]*k_1)
127       dtheta = dy*Sigma[idx]/(cnst*sin_colat[idx]*k_1)
128       rho = np.sqrt((dtheta)**2 + (dphi**2 * sin2_colat[idx])) #
    small-angle approximated distance
129
130       weight_new = Gaussian_weight(rho) # calculating Gaussian
    weight
131       # indices of the corresponding interpolation patch on the
    grid.
132       p = ii+dpixel
133       q = jj+dpixel
134
135       m[p[0]: p[-1]+1, q[0]: q[-1]+1]+= I[idx]*weight_new #
    weighted intensity map
136       weight[p[0]: p[-1]+1, q[0]: q[-1]+1] += weight_new # weight
     array
137     polar = 0
138   # Evaluating for equatorial region
139   if equator == 1:
140     pix_equator = np.where( DEC<= sin_inv)[0] # condition for
    identifying pixels in the equatorial region.
141     # projection in cartesian coordinates.
142     y = (cnst*np.sin(DEC))*k_1
143     x = -(phi-1.25*np.pi)*k_1
144     # rotation by 45 degrees in anti-clockwise direction.
145     jcoord = (x*cos45 - y*sin45)
146     icoord = (x*sin45 + y*cos45)
147
148     for idx in pix_equator:
149       # coordinates of the pixel centre.
150       ic = round(icoord[idx])
151       jc = round(jcoord[idx])
152       # creating a sqaure window centered at the pixel centre.
```

```
153       ii = (irange + ic).astype(int)
154       jj = (irange + jc).astype(int)
155       ifloat = ii + (icoord[idx]-ic)
156       jfloat = jj + (jcoord[idx]-jc)
157
158       # rotation of the interpolation window by 45 degrees in
    clockwise direction
159       dx1 = cos45*jfloat
160       dx2 = sin45*ifloat
161       dy1 = -sin45*jfloat
162       dy2 = cos45*ifloat
163       # creating the differential matrix
164       for j in range(wsize):
165         dx[:, j] = dx1 + dx2[j]
166         dy[:, j] = dy1 + dy2[j]
167
168       dx = dx-x[idx]
169       dy = dy-y[idx]
170       dphi = dx/(k_1)
171       dtheta = dy/(cnst*sin_colat[idx]*k_1)
172       rho = np.sqrt((dtheta)**2 + (dphi**2 * sin2_colat[idx])) #
    small-angle approximated distance
173
174       weight_new = Gaussian_weight(rho) # calculating Gaussian
    weight.
175       # indices of the corresponding interpolation patch on the
    grid.
176       p = ii+dpixel
177       q = jj+dpixel
178       m[p[0]: p[-1]+1, q[0]: q[-1]+1]+= I[idx]*weight_new #
    weighted intensity map
179       weight[p[0]: p[-1]+1, q[0]: q[-1]+1]+= weight_new # weight
    map
180     equator = 0
181   print ('Completed file', file)
182
183 # Stitching the edges of the north polar facets.
184 for i in range(4):
185   te_i_start = i*Nside # starting index of top edge strip in x-
     axis
186   te_i_stop = (i+1)*Nside + dpixel # ending index of top edge
     strip in x-axis
187   te_j_start = (i+2)*Nside # starting index of top edge strip in
     y-axis
188   te_j_stop = te_j_start + wsize # ending index of top edge strip
      in y-axis
189   if ( i == 3):
190     se_i_start = (i-3)*Nside # starting index of side edge strip
     in x-axis
```

```
191     se_i_stop = se_i_start + wsize  # ending index of top edge
        strip in x-axis
192     se_j_start = (i-2)*Nside + dpixel # starting index of top
        edge strip in y-axis
193     se_j_stop = (i-1)*Nside + wsize-1 # ending index of top edge
        strip in y-axis
194   else:
195
196     se_i_start = (i+1)*Nside
197     se_i_stop = se_i_start + wsize
198     se_j_start = (i+2)*Nside + dpixel
199     se_j_stop = (i+3)*Nside + wsize-1
200
201   a = m[se_i_start: se_i_stop, se_j_start: se_j_stop]
202   m_average = (m[te_i_start: te_i_stop, te_j_start: te_j_stop] +
       np.rot90(a,3))/2. # average of top-edge and side-edge for
       weighted intensity map
203
204   b = weight[se_i_start: se_i_stop, se_j_start: se_j_stop]
205   w_average = (weight[te_i_start: te_i_stop, te_j_start:
       te_j_stop]+ np.rot90(b,3))/2. # average of top-edge and side-
       edge for weight map
206
207   m[te_i_start: te_i_stop, te_j_start: te_j_stop] = m_average
208   weight[te_i_start: te_i_stop, te_j_start: te_j_stop] =
       w_average
209   m[se_i_start: se_i_stop, se_j_start: se_j_stop] = np.rot90(
       m_average)
210   weight[se_i_start: se_i_stop, se_j_start: se_j_stop] = np.rot90
       (w_average)
211
212 # Removing the guard rings
213 m = m[dpixel: nsize+dpixel, dpixel:nsize+dpixel]
214 weight = weight[dpixel: nsize+dpixel, dpixel:nsize+dpixel]
215
216 # Facet 6
217 m_average = (m[4*Nside:nsize, 4*Nside:nsize] + m[0:Nside, 0:Nside
      ])/2.
218 weight_average = (weight[4*Nside:nsize, 4*Nside:nsize] + weight
      [0:Nside, 0:Nside])/2.
219 m[4*Nside:nsize, 4*Nside:nsize] = m_average
220 m[0:Nside, 0:Nside] = m_average
221 weight[4*Nside:nsize, 4*Nside:nsize] = weight_average
222 weight[0:Nside, 0:Nside] = weight_average
223
224 # identifying the bad pixels
225 bad_values = np.where(weight == 0)
226 m[bad_values] = 0; weight[bad_values] = 1
227 weighted_map = m/weight
```

```
228
229 # Removing the extra patches near polar facets.
230 for i in range(3):
231   weighted_map[(i+1)*Nside + Nside: nsize, i*Nside:(i+1)*Nside] =
      0
232
233 # setting bad pixels to blank for display purpose
234 bad_values = np.where(weighted_map == 0)
235 weighted_map[bad_values] = np.NaN
236 cmap = matplotlib.cm.jet
237 cmap.set_bad('grey',1.)
238
239 # Displaying the 2D interpolation grid map
240 fig, ax = plt.subplots()
241 ax.set_xlim(0, nsize) # Set axis ranges; by default this will put
      major ticks every nsize.
242 ax.set_ylim(0, nsize)
243 ax.xaxis.set_major_locator(MultipleLocator(512)) # Change major
    ticks to show every 512.
244 ax.yaxis.set_major_locator(MultipleLocator(512))
245 ax.grid(which='major', color='#CCCCCC', linestyle='--') # Turn
    grid on for both major and minor ticks and style minor
    slightly differently.
246 plt.imshow(weighted_map, vmin = np.nanmin(weighted_map), vmax =
    np.nanmax(weighted_map)*0.1, cmap=cm.jet, origin = 'lower')
247
248 # saving the weighted map as a fits file
249 hdu = fits.PrimaryHDU(data=weighted_map)
250 hdu.writeto('/local/scratch/selvaraj/data/hpx_grid.fits')
```

Listing D.1: Python code for interpolation in HEALPix grid

Appendix E

Code developed for elliptical beam interpolation

In this Section, the program developed for implementing elliptical beam interpolation algorithm is given.

```
# importing libraries
from matplotlib.ticker import (AutoMinorLocator, MultipleLocator)
from astropy.io import fits
import matplotlib
import matplotlib.pyplot as plt
from matplotlib import cm
from tqdm import tqdm
import healpy as hp
import numpy as np
import math
import os

# function for elliptical beam interpolation function
def Ellipical_weight(x, y):
  weight = np.exp(-0.5*(x**2/sigma_i2 + y**2/sigma_j2 ))
return weight

# reading input point source fits file.
dir_path = os.getcwd()
file = 'rcat_02000_2020-09-01.fits'
hp.mollview(data_map, nest=False, title="Mollview image RING",min
    = np.min(data_map), max = np.max(data_map)*0.01, cmap = cm.
   jet) # displaying the point source map.
data_map= hp.read_map(file)
Nside = fits.open(file, memmap=False)[1].header['NSIDE']
```

```
24
25 # reading C-BASS poiting fits files.
26 dir_path = os.getcwd()
27 file = open("file_list.txt", "r")
28 file_list = (file.read().splitlines())
29 file.close()
30
31 pixels = 12*Nside**2 # defining the total number of pixels.
32 nsize = 5*Nside # defining the size of the grid.
33
34 # getting the coordinates of the pixel centres and defining
      isolatitude rings.
35 theta_pixel, phi_pixel = hp.pix2ang(Nside,np.arange(pixels),
      lonlat=False) # returns the healpixel coordinates (latitude
      and longitutde) in radians.
36 colatitude_rings = np.unique(theta_pixel)
37 sine_ring = np.sin(colatitude_rings)
38 cos_ring = np.cos(colatitude_rings)
39
40 # Parameters for query_disc
41 radius_degrees = 2
42 fwhm_degrees =  1
43 radmax = np.radians(radius_degrees) #in degree  convert it to
      radians
44 fwhm = np.radians(fwhm_degrees)
45 k_2 = np.sqrt(8*math.log(2))
46 sigma = (fwhm/k_2)
47 two_sigma2 = 2*sigma**2
48
49 #Parameters for elliptical beam interpolation function
50 alpha = 0.5
51 beta = 1.5
52 sigma_i2 = (np.deg2rad(beta)/k_2)**2
53 sigma_j2 = (np.deg2rad(alpha)/k_2)**2
54 delta = np.pi/2.
55 pixelsize = np.sqrt(4*np.pi/pixels)
56 dpixel = np.int(np.ceil(radmax / pixelsize))
57 irange = np.arange(-dpixel, dpixel+1)
58
59 # defining constants
60 wsize = int(dpixel*2+1)
61 cos45 = math.cos(math.radians(45))
62 sin45 = math.sin(math.radians(45))
63 cnst = (3*np.pi/8.)
64 k_1 = (2*np.sqrt(2)*Nside)/np.pi
65 sin_inv = np.arcsin(2/3.)
66
67 # Defining the matrices for grid interpolation
68 dx = np.zeros((wsize, wsize))
```

```
dy = np.zeros((wsize, wsize))
m = np.zeros((nsize+wsize, nsize+wsize))
weight = np.zeros((nsize+wsize, nsize+wsize))

# defining diagonal index matrix.
xi, yj = np.meshgrid(np.arange(nsize+wsize), np.arange(nsize+
    wsize))
diagonal = xi - yj + int(np.floor((4*Nside - 1)/2.))
diagonal[diagonal < 0] = 0
diagonal[diagonal >= 4*Nside-1] = 0

sin_thetaP = np.zeros((nsize+wsize, nsize+wsize))
cos_thetaP = np.zeros((nsize+wsize, nsize+wsize))
for i in range(nsize+wsize):
  for j in range(nsize+wsize):
    index = diagonal[i,j]
    sin_thetaP[i,j] = sine_ring[index]
    cos_thetaP[i,j] = cos_ring[index]

test = 1 # binary condition for testing.
# Looping over the entire input file list.
for file in tqdm(file_list):
  polar = 1 # binary condition to evaluate for polar region.
  equator = 1 3 binary condition to evalute for equatorial region
    .
  infile = fits.open(file)[1]
  # reading declination and right-ascension from C-BASS pointing
    files.
  RA = infile.data['RA']
  DEC = infile.data['DEC']
  nlength = len(RA)

  if test ==1:
    lowdec = np.where( DEC<= 0.01)[0]  # 1 radian value.
    I0 = lowdec[0]
    I1 = I0 + 10
    I2 = I0 + 20
    RA = RA[I0:I2]
    DEC = DEC[I0:I2]
    I = np.zeros(20)
    colatitude =  ((np.pi/2) - DEC) #co-latitude
    phi = RA
  else:
    colatitude =  ((np.pi/2) - DEC)
    phi = RA
    pixint = hp.ang2pix(Nside, colatitude, phi, nest = False,
    lonlat = False)
    I = data_map[pixint]

```

```
114   sin_colat = np.sin(colatitude)
115   sin2_colat = sin_colat**2
116
117   # difference in declination and right-ascension.
118   Ddec_pointing = np.append(DEC[:-1] - DEC[1:], DEC[-2])
119   Dphi_pointing = np.append(RA[:-1] - RA[1:], RA[-2])
120   # position angle calculation
121   PA_pointing = np.arctan(Dphi_pointing * np.cos(DEC)/
       Ddec_pointing)
122   PA_pointing[np.where(np.isnan(PA_pointing) == True)] =
       -1.5707964
123
124   # Evaluating for polar region.
125   if polar == 1:
126     pix_polar = np.where( DEC> sin_inv)[0]
127     Sigma = np.sqrt(3*(1-np.abs(np.sin(DEC))))
128     phi_c = -np.pi + (2*np.floor((phi+np.pi)*4/(2*np.pi))+1)*np.
       pi/4.
129     # projection in cartesian coordinates.
130     y = (np.pi/4)*(2-Sigma) *k_1
131     x = -(phi_c + (phi - phi_c)*Sigma - 1.25*np.pi) *k_1
132     # rotation by 45 degrees in anti-clockwise direction.
133     jcoord = (x*cos45 - y*sin45)
134     icoord = (x*sin45 + y*cos45)
135
136     for idx in pix_polar:
137       # coordinates of the pixel centre.
138       ic = round(icoord[idx])
139       jc = round(jcoord[idx])
140       I[idx] = 1
141       # creating a square window centered at the pixel centre.
142       ii = (irange + ic).astype(int)
143       jj = (irange + jc).astype(int)
144       ifloat = ii + (icoord[idx]-ic)
145       jfloat = jj + (jcoord[idx]-jc)
146
147       # rotation of the interpolatio window by 45 degree in
       clockwise direction.
148       dx1 = cos45*jfloat
149       dx2 = sin45*ifloat
150       dy1 = -sin45*jfloat
151       dy2 = cos45*ifloat
152
153       # creating the differential matrix
154       for j in range(wsize):
155         dx[:, j] = dx1 + dx2[j]
156         dy[:, j] = dy1 + dy2[j]
157       dx = dx-x[idx]
158       dy = dy-y[idx]
```

```
159
160         dphi = dx/(Sigma[idx]*k_1)
161         dtheta = dy*Sigma[idx]/(cnst*sin_colat[idx]*k_1)
162         rho = np.sqrt((dtheta)**2 + (dphi**2 * sin2_colat[idx])) #
    small-angle approximated distance
163
164         sinPhi = sin_thetaP[p[0]: p[-1]+1, q[0]: q[-1]+1]*(dphi/rho
    )
165         sinPhi[np.where(sinPhi > 1)] = 1
166         sinPhi[np.where(sinPhi <-1)] = -1
167         cosPhi = (np.cos(rho)*np.cos(colatitude[idx]) - cos_thetaP[
    p[0]: p[-1]+1, q[0]: q[-1]+1])\
168         /(rho*np.sin(colatitude[idx]))
169         Phi = np.arctan2( cosPhi, sinPhi) # position angle
    calculation
170
171         eps = 2*np.pi - PA_pointing[idx] - delta + Phi
172         xe = rho*np.cos(eps)
173         ye = rho*np.sin(eps)
174         weight_new = Ellipical_weight(xe, ye) # calculating the
    elliptical beam weight
175
176         # indices of the corresponding interpolation patch on the
    grid.
177         p = ii+dpixel
178         q = jj+dpixel
179         m[p[0]: p[-1]+1, q[0]: q[-1]+1]+= I[idx]*weight_new #
    weighted intensity map
180         weight[p[0]: p[-1]+1, q[0]: q[-1]+1] += weight_new # weight
     array
181       polar = 0
182
183   # Evaluating for equatorial region
184   if equator == 1:
185     pix_equator = np.where( DEC<= sin_inv)[0] # condition for
    identifying pixels in the equatorial region.
186     # projection in cartesian coordinates.
187     y = (cnst*np.sin(DEC))*k_1
188     x = -(phi-1.25*np.pi)*k_1
189     # rotation by 45 degrees in anti-clockwise direction.
190     jcoord = (x*cos45 - y*sin45)
191     icoord = (x*sin45 + y*cos45)
192
193     for idx in pix_equator:
194       # coordinates of the pixel centre.
195       ic = round(icoord[idx])
196       jc = round(jcoord[idx])
197       I[idx] = 1
198       # creating a sqaure window centered at the pixel centre.
```

```
199         ii = (irange + ic).astype(int)
200         jj = (irange + jc).astype(int)
201         ifloat = ii + (icoord[idx]-ic)
202         jfloat = jj + (jcoord[idx]-jc)
203 
204         # rotation of the interpolation window by 45 degrees in
    clockwise direction
205         dx1 = cos45*jfloat
206         dx2 = sin45*ifloat
207         dy1 = -sin45*jfloat
208         dy2 = cos45*ifloat
209         # creating the differential matrix
210         for j in range(wsize):
211           dx[:, j] = dx1 + dx2[j]
212           dy[:, j] = dy1 + dy2[j]
213 
214         dx = dx-x[idx]
215         dy = dy-y[idx]
216         dphi = dx/(Nside*k_1)
217         dtheta = dy/(cnst*sin_colat[idx]*Nside*k_1)
218         rho = np.sqrt((dtheta)**2 + (dphi**2 * sin2_colat[idx]))#
    small-angle approximated distance
219 
220         sinPhi = sin_thetaP[p[0]: p[-1]+1, q[0]: q[-1]+1]*(dphi/rho
    )
221         sinPhi[np.where(sinPhi > 1)] = 1
222         sinPhi[np.where(sinPhi <-1)] = -1
223         cosPhi = (np.cos(rho)*np.cos(colatitude[idx]) - cos_thetaP[
    p[0]: p[-1]+1, q[0]: q[-1]+1])/(rho*np.sin(colatitude[idx]))
224         Phi = np.arctan2( cosPhi, sinPhi) # position angle
    calculation
225 
226         eps = 2*np.pi - PA_pointing[idx] - delta + Phi
227         xe = rho*np.cos(eps)
228         ye = rho*np.sin(eps)
229         weight_new = Ellipical_weight(xe, ye) # calculating the
    elliptical beam weight
230         # indices of the corresponding interpolation patch on the
    grid.
231         p = ii+dpixel
232         q = jj+dpixel
233         m[p[0]: p[-1]+1, q[0]: q[-1]+1]+= I[idx]*weight_new #
    weighted intensity map
234         weight[p[0]: p[-1]+1, q[0]: q[-1]+1] += weight_new # weight
     map
235       equator = 0
236     print ('Completed file', file)
237 
238 # Stitching the edges of the north polar facets.
```

```
for i in range(4):
  te_i_start = i*Nside # starting index of top edge strip in x-axis
  te_i_stop = (i+1)*Nside + dpixel # ending index of top edge strip in x-axis
  te_j_start = (i+2)*Nside  # starting index of top edge strip in y-axis
  te_j_stop = te_j_start + wsize # ending index of top edge strip in y-axis

  if ( i == 3):
    se_i_start = (i-3)*Nside # starting index of side edge strip in x-axis
    se_i_stop = se_i_start + wsize # ending index of top edge strip in x-axis
    se_j_start = (i-2)*Nside + dpixel  # starting index of top edge strip in y-axis
    se_j_stop = (i-1)*Nside + wsize-1 # ending index of top edge strip in y-axis
  else:
    se_i_start = (i+1)*Nside
    se_i_stop = se_i_start + wsize
    se_j_start = (i+2)*Nside + dpixel
    se_j_stop = (i+3)*Nside + wsize-1

  a = m[se_i_start: se_i_stop, se_j_start: se_j_stop]
  m_average = (m[te_i_start: te_i_stop, te_j_start: te_j_stop] + np.rot90(a,3))/2. # average of top-edge and side-edge for weighted intensity map

  b = weight[se_i_start: se_i_stop, se_j_start: se_j_stop]
  w_average = (weight[te_i_start: te_i_stop, te_j_start: te_j_stop]+  np.rot90(b,3))/2. # average of top-edge and side-edge for weight map

  m[te_i_start: te_i_stop, te_j_start: te_j_stop] = m_average
  weight[te_i_start: te_i_stop, te_j_start: te_j_stop] = w_average

  m[se_i_start: se_i_stop, se_j_start: se_j_stop] = np.rot90(m_average)
  weight[se_i_start: se_i_stop, se_j_start: se_j_stop] = np.rot90(w_average)

# Removing the guard rings
m = m[dpixel: nsize+dpixel, dpixel:nsize+dpixel]
weight = weight[dpixel: nsize+dpixel, dpixel:nsize+dpixel]

# Facet 6
```

```
m_average = (m[4*Nside:nsize, 4*Nside:nsize] + m[0:Nside, 0:Nside
    ])/2.
weight_average = (weight[4*Nside:nsize, 4*Nside:nsize] + weight
    [0:Nside, 0:Nside])/2.
m[4*Nside:nsize, 4*Nside:nsize] = m_average
m[0:Nside, 0:Nside] = m_average
weight[4*Nside:nsize, 4*Nside:nsize] = weight_average
weight[0:Nside, 0:Nside] = weight_average

# identifying the bad pixels
bad_values = np.where(weight == 0)
m[bad_values] = 0; weight[bad_values] = 1
weighted_map = m/weight

# Removing the extra patches near polar facets.
for i in range(3):
  weighted_map[(i+1)*Nside + Nside: nsize, i*Nside:(i+1)*Nside] =
    0

# setting bad pixels to blank for display purpose
bad_values = np.where(weighted_map == 0)
weighted_map[bad_values] = np.NaN
cmap = matplotlib.cm.jet
cmap.set_bad('grey',1.)

# Displaying the 2D interpolation grid map
fig, ax = plt.subplots()
ax.set_xlim(0, nsize) # Set axis ranges; by default this will put
    major ticks every nsize.
ax.set_ylim(0, nsize)
ax.xaxis.set_major_locator(MultipleLocator(512)) # Change major
    ticks to show every 512.
ax.yaxis.set_major_locator(MultipleLocator(512))
ax.grid(which='major', color='#CCCCCC', linestyle='--') # Turn
    grid on for both major and minor ticks and style minor
    slightly differently.
plt.imshow(weighted_map, vmin = np.nanmin(weighted_map), vmax =
    np.nanmax(weighted_map)*0.1, cmap=cm.jet, origin = 'lower')

# saving the weighted map as a fits file
hdu = fits.PrimaryHDU(data=weighted_map)
hdu.writeto('/local/scratch/selvaraj/data/hpx_grid_ellipse.fits')
```

Listing E.1: Python code for elliptical beam interpolation

Bibliography

Ade P. et al., 2014a, Physical Review Letters, 112

Ade P. A. et al., 2014b, Astronomy & Astrophysics, 571, A16

Ade P. A. et al., 2011, Astronomy & Astrophysics, 536, A1

Ade P. A. et al., 2016, Astronomy & Astrophysics, 594, A4

Aghanim N. et al., 2018, arXiv preprint arXiv:1807.06207

Akrami Y. et al., 2018, arXiv preprint arXiv:1807.06205

Allen C. W., Gum C. S., 1950, Australian Journal of Scientific Research A Physical Sciences, 3, 224

Alvarez H., Aparici J., May J., Olmos F., 1997, Astronomy and Astrophysics Supplement Series, 124, 315

Araujo D. et al., 2012, The Astrophysical Journal, 760, 145

Armitage C., Wandelt B. D., 2004, Physical Review D, 70, 123007

Armitage-Caplan C., Wandelt B. D., 2009, The Astrophysical Journal Supplement Series, 181, 533

Ashdown M. et al., 2007, Astronomy & Astrophysics, 467, 761

Balanis C. A., 2016, Antenna theory: analysis and design. John wiley & sons

Baumgardner J. R., Frederickson P. O., 1985, SIAM Journal on Numerical Analysis, 22, 1107

Bennett C., et al., 2003, Astrophys. J. Suppl, 148

Bennett C. L. et al., 2003, ApJS, 148, 1

Bennett C. L. et al., 2013, The Astrophysical Journal Supplement Series, 208, 20

Bennett C. L. et al., 1992, The Astrophysical Journal, 396, L7

Bersanelli M. et al., 2010, Astronomy & Astrophysics, 520, A4

Beuermann K., Kanbach G., Berkhuijsen E. M., 1985, A&A, 153, 17

Bond J. R., Crittenden R. G., Jaffe A. H., Knox L., 1999, Computing in science & engineering, 1, 21

Bradt H., 2004, Astronomy methods: A physical approach to astronomical observations. Cambridge University Press

Burigana C., Malaspina M., Mandolesi N., Danse L., Maino D., Bersanelli M., Maltoni M., 1999, arXiv preprint astro-ph/9906360

Burke B. F., Graham-Smith F., 2002, An Introduction to Radio Astronomy: Second Edition

Calabretta M. R., Lowe S. R., 2013, Publications of the Astronomical Society of Australia, 30

Calabretta M. R., Roukema B. F., 2007, Monthly Notices of the Royal Astronomical Society, 381, 865

Carretti E. et al., 2013, Nature, 493, 66

Chiang L.-Y., Coles P., Naselsky P. D., Olesen P., 2007, Journal of Cosmology and Astroparticle Physics, 2007, 021

Crittenden R. G., Turok N. G., 1998, arXiv preprint astro-ph/9806374

Cyrille Rosset, Andrea Zonca M. R. L. S. D. L., 2019, Healpy documentation. `https://healpy.readthedocs.io/en/latest/install.html`, [Online; accessed 19-December-2019]

de Bernardis P. et al., 2000, Nature, 404, 955

Delabrouille J., 1998, Astronomy and Astrophysics Supplement Series, 127, 555

Djorgovski S. G., Mahabal A., Drake A., Graham M., Donalek C., 2013, Planets, Stars and Stellar Systems, 223–281

Driscoll J. R., Healy D. M., 1994, Advances in applied mathematics, 15, 202

Dröge F., Priester W., 1956, Zeitschrift fuer Astrophysik, 40, 236

Dupac X., Giard M., 2002, Monthly Notices of the Royal Astronomical Society, 330, 497

Frewin R. A., Polnarev A., Coles P., 1994, Monthly Notices of the Royal Astronomical Society, 266, L21

Gorski K. M., Hivon E., Banday A. J., Wandelt B. D., Hansen F. K., Reinecke M., Bartelmann M., 2005, The Astrophysical Journal, 622, 759

Gorski K. M., Hivon E., Wandelt B. D., 1998, Analysis issues for large cmb data sets

Gorski K. M., Wandelt B. D., Hansen F. K., Hivon E., Banday A. J., 1999, arXiv preprint astro-ph/9905275

Green R. M., Green R. M., 1985, Spherical astronomy. Cambridge University Press

Hanany S. et al., 2000, The Astrophysical Journal, 545, L5–L9

Harrison D., van Leeuwen F., Ashdown M., 2011, Astronomy & Astrophysics, 532, A55

Haslam C., Quigley M., Salter C., 1970, Monthly Notices of the Royal Astronomical Society, 147, 405

Haslam C., Salter C., Stoffel H., Wilson W., 1982, Astronomy and Astrophysics Supplement Series, 47, 1

Haslam C., Wilson W., Graham D., Hunt G., 1974, Astronomy and Astrophysics Supplement Series, 13, 359

Haslam C. G. T., Klein U., Salter C. J., Stoffel H., Wilson W. E., Cleary M. N., Cooke D. J., Thomasson P., 1981, A&A, 100, 209

Haslam C. G. T., Salter C. J., Stoffel H., Wilson W. E., 1982, A&AS, 47, 1

Haslam C. G. T., Wilson W. E., Cooke D. J., Cleary M. N., Graham D. A., Wielebinski R., Day G. A., 1975, Proceedings of the Astronomical Society of Australia, 2, 331

Hauser M. G. et al., 1998, The Astrophysical Journal, 508, 25

HEALPix, 2019, Data Analysis, Simulations and Visualization on the Sphere. `https://healpix.sourceforge.io/index.php`, [Online; accessed 19-December-2019]

Hinshaw G. et al., 2013, The Astrophysical Journal Supplement Series, 208, 19

Hinshaw G. et al., 2009, The Astrophysical Journal Supplement Series, 180, 225

Hu W., White M., 1997, arXiv preprint astro-ph/9706147

Ichiki K., 2014, Progress of Theoretical and Experimental Physics, 2014

Irfan M. O. et al., 2015, MNRAS, 448, 3572

Jansky K. G., 1932, Proceedings of the Institute of Radio Engineers, 20, 1920

Jarosik N. et al., 2011, The Astrophysical Journal Supplement Series, 192, 14

Jonas J., De Jager G., Baart E., 1985, Astronomy and Astrophysics Supplement Series, 62, 105

Jones M. E. et al., 2018, Monthly Notices of the Royal Astronomical Society, 480, 3224

Karttunen H., Kröger P., Oja H., Poutanen M., Donner K. J., 2016, Fundamental astronomy. Springer

Keihänen E., Kiiveri K., Kurki-Suonio H., Reinecke M., 2017, Monthly Notices of the Royal Astronomical Society, 466, 1348

Keihänen E., Kiiveri K., Lindholm V., Reinecke M., Suur-Uski A.-S., 2016, Astronomy & Astrophysics, 587, A27

Keihänen E., Kurki-Suonio H., Poutanen T., Maino D., Burigana C., 2004, Astronomy & Astrophysics, 428, 287

Keihänen E., Lindholm V., Lopez-Caniego M., Maris M., Reinecke M., Sandri M., Suur-Uski A.-S., 2019, Astronomy & Astrophysics, 632, A1

Keihänen E., Reinecke M., 2012, Astronomy & Astrophysics, 548, A110

Kovac J. M., Leitch E., Pryke C., Carlstrom J., Halverson N., Holzapfel W., 2002, Nature, 420, 772

Kunszt P. Z., Szalay A. S., Thakar A. R., 2001, in The hierarchical triangular mesh, Springer, pp. 631–637

Lamarre J.-M. et al., 2010, Astronomy & Astrophysics, 520, A9

Landecker T. L., Wielebinski R., 1970, Australian Journal of Physics Astrophysical Supplement, 16, 1

Langtangen H. P., 2012, A primer on scientific programming with Python. Springer

López-Caniego M. et al., 2014, The quijote cmb experiment: status and first results with the multi-frequency instrument

Maeda K., Alvarez H., Aparici J., May J., Reich P., 1999, Astronomy and Astrophysics Supplement Series, 140, 145

Maino D. et al., 1999, Astronomy and Astrophysics Supplement Series, 140, 383

Mather J. C., Fixsen D. J., Shafer R. A., 1993, in Infrared Spaceborne Remote Sensing, Vol. 2019, International Society for Optics and Photonics, pp. 168–179

Mennella A. et al., 2011, Astronomy & Astrophysics, 536, A3

Miller S., Childers D., 2012, Probability and random processes: With applications to signal processing and communications. Academic Press

Mitra S., Rocha G., Gorski K. M., Huffenberger K., Eriksen H., Ashdown M., Lawrence C., 2011, The Astrophysical Journal Supplement Series, 193, 5

Muciaccia P. F., Natoli P., Vittorio N., 1997, The Astrophysical Journal Letters, 488, L63

Naselsky P. D., Novikov I. D., Chiang L.-Y., 2006, The Astrophysical Journal, 642, 617

Naselsky P. D., Verrkhodanov O. V., 2008, International Journal of Modern Physics D, 17, 179–194

Natoli P., De Gasperis G., Gheller C., Vittorio N., 2001, Astronomy & Astrophysics, 372, 346

Ng T., Landecker T., Cazzolato F., Routledge D., Gray A., Reid R., Veidt B., 2005, Radio science, 40, 1

Penzias A. A., Wilson R. W., 1965, The Astrophysical Journal, 142, 419

Phillipps S., Kearsey S., Osborne J. L., Haslam C. G. T., Stoffel H., 1981, A&A, 103, 405

Polnarev A. G., 1985, Soviet Ast., 29, 607

Readhead A. et al., 2004, The Astrophysical Journal, 609, 498

Reich P., Reich W., 1986, Astronomy and Astrophysics supplement series, 63, 205

Reich W., 1982, Astronomy and Astrophysics supplement series, 48, 219

Reich W., Fürst E., Haslam C., Steffen P., Reif K., 1984, Astronomy and Astrophysics Supplement Series, 58, 197

Robishaw T., Heiles C., 2018, arXiv preprint arXiv:1806.07391

Roukema B. F., Lew B., 2004, arXiv preprint astro-ph/0409533

Saff E. B., Kuijlaars A. B., 1997, The mathematical intelligencer, 19, 5

Seljak U., Zaldarriaga M., 1997, Phys. Rev. Lett., 78, 2054

Sinnott R., 1984, Sky and telescope, 158

Smart W. M., Smart W., Green R., 1977, Textbook on spherical astronomy. Cambridge University Press

Smoot G. F. et al., 1992, The Astrophysical Journal, 396, L1

Spergel D. N. et al., 2007, The Astrophysical Journal Supplement Series, 170, 377

Sullivan W. T., 2009, Experimental Astronomy, 25, 107

Sun X., Han J., Reich W., Reich P., Shi W., Wielebinski R., Fürst E., 2007, Astronomy & Astrophysics, 463, 993

Sutton D., Johnson B., Brown M., Cabella P., Ferreira P., Smith K., 2009, Monthly Notices of the Royal Astronomical Society, 393, 894

Szalay A. S., Gray J., Fekete G., Kunszt P. Z., Kukol P., Thakar A., 2007, arXiv preprint cs/0701164

Taylor A. C., 2018, The c-band all-sky survey

Tegmark M., 1996, arXiv preprint astro-ph/9610094

Tegmark M., 1997, The Astrophysical Journal Letters, 480, L87

Testori J., Reich P., Bava J., Colomb F., Hurrel E., Larrarte J., Reich W., Sanz A., 2001, Astronomy & Astrophysics, 368, 1123

Tinbergen J., 2005, Astronomical polarimetry. Cambridge University Press

Turtle A. J., Baldwin J. E., 1962, MNRAS, 124, 459

Wallis C. G., Brown M. L., Battye R. A., Pisano G., Lamagna L., 2014, Monthly Notices of the Royal Astronomical Society, 442, 1963

Wiener N., 1949, Time Series, with Engineering Applications

Yates K. W., Wielebinski R., Landecker T. L., 1967, Australian Journal of Physics, 20, 595

Zaldarriaga M., Seljak U., 1997, Physical Review D, 55, 1830