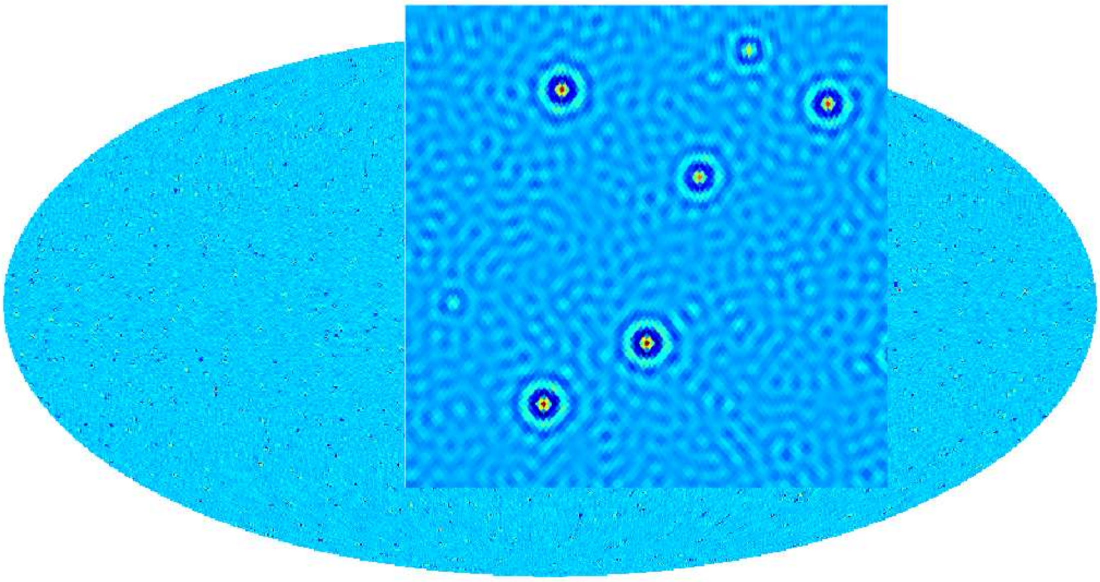
Point Source Detection with Wavelets applied on Cosmic Microwave Radiation Maps



Master of Science thesis by

Michael Katz
Institute of Theoretical Astrophysics
University of Oslo
Norway

April 2008

Contents

Contents iii

List of figures v

List of tables vii

Preface ix

1 Introduction 1

2 Cosmology and the CMB 2
2.1 Cosmology 2
2.1.1 The cosmological parameters 2
2.1.2 Inflation 3
2.1.3 Isotropy and homogeneity 4
2.2 The Cosmic Microwave Background 4
2.2.1 The epoch of recombination 4
2.2.2 The Boltzmann equations 5
2.2.3 The evolution of anisotropies 5

3 CMB analysis 8
3.1 Overview 9
3.2 Simulating correlations 9
3.3 Fourier transformation 11
3.3.1 Fourier theory 11
3.3.2 Fourier transformation of the 2D field 13
3.4 Simulation on the sphere 15
3.4.1 Fourier transformation on the sphere 15
3.4.2 Hierarchical Equal Area isoLatitude Pixelisation (HEALpix) 16
3.5 The power spectrum 17
3.6 Noise, beam and the pixel window 19
3.6.1 Noise 19
3.6.2 Beam 20
3.6.3 The pixel window 21
3.7 The foregrounds 23

4 Point source detection and the wavelet technique 26

4.1 Point source detection 26
4.1.1 Resolved point sources 26
4.1.2 Unresolved point sources 28
4.2 Wavelets 29
4.2.1 Wavelet theory 29
4.2.2 The Spherical Mexican Hat Wavelet and Spherical Needlets 30

5 Method and implementation **35**
5.1 Problem 35
5.2 Implementation of resolved point source detection 35
5.2.1 Without wavelets 35
5.2.2 With wavelets 39
5.3 Implementation of unresolved point source detection 41
5.4 Introducing noise in the analysis 41

6 Results **44**
6.1 Detection of resolved point sources 44
6.1.1 Without wavelets 44
6.1.2 Spherical Mexican Hat wavelets 45
6.1.3 Needlets 47
6.1.4 Unknown point source locations 54
6.2 Detection of unresolved point sources 56
6.2.1 Spherical Mexican Hat wavelets 56
6.2.2 Needlets 58
6.2.3 Estimation of skewness and kurtosis for a given amplitude 60
6.2.4 χ^2 minimization and correction to the power spectrum 62
6.3 Detection with noise 63
6.3.1 Detection of resolved point sources 63
6.3.2 Detection of unresolved point sources 67

7 Summary and conclusions **72**

A Source code **75**
A.1 Detection of resolved point sources 75
A.2 Detection of unresolved point sources 85

Bibliography **94**

List of Figures

3.1 The Wilkinson Microwave Anisotropy Probe (WMAP) 8
3.2 Uncorrelated two dimensional field and its histogram 12
3.3 Correlated two dimensional field using Cholesky decomposition and Fourier transformation 12
3.4 Division of the Mollweide projected sky map using HEALpix 16
3.5 Analysis of the mean power spectrum of 100 simulated universes 18
3.6 Three sectional maps and their C_l showing how the noise is reduced with the number of simulations 20
3.7 $N_{\mathrm{side}} = 128$ map and its power spectrum with no beam and a $1°$ beam 22
3.8 $N_{\mathrm{side}} = 512$ map and its power spectrum with no beam and a $14'$ beam 22
3.9 Foreground emission where the CMB predominates 23
3.10 Mask map example 25
3.11 The effect masks make on the power spectrum 25

4.1 Flux values at each pixel for a small sky map with marked thresholds 27
4.2 Waves and wavelets in real space. 30
4.3 Needlets in pixel space 32
4.4 The power spectrum C_l after 10 simulations, now with a greater amplitude at the small scales caused by the point sources 33
4.5 Wavelet transformation of the power spectrum with an SMH Wavelet 34
4.6 Wavelet transformation of the power spectrum with a needlet 34

5.1 Sky map before and after added point sources 36
5.2 Smoothing of the nearby pixels around the point source 37
5.3 Determining the mask size from a needlet with many troughs 40
5.4 Sky map filtered with the SMH wavelet of scale $R = 7.5'$ 40
5.5 Sky maps after transformation with two different needlets 42
5.6 The RMS noise map from the V band of WMAP 43

6.1 Histograms of sky map with and without point sources 46
6.2 Number of true and false detections after wavelet transformation with the SMH wavelet for selected R values 46
6.3 A finer plot of the number of true detections around $R = 7.5'$ 47
6.4 Plot of the number of true detections of the needlets in table 6.4 and 6.5 50
6.5 Associated l_{max} and l_{min} for the needlets in table 6.4 and 6.5 51
6.6 Associated number of false detections for the needlets in table 6.4 and 6.5 . . . 51
6.7 Finer plots of the number of true and false detections around $a = 1.09898$ . . . 52

6.8 Number of true and false detections at all j for the best scale $a = 1.096259$ . . 52
6.9 Needlet $a = 1.096259$ of scale $j = 75$ and $j = 70$ with maximum at 5σ, show why higher j give more false point sources 53
6.10 Number of point source detections at all j for the scale $a = 1.44428$ when the point source locations are unknown 56
6.11 Skewness and kurtosis with confidence levels for the SMH wavelet map up to very large scales 57
6.12 Skewness and kurtosis as a factor of σ_S and σ_K for the SMH wavelet map at the small scales 57
6.13 Finer plots of skewness and kurtosis for the SMH wavelet map 58
6.14 Skewness and kurtosis as a factor of σ_S and σ_K for various values of a 59
6.15 Finer plots of skewness and kurtosis of the a's that in figure 6.14 measured the largest deviations 59
6.16 Skewness and kurtosis with confidence intervals for all j for the needlets that measured the largest deviation 60
6.17 Skewness and kurtosis as a function of constant point source amplitude 62
6.18 The χ^2 minimization for one simulation using several scales of the SMH wavelet and one needlet. 63
6.19 Corrected power spectrum with error bars using the SMH wavelet 64
6.20 Corrected power spectrum with error bars using one needlet 65
6.21 The standard deviation of the power spectra in figure 6.19 and 6.20 65
6.22 Number of true detections on sky maps with noise after wavelet transformation with the SMH wavelet for selected R values 66
6.23 Number of true detections on sky maps with noise after wavelet transformation with a selection of needlets 67
6.24 A finer plot of the number of true detections around the largest peak in figure 6.23 68
6.25 Non-filtered, SMH wavelet filtered and Needlet filtered sky maps with noise . . 69
6.26 Skewness and kurtosis for the SMH wavelet transformed maps with noise 70
6.27 Finer plots of the skewness and kurtosis in figure 6.26 for the SMH wavelet transformed maps with noise 70
6.28 Skewness and kurtosis for sky maps with noise filtered with various needlets . . 71
6.29 Skewness and kurtosis around the peaks in figure 6.28 based on data from 1000 simulations 71

List of Tables

6.1 Number of true and false detections for three thresholds ($N = 10$) 45
6.2 Number of true and false detections for three thresholds ($N = 100$) 45
6.3 Average number of unique detections for each scale compared to $R = 7.5'$ 47
6.4 Number of true and false detections for a selection of needlets at their smallest possible scale j 48
6.5 Number of true and false detections for a selection of needlets with l_{max} reaching 1080 49
6.6 Detection rate for the highest values of j for the needlet with $a = 1.096259$ . . . 53
6.7 Average number of unique detections for the highest values of j for $a = 1.096259$ compared to $j = 75$ 54
6.8 Detection rate for the highest values of j for the needlet with $a = 1.096259$ when the point source locations are unknown 55
6.9 Detection rate for needlets that had low numbers of false detections 55

7.1 Summary of the best results for the detection of resolved point sources 73
7.2 Summary of the best results for the detection of unresolved point sources . . . 74

Preface

The main analysis in this thesis was performed using the FORTRAN programming language. FORTRAN is a powerful language for scientific simulations by being highly efficient, featuring special adaption for scientific calculations and being quite suitable for supercomputing. The parallelization of the programs for use in supercomputing was done using the Message Passing Interface (MPI). C++ is also widely used for simulation of physical processes, but needs more optimization to reach the same performance level as FORTRAN. For making plots and quick analysis of the data outputted by FORTRAN, IDL was used. IDL is short for Interactive Data Language, and is widely used for data analysis and visualization of data. IDL is a much slower programming language than FORTRAN, and is thus not wise to use when dealing with heavy computations. Both FORTRAN and IDL was used together with the HEALPix (Hierarchical Equal Area isoLatitude Pixelisation) package made available by the Jet Propulsion Laboratory (JPL) at NASA. The HEALPix package features premade programs for use with CMB analysis on the sphere. The thesis is written in the document markup language LaTeX, and the GIMP was used when image editing or merging of the images was required.

My father first triggered my interest in astronomy, by reading astronomy books as good-night stories by my bedside when I was a child. I would like to give gratitude to him and to all my family in Norway and Australia, my girlfriend, Kristin Charlotte Carlsson, and my friends, above all Silje Bjølseth, Espen Raugstad, Zoya Shah and the great people in the study hall, for all their support throughout my studies. A special thanks also to my supervisor, Frode K. Hansen. Without his great pedagogical efforts, the subject of this thesis would be less comprehensible. An additonal thanks to all the artists of the music I listened to, in particular John Powell, Nightwish, Tristania, george and Within Temptation, which made me more efficient.

Front page figure is a wavelet transformed point source contaminated sky map with a needlet with parameters $a = 1.35483$ and $j = 22$.

Michael Katz
April 2008

Chapter 1

Introduction

The Cosmic Microwave Background (CMB) is the remnants of the early universe, and can be considered to be one of the best evidences for the Hot Big Bang model. When the Big Bang occurred 13.7 billion years ago the universe was hot and dense, but the universe expanded and eventually cooled down. In the beginning, the photons were coupled to matter and the universe was opaque. The CMB marks the transition where the photons decoupled and were able to move freely across the universe. This is the radiation we actually observe as the CMB, and it is among the oldest information of the early universe. The CMB covers all the universe and has the same statistical properties regardless of the direction in which you observe (isotropy). However, very small deviations from isotropy (anisotropy) have been observed, and these are of interest to cosmologists.

The data from our observations of the CMB are not without unwelcome signals. The photons from the last scattering surface have traveled a long way, and on their path towards us there are foregrounds also emitting in the microwave range (e.g. galaxies). After this long journey, when the photons finally arrive at our instrument, additional instrumental artifacts get added to the final data. Cosmologists, who yearn for the best description of our universe, need the most accurate information they can get. It is therefore of utter importance that the contaminants are removed. There is, however, a second gain from removing all the foreground emissions. If the CMB signal still, after the extraction, showed unexpected non-Gaussianity and after other astrophysical phenomena had been excluded as sources, great revisions to how we currently model our universe would have been necessary.

This thesis will investigate different methods, mainly using wavelet analysis, for removing certain classes of foreground contaminants. Chapter 2 reviews basic cosmology and its relation to the CMB, and the third chapter introduces important concepts in CMB analysis. The cosmology in these chapters is mainly based on the texts by Dodelson [8], Ryden [21] and Elgarøy [9], while the statistics are based on the texts by Moore [20] and Murray [22]. Chapter 4 encompasses an introduction to the detection of foreground emission and to wavelets, while chapter 5 will make an approach to the problem and show how the simulations with the wavelet analysis will be implemented. Chapter 6 contains the results generated in the analysis, and the last chapter summarizes the thesis, concludes the results, and suggests further research on the subject.

Chapter 2

Cosmology and the CMB

The study of the CMB is just one subject of cosmology, and it is important to get a grasp on concepts linked to the CMB to fully understand it. Cosmology is the study of the very largest scales of the universe, where even galaxy clusters become inconceivably small. It ultimately comprises of understanding what the universe is, what it consists of, how it all started and how it all will end. These are questions humans have asked themselves for all time, but cosmology as a science is relatively new.

2.1 Cosmology

2.1.1 The cosmological parameters

In 1922, Alexander Friedmann, inspired by Einstein's general relativity, constructed a set of equations linking different *cosmological parameters* together:

$$\dot{a}^2 + kc^2 = \frac{8\pi G}{3}\rho a^2 + \frac{\Lambda}{3}a^2 \tag{2.1}$$

$$\ddot{a} = -\frac{4\pi G}{3}\left(\rho + \frac{3p}{c^2}\right)a + \frac{\Lambda}{3}a, \tag{2.2}$$

where c is the speed of light, G the gravitational constant and $\dot{} = \frac{\mathrm{d}}{\mathrm{d}t}$. The equations can be used to derive complex descriptions of how a universe behaves, when fed with values or conditions for the different parameters. The parameter a, called the scale factor, is a measurement of the expansion of the universe from a given reference point. The parameter k tells us how the universe curves or if it is flat, while ρ is the energy density of the universe, given by[1]

$$\rho = \rho_0 a, \tag{2.3}$$

where ρ_0 denotes the density today. The amount of matter or energy is more conveniently expressed as a ratio to the critical density ρ_c, called the density parameter:

$$\Omega \equiv \frac{\rho}{\rho_c},$$

[1]The equation is obtained by solving the fluid equation $\dot{\rho} = -3\frac{\dot{a}}{a}(\rho + \frac{p}{c^2})$, having introduced the equation of state $p = w\rho c^2$, where w is a constant depending on the universe model.

and the critical density itself depends on the Hubble parameter:

$$\rho_c \equiv \frac{3H^2}{8\pi G}.$$

When $a = 1$ today, the density parameter is

$$\Omega_0 = \frac{\rho_0}{\rho_c} = \frac{8\pi G}{3H_0^2}\rho_0.$$

The last parameter in equations (2.1) and (2.2), the cosmological constant Λ, gives the amount of exotic energy in the universe (in our universe, this might be the dark energy). But how do we determine what these parameters are for our universe? The answer is the cosmic microwave background, as well as other observations. From observations of the CMB we can extract the so called power spectrum, which tells us what the parameters are. The power spectrum will be examined in greater detail in chapter 3. The ability of the above equations to model universes is best demonstrated using a particular example. A good example must be the current observations, which favour the flat ΛCDM-model, a universe dominated by dark energy and dark matter, with $\Omega_{\Lambda 0} > 0$. In this example k must be zero, and since dark matter and energy dominates, the radiation term is negligible:

$$\frac{H(t)^2}{H_0^2} = \frac{\Omega_{m0}}{a^3} + 1 - \Omega_{m0},$$

where conservation of elements implies $\Omega_{\Lambda 0} = 1 - \Omega_{m0}$. When integrated over all time, the following scale factor is the result

$$a(t) = \left(\frac{\Omega_{m0}}{1-\Omega_{m0}}\right)^{1/3} \left\{\sinh\left(\frac{3}{2}\sqrt{1-\Omega_{m0}}H_0 t\right)\right\}^{2/3}.$$

From this equation we see that we live in a universe that expands with a rate of $a \propto \sinh t^{2/3}$, that is, it expands faster the larger it gets. Thus, from the knowledge of what certain cosmological parameters are, one can conclude that the universe has an accelerating expansion.

2.1.2 Inflation

The ΛCDM-model has no end, but it has a beginning, the Big Bang. However, the Hot Big Bang model introduces a couple of problems: the flatness problem, the monopole problem and the horizon problem. It will suffice to explain the latter to understand the context. In every universe there is a *particle horizon*. The horizon defines a limit of how far two points in the universe can be separated, in order to have exchanged information some time in the history of the universe. Consider two points, 180° opposite to each other on the CMB with an observer at Earth. Last scattering occurred much closer to Big Bang than our time, so the distance from us to the surface is almost as big as the particle horizon. Therefore the distance between the two points is longer than the distance to the particle horizon, and the points can not be causally connected. But the problem arises when we notice that the points on the CMB have nearly the same temperature. How can points that have not been able to exchange information have more or less the same temperature? The three problems were solved with the introduction of *inflation* into the Hot Big Bang model. In the inflation phase, the universe went through an extremely rapid expansion for a short period very early on the time line, before it continued to evolve as today. Before this phase the two points of the horizon problem were in causal contact, and could get the same physical properties. The scale of the universe was blown up considerably during the inflation phase, causing the physical properties of the two points to stay the same after this phase, despite appearing to be far from each other.

2.1.3 Isotropy and homogeneity

The universe is *isotropic* and *homogeneous* on very large scales; isotropic because it looks the same regardless of the direction you look from one reference point, and homogeneous because it looks the same regardless of reference point. However, this is surely not the case on smaller scales, considering that we do have matter clumping together to form structures like galaxies. The scalar field set up by quantum fluctuations in the inflation phase induced fluctuations in the radiation and matter, which eventually ended up as *anisotropies* in the CMB and *inhomogeneities* in the matter. We will take a closer look at the anisotropies in the CMB in the next section.

2.2 The Cosmic Microwave Background

The *Cosmic Microwave Background* was accidentally discovered by Penzias & Wilkinson in 1965 when they were observing microwave signals at wavelengths of $\lambda = 7.35$ cm, and received a stronger signal than expected. The energy of a CMB photon is so low, just $E_\gamma = 6 \times 10^{-4}$ eV, that the discoverers first believed they found nothing more than additional noise. The signal was isotropic and constant in time, and contributed with a temperature of 3.5 K to the antenna. Great effort was put in finding the source of and removing the "noise", but without success, until they were put in contact with Robert Dicke, who had predicted the CMB theoretically. When the COsmic Background Explorer (COBE) satellite observed the CMB in 1989, the information about it could be much more accurately measured. The CMB was found to radiate very close to an ideal blackbody at

$$\langle T \rangle = 2.725\,\mathrm{K} \tag{2.4}$$

peaking at $\lambda = 0.19$ cm. In addition, the temperature fluctuations across the sky map were measured at only 30 mK, that is, the background is *very* close to isotropic. These observations fit neatly with the Hot Big Bang model.

2.2.1 The epoch of recombination

The background radiation we see today appeared approximately when the universe was $t = 3 \times 10^5$ years old. Before this time the radiation and matter were both part of an ionized plasma, and the universe was in an opaque state. If, for example, a hydrogen atom was formed in this plasma, it would be dissolved again quickly by high-energy photons. Being tightly coupled with the electron-proton fluid, the photons interacted with the electrons by Compton scattering. This occurred as long as the scattering rate stayed above the rate of which the universe was expanding, or in other words, as long as the mean free path of the photon was smaller than the horizon distance. At some point in time, when the universe was cooled down enough, significant amounts of protons and electrons could combine, making the ionized plasma a neutral fluid. This is called the epoch of *recombination*. When the expansion rate became larger than the scattering rate, the photons could *decouple* and were able to move freely without further interactions with electrons, making the universe increasingly transparent. The point in time where most photons decoupled, is called the *last scattering*, and surrounding us today is the *last scattering surface* from which the CMB photons have traveled freely towards us. Recombination, decoupling and last scattering are so close in time compared to the age of the universe, that they are often approximated as events occurring at the same time. This is what will be done from here on, denoting it η_*[2].

2.2.2 The Boltzmann equations

The primordial perturbations in the matter and radiation were predicted theoretically before they were observed. The physics of the particles in the early universe is described by *Boltzmann equations*,

$$\frac{\mathrm{d}f(\mathbf{x},\mathbf{p},t)}{\mathrm{d}t} = C(f),$$

giving statistical information about particles with momentum $\mathbf{p}$ passing a point in space $\mathbf{x}$ per unit time t, with the collision terms on the right hand side of the equation. Since the universe consists of many unique types of particles, a set of Boltzmann equations is needed, and since many particles interact, the equations must be solved as a whole. The following phrase from general relativity may be familiar: *Mass tells space-time how to curve, and space-time tells matter how to move.* In other words, this phrase tells us that all the particles set up a *gravitational potential*, which again influence the behaviour of the particles. In the end, there are eight differential equations dictating the evolution of perturbations that must be solved: two for the non-baryonic dark matter (density and velocity), two for the baryons, one for the massless photons (no density), one for the massless neutrinos and two for the gravitational potential (Newtonian potential and potential in the space-time curvature). Recall that inflation initially perturbed the energy density field, and naturally it is from inflationary theory that the initial conditions for the differential equations originate from. We see now that the fluctuations in the radiation field and the perturbations in the matter components influence the evolution of one another, and one must therefore include the matter perturbations in the study of how the anisotropies in the CMB appeared.

2.2.3 The evolution of anisotropies

The perturbations in the distribution of photons as seen from a point $\mathbf{x}$ in space with photon directions $\hat{p}$ is denoted $\Theta(\hat{p},\mathbf{x})$. To easily distinguish between the various scales in the perturbations, it is customary to Fourier transform Θ into harmonic space (Fourier theory will be covered in greater detail in the next chapter), where the perturbations now become a function of the wave number $\mathbf{k}$. This enables us to study each perturbation scale separately from all the others. Most generally, the perturbations can be defined in terms of Legendre polynomials $\mathcal{P}_l$:

$$\Theta_l = \frac{1}{(-i)^l}\int_{-1}^{1}\frac{\mathrm{d}\mu}{2}\mathcal{P}_l(\mu)\Theta(\mu),$$

where the variable μ now defines the photon propagation direction in regards to the wave number. The smaller the angular scales of the perturbations in the temperature field as seen

[2]The conformal time is defined in terms of the scale factor: $\mathrm{d}\eta = \frac{\mathrm{d}t}{a}$.

from one point, the more coefficients Θ_l are needed to describe it. When there is no variation in the temperature field, the perturbations only need to be described by the *monopole moment* Θ_0.

Now that a quick overview of the fundamentals has been made, it is time to discuss how the fluctuations in the radiation field evolved into the anisotropies we observe today. We will start with the very early perturbations that appeared before recombination. The large scale perturbations must be separated from the small scale perturbations, which evolve differently. Fourier modes larger than the particle horizon, super horizon perturbations, will not evolve much, since all points on the structure can not interfere with each other. Therefore, when the large scale modes is observed today, they look very much like they did early in the history of the universe. When the differential equations described in the last paragraph are solved for radiation and large scales at the time of recombination, the following relation is found:

$$(\Theta_0 + \Psi)(k, \eta_*) = -\frac{1}{6}\delta(\eta_*). \tag{2.5}$$

Here δ denotes the perturbations in matter and Ψ the gravitational potential in the curvature. With equation (2.5) we can directly relate the observed anisotropies in the temperature to the perturbations in the matter. The equation informs us that for overdense regions in the matter perturbations, the anisotropy in the temperature will be observed as cold. This contradicts common sense, but does in fact have an explanation. Before the photons can start their long journey towards us, they must get out of the potential well set up by the overdensity, which causes the photons to lose energy before we observe them. In short, for large scale anisotropies, the overdense regions are observed as cold spots on the sky, while the underdense regions are observed as hot spots. The phenomenon is called the *Sachs-Wolfe effect*.

Smaller scales enter the horizon earlier than large scales, that is, as the horizon gradually gets bigger, larger and larger scales end up inside the horizon and causal physics can begin to act. When the density becomes large enough, the perturbations will start to collapse under their own gravity and at the same time create an overpressure that halts the collapse. The perturbations are then dissolved, causing an underpressure which reinitiates the collapse. In such a way, the perturbations create acoustic oscillations. The modes enter the horizon at different times, and the fluctuations on these scales at the time of recombination will vary accordingly. Before recombination, the last scattering surface was much smaller than the horizon, due to the short mean free path of the photons. When the photons all originate from points close to the observer, the photons must be in thermal equilibrium, and the anisotropies are dominated by the monopole moment Θ_0. In addition, we must account for the *dipole moment* Θ_1 caused by our movement relative to the photons, where the photons behind us appear as colder than those in front (Doppler shift). Other moments of the perturbations can not be observed in the opaque fluid, except for the *quadrupole moment* Θ_2 from photon diffusion, which has a small nonnegligible effect. When the Boltzmann equations are solved for each of these moments and the resulting equations combined, the perturbations will manifest themselves as alternating high and low peaks in k space with a dampening at small scales, corresponding to the physical description above.

The early evolution of the perturbations affects how they look when observed today. Right after recombination the anisotropy photons can move freely towards us, but are influenced by some other effects prior to arrival. The potentials between us and the last scattering surface are not constant, since there still is a transition between radiation dominated universe and matter dominated. Later, the dark energy starts to have its influence on the potentials. The wells created by these potentials can cause the photons to loose or gain energy, according to how they vary in time. If the well gets smaller while the photon is caught in it, the photon gains energy. If it increases, the photon loses energy. When the photons originated from a non-varying potential well at the time of recombination, it was called the Sachs-Wolfe effect. The influence of time-varying wells after recombination is called the *integrated Sachs-Wolfe effect*. Also, last scattering happened early in the history of the universe, and the photons have used quite some time to reach us. During this time the universe expanded, and stretched the photons. With the knowledge of every influence to the photons, the most pristine form of the power spectrum is now known. The cosmological parameters each have an additional effect on how the power spectrum looks, making it possible to read their values by finding deviations from its predicted form.

Chapter 3

CMB analysis

The microwave radiation is registered with a radio antenna on a satellite or on the ground (preferably with a balloon), and is pointed in every direction on the stellar vault. As mentioned in the last chapter, the first satellite to observe the CMB, the COBE, achieved some remarkable discoveries that eventually made it to the Nobel prize in physics in 2006. The COBE was launched in 1989 with the goal of measuring the primordial radiation in our universe. It carried three instruments: the Diffuse InfraRed Background Experiment (DIRBE), the Differential Microwave Radiometer (DMR) and the Far InfraRed Absolute Spectrophotometer (FIRAS). The DMR instrument was the first to observe the anisotropies in the CMB [13], and the FIRAS instrument showed that the CMB radiates very close to an ideal blackbody [10]. Its successor, the Wilkinson Microwave Anisotropy Probe (WMAP) (figure 3.1), carried microwave instruments that could measure the full sky with higher accuracy and resolution, but will soon be succeeded by Planck, to be launched in July 2008.

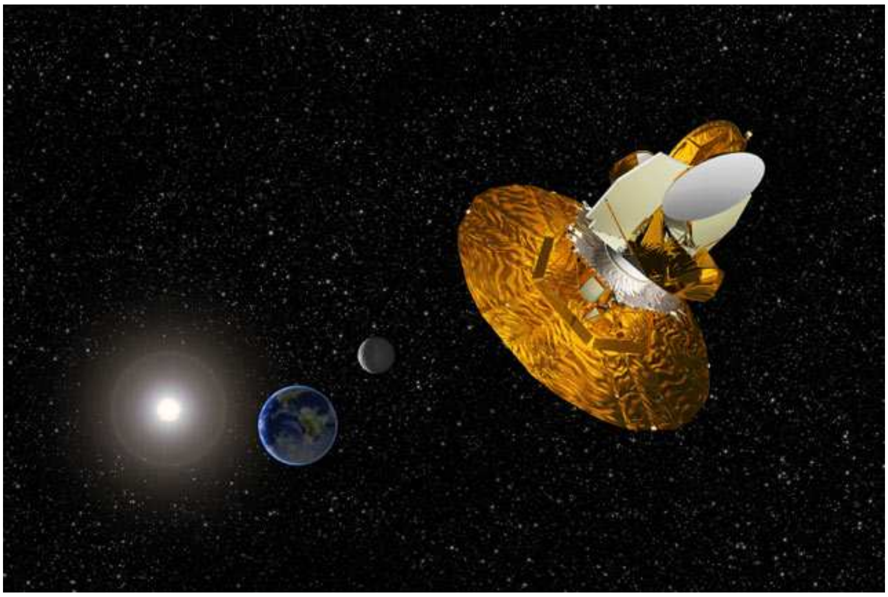


Figure 3.1: A concept image of the Wilkinson Microwave Anisotropy Probe (WMAP) in orbit around the second Lagrangian point. Courtesy NASA / WMAP Science Team.

The radio sensors carried on the satellites register hot or cold regions on the sky depending on the direction it is pointed. Before any physical analysis is performed, the signal data of the CMB over the full sky is visualized on a spherical flat surface (see figure 5.1(a) on p. 36), just like when the Earth is shown on a flat surface (Mollweide projection). The color coding on the map represents the registered temperature fluctuations from the mean temperature given in equation (2.4), but in CMB theory we adjust the scale such that the mean becomes 0. Red areas represent hot regions, and the blue areas represent cold regions. This chapter will describe how realizations of the CMB is generated from theory, discuss the power spectrum and how it is extracted from the CMB data and look at phenomena that complicates our observations of the CMB.

3.1 Overview

Current inflationary models predict that the very initial perturbations during the inflation phase are Gaussian distributed, which means the small temperature fluctuations in the CMB are Gaussian distributed. This is very advantageous since we then can make use of the statistical properties of that distribution in the analysis of the data. The density curve of a Gaussian distribution peaks at the mean μ, and since the frequency of occurrences on each side of the mean is the same, it is neither skewed to the left nor right. To measure the spread of the data we use the standard deviation σ. If the spread is small, the deviation from the mean is small, and the curve falls off quickly. If the spread is large, the deviation from the mean is large, and the curve falls off slowly. The probability density function of Gaussian data is

$$P(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right),$$

where x is the data which here represents the temperature fluctuations. With this function the probability of a specific temperature value can be calculated. Gaussian distributed data have certain characteristics that will be utilized throughout the thesis.

The CMB map is the stage of data representation prior to any physical analysis of the data. Real CMB data can consist of very large data sets, and an effective method of data representation is therefore desirable. In the analysis of this thesis, observations of the CMB will be simulated. When the distribution of the physical system is known, several simulations by random sampling from the distribution can be performed through the Monte Carlo method. The Monte Carlo method is simply a stochastic numerical process which repeats an algorithm a given number of times. The results from each simulation are averaged over the number of simulations, and an accurate measurement of the system with an estimation of the error can be achieved. To understand how to make a simulated CMB map from theory, we must start simple and work our way upwards. The correlations in the field of the background are complicated, and the principles of correlations and the most implicit method of simulating it will first be explained. In the end we will see a quicker method of simulation, and extend what we have learned to the sphere.

3.2 Simulating correlations

We begin with a two dimensional grid, where each grid point is one unique pixel. The previous section showed that the temperature fluctuations in the CMB are Gaussian distributed. This will be the starting point when random data is now created for making a simple field in IDL with equal variance in all pixels. The idea behind the program is simple: make an $N \times N$ grid, take random numbers from a Gaussian distribution and insert them into each grid point. In figure 3.2(a) you find the grid visualized. Light areas correspond to positive numbers, and dark areas to negative numbers. This of course coincides with hot and cold areas on the CMB map, but the created grid in the figure bears no resemblance to a CMB map at all. Because of the physical processes that initially generated the perturbations in the background, the temperature in the different pixels are correlated. According to statistical theory, the correlation between two random variables is

$$\langle x_i x_j \rangle = C_{ij}\sigma^2,$$

where C_{ij} is the correlation matrix given by

$$C_{ij} = \xi(|\mathbf{x_i} - \mathbf{x_j}|). \tag{3.1}$$

Here ξ is the two-point correlation function for an isotropic field (i.e. random variables drawn from a distribution aiming for isotropy), which depends on the norm of the relative distance between the two random variables x_i and x_j. If an element of the matrix is 1, then x_i and x_j is the same number, and a number closer to zero means less influence between the two variables. Since the correlation matrix must explain the relation between each and every variable in the entire $N \times N$ grid, the dimension of C_{ij} becomes $N^2 \times N^2$.

The correlation function [9]

$$\xi = \frac{\sin kr}{kr}$$

with constant wave number k will be used as an example for generating structures, where $r = |x_i - x_j|$ is the distance between the two variables. The tricky part here is to find the distance r, but this can be done by using the indices of the matrix. A very small grid will be used to illustrate how to find the distance r. Consider the 2×2 grid

0	1
2	3

with the associating 4×4 correlation matrix

$$\mathbf{C} = c_{ij} = \begin{pmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{pmatrix}.$$

The correlation matrix gives the correlation value between each pixel i and j, but if the correlation function is to be used, one need the coordinates of these two pixels in the grid. To find a variable's position y along the horizontal axis we take the index modulu[1] the size of the matrix, and to find the position z along the vertical axis we take the floored index divided by the size. This is followed by taking the difference between the y coordinates and the z coordinates of the variables, and eventually using the well-known

$$r^2 = \Delta y^2 + \Delta z^2.$$

[1]Modulu is the remainder after a division between the involved numbers.

For the 2×2 example grid, let us find the distance between pixel 0 and 3. The coordinates for pixel 0 is $y_0 = i \mod N = 0 \mod 2 = 0$ and $z_0 = \lfloor \frac{j}{N} \rfloor = \lfloor \frac{0}{2} \rfloor$, and similarly the coordinates for pixel 3 is $y_3 = 1$ and $z_3 = 1$, that is a distance of $r = \sqrt{(y_3 - y_0)^2 + (z_3 - z_0)^2} = \sqrt{2}$. To successfully draw a correlated map using C_{ij}, we must take the Cholesky decomposition[2] of the matrix

$$\mathbf{C} = \mathbf{L}\mathbf{L}^T, \tag{3.2}$$

while finalizing with a matrix multiplication between the resulting lower triangular matrix $\mathbf{L}$ and the Gaussian distributed grid $\mathbf{x}$:

$$\mathbf{x}^{'} = \mathbf{L}\mathbf{x}, \tag{3.3}$$

This is possible since the correlated pixels are linked to the correlation matrix through Cholesky decomposition. To see this, let $\langle x_i^{'} x_j^{'} \rangle$ be the correlated pixels, and x_i and x_j the uncorrelated ones. The discrete form of equation (3.3) is

$$x_i^{'} = \sum_k L_{ik} x_k,$$

and inserting this into the expression for correlated pixels yields

$$\langle x_i^{'} x_j^{'} \rangle = \sum_k \langle x_k^2 \rangle L_{ik} L_{jk}.$$

The statistic $\langle x_k^2 \rangle$ is the variance, and the rest of the sum is the discretized Cholesky decomposition, such that

$$\langle x_i^{'} x_j^{'} \rangle = \sigma^2 C_{ij}.$$

The resulting correlated map is drawn figure 3.3(a). As previously indicated, the correlation matrix has a dimension $N^2 \times N^2$, and in addition the Cholesky decomposition requires $O(N^3)$ operations. Thus, using this method to create a CMB map is bad when one is dealing with a real CMB map and large N. To wait longer than the lifetime of the universe is not preferential, so another method will be considered.

3.3 Fourier transformation

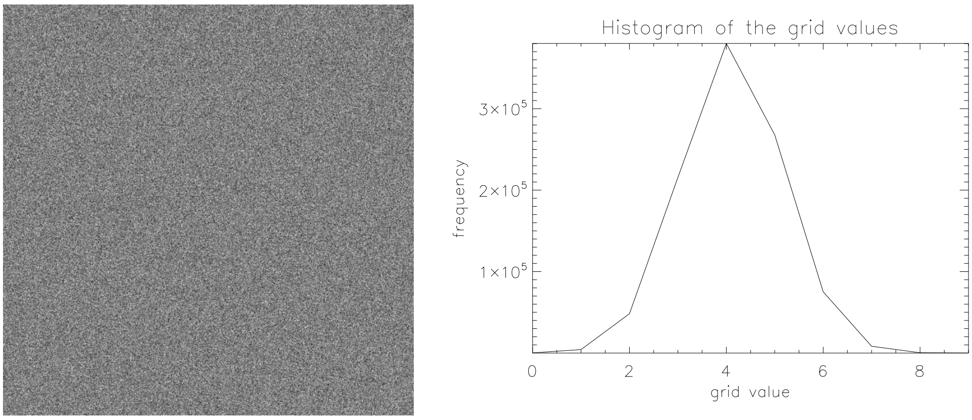
An intelligent way of avoiding the computation intensive Cholesky decomposition is to start out in Fourier space. In real space the pixels have complicated correlations, but in Fourier space there are no correlations between them. It is therefore computationally convenient to generate the correlations in Fourier space. Before continuing, we must first know what Fourier transformations are all about.

3.3.1 Fourier theory

Any function $f(x)$ can be transformed into a sum of many sinusoidal functions in the frequency domain. The transformation results in a Fourier series, whose terms are given different weights according to importance, e.g.

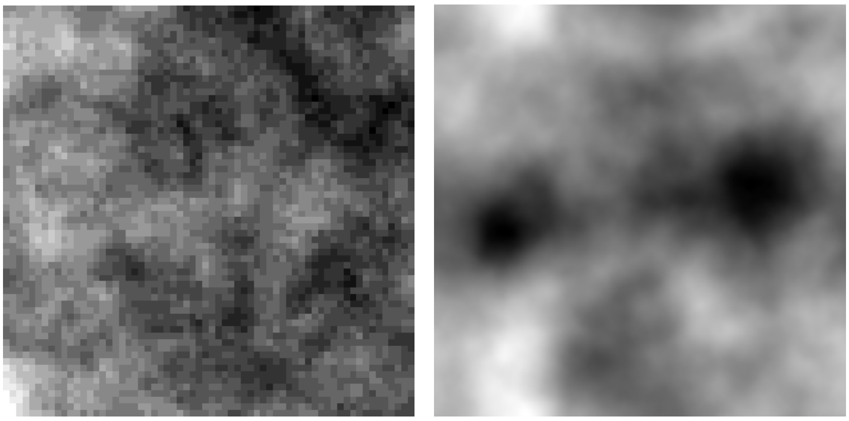
$$f(x) = \sum_k c_k \cos kx + \sum_k d_k \sin kx,$$

[2]If a matrix is symmetric and positive-definite, it can be decomposed into a lower triangle matrix and the transpose of the lower triangle matrix.



(a) **(b)**

Figure 3.2: (a) Visualization of the uncorrelated two dimensional field with 10^6 pixel points. The result is as expected with no sign of structure whatsoever. (b) The histogram of the grid values clearly indicates that the values are Gaussian distributed as they should be.



(a) **(b)**

Figure 3.3: When nearby pixels are correlated, they form structures as shown here. Figure (a) is a visualization of the correlated two dimensional field using Cholesky decomposition (3600 pixels), while figure (b) is a visualization using Fourier transformation (10^6 pixels). Since the latter uses less CPU time than the former, it allows higher resolutions.

where c_k and d_k are the weights. A Fourier transformation is a generalized Fourier series, and is useful when faced with functions on an infinite interval. A Fourier transformation is carried out by finding the Fourier transform of a function $f(x)$:

$$\tilde{f}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-ikx} \mathrm{d}x, \tag{3.4}$$

where its inverse is

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \tilde{f}(k) e^{ikx} \mathrm{d}k.$$

Extending the Fourier transform in equation (3.4) into 2D space, we have

$$\tilde{f}(\mathbf{k}) = \frac{1}{2\pi} \int f(\mathbf{x}) e^{-i\mathbf{kx}} \mathrm{d}^2\mathbf{x}.$$

The frequency space collects the information in the original data in a different manner, which makes it easier to separate signal from noise, allowing measurements otherwise difficult in the spatial domain. One application might be easy removal of unwanted signatures in the signal and, if necessary, inverse transformation back to the original function with the unwanted signatures removed also in the spatial domain. In cosmology, Fourier transformation can simplify our equations significantly. While variables in regular space are dependent on each other, they are not in Fourier space[3]. The dependency is limited to the individual size and orientation of the variable, and thus each mode k in Fourier space may be dealt with individually. For example, consider the temperature fluctuations in the CMB. In the spatial domain, all the different looking fluctuations are located randomly across the sky, where only the distance from other fluctuations determine how they look. When a Fourier transformation is performed, it is possible to keep all fluctuations of a certain mode separate from all the other modes, and information about those kind of fluctuations is more easily measured. The resulting plot of the structures in the frequency domain is known as the power spectrum (see section 3.5).

3.3.2 Fourier transformation of the 2D field

Just like with the field based on Cholesky decomposition, an uncorrelated Gaussian distributed grid is first created, but this time there is no statistical properties through a correlation function. Therefore, the random Gaussian distributed pixels must get a realistic variance. The power spectrum is the variance of the temperature fluctuations, and since inflation predicts a power spectrum that goes like $P(k) \propto k^{-3}$, that value will be used as variance. After the creation of the uncorrelated Gaussian distributed grid, now complex due to being in Fourier space, there is another fact that must be implemented in the program. The spatial domain describing each temperature pixel on the sky map has N values, but when a transformation to harmonic space is made, each pixel will be described by the real and the imaginary part of the complex number, giving $2N$ values. Therefore, there will be some superfluous information, here by half of the matrix being the complex conjugate of the other half. It is easier to see how if a one dimensional example is first considered. The discretized Fourier transform is

$$\tilde{f}_j = \sum_n f(x_n) e^{-i\frac{2\pi}{N}nj}, \quad j = 0, 1, \ldots, N-1. \tag{3.5}$$

[3]This is true only if the variables do not depend on location, that is, if the correlation matrix C_{ij} only depends on $|x - y|$ and not on x and y individually.

Here the wave number has been replaced by $k = \frac{2\pi}{N}j$, j is each pixel point in Fourier space, n is each pixel point in regular space and N is the number of pixels. Using equation (3.5), the relation

$$\tilde{f}_{-j} = \sum_n f(x_n)e^{i\frac{2\pi}{N}nj} = \tilde{f}_j^*$$

is found for $-j$, and the relation

$$\tilde{f}_{N-j} = \sum_n f(x_n)e^{-i\frac{2\pi}{N}n(N-j)} = \sum_n f(x_n)e^{i\frac{2\pi}{N}nj} = \tilde{f}_j^*$$

for $N - j$. In the last relation we have used that

$$e^{-i2\pi n} = cos(2\pi n) - i\sin(2\pi n) = 1$$

according to periodicity. The first relation tells us that $\tilde{f}_0$ is real, and the second relation that all elements have a complex conjugate in the other half of the one dimensional array, except $\tilde{f} = \frac{N}{2}$ which is real. In a similar fashion we get the following relation in a two dimensional grid:

$$\begin{aligned} \tilde{f}(N - j_x, N - j_y) &= \sum_{n_x n_y} f(n_x, n_y)e^{i\frac{2\pi}{N}((N-j_x)n_x+(N-j_y)n_y)} \\ &= \sum_{n_x n_y} f(n_x, n_y)e^{-i\frac{2\pi}{N}(j_x n_x+j_y n_y)} = \tilde{f}^*(j_x, j_y). \end{aligned}$$

Note particularly that

$$\tilde{f}(0,0) = \sum_{n_x n_y} f(n_x, n_y)$$

and

$$\tilde{f}\left(\frac{N}{2}, \frac{N}{2}\right) = \pm \sum_{n_x n_y} f(n_x, n_y)$$

both are real. Finally, after adjusting according to the rules above, we take the inverse fast Fourier transform to get the correlated map. Now that each mode k has the correct signal or amplitude from the start, when transforming "back" to the spatial domain, the temperature fluctuations are correctly distributed too. Figure 3.3(b) shows the resulting map with this method. Using fast Fourier transforms instead decreases the number of operations to only $O(N^2 \ln^2 N)$, which is much more satisfactory than the $O(N^3)$ operations one had when using Cholesky decomposition to make the correlated map.

We have now seen the underlying physics of the correlations between the temperature fluctuations in the background, and how to simulate this process on a sky map. A straightforward method of simulating correlations was first considered, before it was shown how it can be more effectively done by using Fourier space. Summarized, making realizations of CMB maps progresses as follows:

- Create Gaussian distributed data in Fourier space with a variance $P(k)$ that describes the physics of our universe.
- Perform an inverse harmonic transformation of the map.

Now it is time to extend this theory to the spherical domain.

3.4 Simulation on the sphere

3.4.1 Fourier transformation on the sphere

When analyzing the temperature fluctuations, we divide the full sky into grids or pixels, and assign a temperature value T_i to each. The temperature fluctuations have a relative deviation from the mean of

$$\frac{\Delta T(\theta,\phi)}{\langle T\rangle} = \frac{T(\theta,\phi) - \langle T\rangle}{\langle T\rangle},$$

but since the mean $\langle T\rangle$ is zero by definition, the deviation is simply

$$\Delta T(\theta,\phi) = T(\theta,\phi).$$

The temperature fluctuations measured in each and every direction on the sky must be visualized on a spherical surface. The end result of extending the past theory onto the sphere results in

$$T(\theta,\phi) = \sum_{lm} a_{lm} Y_{lm}(\theta,\phi),$$

where Y_{lm} is the spherical harmonic function and a_{lm} is the Fourier coefficients, given by

$$a_{lm} = \int Y_{lm}(\theta,\phi) T(\theta,\phi) \mathrm{d}\cos\theta \mathrm{d}\phi, \tag{3.6}$$

or on discretized forms,

$$T_i = \sum_{lm} a_{lm} Y^i_{lm} \tag{3.7}$$

and

$$a_{lm} = \sum_i T_i Y^i_{lm}. \tag{3.8}$$

The coefficients a_{lm} measure the amplitude or intensity of the spots at each individual scale on the map. When the transformation (3.7) is used, the coefficients are summed over many different spot orientation and sizes, which together give a total description of how the map looks at the pixel i in question. The index l, the multipole moment, describes the size of the spots, while m describes the orientation and location. Note that $l \in [0, l_{\max}]$ and $m \in [-l, l]$. The angular diameter of the spots on the sky map becomes smaller for larger multipoles according to

$$\Delta\theta = \frac{\pi}{l}.$$

Let us now see the connection with the discussion in the last two sections. The equivalent of the Fourier coefficients $\tilde{f}$ is the coefficients a_{lm}, the equivalent of scale k in two dimensions is scale l, and in addition we get the extra variable m in the spherical domain. The temperature pixels T_i are correlated, but the a_{lm} coefficients are not. The fluctuations in the temperature are now generated by drawing a set of a_{lm} coefficients from a Gaussian distribution with variance C_l, the spherical representation of the power spectrum.

3.4.2 Hierarchical Equal Area isoLatitude Pixelisation (HEALpix)

A spherical surface is usually projected onto a two dimensional surface with a method called Mollweide projection. The projection enables you to see the whole sphere in one go, where a familiar analogy is this type of projection of the world map. Pixelating a square surface into equally sized pixels is a simple matter, but splitting up a sphere in the same manner is more problematic. One of the solutions lies in HEALpix (developed by Górski et al. [14, 1]), which is the most used method for representing the CMB data on the sphere in CMB analysis today.

HEALpix divides the pixels into 12 basic diamonds of equal area with center points lying on one of three latitudes, spaced equally from each other (see figure 3.4). Each of these diamonds are infused with smaller diamonds if a higher resolution is desired. The resolution of a HEALpixelated map is defined by the parameter N_{side}, which is the number of pixels along one of the sides of one of the 12 basic pixels. The lowest resolution possible is $N_{\text{side}} = 1$, i.e. one pixel along the side of a basic pixel. The formula for finding the number of pixels in a map of a certain resolution is

$$N_{\text{pix}} = 12N_{\text{side}}^2,$$

since maps with the most basic resolution has 12 diamonds and the number of pixels in each basic diamond is N_{side}^2. This also means that only an N_{side} of 2^n is allowed, where $n = 0, 1, \ldots, \infty$. The resolution limits the details we are able see, so it is no point in using a large l for a small N_{side}. In any case, too large l might give a wrong map out of the a_{lm}'s, since the information about the smallest scales will manifest themselves on the larger scales. For a given N_{side} it is common to use a maximum l in between $2N_{\text{side}}$ and $3N_{\text{side}}$. If the lowest l_{max} in this interval is used, the map can be reconstructed to numerical precision, while the highest can not reconstruct multipoles in the interval completely.

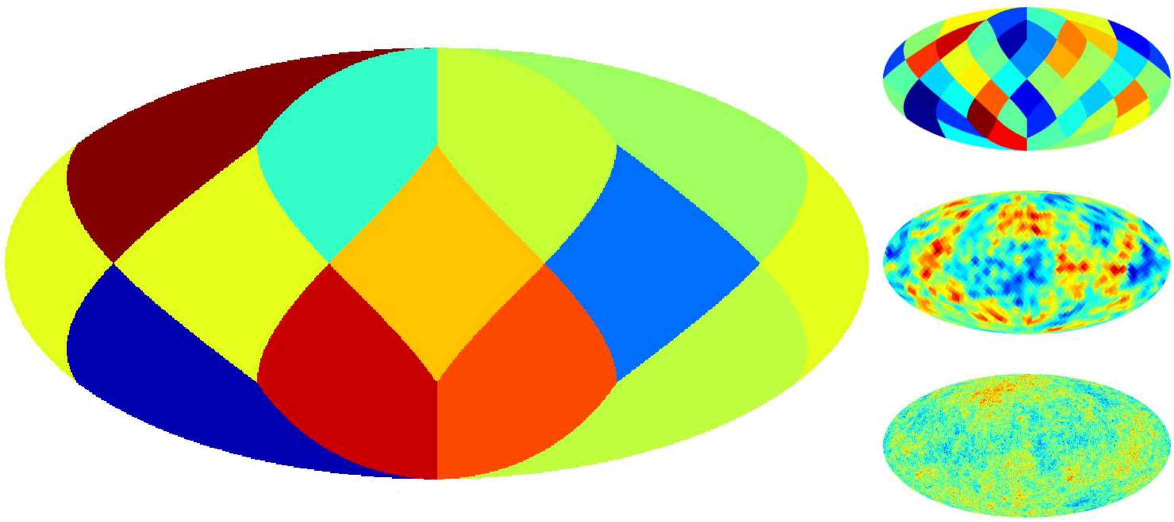


Figure 3.4: Using HEALpix, the Mollweide projected sky map is divided into 12 basic diamonds of equal area. When the resolution is increased, each basic diamonds is split into smaller diamonds as seen on the three small maps to the right. The first with $N_{\text{side}} = 2$, the second with $N_{\text{side}} = 16$ and the third with $N_{\text{side}} = 512$.

3.5 The power spectrum

What one is eventually interested in is the physics of the universe, not where all the spots are located and what form they have. The exact positions of the temperature fluctuations are entirely random, and not dependent on the physics that generated the fluctuations. However, the underlying distribution function is dependent on the cosmological parameters, and therefore it is possible to find the cosmological parameters by looking at the variance

$$\langle a_{lm} a^*_{l'm'} \rangle = \delta_{ll'} \delta_{mm'} \langle C_l \rangle$$

of the spots at each scale. Here δ is the Kronecker delta function, and C_l the power spectrum that gives us this information. To estimate the variance statistic, a sample with many observations is needed, thus the equation for the power at each scale is averaged over all m across the full sky:

$$C_l = \frac{1}{2l+1} \sum_{m=-l}^{l} a_{lm} a^*_{lm}. \tag{3.9}$$

The power spectrum tells on what scale l the signal is strongest, that is on what scale most fluctuations or structure is found. It is common to plot $l(l+1)C_l$ instead of C_l, due to the large influence of the Sachs-Wolfe effect at low l.

We shall now take a closer look at a simulated power spectrum. Using IDL, a program that creates a given number of universes with the Monte Carlo method has been developed. The program uses the `create_alm` function in the HEALPix environment. The output of the function is an array with a_{lm} values up to a defined $l_{\max}$ (i.e. a CMB map) based on the theoretical best fit C_l. Finally, the a_{lm} values are used to calculate a mean power spectrum. The best fit to the first year observational data of WMAP has been used to create the last mentioned power spectrum, together with theoretical calculations and some educated guess, a method called Likelihood analysis.

Figure 3.5 illustrates a number of plots outputted by the program. The first plot shows the mean C_l of the simulated maps on top of the best fit. They compare very well to one another due to the large number of simulations. Another way of checking if the simulations are correct according to observations is used in the second plot, where the best fit power spectrum is subtracted from the mean power spectrum for low l. A curve centered around $y = 0$ means the simulated data matches the theoretical data with some error, and a curve centered at another value than $y = 0$ means the values divert. In our case the curve jumps slightly above and below 0, which means that the data is similar. Note that there are bigger deviations for the smallest l's (large scales). The next three plots show how the power spectrum for a fixed value of l varies for each realization. It is apparent that there are most deviations for large scales. The sixth plot shows an extreme decrease in variance after the very first values of l. All plots tell us that the simulations are less accurate for low values of l, but why is it so? When the scale is large there are less a_{lm} coefficients to average over according to equation (3.9), producing a larger inaccuracy in the final C_l for that scale. This is called *cosmic variance*. The largest scale describes our complete universe, and for that scale there is only one sample available in the distribution describing our universe. As the scales get smaller, there are more and more samples available. The more samples used to describe the fluctuations at a certain scale, the more accurate the simulation at that scale becomes. The last three histograms for three different scales show how a Gaussian shape appears when more samples are available.

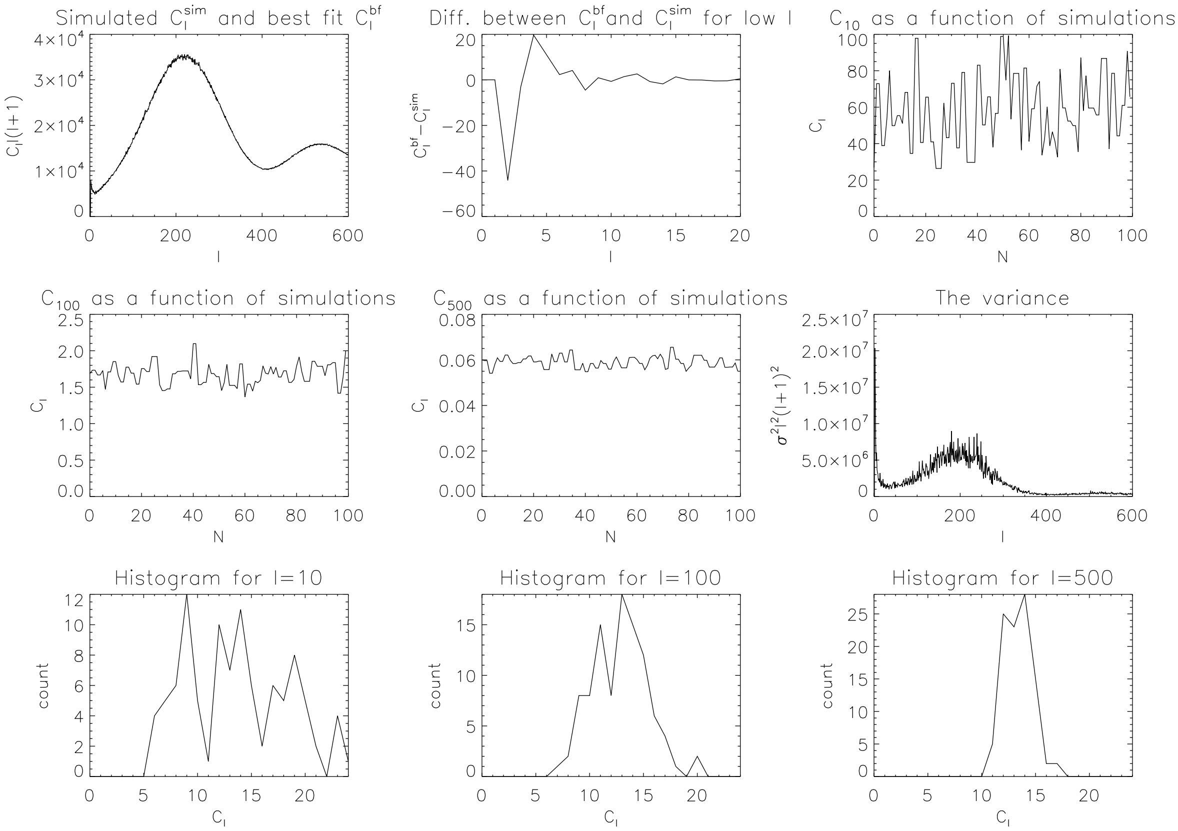


Figure 3.5: The analysis of the mean power spectrum of 100 simulated universes. The first plot is the mean C_l plotted on top of the best fit C_l. In the second plot the best fit power spectrum is subtracted from the mean power spectrum. The third, fourth and fifth plot show the power spectrum values for all simulations at $l = 10$, $l = 100$ and $l = 500$. The sixth plot is the variance of the power spectrum at each scale, and the last three plots are histograms for the three fixed l.

Statistically, this is described by

$$\sigma_{\bar{x}}^2 = \frac{\sigma^2}{n},$$

where σ^2 is the population variance. We see that more samples n is needed to reduce the variance $\sigma_{\bar{x}}^2$ of the sampling distribution.

3.6 Noise, beam and the pixel window

The anisotropy temperature readings are modified by environmental conditions and instrumental properties. Noise on the sky map may be caused by strains on the instrument or by other sources of noise like heat and atmospheric noise (for ground-based instruments). Instrumental limitations are also important to consider, like the limits to the detail level we are able to see in the CMB.

3.6.1 Noise

If the background radiation of each pixel i is s_i, then the noise is simply added to this value to get the measured temperature value:

$$T_i = s_i + n_i.$$

The noise n is Gaussian distributed with a mean value of $\langle n \rangle = 0$, and a variance $\langle n^2 \rangle = \sigma^2$. It is a good approximation to assume ideal noise, which has no correlations between each noise pixel[4], giving the noise correlation matrix

$$\mathbf{N} = \langle n_i n_j \rangle = \delta_{ij}\sigma^2 \tag{3.10}$$

between pixel i and j, where δ_{ij} is the Kronecker delta function. As noted in the last section, the variance of Gaussian distributed data may be reduced by collecting many observations, if the data is changing from each observation. We are interested in finding the effective variance $\langle n_{\text{eff}}^2 \rangle = \sigma^2$ to back up this fact. When the noise of all the observations is summed up, we get the effective noise

$$n_{\text{eff}} = \frac{1}{N}\sum_i n_i.$$

Squaring the effective noise yields

$$\langle n_{\text{eff}}^2 \rangle = \frac{1}{N^2}\sum_i \sum_j \langle n_i n_j \rangle,$$

which becomes

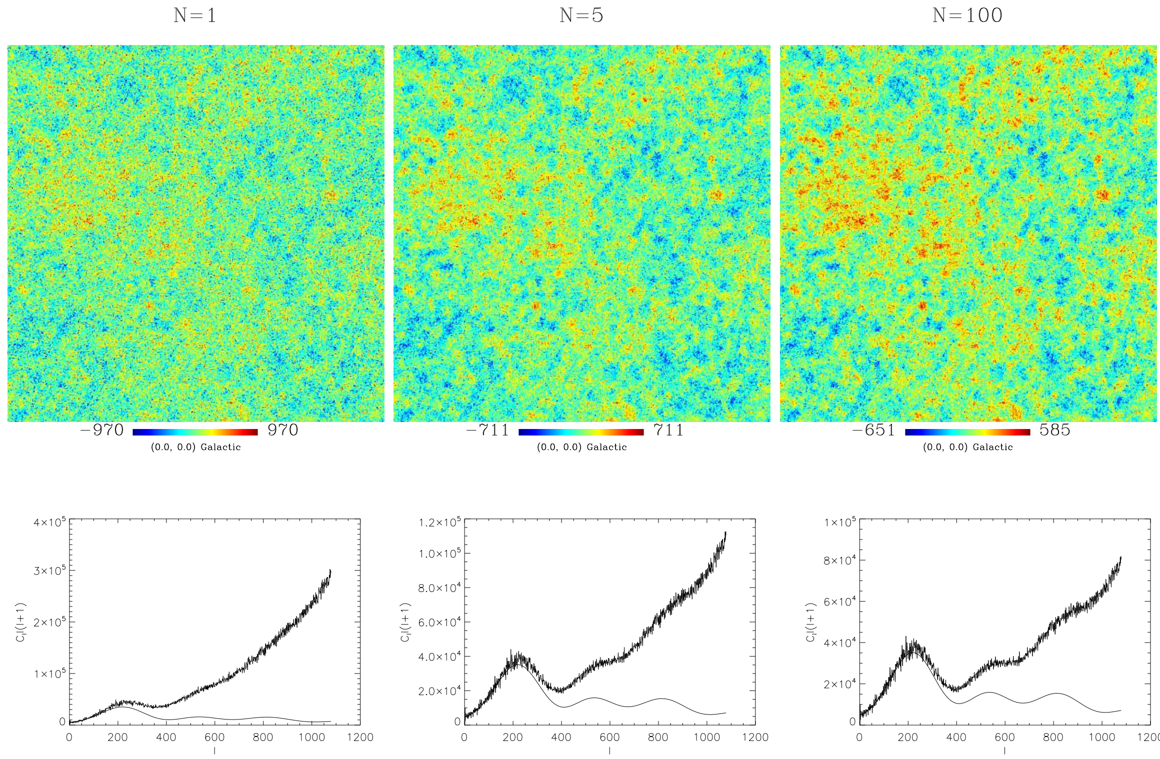
$$\sigma_{\text{eff}} = \frac{\sigma^2}{N^2}\sum_{ij} \delta_{ij} \tag{3.11}$$

[4]Some experiments might induce larger correlations between noise components than others. The WMAP and Planck satellites collect data in different ways, such that the noise correlations in the Planck data will be larger than the correlations in the WMAP data. The noise correlations in the WMAP data can be ignored due to the high signal-to-noise ratio at low l and insignificant effect at high l [15]. For polarization, however, the signal-to-noise ratio is much smaller, and correlations must be taken into account also for WMAP.

when recalling from equation (3.10) that each noise pixel is not correlated. Equation (3.11) can now be reduced to

$$\sigma_{\text{eff}} = \frac{\sigma^2}{N},$$

showing that the total variance is decreased the larger N is. For simplicity the same noise variance σ has been used for all pixels, but in reality this is not true. Consider WMAP orbiting around the second Lagrangian point. The variance changes according to distance from the sun, but also when the satellite crosses some of the same points from earlier orbits. The latter is an additional observation, and reduces the variance significantly. Figure 3.6 shows the difference between a mean map created after 1, 5 and 100 observations, visualizing what was deduced in the last paragraph.



(a) N=1 **(b)** N=5 **(c)** N=100

Figure 3.6: Three sectional maps of $N_{\text{side}} = 512$ with noise of constant standard deviation $\sigma = 100$. The effect of noise can also be seen on the power spectrum (the best fit C_l is the solid line), where the small angular scales have more power. When the number of simulations is increased, less noise is observed on the map and the amplitude of the power spectrum decreases.

3.6.2 Beam

The telescope cannot be pointed at just one single point, but observes a finite solid angle at a time. This is the instrumental beam, and limits the resolution it is possible to obtain with the telescope. In the case of radio telescopes the beam function often has a near Gaussian profile, and its width is defined by the full-width at half maximum[5] of the beam function. The beam causes each pixel on the sky map to smear outwards in all directions from the pixel, canceling out the small scales. The measured temperature value with the beam function B is

$$T_i^{\text{beam}} = T_i B,$$

or for all pixels,

$$T^{\text{beam}}(\theta, \phi) = \int T(\theta', \phi') B\left(|(\theta, \phi)| - |(\theta', \phi')|\right) \mathrm{d}\cos\theta \mathrm{d}\phi. \tag{3.12}$$

Like noise, the beam should be incorporated in our program, so that the simulated data includes more of the effects observations would produce. Adding the beam numerically pixel by pixel like in equation (3.12) is computationally very heavy, and it is thus reasonable to generate a_{lm} coefficients containing the beam. The coefficients for a beam with a Gaussian profile is

$$a_{lm}^B = a_{lm} \exp\left(-l(l+1)\frac{\sigma^2}{2}\right), \tag{3.13}$$

where

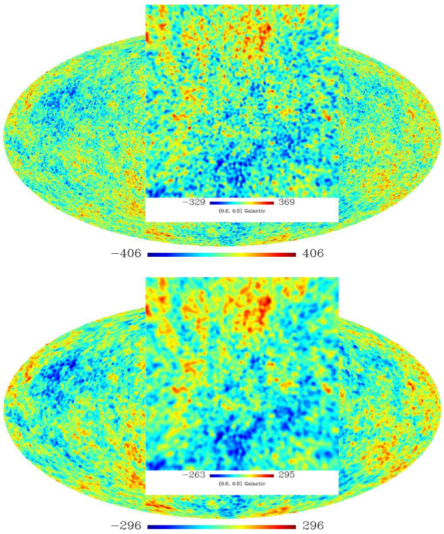
$$\sigma^2 = \frac{\text{FWHM}^2}{8\ln 2}$$

is the width of the beam. It is important to avoid a pixel size that surpasses the beam size, since information that can be resolved will be lost. The instruments in WMAP have a beam size of $14'$, allowing a minimum pixel size of 1.6×10^{-5} radians. This would hold 785399 pixels on a sphere, corresponding to a map with $N_{\text{side}} = 256$. However, the size of the beam is measured at its FWHM, which means that the beam is actually a bit larger than $14'$. Therefore $N_{\text{side}} = 512$ would be more appropriate. As already noted, it is the beam that limits the scales that it is possible to resolve, thus using a resolution any larger than this would make computations heavier than necessary. Figure 3.7 and 3.8 show the effect of a beam for $N_{\text{side}} = 128$ and $N_{\text{side}} = 512$ using this formula, accompanied with an illustration of how the power spectrum changes after adding a beam.

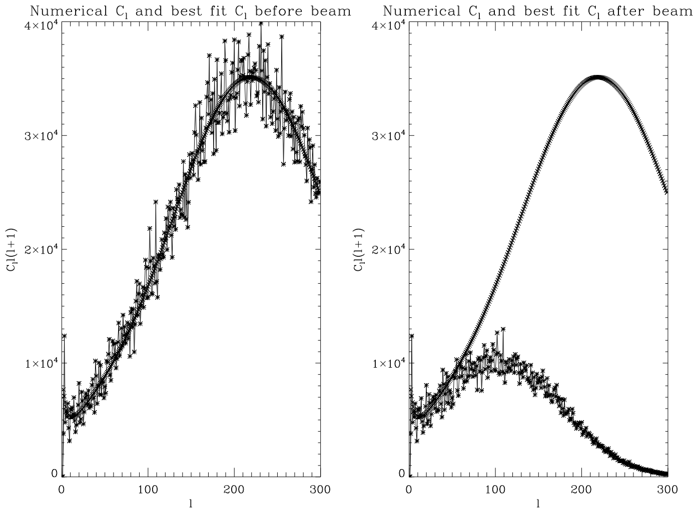
3.6.3 The pixel window

Information about the smallest scales is forfeit when the information contained in each a_{lm} coefficient is converted to a pixel on the sky map. Each pixel can only be one-valued, so the temperature is averaged for each pixel. This is of concern if we want to convert the map back to a_{lm} coefficients, where we would have observed the power spectrum converging towards zero at large l. Infinite pixels would have demanded infinite computing power and hard drive space, making it desirable using the so called pixel window function $W_l(N_{\text{side}})$ to solve the problem. A window function is a form of filter, and has similar effects on the map as a beam. The window function is zero outside a given interval in real space, and when another function is multiplied with it, only the parts defined within the interval gets through, hence the window name. The pixel window function applies a controlled smoothing to each pixel. The factor must be applied to either the a_{lm} coefficients or the power spectrum C_l before the conversion commences, and must be inversely applied again after converting back from the map to get

[5]Full-width at half maximum (FWHM) is the distance from half the maximum at one side of the maximum of a Gaussian function to half the maximum at the other side.

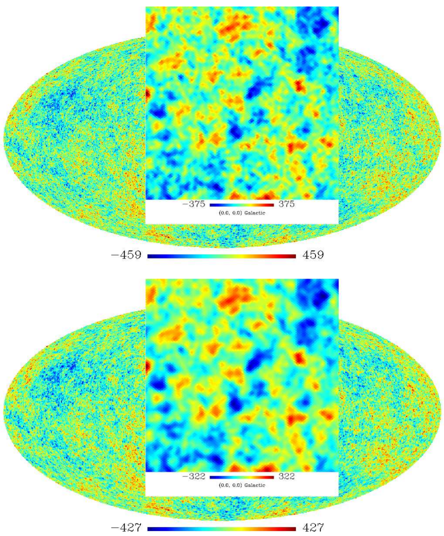


(a) Maps

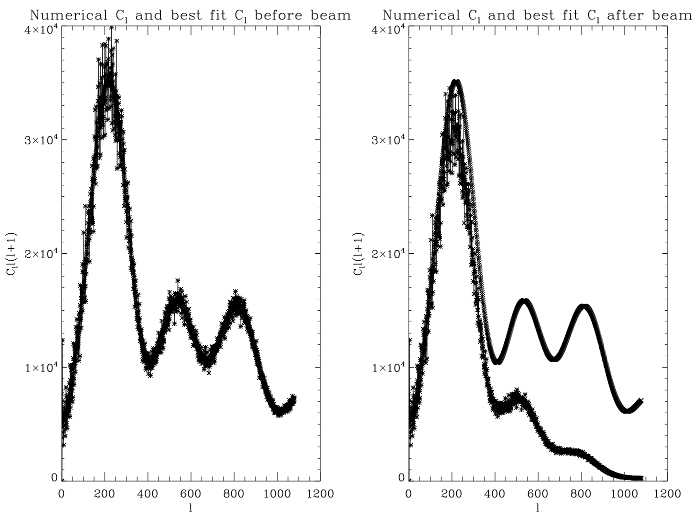


(b) C_l plot

Figure 3.7: The effect of an artificial beam of 1° on a sky map with resolution $N_{\text{side}} = 128$. The upper sky map has no beam added, while the lower has beam added, and a noticeable smooth out is visible. When the structures are smoothed out, the smallest scales gets wiped out. The rightmost plot illustrates how the power spectrum, due to this fact, is degraded for large l.



(a) Maps



(b) C_l plot

Figure 3.8: When the beam is smaller, it is possible to resolve smaller scales. This sky map with a resolution of $N_{\text{side}} = 512$ has been artificially applied a small beam of $14'$. See figure 3.7 for details.

the original a_{lm}'s back again. The pixel window function is a function of N_{side}, since we need less smoothing for higher resolutions, which can contain more pixels with more information about the temperatures.

3.7 The foregrounds

The best parts of the frequency spectrum to observe the background is the area where it dominates, namely the range 30 to 150 GHz, as pictured in figure 3.9. Both COBE and WMAP observed the sky within this range; COBE centered at three frequencies (31.5, 53 and 90 GHz) and WMAP centered at five (23, 33, 41, 61 and 94 GHz)[6] [2]. Unfortunately there are many other sources in the universe emitting radiation at the same frequencies as the CMB. The objects are all in between us and the background, and contaminate it with alien signals[7]. This has caused an extra headache for cosmologists, and great effort has been put in filtering out the signals. Ways of dealing with the challenge is to observe at frequencies and at sky locations with less contamination. But these are not final solutions when lots of information is overlooked, and after all, no frequency nor location comes contamination free. Some pixels can be used for information retrieval after foreground reduction techniques have been applied to the map, but for other pixels this will not work.

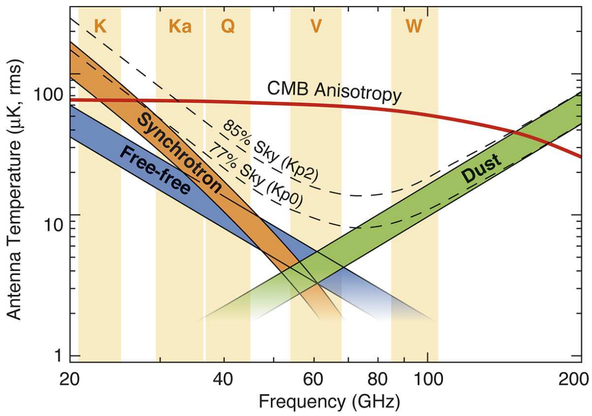


Figure 3.9: Even the frequency domain where the CMB predominates is not free from foreground emission (the emission shown here mainly originates from normal galaxies). The five channels of WMAP are marked at the top [5].

The spectra of the foreground emission sources are different from that of the CMB, which makes it possible to use observations from several frequencies to extract the background emission from the signal. Observations of the five separated frequencies of the WMAP mission was used to determine what signals belonged to CMB and what signals came from other sources. However, some sky locations are so full of contamination that foreground reduction cannot be performed satisfactory, and retrieving information from the CMB in these locations is hopeless. This class of contaminants are extragalactic or originates from the Milky Way galaxy.

[6]Planck will measure the full sky in nine frequency bands, centered at 30, 44, 70, 100, 143, 217, 353, 545 and 857 GHz. This will enable a full sky map of all anisotropies over all frequency channels [3].

[7]The foregrounds are also a very important source of information for cosmologists, although, in a CMB analysis context, the foreground is unwanted.

The temperature contribution from these points are dealt with by applying a mask on the affected pixels, so that they do not impact the power spectrum. A mask is simply a map with zeroes for the pixels to be removed and ones for those we do not want to remove (see figure 3.10). When pixels are removed their contribution to the total amplitude on all scales is reduced. This is illustrated for three maps with very large masks and their power spectrum in figure 3.11, where the largest reduction occur for the largest mask. Most contamination comes from the Milky Way galaxy, and this is where the largest mask, the galactic cut, is found on maps from observations. Finding the cut-off point for the galactic cut is not an easy task as the galactic plane has decreasing intensity the further you get from its center. Here one must achieve a fine line between filtering out the galactic contamination and keeping as many pixels belonging to the CMB as possible. It is common to make a set of different mask maps where different intensity cut-off points have been chosen, so that the level of cut-off can be easily adapted to the analysis performed.

The microwave emission from the galactic plane has many origins. The most prominent are free-free emission (bremsstrahlung), synchrotron emission and dust radiation. Free-free and synchrotron emission is prominent in the lower part of the spectrum, and emission from dust in the upper part, as can be seen from figure 3.9. The free-free emission is caused by free electrons being slowed down by ions, such that a photon is emitted. Synchrotron emission originates from cosmic ray electrons in supernova remnants or from diffuse electrons around the galaxy, where ultrarelativistic electrons are caught spiraling in a magnetic field, emitting photons while being accelerated in the field. The spectrum from synchrotron emission varies according to several factors (such as number density, magnetic field strength, energy loss etc.), and the effect will therefore vary greatly according to the frequency at which the CMB is observed. Thermal emission from dust also originate from locations with star formation processes, and as such, can be found in many of the same areas as synchrotron radiation. Thermal emission radiates significantly in the infrared and the microwave part of the spectrum, but dust can also emit microwave photons due to spin and thermal fluctuations.

Since many of the extragalactic sources are also galaxies, we can expect some of the same radiation from these as from the galactic plane. However, this group of contaminants contains mostly radio sources like radio galaxies and quasars, rather than normal galaxies like the Milky Way galaxy. These extragalactic sources are *point sources* that occupy just a few pixels on the sky map, but there are also larger extragalactic sources, like the Coma Cluster. The Coma Cluster is a source of the Sunyaev-Zeldovich effect. When a CMB photon passes the hot gas in the galaxy cluster, it Compton scatters off the hot electrons, causing a frequency shift that lowers the temperature readings. The next chapter will discuss how the point source contaminants are detected and removed.

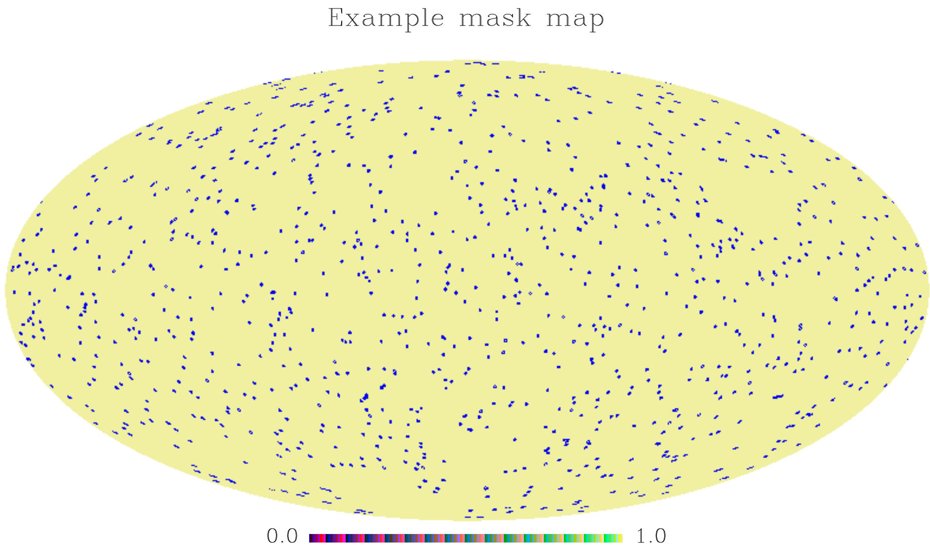
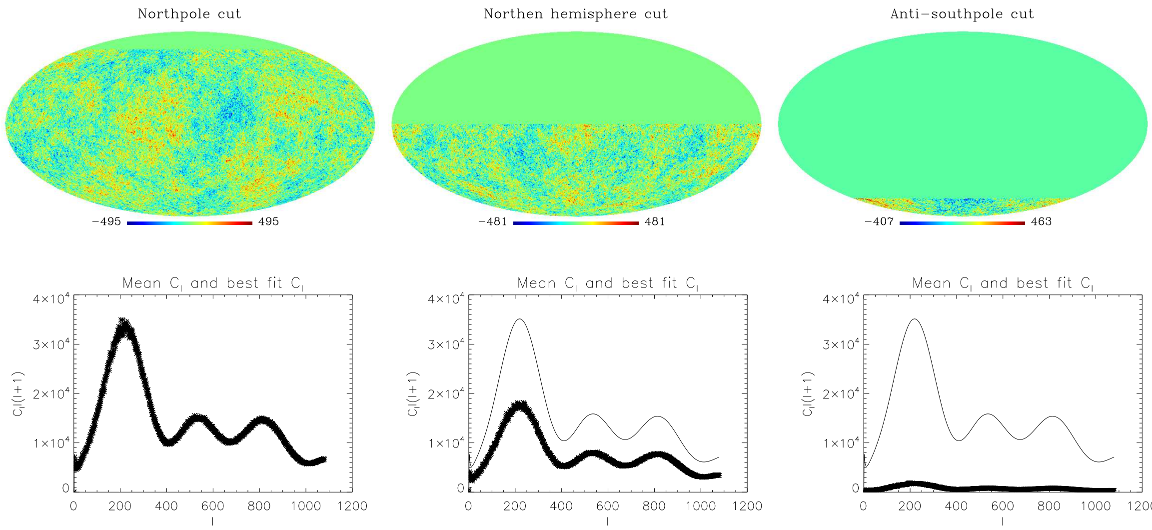


Figure 3.10: An example mask map with a value of zero at the contaminated pixels.



(a) Northpole cut **(b)** Northern hemisphere cut **(c)** Anti-southpole cut

Figure 3.11: Three maps with three different mask sizes, show how the power spectrum is reduced when less pixels contribute to its amplitude.

Chapter 4

Point source detection and the wavelet technique

Before masks can be applied on the point sources, they must be located. The point sources occupy just a few pixels, are scattered randomly across the sky map and are mostly found at the smaller angular scales. Detecting the point sources in the temperature data has become a more and more important part of CMB analysis. The COBE had too low resolution to resolve any point sources, while the WMAP data was contaminated by a few. Planck will resolve an even greater amount, calling for methods with the ability to locate point sources more efficiently than the methods used on the WMAP data.

The point sources can be detected by determining their flux compared to the CMB directly from the sky map as is, an approach used on the WMAP data. However, if the map is applied with wavelets, the flux of the point sources can be enhanced so that they stand out from the CMB. There are two types of point sources, resolved and unresolved, and different procedures are used to detect them. This chapter will explain how point source detection is performed for both the resolved and the unresolved point sources, and what the wavelet technique is and how it is used.

4.1 Point source detection

4.1.1 Resolved point sources

The resolved point sources can be filtered by looking at their total flux, which is generally larger than the flux of the CMB. The method involves using a fixed σ limit, i.e. a factor of the standard deviation of the data. When filtering out the point sources using this method, there will always be some pixels falsely identified as point sources, as some pixels of the background do have a larger flux. According to Gaussian statistics, 68 % of the observations x happen to have a deviation less than $\pm\sigma$ from the mean $\langle x \rangle$, 95 % of the data the values $|x - \langle x \rangle| < 2\sigma$, 99.7 % the values $|x - \langle x \rangle| < 3\sigma$, and so forth. The flux of the point sources does not fall on the lower tail of the temperature distribution. Therefore, if 3σ is used as filtering limit, 0.15 % of the data outside the limit will belong to the CMB. In other words, if an N_{side} of 64 is used, ~ 74 out of 49152 pixels will be identified as point sources and removed even though they are not all point sources. Figure 4.1 illustrates this using a plot of the flux at each pixel for a small sky map containing three point sources. The number of falsely identified point sources

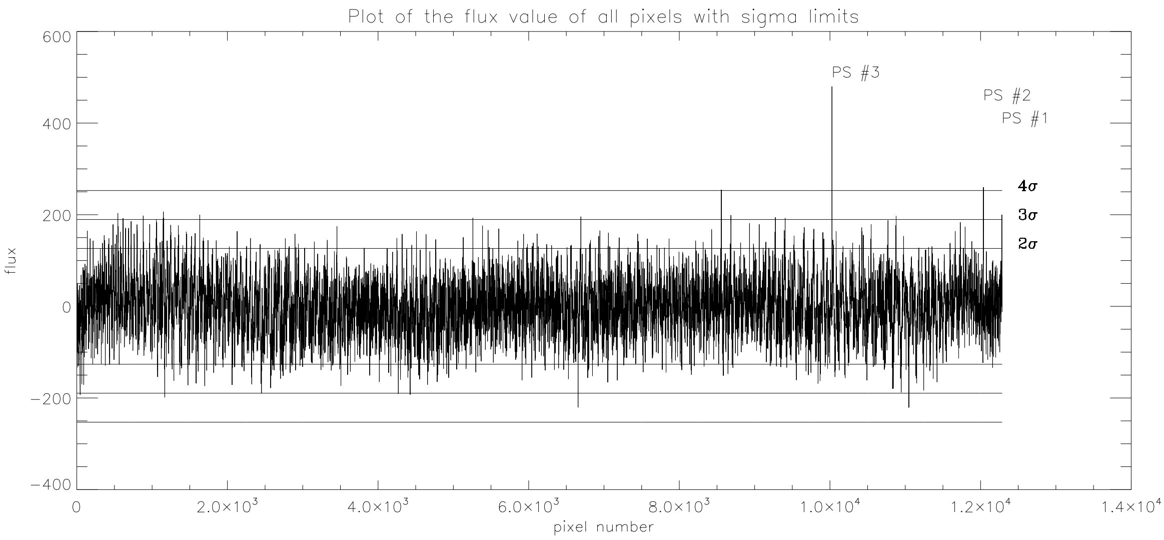


Figure 4.1: The flux values at each pixel for a small sky map with just 12286 pixels, i.e. $N_{\text{side}} = 32$ (note that the small sky map is used for illustrative purposes), where 3 point sources have been added (see marked pixels). The largest threshold has localized two point sources, while a third is considered to be coming from the CMB. A threshold of 3σ localizes all point sources, however, a handful of pixels originating from the CMB is falsely identified as point sources.

must be kept as low as possible, and the total number of located true point sources as high as possible. If, on the one hand, too many false point sources are removed, the background at these points is ignored, and the cosmological parameters will get larger uncertainties. On the other hand, if too few true point sources are removed, the estimation of the cosmological parameters will be wrong. The last section in the previous chapter revealed that observations of the CMB is performed on several frequency channels. Since the flux of the point sources vary more between each channel than the flux of the CMB, the point sources can be separated more easily from the CMB by performing detections on some or all of the channels. The problem with false point sources is therefore near negligible when analyzing real CMB data. In this thesis, however, only one channel has been used, and thus the false point sources must be taken into account.

A quite similar approach to the above was used to detect point sources in the WMAP data. In the WMAP analysis, a list was first made of all pixels larger than a flux limit of 5σ in all frequency bands WMAP operates. If any of these pixels were found in any other band with a flux larger than 2σ, they were also added for that band in the list. After detection of the resolved point sources from the data, the detections were cross-checked against existing catalogues of known radio and infrared sources. If the angular position of the point sources in the WMAP list corresponded well with the sources from the surveys (within two times the beam size), they were identified as true point sources. The first year analysis of the WMAP data revealed 208 point sources [5]. After cross-checking with the catalogues, 203 of the detected point sources had counterparts, where all of the counterparts were found in the catalogues of radio sources. Since the remaining 5 point sources were found close to the 5σ threshold, they were assumed to be spurious, which was consistent with the predictions made by Gaussian statistics. Thus no point sources were found without a known counterpart.

According to a later paper [23], the 203 sources were found to be 141 quasars, 42 galaxies or active galactic nuclei, 19 BL Lac-type objects and the IC418 nebula. After more accurate results had been obtained when three and five years of observations had been concluded, the number of resolved point sources was adjusted to 323 [15] and 390 [24] respectively.

4.1.2 Unresolved point sources

Point sources that are not above the defined standard deviation threshold of the CMB are called unresolved point sources, and have a flux near the peak flux of the CMB. Finding these point sources is more tricky, but one can use the statistical properties of the data to get knowledge about their amplitudes. The CMB is Gaussian distributed, but the point sources destroy the Gaussian distribution. This deviation from Gaussianity can be measured by using the third and fourth order moments of the distribution, known as skewness and kurtosis. It is not possible to tell which pixels are contaminated by point sources, but one is be able to know the amount of point sources or their amplitudes. When a model for the amount of point sources or the amplitudes of the point sources has been determined, one can compensate for the deviation, and thereby get a more correct power spectrum.

The skewness statistic is a measure of how asymmetric the data is, and is defined as

$$S' = \frac{1}{N\sigma^3}\sum_i (T_i - \langle T\rangle)^3,$$

where the variance is defined by

$$\sigma'^2 = \frac{1}{N}\sum_i (T_i - \langle T\rangle)^2.$$

Positive skewness signifies a distribution with a tail towards the right, and negative skewness signifies a distribution with a tail towards the left. Gaussian distributed data has no skew, and naturally the skewness for such data is 0. Kurtosis is a measure of the peakedness of the data, and is defined as

$$K' = \frac{1}{N\sigma^4}\sum_i (T_i - \langle T\rangle)^4.$$

Positive kurtosis signifies that the data quickly falls off from its peak value (sharp peak), and negative kurtosis signifies that the data falls off slowly from its peak value (flat peak). The temperature mean is zero, so the above equations become a bit simplified:

$$S = \frac{1}{N\sigma^3}\sum_i T_i^3 \tag{4.1}$$

$$K = \frac{1}{N\sigma^4}\sum_i T_i^4 - 3, \tag{4.2}$$

where

$$\sigma^2 = \frac{1}{N}\sum_i T_i^2. \tag{4.3}$$

The kurtosis for Gaussian distributed data is 3. Therefore, a normalization factor has been included in the last term of the equation for kurtosis, so that the relative deviation from the Gaussian distribution is illustrated more conveniently.

The deviation of the power spectrum due to the amplitudes of the unresolved point sources is found by performing a χ^2 minimization. The χ^2 minimization determines how well the data x_i from n observations, with standard deviation σ_i, fit a given model m_i, and is given by

$$\chi^2 = \sum_{i=1}^{n} \left[\frac{x_i - m_i}{\sigma_i} \right]^2 .$$

The accuracy of the χ^2 minimization is better if many statistical moments are used. In this thesis, the skewness and kurtosis for a given amplitude A are the models, and the skewness and kurtosis for an unknown amplitude are the data. The χ^2 statistic is then:

$$\chi^2(A) = \sum_{i=1}^{n} \left[\frac{S_i^{\text{obs}} - S_i^{\text{mod}}(A)}{\sigma_{S,i}} \right]^2 + \sum_{i=1}^{n} \left[\frac{K_i^{\text{obs}} - K_i^{\text{mod}}(A)}{\sigma_{K,i}} \right]^2 . \tag{4.4}$$

The amplitude where χ^2 is at its minimum is the amplitude that fits the data best. Finally, the power spectrum of the model with this amplitude is subtracted from the modelled power spectrum with no amplitude, and the correction applied to the observed power spectrum.

4.2 Wavelets

4.2.1 Wavelet theory

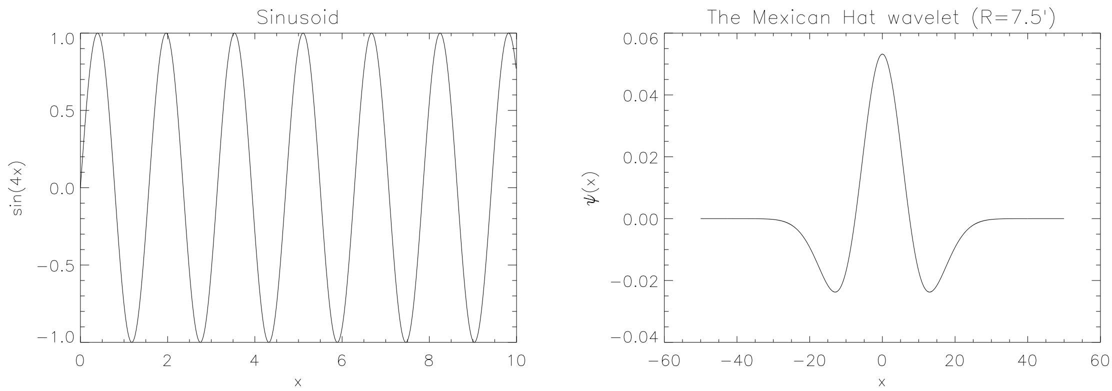
The first work on wavelet theory was done in France during the 1980's [12, 7, 17], and was first used in CMB analysis by Forni and Aghanim in 1999 [11], who developed statistical tools for finding non-Gaussianity in signals and later applied it in a cosmological context [4]. This recent research show that attempting to search for non-Gaussianity in regular space does not give as good results as going through wavelet space. In the isotropic field of the background, isotropic non-Gaussian signals are not better to look for in wavelet space, but for non-isotropic signals, like point sources, wavelets give better results. The idea is to use wavelets to enhance the scales where the point sources are located, and in that way find more of them. Since the wavelet transformation is a linear process, the Gaussian properties of the CMB are preserved. Before we can continue applying this method for our use, it is necessary with a basic understanding of what wavelet analysis really is.

Wavelet analysis compares well to Fourier analysis. In Fourier analysis, a function or a signal of a continuous variable (usually time or space) is represented as a sum of waves localized in frequency space or harmonic space. In wavelet analysis, the wave is replaced with a wavelet[1]. While a wave oscillates with an amplitude distributed over all points, the wavelet's energy is concentrated around one point (illustrated in figure 4.2), and contrary to a Fourier series, a wavelet enjoys good localization properties in both the original space and in frequency space at the same time. Wavelets are highly configurable, and are scaled and translated through a mother wavelet. This is the one of the main sets of characteristics defining a wavelet. The mother wavelet $\psi(\mathbf{x})$ is defined through the continuous isotropic wavelet transform of a signal $f(\mathbf{x})$, given by

$$w(R, \mathbf{b}) = \int \mathrm{d}\mathbf{x} f(\mathbf{x}) \frac{1}{R} \psi \left(\frac{|\mathbf{x} - \mathbf{b}|}{R} \right) .$$

[1] The word wavelet is in fact a light mix between English and French, where the last syllable means small, i.e. wavelet means a small wave.

The isotropic wavelet is spherically symmetric, and can therefore be used anywhere on the sphere without changing its properties. The variable **b** translates the wavelet, making it possible to change the position we want to look at, while R scales (not to be confused with the scale l) the wavelet, which can be varied to change the detail level or resolution of a certain position. This feature makes wavelet analysis useful for non-periodic phenomena and local events, such as the localized point sources and their lack of following a pattern on the sky map.



(a) Sinus wave **(b)** Mexican hat wavelet

Figure 4.2: Waves and wavelets in real space.

4.2.2 The Spherical Mexican Hat Wavelet and Spherical Needlets

This thesis will compare the efficency of two types of wavelets, the Spherical Mexican Hat wavelet (SMH wavelet) and Spherical needlets. Previously only the SMH wavelet and Sphercial Haar wavelets have been used in the search for non-Gaussianity. Due to the isotropic features on the sphere and good performance of the SMH wavelet over the Haar wavelet, the SMH wavelet was until recently considered to be the better wavelet for detecting non-Gaussianity [6, 19], but needlets are showing promising additional features that make them a possible candidate for succession [18]. The mathematical details of the SMH wavelets will now be briefly examined. The mathematical groundwork of needlets, though, is more complex, but is thoroughly derived in the paper by Marinucci et al. [18].

The SMH wavelet is a spherical version of the more known flat Mexican Hat wavelet (pictured in figure 4.2(b)):

$$\psi(x;R) = \frac{1}{R\sqrt{2\pi}}\left(2 - \frac{x^2}{R^2}\right)\exp\left(-\frac{1}{2}\frac{x^2}{R^2}\right). \tag{4.5}$$

Since we are dealing with spherical data, extending the Mexican Hat wavelet to the sphere is necessary, but this has been a hard issue to solve. In 1998, Antoine and Vanderheynst proposed using a stereographic projection on the sphere [19], a method that conserves the basic properties of the Mexican Hat wavelet. The projection gives the SMH wavelet

$$\Psi(\theta; R) = N\frac{4}{(1+\cos\theta)^2}\psi\left(\frac{x}{R} = \frac{2}{R}\tan\frac{\theta}{2}\right),$$

where

$$N = \frac{1}{R}\left(1 + \frac{R^2}{2} + \frac{R^4}{4}\right)^{-1/2}$$

is the normalization constant and $\psi(x)$ is the flat Mexican Hat wavelet defined in equation (4.5). When the theory for the continuous Spherical Mexican Hat wavelet is discretized, it is deprived of some of the properties of a wavelet. One of the disadvantages is that we no longer can inversely transform the sky map after the wavelet has been applied. However, this is a handicap for some applications, but not for locating the point sources. While other wavelets used for point source detection have to be stereographically projected onto the sphere, the needlets are by definition spherical, and can make use of the properties of a sphere. The disadvantage noted above therefore does not apply to needlets. Needlets are also more friendly towards numerical adaptation, since the starting point is in the spherical domain.

The SMH wavelet is defined for a large range of scales, but is more localized on some scales in multipole space than on others. In this way, information of other scales are not lost, though it is harder to find the point sources on the scales where the wavelet is less localized. Needlets, on the other hand, have an exact localization on all scales they are defined. Needlets are more scalable than the SMH wavelet, since they consist of a wide range of mother wavelets. The SMH wavelets only have one underlying mother wavelet. In multipole space the SMH wavelet is therefore defined by one variable R, which scales the wavelet, and needlets defined by two variables, one variable a that defines how the mother wavelet looks and one variable j that scales it (see figure 4.3). Translation occur when the wavelet is applied to different locations on the sphere. There is a relation between the two variables a and j for needlets, and the multipole l of which the wavelets are defined. The lower limit of this range is

$$l_{\min} = a^{j-1},$$

and the higher limit is

$$l_{\max} = a^{j+1}, \tag{4.6}$$

where $a > 1$. The smaller this range is, that is when a is small, the more sharply the wavelet is localized in spherical harmonic space, but at the same time it becomes harder to localize it in spherical space. The exact opposite case applies for a large range or when a is large, where the wavelet is sharply localized on the sphere, while being harder to localize in spherical harmonic space. The phenomenon is easier to understand when considering its analogy to Heisenberg's uncertainty principle[2].

A wavelet can be compared to the effect of a beam of a certain scale on the map, though wavelets are artificially applied to the map for filtering out different scales. Because of this similarity, the a_{lm} coefficients after the wavelet has been applied is calculated in the same manner as in equation (3.13), however, the Gaussian beam is replaced by the multipole representation of the wavelet g_l^s of scale s:

$$a_{lm}^{\text{wavelet}} = a_{lm} g_l^s. \tag{4.7}$$

[2]Heisenberg's uncertainty principle states that when a particle is sharply localized in space, it is less localized in momentum space. When it is sharply localized in momentum space, it is harder to locate in momentum space.

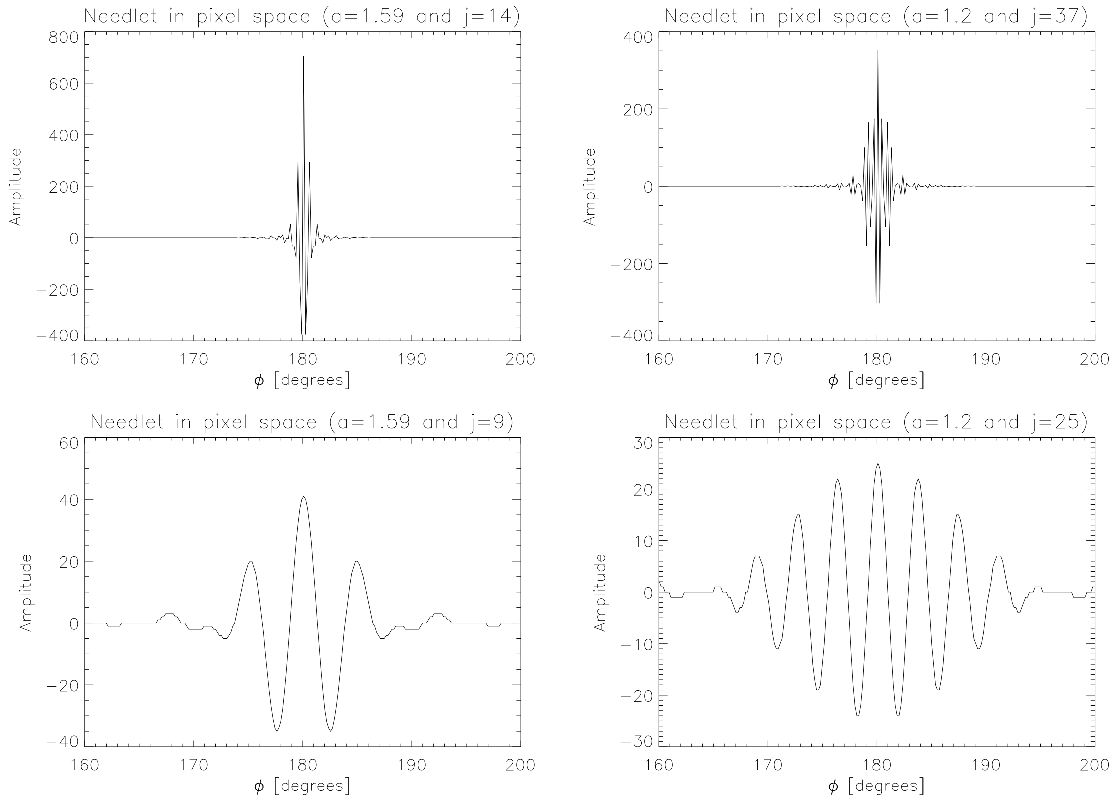


Figure 4.3: Needlets in pixel space. The value of j determines the scale of the wavelets. For $a = 1.59$ of scale $j = 14$ (top left panel) and $a = 1.2$ of scale $j = 37$ (top right panel), j is a high value, so the wavelets are small. For $a = 1.59$ of scale $j = 9$ (bottom left panel) and $a = 1.2$ of scale $j = 25$ (bottom right panel), j is intermediate, so the wavelets are larger. In harmonic space, from top left to bottom right, the needlets are defined for $l \in [415, 1049]$, $l \in [709, 1021]$, $l \in [41, 103]$ and $l \in [79, 114]$

In figure 4.5(a) and 4.6(a), an SMH wavelet with $R = 7.5'$ and a needlet with $a = 1.59$ and $j = 14$ have been plotted in harmonic space. When the wavelets are applied to the power spectrum in figure 4.4, the interesting scales are enhanced and the uninteresting are damped. The power spectrum after the SMH wavelet transformation is shown in figure 4.5(b), where the largest scales have been thoroughly damped, but not completely. The SMH wavelet is known for this "leakage" at large scales [18], but this does not constitute a problem for needlets, as seen in figure 4.6(b). The large scales are affected by cosmic variance and make a greater contribution to the temperature, thus the point sources may be more notable with needlets. Furthermore, a large peak can be seen at smaller scales, but the needlet transformed power spectrum is more localized in harmonic space than the SMH wavelet transformed power spectrum.

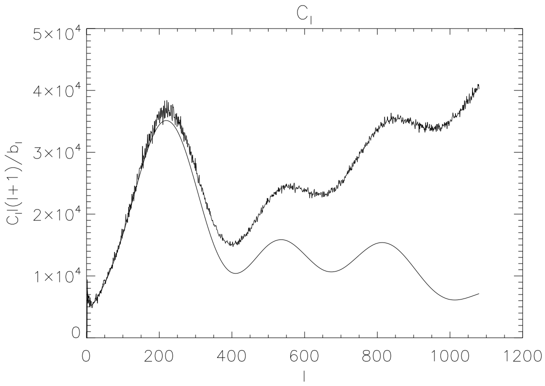
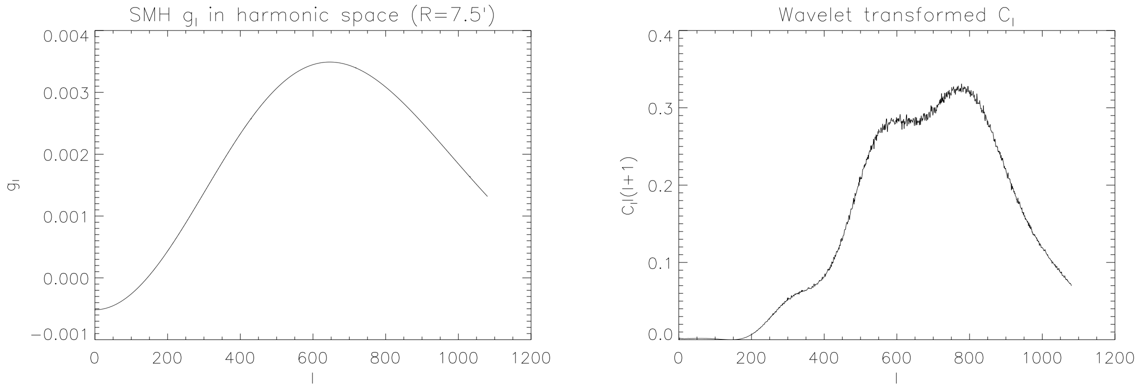


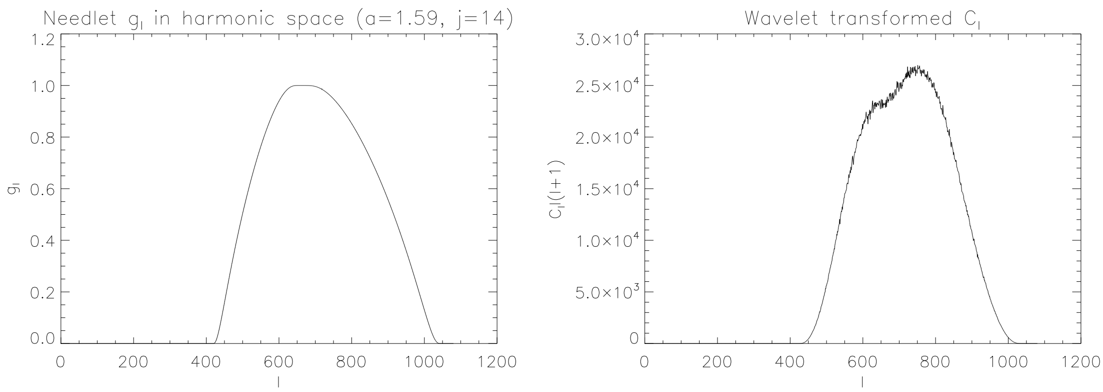
Figure 4.4: The power spectrum C_l after 10 simulations, now with a greater amplitude at the small scales caused by the point sources. The best fit C_l is the solid line.



(a) g_l for $R = 7.5'$

(b) C_l after transformation

Figure 4.5: Wavelet transformation of the power spectrum in figure 4.4 with an SMH Wavelet of scale $R = 7.5'$. Figure (a) is the wavelet in multipole space g_l, while figure (b) is the power spectrum after the transformation. The large scales have been notably damped, while the small scales have been enhanced.



(a) g_l for $a = 1.59/j = 14$

(b) C_l after transformation

Figure 4.6: Wavelet transformation of the power spectrum in figure 4.4 with a needlet with $a = 1.59$ and $j = 14$. Figure (a) is the needlet in multipole space g_l, while figure (b) is the power spectrum after the transformation. The large scales have been fully damped, while the small scales have been enhanced.

Chapter 5

Method and implementation

5.1 Problem

The previous chapters have illustrated how the point sources contaminate the signal from the CMB at small angular scales. Some of the point sources are hard to separate from the temperature variations, and can give large errors in the estimates of the cosmological parameters. It is desirable to get better measurements of the cosmological parameters by detecting as many point sources as possible. Our primary goal is to improve the methods of point source subtraction, and to achieve that a comparison will be made of the efficiency of no wavelets, Spherical Mexican Hat wavelets and needlets applied to CMB maps.

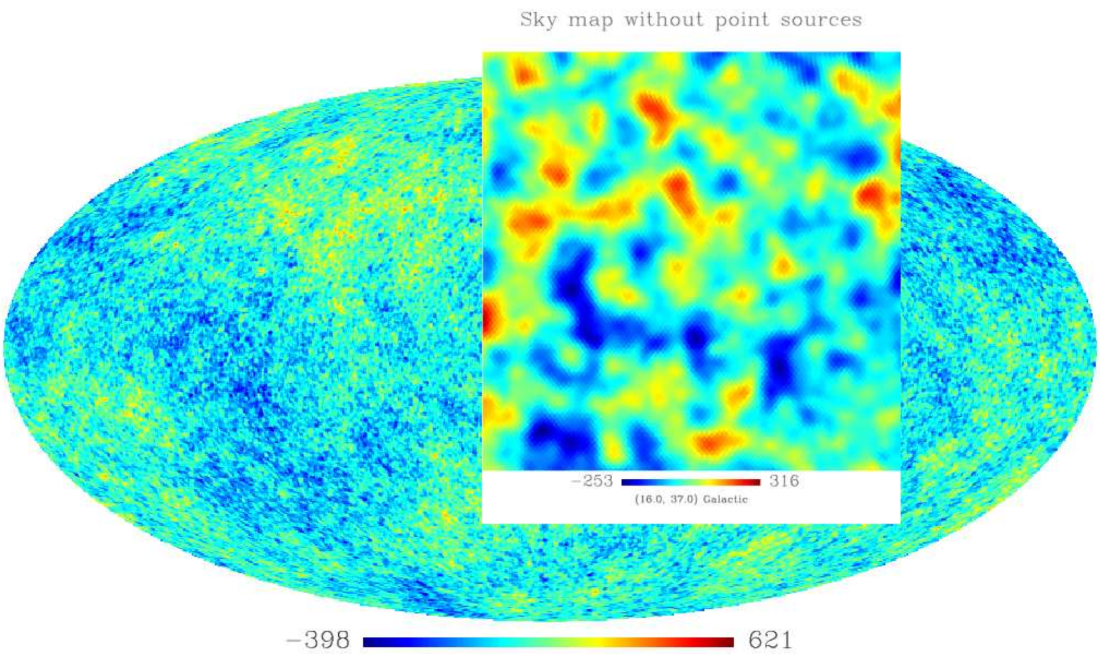
5.2 Implementation of resolved point source detection

The source code for the detection of resolved point sources in Appendix A.1 allows tweaking of several variables, however, some variables will be fixed throughout the analysis. In the simulations, $N_{\text{side}} = 512$ will be used with $l_{\text{max}} = 1080$. The resolution is a good compromise between high resolution and effective code, and in addition, WMAP resources can be more easily implemented. It will be assumed that at this resolution a sky map will contain 2000 resolved and unresolved point sources. WMAP detected 323 resolved point sources at the 5σ level[1], and to achieve about the same amount of detections with our program, the intensity of the point sources must be maximum $I = 65\sigma_{\text{cmb}}$, where σ_{cmb} is the standard deviation of the temperatures. The intention is to compare the efficiency of the wavelets, and even though the intensity and the number of point sources may not be true to reality, they will nevertheless be identical for the subjects to be compared.

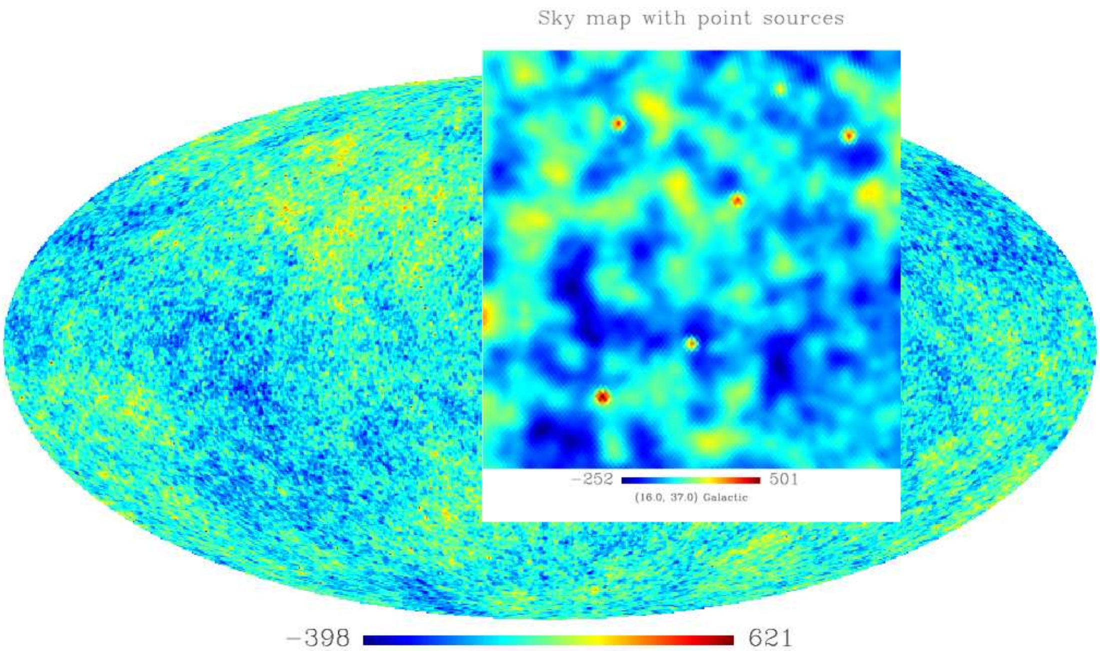
5.2.1 Without wavelets

The first part of the program simulates N sky maps with the Monte Carlo method for retrieving an accurate estimate of the variance of the maps by using equation (4.3). A beam taken from the V band (61 GHz) of WMAP is added in the process (this beam includes the pixel window function), while noise is not simulated to begin with. The noise adds additional complications to the detection of point sources by affecting the small angular scales. The simulation uses

[1]The 5-year analysis of the WMAP observations was released after the simulations in this thesis commenced, and therefore the 3-year analysis was used as reference.

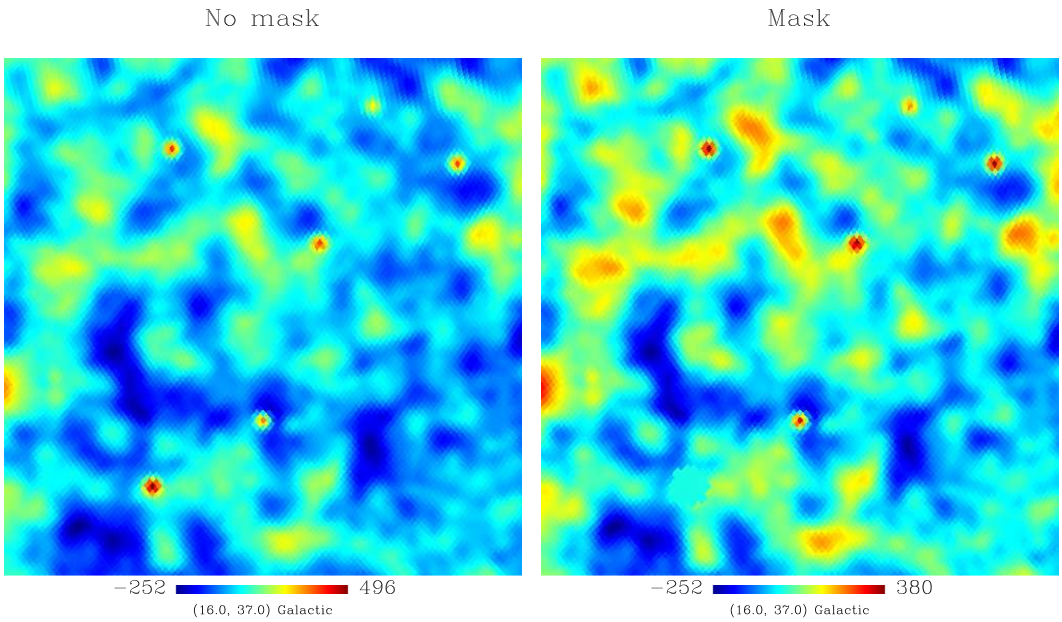


(a) Without point sources



(b) With point sources

Figure 5.1: A single simulation of a sky map before (a) and after (b) point sources have been added. Note the increase in sky temperature in the magnified illustration from a maximum of 316 μK to 501 μK.



(a) No mask **(b)** Mask

Figure 5.2: The smoothing of pixels surrounding the point source adds additional false point sources. When a point source is detected, it is therefore covered with a mask that reduces the number of false detections. Figure (a) shows the map without mask, while figure (b) shows the map with mask after detection at the 5σ level. Only one point source has high enough flux to be detected at this level (bottom left).

the existing `create_alm`, `alm2map` and `map2alm` functions in the HEALpix package. The first function makes random a_{lm} coefficients from the best fit C_l, and the two latter functions transform back and forth between map and a_{lm} coefficients. After the beam has been applied to the map, the standard deviation is calculated from a sum over all pixels and simulations. Extra transformations between the map and the a_{lm} coefficients are performed at the point in the code where the point sources will be added in a second Monte Carlo loop. To avoid computational differences, the transformations must be performed an equal number of times for the calibration of the variance and during the detection of point sources.

In the second Monte Carlo loop, following the calculation of the standard deviation of the simulations, the point sources are added to random locations on the map (see figure 5.1). The flux and the position of the point sources are taken from a uniform distribution with values between zero and a defined maximum limit. Realistically their intensity is distributed differently, but a uniform distribution is a sufficient approximation for comparison of the detection rate. The position is limited to the dimensions of the sphere, $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi]$, while the intensity is limited to the maximum intensity $I = 65\sigma_{\mathrm{cmb}}$. To be able to distinguish between real point source detections and false detections, the location of all the point sources are saved. The point source detection may now commence. If the flux of a given pixel is larger than a desired factor of σ_{cmb}, the program checks if the location of this pixel corresponds to the location of a point source. The number of true detections is counted together with the number of false detections for all the N simulations. The final detection numbers are averages over the number of detections for each simulation.

The beam adds some complications. When the beam is applied to the map, it averages the temperature in each pixel over the temperatures in the nearby pixels. Thus, for each true point source, a series of extra false point sources is created, which are not associated with the false point sources predicted by Gaussian statistics. A pixel larger than the defined threshold might be a byproduct of a nearby point source, that is, the pixel may not necessarily be the point source itself. Therefore, the program checks which of the surrounding pixels within a chosen radius around the detected pixel has the largest temperature, before a mask is applied around the pixel. If any other true point sources are found within the radius, the mask is removed at their pixel locations, so that they may be counted and their surrounding pixels masked when the program checks their locations. A larger mask than necessary for each pixel combined with many detections can result in too much of the background ignored. The beam is slightly larger than its FWHM, so with a mask size of $28'$ we should get all the extra false point sources without exaggerating the size of the mask[2]. Figure 5.2 shows one part of the sky map before and after masks have been applied to the point sources.

The interpretation of true point source detection, false point source detection and extra false point source detection should be emphazised. A *true detection* means a point source in the temperature map, a *false detection* means that the detected point source originates from the background and an *extra false detection* means false point sources created by the smoothing of a true or false point source. Summarized, the simulations described in this section proceed as follows:

- A first Monte Carlo loop simulates N CMB maps with beam, and calculates their standard deviation.

[2]It is difficult to determine the size of the WMAP beam, but the effective size of $21'$ has been assumed throughout the thesis [2].

- A second Monte Carlo loop makes new simulations of maps, and adds the point sources to random locations on the sphere with flux up to a defined maximum intensity.
- Point source detection is performed within the loop. All true and false detections are counted, and a mask is put at the locations of the true point sources to avoid counts from extra false point sources.
- The point source counts are averaged over all simulations.

5.2.2 With wavelets

When the program in Appendix A.1 is run with wavelets it works in a similar fashion as without wavelets, and the same parameters will be used as before. This time, however, the temperature map is filtered with a wavelet prior to calculating the standard deviation and checking what pixels are above the threshold. The wavelet generation subroutines for the Spherical Mexican Hat wavelets and needlets are developed by Frode K. Hansen.

Both the SMH wavelets and needlets have been implemented in the same program, and the choice of wavelet type, mother wavelet and wavelet scale can be selected by a single parameter each. The large angular scales are not interesting in this setting, since the point sources affect the small scales. Therefore, for the SMH wavelet, the scale of the wavelet will be limited to about $20'$ depending on what scale is most efficient at detecting the point sources. The smallest scales must also be avoided, as there is no purpose with a wavelet size smaller than the pixel size. The smallest possible size of the SMH wavelet is slightly less than the pixel size of $6.81'$, so we will choose $5'$ as the smallest angular scale. Needlets reach larger scales at small j for all a and at all values of j for large a, and thus we can expect small j and large a to be less efficient at finding point sources. Smaller scales than the chosen $l_{\max}$ for needlets is not allowed, as seen from equation (4.6).

When wavelets were not applied to the map, a fixed mask size was used on the point sources. If many extra false point sources are present during point source extraction from real observations, a larger percentage of the sky map will be masked. In order to compare this effect between the different wavelets, a dynamic mask has been introduced. Since the level of smoothing changes with each mother wavelet and wavelet scale, a unique mask is required for each wavelet to be able to remove all the extra false point sources. The mask size is selected by finding the radius at which the value of a wavelet transformed single pixel is reduced by 90 %. Outlined, the dynamic mask is created as follows:

- Add one pixel to an empty map, and apply the wavelet on it.
- Check all values in the map, and create a mask from all surrounding pixels that is larger than 10 % of the original pixel.
- Calculate the radius of the mask from the total area of the pixels.

Certain needlets, however, generate more troughs and peaks in pixel space than the SMH wavelet, due to the bad localization properties in pixel space when localization in harmonic space is good, as mentioned in the last chapter. The above code for determining the mask will make a smaller mask than required when there are many troughs less than 90 % the flux of the single pixel in between the peaks, as illustrated in figure 5.3. The needlets therefore receive an additional 10 % to the size of the mask.

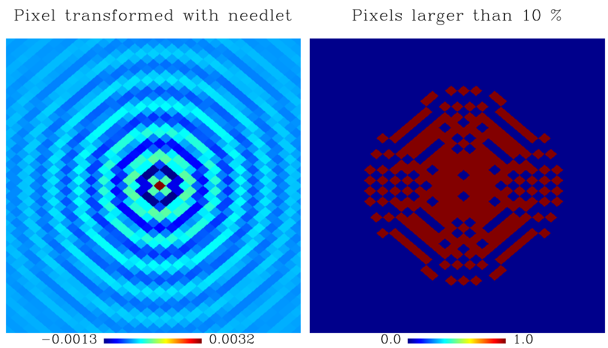
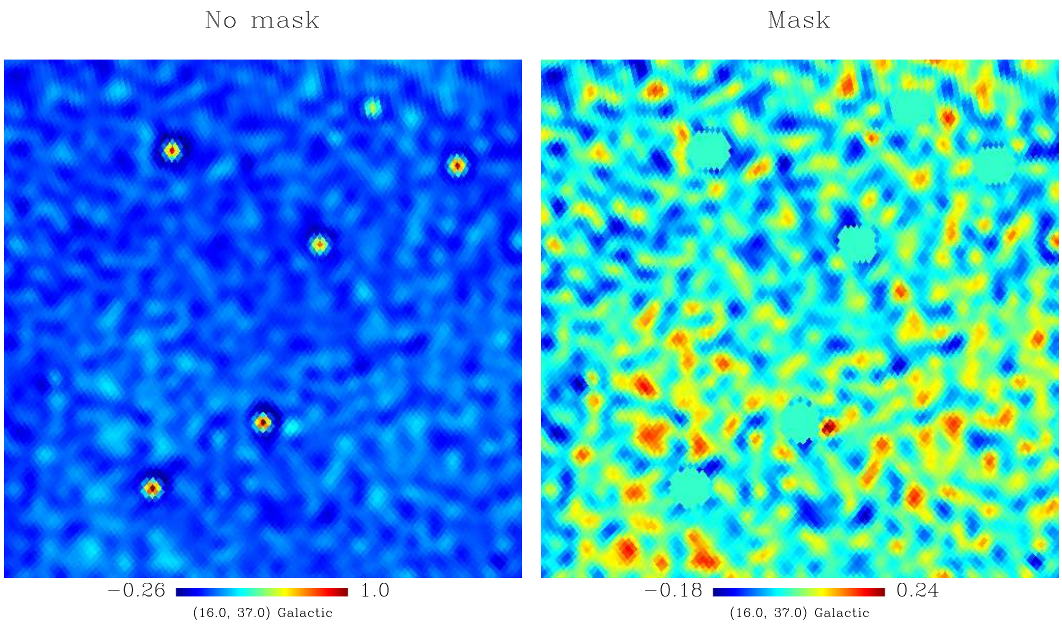


Figure 5.3: The size of the dynamic mask is determined from the area of all surrounding pixels larger than 10 % the size of a wavelet transformed pixel. The left panel is a single pixel transformed with the needlet $a = 1.096259$ of scale $j = 75$, while the right panel shows all pixels larger than 10 % its size. The bad localization properties of the needlet generates troughs that make the calculation of the mask harder.



(a) No mask **(b)** Mask

Figure 5.4: (a) A sky map filtered with the SMH wavelet of scale $R = 7.5'$. The flux of the point sources is enhanced, and they are more easily detected. (b) The map after the mask has been applied to point sources detected at the 5σ level. The apparent point sources that were not detected without wavelets in figure 5.2 are now detected.

Figure 5.4 is a visualization of the map after transformation with the SMH wavelet. The figure shows how the transformation elevates the point sources, and the better ability of the SMH wavelet to locate them. Maps transformed with two types of needlets are illustrated in figure 5.5. One of the wavelets generates many waves on the sphere, while the other generates less. Also some of the false point sources above the threshold create a series of extra false point sources. This did not constitute a significant problem without wavelets. Since the background generally has lower flux than the point sources, the effect is less extensive. A mask size 40 % the size of the mask for the true point sources will be used for the false point sources, which have been tested to get most of the extra false point sources.

5.3 Implementation of unresolved point source detection

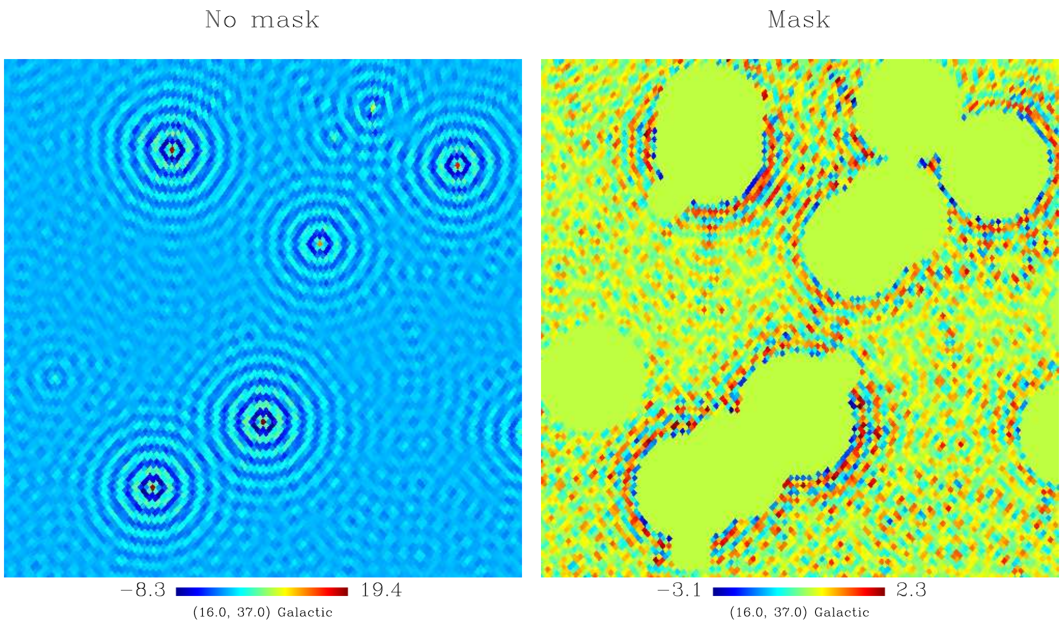
The code for detecting the unresolved point sources in Appendix A.2 is based on the code for the resolved point sources, but this time no direct point source detection is performed in the code. The first Monte Carlo loop, used for determining the standard deviation of the sky maps earlier, is this time carried out for calculating expected skewness and kurtosis using equations (4.1) and (4.2). The data from this run will be used for generating confidence levels for the skewness and kurtosis of the CMB wavelet maps. A larger number of simulations than earlier is needed before these statistics stabilize at their theoretical values, since the confidence intervals for skewness and kurtosis can only be calculated from many distributions of pixels. In the second Monte Carlo loop, the point sources are added to the maps, and the skewness and kurtosis are once again measured. The data from the simulation of CMB maps with point sources will then be compared to the theoretical confidence levels to detect any deviation from Gaussianity.

Before any detection of unresolved point sources is performed in CMB data, the located resolved point sources are masked out, and do not contribute to the amplitude of the temperature map any more. Therefore, for the simulation of detecting deviations caused by the unresolved point sources, the maximum intensity of the point sources have been reduced to $I = 35\sigma_{\mathrm{cmb}}$. With this intensity there are almost no point sources with amplitude above the 5σ level of the temperature.

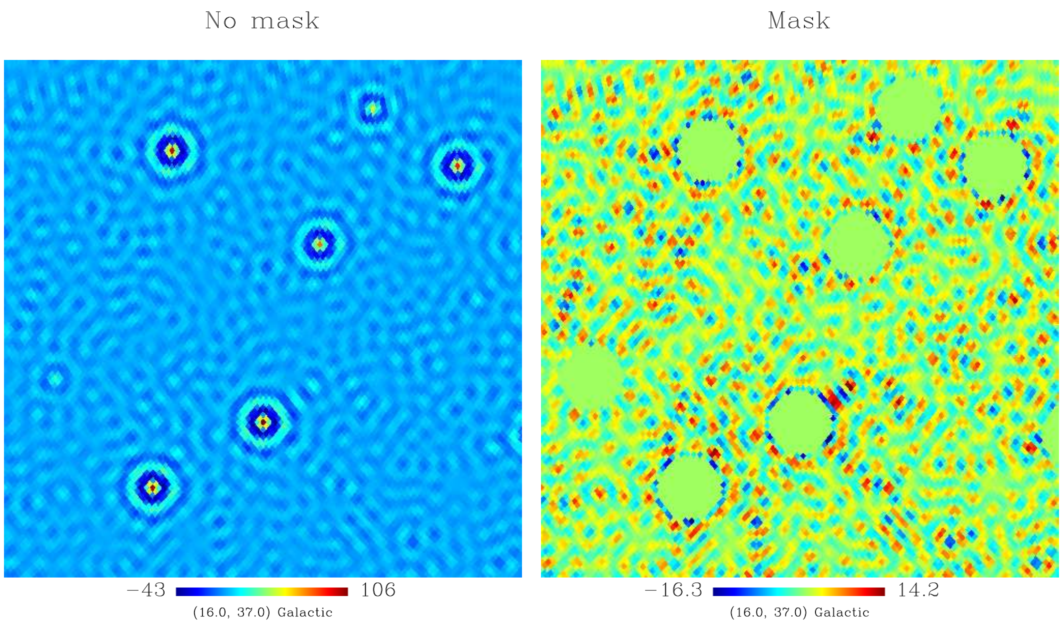
The χ^2 statistic in equation (4.4) is used to find the deviation to the power spectrum caused by the unresolved point sources. The formula is summed over several wavelet scales (corresponds to the observations n in the equation) to get as much information about the deviation as possible. The χ^2 minimization finds the deviation that best compares to the deviation of the observations. First, an attempt will be made to find an analytical model for the skewness and kurtosis for different point source amplitudes. If an analytical expression is not found, the models will be simulated and their mean will be used in the equation. For simplicity, the simulated observations the model is compared with will have point sources with a constant intensity of $I = 20\sigma_{\mathrm{cmb}}$.

5.4 Introducing noise in the analysis

Section 3.6.1 in the chapter about CMB analysis illustrated how instrumental noise affects the observations of the CMB. To make the comparison of the point source detection performance more realistic, the noise should be added in the analysis. The noise is present on the small angular scales, and since the point sources are also found on these scales, the noise should



(a) No mask ($a = 1.096259$, $j = 75$) **(b)** Mask ($a = 1.096259$, $j = 75$)



(c) No mask ($a = 1.35483$, $j = 22$) **(d)** Mask ($a = 1.35483$, $j = 22$)

Figure 5.5: Figure (a) and (c) are sky maps after transformation with two different needlets. The needlet with $a = 1.096259$ generates more troughs and peaks on the sphere than the needlet with $a = 1.35483$. A much larger mask is therefore needed for the first mentioned needlet, as seen in figure (b) and (d), but the code is not able to get all the extra false point sources for the first case. The smaller masks seen in figure (b) is the 40% masks used on the false sources.

naturally complicate the detection of the point sources. This analysis will simulate the effective noise from the V band of WMAP. The standard deviation of the noise as observed by WMAP in this channel at each point on the full sky is visualized in figure 5.6. The standard deviation in each point varies according to how many times WMAP has observed that point on the full sky, and many observations reduces the standard deviation. For each simulation and each pixel on the full sky, a random number from Gaussian distributed data, with the standard deviation of that pixel in the noise map, is drawn and added to the map. Since the standard deviation of each noise pixel is different from the others, a unique standard deviation for each pixel will need to be calculated, rather than a total standard deviation for the complete map. This also means that we will have to use more simulations before the standard deviation converges towards the correct value. The skewness and kurtosis statistics in the unresolved analysis, however, are dependent on the complete distribution, so the standard deviation for this analysis must be calculated as before.

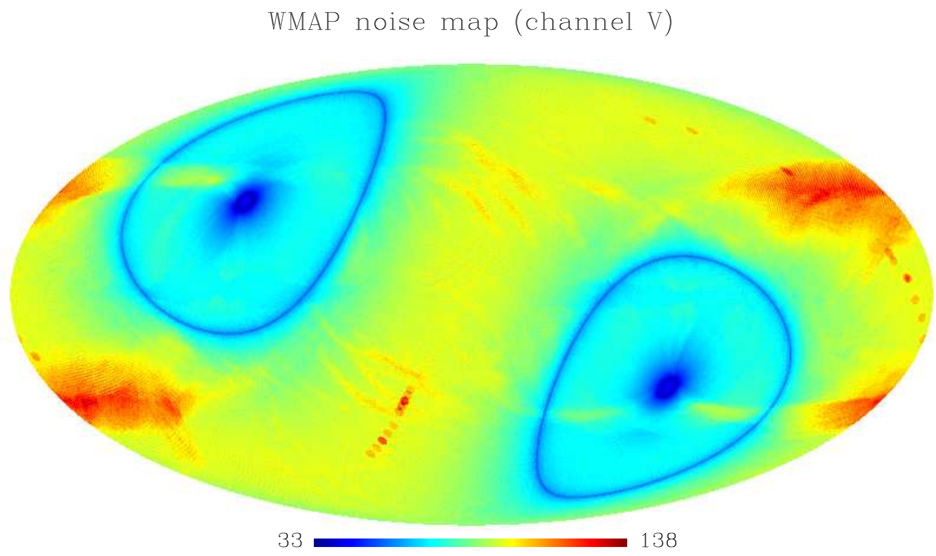


Figure 5.6: The RMS noise map from the V band of WMAP. WMAP has performed most observations in the ecliptic poles, and in rings around 141° from the poles, where the standard deviation is at its lowest. Few observations have been performed in the ecliptic plane and in the positions of planets (the small circular masks), which contaminate the CMB signal [15].

Chapter 6

Results

The results chapter is divided into three main parts. The first section compares the performance of the SMH wavelet and needlet for the resolved point sources, and section 6.2 will test their performance for the unresolved point sources. Both the two first sections of the analysis deal with less complicated noiseless sky maps, but the last section will add noise for a more realistic approach, and some of the tests done in the two first sections will be performed again.

Before the results presented in this chapter were generated, the code went through a thorough consistency and bug check. One very important consistency check was the correctness of the number of false and true detections. The number of false detections must coincide with the number of false detections predicted by Gaussian statistics. Due to the extra false point sources, the true point sources were removed from the map before the accordance with Gaussian statistics was checked. The consistency of the true point sources was tested by varying all actuating parameters. It was especially important that the applied mask did not hinder true point sources to be detected, as this would have given fewer true detections than one is supposed to find.

6.1 Detection of resolved point sources

6.1.1 Without wavelets

Table 6.1 and 6.2 list the results for the run with no wavelets applied to the map with $N = 10$ and $N = 100$ respectively, where N is the number of simulations. It is apparent that a confidence limit of 3σ yields too many false detections to be reliably used. If the 3σ level was used on real observational data, where true and false point sources cannot be distinguished, the number of pixels masked out (removed from the map due to the point sources) would be very large, which again would lead to bigger uncertainties in the estimation of the power spectrum. Both the 4σ and 5σ levels, however, are good candidates. The first gives a larger number of detected sources, while the latter gives practically no false detections, but at the expense of true detections. It is pointless using a larger threshold than 5σ, when the contribution from false point sources has already nearly vanished. The number of false detections stabilizes around the number predicted by Gaussian statistics when the number of simulations rise. The predicted numbers for the 3σ, 4σ and 5σ levels are 4250, 100 and 1 respectively. The number of false detections for the 4σ case has a slightly higher value than predicted, but this is only due to the chosen seed, and it stabilizes more when N is increased a bit further. Throughout the analysis, 100 simulations will be maintained for final data, while 10 simulations will be used to get quick overviews. The histogram in figure 6.1 illustrates how the pixels get larger temperatures when point sources are present.

	# of true	# of false	% masked
3σ	954	3778	1.6
4σ	618	78	1.0
5σ	310	0.5	0.5

Table 6.1: The number of true and false detections for three thresholds for $N = 10$. Level 3σ yields too many false detections, but the 4σ and 5σ levels can be used. The percentage of the map ignored by masking out the detected point sources is also shown.

	# of true	# of false	% masked
3σ	959	4306	1.6
4σ	614	101	1.0
5σ	309	1	0.5

Table 6.2: The number of true and false detections for three thresholds for $N = 100$, where the number of false detections has become more stabilized around the predicted value. See table 6.1 for details.

6.1.2 Spherical Mexican Hat wavelets

From this point results are focused on the 5σ level. The efficiency of each scale for the SMH wavelets is plotted in figure 6.2(a), where scales from $5'$ up to $15'$ have been tested. The best scale is the peak of the curve, which lies around $R = 7.5'$. At this scale 1425 true point sources are detected, a lot more than detected without wavelets. The percentage of the pixels in the map covered with a wavelet mask for this scale is 2.7 %. Figure 6.2(b) shows the associated number of false detections, which is ~ 1 for $R = 7.5'$. From the figure it is apparent that the number of false detections increases with the scale. The false sources contributing to this count are due to extra false sources near true sources smaller than the threshold. When the wavelet takes an average over pixels near each other on the map in these cases, the extra false sources are averaged larger than the threshold, while the true sources are not. The program could have counted such cases as true sources, but there are not many of them and they only have an effect were the number of true detections is smaller. The finer plot around $R = 7.5'$ in figure 6.3 reveals no significant change in the number of detected point sources, but $R = 7.32'$ is slightly better with 1426 true detections.

If the different scales discover different point sources, several scales could be combined to reach a larger number of true point source detections. However, the number of unique detections for each scale is negligible, as seen in table 6.3. The table lists the mean number of unique pixels detected compared to those detected by the scale $R = 7.5'$. The total number of unique detections is even smaller than the sum of these numbers, since other scales also have some equal detections. In a finer search for unique pixels around $R = 7.32'$, the number of equal detections between all scales is larger.

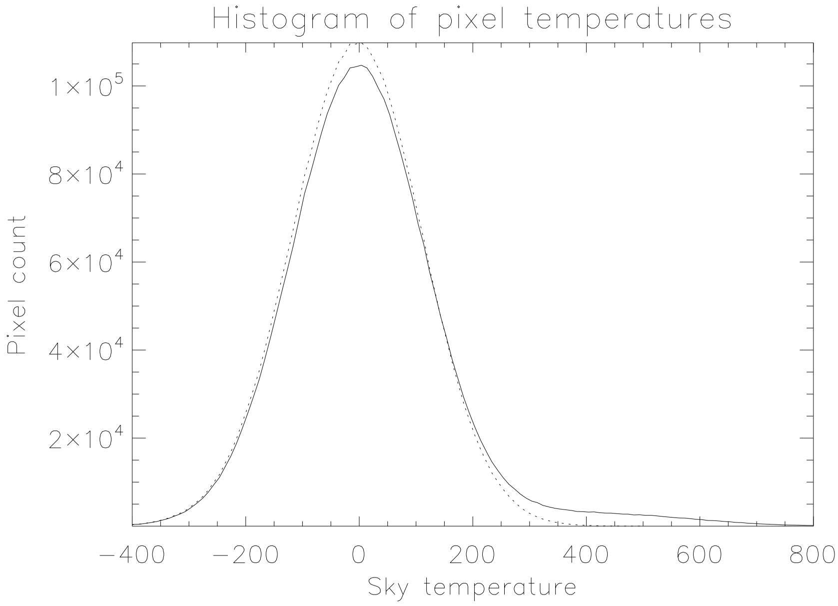
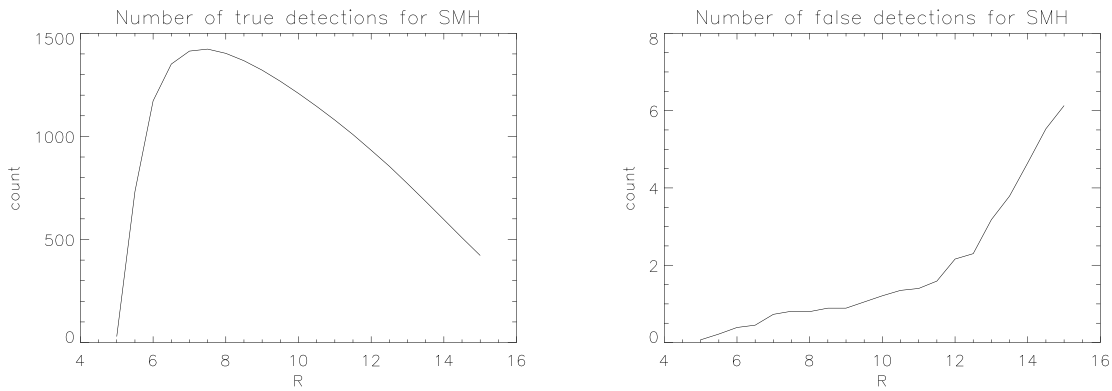


Figure 6.1: The solid line pictures a histogram of a sky map with point sources added, while the dotted line is a clean sky map with no foreground. The pixels with point sources added increases the temperature for some pixels, as can be seen by the difference in the frequency at the peak temperature and the longer tail.



(a) True detections

(b) False detections

Figure 6.2: The number of true (a) and false (b) detections after wavelet transformation with the SMH wavelet for selected R values after 100 simulations. The small scale $R = 7.5'$ is best with around 1425 true detections and 1 false detection.

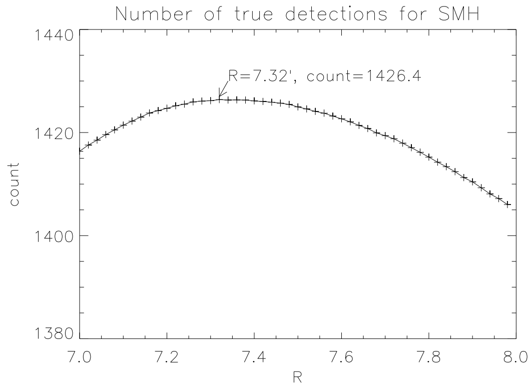


Figure 6.3: A finer plot of the number of true detections around $R = 7.5'$. The best result of 1426 true detections is achieved with $R = 7.32'$.

R	count	R	count	R	count	R	count	R	count	R	count
$6'$	0.2	$8'$	2.6	$9.5'$	0.9	$11'$	0.5	$12.5'$	0.2	$14'$	0.2
$6.5'$	2.1	$8.5'$	1.8	$10'$	0.7	$11.5'$	0.3	$13'$	0.2	$14.5'$	0.2
$7'$	4.4	$9'$	1.2	$10.5'$	0.7	$12'$	0.2	$13.5'$	0.1	$15'$	0.1

Table 6.3: The average number of unique detections after 10 simulations for each scale compared to $R = 7.5'$ is small. If the other scales are compared to each other, the total number of unique detections becomes even smaller.

6.1.3 Needlets

As seen in chapter 4, needlets come in many variations. The mother wavelet is changed by varying the parameter a, while the scale of each of these mother wavelets is changed by varying the parameter j. A selection of the tested needlets from $a = 1$ to $a = 2$ are listed in table 6.4. All the needlets are of the smallest possible scale j, which almost always detects most point sources. The discrepancy here is for very small a, where the difference between each scale is so small that statistical variation occurs. At first glance, the table tells us that mother wavelets which do not allow $l_{\max}$ to reach its highest value, detect fewer point sources. Furthermore, the highest number of detections is seen for $a = 1.09898$, which detects 1741 true point sources. However, for the most part, smaller a means a greater number of false sources. Recall that the dynamic mask was introduced because of the changing number of extra false point sources between the different wavelets. For needlets, the mask was increased by an additional 10 %. Even with this increase the code does not manage to catch all the extra false point sources when the number of peaks and troughs generated by the wavelet in pixel space gets larger with a. Nevertheless, the mask in these cases is already so large that too much of the background is removed. How this works out will become more apparent in the next section, when the performance of the best needlet is simulated without knowledge of the locations of the true point sources, and with a mask size comparable to the size used on real observations.

As long as $l_{\max}$ is not too small, needlets perform better than the SMH wavelet for all mother wavelets approximately in the range $1.02 < a < 2.0$. The observations in table 6.4 are

a	j	$l_{\min}$	$l_{\max}$	# of true	# of false	% masked
1.001	6387	1077	1080	414	7068	81
1.01	700	1049	1070	1361	3317	80
1.02	351	1023	1065	1538	2277	70
1.03	235	1009	1070	1620	1424	57
1.04	177	995	1076	1667	1413	48
1.05	142	972	1072	1697	1269	42
1.062	115	951	1073	1720	1064	35
1.07	102	928	1063	1609	154	31
1.07016	102	943	1079	1733	755	31
1.08	80	874	1019	1660	511	27
1.08069	89	924	1079	1737	1234	28
1.09	80	905	1075	1738	838	24
1.09898	73	894	1080	1741	582	21
1.1	72	869	1051	1708	287	22
1.1961	38	754	1079	1689	45	10
1.2	37	709	1020	1642	7	11
1.3	25	543	917	1595	8	10
1.308	25	629	1076	1678	24	9
1.4175	19	534	1073	1670	2	5
1.5	16	438	935	1590	1	6
1.50812	16	475	1080	1649	2	6
1.5922	14	423	1071	1618	1	6
1.7	12	343	990	1544	14	3
1.71	12	366	1069	1586	25	3
1.7896	11	337	1079	1574	9	3
1.886	10	302	1074	1554	4	4
1.9	9	170	613	571	11	3
2.0	9	256	1024	1490	2	4

Table 6.4: Number of true and false detections for a selection of needlets at their smallest possible scale j, together with their associated $l_{\min}$, $l_{\max}$ and mask size, averaged over 10 simulations. See figure 6.4, 6.6 and 6.5 for plots of these numbers.

a	j	$l_{\min}$	$l_{\max}$	# of true	# of false	% masked
1.001	6387	1077	1080	414	7068	81
1.0101	694	1058	1080	1362	3342	81
1.01987	354	1038	1080	1544	2681	68
1.0288	245	1020	1080	1610	1723	56
1.03939	178	999	1080	1668	1419	47
1.0497	143	980	1080	1698	958	41
1.06376	112	954	1080	1726	1021	34
1.07234	99	939	1080	1733	1111	30
1.08164	88	924	1080	1739	1189	28
1.09244	78	905	1080	1742	796	24
1.11163	65	874	1080	1738	259	18
1.1532	48	812	1080	1711	220	14
1.20178	37	748	1080	1688	31	10
1.24392	31	698	1080	1678	231	9
1.30818	25	631	1080	1680	25	9
1.35483	22	588	1080	1680	6	7
1.39459	20	555	1080	1677	3	5
1.44428	18	518	1080	1667	2	6
1.50812	16	475	1080	1649	2	6
1.54735	15	451	1080	1637	1	6
1.593	14	425	1080	1623	2	6
1.6469	13	398	1080	1608	15	5
1.7113	12	369	1080	1592	26	3
1.78972	11	337	1080	1575	9	3
1.88692	10	303	1080	1557	3	4
2.01067	9	267	1080	1525	1	4

Table 6.5: Number of true and false detections for a selection of needlets with $l_{\max}$ reaching 1080. These numbers are the dashed lines in figure 6.4, 6.6 and 6.5. The discrepancy in the number of true point source detections for $a = 1.24392$ is due to the low number of simulations.

more evident in figure 6.4, where all the true detection counts of each mother wavelet in the table have been plotted. Especially notable are the dips in the plot. These are caused by the mother wavelets that can not reach $l_{\max} = 1080$, as can be seen by the nice correspondence with the dips in figure 6.5, which is a plot of the associating $l_{\min}$ and $l_{\max}$ for each a. The dashed line in the figures illustrates the same case for only values of a that can reach the smallest scales (given in table 6.5), and thereby the largest numbers of true detections. The number of false detections, plotted on a logarithmic scale in figure 6.6, sinks drastically to start with before it settles. Some increase in between can be observed, which is caused by the mentioned deficient mask.

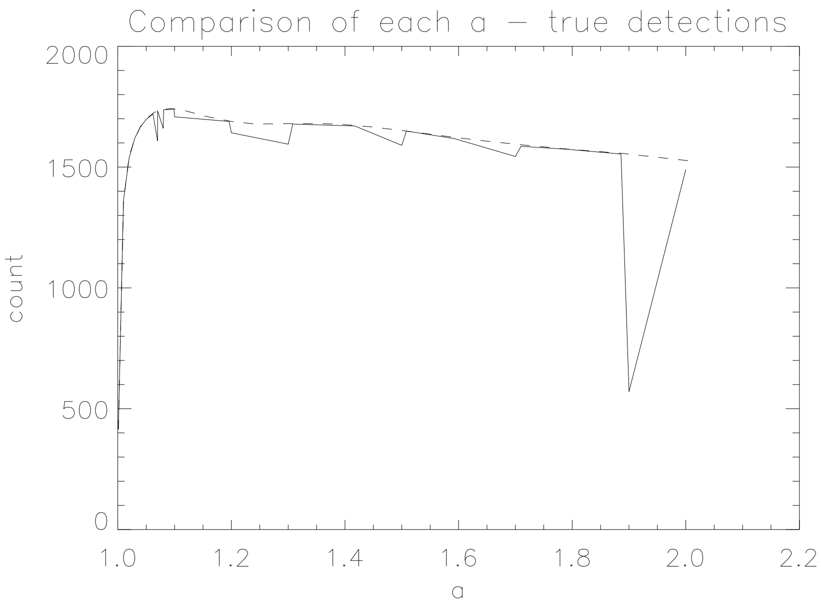


Figure 6.4: The solid line is a plot of the number of true detections of the needlets in table 6.4. The dips in the figure are the needlets which do not reach the largest possible $l_{\max}$. The dashed line is the same plot for the needlets in table 6.5, which are all defined up to the largest $l_{\max}$.

In figure 6.7 you find a finer plot around the scale that was determined best from table 6.5 and figure 6.4. The best scale here is $a = 1.096259$ with a slightly better count of 1744 true point sources. Now that the mother wavelet and scale which gives the best number of true detections for needlets is known, its performance for each j should also be looked at. This is shown in table 6.6 and figure 6.8. The number of true detections increases quickly and steadily from $j = 62$. The number of false detections, however, does not start to grow significantly before $j = 74$. At high j, the amplitude of the peaks and troughs the wavelet generates on the map is greater (see figure 6.9), and in addition the code presented in chapter 5 makes a smaller mask than necessary. Thus, a greater number of surrounding pixels pass the 5σ threshold, and cause the sudden increase in the number of false detections seen in the plot. If the false detections become a problem when the code is tested on the case where the locations of the point sources are unknown, it may be reasonable to use a slightly smaller j, if they perform better than for values of a where the number of false detections is small for the largest j.

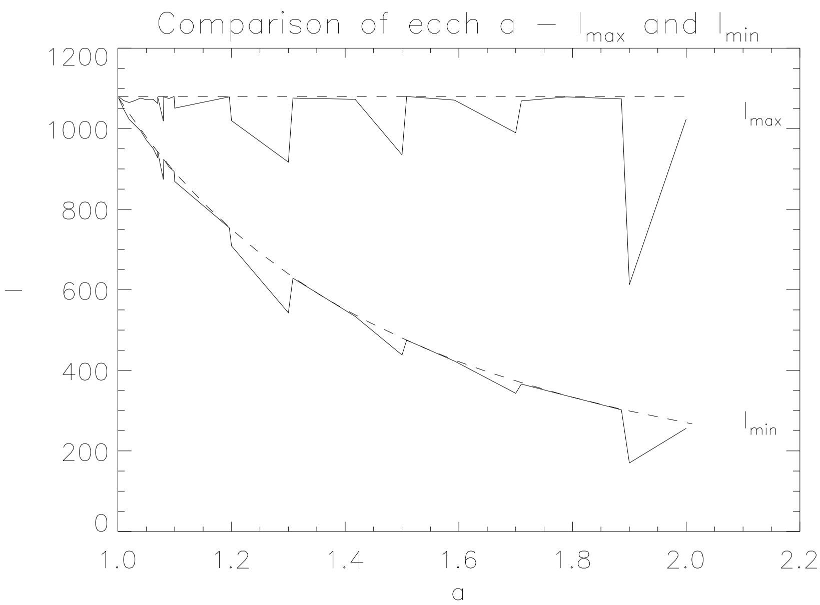


Figure 6.5: The associated $l_{\max}$ and $l_{\min}$ for the needlets in table 6.4. The dips are located at same a's as in figure 6.4, confirming that smaller $l_{\max}$ results in fewer detections.

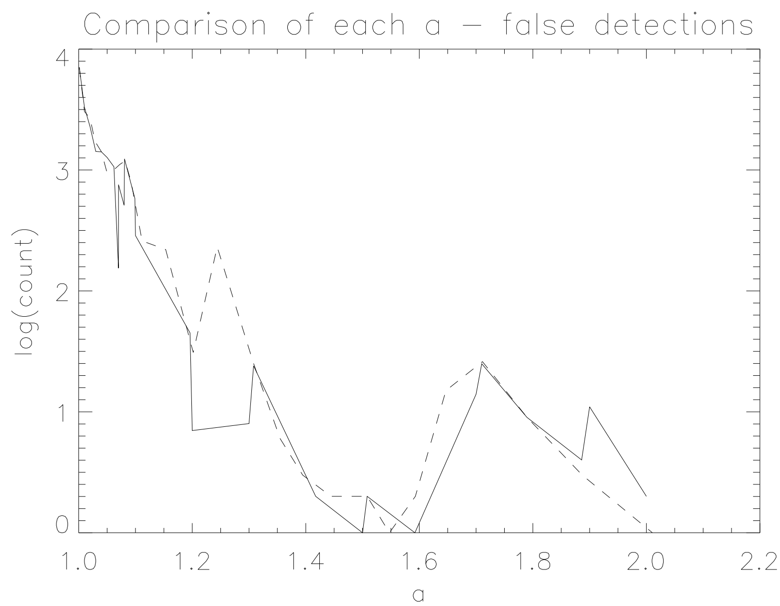
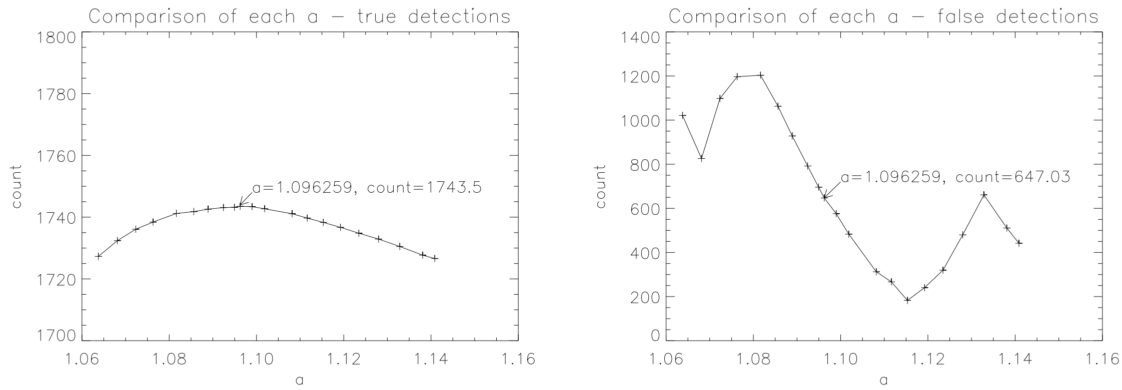
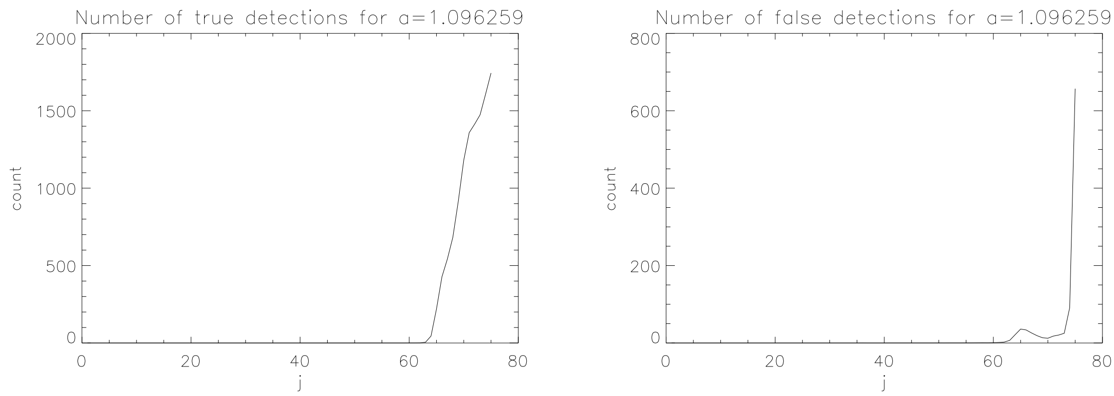


Figure 6.6: The associated number of false detections for the needlets in table 6.4. See figure 6.4 for details.



(a) True detections **(b)** False detections

Figure 6.7: Finer plots of the number of true (a) and false (b) detections around $a = 1.09898$ averaged over 100 simulations. The best scale is $a = 1.096259$ with 1744 true detections.



(a) True detections **(b)** False detections

Figure 6.8: The number of true (a) and false (b) detections at all scales j for the best needlet $a = 1.096259$ averaged over 100 simulations. The number of false detections increases rapidly in terms of j later than the number of true detections. The increase is due to the greater amplitudes of the peaks and troughs the wavelet generates at high j, and more of the surrounding pixels then pass the threshold. The numerical values for the higher j can be found in table 6.6.

j	$l_{\min}$	$l_{\max}$	# of true	# of false	% masked
69	518	622	914	14	28.2
70	568	682	1180	12	31.3
71	622	745	1358	18	28.9
72	682	820	1414	21	26.1
73	745	899	1474	25	24.2
74	820	985	1606	91	22.3
75	899	1080	1743	657	22.7

Table 6.6: The detection rate for the highest values of j for the needlet with $a = 1.096259$. The data has been plotted in figure 6.8

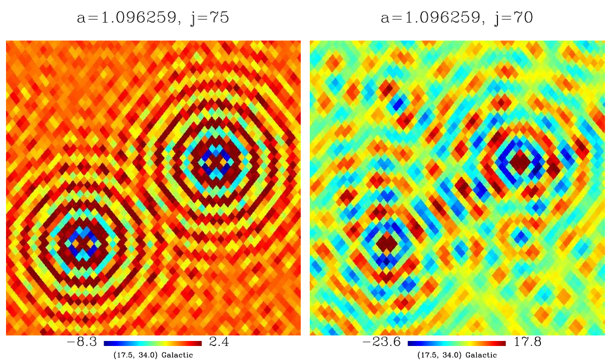


Figure 6.9: Needlet $a = 1.096259$ of scale $j = 75$ (left panel) and $j = 70$ (right panel) with maximum at 5σ, that is, the pixels shown in red are above this threshold. When $j = 75$, more of the surrounding pixels pass the 5σ threshold than when $j = 70$, and lead to a greater amount of false point sources.

For the SMH wavelet it was checked if the different scales detected different point sources, but there were not many unique detections between the scales. The case might be different for needlets. Needlets are more sharply defined in multipole space, and the detections at lower j might contain many differences from those at the highest j. Table 6.6 shows that $a = 1.096259$ has $l \in [899, 1080]$ for $j = 75$, and $l \in [745, 899]$ for $j = 73$. That is, every other j covers completely different scales. The results from table 6.7, however, clearly indicate that there is not much to obtain from using different scales. Since other needlets cover the same interval in multipole space for all j, it is satisfactory to only test the different scales of the best needlet against each other.

j	count	j	count
69	0.5	72	7.1
70	2.4	73	8.4
71	5.9	74	12.1

Table 6.7: The average number of unique detections after 10 simulations for the highest values of j for $a = 1.096259$ compared to $j = 75$. The number of unique detections is slightly higher than for the SMH wavelet, but also here the total number of unique detections becomes smaller when each individual scale is compared.

6.1.4 Unknown point source locations

Until now it has been assumed that the positions of the point sources are known, and thus it has been possible to distinguish between true and false point source detections. With a correct count of the number of true point sources detected, a precise way of comparing the ability of each wavelet to enhance the point sources was possible. A dynamic mask was used on the different wavelets to take the extra false detections into account, which varied a lot between the wavelets. When point source detections are performed on real data, however, the point source locations are not always known. It will now be assumed that the point source locations are unknown, and therefore any pixel larger than the threshold is masked out. Even though the location of the true point sources are not supposed to be known, the number of incorrect discoveries will still be kept track of to get a measurement of the performance level. Note that the incorrect detection count is now the number of wrong detections within the total count of detections. When point source detection is performed on real data, it is the beam mask that is used to remove the point sources from the map. The mask size will be kept at a fixed size comparable to the size used in the WMAP analysis [24], that is, two times the size of the beam ($42'$). This means that some wavelets will have many incorrect detections, but it is more important to detect and remove the true point sources that can give wrong estimates of the power spectrum.

The needlet previously found to have the best number of true detections was the first applied to the sky map, and the results can be viewed in table 6.8. As expected, the best needlet generates so many troughs and peaks in pixels space, such that the the true detections drown in false detections. The $42'$ mask is too small for needlets that require large masks (i.e. needlets with high a and high j), which is the main cause for the large number of incorrect detections. The code blindly chooses the first largest pixel it detects within the radius of the mask. Smaller true point sources within the radius are covered with the mask, and are not detected. Extra false point sources from these true point sources will then also go clear of the mask, increasing the number of false detections within the total point source detection count. It is also apparent from the table that smaller values of j are not better choices, as the number of detections gets too small when the number of incorrect detections is minimized.

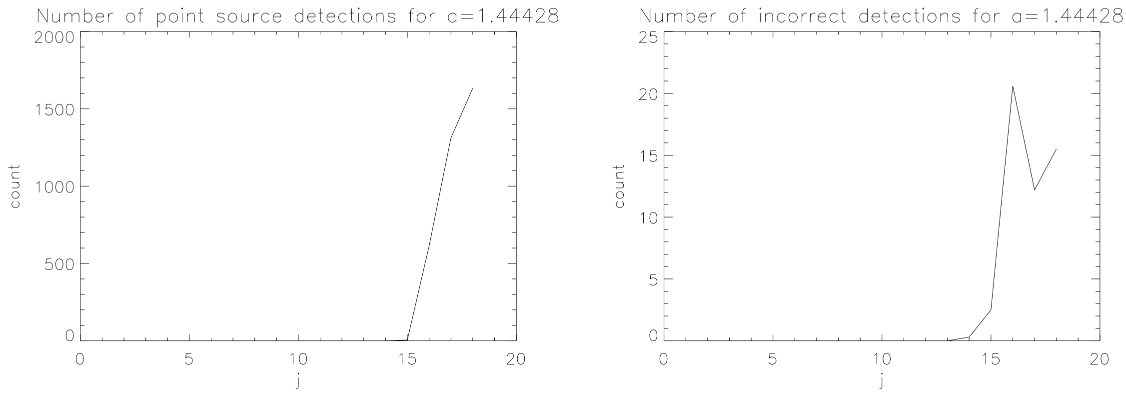
Some needlets, which had few false detections in table 6.5, have been listed in table 6.9. From $a = 1.44428$ and upwards the number of incorrect detections within the count does not seem to sink much more. This needlet detects 1634 point sources, where 15 of them are incorrect detections. Figure 6.10 shows its performance for all j. When the code is used on the best scale $R = 7.32'$ for the SMH wavelet, 1397 point sources are detected with 7 incorrect detections and 5.2 % masked map. This is about 240 detections less than the needlet above can yield, but with a slightly lower incorrect detection rate.

j	$l_{\min}$	$l_{\max}$	count	incorrect	% masked
69	518	622	1098	201	3.9
70	568	682	1507	357	5.4
71	622	745	1997	677	7.1
72	682	820	2199	821	7.6
73	745	899	2520	1079	8.4
74	820	985	3953	2381	11.8
75	899	1080	7268	5561	18.9

Table 6.8: The detection rate for the highest values of j for the needlet with $a = 1.096259$ when the point source locations are unknown. The incorrect detections within the count is also listed. The number of incorrect detections is still too high even after the number of detections has sunk below an acceptable amount. 100 simulations have been used.

a	j	$l_{\min}$	$l_{\max}$	count	incorrect	% masked
1.24392	31	698	1080	2026	393	7.1
1.30818	25	631	1080	1705	76	6.3
1.35483	22	588	1080	1661	31	6.2
1.39459	20	555	1080	1648	21	6.1
1.44428	18	518	1080	1634	15	6.1
1.50812	16	475	1080	1616	13	6.0
1.54735	15	451	1080	1605	14	6.0
1.593	14	425	1080	1594	15	5.9
1.64692	13	398	1080	1582	16	5.9
1.7113	12	369	1080	1569	16	5.8

Table 6.9: The detection rate for needlets that had low numbers of false detections in table 6.5 averaged over 100 simulations. From $a = 1.44428$ and upwards, the number of incorrect detections within the count does not sink much more.



(a) All detections

(b) Incorrect detections

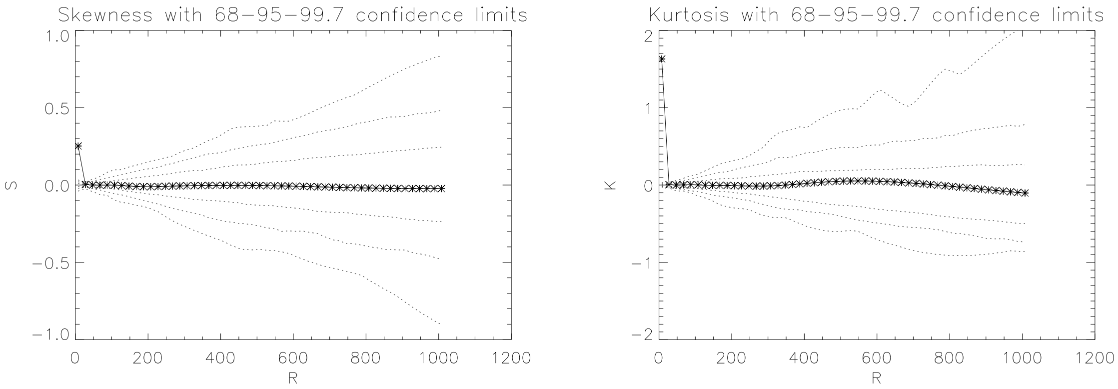
Figure 6.10: The number of all point source detections (a) and the associating number of incorrect detections (b) at all scales j for the needlet $a = 1.44428$, when the point source locations are unknown. The needlet detects a high number of point sources, with few incorrect detections.

6.2 Detection of unresolved point sources

To detect the unresolved point sources, the deviation from Gaussianity was measured using the skewness and kurtosis statistics. If the skewness and kurtosis at a certain scale are close to zero and within the confidence limits, there are no measured deviation from Gaussianity. The deviation is presented as a factor of the 68 % confidence level, which will allow comparison of the measured deviation between the different wavelets and wavelet scales. When the deviation on the different scales is known, one can use the χ^2 minimization technique to find the effect of the unresolved point sources on the power spectrum. The last part of this section will apply this technique to the discoveries of the first part.

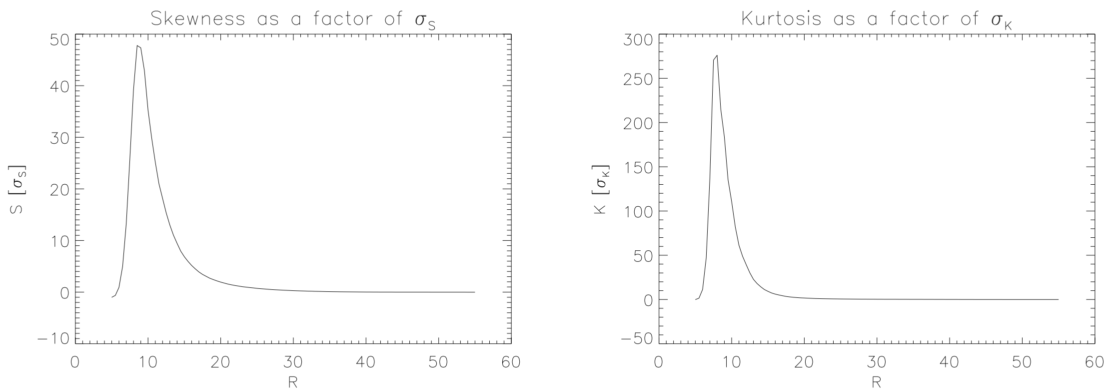
6.2.1 Spherical Mexican Hat wavelets

Figure 6.11 gives an overview over the skewness and kurtosis with 68 %, 95 % and 99.7 % confidence intervals up to very large scales for the SMH wavelet. The size of the confidence interval increases the larger the scale, where the cosmic variance comes into play. Large deviations from Gaussianity are only spotted at the very smallest scales. The skewness and kurtosis have been plotted around the smallest scales in figure 6.12 as a factor of their standard deviations σ_S and σ_K. The largest deviation occur around $R = 8'$, which is close to the scale that detected most resolved point sources. A finer plot around these scales in figure 6.13 reveal the largest deviation in skewness of $S = 54\sigma_S$ at $R = 8.52'$, and the largest deviation in kurtosis of $K = 283\sigma_K$ at $R = 7.68'$. The registered deviations are very large, since there is no noise present to disrupt the detection of non-Gaussianity (will be covered in section 6.3). The noise enhances the Gaussian form of the CMB distribution, and makes detection of deviations harder.



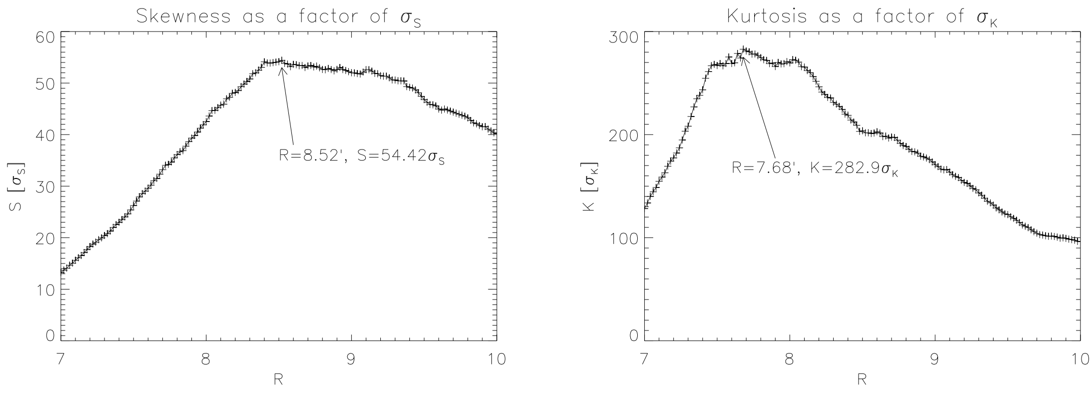
(a) Skewness **(b)** Kurtosis

Figure 6.11: An overview of the skewness (a) and kurtosis (b) for the SMH wavelet map up to very large scales. No deviation from Gaussianity is observed at large scales, although deviation is observed at the very smallest scales. The confidence intervals at the 68 %, 95 % and the 99.7 % levels are based on data from 1000 simulations, while the kurtosis and skewness values are generated from 100 simulations.



(a) Skewness **(b)** Kurtosis

Figure 6.12: Skewness (a) and kurtosis (b) as a factor of σ_S and σ_K for the SMH wavelet map at the small scales after 1000 simulations. The largest deviation occur around $R = 8'$, close to the scale where most point sources were detected for the resolved point sources.



(a) Skewness

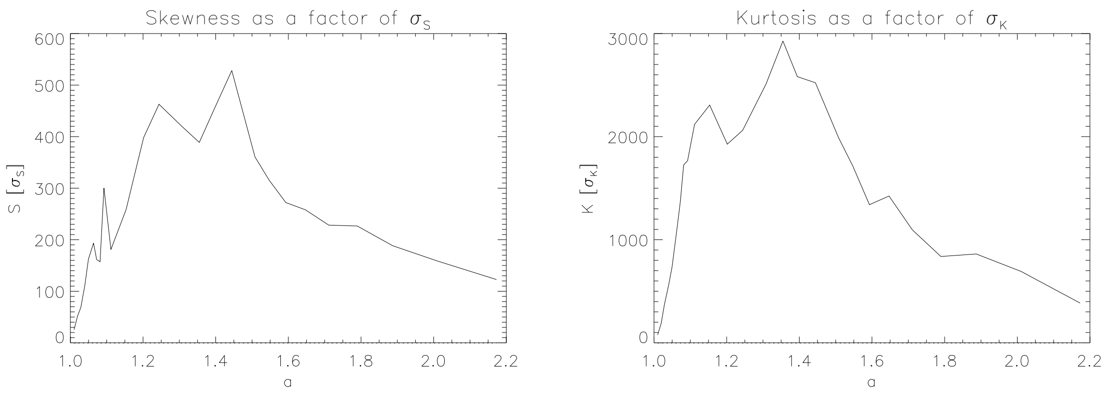
(b) Kurtosis

Figure 6.13: Finer plots around $R = 8'$ of skewness (a) and kurtosis (b) for the SMH wavelet map. The largest deviation occur at $R = 8.52'$ for skewness and at $R = 7.68'$ for kurtosis.

6.2.2 Needlets

For needlets a similiar approach has been adapted as for the resolved point sources. All the needlets in table 6.5 were tested and plotted, and the measured skewness and kurtosis are shown as a factor of σ_S and σ_K in figure 6.14. The largest deviations are always measured at the highest j, although due to small differences between the scales at very small a, the highest j is not always best for these. Some needlets which cannot reach an $l_{\max}$ of 1080 at the highest j have also been tested (not shown in the plots), but they have smaller deviations than the needlets in their vicinity with $l_{\max} \sim 1080$. The standard deviation in the plots were calculated after 100 simulations to obtain an overview, and have therefore not completely stabilized at their theoretical values. Apparent peaks are spotted approximately between $a = 1.2$ and $a = 1.6$, but the peak of the kurtosis does not begin to decrease significantly before around $a = 1.1$. Thus, all values of a (all that can reach $l_{\max} \sim 1080$) have been tested in the very large interval $a \in [1.1, 1.6]$ to make sure there are no signs of larger peaks for low a.

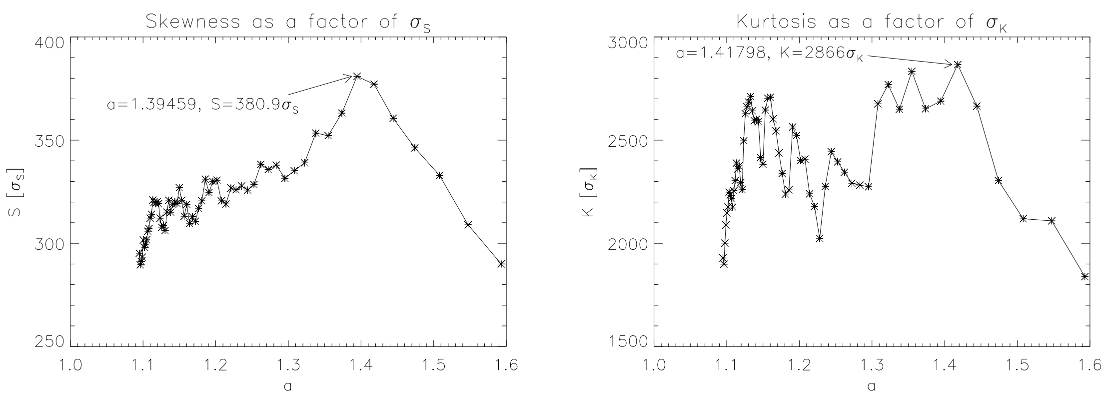
The results of the 1000 simulations of $a \in [1.1, 1.6]$ have been plotted in figure 6.15. The largest deviation for skewness is observed at $a = 1.39459$ with $S = 381\sigma_S$. The measurements of kurtosis, however, may still seem unstable. The plot shows many peaks for the tested interval, with the most apparent for $a \in [1.1, 1.2]$ and $a \in [1.3, 1.6]$. The peaks in the first mentioned interval have been tested with several seeds and an increased number of simulations, and have been confirmed to be genuine. The largest deviation for kurtosis, although not much larger than the the other peaks for $a \in [1.3, 1.6]$, is found at $a = 1.41798$ with $K = 2866\sigma_K$. The measured deviations for both skewness and kurtosis are much larger than what was measured from the SMH transformed maps. Figure 6.16 shows the deviation at all j for the best needlets with confidence intervals.



(a) Skewness

(b) Kurtosis

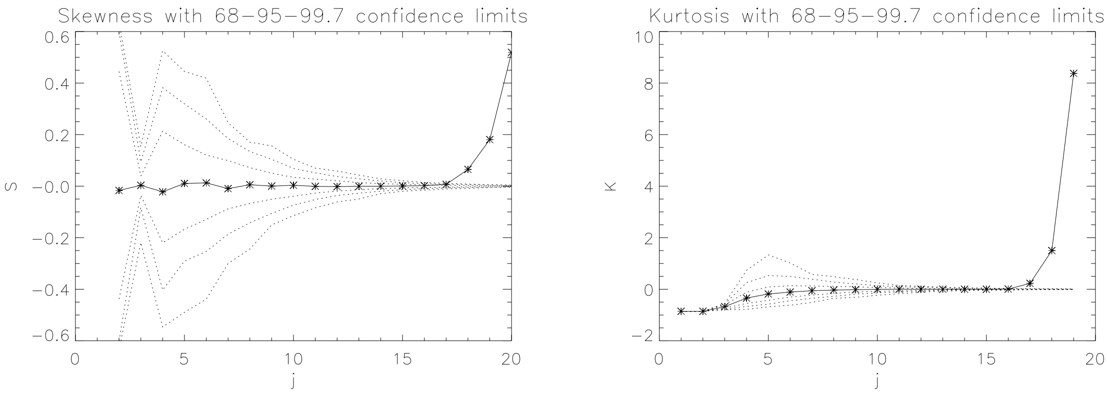
Figure 6.14: The skewness (a) and kurtosis (b) as a factor of σ_S and σ_K for various values of a. The standard deviations have been averaged over 100 simulations, hence the instability. The largest deviation occur somewhere around the peaks between $a = 1.2$ and $a = 1.6$.



(a) Skewness

(b) Kurtosis

Figure 6.15: Finer plots of skewness (a) and kurtosis (b) around the a's that in figure 6.14 measured the largest deviations, now averaged over 1000 simulations. The largest skewness of $S = 381\sigma_S$ is measured at $a = 1.39459$, and the largest kurtosis of $K = 2866\sigma_K$ is measured at $a = 1.41798$.



(a) Skewness **(b)** Kurtosis

Figure 6.16: The skewness (a)and kurtosis (b) with 68 %, 95 % and 99.7 % confidence intervals for all j for the needlets that measured the largest deviation from Gaussianity. For skewness, the needlet with $a = 1.39459$ is shown, and for kurtosis, the needlet with $a = 1.41798$ is shown. The confidence intervals are based on data from 1000 simulations, while the skewness and kurtosis have been calculated after 100 simulations. The measurements at large scales (small j) are unreliable

6.2.3 Estimation of skewness and kurtosis for a given amplitude

As a first step towards calculating the deviation of the power spectrum due to the unresolved point sources, the skewness and kurtosis will be expressed as a function of the amplitude A of the point sources. Skewness and kurtosis models for the χ^2 minimization introduced in section 4.1.2 may be more quickly calculated analytically than through simulations. Thus, it is desirable with analytical expressions for $S(A)$ and $K(A)$.

The total temperature of the sky map in pixel i is a sum of the temperature of the background and the temperature of the foreground:

$$T_i^{\text{tot}} = T_i^{\text{cmb}} + T_i^{\text{ps}} = T_i^{\text{cmb}} + \sum_j A\delta_{ij}, \tag{6.1}$$

where equal amplitude for all the point sources has been assumed for simplicity. If a point source exists at the index i, that is if index i and j are equal, the amplitude A is added to the temperature value of the background at this pixel. Using equations (3.8) and (4.7), the wavelet coefficients of the sky map can be expressed as

$$C_i = \sum_{lm} a_{lm} g_l Y_{lm}^i,$$

and inserting this into equation (4.1) for skewness yields

$$S = \frac{1}{N\sigma^3} \sum_i \left(\sum_{lm} a_{lm} g_l Y_{lm}^i \right)^3.$$

The amplitude is gained into this expression by using equations (3.8) and (6.1):

$$S = \frac{1}{N\sigma^3} \sum_i \left\{ \sum_{lm} \left(\sum_k \left[\left(T_k^{\mathrm{cmb}} + AD_k \right) Y_{lm}^k \right] \right) g_l Y_{lm}^i \right\}^3 , \tag{6.2}$$

where

$$D_k = \sum_j \delta_{kj}.$$

The sum over k in equation (6.2) can be split up:

$$S = \frac{1}{N\sigma^3} \sum_i \left\{ \sum_{lm} \left(\sum_k T_k^{\mathrm{cmb}} g_l Y_{lm}^k Y_{lm}^i + A \sum_k D_k g_l Y_{lm}^k Y_{lm}^i \right) \right\}^3 .$$

If one defines

$$P_i = \sum_{lm} \sum_k T_k^{\mathrm{cmb}} g_l Y_{lm}^k Y_{lm}^i$$

and

$$Q_i = \sum_{lm} \sum_k D_k g_l Y_{lm}^k Y_{lm}^i,$$

the equation can be simplified to

$$S = \frac{1}{N\sigma^3} \sum_i \left(P_i + AQ_i \right)^3 . \tag{6.3}$$

A similar approach with the variance in equation (4.3) gives

$$\sigma^2 = \frac{1}{N} \sum_i \left(P_i + AQ_i \right)^2 ,$$

such that equation (6.3) finally becomes

$$S(A) = N^{1/2} \frac{\sum_i \left(P_i + AQ_i \right)^3}{\left(\sum_i \left(P_i + AQ_i \right)^2 \right)^{3/2}},$$

where

$$\left(P + AQ \right)^3 = P^3 + 3P^2 AQ + 3P(AQ)^2 + (AQ)^3$$

and

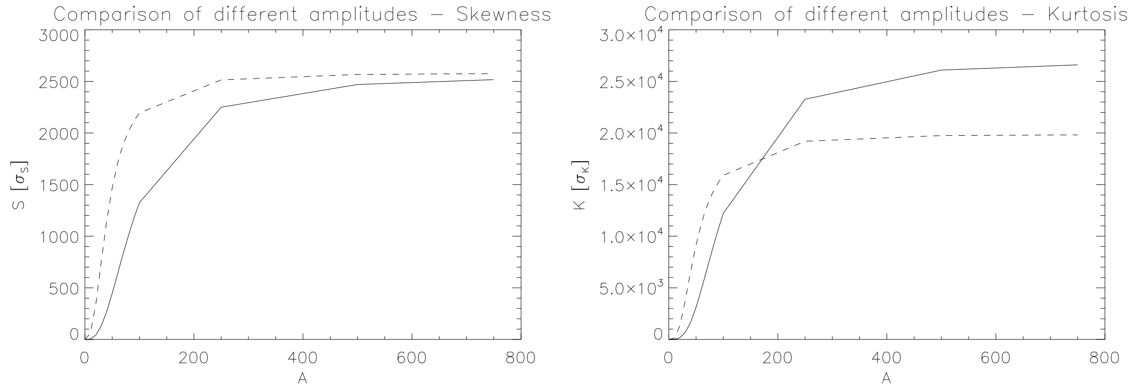
$$\left(P + AQ \right)^2 = P^2 + 2PAQ + (AQ)^2.$$

When the amplitude is large, the cubic term in the numerator and the quadratic term in the denominator dominate. The denominator ends up going as A^3, such that S approaches a constant. Similarly, the expression for the kurtosis becomes

$$K(A) = N \frac{\sum_i \left(P_i + AQ_i \right)^4 - 3}{\left(\sum_i \left(P_i + AQ_i \right)^2 \right)^2},$$

which also approaches a constant when A is large.

The analytical expressions above is confirmed by simulations for large A. In figure 6.17, which show how the skewness and kurtosis behave with different point source amplitudes for the best SMH wavelet and needlet, the curves converge towards a constant for large amplitudes. The SMH wavelet and needlet follow approximately the same pattern. Unfortunately, the analytical expressions above are complicated, and complicated analytical expressions may themselves require numerical approaches. Simulations must therefore be used to calculate the skewness and kurtosis of different amplitudes needed for the χ^2 minimization.



(a) Skewness

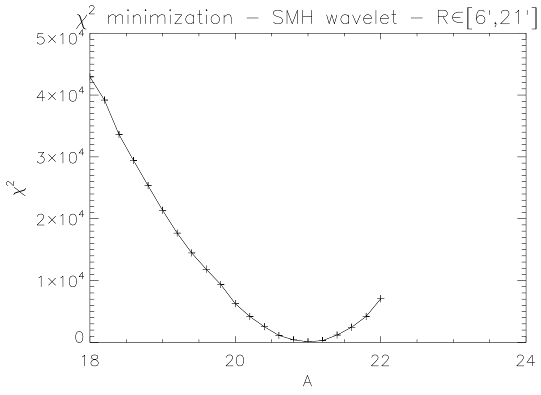
(b) Kurtosis

Figure 6.17: Skewness and kurtosis as a function of constant point source amplitude. (a) Skewness for the SMH wavelet of scale $R = 8.52'$ (solid line) and needlet $a = 1.39459$ of scale $j = 20$ (dashed line). (b) Kurtosis for the SMH wavelet of scale $R = 7.68'$ and needlet $a = 1.41798$ of scale $j = 19$.

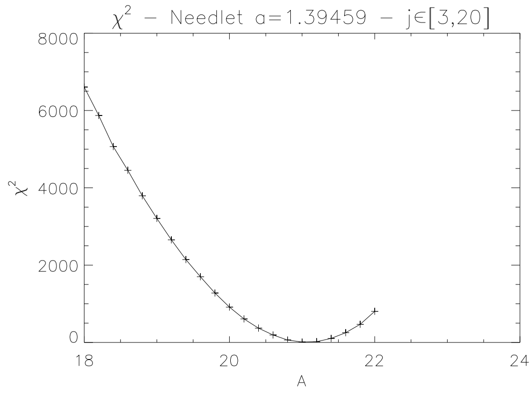
6.2.4 χ^2 minimization and correction to the power spectrum

The χ^2 minimization gives a measure of which model fits the data best. Since the analytical models for $S(A)$ and $K(A)$ were intricate, simulations were made to create the models. For each constant A, 100 simulations were performed, and their mean skewness and kurtosis were used in equation 4.4. The observations to be compared to the model were simulated with the same number of simulations. The amplitude of the simulated observations was chosen to be $A = 20$, but recall that in reality this amplitude is unknown and non-constant. The different scales of the SMH wavelet and a chosen needlet serve as the observations n in the equation. For the SMH wavelet, the equation was summed over scales around the largest deviation in figure 6.12, that is where $R \in [6', 21']$, where each scale is separated by $0.5'$. A needlet around the largest peak for both skewness and kurtosis in the plot of the a's in figure 6.15 was chosen, but the largest scales were avoided due to the uncertainty.

The χ^2 minimizations from one simulation for the SMH wavelet and the needlet are shown in figure 6.18, where the amplitude of the simulation has been determined to be near $A = 21$ in both cases. The power spectrum of each simulation is then corrected by using the model power spectra for the amplitude determined by the χ^2 minimization and for no amplitude. The corrected power spectra by using the SMH wavelet and the needlet are shown in figure 6.19 and 6.20 with the uncertainty of the estimation, where the impact of the cosmic variance has been subtracted from the uncertainty. The error bars increase towards the small angular scales, where larger corrections to the amplitude of the power spectrum has been performed due to the point sources. The error bars are marginally larger at the smallest angular scales for the SMH wavelet. This small difference and the general increase in variance towards smaller scales are slightly more evident in figure 6.21, which shows a plot of the standard deviation of the two power spectra.



(a) SMH wavelet



(b) Needlet

Figure 6.18: The χ^2 minimization for one simulation using several scales of the SMH wavelet (a) and the needlet $a = 1.39459$ (b). Both minimizations find the amplitude of the data to be closest to the amplitude near $A = 21$ of the model.

6.3 Detection with noise

The noise has been introduced in the analysis for a more realistic comparison of the performance of the different wavelets. The next couple of tests will briefly go through the highlights of the tests done in the previous sections.

6.3.1 Detection of resolved point sources

When point source detection[1] is performed at the 5σ level on the sky map with no wavelets applied, 59 true point sources and 14 false point sources are detected. 1000 simulations were used for generating the thresholds, and 100 was used for the detection of point sources. The number of true detections has decreased by about 250 compared to the noiseless simulations, while the number of false detections has increased by a a little less than 15. If the point sources are removed from the map, the latter number becomes 1, which is consistent with Gaussian statistics. Therefore, the larger false detection count is due to the point sources.

[1]One note on the chosen intensity of the point sources in section 5.2. Recall that the intensity was based on when the code detected about the same amount of point sources as the 3-year analysis of the WMAP data. Since noise was not present in the analysis at that stage, the intensity does not correspond in any way to the measurements of WMAP. Nevertheless, the same intensity has been used also in the simulations with noise, so that the comparison between the earlier simulations will still be valid.

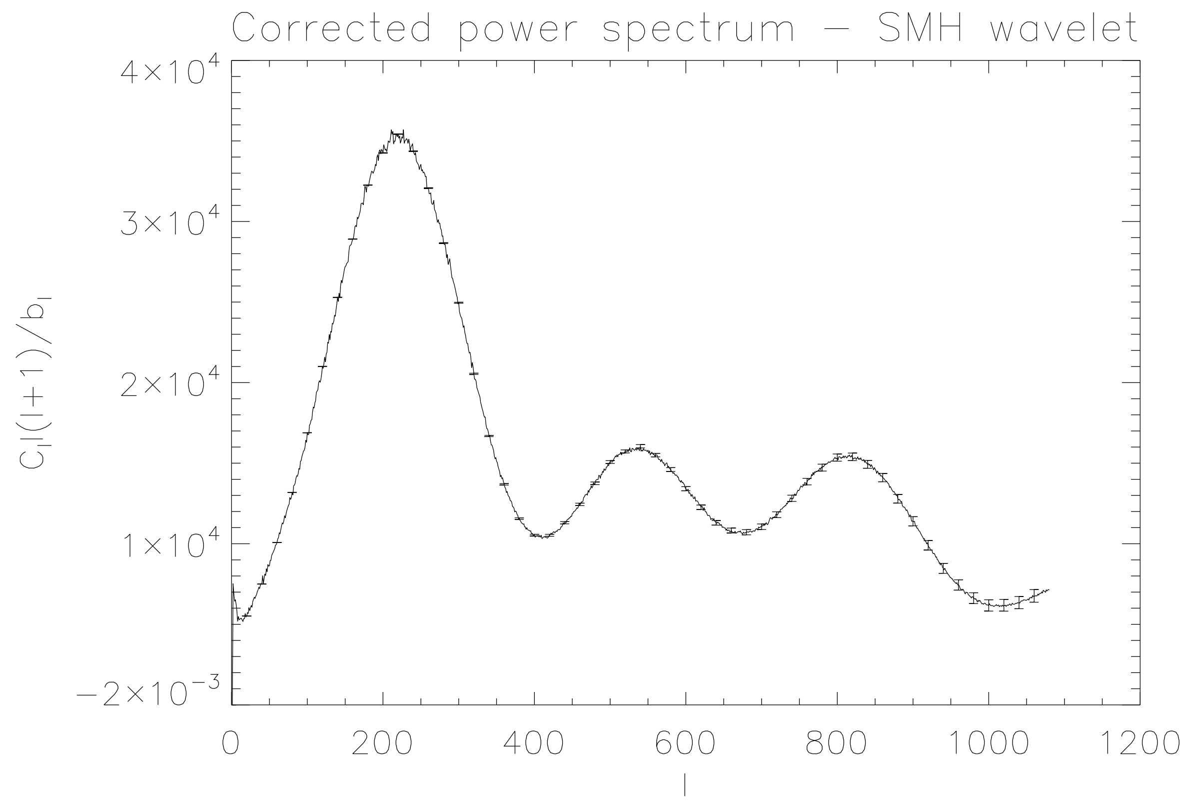


Figure 6.19: Corrected power spectrum with error bars using the SMH wavelet, where the cosmic variance has been subtracted from the error bars. The error increases towards the smaller angular, where the amplitude caused by the point sources was greater before the correction. The error is marginally larger at the smallest angular scales compared to the corrected power spectrum by using needlets, shown in figure 6.20. The small difference is slightly more evident in 6.21.

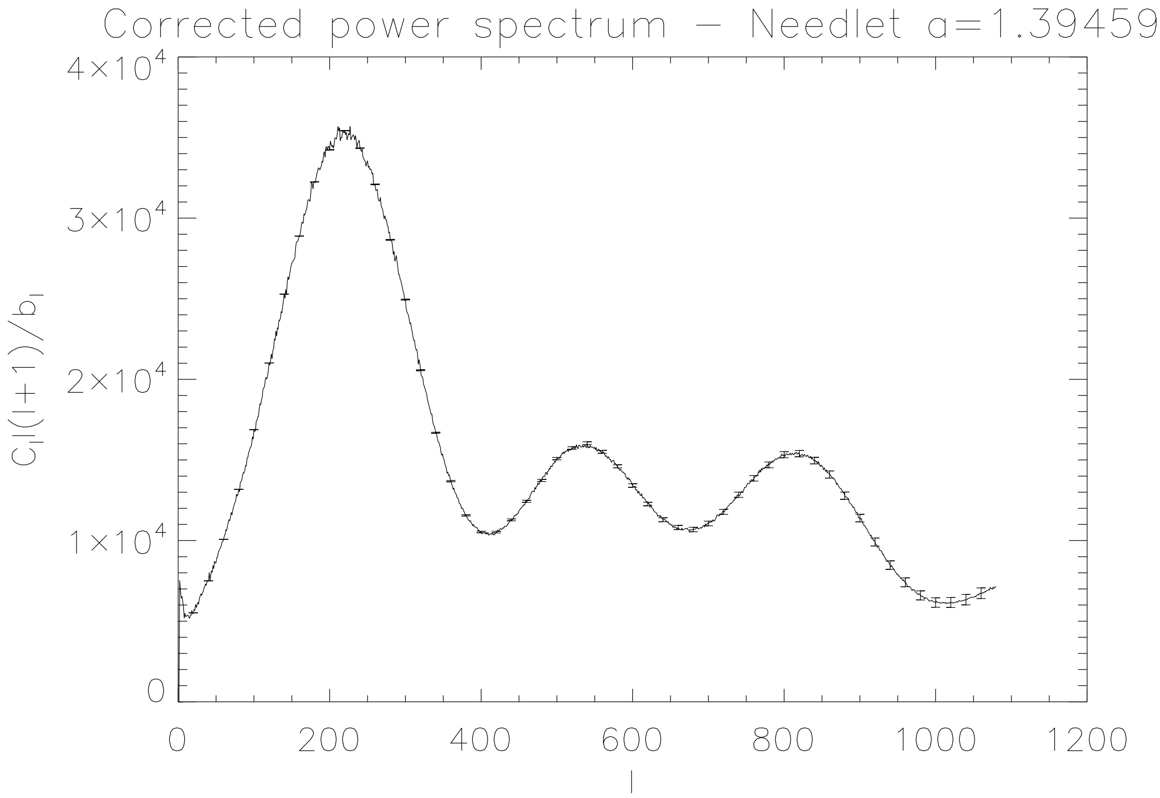
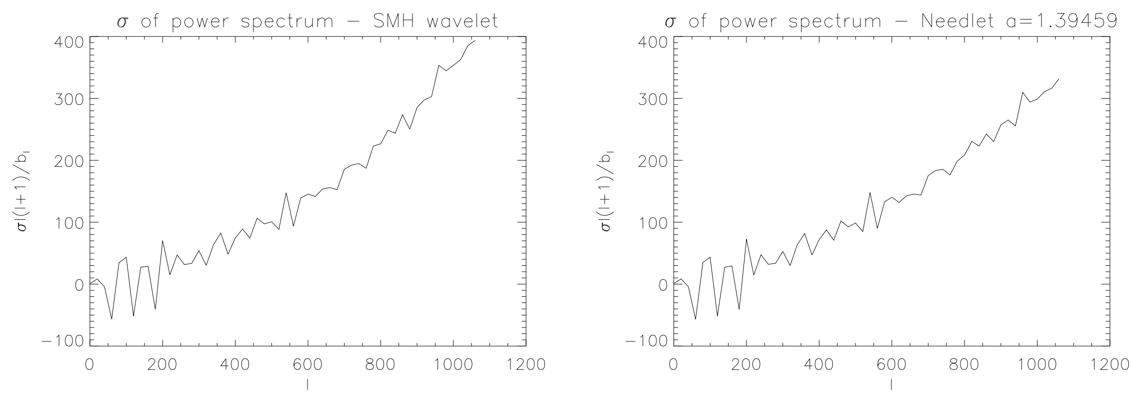


Figure 6.20: Corrected power spectrum with error bars using the needlet $a = 1.39459$. See figure 6.19 for details.



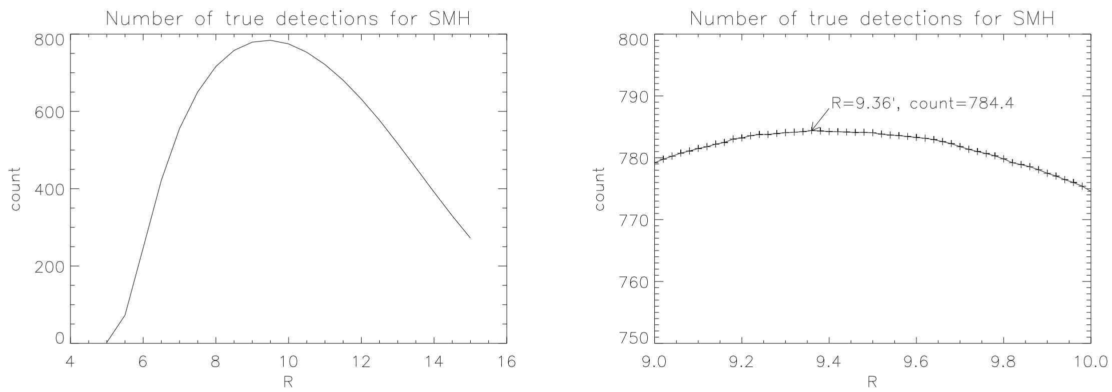
(a) SMH wavelet

(b) Needlet

Figure 6.21: The standard deviation of the power spectra in figure 6.19 and 6.20, where the cosmic variance has been subtracted from the variance. The error increases towards the smaller scales, and a marginally larger error for the SMH wavelet compared to the needlet is noted at the smallest scales.

The extra false detections are caused by pixels near smoothed true point sources, where all the pixels are slightly smaller than the threshold. Due to the noise, some of these pixels can get a flux slightly above the threshold, while the true point sources remain smaller. A code was not incorporated to count such detections as true also when the noise was not simulated, since the number of these occurrences were few when the number of detections was large.

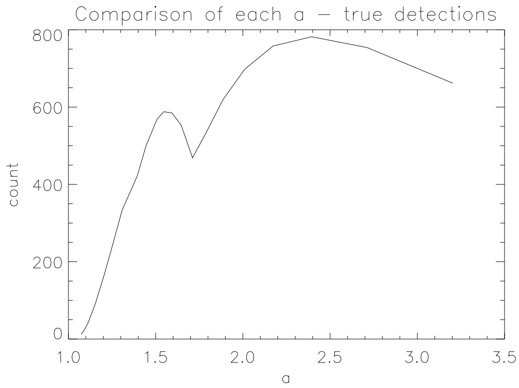
The number of true detections for the SMH wavelet filtered sky map is plotted in figure 6.22. The number of false detections is not plotted, but generally lies around a count of 10 for each scale, and appear due to the noise and the few true detections as stated above. Compared to the case without noise in figure 6.2, the true detection count is smaller and the scale giving the highest number of true detections has shifted to larger scales where the noise is less present. According to the fine plot around the peak in the figure, the best scale is now $R = 9.36'$ with 784 true detections.



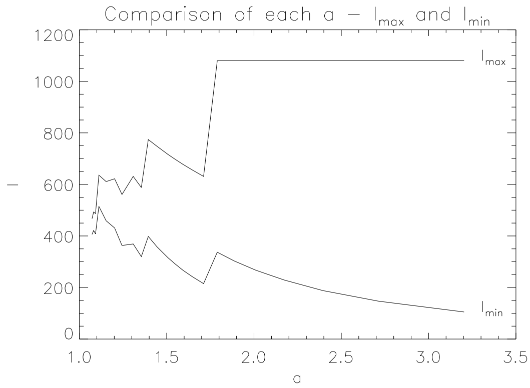
(a) $R \in [5', 15']$ **(b)** $R \in [9', 10']$

Figure 6.22: The number of true detections on sky maps with noise after wavelet transformation with the SMH wavelet for selected R values. The plot (a) has shifted to higher scales where the noise is less present. According to the rightmost fine plot (b), the best scale is now $R = 9.36'$ with around 784 true detections.

In figure 6.23, the needlets from table 6.5 have been plotted with an accompanying plot of the multipoles the needlet is defined for. This time, however, the highest j does not always give the best results, and the j with the largest detection rate for each needlet is therefore used in the plot. For the largest a these are the highest j, while for the smaller a, the best j is slightly lower than the highest. There are two peaks in the curve of the figure, one at $a = 1.54735/j = 14$ with $l \in [292, 698]$ and the other at $a = 2.39429/j = 7$ with $l \in [188, 1080]$. The two peaks are due to the differences in the covered multipoles. Around the first peak, the scales reaching for lower multipoles gave higher detection counts than those that covered the higher multipoles at the expense of the low multipoles. At the turning point between the curves, this changes to the opposite. Now covering high multipoles give better results, and the detection rate increases as more and more of the lower multipoles are once again covered. The number of false detections lies steadily around a count of 10-20 for all a, since the best detection rate is achieved at lower j than earlier. In figure 6.24, more needlets with multipole ranges close to that of the peak in figure 6.23(a) have been tested, but the figure reveals minor changes. The best detection rate of 796 point sources is accomplished with the needlet $a = 2.385/j = 7$, which is not defined for the very largest l, where there is more noise.



(a) true detections



(b) l interval

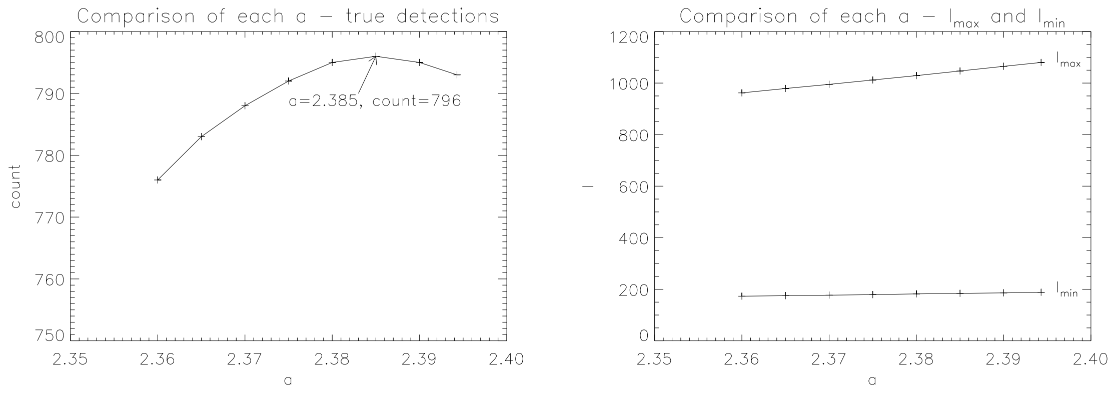
Figure 6.23: (a) The number of true detections on sky maps with noise after wavelet transformation with a selection of needlets. (b) The accompanying multipole range of the needlets. The largest peak occur for needlets that cover a large range of multipoles.

Overall, the number of true detections sinks when there is noise present compared to when there is not, and the larger wavelet scales are less affected by noise. In figure 6.25, three sky maps with noise applied are shown. The first sky map is non-filtered, the second is filtered with the SMH wavelet of scale $R = 7.32'$ and the third is filtered with the $a = 1.096259$ needlet of scale $j = 75$. Previously, both these wavelets detected most point sources. Note the difference between these sky maps to the ones in figure 5.4(a) and 5.5(a). Since the noise is present on the small scales where the point sources dwell, they do not get filtered with the small scale wavelet. This entails an increase in the variance of the filtered sky map, the threshold become higher and fewer point sources are detected. In particular, note that the performance of needlets is now on level with SMH wavelets, with only a few detections more. The needlets performed best when localized at high l in multipole space, but these multipoles are now dominated by noise.

6.3.2 Detection of unresolved point sources

The deviation of skewness and kurtosis as a function of σ_S and σ_K for the SMH wavelet transformed maps with noise for $R \in [5', 50']$ is presented in figure 6.26. Previously, without noise, kurtosis registered the largest deviation, but now skewness performs better. The peak has now shifted to around $R = 13'$, compared to around $R = 8'$ before. Plots in a smaller interval around the peak is given in figure 6.27, where the largest deviation is measured at $R = 12.12'$ with $S = 6.9\sigma_S$ and at $R = 13.34'$ with $K = 3.1\sigma_K$.

An overview of kurtosis and skewness for all needlets where the deviation is larger than the 99.7% confidence level is found in figure 6.28, and a finer plot around the peaks is found in figure 6.29. The standard deviations of the data in the first mentioned plots was made from measurements of 100 simulations, while they in the latter was made from 1000 simulations.



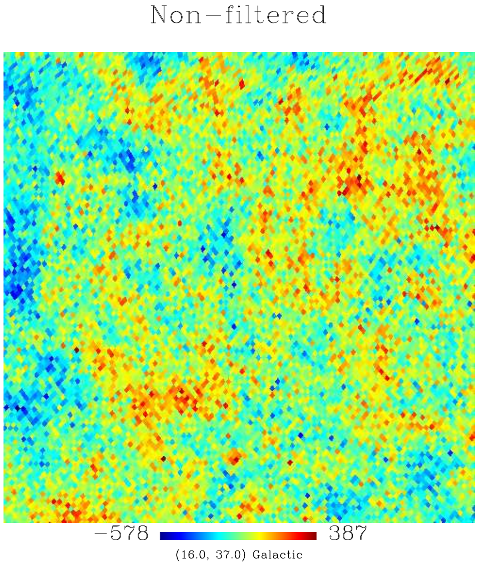
(a) true detections

(b) l interval

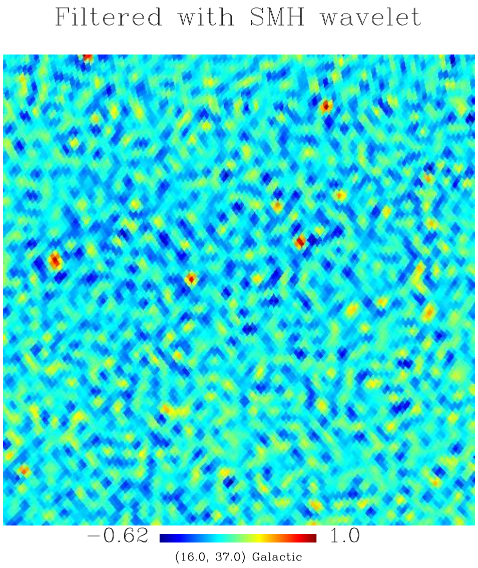
Figure 6.24: (a) Finer plot of the number of true detections around the largest peak in figure 6.23. (b) The accompanying multipole range of the needlets. The detection rate reaches its height of 796 point sources for $a = 2.385/j = 7$, where $l \in [184, 1047]$.

For needlets, the largest measured deviation has moved to much larger scales than before, in particular for kurtosis. Smaller peaks can also be seen prior to the largest measured deviation. For skewness the largest peak is found at $a = 2.39429/j = 7$ with a deviation of $S = 15.7\sigma_S$, while the peak for kurtosis is found at $a = 4.04282/j = 4$ with $K = 2.7\sigma_K$. These needlets and their scales correspond to the multipole ranges $l \in [188, 1080]$ and $l \in [66, 1080]$ respectively.

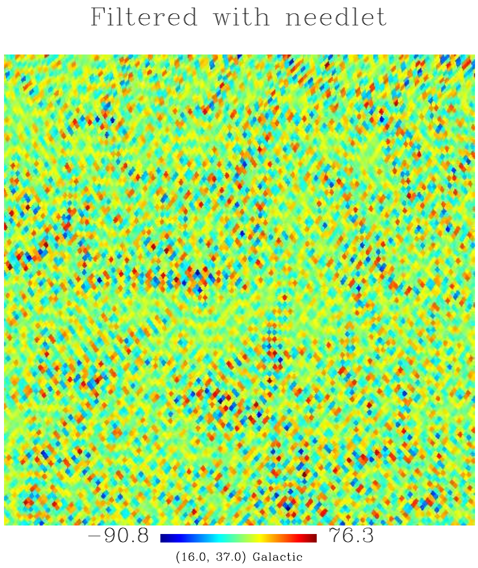
For the resolved point sources, the differences in performance between the SMH wavelet and needlets were minor. For the unresolved point sources, however, needlets measure larger deviations than the SMH wavelet for skewness, while there is nearly no difference for kurtosis. The multipole ranges are very large for the best needlets. It is apparent that it is favourable with information from most multipoles when there is noise present, and since the SMH wavelets are also defined for a large range of multipoles, the differences between the two wavelet types have evened out.



(a) Non-filtered

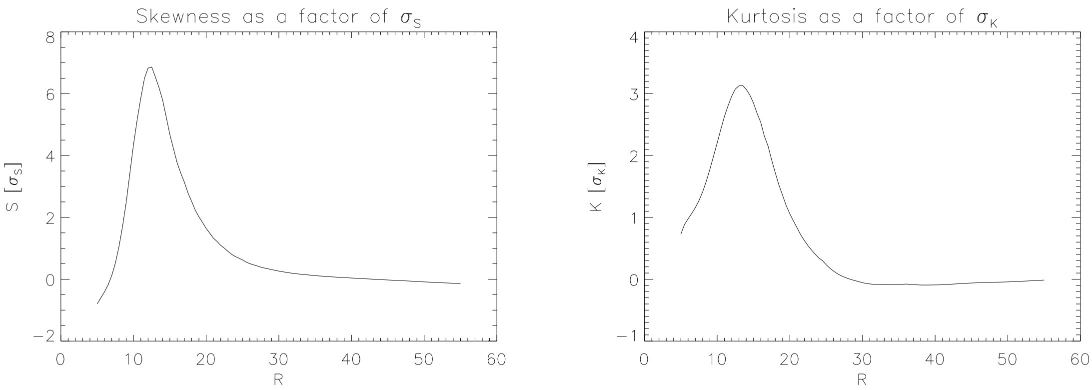


(b) SMH wavelet $R = 7.32'$



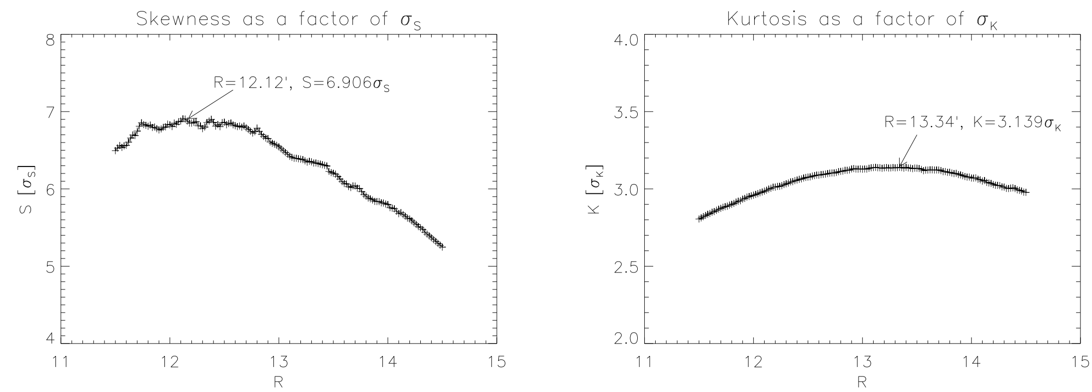
(c) Needlet $a = 1.096259/j = 75$

Figure 6.25: Three maps (at the same galactic coordinates) affected by noise. Figure (a) is non-filtered, figure (b) is filtered with the SMH wavelet of scale $R = 7.32'$ and figure (c) is filtered with the needlet $a = 1.096259$ of scale $j = 75$. Both the two latter cases detected the largest numbers of point sources when there was no noise present in the maps. The figures show how the noise, due to being present at the small scales as the point sources, still is present after the filtration process.



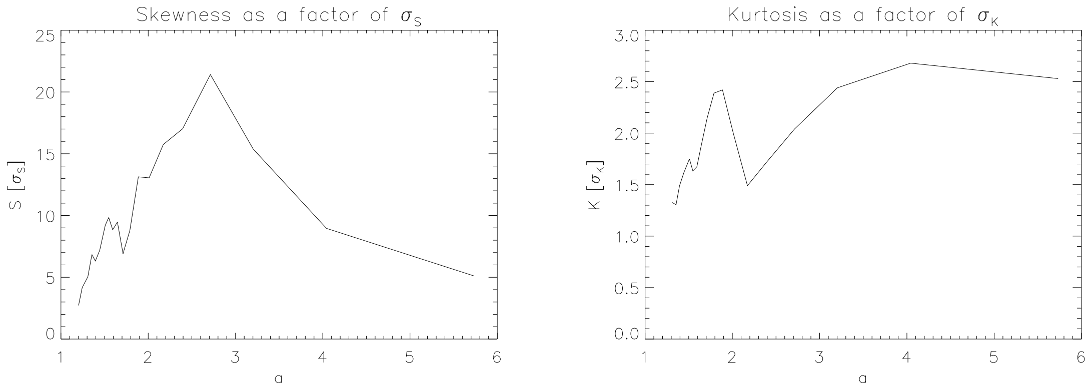
(a) Skewness **(b)** Kurtosis

Figure 6.26: Skewness (a) and kurtosis (b) for the SMH wavelet transformed maps with noise. The peak of the curve lies around $R = 13'$, which is on a larger scale than the case without noise.



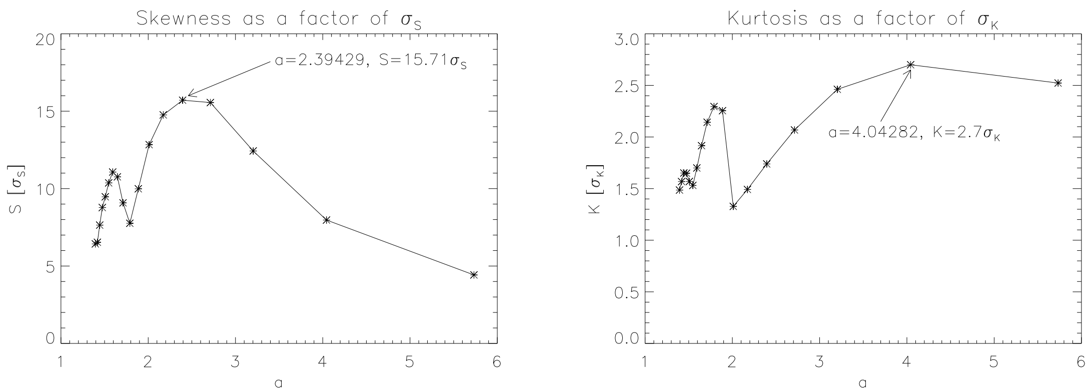
(a) Skewness **(b)** Kurtosis

Figure 6.27: Finer plots of the skewness (a) and kurtosis (b) in figure 6.26 for the SMH wavelet transformed maps with noise. The largest deviation is measured at $R = 12.12'$ with $S = 6.9\sigma_S$ and at $R = 13.34'$ with $K = 3.1\sigma_K$.



(a) Skewness **(b)** Kurtosis

Figure 6.28: Skewness (a) and kurtosis (b) based on simulations of 100 sky maps with noise, filtered with various needlets. The scale j giving the largest deviation is shown. For skewness the largest deviation occur between the very large scales $a = 2$ and $a = 3$, and for kurtosis at even larges scales around $a = 4$. Smaller peaks are also found prior to the large peaks at the large scales.



(a) Skewness **(b)** Kurtosis

Figure 6.29: Plots of skewness (a) and kurtosis (b) concentrated around the peaks in figure 6.28, and based on data from 1000 simulations instead of 100. The largest deviation for skewness occur for $a = 2.39429/j = 7$ with $S = 15.7\sigma_S$, and for kurtosis for $a = 4.04282/j = 4$ with $K = 2.7\sigma_K$.

Chapter 7

Summary and conclusions

In between the Cosmic Microwave Background (CMB) and the instrument there are other sources of radiation, called foregrounds. The foregrounds radiate in the same frequency bands as the background, and cause contamination in the signal of the CMB. To ensure no wrong estimations of the power spectrum, these contaminants must be removed. There are several classes of contaminants, and this thesis has investigated techniques of removing the point sources (class of extragalactic sources), which occupy just a few pixels on the full sky and are located at the small angular scales.

The wavelet filtering technique was used for point source detection, and in particular, new types of wavelets called needlets. The wavelets have good localization properties in both real and harmonic space, and are characterized by an underlying mother wavelet which can be fine-tuned by scaling and translation. These properties make wavelets particularly useful for point source detection. The wavelets can enhance the scales in harmonic space where the point sources are located, such that they are more easily separated from the background. The most used wavelet for detection of non-Gaussianity today is the Spherical Mexican Hat (SMH) wavelet, but needlets are showing promising additional features that make them a possible candidate of succession. These wavelets enjoy direct definition on the sphere, can be more localized in harmonic space and are more scalable than the SMH wavelet. The mother wavelet of needlets is defined by the parameter a and their scale by j, while the SMH wavelet has only one mother wavelet and is scaled through R. This thesis has investigated the performance of needlets compared to the SMH wavelet.

Detecting the point sources that can be resolved from the distribution of the CMB was the first goal, and several Monte Carlo simulations of the detection ability of each wavelet was performed to achieve statistically significant results. At the 5σ threshold, the SMH wavelet detected 1426 point sources at the best scale $R = 7.32'$. Needlets performed best for $a = 1.096259$ of scale $j = 75$ with 1744 detections. However, this needlet is badly localized in pixel space, and generates many extra false point source detections not associated with those predicted by Gaussian statistics. In general, high a have very bad localization properties on the sphere, and are not suitable for point source detection. These findings are consistent with those made by Marrinucci et al. [18]. If the needlets at high a was to be used in real observational data analysis, too much of the information in the CMB would be removed (masked). The best needlet, that does not generate many extra false point sources and leave a high percentage of the full sky masked, was found to be $a = 1.44428$ of scale $j = 18$ with 1667 detections, ~ 240 more than the SMH wavelet.

Real observations, however, contain noise that are located on the small angular scales like the point sources, and makes the detection of point sources harder. The first part of the analysis showed that needlets perform better than the SMH wavelet at the smallest angular scales, but the noise are more present at these scales and makes them unusable for point source detection. With the introduction of noise, the performance of the needlets compared to the SMH wavelet evened out. Now the best scale for the SMH wavelet was $R = 9.36'$ with 784 detections, and the best scale for needlets was $a = 2.385$ of scale $j = 7$ with 796 detections.

The second goal of the thesis was to see the ability of the two wavelets to detect point sources that can not be resolved from the distribution of the CMB. Using the skewness and kurtosis statistics, the deviation of the distribution from Gaussianity caused by the point sources was checked. Without noise, the largest deviation was measured with needlets, but when the power spectrum was corrected using the χ^2 minimization technique, very little difference in the ability to correct the power spectrum between the two wavelets was found. With noise, the ability of needlets to measure deviations with the kurtosis statistic was evened out, while slightly larger deviations were still measured with skewness. Corrections to the power spectrum in the noise analysis was not performed, but due to the smaller skewness and kurtosis deviations measured with noise, one can assume larger errors and smaller differences in the ability to correct the power spectrum between the wavelets than without noise.

Table 7.1 and 7.2 summarizes the best measured performance of the needlets and the SMH wavelet, with and without noise. When confronted with realistic simulations, this work has shown no indication of one wavelet being significantly better than the other.

It is important to note that the main goal of the analysis of the thesis was to compare the efficiency of the SMH wavelet compared to needlets, and therefore the simple case of one single frequency was used. Simulation at different frequencies should be considered for further investigation of how one can best take advantage of the properties of Spherical needlets. Also this thesis has not tested the point source detection algorithm of both the SMH wavelet and needlets on WMAP data, and no comparison with the method used by Wright et al. [24] could then be performed. The WMAP data can contain different amount of point sources than what was assumed here, and the Milky Way Galaxy must also be taken into account in such an analysis. Corrections of the power spectrum due to the point source amplitudes was briefly covered here, and should be further examined, both including noise and for the resolved point sources. Making precise corrections to the power spectrum is of crucial importance in cosmology, which is reconfirmed by the recent discoveries of Huffenberger et al. [16].

	a	R/j	Detections
SMH wavelet (no noise)	-	$7.32'$	1426
Needlet (no noise)	1.44428	18	1667
SMH wavelet (noise)	-	$9.36'$	784
Needlet (noise)	2.385	7	796

Table 7.1: Summary of the best results for the detection of resolved point sources

	a	R/j	S [σ_S]	a	R/j	K [σ_K]
SMH wavelet (no noise)	-	8.52	54.4	-	7.68	283
Needlet (no noise)	1.39459	20	381	1.41798	19	2870
SMH wavelet (noise)	-	12.12	6.9	-	13.34	3.1
Needlet (noise)	2.39429	7	16	4.04282	4	2.7

Table 7.2: Summary of the best results for the detection of unresolved point sources

Appendix A

Source code

A.1 Detection of resolved point sources

Listing A.1: psw_par.f90

```
PROGRAM psw
  USE psw_sub

  INTEGER(I4B)          :: n_pols, iseed, midpix
  CHARACTER(LEN=128)  :: healpixdir, filename
  TYPE(PLANCK_RNG)  :: rng_handle

  ! Necessary for parallelization
  CALL MPI_INIT(ierr)
  CALL MPI_COMM_SIZE(MPI_COMM_WORLD, ntasks, ierr)
  CALL MPI_COMM_RANK(MPI_COMM_WORLD, me, ierr)

  ! Find Healpix-directory
  CALL getEnvironment("HEALPIX",healpixdir)

  ! Set parameters
  filename = 'params_psw.txt'
  CALL get_params(filename)

  ! Set standard values derived from the parameters
  npix=nside**2*12
  n_pols = 1 + 2*polar ! either 1 or 3
  iseed = start_seed+me
  midpix = npix/2
  pixsize = SQRT((4_dp*pi)/npix)

  ! Necessary for parallelization
  ALLOCATE(N_max_pp(0:ntasks-1))
  ALLOCATE(stat(0:MPI_STATUS_SIZE-1))

  ! Code to distribute number of N evenly to each CPU, and if there's a
  ! remainder from the division, the remaining N are added to the first CPUs
  N_max_pp=N_max1/ntasks
  IF(MOD(N_max1,ntasks).NE.0) THEN
     N_max_pp(0:MOD(N_max1,ntasks)-1)=N_max_pp(0:MOD(N_max1,ntasks)-1)+1
  END IF

  ! Allocate memory for arrays
  CALL alloc(n_pols)

  ! Preperation of the array containing the scales
  IF(wavelet) THEN
     IF(smh) THEN
        scales(1) = scale_start
        DO i=2,nscales
           scales(i) = scales(i-1) + add
        END DO
     END IF
  ELSE
     scales(1) = 1
  END IF

  zbounds=[-1,1]
  w8ring_TQU=1
  mask = 1
  sigma_s = 0d0
  sigma_noise = 0d0
  false_mean = 0
```

```
true_mean = 0
mask_mean = 0

! The unit for file opening is different for each CPU, where me is
! the CPU number
unit=10+me

! Fetches the beam from file
OPEN(unit, file='MAP_blxwl_avgv_opt.unf', form='unformatted',status='old')
REWIND(unit)
READ(unit) beam
CLOSE(unit)

! Fetches the noise from file
OPEN(unit, file='MAP_noise_avgv.unf', form='unformatted',status='old')
REWIND(unit)
READ(unit) noise
CLOSE(unit)

IF(wavelet) THEN
   IF(me .EQ. 0) PRINT *, "Generating wavelets..."
   IF(smh) THEN

      ! Finds g_l for the SMH wavelets at the defined scales
      CALL calc_gl_smh(nside, lmax, nscales, scales, gl, .FALSE., .TRUE., glfile, me)
   ELSE
      ! Finds g_l for the needlets at the defined scales
      CALL calc_f2(f2,nn)
      CALL calc_gl(f2,nn,j0,nj,lmax,gl,aa)

      OPEN(unit, file='gl_psw.unf', form='unformatted',status='unknown')
      REWIND(unit)
      WRITE(unit) gl
      CLOSE(unit)
   END IF
END IF

! Transfer iseed to rng_handle, from now on, use rng_handle in calls to
! routines using random generator
CALL rand_init(rng_handle, iseed) ! takes up to 4 seeds simultaneously

! Generates CMB maps to determine a value for sigma_CMB
IF(me .EQ. 0) PRINT *, "Calibrating sigma..."
DO, i_N=1, N_max_pp(me)

   ! Finds which N to give to this CPU
   IF (me .EQ. 0) THEN
      N=i_N ! The first cpu just gets the first N
   ELSE
      ! Sums up all the N given to the previous CPUs such that
      ! the index starts off at the correct N
      N=SUM(N_max_pp(0:me-1))+i_N
   END IF

   CALL find_sigma(iseed, rng_handle, fwhm_arcmin)
END DO

! Takes the sum of sigma_CMB and sigma_s from all cpus, and puts the
! result in all CPUs
CALL reduce_sigma

! Necessary for parallelization
DEALLOCATE(N_max_pp)
DEALLOCATE(stat)
ALLOCATE(N_max_pp(0:ntasks-1))
ALLOCATE(stat(0:MPI_STATUS_SIZE-1))

! Code to distribute number of N evenly to each CPU, and if there's a
! remainder from the division, the remaining N are added to the first CPUs
N_max_pp=N_max2/ntasks
IF(MOD(N_max2,ntasks).NE.0) THEN
   N_max_pp(0:MOD(N_max2,ntasks)-1)=N_max_pp(0:MOD(N_max2,ntasks)-1)+1
END IF

CALL realloc(n_pols)

! Generates CMB maps to simulate detection of point sources
IF(me .EQ. 0) PRINT *, "Detecting point sources..."
DO, i_N=1, N_max_pp(me)

   IF (me .EQ. 0) THEN
      N=i_N
   ELSE
      N=SUM(N_max_pp(0:me-1))+i_N
   END IF

   CALL detect_ps(iseed, rng_handle, fwhm_arcmin, midpix)
END DO

! Print results to screen and write to file
```

```
  CALL dump_results

  ! Deallocate the memory used for arrays
  CALL dealloc

  ! Necessary for parallelization
  CALL MPI_FINALIZE(ierr)

END PROGRAM psw
```

Listing A.2: psw_sub_par.f90

```
MODULE psw_sub

  USE healpix_types
  USE alm_tools
  USE ran_tools
  USE pix_tools
  USE extension
  USE mod_domwav
  USE rngmod,    ONLY: rand_init, rand_gauss, planck_rng

  IMPLICIT NONE
  INCLUDE 'mpif.h' ! Necessary for parallelization

  INTEGER(I4B), DIMENSION(:), ALLOCATABLE     :: listpix, otherpix
  INTEGER(I4B), DIMENSION(:,:), ALLOCATABLE   :: ps_index
  REAL(SP), DIMENSION(:), ALLOCATABLE         :: map_TQU, scales, true_mean, false_mean, true_mean0,
       false_mean0, mask, map_pixel, pixel_mask, otherflux, mask_mean, mask_mean0, noise, noise_map,
       ps_amp, ps_amp0
  REAL(SP), DIMENSION(:,:), ALLOCATABLE       :: gl
  REAL(DP), DIMENSION(:), ALLOCATABLE         :: zbounds, vector, f2, sigma_s, sigma_s0
  REAL(DP), DIMENSION(:,:), ALLOCATABLE       :: beam, w8ring_TQU, ps_flux, sigma_noise,
      sigma_noise0
  COMPLEX(SPC), DIMENSION(:,:,:), ALLOCATABLE :: alm_TGC, alm_g, alm_test, alm_g_test, alm_noise,
      alm_g_noise
  CHARACTER(LEN=80), DIMENSION(1:180)         :: header_PS
  CHARACTER(LEN=128) :: clfile, glfile

  INTEGER(I4B) :: nside, lmax, polar, no_of_sources, nscales, N, i_N, l, i, s, j0, nj, nn, start,
      finish, start_seed
  INTEGER(I8B) :: N_max1, N_max2, npix
  REAL(SP)     :: fwhm_arcmin, add, scale_start, source_intensity, sigma_limit, sigma_test
  REAL(DP)     :: sigma_CMB, sigma_CMB0, aa, pixsize, disc_size
  LOGICAL(LGT) :: smh, knowps, addnoise, wavelet

  ! Necessary for parallelization
  INTEGER(I4B), DIMENSION(:), ALLOCATABLE     :: N_max_pp, stat
  INTEGER(I4B) :: ierr, ntasks, unit, cnt, dest, tag, src, me
CONTAINS

  SUBROUTINE get_params(filename)

    IMPLICIT NONE
    CHARACTER(LEN=128) :: line, name, value, filename, wlet, kps, addn
    INTEGER(I4B)       :: rstat
    LOGICAL(LGT)       :: exist

    ! Checks if the file exists on disk. trim cuts the blank characters
    ! away from filename
    INQUIRE(file=filename, exist=exist)
    IF (.NOT. exist) THEN
       PRINT *, "Error: File ", TRIM(filename) ," not found."
       STOP
    END IF

    ! Reads the file line for line. Scan finds the index of the specified
    ! character in the string. If there is no '=' on the line being read,
    ! or there is a comment '#' on the line, the do loop skips to the next
    ! line with cycle. Name contains the variable name, and value the value
    ! of the variable. If the name corresponds to one of the cases, the value
    ! of that name is inserted into the correct variable
    OPEN(unit, file=filename, form='formatted', iostat=rstat)
    DO WHILE(rstat .EQ. 0)
       READ(unit, fmt='(A)',iostat=rstat) line
       i = SCAN(line, '=')
       IF ((i .EQ. 0) .OR.(line(1:1) .EQ. '#')) CYCLE
       name = TRIM(ADJUSTL(line(:i-1)))
       value = TRIM(ADJUSTL(line(i+1:)))

       SELECT CASE(TRIM(name))
       CASE('nside')
          READ(value,*) nside
       CASE('lmax')
          READ(value,*) lmax
       CASE('N_max1')
          READ(value,*) N_max1
       CASE('N_max2')
```

```
            READ(value,*) N_max2
         CASE('no_of_sources')
            READ(value,*) no_of_sources
         CASE('source_intensity')
            READ(value,*) source_intensity
         CASE('sigma_limit')
            READ(value,*) sigma_limit
         CASE('polar')
            READ(value,*) polar
         CASE('start_seed')
            READ(value,*) start_seed
         CASE('fwhm_arcmin')
            READ(value,*) fwhm_arcmin
         CASE('wavelet')
            READ(value,*) wlet
            IF(wlet .EQ. 'smh') THEN
               wavelet = .TRUE.
               smh = .TRUE.
            ELSE IF(wlet .EQ. 'needlets') THEN
               wavelet = .TRUE.
               smh = .FALSE.
            ELSE IF(wlet .EQ. 'no') THEN
               wavelet = .FALSE.
               smh = .TRUE.
            ELSE
               PRINT *, "Error: ", TRIM(wlet) ," is not a valid wavelet."
               STOP
            END IF
         CASE('nscales')
            READ(value,*) nscales
         CASE('add')
            READ(value,*) add
         CASE('scale_start')
            READ(value,*) scale_start
         CASE('j0')
            READ(value,*) j0
         CASE('nj')
            READ(value,*) nj
         CASE('aa')
            READ(value,*) aa
         CASE('nn')
            READ(value,*) nn
         CASE('disc_size')
            READ(value,*) disc_size
         CASE('knowps')
            READ(value,*) kps
            IF(kps .EQ. 'true') THEN
               knowps = .TRUE.
            ELSE IF(kps .EQ. 'false') THEN
               knowps = .FALSE.
            ELSE
               PRINT *, "Error: knowps must be true or false."
               STOP
            END IF
         CASE('addnoise')
            READ(value,*) addn
            IF(addn .EQ. 'true') THEN
               addnoise = .TRUE.
            ELSE IF(addn .EQ. 'false') THEN
               addnoise = .FALSE.
            ELSE
               PRINT *, "Error: addnoise must be true or false."
               STOP
            END IF
         CASE('clfile')
            READ(value,*) clfile
         CASE('glfile')
            READ(value,*) glfile
         END SELECT
      END DO
      CLOSE(unit)

   END SUBROUTINE get_params

   SUBROUTINE alloc(n_pols)

      IMPLICIT NONE
      INTEGER(I4B) :: n_pols

      ALLOCATE(alm_TGC(1:n_pols,0:lmax,0:lmax))
      ALLOCATE(alm_test(1:n_pols,0:lmax,0:lmax))
      ALLOCATE(alm_g(1:n_pols,0:lmax,0:lmax))
      ALLOCATE(alm_noise(1:n_pols,0:lmax,0:lmax))
      ALLOCATE(alm_g_test(1:n_pols,0:lmax,0:lmax))
      ALLOCATE(alm_g_noise(1:n_pols,0:lmax,0:lmax))
      ALLOCATE(map_TQU(0:npix-1))
      ALLOCATE(map_pixel(0:npix-1))
      ALLOCATE(noise(0:npix-1))
      ALLOCATE(noise_map(0:npix-1))
      ALLOCATE(pixel_mask(0:npix-1))
```

```
    ALLOCATE(beam(0:lmax,1:n_pols))
    ALLOCATE(zbounds(1:2))
    ALLOCATE(w8ring_TQU(1:2*lmax, 1))
    ALLOCATE(mask(0:npix-1))
    ALLOCATE(vector(1:3))
    ALLOCATE(listpix(0:npix-1))
    ALLOCATE(otherpix(0:npix-1))
    ALLOCATE(otherflux(0:npix-1))

    IF(wavelet) THEN
       IF(smh) THEN
          start=1
          finish=nscales
          ALLOCATE(scales(start:finish))
       ELSE
          start=j0
          finish=j0+nj-1
          ALLOCATE(f2(0:nn-1))
       END IF
    ELSE
       start=1
       finish=1
       ALLOCATE(scales(start:finish))
    END IF

    ALLOCATE(gl(0:lmax,start:finish))
    ALLOCATE(sigma_s(start:finish))
    ALLOCATE(sigma_s0(start:finish))
    ALLOCATE(sigma_noise(start:finish, 0:npix-1))
    ALLOCATE(sigma_noise0(start:finish, 0:npix-1))
    ALLOCATE(true_mean(start:finish))
    ALLOCATE(false_mean(start:finish))
    ALLOCATE(mask_mean(start:finish))
    ALLOCATE(ps_amp(start:finish))
    ALLOCATE(true_mean0(start:finish))
    ALLOCATE(false_mean0(start:finish))
    ALLOCATE(mask_mean0(start:finish))
    ALLOCATE(ps_amp0(start:finish))

  END SUBROUTINE alloc

  SUBROUTINE realloc(n_pols)

    IMPLICIT NONE
    INTEGER(I4B) :: n_pols

    ALLOCATE(ps_index(1:no_of_sources, 1:N_max_pp(me)))
    ALLOCATE(ps_flux(1:no_of_sources, 1:N_max_pp(me)))

  END SUBROUTINE realloc

  SUBROUTINE dealloc

    DEALLOCATE(alm_TGC, alm_g, alm_test, alm_g_test, alm_noise, alm_g_ noise, map_TQU, map_pixel,
         noise, noise_map, pixel_mask, beam, ps_index, ps_flux, sigma_s, sigma_s0, sigma_noise,
         sigma_noise0, zbounds, w8ring_TQU, true_mean, false_mean, mask_mean, ps_amp, mask,vector,
         listpix, otherpix, otherflux, true_mean0, false_mean0, mask_mean0)

    IF(wavelet) THEN
       DEALLOCATE(gl)
       IF(smh) THEN
          DEALLOCATE(scales)
       ELSE
          DEALLOCATE(f2)
       END IF
    ELSE
       DEALLOCATE(scales)
    END IF

    ! Necessary for parallelization
    DEALLOCATE(N_max_pp, stat)

  END SUBROUTINE dealloc

  SUBROUTINE find_sigma(iseed, rng_handle, fwhm_arcmin)

    IMPLICIT NONE
    INTEGER(I4B)        :: iseed
    REAL(SP)            :: fwhm_arcmin
    TYPE(PLANCK_RNG)    :: rng_handle

    IF(me .EQ. 0) PRINT *, "Entering N=", i_N

    ! Creates a simulated CMB map
    CALL create_alm(nside, lmax, lmax, polar, clfile, rng_handle, fwhm_arcmin, alm_TGC, header_PS)

    ! Transforms the alm's to map and back again at the point in the code
    ! where point sources are added in the second MC loop
    CALL alm2map(nside, lmax, lmax, alm_TGC, map_TQU)
    CALL map2alm(nside,lmax,lmax,map_TQU,alm_TGC,zbounds,w8ring_TQU)
```

```

    ! Add beam effects to the created map (note that the pixel window
    ! function is included in this map)
    DO l=0, lmax
       alm_TGC(1,l,:) = alm_TGC(1,l,:)*beam(l,1)
    END DO

    ! Need the map for calculation of sigma_CMB
    CALL alm2map(nside, lmax, lmax, alm_TGC, map_TQU)

    ! Calculates the variance of the CMB map before noise
    sigma_CMB = sigma_CMB + SUM(map_TQU**2)

    ! Add noise effects to the created map
    IF(addnoise) THEN
       DO i=0,npix-1
          noise_map(i) = noise(i)*randgauss_boxmuller(iseed)*1000
       END DO
    END IF

    ! Need alm's for wavelet transformation
    CALL map2alm(nside,lmax,lmax,map_TQU,alm_TGC,zbounds,w8ring_TQU)
    CALL map2alm(nside,lmax,lmax,noise_map,alm_noise,zbounds,w8ring_TQU)

    ! Loop over all the chosen scales, which calculates the alm's for the
    ! wavelets and transforms them to map
    DO s=start, finish
       IF(me .EQ. 0) PRINT *,"Processing scale ",s, " of ", finish

       IF(wavelet) THEN
          DO l=0, lmax
             alm_g(1,l,:) = alm_TGC(1,l,:)*gl(l,s)
             alm_g_noise(1,l,:) = alm_noise(1,l,:)*gl(l,s)
          END DO
          CALL alm2map(nside, lmax, lmax, alm_g, map_TQU)
          CALL alm2map(nside, lmax, lmax, alm_g_noise, noise_map)
       END IF

       ! Calcuate the variance of the wavelet coeffisients at each scale
       sigma_s(s) = sigma_s(s) + SUM(map_TQU**2)
       IF(addnoise) THEN
          DO i=0, npix-1
             sigma_noise(s,i) = sigma_noise(s,i) + noise_map(i)**2
          END DO
       END IF
    END DO

  END SUBROUTINE find_sigma

  SUBROUTINE reduce_sigma

    IMPLICIT NONE
    REAL(DP) :: temp

    cnt = 1
    CALL MPI_AllReduce(sigma_CMB,sigma_CMB0,cnt,MPI_DOUBLE_PRECISION,MPI_SUM,MPI_COMM_WORLD,ierr)
    sigma_CMB = SQRT(sigma_CMB0/(N_max1*npix))

    IF(wavelet) THEN
       IF(smh) THEN
          cnt = nscales
       ELSE
          cnt = nj
       END IF
    ELSE
       cnt = 1
    END IF

    CALL MPI_AllReduce(sigma_s,sigma_s0,cnt,MPI_DOUBLE_PRECISION,MPI_SUM,MPI_COMM_WORLD,ierr)
    sigma_s = SQRT(sigma_s0/(N_max1*npix))

    IF(addnoise) THEN

       IF(wavelet) THEN
          IF(smh) THEN
             cnt = nscales*npix
          ELSE
             cnt = nj*npix
          END IF
       ELSE
          cnt = npix
       END IF

       CALL MPI_AllReduce(sigma_noise,sigma_noise0,cnt,MPI_DOUBLE_PRECISION,MPI_SUM,MPI_COMM_WORLD,
            ierr)

       DO s=start, finish
          sigma_noise(s,:) = SQRT((sigma_noise0(s,:)/N_max1) + (sigma_s0(s)/(N_max1*npix)))
       END DO
    END IF
```

```

END SUBROUTINE reduce_sigma

SUBROUTINE detect_ps(iseed, rng_handle, fwhm_arcmin, midpix)

  IMPLICIT NONE
  INTEGER(I4B)        :: iseed, pix, j, k, true_source, false_source, nlist, largest_pix, midpix,
       foundcount
  REAL(SP)            :: fwhm_arcmin
  REAL(DP)            :: costheta, radius, false_radius, phi, theta
  TYPE(PLANCK_RNG)    :: rng_handle
  LOGICAL(LGT)        :: found

  IF(me .EQ. 0) PRINT *, "Entering N=", i_N

  ! Reset the map. Add one point source to a clean map for finding mask size
  map_pixel = 0
  map_pixel(midpix) = 1

  ! Creates a simulated CMB map
  CALL create_alm(nside, lmax, lmax, polar, clfile, rng_handle, fwhm_arcmin, alm_TGC, header_PS)

  ! Transform the alm's to map
  CALL alm2map(nside, lmax, lmax, alm_TGC, map_TQU)

  ! Create random point sources
  DO i=1, no_of_sources

     ! Random direction and flux
     costheta = ran_mwc(iseed)*2 - 1
     phi = ran_mwc(iseed)*2*3.14159265
     ps_flux(i,i_N) = ran_mwc(iseed)*source_intensity*sigma_CMB

     ! Convert the angular coordinates to a pixel index
     CALL ang2pix_ring(nside, ACOS(costheta), phi, pix)
     ps_index(i,i_N) = pix

     ! Add the point sources to the map
     map_TQU(ps_index(i,i_N)) = map_TQU(ps_index(i,i_N)) + ps_flux(i,i_N)
  END DO

  ! Converts the map back to alm's
  CALL map2alm(nside,lmax,lmax,map_TQU,alm_TGC,zbounds,w8ring_TQU)
  CALL map2alm(nside,lmax,lmax,map_pixel,alm_test,zbounds,w8ring_TQU)

  ! Add beam effects to the created map (note that the pixel window
  ! function is included in this map)
  DO l=0, lmax
     alm_TGC(1,l,:) = alm_TGC(1,l,:)*beam(l,1)
     alm_test(1,l,:) = alm_test(1,l,:)*beam(l,1)
  END DO

  ! Point source detection must do the same processes to the map
  ! as for the calibration of sigma
  CALL alm2map(nside, lmax, lmax, alm_TGC, map_TQU)
  CALL alm2map(nside, lmax, lmax, alm_test, map_pixel)

  ! Adds noise effects to the created map
  IF(addnoise) THEN
     DO i=0,npix-1
        map_TQU(i) = map_TQU(i) + noise(i)*randgauss_boxmuller(iseed)*1000
     END DO
  END IF

  CALL map2alm(nside,lmax,lmax,map_TQU,alm_TGC,zbounds,w8ring_TQU)
  CALL map2alm(nside,lmax,lmax,map_pixel,alm_test,zbounds,w8ring_TQU)

  ! Loop over all the chosen scales
  DO s=start, finish
     IF(me .EQ. 0) PRINT *,"Processing scale ",s, " of ", finish

     IF(wavelet) THEN
        ! Calculate the alm's for the wavelets
        DO l=0, lmax
           alm_g(1,l,:) = alm_TGC(1,l,:)*gl(l,s)
           alm_g_test(1,l,:) = alm_test(1,l,:)*gl(l,s)
        END DO

        ! Transform the alm's to map for the wavelet coeffisients
        CALL alm2map(nside, lmax, lmax, alm_g, map_TQU)
        CALL alm2map(nside, lmax, lmax, alm_g_test, map_pixel)
     END IF

     true_source = 0
     false_source = 0
     mask = 1.

     ! Find total point source amplitude
     DO i=1, no_of_sources
        ps_amp(s) = ps_amp(s) + map_TQU(ps_index(i,i_N))
```

```
        END DO

        ! Find mask for the current wavelet
        CALL mask_finder(radius, false_radius, midpix)
        if(me .eq. 0) print *, "sigma-limit=",sigma_limit*sigma_s(s)

        ! Routine for locating point sources larger than X*sigma_CMB
        DO pix=0,npix-1

           IF(addnoise) THEN
              sigma_test = sigma_noise(s,pix)
           ELSE
              sigma_test = sigma_s(s)
           END IF

           ! Tests if the current temperature value is greater than a
           ! chosen factor of sigma_CMB. These pixels are possible
           ! point source candidates. The test will avoid cases where
           ! the map is 0, since these cases slow down the program
           IF((map_TQU(pix) .GE. sigma_limit*sigma_test) .AND. map_TQU(pix) .NE. 0) THEN

              found = .FALSE.
              foundcount = -1
              largest_pix = pix

              ! Checks if any of the surrounding pixels are larger
              CALL pix2vec_ring(nside, pix, vector)
              CALL query_disc(nside, vector, radius, listpix, nlist)
              DO j=0, nlist-1
                 IF(map_TQU(listpix(j)) .GT. map_TQU(largest_pix)) THEN
                    largest_pix = listpix(j)
                 END IF
              END DO

              ! Centers the search around the largest pixel
              CALL pix2vec_ring(nside, largest_pix, vector)
              CALL query_disc(nside, vector, radius, listpix, nlist)

              ! Either known point source locations or unknown
              IF(knowps) THEN
                 DO j=1,no_of_sources

                    ! Checks if the point source really is a point source or just
                    ! wrongly accused of being such a point. If true, then the
                    ! point source is counted as a true point source
                    IF(pix .EQ. ps_index(j,i_N)) THEN
                       found = .TRUE.
                       true_source = true_source + 1
                    END IF

                    ! All the surrounding pixels might be point sources. The test
                    ! fails if any of these point sources are smaller than the sigma
                    ! limit, that is if there are point sources within the mask that
                    ! are smaller than the point sources we are supposed to find
                    DO k=0, nlist-1

                       IF(addnoise) THEN
                          sigma_test = sigma_noise(s,listpix(k))
                       END IF

                       IF((listpix(k) .EQ. ps_index(j,i_N)) .AND. (map_TQU(listpix(k)) .GE.
                            sigma_limit*sigma_test)) THEN

                          foundcount = foundcount + 1
                          otherpix(foundcount) = listpix(k)
                          otherflux(foundcount) = map_TQU(listpix(k))
                          found = .TRUE.
                       END IF
                    END DO
                 END DO

                 ! A mask is put around the largest pixel. If other true
                 ! point sources were found in the above test, they are not
                 ! masked, so that they can be found later on. If no true
                 ! point sources were found, the point originates from CMB,
                 ! and the case is registered as a false point source and is
                 ! masked with a smaller mask
                 IF(found) THEN
                    CALL pix2vec_ring(nside, largest_pix, vector)
                    CALL query_disc(nside, vector, radius, listpix, nlist)
                    mask(listpix(0:nlist-1)) = 0.
                    map_TQU(listpix(0:nlist-1)) = 0.
                    map_TQU(otherpix(0:foundcount)) = otherflux(0:foundcount)

                 ELSE
                    false_source = false_source + 1

                    CALL pix2vec_ring(nside, largest_pix, vector)
                    CALL query_disc(nside, vector, false_radius, listpix, nlist)
                    mask(listpix(0:nlist-1)) = 0.
```

```
                        map_TQU(listpix(0:nlist-1)) = 0.
                    END IF

                ELSE
                    CALL pix2vec_ring(nside, largest_pix, vector)
                    CALL query_disc(nside, vector, radius, listpix, nlist)

                    ! Checks if the point source is real. Note that
                    ! false_source is used to count correct detections here
                    DO j=1,no_of_sources
                        DO k=0, nlist-1

                            IF(addnoise) THEN
                                sigma_test = sigma_noise(s,listpix(k))
                            END IF

                            IF((listpix(k) .EQ. ps_index(j,i_N)) .AND. (.NOT. found) .AND. (map_TQU(
                                listpix(k)) .GE. sigma_limit*sigma_test)) THEN
                                false_source = false_source + 1
                                found = .TRUE.
                            END IF
                        END DO
                    END DO

                    ! Unknown point sources are all treated as true point
                    ! sources, and masked with a standard beam mask
                    mask(listpix(0:nlist-1)) = 0.
                    map_TQU(listpix(0:nlist-1)) = 0.
                    true_source = true_source + 1
                END IF
            END IF
        END DO

        ! Calculation of the means
        false_mean(s) = false_mean(s) + false_source
        true_mean(s) = true_mean(s) + true_source
        mask_mean(s) = mask_mean(s) + 100*(1_dp-(SUM(mask)/npix))

        IF(N .EQ. 1) THEN
            ! Counts the number of pixels remaining in map with mask
            PRINT *, 'Number of pixels in mask is ', npix-INT(SUM(mask))
            PRINT *, 'Percentage in mask', 100*(1_dp-(SUM(mask)/npix))
            PRINT *

            ! Prints the results to screen
            IF(wavelet) THEN
                IF(smh) THEN
                    PRINT *, 'CURRENT SCALE IS s= ', scales(s)
                ELSE
                    PRINT *, 'CURRENT SCALE IS j= ', s
                END IF
            END IF
            PRINT *, true_source, ' of ', no_of_sources, ' point sources were found'
            IF(knowps) THEN
                PRINT *, false_source, ' found point sources were false'
            ELSE
                PRINT *, true_source-false_source, ' of these are incorrect detections'
            END IF
            PRINT *
        END IF
    END DO

END SUBROUTINE detect_ps

SUBROUTINE mask_finder(radius, false_radius, midpix)

    IMPLICIT NONE
    INTEGER(I4B) :: j, nlist, midpix
    REAL(DP)     :: radius, false_radius, map_sum, area

    ! Either fixed or dynamic mask
    IF(disc_size .EQ. 0) THEN

        ! Creates a mask of a single wavelet transformed pixel. If a pixel
        ! is larger than 0.1 the pixel size, it will be within the mask
        pixel_mask = 0
        DO i=0,npix-1
            IF(ABS(map_pixel(i)) .GT. 0.1*map_pixel(midpix)) pixel_mask(i) = 1
        END DO

        ! Finds the area of the mask, and calculates its radius
        map_sum = SUM(pixel_mask)
        area = pixsize**2*map_sum
        radius = SQRT(area/pi)

        ! Compensates for the pixels less than 0.1 the pixel size that
        ! was within the radius, and increases the radius by 10 % in
        ! case some false sources were missed. The radius then becomes
        ! a bit larger. Mask sizes larger than the sphere must be avoided,
        ! and these are set to pi.
```

```
         IF(radius .LT. pi) THEN

            CALL pix2vec_ring(nside, midpix, vector)
            CALL query_disc(nside, vector, radius, listpix, nlist)

            DO j=0, nlist-1
               IF(pixel_mask(listpix(j)) .EQ. 0) map_sum = map_sum + 1
            END DO

            area = pixsize**2*map_sum
            radius = SQRT(area/pi)
            IF(.NOT. smh) radius = radius*1.1
            IF(radius .GT. pi) radius = pi

            IF(N .EQ. 1) PRINT *, 'Use mask size ', radius*(180_dp/pi)*60_dp
         ELSE
            radius = pi
         END IF
      ELSE
         radius = disc_size/((180_dp/pi)*60_dp)
         IF(N .EQ. 1) PRINT *, 'Use fixed mask size ', radius*(180_dp/pi)*60_dp
      END IF

      ! Determine mask for false sources from mask for true sources
      false_radius = radius*0.4
      IF(N .EQ. 1) PRINT *, 'False mask size ', false_radius*(180_dp/pi)*60_dp

   END SUBROUTINE mask_finder

   SUBROUTINE dump_results

      ! Takes the sum of false_mean and true_mean from all cpus, and puts it
      ! into cpu #0
      IF(wavelet) THEN
         IF(smh) THEN
            cnt = nscales
         ELSE
            cnt = nj
         END IF
      ELSE
         cnt = 1
      END IF

      dest = 0
      CALL MPI_Reduce(false_mean,false_mean0,cnt,MPI_REAL,MPI_SUM,dest,MPI_COMM_WORLD,ierr)
      CALL MPI_Reduce(true_mean,true_mean0,cnt,MPI_REAL,MPI_SUM,dest,MPI_COMM_WORLD,ierr)
      CALL MPI_Reduce(mask_mean,mask_mean0,cnt,MPI_REAL,MPI_SUM,dest,MPI_COMM_WORLD,ierr)
      CALL MPI_Reduce(ps_amp,ps_amp0,cnt,MPI_REAL,MPI_SUM,dest,MPI_COMM_WORLD,ierr)

      IF(me .EQ. 0) THEN
         false_mean0 = false_mean0/N_max2
         true_mean0 = true_mean0/N_max2
         mask_mean0 = mask_mean0/N_max2
         ps_amp0 = ps_amp0/(N_max2*no_of_sources)

         IF(wavelet) THEN
            IF(smh) THEN
               DO s=start, finish
                  PRINT *, 'MEAN VALUES FOR THE SCALE ', scales(s), ' ARC MINUTES'
                  PRINT *, true_mean0(s), ' of ', no_of_sources, ' point sources were found'
                  IF(knowps) THEN
                     PRINT *, false_mean0(s), ' found point sources were false'
                  ELSE
                     PRINT *, true_mean0(s)-false_mean0(s), ' of these are incorrect detections'
                  END IF
                  PRINT *, 'Percentage in mask ', mask_mean0(s)
                  PRINT *, 'Total PS amplitude ', ps_amp0(s)
                  PRINT *
               END DO
            ELSE
               DO s=start, finish
                  PRINT *, 'MEAN VALUES FOR THE SCALE j=', s
                  PRINT *, true_mean0(s), ' of ', no_of_sources, ' point sources were found'
                  IF(knowps) THEN
                     PRINT *, false_mean0(s), ' found point sources were false'
                  ELSE
                     PRINT *, true_mean0(s)-false_mean0(s), ' of these are incorrect detections'
                  END IF
                  PRINT *, 'Percentage in mask ', mask_mean0(s)
                  PRINT *, 'Total PS amplitude ', ps_amp0(s)
                  PRINT *
               END DO
            END IF
         ELSE
            DO s=start, finish
               PRINT *, 'MEAN VALUES'
               PRINT *, true_mean0(s), ' of ', no_of_sources, ' point sources were found'
               IF(knowps) THEN
                  PRINT *, false_mean0(s), ' found point sources were false'
               ELSE
```

```
                PRINT *, true_mean0(s)-false_mean0(s), ' of these are incorrect detections'
             END IF
             PRINT *, 'Percentage in mask ', mask_mean0(s)
             PRINT *, 'Total PS amplitude ', ps_amp0(s)
             PRINT *
          END DO
       END IF

       open(unit, file='detections.unf', form='unformatted',status='unknown')
       rewind(unit)
       write(unit) true_mean0, false_mean0
       close(unit)
    END IF

  END SUBROUTINE dump_results
END MODULE psw_sub
```

A.2 Detection of unresolved point sources

Listing A.3: psks_par.f90

```
PROGRAM psks
  USE psks_sub

  INTEGER(I4B)          :: n_pols, iseed
  REAL(DP)              :: radius
  CHARACTER(LEN=128)    :: healpixdir, filename
  TYPE(PLANCK_RNG)      :: rng_handle

  ! Necessary for parallelization
  CALL MPI_INIT(ierr)
  CALL MPI_COMM_SIZE(MPI_COMM_WORLD, ntasks, ierr)
  CALL MPI_COMM_RANK(MPI_COMM_WORLD, me, ierr)

  ! Find Healpix-directory
  CALL getEnvironment("HEALPIX",healpixdir)

  ! Set parameters
  filename = 'params_psks3.txt'
  CALL get_params(filename)

  ! Set standard values derived from the parameters
  radius = disc_size*3.1416/(180d0*60d0)
  npix=nside**2*12
  n_pols = 1 + 2*polar ! either 1 or 3
  iseed = start_seed+me

  ! Necessary for parallelization
  ALLOCATE(N_max_pp(0:ntasks-1))
  ALLOCATE(stat(0:MPI_STATUS_SIZE-1))

  ! Code to distribute number of N evenly to each CPU, and if there's a
  ! remainder from the division, the remaining N are added to the first CPUs
  N_max_pp=N_max1/ntasks
  IF(MOD(N_max1,ntasks).NE.0) THEN
     N_max_pp(0:MOD(N_max1,ntasks)-1)=N_max_pp(0:MOD(N_max1,ntasks)-1)+1
  END IF

  ! Allocate memory for arrays
  CALL alloc(n_pols,.TRUE.)

  ! Preperation of the array containing the scales
  IF(wavelet) THEN
     IF(smh) THEN
        scales(1) = scale_start
        DO i=2,nscales
           scales(i) = scales(i-1) + add
        END DO
     END IF
  ELSE
     scales(1) = 1
  END IF

  ! Preperation of the array containing the amplitudes
  IF(amp) THEN
     source_intensity(1) = midamp - (N_amp/2)*addamp
     DO i=2,N_amp
        source_intensity(i) = source_intensity(i-1) + addamp
     END DO
  ELSE
     source_intensity(1) = midamp
  END IF

  ! Set values after allocation
```

```fortran
zbounds=[-1,1]
w8ring_TQU=1
mask = 1.
sigma_s = 0d0
sigma_noise = 0d0

! The unit for file opening is different for each CPU, where me is
! the CPU number
unit=10+me

! Fetches the beam from file
OPEN(unit, file='MAP_blxwl_avgv_opt.unf', form='unformatted',status='old')
REWIND(unit)
READ(unit) beam
CLOSE(unit)

! Fetches the noise from file
OPEN(unit, file='MAP_noise_avgv.unf', form='unformatted',status='old')
REWIND(unit)
READ(unit) noise
CLOSE(unit)

IF(wavelet) THEN
   IF(me .eq. 0) PRINT *, "Generating wavelets..."
   IF(smh) THEN

      ! Finds g_l for the SMH wavelets at the defined scales
      CALL calc_gl_smh(nside, lmax, nscales, scales, gl, .FALSE., .TRUE., glfile, me)
   ELSE
      ! Finds g_l for the needlets at the defined scales
      CALL calc_f2(f2,nn)
      CALL calc_gl(f2,nn,j0,nj,lmax,gl,aa)

      OPEN(unit, file='gl_psks.unf', form='unformatted',status='unknown')
      REWIND(unit)
      WRITE(unit) gl
      CLOSE(unit)
   END IF
END IF

! Transfer iseed to rng_handle, from now on, use rng_handle in calls to
! routines using random generator
CALL rand_init(rng_handle, iseed) ! takes up to 4 seeds simultaneously

! Generates CMB maps to determine a value for sigma_CMB
IF(me .EQ. 0) PRINT *, "Calibrating confidence..."
DO, i_N=0, N_max_pp(me)-1

   ! Finds which N to give to this CPU
   IF (me .EQ. 0) THEN
      N=i_N ! The first cpu just gets the first N
   ELSE
      ! Sums up all the N given to the previous CPUs such that
      ! the index starts off at the correct N
      N=SUM(N_max_pp(0:me-1))+i_N
   END IF

   CALL find_data(iseed, rng_handle, fwhm_arcmin)
END DO

! Takes the sum of sigma_CMB and sigma etc. from all cpus, and puts the
! result in all CPUs
CALL reduce(.TRUE.)
CALL dealloc(.FALSE.)

! Necessary for parallelization
ALLOCATE(N_max_pp(0:ntasks-1))
ALLOCATE(stat(0:MPI_STATUS_SIZE-1))

! Code to redistribute number of N evenly to each CPU, and if there's a
! remainder from the division, the remaining N are added to the first CPUs
N_max_pp=N_max2/ntasks
IF(MOD(N_max2,ntasks).NE.0) THEN
   N_max_pp(0:MOD(N_max2,ntasks)-1)=N_max_pp(0:MOD(N_max2,ntasks)-1)+1
END IF

! Reallocate memory for some arrays
CALL alloc(n_pols,.FALSE.)

! Generates CMB maps to simulate detection of point sources
IF(me .EQ. 0) PRINT *, "Detecting point sources..."
DO, i_N=0, N_max_pp(me)-1

   IF (me .EQ. 0) THEN
      N=i_N
   ELSE
      N=SUM(N_max_pp(0:me-1))+i_N
   END IF

   CALL sim(iseed, rng_handle, fwhm_arcmin, radius)
```

```
  END DO

  ! Takes the sum of sigma_CMB and sigma etc. from all cpus, and puts the
  ! result in all CPUs
  CALL reduce(.FALSE.)

  ! Necessary for parallelization
  CALL MPI_FINALIZE(ierr)

  ! Deallocate the memory used for arrays
  CALL dealloc(.TRUE.)

END PROGRAM psks
```

Listing A.4: psks_sub_par.f90

```
MODULE psks_sub

  USE healpix_types
  USE alm_tools
  USE ran_tools
  USE pix_tools
  USE extension
  USE mod_domwav
  USE rngmod,     ONLY: rand_init, rand_gauss, planck_rng

  IMPLICIT NONE
  INCLUDE 'mpif.h' ! Necessary for parallelization

  INTEGER(I4B), DIMENSION(:), ALLOCATABLE        :: listpix
  INTEGER(I4B), DIMENSION(:,:), ALLOCATABLE      :: ps_index
  REAL(SP), DIMENSION(:), ALLOCATABLE            :: map_TQU, scales, mask, noise, source_intensity,
      start_map
  REAL(SP), DIMENSION(:,:), ALLOCATABLE          :: gl
  REAL(DP), DIMENSION(:), ALLOCATABLE            :: zbounds, vector, f2
  REAL(DP), DIMENSION(:,:), ALLOCATABLE          :: beam, w8ring_TQU, ps_flux
  REAL(DP), DIMENSION(:,:,:), ALLOCATABLE        :: sigma, sigma0, skew, skew0, kurt, kurt0, cl, cl0
  COMPLEX(SPC), DIMENSION(:,:,:), ALLOCATABLE :: alm_TGC, alm_g, alm_corr
  CHARACTER(LEN=80), DIMENSION(1:180)            :: header_PS
  CHARACTER(LEN=128) :: clfile, glfile

  INTEGER(I4B) :: nside, lmax, polar, no_of_sources, nscales, N, i_N, l, i, s, j0, nj, nn, start,
      finish, start_seed, N_amp
  INTEGER(I8B) :: npix, N_max1, N_max2
  REAL(SP)     :: fwhm_arcmin, add, scale_start, addamp, midamp
  REAL(DP)     :: sigma_CMB, sigma_CMB0, aa, disc_size
  LOGICAL(LGT) :: smh, wavelet, addnoise, amp, constamp

  ! Necessary for parallelization
  INTEGER(I4B), DIMENSION(:), ALLOCATABLE        :: N_max_pp, stat
  INTEGER(I4B) :: ierr, ntasks, unit, cnt, cnt_cl, dest, tag, src, me

CONTAINS
  SUBROUTINE get_params(filename)

    CHARACTER(LEN=128) :: line, name, value, filename, wlet, addn, ampamp, camp
    INTEGER(I4B)       :: rstat
    LOGICAL(LGT)       :: exist

    ! Checks if the file exists on disk. trim cuts the blank characters
    ! away from filename
    INQUIRE(file=filename, exist=exist)
    IF(.NOT. exist) THEN
       PRINT *, "Error: File ", TRIM(filename) ," not found."
       STOP
    END IF

    ! Reads the file line for line. Scan finds the index of the specified
    ! character in the string. If there is no '=' on the line being read,
    ! or there is a comment '#' on the line, the do loop skips to the next
    ! line with cycle. Name contains the variable name, and value the value
    ! of the variable. If the name corresponds to one of the cases, the value
    ! of that name is inserted into the correct variable
    OPEN(unit, file=filename, form='formatted', iostat=rstat)
    DO WHILE(rstat .EQ. 0)
       READ(unit, fmt='(A)',iostat=rstat) line
       i = SCAN(line, '=')
       IF ((i .EQ. 0) .OR.(line(1:1) .EQ. '#')) CYCLE
       name = TRIM(ADJUSTL(line(:i-1)))
       value = TRIM(ADJUSTL(line(i+1:)))

       SELECT CASE(TRIM(name))
       CASE('nside')
          READ(value,*) nside
       CASE('lmax')
          READ(value,*) lmax
       CASE('N_max1')
          READ(value,*) N_max1
```

```
        CASE('N_max2')
           READ(value,*) N_max2
        CASE('no_of_sources')
           READ(value,*) no_of_sources
        CASE('polar')
           READ(value,*) polar
        CASE('start_seed')
           READ(value,*) start_seed
        CASE('amp')
           READ(value,*) ampamp
           IF(ampamp .EQ. 'true') THEN
              amp = .TRUE.
           ELSE IF(ampamp .EQ. 'false') THEN
              amp = .FALSE.
           ELSE
              PRINT *, "Error: amp must be true or false."
              STOP
           END IF
        CASE('constamp')
           READ(value,*) camp
           IF(camp .EQ. 'true') THEN
              constamp = .TRUE.
           ELSE IF(camp .EQ. 'false') THEN
              constamp = .FALSE.
           ELSE
              PRINT *, "Error: constamp must be true or false."
              STOP
           END IF
        CASE('N_amp')
           READ(value,*) N_amp
        CASE('source_intensity')
           READ(value,*) midamp
        CASE('addamp')
           READ(value,*) addamp
        CASE('fwhm_arcmin')
           READ(value,*) fwhm_arcmin
        CASE('wavelet')
           READ(value,*) wlet
           IF(wlet .EQ. 'smh') THEN
              wavelet = .TRUE.
              smh = .TRUE.
           ELSE IF(wlet .EQ. 'needlets') THEN
              wavelet = .TRUE.
              smh = .FALSE.
           ELSE IF(wlet .EQ. 'no') THEN
              wavelet = .FALSE.
              smh = .TRUE.
           ELSE
              PRINT *, "Error: ", TRIM(wlet) ," is not a valid wavelet."
              STOP
           END IF
        CASE('nscales')
           READ(value,*) nscales
        CASE('add')
           READ(value,*) add
        CASE('scale_start')
           READ(value,*) scale_start
        CASE('j0')
           READ(value,*) j0
        CASE('nj')
           READ(value,*) nj
        CASE('aa')
           READ(value,*) aa
        CASE('nn')
           READ(value,*) nn
        CASE('disc_size')
           READ(value,*) disc_size
       CASE('addnoise')
           READ(value,*) addn
           IF(addn .EQ. 'true') THEN
              addnoise = .TRUE.
           ELSE IF(addn .EQ. 'false') THEN
              addnoise = .FALSE.
           ELSE
              PRINT *, "Error: addnoise must be true or false."
              STOP
           END IF
        CASE('clfile')
           READ(value,*) clfile
        CASE('glfile')
           READ(value,*) glfile
        END SELECT
     END DO
     CLOSE(unit)

  END SUBROUTINE get_params

  SUBROUTINE alloc(n_pols,firsttime)

     IMPLICIT NONE
```

```
    INTEGER(I4B) :: n_pols
    LOGICAL(LGT) :: firsttime

    ! Allocates only the first time
    IF(firsttime) THEN
       ALLOCATE(alm_TGC(1:n_pols,0:lmax,0:lmax))
       ALLOCATE(alm_g(1:n_pols,0:lmax,0:lmax))
       ALLOCATE(alm_corr(1:n_pols,0:lmax,0:lmax))
       ALLOCATE(map_TQU(0:npix-1))
       ALLOCATE(start_map(0:npix-1))
       ALLOCATE(noise(0:npix-1))
       ALLOCATE(beam(0:lmax,1:n_pols))
       ALLOCATE(zbounds(1:2))
       ALLOCATE(w8ring_TQU(1:2*lmax, 1))
       ALLOCATE(mask(0:npix-1))
       ALLOCATE(vector(1:3))
       ALLOCATE(listpix(0:npix-1))
       ALLOCATE(source_intensity(1:N_amp))

       IF(wavelet) THEN
          IF(smh) THEN
             start=1
             finish=nscales
             ALLOCATE(scales(start:finish))
          ELSE
             start=j0
             finish=j0+nj-1
             ALLOCATE(f2(0:nn-1))
          END IF
       ELSE
          start=1
          finish=1
          ALLOCATE(scales(start:finish))
       END IF

       IF(.NOT. amp) N_amp = 1

       ALLOCATE(gl(0:lmax,start:finish))
       ALLOCATE(sigma(0:N_max_pp(me)-1,start:finish,1:N_amp))
       ALLOCATE(skew(0:N_max_pp(me)-1,start:finish,1:N_amp))
       ALLOCATE(kurt(0:N_max_pp(me)-1,start:finish,1:N_amp))
       sigma = 0.; skew = 0.; kurt = 0.

       ! sigma0 etc. is an array that contain all the results. This array is
       ! only given to CPU #0
       IF(me .EQ. 0) THEN
          ALLOCATE(sigma0(0:N_max1-1,start:finish,1:N_amp))
          ALLOCATE(skew0(0:N_max1-1,start:finish,1:N_amp))
          ALLOCATE(kurt0(0:N_max1-1,start:finish,1:N_amp))
          sigma0 = 0.; skew0 = 0.; kurt0 = 0.
       END IF

    ELSE ! For reallocation of a couple of arrays
       ALLOCATE(ps_index(1:no_of_sources, 0:N_max_pp(me)-1))
       ALLOCATE(ps_flux(1:no_of_sources, 0:N_max_pp(me)-1))
       ALLOCATE(sigma(0:N_max_pp(me)-1,start:finish,1:N_amp))
       ALLOCATE(skew(0:N_max_pp(me)-1,start:finish,1:N_amp))
       ALLOCATE(kurt(0:N_max_pp(me)-1,start:finish,1:N_amp))
       ALLOCATE(cl(0:lmax,0:N_max_pp(me)-1,1:N_amp))
       sigma = 0.; skew = 0.; kurt = 0.; cl=0

       IF(me .EQ. 0) THEN
          ALLOCATE(sigma0(0:N_max2-1,start:finish,1:N_amp))
          ALLOCATE(skew0(0:N_max2-1,start:finish,1:N_amp))
          ALLOCATE(kurt0(0:N_max2-1,start:finish,1:N_amp))
          ALLOCATE(cl0(0:lmax,0:N_max2-1,1:N_amp))
          sigma0 = 0.; skew0 = 0.; kurt0 = 0.; cl0 = 0.
       END IF
    END IF

  END SUBROUTINE alloc

  SUBROUTINE dealloc(lasttime)

    IMPLICIT NONE
    LOGICAL(LGT) :: lasttime

    IF(lasttime) THEN
       DEALLOCATE(alm_TGC, alm_g, alm_corr, map_TQU, start_map, noise, beam, ps_index, ps_flux,
            sigma, skew, kurt, zbounds, w8ring_TQU, mask, vector, listpix, gl, cl

       IF (me .EQ. 0) DEALLOCATE(sigma0, skew0, kurt, cl0)

       IF(wavelet) THEN
          IF(smh) THEN
             DEALLOCATE(scales)
          ELSE
             DEALLOCATE(f2)
          END IF
       ELSE
```

```
          DEALLOCATE(scales)
        END IF

        ! Necessary for parallelization
        DEALLOCATE(N_max_pp, stat)
     ELSE
        DEALLOCATE(sigma, skew, kurt)

        IF (me .EQ. 0) THEN
           DEALLOCATE(sigma0, skew0, kurt0)
        END IF

        ! Necessary for parallelization
        DEALLOCATE(N_max_pp, stat)
     END IF

  END SUBROUTINE dealloc

  SUBROUTINE find_data(iseed, rng_handle, fwhm_arcmin)

     IMPLICIT NONE
     INTEGER(I4B)         :: iseed
     REAL(SP)             :: fwhm_arcmin
     TYPE(PLANCK_RNG)     :: rng_handle

     IF(me .eq. 0) PRINT *, "Entering N=", i_N+1

     ! Creates a simulated CMB map
     CALL create_alm(nside, lmax, lmax, polar, clfile, rng_handle, fwhm_arcmin, alm_TGC, header_PS)

     ! Transforms the alm's to map and back again at the point in the code
     ! where point sources are added in the second MC loop
     CALL alm2map(nside, lmax, lmax, alm_TGC, map_TQU)
     CALL map2alm(nside,lmax,lmax,map_TQU,alm_TGC,zbounds,w8ring_TQU)

     ! Add beam effects to the created map (note that the pixel window
     ! function is included in this map)
     DO l=0, lmax
        alm_TGC(1,l,:) = alm_TGC(1,l,:)*beam(l,1)
     END DO

     ! Need the map for calculation of sigma_CMB
     CALL alm2map(nside, lmax, lmax, alm_TGC, map_TQU)

     ! Calculates the variance of the CMB map
     sigma_CMB = sigma_CMB + SUM(map_TQU**2)

     ! Add noise effects to the created map
     IF(addnoise) THEN
        DO i=0,npix-1
           map_TQU(i) = map_TQU(i) + noise(i)*randgauss_boxmuller(iseed)*1000
        END DO
     END IF

     ! Need alm's for wavelet transformation
     CALL map2alm(nside,lmax,lmax,map_TQU,alm_TGC,zbounds,w8ring_TQU)

     ! Loop over all the chosen scales, which calculates the alm's for the
     ! wavelets and transform them to map
     DO s=start, finish
        IF(me .eq. 0) PRINT *,"Processing scale ",s, " of ", finish

        IF(wavelet) THEN
           DO l=0, lmax
              alm_g(1,l,:) = alm_TGC(1,l,:)*gl(l,s)
           END DO
           CALL alm2map(nside, lmax, lmax, alm_g, map_TQU)
        END IF

        ! Calculate the variance, skewness and kurtosis of the wavelet
        ! coeffisients at each scale
        sigma(i_N,s,1) = SUM(map_TQU**2)
        skew(i_N,s,1) = SUM(map_TQU**3)
        kurt(i_N,s,1) = SUM(map_TQU**4)
     END DO

  END SUBROUTINE find_data

  SUBROUTINE sim(iseed, rng_handle, fwhm_arcmin, radius)

     IMPLICIT NONE
     INTEGER(I4B)         :: iseed, ipix, nlist, Ai
     REAL(SP)             :: fwhm_arcmin
     REAL(DP)             :: costheta, phi, radius
     TYPE(PLANCK_RNG)     :: rng_handle

     IF(me .EQ. 0) PRINT *, "Entering N=", i_N+1

     ! Creates a simulated CMB map
     CALL create_alm(nside, lmax, lmax, polar, clfile, rng_handle, fwhm_arcmin, alm_TGC, header_PS)
```

```
   ! Transform the alm's to map
   CALL alm2map(nside, lmax, lmax, alm_TGC, start_map)

   ! Loop over all amplitudes
   DO Ai=1, N_amp

      IF(me .EQ. 0) PRINT *, "Testing␣amplitude␣", source_intensity(Ai)
      map_TQU = start_map

      ! Create random point sources
      DO i=1, no_of_sources

         ! Random direction and flux
         costheta = ran_mwc(iseed)*2 - 1
         phi = ran_mwc(iseed)*2*3.14159265
         IF(constamp) THEN ! For constant amplitude
            ps_flux(i,i_N) = source_intensity(Ai)*sigma_CMB
         ELSE
            ps_flux(i,i_N) = ran_mwc(iseed)*source_intensity(Ai)*sigma_CMB
         ENDIF

         ! Convert the angular coordinates to a pixel index
         CALL ang2pix_ring(nside, ACOS(costheta), phi, ipix)
         ps_index(i,i_N) = ipix

         ! Add the point sources to the map
         map_TQU(ps_index(i,i_N)) = map_TQU(ps_index(i,i_N)) + ps_flux(i,i_N)
      END DO

      ! Converts the map back to alm's
      CALL map2alm(nside,lmax,lmax,map_TQU,alm_TGC,zbounds,w8ring_TQU)

      ! Add beam effects to the created map (note that the pixel window
      ! function is included in this map)
      DO l=0, lmax
         alm_TGC(1,l,:) = alm_TGC(1,l,:)*beam(l,1)
      END DO

      ! Point source simulation must do the same processes to the map
      ! as for the calibration of the confidence limits
      CALL alm2map(nside, lmax, lmax, alm_TGC, map_TQU)

      ! Add noise effects to the created map
      IF(addnoise) THEN
         DO i=0,npix-1
            map_TQU(i) = map_TQU(i) + noise(i)*randgauss_boxmuller(iseed)*1000
         END DO
      END IF

      CALL map2alm(nside, lmax, lmax, map_TQU, alm_TGC, zbounds, w8ring_TQU)

      ! Correct with beam before creating C_l
      DO l=0, lmax
         alm_corr(1,l,:) = alm_TGC(1,l,:)/beam(l,1)
      END DO

      ! Calculates cl
      DO l=0, lmax
         cl(l,i_N,Ai) = (1d0/(2d0*l + 1d0))*(alm_corr(1,l,0)**2+2d0*SUM(alm_corr(1,l,1:l)*CONJG(
              alm_corr(1,l,1:l))))
      END DO

      ! Loop over all the chosen scales
      DO s=start, finish
         IF(me .eq. 0) PRINT *,"Processing␣scale␣",s, "␣of␣", finish

         IF(wavelet) THEN
            ! Calculate the alm's for the wavelets
            DO l=0, lmax
               alm_g(1,l,:) = alm_TGC(1,l,:)*gl(l,s)
            END DO

            ! Transform the alm's to map for the wavelet coeffisients
            CALL alm2map(nside, lmax, lmax, alm_g, map_TQU)
         END IF

         ! Calculate the variance, skewness and kurtosis of the wavelet
         ! coeffisients at each scale
         sigma(i_N,s,Ai) = sum(map_TQU**2)
         skew(i_N,s,Ai) = sum(map_TQU**3)
         kurt(i_N,s,Ai) = sum(map_TQU**4)
      END DO
   END DO

END SUBROUTINE sim

SUBROUTINE reduce(data)

   IMPLICIT NONE
```

```
  LOGICAL(LGT) :: data
  CHARACTER(LEN=128) :: sigma_file, skew_file, kurt_file

  IF(data) THEN
     cnt = 1
     CALL MPI_AllReduce(sigma_CMB,sigma_CMB0,cnt,MPI_DOUBLE_PRECISION,MPI_SUM,MPI_COMM_WORLD,ierr)
     sigma_CMB = SQRT(sigma_CMB0/(N_max1*npix))
  END IF

  ! All cpus, except cpu #0, must wait before program can continue. All
  ! the cpus must finish calculation of sigma etc. first
  dest = 0 ! The destination CPU of the data
  IF(me .NE. 0) THEN

     ! Total number of elements to be sent
     IF(wavelet) THEN
        IF(smh) THEN
           cnt = nscales*N_max_pp(me)*N_amp
        ELSE
           cnt = nj*N_max_pp(me)*N_amp
        END IF
     ELSE
        cnt = 1*N_max_pp(me)*N_amp
     END IF
     cnt_cl = (lmax+1)*N_max_pp(me)*N_amp

     tag = me
     PRINT*,'****Sending sigma from ',me
     CALL MPI_SSEND(sigma,cnt,MPI_DOUBLE_PRECISION,dest,tag,MPI_COMM_WORLD,ierr)
     PRINT*,'****Sent sigma from ',me

     tag = me+ntasks ! Tag must be different from the previous
     PRINT*,'****Sending skewness from ',me
     CALL MPI_SSEND(skew,cnt,MPI_DOUBLE_PRECISION,dest,tag,MPI_COMM_WORLD,ierr)
     PRINT*,'****Sent skewness from ',me

     tag = me+ntasks*2 ! Tag must be different from the previous
     PRINT*,'****Sending kurtosis from ',me
     CALL MPI_SSEND(kurt,cnt,MPI_DOUBLE_PRECISION,dest,tag,MPI_COMM_WORLD,ierr)
     PRINT*,'****Sent cl from ',me

     IF(.NOT. data) THEN
        tag = me+ntasks*3 ! Tag must be different from the previous
        PRINT*,'****Sending cl from ',me
        CALL MPI_SSEND(cl,cnt_cl,MPI_DOUBLE_PRECISION,dest,tag,MPI_COMM_WORLD,ierr)
        PRINT*,'****Sent cl from ',me
     END IF

  ELSE
     DO i=0, ntasks-1

        ! Total number of elements to be received
        IF(wavelet) THEN
           IF(smh) THEN
              cnt = nscales*N_max_pp(i)*N_amp
           ELSE
              cnt = nj*N_max_pp(i)*N_amp
           END IF
        ELSE
           cnt = 1*N_max_pp(me)*N_amp
        END IF
        cnt_cl = (lmax+1)*N_max_pp(me)*N_amp

        IF (i .EQ. 0) THEN
           sigma0(0:N_max_pp(0)-1,:,:) = sigma
           skew0(0:N_max_pp(0)-1,:,:) = skew
           kurt0(0:N_max_pp(0)-1,:,:) = kurt
           IF(.NOT. data) cl0(:,0:N_max_pp(0)-1,:) = cl
        ELSE

           src=i ! The source CPU of the data
           tag=i ! The tag must match the one it's receiving from
           PRINT*,'****Receiving sigma from ',i
           CALL MPI_RECV(sigma0(SUM(N_max_pp(0:i-1)):SUM(N_max_pp(0:i))-1,:,:),cnt,
                MPI_DOUBLE_PRECISION,src,tag,MPI_COMM_WORLD,stat,ierr)
           PRINT*,'****Received sigma from ',i

           tag=i+ntasks
           PRINT*,'****Receiving skewness from ',i
           CALL MPI_RECV(skew0(SUM(N_max_pp(0:i-1)):SUM(N_max_pp(0:i))-1,:,:),cnt,
                MPI_DOUBLE_PRECISION,src,tag,MPI_COMM_WORLD,stat,ierr)
           PRINT*,'****Received skewness from ',i

           tag=i+ntasks*2
           PRINT*,'****Receiving kurtosis from ',i
           CALL MPI_RECV(kurt0(SUM(N_max_pp(0:i-1)):SUM(N_max_pp(0:i))-1,:,:),cnt,
                MPI_DOUBLE_PRECISION,src,tag,MPI_COMM_WORLD,stat,ierr)
           PRINT*,'****Received kurtosis from ',i

           IF(.NOT. data) THEN
```

```
                  tag=i+ntasks*3
                  PRINT*,'****Receiving cl from ',i
                  CALL MPI_RECV(cl0(:,SUM(N_max_pp(0:i-1)):SUM(N_max_pp(0:i))-1,:),cnt_cl,
                       MPI_DOUBLE_PRECISION,src,tag,MPI_COMM_WORLD,stat,ierr)
                  PRINT*,'****Received cl from ',i
               END IF
            END IF
         END DO

         ! Takes the middle of std.dev, skewness and kurtosis of all the pixels
         sigma0 = SQRT(sigma0/npix)
         skew0 = skew0/(npix*sigma0**3)
         kurt0 = kurt0/(npix*sigma0**4)-3
      END IF

      ! Print results to screen and write to file
      IF(data) THEN
         sigma_file = 'dummy.unf'
         skew_file = 'skew_data.unf'
         kurt_file = 'kurt_data.unf'
      ELSE
         sigma_file = 'cl_sim.unf'
         skew_file = 'skew_sim.unf'
         kurt_file = 'kurt_sim.unf'
      END IF
      CALL dump_results(sigma_file, skew_file, kurt_file)

   END SUBROUTINE reduce

   SUBROUTINE dump_results(cl_file, skew_file, kurt_file)

      IMPLICIT NONE
      CHARACTER(LEN=128) :: cl_file, skew_file, kurt_file

      IF(me .EQ. 0) THEN
         OPEN(unit, file=cl_file, form='unformatted',status='unknown')
         REWIND(unit)
         WRITE(unit) cl0
         CLOSE(unit)

         OPEN(unit, file=skew_file, form='unformatted',status='unknown')
         REWIND(unit)
         WRITE(unit) skew0
         CLOSE(unit)

         OPEN(unit, file=kurt_file, form='unformatted',status='unknown')
         REWIND(unit)
         WRITE(unit) kurt0
         CLOSE(unit)
      END IF

   END SUBROUTINE dump_results
END MODULE psks_sub
```

Bibliography

[1] Jet Propulsion Laboratory: HEALPix. `http://healpix.jpl.nasa.gov/`, as of April 2008.

[2] Legacy Archive for Microwave Background Analysis. `http://lambda.gsfc.nasa.gov/`, as of April 2008.

[3] Planck Science Team Home. `http://www.rssd.esa.int/Planck/`, as of April 2008.

[4] N. Aghanim and O. Forni. Searching for the non-gaussian signature of the CMB secondary anisotropies. *Astronomy & Astrophysics*, 347:409–418, 1999.

[5] C. L. Bennett, R. S. Hill, G. Hinshaw, M. R. Nolta, N. Odegard, L. Page, D. N. Spergel, J. L. Weiland, E. L. Wright, M. Halpern, N. Jarosik, A. Kogut, M. Limon, S.S. Meyer, G. S. Tucker, and E. Wollack. First Year Wilkinson Microwave Anisotropy Probe (WMAP) Observations: Foreground Emission. *ApJ*, 148:97, 2003.

[6] L. Cayón, J. L. Sanz, E. Martínez-González, A.J Banday, F. Argüeso, J. E. Gallegos, K. M. Górski, and G. Hinshaw. Spherical Mexian Hat wavelet: an application to detect non-Gaussianity in the COBE-DMR maps. *MNRAS*, 326:1243, 2001.

[7] I. Daubechies. Orthonormal Bases of Compactly Supported wavelets. *Communications On Pure & Applied Mathematics*, 41:909–996, 1988.

[8] Scott Dodelson. *Modern Cosmology.* Academic Press, 2003.

[9] Øystein Elgarøy. *AST4220: Cosmology I.* Insitute of Theoretical Astrophysics, University of Oslo, 2006.

[10] D. J. Fixsen, E. S. Cheng, J. M. Gales, J. C. Mather, R. A. Shafer, and E. L. Wright. The Cosmic Microwave Background Spectrum from the Full COBE FIRAS Data Set. *ApJ*, 473:576, 1996.

[11] O. Forni and N. Aghanim. Searching for non-gaussianity: Statistical tests. *Astronomy & Astrophysics*, 1999.

[12] A. Grossman and J. Morlet. Decomposition of Hardy functions into square integrable wavelets of constant shape. *SIAM Journal of Mathematical Analysis*, 15:723–736, 1984.

[13] K. M. Górski, G. Hinshaw, A.J. Banday, C.L. Bennett, E.L. Wright, A. Kogut, G.F. Smoot, and P. Lubin. On determining the spectrum of primordial inhomogeneity from the COBE DMR sky maps: Results of two-year data analysis. *ApJ*, 430:85, 1994.

[14] K. M. Górski, E. Hivon, and B. D. Wandelt. Analysis issues for large CMB data sets. *to appear in Proceedings of the MPA/ESO Cosmology Conference on Evolution of Large-Scale Structure: from Recombination to Garching 2-7 August 1998; eds A.J. Banday, R. K. Sheth and L. Da Costa*, 1998.

[15] G. Hinshaw, M. R. Nolta, C. L. Bennet, R. Bean, O. Doré, M. R. Greason, M. Halpern, R. S. Hill, N. Jarosik, A. Kogut, E. Komatsu, M. Limon, N. Odegard, S. S. Meyer, L. Page, H. V. Peiris, D. N. Spergel, G. S. Tucker, L. Verde, J. L. Weiland, E. Wollack, and E. L. Wright. Three Year Wilkinson Microwave Anisotropy Probe (WMAP) Observations: Temperature Analysis. *ApJ*, 170:288, 2007.

[16] K. M. Huffenberger, H. K. Eriksen, and F. K. Hansen. Point-Source Power in 3 Year Wilkinson Microwave Anisotropy Probe Data. *ApJ*, 651:81–84, 2006.

[17] S. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:674–693, 1989.

[18] D. Marinucci, D. Pietrobon, A. Balbi, P. Baldi, P. Cabella, G. Kerkyacharian, P. Natoli, D. Picard, and N. Vittorio. Spherical Needlets for CMB Data Analysis. *MNRAS*, submitted, 2007.

[19] E. Martínez-González, J. E. Gallegos, F. Argüeso, L. Cayón, and J. L. Sanz. The performance of spherical wavelets to detect non-Gaussianity in the CMB sky. *MNRAS*, 336:22, 2002.

[20] David S. Moore and George P. McCabe. *Introduction to the Practice of Statistics*. W. H. Freeman and Company, 1989.

[21] Barbara Ryden. *Introduction to Cosmology*. Pearson Education, 2003.

[22] Murray R. Spiegel and Larry J. Stephens. *Statistics*. McGraw-Hill, 1999.

[23] S. A. Trushkin. Radio spectra of the WMAP catalog sources. *Bulletin of the Special Astrophysical Observatory*, 55:90, 2003.

[24] E. L. Wright, X. Chen, N. Odegard, C. L. Bennett, R. S. Hill, G. Hinshaw, N. Jarosik, E. Komatsu, M.R. Nolta, L. Page, D. N. Spergel, J. L. Weiland, E. Wollack, J. Dunkley, B. Gold, M. Halpern, A. Kogut, D. Larson, M. Limon, S. S. Meyer, and G. S. Tucker. The Wilkinson Microwave Anisotropy (WMAP) Source Catalog. *ApJ*, submitted, 2008.