Designing Software to Calculate the Power Spectrum and Bispectrum of the Thermal Sunyaev-Zeldovich Effect Caused by Filaments in the Universe

by

Jonathan Fawcett

A thesis submitted to the

Department of Physics, Engineering Physics, and Astronomy

in conformity with the requirements for

the degree of Bachelor of Applied Science

Queen's University

Kingston, Ontario, Canada

April 2023

Abstract

Predicting the expected signal of cosmological observations using simulations is a frequently used method to validate models of cosmological evolution, but no open-source software exists to do so. The design process of a modular, extensible, and verifiable open-source Python package capable of calculating the expected statistical properties of the thermal Sunyaev-Zeldovich effect for cosmological structures is described. The code is structured using object-oriented programming principles, with vectorized calculations and parallelization used to improve computation speed. Template classes are provided for future development, with concrete classes written that reproduce published results in unit tests. Comparisons with data on galaxy clusters from the Planck observatory show that the calculator performs well, although further investigation is needed to ensure the slight discrepancies are due to uncertainties in the cosmological models. Results for cosmological filaments show the signal is extremely weak, but the software can be used to determine an ideal angular scale to search for large-scale structures at future observatories.

Table of Contents

List of Tables

List of Figures

1 Introduction and Motivation

As new telescopes begin to collect data on the early stages of the universe, scientists will be able to compare models of the universe's evolution with observations. One major source of information is variations in the cosmic microwave background (CMB) caused by the thermal Sunyaev-Zeldovich effect, where low-energy photons are boosted by hot electrons present in cosmological structures [1]. This causes a shift in the CMB spectrum, as seen in Figure 1.

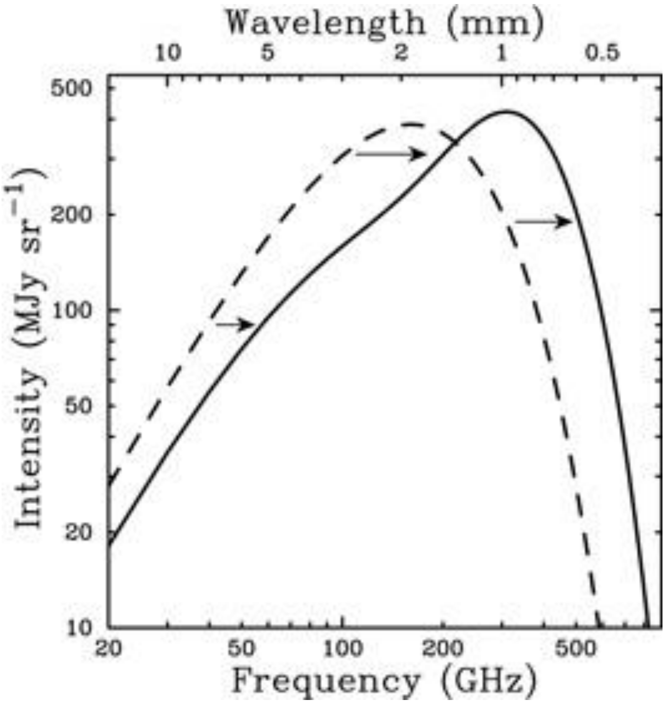

Figure 1: Illustration of the Sunyaev-Zeldovich effect, showing the shift from the undistorted spectrum (dashed line) to the distorted spectrum (solid line) [1].

By analyzing correlations of this shift between angular scales, scientists can probe the structure of the universe. Currently, no open-source software exists to compute these values for cosmological structures, although there has been research into the optimal method of computing them [2] [3].

Systemic issues in open-source scientific software include unintuitive design and insufficient documentation, as code is often written by researchers looking to publish results, rather than software engineers looking to develop tools for later use [4]. Most software written for publications is also closed-source and not freely available, due to issues such as the competitive nature of academia and restrictions from sponsoring institutions [5]. Along with insufficient testing, these factors have led to calculation errors and confusion when attempting to reconcile data from different publications [5].

The goal of this thesis is to design an open-source software tool for scientists to calculate the power spectrum and bispectrum of the thermal Sunyaev-Zeldovich effect caused by filaments in the universe, while incorporating software engineering design standards to improve usability and allow for future development.

2 Background

The large-scale structure of the universe consists of galaxies clustered into groups, with halos of hydrogen gas surrounding galaxies, and filaments of hot, diffuse gas connecting them [6]. The halos surrounding galaxies have been observed by telescopes, most notably by the Planck Space Observatory, however the filaments have extremely low density, making direct observation challenging [2] [6]. With new observatories, these filaments may soon be measured in greater detail, and therefore used to validate models of the universe's evolution.

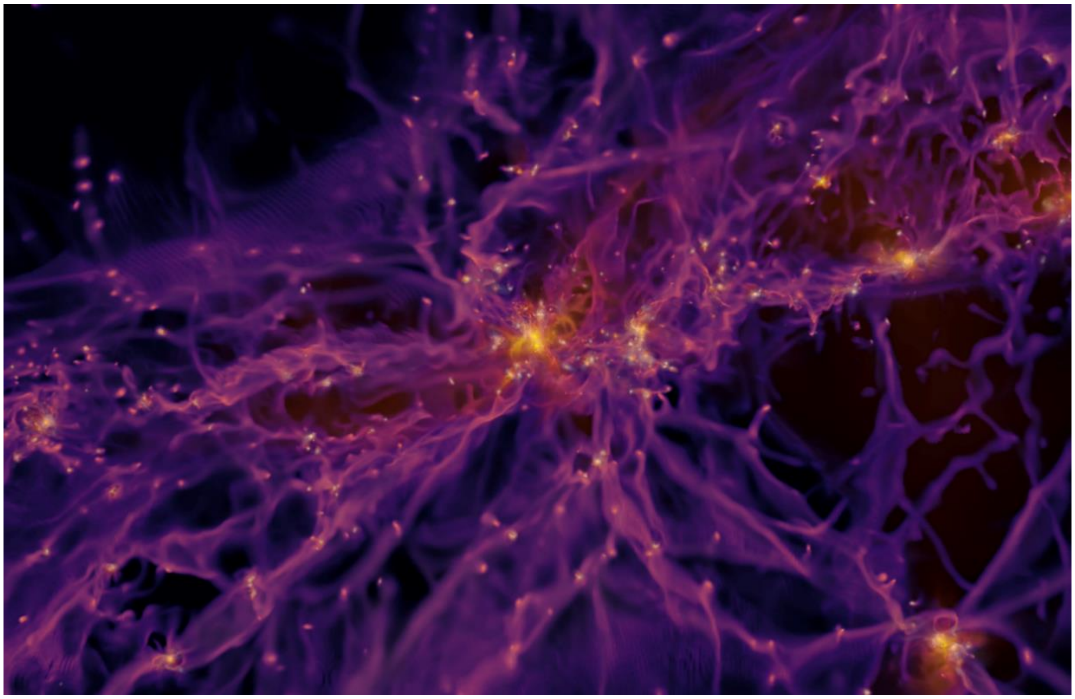

Figure 2: Results from a simulation of the universe conducted by the SPHINX project. Intergalactic gas is shown in purple, ionising radiation is yellow-red, and galaxies are white. The filamentary structure of the intergalactic medium can be clearly seen [7].

The magnitude of the effect of the thermal Sunyaev-Zeldovich effect on the CMB is measured by the Compton y-parameter, which represents the shift in the blackbody radiation spectrum caused by the scattering of photons off hot electrons [1]. This effect is significant due to the high temperature of the filament gas, in the range of 10^5 to 10^7 K [6]. The angular power spectrum for the intra-filament effects, which represents the expected signal for different angular scales across the sky, can be calculated by integrating the Fourier transform of the y-parameter along the line of sight [2]:

$$C_\ell^{1filament} = \int_{z_{min}}^{z_{max}} dz \frac{dV_c}{dzd\Omega} \int_{M_{min}}^{M_{max}} dM \frac{dN(M,z)}{dM} |\tilde{y}_\ell(M,z)|^2 , \tag{1}$$

where the variables are described in Table 1. The C_ℓ values calculated are the coefficients for a spherical harmonic expansion of the signal distribution on the sky, which are commonly reported when analyzing cosmological observations [2].

Table 1: Description of variables used to calculate the angular power spectrum [8].

Variable	Description
$C_\ell^{1filament}$	Angular power spectrum of the Compton y-parameter signal from intra-filament effects, for a given angular scale ℓ in spherical harmonics
z	Redshift of the filament, which corresponds to a spacetime distance
$\frac{dV_c}{dzd\Omega}$	Comoving volume of space per unit redshift per unit solid angle, which is formulated by modelling the expansion of the universe
M	Mass of the filament
$\frac{dN(M,z)}{dM}$	Number density of filaments per unit mass
$\tilde{y}_\ell(M,z)$	Fourier transform of the Compton y-parameter at a specific angular scale ℓ

The use of intra-filament effects implicitly stacks all filaments in the sky, which is a technique commonly used when analyzing telescope observations to discern the weak signal from the background [9]. The inter-filament effects are neglected in this project.

The bispectrum represents the correlation between angular scales, and is especially useful for analyzing filaments due to their non-Gaussian distribution [10]:

$$b(\ell_1\ell_2\ell_3) = \int_{z_{min}}^{z_{max}} dz \frac{dV_c}{dzd\Omega} \int_{M_{min}}^{M_{max}} dM \frac{dN(M,z)}{dM} \tilde{y}_{\ell_1}(M,z)\tilde{y}_{\ell_2}(M,z)\tilde{y}_{\ell_3}(M,z)\,, \qquad (2)$$

where b denotes the bispectrum, and ℓ_1, ℓ_2, and ℓ_3 form a triangle [10]. The relative magnitudes of $\ell_1, \ell_2,$ and ℓ_3 determine the shape of the triangle, where equilateral triangles with $\ell_1 = \ell_2 = \ \ell_3$ correspond to circular overdensities surrounded by underdense regions, and flattened triangles with $\ell_1 = \ell_2 = \frac{1}{2}\ell_3$ correspond to pancake-shaped overdensities with underdense planes between them [10]. Other arrangements are also possible, following the selection rules for angular momenta [10].

Both the angular power spectrum and bispectrum are sometimes multiplied by a constant factor of $f(x_\nu)$, which accounts for the variation in the y-parameter as a function of the measured frequency of light [11]. As seen in Figure 1, certain frequencies will show increased power, while others will show decreased power, or no change at all. Since the papers used to validate this software report values without the frequency factor, $f(x_\nu)$ will not be calculated [2] [9].

The Compton y-parameter in three dimensions is parameterized simply by the electron-pressure profile, assuming a cylindrical filament model [12]:

$$y_{3D}(x) = \frac{\sigma_T}{m_e c^2} P_e(x), \qquad (3)$$

where $x \equiv \frac{r}{R_{fil}(M,z)}$, for a given radial distance r from the centre of a cylindrical filament of characteristic radius R_{fil}. Note that the Thomson cross section, σ_T, the mass of the electron, m_e, and the speed of light in vacuum, c, are all constants. For the general calculator, the Fourier transform of this function is projected along the line of sight to yield $\tilde{y}$, which is then used in Equations 1 and 2 to calculate the statistical properties. This method can be combined with the radial pressure profile of galaxy clusters to calculate their expected Sunyaev-Zeldovich signal, which has been measured by various observatories and is therefore used for software validation [2].

For the expected signals from filaments, it is useful to parameterize the cylindrical filaments as having a uniform pressure along their length and a decreasing radial pressure that can be approximated as a double Gaussian [8]:

$$P_e(x) = A_1 e^{-\frac{x^2}{c_1^2}} + A_2 e^{-\frac{x^2}{c_2^2}}, \tag{4}$$

where A_1, A_2, c_1, and c_2 are all constants derived from a best fit to the predicted filament shape.

The slice-projection theorem states that a projection of the filament followed by a two-dimensional Fourier transform is equivalent to a three-dimensional Fourier transform followed by a one-dimensional slice of the filament. This leads to a much simpler calculation for the Fourier transform, $\tilde{y}_\ell$, as the Gaussians remain as Gaussians while the long axis of the cylinder becomes a $sinc$ function [8]:

$$\tilde{y}_\ell = \frac{1}{4\pi d_A^2} \frac{\sigma_T}{m_e c^2} \frac{2 \sin\left(\frac{k_z L}{2}\right)}{k_z} R_{fil}^2 \left(A_1 c_1^2 e^{-\frac{k_r^2 R_{fil}^2 c_1^2}{4}} + A_2 c_2^2 e^{-\frac{k_r^2 R_{fil}^2 c_2^2}{4}} \right), \tag{5}$$

where L is the characteristic length of the filaments and d_A is the angular diameter distance, defined by the model of the expansion of the universe. Since the filaments are aligned in random orientations with respect to the observer, the axes of the filament must first be aligned with the axes of the observer to evaluate the filament scales k_z and k_r, as shown in Figure 3.

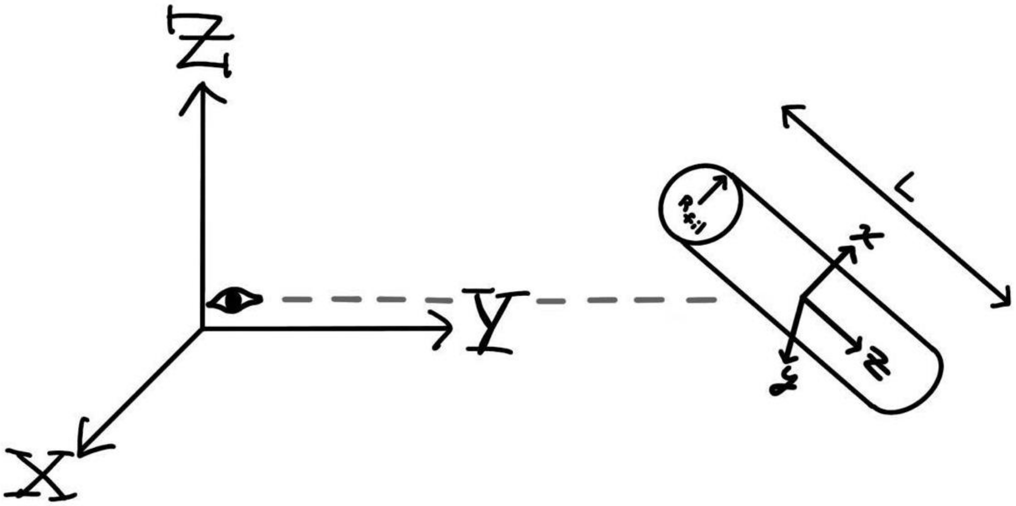

Figure 3: Diagram depicting the orientation of an arbitrary filament with axes x, y, z compared to the observer with axes X, Y, Z. The filament has a characteristic length L and a radius R_{fil}.

The rotation matrix can be derived using ZYZ Euler angles:

$$R(\phi_z, \theta_y) = R_{z,\phi_z} R_{y,\theta_y} \tag{6}$$

$$= \begin{bmatrix} cos\phi_z & -sin\phi_z & 0 \\ sin\phi_z & cos\phi_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\theta_y & 0 & sin\theta_y \\ 0 & 1 & 0 \\ -sin\theta_y & 0 & cos\theta_y \end{bmatrix} \tag{7}$$

$$= \begin{bmatrix} c_{\phi_z} c_{\theta_y} & -s_{\phi_z} & c_{\phi_z} s_{\theta_y} \\ s_{\phi_z} c_{\theta_y} & c_{\phi_z} & s_{\phi_z} s_{\theta_y} \\ -s_{\theta_y} & 0 & c_{\theta_y} \end{bmatrix}, \tag{8}$$

where $c_\theta = \cos(\theta)$, and $s_\theta = \sin(\theta)$. The final z Euler angle is not necessary since the orientation of the x and y axes of the filament are irrelevant for a cylinder. From the perspective of the observer, the size of the filament in each direction is simply the overall scale $k \equiv \frac{\ell}{d_A}$ multiplied by the projection of the axis onto the observer's frame, which are the column vectors of $R(\phi_z, \theta_y)$:

$$k_x = k \begin{bmatrix} c_{\phi_z} c_{\theta_y} \\ s_{\phi_z} c_{\theta_y} \\ -s_{\theta_y} \end{bmatrix}, \tag{9}$$

$$k_y = k\begin{bmatrix} -s_{\phi_z} \\ c_{\phi_z} \\ 0 \end{bmatrix}, \tag{10}$$

$$k_z = k\begin{bmatrix} c_{\phi_z} s_{\theta_y} \\ s_{\phi_z} s_{\theta_y} \\ c_{\theta_y} \end{bmatrix}. \tag{11}$$

By taking a slice at a constant $Y = 0$, the Y components go to zero and the magnitudes of the vectors are used as the relevant parameters:

$$k_x = k\sqrt{c_{\phi_z}^2 c_{\theta_y}^2 + s_{\theta_y}^2}, \tag{12}$$

$$k_y = k s_{\phi_z}, \tag{13}$$

$$k_z = k\sqrt{c_{\phi_z}^2 s_{\theta_y}^2 + c_{\theta_y}^2}. \tag{14}$$

Averaging over all orientations of the filaments requires integrating across the Euler angles:

$$\tilde{y}_\ell = \frac{1}{4\pi d_A^2}\frac{\sigma_T}{m_e c^2}\frac{1}{4\pi}\int_0^{2\pi}\int_0^{\pi}\frac{2\sin\left(\frac{k_z L}{2}\right)}{k_z} R_{fil}^2\left(A_1 c_1^2 e^{-\frac{k_r^2 R_{fil}^2 c_1^2}{4}} + A_2 c_2^2 e^{-\frac{k_r^2 R_{fil}^2 c_2^2}{4}}\right)\sin\theta_y \, d\theta_y d\phi_z, \tag{15}$$

where the radial scale is defined as:

$$k_r \equiv k_x^2 + k_y^2 = k^2\left(c_{\phi_z}^2 c_{\theta_y}^2 + s_{\theta_y}^2 + c_{\phi_z}^2\right). \tag{16}$$

This equation for $\tilde{y}_\ell$ can be used to calculate the angular power spectrum and bispectrum at arbitrary ℓ angular scales, per Equations 1 and 2. Since Equation 15 does not have an analytical solution, numerical integration is required to compute each term.

3 Constraints

The main stakeholders are scientists, who require software to be understandable, verifiably accurate, versatile, and easily integrated into their work [5]. Documentation is therefore very important, as are unit tests for both individual functions and the entire program that reproduce published results. Software engineers may further develop the program, and will desire code that is structured according to industry norms with thorough documentation.

Numerous technical challenges and risks were identified and mitigated to ensure the success of this project. The precision of data types was a concern, as it can cause significant rounding errors if not accounted for, while increased memory usage for higher precision data could affect computation time. Each variable was assigned an appropriate type during the initial design phase, and the use of floating-point variables was minimized to reduce memory usage. There was also risk in the time taken to perform the required nested numerical integrations, which was mitigated using vectorization and parallel computing methods to improve performance.

Because no publications have studied the power spectrum and bispectrum of filaments, and there is no open-source software to compare these calculations against, there was significant risk associated with the final accuracy of the software [8]. Mitigation was accomplished by creating unit tests to validate the behaviour of each class, and recreating a variety of published results to verify the results for filaments and galaxy clusters.

Environmental effects of this project are generally minimal, however large calculations can use a significant amount of electricity. Depending on the region where the computers are located, this could cause greenhouse gas emissions, so there is incentive to reduce computation time for environmental reasons.

Economic constraints are mainly due to the cost of computation time, as computer clusters commonly used for physics can cost $124 per core-year, with some projects costing up to $346,000 [13]. This means a modest 10% improvement in efficiency could save scientists tens of thousands of dollars. The time required for scientists to understand and integrate the code could also be significant if it is not properly documented and tested. As such, making the software efficient and easy to use will reduce its cost, while its open-source nature removes any requirement for licensing fees.

Releasing the software under an open-source license allows all scientists to freely access the software and reproduce published work, which is an ethical choice that prioritizes the accessibility of science. As many published works are first uploaded onto pre-print servers to offer open access to cosmological research, allowing open access to the software is a natural extension of this trend. Multiple open-source licenses are available, although the two most popular are the General Public License (GPL) and the Berkeley Software Distribution (BSD) license [14]. The BSD license is quite simple, and allows users to use, modify, and redistribute the code in any way, albeit with no warranty from the developers [14]. By contrast, the GPL license is significantly longer, forbids selling the software, and requires all derivative works to carry the same license [14]. This software is released under the BSD license due to its simplicity and standard usage in scientific programs [14], although the potential privatization of software used to benefit cosmological exploration is a downside to this approach.

There is no reasonable chance that this cosmological software could be used in an unsafe manner, so there are no major safety concerns.

4 Methodology

The programming languages considered for this project were Python, C++, and Fortran. Fortran is a lower-level compiled language that has historically been used in scientific software, but is currently being phased out in favour of modern languages due to their ease of use [5] [8]. C++ is also a compiled language that focuses on object-oriented programming, and is an industry standard for high-performance applications, but it can be challenging to learn and has little support for open-source packages [5]. Python supports object-oriented programming, has a more human-readable syntax and many community-made packages, and is increasingly used in open-source scientific software [5]. Because the modularity, usability, and future development of this software were the most important considerations for this design, Python was chosen as the programming language [8].

Multiple software design strategies were considered, including Jupyter notebooks, functional programming, and well-structured large scripts. Object-oriented programming was chosen due to it being the industry standard for modern programming, as well as its superior modularity, ease of integration, and potential for future development [15]. Existing scientific software suites, such as HEALPix and nuFATE, are programmed in Python and follow object-oriented programming [16] [17].

Each term of Equations 1, 2, 3, and 5, such as the number density of filaments, electron pressure distribution, filament shape and radial distribution, are divided into separate classes to facilitate testing and further development. For example, adding a new electron pressure distribution is facilitated by the existence of an abstract template class, with unit tests that ensure the output is reasonable and follows the data type and format required by the software. For each template class, a concrete class that implements a published result is included to provide verification of results.

Additionally, the software uses cosmology models from the Astropy project [18], which simplifies integration into existing code and allows future cosmological models to be quickly imported. All inputs and outputs from the software use Astropy's Quantity object, which tracks and automatically converts the units associated with each variable. This allows inputs to be provided in any unit, and provides an unambiguous, easily verifiable output format that can then be used with any of Astropy's other cosmological software.

The calculation functions are written to work vectorially with NumPy arrays, which significantly improves performance at the cost of increased memory usage. The precision of the calculations is also user-selectable by varying the number and distribution of evaluation points, and changing the integration method to a fixed or variable-step algorithm.

Although this project focuses on the angular power spectrum and bispectrum, the software can also be used for higher-order statistics by increasing the number of ℓ values in each row of the input matrix. Throughout the code, few assumptions are made about the magnitude or type of data supplied, which ensures the software is usable for a variety of applications.

A Unified Modeling Language (UML) diagram was created for the software to visually illustrate the design, which can be seen in Figure 4.

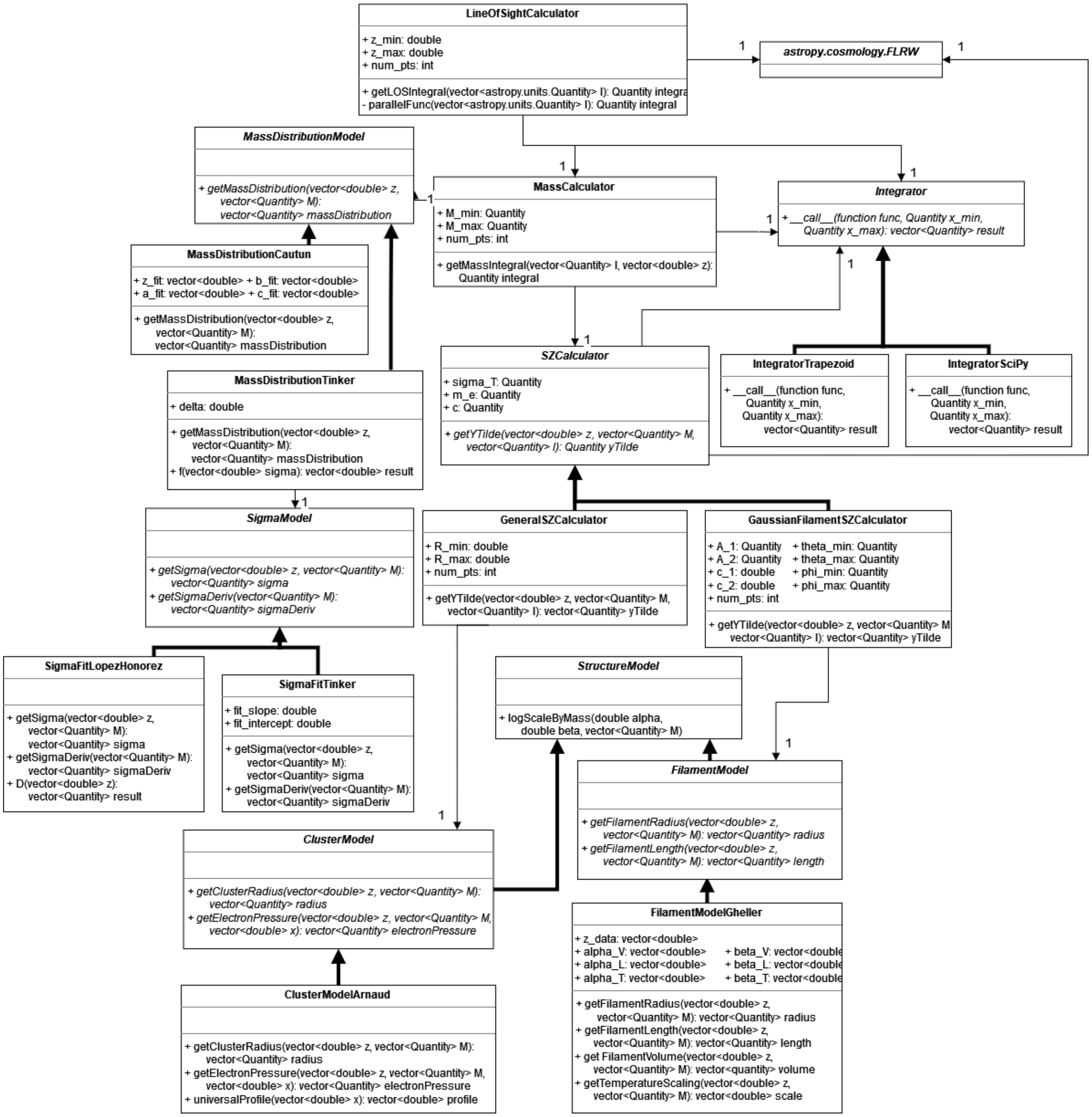
LineOfSightCalculator
+ z_min: double
+ z_max: double
+ num_pts: int
+ getLOSIntegral(vector<astropy.units.Quantity> I): Quantity integra
- parallelFunc(vector<astropy.units.Quantity> I): Quantity integral
1
astropy.cosmology.FLRW
1
MassDistributionModel
+ getMassDistribution(vector<double> z, vector<Quantity> M): vector<Quantity> massDistribution
1
MassCalculator
+ M_min: Quantity
+ M_max: Quantity
+ num_pts: int
+ getMassIntegral(vector<Quantity> I, vector<double> z): Quantity integral
1
Integrator
1
+ __call__(function func, Quantity x_min, Quantity x_max): vector<Quantity> result
1
1
MassDistributionCautun
+ z_fit: vector<double> + b_fit: vector<double>
+ a_fit: vector<double> + c_fit: vector<double>
+ getMassDistribution(vector<double> z, vector<Quantity> M): vector<Quantity> massDistribution
1
SZCalculator
+ sigma_T: Quantity
+ m_e: Quantity
+ c: Quantity
+ getYTilde(vector<double> z, vector<Quantity> M, vector<Quantity> I): Quantity yTilde
IntegratorTrapezoid
+ __call__(function func, Quantity x_min, Quantity x_max): vector<Quantity> result
IntegratorSciPy
+ __call__(function func, Quantity x_min, Quantity x_max): vector<Quantity> result
MassDistributionTinker
+ delta: double
+ getMassDistribution(vector<double> z, vector<Quantity> M): vector<Quantity> massDistribution
+ f(vector<double> sigma): vector<double> result
1
SigmaModel
+ getSigma(vector<double> z, vector<Quantity> M): vector<Quantity> sigma
+ getSigmaDeriv(vector<Quantity> M): vector<Quantity> sigmaDeriv
GeneralSZCalculator
+ R_min: double
+ R_max: double
+ num_pts: int
+ getYTilde(vector<double> z, vector<Quantity> M, vector<Quantity> I): vector<Quantity> yTilde
GaussianFilamentSZCalculator
+ A_1: Quantity + theta_min: Quantity
+ A_2: Quantity + theta_max: Quantity
+ c_1: double + phi_min: Quantity
+ c_2: double + phi_max: Quantity
+ num_pts: int
+ getYTilde(vector<double> z, vector<Quantity> M vector<Quantity> I): vector<Quantity> yTilde
SigmaFitLopezHonorez
+ getSigma(vector<double> z, vector<Quantity> M): vector<Quantity> sigma
+ getSigmaDeriv(vector<Quantity> M): vector<Quantity> sigmaDeriv
+ D(vector<double> z): vector<Quantity> result
SigmaFitTinker
+ fit_slope: double
+ fit_intercept: double
+ getSigma(vector<double> z, vector<Quantity> M): vector<Quantity> sigma
+ getSigmaDeriv(vector<Quantity> M): vector<Quantity> sigmaDeriv
StructureModel
+ logScaleByMass(double alpha, double beta, vector<Quantity> M)
1
FilamentModel
+ getFilamentRadius(vector<double> z, vector<Quantity> M): vector<Quantity> radius
+ getFilamentLength(vector<double> z, vector<Quantity> M): vector<Quantity> length
1
ClusterModel
+ getClusterRadius(vector<double> z, vector<Quantity> M): vector<Quantity> radius
+ getElectronPressure(vector<double> z, vector<Quantity> M, vector<double> x): vector<Quantity> electronPressure
FilamentModelGheller
+ z_data: vector<double>
+ alpha_V: vector<double> + beta_V: vector<double
+ alpha_L: vector<double> + beta_L: vector<double
+ alpha_T: vector<double> + beta_T: vector<double
+ getFilamentRadius(vector<double> z, vector<Quantity> M): vector<Quantity> radius
+ getFilamentLength(vector<double> z, vector<Quantity> M): vector<Quantity> length
+ get FilamentVolume(vector<double> z, vector<Quantity> M): vector<quantity> volume
+ getTemperatureScaling(vector<double> z, vector<Quantity> M): vector<double> scale
ClusterModelArnaud
+ getClusterRadius(vector<double> z, vector<Quantity> M): vector<Quantity> radius
+ getElectronPressure(vector<double> z, vector<Quantity> M, vector<double> x): vector<Quantity> electronPressure
+ universalProfile(vector<double> x): vector<double> profile

Figure 4: UML class diagram of the software. Bold lines represent inheritance, while thin lines represent possession. Italics denote abstract classes and methods.

5 Results and Validation

Unit tests are implemented for each class, reproducing results from the corresponding publication using data from digitized plots, and the overall calculator has unit tests to reproduce the observed power spectrum from the Planck mission. For published results with errors included, the test verifies that the calculated value is within standard error of the results.

For the Planck results, the calculated value must be within one order of magnitude of the published results due to the large uncertainties associated with each term of the calculation. Additional tests are included to demonstrate the speed improvements from just-in-time compilation and parallelization. A full list of tests can be found in Table 2.

Table 2: Unit tests implemented in the software, allowing users to easily verify accuracy and performance.

Test Description	**Classes Tested**	**Data Source(s)**
Calculate differential comoving volume across redshifts	Astropy cosmology model	[19]
Calculate length and radius of filaments across masses and redshifts	FilamentModelGheller	[6]
Calculate filament mass distribution across redshifts	MassDistributionCautun	[20]
Calculate cluster radius across masses and redshifts	ClusterModelArnaud	[21]
Calculate cluster mass distribution across redshifts	MassDistributionTinker, SigmaFitLopezHonorez	[22] [23]
Calculate angular power spectrum from clusters	LineOfSightCalculator (angular power spectrum), ClusterModel, GeneralSZCalculator	[2]
Calculate bispectrum from filaments that are fully parallel/fully perpendicular to observer	LineOfSightCalculator (bispectrum), GaussianFilamentSZCalculator	[10]
Numerical integration of analytical functions	Integrator	Self-derived
Test calculation speed with just-in-time compilation enabled/disabled	Performance of compiled functions	N/A
Test calculation speed with parallelization enabled/disabled	Performance of all classes when parallelized	N/A

The unit tests can optionally produce plots comparing the outputs and the published results, an example of which can be seen in Figure 5.

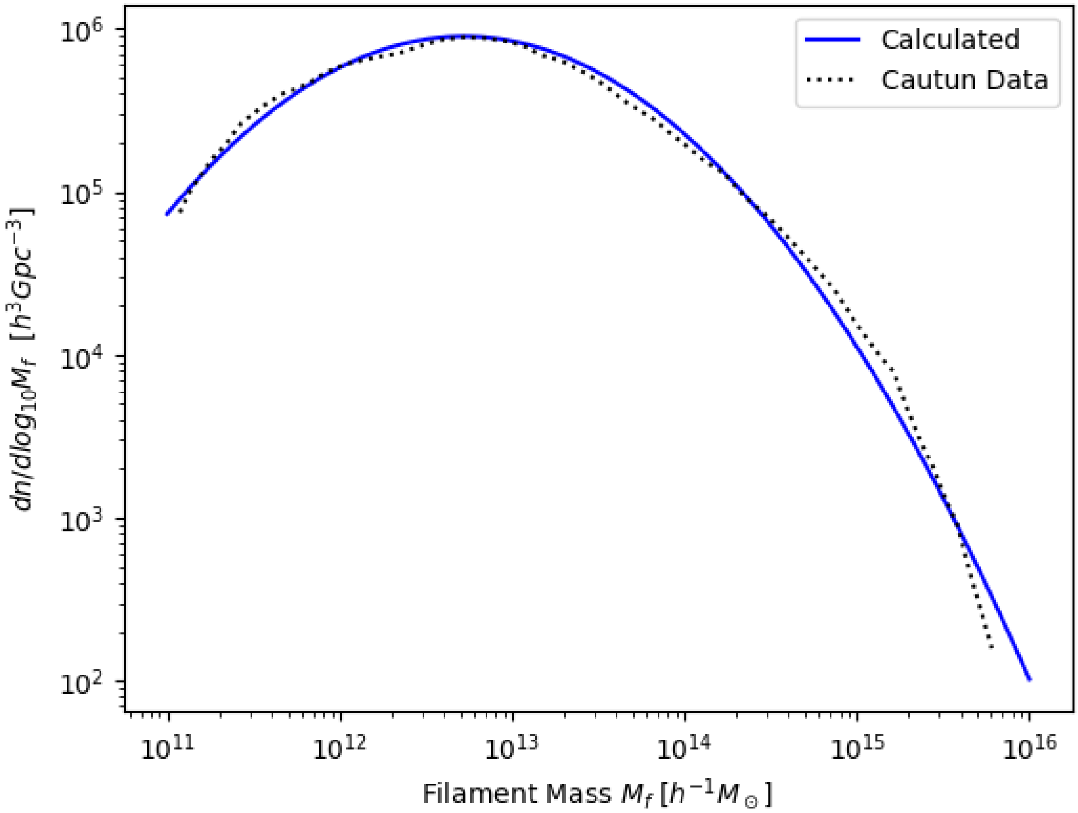

Figure 5: Comparison of published filament number density and the quantity calculated by the MassDistributionCautun class using a parabolic fit, at redshift $z = 0$ [20].

Additionally, the software was used to reproduce measurements from the Planck observatory, using cluster electron pressures and distributions from other published works. Only the intra-cluster effects are modelled, while the inter-cluster effects are neglected for this calculation. The results can be seen in Figure 6.

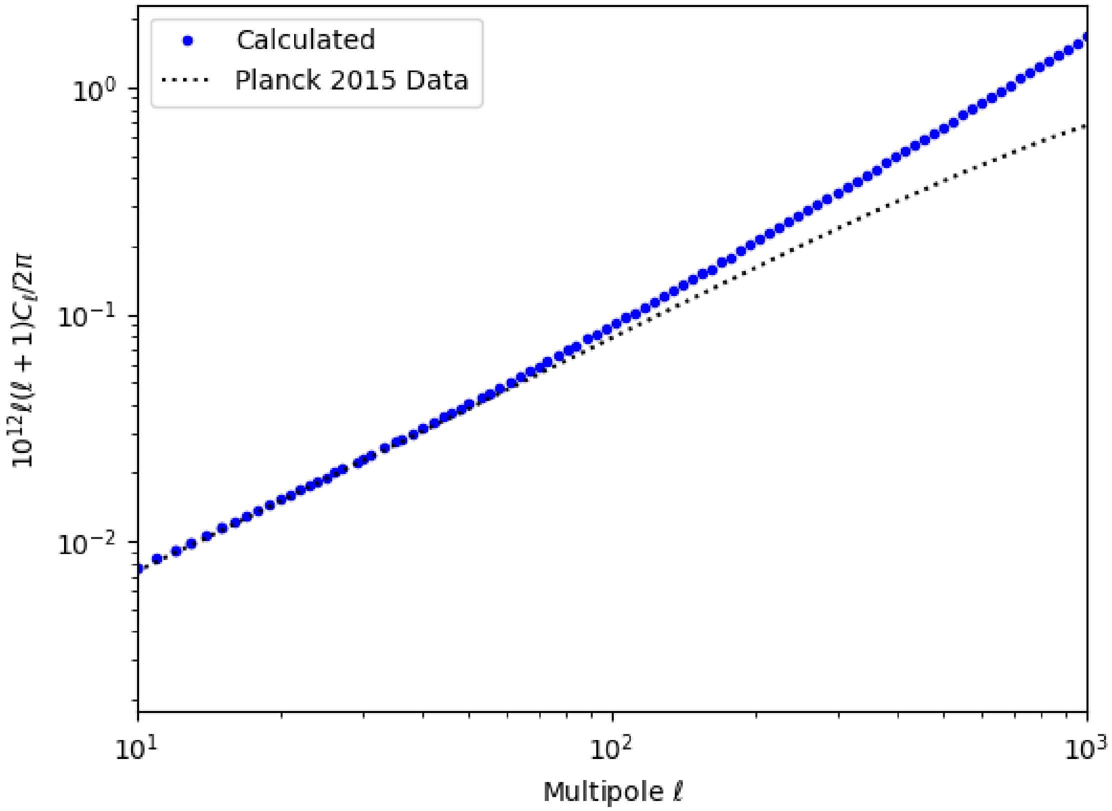

Figure 6: Angular power spectrum of galaxy clusters, calculated using Astropy's Planck 2018 cosmology model, SigmaFitLopezHonorez, MassDistributionTinker, ClusterModelArnaud, and the GeneralSZCalculator. The dotted line shows the Sunyaev-Zeldovich contribution to Planck observations of clusters [2].

The magnitude of the calculated angular power spectrum is less than an order of magnitude away from the published measurements at all angular scales, although it does overestimate the magnitude for $\ell > 100$. The equilateral bispectrum is within two orders of magnitude of the Planck observations, as seen in Figure 7. This is a larger error than was allowed in the quantitative objectives, but similar attempts to recreate the Planck observations have also had errors around two orders of magnitude [8]. Therefore, the failure of this test is more likely due to errors in the current models of galaxy clusters, rather than the calculator itself.

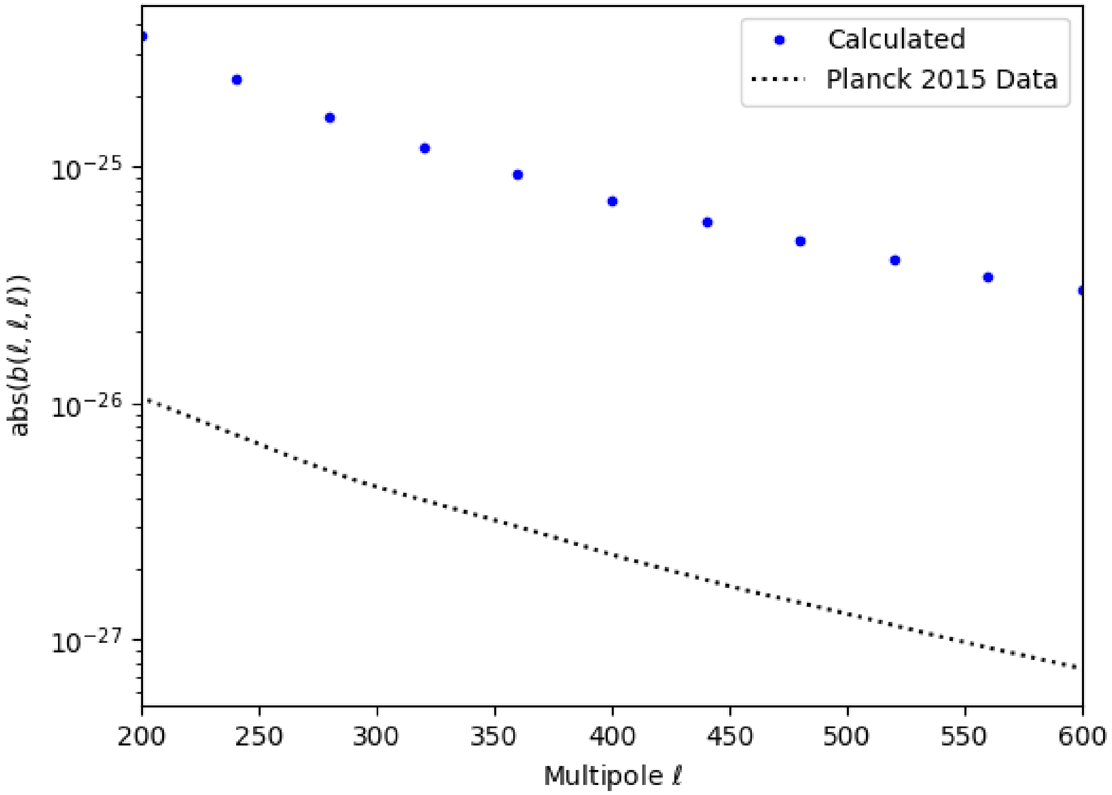

Figure 7: Equilateral bispectrum of clusters, calculated using Astropy's Planck 2018 cosmology model, SigmaFitLopezHonorez, MassDistributionTinker, ClusterModelArnaud, and the GeneralSZCalculator. The dotted line shows the Planck observations of real clusters [2].

There are no published values for filaments to compare against, however by testing the angular power spectrum and bispectrum at various angular arrangements of the filaments, the behaviour of the calculator can be partially verified. A large positive equilateral bispectrum corresponds to a universe with large circular overdensities surrounded by underdense regions [10]. By fixing the angular limits of θ_y and ϕ_z in Equation 15 to assume all filaments are fully parallel to the line of sight, or fully perpendicular, the relative magnitudes can be compared, as seen in Figure 8.

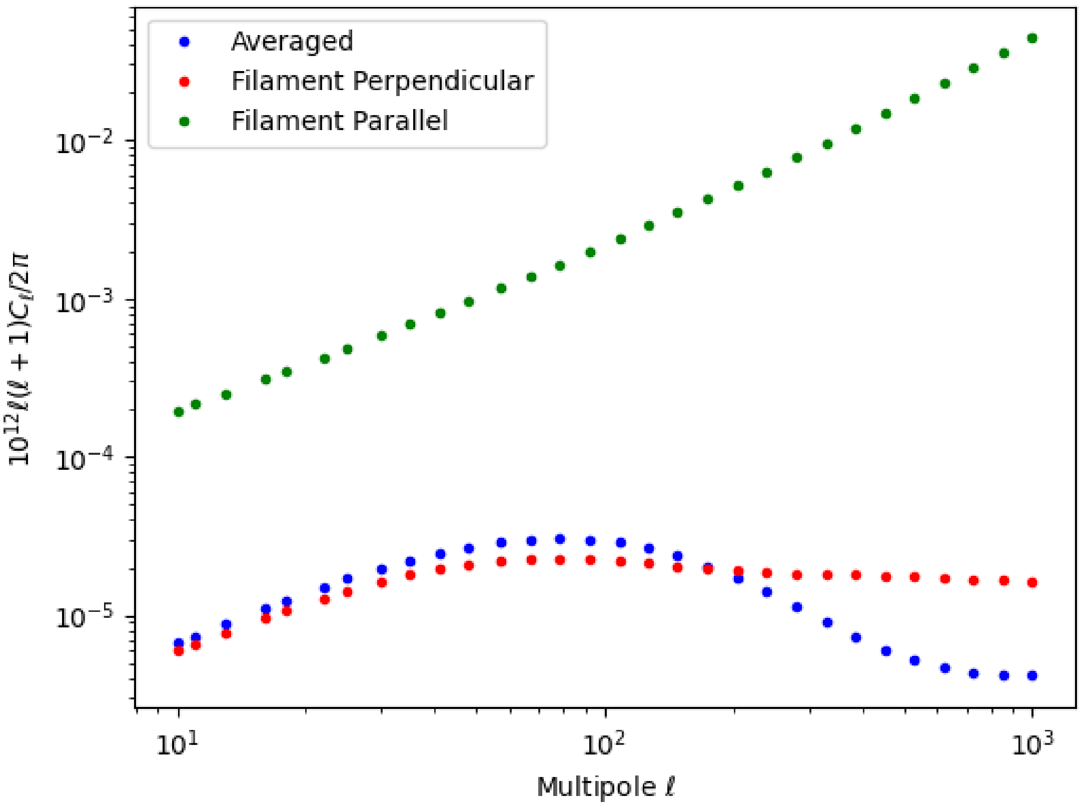

Figure 8: Angular power spectrum for filaments, calculated using Astropy's Planck 2018 cosmology model, MassDistributionCautun, FilamentModelGheller, and the GaussianFilamentSZCalculator. Filaments averaged over all orientations are blue, while filaments whose lengths are perpendicular and parallel to the line of sight are green and red, respectively.

Filaments that are parallel to the line of sight appear similar to clusters, so the angular power spectrum should be roughly linearly dependent on angular scale ℓ, as in Figure 6. The angular power spectrum of the averaged filaments has a maximum near $\ell = 100$, which suggests that this is an ideal angular scale to probe for filament signals. However, the signal is approximately three orders of magnitude weaker than that of the clusters, complicating this search. Because the filaments are strongly non-Gaussian, they should also have a noticeable bispectrum, which is shown in Figure 9.

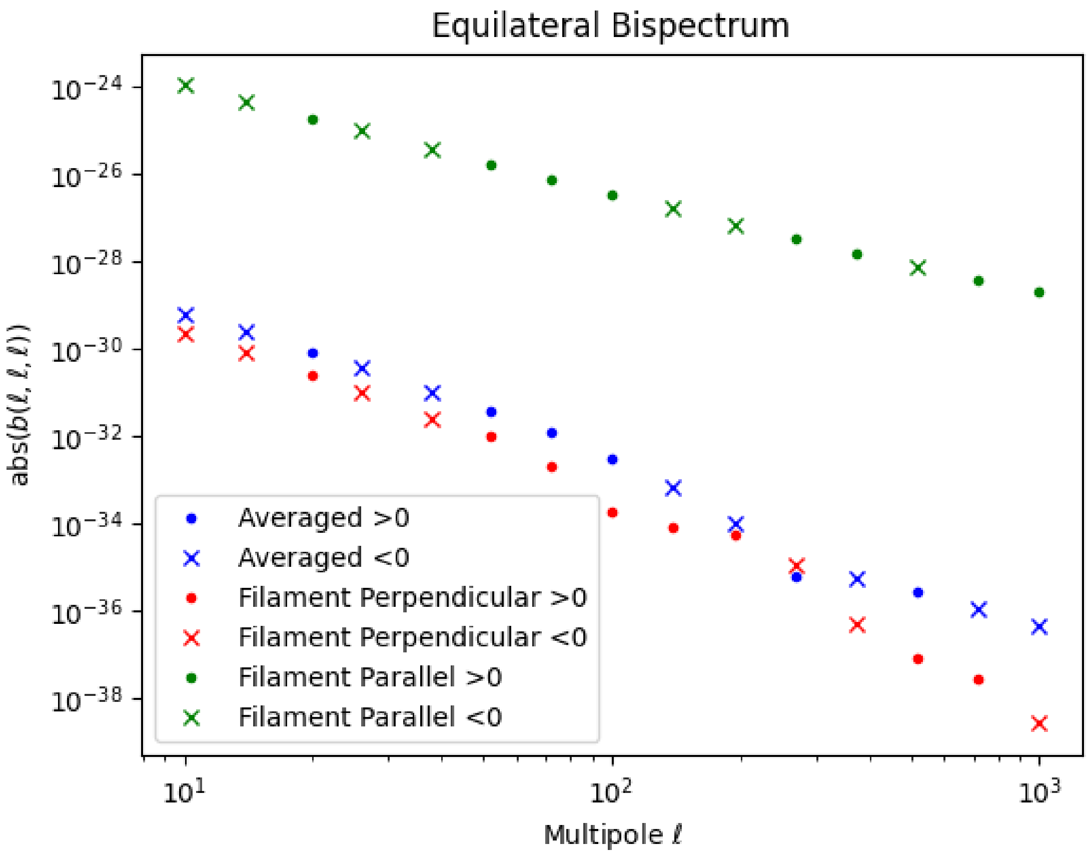

Figure 9: Equilateral bispectrum of filaments, calculated using Astropy's Planck 2018 cosmology model, MassDistributionCautun, FilamentModelGheller, and the GaussianFilamentSZCalculator. Filaments averaged over all orientations are blue, while filaments whose lengths are perpendicular and parallel to the line of sight are green and red, respectively. Circles denote positive values, while crosses denote negative values.

The equilateral bispectrum for the parallel filaments is larger than that of the averaged and perpendicular filaments, as expected by their appearance as circular overdensities. The flattened bispectrum shows a similar pattern in Figure 10, although the averaged filaments switch from positive to negative magnitudes near $\ell = 150$. This implies that the predominant distribution switches from dense pancakes with interstitial voids to diffuse regions with prominent voids at small angular scales [10].

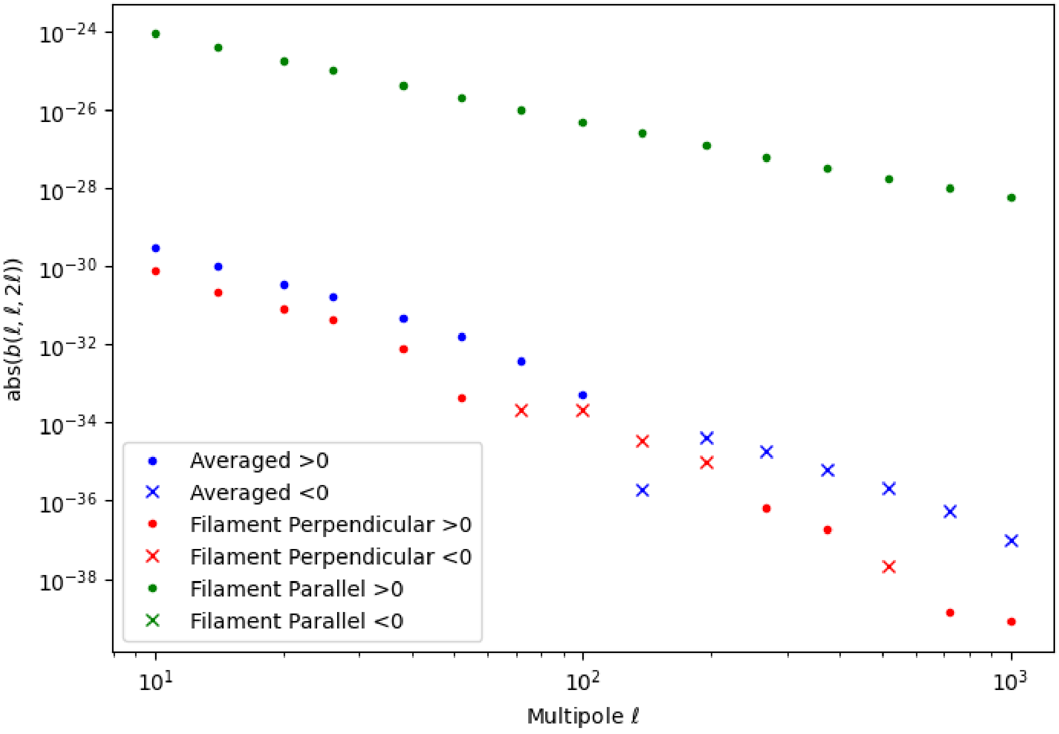

Figure 10: Flattened bispectrum of filaments, calculated using Astropy's Planck 2018 cosmology model, MassDistributionCautun, FilamentModelGheller, and the GaussianFilamentSZCalculator. Filaments averaged over all orientations are blue, while filaments whose lengths are perpendicular and parallel to the line of sight are green and red, respectively. Circles denote positive values, while crosses denote negative values.

The parallel filaments still show a stronger flattened bispectrum than the perpendicular or averaged filaments, which is an unintuitive result. Since the parallel filaments are similar to clusters, they should have a lower flattened bispectrum, but this effect is overwhelmed by the significantly stronger signal coming from the parallel filaments. To fully verify the filament calculation, high-confidence observations of real filaments would need to be made, which is not yet possible.

These results show that the current calculator is verifiably accurate for current models and observations, however it does have limitations. It currently only accounts for the thermal Sunyaev-Zeldovich effect and neglects kinetic and polarization effects. As well, the current classes only implement a handful of published results, and filaments are assumed to follow a double Gaussian radial profile with a constant density along their lengths. Due to the modular nature of the software, these limitations can be addressed by future developers using extensions to the abstract classes.

6 Iteration and Future Work

While initial verification shows that the results are similar to observations, there are still discrepancies that should be investigated to ensure the software is accurate. The angular power spectrum of galaxy clusters is overestimated at high ℓ values, and the bispectrum of clusters is also significantly higher than observed. Although previous attempts to recreate these results have shown a similar overestimate [8], the underlying cause should be investigated. The discrepancies may be caused by the mass distribution and radial profile models used, or they may be caused by rounding errors, unit discrepancies, insufficient integral accuracy, or errors in the software.

Following initial verification, optimization was done to improve performance and reduce unnecessary calculations. The code was parallelized so that multiple ℓ values are calculated simultaneously, which reduced computation time by 40% on an Intel i5-1035G1 CPU. The benefits from parallelization are potentially much larger on computers with more threads or more memory.

The Numba just-in-time compiler can automatically compile some Python code into machine code dynamically during runtime, significantly increasing the speed of frequently-used functions [24]. This was implemented for computationally intensive functions, resulting in an average reduction in computation time of 70%. Astropy Quantity objects are not compatible with Numba, so many functions could not be easily compiled using this strategy. To work around this, the functions could be modified to accept non-Quantity inputs, and the calls to the functions could explicitly provide the underlying primitive data type, then translated back to Quantity types after the function returns. This would make the code much harder to understand, and the performance benefits would be mitigated by the type switching, so this was not pursued. Alternative packages such as Cython or PyPy could also be investigated to improve computation speed.

To reduce unnecessary calculations, the MassCalculator class filters the array of ℓ values provided to only calculate the unique entries, reducing the time to compute the equilateral bispectrum by approximately 66%. This could also have been solved by requiring the user to manually filter ℓ values used as input, but to be as easy to use as possible the software will automatically filter extra inputs without any loss of accuracy.

A more complex algorithm to compute the power spectrum and bispectrum, which batches ℓ values together, could be implemented to further reduce computation time [3]. However, because computing the power spectrum and bispectrum of filaments at 180 ℓ values takes less than five minutes on an Intel i5-1035G1 CPU, the current performance was deemed adequate.

The bispectrum of filaments averaged over all orientations is approximately four orders of magnitude lower than that of galaxy clusters, which will makes searching for them extremely difficult, even for next-generation space telescopes. This software could be used to investigate the ideal bispectrum arrangement by fixing the magnitude of ℓ_3 and varying ℓ_1 and ℓ_2, thereby identifying the non-Gaussianity most prominent in the data. However, this is dependent on the filament model used to characterize the electron pressure, so the results will depend on future cosmological simulations providing more accurate models of filament structure.

7 Conclusion

The final product of this thesis is a modular, extensible, and verifiable open-source software package that can efficiently calculate the power spectrum and bispectrum of cosmological structures, including galaxy clusters and filaments. Usability was prioritized over optimization, although the final result is vectorized and parallelized so that it can produce a full set of results in less than five minutes. The validity of each class in the object-oriented design has been verified using published results, and example scripts are provided to produce the plots included in this thesis. The final software can be found on GitHub, released under the Berkeley Software Distribution license, at [25].

Word Count: 4,122

References

[1] J. E. Carlstrom, G. P. Holder and E. D. Reese, "The Sunyaev-Zel'Dovich Effect," 2002. [Online]. Available: https://ned.ipac.caltech.edu/level5/Sept05/Carlstrom/Carlstrom2.html. [Accessed 19 October 2022].

[2] N. Aghanim, C. Baccigalupi, A. J. Banday, R. B. Barreiro, E. Battaner, R. Battye, K. Benabed, J. J. Bock, J. R. Bond, J. Borrill, F. R. Bouchet, R. C. Butler, E. Calabrese, J.-F. Cardoso, A. Catalano and A. Challinor, "Planck 2015 results: XXII. A map of the thermal Sunyaev-Zeldovich effect," *Astronomy and astrophysics (Berlin),* vol. 594, p. A22, 2016.

[3] M. Bucher, B. van Tent and C. S. Carvalho, "Detecting bispectral acoustic oscillations from inflation using a new flexible estimator," *Monthly Notices of the Royal Astronomical Society,* vol. 407 (4), pp. 2193-2206, 2010.

[4] J. Swarts, "Open-Source Software in the Sciences: The Challenge of User Support," *Journal of business and technical communication,* vol. 33 (1), pp. 60-90, 2019.

[5] R. Sanders, "THE DEVELOPMENT AND USE OF SCIENTIFIC SOFTWARE," Queen's University, April 2008. [Online]. Available: https://qspace.library.queensu.ca/bitstream/handle/1974/1188/Sanders_Rebecca_J_200804_MSc.pdf. [Accessed 19 October 2022].

[6] C. Gheller, F. Vazza, M. Bruggen, M. Alpaslan, B. W. Holwerda, A. M. Hopkins and J. Liske, "Evolution of cosmic filaments and of their galaxy population from MHD cosmological simulations," *Monthly Notices of the Royal Astronomical Society,* vol. 462, no. 1, pp. 448-463, 2016.

[7] SPHINX Project, "The SPHINX Project - Cosmological radiation-hydrodynamical simulations of reionization," Centre de Recherche Astrophysique de Lyon, 30 June 2021. [Online]. Available: https://sphinx.univ-lyon1.fr/. [Accessed 19 November 2022].

[8] A. Vincent, Interviewee, *Private Communication.* [Interview]. 2022.

[9] T. Yang, M. J. Hudson and N. Afshordi, "A Universal Profile for Stacked Filaments from Cold Dark Matter Simulations," *Monthly Notices of the Royal Astronomical Society,* vol. 516 (4), pp. 6041-6054, 2022.

[10] A. Lewis, "The real shape of non-Gaussianities," *Journal of cosmology and astroparticle physics,* vol. 2011 (10), pp. 026-26, 27 July 2011.

[11] S. Bhattacharya, D. Nagai, L. Shaw, T. Crawford and G. P. Holder, "BISPECTRUM OF THE SUNYAEV–ZEL'DOVICH EFFECT," The Astrophysical Journal, 29 October 2012. [Online]. Available: https://iopscience.iop.org/article/10.1088/0004-637X/760/1/5. [Accessed 21 November 2022].

[12] E. Komatsu and U. Seljak, "The Sunyaev–Zel'dovich angular power spectrum as a probe of cosmological parameters," *Monthly Notices of the Royal Astronomical Society,* vol. 336, no. 4, pp. 1256-1270, 2002.

[13] Digital Research Alliance of Canada, "2022 Resource Allocations Competition Results," 9 February 2022. [Online]. Available: https://alliancecan.ca/en/services/advanced-research-computing/accessing-resources/resource-allocation-competitions/2022-resource-allocations-competition-results. [Accessed 27 November 2022].

[14] M. Bruce, "COMPARING THE BSD AND GPL LICENSES," *The Open Source Business Resource,* vol. 2007, no. October, pp. 20-23, 2007.

[15] D. Phillips, Python 3 Object Oriented Programming, Birmingham: Packt Publishing , 2010.

[16] A. J. Banday, K. M. Gorski, E. Hivon, E. Joliet, W. O'Mullane, M. Reinecke, L. Singer and A. Zonca, "HEALPix," 19 December 2019. [Online]. Available: https://healpix.sourceforge.io/. [Accessed 25 November 2022].

[17] A. C. Vincent, C. A. Arguelles, A. Kheirandish, I. Safa and K. Hoshina, "nuFATE: Neutrino Fast Attenuation Through Earth," 14 November 2020. [Online]. Available: https://github.com/aaronvincent/nuFATE. [Accessed 25 November 2022].

[18] The Astropy Collaboration, "Astropy," 2023. [Online]. Available: https://www.astropy.org/. [Accessed 31 March 2023].

[19] J. Carlstrom and e. al., "Cosmology with the Sunyaev-Zel'Dovich Effect," Annual Review of Astronomy and Astrophysics, 2002. [Online]. Available: https://ned.ipac.caltech.edu/level5/Sept05/Carlstrom/Carlstrom4.html. [Accessed 11 February 2023].

[20] M. Cautun, R. v. d. Weygaert, B. J. T. Jones and C. S. Frenk, "Evolution of the cosmic web," *Monthly Notices of the Royal Astronomical Society,* vol. 441, no. 4, p. 2923–2973, 2014.

[21] M. Arnaud, G. W. Pratt, R. Piffaretti, H. Bohringer, J. H. Croston and E. Pointecouteau, "The universal galaxy cluster pressure profile from a representative sample of nearby systems (REXCESS) and the Y_SZ - M_500 relation," *Astronomy & Astrophysics,* vol. 517, p. A92, 2010.

[22] J. Tinker, A. V. Kravtsov, A. Klypin, K. Abazajian, M. Warren, G. Yepes, S. Gottlöber and D. E. Holz, "Toward a Halo Mass Function for Precision Cosmology: The Limits of Universality," *The Astrophysical Journal,* vol. 688, no. 2, pp. 709-728, 2008.

[23] L. Lopez-Honorez, O. Mena, S. Palomares-Ruiz and A. C. Vincent, "Constraints on dark matter annihilation from CMB observations before Planck," *Journal of Cosmology and Astroparticle Physics,* vol. 2013, no. 07, 2013.

[24] Anaconda, "Numba: A High Performance Python Compiler," 2018. [Online]. Available: https://numba.pydata.org/. [Accessed 29 March 2023].

[25] J. Fawcett, "PySZ," March 2023. [Online]. Available: https://github.com/JonathanFawcett/PySZ. [Accessed 29 March 2023].