

# User Manual for FFTLog-and-Beyond

Xiao Fang

## 1 Versions

- Version 2.0: 2021/07/14. `fftlogx` released. Supported new forms of integrals. Upgraded for use in LSST-DESC N5K (Non-Limber method) Challenge.
- Version 1.0: 2019/11/22. Independent pure Python version and C version released.

## 2 Problem Description

The integrals we are solving are

$$F(y) = \int_0^\infty \frac{dx}{x} f(x) j_\ell^{(n)}(xy) , \quad (1)$$

$$H(y) = \int_0^\infty \frac{dx}{x} f(x) |j_\ell(xy)|^2 , \quad (2)$$

where  $f(x)$  is an input array,  $j_\ell$  is the order- $\ell$  spherical Bessel function of the first kind, the superscript  $^{(n)}$  denotes the order of derivative. This type of integrals are numerically challenging due to the rapidly oscillatory nature of the spherical Bessel functions, especially when the input  $f(x)$  data array correspond to sampling array  $x$  over a large range (*i.e.*, over several orders of magnitude).

In weak lensing, the integral often takes form of

$$S(y) = \int_0^\infty \frac{dx}{x} f(x) \frac{j_\ell(xy)}{(xy)^2} , \quad (3)$$

which could be solved by moving  $y$  dependence outside of the integral, and reducing to form  $F(y)$ . However, we will present a numerically better approach.

## 3 Efficient Computation with FFTLog-and-Beyond

The essential idea is to expand  $f(x)$  into a series of power-laws and solve each component integral analytically. We require a logarithmic sampling of  $x$  with linear spacing in  $\ln(x)$  equal to  $\Delta_{\ln x}$ , *i.e.*,  $x_q = x_0 \exp(q\Delta_{\ln x})$  with  $x_0$  being the smallest value in the  $x$  array. The power-law decomposition then means

$$f(x_q) = \frac{1}{N} \sum_{m=-N/2}^{N/2} c_m x_0^\nu \left( \frac{x_q}{x_0} \right)^{\nu+i\eta_m} , \quad (4)$$

where  $N$  is the sample size of the input function,  $\eta_m = 2\pi m/(N\Delta_{\ln x})$ , and  $\nu$  is the bias index. The Fourier coefficients satisfy  $c_m^* = c_{-m}$  since function  $f(x)$  is real, and are computed by discrete Fourier transforming the “biased” input function  $f(x)/x^\nu$  as

$$c_m = W_m \sum_{q=0}^{N-1} \frac{f(x_q)}{x_q^\nu} e^{-2\pi i m q/N} , \quad (5)$$

where  $W_m$  is a window function which smooths the edges of the  $c_m$  array and takes the form of Eq. (C.1) in McEwen et al (2016, arXiv: 1603.04826). This filtering is found to reduce the ringing effects.

Each term is now analytically solvable. For  $n = 0$ , *i.e.*, no derivative,

$$\begin{aligned} F(y) &= \frac{1}{N y^\nu} \sum_{m=-N/2}^{N/2} c_m x_0^{-i\eta_m} y^{-i\eta_m} \int_0^\infty \frac{dx}{x} x^{\nu+i\eta_m} j_\ell(x) \\ &= \frac{\sqrt{\pi}}{4N y^\nu} \sum_{m=-N/2}^{N/2} c_m x_0^{-i\eta_m} y^{-i\eta_m} g_\ell(\nu + i\eta_m) , \end{aligned} \quad (6)$$

where the first equality uses change of variable  $xy \rightarrow x$ . Function  $g_\ell(z)$  is given by

$$g_\ell(z) = 2^z \frac{\Gamma\left(\frac{\ell+z}{2}\right)}{\Gamma\left(\frac{3+\ell-z}{2}\right)} , \quad -\ell < \Re(z) < 2 , \quad (7)$$

giving the range of bias index  $-\ell < \nu < 2$ .

Finally, assuming that  $y$  is logarithmically sampled with the same linear spacing  $\Delta_{\ln y} = \Delta_{\ln x}$  in  $\ln y$ , we can write the last summation in Eq. (6) as

$$F(y_p) = \frac{\sqrt{\pi}}{4y_p^\nu} \text{IFFT} [c_m^*(x_0 y_0)^{i\eta_m} g_\ell(\nu - i\eta_m)] , \quad (8)$$

where  $y_p$  ( $p = 0, 1, \dots, N-1$ ) is the  $p$ -th element in the  $y$  array. IFFT stands for the Inverse Fast Fourier Transform. In summary, this method performs two FFT operations, one in computing  $c_m$ , one in the final summation over  $m$ . Thus, the total time complexity is  $\mathcal{O}(N \log N)$ . So far we have described the principle of the FFTLog algorithm.

For  $n > 0$ , following the same procedure of power-law decomposition, we have

$$F_n(y) = \frac{1}{Ny^\nu} \sum_{m=-N/2}^{N/2} c_m x_0^{-i\eta_m} y^{-i\eta_m} \int_0^\infty \frac{dx}{x} x^{\nu+i\eta_m} j_\ell^{(n)}(x) . \quad (9)$$

Again, the integral for each  $m$  has an analytic solution, which can be shown with integration by parts. We write the solution in the same form with the FFTLog, *i.e.*,

$$F_n(y) = \frac{\sqrt{\pi}}{4Ny^\nu} \sum_{m=-N/2}^{N/2} c_m x_0^{-i\eta_m} y^{-i\eta_m} \tilde{g}_\ell(n, \nu + i\eta_m) , \quad (10)$$

and its discrete version assuming  $\Delta_{\ln y} = \Delta_{\ln x}$ ,

$$F_n(y_p) = \frac{\sqrt{\pi}}{4y_p^\nu} \text{IFFT} [c_m^*(x_0 y_0)^{i\eta_m} \tilde{g}_\ell(n, \nu - i\eta_m)] , \quad (11)$$

where  $\tilde{g}_\ell(n, z) = 4\pi^{-1/2} \int_0^\infty dx x^{z-1} j_\ell^{(n)}(x)$ . For  $n = 0$ ,  $\tilde{g}_\ell(0, z) = g_\ell(z)$ , and for  $n = 1, 2$ , it is given by

$$\tilde{g}_\ell(1, z) = -2^{z-1}(z-1) \frac{\Gamma(\frac{\ell+z-1}{2})}{\Gamma(\frac{4+\ell-z}{2})} , \quad \left( \begin{array}{ll} 0 < \Re(z) < 2 , & \text{for } \ell = 0 \\ 1 - \ell < \Re(z) < 2 , & \text{for } \ell \geq 1 \end{array} \right) , \quad (12)$$

$$\tilde{g}_\ell(2, z) = 2^{z-2}(z-1)(z-2) \frac{\Gamma(\frac{\ell+z-2}{2})}{\Gamma(\frac{5+\ell-z}{2})} , \quad \left( \begin{array}{ll} -\ell < \Re(z) < 2 , & \text{for } \ell = 0, 1 \\ 2 - \ell < \Re(z) < 2 , & \text{for } \ell \geq 2 \end{array} \right) . \quad (13)$$

We choose  $\nu = 1$  for all  $\ell$ 's. With this generalized FFTLog algorithm, the integral containing one derivative of a spherical Bessel function also takes 2 FFT operations to compute.

Most generally, for  $n \in \mathcal{N}$ ,

$$\int_0^\infty dx x^{\alpha-1} j_\ell^{(n)}(x) = (-1)^n \frac{\sqrt{\pi}}{4} 2^{\alpha-n} \frac{\Gamma(\alpha)}{\Gamma(\alpha-n)} \frac{\Gamma(\frac{\ell+\alpha-n}{2})}{\Gamma(\frac{3+n+\ell-\alpha}{2})} , \quad \left( \begin{array}{ll} -\ell < \Re(\alpha) < 2 , & \text{for } \ell < n \\ n - \ell < \Re(\alpha) < 2 , & \text{for } \ell \geq n \end{array} \right) . \quad (14)$$

For  $H(y)$ , a similar method can be developed.

$$\begin{aligned} H(y) &= \frac{1}{Ny^\nu} \sum_{m=-N/2}^{N/2} c_m x_0^{-i\eta_m} y^{-i\eta_m} \int_0^\infty \frac{dx}{x} x^{\nu+i\eta_m} |j_\ell(x)|^2 \\ &= \frac{\sqrt{\pi}}{4Ny^\nu} \sum_{m=-N/2}^{N/2} c_m x_0^{-i\eta_m} y^{-i\eta_m} h_\ell(\nu + i\eta_m) , \end{aligned} \quad (15)$$

where

$$h_\ell(z) = \frac{4}{\sqrt{\pi}} \frac{\pi}{8} 2^z \frac{\Gamma(\ell + \frac{z}{2}) \Gamma(2-z)}{\Gamma(2+\ell-\frac{z}{2}) [\Gamma(\frac{3-z}{2})]^2} = \frac{\Gamma(\ell + \frac{z}{2}) \Gamma(\frac{2-z}{2})}{\Gamma(2+\ell-\frac{z}{2}) \Gamma(\frac{3-z}{2})} , \quad (16)$$

where the second equality utilizes the gamma function's duplication formula,  $\Gamma(2-z) = \frac{2^{1-z}}{\sqrt{\pi}} \Gamma(\frac{2-z}{2}) \Gamma(\frac{3-z}{2})$ . The final result in discrete form is then

$$H(y_p) = \frac{\sqrt{\pi}}{4y_p^\nu} \text{IFFT} [c_m^*(x_0 y_0)^{i\eta_m} h_\ell(\nu - i\eta_m)] , \quad (17)$$

For  $S(y)$ , following the power-law decomposition, we have

$$S(y) = \frac{1}{Ny^\nu} \sum_{m=-N/2}^{N/2} c_m x_0^{-i\eta_m} y^{-i\eta_m} \int_0^\infty \frac{dx}{x} x^{\nu+i\eta_m} \frac{j_\ell(x)}{x^2} . \quad (18)$$

Now the discrete result becomes

$$S(y_p) = \frac{\sqrt{\pi}}{4y_p^\nu} \text{IFFT} [c_m^*(x_0 y_0)^{i\eta_m} g_\ell(\nu - 2 - i\eta_m)] . \quad (19)$$

Similar expressions can be derived for integrals with derivatives.

## 4 Using the Code

This repository contains independent python module and C module for computing Eqs. (1-3).

### 4.1 fftlogx Version: C implementation + Python wrapper

This is the most recent and most recommended version. It has C code in `src/` and a python wrapper in `fftlogx/`. The code contains all functionality presented in this Note.

To use the code, run `make` command at the repository, which will build the C file and generate a shared library file `libfftlogx.so`. The python wrapper should then be used. See test files in `fftlogx/` for examples.

### 4.2 Pure Python Version (Version 1, not recommended)

The main code is all in `python/fftlog.py`, along with `python/test.py` which provides an example of calling the module to compute the integrals with  $f(x)$  being the power spectrum in `Pk_test`, and  $n = 0, 1, 2$ , respectively. We also have a Hankel function, defined as replacing  $j_\ell$  as  $J_n$  in Eq. (1). If you find them useful in your research, please cite Fang et al (2019).

### 4.3 C Version (Version 1, not recommended)

The code, sitting in `cfftlog` folder, uses FFTW library. The main code is in `cfftlog.c`, with auxiliary functions (*e.g.*, extrapolation, window function,  $g_\ell$  functions) defined in `utils.c` and `utils_complex.c`.

`cfftlog.c` provides different ways to run the computation. `cfftlog` provides the simplest usage. `cfftlog_ells` function enables to compute the same integral with an array of  $\ell$  values. This is more optimal than calling single function `cfftlog` many times since FFTW plans have to be re-created and destroyed over and over again, and same for the FFTW\_COMPLEX arrays used for the FFTs.

`cfftlog_ells_increment` does the same thing as `cfftlog_ells`, but increment the results (instead of refreshing). This saves memory for some applications.

`test.c` and `test_ells.c` show examples of calling those functions.