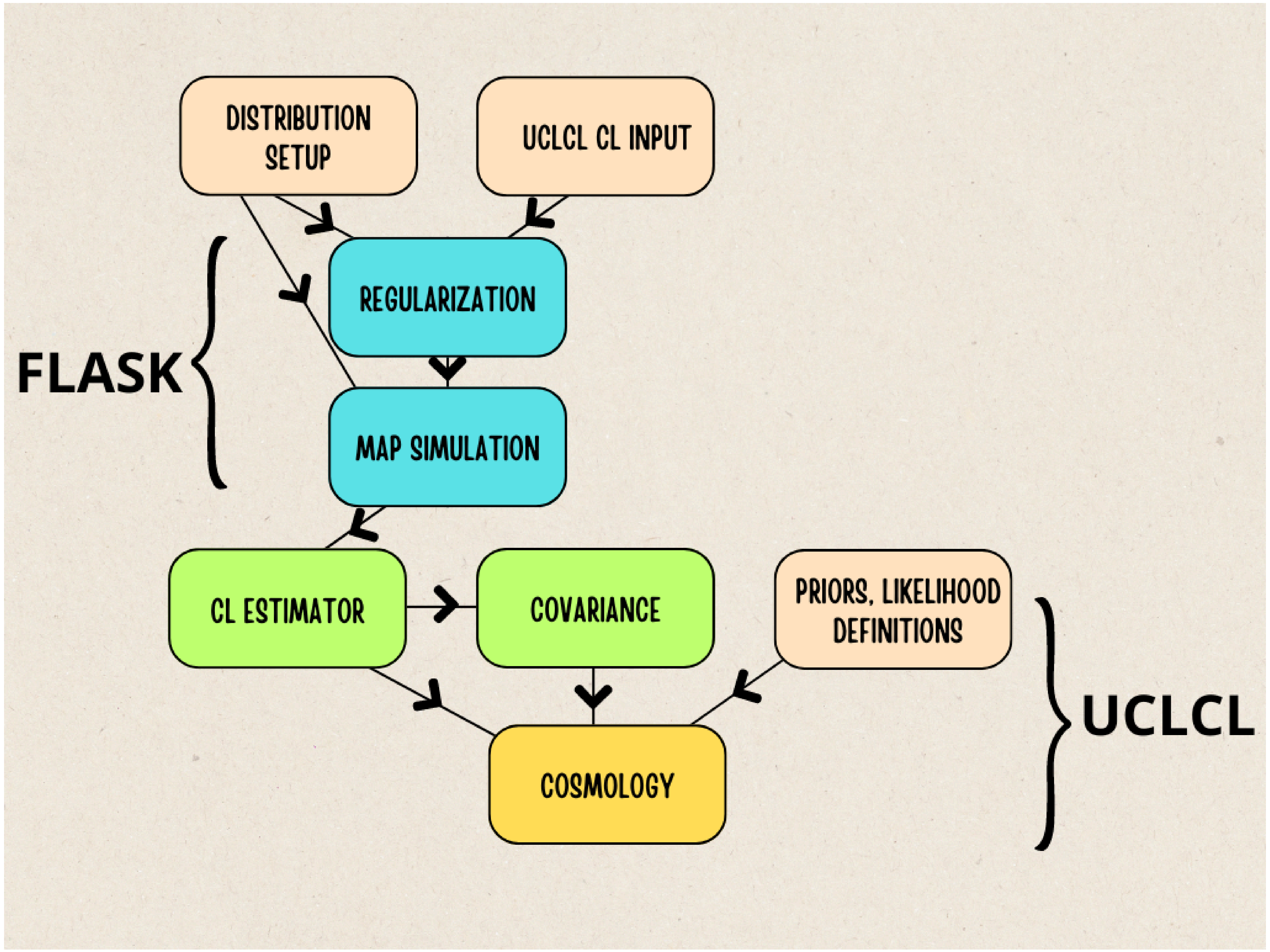
Exploring BINGO 21cm Cls for Bayesian inference

PABLO MOTTA

UNIVERSIDADE DE SÃO PAULO

Project flow chart

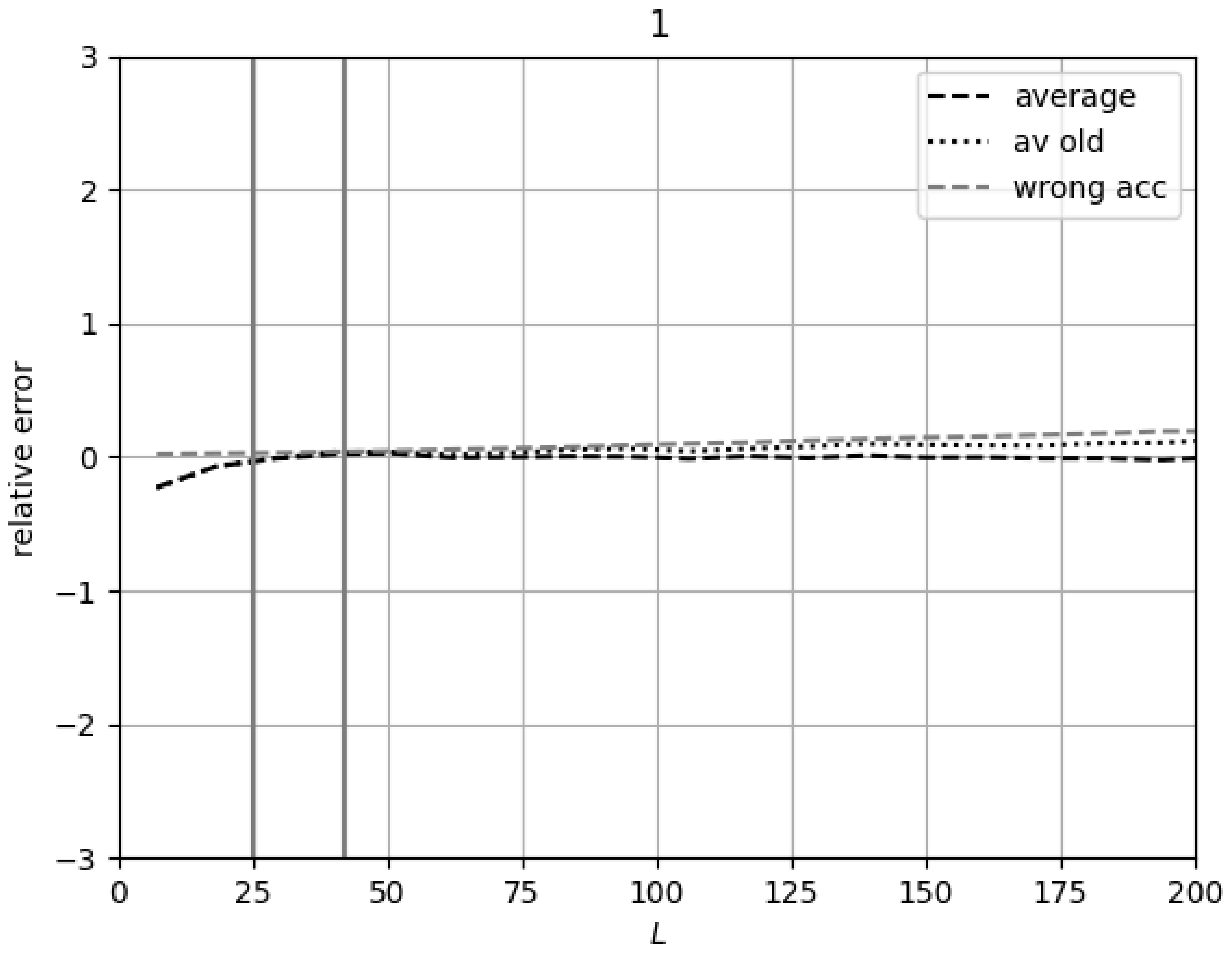


Things that impact the CL fitting

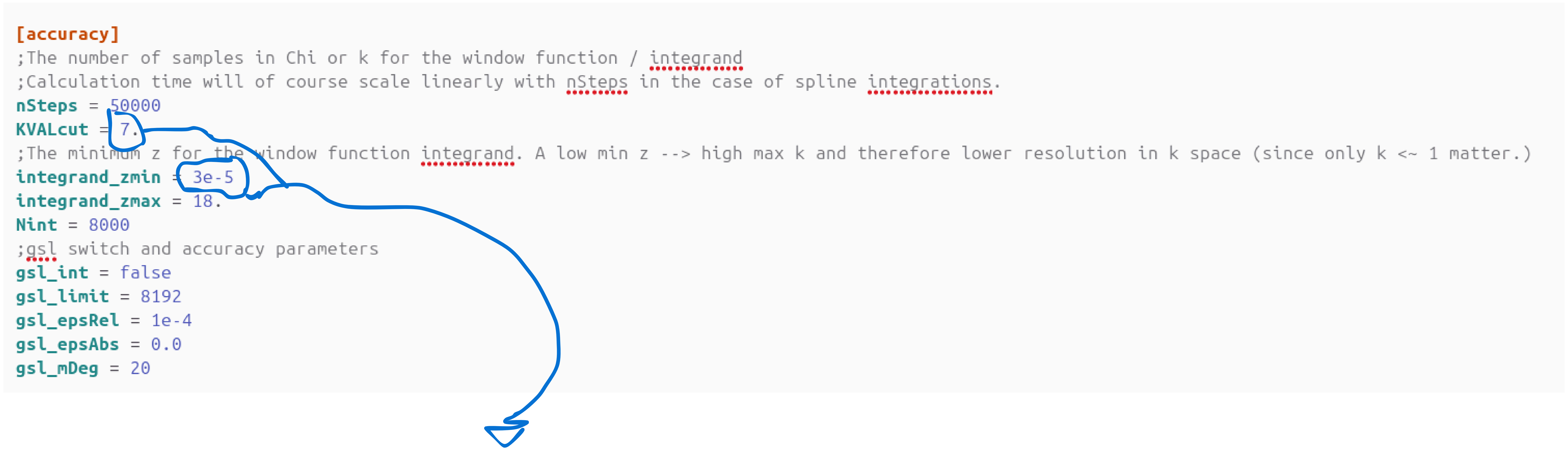
- Flask regularization.
- The 1-point distribution of the temperature.
- Cl estimator from maps
- The theoretical Cl that is a FLASK input to be consistent with the model being fitted in terms accuracy parameters and other non-cosmological paramters in UCLCL ini.

Errors on the average Cl

Average follows the theory



Fixing wrong accuracy values

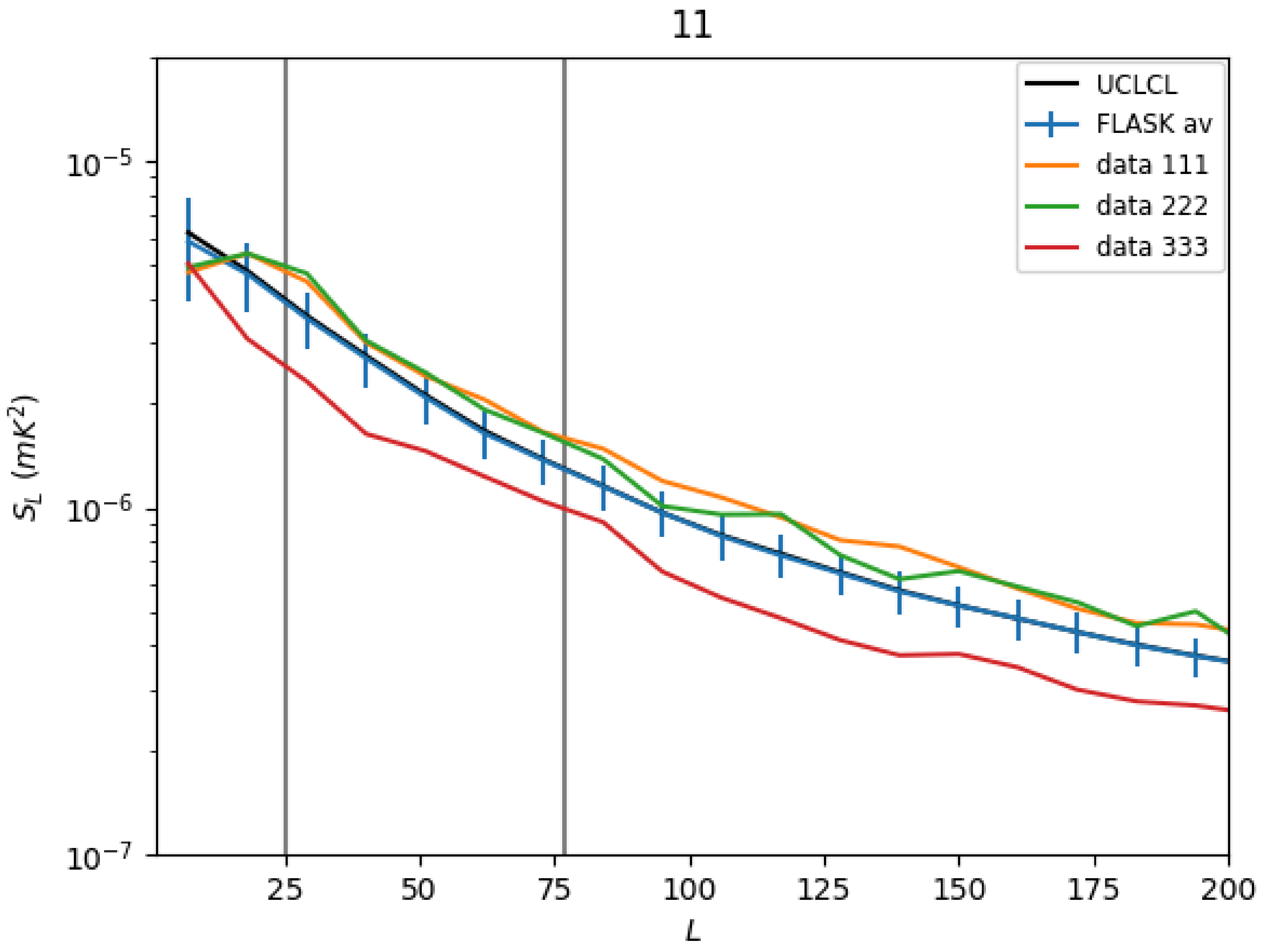


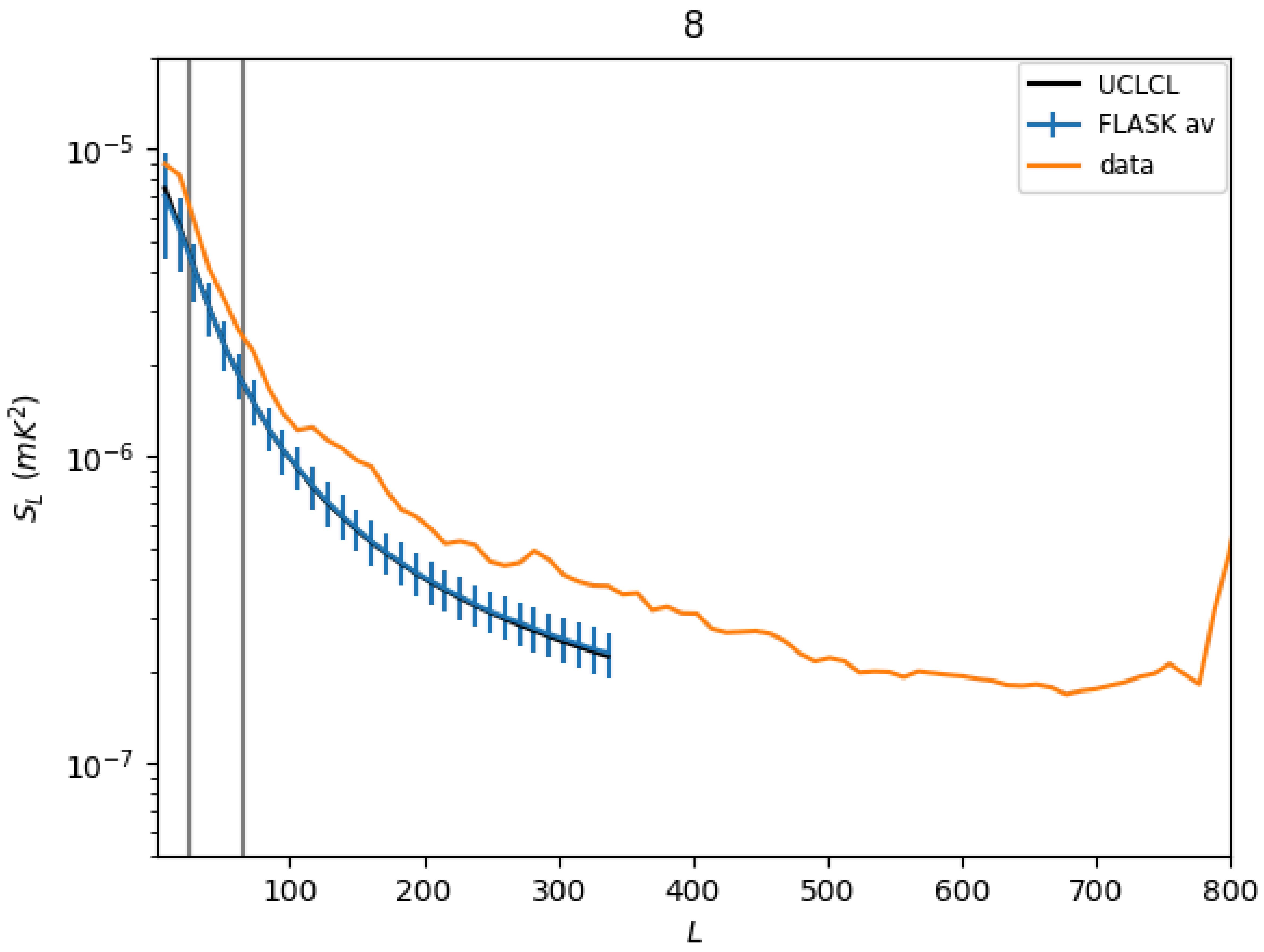
```
[accuracy]
;The number of samples in Chi or k for the window function / integrand
;Calculation time will of course scale linearly with nSteps in the case of spline integrations.
nSteps = 50000
KVALcut = 7.
;The minimum z for the window function integrand. A low min z --> high max k and therefore lower resolution in k space (since only k <~ 1 matter.)
integrand_zmin = 3e-5
integrand_zmax = 18.
Nint = 8000
;gsl switch and accuracy parameters
gsl_int = false
gsl_limit = 8192
gsl_epsRel = 1e-4
gsl_epsAbs = 0.0
gsl_mDeg = 20
```

3, 1e-4 -> 7, 3e-5

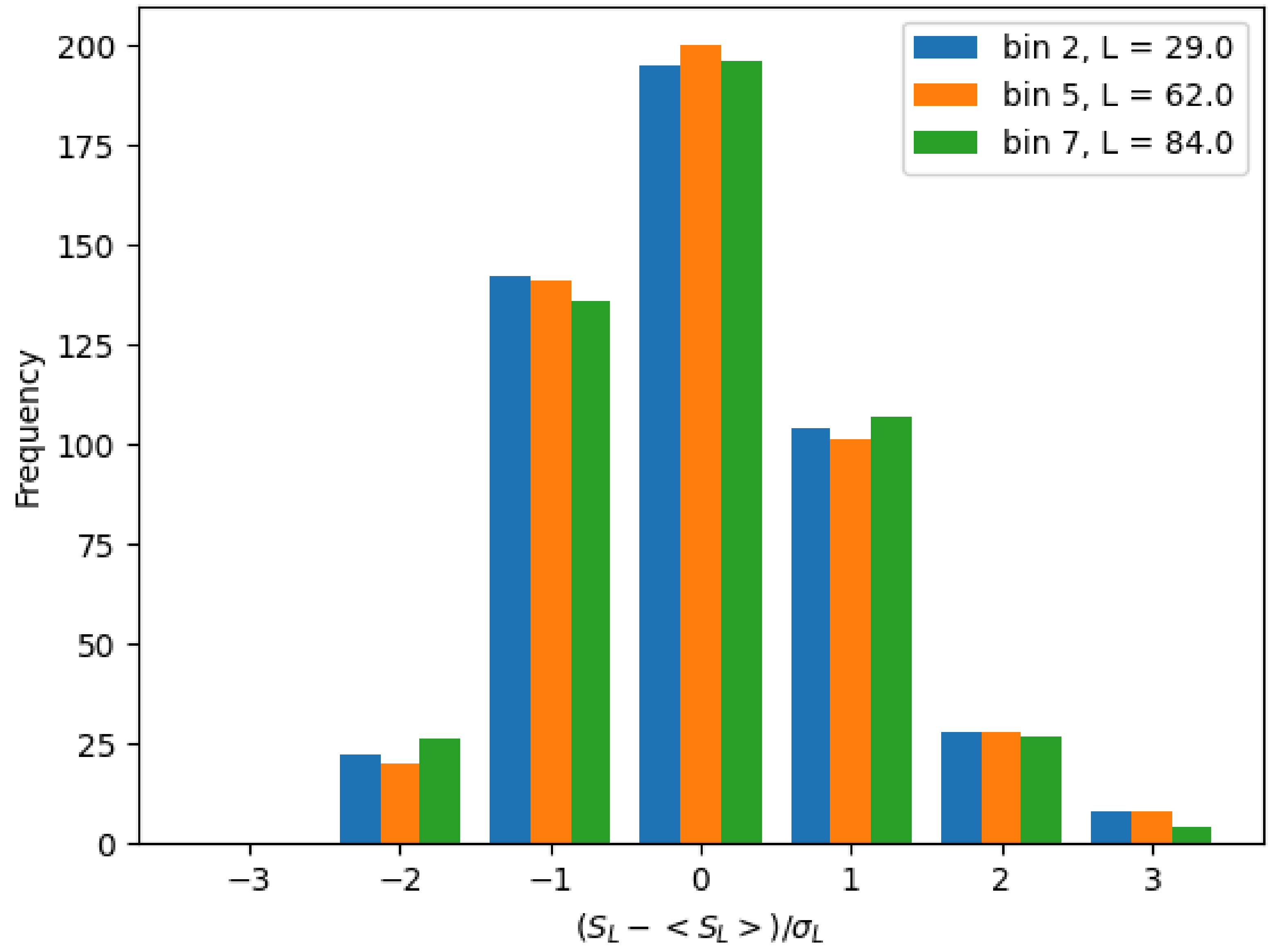
Errors on the Cl of a realization map

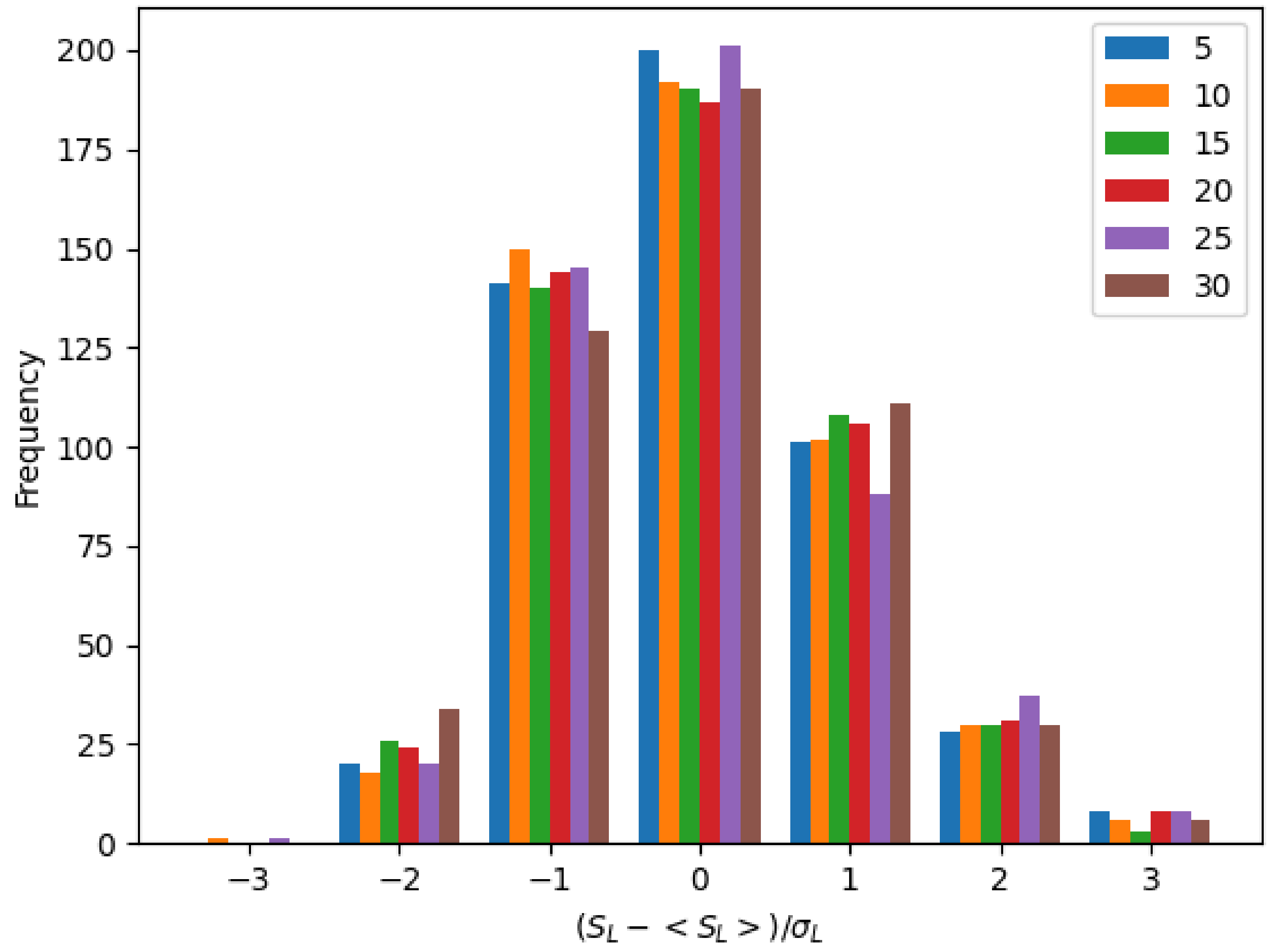
Correlated cosmic variance




8
UCLCL
FLASK av
data
10^{-5}
10^{-6}
10^{-7}
$S_L\ (mK^2)$
100
200
300
400
500
600
700
800
L

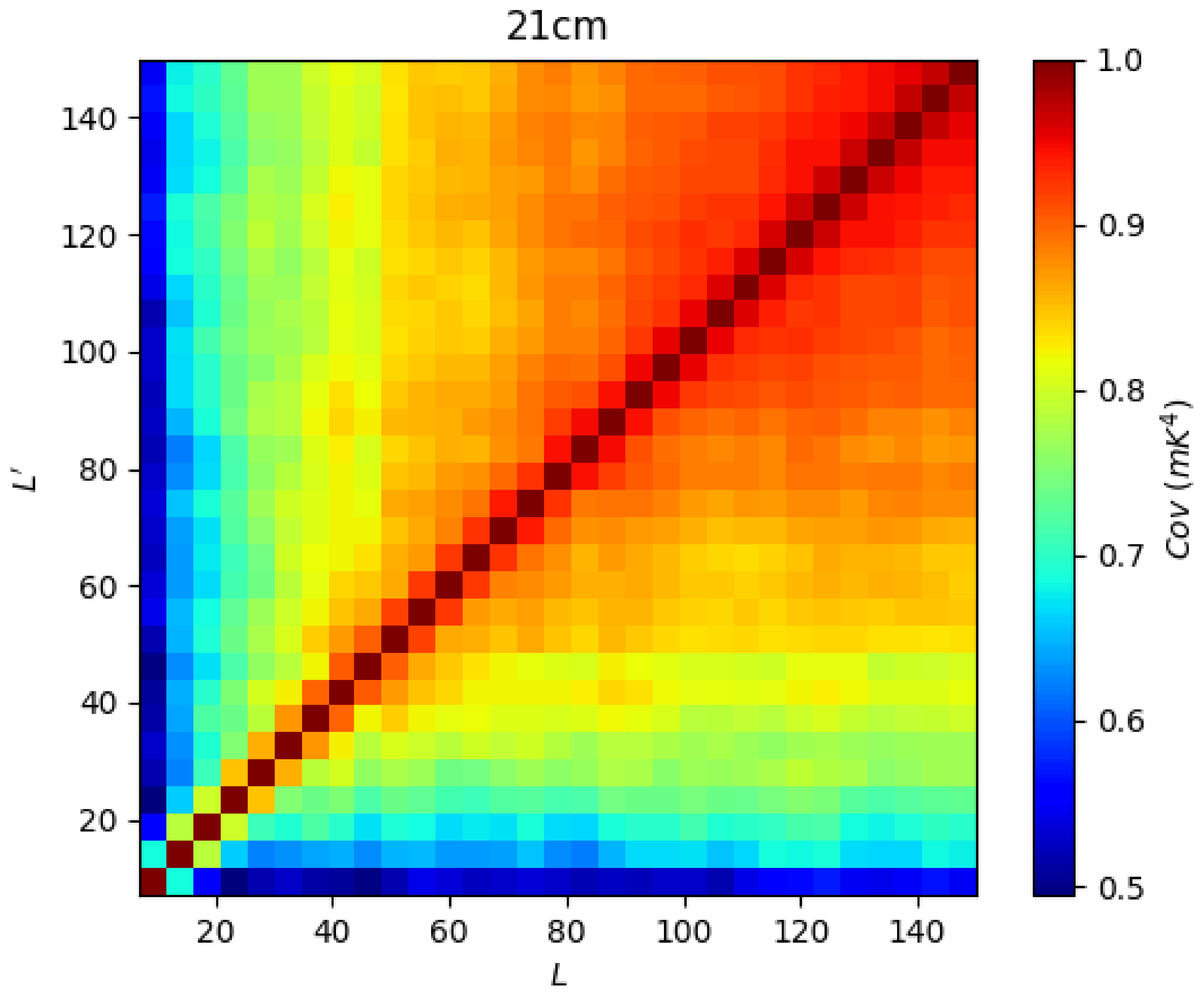
Histograms of Cls





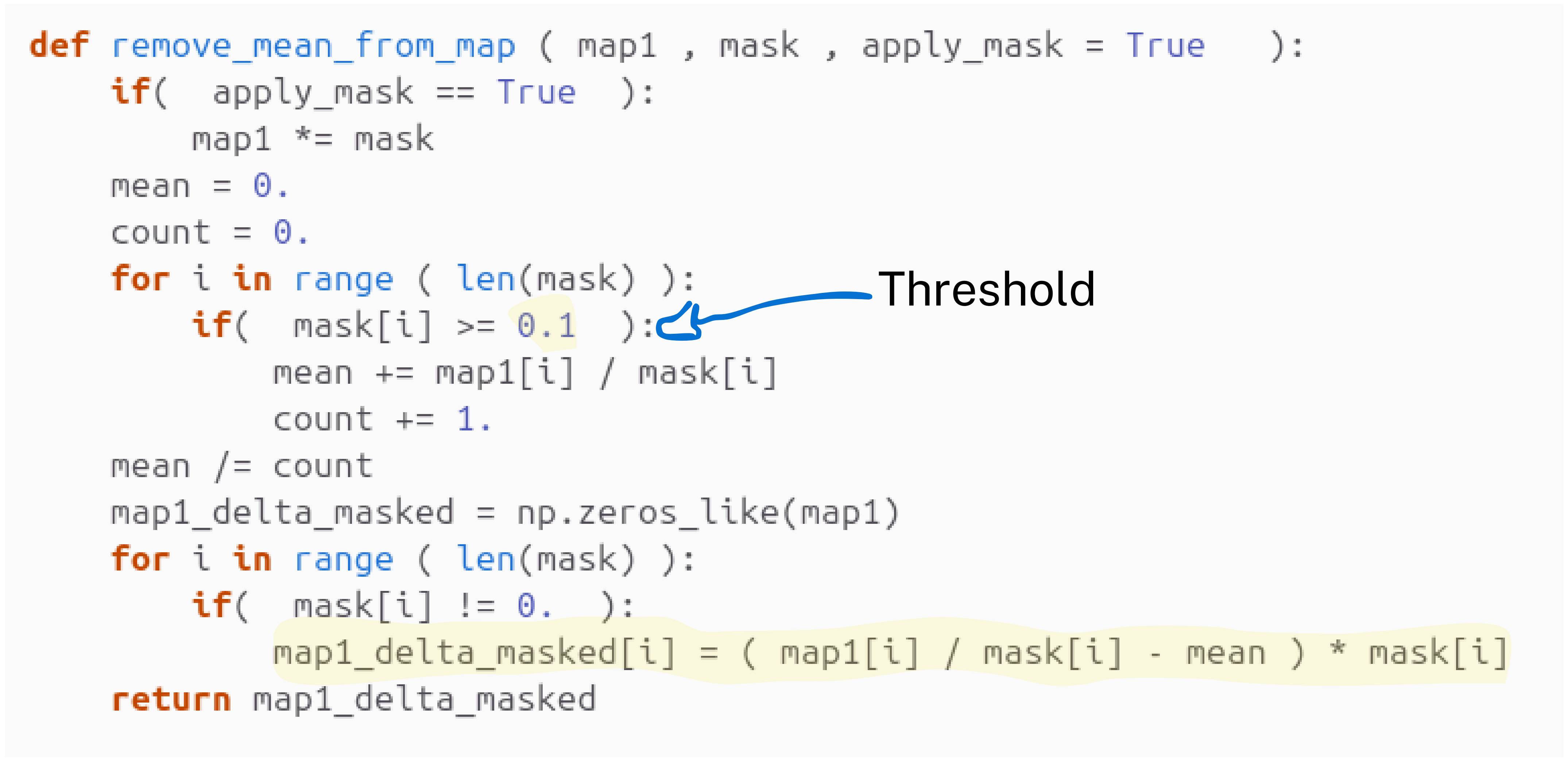
Histograms are what we expect although the realization values are weird

Pearson correlation



Cl estimator

Removing mean of map



```python
def remove_mean_from_map ( map1 , mask , apply_mask = True   ):
    if(  apply_mask == True   ):
        map1 *= mask
    mean = 0.
    count = 0.
    for i in range ( len(mask) ):
        if(  mask[i] >= 0.1   ):
            mean += map1[i] / mask[i]
            count += 1.
    mean /= count
    map1_delta_masked = np.zeros_like(map1)
    for i in range ( len(mask) ):
        if(  mask[i] != 0.   ):
            map1_delta_masked[i] = ( map1[i] / mask[i] - mean ) * mask[i]
    return map1_delta_masked
```

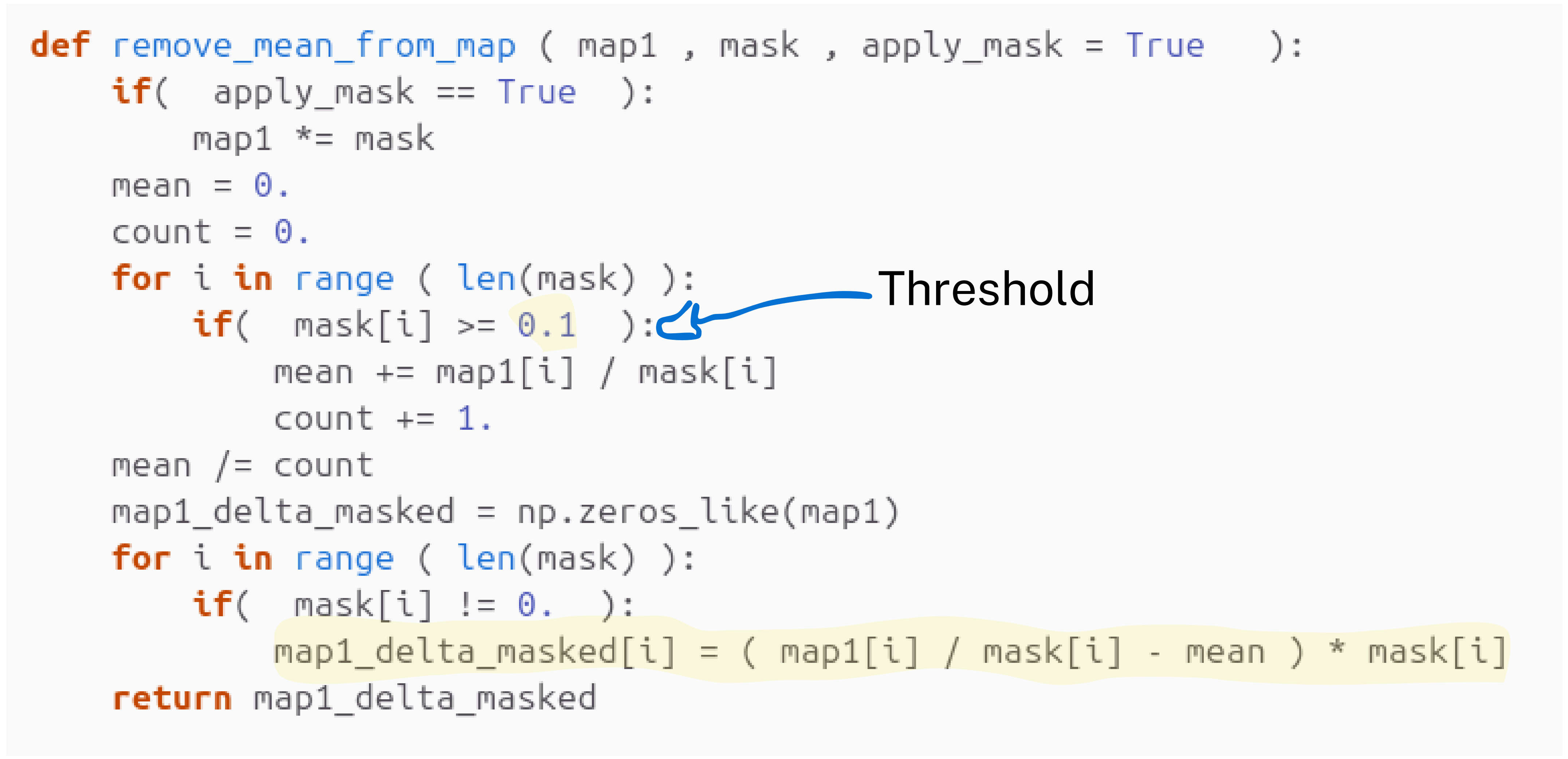
Threshold

Estimator itself

$$\hat{S}_\ell^{ij} = \frac{1}{\underbrace{f_{sky}}_{\text{Normalization}} b_\ell^2 w_\ell^2} \sum_{m=-\ell}^{\ell} a_{\ell m} a_{\ell m}^*$$

Cl estimator

Removing mean of map



```
def remove_mean_from_map ( map1 , mask , apply_mask = True    ):
    if(  apply_mask == True   ):
        map1 *= mask
    mean = 0.
    count = 0.
    for i in range ( len(mask) ):
        if(  mask[i] >= 0.1   ):
            mean += map1[i] / mask[i]
            count += 1.
    mean /= count
    map1_delta_masked = np.zeros_like(map1)
    for i in range ( len(mask) ):
        if(  mask[i] != 0.  ):
            map1_delta_masked[i] = ( map1[i] / mask[i] - mean ) * mask[i]
    return map1_delta_masked
```

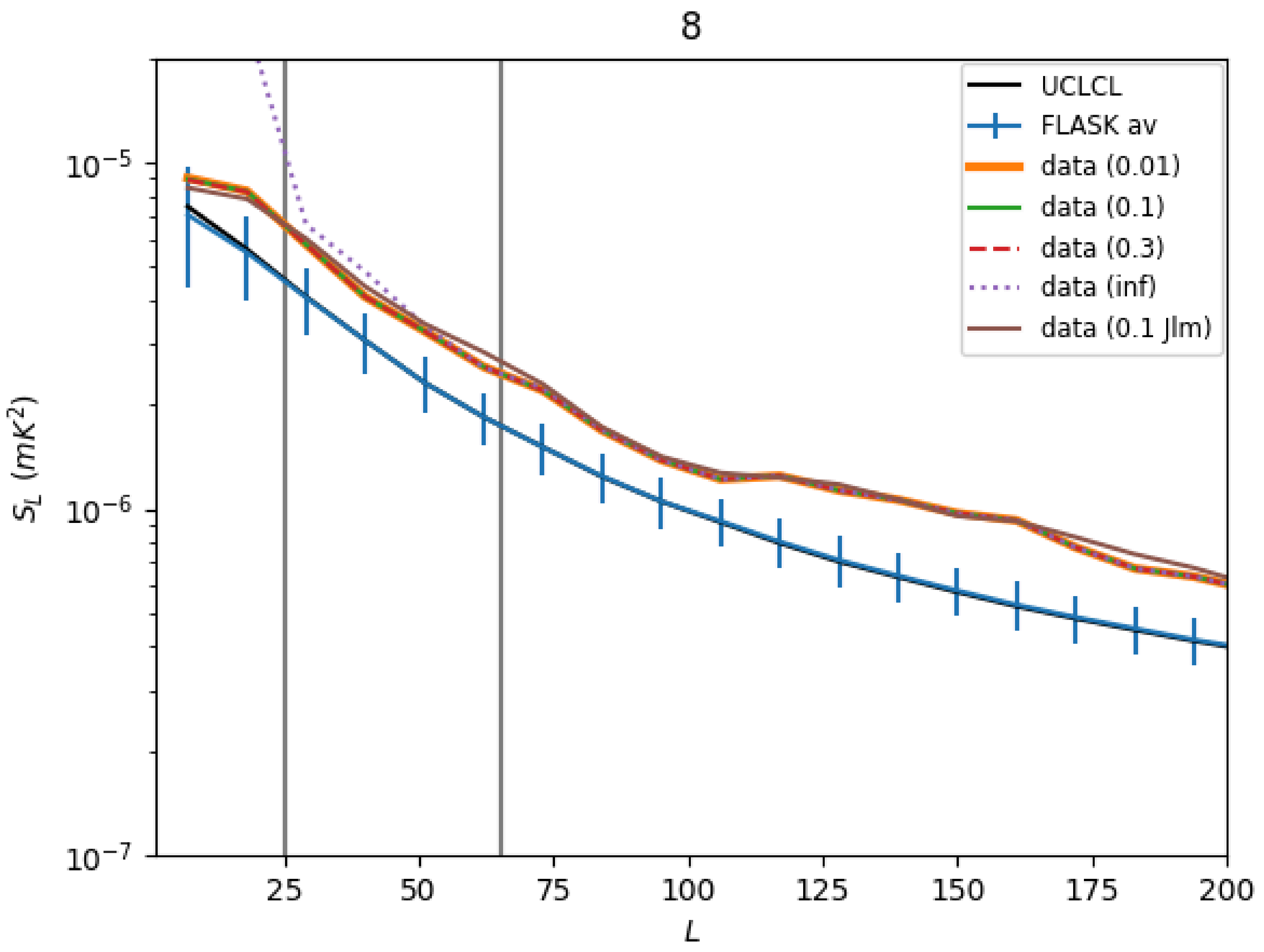
Threshold

Estimator itself

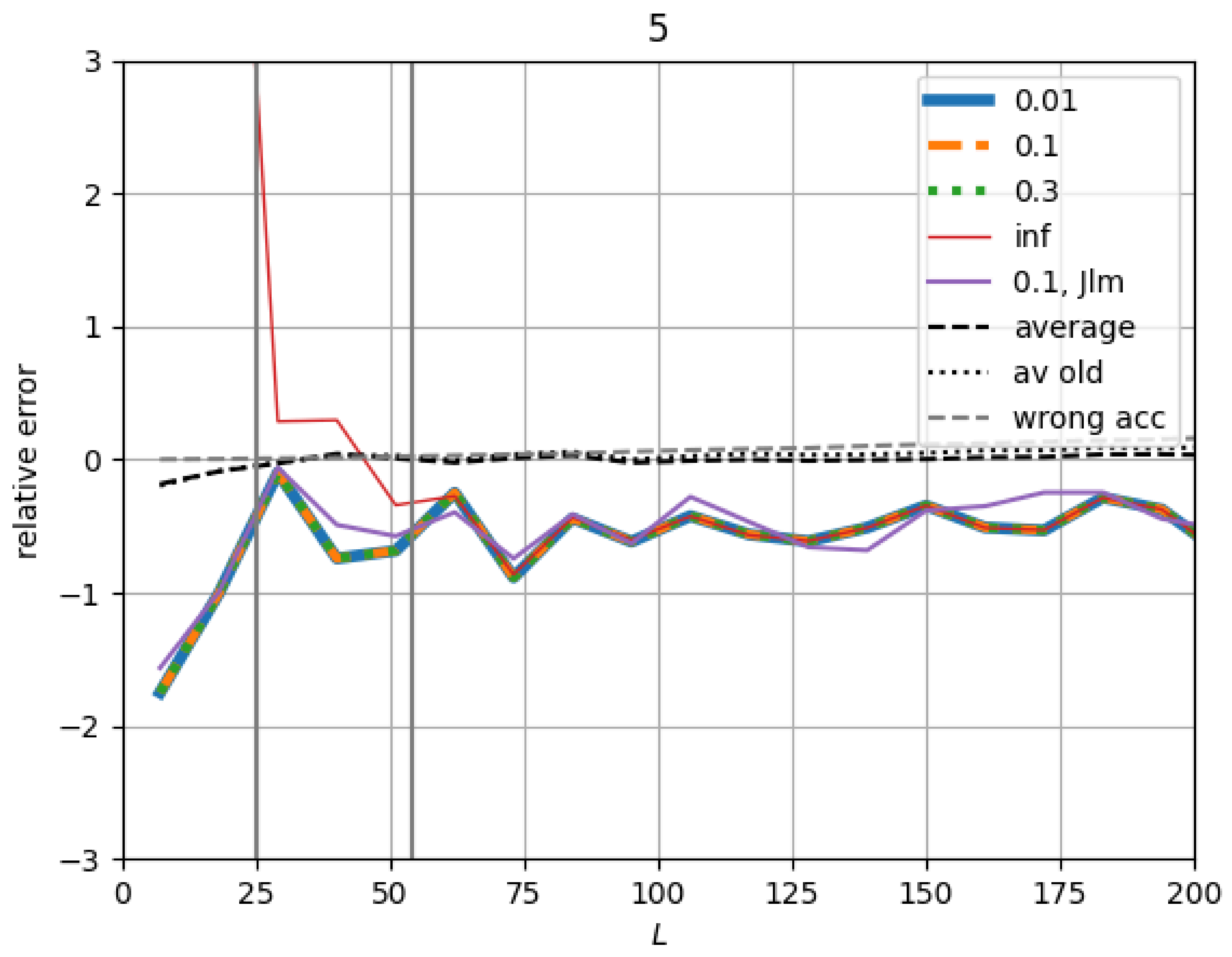
$$\hat{S}_\ell^{ij} = \frac{1}{\cancel{f_{sky}} b_\ell^2 w_\ell^2} \sum_{m=-\ell}^{\ell} \frac{a_{\ell m} a^*_{\ell m}}{J_{\ell m}}$$

Normalization

Cl estimator



Cl estimator



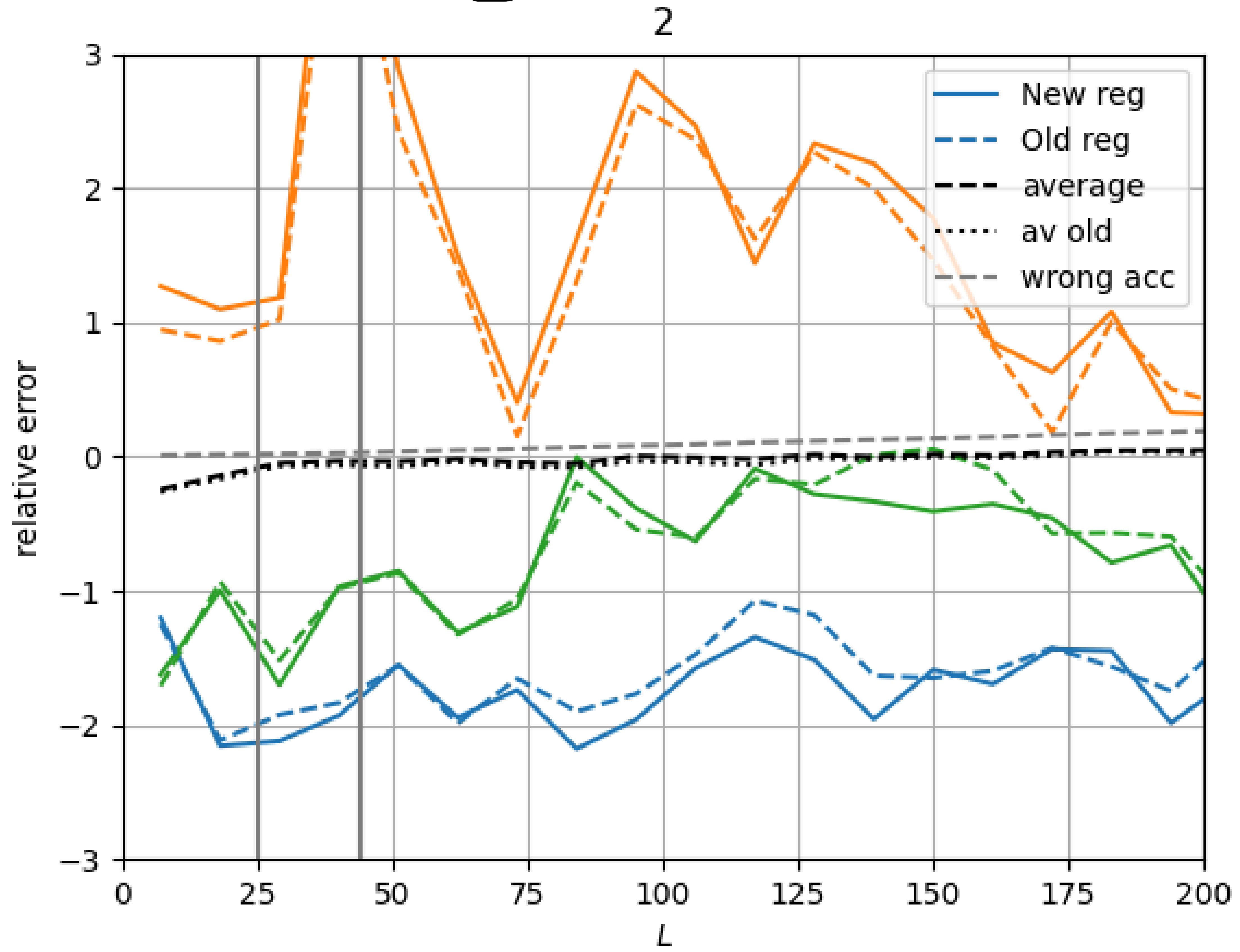
Improving Regularization

We improved the FLASK regularization by tuning some FLASK parameters

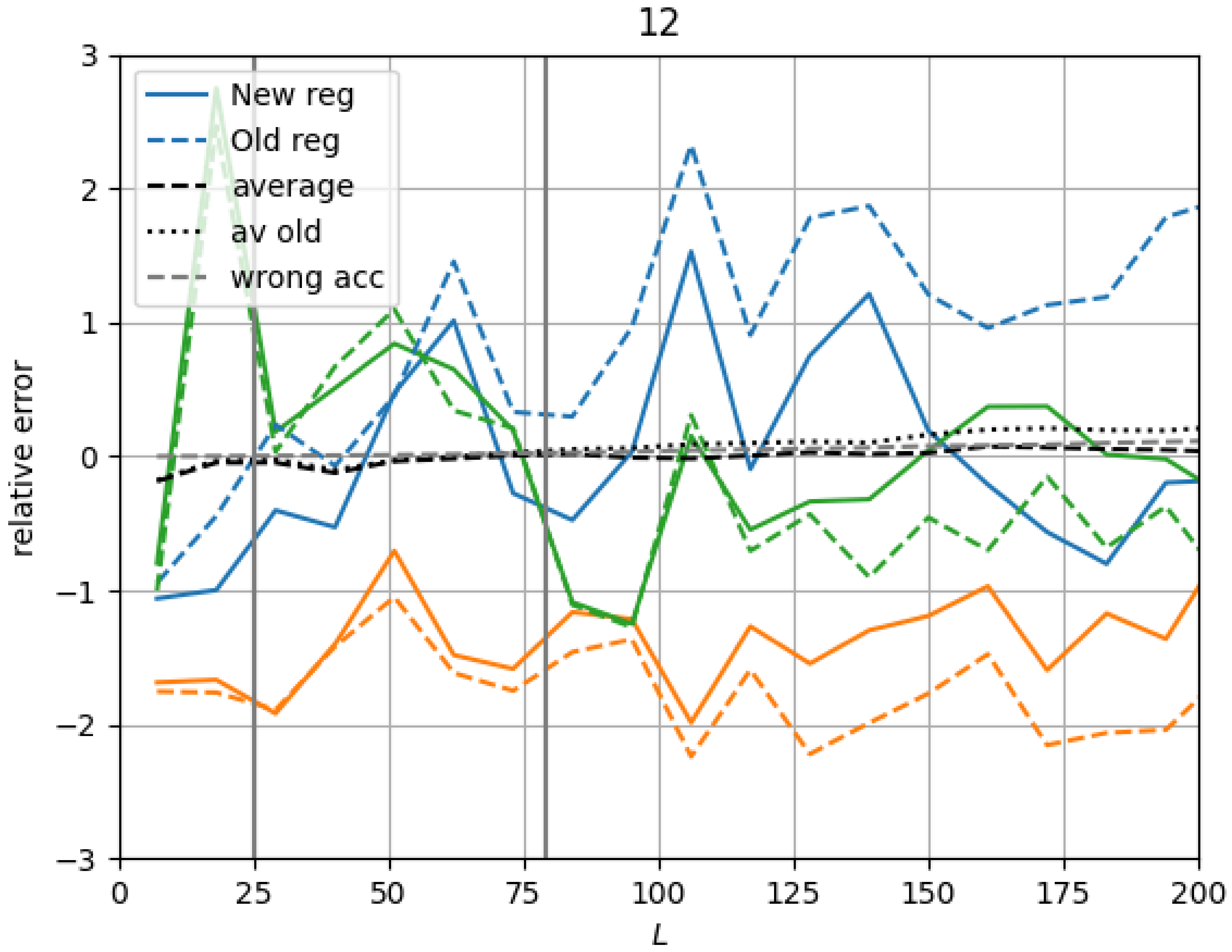
- **Old regularization:** Nside = X, Lmax=2500
- **New regularization:** Nside = 4096, Lmax = 8000 (where for L>2500 we define Cl=0)

The new regularization decreased the sum of absolute errors on the average Cls from 1.6 sigma to 1.1 sigma! However it has a different impact on the realization Cl.

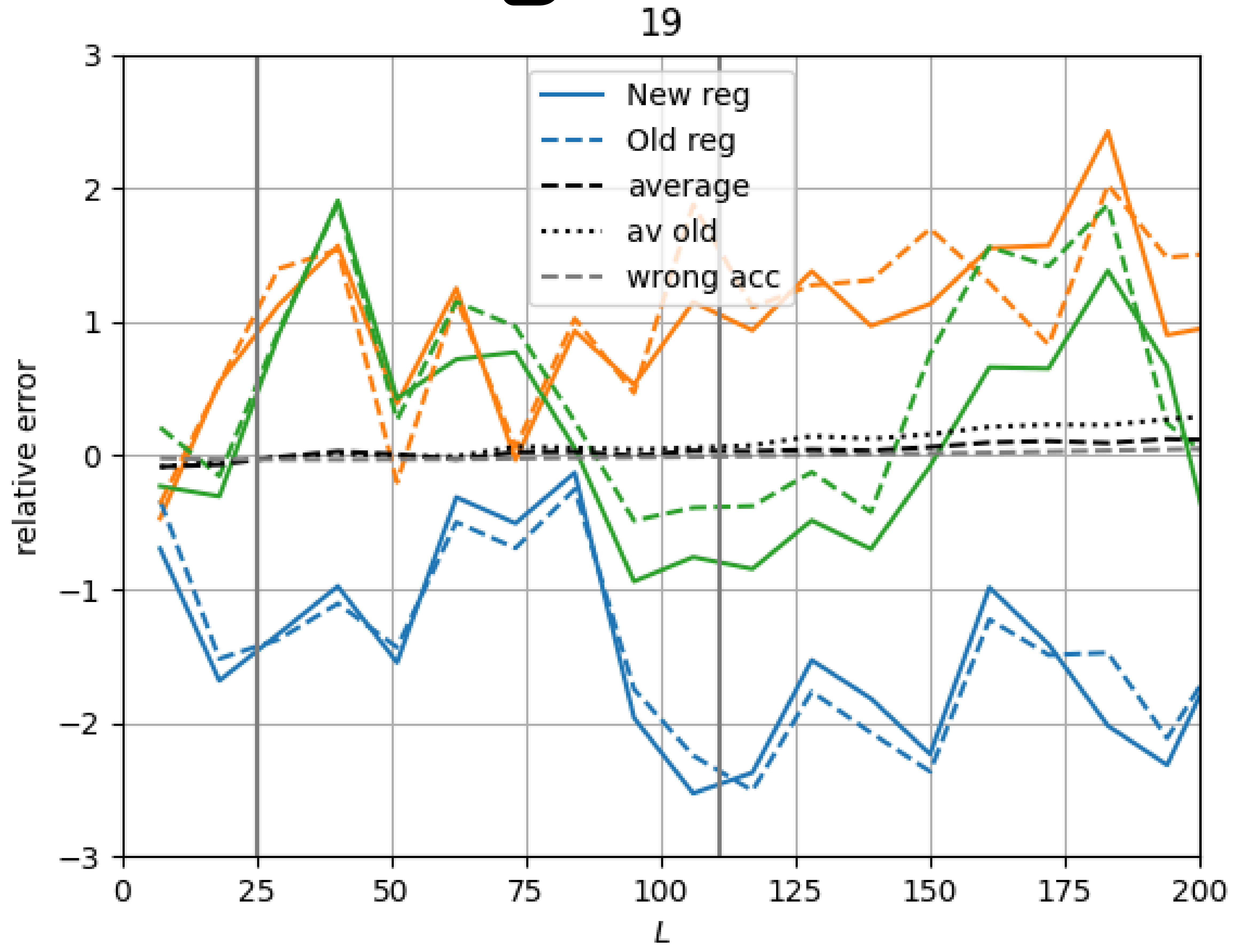
About Regularization



About Regularization

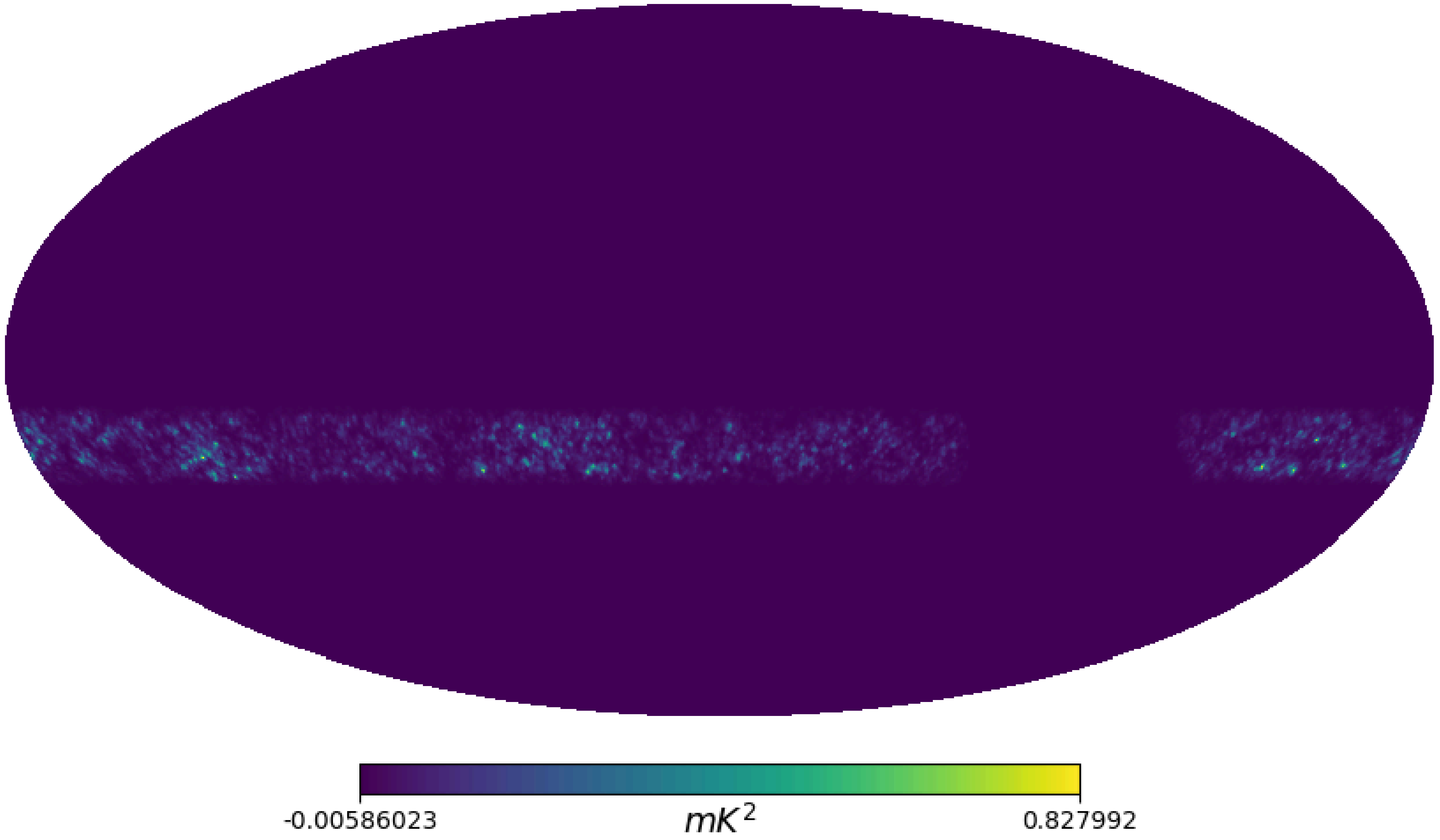


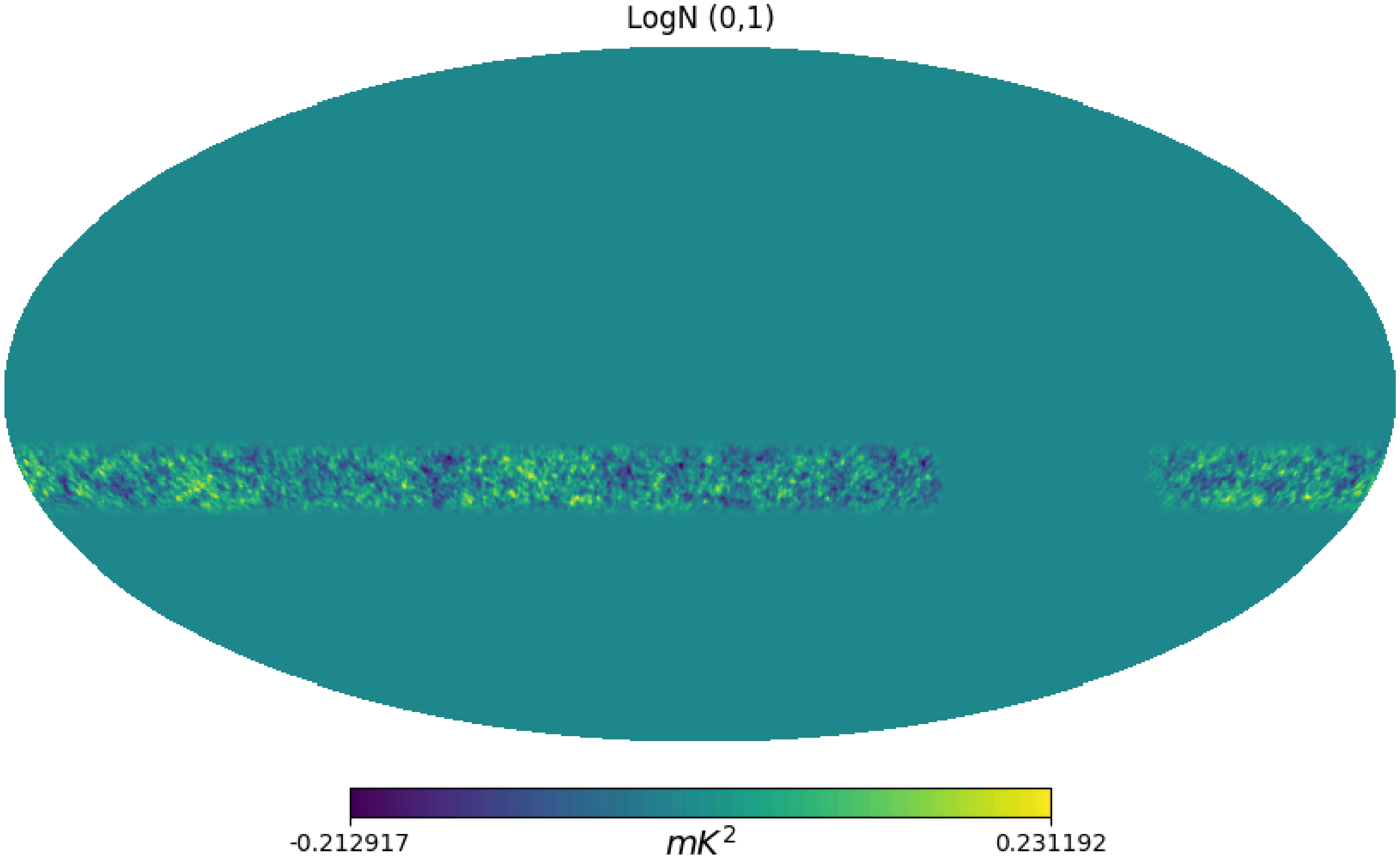
About Regularization

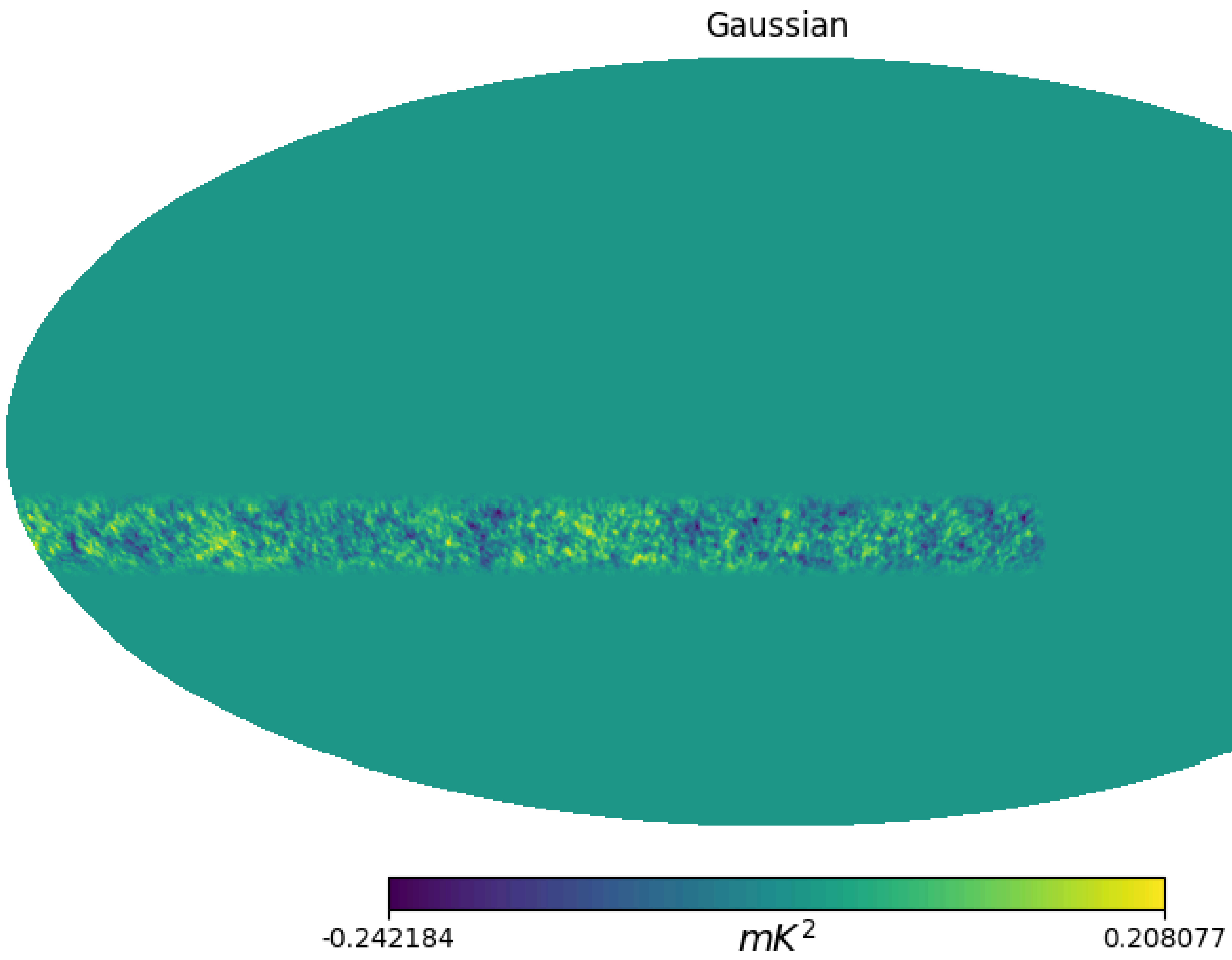


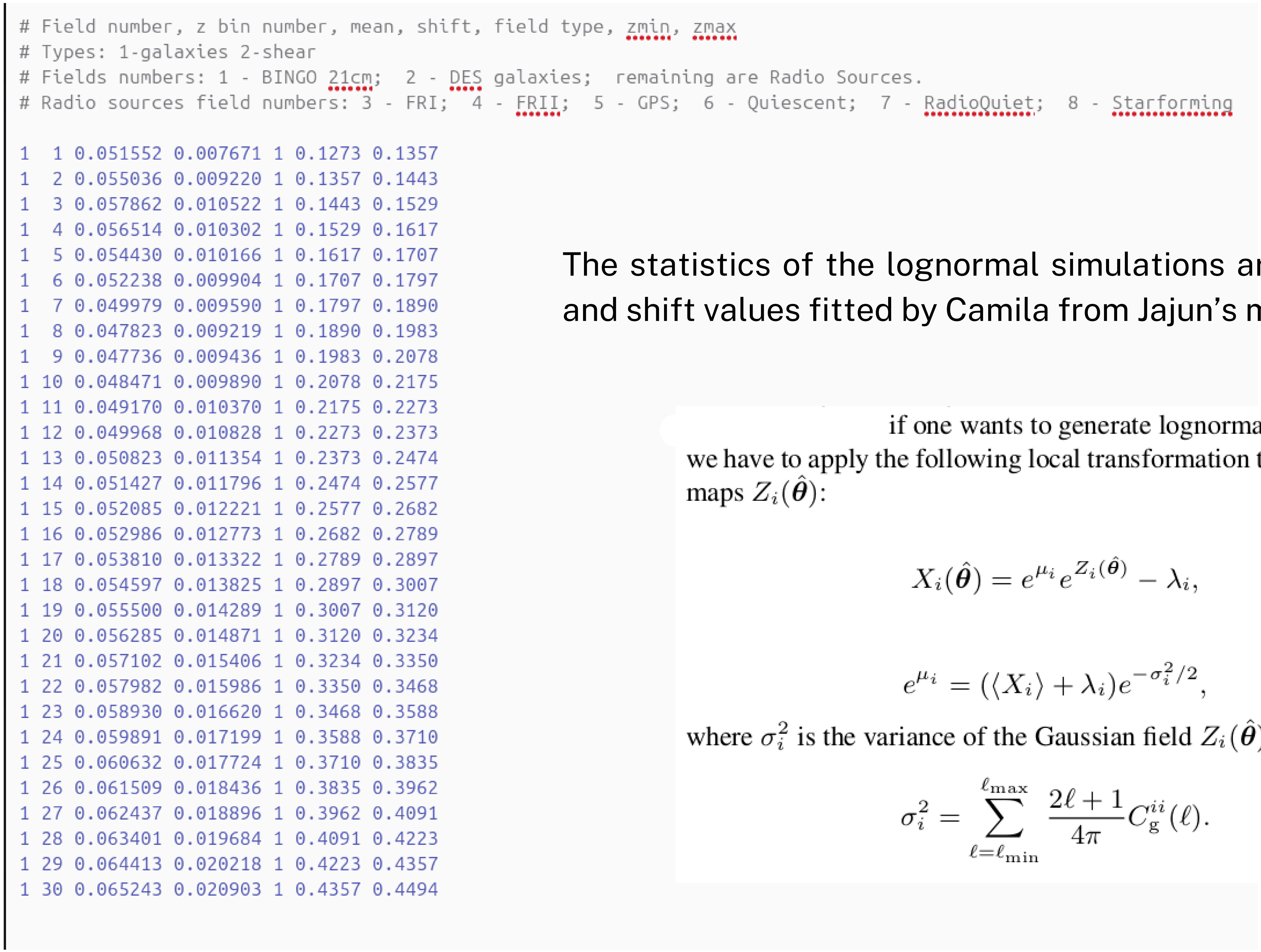
Distribution comparison

LogN Camila




LogN (0,1)
-0.212917
mK^2
0.231192

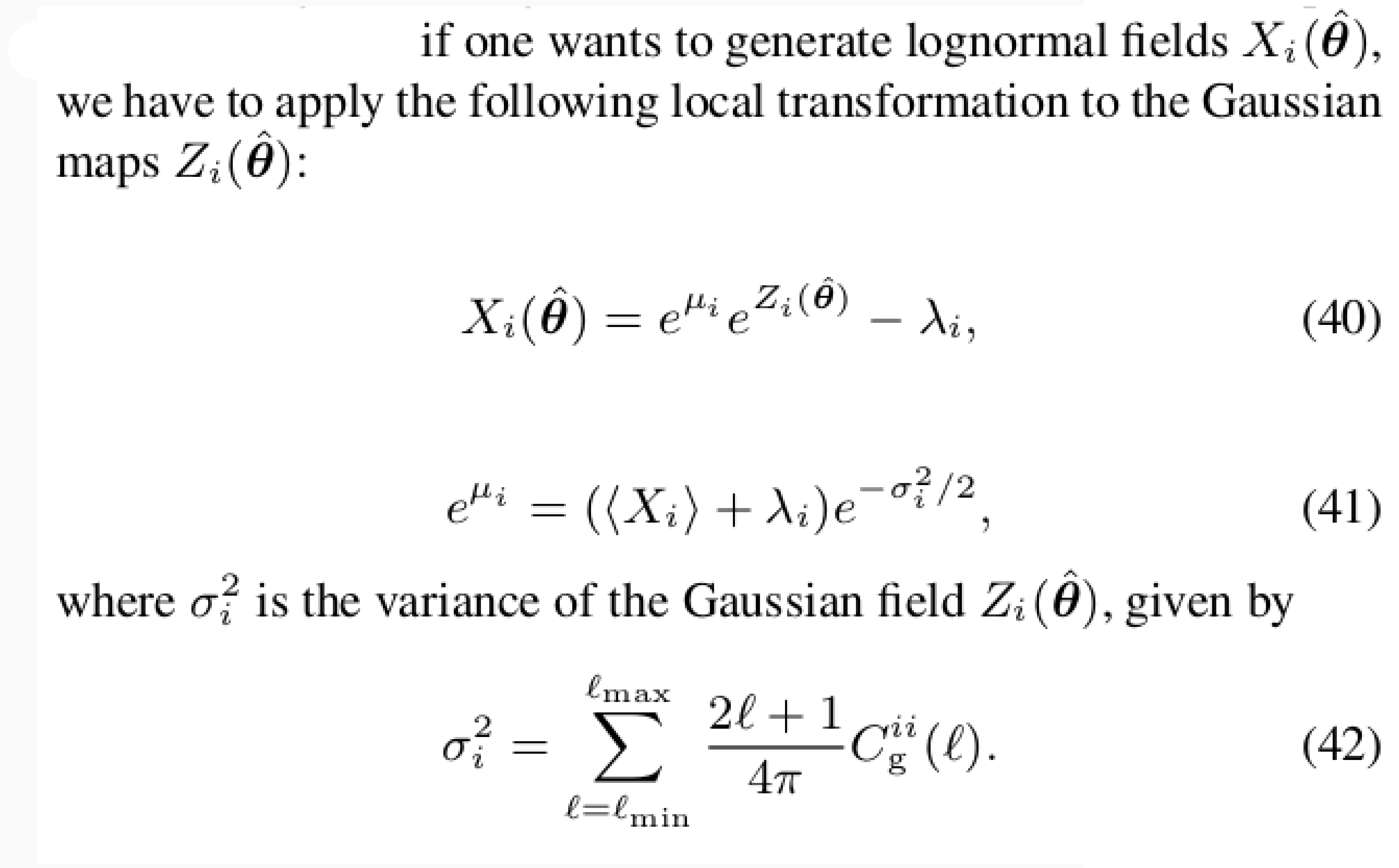

Gaussian
-0.242184
mK^2
0.208077



```
# Field number, z bin number, mean, shift, field type, zmin, zmax
# Types: 1-galaxies 2-shear
# Fields numbers: 1 - BINGO 21cm;  2 - DES galaxies;  remaining are Radio Sources.
# Radio sources field numbers: 3 - FRI;  4 - FRII;  5 - GPS;  6 - Quiescent;  7 - RadioQuiet;  8 - Starforming

1  1 0.051552 0.007671 1 0.1273 0.1357
1  2 0.055036 0.009220 1 0.1357 0.1443
1  3 0.057862 0.010522 1 0.1443 0.1529
1  4 0.056514 0.010302 1 0.1529 0.1617
1  5 0.054430 0.010166 1 0.1617 0.1707
1  6 0.052238 0.009904 1 0.1707 0.1797
1  7 0.049979 0.009590 1 0.1797 0.1890
1  8 0.047823 0.009219 1 0.1890 0.1983
1  9 0.047736 0.009436 1 0.1983 0.2078
1 10 0.048471 0.009890 1 0.2078 0.2175
1 11 0.049170 0.010370 1 0.2175 0.2273
1 12 0.049968 0.010828 1 0.2273 0.2373
1 13 0.050823 0.011354 1 0.2373 0.2474
1 14 0.051427 0.011796 1 0.2474 0.2577
1 15 0.052085 0.012221 1 0.2577 0.2682
1 16 0.052986 0.012773 1 0.2682 0.2789
1 17 0.053810 0.013322 1 0.2789 0.2897
1 18 0.054597 0.013825 1 0.2897 0.3007
1 19 0.055500 0.014289 1 0.3007 0.3120
1 20 0.056285 0.014871 1 0.3120 0.3234
1 21 0.057102 0.015406 1 0.3234 0.3350
1 22 0.057982 0.015986 1 0.3350 0.3468
1 23 0.058930 0.016620 1 0.3468 0.3588
1 24 0.059891 0.017199 1 0.3588 0.3710
1 25 0.060632 0.017724 1 0.3710 0.3835
1 26 0.061509 0.018436 1 0.3835 0.3962
1 27 0.062437 0.018896 1 0.3962 0.4091
1 28 0.063401 0.019684 1 0.4091 0.4223
1 29 0.064413 0.020218 1 0.4223 0.4357
1 30 0.065243 0.020903 1 0.4357 0.4494
```

The statistics of the lognormal simulations are the mean and shift values fitted by Camila from Jajun's mocks.



if one wants to generate lognormal fields $X_i(\hat{\boldsymbol{\theta}})$, we have to apply the following local transformation to the Gaussian maps $Z_i(\hat{\boldsymbol{\theta}})$:

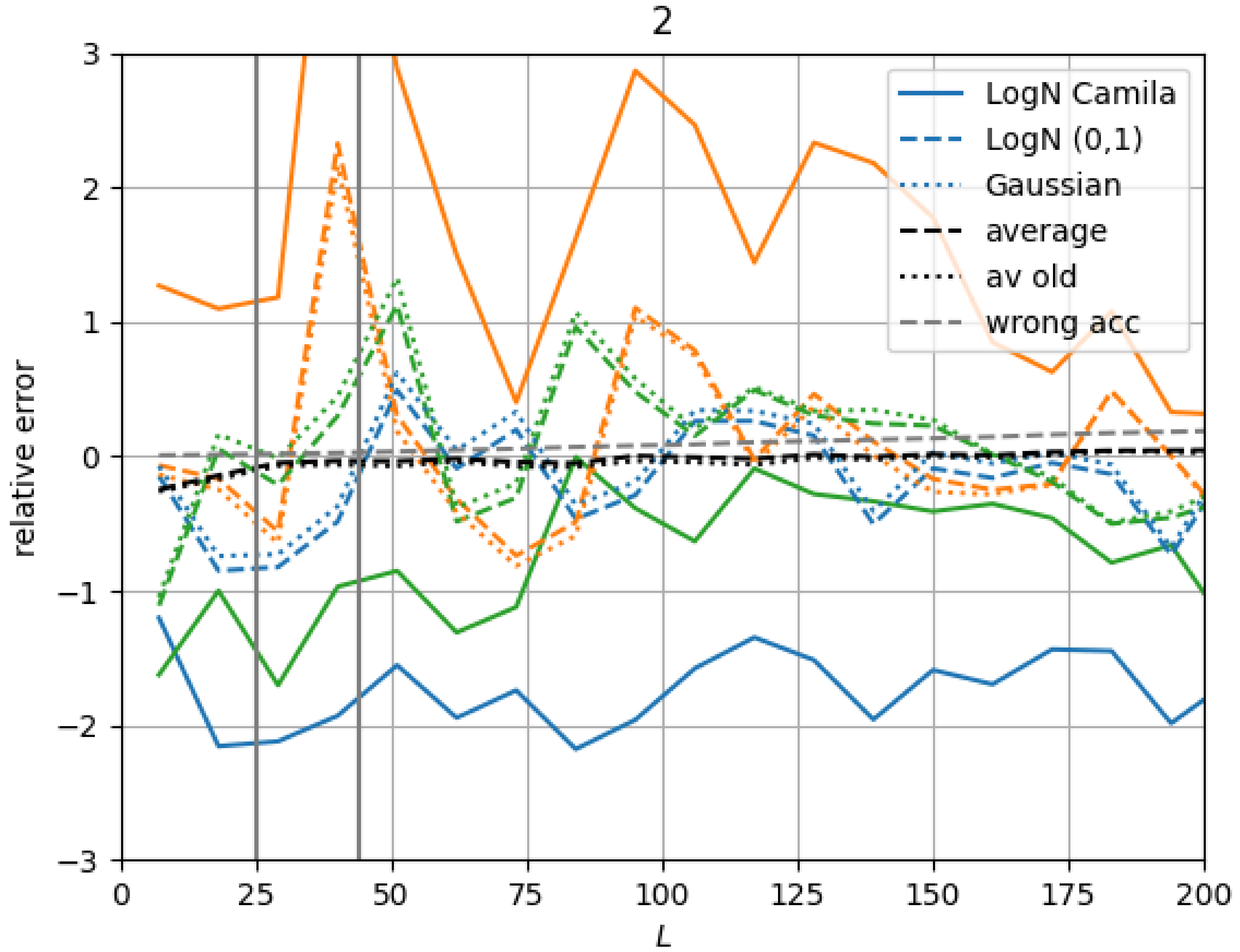
$$X_i(\hat{\boldsymbol{\theta}}) = e^{\mu_i} e^{Z_i(\hat{\boldsymbol{\theta}})} - \lambda_i, \tag{40}$$

$$e^{\mu_i} = (\langle X_i \rangle + \lambda_i) e^{-\sigma_i^2/2}, \tag{41}$$

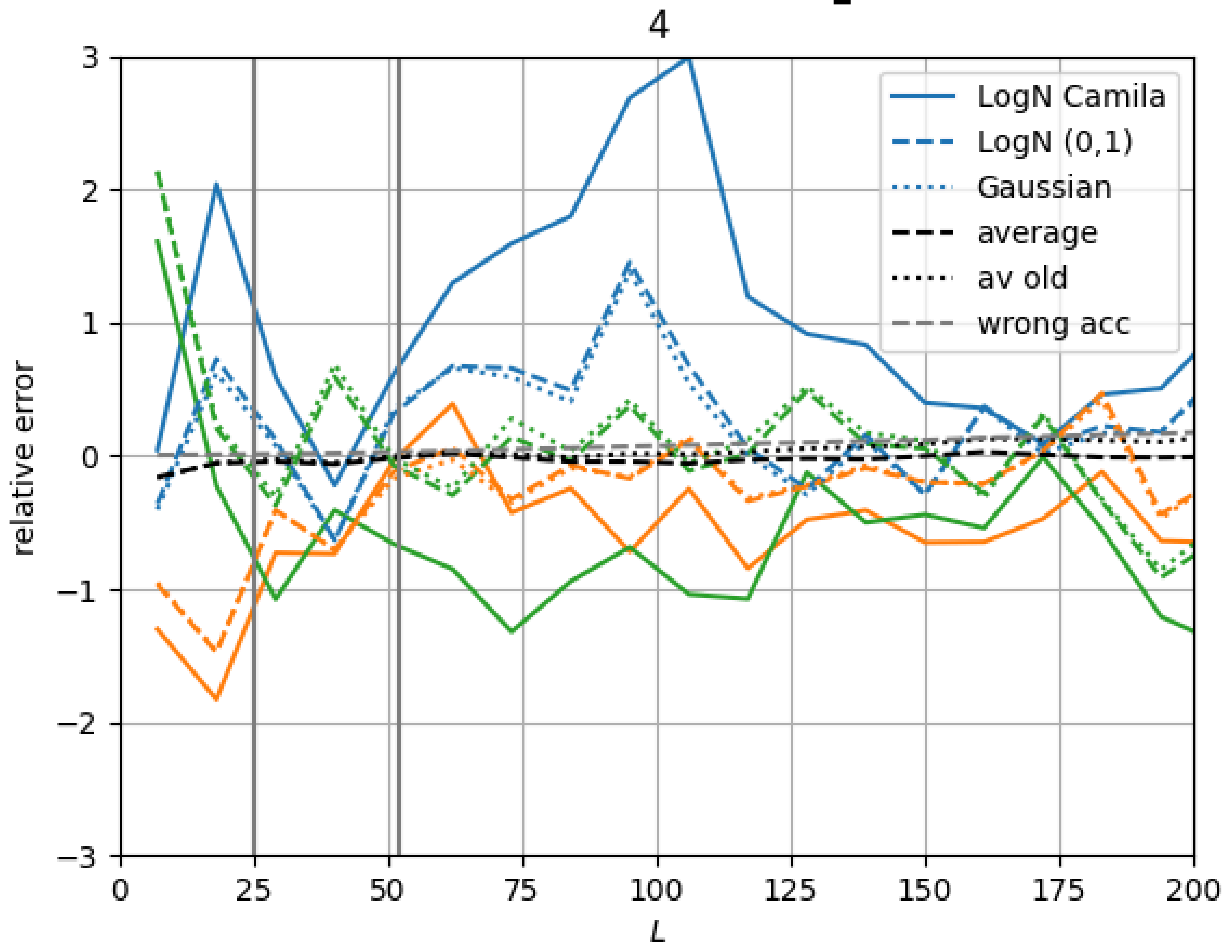
where σ_i^2 is the variance of the Gaussian field $Z_i(\hat{\boldsymbol{\theta}})$, given by

$$\sigma_i^2 = \sum_{\ell=\ell_{\min}}^{\ell_{\max}} \frac{2\ell+1}{4\pi} C_{\mathrm{g}}^{ii}(\ell). \tag{42}$$

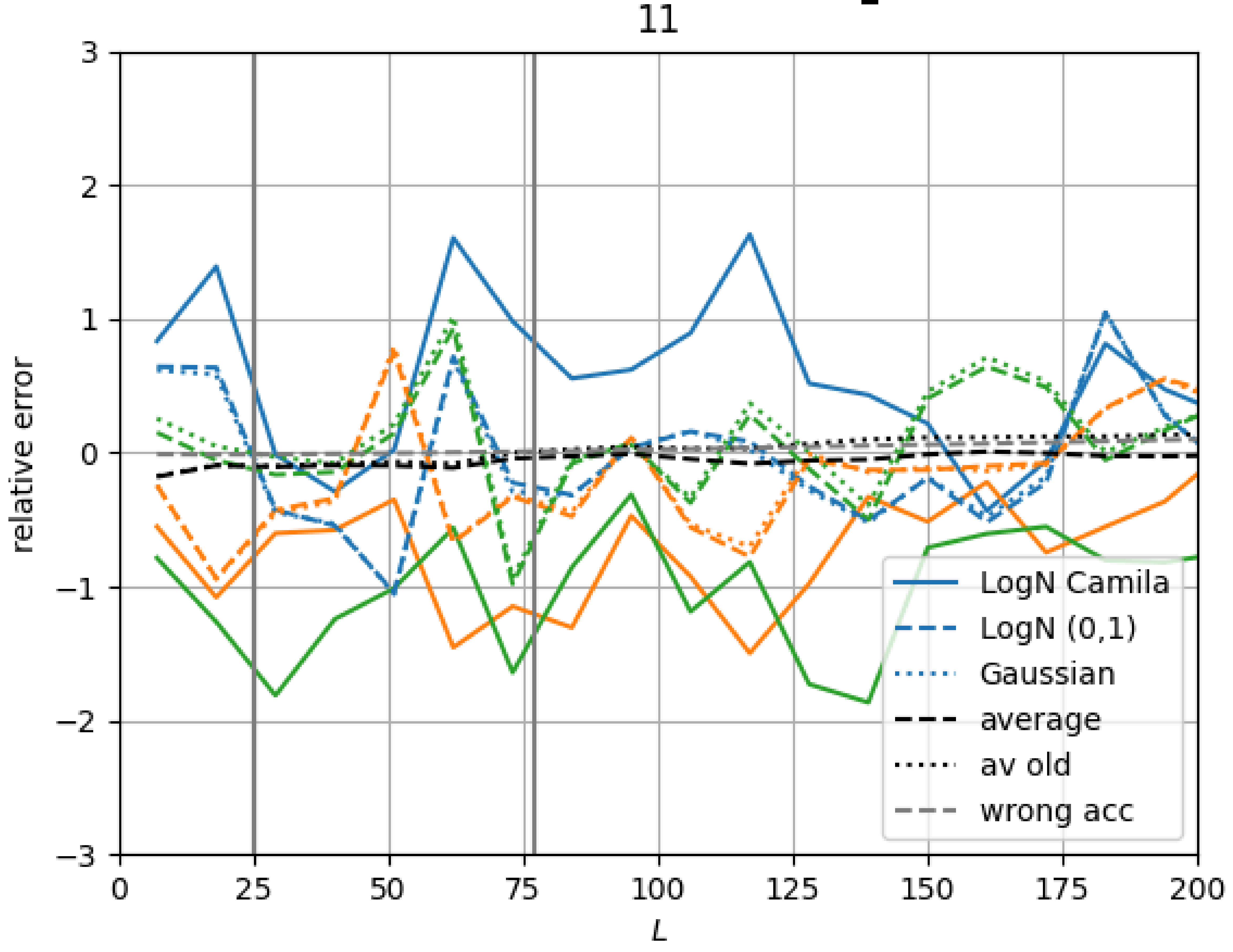
Distribution comparison



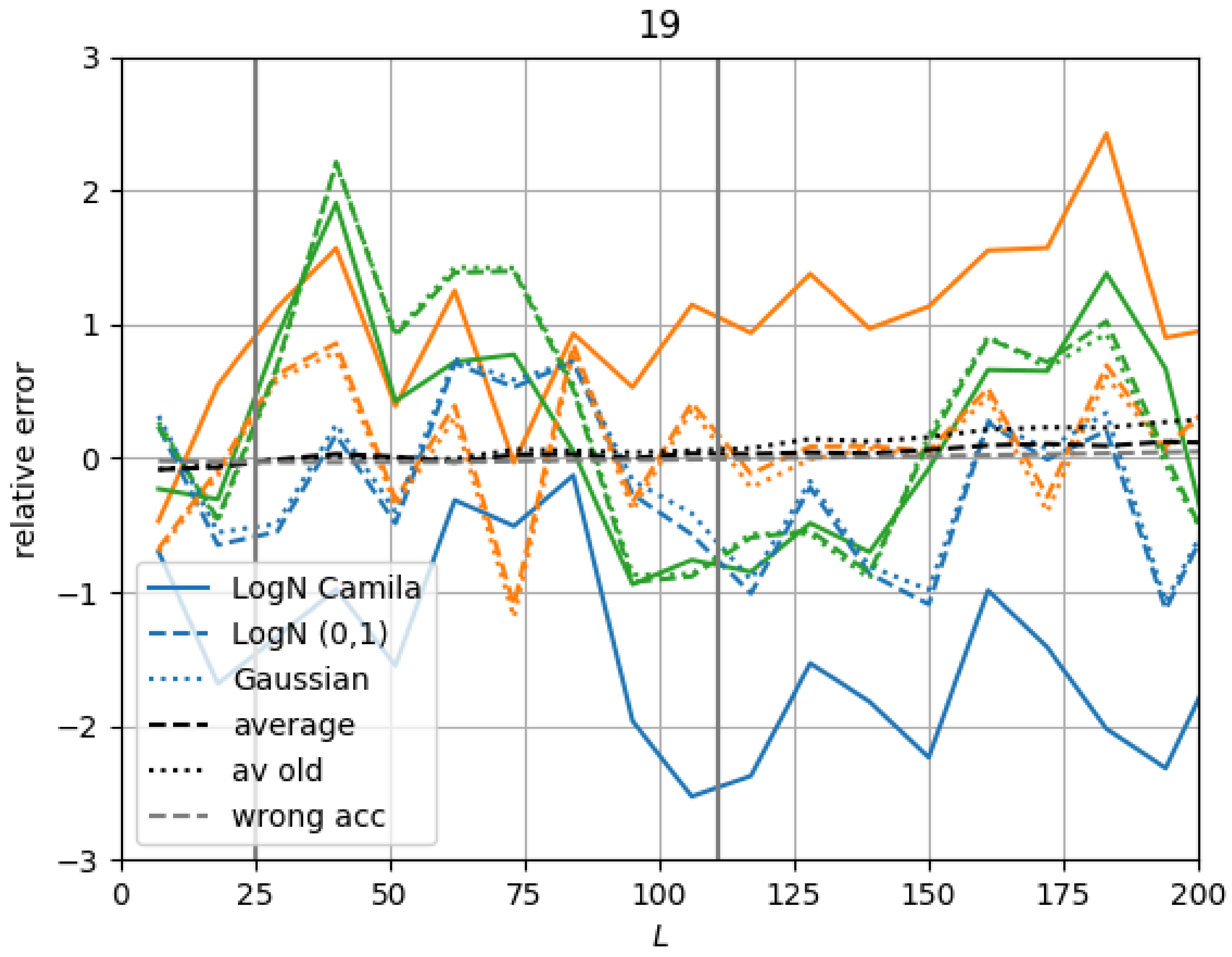
Distribution comparison



Distribution comparison



Distribution comparison



Conclusions

- We improved Flask regularization in terms of the average Cl.
- We found inconsistencies in Flask input Cl and the model being fitted, which are due to the UCLCL *zmin* and *Kvalcut* accuracy values.
- The cosmic variance errors are highly correlated and this might impact the fitting. After tests we found that this error are highly impacted by the 1-point distribution. Using another distributions fixes the issue.