An efficient parallel algorithm for estimating *higher*-order polyspectra

Donghui Jeong
(Penn State)

PTChat@Kyoto, 10 April 2019

Work done by

Prof. Juhan Kim @ KIAS   Joseph Tomlinson @ Penn State

Higher order polyspectra

- In an statistically homogeneous universe
  - Bispectrum

$$\langle \delta(\mathbf{k}_1)\delta(\mathbf{k}_2)\delta(\mathbf{k}_3)\rangle = (2\pi)^3 B(\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3)\delta^D(\mathbf{k}_1 + \mathbf{k}_2 + \mathbf{k}_3)$$

  - Trispectrum

$$\langle \delta(\mathbf{k}_1)\delta(\mathbf{k}_2)\delta(\mathbf{k}_3)\delta(\mathbf{k}_4)\rangle = (2\pi)^3 T(\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4)\delta^D(\mathbf{k}_1 + \mathbf{k}_2 + \mathbf{k}_3 + \mathbf{k}_4)$$

  - Quadspectrum, Pentaspectrum, etc

$$\langle \delta(\mathbf{k}_1)\delta(\mathbf{k}_2)\cdots\delta(\mathbf{k}_n)\rangle = (2\pi)^3 P_n(\mathbf{k}_1, \mathbf{k}_2, \cdots, \mathbf{k}_n)\delta^D(\mathbf{k}_1 + \mathbf{k}_2 + \cdots + \mathbf{k}_n)$$

Naive estimator: bispectrum

- Let's focus on the monopole, because the extension is trivial
  - From the definition:

$$\langle \delta(\mathbf{k}_1)\delta(\mathbf{k}_2)\delta(\mathbf{k}_3)\rangle = (2\pi)^3 B(\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3)\delta^D(\mathbf{k}_1 + \mathbf{k}_2 + \mathbf{k}_3)$$

  - We can estimate the bispectrum from direct sampling:

$$B(k_1, k_2, k_3) = \frac{V_f}{V^B_{(123)}(2\pi)^3}\int_{k_1} d^3q_1 \int_{k_2} d^3q_2 \delta(\mathbf{q}_1)\delta(\mathbf{q}_2)\delta(-\mathbf{q}_{12})$$

Complexity of Naive estimator

- To measure bispectrum from $(k_1,k_2,k_3) = (1,1,1)k_F$ to $(k_1,k_2,k_3) = (N_{max},N_{max},N_{max})k_F$, we need to loop over k_{1x}, k_{1y}, k_{1z}, k_{2x}, k_{2y}, k_{2z} (then, k_3 is determined from triangle condition)
- Complexity = $(N_{max})^6$

$$B(k_1,k_2,k_3) = \frac{V_f}{V^B_{(123)}(2\pi)^3}\int_{k_1} d^3q_1 \int_{k_2} d^3q_2 \delta(\mathbf{q}_1)\delta(\mathbf{q}_2)\delta(-\mathbf{q}_{12})$$

Roman’s estimator

$$\begin{aligned}
B(k_1,k_2,k_3) =& \frac{V_f}{V^B_{(123)}(2\pi)^3}\int_{k_1} d^3q_1 \int_{k_2} d^3q_2 \int_{k_3} d^3q_3 \delta(\mathbf{q}_1)\delta(\mathbf{q}_2)\delta(\mathbf{q}_3)\delta_D(\mathbf{q}_{123}) \\
=& \frac{V_f}{V^B_{(123)}(2\pi)^3}\int_{k_1} d^3q_1 \int_{k_2} d^3q_2 \int_{k_3} d^3q_3 \delta(\mathbf{q}_1)\delta(\mathbf{q}_2)\delta(\mathbf{q}_3) \int \frac{d^3x}{(2\pi)^3} e^{i\mathbf{x}\cdot\mathbf{q}_{123}} \\
=& \frac{V_f}{V^B_{(123)}(2\pi)^3}\int \frac{d^3x}{(2\pi)^3} \left(\int_{k_1} d^3q_1 \delta(\mathbf{q}_1) e^{i\mathbf{x}\cdot\mathbf{q}_1}\right)\left(\int_{k_2} d^3q_2 \delta(\mathbf{q}_2) e^{i\mathbf{x}\cdot\mathbf{q}_2}\right)\left(\int_{k_3} d^3q_3 \delta(\mathbf{q}_3) e^{i\mathbf{x}\cdot\mathbf{q}_3}\right) \\
=& \frac{V_f}{V^B_{(123)}(2\pi)^3}\int \frac{d^3x}{(2\pi)^3} I_{k_1}(\mathbf{x}) I_{k_2}(\mathbf{x}) I_{k_3}(\mathbf{x})
\end{aligned}$$

$$I_{k_i}(\mathbf{x}) = \int_{k_i} d^3q_1 \delta(\mathbf{q}) e^{i\mathbf{x}\cdot\mathbf{q}} = \int d^3q_1 \tilde{I}_{k_i}(\mathbf{q}) e^{i\mathbf{x}\cdot\mathbf{q}}$$

Complexity of Roman's estimator

- To measure bispectrum
  from $(k_1,k_2,k_3) = (1,1,1)k_F$ to $(k_1,k_2,k_3) = (N_{max},N_{max},N_{max})k_F$,
  we need to loop over
  k_1, k_2, k_3
  and need to calculate the inner product of 3D matrices

- Complexity = $(N_{max})^6$

$$B(k_1, k_2, k_3) = \frac{V_f}{V^B_{(123)}(2\pi)^3} \int \frac{d^3x}{(2\pi)^3} I_{k_1}(\mathbf{x}) I_{k_2}(\mathbf{x}) I_{k_3}(\mathbf{x})$$

Yet, Roman's estimator is much faster!

- Loop is only for k_1, k_2, k_3 : much fewer computation
- Matrix inner product is much faster than irregular sampling of the matrix

$$B(k_1, k_2, k_3) = \frac{V_f}{V^B_{(123)}(2\pi)^3} \int_{k_1} d^3q_1 \int_{k_2} d^3q_2 \delta(\mathbf{q}_1)\delta(\mathbf{q}_2)\delta(-\mathbf{q}_{12})$$

$$B(k_1, k_2, k_3) = \frac{V_f}{V^B_{(123)}(2\pi)^3} \int \frac{d^3x}{(2\pi)^3} I_{k_1}(\mathbf{x}) I_{k_2}(\mathbf{x}) I_{k_3}(\mathbf{x})$$

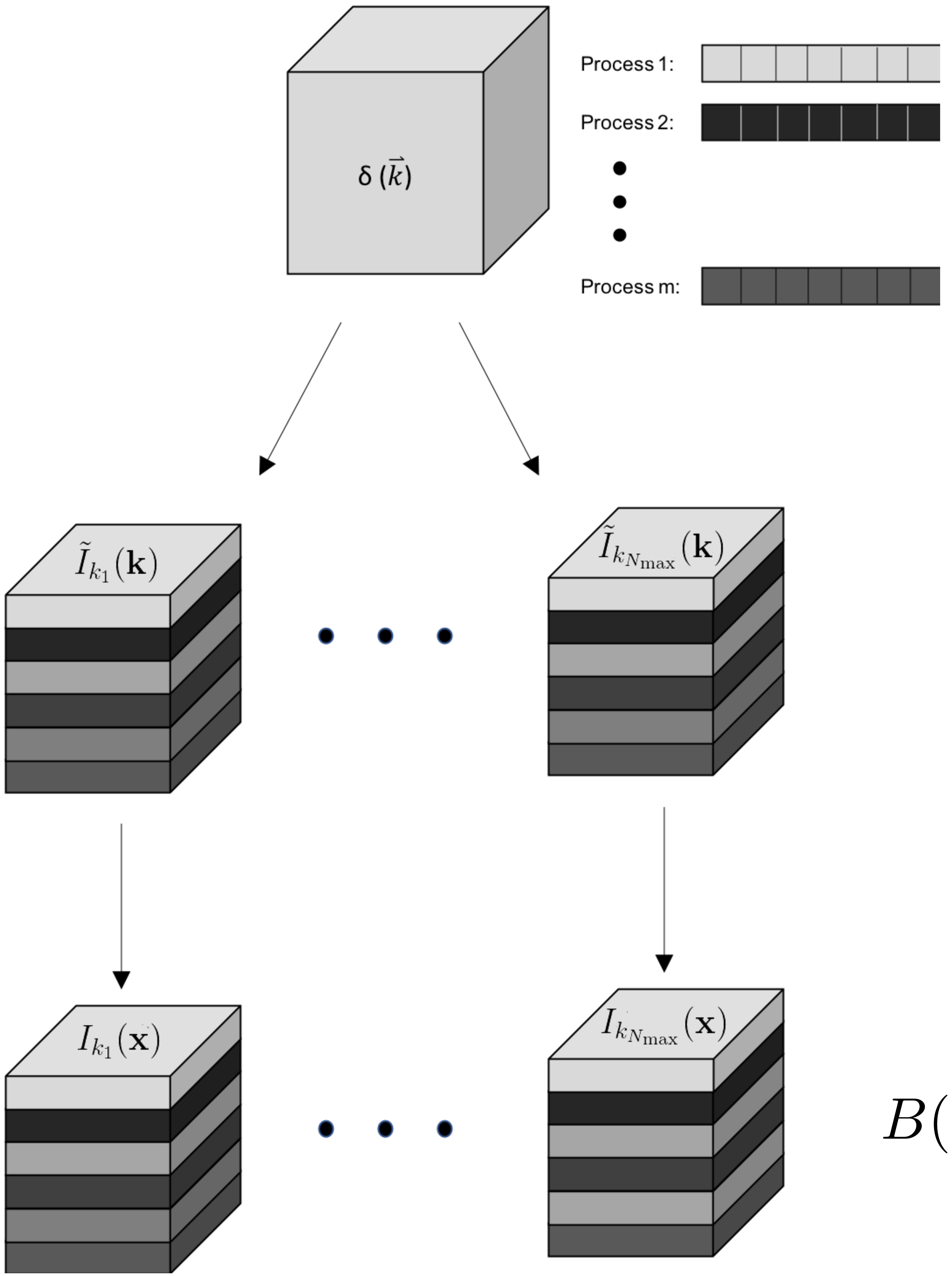
Memory requirement

- To get an unbiased bispectrum,
  $N_{mesh} > 3\ N_{max}$
  for each $I_{ki}(x)$, and we need N_{max} of them.
- We therefore need memory space for at least
  $N_{max}\,(N_{mesh})^3 > 27\ (N_{max})^4$
  numbers
- With single precision (Float32), already 27 GB for N_{max}=128.

Naive parallelization of the estimator

- Naive parallelization : Run each $I_{ki}(x)$ on one CPU
- Why bad?
  We need to pull out the full 3D array to calculate the inner product

$$B(k_1, k_2, k_3) = \frac{V_f}{V^B_{(123)}(2\pi)^3} \int \frac{d^3x}{(2\pi)^3} I_{k_1}(\mathbf{x}) I_{k_2}(\mathbf{x}) I_{k_3}(\mathbf{x})$$

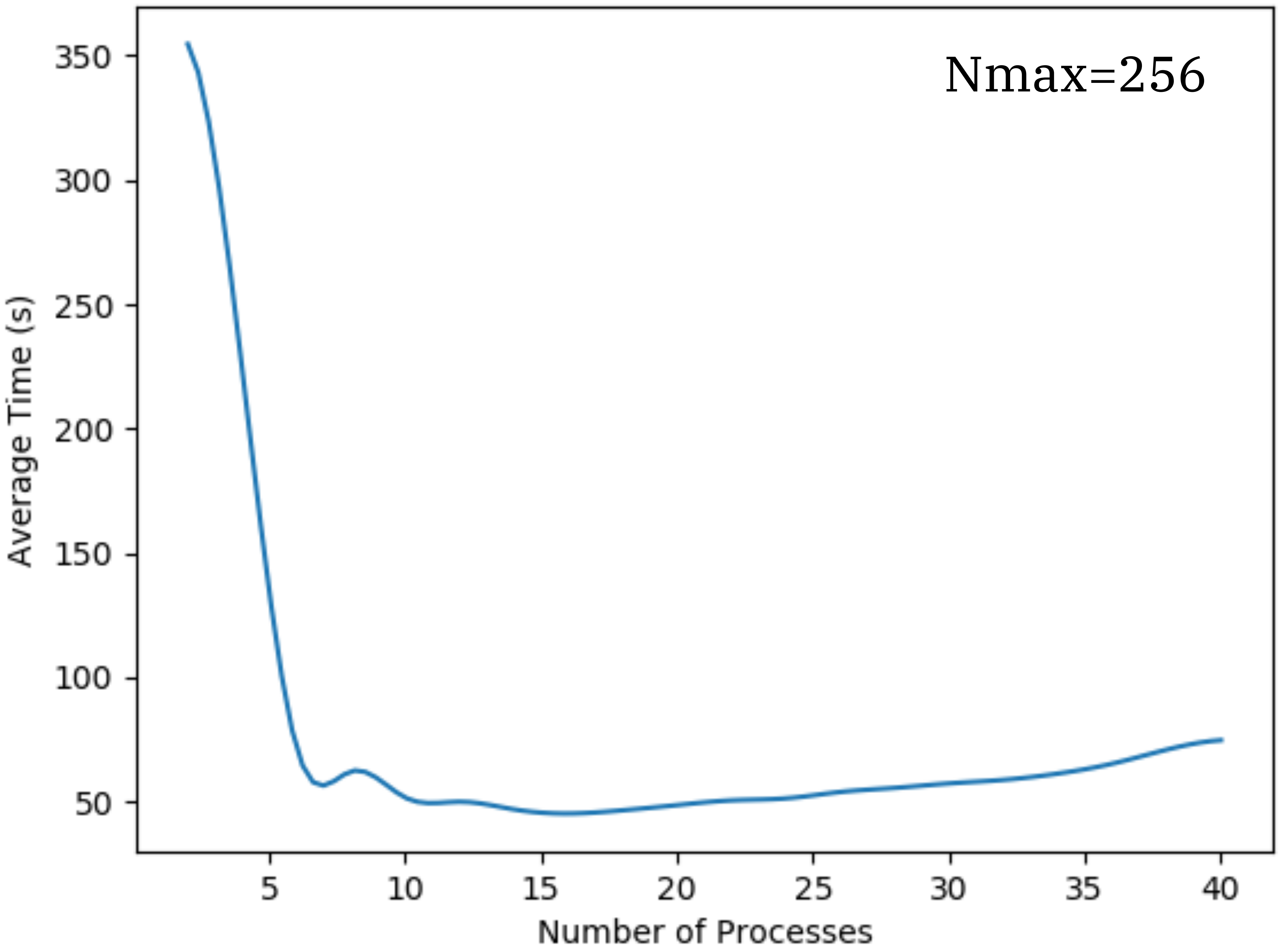


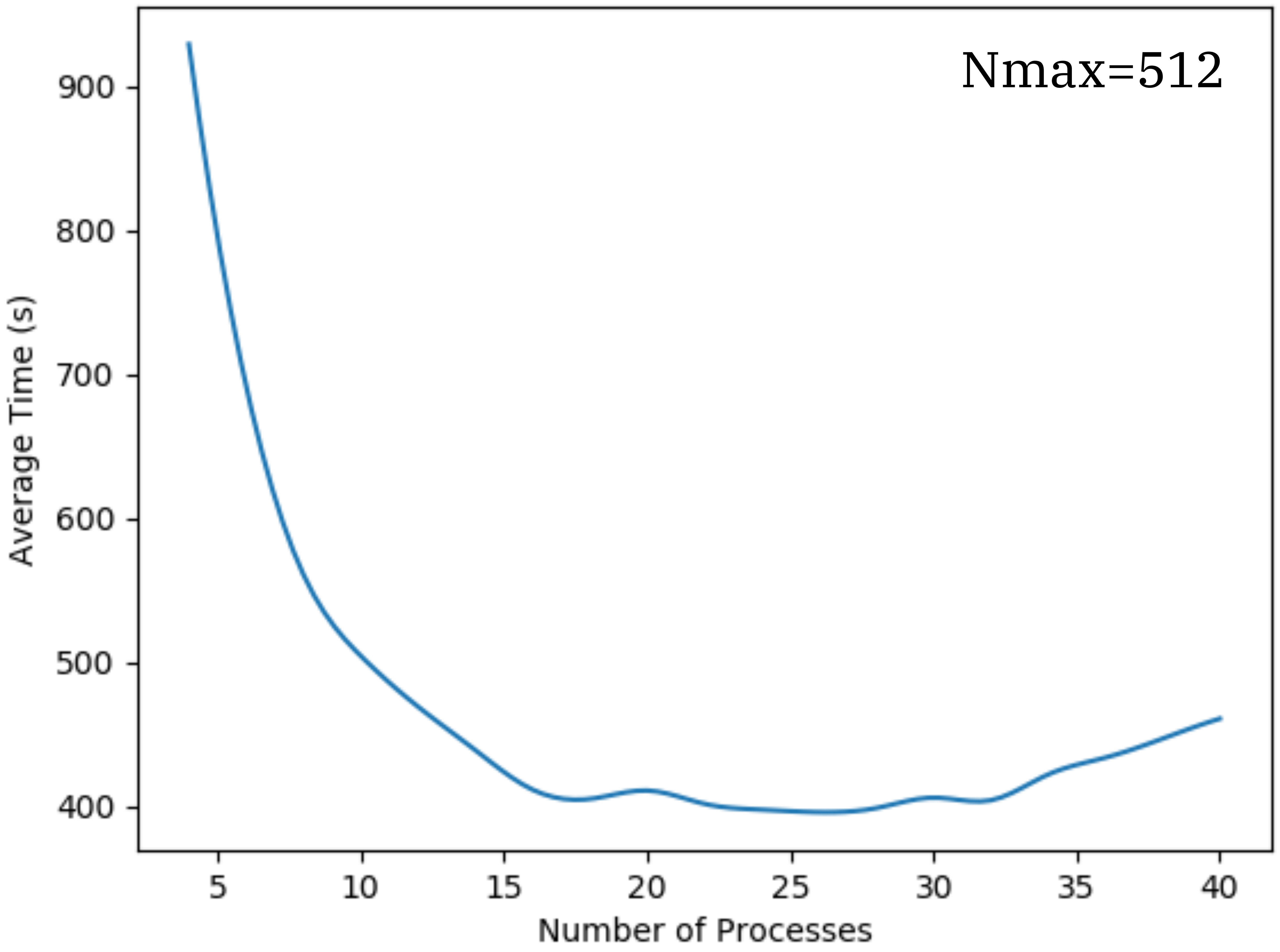
Efficient parallelization

- Multiplication only done locally!
- Minimize the interCPU communication:
  - When FFT the last dimension
  - When reducing the sum

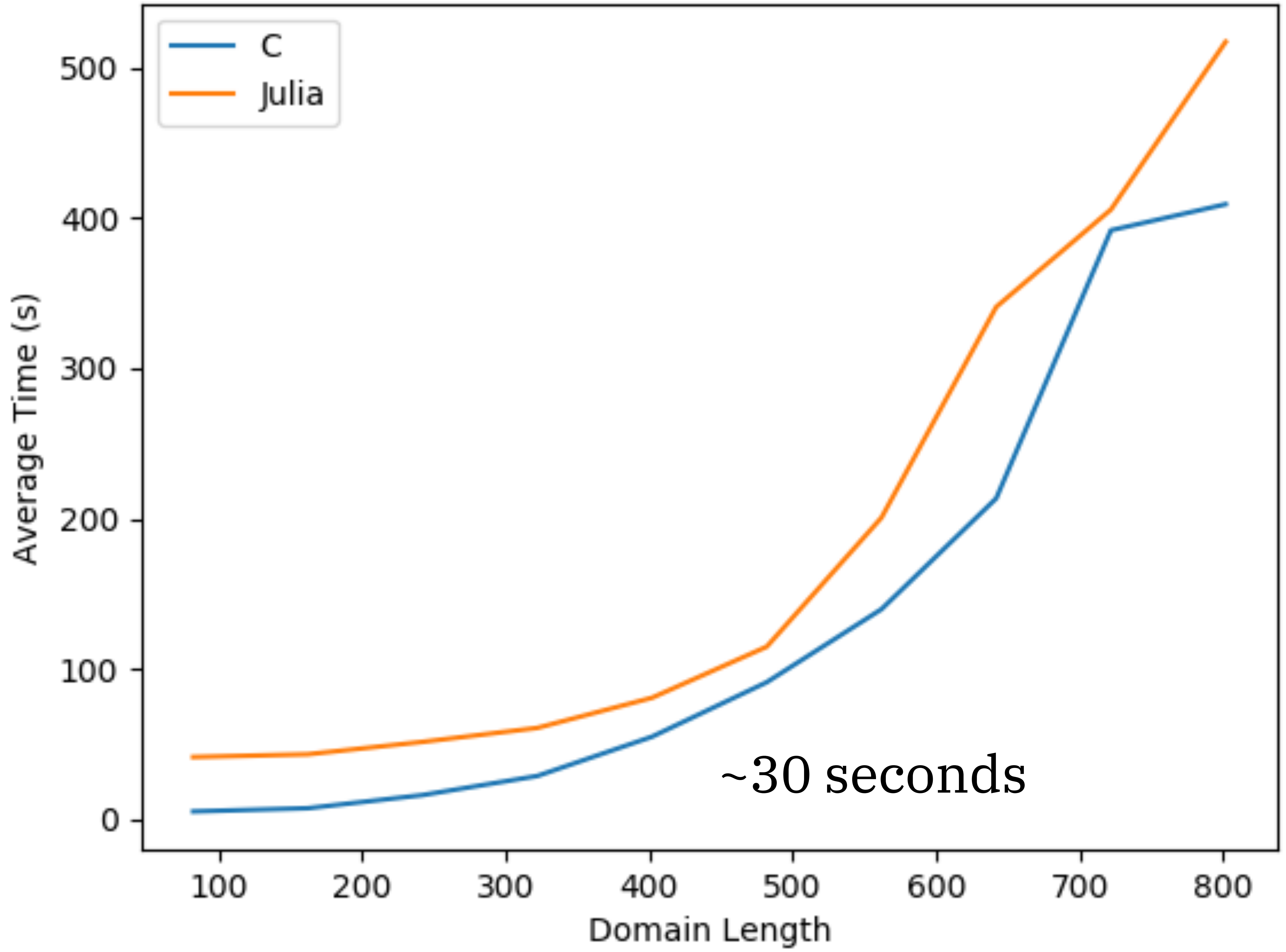
$$B(k_1, k_2, k_3) = \frac{V_f}{V^B_{(123)}(2\pi)^3} \int \frac{d^3x}{(2\pi)^3} I_{k_1}(\mathbf{x}) I_{k_2}(\mathbf{x}) I_{k_3}(\mathbf{x})$$

Efficient parallelization, result





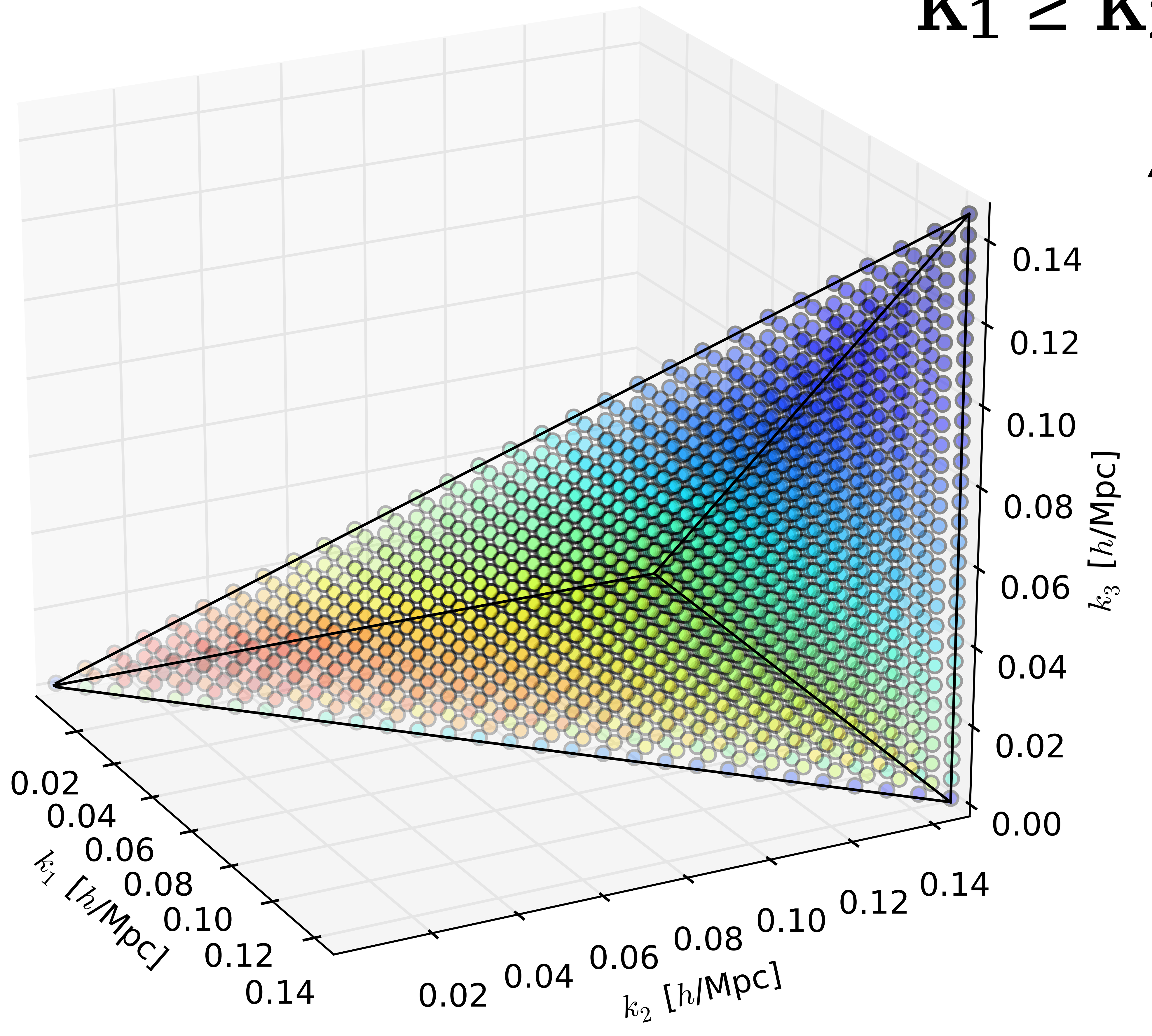
C vs. Julia

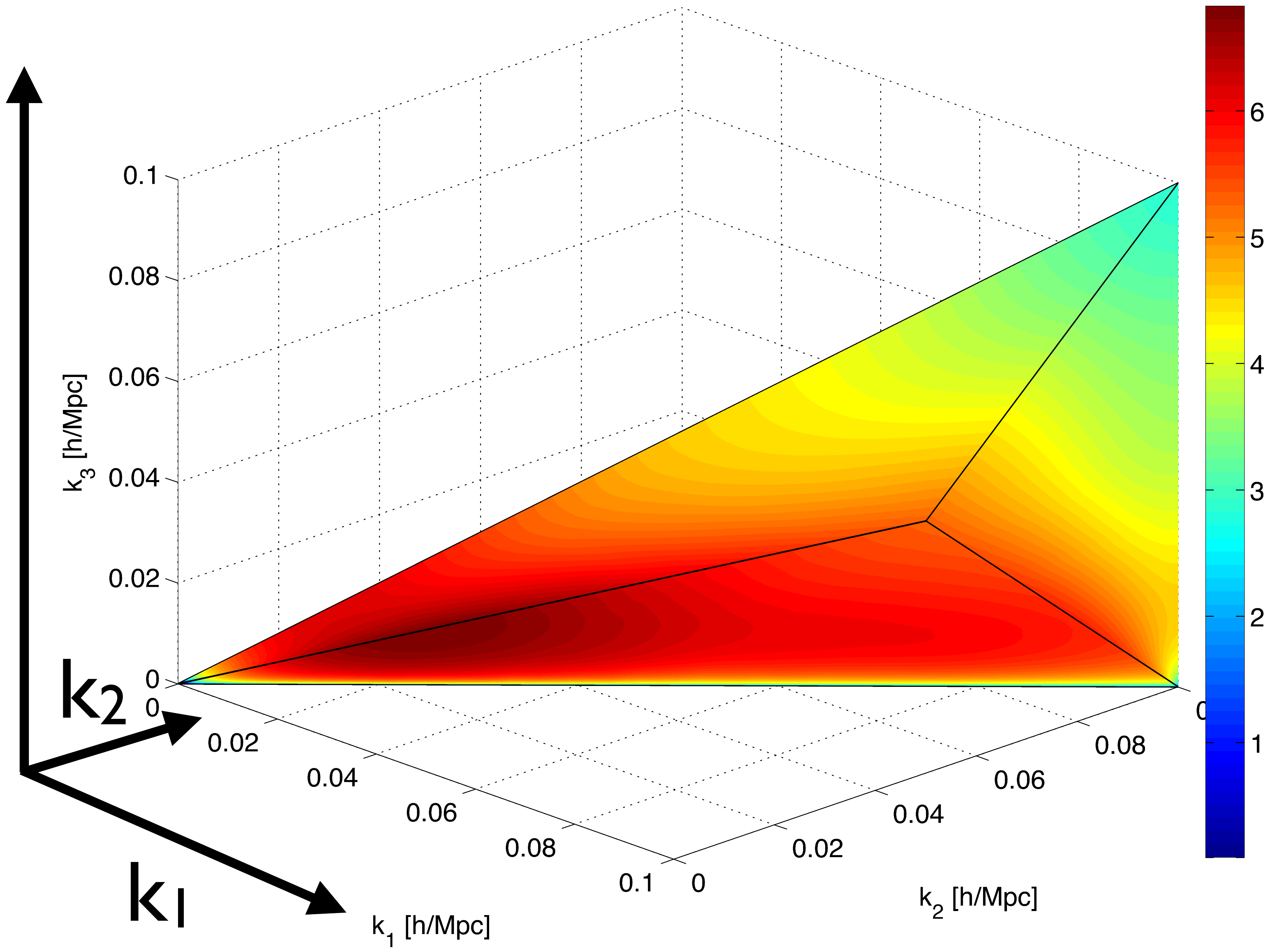


C
Julia
500
400
300
200
100
0
Average Time (s)
100
200
300
400
500
600
700
800
Domain Length
~30 seconds

Visualizing bispectrum

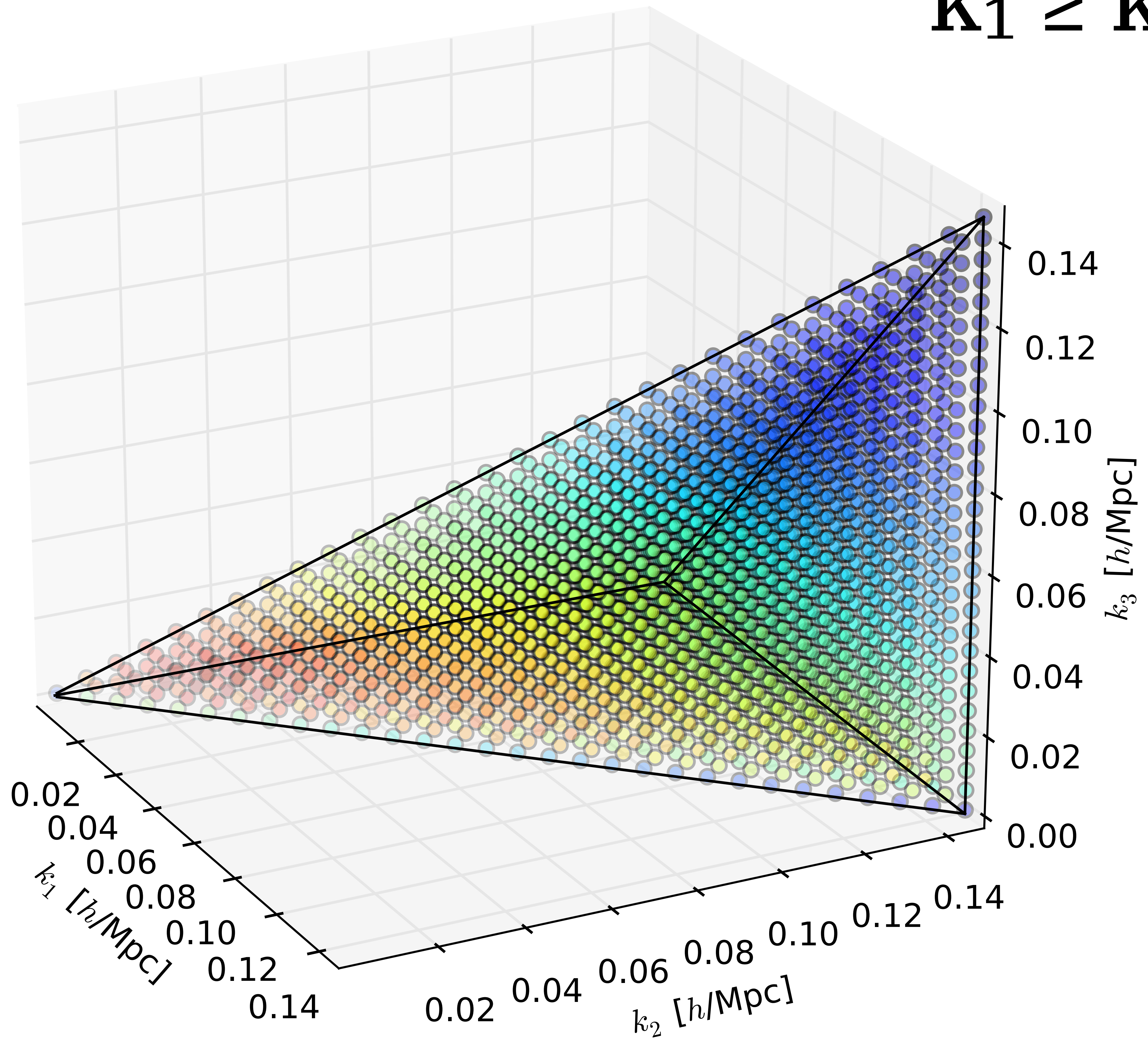
$k_1 \geq k_2 \geq k_3$

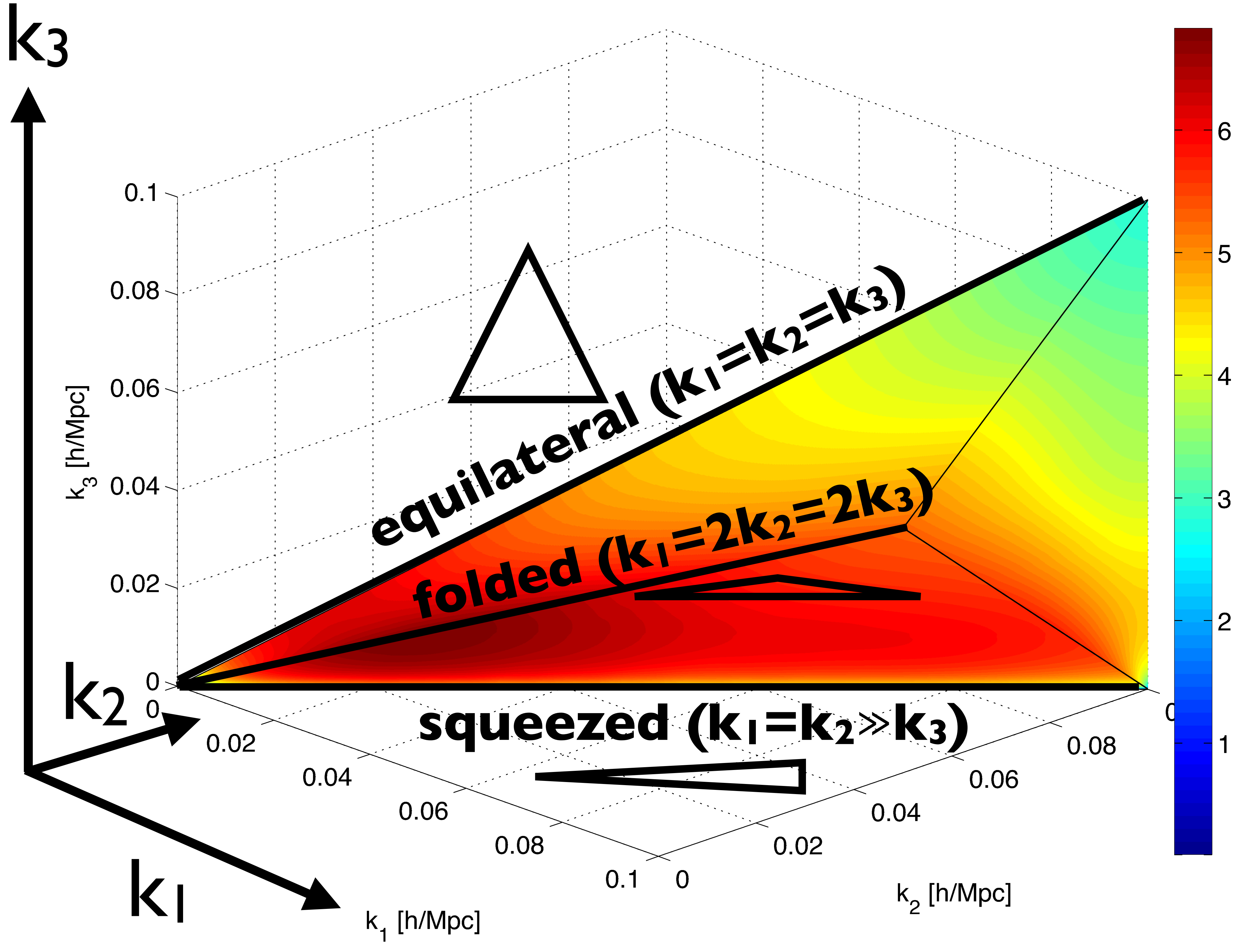




Visualizing bispectrum

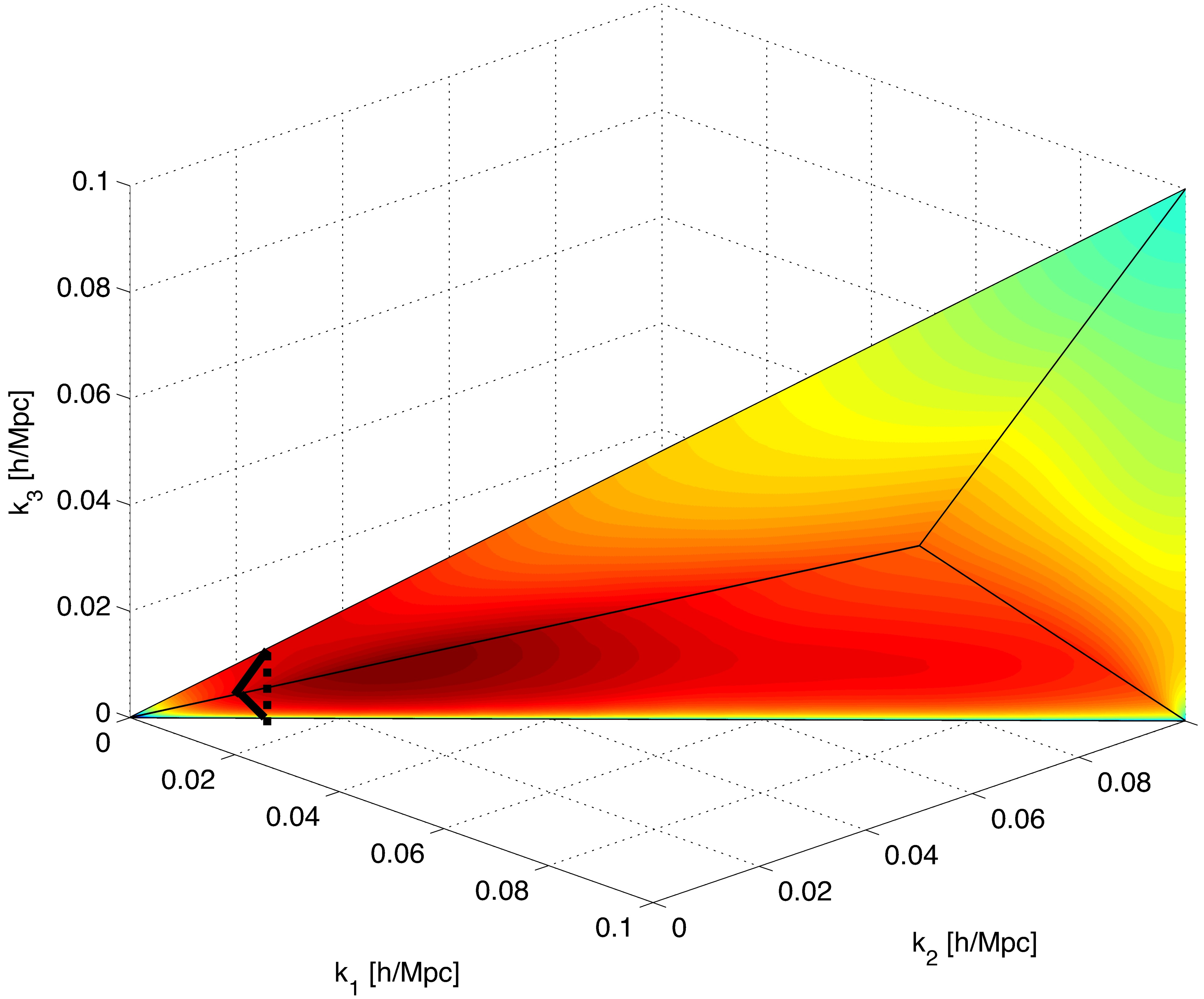
$k_1 \geq k_2 \geq k_3$

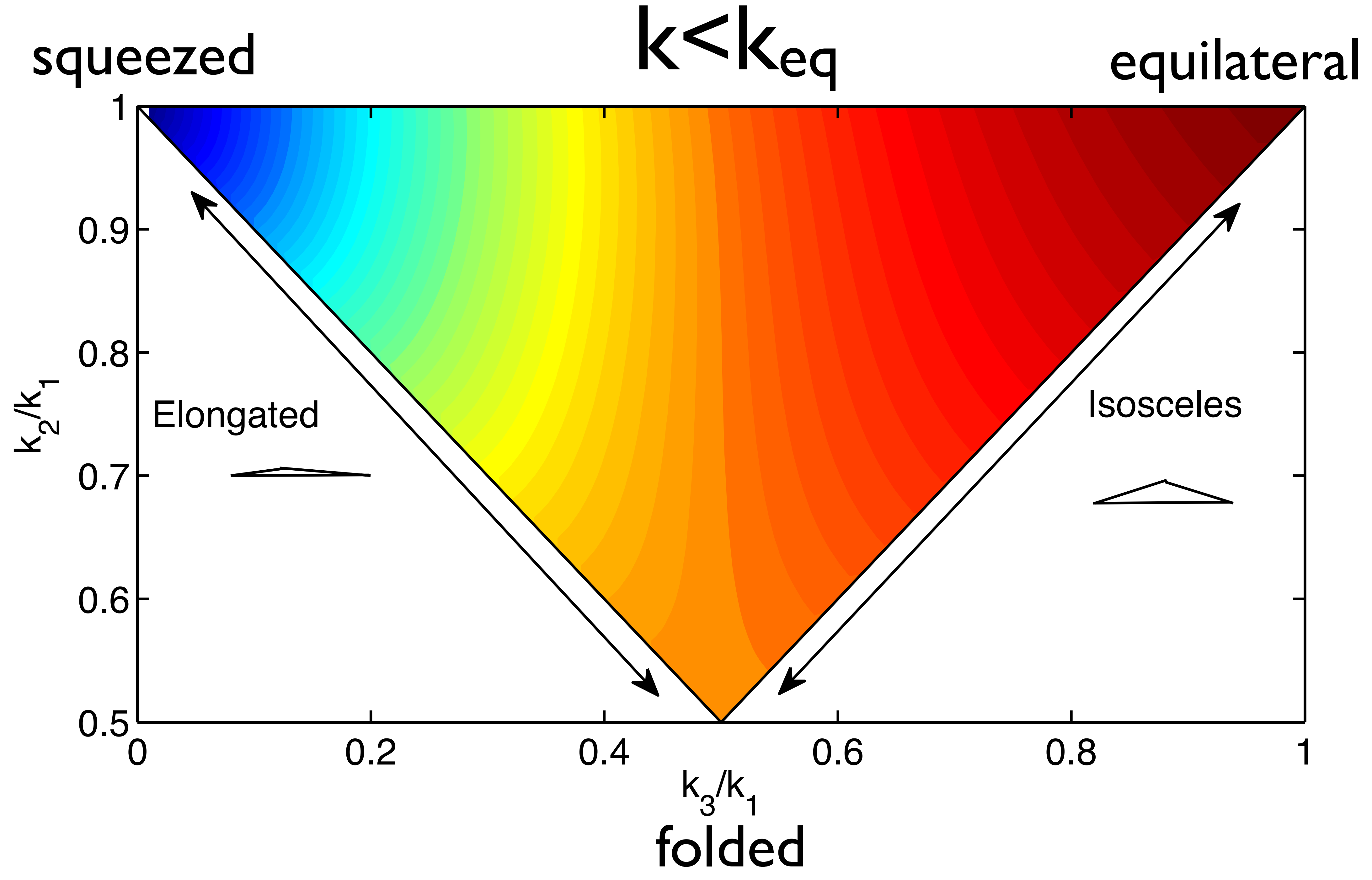




Visualizing bispectrum

$k_1 \geq k_2 \geq k_3$



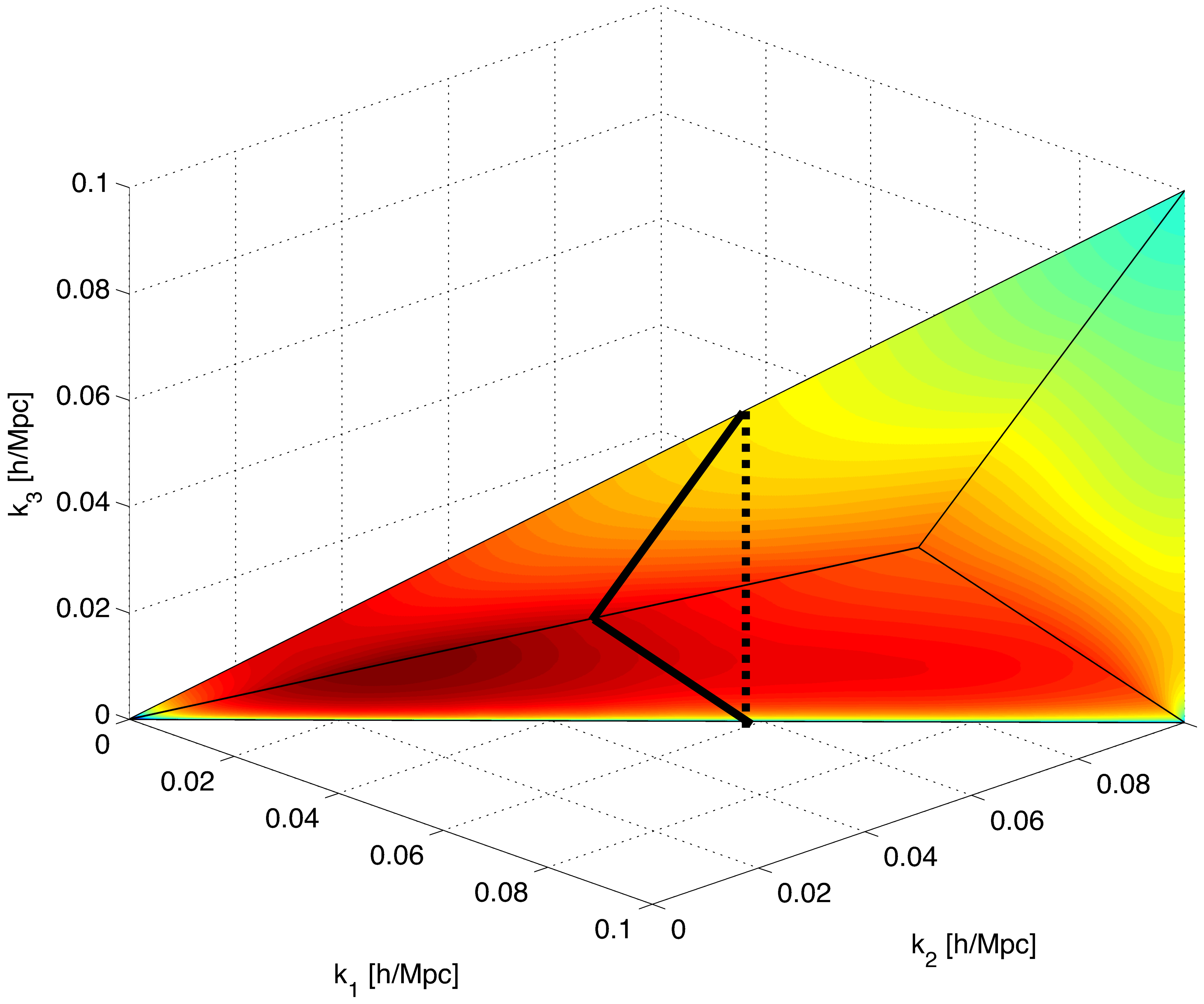


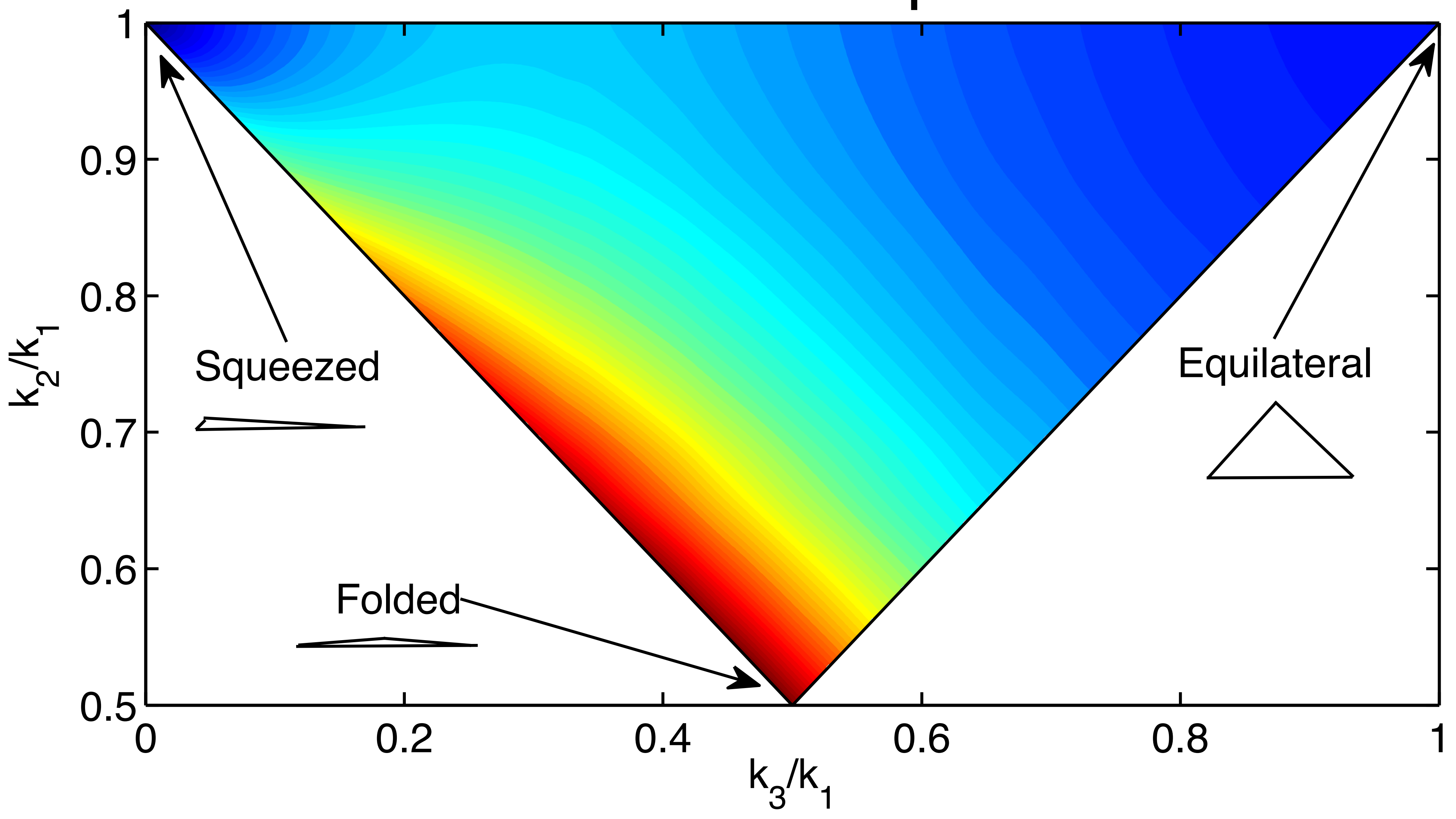
The slope of power spectrum

dlnP/dlnk > 0

Visualizing bispectrum

$k_1 \geq k_2 \geq k_3$



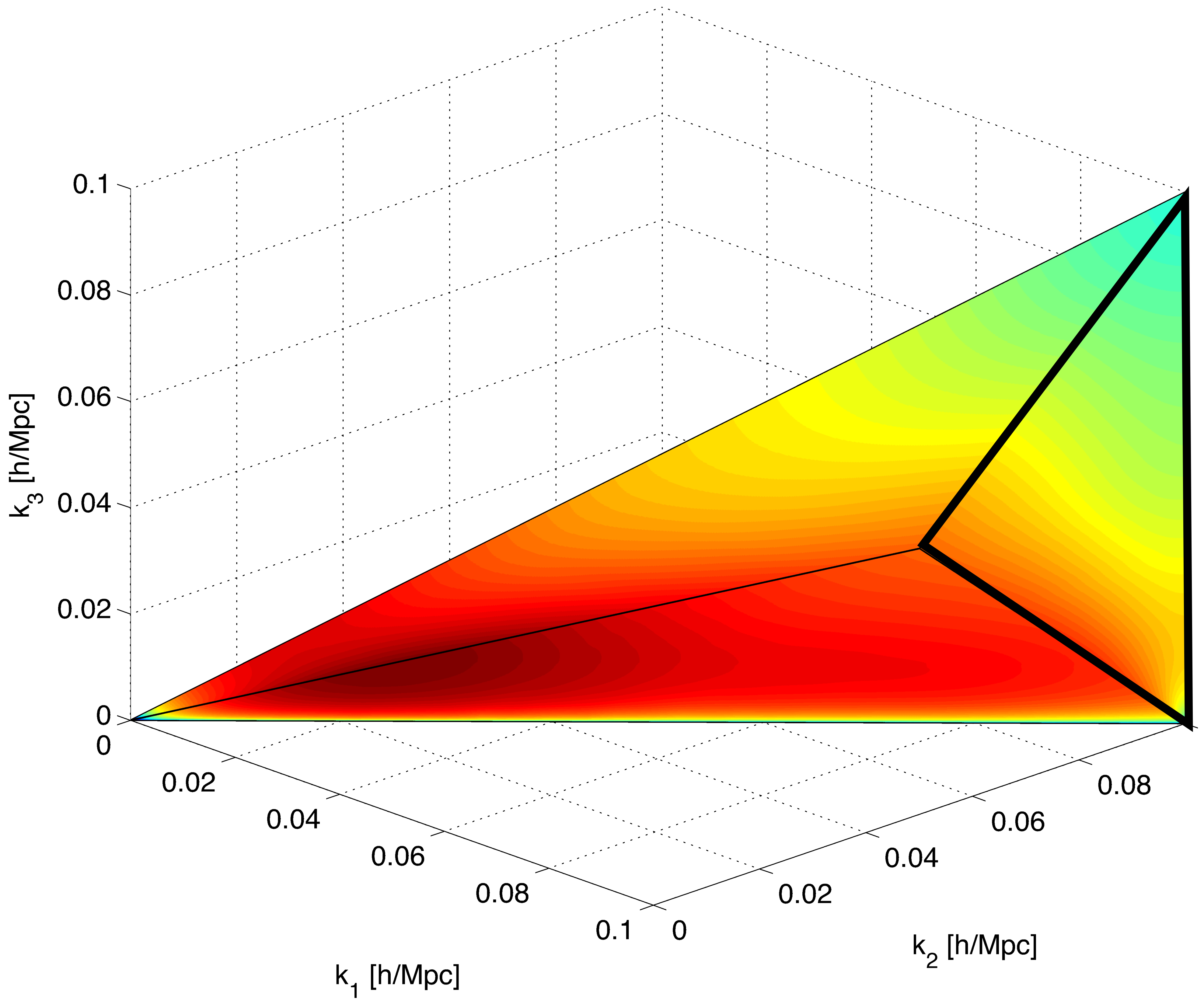


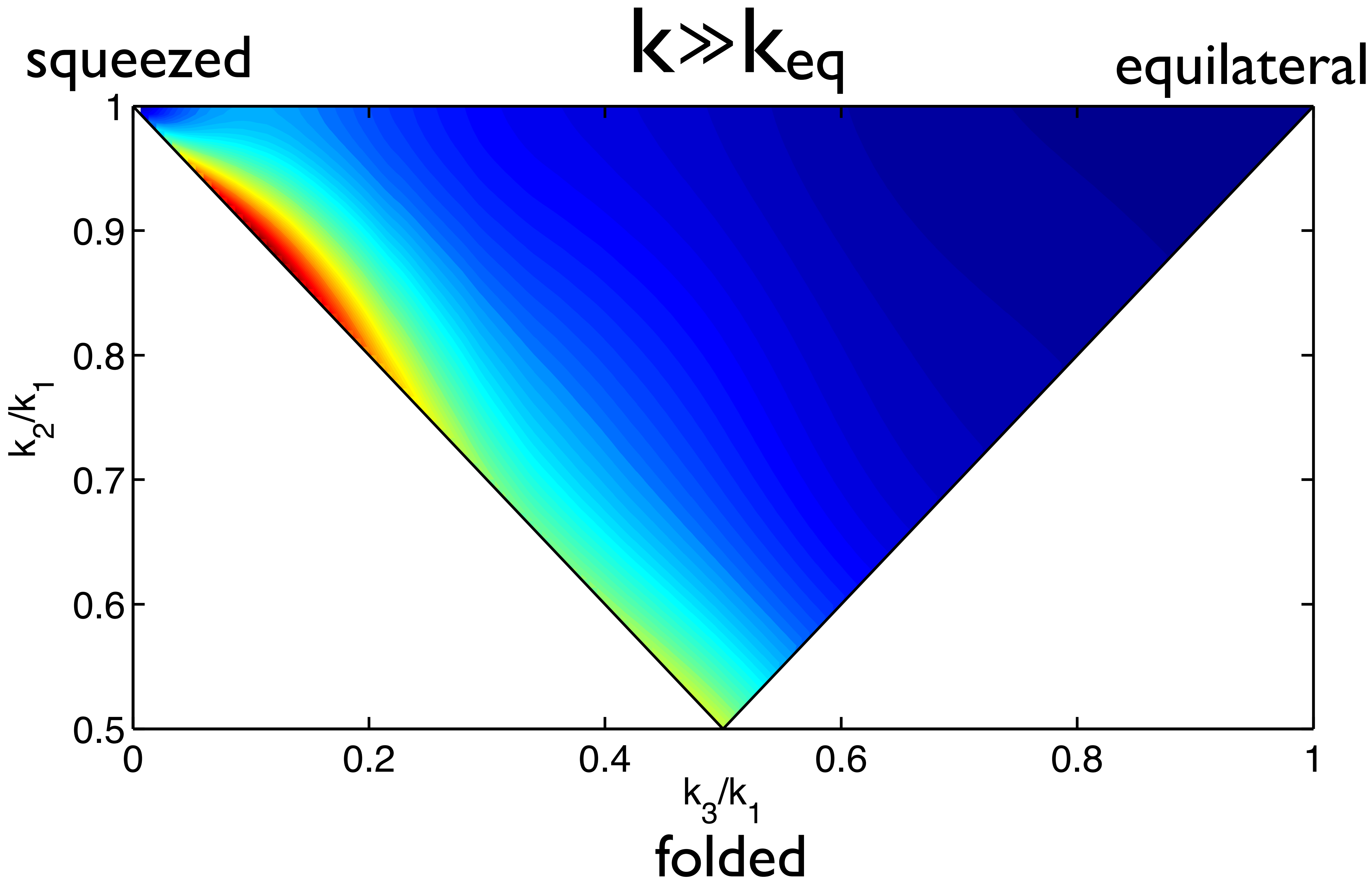
The slope of power spectrum

$d\ln P/d\ln k \lesssim 0$

Visualizing bispectrum

$k_1 \geq k_2 \geq k_3$



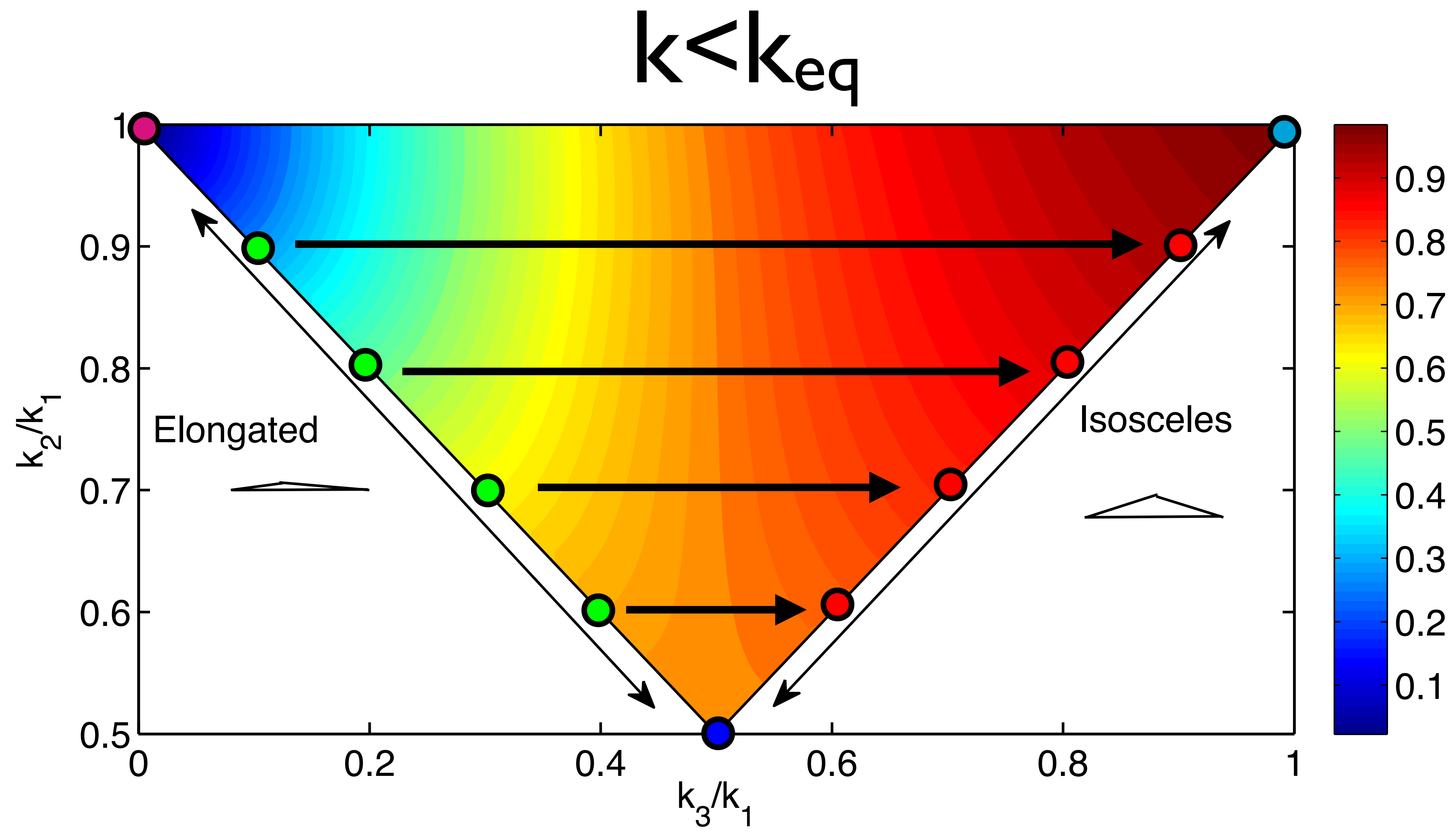


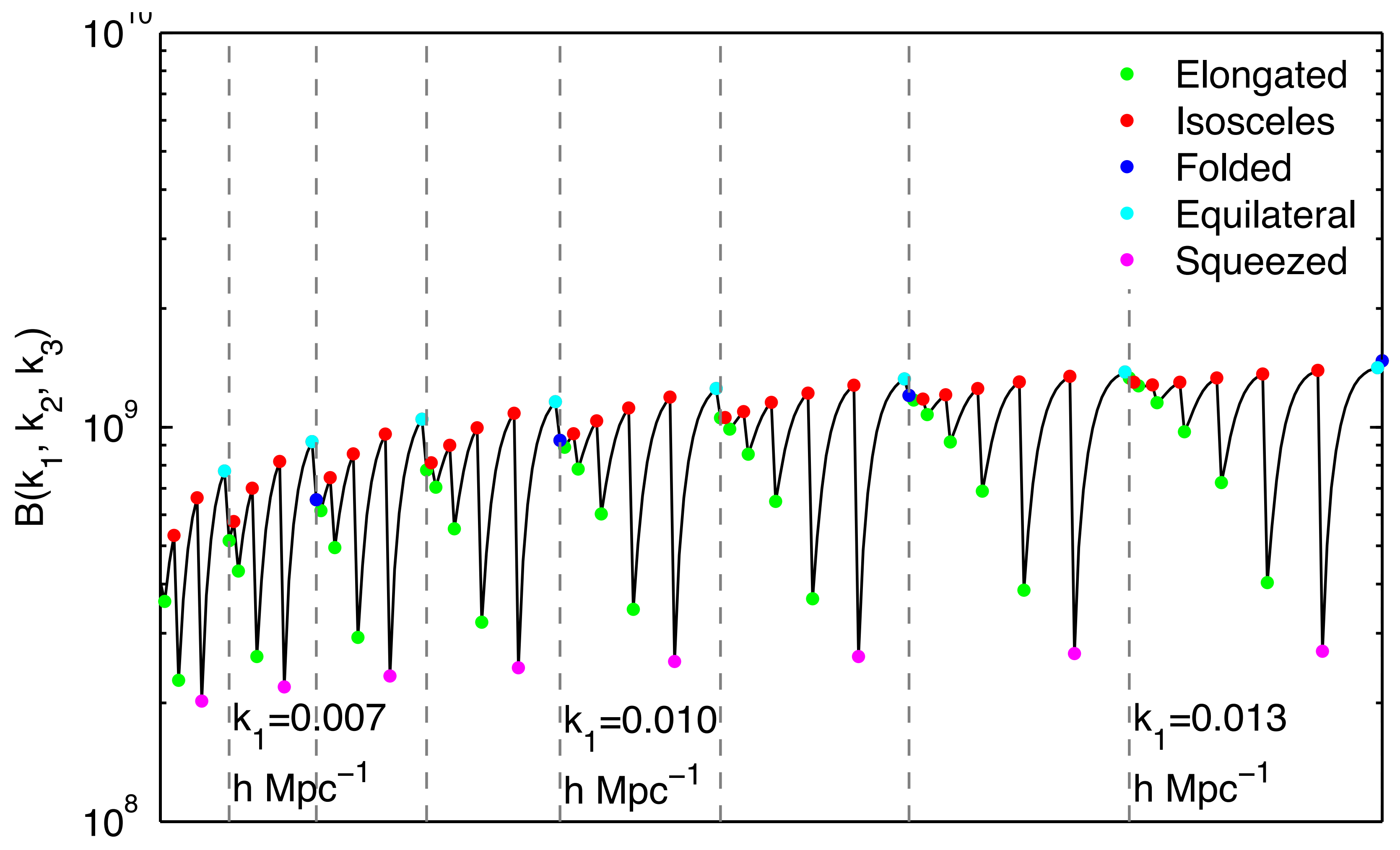
The slope of power spectrum

$d\ln P/d\ln k < 0$

Visualizing bispectrum

$k_1 \geq k_2 \geq k_3$

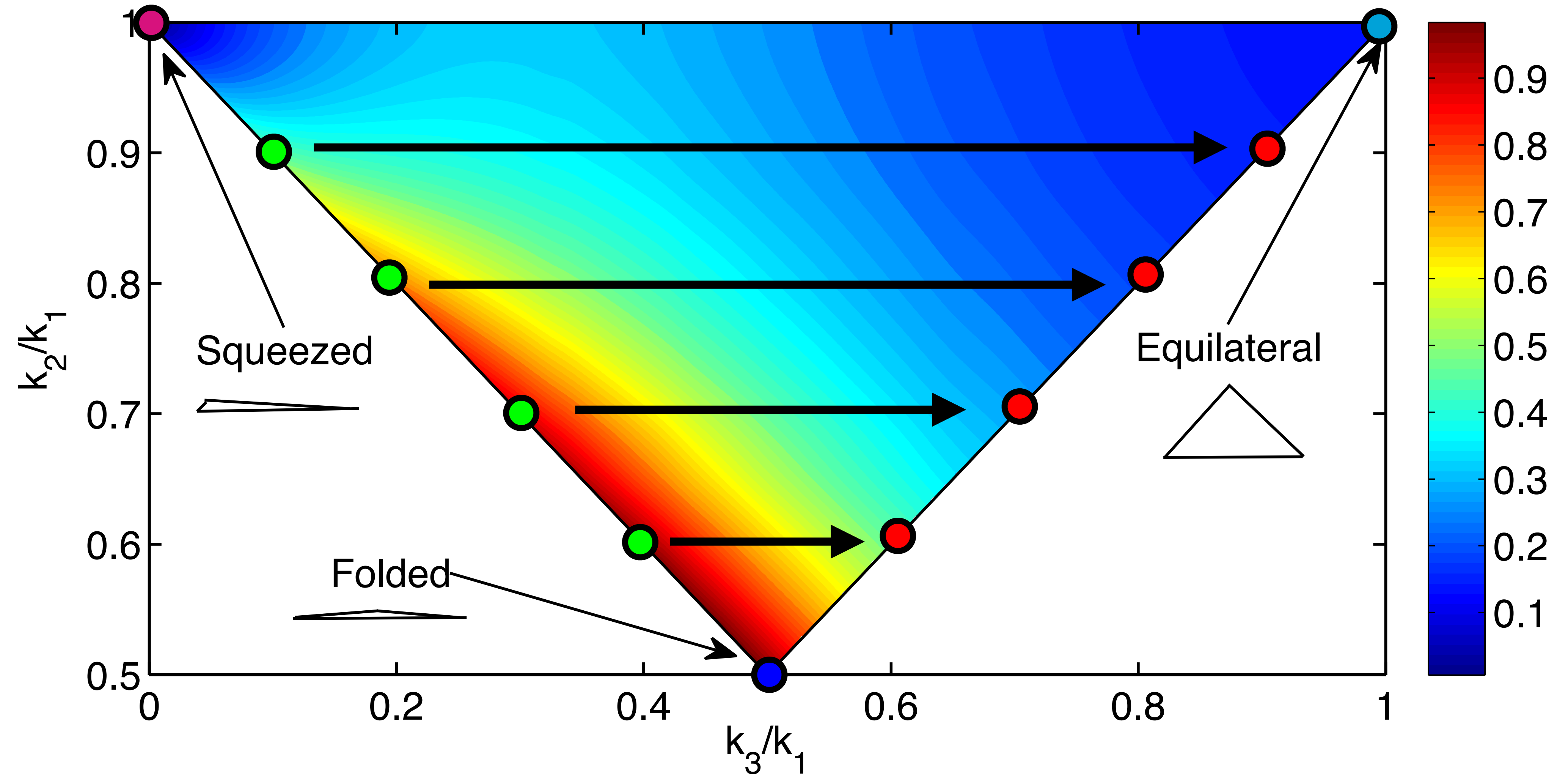


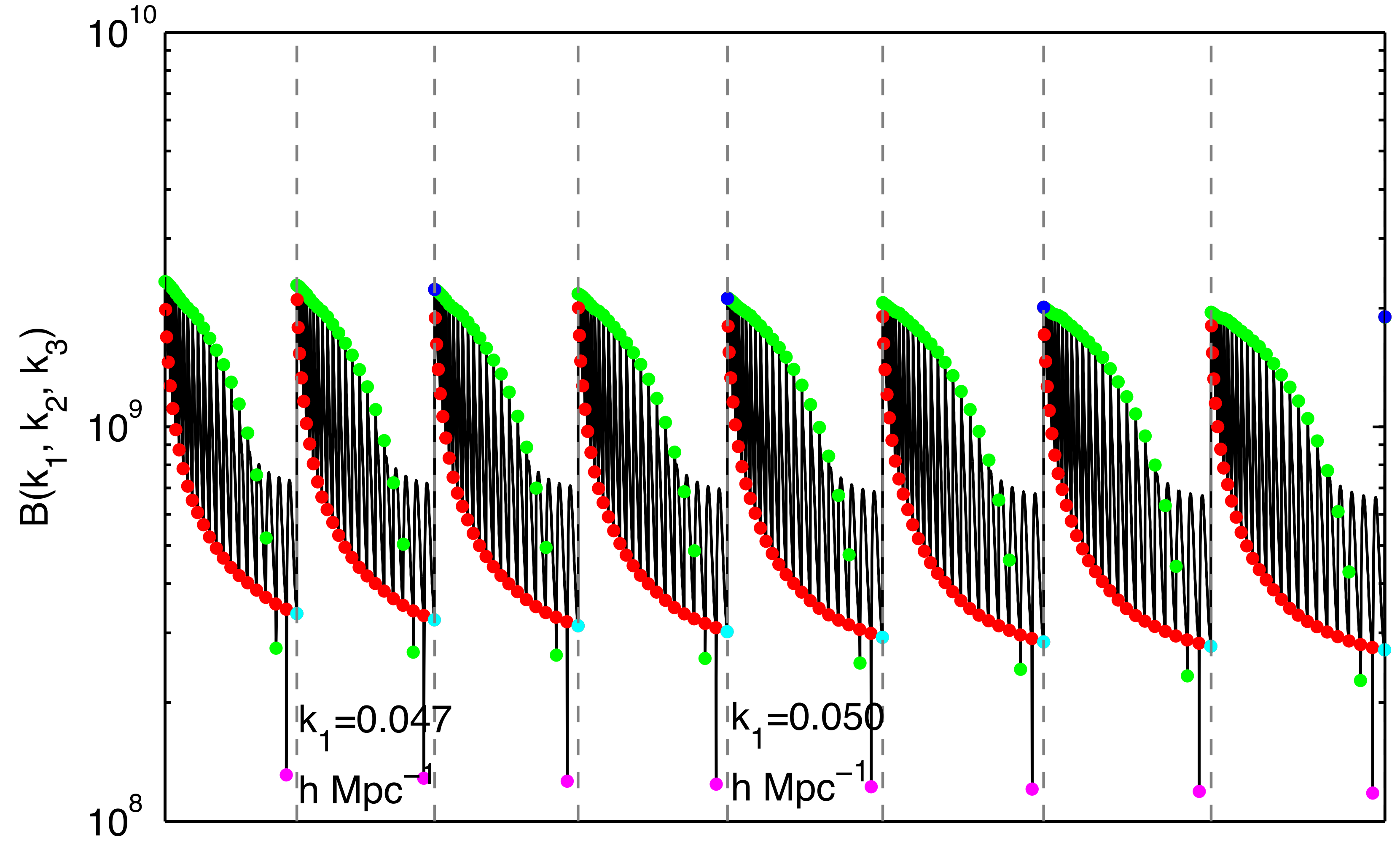


Visualizing bispectrum

$k_1 \geq k_2 \geq k_3$

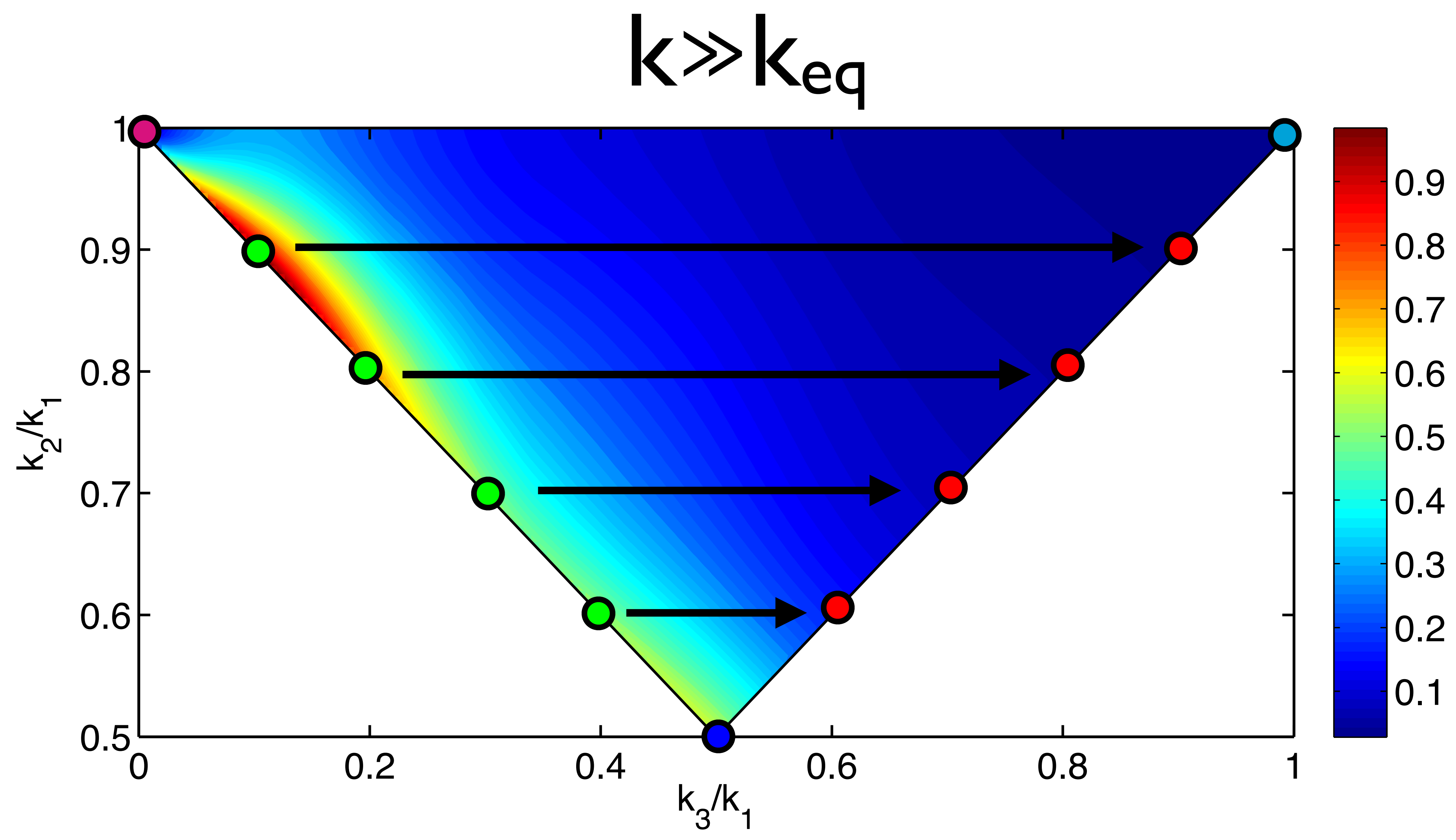
$k \gtrsim k_{eq}$

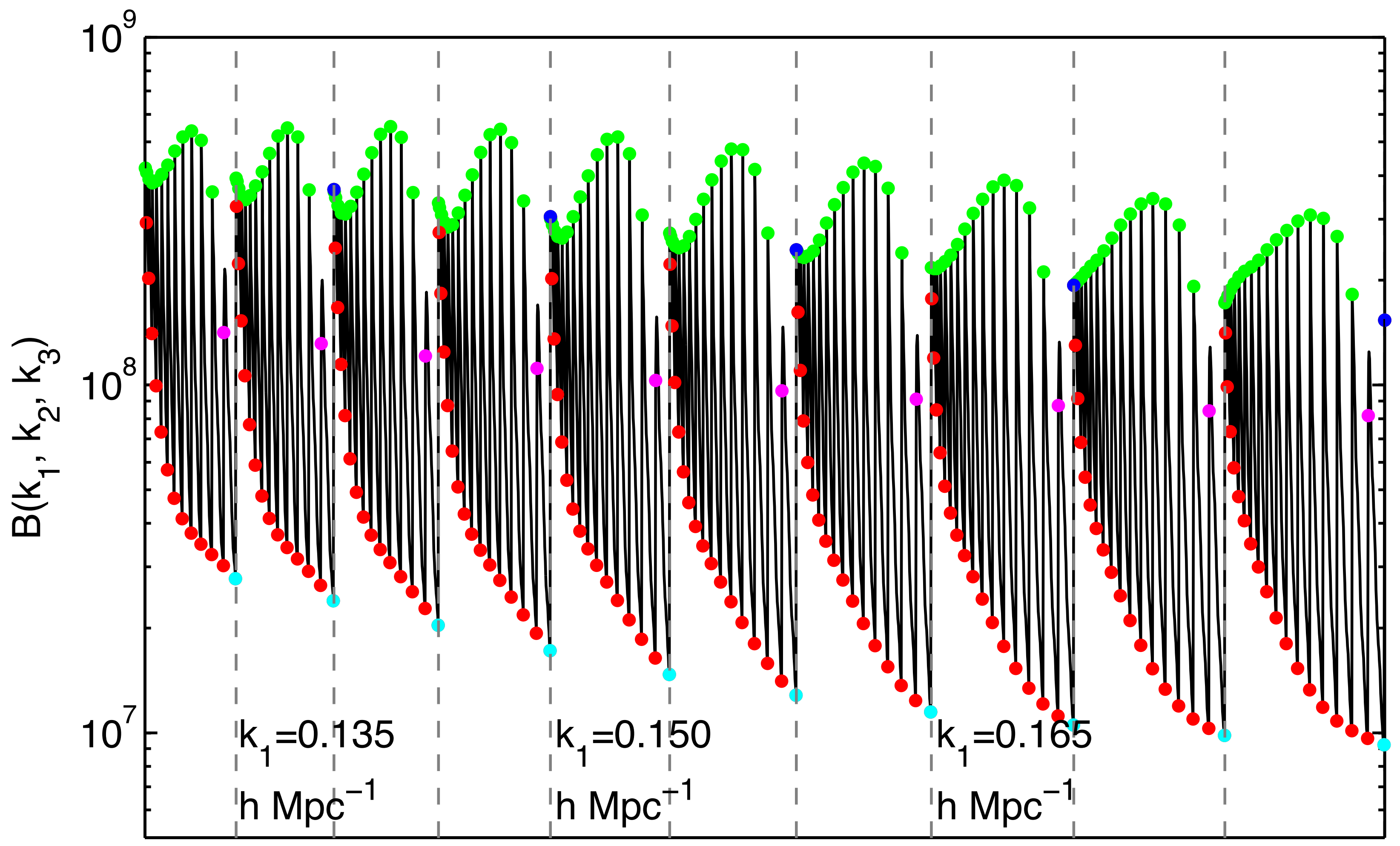


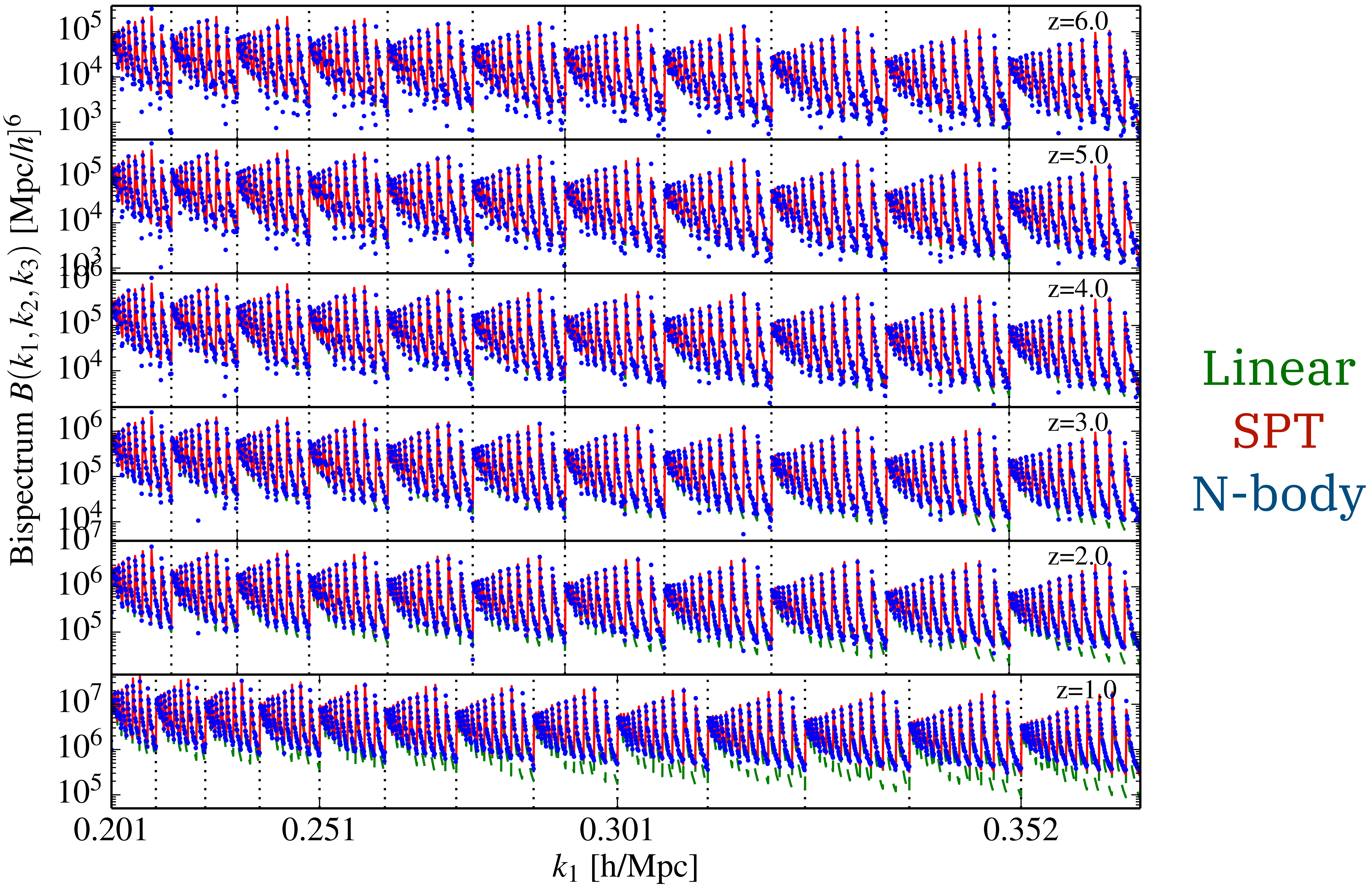


Visualizing bispectrum

$k_1 \geq k_2 \geq k_3$



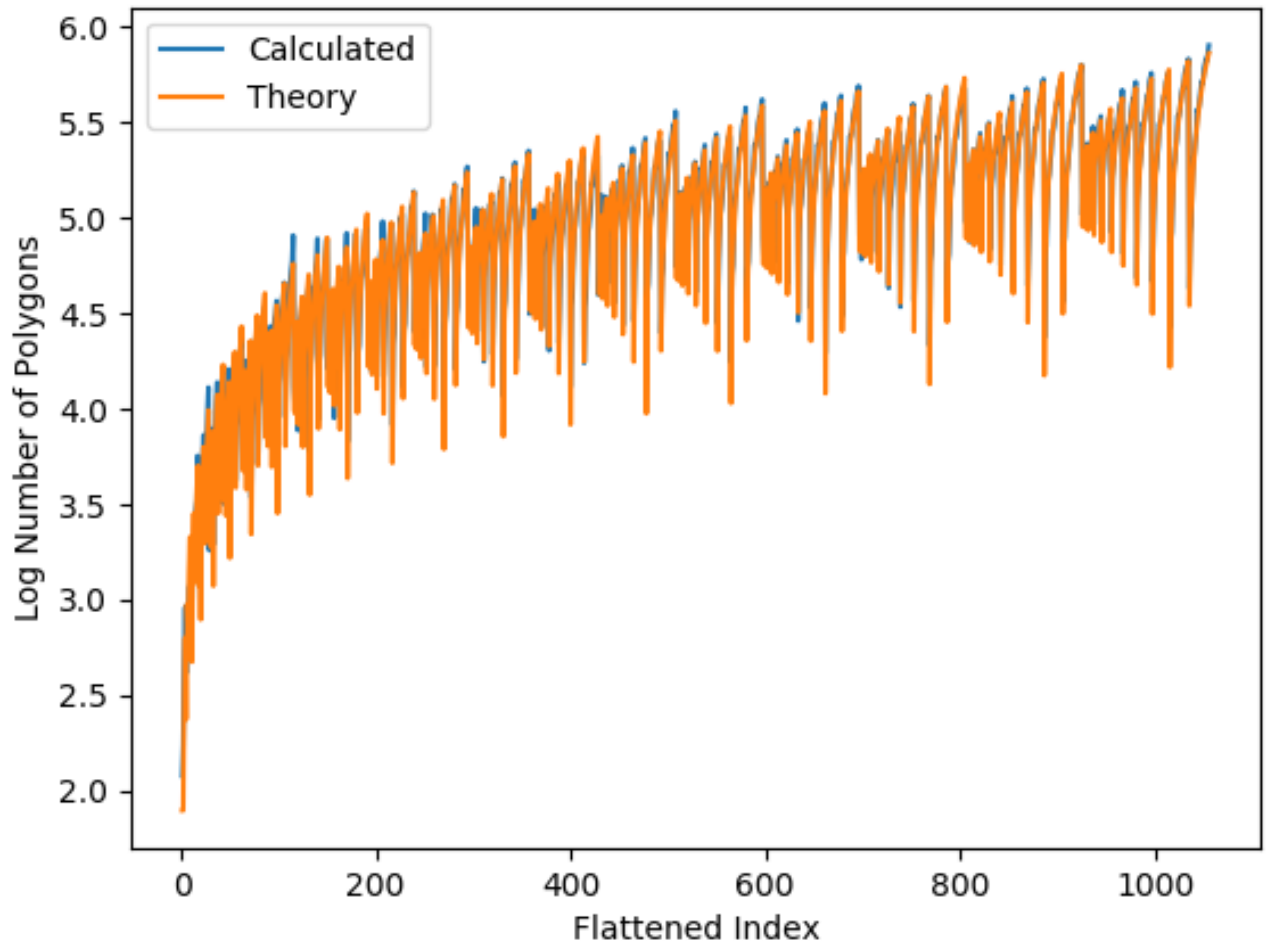


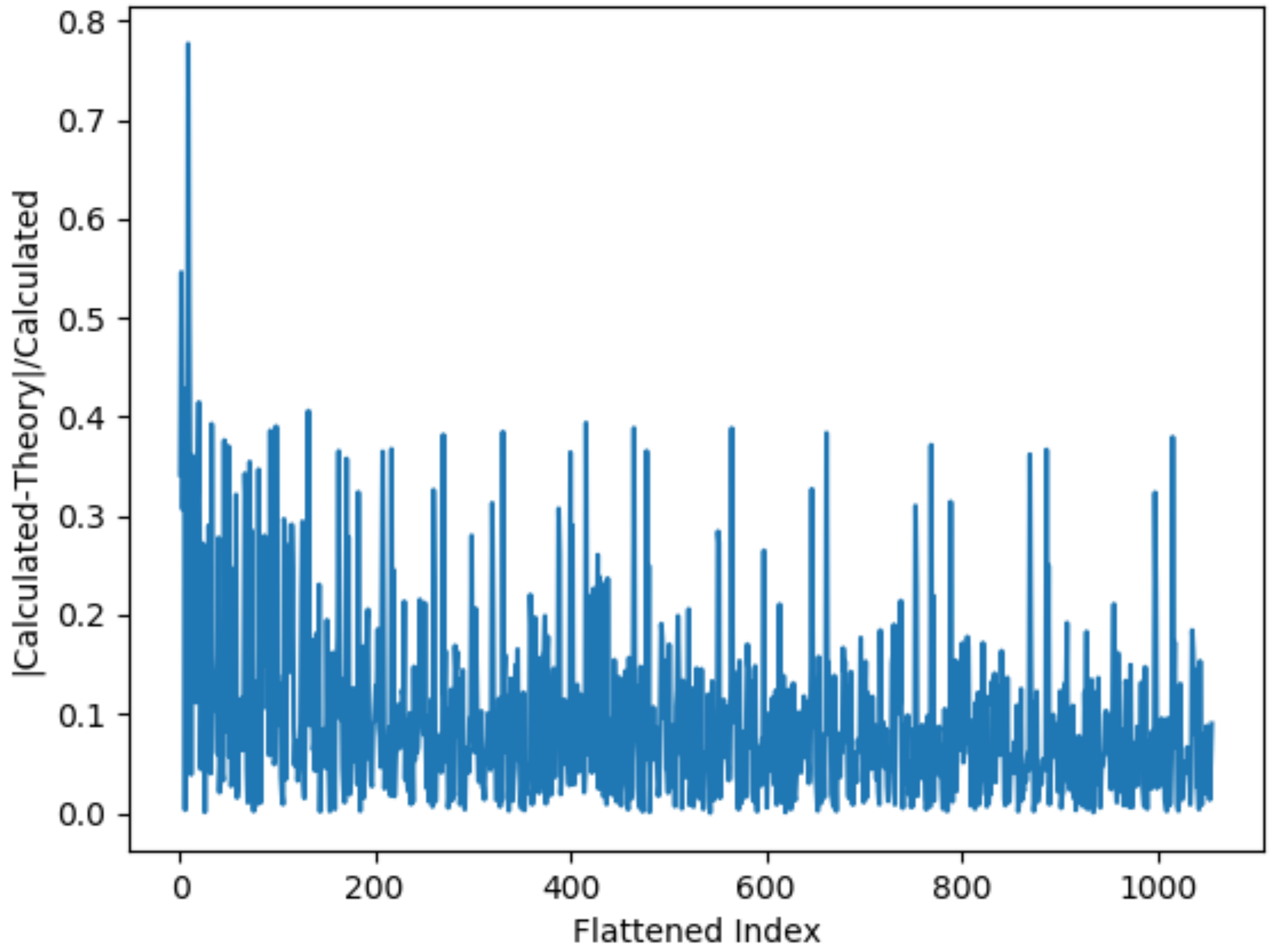


Linear
SPT
N-body
Bispectrum $B(k_1, k_2, k_3)$ [Mpc/h]^6
z=6.0
z=5.0
z=4.0
z=3.0
z=2.0
z=1.0
0.201
0.251
0.301
0.352
k_1 [h/Mpc]

Number of triangles

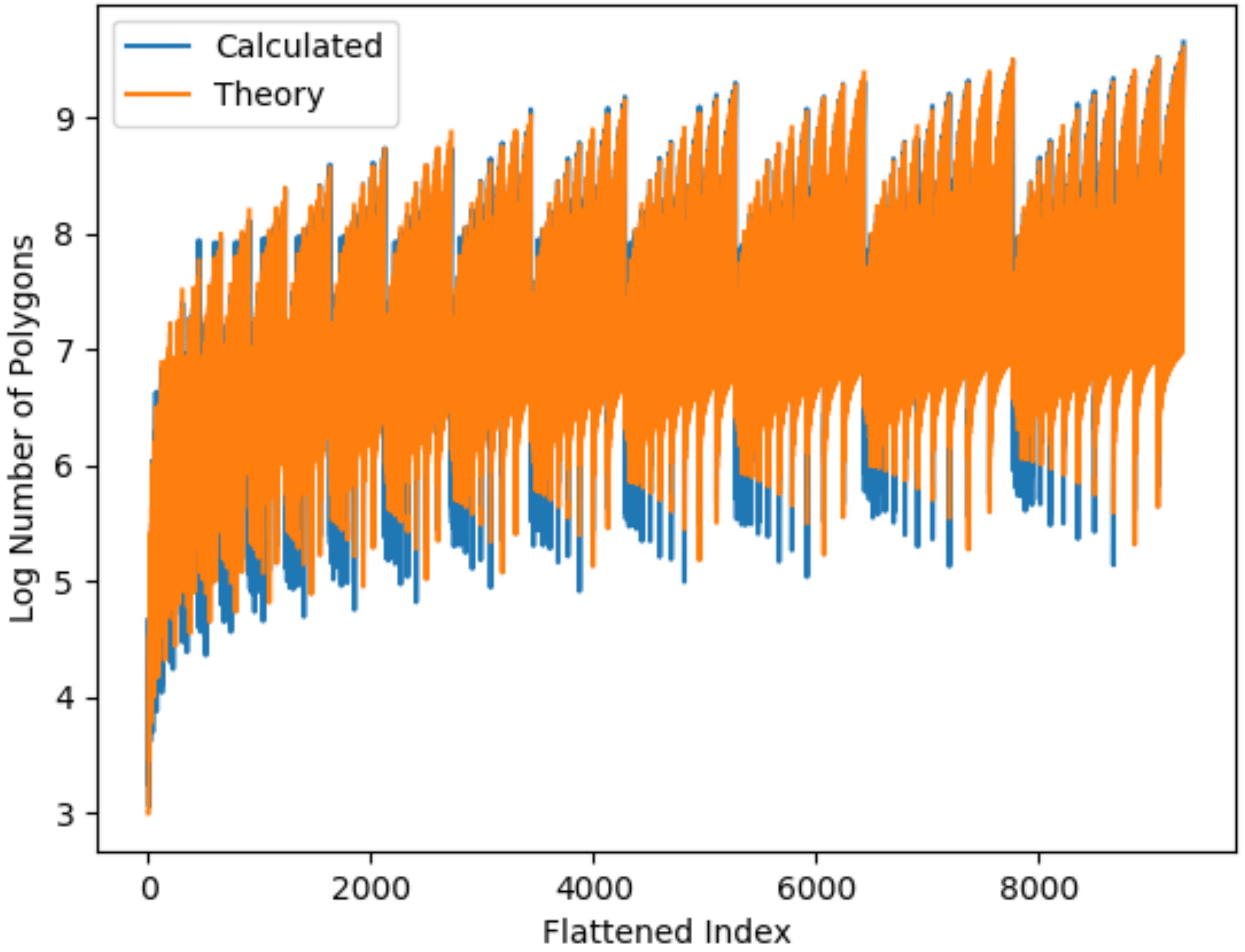
- Using the same estimator, but with δ=1, we can calculate the total number of triangles. Spikes at $k_1 = k_2 + k_3$

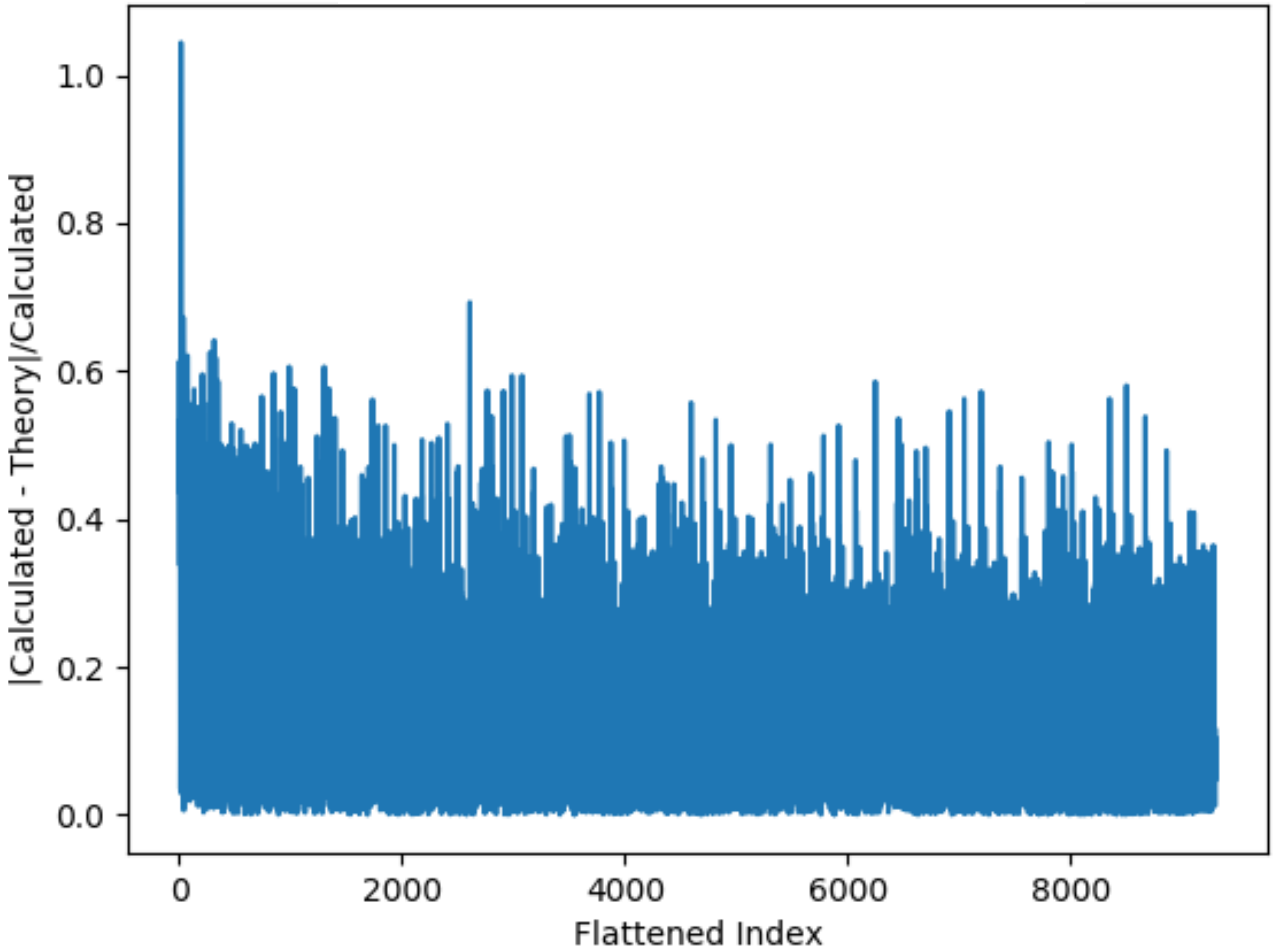




Number of quadrilaterals

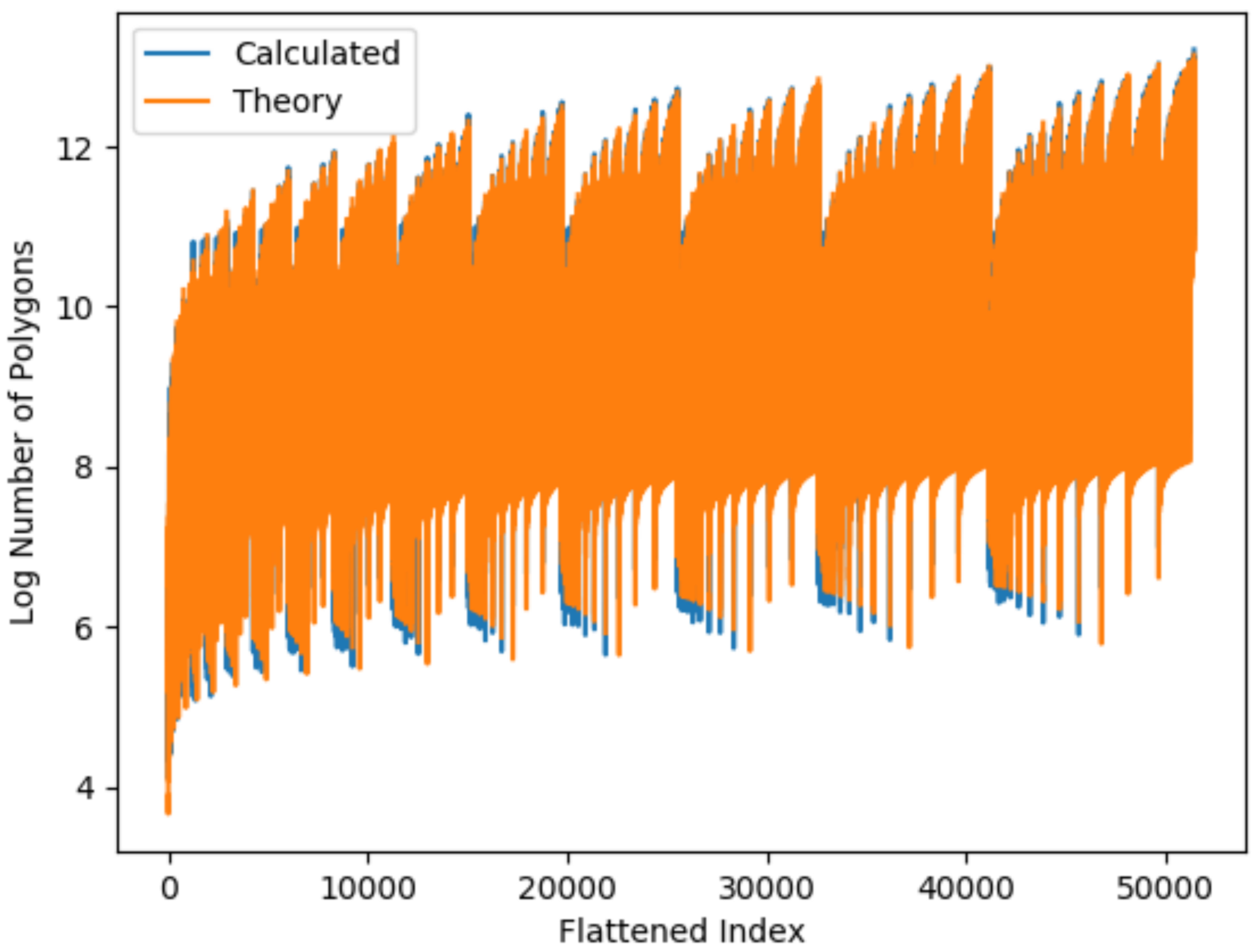
Why stop at triangle? Here's the angle averaged Trispectrum!

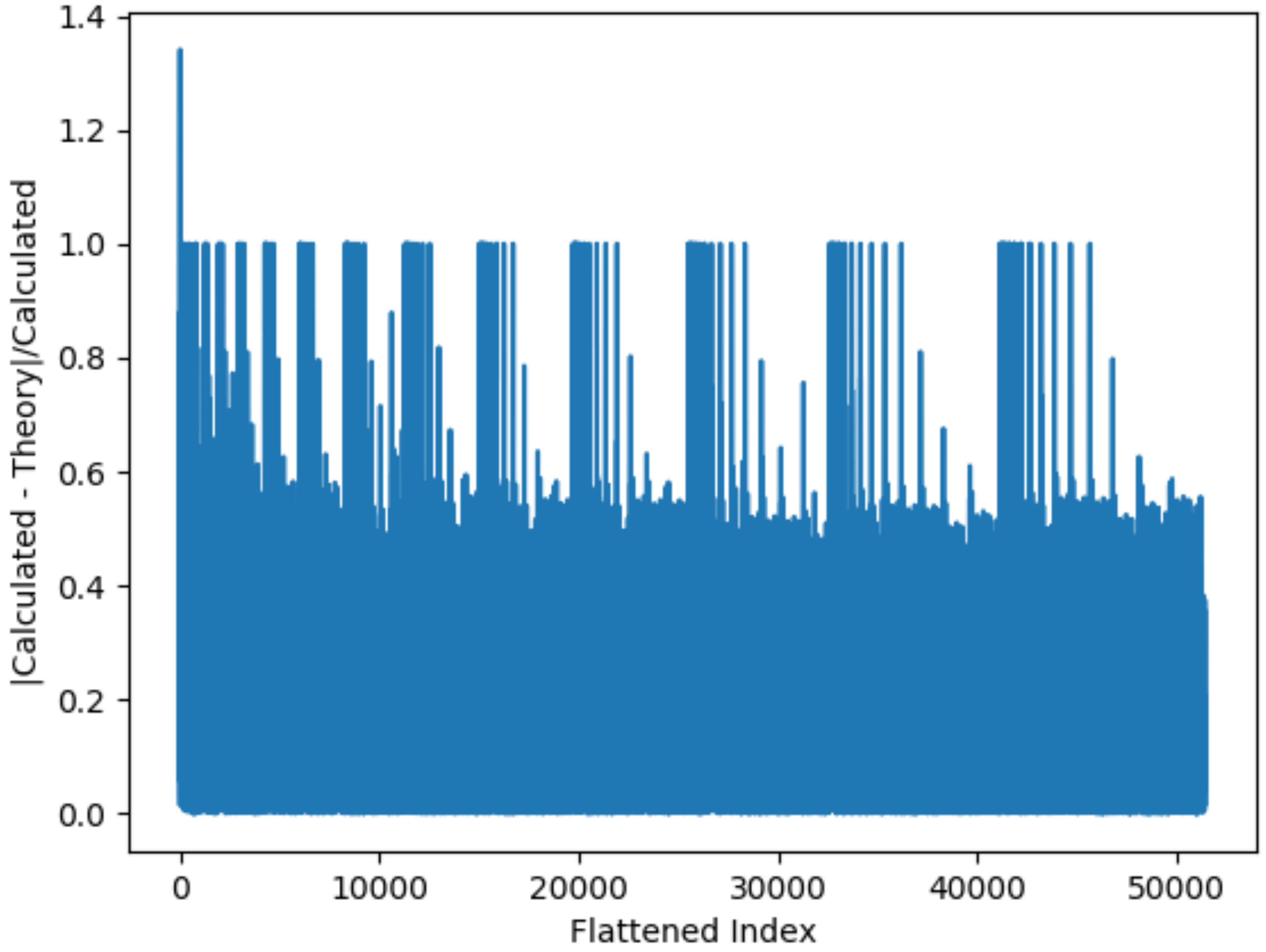




Number of pentagons

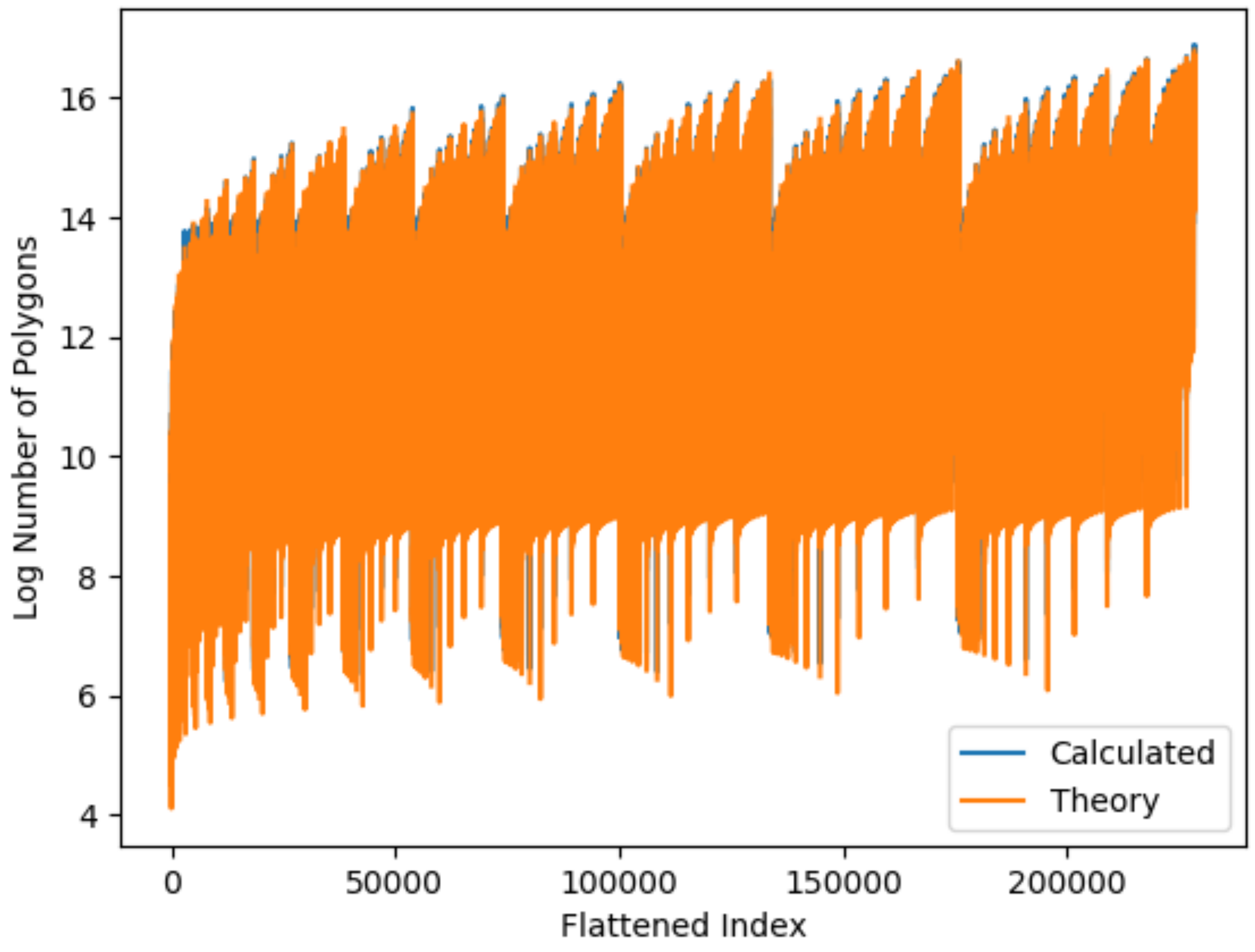
Why stop at trispectrum? Here's the angle averaged quadspectrum!

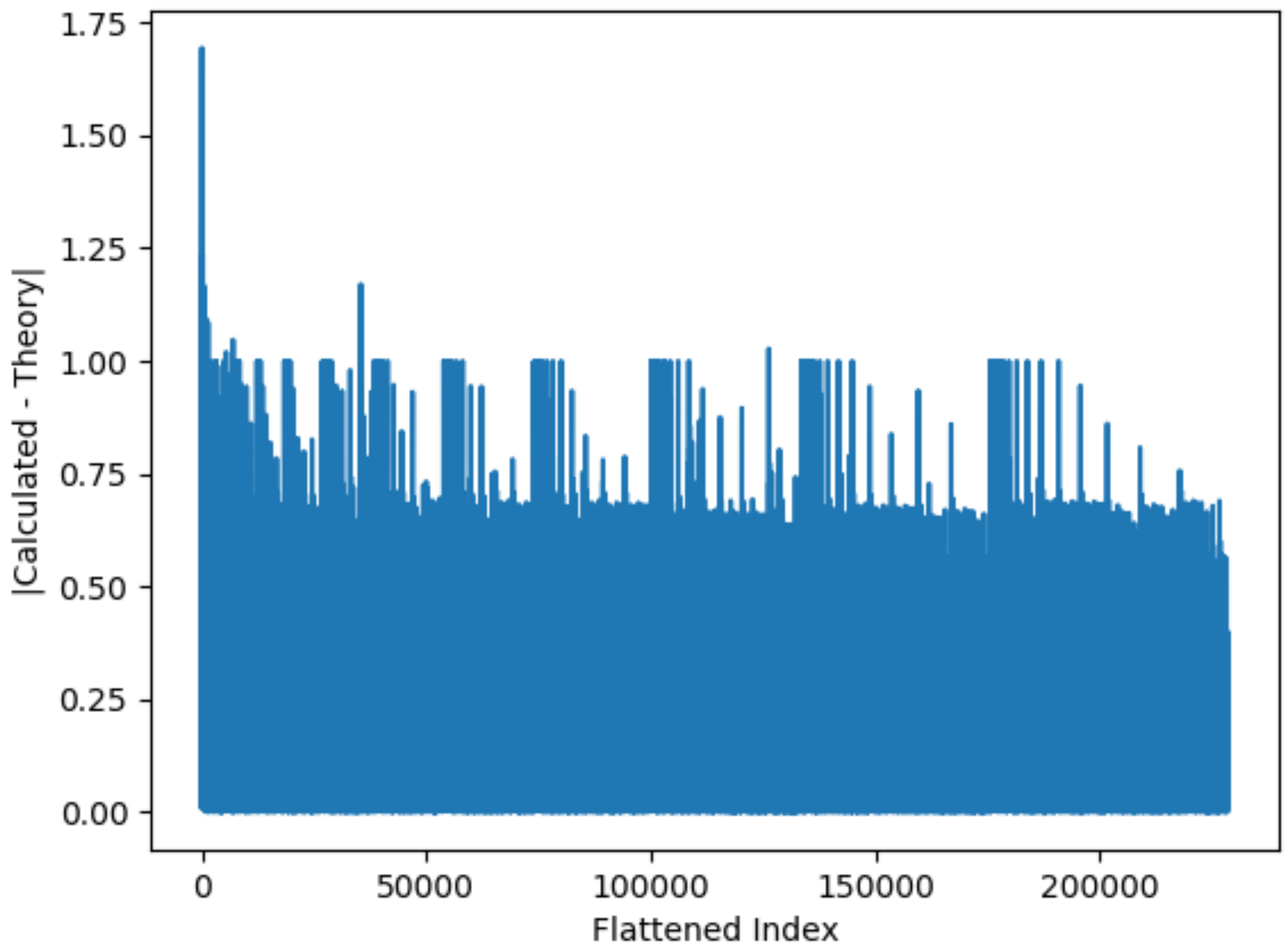




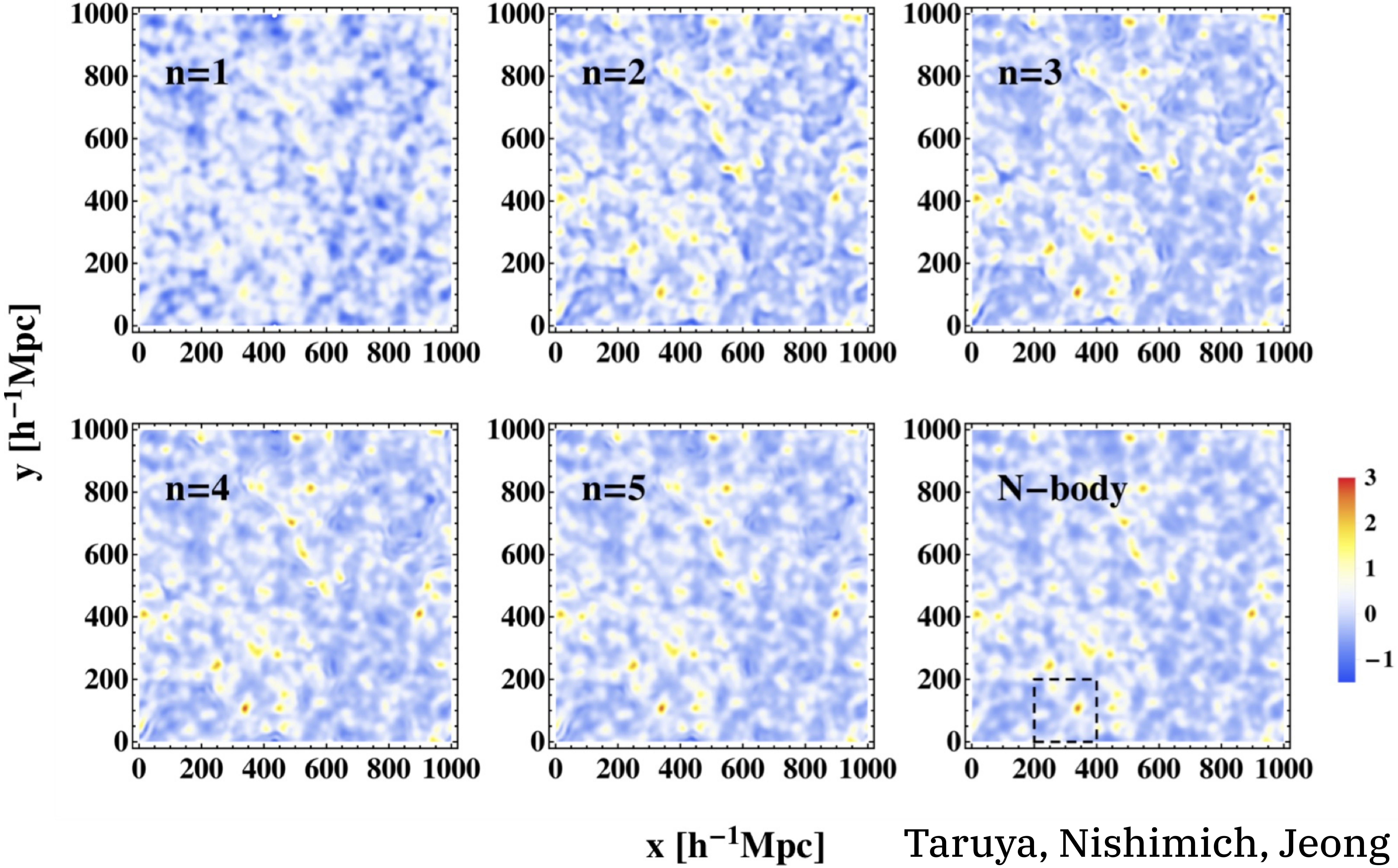
Number of hexagons

Why stop at quadspectrum? Here's the angle averaged pentaspectrum!





Application: polyspectra with GridSPT



Taruya, Nishimich, Jeong (2018)

Conclusion

- We present an efficient parallel algorithm for calculating higher-order polyspectra
- With the parallelization, we can overcome the high memory requirement of Scoccimarro estimator, and the parallel version is quite fast!
- Applying it to GridSPT, we can calculate the SPT prediction for the higher-order polyspectra!