University of Zurich[UZH]

Euclid consortium

The Euclid Emulator

M. Knabenhans (UZH), D. Potter (UZH), J. Stadel (UZH), R. Teyssier (UZH),
L. Legrand (Groningen/UZH), S. Marelli (ETH)

Fast and Accurate Power Spectrum Estimation

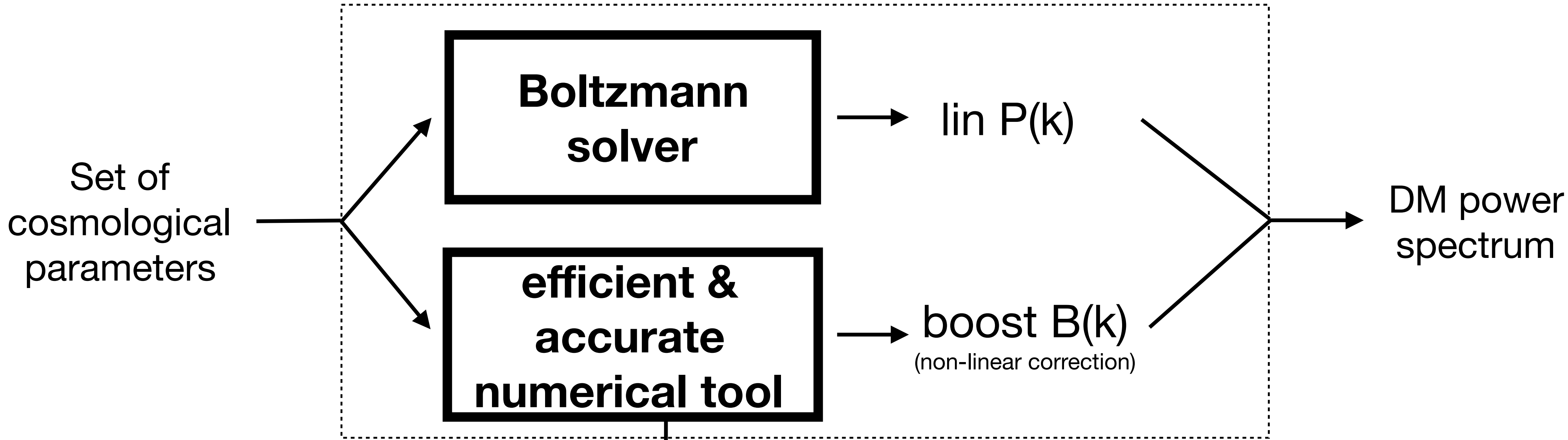
Set of cosmological parameters → **efficient & accurate numerical tool** → DM power spectrum

<u>Goals/Requirements:</u>

Efficiency: Compute a power spectrum in a node-second or less

Accuracy: Relative error maximally 1% up to non-linear scales (k_{max} ~ 10 h/Mpc)

Fast and Accurate Power Spectrum Estimation



Tool 1: N-body code

+ very accurate
- very expensive
- hard to make less expensive

Tool 2: Emulator (= „statistical predictor“)

+ very cheap
o less accurate but accurate enough

A cosmic emulator makes a MCMC-like MLE in the high dimensional cosmological parameter space feasible

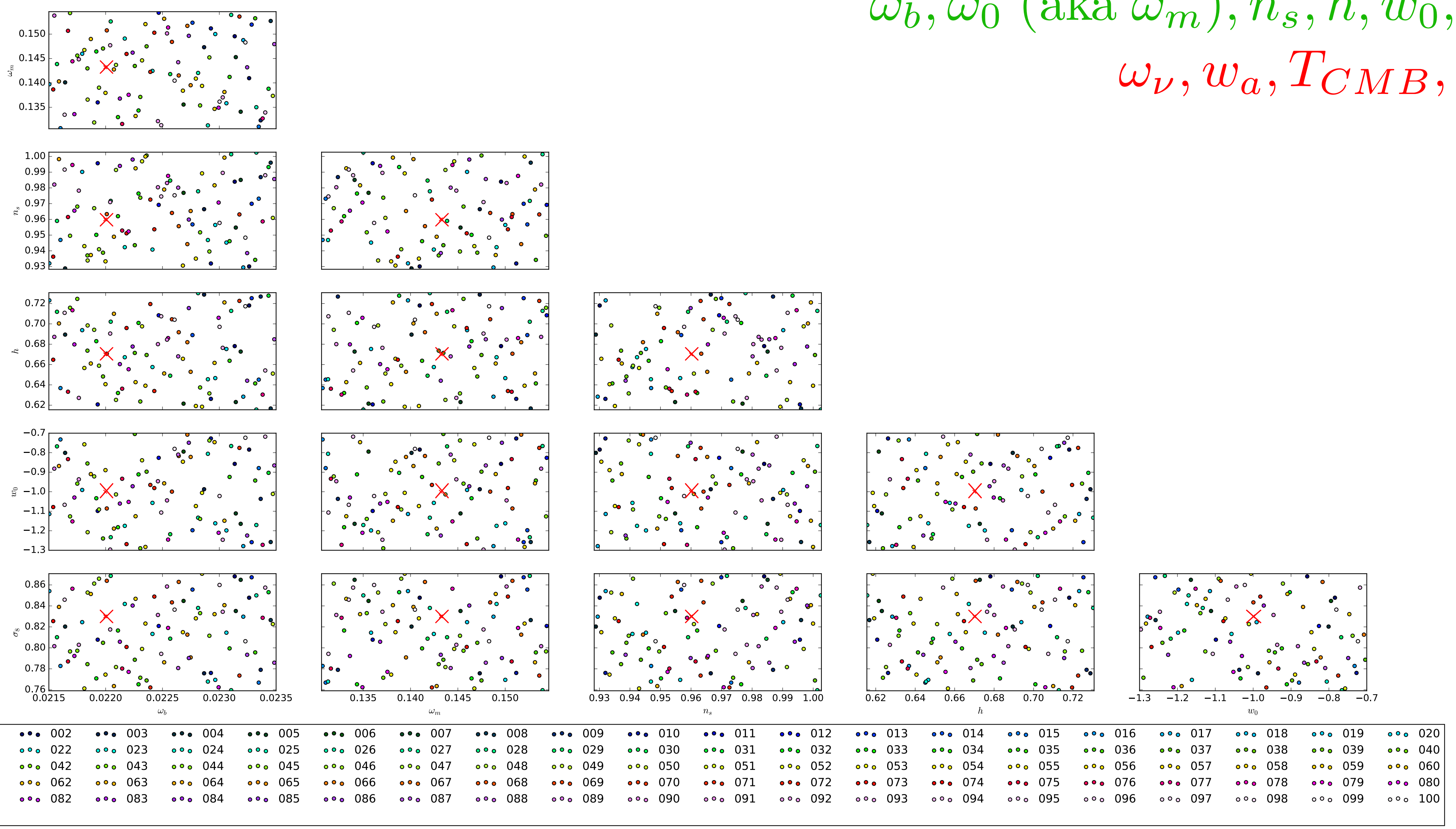
MAIN APPLICATION: PARAMETER FORECASTING!!!

BUT: Simulation and emulation errors need to be optimal!!!

Sampling the Cosmological Parameter Space with LHS

$\omega_b, \omega_0 \text{ (aka } \omega_m), n_s, h, w_0, \sigma_8$

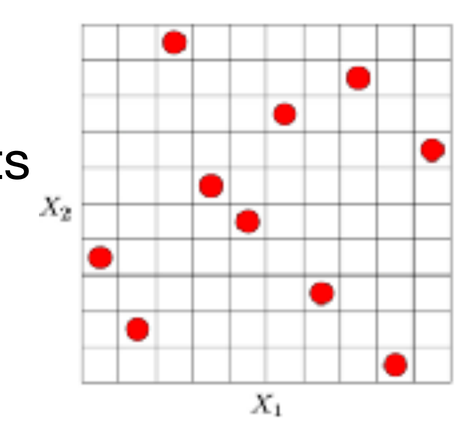
$\omega_\nu, w_a, T_{CMB}, \ldots$



<u>Sampling strategy:</u>

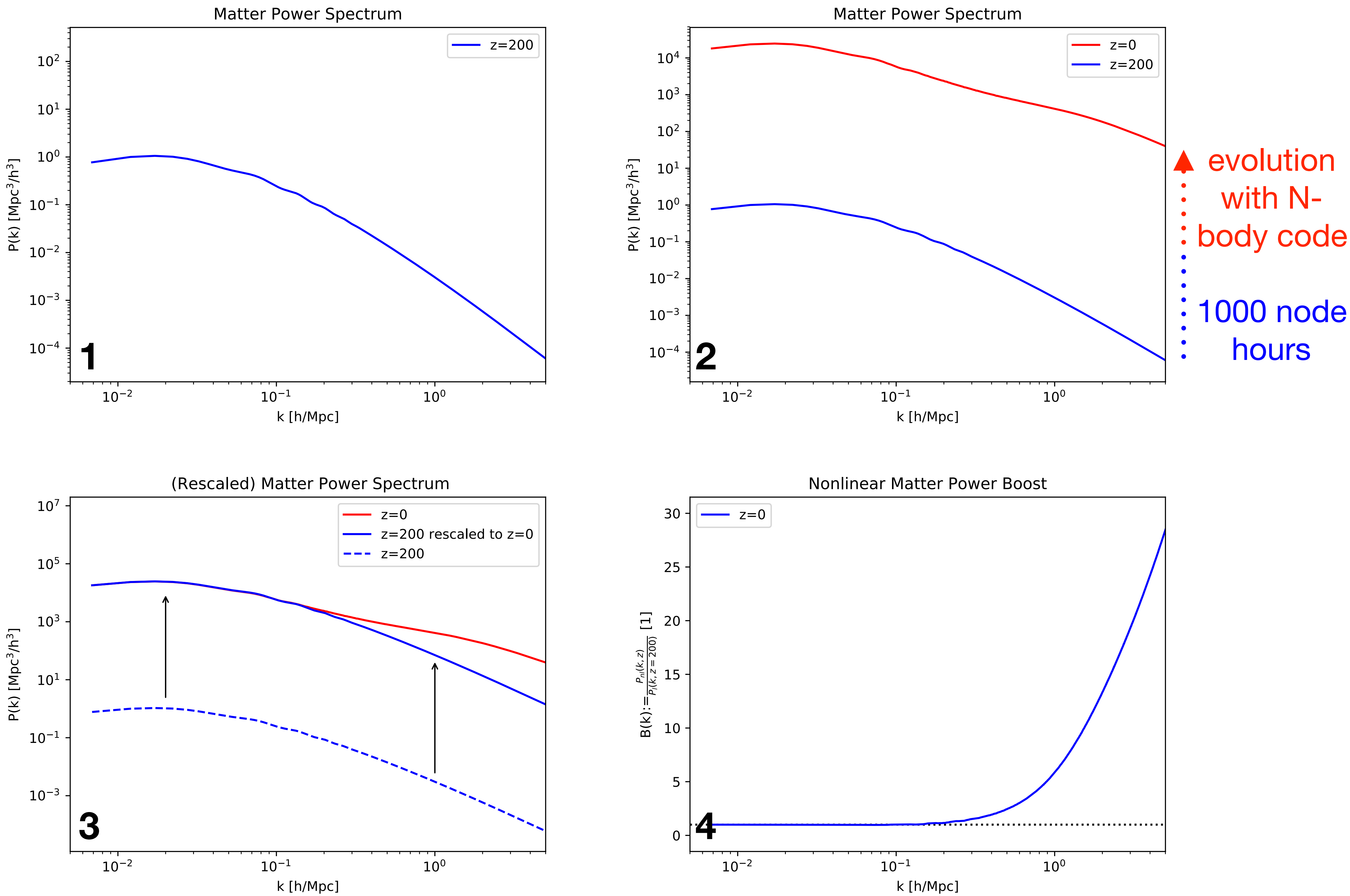
Step 1: 10^5 latin hypercube sampling (LHS) realizations of N=100 data points

Step 2: Maximize the minimal distance between the data points



LHS looks like a sudoku!

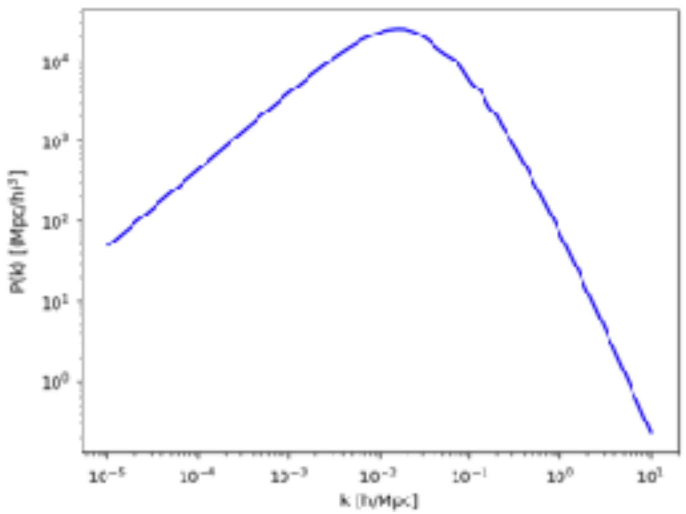
Definition & Computation of the Boost Factor



$$B(k,z) := \frac{P_{non-linear}(k,z)}{P_{linear}(k,z)}$$

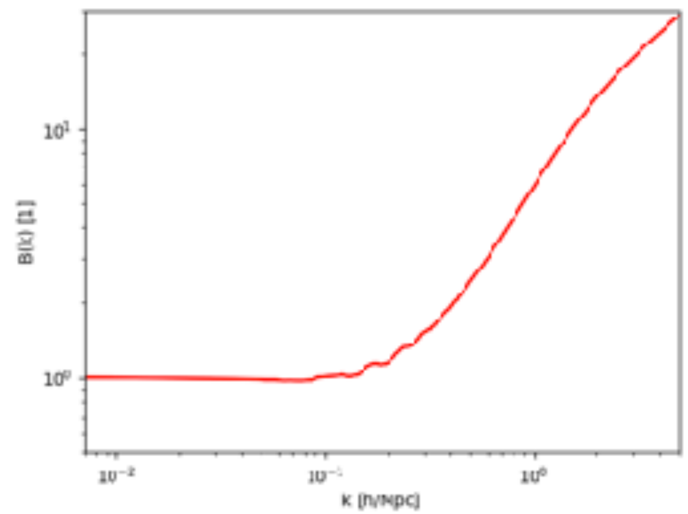
Power Spectrum vs Boost Factor Emulation — Transparency

linear power spectrum



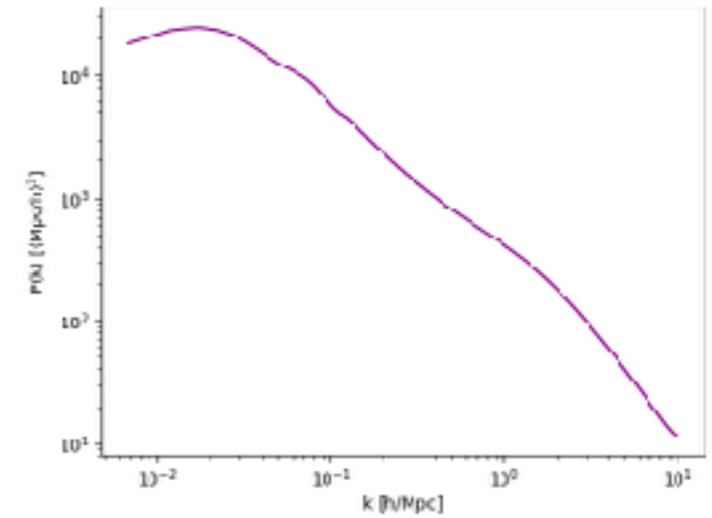
×

boost factor



=

non-linear power spectrum



Boltzmann solver

+ very fast
+ precise up to linear order
+ includes lots of complicated physics (e.g. GR effects)

Emulator

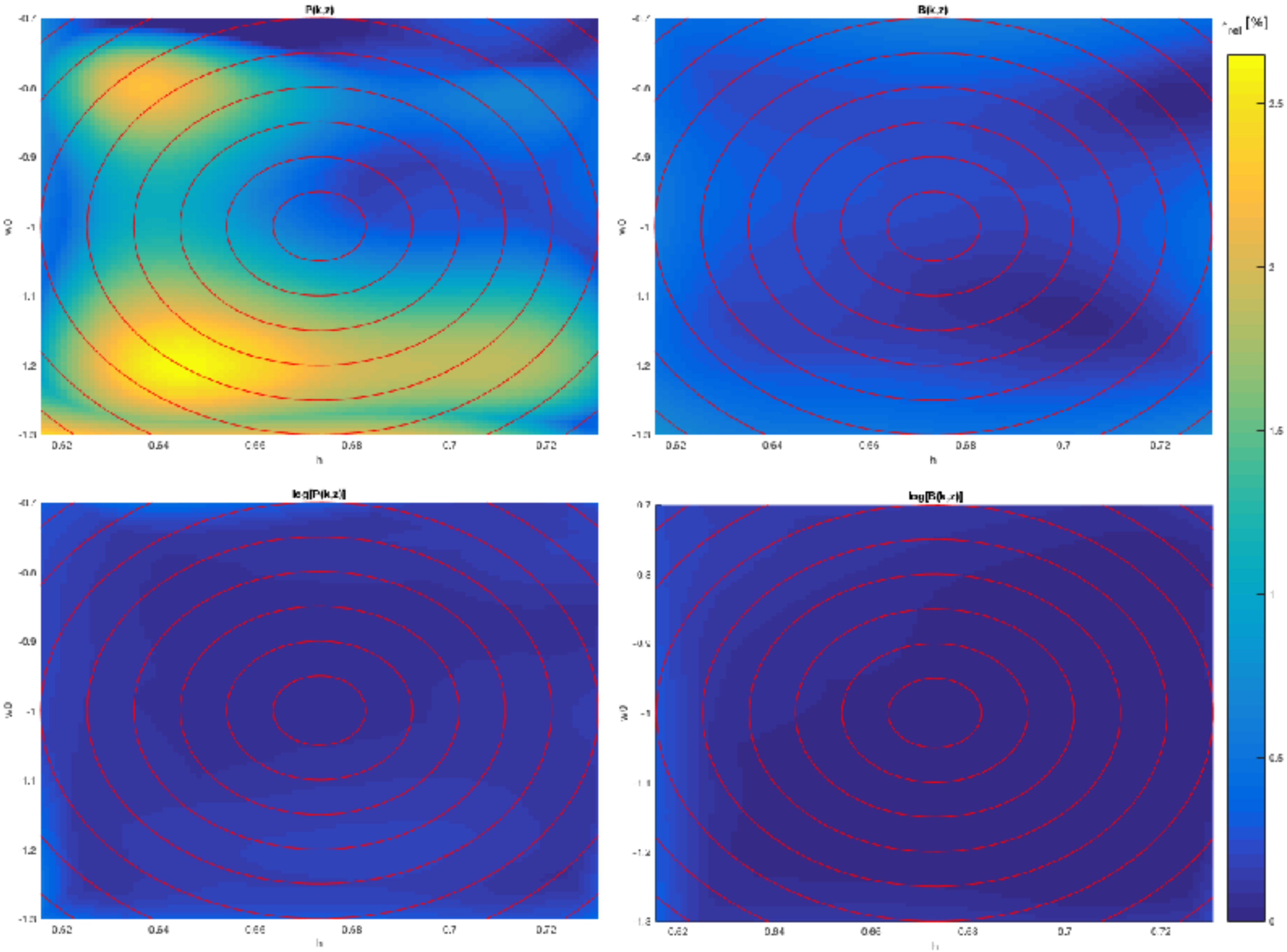
+ very fast
+ very accurate
+ includes non-linearities in DM clustering

Result

+ very fast
+ very accurate on all scales
+ includes DM non-linearities
+ includes complicated physics to linear order

BOOST FACTOR EMULATION
COMBINES THE BEST OF BOTH WORLDS!

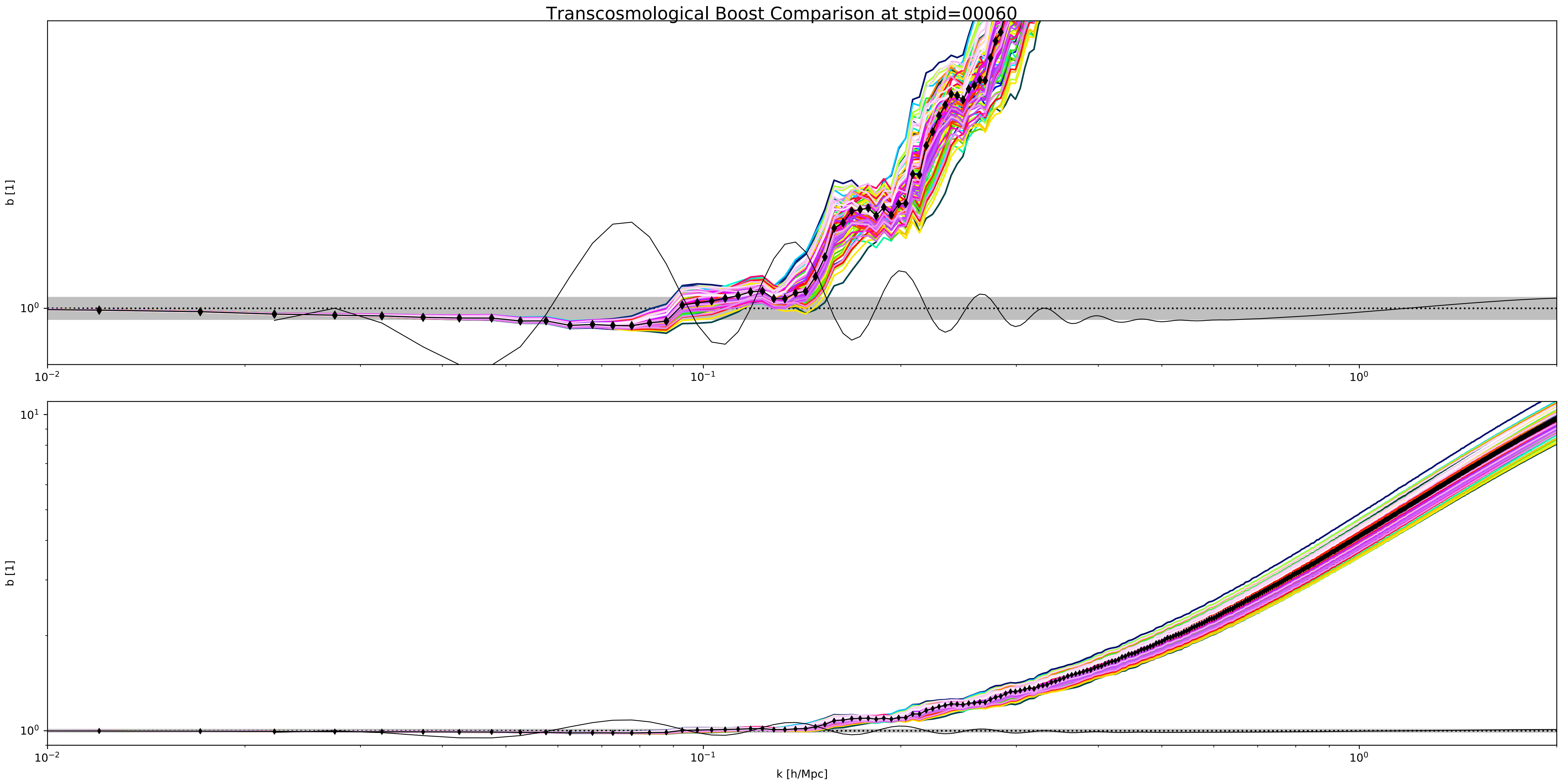
Power Spectrum vs Boost Factor Emulation — Accuracy



Emulation of the boost factor leads to smaller emulation errors!

Methods

Input Simulations - The Experimental Design Data



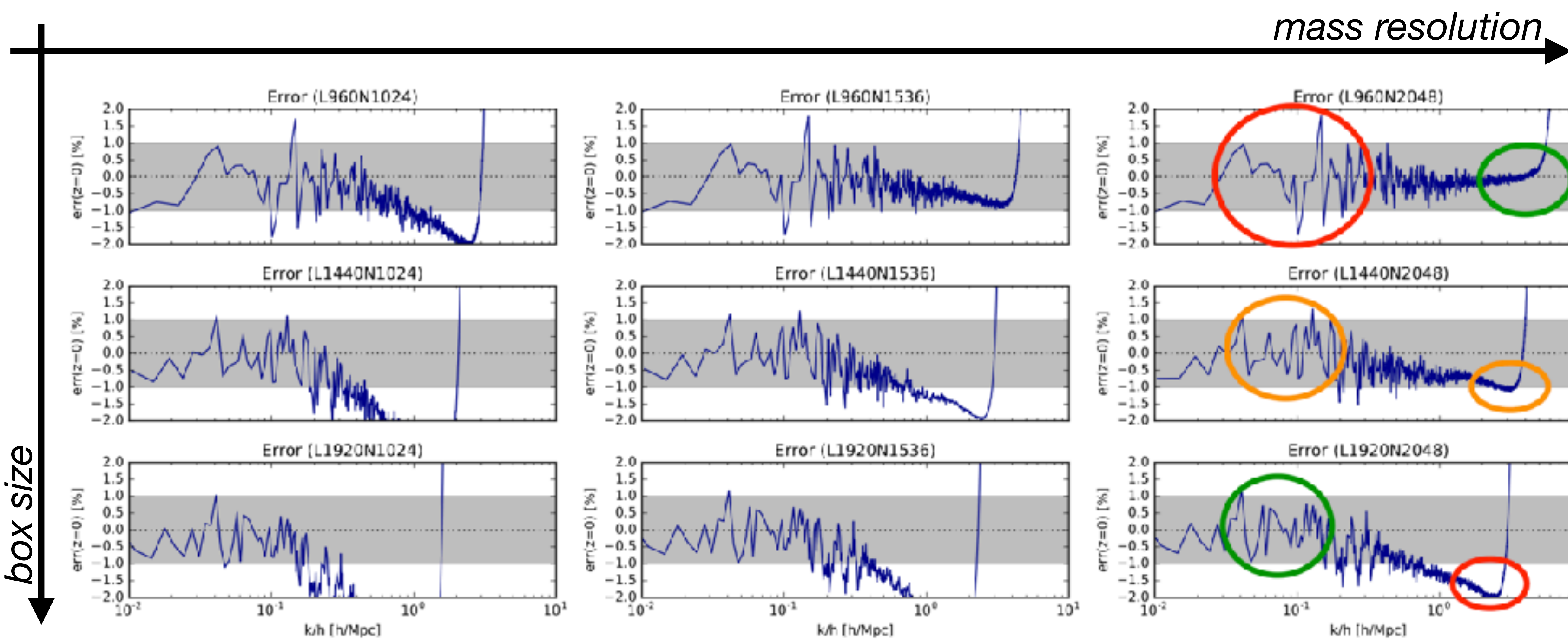
Simulations run with PKDGRAV3 (J. Stadel & D. Potter)
on UZH-based cluster zBox4

Step 3: Run N-body simulation for each of these 100 cosmologies and measure the DM power spectrum/boost factor (= construction of the experimental design)

Step 4: Construct a statistical predictor for DM power spectrum/boost factor based on these simulations

Simulation Convergence Test

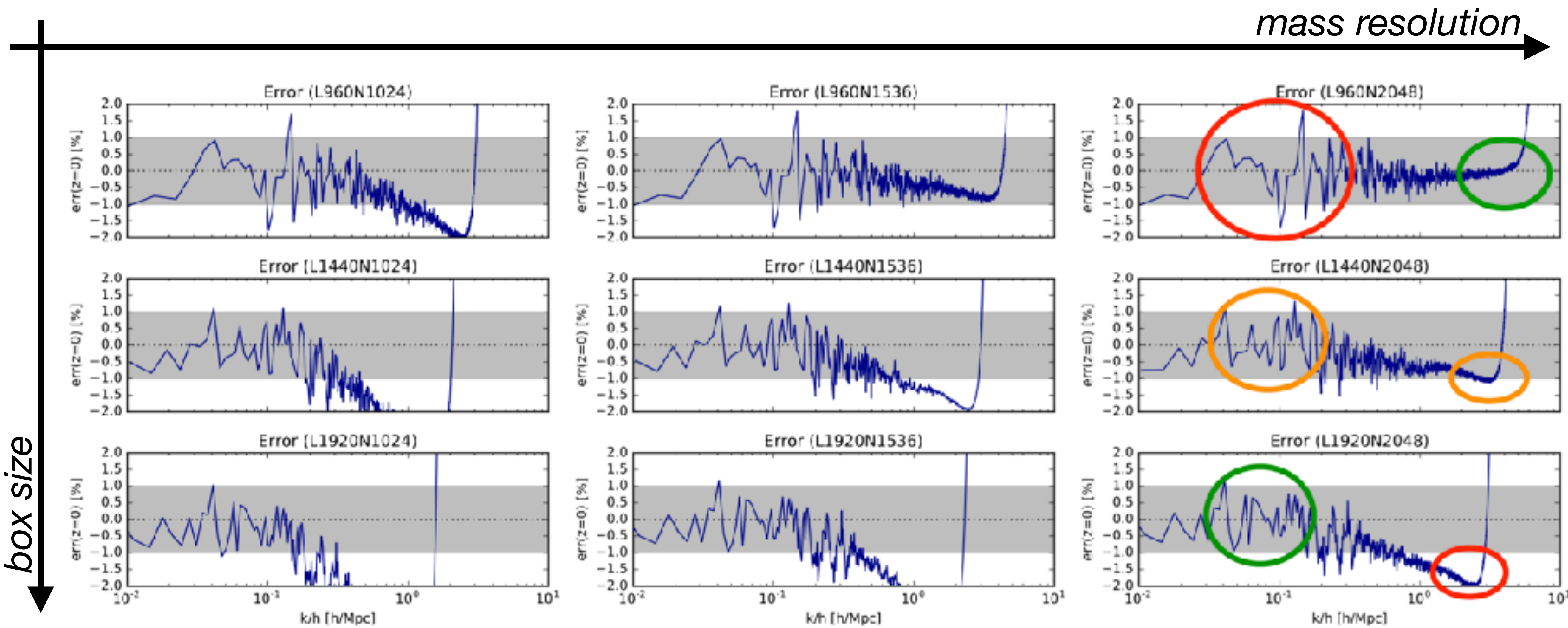
Comparison of L[…]N[…] runs against a L1920N8000 run (relative error of boost factor curves):



We can expect good results from L ~ 2000 Mpc/h and R := N/L ~ 2.05 (corresponding to N^3 ~ 4096^3 particles)

Simulation Convergence Test

Comparison of L[…]N[…] runs against a L1920N8000 run (relative error of boost factor curves):

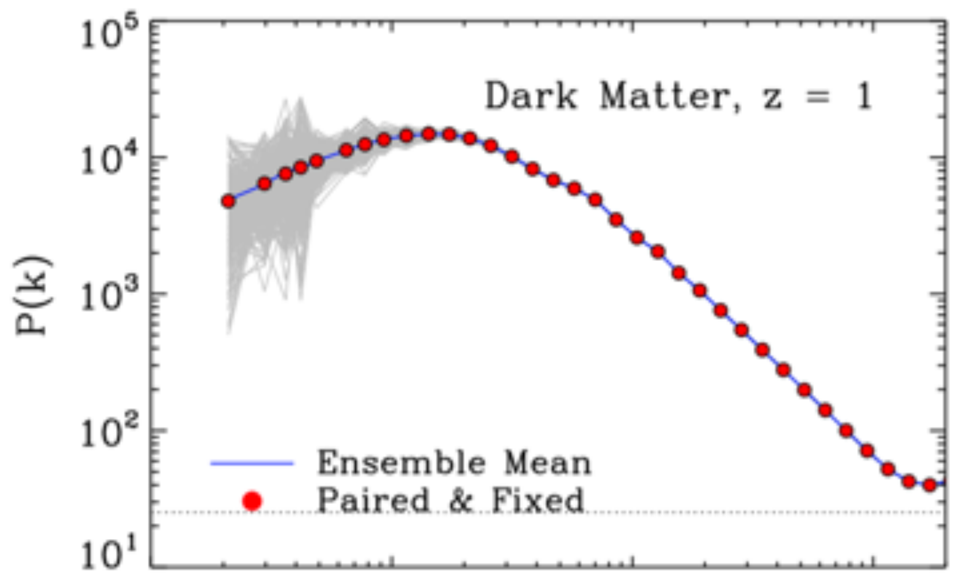


!!! ca. 1.3×10^6 node hours for entire suite —> TOO EXPENSIVE !!!

The bare minimum is given by L ~ 1250 Mpc/h and R ~ 1.64 (corresponding to N^3 ~ 2048^3 particles)

(still ca. 200’000 node hours for entire suite)

Improving Quality of Input Simulations - ``Pairing & Fixing“ (PF)



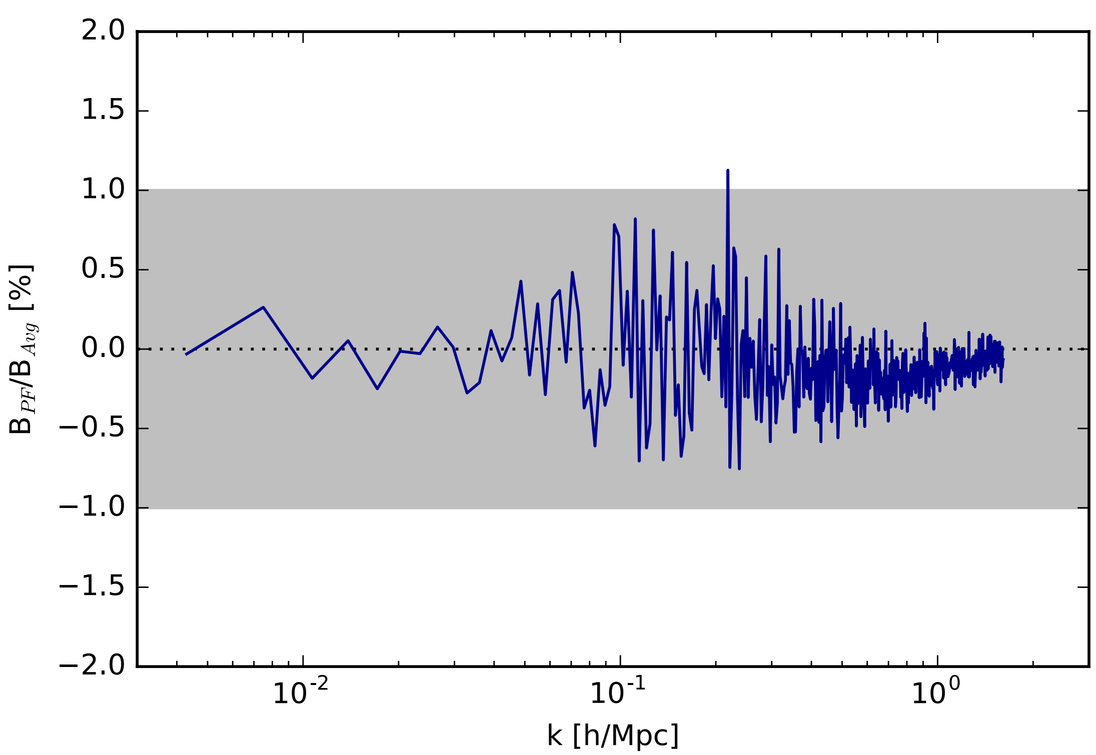
credits: Angulo & Pontzen (2016)

Probability distribution of Gaussian random field:

$$\mathrm{Pr}_g\left(\left|\delta_i^{\mathrm{L}}\right|, \theta_i\right) \equiv \prod_i \frac{1}{2\pi}\sqrt{\frac{2}{\pi P_i}}\exp\left(-\frac{\left|\delta_i^{\mathrm{L}}\right|^2}{2P_i}\right),$$

``Fixed” probability distribution:

$$\mathrm{Pr}_f\left(\left|\delta_i^{\mathrm{L}}\right|, \theta_i\right) \equiv \prod_i \frac{1}{2\pi}\delta_D\left(\left|\delta_i^{\mathrm{L}}\right| - \sqrt{P_i}\right)$$



Step 1: For each simulation we run two realisations based on two different ICs (one using δ_i^L and one using $\delta_i^L + \pi$).

Step 2: Average the resulting power spectra

CONCLUSION:
The average of a pair of fixed IC simulations corresponds to the average of 10 random IC simulations.

Emulation: Strategy & Software

Emulation means finding a surrogate for the model underlying a data set

There are two main families of emulators

Interpolation

Done by Heitmann et al. (cf Heitmann et al 2010, Heitmann et al 2017)

+ Reproduces input data exactly
- May feature a relatively large global error

Regression

Done by us

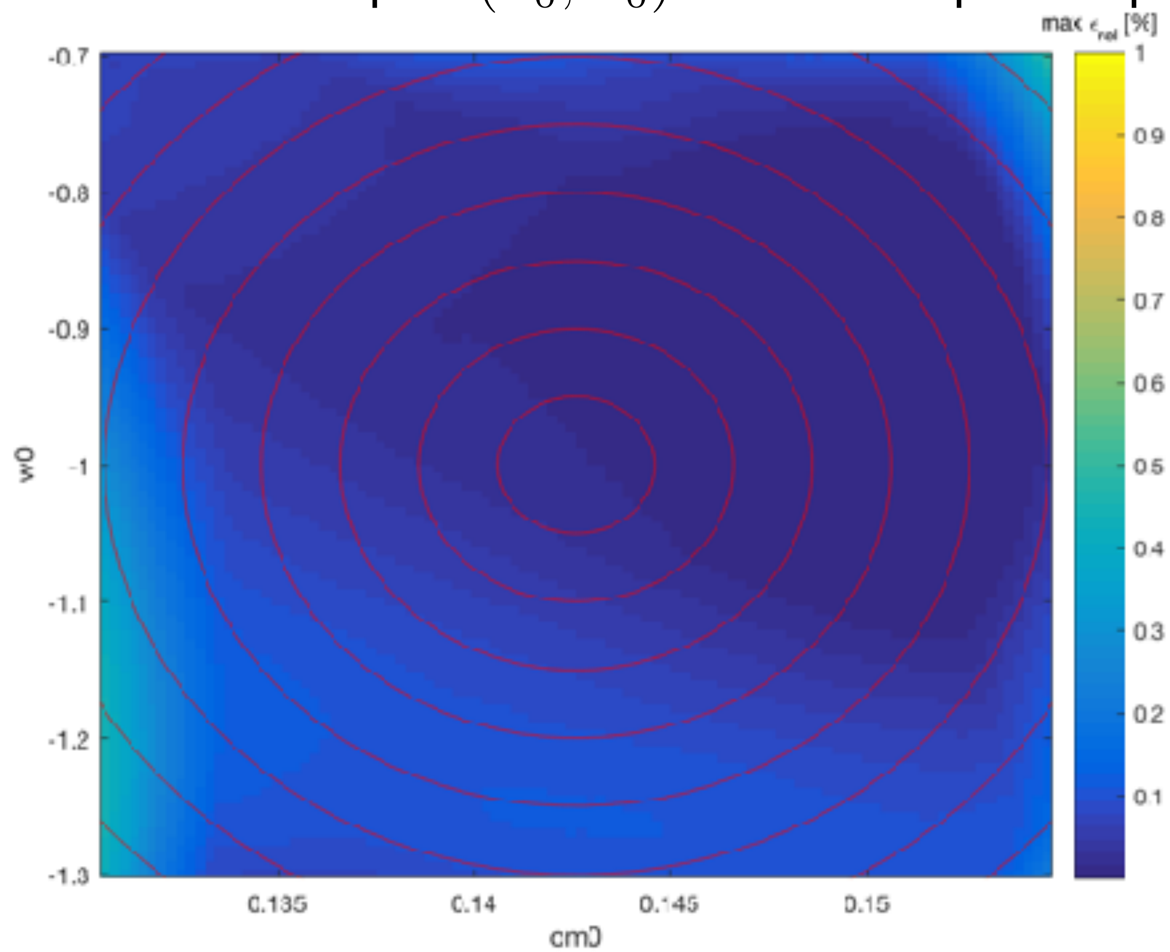
+ Global error can be small
- Does generally not reproduce input data

Our emulator is constructed using UQLab

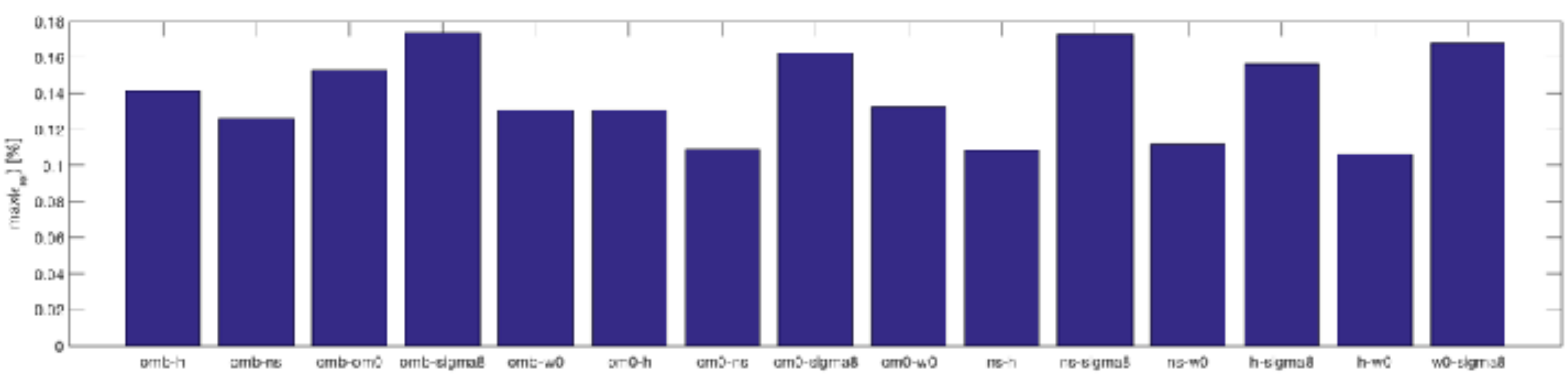
UQLab is an open source uncertainty quantification, reliability and sensitivity analysis software based on Matlab developed by B. Sudret et al at ETH, cf. www.uqlab.com

Emulation Error Prediction (based on Halofit)

Relative error map of (ω_0, w_0) coordinate plane up to 9σ



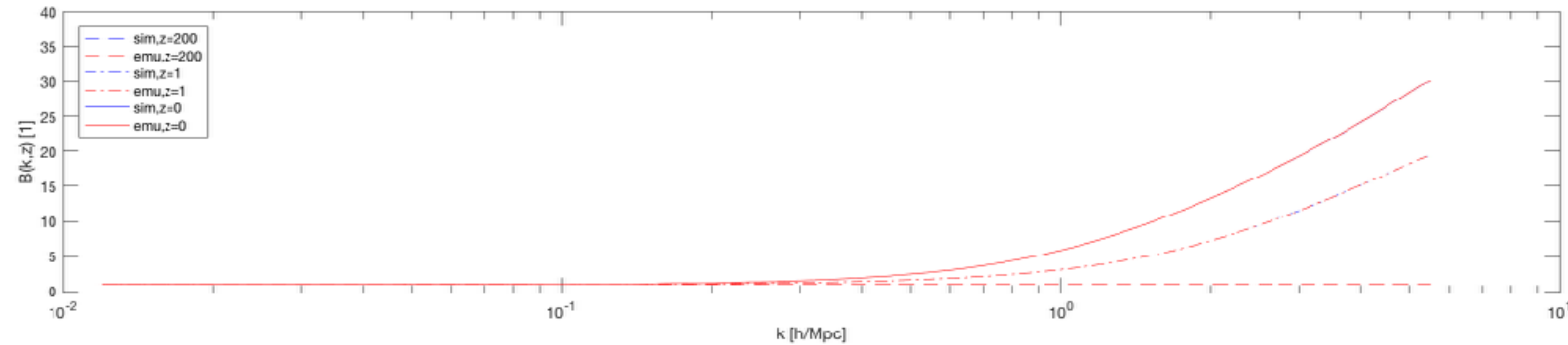
Maximal relative error in all coordinate planes up to 5σ

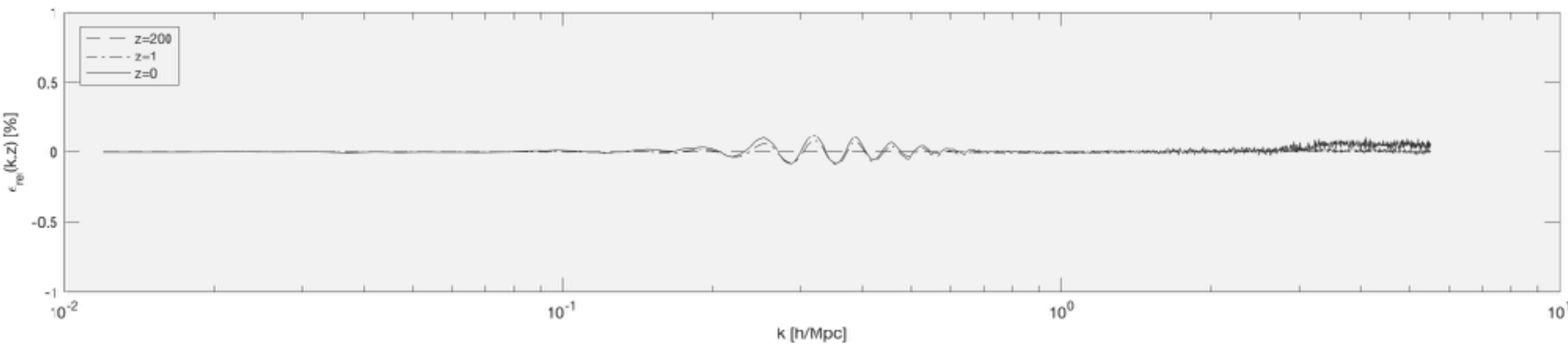


No outliers!

RESULTS

End-to-End Test using Euclid Reference Cosmology





The Euclid Emulator meets the requirements:

1) emulated boosts are within predicted region (shaded area)

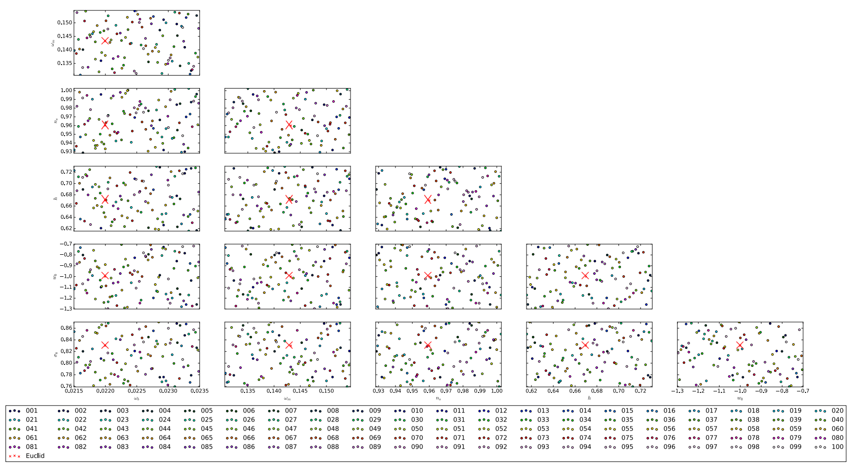
=> **accuracy: dominated by simulation errors**

2) boosts are emulated within less than 0.02 second

=> **efficiency: check!**

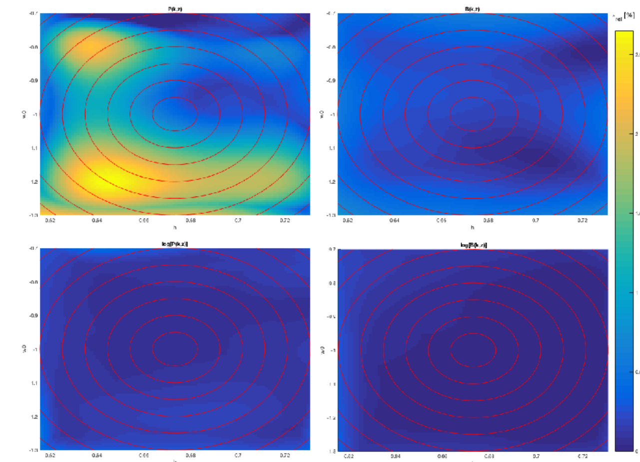
Requirements for Fast and Accurate Power Spectrum Estimation

1) <u>Size of experimental design:</u>



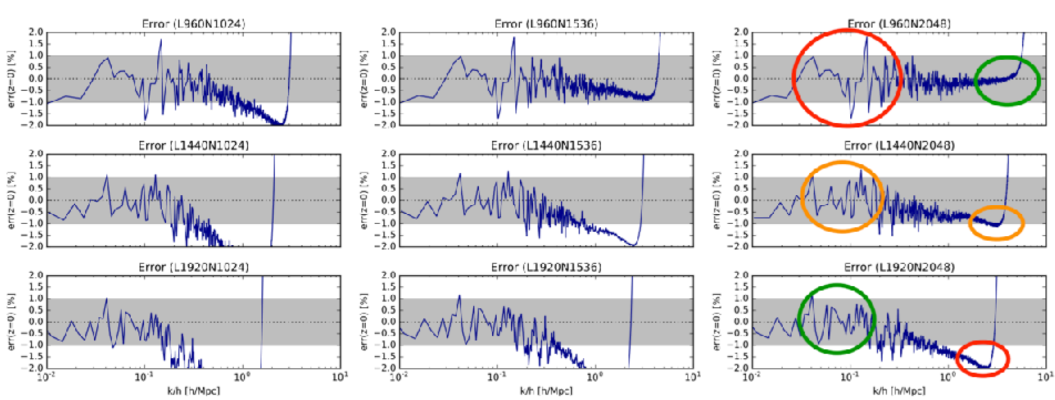
≤ 100 simulated data points

2) <u>Data preprocessing:</u>



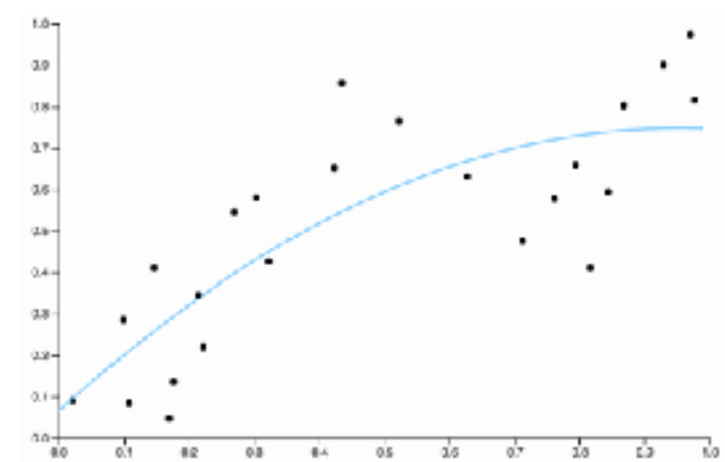
Log of boost factor

3) <u>Quality of experimental design/Convergence:</u>



L ≥ 1250 Mpc/h, N/L ≥ 1.6
Pairing & Fixing

4) <u>Choice of emulation strategy:</u>



Regression

Future Projects

- construct a Euclid Emulator with higher resolution input simulation (R~2)
- include neutrinos into parameter space
- Extend emulator from power spectra to other observables
  (e.g. covariance, halo mass functions, particle light cones)

Stay tuned…

… the Euclid Emulator will be published later this year

BACK UP

The Cosmological Parameter Space

- based on Planck2015 results with most conservative error bars
- parameter ranges are given by $\mu \pm 6\sigma$ (only for ω_b we use the range used by Heitmann et al.)

$$\omega_b \in [0.0215, 0.0235]$$
$$\omega_m \in [0.1306, 0.1546]$$
$$n_s \in [0.9283, 1.0027]$$
$$h \in [0.6155, 0.7307]$$
$$w_0 \in [-1.279, -0.751]$$
$$\sigma_8 \in [0.7591, 0.8707]$$

<u>Sampling strategy:</u>

Step 1: 10^5 latin hypercube sampling (LHS) realizations of N=100 data points

Step 2: Maximize the minimal distance between the data points

Requirements for Fast and Accurate Power Spectrum Estimation

1) <u>Size of experimental design:</u>

    How many simulated data points?

2) <u>Quality of experimental design/Convergence:</u>

    What simulation box size and mass resolution?
    ``Tricks” to reduce numerical artefacts

3) <u>Data preprocessing:</u>

    Power spectrum, boost factor or yet another quantity?
    How many included principle components?

4) <u>Choice of emulation strategy:</u>

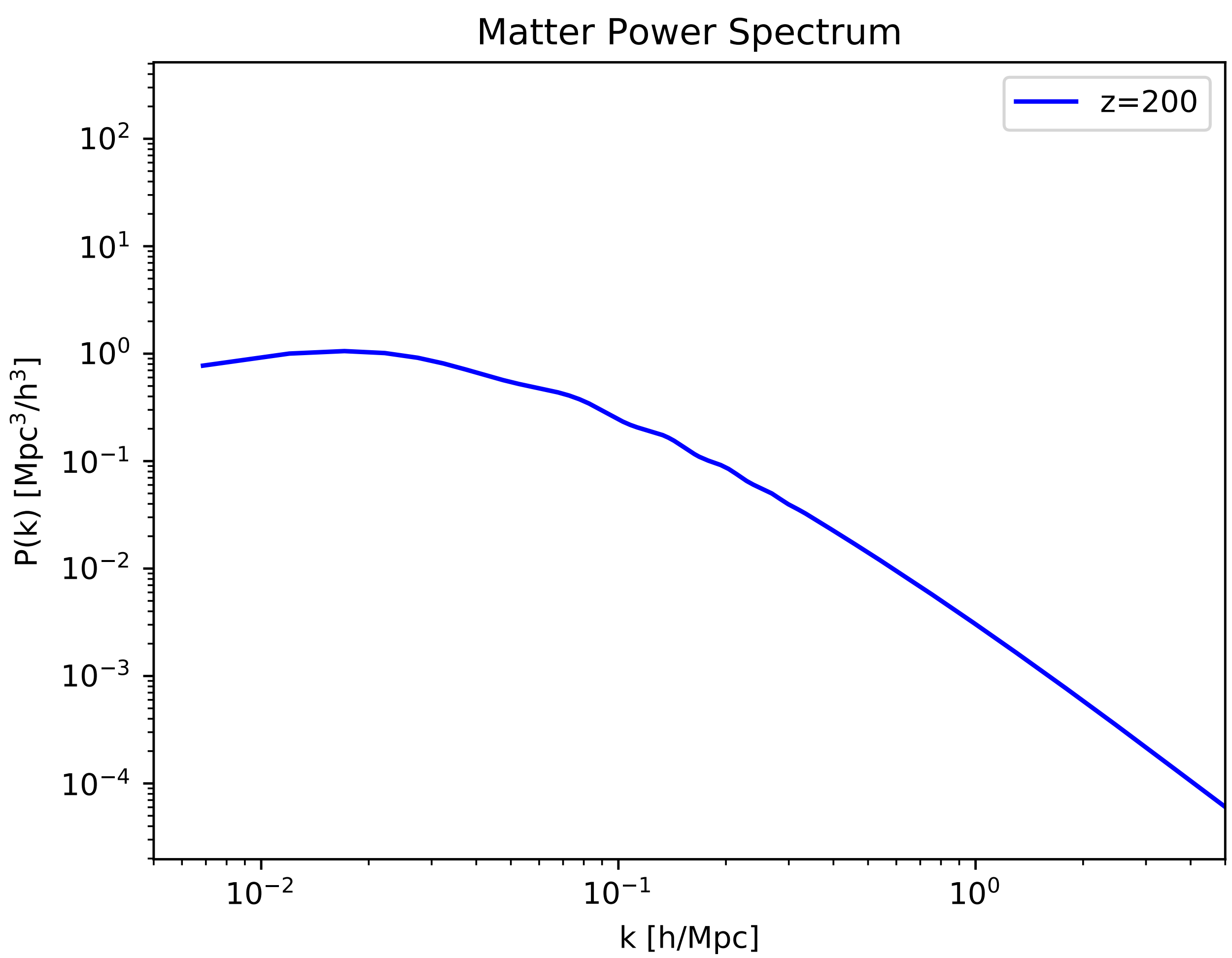
    Interpolation or regression?

5) <u>Choice of emulation parameters:</u>

    In our case: Where to truncate expansion series?

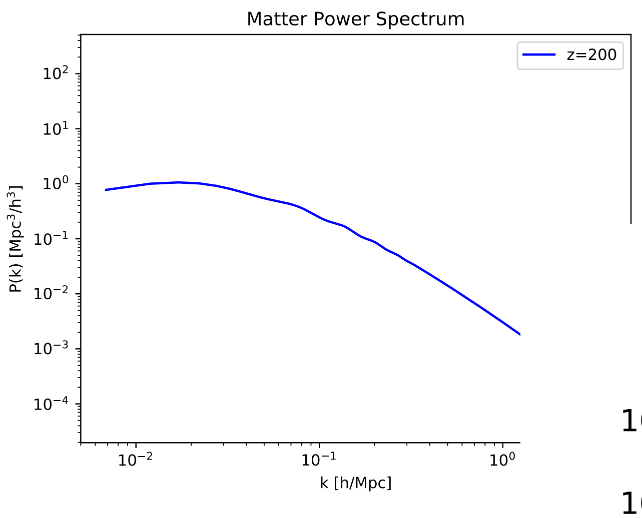
Input Simulations - Initial Power Spectrum

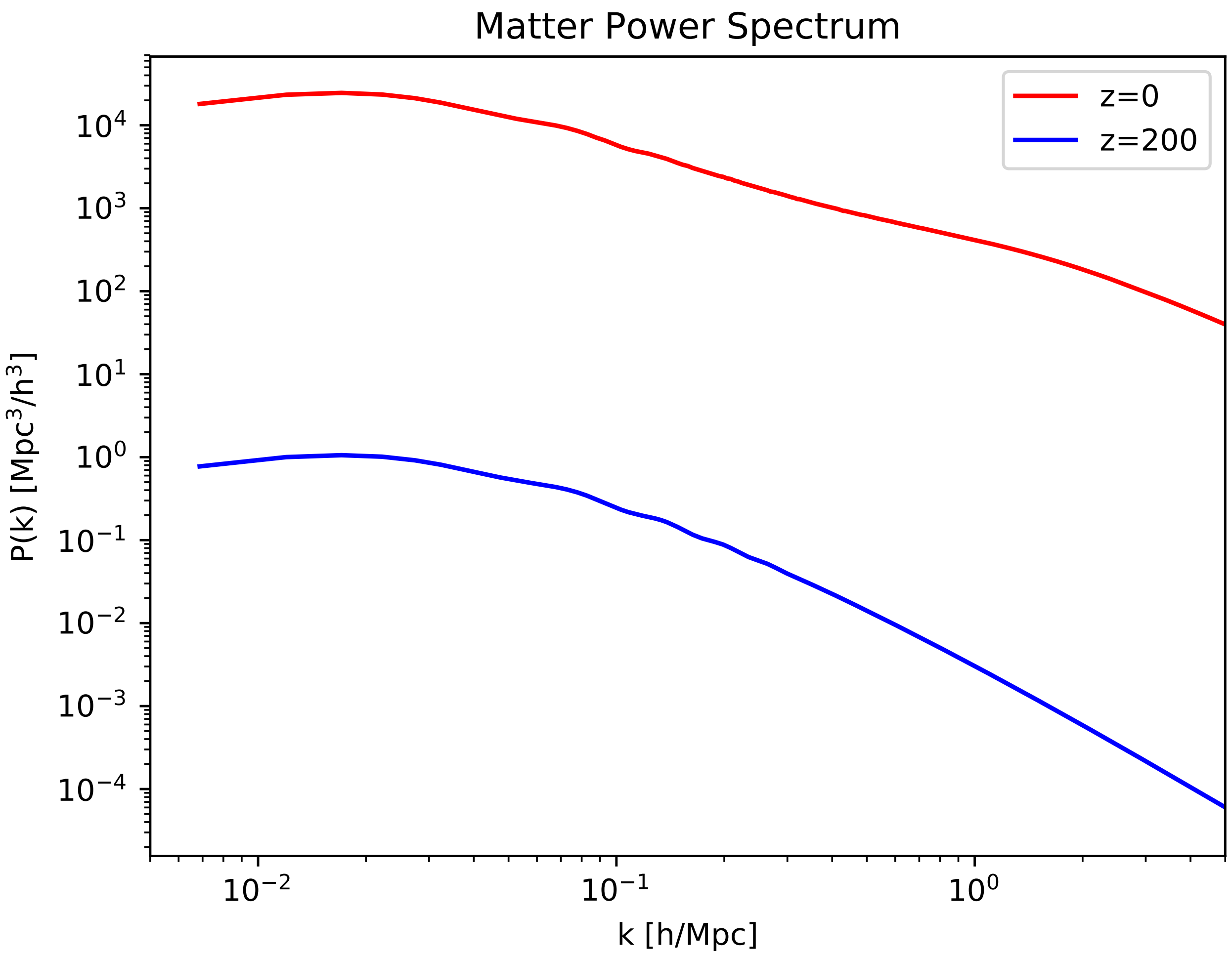
Preliminaries: Compute transfer function at z=0 and scale it back to initial redshift (here z=200)
This will lead to accurate results at low redshifts and wrong results at high redshifts



Step 1: Generate ICs with Class and measure initial P(k) with PKDGRAV (by J. Stadel & D. Potter)

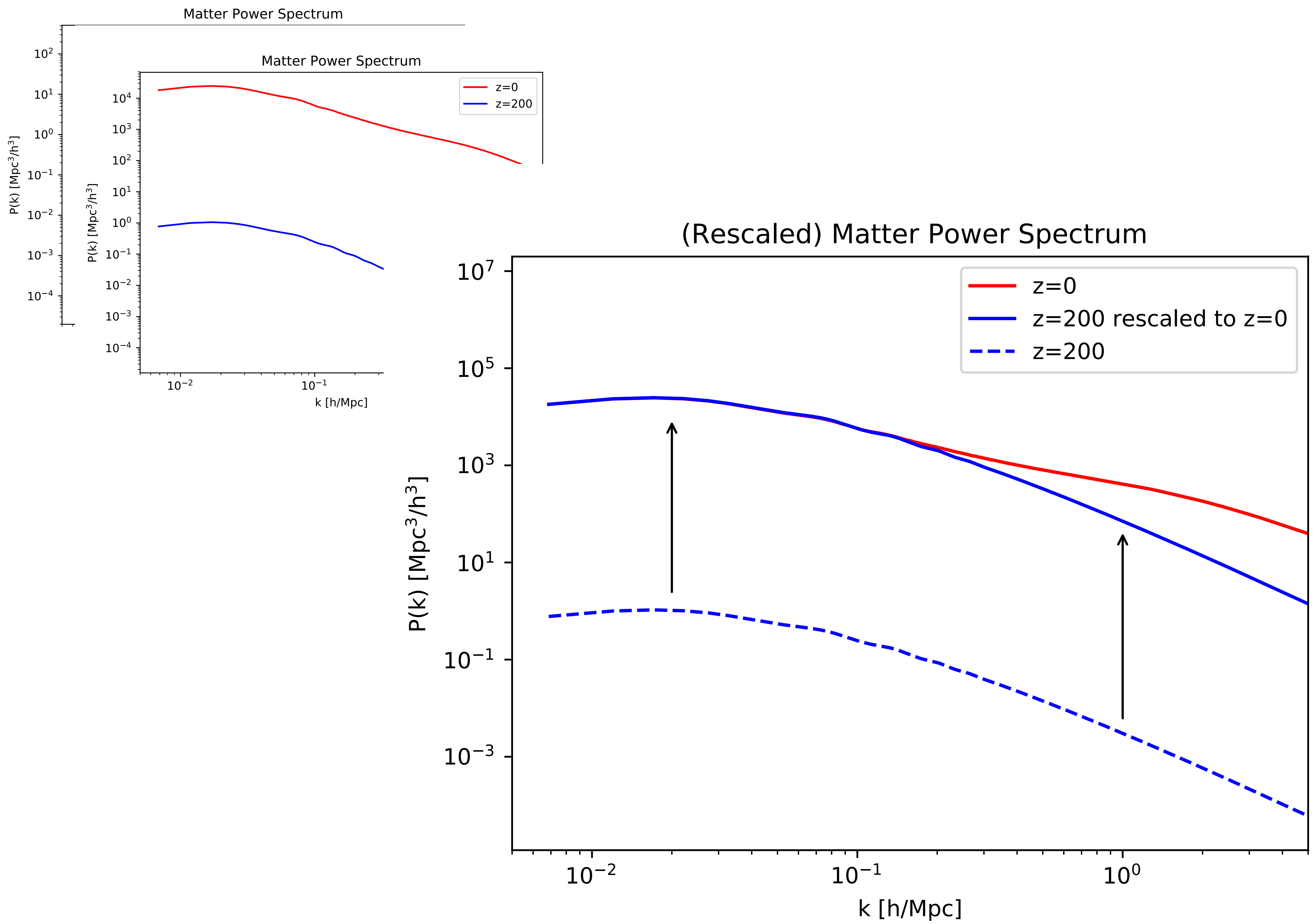
Input Simulations - Final Power Spectrum





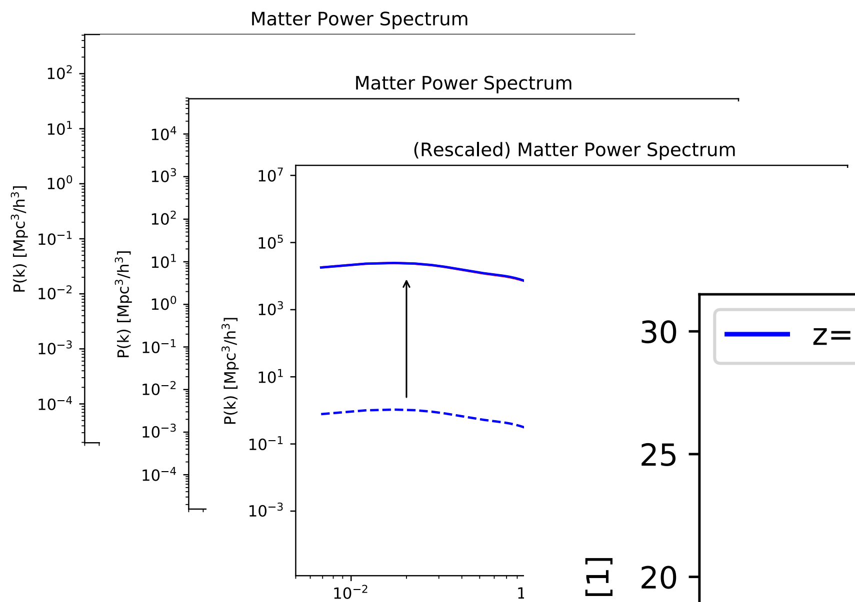
Step 2: N-body simulation/evolution to lower redshifts and P(k) measurement with PKDGRAV3

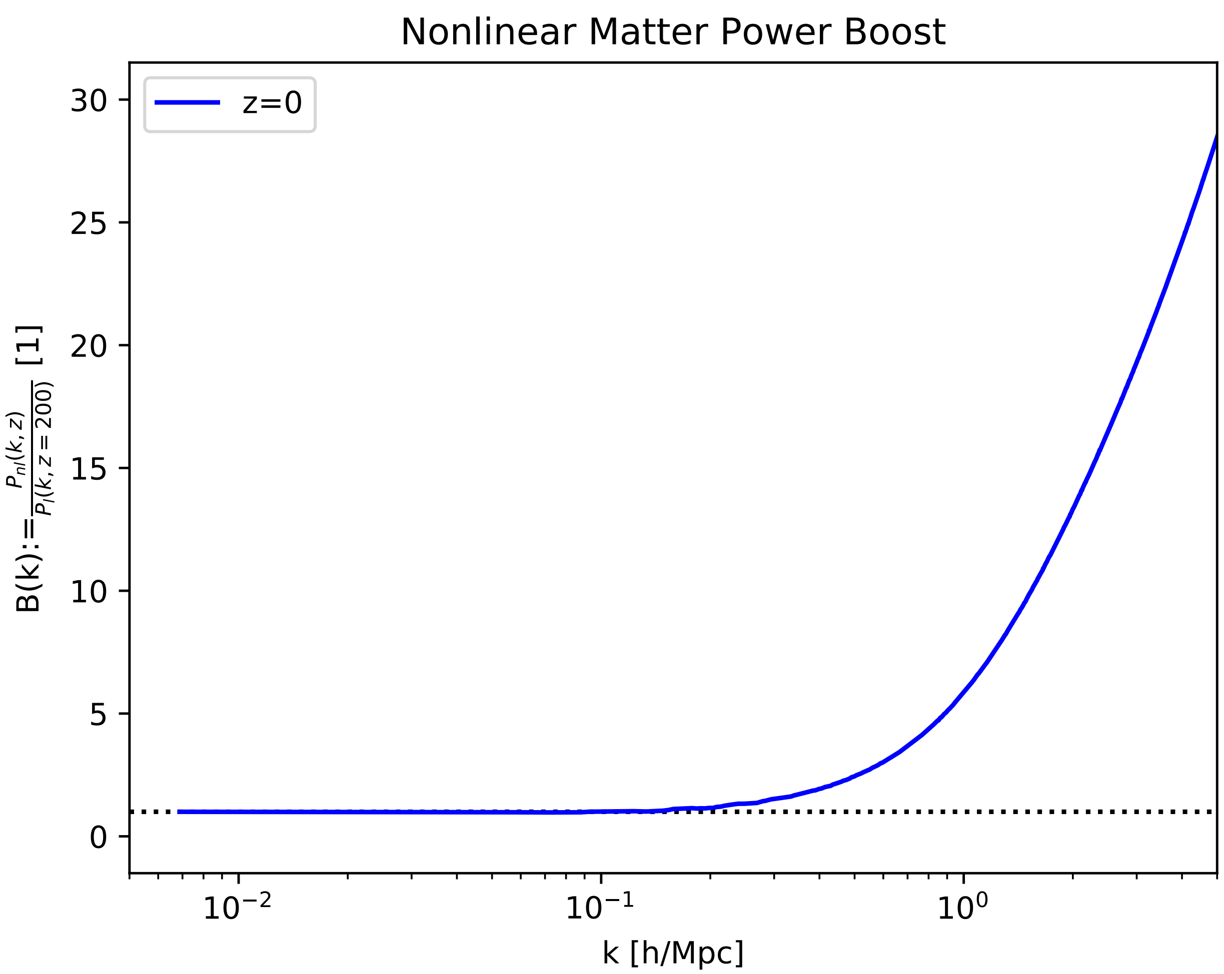
Input Simulations - Rescaling



Step 3: Rescale initial power spectrum to redshift of interest (using linear growth factor $D_1(z)$)

Input Simulations - The Boost Factor

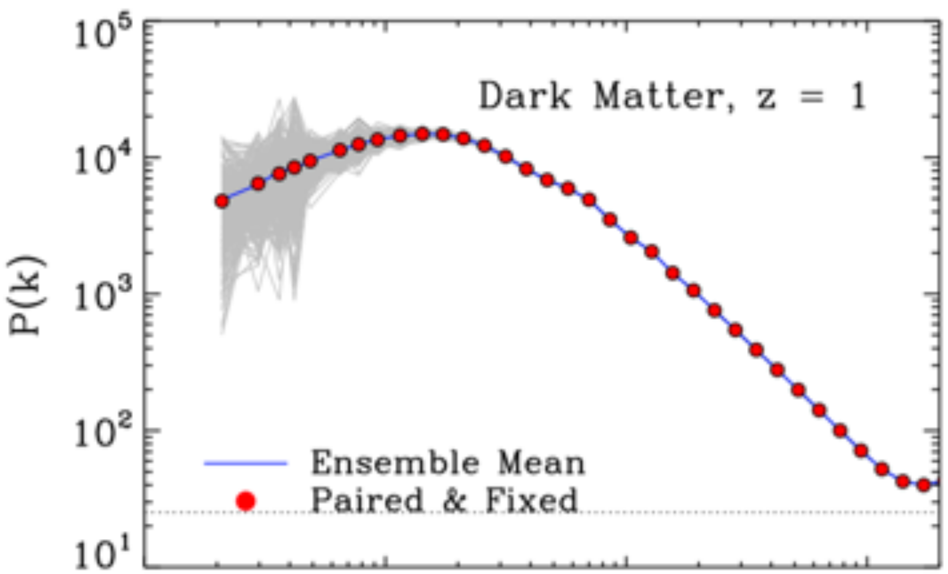




Step 4: Compute the non-linear boost $B(k,z) := \frac{P_{non-linear}(k,z)}{P_{linear}(k,z)}$

Emulation Strategy - Sparse Polynomial Chaos Expansion (SPCE)

Improving Quality of Input Simulations - ``Pairing & Fixing“ (PF)



credits: Angulo & Pontzen (2016)

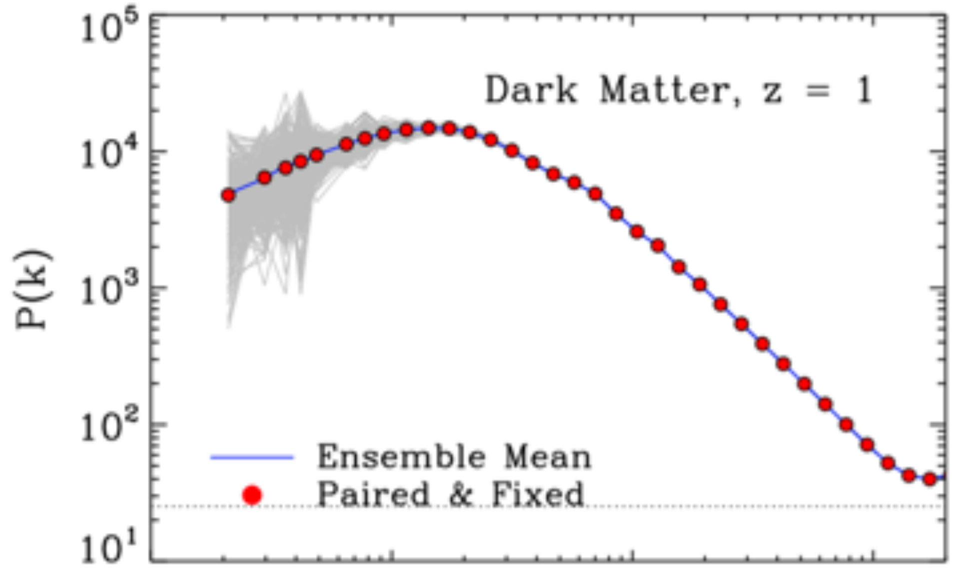
Standard probability distribution:

$$\Pr_g\left(\left|\delta_i^{\rm L}\right|, \theta_i\right) \equiv \prod_i \frac{1}{2\pi}\sqrt{\frac{2}{\pi P_i}}\exp\left(-\frac{\left|\delta_i^{\rm L}\right|^2}{2P_i}\right),$$

``Fixed” probability distribution:

$$\Pr_f\left(\left|\delta_i^{\rm L}\right|, \theta_i\right) \equiv \prod_i \frac{1}{2\pi}\delta_D\left(\left|\delta_i^{\rm L}\right| - \sqrt{P_i}\right)$$

Improving Quality of Input Simulations - ``Pairing & Fixing“ (PF)



credits: Angulo & Pontzen (2016)

Standard probability distribution:

$$\mathrm{Pr}_g\left(\left|\delta_i^{\mathrm{L}}\right|, \theta_i\right) \equiv \prod_i \frac{1}{2\pi}\sqrt{\frac{2}{\pi P_i}}\exp\left(-\frac{\left|\delta_i^{\mathrm{L}}\right|^2}{2P_i}\right),$$

``Fixed” probability distribution:

$$\mathrm{Pr}_f\left(\left|\delta_i^{\mathrm{L}}\right|, \theta_i\right) \equiv \prod_i \frac{1}{2\pi}\delta_D\left(\left|\delta_i^{\mathrm{L}}\right| - \sqrt{P_i}\right)$$

Step 1: For each simulation we run two realisations based on two different ICs (one using δ_i^L and one using $\delta_i^L + \pi$).

Step 2: Average the resulting power spectra