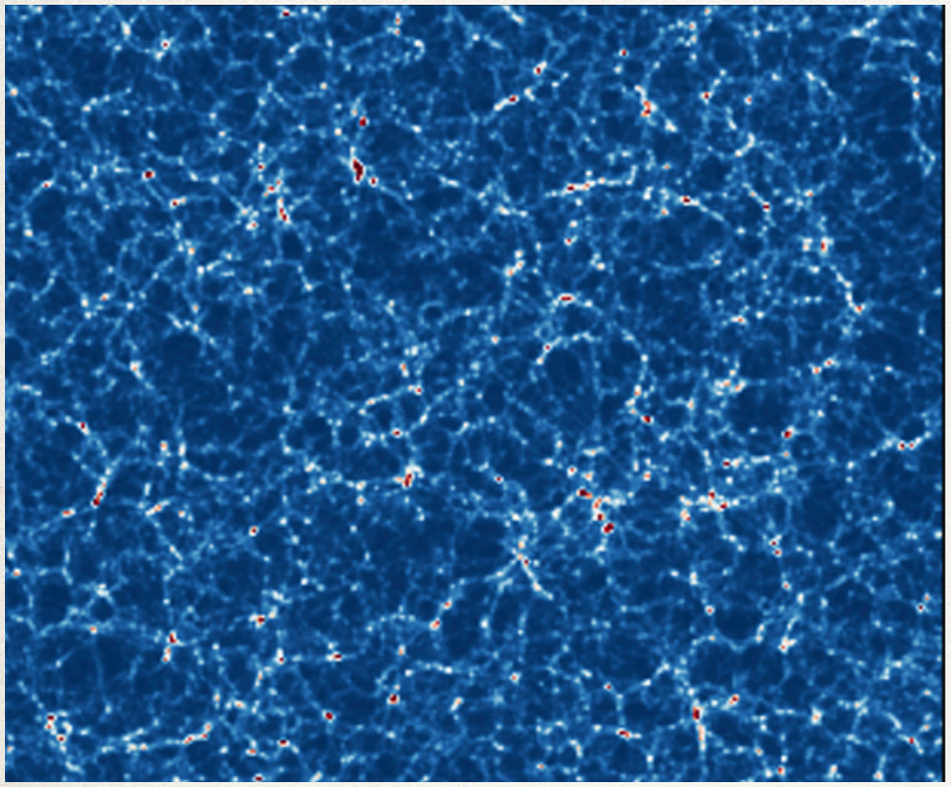
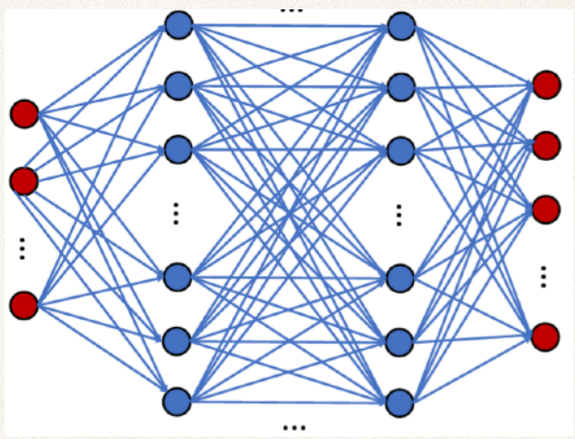
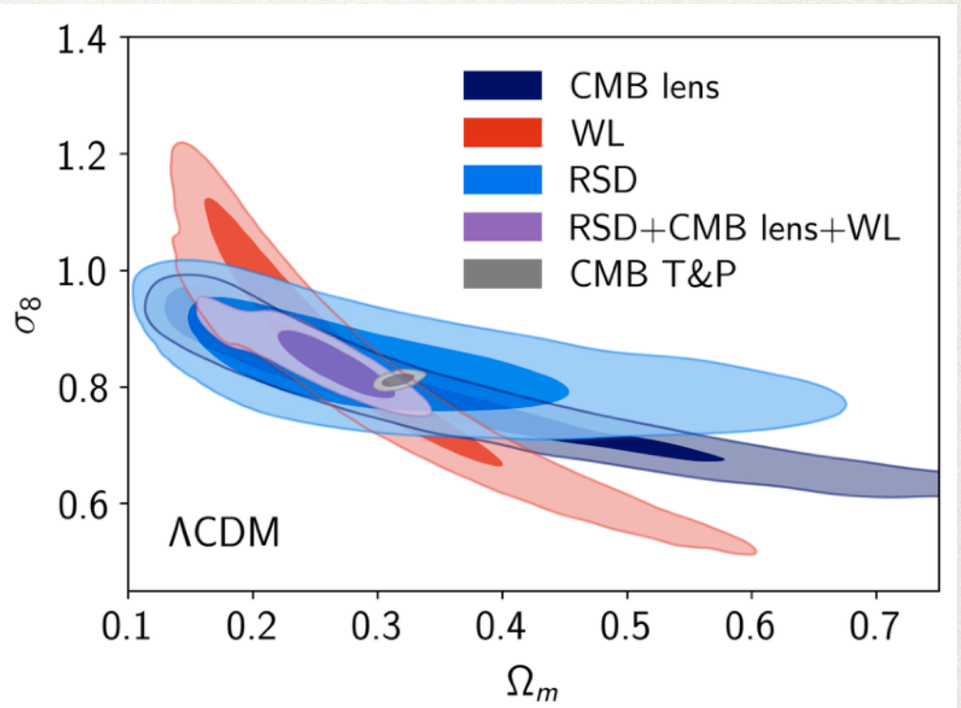






Emulation in Cosmic Frontier Theory

Joe DeRose — Lawrence Berkeley National Laboratory

8/16/23 — SLAC — SSI

BERKELEY LAB

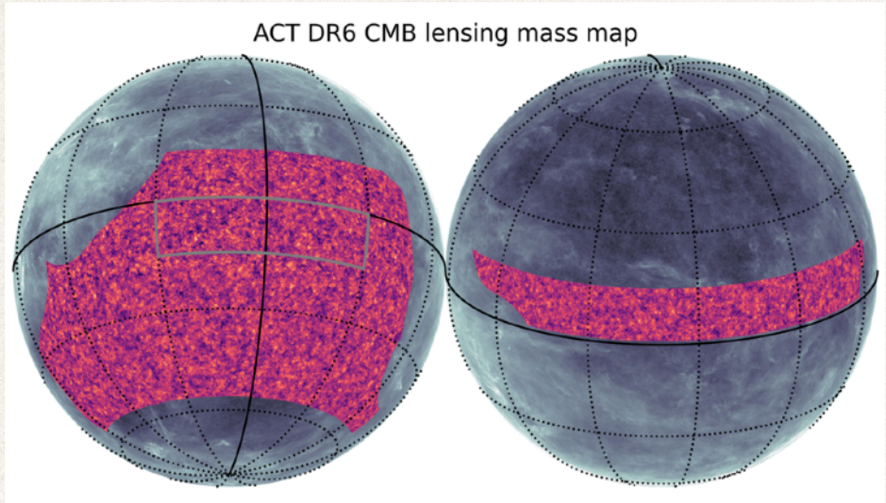
Outline

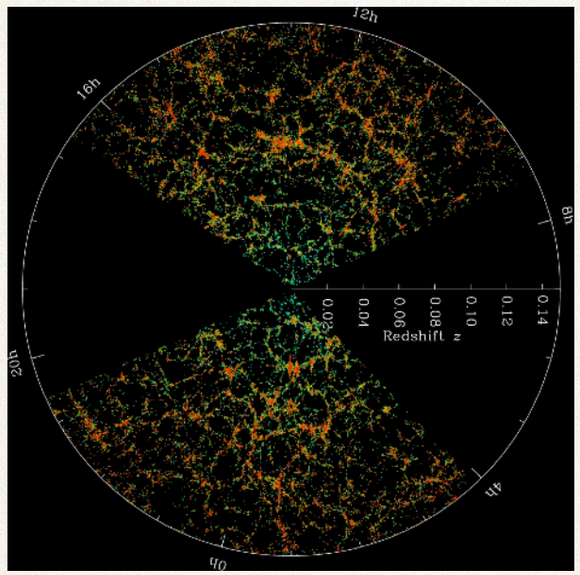
- Computational Challenges in Cosmic Frontier Theory
- Experimental design
  - Parameter space sampling, bayesian optimization, variance-reduction
- Emulation techniques and when to use them
  - Gaussian processes, polynomial chaos cxpansions, and neural networks
- I trained an emulator, now what?

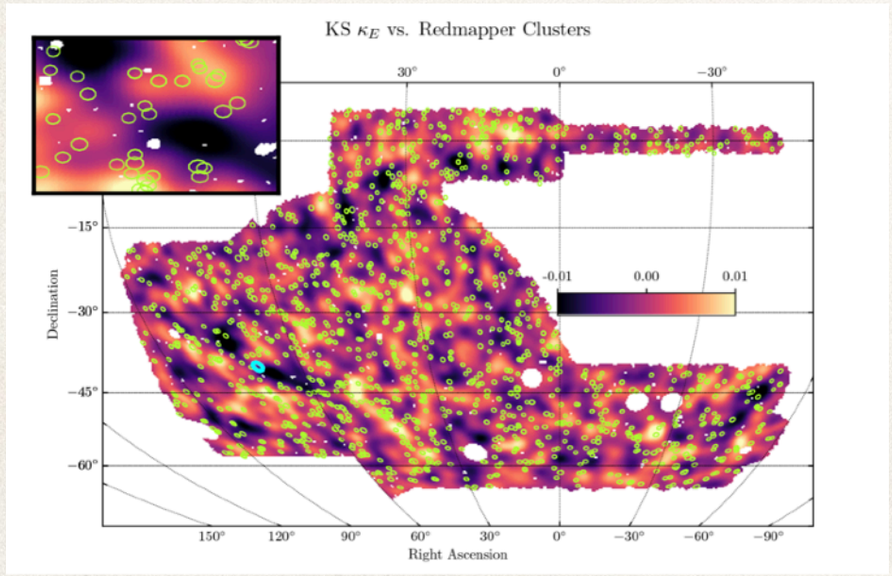
Outline

- Computational Challenges in Cosmic Frontier Theory
- Experimental design
  - Parameter space sampling, bayesian optimization, variance-reduction
- Emulation techniques and when to use them
  - Gaussian Processes, Polynomial Chaos Expansions, and Neural Networks
- I trained an emulator, now what?

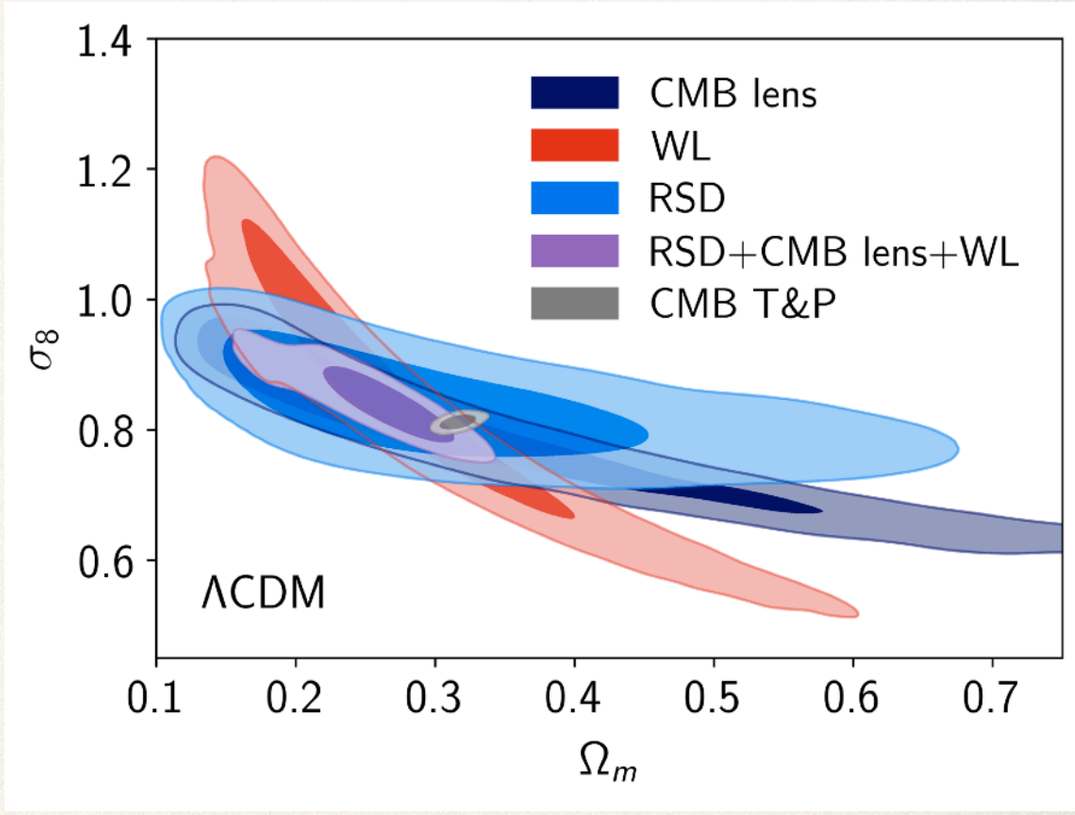
Computational Challenges in CF Theory







(Apologies in advance for the LSS/CMB bias)

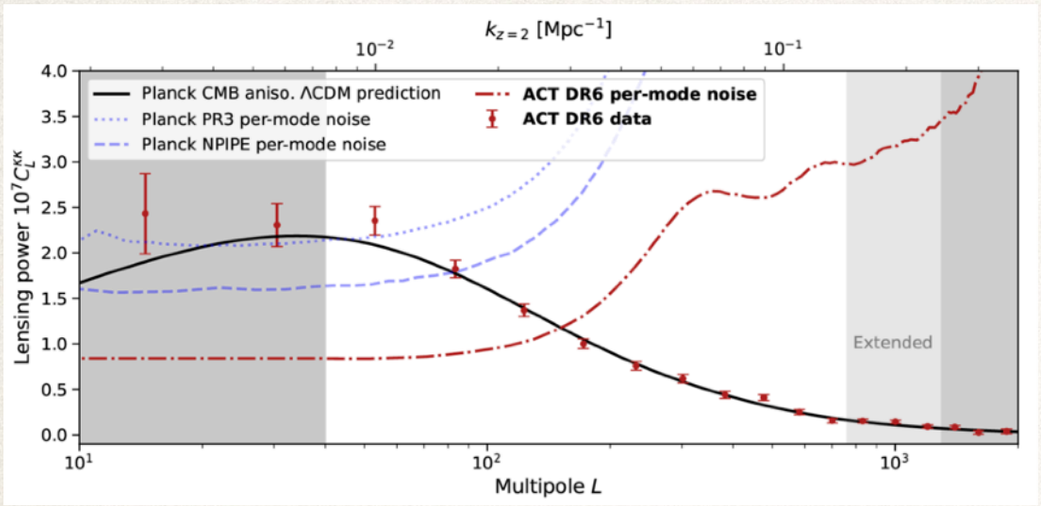


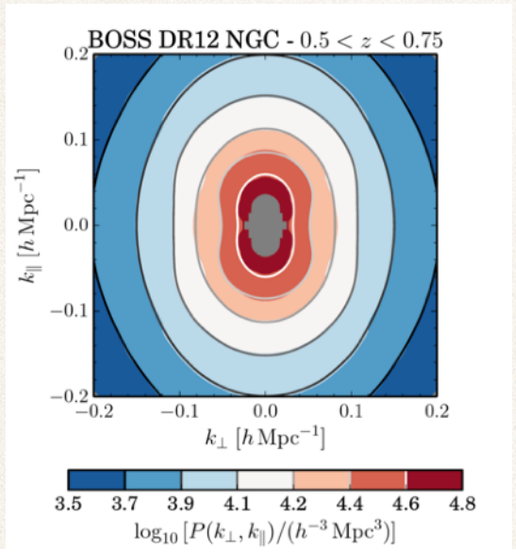
Computational Challenges in CF Theory

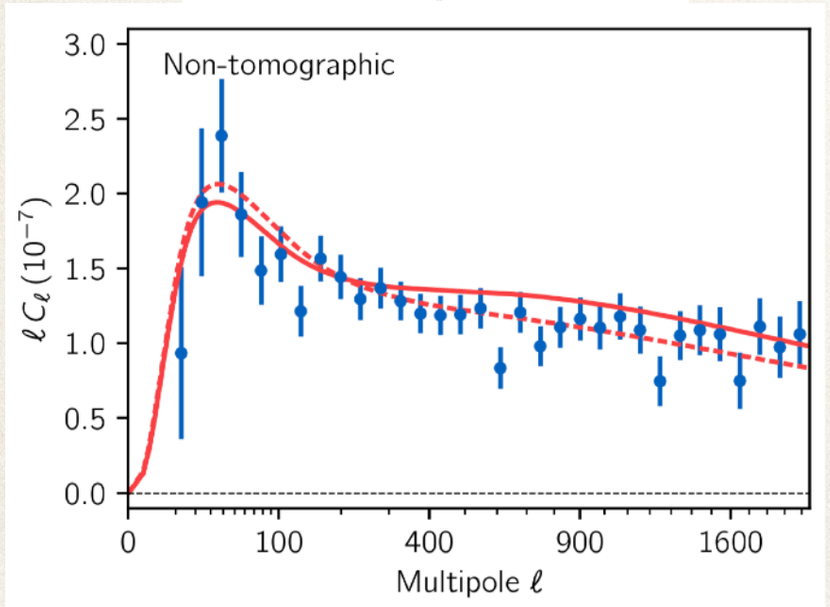
Since we believe large scale structure is sourced by a stochastic process, all inference in cosmology is statistical.

- Positions of individual objects are not predictable, but we can make predictions for their statistical distribution.
- Most analyses use two-point correlations, e.g $\langle \delta(\mathbf{x_1}) \delta(\mathbf{x}_2) \rangle$
  - Equivalently in Fourier space:
    $\langle \delta(\mathbf{k_1}) \delta(\mathbf{k}_2) \rangle = (2\pi)^3 \delta^{(D)}(\mathbf{k}_1 + \mathbf{k}_2) P(k)$
  - Cross power spectrum:
    $\langle \delta_m(\mathbf{k_1}) \delta_g(\mathbf{k}_2) \rangle = (2\pi)^3 \delta^{(D)}(\mathbf{k}_1 + \mathbf{k}_2) P_{gm}(k)$
- Compare measured and predicted correlations to constrain cosmological models.

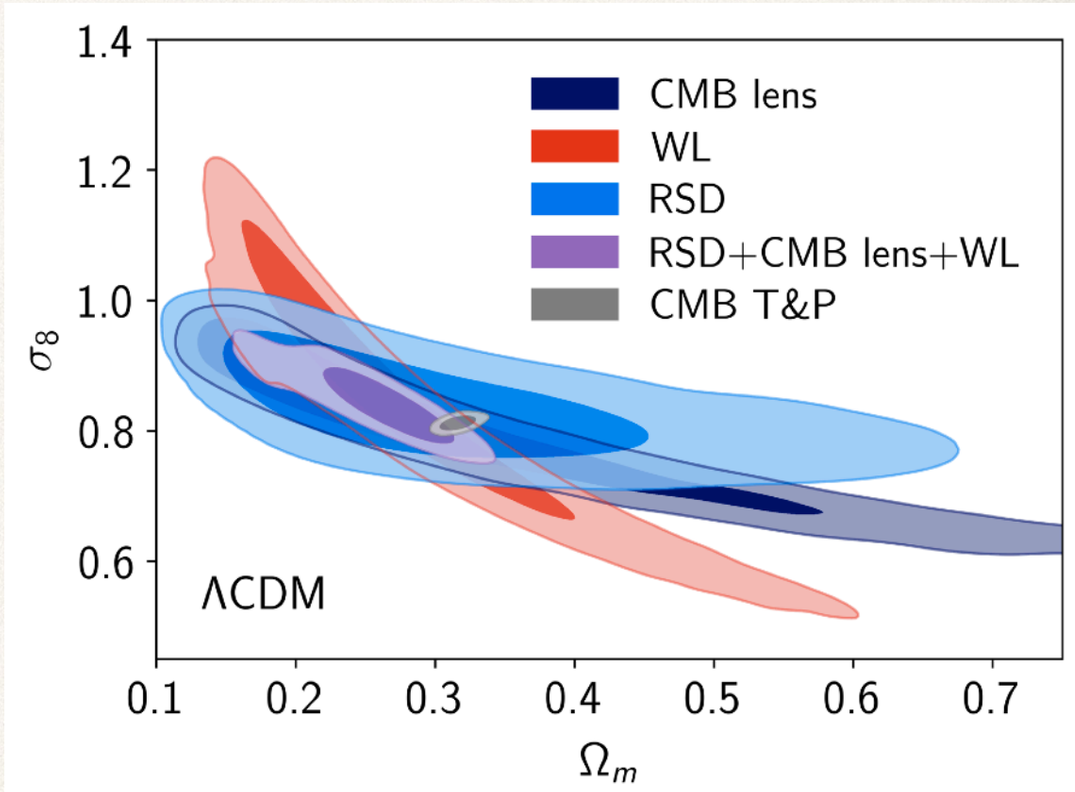
Computational Challenges in CF Theory



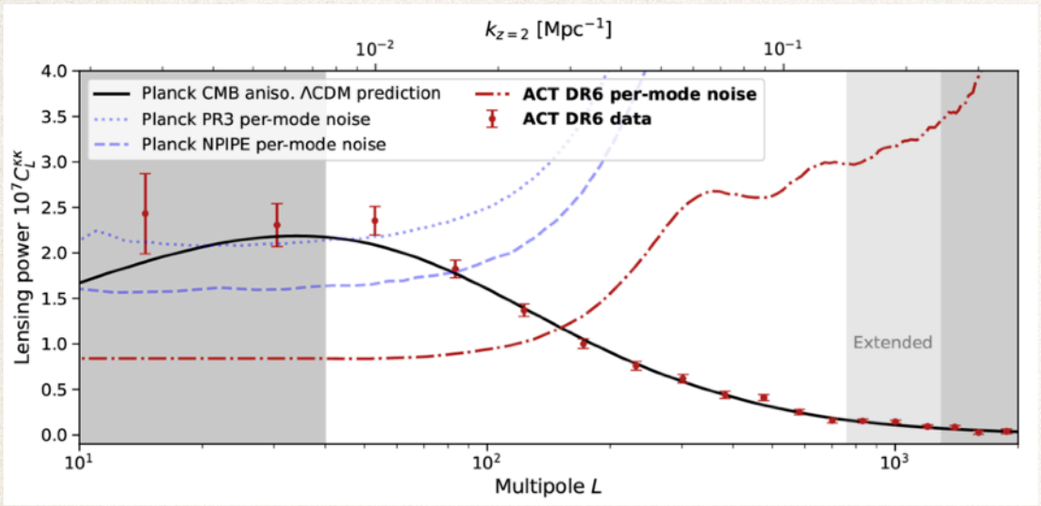


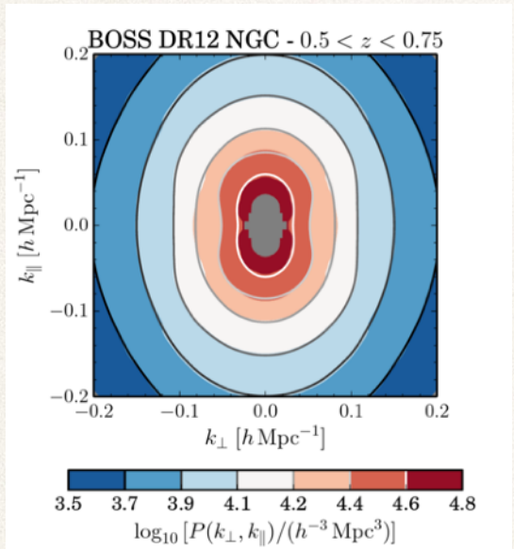


Inference

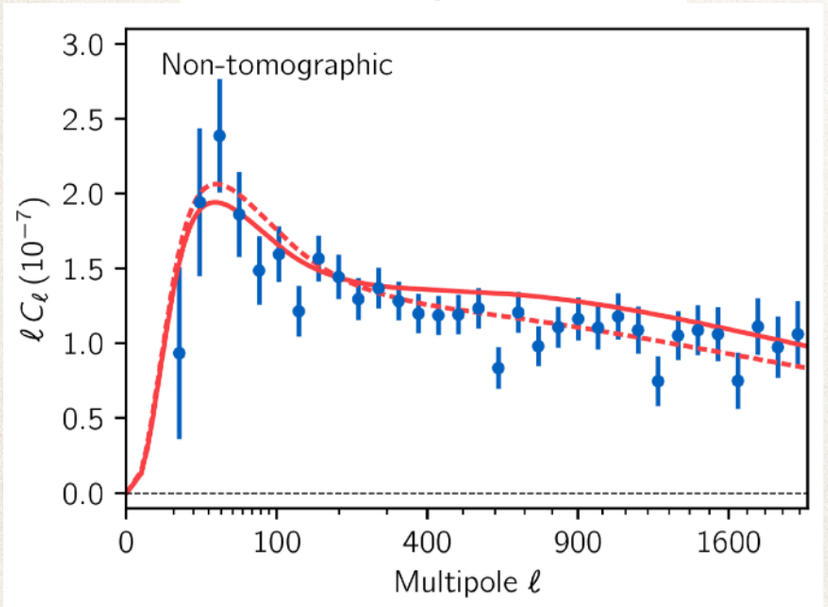


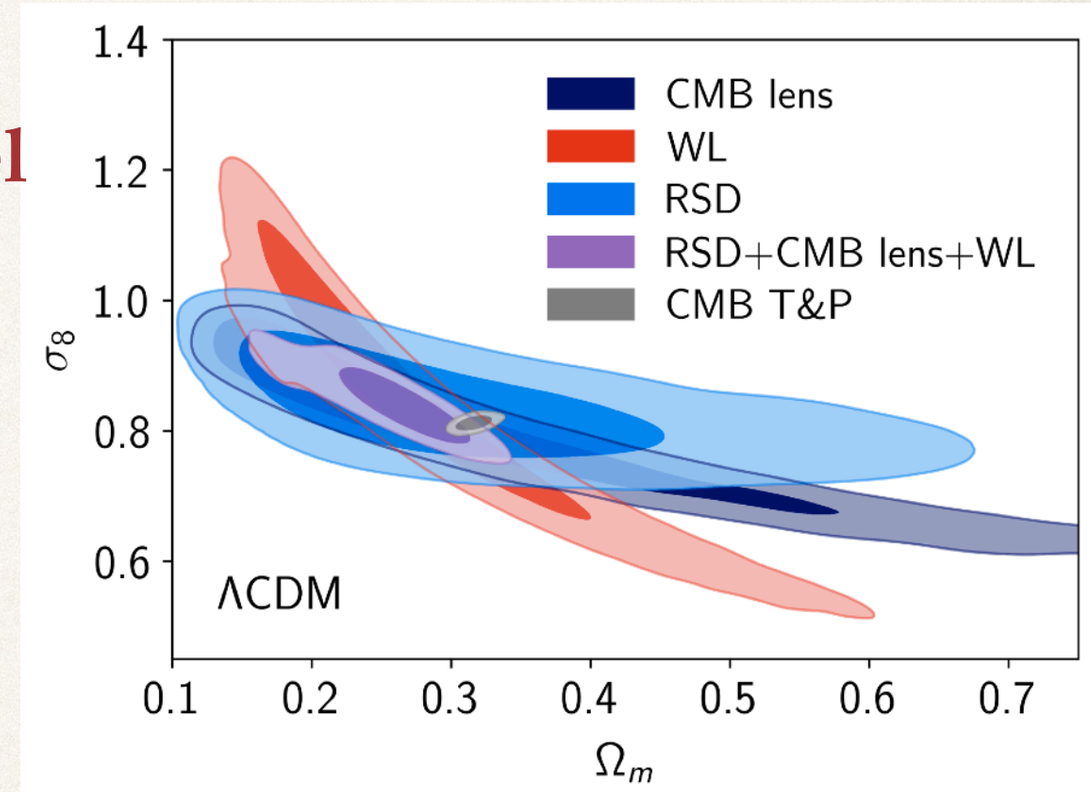
Computational Challenges in CF Theory





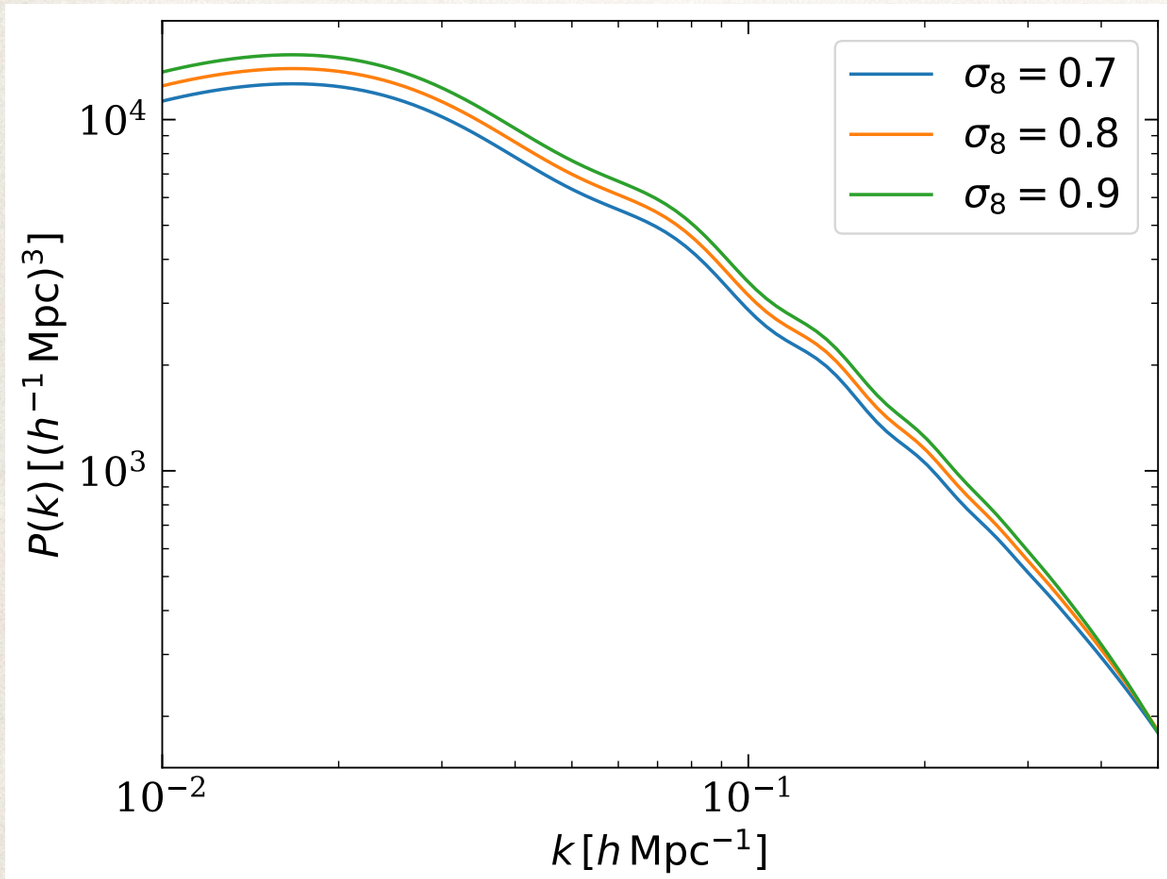
Forward model

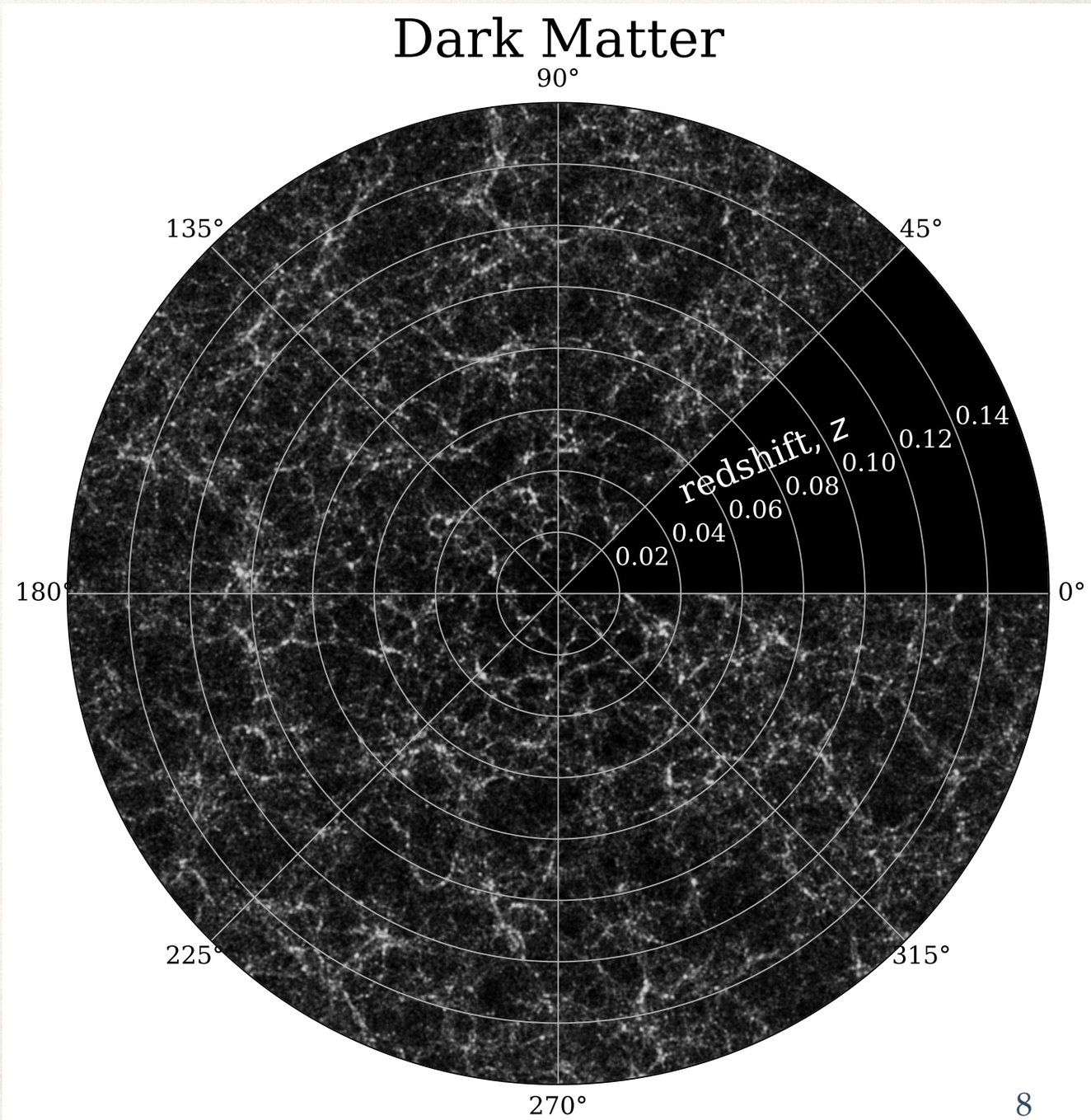




Large Scale Structure Forward Modeling

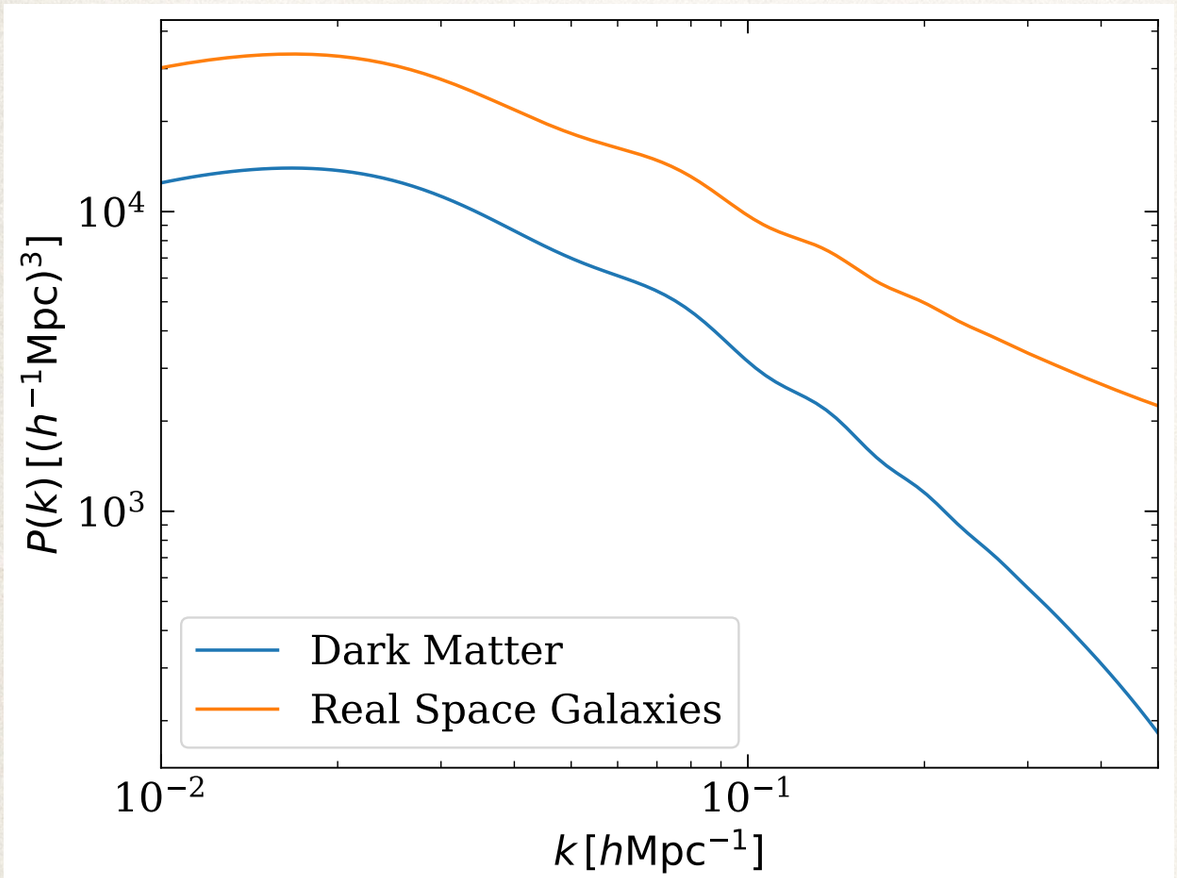
First step is to model the dark matter distribution as a function of relevant cosmological parameters.

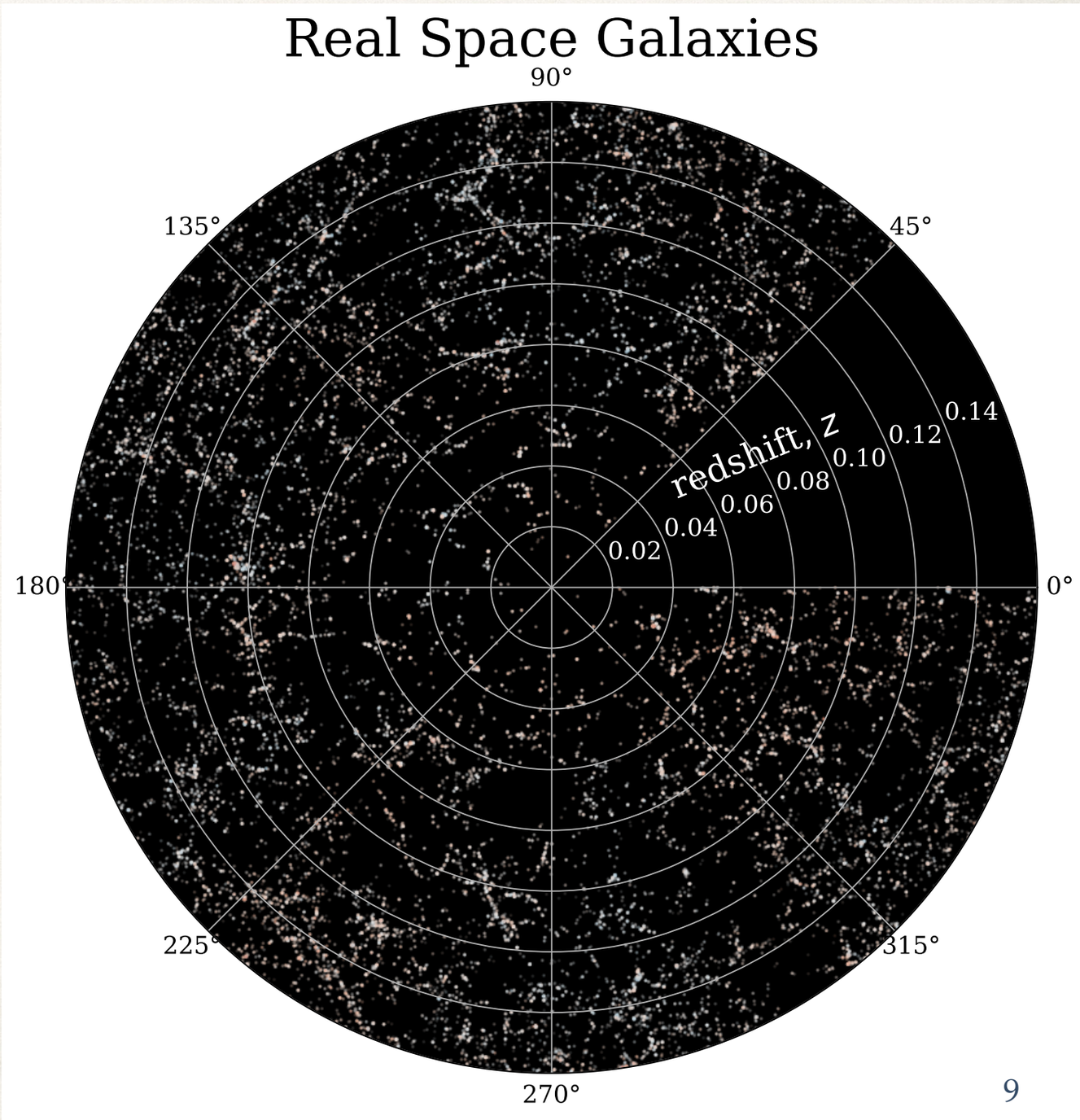




Large Scale Structure Forward Modeling

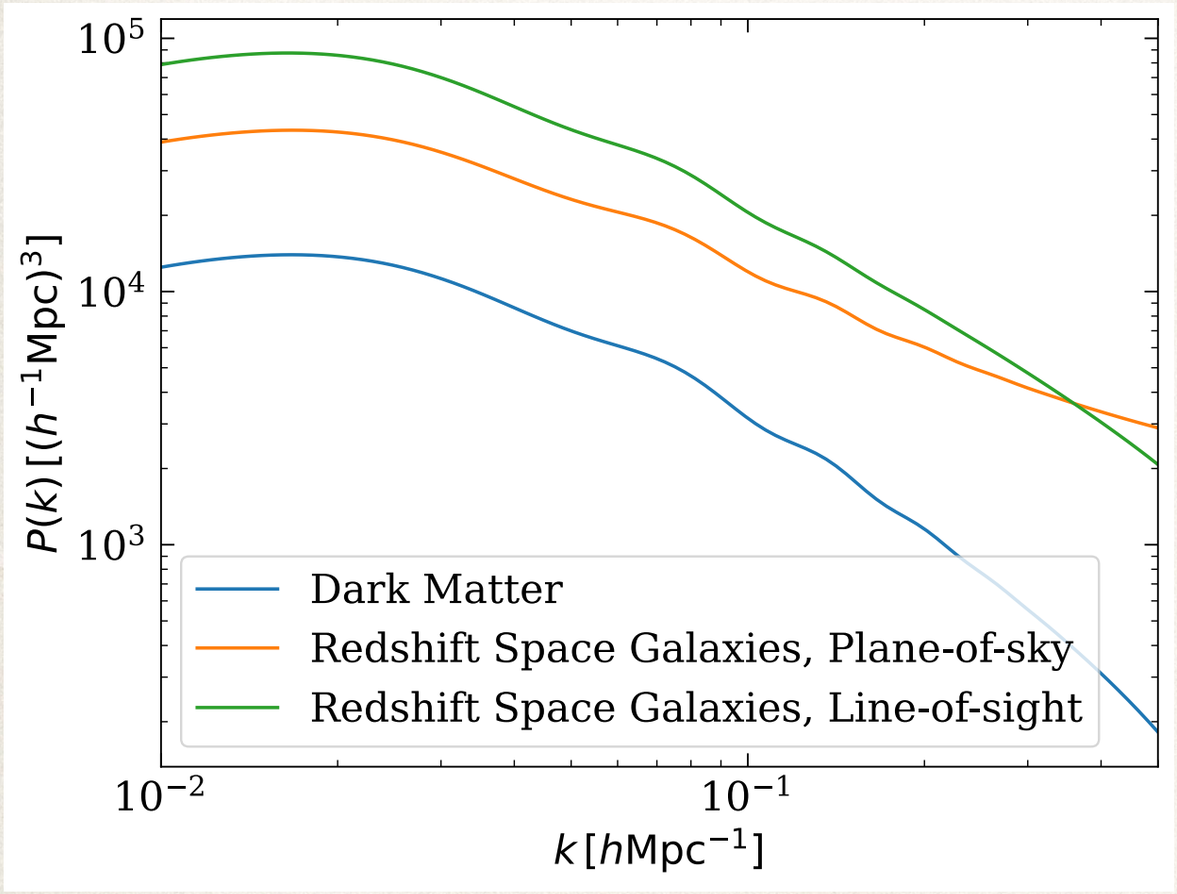
Modeling complicated by fact that galaxies are discrete and biased tracers of the matter field.

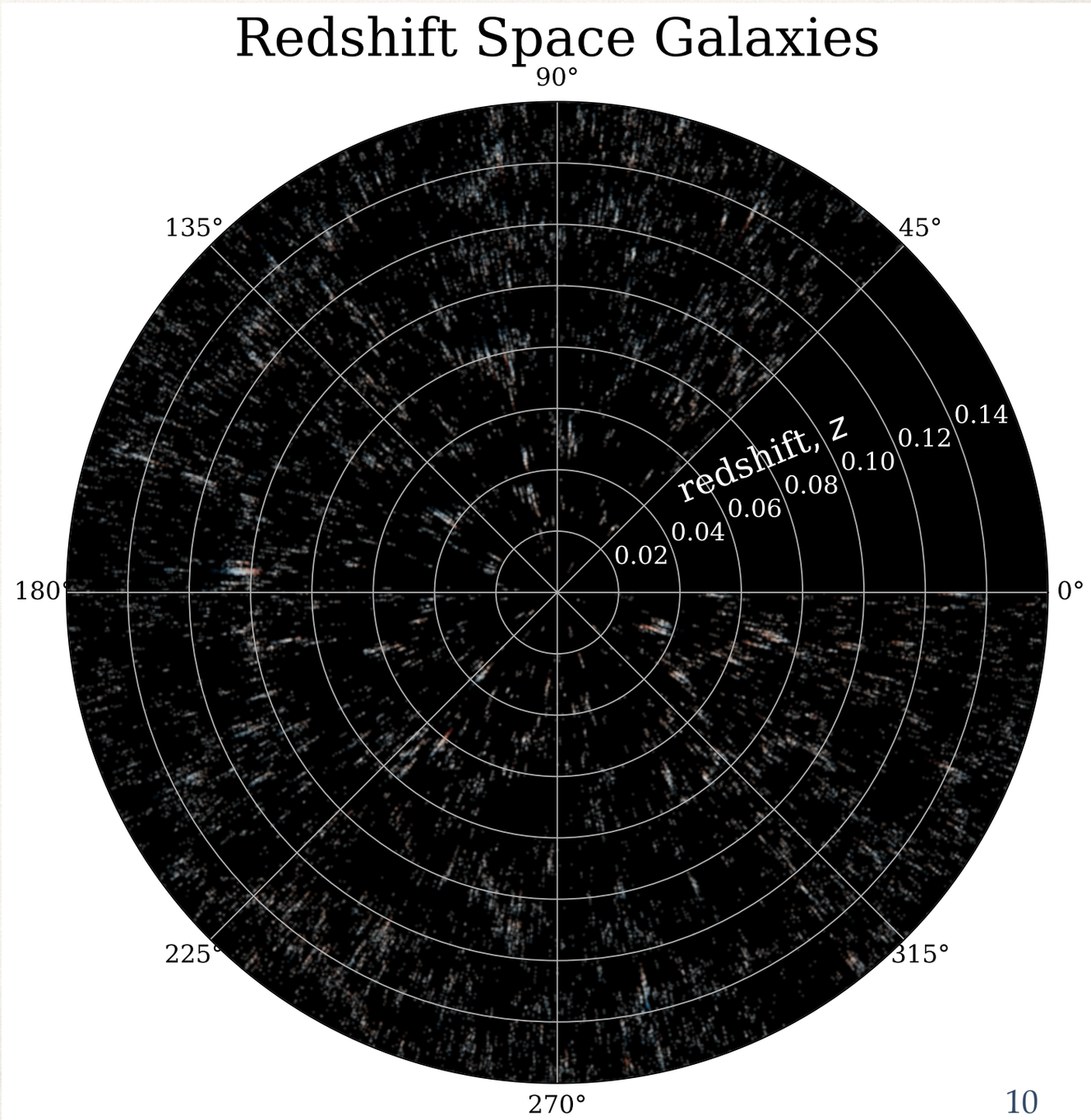




Large Scale Structure Forward Modeling

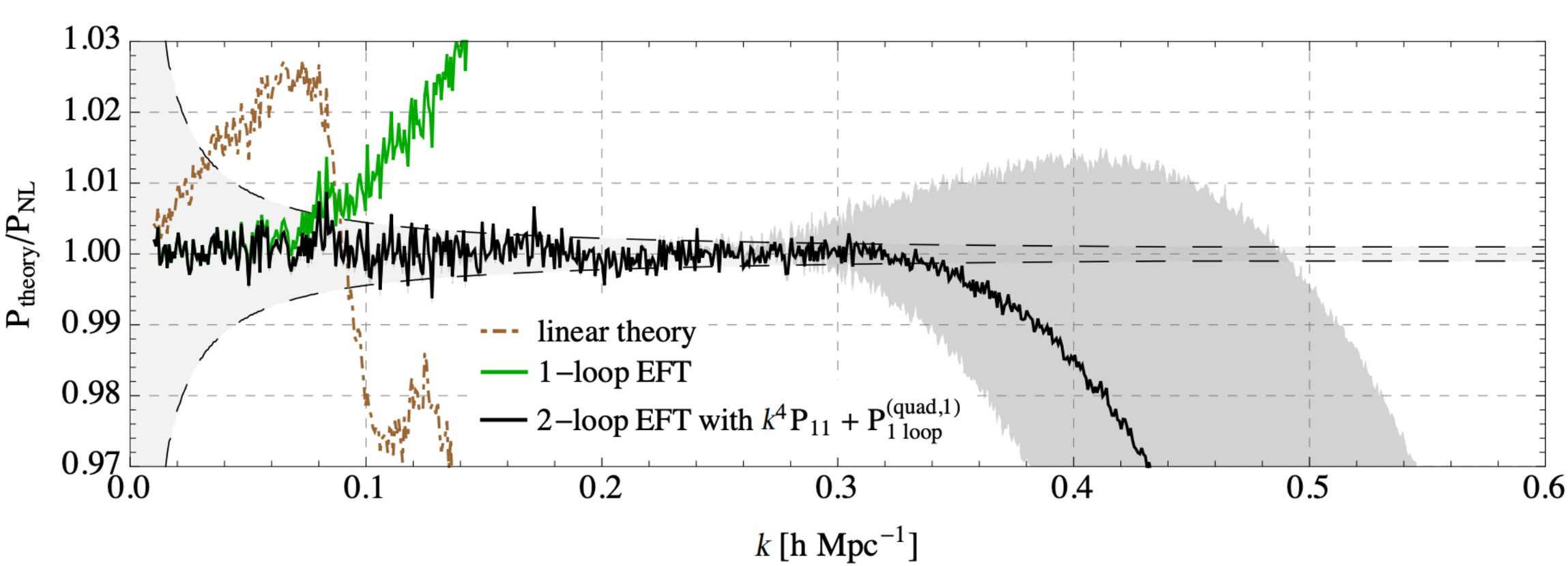
Observational effects mix and destroy information.





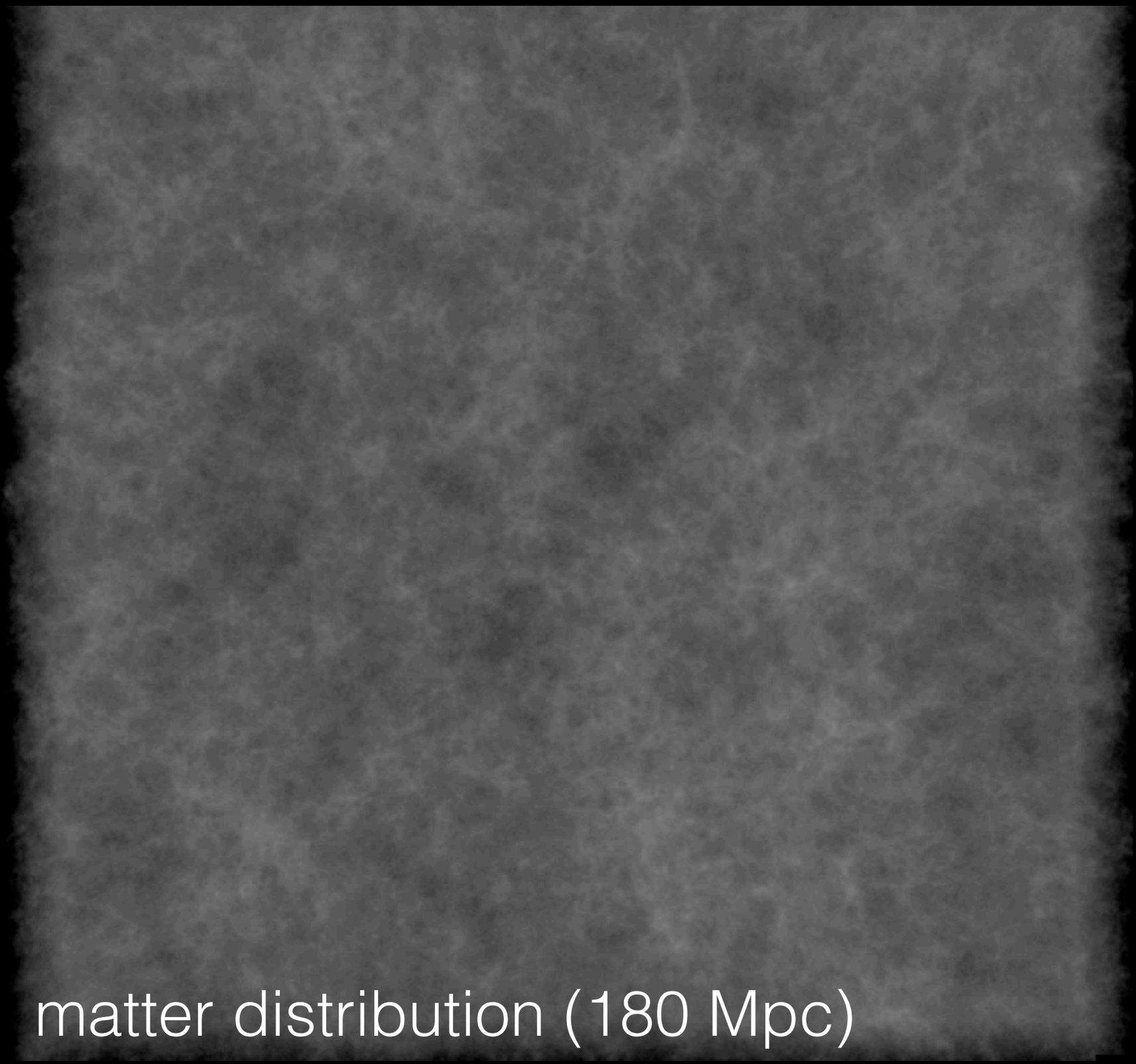
LSS Forward Models — Perturbation Theory

- Perturbation theory:
  - Ansatz: $\delta_m(\mathbf{x}) = \sum_n \delta_m^n(\mathbf{x});\ \delta_g(\mathbf{x}) \sim \sum_i b_i \delta_m(\mathbf{x})^i$
  - Accurate, flexible, fast, but low k reach.
    - Evaluation speed still problematic for MCMC.



Foreman et al 2015

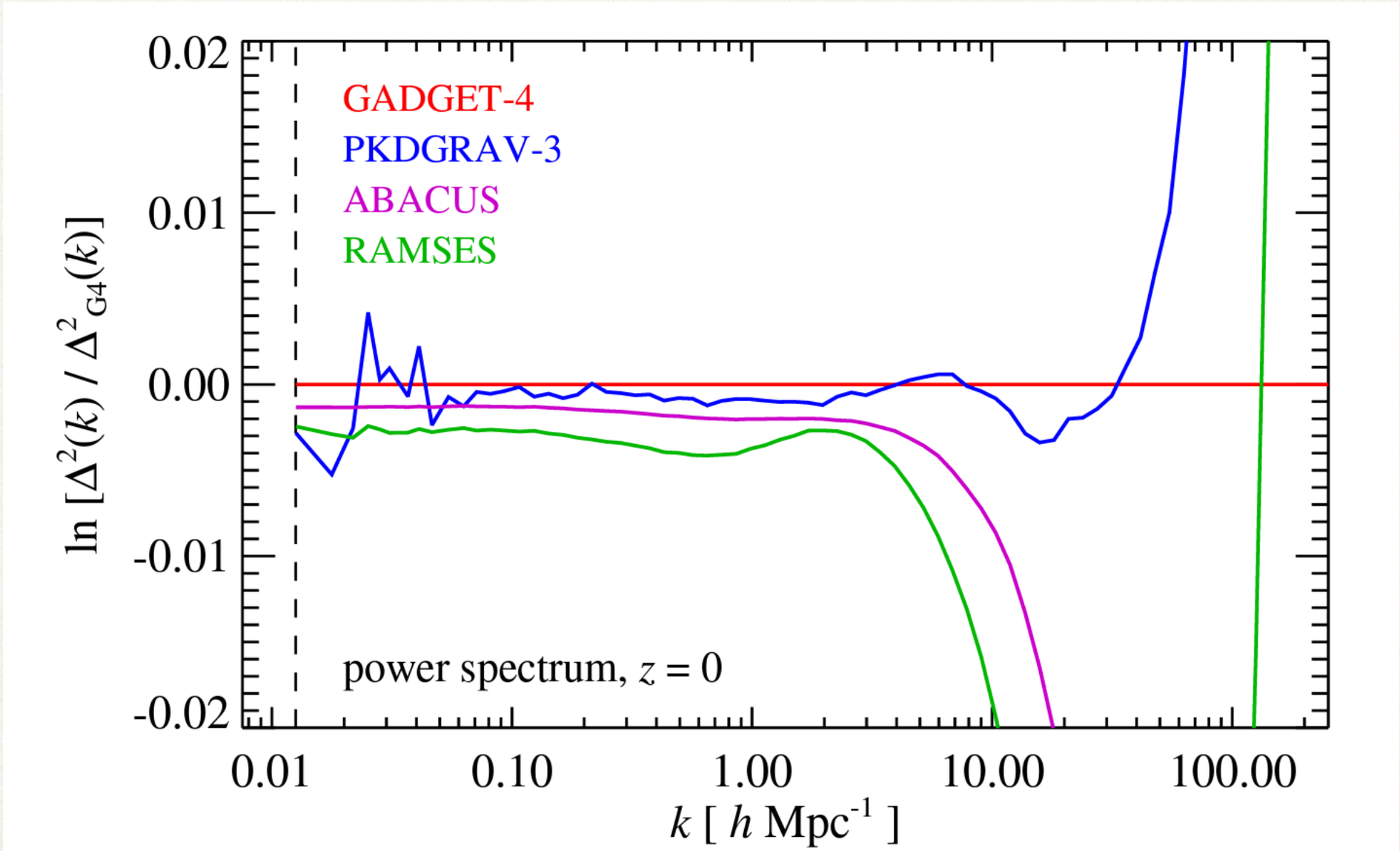
N-body Simulations



movie, simulation, statistics: Matt Becker, Ralf Kaehler, Yao-Yuan Mao, Rachel Reddick, Risa Wechsler (Stanford/SLAC)

LSS Forward Models — N-body Simulations

N-body simulations are converged to $k \sim 1$ but extreme expense prevents them from being used directly in analyses.

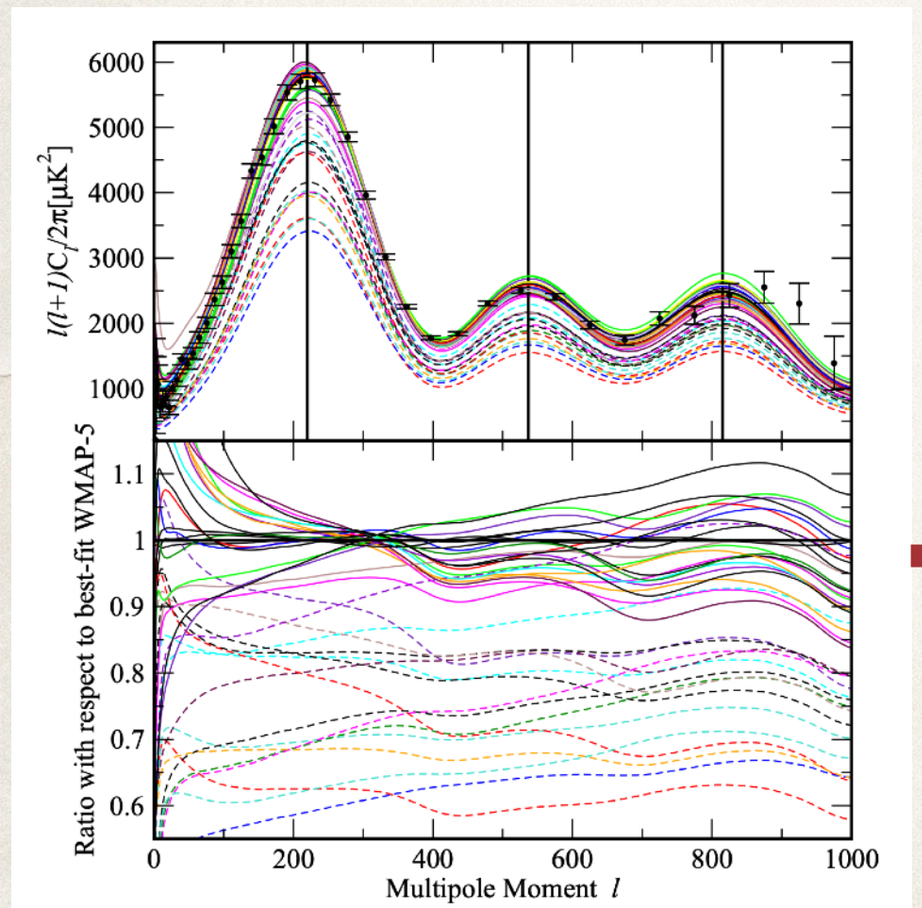


Springel et al 2021

CF Theory Emulation

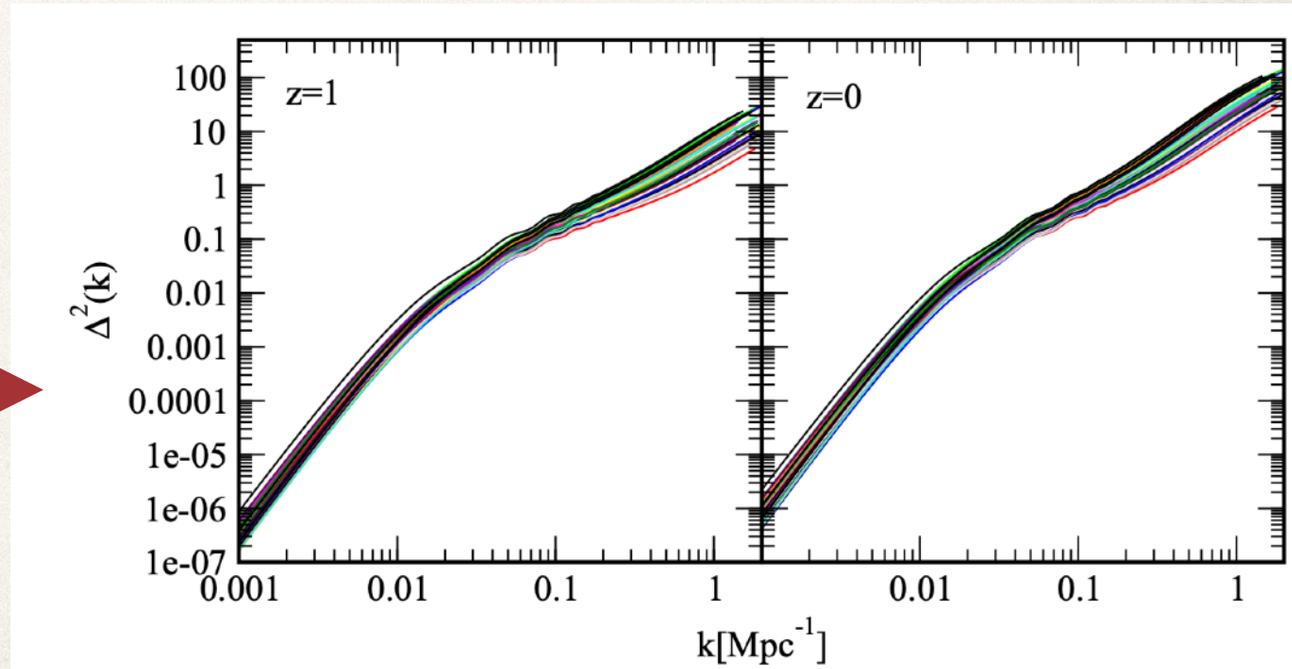
A recipe for high accuracy models for non-linear galaxy and halo statistics:

1. Run suites of simulations spanning currently-allowed cosmological space



N-body

10^4 CPU Hours

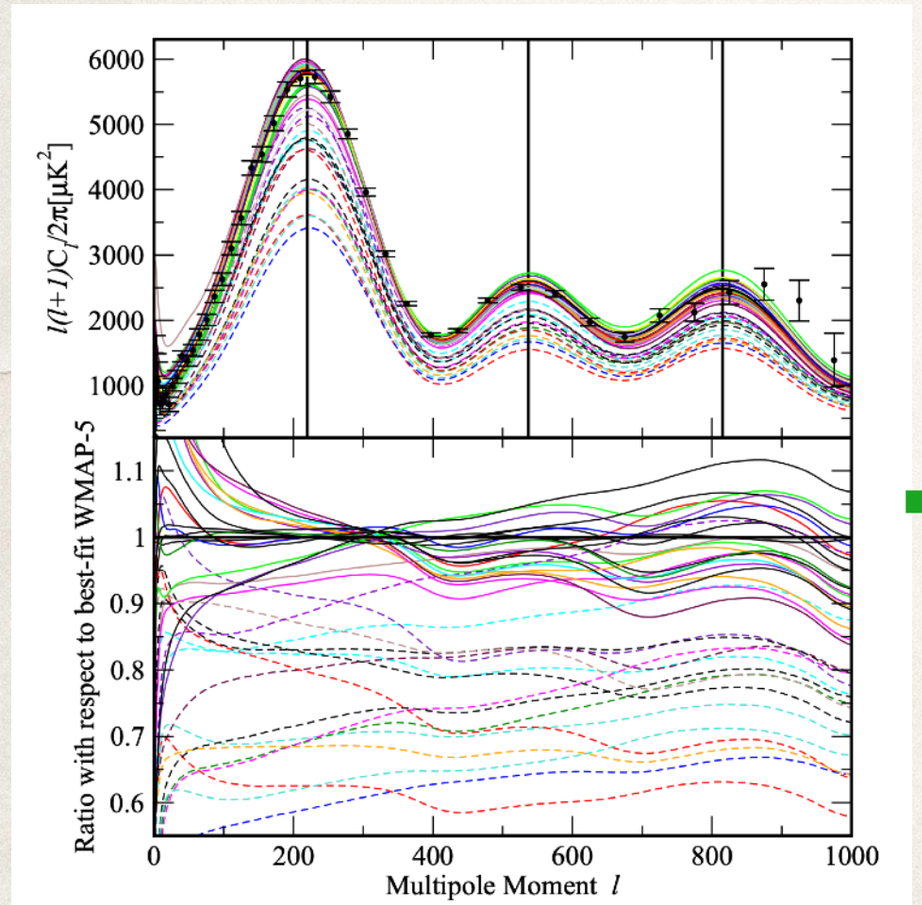


Heitmann et al. 2013

CF Theory Emulation

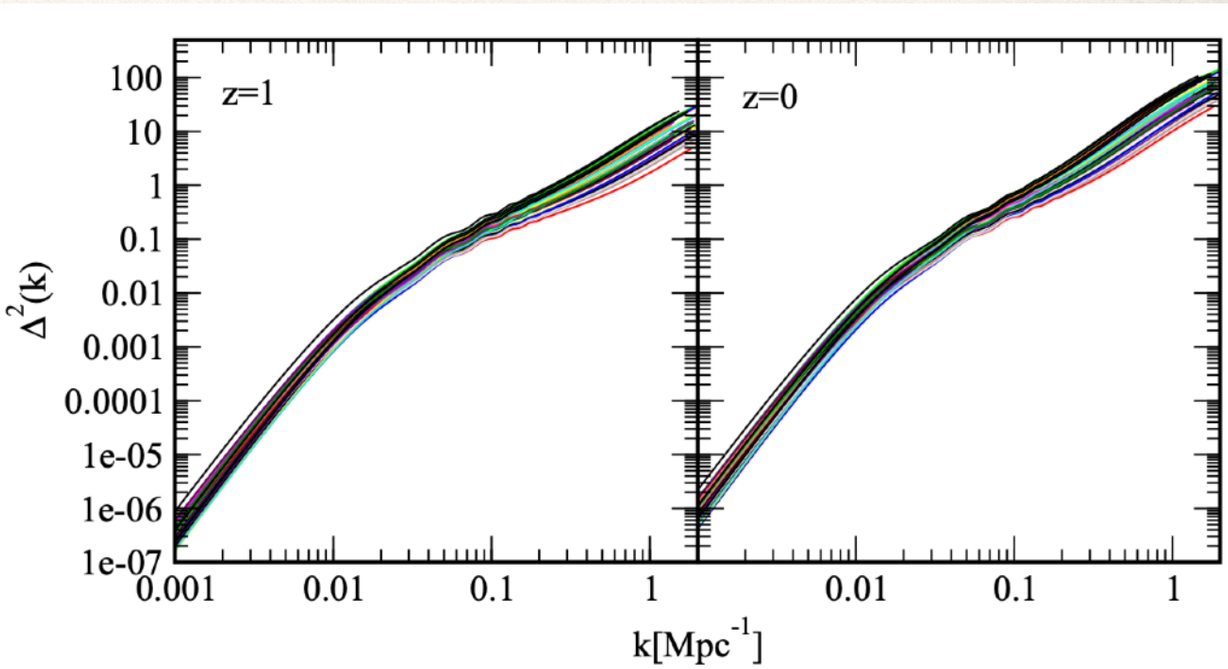
A recipe for high accuracy models for non-linear galaxy and halo statistics:

1. Run suites of simulations spanning currently-allowed cosmological space
2. Interpolate statistics within cosmological + galaxy formation model space



Emulator

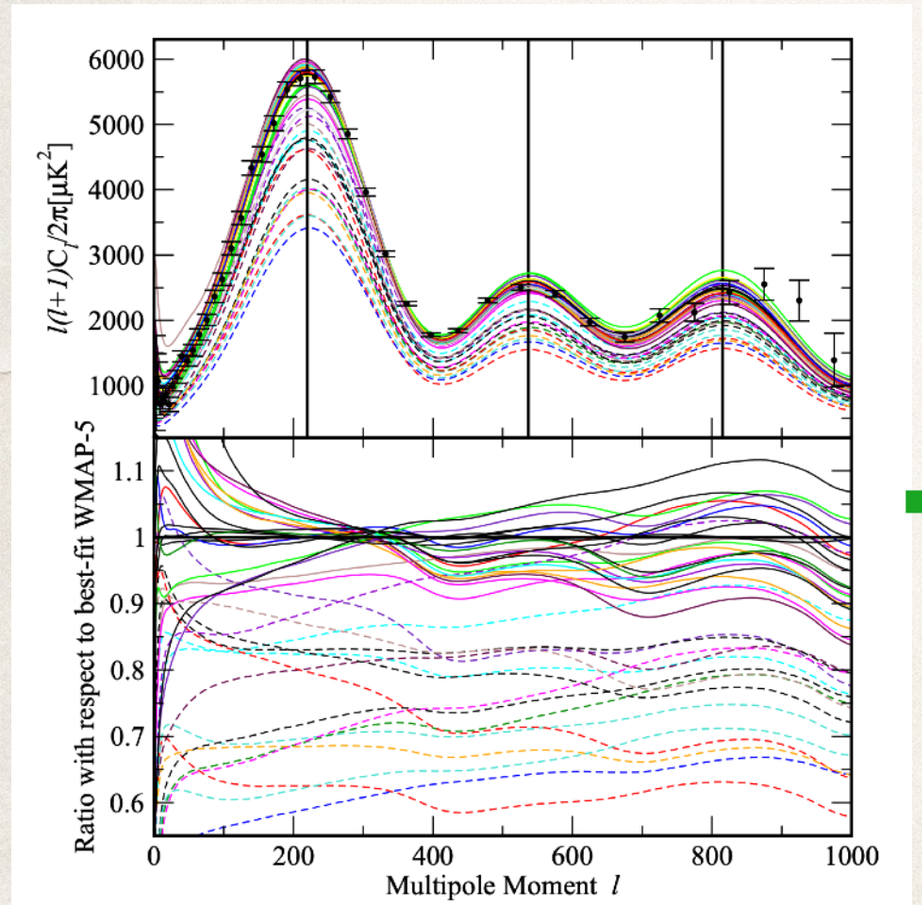
~1 ms



CF Theory Emulation

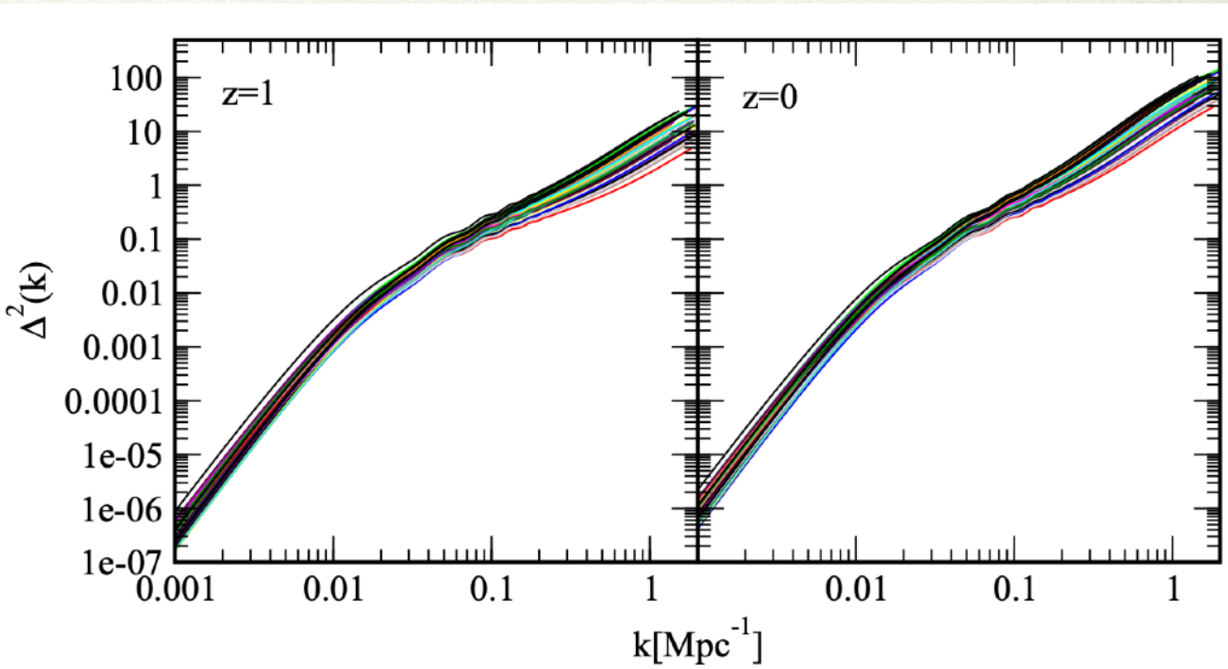
A recipe for high accuracy models for non-linear galaxy and halo statistics:

1. **Run suites of simulations spanning currently-allowed cosmological space**
2. Interpolate statistics within cosmological + galaxy formation model space



Emulator

~1 ms



Outline

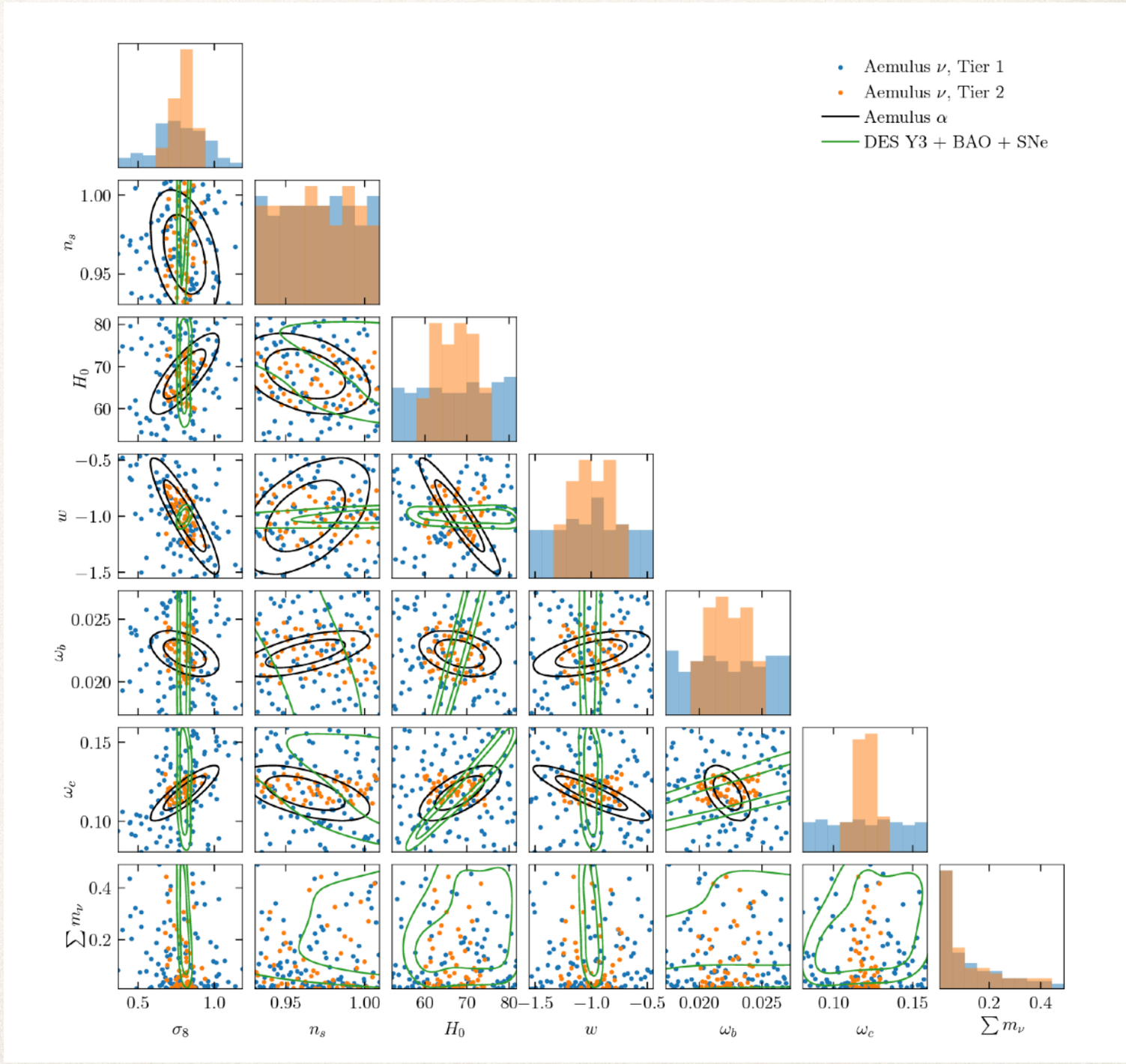
- Computational Challenges in Cosmic Frontier Theory
- **Experimental design**
  - Parameter space sampling, bayesian optimization, variance-reduction
- Emulation techniques and when to use them
  - Gaussian Processes, Polynomial Chaos Expansions, and Neural Networks
- I trained an emulator, now what?!

Experimental Design

Choosing where to generate training data is a key determining factor for how useful an emulator will be, but…

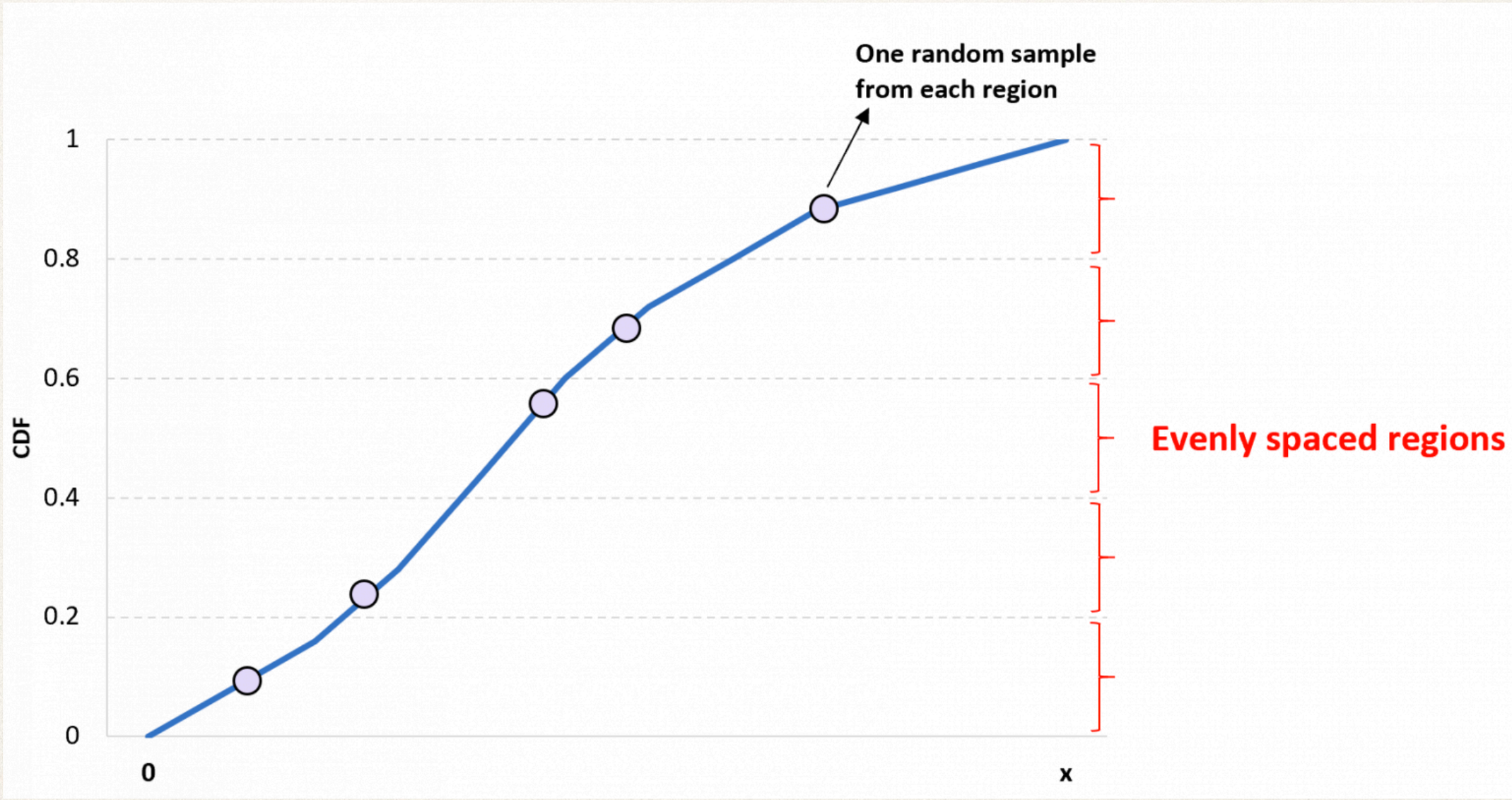
- Dimensionality of parameter space precludes naive grid based sampling.
  - 3 points per dimension is already >2000 sims.
- Monte Carlo sampling is wasteful.

Need something better!



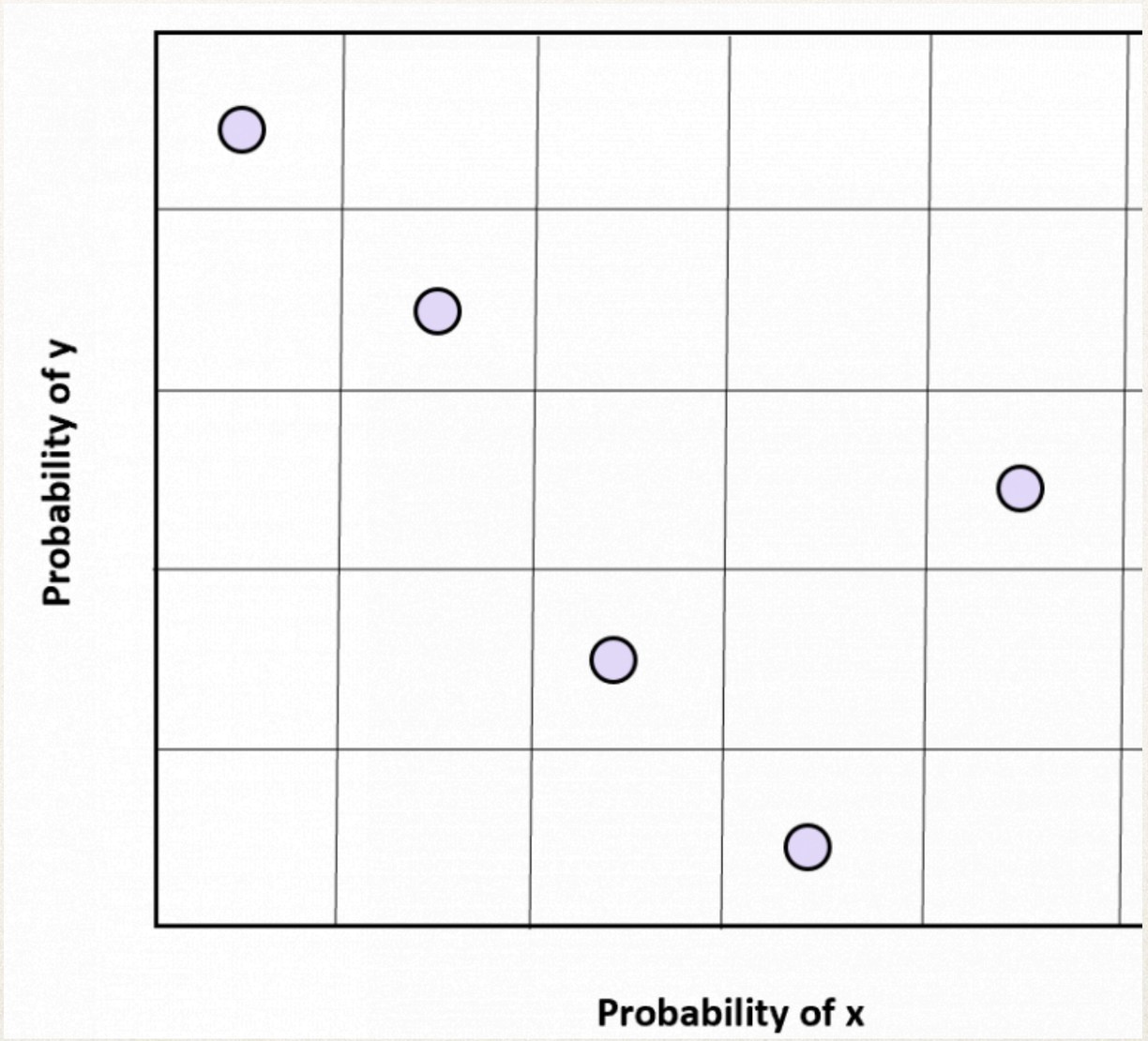
Parameter Space Sampling — Latin Hybercube

Idea: decompose parameter space into regions of equal probability. Sample uniformly from these. In 1-dimension:



Parameter Space Sampling — Latin Hybercube

To sample **S** dimensions with **N** points, divide up each dimension into **N** regions, sample **N** points such that only one point in each row and column. Similar to having **N** rooks on a chess board without threatening each-other.



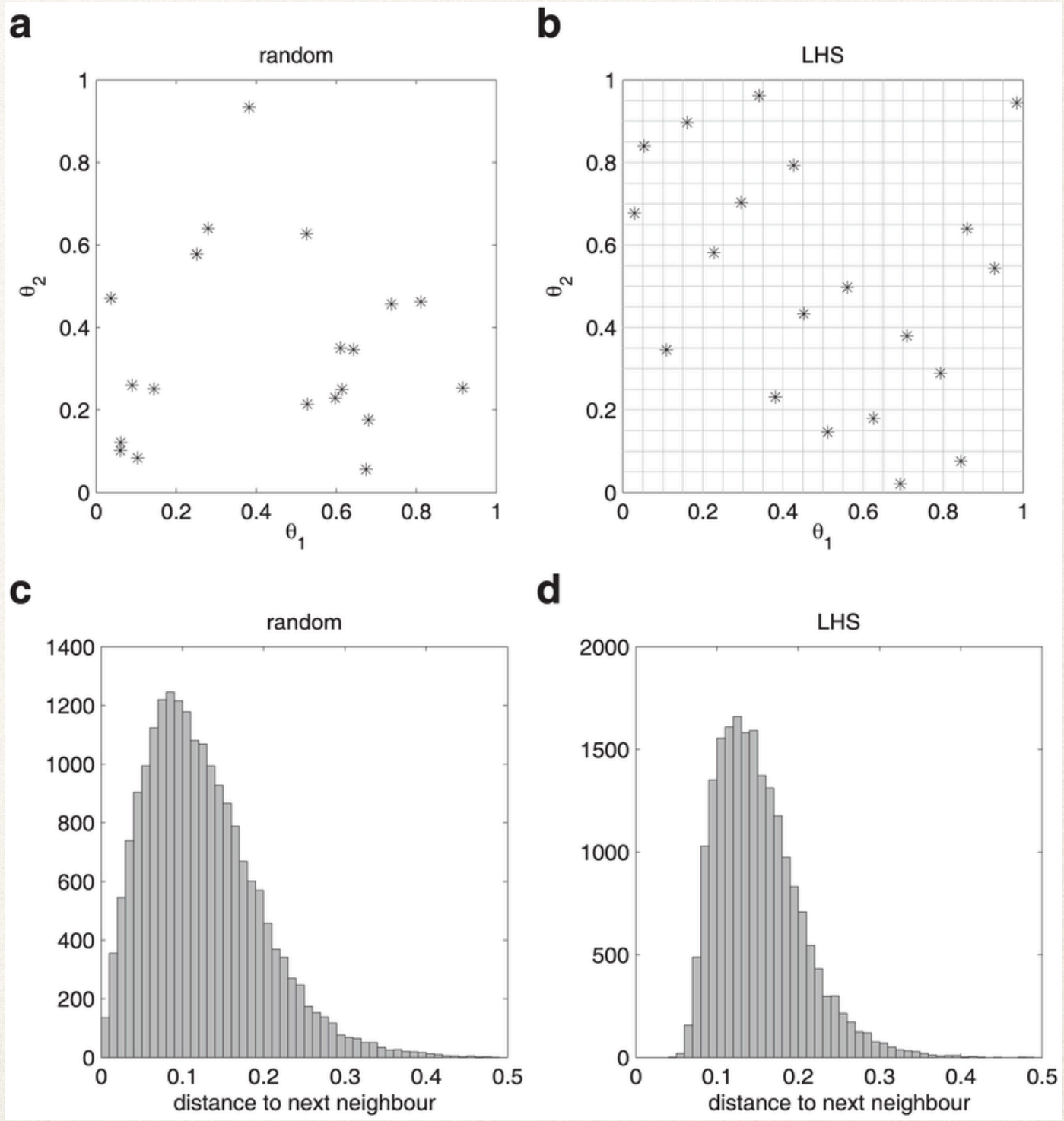
Sampling — Latin Hybercube

Pros:

- Covers space much more evenly than MC sampling.
- Reduces tails in nearest neighbors distances.

Cons:

- Need to know how many samples you want (N) beforehand.
- More computationally expensive to generate large numbers of samples.



Sampling — Low-discrepancy Sequences

The study of low-discrepancy sequences came about in the study of the convergence properties of Monte-Carlo integration:

$$\left| \frac{1}{N} \sum_{i=1}^{N} f(x_i) - \int_{\bar{I}^s} f(u)\, du \right| \leq V(f)\, D_N^*(x_1, \ldots, x_N).$$

where the “discrepancy”

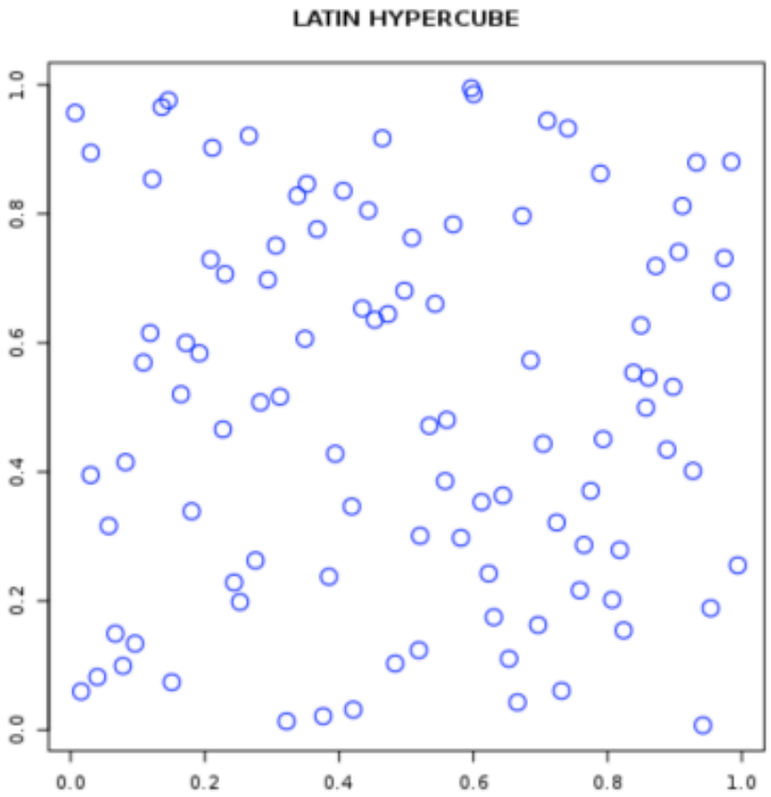
$$D_N(P) = \sup_{B \in J} \left| \frac{A(B;P)}{N} - \lambda_s(B) \right|$$

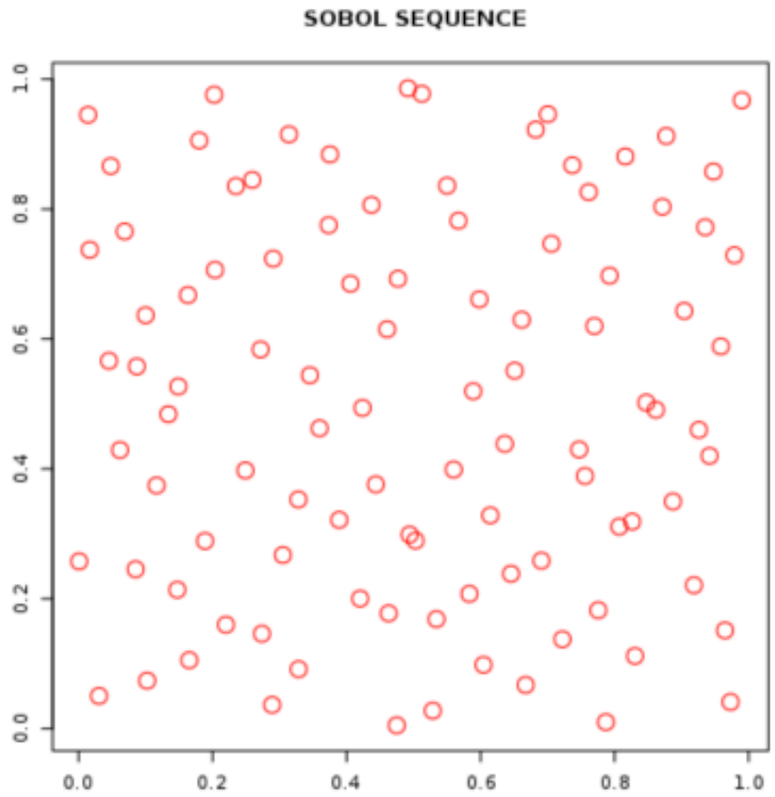
can be interpreted as the worst-case maximum point density deviation over a uniform set.

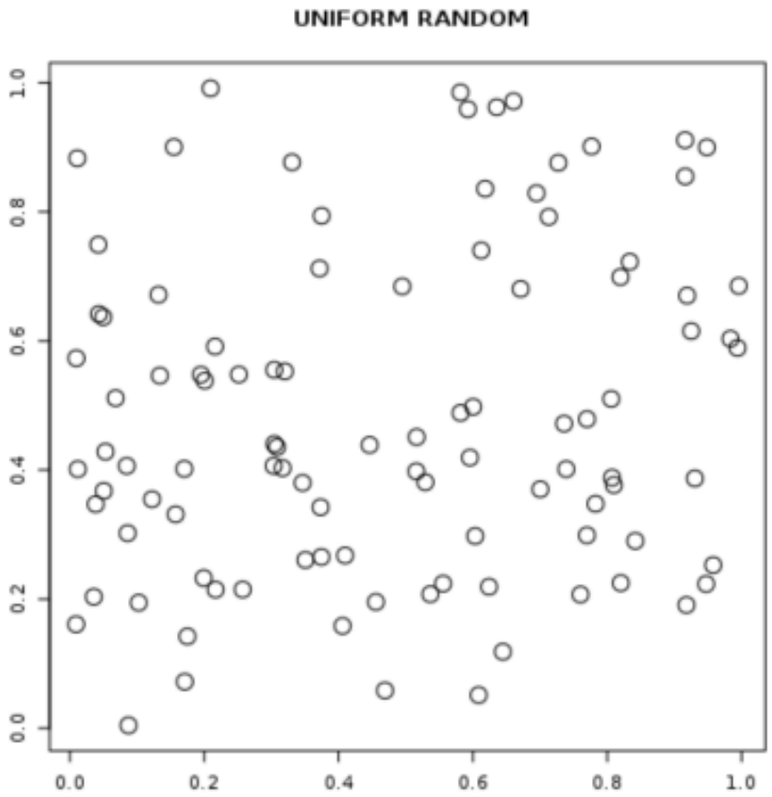
Sampling — Low-discrepancy Sequences

Low-discrepancy sequences minimize D, subject to the constraint that the N-th element of the set can be computed without computing the previous N-1 elements.

The "Sobol" and "Halton" sequences can be shown to minimize discrepancy in the large N limit. **Sobol is preferred** due to issues with Halton at small N.







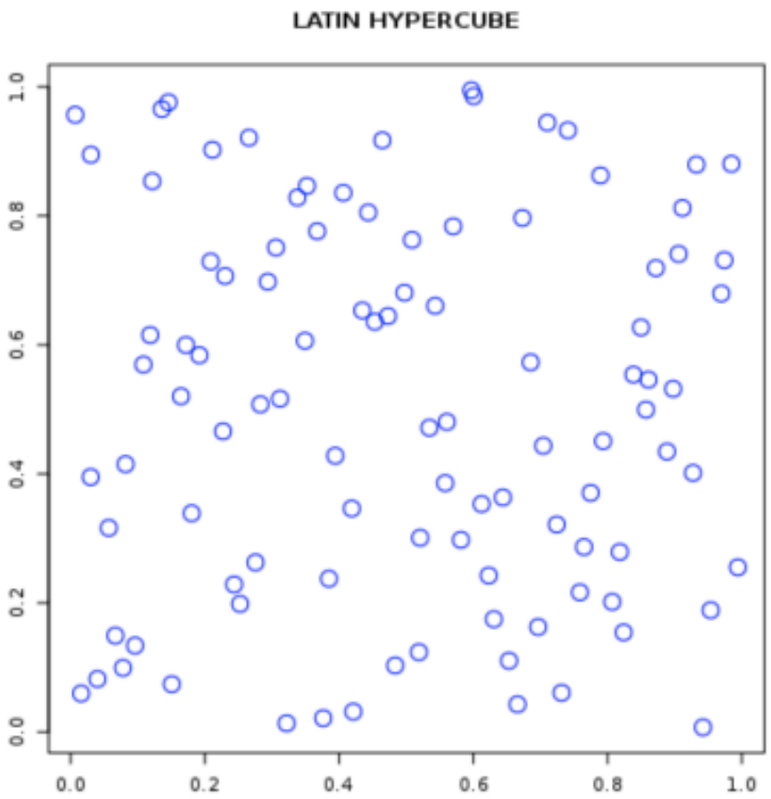
Sampling — Low-discrepancy Sequences

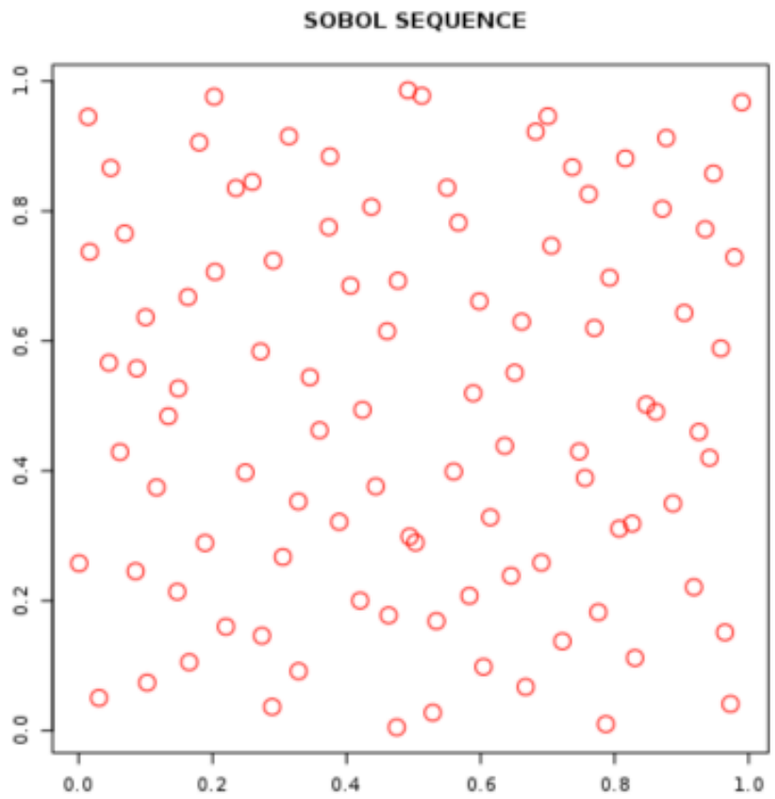
Pros:

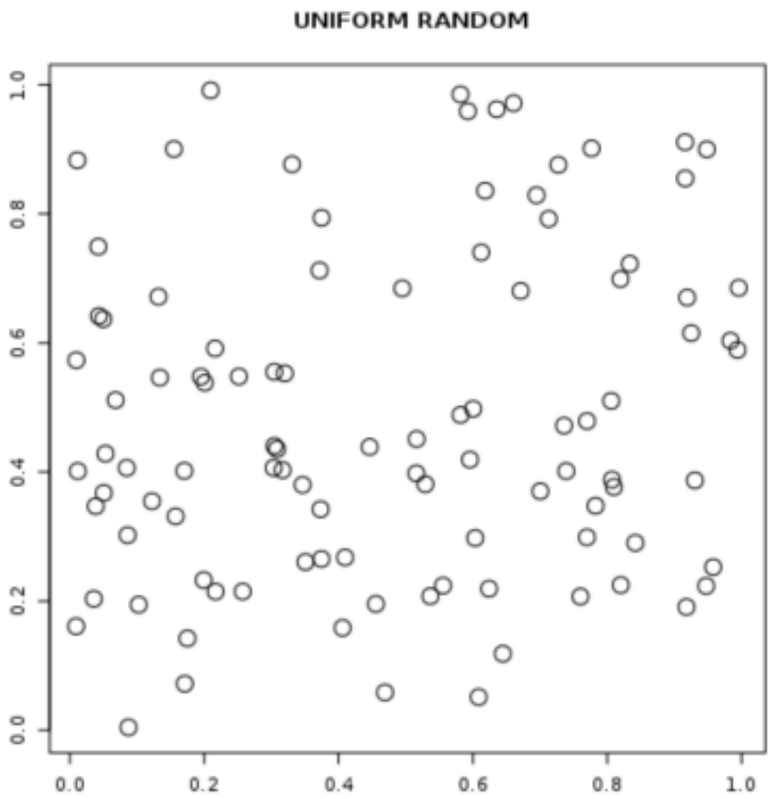
- Computationally efficient to compute
- Can easily generate additional samples after original design.

Cons:

- ??



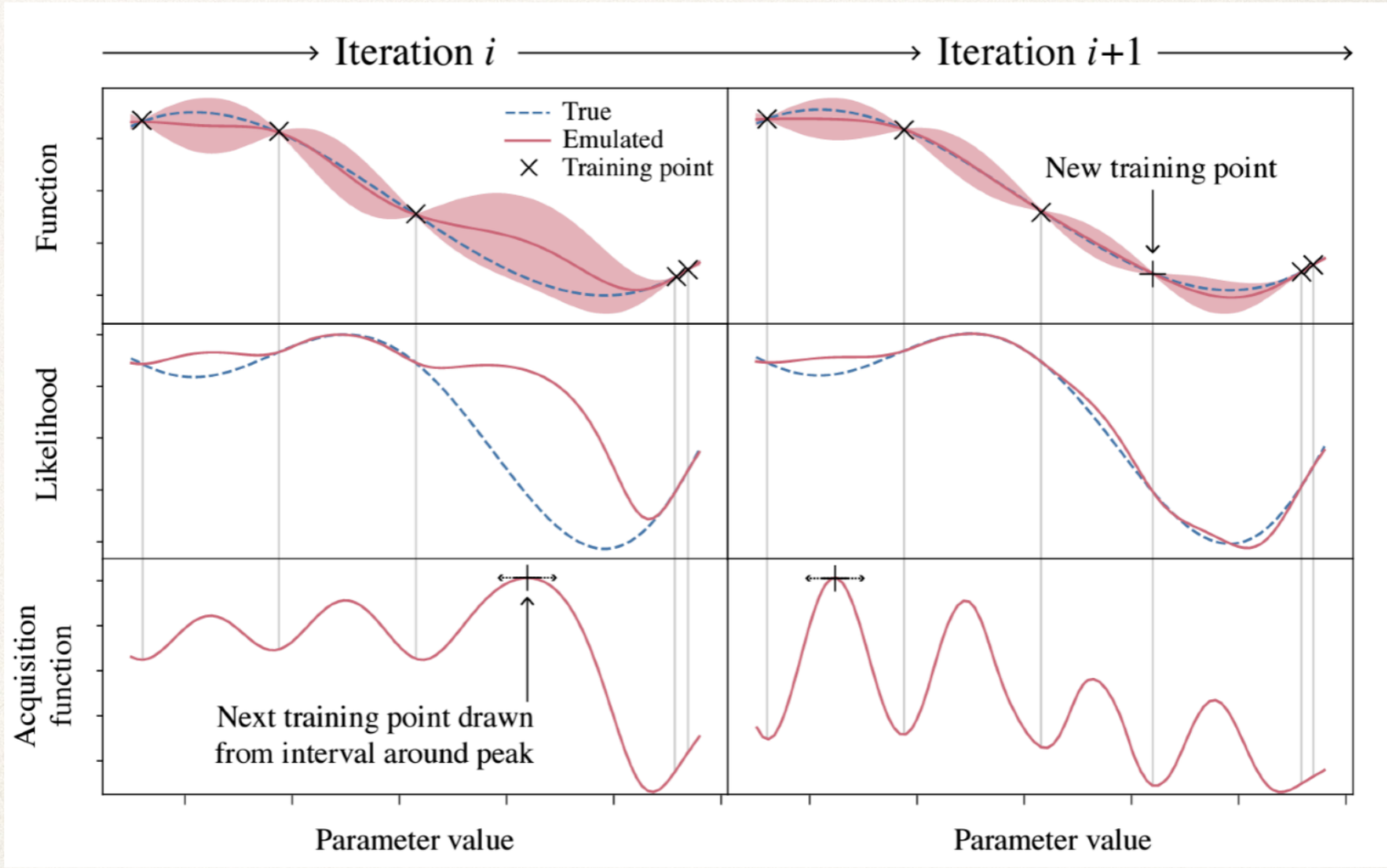




Sampling — Bayesian Optimization

1. Train emulator using an initial set of simulations.
2. Use initial model and uncertainty estimate (plus additional info) to choose new point to run simulation.
3. Profit!

$$\mathcal{A}(\tilde{\boldsymbol{\theta}}) = \mathcal{P}(\tilde{\boldsymbol{\theta}}|\boldsymbol{d}) + \alpha \boldsymbol{\sigma}^{\mathrm{T}}(\tilde{\boldsymbol{\theta}}) \Sigma^{-1} \boldsymbol{\sigma}(\tilde{\boldsymbol{\theta}}),$$

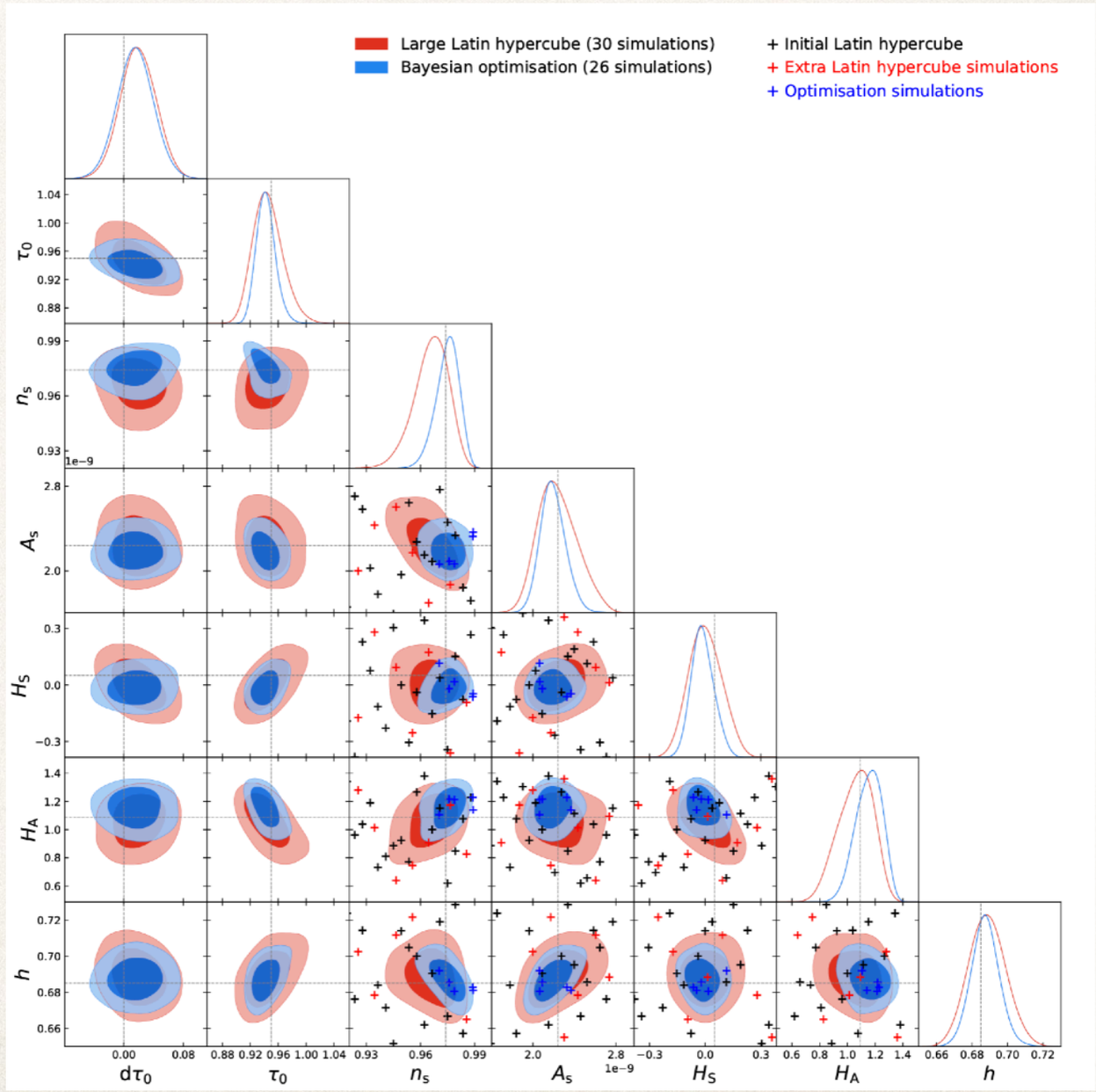


Rogers et al. 2018 (1812.04631)

Sampling — Bayesian Optimization

$$\mathcal{A}(\tilde{\boldsymbol{\theta}}) = \mathcal{P}(\tilde{\boldsymbol{\theta}}|\boldsymbol{d}) + \alpha \boldsymbol{\sigma}^{\mathrm{T}}(\tilde{\boldsymbol{\theta}}) \Sigma^{-1} \boldsymbol{\sigma}(\tilde{\boldsymbol{\theta}}),$$

- Don't need a particularly accurate initial model to converge to something very accurate.
- Some caution required about how much to weight actual posterior (what to set alpha to?)



Rogers et al. 2018 (1812.04631)

Preprocessing — Sample Variance Reduction

There exists a bias-variance trade off in generating training data:

- The volume of a simulation controls the statistical error on quantities of interest (e.g. power spectra)

$$\frac{\sigma^2_{P(k)}}{P(k)} \propto \frac{1}{V} \frac{1}{\Delta k k^2}$$

- The resolution of the simulation controls the smallest usable scale in the simulation.

$$r_{min} \propto \frac{V^{1/3}}{N_{\text{mesh}}} \qquad N_{\text{sim}} \propto \frac{\text{Compute hours}}{N_{\text{mesh}}}$$

Want to minimize $\sigma_{P(k)}$, r_{min} and maximize N_{sim} in order to build the best possible emulator…

Sample Variance Reduction — Control Variates

Want to maximize $N_{\rm sim}$ subject to constraints on $\sigma_{P(k)}$ and $r_{\rm min}$. Would be great if we could increase effective volume without increasing $r_{\rm min}$?

We can via control variates:

Given a noisy quantity (i.e. a measurement from simulations),
but have access to a cheap correlated "control variate", we can construct:

$$\hat{y} \equiv \hat{x} - \beta(\hat{c} - \mu_c)$$

we can then optimize β to minimize the variance of $\hat{y}$, giving

$$\hat{\beta} = \frac{\mathrm{cov}[\hat{\mathrm{x}}, \hat{\mathrm{c}}]}{\mathrm{var}[\hat{\mathrm{c}}]}$$

leading to a reduction in variance of

$$\frac{\mathrm{var}[\hat{\mathrm{y}}]}{\mathrm{var}[\hat{\mathrm{x}}]} = 1 - \frac{\mathrm{Cov}^2[\hat{\mathrm{x}}, \hat{\mathrm{c}}]}{\mathrm{Var}[\hat{\mathrm{c}}]\mathrm{Var}[\hat{\mathrm{x}}]} + \frac{\hat{\beta}^2 \mathrm{Var}[\hat{\mathrm{c}}]}{M\mathrm{Var}[\hat{\mathrm{x}}]} = 1 - \rho_{xc}^2 + \frac{\hat{\beta}^2 \mathrm{Var}[\hat{\mathrm{c}}]}{M\mathrm{Var}[\hat{\mathrm{x}}]}$$

Sample Variance Reduction — Control Variates

$$\hat{y} \equiv \hat{x} - \beta(\hat{c} - \mu_c) \qquad \frac{\mathrm{var}[\hat{\mathrm{y}}]}{\mathrm{var}[\hat{\mathrm{x}}]} = 1 - \rho_{xc}^2 + \frac{\hat{\beta}^2 \mathrm{Var}[\hat{\mathrm{c}}]}{M \mathrm{Var}[\hat{\mathrm{x}}]}$$

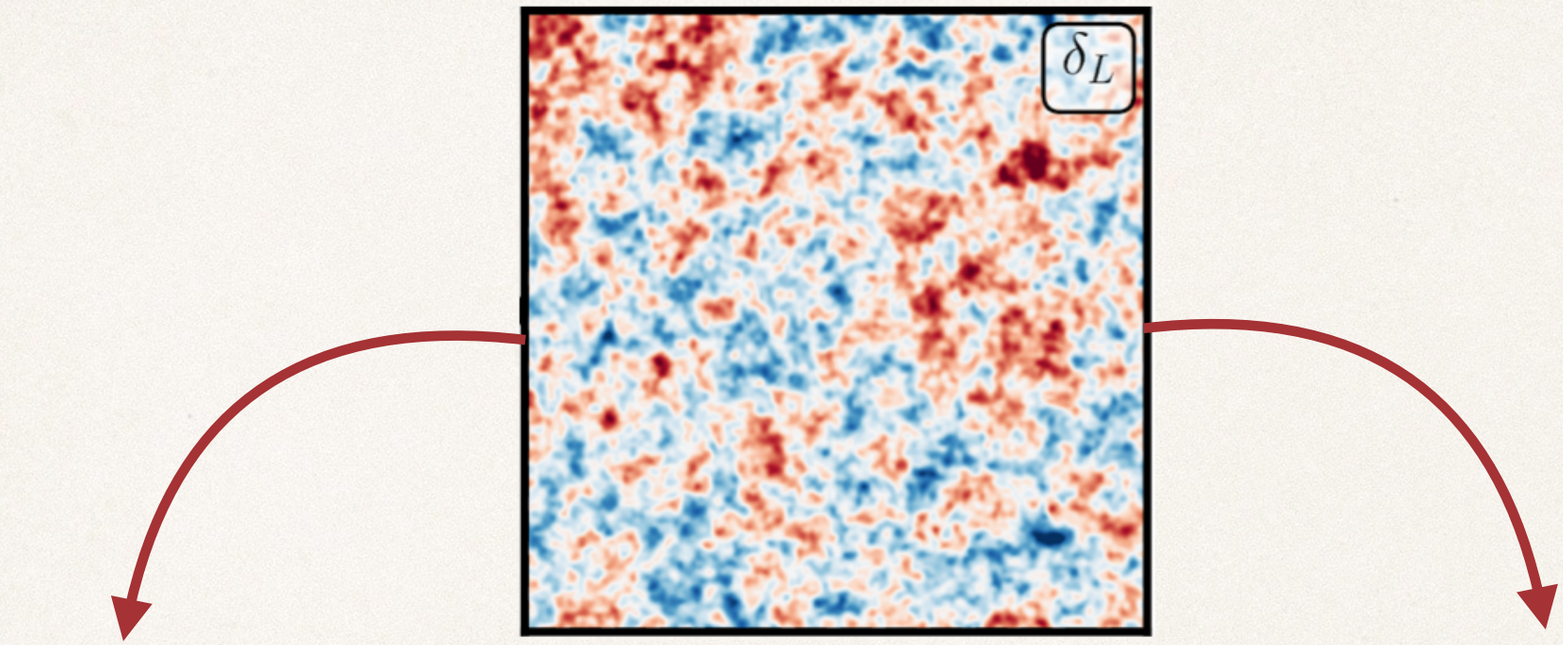
Application to cosmology introduced as CARPool in Chartier et al 20, Chartier & Wandelt 21, Chartier & Wandelt 22.

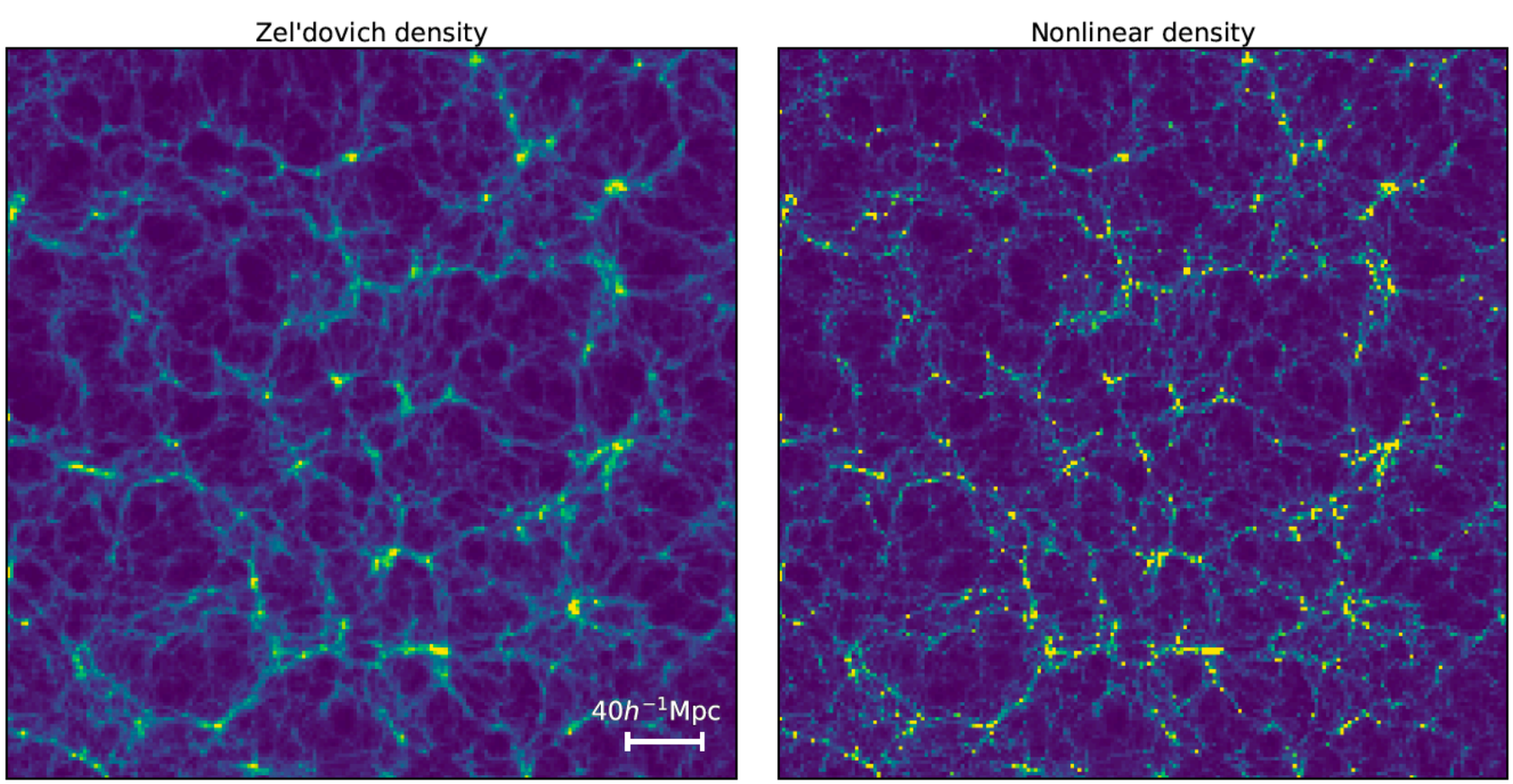
- Usually use approximate N-body simulations for $\hat{c}$.
- e.g. DESI FastPM effort (Ding et al 2022) used 500 FastPM mocks requiring **21M CPU hours in total**

Need a control variate that is:

- **Inexpensive**
- **Highly correlated with $\hat{x}$**
- **Analytically known μ_c**

Zel'dovich approximation to the rescue

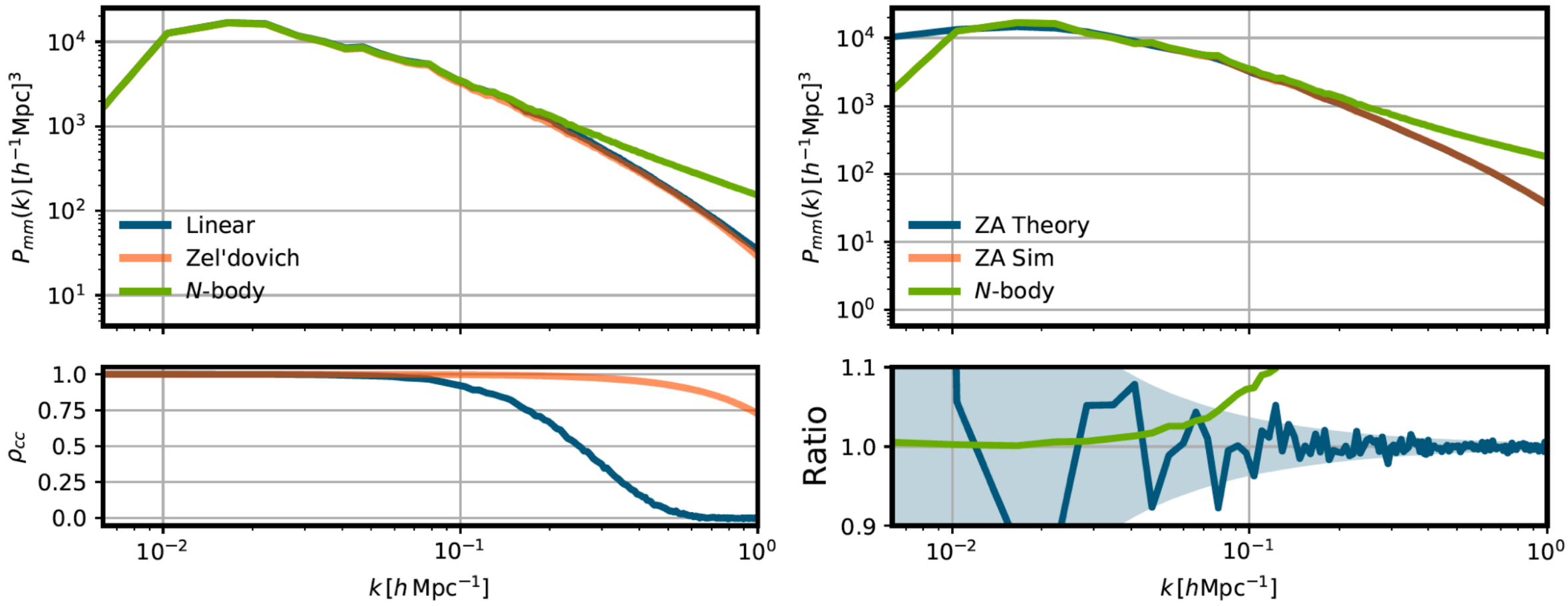




Kokron et al 2022

Zel'dovich approximation to the rescue

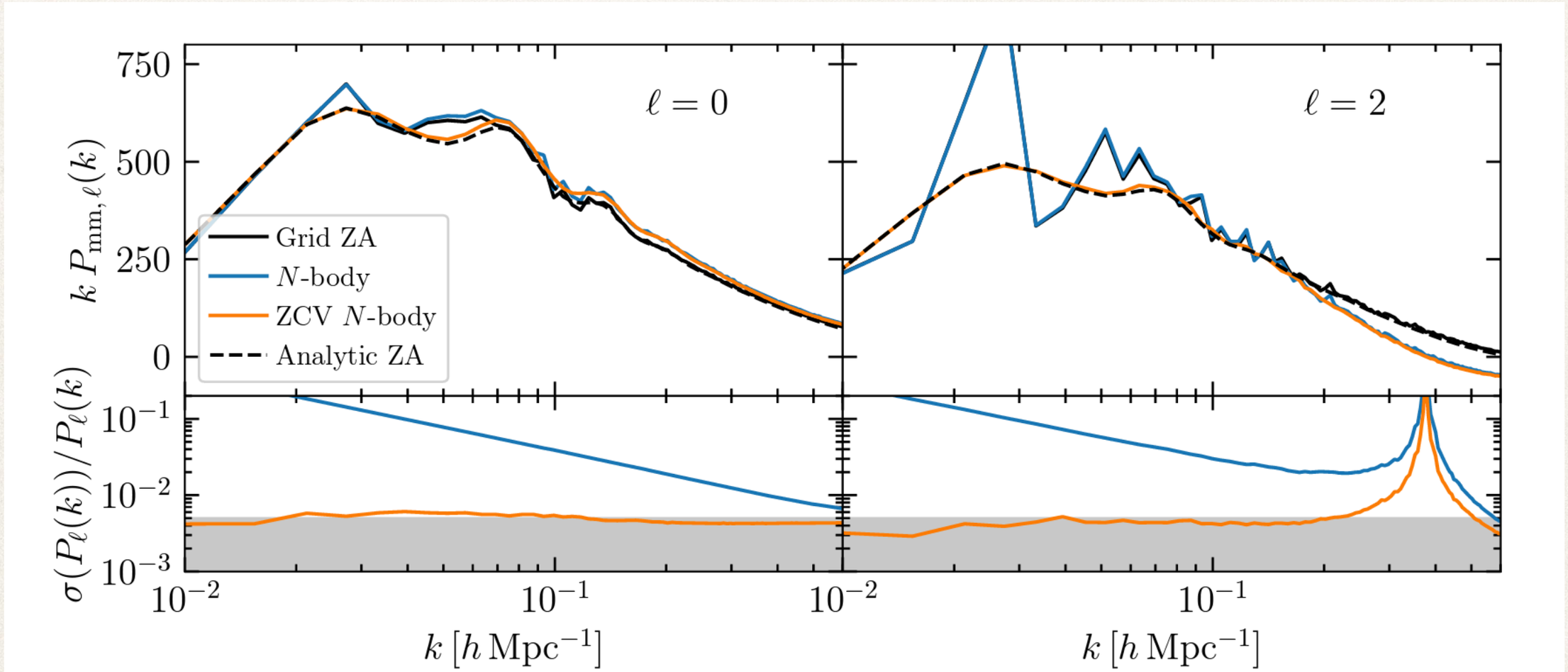
Kokron et al 2022 (incl. JDR)



ZA is **inexpensive, highly correlated** with the non-linear matter field, and **we can predict its mean** exactly.

Zel'dovich approximation to the rescue: An illustrative example

DeRose et al. 2022

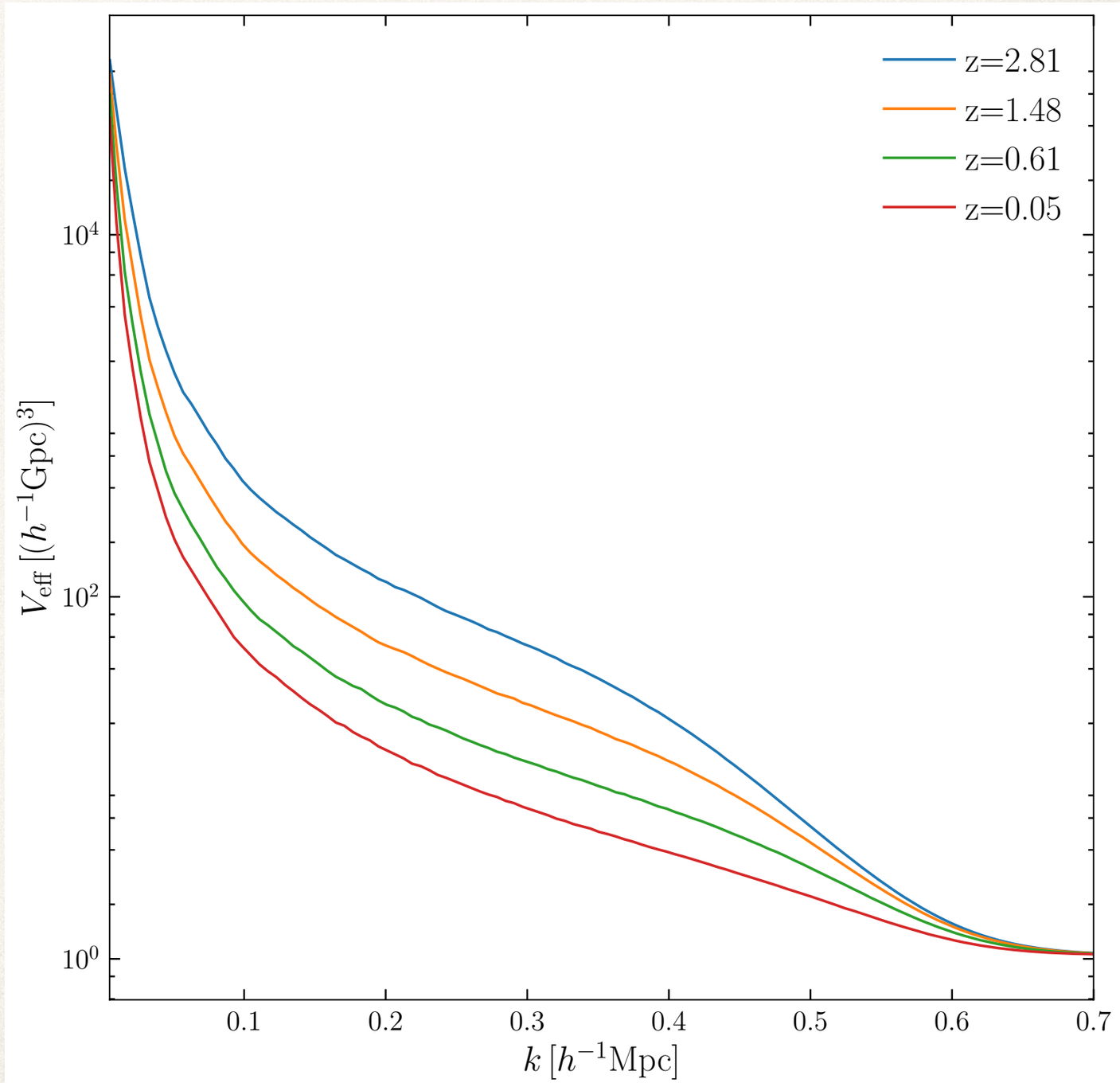


Because the Zel'dovich realization is highly correlated with the N-body, we can simply subtract the difference between grid ZA and analytic ZA from the N-body measurement to remove noise.

Zel'dovich approximation to the rescue: Upshot

- Leads to orders of magnitude increases in effective volume.
- Highly versatile, can be used on a large number of observables measured from a variety of different types of simulations
- Control variates in general can be a very powerful technique for mitigating noise in simulation Monte Carlo estimates

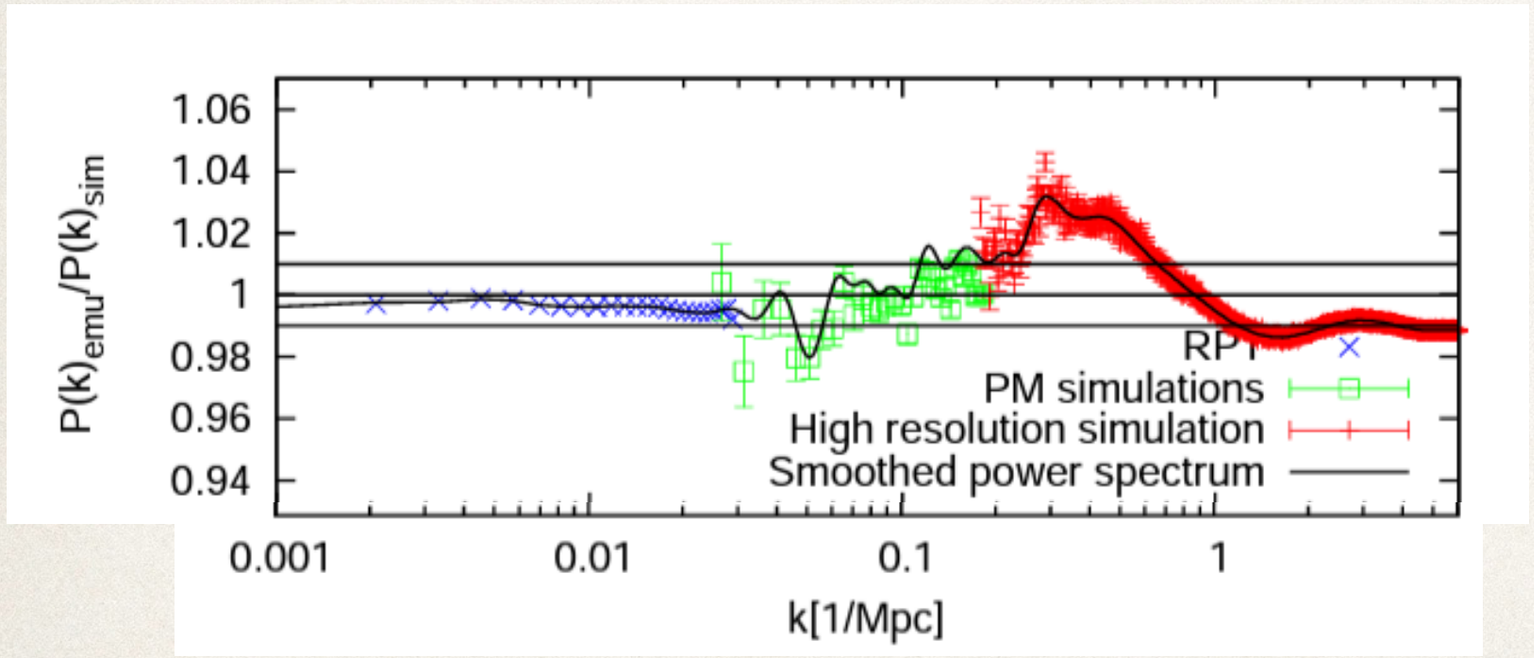
DeRose et al. 2023



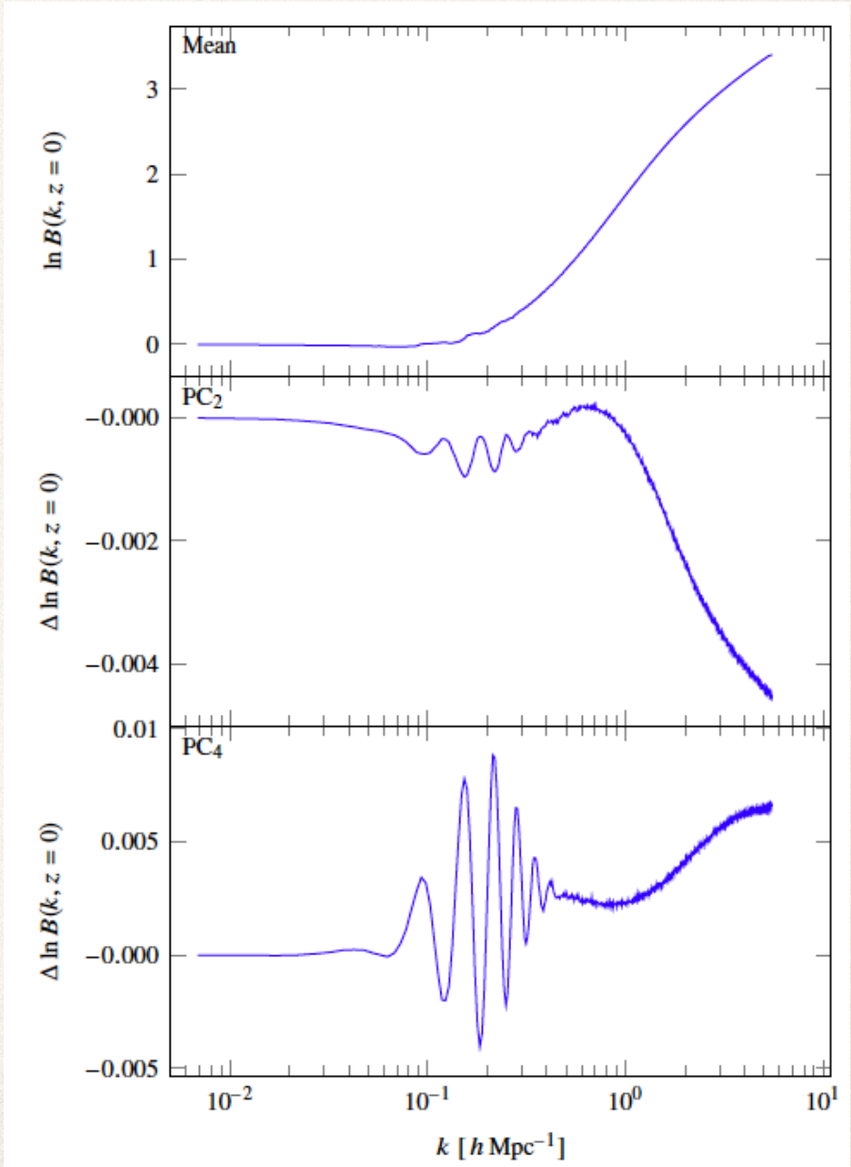
Preprocessing — Miscellaneous

Some common preprocessing steps:

- Reducing dynamic range (e.g. taking the logarithm)
- Whitening
- Smoothing (e.g. via Savitsky-Golay)
- Principle component decomposition



Heitmann et al. 2013



Knabenhans et al 2019

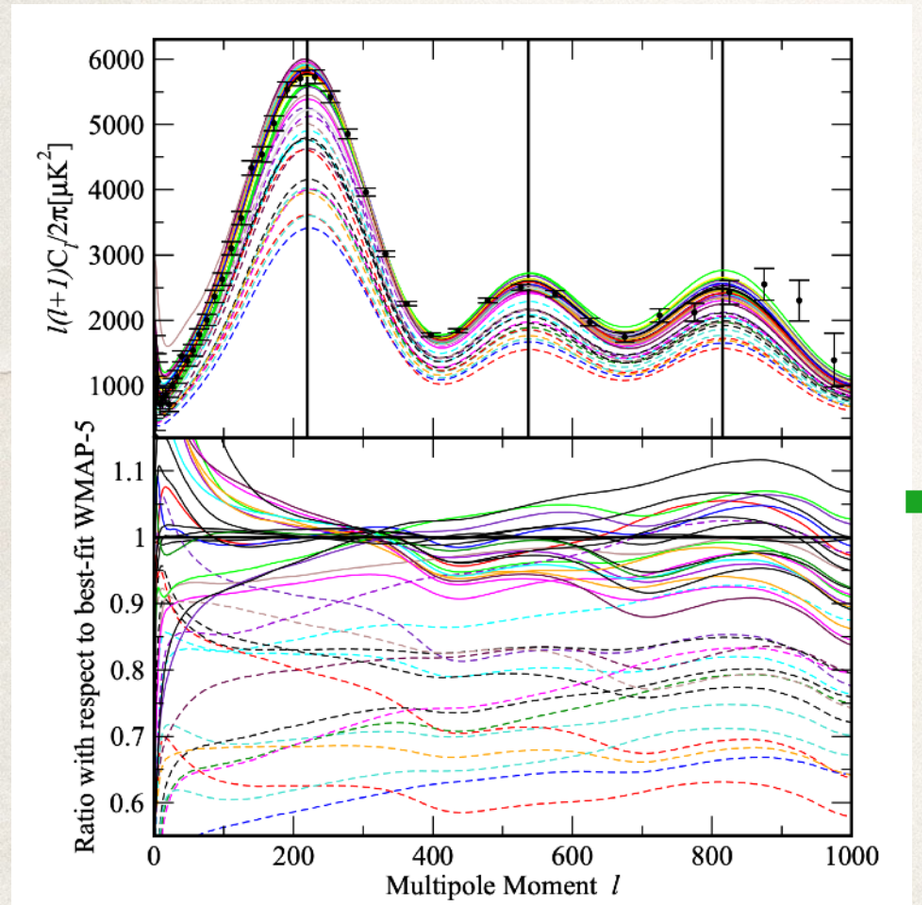
Outline

- Computational Challenges in Cosmic Frontier Theory
- Experimental design
  - Parameter space sampling, bayesian optimization, variance-reduction
- Emulation techniques and when to use them
  - Gaussian Processes, Polynomial Chaos Expansions, and Neural Networks
- I trained an emulator, now what?!

CF Theory Emulation

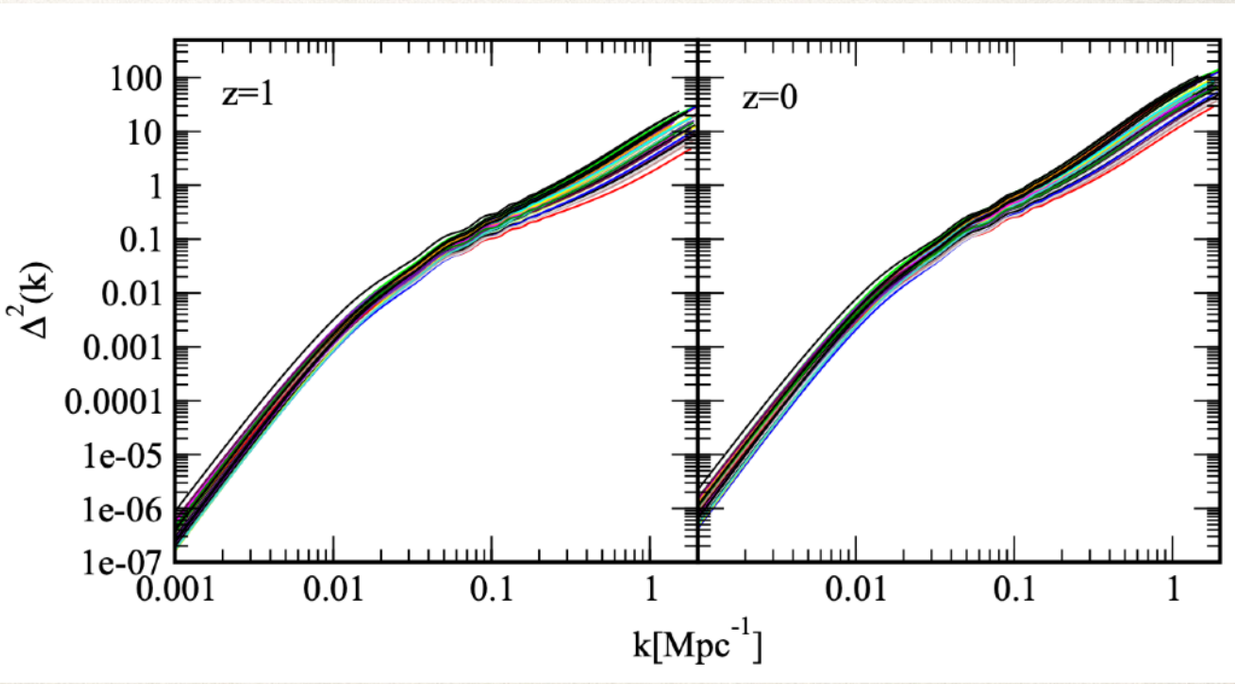
A recipe for high accuracy models for non-linear galaxy and halo statistics:

1. Run suites simulations spanning currently-allowed cosmological space
2. **Interpolate statistics within cosmological + galaxy formation model space**



Emulator

~1 ms

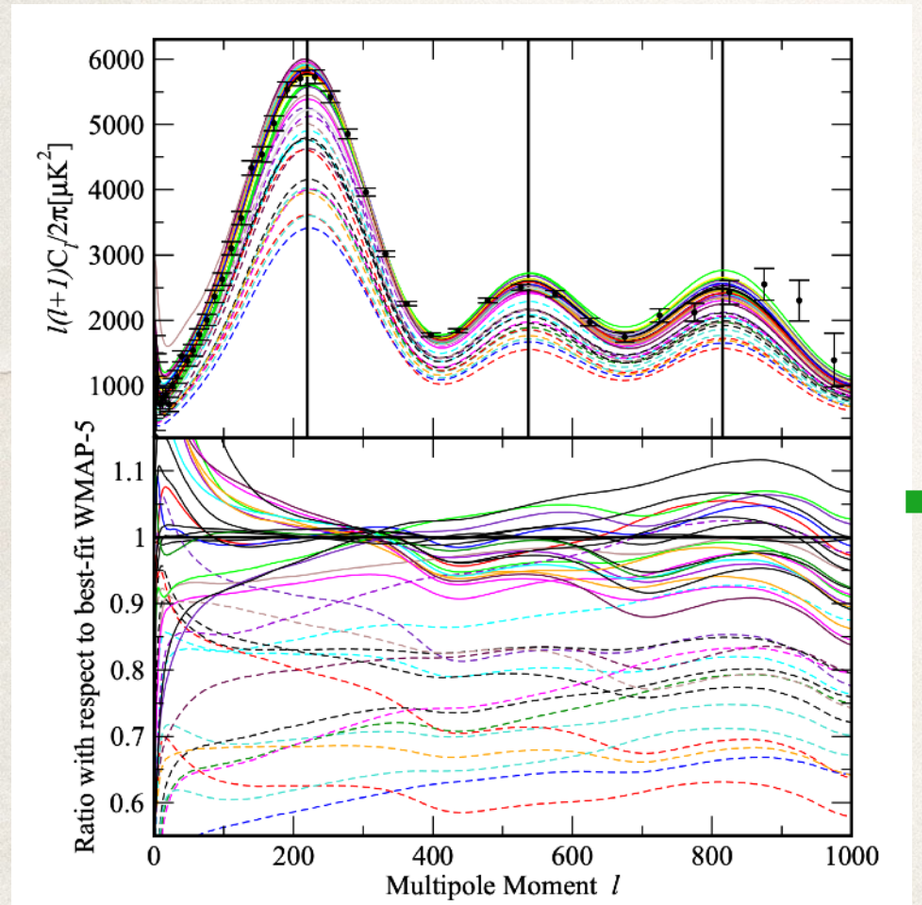


CF Theory Emulation

There is now a large literature on emulation in the cosmology literature.

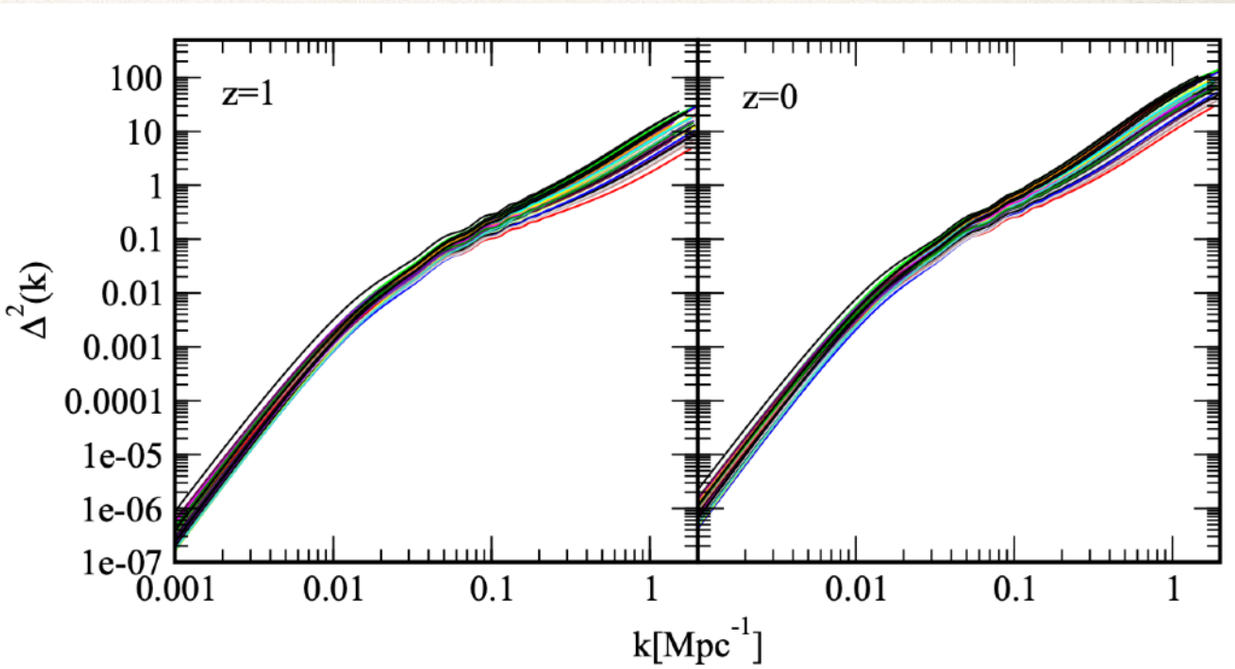
Some commonly used options for surrogate models are:

- Gaussian processes (GP)
- Polynomial Chaos Expansions (PCE)
- Neural networks (usually simple fully connected MLPs)



Emulator

~1 ms



Gaussian Processes

Definition 2.1 *A* Gaussian process *is a collection of random variables, any finite number of which have a joint Gaussian distribution.* □

A Gaussian process is completely specified by its mean function and covariance function. We define mean function $m(\mathbf{x})$ and the covariance function $k(\mathbf{x}, \mathbf{x}')$ of a real process $f(\mathbf{x})$ as

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))], \end{aligned} \tag{2.13}$$

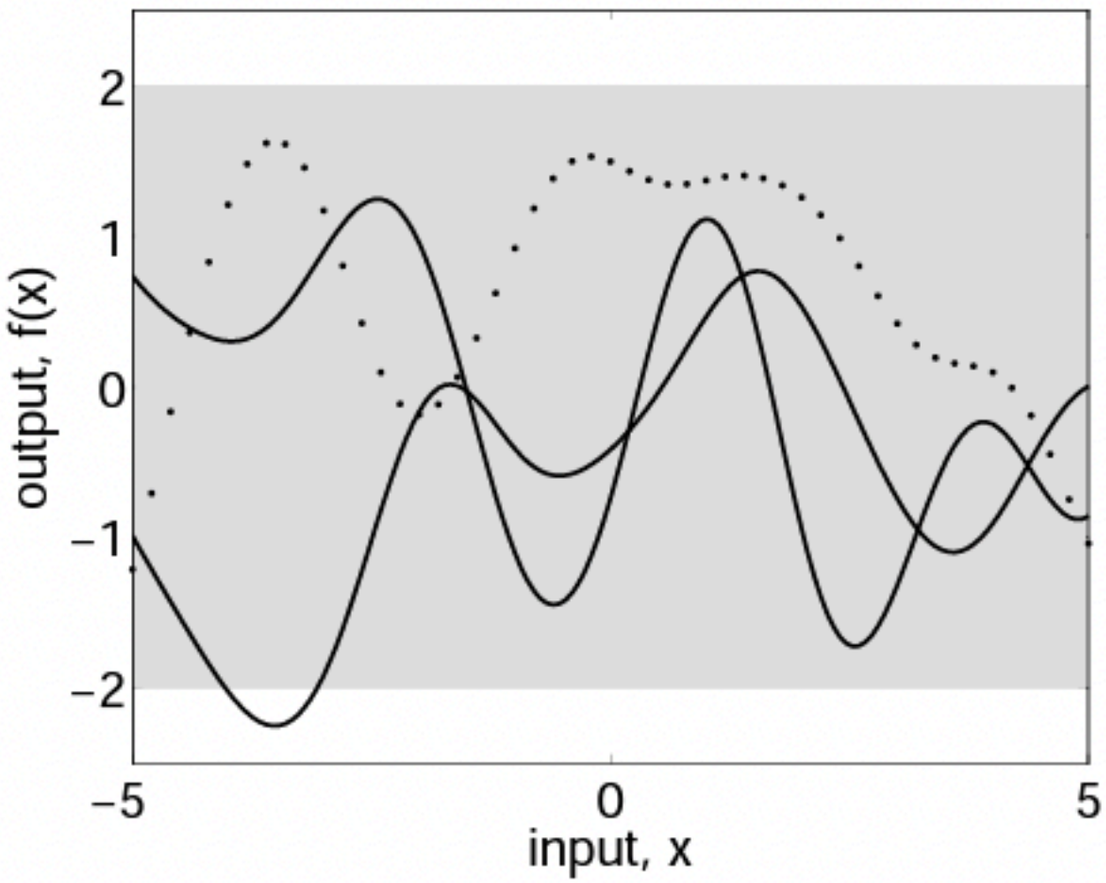
and will write the Gaussian process as

$$f(\mathbf{x}) \sim \mathcal{GP}\big(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\big). \tag{2.14}$$

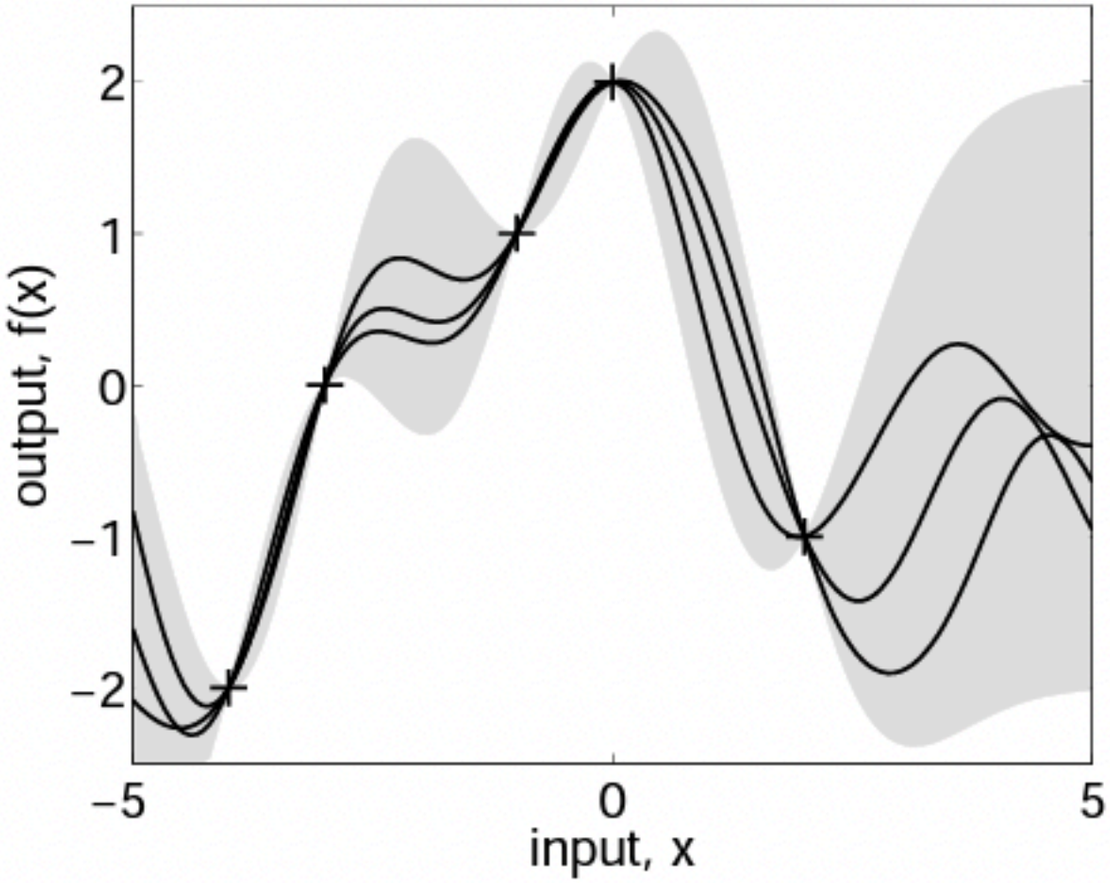
Rasmussen & Williams 2005

Gaussian Processes

$$\mathbf{f}_*|X_*, X, \mathbf{f} \sim \mathcal{N}\big(K(X_*, X)K(X, X)^{-1}\mathbf{f},$$
$$K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)\big).$$



(a), prior



(b), posterior

Rasmussen & Williams 2005

Gaussian Processes — Kernels

Much of the secret sauce of Gaussian process regression is in the choice of kernel.

Common kernel choices are the **Radial basis function (RBF)** AKA a Gaussian…

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$

and a common generalization called the **Matern** kernel, which converges to a Gaussian as $\nu \to \infty$

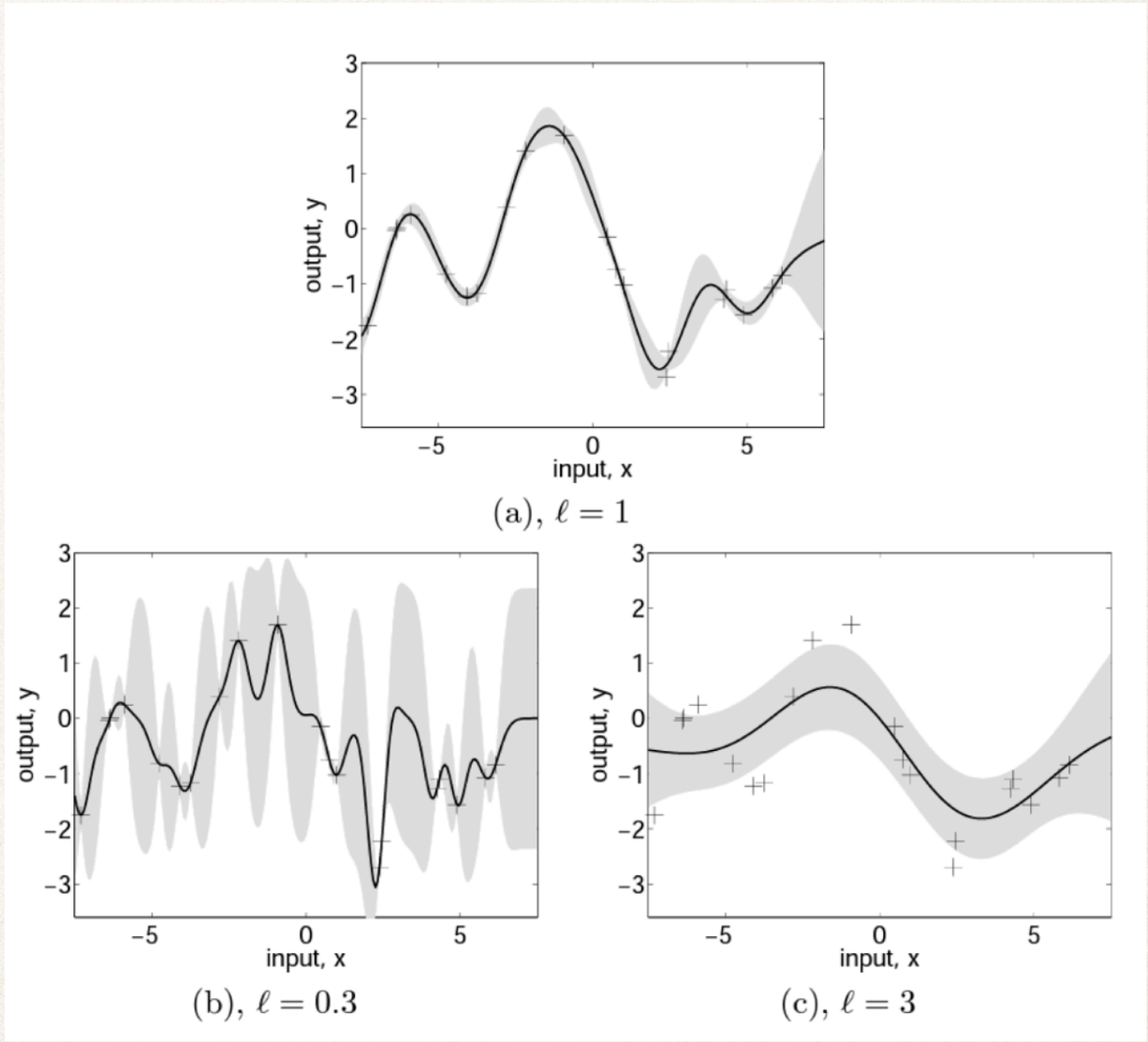
$$C_\nu(d) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\frac{d}{\rho}\right)^\nu K_\nu\left(\sqrt{2\nu}\frac{d}{\rho}\right)$$

and K_ν is the modified Bessel function of the second kind. Common choices for ν are $5/2$ and $3/2$ due to simplicity of expression for kernel in these cases.

Gaussian Processes — Hyper parameter Optimization

Hyper-parameter optimization of the kernel function is often the most important and most expensive component of GP regression.

$$k_y(x_p, x_q) = \sigma_f^2 \exp\left(-\frac{1}{2\ell^2}(x_p - x_q)^2\right) + \sigma_n^2 \delta_{pq}.$$



Common algorithms with give maximum likelihood estimates of these, but posterior distributions of them can be useful for robust error estimates from the GP.

Rasmussen & Williams 2005

Gaussian Processes — Scalability

Gaussian process regression is computationally **limited by the computational expense of matrix inversion**:

$$\mathbf{f}_*|X_*, X, \mathbf{f} \sim \mathcal{N}\big(K(X_*, X)K(X, X)^{-1}\mathbf{f}, \\ K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)\big).$$

if $K(X, X)$ is not full rank, then this issue can be partially alleviated. A common example is if $\frac{\partial f}{\partial \theta_i \partial \theta_j} = 0$ for parameters θ_i , θ_j then $K(X, X)$ is block diagonal and "Kroneker" representations of the kernel can be used.

There are also a number of "approximate" GP models on the market, some of which scale quite well. See, e.g. KISS-GP, LOVE, SKIP…

Gaussian Processes — Summary

Gaussian processes have become the most common choice for emulation in cosmology.

Pros:

- Highly flexible
- Relatively few parameters
- Built in uncertainty estimates

Cons:

- Not as scalable to many dimensions/large amounts of training data as other options.
  - training and prediction times scale with N
- Kernel choice and hyper-parameter optimization can be finicky.

Gaussian Processes — References

Common libraries:

- GPy: https://github.com/SheffieldML/GPy
- GPyTorch: https://github.com/cornellius-gp/gpytorch

A few useful papers/textbooks:

- Rasmussen & Williams 2005
- KISS-GP (http://proceedings.mlr.press/v37/wilson15.pdf)
- SKIP (https://arxiv.org/pdf/1802.08903.pdf)

Polynomial Chaos Expansions

A common alternative to GPs. Trade flexibility for simplicity speed. A PCE of order p approximates a function, $A(\mathbf{\Omega})$, as

$$\tilde{A}(\mathbf{\Omega}) = \sum_{|\beta| \leq p} \eta_\beta \Psi_\beta(\mathbf{\Omega})$$

where η_β are expansion coefficients, β, is a multi-index, and $\Psi_\beta(\mathbf{\Omega})$ are polynomials chosen to be orthonormal with respect to the PDF, $p(\mathbf{\Omega})$.

For uniformly or gaussian distributed inputs over $[-1,1]^D$, this results in the Legendre and Hermite polynomials respectively.

Polynomial Chaos Expansions — Truncation

Once a polynomial order is specified, fitting a PCE is a simple linear regression problem. All of the work in fitting PCEs is thus in specifying at what order to truncate the polynomial expansion.

If $\mathscr{A}^{d,p} = \{\beta \in \mathbb{N}^d, ||\alpha||_1 \leq p\}$ then

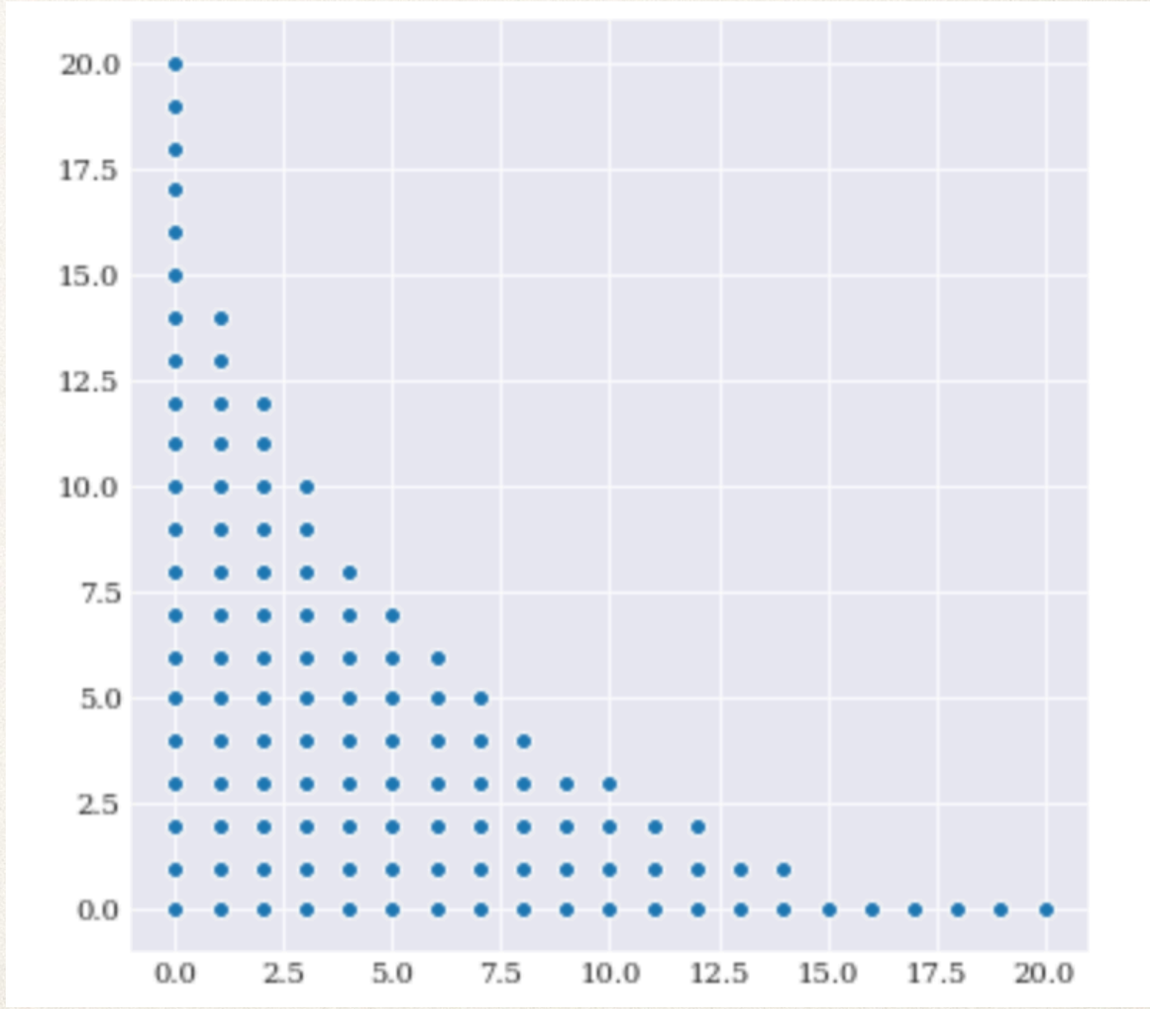
$$\text{card}(\mathscr{A}^{d,p}) = \binom{p+d}{p}$$

Additional truncations often applied:

$\mathscr{A}^{d,p,q} = \{\beta \in \mathscr{A}^{d,p}, ||\alpha||_q \leq p\}$ where

$$||\beta||_q = \left(\sum_{i=1}^{d} \beta_i^q\right)^{1/q}$$

q=0.6 hyperbolic trunc.



Polynomial Chaos Expansions — Summary

PCE is becoming a more commonly adopted strategy for relatively low dimensional problems. Often combined with PCA to reduce number of polynomials required.

Pros:

- Extremely easy to fit.
- Evaluation time independent of N.

Cons:

- Not as flexible as GPs.

Common libraries:

- Chaospy: https://github.com/jonathf/chaospy
- UQLab: https://sudret.ibk.ethz.ch/software/uqlab.html
- Uncertainpy: https://github.com/simetenn/uncertainpy

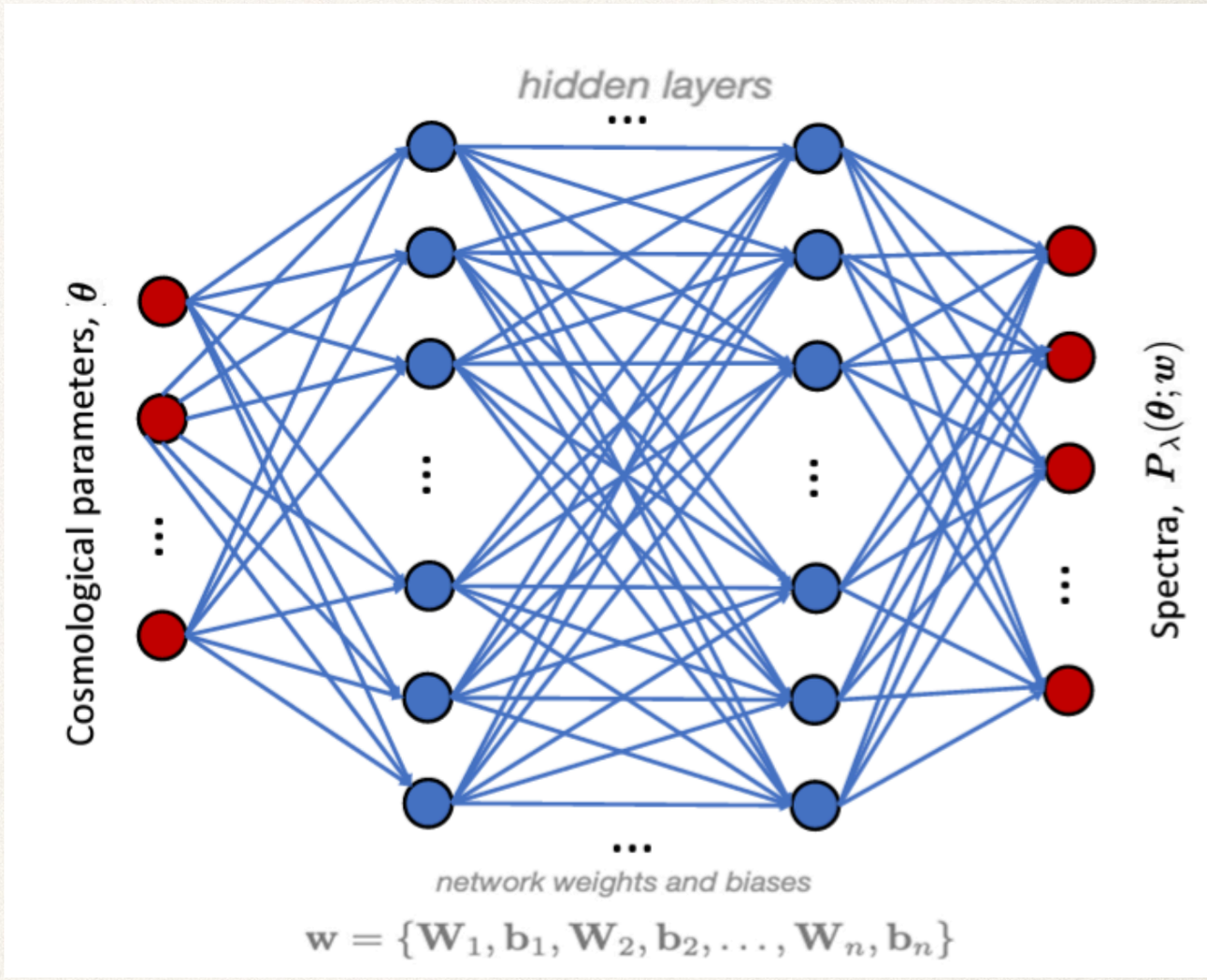
Neural Networks

Pros:

- Highly flexible.
- (can be) very fast at evaluation time.

Cons:

- Very easy to overfit.
- Requires large amounts of training data.
- Training very temperamental.



Spurio Mancini et al. 2021

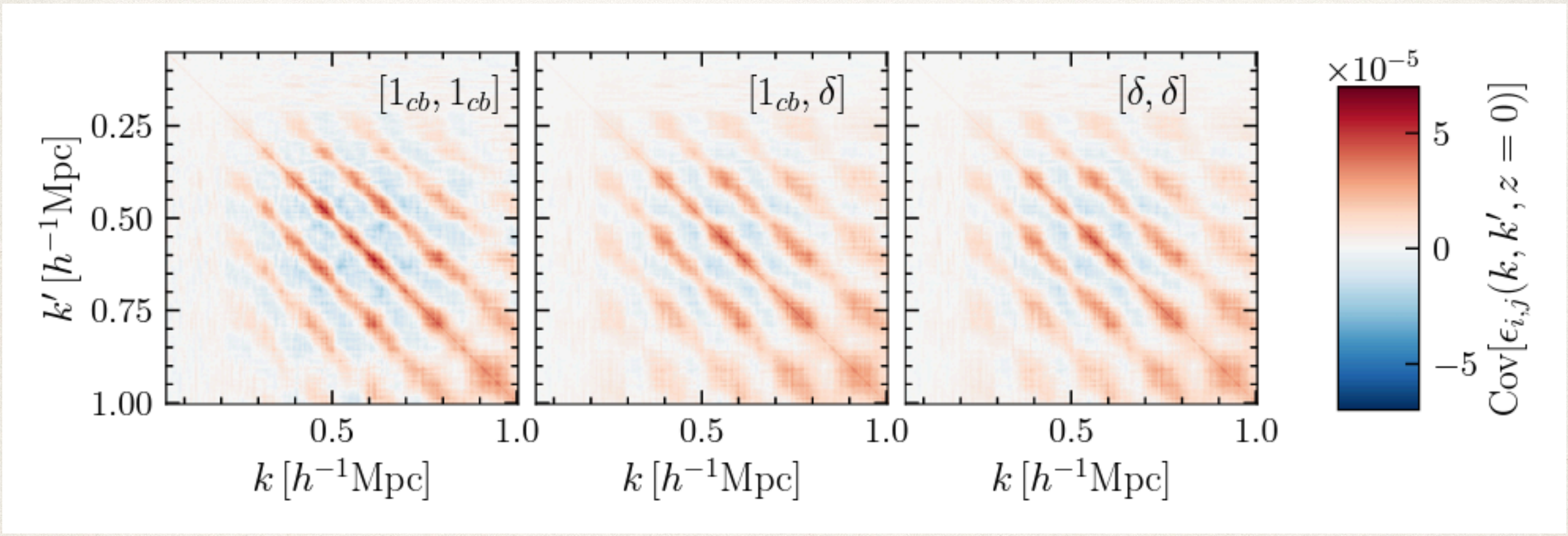
Outline

- Computational Challenges in Cosmic Frontier Theory
- Experimental design
  - Parameter space sampling, bayesian optimization, variance-reduction
- Emulation techniques and when to use them
  - Gaussian Processes, Polynomial Chaos Expansions, and Neural Networks
- I trained an emulator, now what?

Error quantification

Emulator error structure can be highly non-trivial.

If you're making your emulator publicly available, it should come with a quantitative way to incorporate emulator error into analyses. You can usually do this for free via k-fold cross validation.

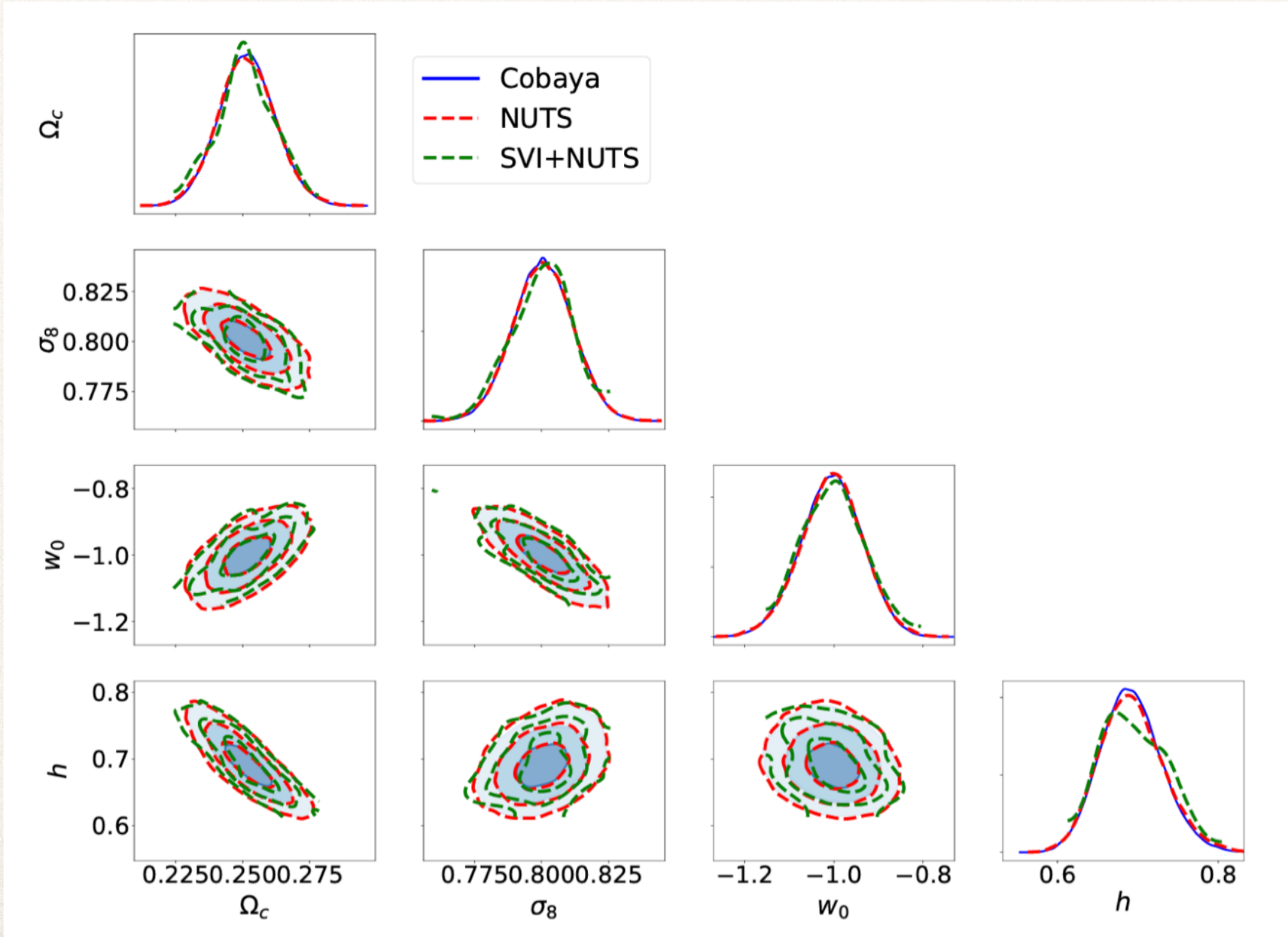


DeRose et al. 2023

Highlights — Model Differentiability

One major upside to emulators is that they are often very trivially differentiable. This enables fast inference via HMC.

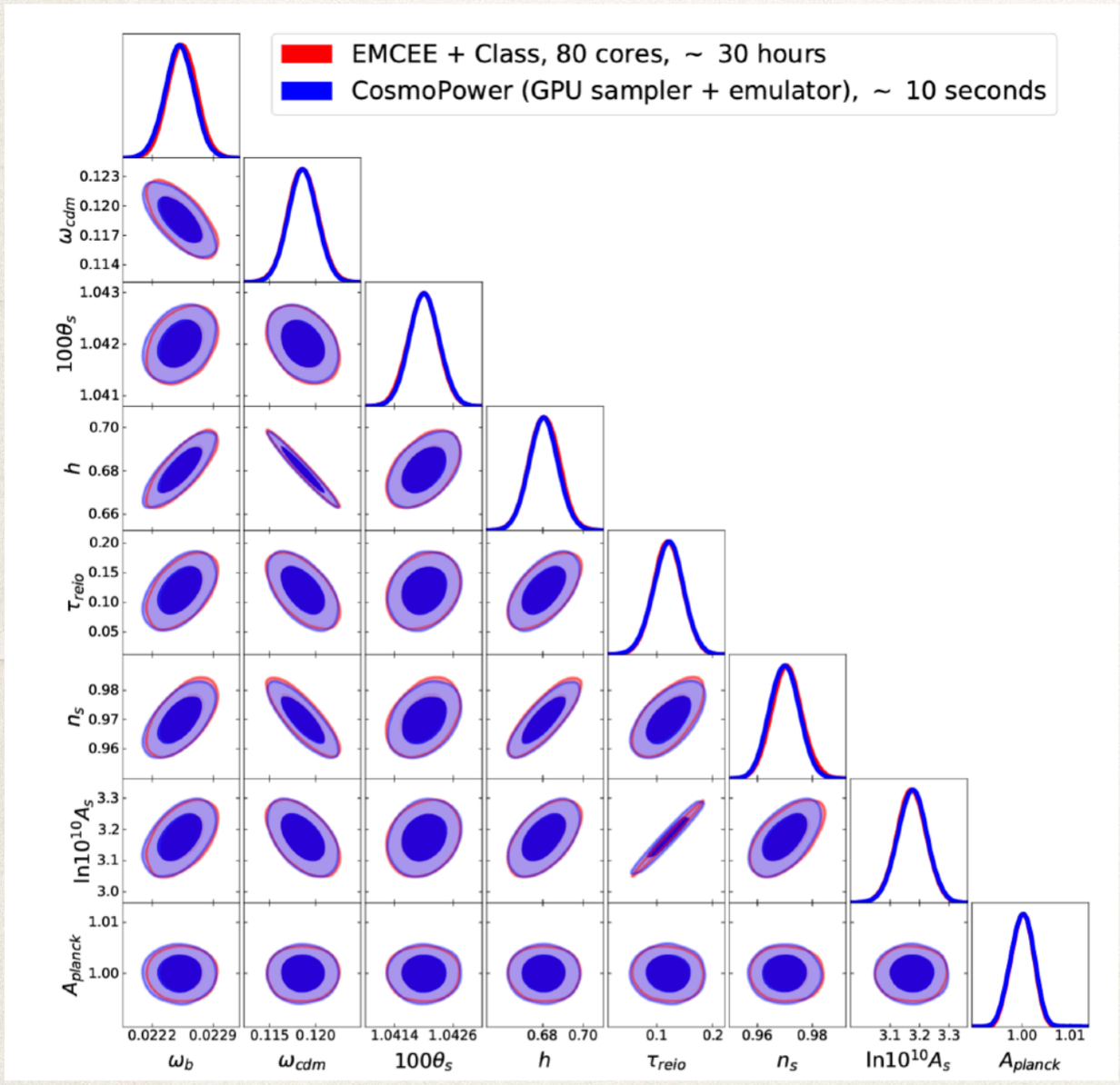
NUTS posterior required 10x fewer samples that MH (Cobaya)



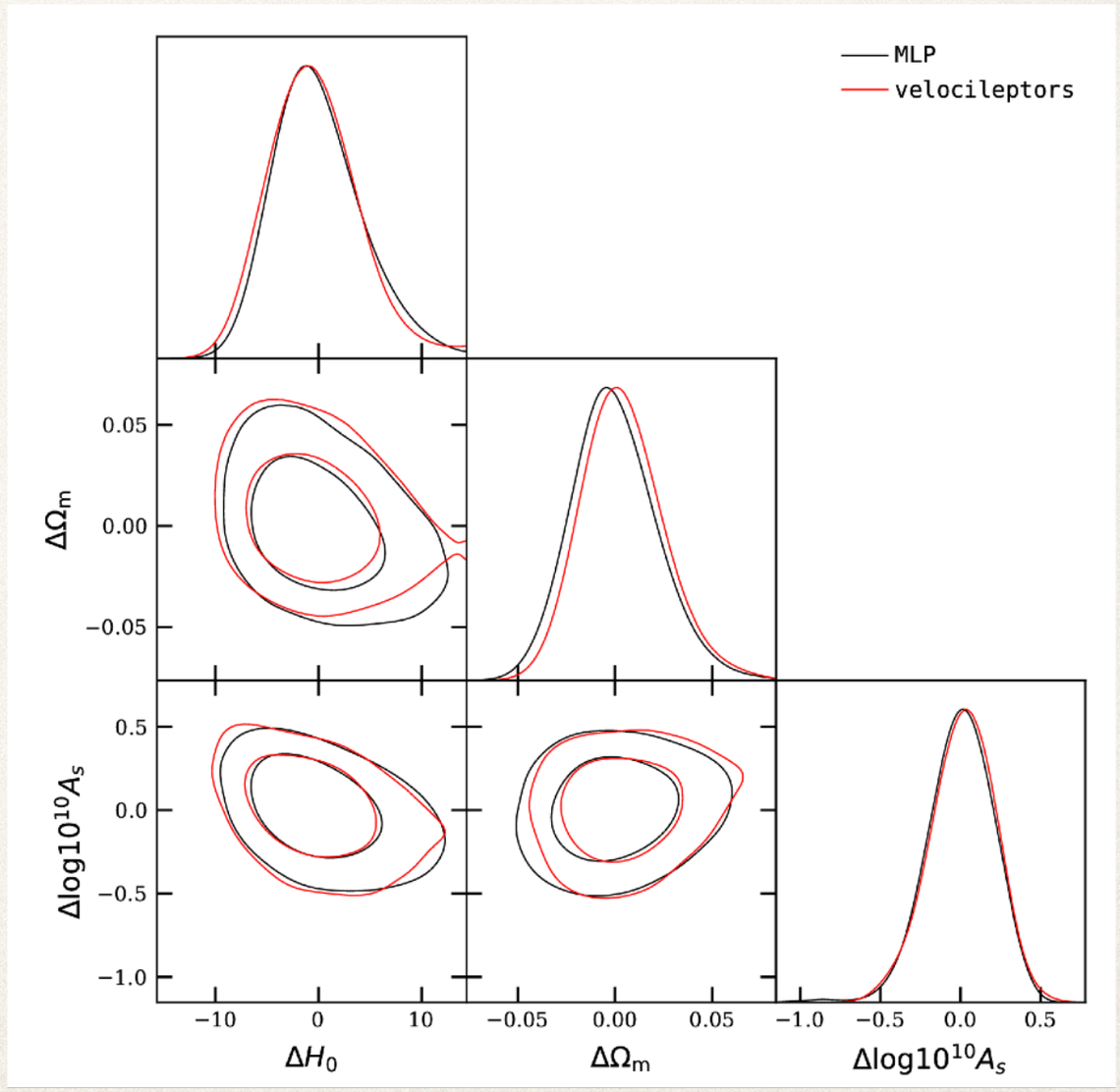
Campagne et al 2023

Highlights — Boltzmann Solvers/Perturbation Theory

Acceleration of expensive theoretical calculations is perfect application for simple NN architectures.



Spurio-Mancini et al. 2021

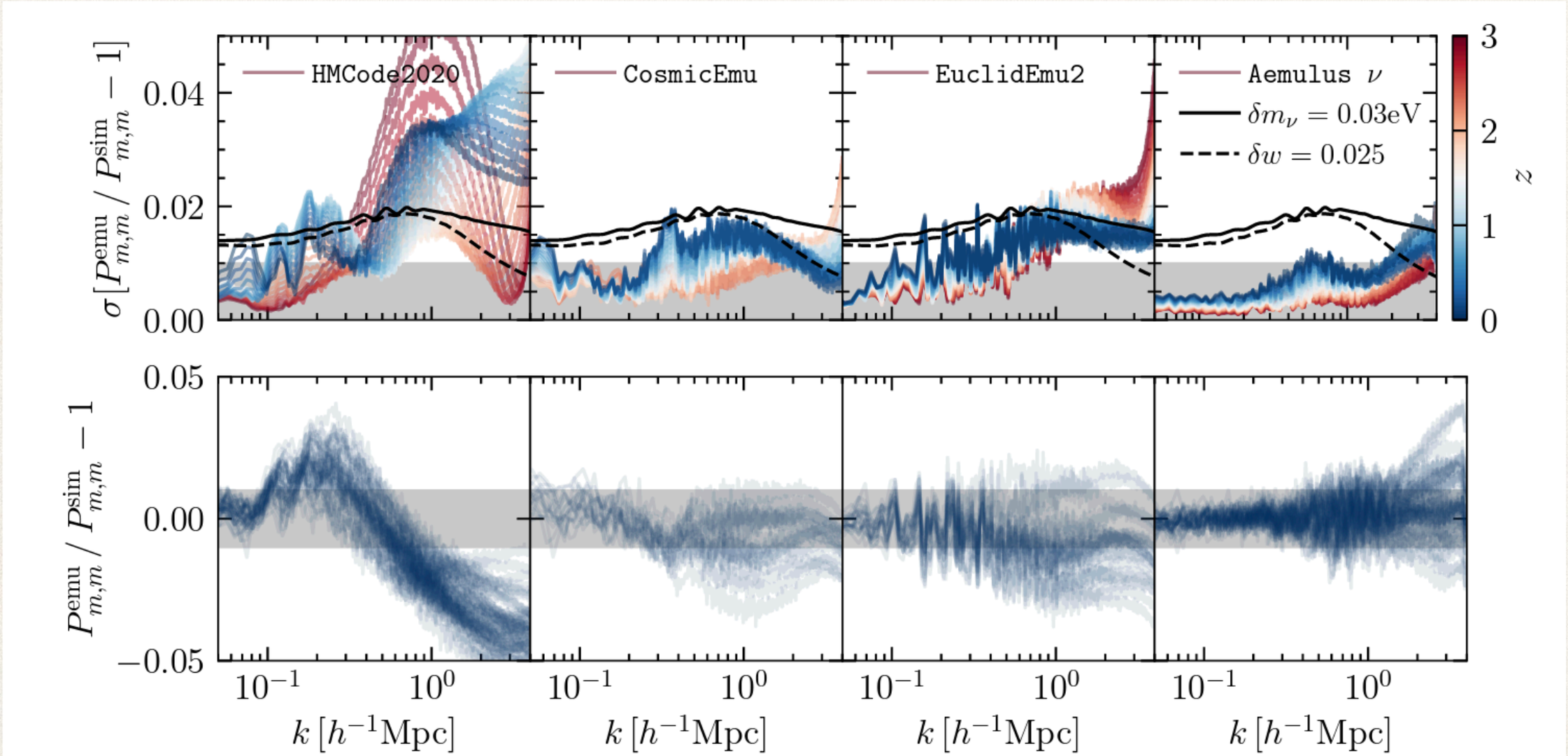


DeRose et al. 2021

Highlights — Non-linear matter power spectrum

Emulators have lead to a huge leap in how accurately we can model the matter power spectrum.

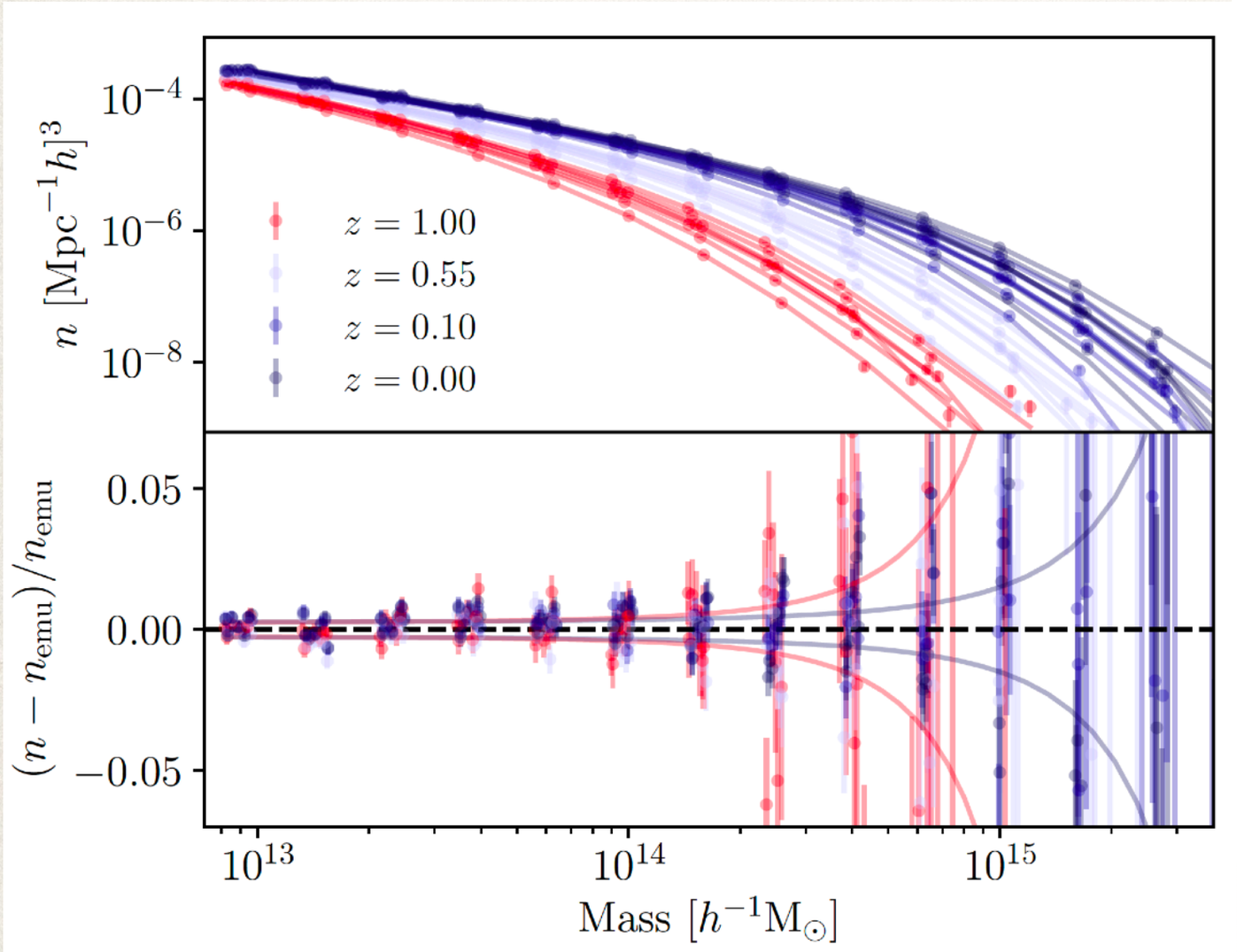
Semi-analytic GP PCA+PCE PCA+PCE



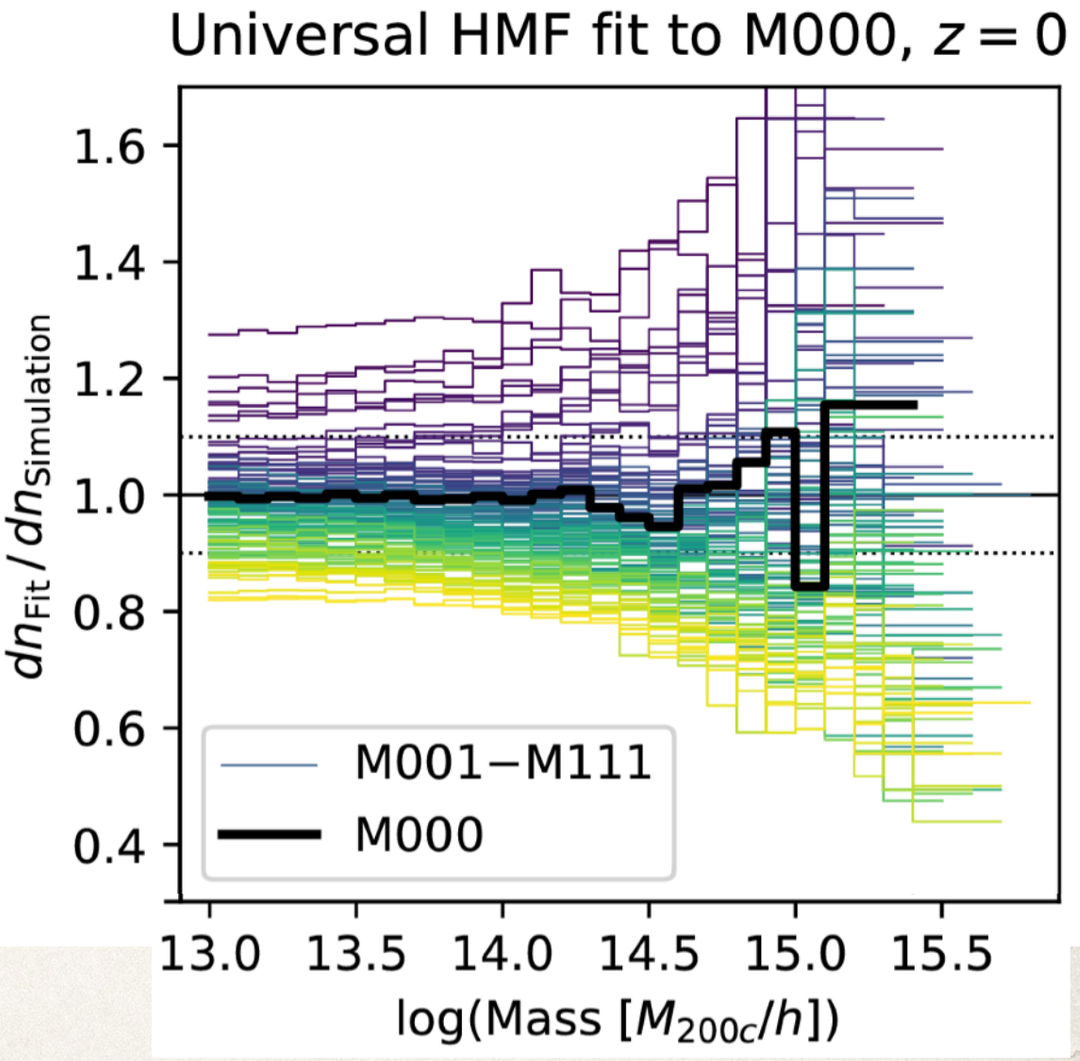
DeRose et al 2023

Highlights — Dark matter halo statistics

And dark matter halo statistics…



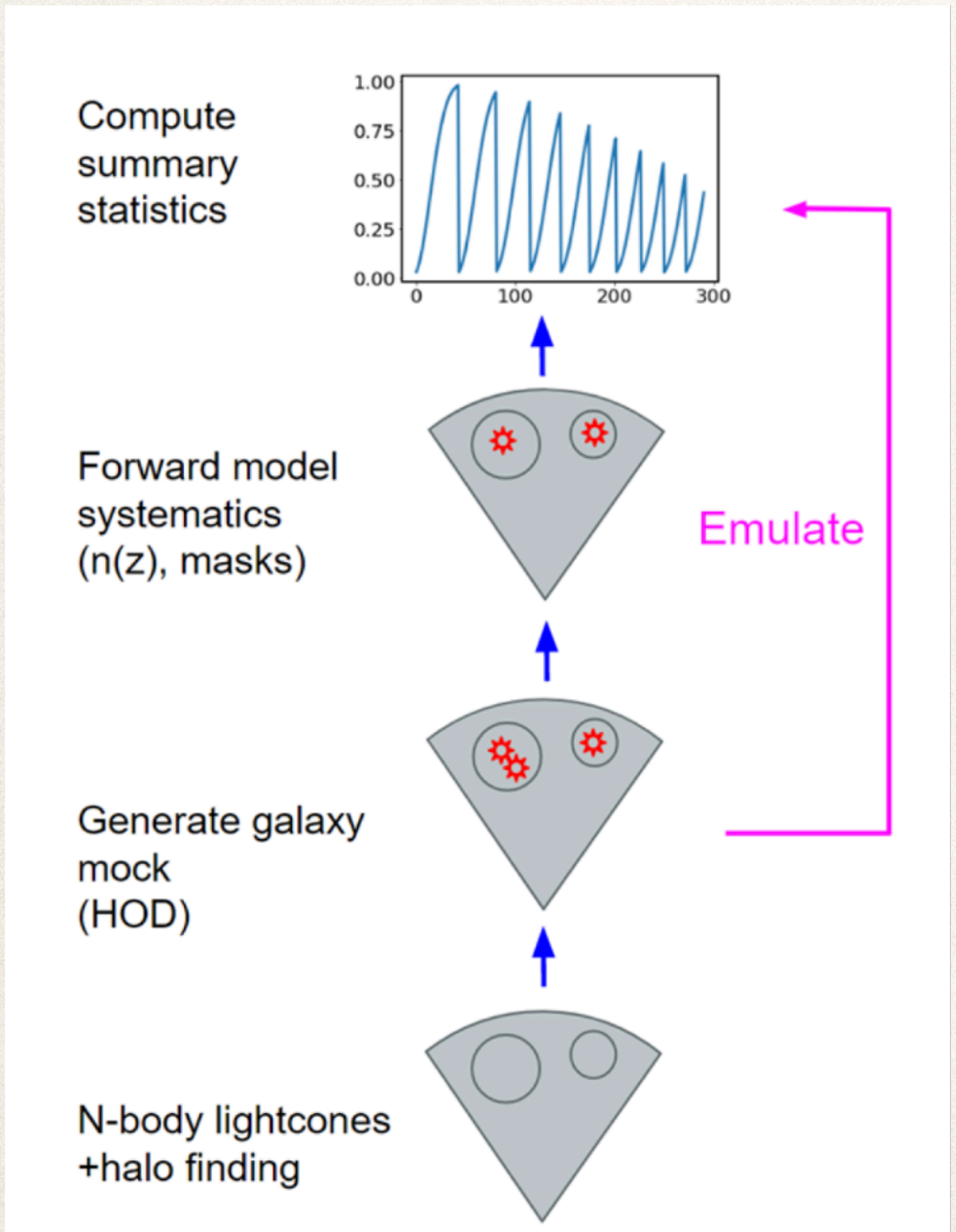
McClintock et al 2018

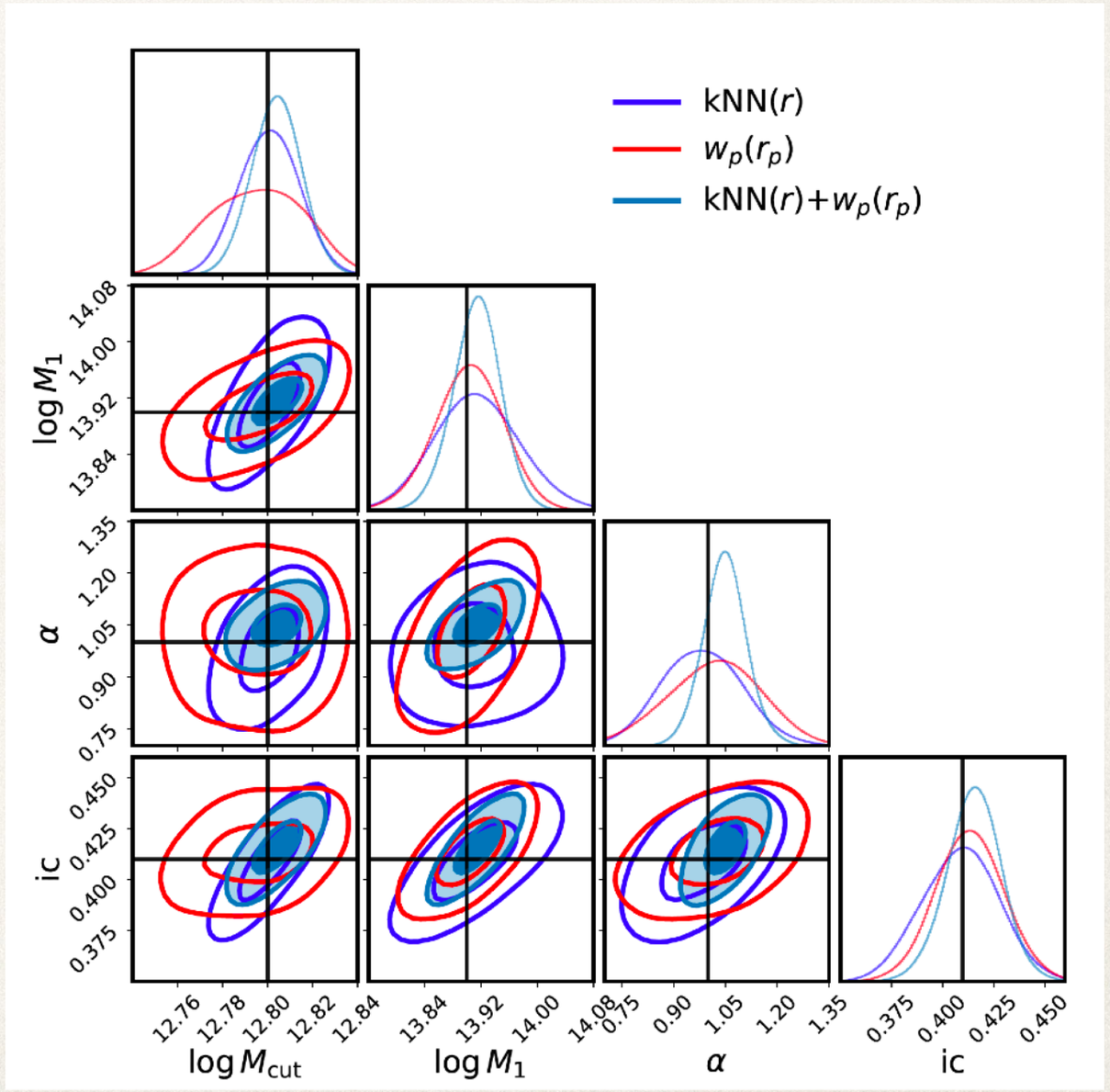


Bocquet et al 2020

Highlights — Galaxy Clustering

Emulating outputs of lightcone simulations with observational systematics applied enables use of exotic, and more constraining statistics.

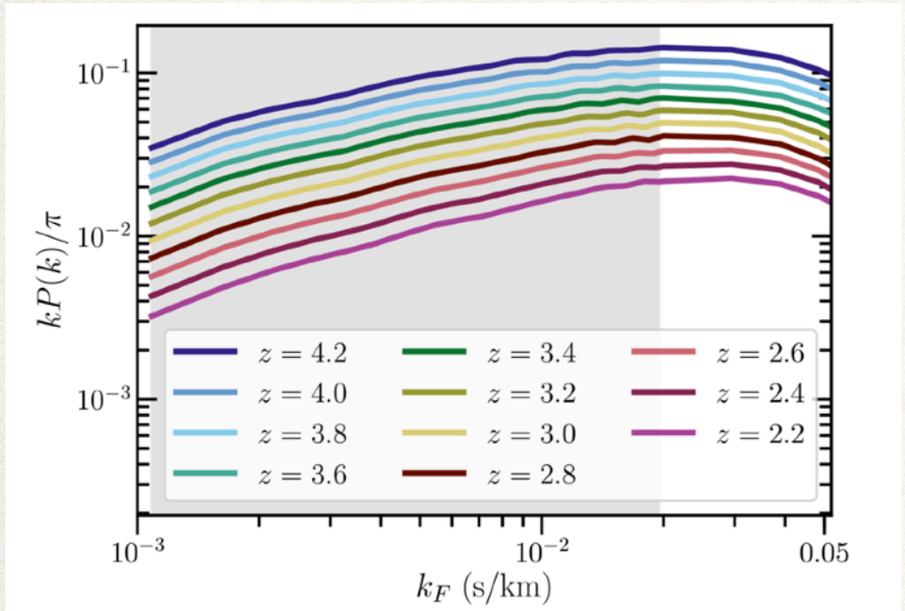


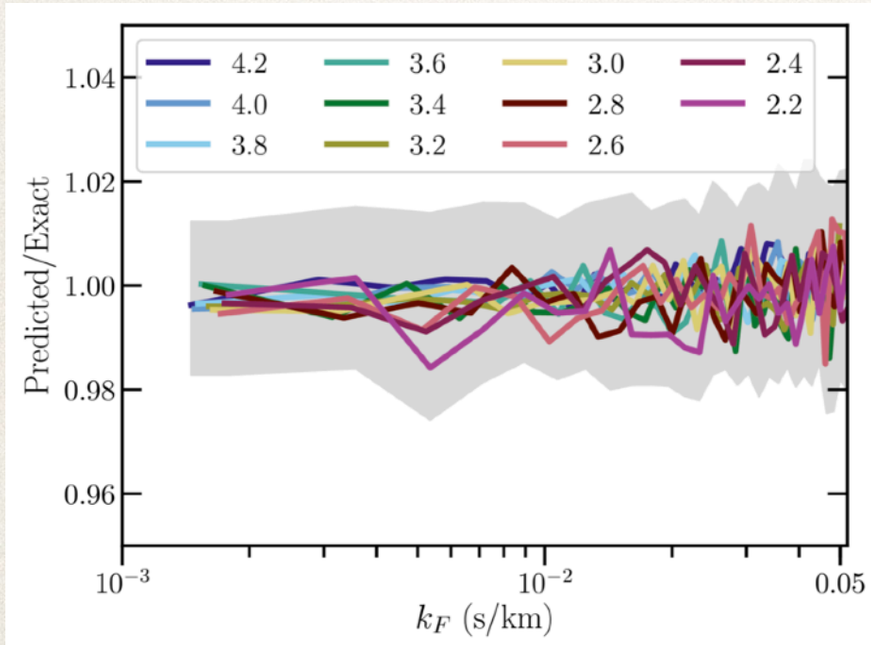


Yuan et al. 2023

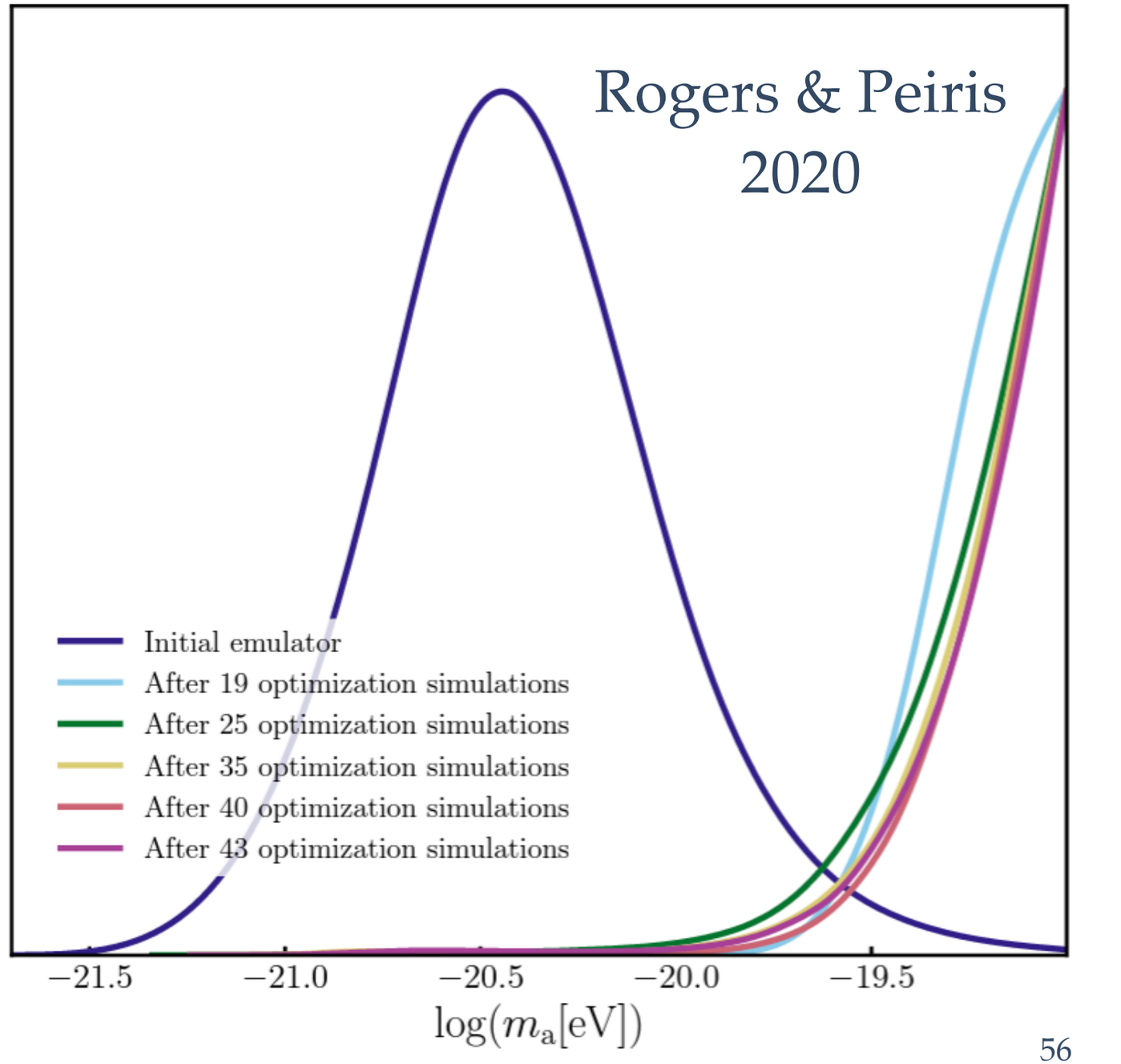
Highlights — Lyman Alpha Forest

Emulating hydrodynamical simulation outputs enables use of Lyman alpha forest to constrain light relics.

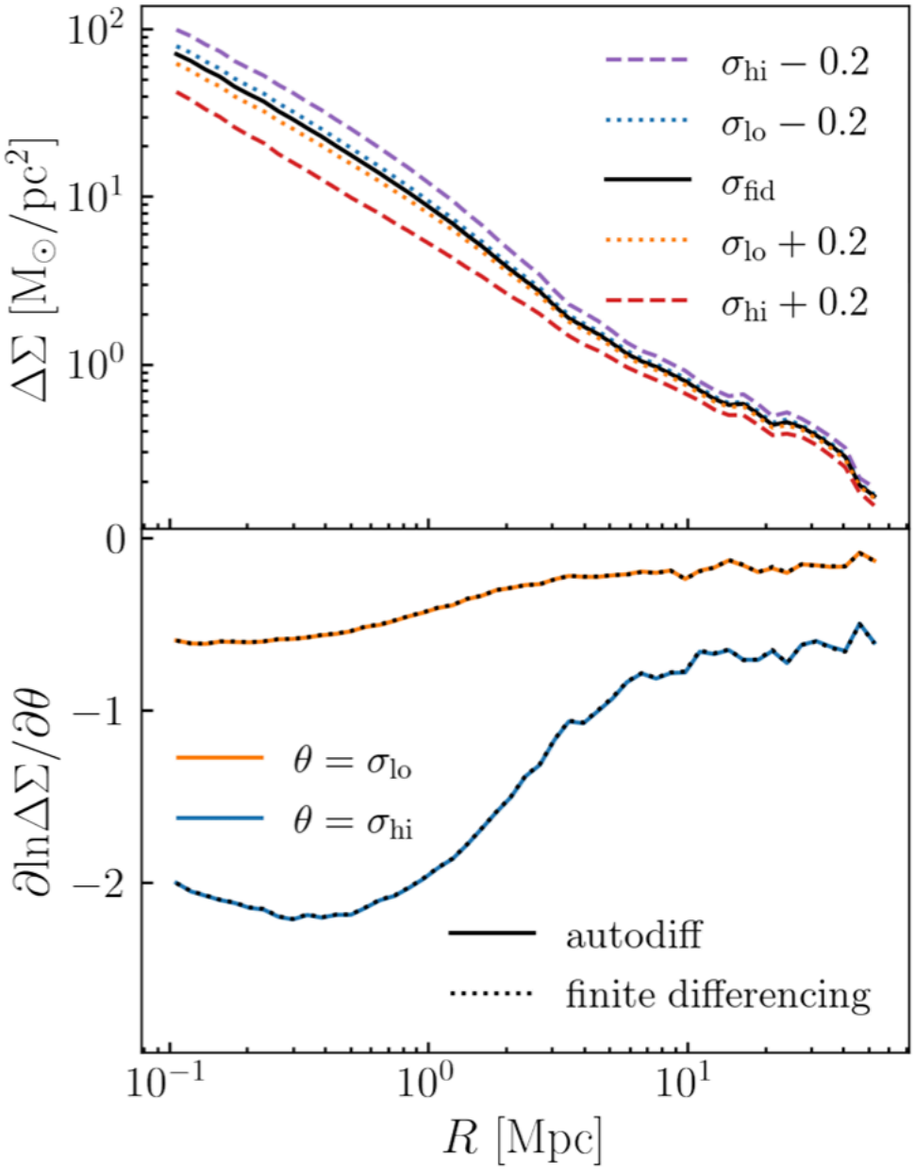




Bird et al. 2018

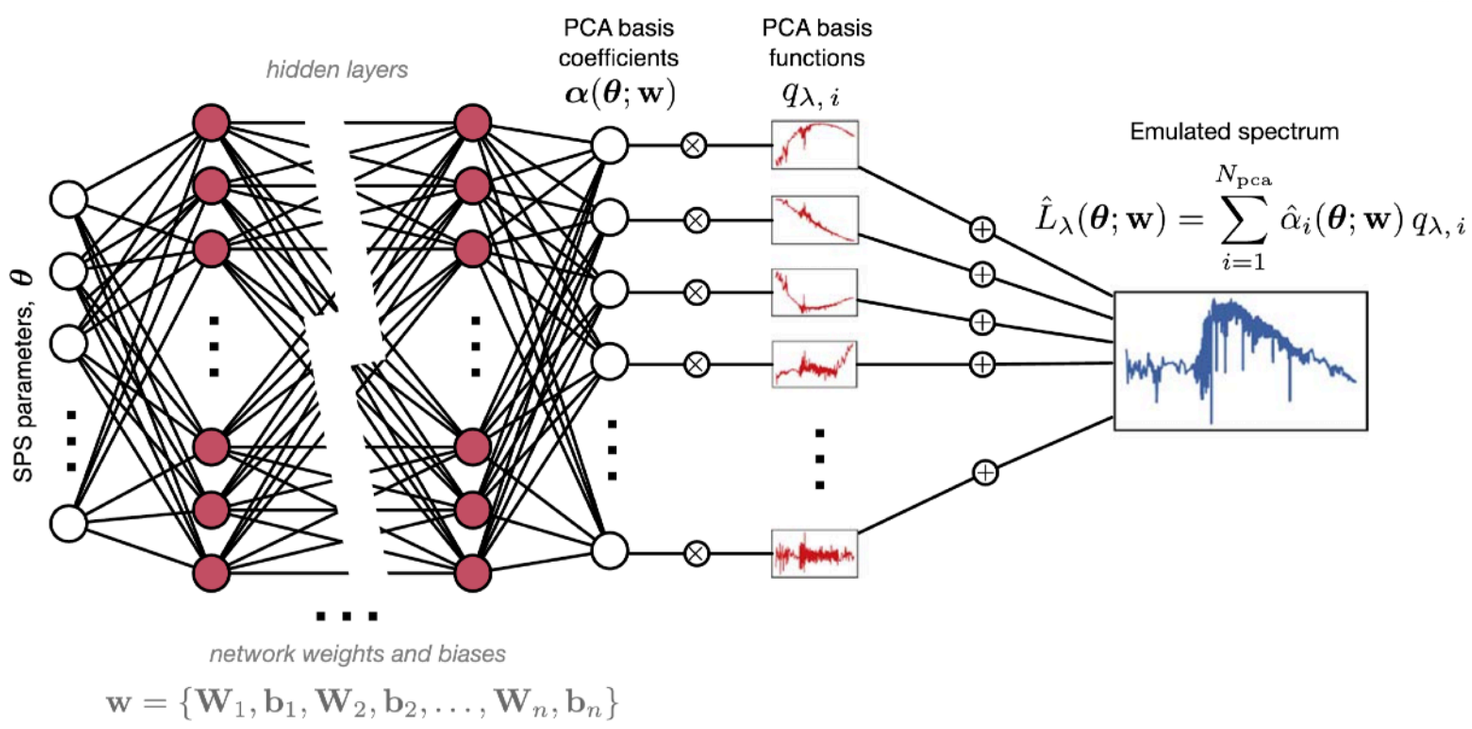


Highlights — Galaxy Formation Models



Hearin et al. 2022

Explosion of applications to emulating complex galaxy formation models (e.g. Semi-analytic models and Stellar population synthesis)



Alsing et al. 2022

Summary

- Choice of how to sample training data, and how to preprocess that training data is often the most important determining factor in the accuracy of emulators.

- Gaussian processes are the workhorse of CF emulation, but Polynomial Chaos Expansions are worth considering for low dimensional problems.

- Neural network emulation is becoming essential for fast analytic predictions.

- The concept of emulation has revolutionized the accuracy that we can attain in CF models.