# Accelerating MCMC for Cosmological Parameter Inference

Minas Karamanis

(he/him)

minas.karamanis@ed.ac.uk

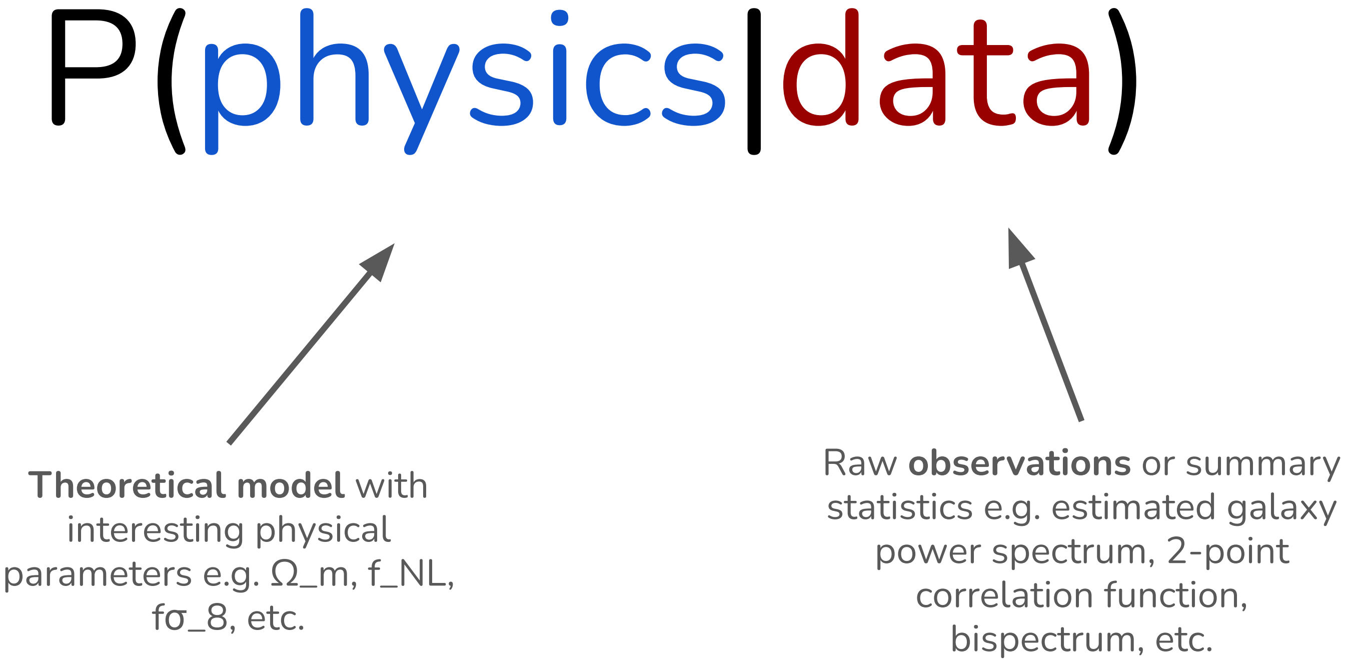
github.com/minaskar

Cosmology from Home 2021

# Outline

- Bayesian parameter inference
- Markov chain Monte Carlo (MCMC)
- Preconditioning
- Normalising flows (NF)
- Normalising flow preconditioning (NFP)
- Antithetic sampling
- Empirical evaluation
- Conclusions

# A bit of context...



P(physics|data)

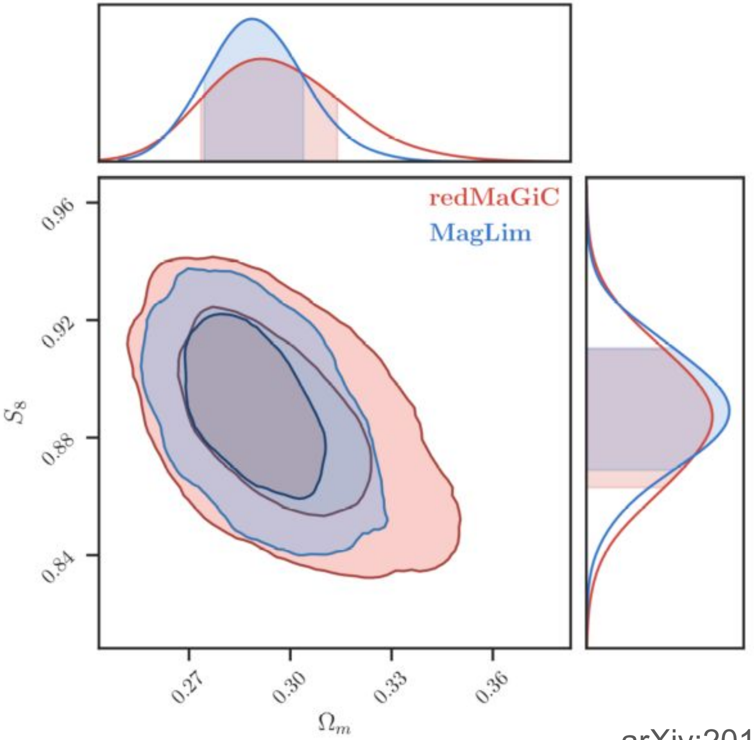
**Theoretical model** with interesting physical parameters e.g. Ω_m, f_NL, fσ_8, etc.

Raw **observations** or summary statistics e.g. estimated galaxy power spectrum, 2-point correlation function, bispectrum, etc.

# Why is the Posterior distribution so useful?

One can use the Posterior to compute expectation values (a.k.a. integrals):

- Mean values of parameters,
- Variance of parameters,
- 1D marginal Posteriors,
- 2D marginal Posteriors.

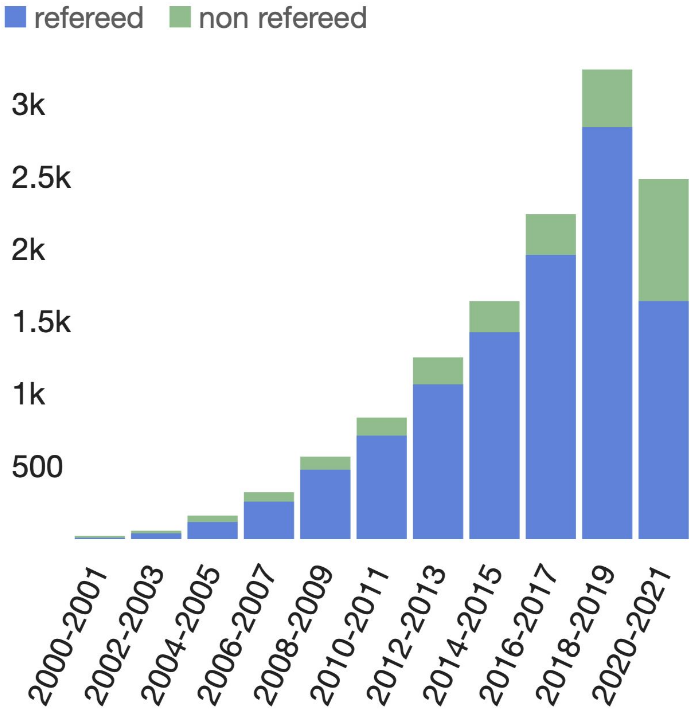


arXiv:2011.03411

# Can we compute the Posterior then?

**Almost always no**, but we can “sample” from it using:

- Rejection sampling,
- Importance sampling,
- **Markov Chain Monte Carlo (MCMC)**,
- Nested sampling,
- Others.

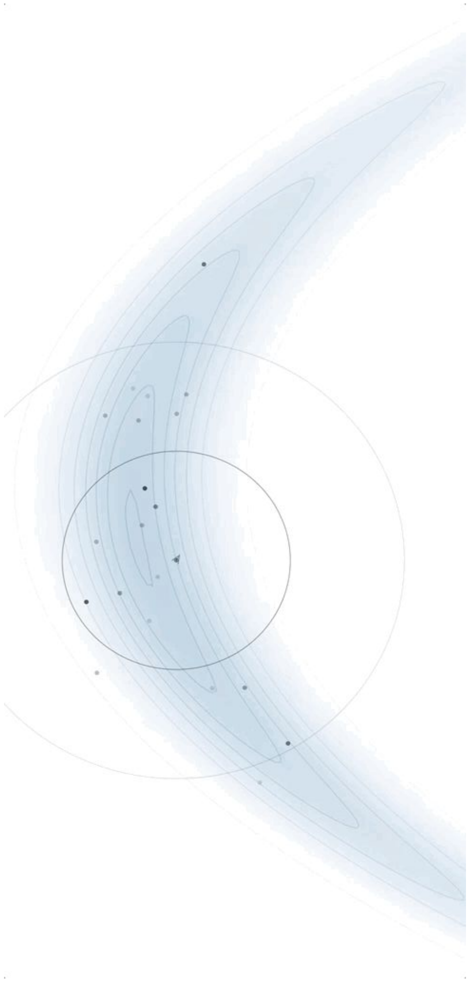


# Markov Chain Monte Carlo (MCMC)

MCMC methods generate samples (chain) from a distribution by **locally** exploring the posterior mass.

MCMC accomplishes this by creating a chain of samples $\boldsymbol{\theta}_i$ over **n** iterations such that the number of iterations $\mathbf{n}(\boldsymbol{\theta}_i)$ spend in any particular region is proportional to the posterior density in that region.

$$\mathrm{E}[f(\theta)] = \int f(\theta)P(\theta)d\theta = \frac{1}{N}\sum_{i=1}^{N} f(\theta_i)$$



# Preconditioning poorly scaled distributions

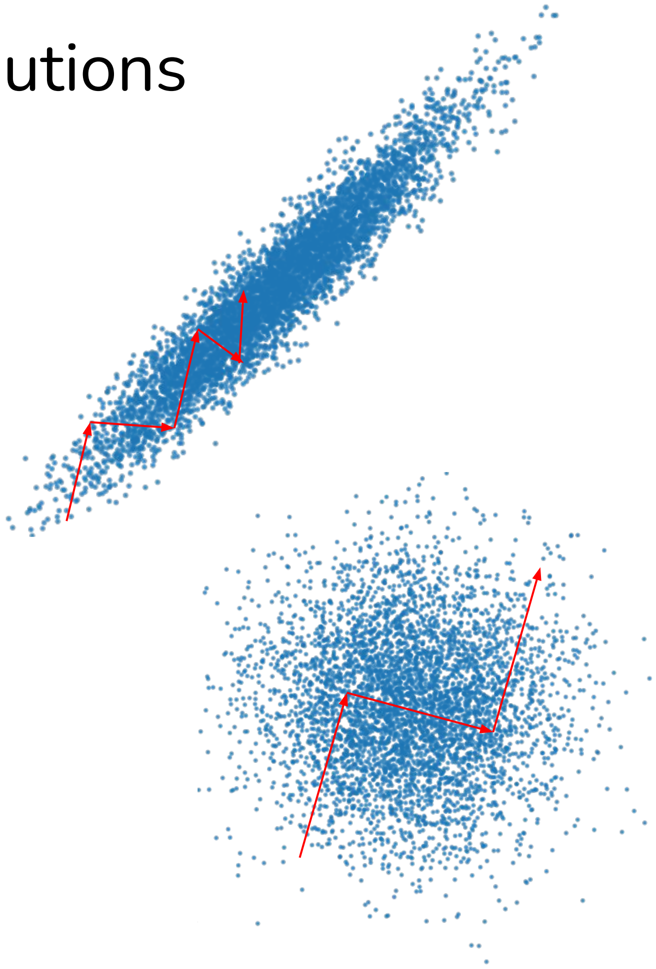
- Poorly scaled distributions can hinder the performance of MCMC,
- Lots of small steps are needed to cross the distribution,
- This leads to high computational costs.
- Preconditioning can help with that:

“A change of variables that attempts to decorrelate the parameters”

e.g. the affine transformation:

$$\mathbf{x} \leftarrow \mathbf{L}^{-1}\mathbf{x}$$

$$\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^{-1}$$



# Finding the optimal parameterisation

Finding the optimal parameterisation is not trivial and often requires expert knowledge about the problem. Examples include:

- Gravitational Wave analyses: Fitting the Chirp mass and the mass ratio instead of the two BH masses.
- Exoplanet analyses: Fitting $\mathit{cos}\omega\sqrt{\mathbf{e}}$ and $\mathit{sin}\omega\sqrt{\mathbf{e}}$ instead of $\mathbf{e}$ and $\omega$.
- BAO analyses: Fitting $\Sigma^2$ instead of $\Sigma$.

**Is there a way to find the optimal parameterisation automatically for every problem?**

# Normalising flows (NF)

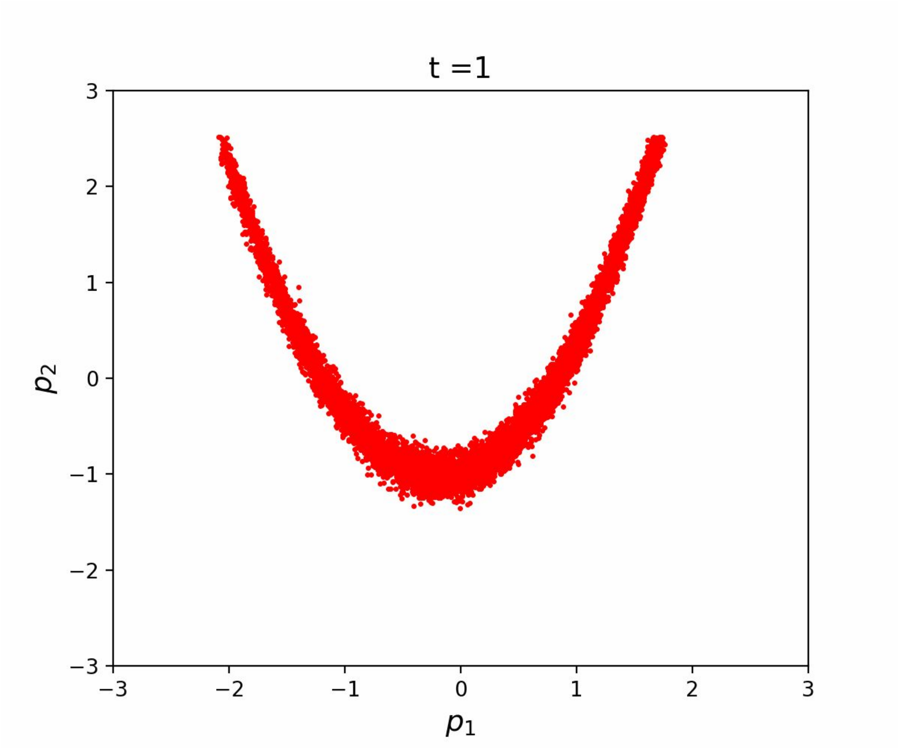
NFs are series of bijective (i.e. invertible) mappings, often parameterised by neural networks.

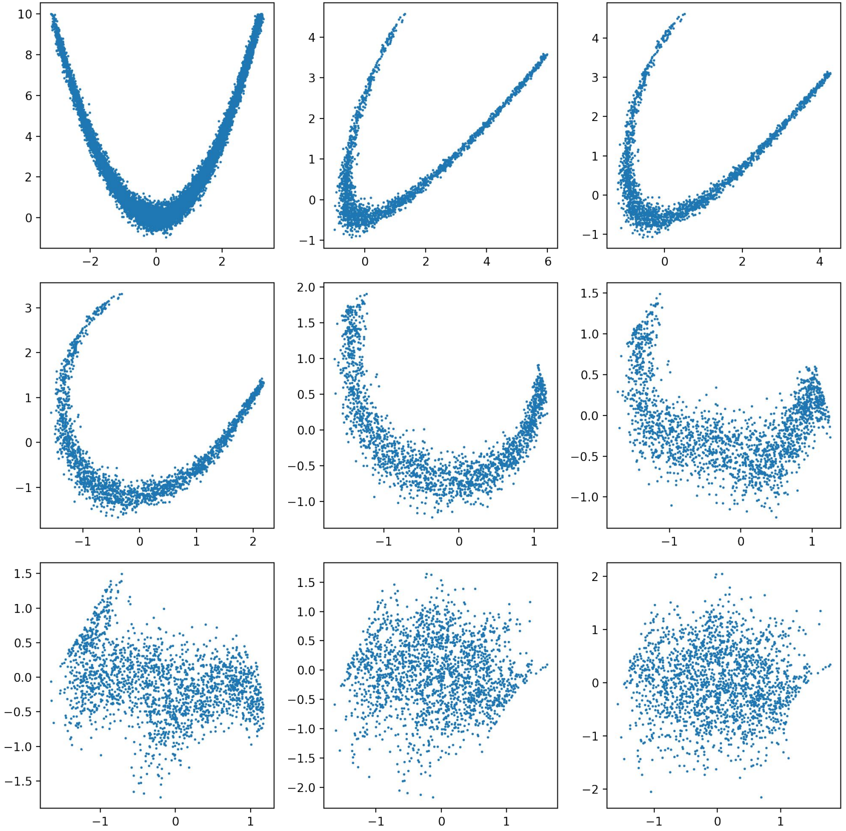
We can transform a probability distribution using an invertible mapping. Let Z and X be two random variables which are related by a mapping $f : \mathbb{R}^n \to \mathbb{R}^n$such that $X = f(Z)$ and $Z = f^{-1}(X)$, then:

$$p_X(\mathbb{X}) = p_Z(\mathbb{Z}) \left| \det \left( \frac{\partial f(\mathbb{Z}))}{\partial \mathbb{Z}} \right) \right|^{-1}$$

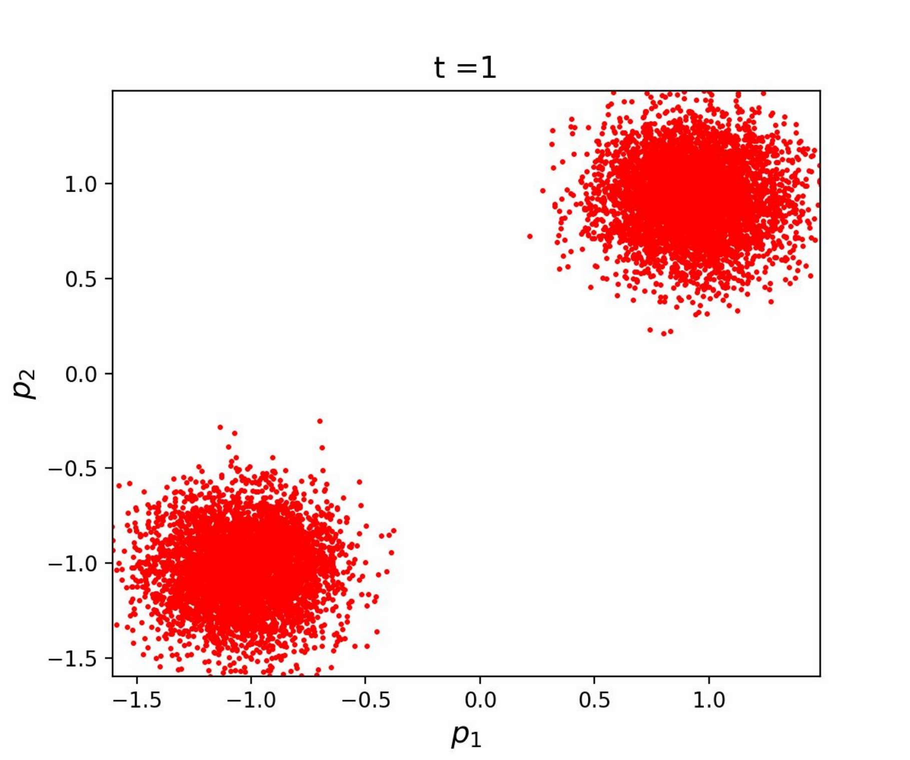
By applying a series of such transformations, one obtains a NF.

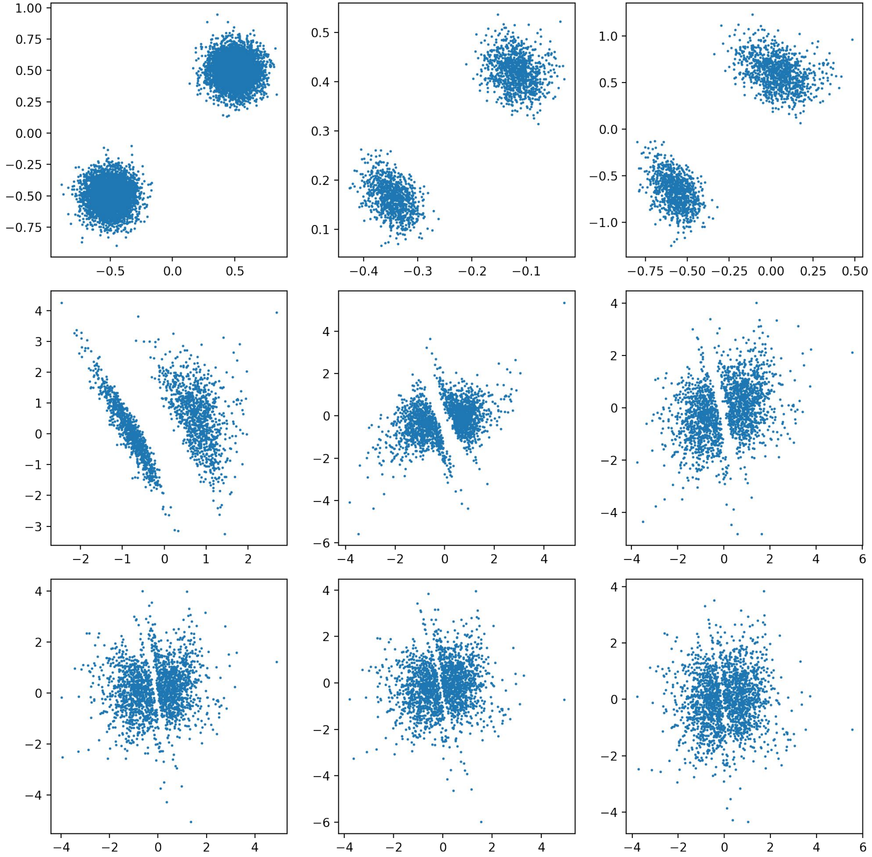
# NF example: 20D Rosenbrock distribution





# NF example: 20D Gaussian Mixture

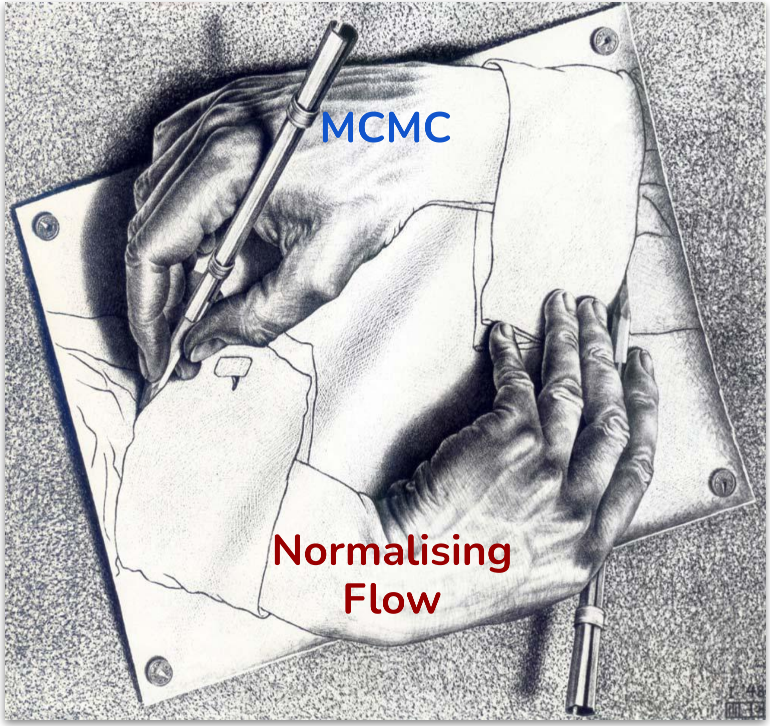




# Normalising Flow Preconditioning (NFP)

1. Use NF to transform any target distribution into a normal distribution.
2. Use the NF forward transformation to precondition the sampler.
3. Use MCMC to sample from the preconditioned distribution.
4. Push the samples through the inverse NF transformation to get the final samples.

In order to train the NF we can use an iterative scheme, gradually improving the accuracy of the transformation.



# Initial approximation

How to build the initial approximation in order to train the NF?

- Short MCMC or Nested sampling run.
- Iterative MCMC scheme, e.g.:
  a. Run MCMC for 1000 iterations/steps,
  b. Train the NF using those samples,
  c. Run MCMC for another 1000 iterations sampling from the transformed distribution,
  d. Re-train the NF using the additional samples,
  e. Check the accuracy of the NF, if it is bad then go to c,
  f. Continue running MCMC until enough samples are collected.

# Antithetic sampling

“Proposing samples on the other side of the mode introduces anticorrelations”

Suppose that you want to estimate $\theta = E(Y)$

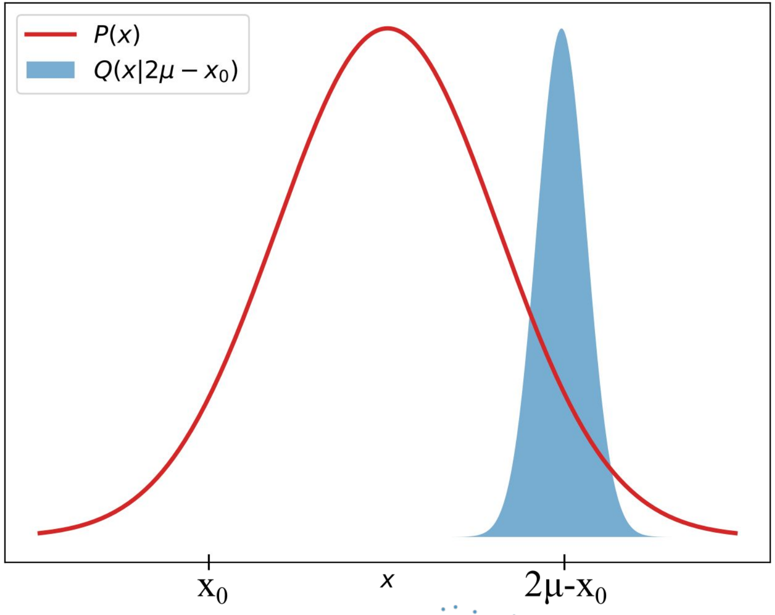
You have generated 2 samples $Y_1$ and $Y_2$

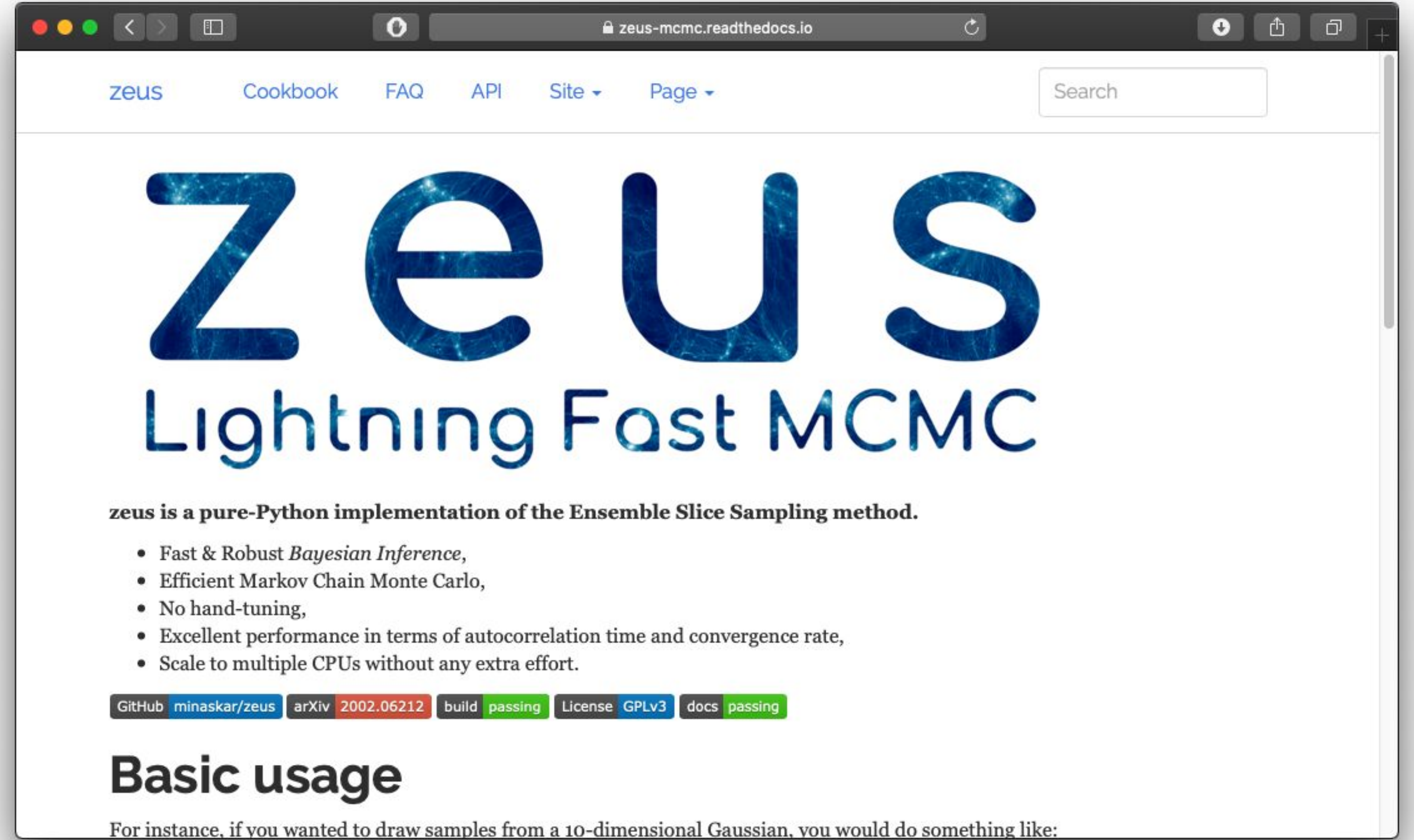
An unbiased estimate is $\hat{\theta} = \frac{\hat{\theta}_1 + \hat{\theta}_2}{2}$

The variance is

$$Var(\hat{\theta}) = \frac{Var(Y_1) + Var(Y_2) + 2Cov(Y_1, Y_2)}{4}$$

Which is reduced if the covariance is negative





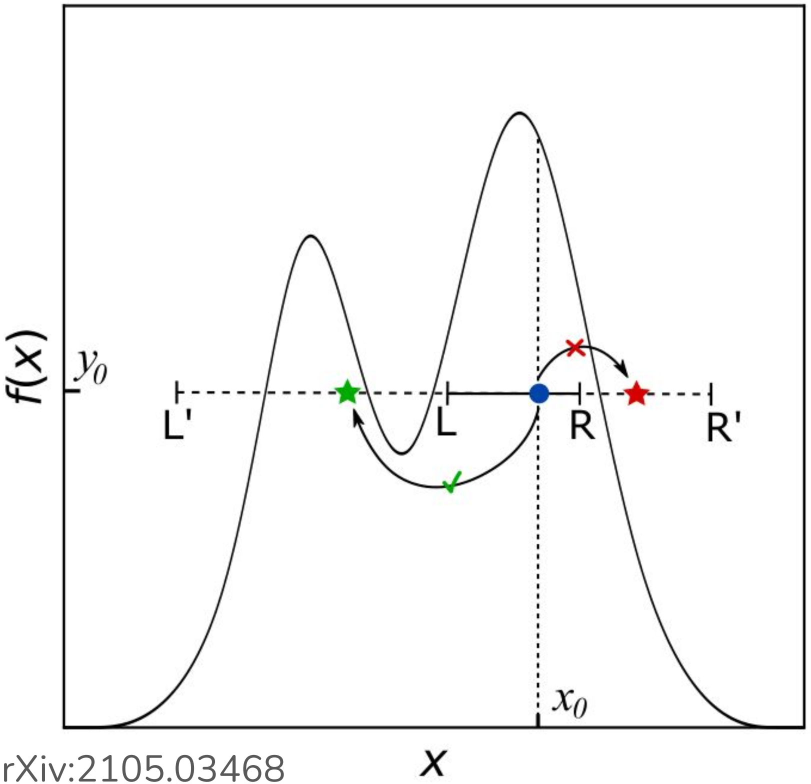
**zeus is a pure-Python implementation of the Ensemble Slice Sampling method.**

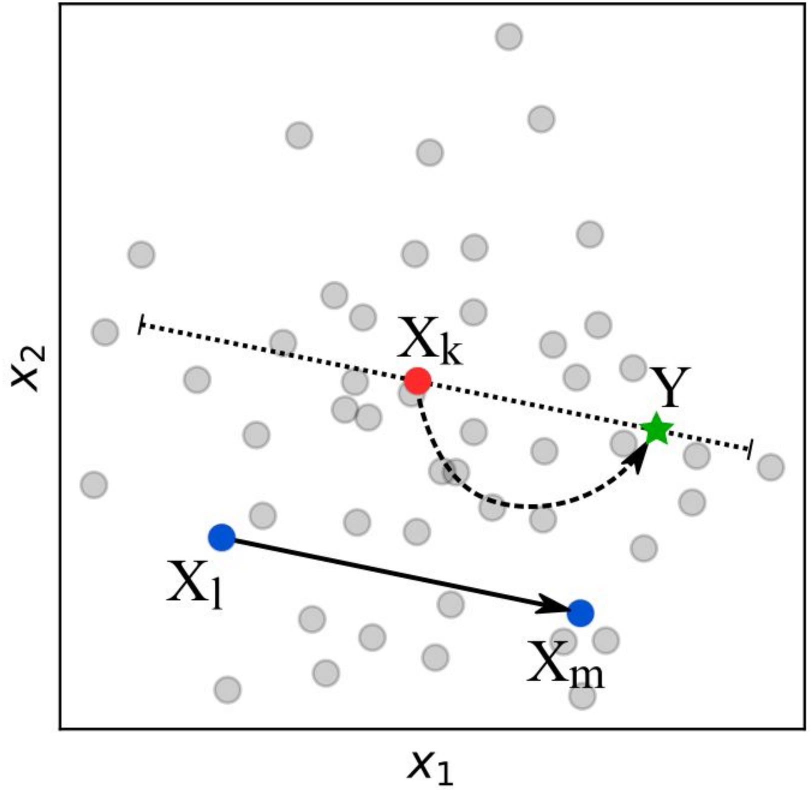
- Fast & Robust *Bayesian Inference*,
- Efficient Markov Chain Monte Carlo,
- No hand-tuning,
- Excellent performance in terms of autocorrelation time and convergence rate,
- Scale to multiple CPUs without any extra effort.

# Basic usage

For instance, if you wanted to draw samples from a 10-dimensional Gaussian, you would do something like:

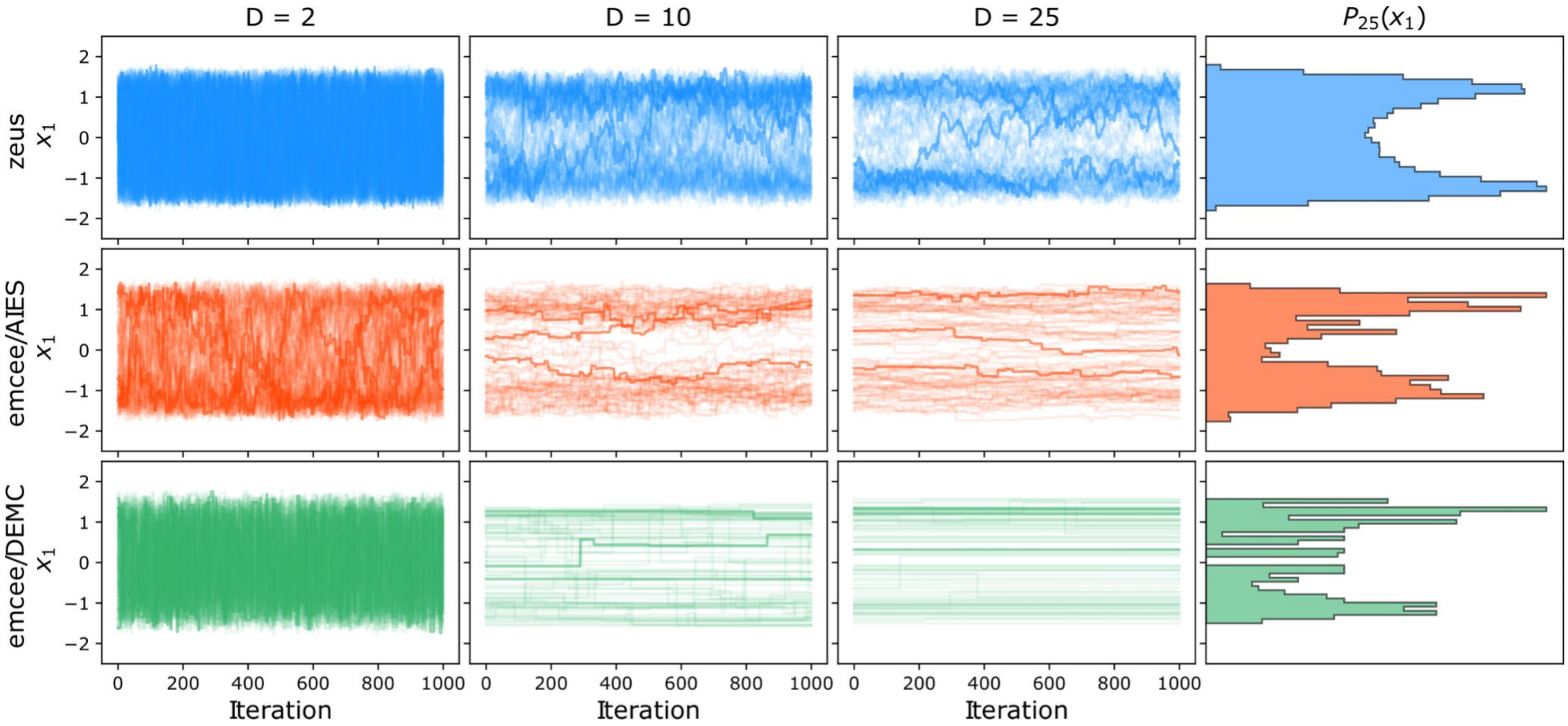
# Zeus and Ensemble Slice Sampling





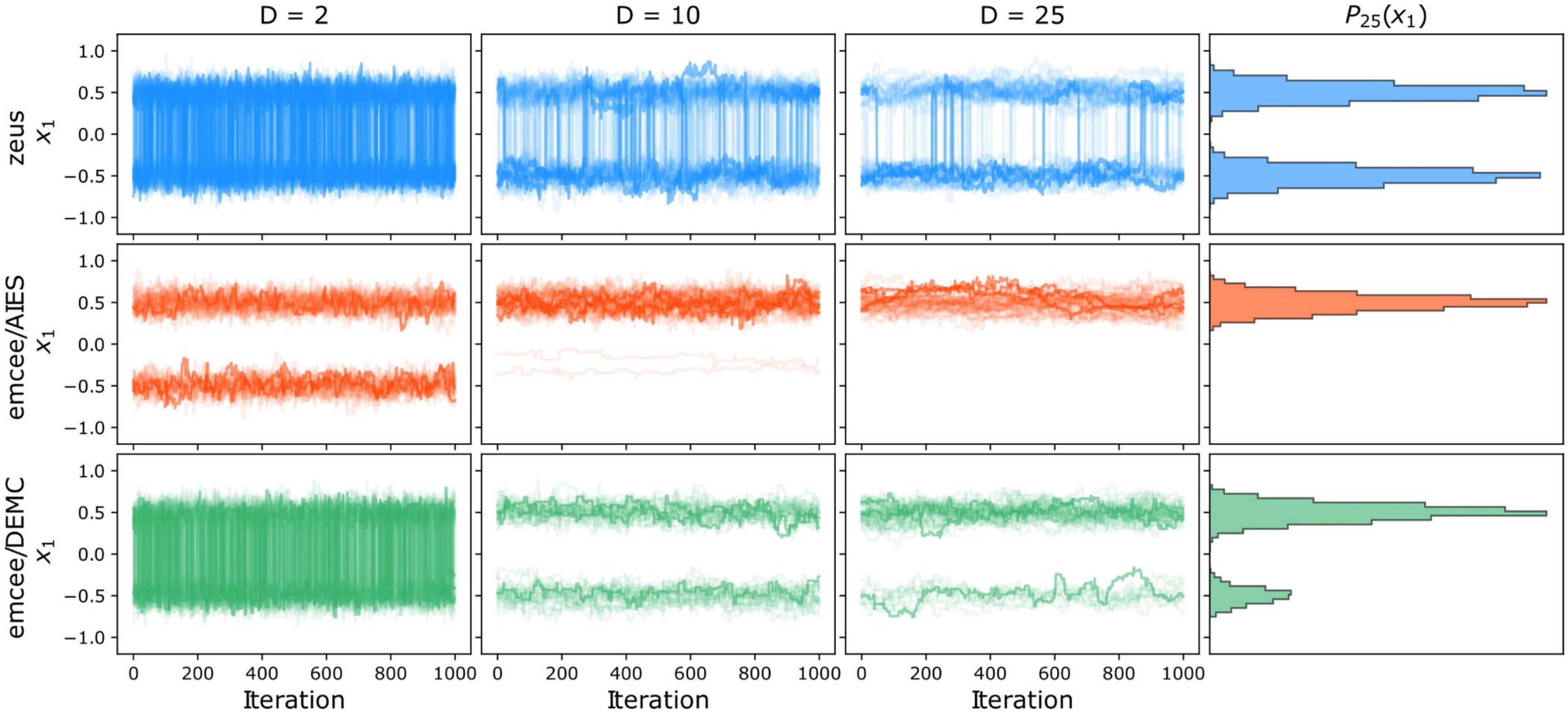
arXiv:2105.03468

# Ring distribution



arXiv:2105.03468

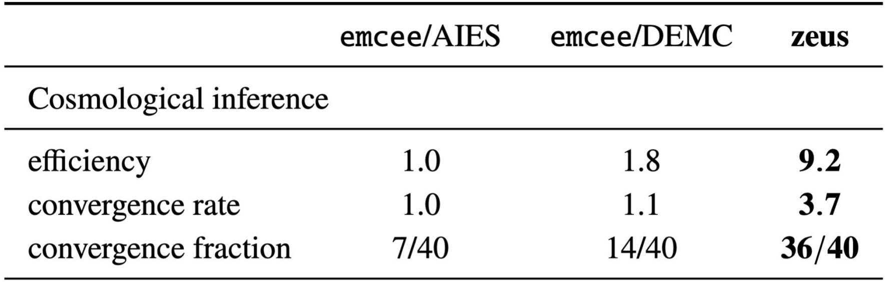
# Gaussian mixture



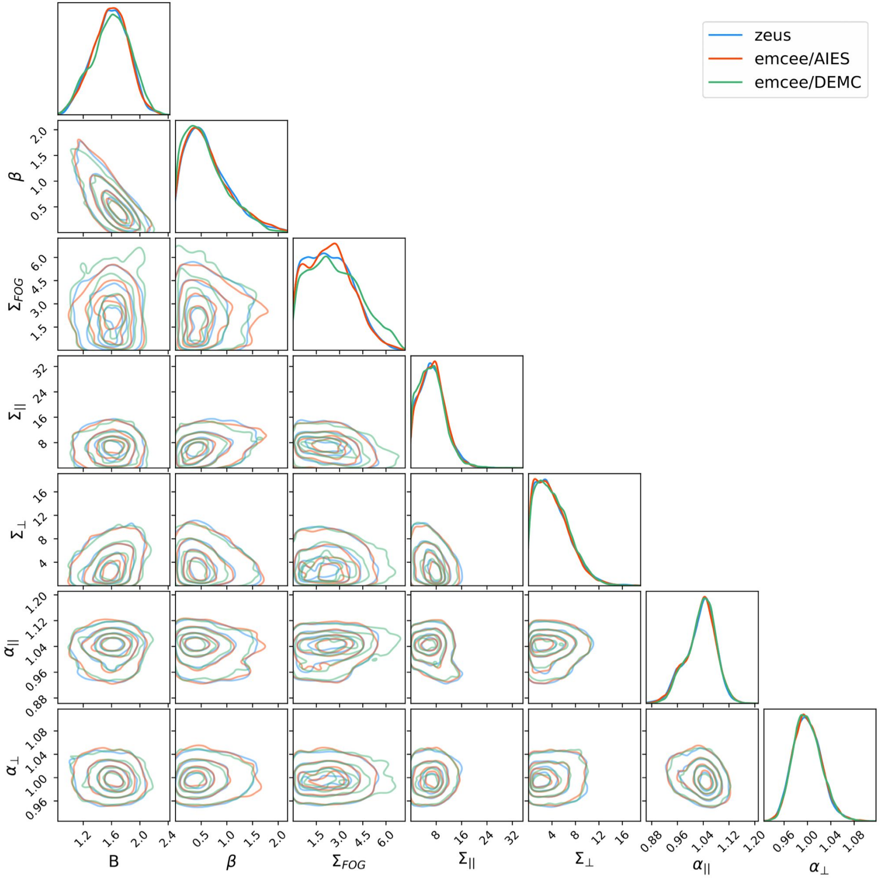
arXiv:2105.03468

# Baryon Acoustic Oscillations

- 22 free parameters
- BOSS DR12 NGC dataset
- P0, P2, P4 model

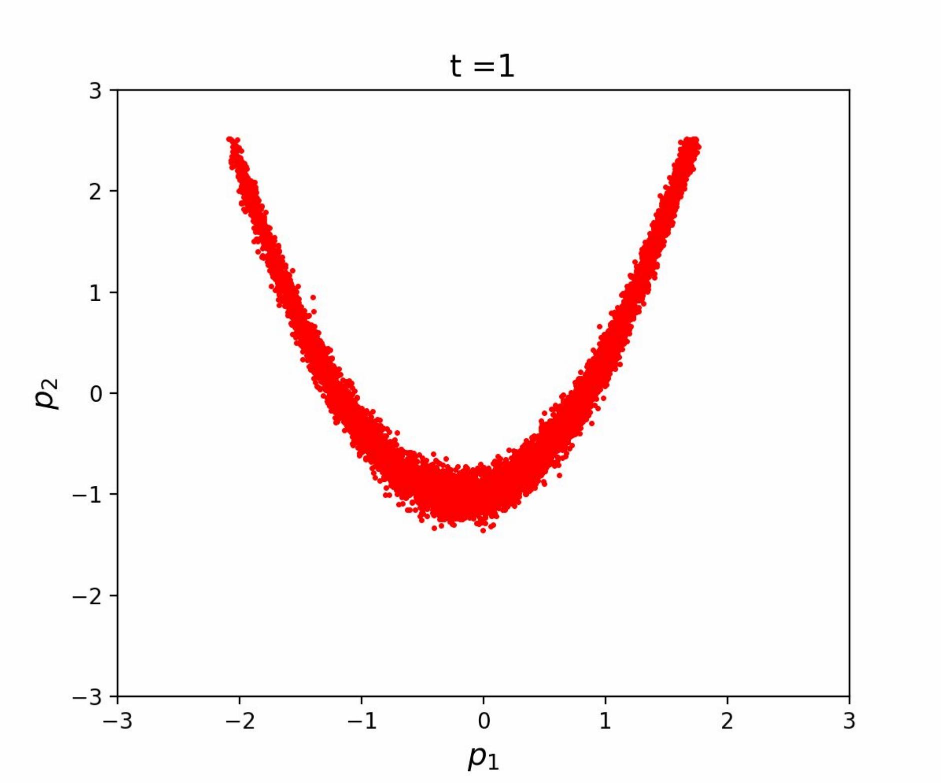


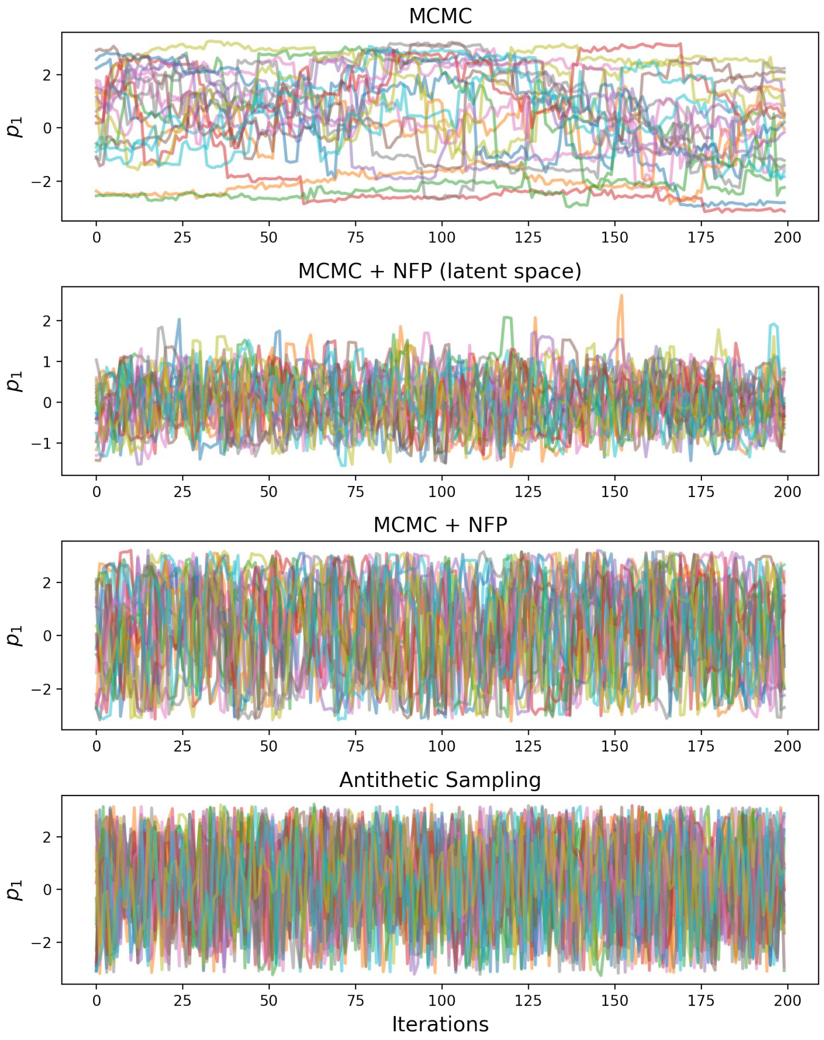
| | emcee/AIES | emcee/DEMC | **zeus** |
|---|---|---|---|
| Cosmological inference | | | |
| efficiency | 1.0 | 1.8 | **9.2** |
| convergence rate | 1.0 | 1.1 | **3.7** |
| convergence fraction | 7/40 | 14/40 | **36/40** |



arXiv:2105.03468

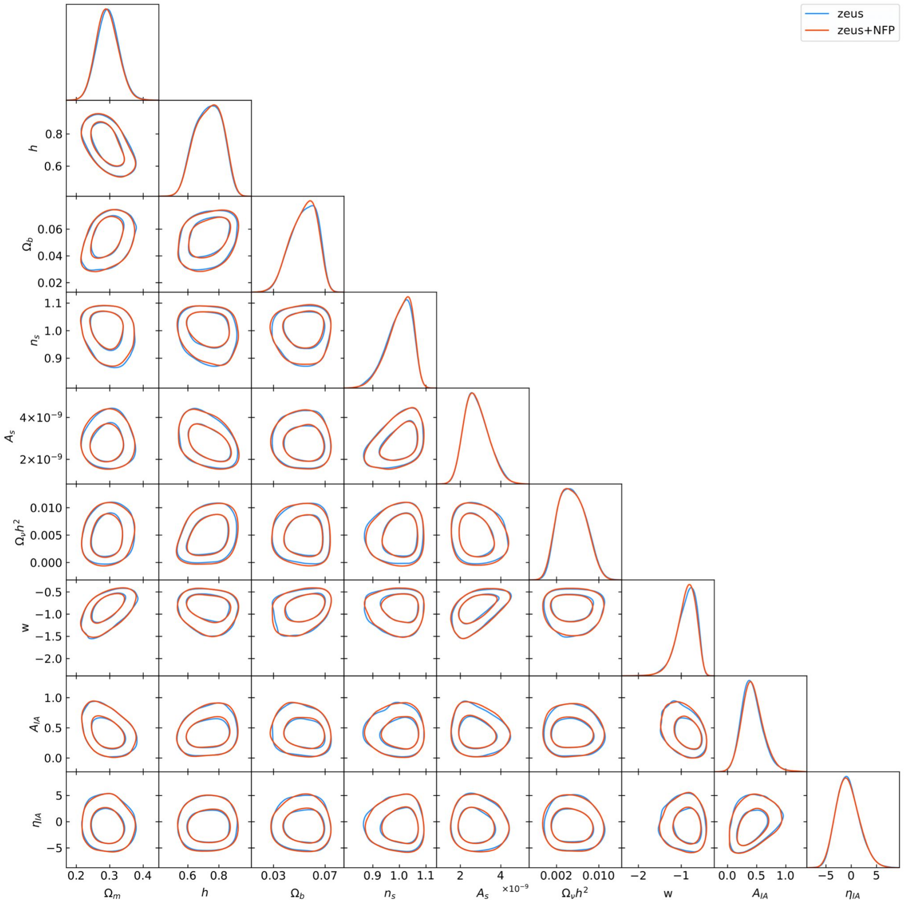
# 20D Rosenbrock distribution





# DES Y1 analysis

- Augmenting **zeus** with NFP results in increased sampling efficiency (effective samples per model evaluation) by a factor of 13.
- Further sampling further with the antithetic sampler results in increased efficiency by a factor of 200 yielding one effectively independent sample per 5-7 model evaluations.



# Conclusions

- NFP can be applied to any target distribution,
- NFP can reinforce any MCMC method,
- Using NFP to augment zeus using an iterative training scheme results in a very efficient sampling algorithm. In the case of the DES Y1 this resulted in a 13 times increase of the sampling efficiency.
- At the final stages of sampling, zeus can be replaced by the antithetic sampler to produce an even higher efficiency. In the case of the DES Y1 this resulted in a ~200 times increase of the sampling efficiency.

**zeus is publicly available at** https://zeus-mcmc.readthedocs.io