Markov Chain Monte Carlo (MCMC) Simulations for Early Universe Cosmology with *subMIT*

subMIT Workshop, MIT

January 23, 2026

Sarah R. Geller
NSF Mathematical and Physical Sciences Ascend Fellow
Yale University & SCIPP UCSC

Markov Chain Monte Carlo (MCMC) for Early Universe Cosmology

Why do MCMC's for early universe cosmology? Why on the cluster?

How do parallelized MCMC's work for these types of problems: some examples from past/ongoing work.

Utilizing subMIT with *Emcee* and built-in *blobs* feature to store observables to chains

Tips and wishes

A frequent problem in (early universe) cosmology:

We have **some model(s)** for an early universe process (eg. inflation, supermassive seed formation, baryogenesis etc) **and some observational constraints**.

A frequent problem in (early universe) cosmology:

We have **some model(s)** for an early universe process (eg. inflation, supermassive seed formation, baryogenesis etc) **and some observational constraints**.

The model(s) have **some free parameters (**p_i**)** and predict **some observables (**$\mathcal{O}_j$**)**

A frequent problem in (early universe) cosmology:

We have **some model(s)** for an early universe process (eg. inflation, supermassive seed formation, baryogenesis etc) **and some observational constraints**.

The model(s) have **some free parameters (**p_i**)** and predict **some observables (**$\mathcal{O}_j$**)**

The parameter space spanned by p_i's is large/complicated/curvy/>>1 dim

The question: For what (if any) values of p_i does this model predict observables $\mathcal{O}_j$ in compliance with the observation constraints?

The question: For what (if any) values of p_i does this model predict observables $\mathcal{O}_j$ in compliance with the observation constraints?

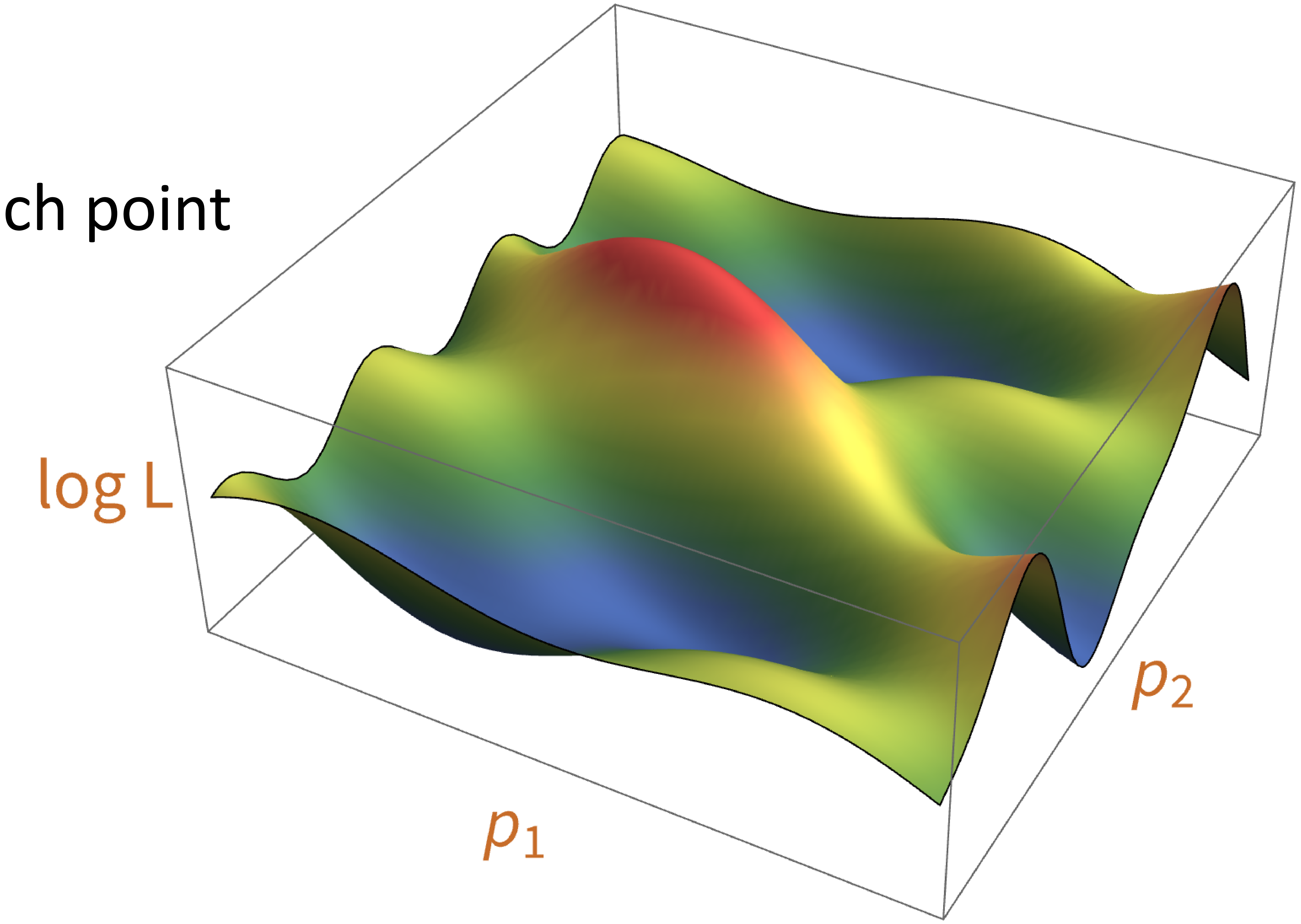
Markov Chain Monte Carlo (MCMC) for Early Universe Cosmology

The question: For what (if any) values of p_i does this model predict observables $\mathcal{O}_j$ in compliance with the observation constraints?

Ultimate goals:

1. Figure out the **best fit regions of parameter space** by assigning a likelihood to each point $(p_1 \ldots p_n)$ based on how well the predicted observables $(\mathcal{O}_1 \ldots \mathcal{O}_n)$ match constraints (real observables)

Parameters vs Likelihood



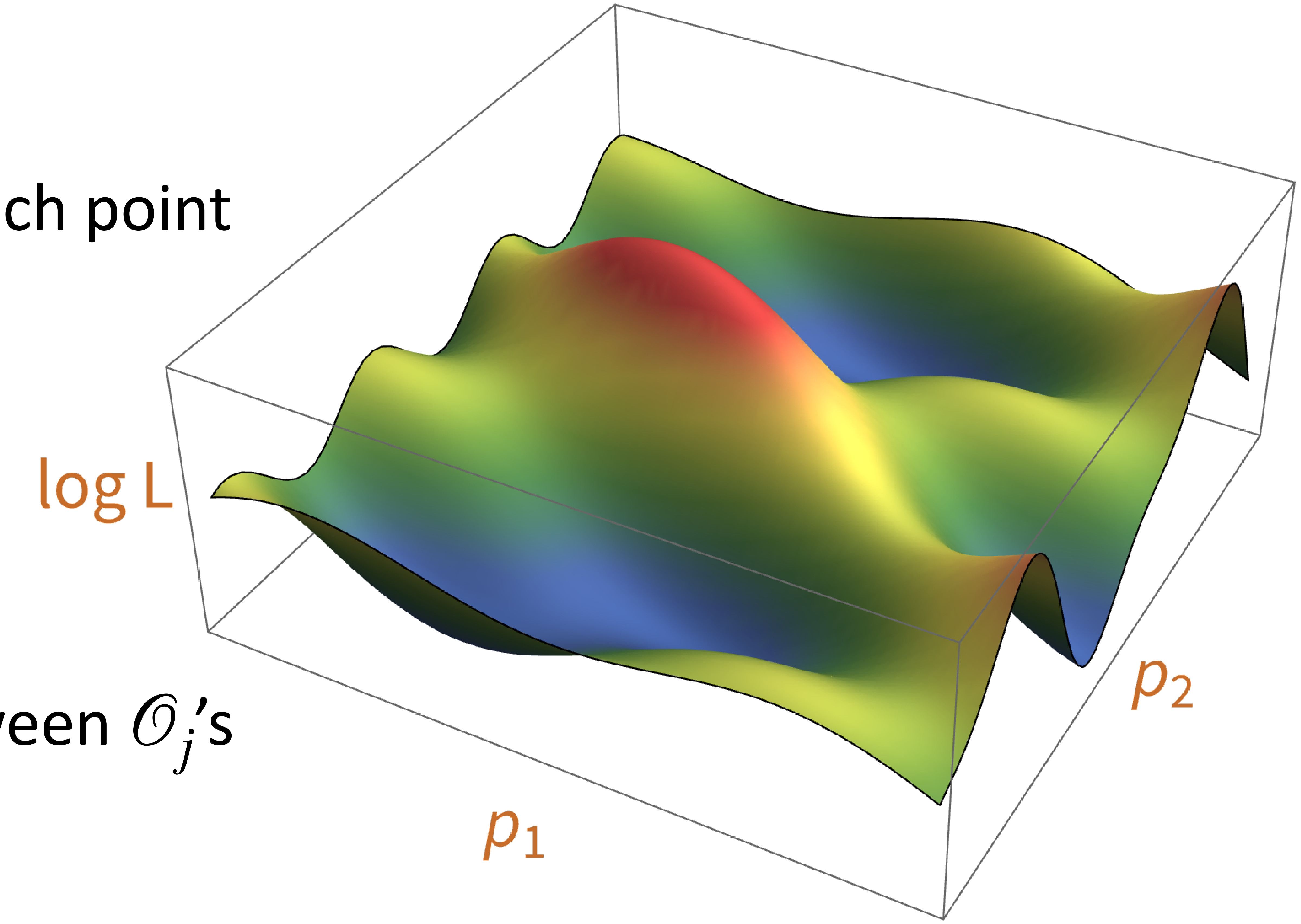
The question: For what (if any) values of p_i does this model predict observables $\mathcal{O}_j$ in compliance with the observation constraints?

Ultimate goals:

1. Figure out the **best fit regions of parameter space** by assigning a likelihood to each point $(p_1 \ldots p_n)$ based on how well the predicted observables $(\mathcal{O}_1 \ldots \mathcal{O}_n)$ match constraints (real observables)

2. Figure out correlations between p_i's and between $\mathcal{O}_j$'s

Parameters vs Likelihood



Simplest approach is usually prohibitively slow/expensive: simply search a fine-grained grid over the parameter space

MCMC uses a combination of Monte Carlo Sampler and Markov Chain

Monte Carlo

Random sampling of posterior distribution $\approx$ throwing darts

This is *highly* inefficient in a high-dim parameter space

Markov chain

Chain $\Longrightarrow$ sequential

Process with 1-step memory

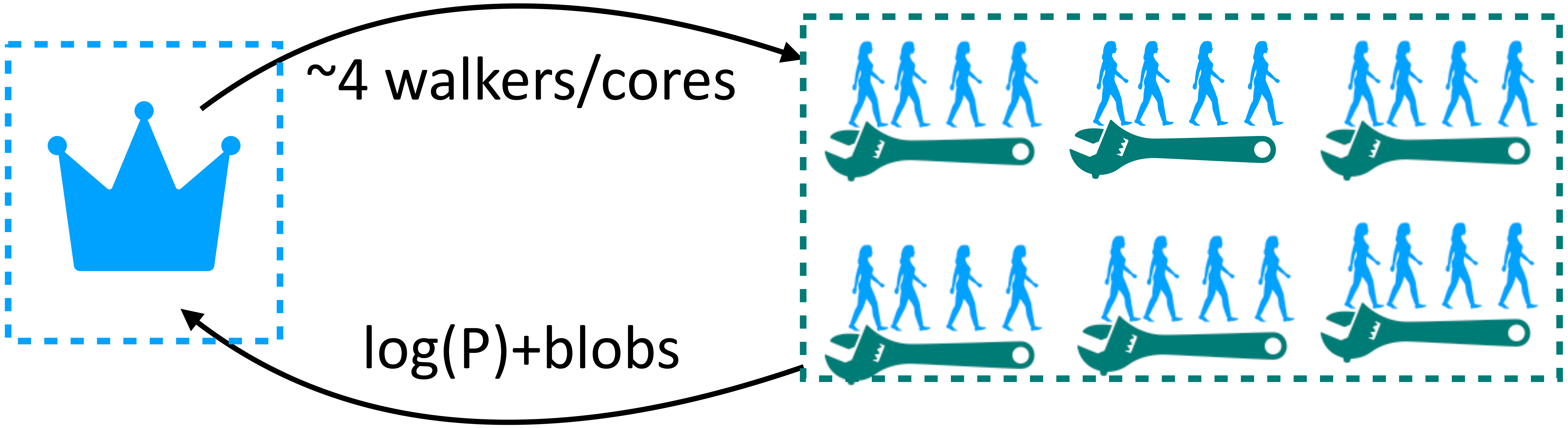
Propose step- accept if likelihood is better (and sometimes even if worse)

Ensemble MCMC Architecture: parallelization with many walkers

Emcee sampler uses an affine-invariant “stretch move”— naturally enables parallelization

Emcee uses a “worker/master” pattern: master level farms out walkers to workers (in subMIT we have 48 workers per node)

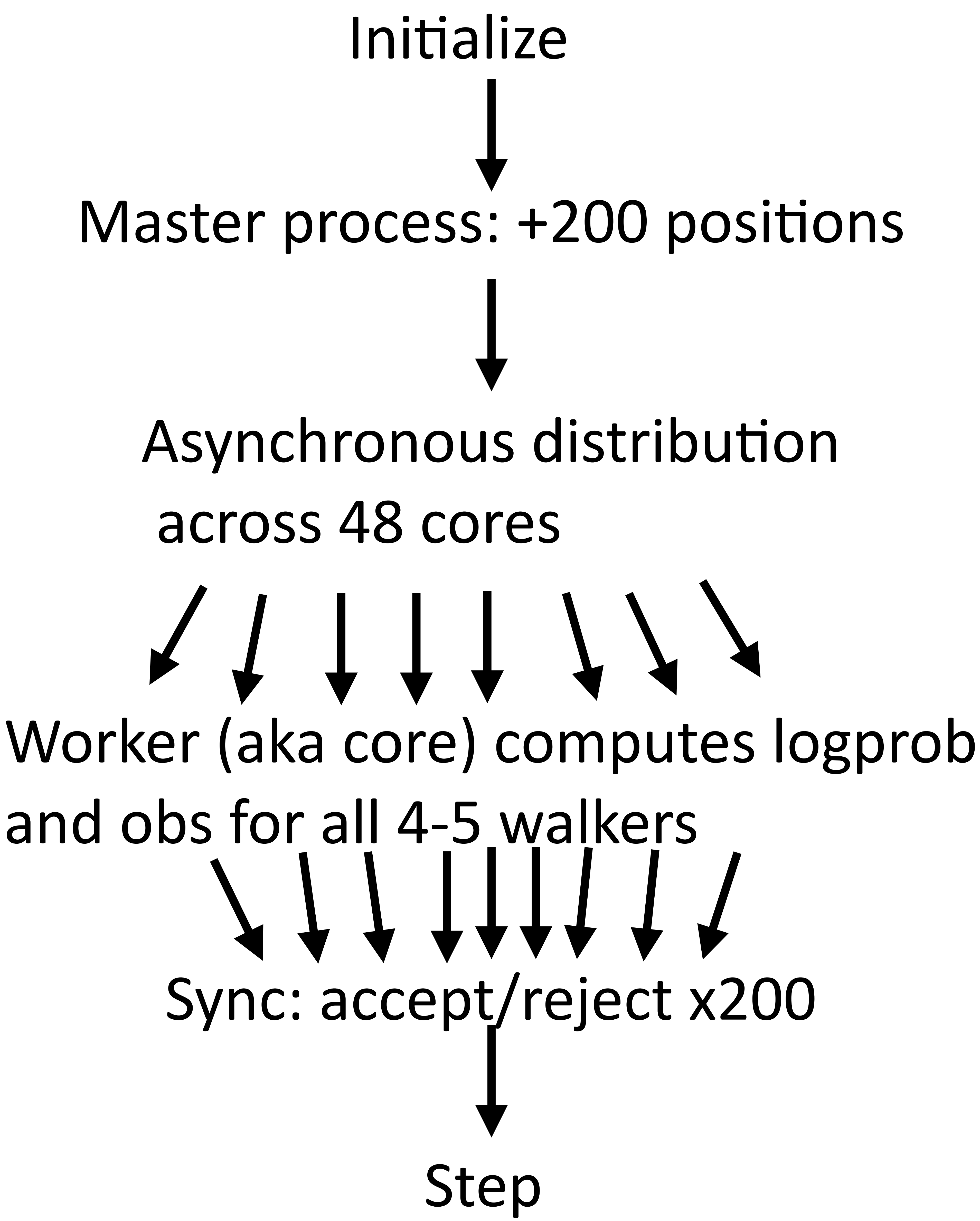
Affine-stretch allows workers to remain independent but still proposes new positions based on other walkers because they all communicate to master process.



Python multiprocessing.pool

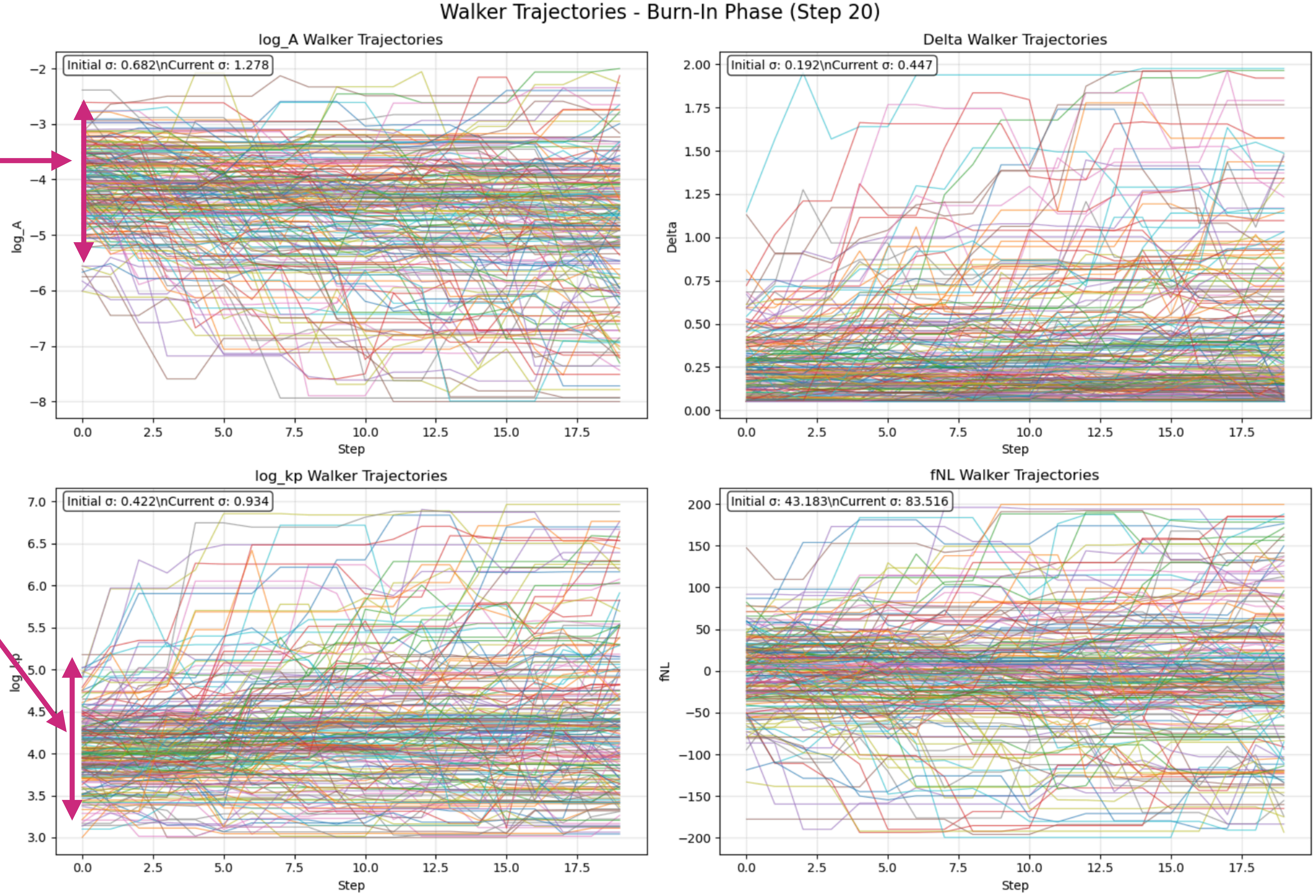
Ensemble MCMC Architecture: parallelization with many walkers

```python
from multiprocessing import Pool
import emcee

# Create worker pool (48 processes)
with Pool(processes=48) as pool:
    sampler = emcee.EnsembleSampler(
        nwalkers=200,          # 200 walkers
        ndim=4,                # 4 parameters
        log_prob_fn=log_posterior,  # Our physics function
        pool=pool              # Enable parallelization
    )

    # Run MCMC
    sampler.run_mcmc(initial_pos, nsteps=10000)
```



Example: Burn-in phase for 200 walkers in 4D parameter space

Initialize
200 walkers by drawing
from
Gaussian ball
around hardcoded
init values

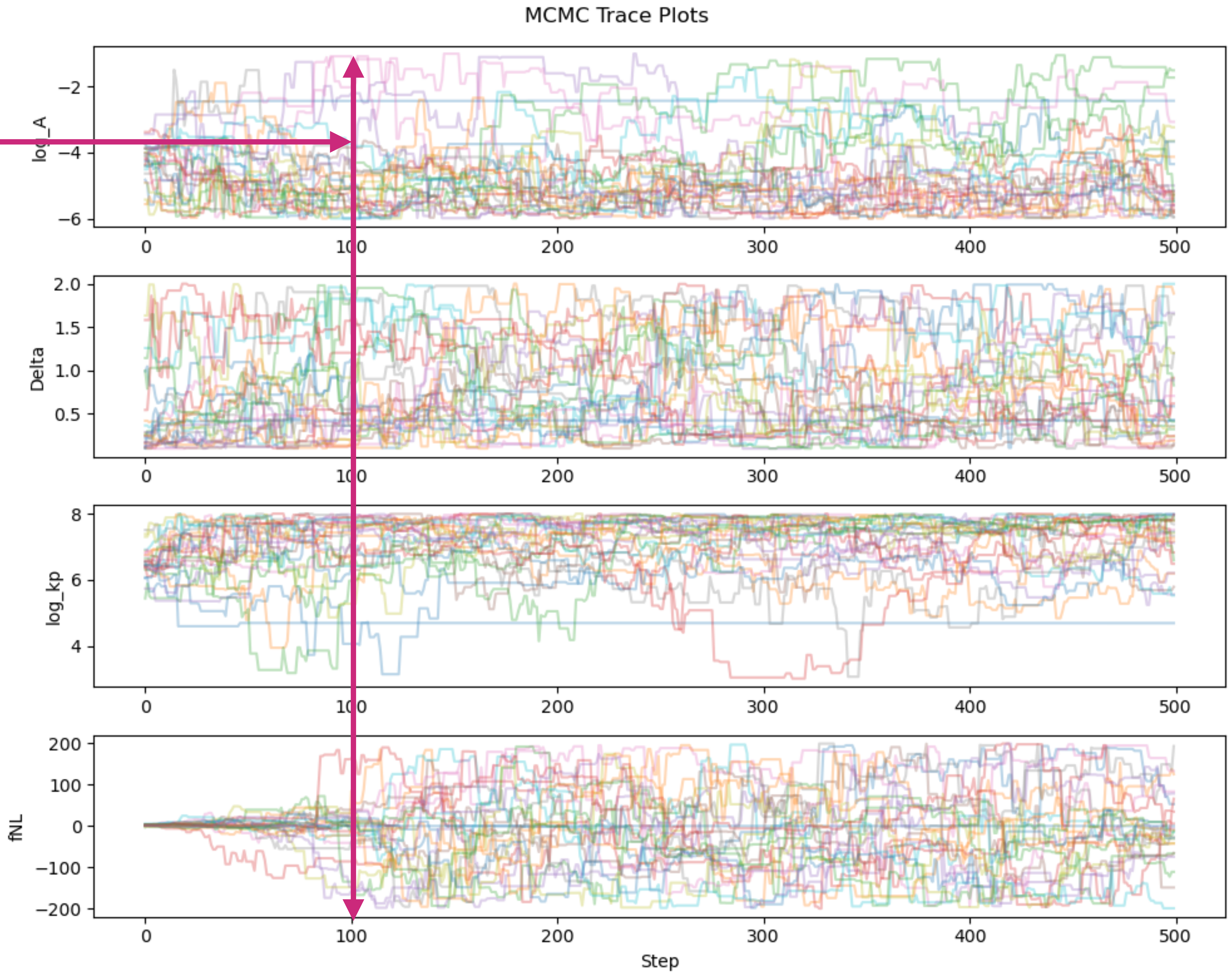


Markov chain induces correlations
Discard a burn in phase

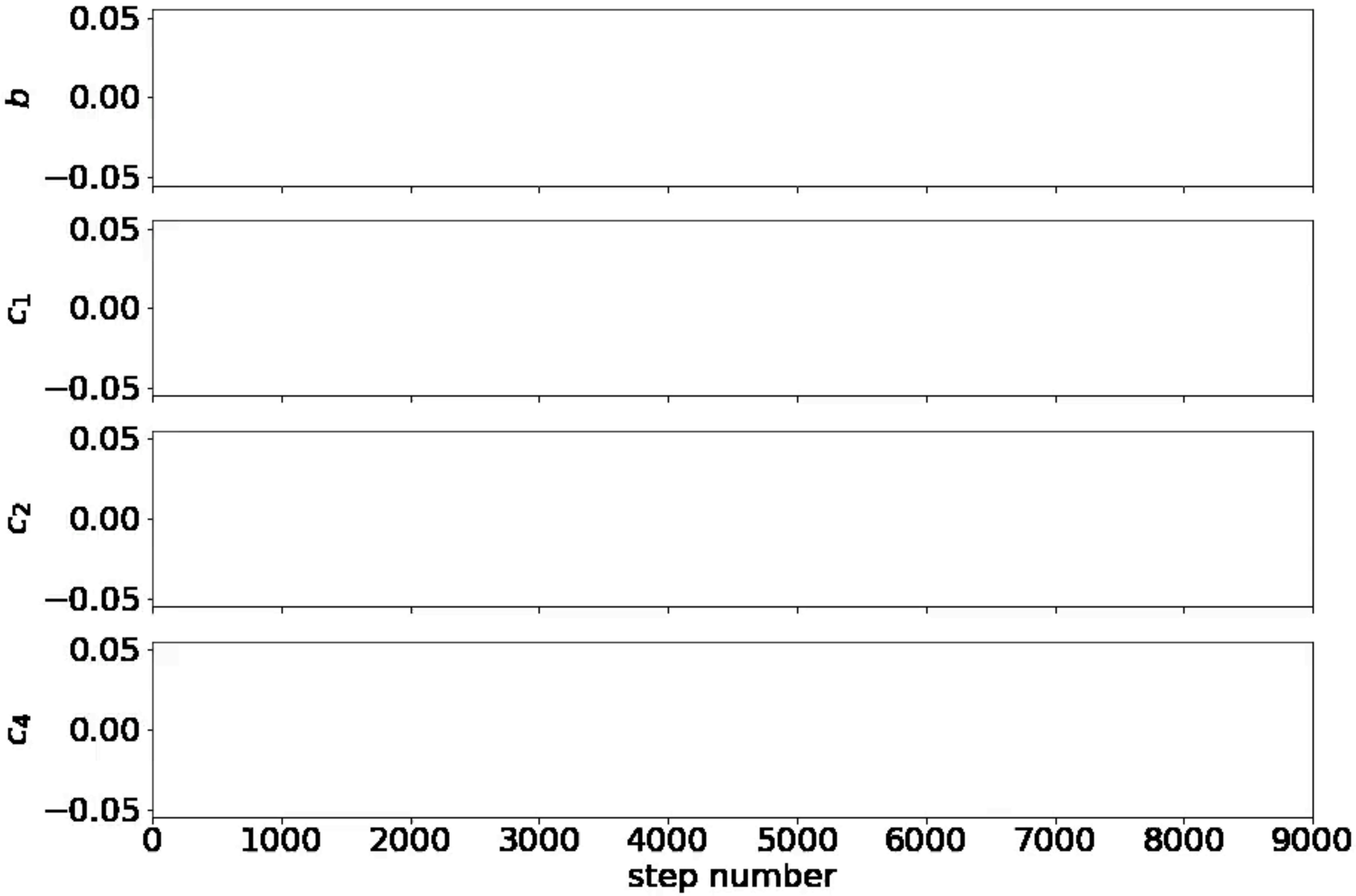
Example: Burn-in phase for 200 walkers in 4D parameter space

Initialize
200 walkers by drawing
 from
Gaussian ball
 around hardcoded
init values

Markov chain induces correlations
Discard a burn in phase



Markov Chain Monte Carlo (MCMC) for Early Universe Cosmology



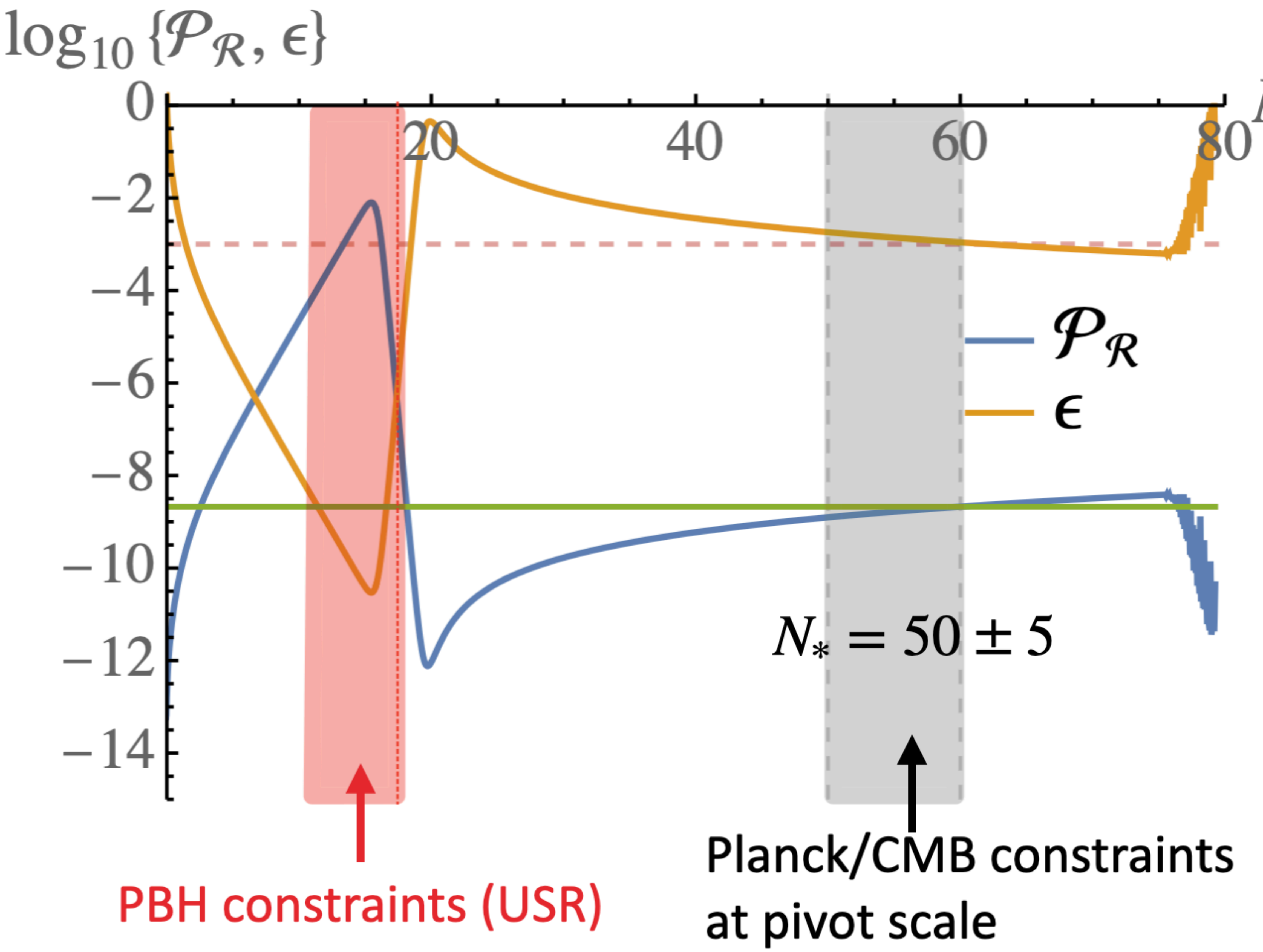
Case 1: A 2-field inflation model for PBH dark matter with 4D parameter space

SRG, Qin, McDonough, Kaiser ’22
Qin, SRG, Balaji, McDonough, Kaiser ’23

MCMC, 200 walkers each taking 10,000 steps through a 4-dim parameter space (b, c_1, c_2, c_4)

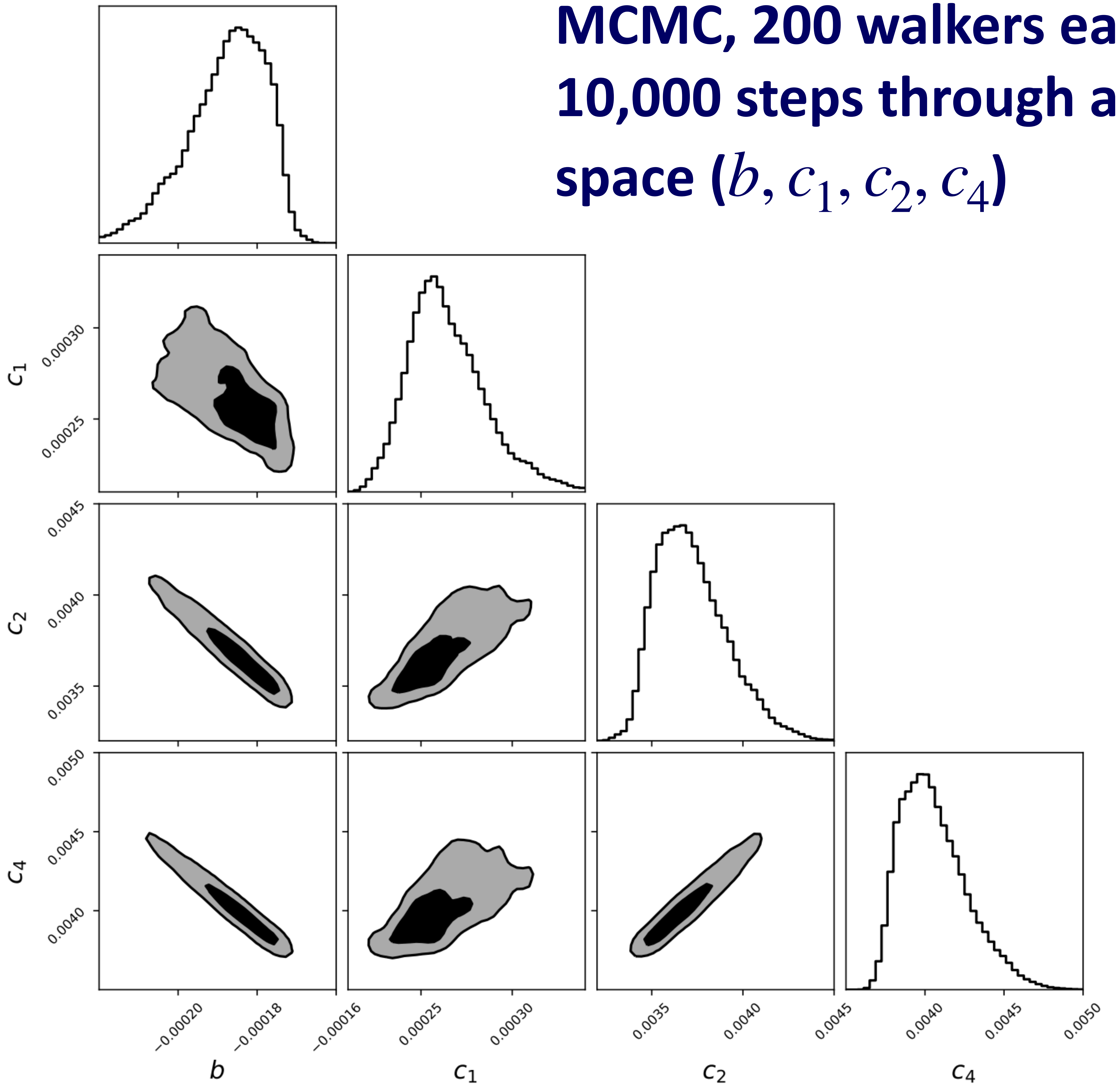
Planck 2018: gives constraints at CMB scales
n_s, A_s, α_*, r_*

Requiring that PBHs form give an additional (self-imposed) constraints at smaller scales



* part of the computing for this project was done on the Chicago cluster

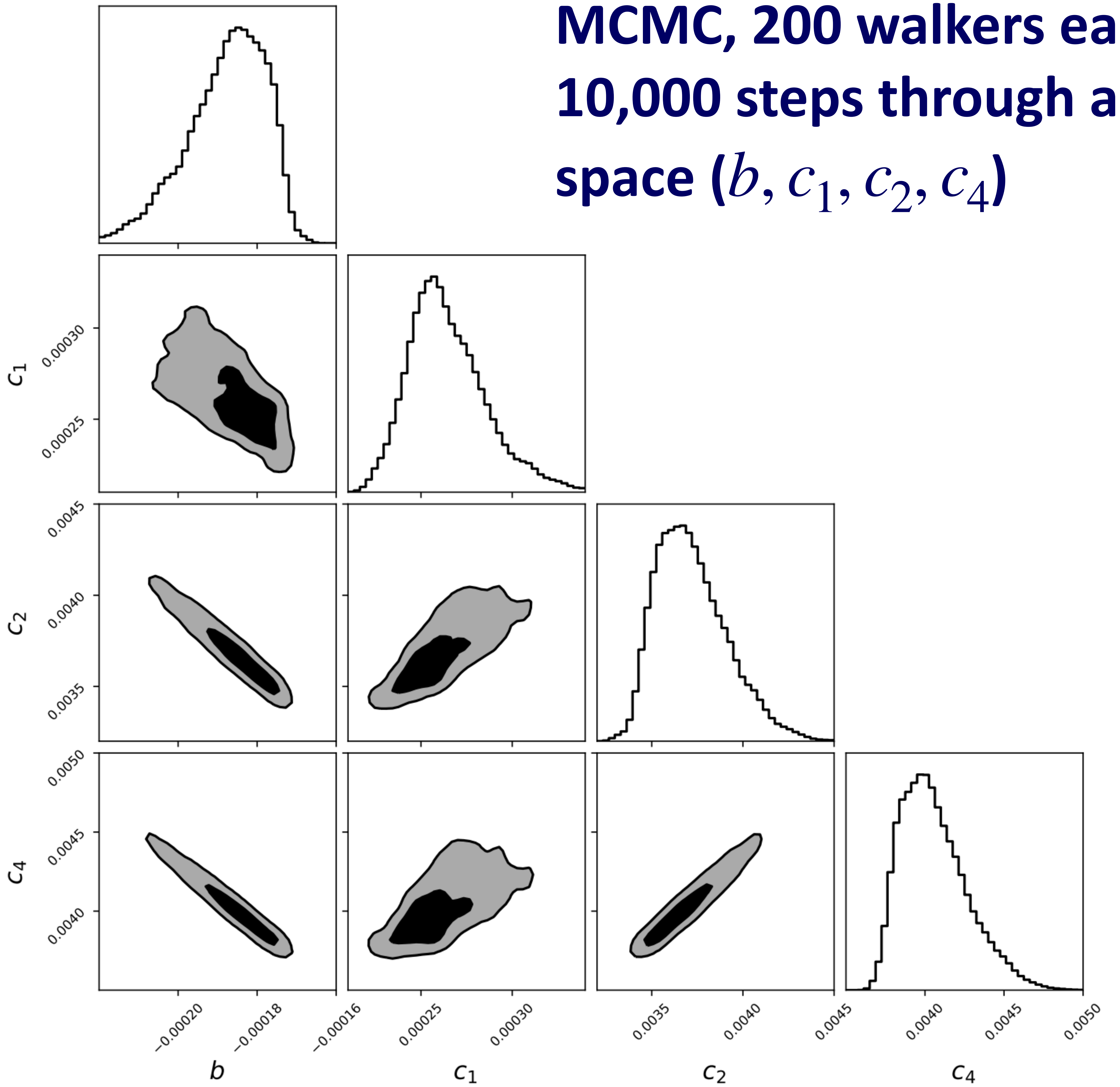
A 2-field inflation model for PBH dark matter with 4D parameter space



MCMC, 200 walkers each taking 10,000 steps through a 4-dim parameter space (b, c_1, c_2, c_4)

Percent-level fine-tuning

A 2-field inflation model for PBH dark matter with 4D parameter space



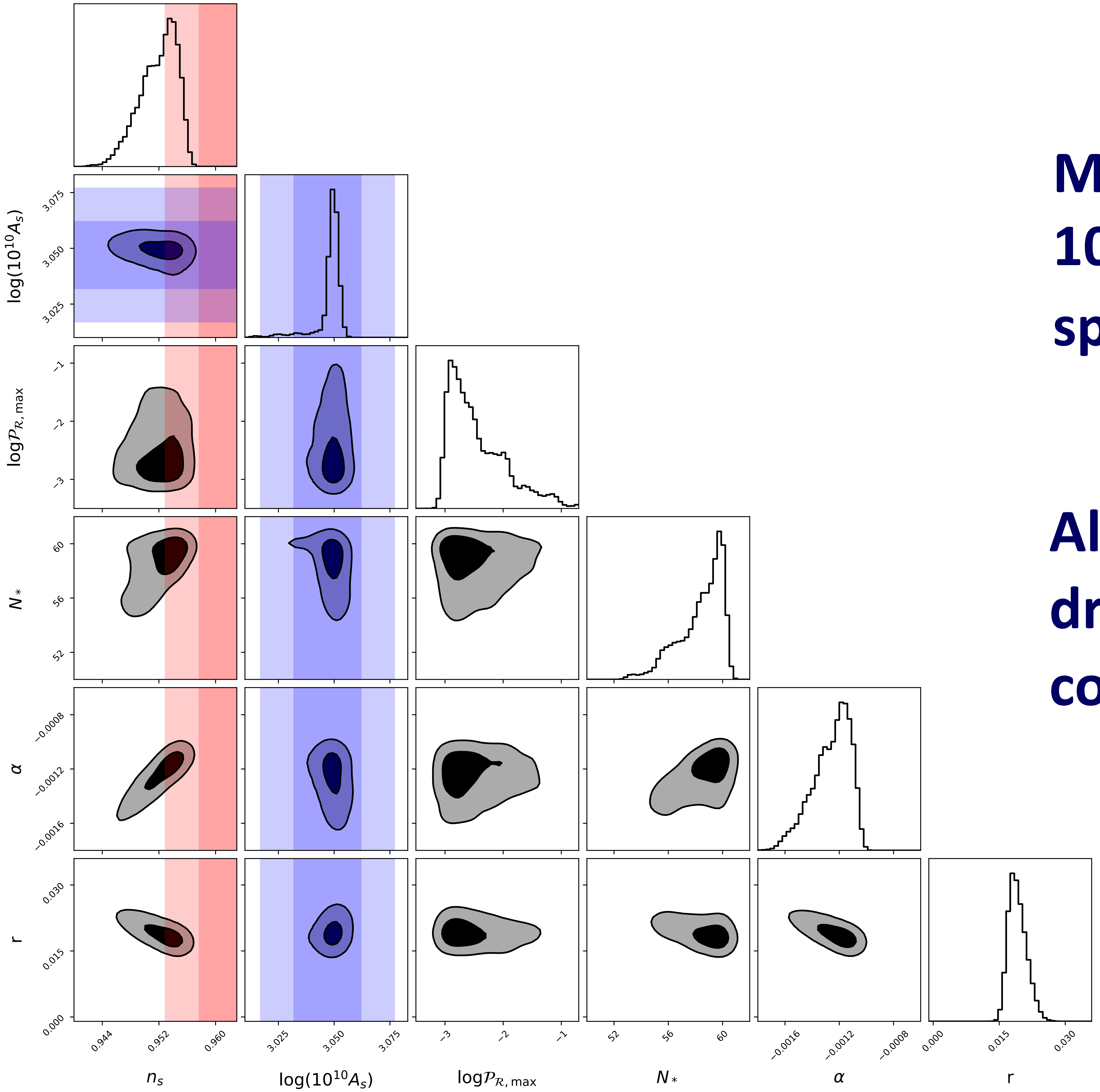
MCMC, 200 walkers each taking 10,000 steps through a 4-dim parameter space (b, c_1, c_2, c_4)

Constraints from requiring PBH DM and satisfying *Planck* 2018 data

Parameter	Constraint
$\boldsymbol{b}$	$-1.87\,(-1.73)^{+0.09}_{-0.11} \times 10^{-4}$
$\boldsymbol{c_1}$	$2.61\,(2.34)^{+0.24}_{-0.17} \times 10^{-4}$
$\boldsymbol{c_2}$	$3.69\,(3.42)^{+0.22}_{-0.16} \times 10^{-3}$
$\boldsymbol{c_4}$	$4.03\,(3.75)^{+0.24}_{-0.17} \times 10^{-3}$
$n_s(k_*)$	$0.952\,(0.956)^{+0.002}_{-0.003}$
$\log(10^{10} A_s)$	$3.049\,(3.048)^{+0.001}_{-0.001}$
N_*	$58.8\,(60.0)^{+1.2}_{-2.2}$
$\alpha(k_*)$	$-0.0012\,(-0.0010)^{+0.0001}_{-0.0002}$
$r(k_*)$	$0.019\,(0.016)^{+0.002}_{-0.001}$
b/c_2	$-5.04(-5.05)^{+0.03}_{-0.05} \times 10^{-2}$
c_1/c_2	$7.07(6.84)^{+0.32}_{-0.26} \times 10^{-2}$
c_4/c_2	$1.091(1.096)^{+0.009}_{-0.008}$

Percent-level fine-tuning

Markov Chain Monte Carlo (MCMC) for Early Universe Cosmology



MCMC, 200 walkers each taking 10,000 steps through a 4-dim parameter space (b, c_1, c_2, c_4)

Allows us to extract which observables are driving the physics, how observables are correlated

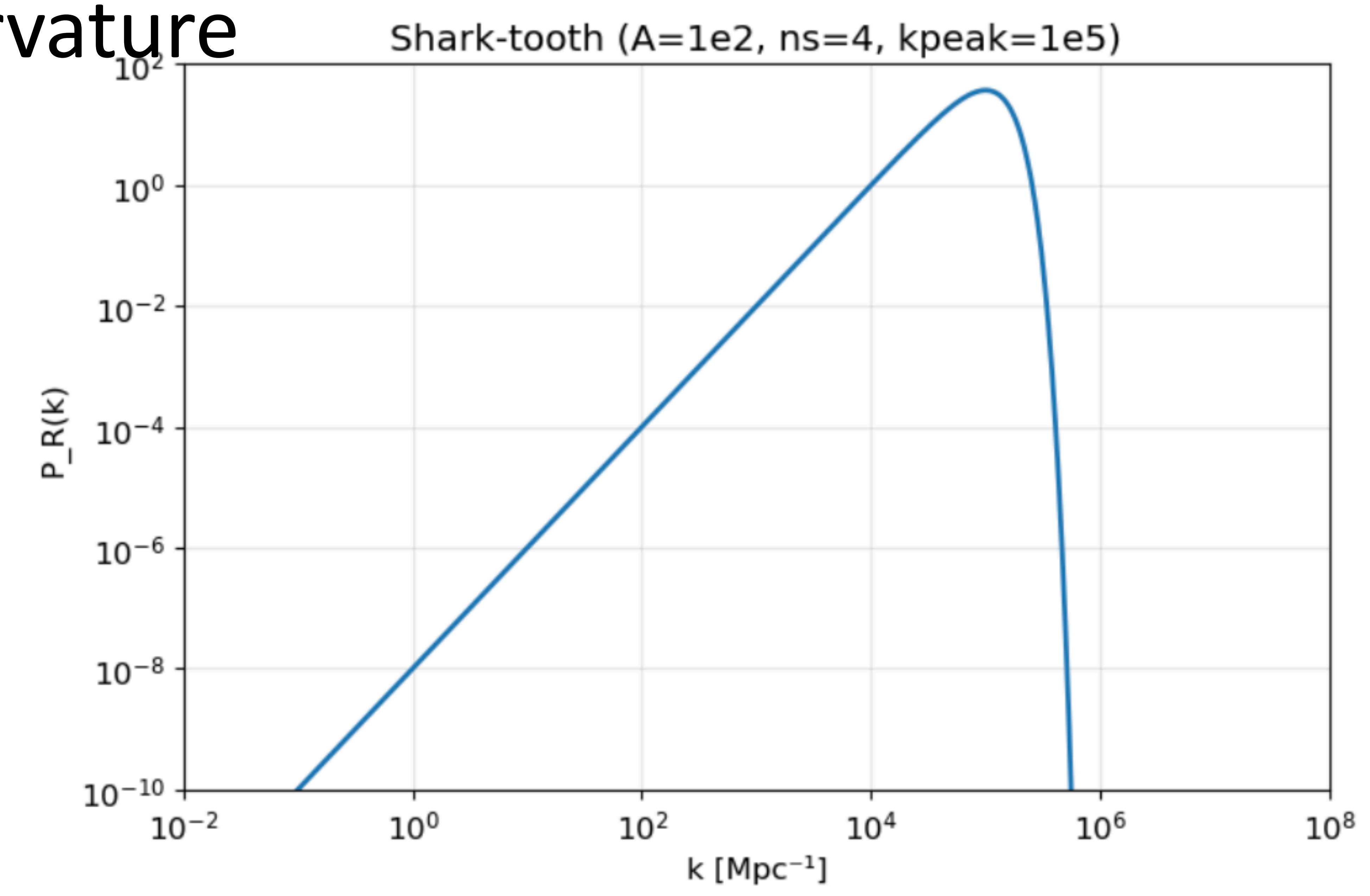
Case 2: Supermassive seeds from PBHs: JWST and μ-Distortion constraints

Balaji, Cyr, SRG, Kaiser, Lorenzoni, McDonough, Qin, 2026

$$\mathscr{P}_{\mathscr{R}}(k) = \mathscr{A}\left(\frac{k}{k_*}\right)^{n_*}\exp\left[-\left(\frac{k}{k_*}\right)^2\right]$$

Model: supermassive seeds form from collapse of over-densities post-inflation

Instantiate the model by passing 3 parameters to the curvature power-spectrum $\mathscr{P}_R(k)$: $\mathscr{A}, k_*, n_*$

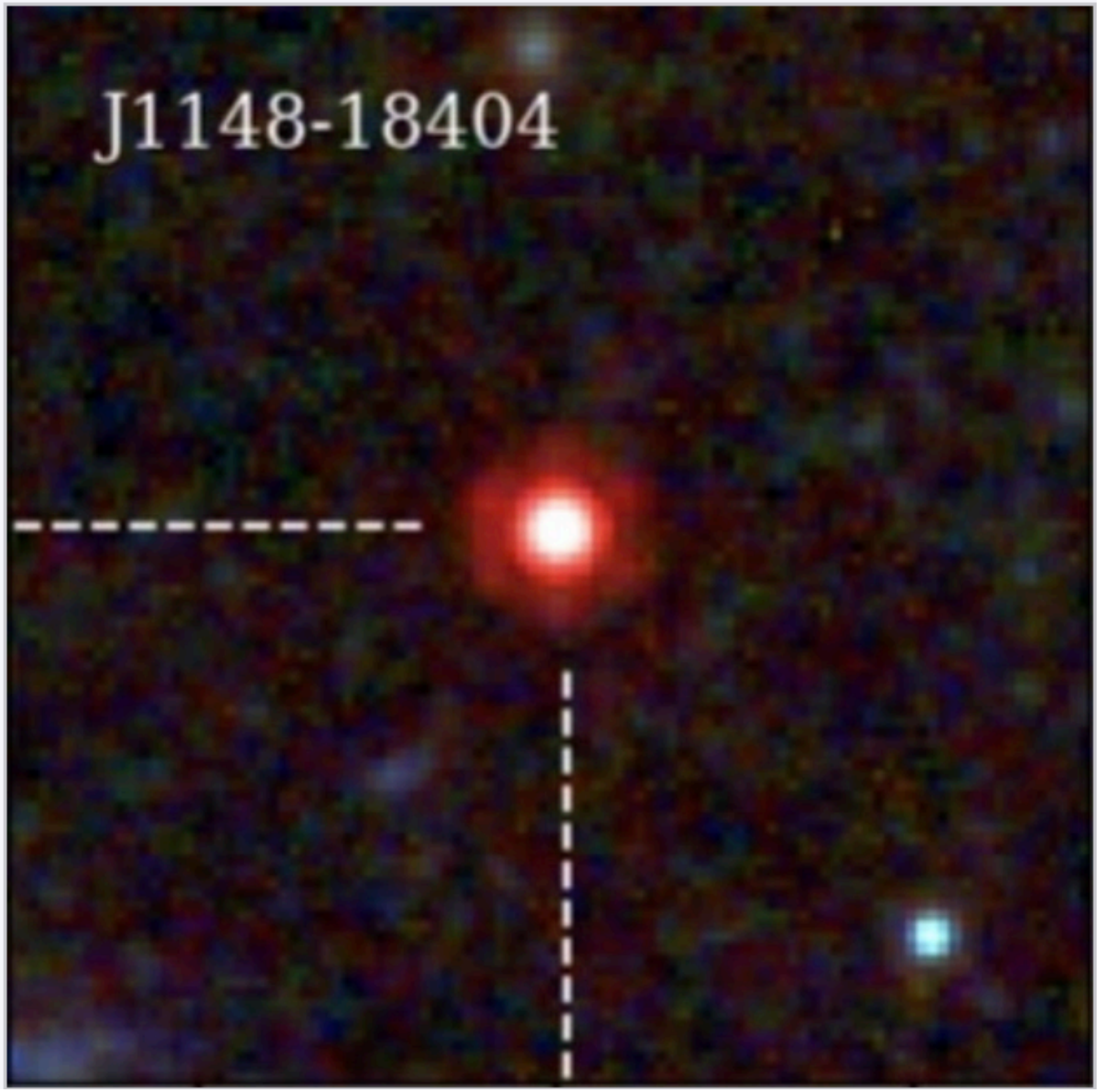


Case 2: Supermassive seeds from PBHs: JWST and μ-Distortion constraints

Balaji, Cyr, SRG, Kaiser, Lorenzoni, McDonough, Qin, 2026

Observable data comes from LRD
("Little Red Dot" - suspected high-redshift quasars)
number densities measured recently by
JWST combined with total μ-distortion
constraints from the CMB.

We allow arbitrary amount of local-type
non-Gaussianity



Goal: map viable region of parameter space that reconciles JWST high-z SMBH population with inflationary theory and show we can constrain with spectral distortions

Blobs: Saving Observables in Case 2

In each MCMC evaluation *Emcee* saves the likelihood and parameters to the chain

But, to get the likelihood it must compute observables (expensive).

Usually if MCMC is parallelized these are lost because they can't be saved (individual workers have independent memory spaces so if observables can't be passed back to master process they can't get written into the chain.

Blobs: Saving Observables in Case 2

In each MCMC evaluation *Emcee* saves the likelihood and parameters to the chain

But, to get the likelihood it must compute observables (expensive).

Usually if MCMC is parallelized these are lost because they can't be saved (individual workers have independent memory spaces so if observables can't be passed back to master process they can't get written into the chain.

With blobs: create another return object and passes it with log(prob) without adding overhead

Eliminates the need to redo all those calculations in post-processing!

Implementation on subMIT

Directory structure and implementation on *subMIT*

Root:
- pipeline files (contain all the physics)
- Mcmc files (wrap emcee and handle checkpointing)
- */cluster*
  - *SLURM* submission scripts
  - /results
    - Output hd5 files with chains

-/logs
- .out and .err log files containing checkpoints and error printing

Request: 1 exclusive node (using nproc to detect cores
Per node)

Personal subMIT project guide

1 Connection & Authentication

Task	Command
SSH to cluster	`ssh sgeller@submit-1.mit.edu`
Exit cluster	`exit` or `Ctrl+D`

2 File Transfer

2.1 Transfer from Local Machine to Cluster

Task	Command
Copy single file TO cluster	`scp /path/to/local/file.py sgeller@submit-1.mit.edu:~/spectral_distortions/`
Copy multiple files TO cluster	`scp file1.py file2.py sgeller@submit-1.mit.edu:~/spectral_dis`
Copy entire directory TO cluster	`scp -r /path/to/local/dir sgeller@submit-1.mit.edu:~/`
Copy file FROM cluster	`scp sgeller@submit-1.mit.edu:~/spectral_distortions/file.py /local/path/`
Copy results FROM cluster	`scp sgeller@submit-1.mit.edu:~/spectral_distortions/*.h5 ./results/`

2.2 Specific Files for This Project

```
# Transfer all updated Python files to cluster
cd "/Users/sarahgeller/MIT␣Dropbox/Sarah␣Geller/Ongoing␣Research␣Projects/
    spectral_distortions"

scp lrd_pipeline.py lrd_mcmc.py merged_pipeline.py mcmc_analysis.py \
    cluster_diagnostic.py cluster_diagnostic_standard.py \
    sgeller@submit-1.mit.edu:~/spectral_distortions/
```

3 Environment Setup on Cluster

Task	Command
Navigate to project directory	`cd ~/spectral_distortions`
Check Python version	`python3 --version`
List available modules	`module avail`
Load Python module (if needed)	`module load python/3.9`
Virtual Environment (venv)	
Activate venv	`source ~/venv/bin/activate`
Create new venv	`python3 -m venv ~/venv`
Deactivate venv	`deactivate`

2