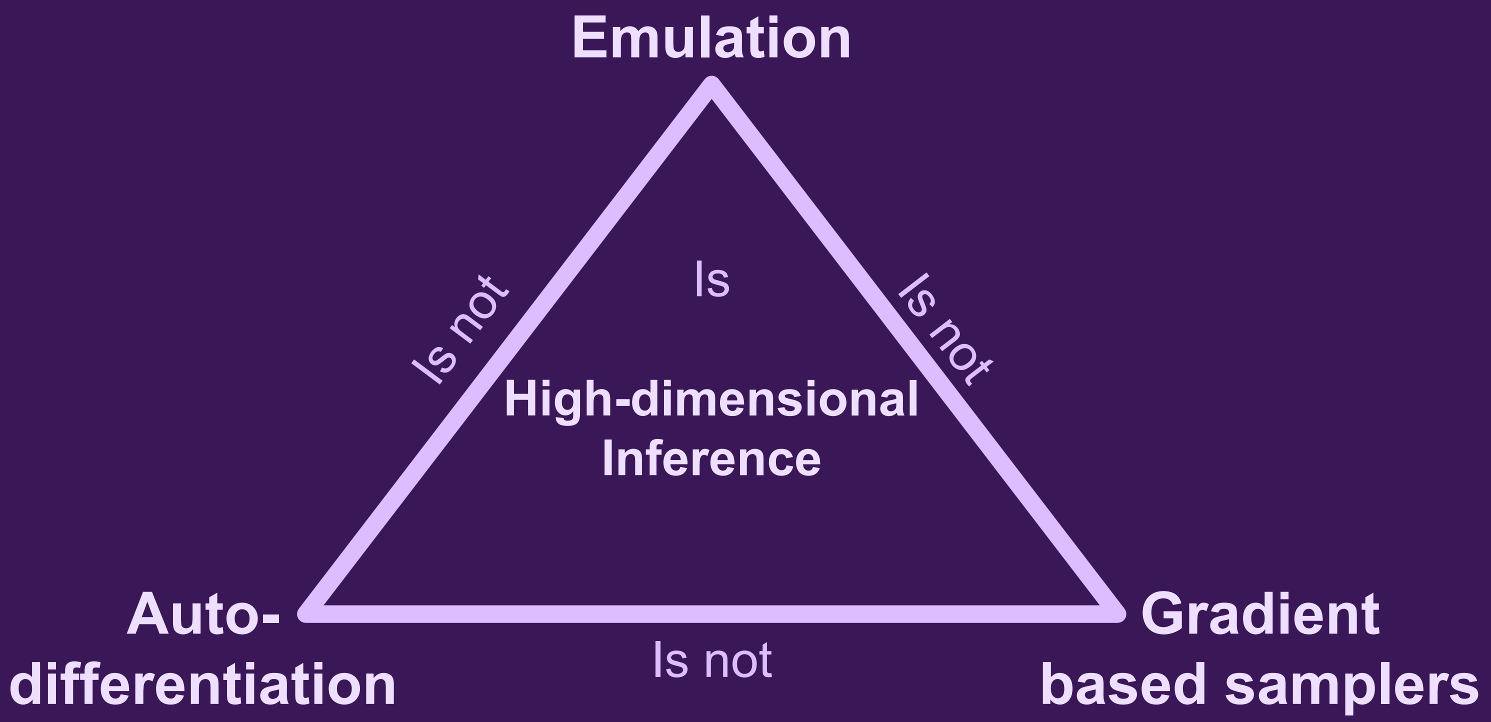
Bayesian Inference for Stage-IV and Beyond

Jaime Ruiz Zapatero (RSE @ UCL)

12th Feb, CosmoForward (Tenerife)

Conclusions:

Fast bayesian inference in the next 5 years will depend on the implementation of 3 technologies



(Piras+24)

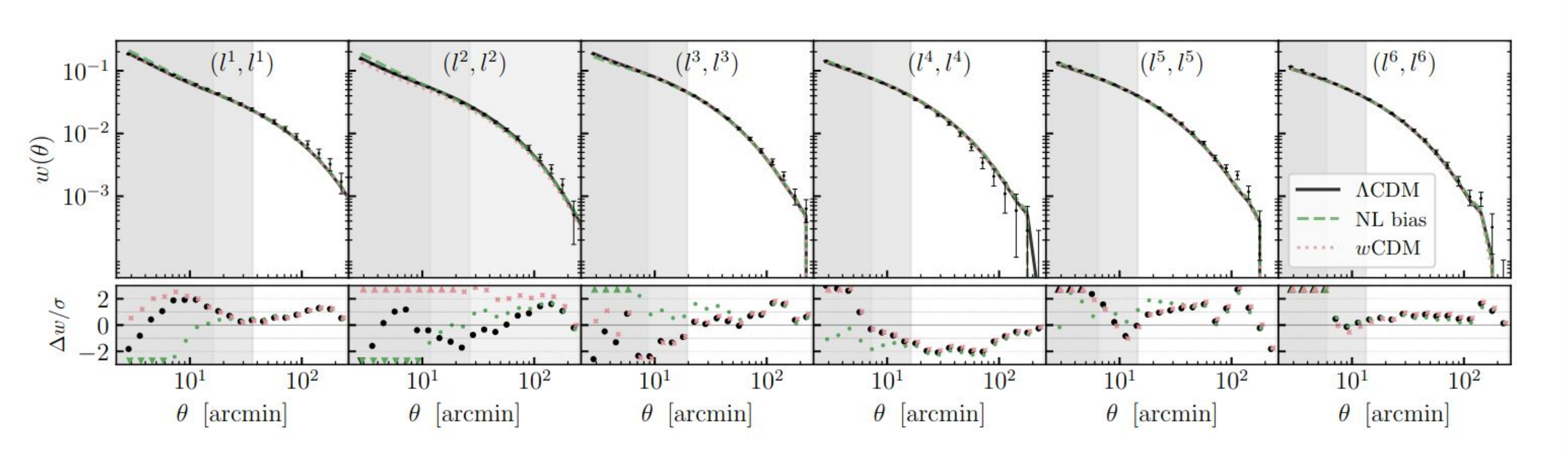





Sections

1. Introduction to Sampling
2. Gradient based Sampling
3. Auto-Differentiation
4. Emulation
5. Analytical Marginalization
6. Practical Advice

Introduction: Many Parameters, Many problems



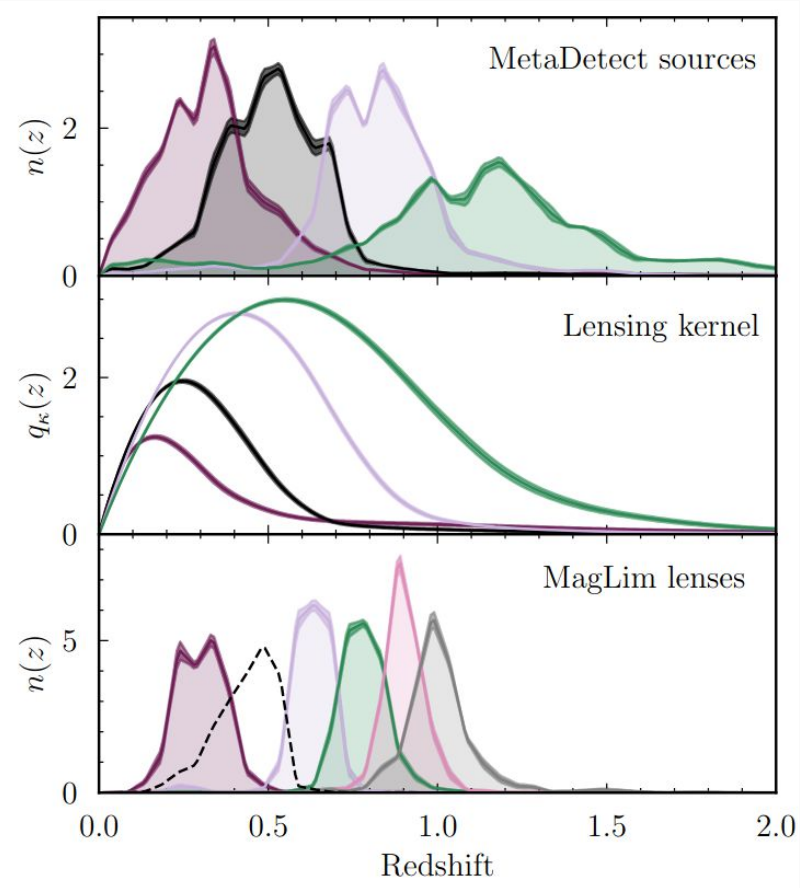
Abbott+26

Introduction: Many Parameters, Many problems

Number of parameters in the systematics modelling increases with the number of tomographic bins.

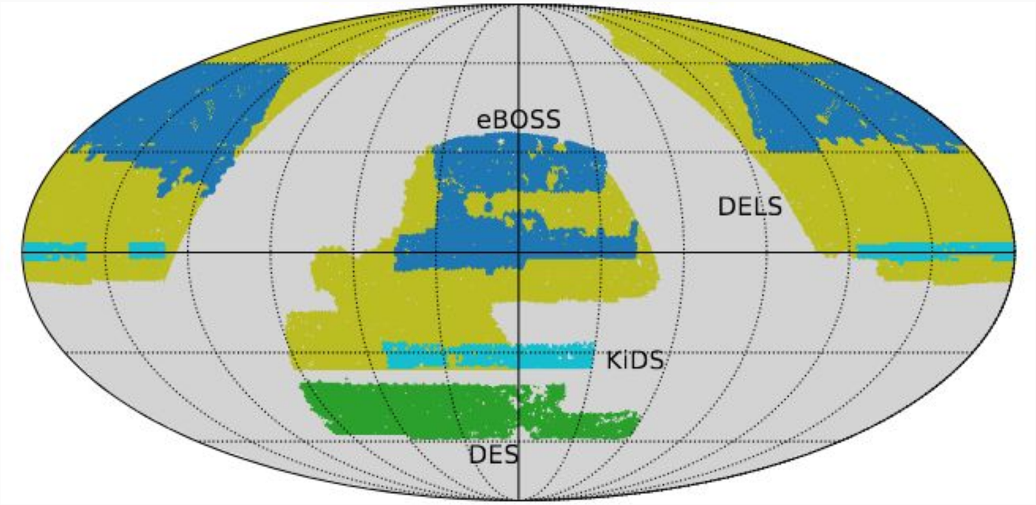
- 2 Intrinsic alignment
- 5 n(z) uncertainty
- 1 multiplicative bias
- 2 galaxy bias

✖ 10 bins

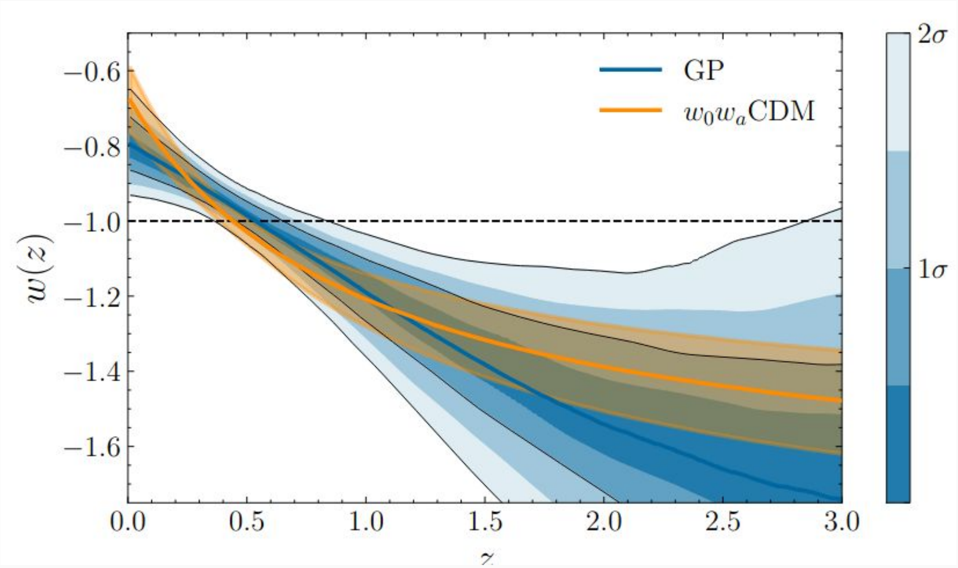


Abbott+26

Introduction: Many Parameters, Many problems



garcia-garcia+21



Lodha+25

Even more parameters if you want to do:

- Cross correlations between different bands with different galaxy samples
- Data-driven resconstructions

Introduction: Bayesian Inference (MCMC)

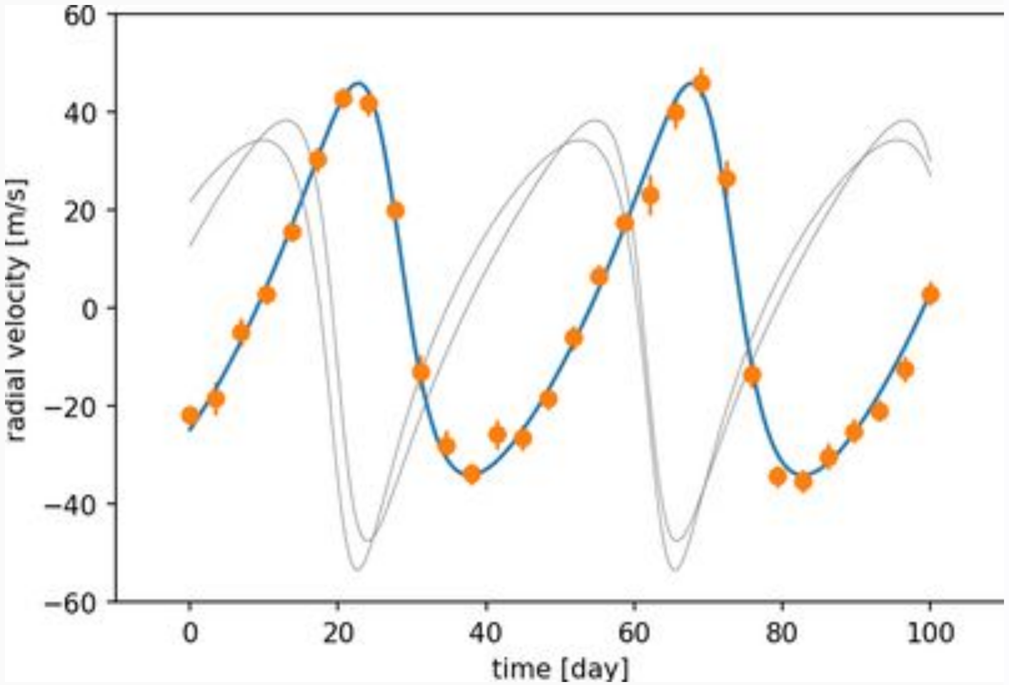
Will not cover:

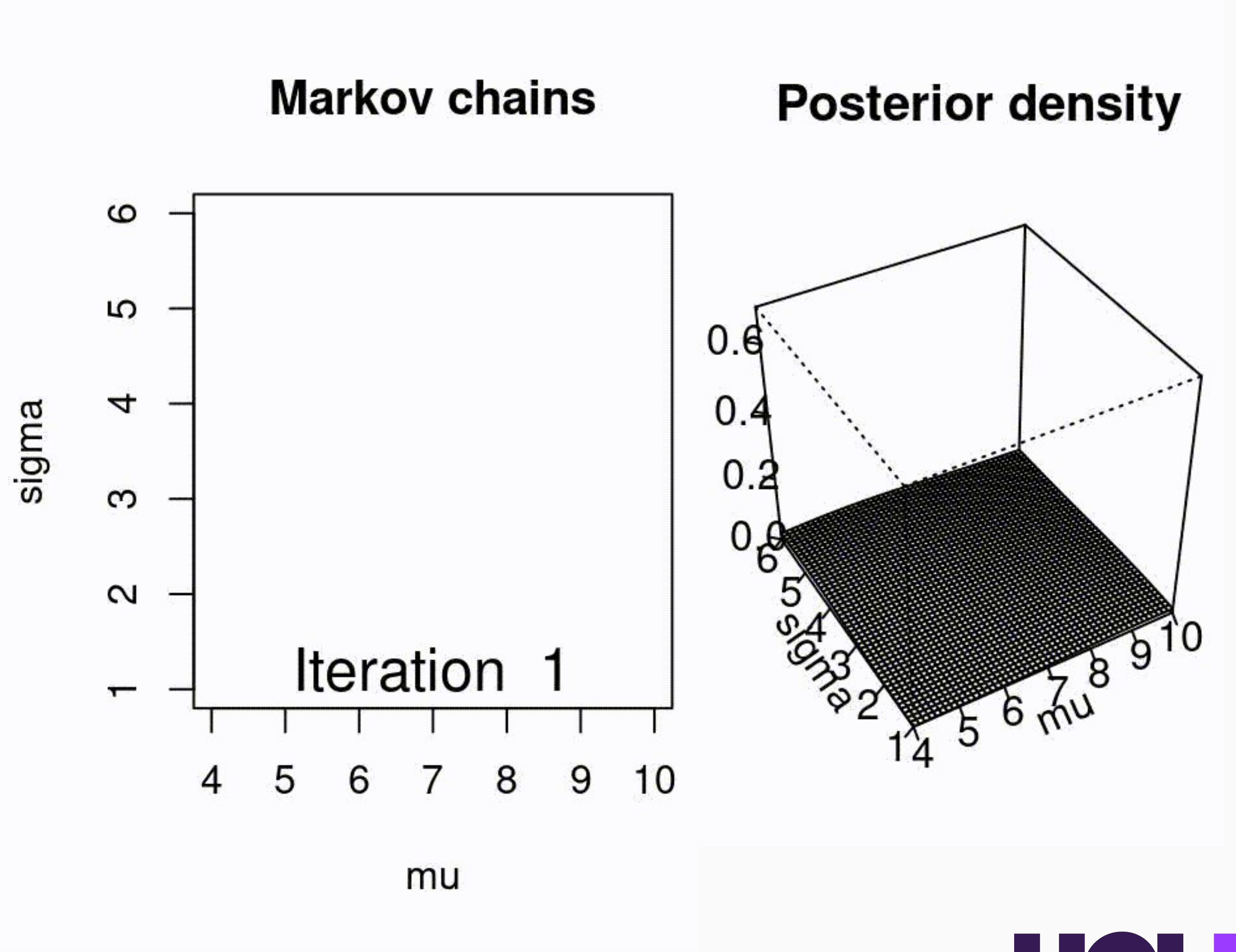
Implicit Likelihood Inference

Variational Inference

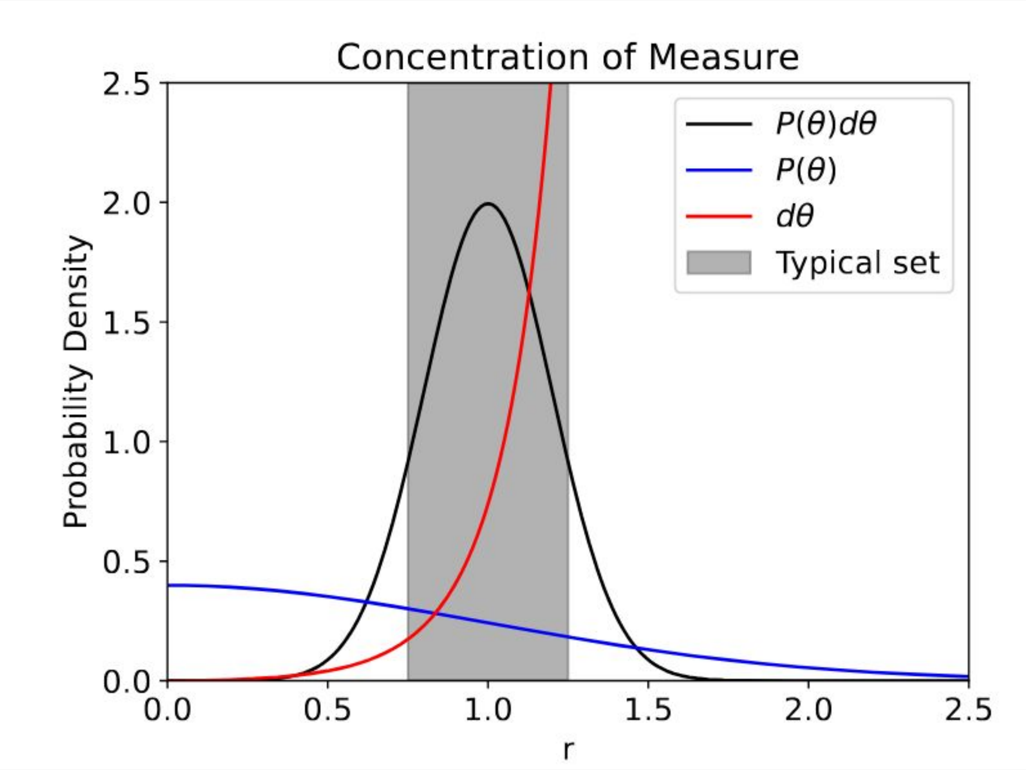
Introduction: Bayesian Inference (MCMC)

Data fit



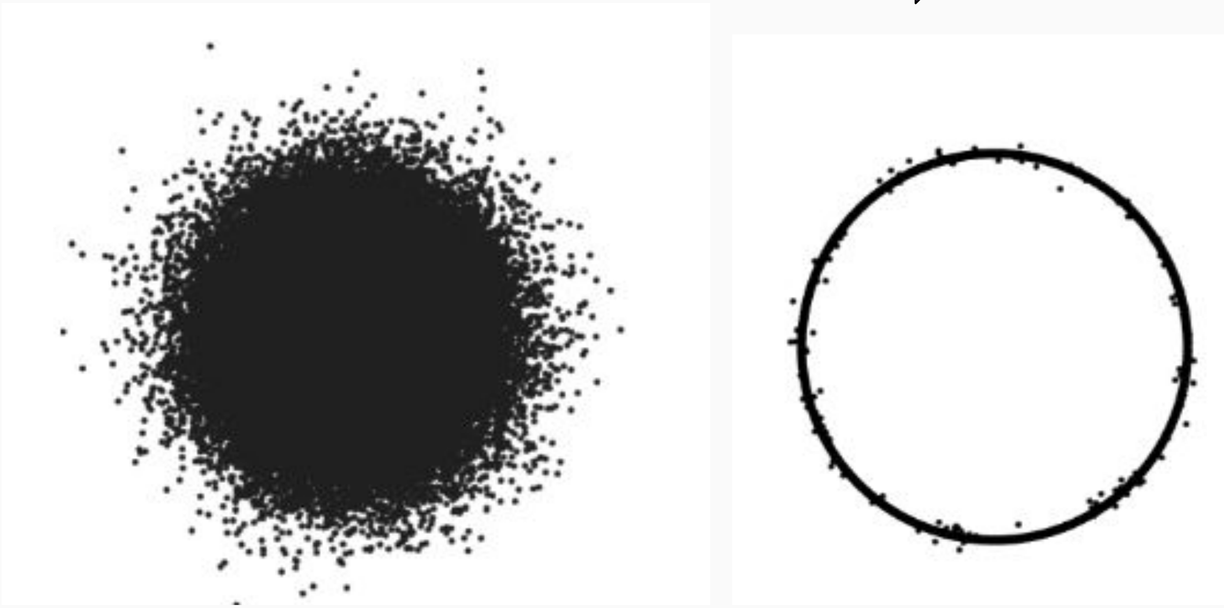


Introduction:
Why is high-dimensional inference expensive?



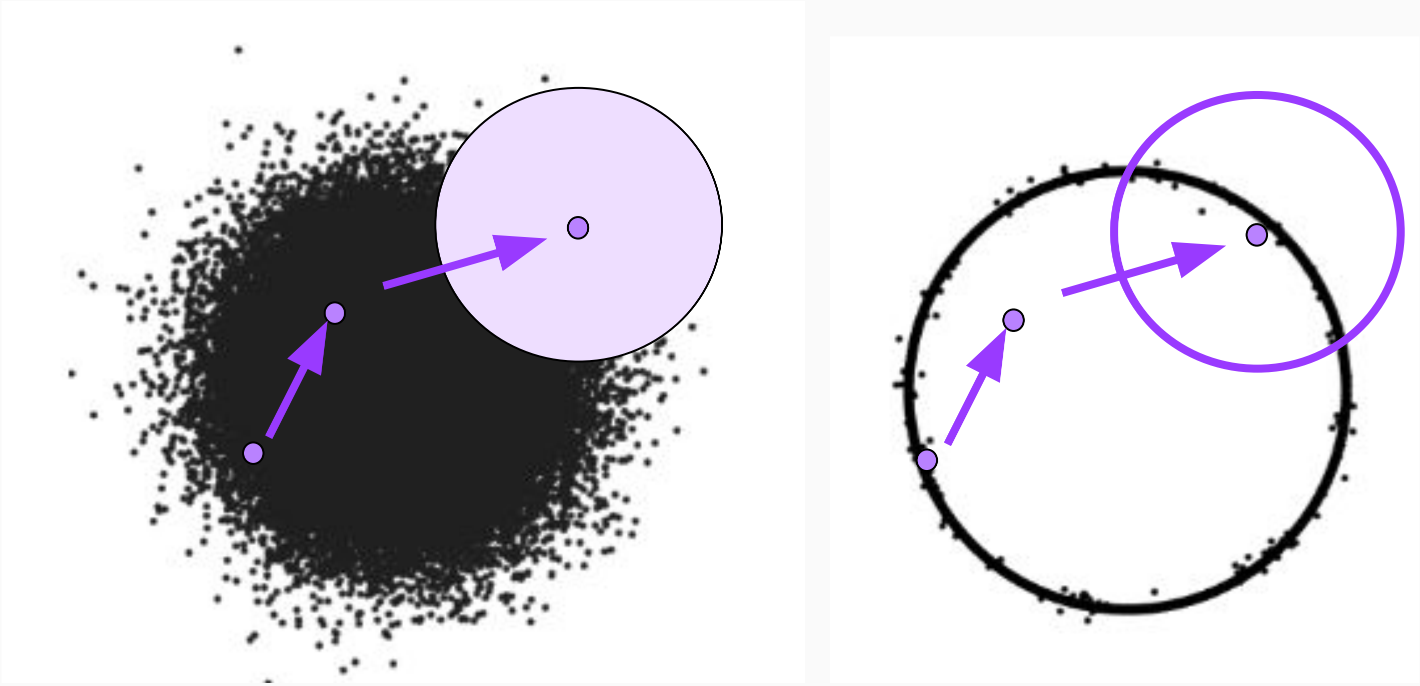
Betancourt, 2017

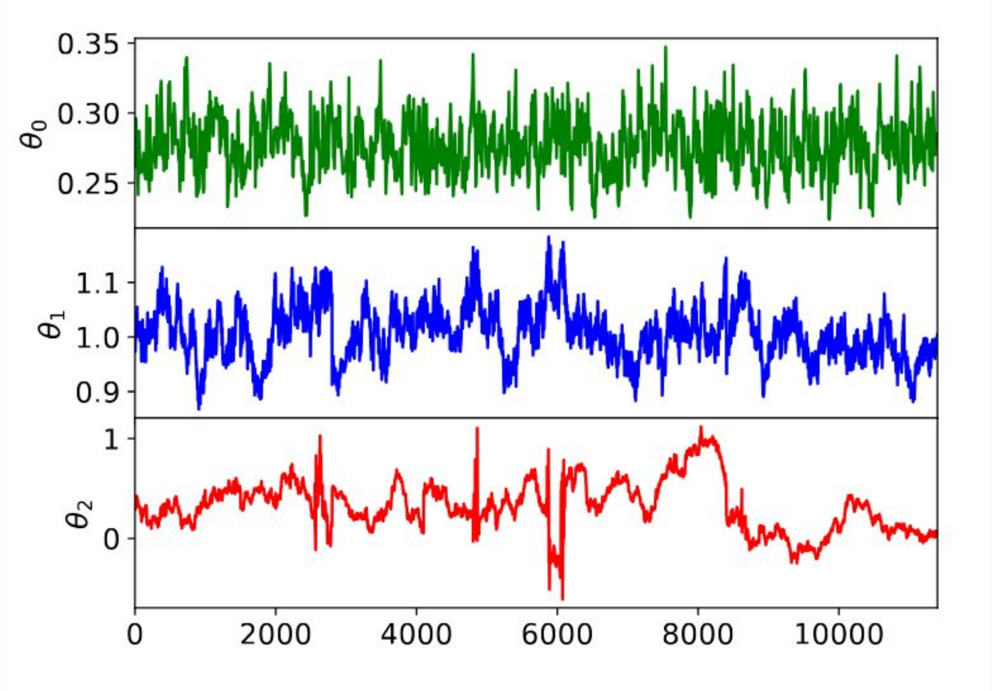
Number of parameters



2D slice of a 2D Gaussian vs 2D slice of a N-Gaussian

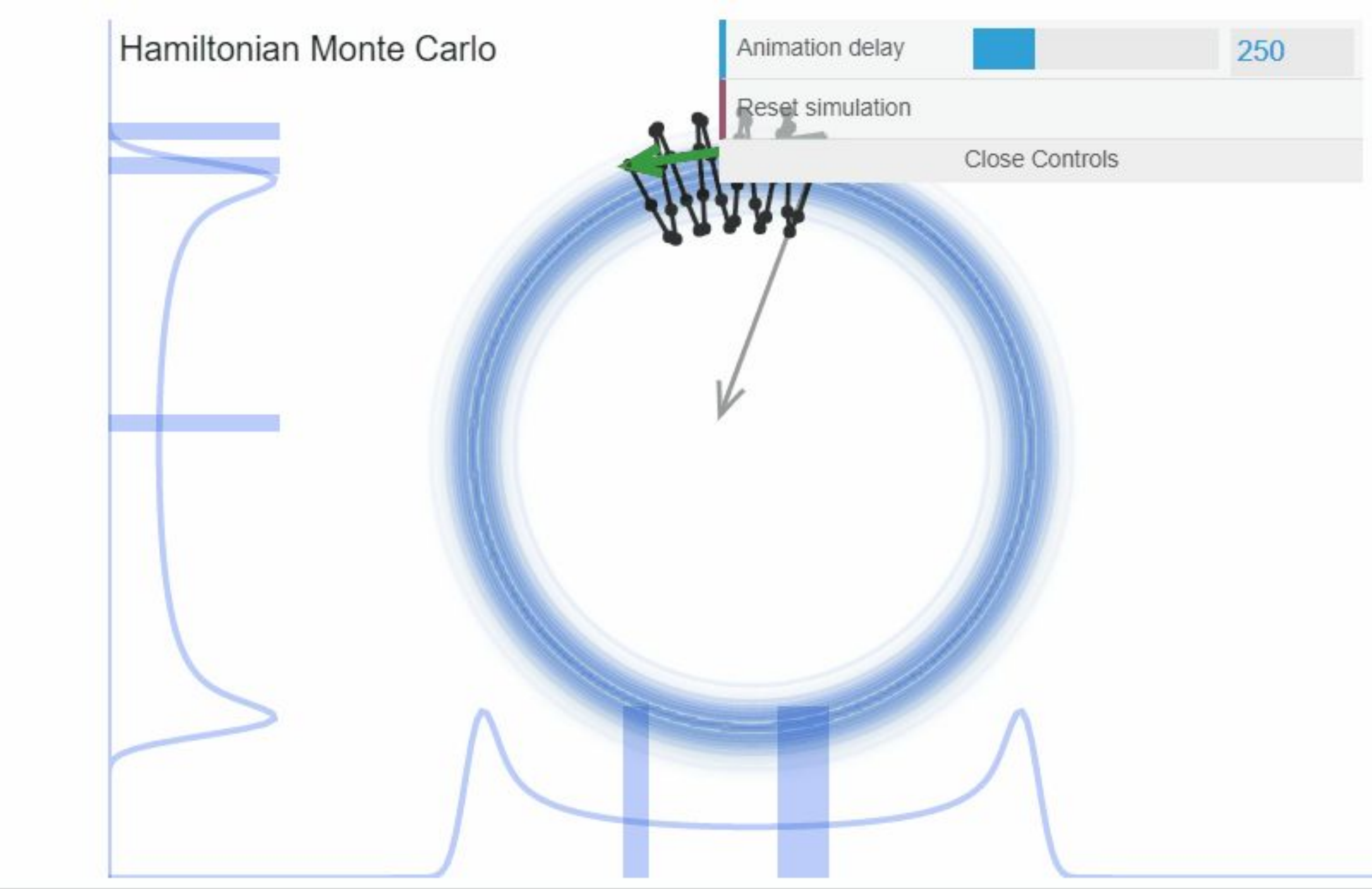
Introduction:
What does Metropolis Hastings do?





2D slice of a 2D Gaussian vs 2D slice of a N-Gaussian

Gradient based Sampling



Gradient based Sampling: HMC

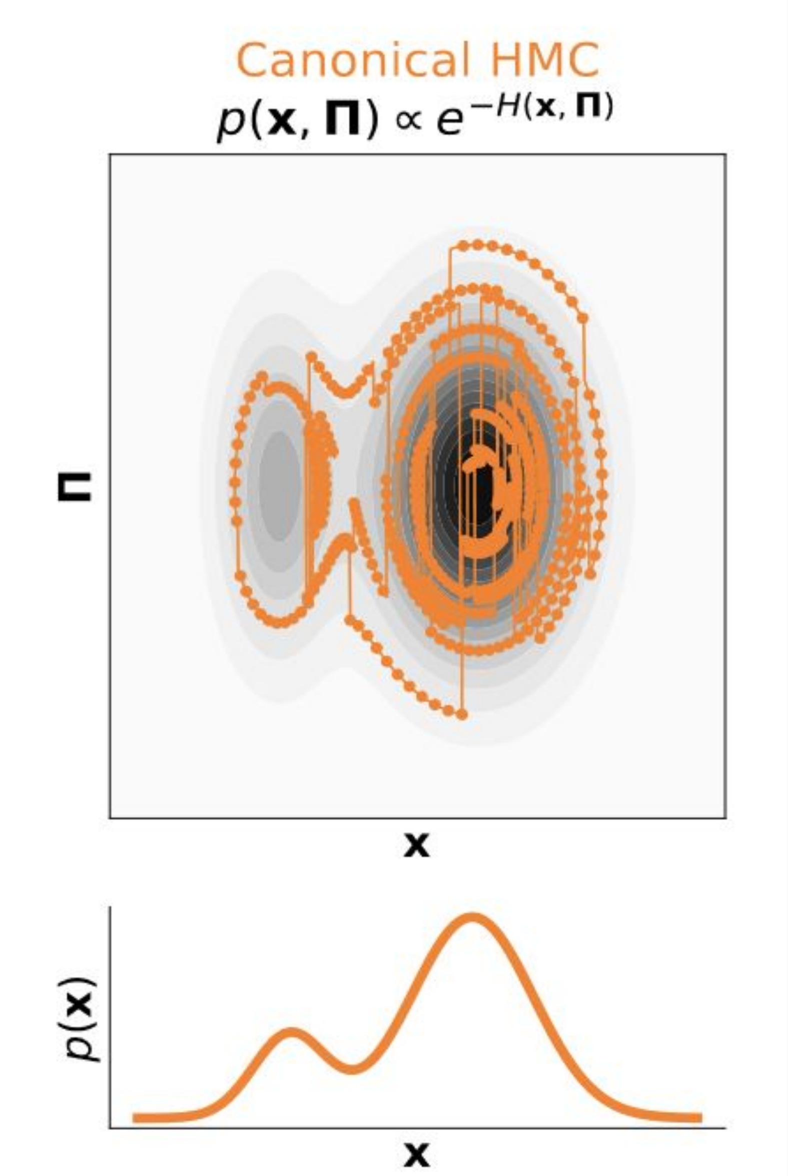
1. Go from a D dimensional space (x) to a 2D dimensional space (x, p) by sampling momenta from a multivariate Gaussian

2. Draw a Hamiltonian

$$H(x, p) = \frac{p^2}{2\Lambda} - log(\mathcal{L}(x))$$

3. Marginalize momenta

$$P(x) = \int e^{-H(x,p)} dp$$



Robnik+23

Gradient based Sampling: MCLMC

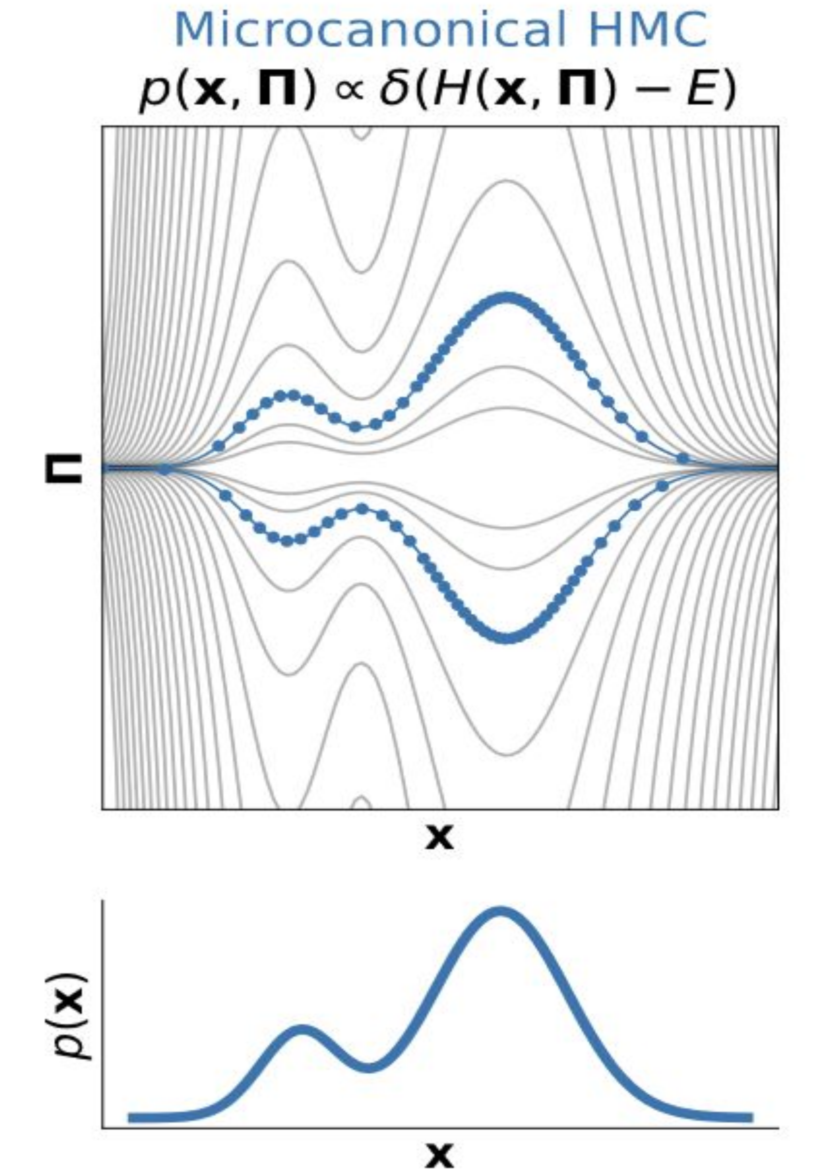
1. Go from a D dimensional space (x) to a 2D dimensional space (x, p) by sampling momenta from a multivariate Gaussian

2. Draw a Hamiltonian

$$H(x,p) = \frac{p^2}{2m(x)} = \frac{p^2}{2e^{-2\mathcal{L}/d}}$$

3. Marginalize over momenta

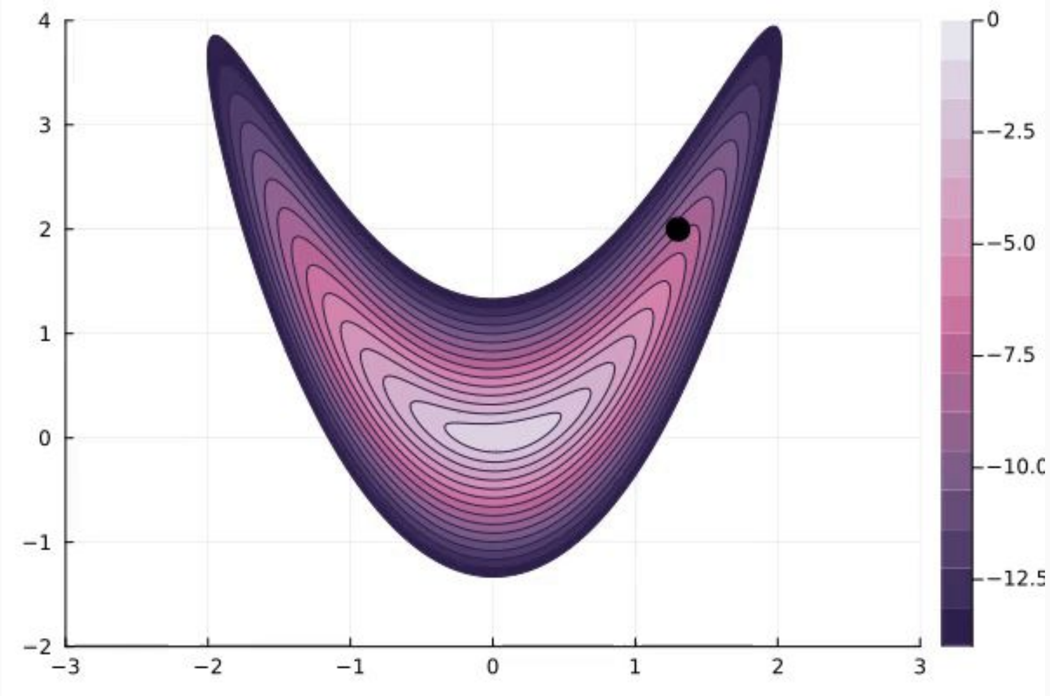
$$P(x) = \int \delta(H(x,p) - E)dp$$



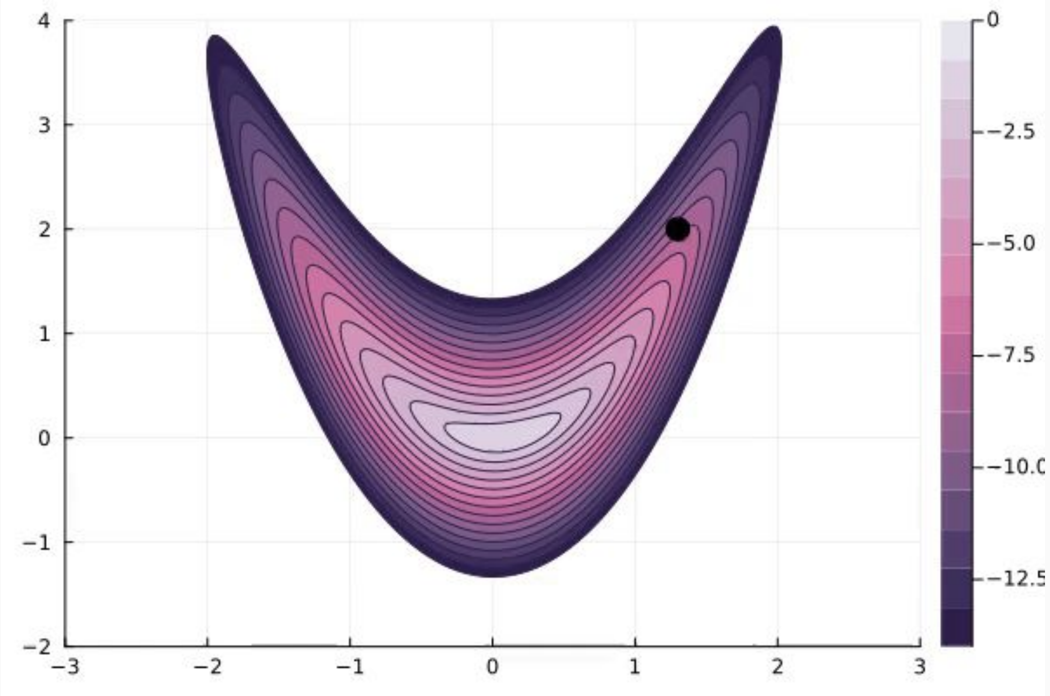
Robnik+23

Gradient based Sampling

Hamiltonian (canonical) (adjusted) Monte Carlo



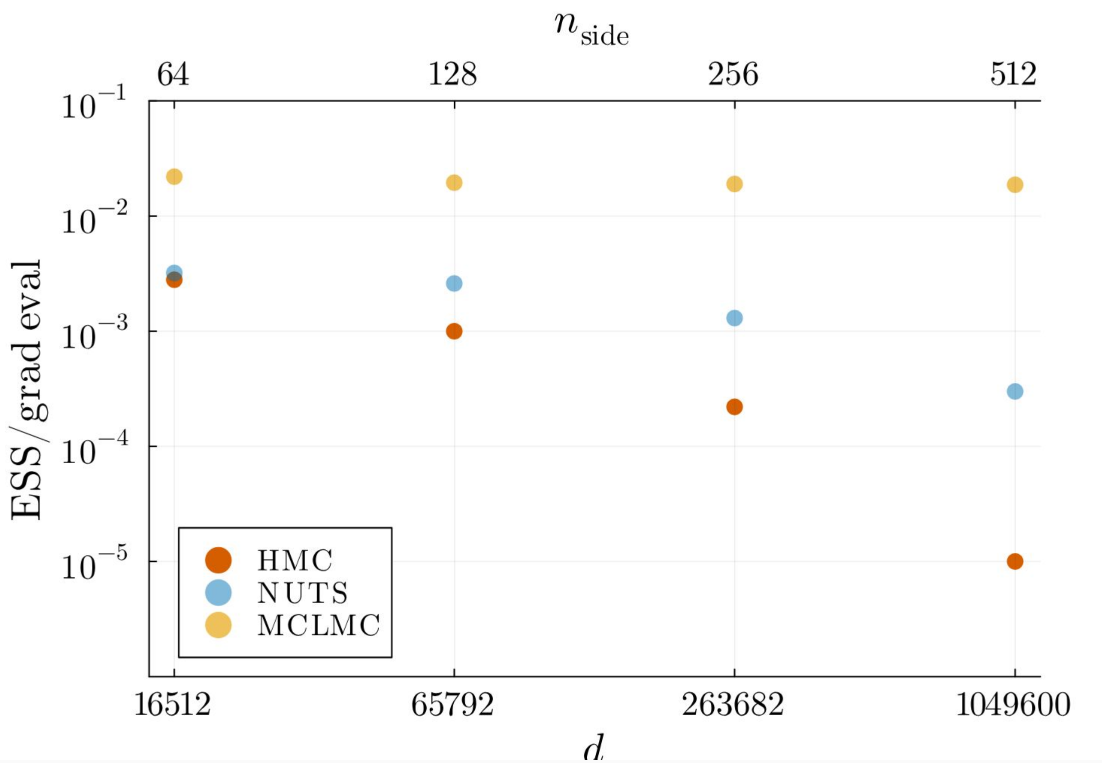
Langevine Micro-Canonical (unadjusted) Monte Carlo



Gradient based Sampling

	Planck	
	ESS/s	Comp. (s)
`PlanckLite.jl`+NUTS	13	1031
MOPED+NUTS	60	228
`PlanckLite.jl`+MCHMC	32	605
MOPED+MCHMC	150	157
`PlanckLite.jl`+Pathfinder	N/A	1.5
MOPED+Pathfinder	N/A	0.5

Planck 2018 analysis (Bonici+23)

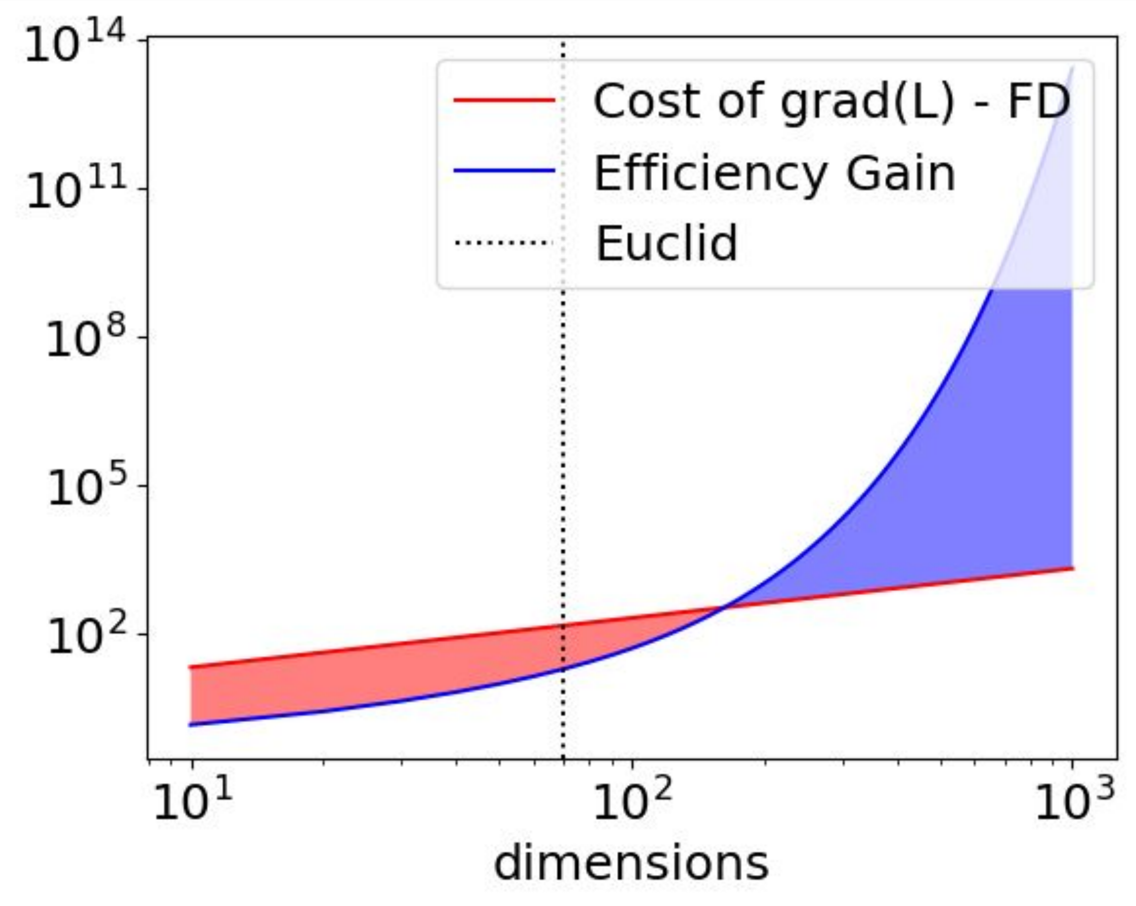


CMB field level inference (Crespi+26)

Gradient based sampling

Choice of:			
Dynamics	Hamiltonian	Langevin	
Ensemble	Canonical	Micro-canonical	Macro-Canonical
Safety	Adjusted	Unadjusted	
Space	Euclidean	Preconditioned	Riemannian

Gradient based Sampling



The cost of the gradient using finite difference is always at least 2D the cost of the likelihood

Auto-differentiation

Humans

$$\frac{d}{dx}\left[f\left(g(x)\right)\right] = f'\left(g(x)\right)g'(x)$$

VS

Robots

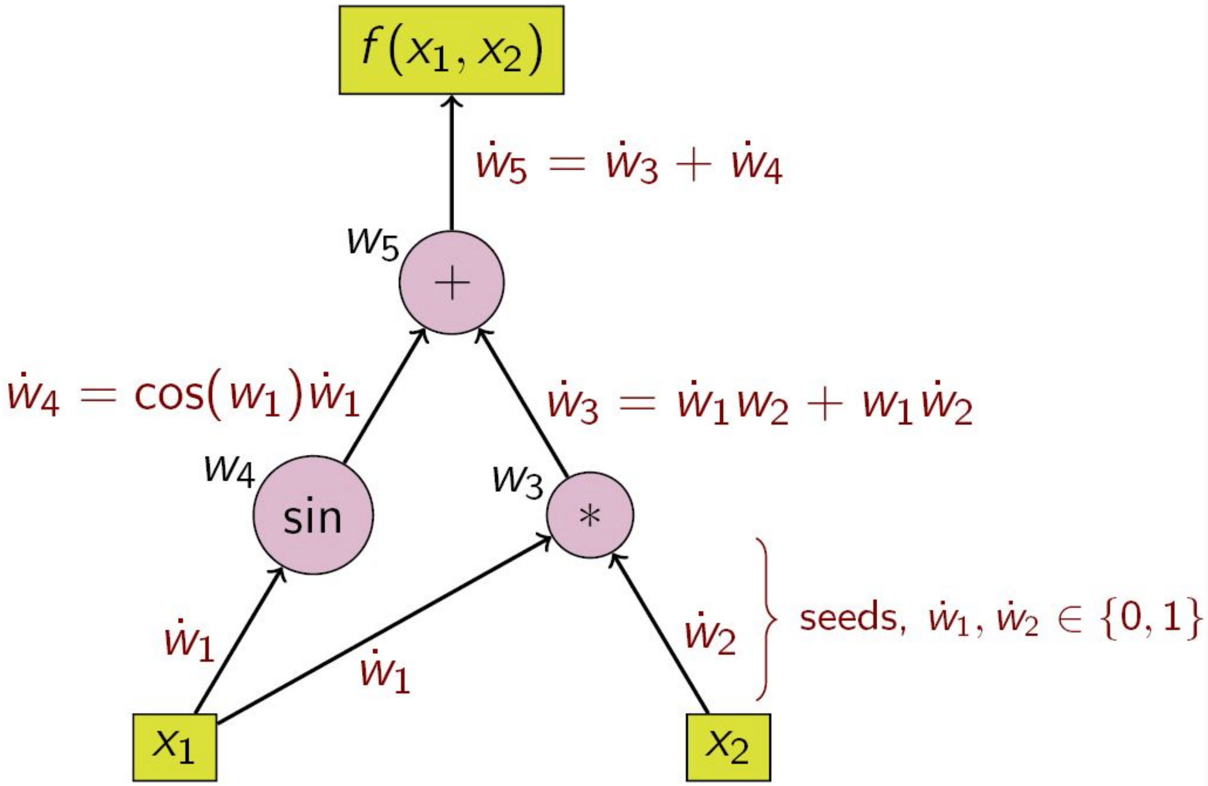
$$\frac{dy}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Auto-differentiation

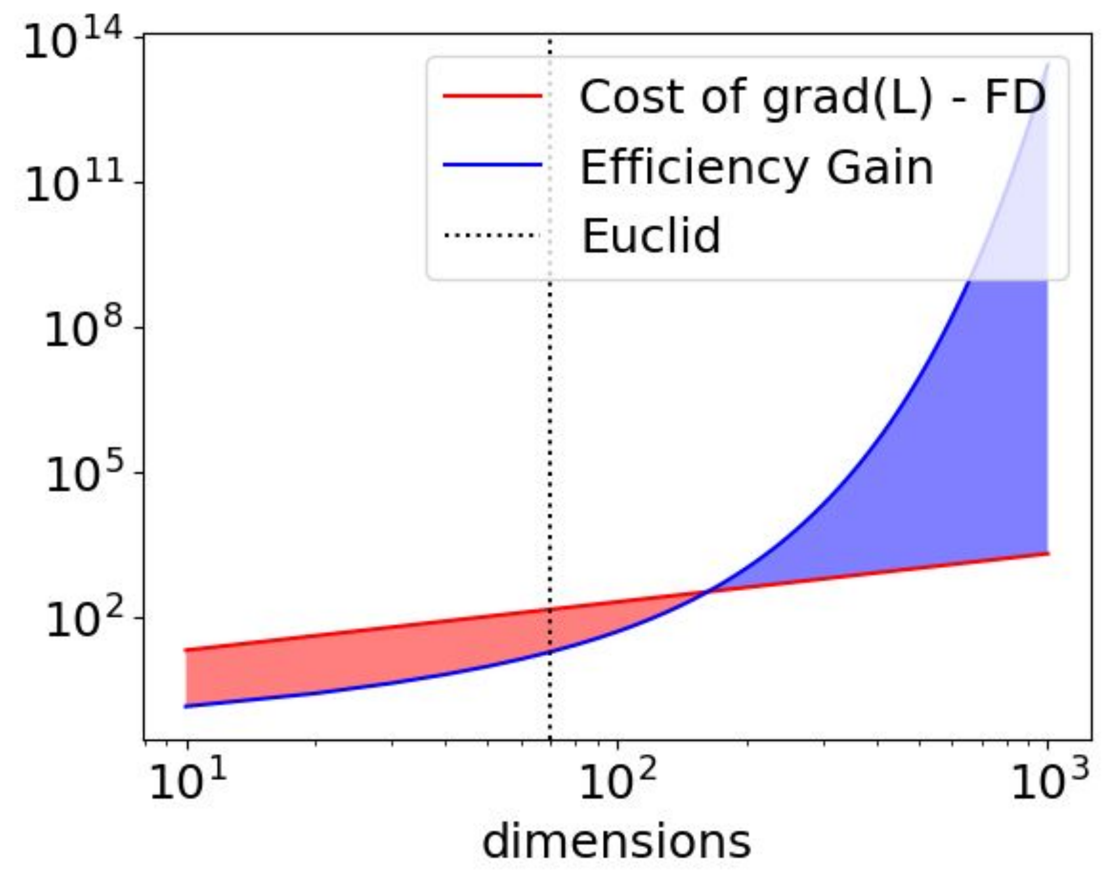
Teach your computer a representation of the programs it runs
Assume a program:
f(R^N) = R^M:

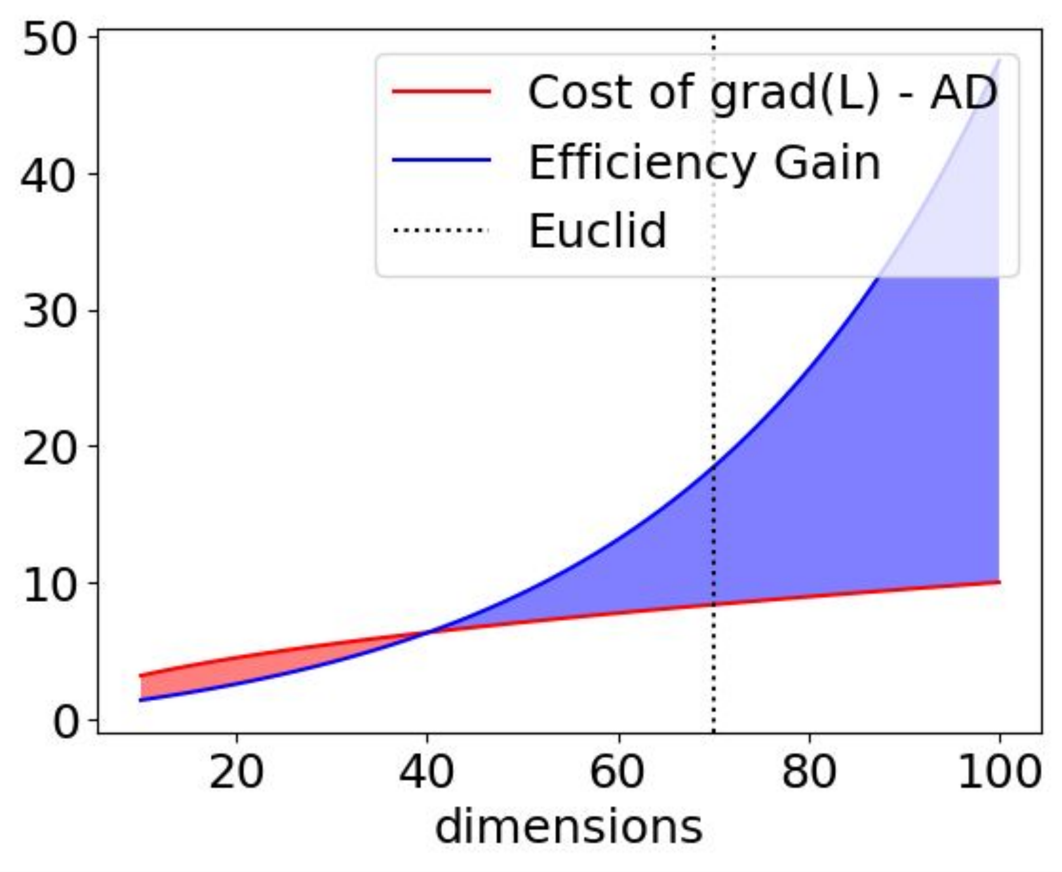
- Forwards (Wengert 64)
  Scales with N
- Backwards (Linnainmaa 76)
  Scales with M.
  For M=1 and If done optimally
  It can be as low as 5 times the cost of f (Baur & Strassen 83)

Implementations in Julia and JAX



Auto-differentiation



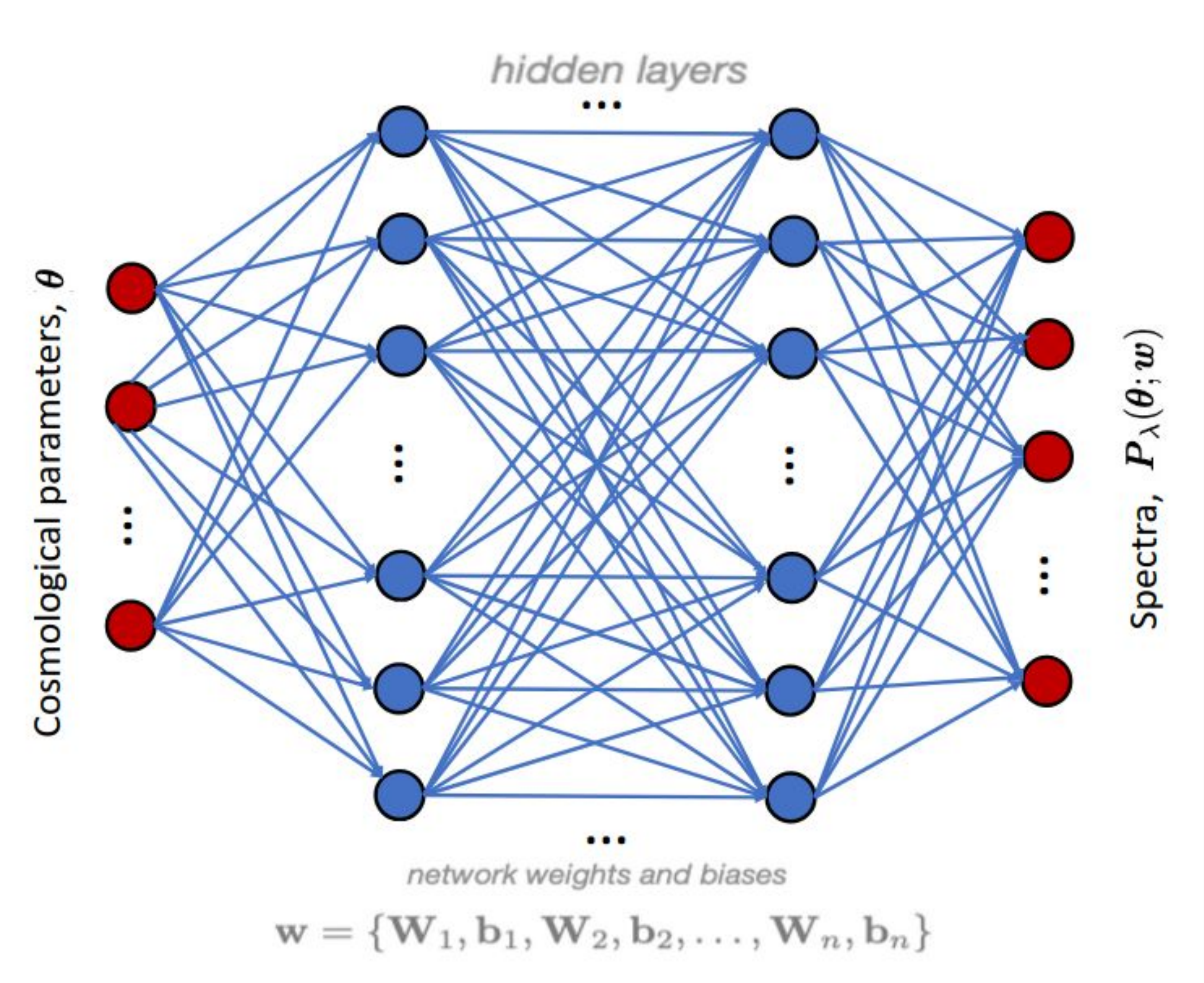


Emulation

AD and GPU's require code to become a series of matrix operations.

Operations such as root finding (Halofit) or ODE solving (Primordial Pk) tend not to comply with this

ML models turn small complex operations into large simple ones



Spurio-Macini+21

Emulation

	`ClassNet`	`CosmoPower`	`Capse.jl`
Speedup	3	1,000	1,000,000
Architecture	NN*	4x512-NN + PCA	5x64-NN + Chebyshev
Training points	10^4	10^5	10^4
Tessellation	Hyperellipsoid	LHC	LHC
Emulation domain			
$\ln 10^{10} A_\mathrm{s}$	N/A	[2.5, 2.5]	[2.5, 3.5]
n_s	N/A	[0.8812, 1.0492]	[0.88, 1.06]
τ	$N(0.0561, 0.0071)$	[0.02, 0.12]	[0.02, 0.12]
H_0[km/s/Mpc]	$N(67.66, 0.42)$	[39.99, 100.01]	[40, 100]
ω_b	$N(0.02242, 0.00014)$	[0.0193, 0.02533]	[0.0193, 0.02533]
ω_c	$N(0.11933, 0.00091)$	[0.08, 0.2]	[0.08, 0.2]
ℓ_max	2,500	11,000	5,000

Bonici+23

The Laplace Approximation

$$n_*(\boldsymbol{\Omega}) \equiv \arg\ \max_{\boldsymbol{n}} p(\boldsymbol{\Omega}, \boldsymbol{n}|\boldsymbol{d}) \quad \Rightarrow \quad \left.\frac{\partial \chi^2}{\partial \boldsymbol{n}}\right|_{\boldsymbol{n}_*} = 0.$$

Now expand your log likelihood around n*

$$\chi^2(\boldsymbol{\Omega}, \boldsymbol{n}) \simeq \chi^2_*(\boldsymbol{\Omega}) + \Delta\boldsymbol{n}^T \mathcal{F}_* \Delta\boldsymbol{n},$$

Where $\Delta\boldsymbol{n} = \boldsymbol{n} - \boldsymbol{n}*$ and $\mathcal{F}_{*,ij} = \left.\frac{1}{2}\frac{\partial^2\chi^2}{\partial n_i \partial n_j}\right|_{\boldsymbol{n}_*}$

The Laplace Approximation

In this limit, the distribution is locally (i.e. at each Ω) is a multivariate normal distribution in n, and thus

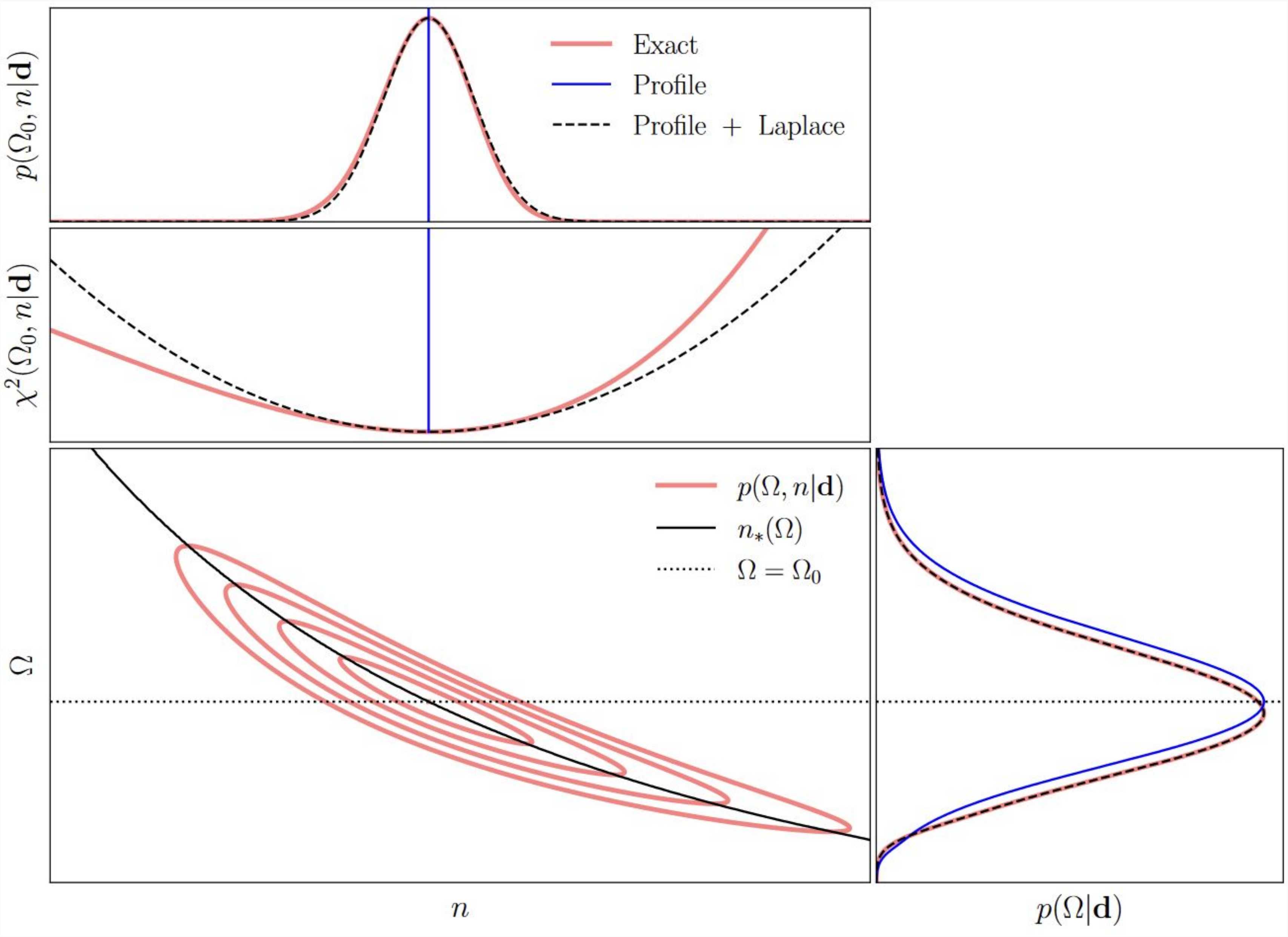
$$P(\boldsymbol{\Omega}|\boldsymbol{d}) \propto \int d\boldsymbol{n} P(\boldsymbol{\Omega}, \boldsymbol{n}|\boldsymbol{d})$$

can be solved analytically leading to the following approximated log likelihood:

$$\chi^2_m(\boldsymbol{\Omega}) \simeq \chi^2_*(\boldsymbol{\Omega}) + \log\left\{\det\left[\mathcal{F}_*(\boldsymbol{\Omega})\right]\right\} + \text{const.}.$$

Profile likelihood

Laplace term



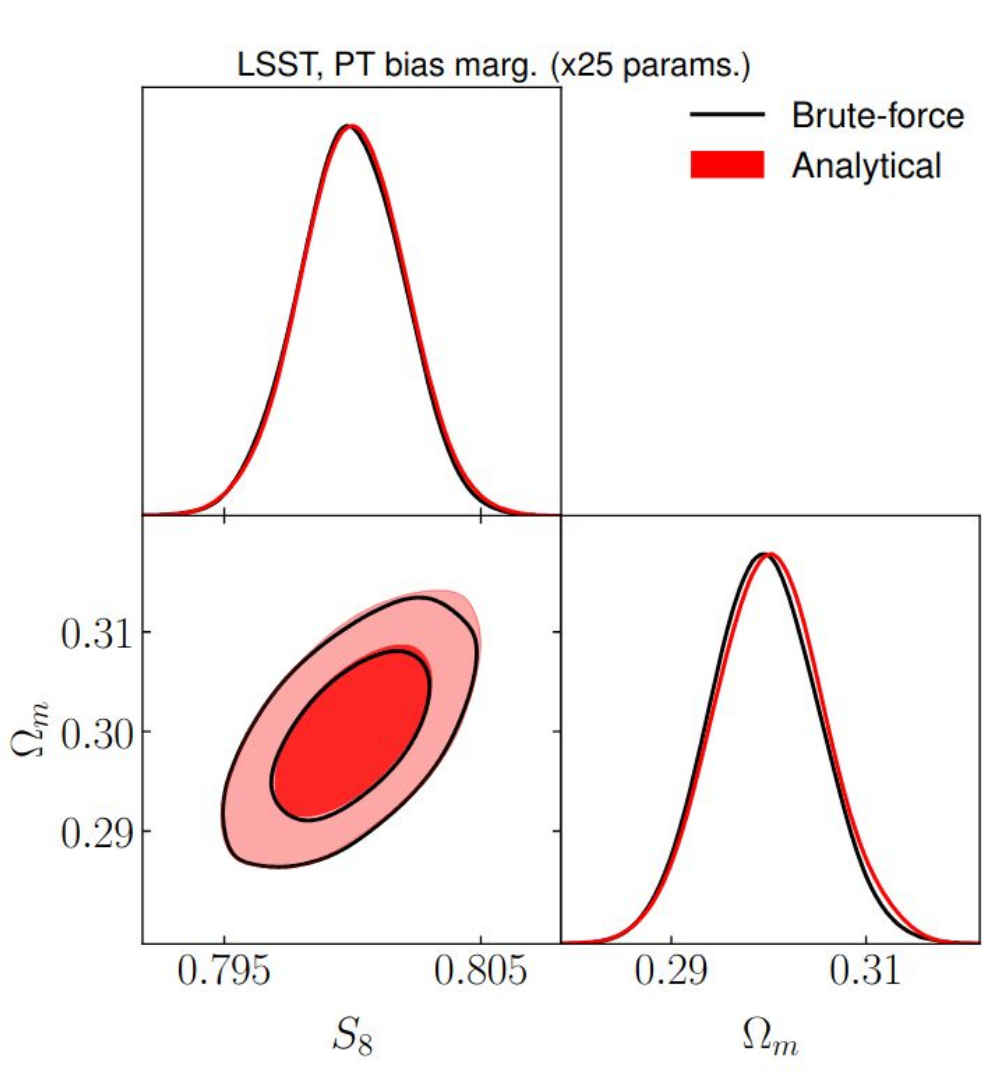
$p(\Omega_0, n|\mathbf{d})$
Exact
Profile
Profile + Laplace
$\chi^2(\Omega_0, n|\mathbf{d})$
$p(\Omega, n|\mathbf{d})$
$n_*(\Omega)$
$\Omega = \Omega_0$
Ω
n
$p(\Omega|\mathbf{d})$

The Laplace Approximation

Tested on DESY1 and LSSTY1-like analyses.

Cosmology with 6 parameters in 1/20 of the time.

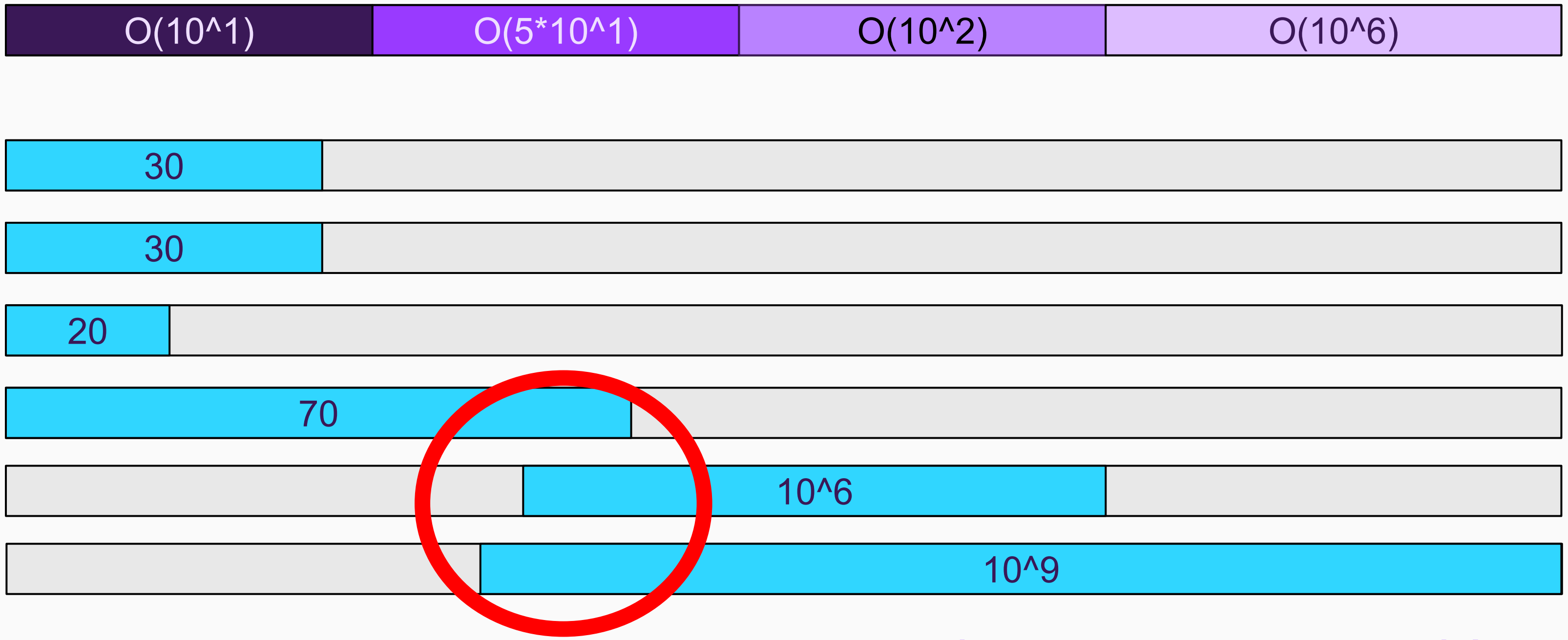
Also requires the gradient.



Hadzhiyska+23
Ruiz-Zapatero+24

Practical advise: what sampler to use

Number of parameters



- Cobaya (MH)
- EMCEE (ensemble)
- Polychord (nested)
- Nautilus (Surrogate)
- NUTS (HMC)
- MCLMC

LimberJack (Ruiz-Zapatero+23)
SiwftCl (Reymond+25)
PyBird (Reeves+25)
Capse (Bonici+23)
JAXCOSMO (Campagne+23)
(Mootoovaloo+24)

(My best guess)*

Practical advise: what framework to use

	Julia	JAX
Easy of use	New language / self-contained,	Python based / Interface to LAX,
Performance	Very performant forwards / Lacking backwards	Very performant backwards
GPU	Requires extra effort	Out of the box if you stay vanilla

Practical advise: How to write your likelihood

LCDM analysis

In [8]:
```
@model function LCDM_model(data, data_cov)
    # Priors
    omegam_pr ~ Uniform(0.1, 0.5)
    h0_pr ~ Uniform(0.6, 0.8)
    s8_pr ~ Uniform(0.6, 0.9)

    # Create cosmology
    cosmo_pr = LimberJack.Cosmology(Ωm=cosmo1.cpar.Ωm, Ωb=cosmo1.cpar.Ωb, h=cosmo1.cpar.h, σ8=s8_pr);

    # Model predictions
    ez_model = Ez(cosmo_pr, z_integ1)
    hz_model := hz_from_ez(z_integ1, h0_pr, ez_model)
    dm_model := dm_from_ez(z_integ1, h0_pr, ez_model)
    fs8_model := fs8_from_ez(z_integ1, omegam_pr, s8_pr, h0_pr, ez_model, eltype(ez_model))

    # Create interpolators
    hz_itp = LinearInterpolation(z_integ1, hz_model, extrapolation_bc=Line())
    dm_itp = LinearInterpolation(z_integ1, dm_model, extrapolation_bc=Line())
    fs8_itp = LinearInterpolation(z_integ1, fs8_model, extrapolation_bc=Line())

    hz_theory := hz_itp(fakedatahz.z)
    dm_theory := dm_itp(fakedatadm.z)
    fs8_theory := fs8_itp(fakedatafs8.z)

    theory = [hz_theory;dm_theory;fs8_theory]

    # Likelihood
    data ~ MvNormal(theory, data_cov)
end
```

```
LCDM_model (generic function with 2 methods)
```

In [9]:
```
cond_LCDM_model = LCDM_model(data_obs, covariance_obs)
chain_LCDM = sample(cond_LCDM_model, NUTS(20, 0.65), 100)
```

Probabilistic
Programming
Languages's

Practical advise: What infrastructure to build

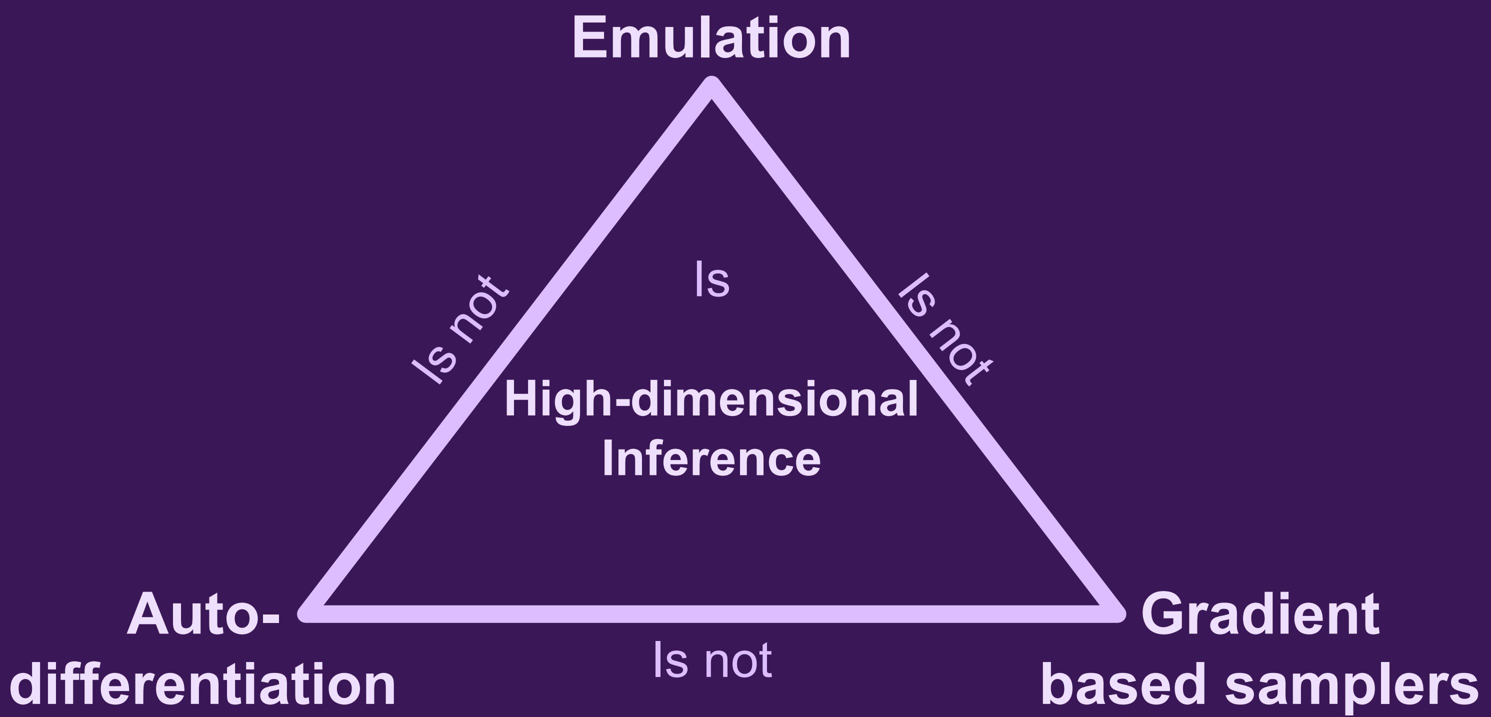
- JAX pipeline
- GPU ensemble MH
- JAX emulators
- Analytical Marginalization
- PPL's

- Auto-differentiable pipeline for extended analyses
- MCLMC for field level

DR1 Analysis	DR2 Analysis
(First 2 years)	(Years 2-5)

Conclusions:

Fast bayesian inference in the next 5 years will depend on the implementation of 3 technologies



(Piras+24)