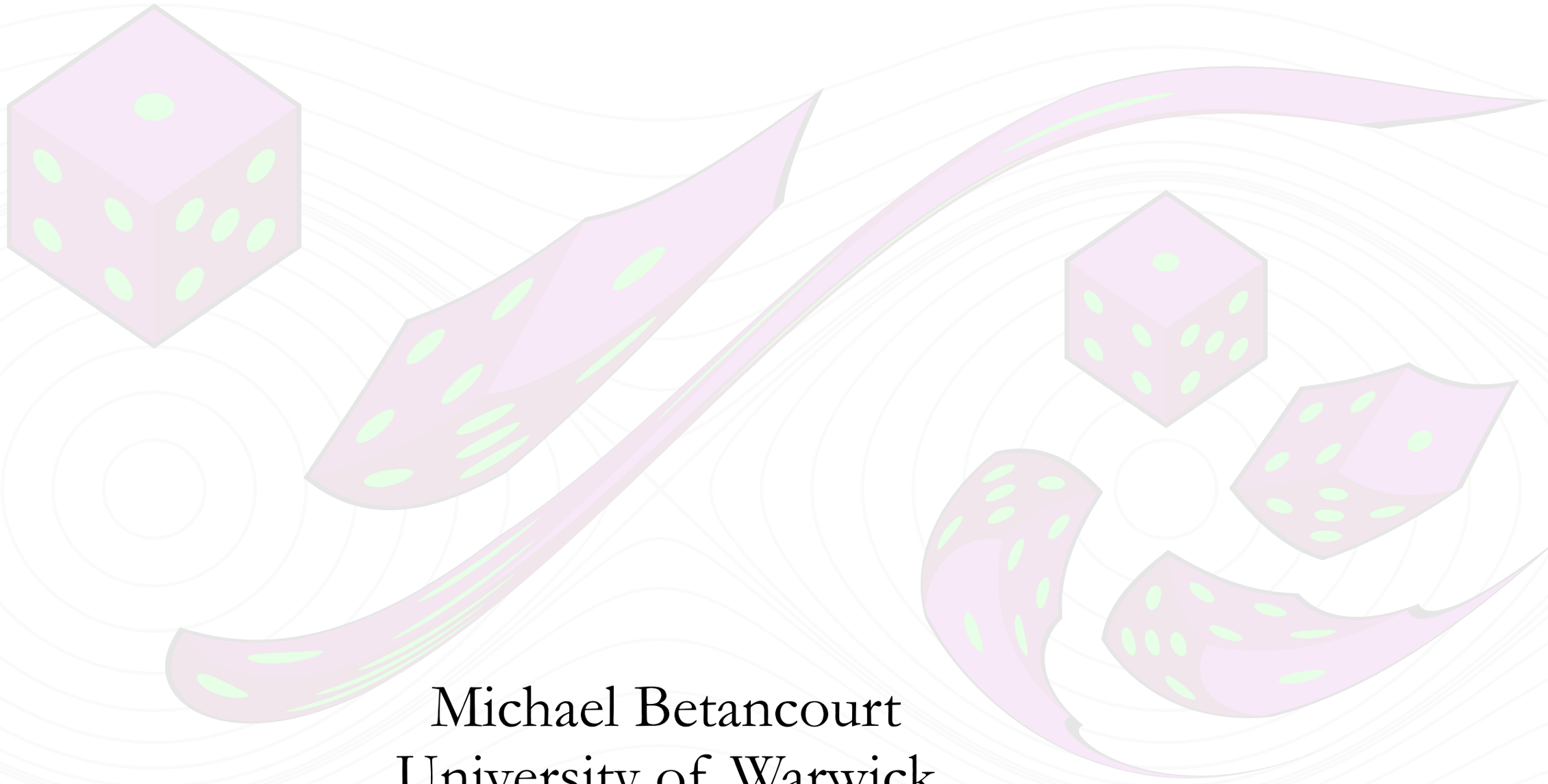
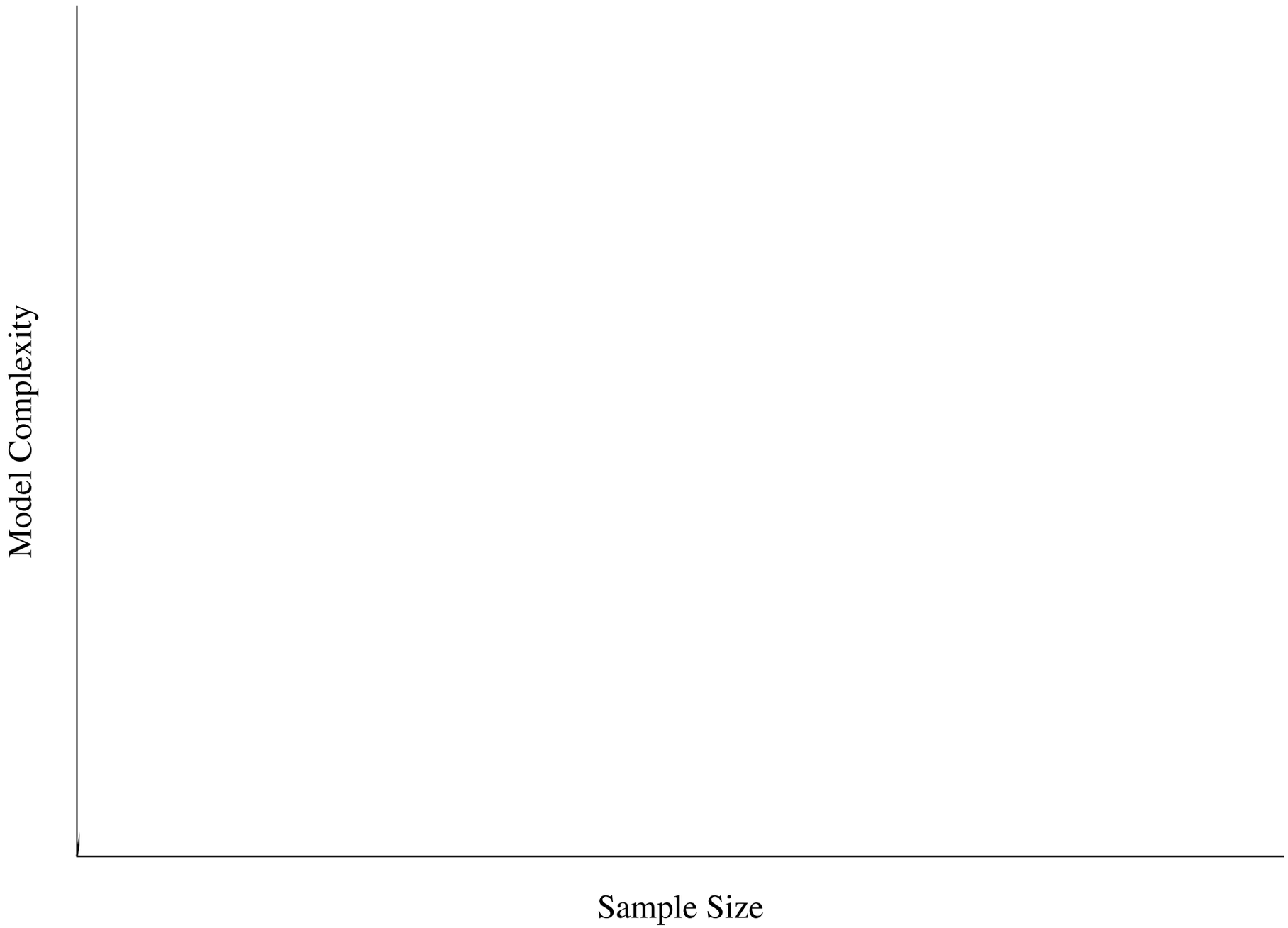
Efficient Bayesian inference with Hamiltonian Monte Carlo

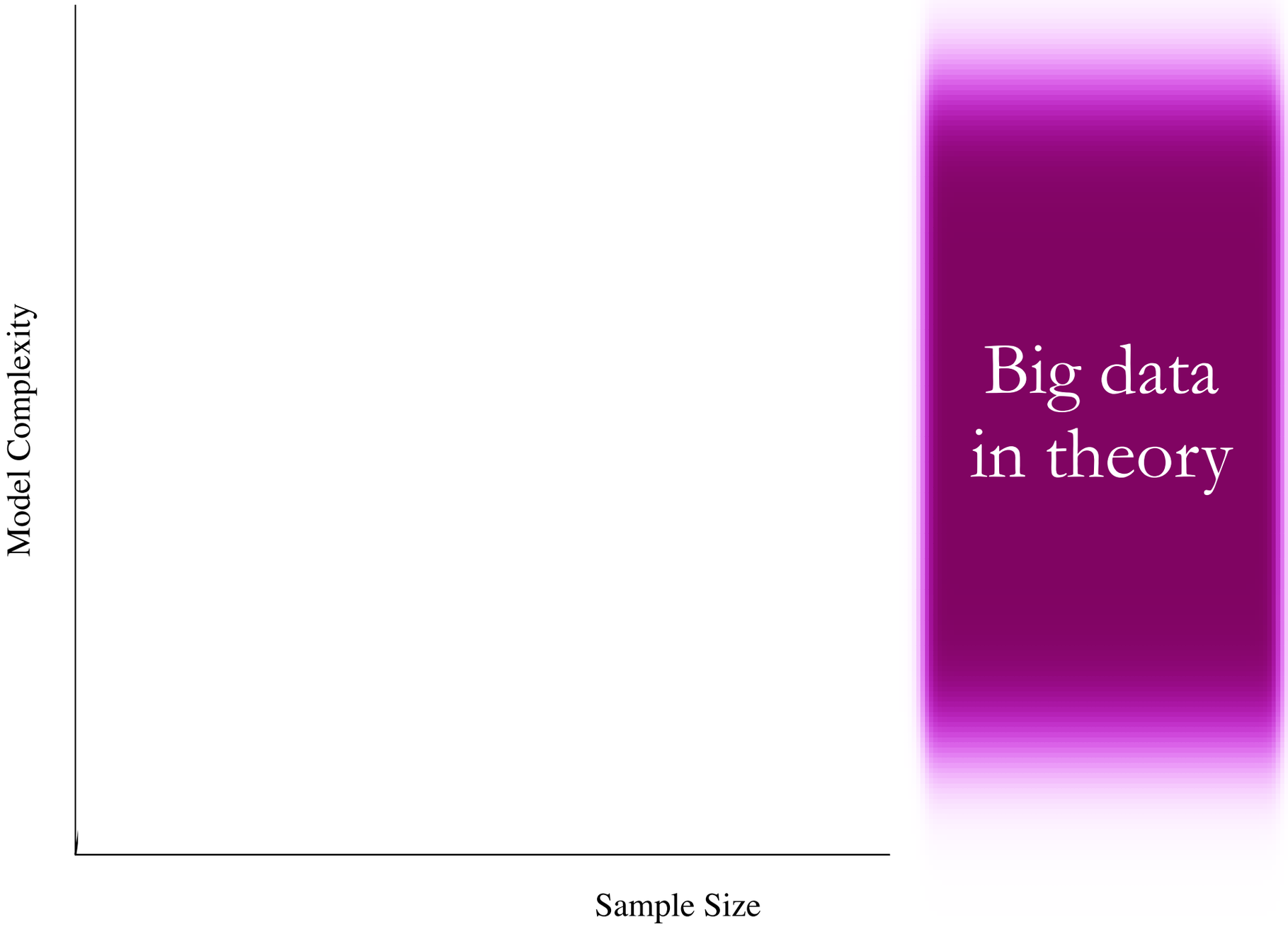


Michael Betancourt
University of Warwick
Machine Learning Summer School 2014
April 29, 2014

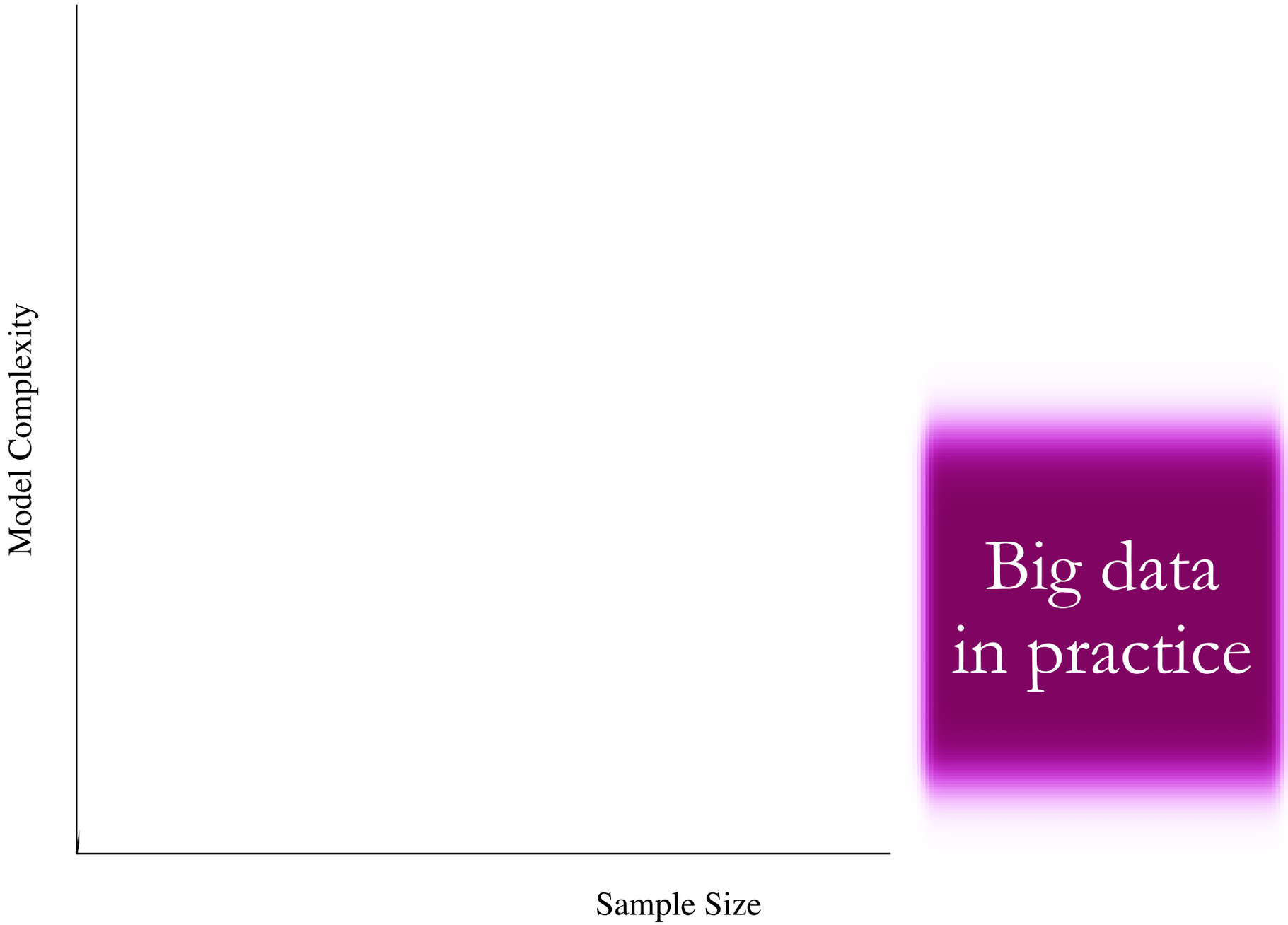
Can big data support big models?



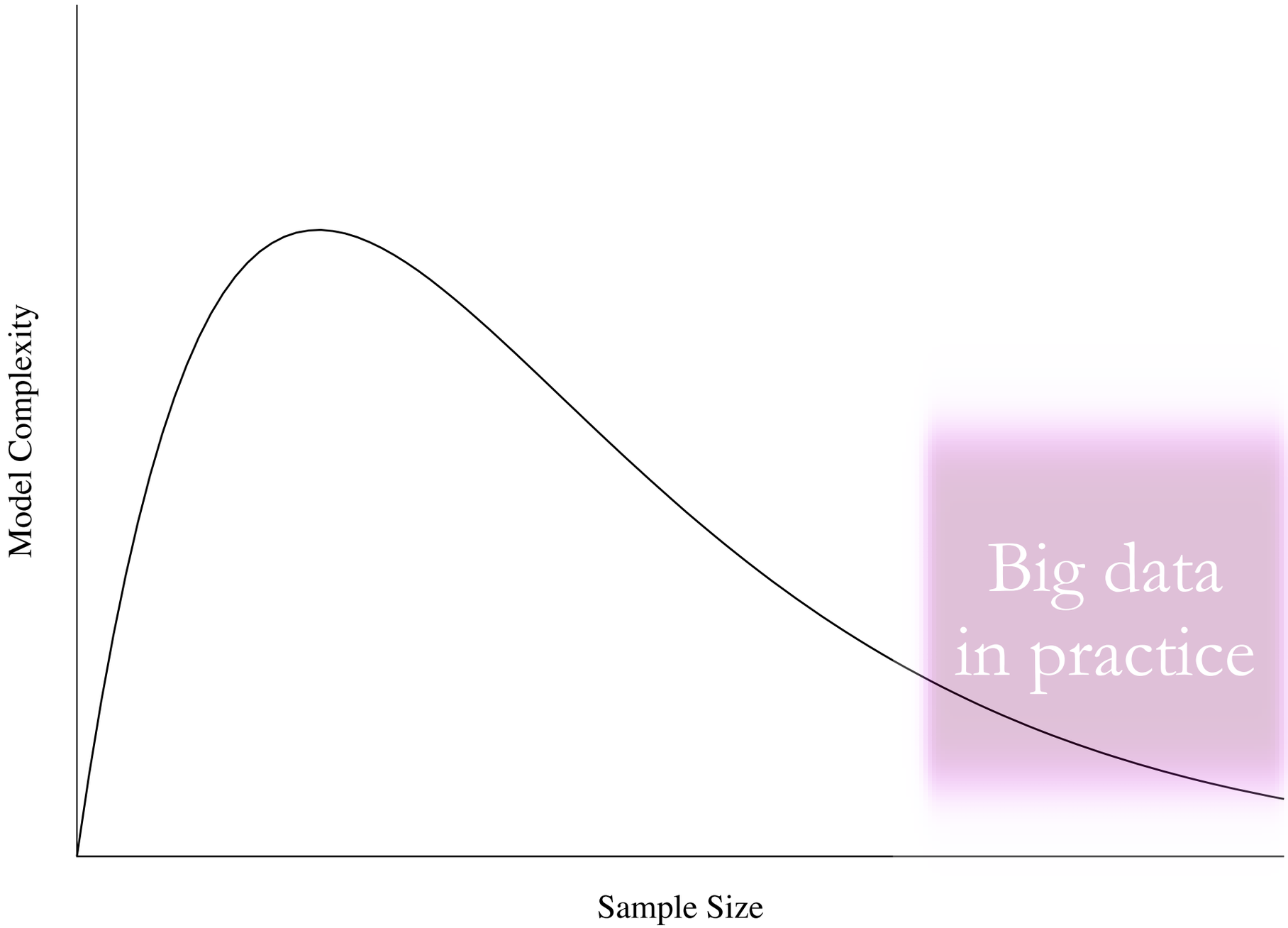
Can big data support big models?



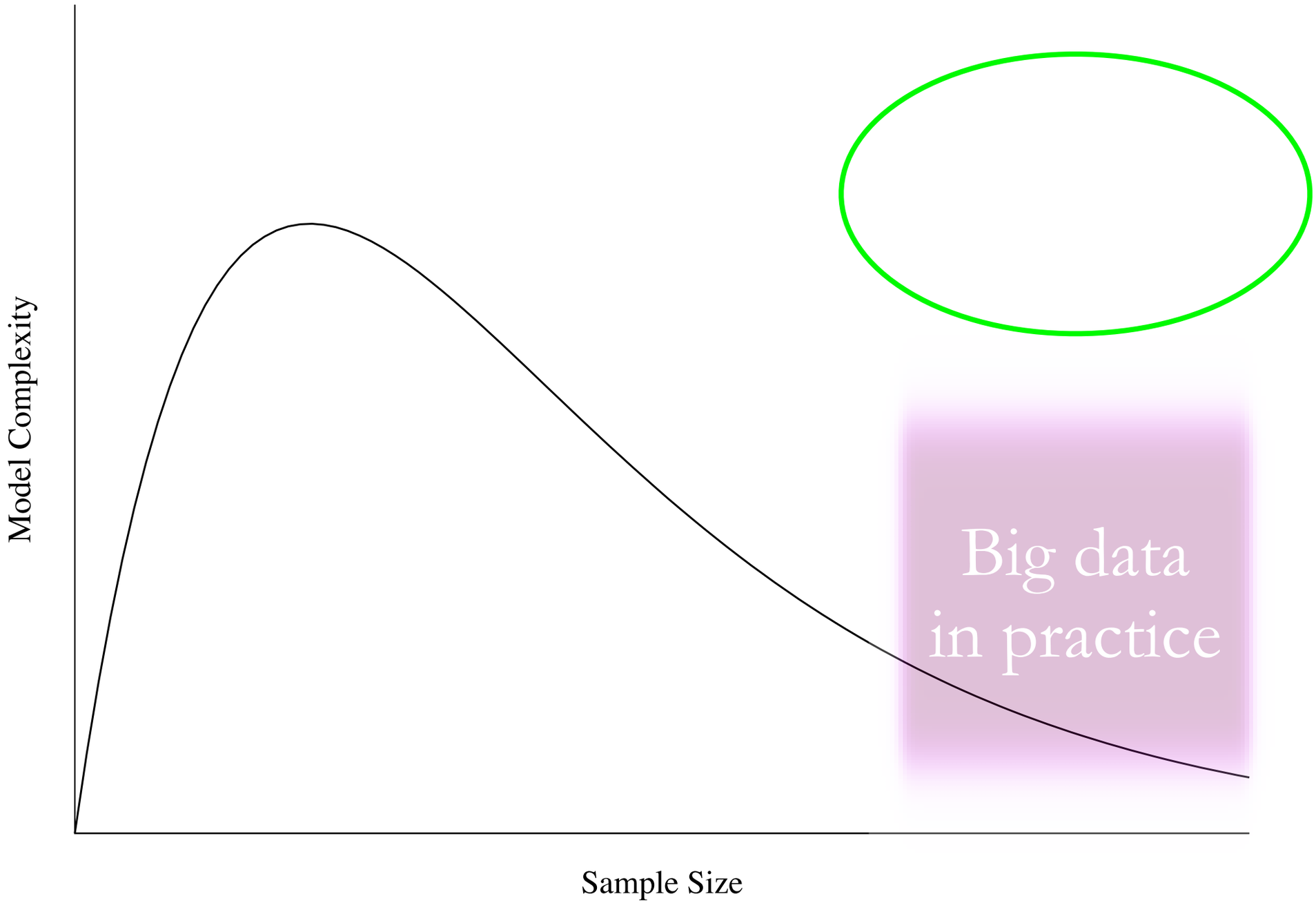
Can big data support big models?



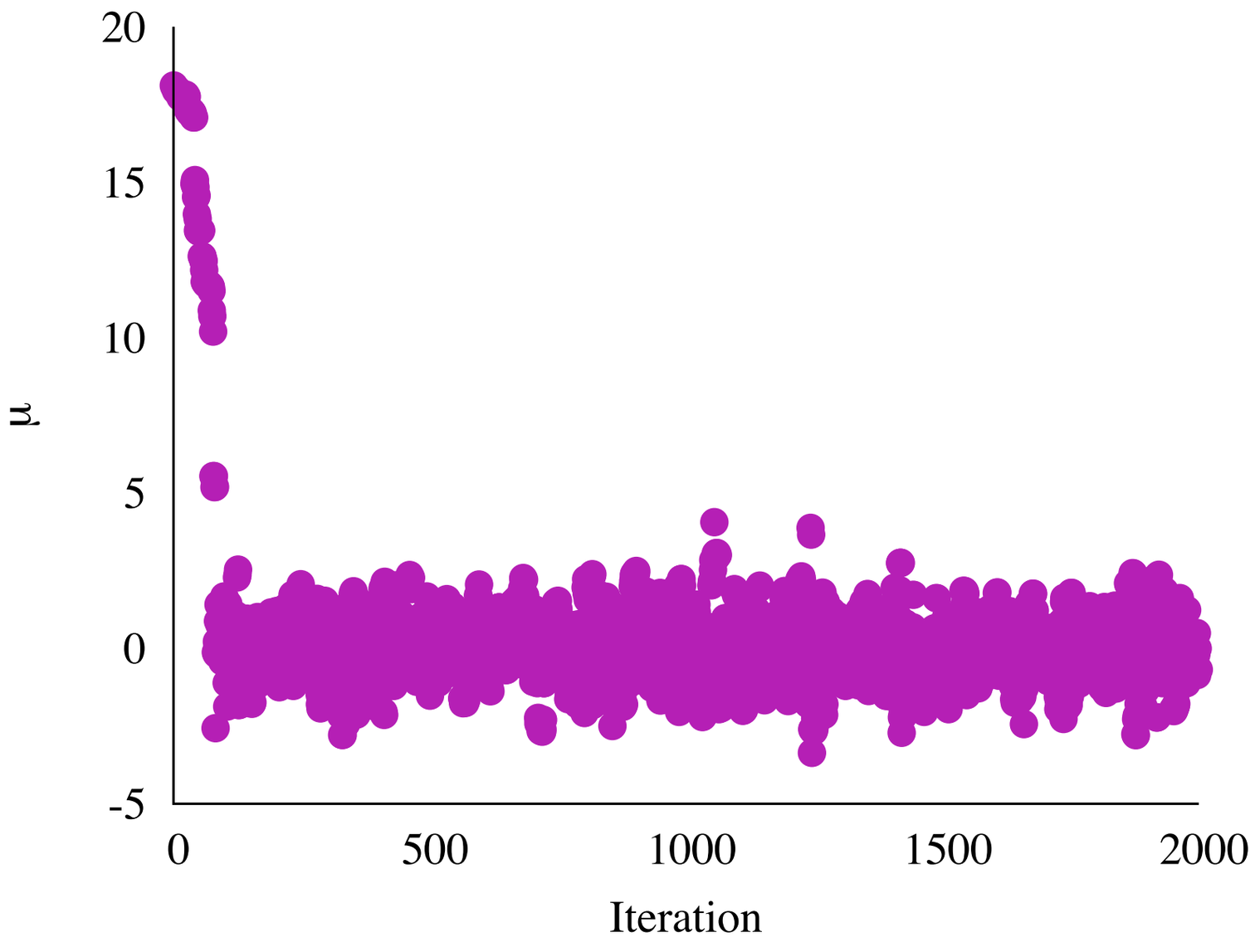
Can big data support big models?



Can big data support big models?



Markov Chain Monte Carlo in Practice



Bayesian inference is a powerful tool
for asking germane statistical questions

Bayesian inference is a powerful tool
for asking germane statistical questions

$$\pi(\theta)$$

Bayesian inference is a powerful tool
for asking germane statistical questions

$$\pi(\mathcal{D}|\theta)\,\pi(\theta)$$

Bayesian inference is a powerful tool
for asking germane statistical questions

$$\pi(\theta|\mathcal{D}) \propto \pi(\mathcal{D}|\theta)\,\pi(\theta)$$

But what makes a good statistical question?

But what makes a good statistical question?

$$f(\hat{\theta}),\ \hat{\theta} = \operatorname{argmax} \pi(\theta)$$

But what makes a good statistical question?

$$f(\hat{\theta}),\ \hat{\theta} = \operatorname{argmax} \pi(\theta)$$

$$\mathbb{E}[f(\theta)] = \int \mathrm{d}\theta\, \pi(\theta)\ f(\theta)$$

Probability densities are a computational convenience --
our questions should not rely on them

Probability densities are a computational convenience --
our questions should not rely on them

$$\pi : \mathcal{B}(\Omega) \to [0, 1]$$

Probability densities are a computational convenience --
our questions should not rely on them

$$\pi : \mathcal{B}(\Omega) \to [0, 1]$$

$$\theta : \Omega \to \mathbb{R}^n$$

$$\mathrm{d}\pi(\theta) = \mathrm{d}\theta \, \pi(\theta)$$

Probability *mass* is fundamental, not density!

$$f(\hat{\theta}),\ \hat{\theta} = \operatorname{argmax} \pi(\theta)$$

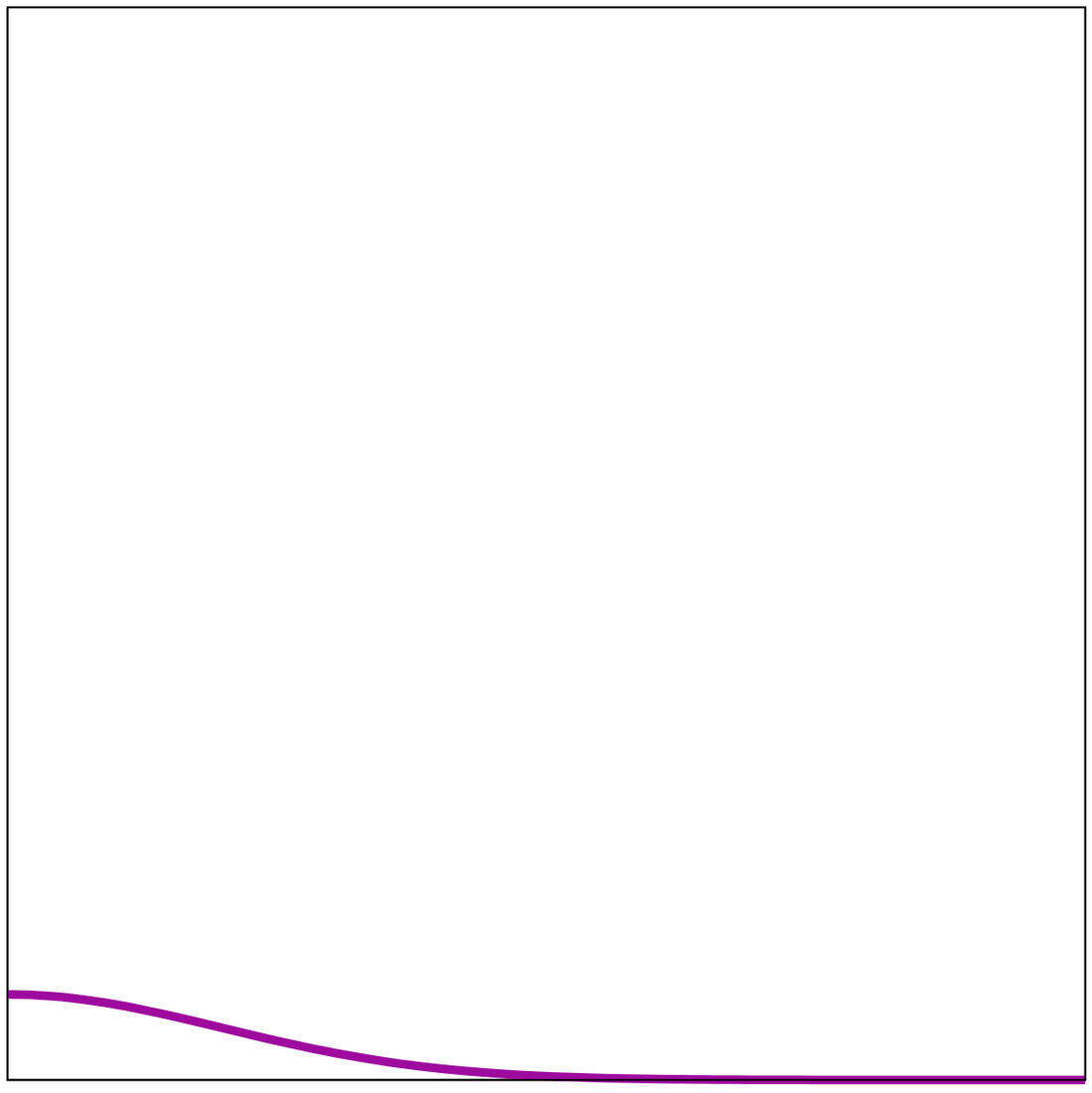
$$\mathbb{E}[f(\theta)] = \int \mathrm{d}\theta\, \pi(\theta)\ f(\theta)$$

Probability *mass* is fundamental, not density!

$$f(\hat{\theta}),\ \hat{\theta} = \operatorname{argmax} \pi(\theta)$$

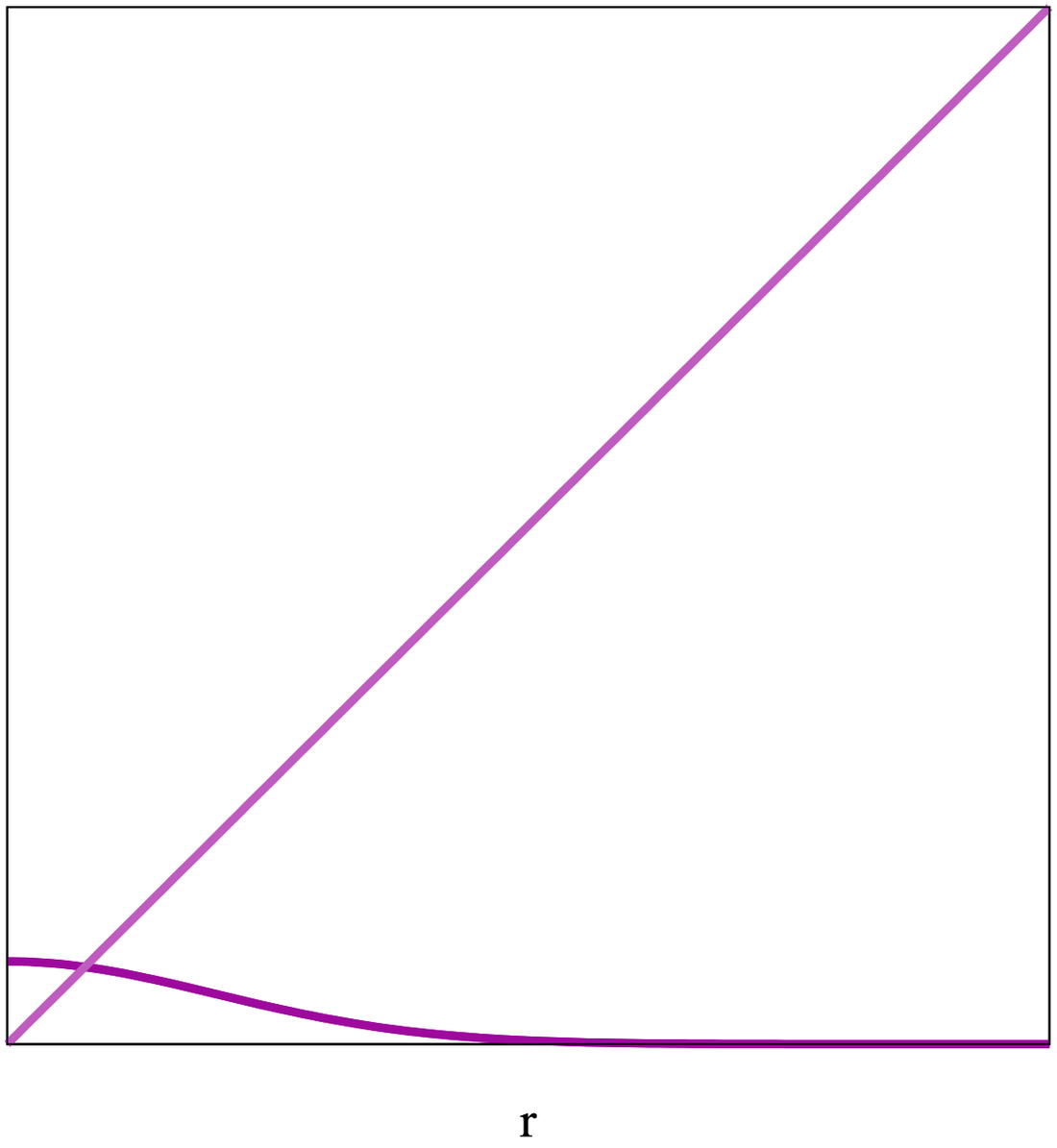
$$\mathbb{E}[f(\theta)] = \int \mathrm{d}\theta\, \pi(\theta)\ f(\theta)$$

And mass can be very far away from density



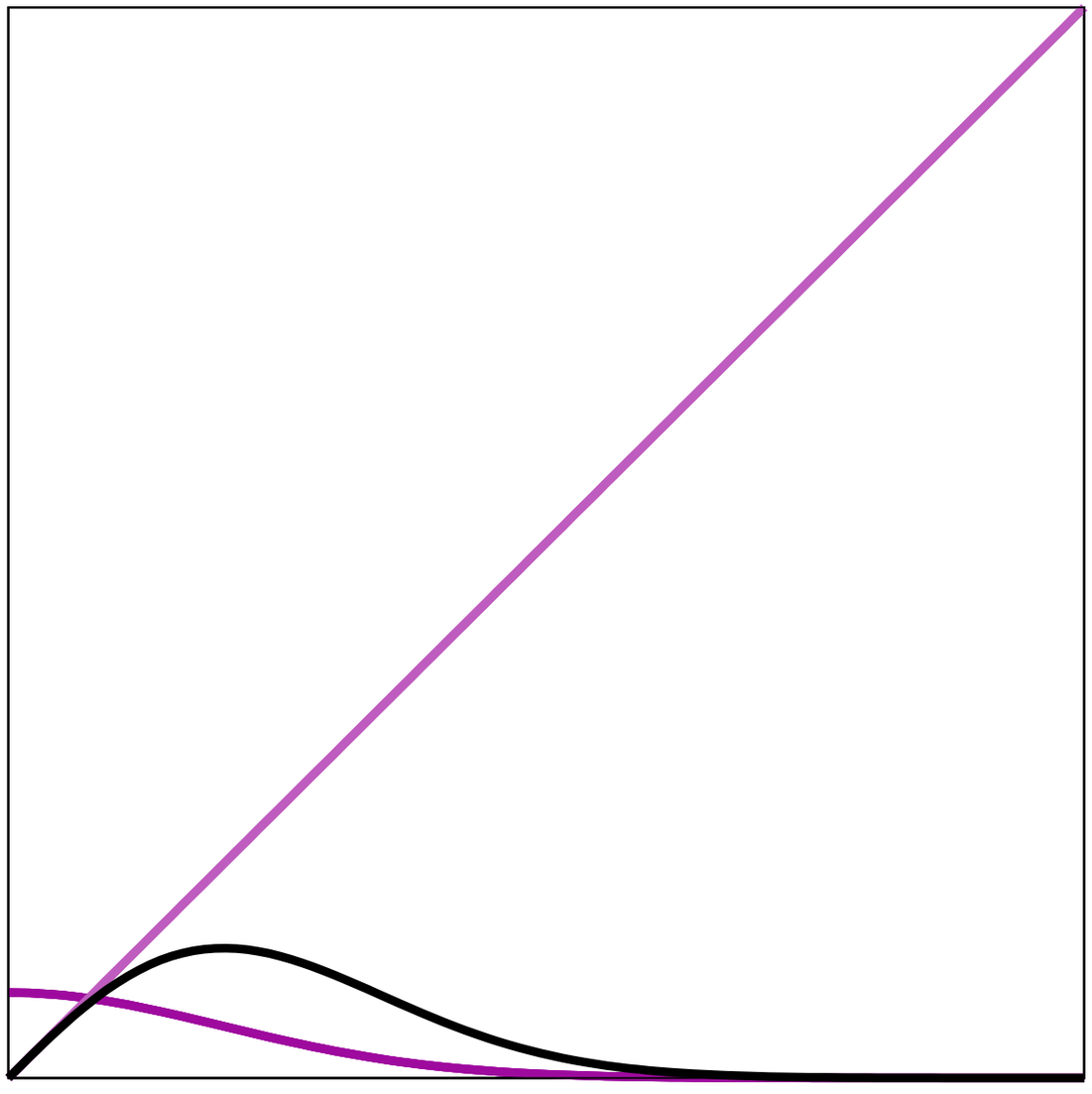
r

And mass can be very far away from density



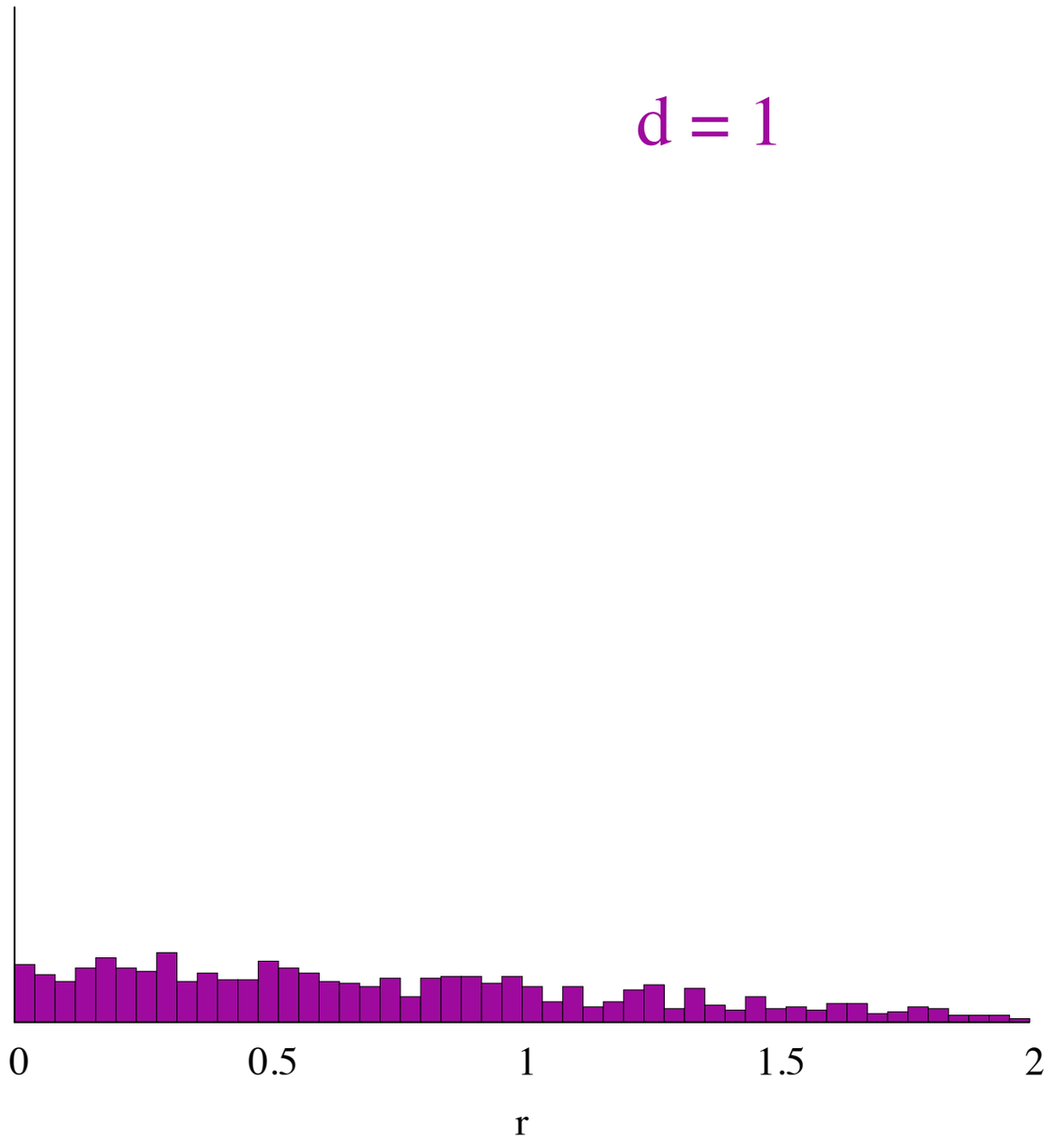
r

And mass can be very far away from density

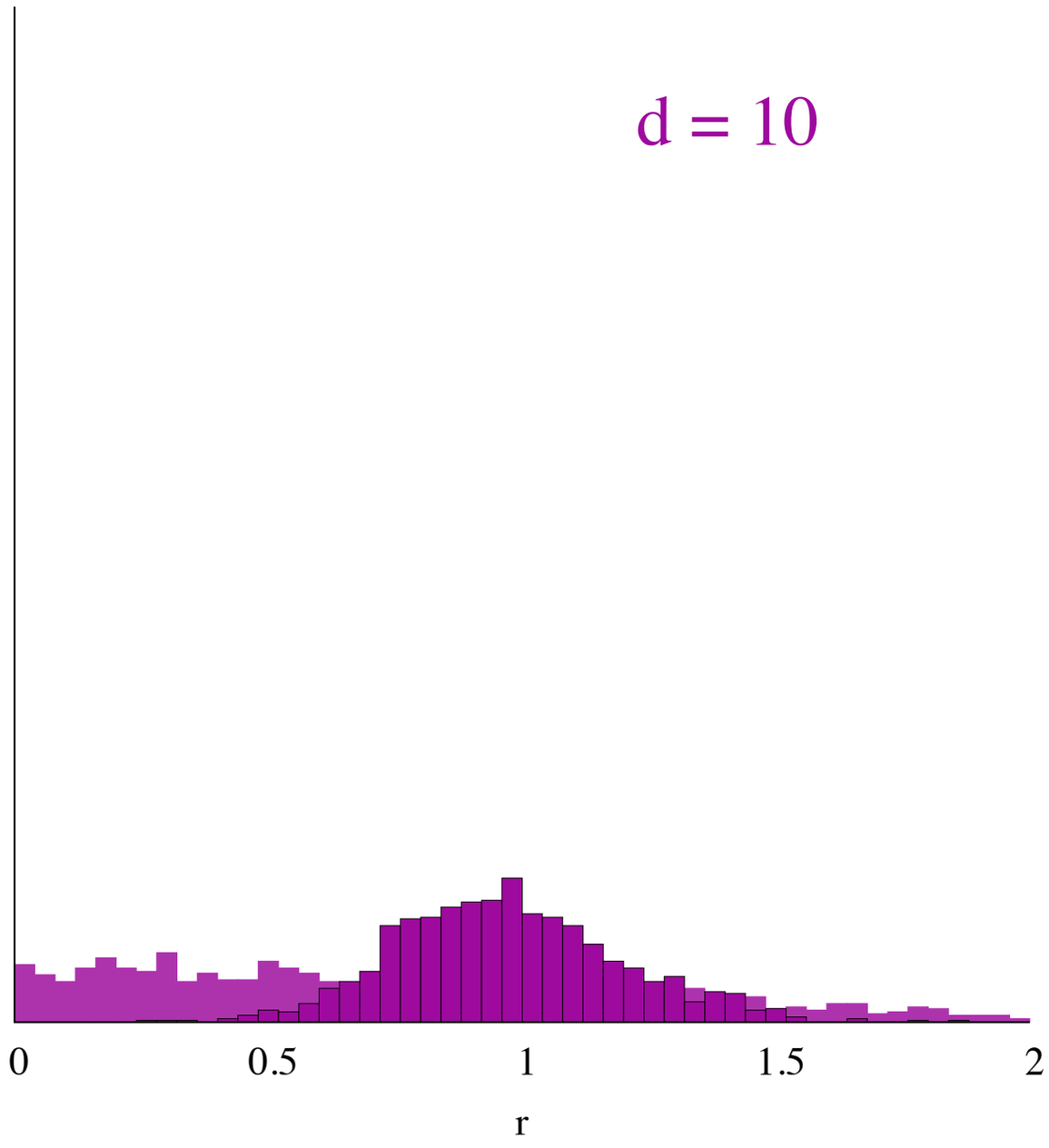


r

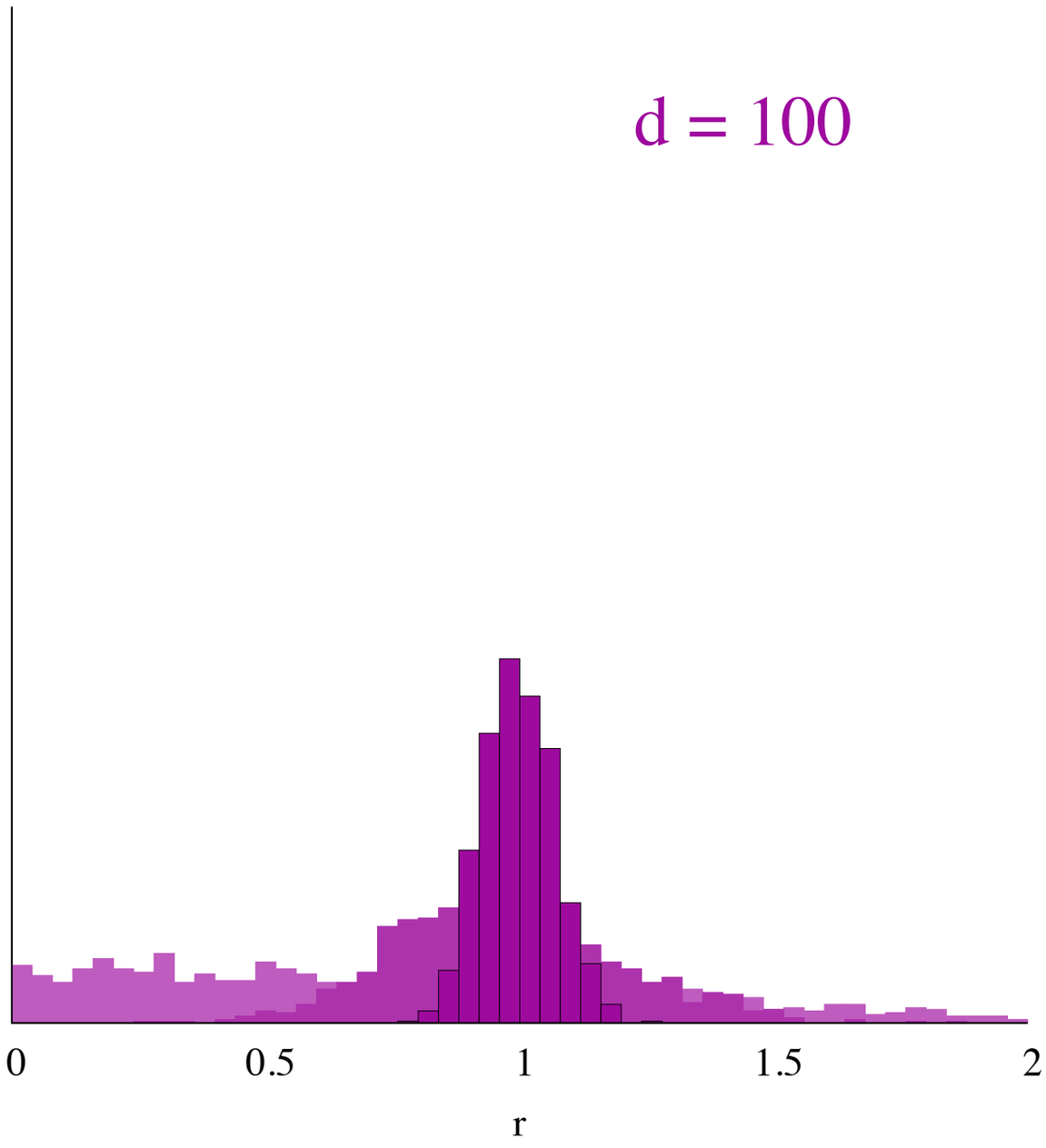
And mass can be very far away from density



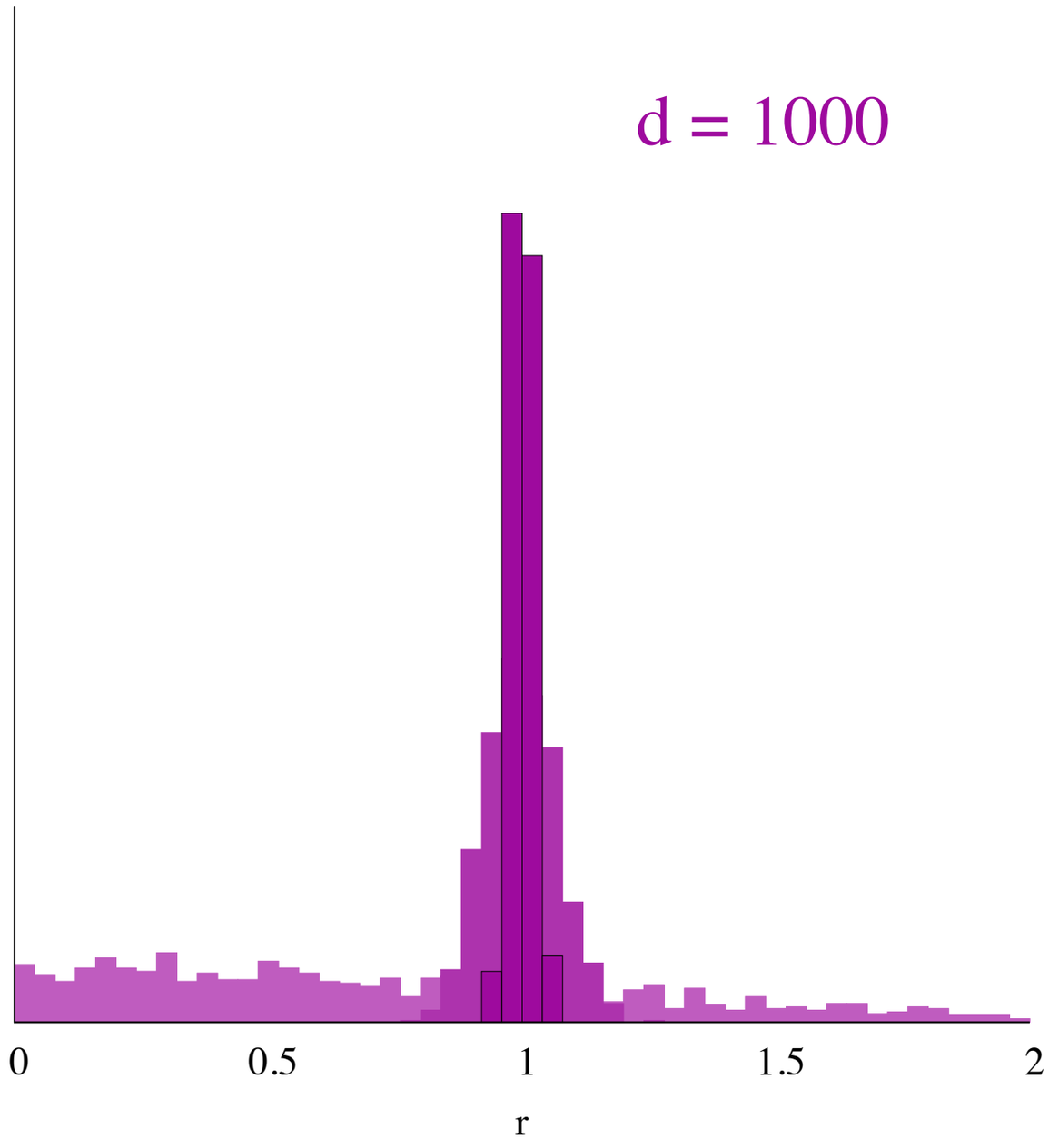
And mass can be very far away from density



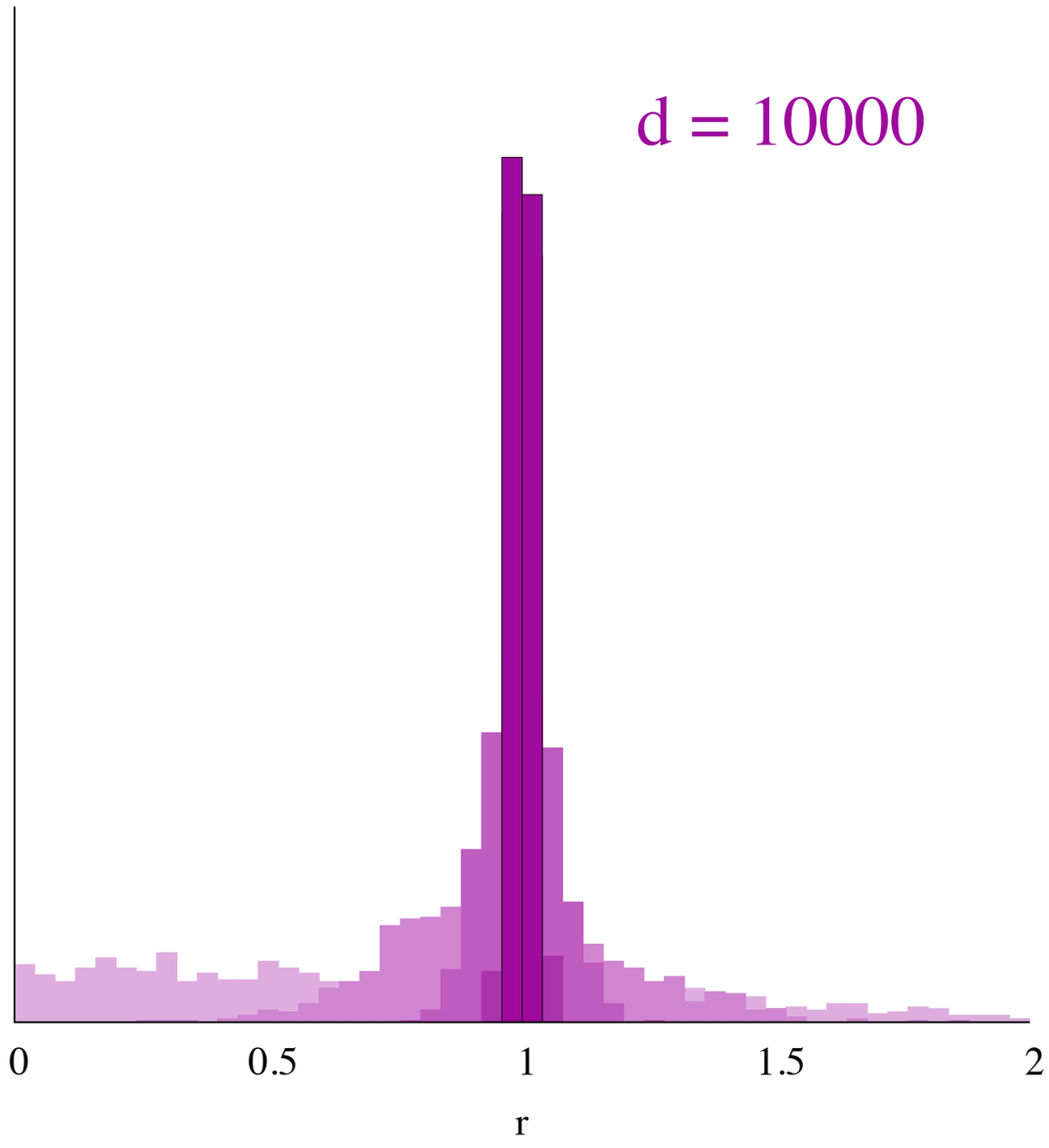
And mass can be very far away from density



And mass can be very far away from density



And mass can be very far away from density



Well-posed queries can be answered
by integrating the posterior

$$\mathbb{E}[f(\theta)] = \int \mathrm{d}\theta \, f(\theta) \, \pi(\theta|\mathcal{D})$$

Well-posed queries can be answered
by integrating the posterior

$$\mathbb{E}[f(\theta)] = \int \mathrm{d}\theta \, f(\theta) \, \pi(\theta|\mathcal{D})$$

$$\pi(\theta_2, \ldots, \theta_n|\mathcal{D}) = \int \mathrm{d}\theta_1 \, \pi(\theta|\mathcal{D})$$

Well-posed queries can be answered
by integrating the posterior

$$\mathbb{E}[f(\theta)] = \int \mathrm{d}\theta \, f(\theta) \, \pi(\theta|\mathcal{D})$$

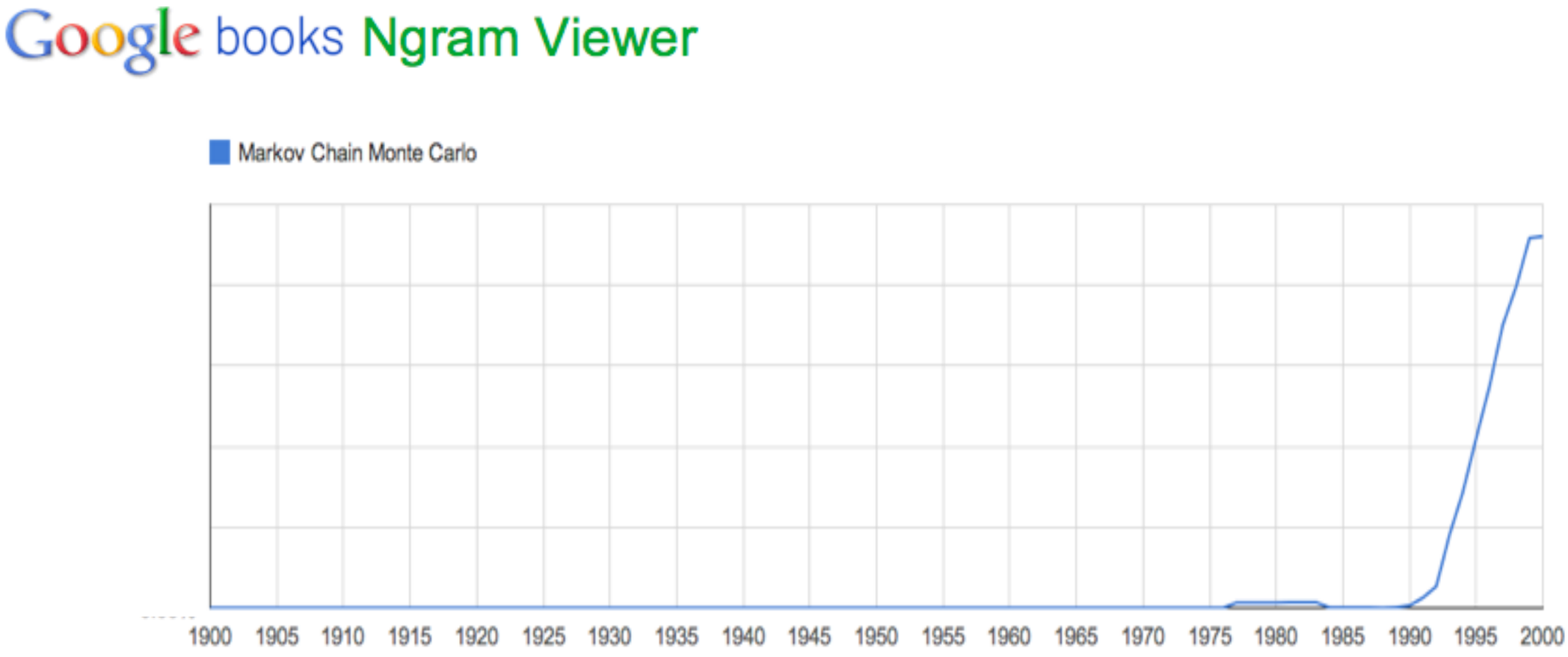
$$\pi(\theta_2, \ldots, \theta_n|\mathcal{D}) = \int \mathrm{d}\theta_1 \, \pi(\theta|\mathcal{D})$$

$$\pi(\theta_2, \ldots, \theta_n|\theta_1, \mathcal{D}) = \frac{\pi(\theta|\mathcal{D})}{\int \mathrm{d}\theta_1 \, \pi(\theta|\mathcal{D})}$$

Building a posterior is straightforward:
Bayesian inference is hard because integration is hard

$$\mathbb{E}[f(\theta)] = \int \mathrm{d}\theta \, f(\theta) \, \pi(\theta|\mathcal{D})$$

The key to efficient integration is Markov Chain Monte Carlo



Here the posterior is represented with a set of samples from which expectations can be efficiently computed

$$p(\theta|\mathcal{D}) \rightarrow \{\theta_1, \ldots, \theta_n\}$$

Here the posterior is represented with a set of samples from which expectations can be efficiently computed

$$p(\theta|\mathcal{D}) \rightarrow \{\theta_1, \ldots, \theta_n\}$$

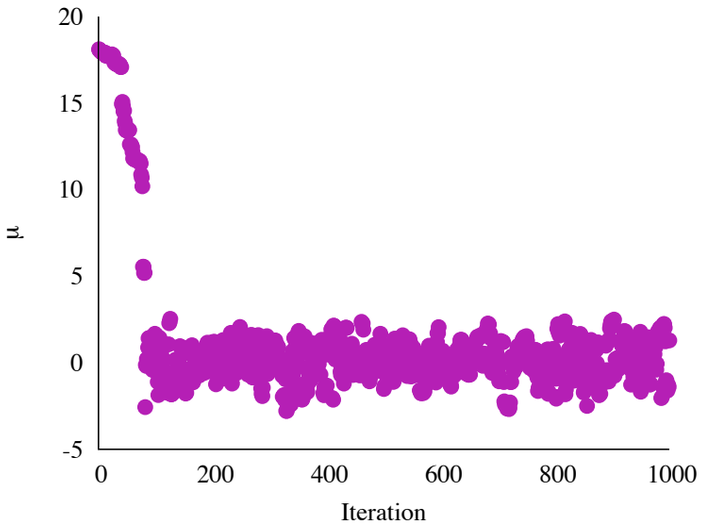
$$\mathbb{E}[f(\theta)] \approx \frac{1}{N} \sum_{n=1}^{N} f(\theta_i)$$

We generate those samples with a Markov chain,
typically defined by its transition kernel

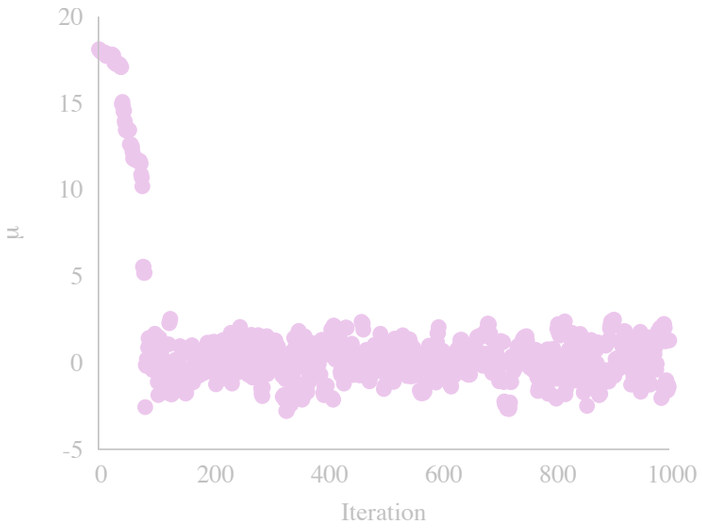
$$\pi'(\theta) = \int \mathrm{d}\theta\, T(\theta, \theta')\, \pi(\theta')$$

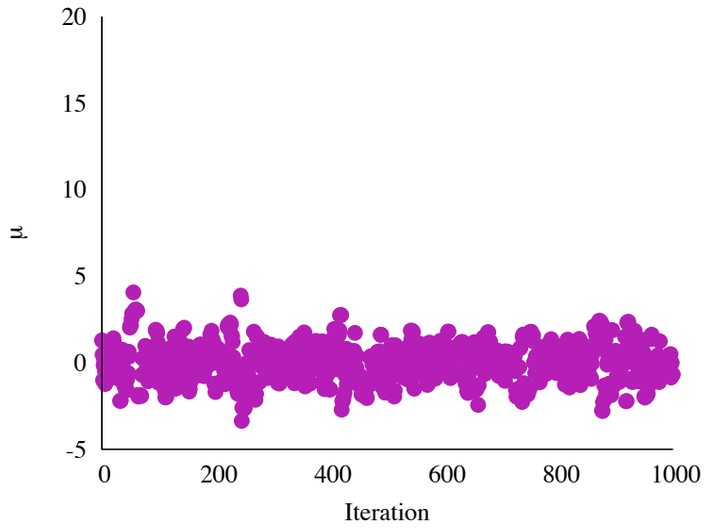
In practice, MCMC proceeds in three stages

In practice, MCMC proceeds in three stages

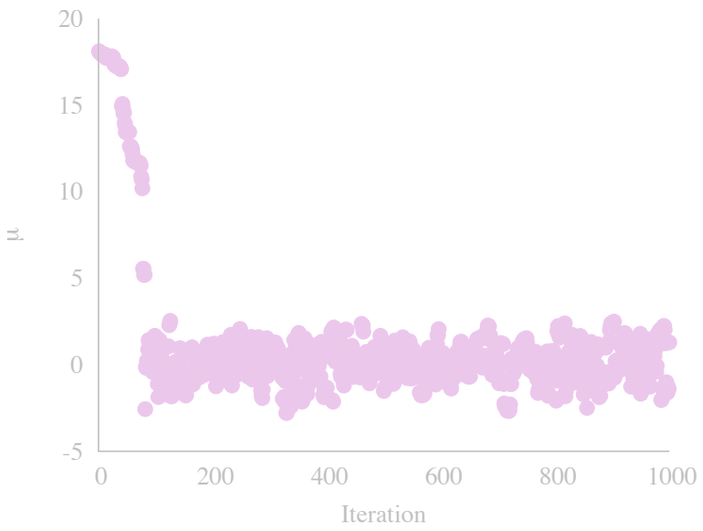


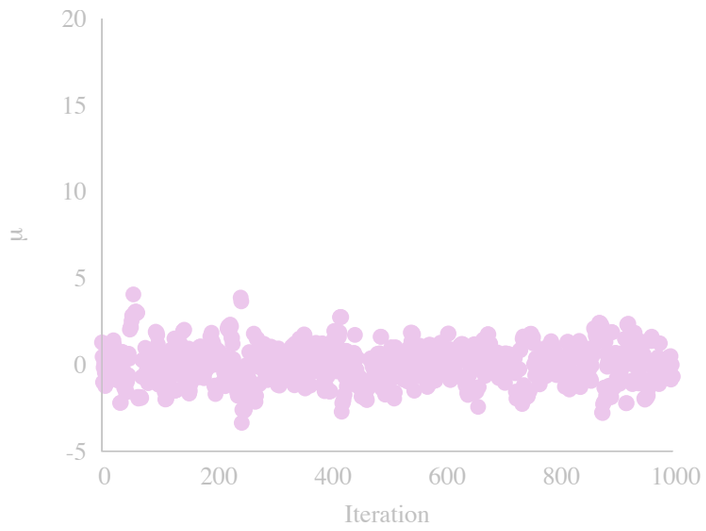
In practice, MCMC proceeds in three stages

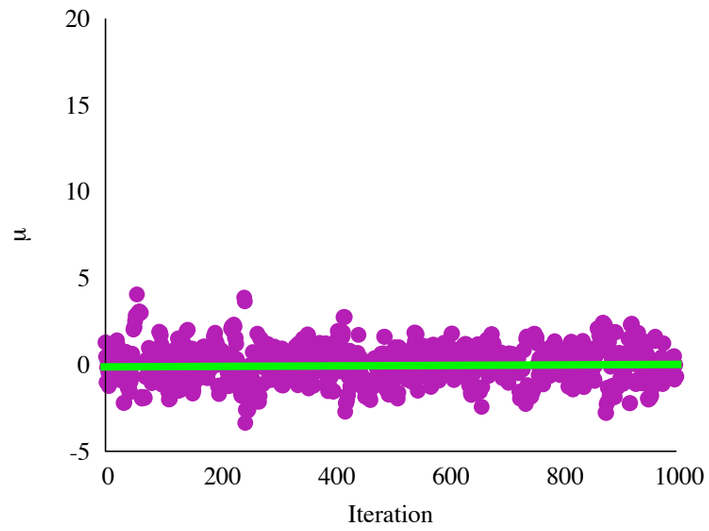




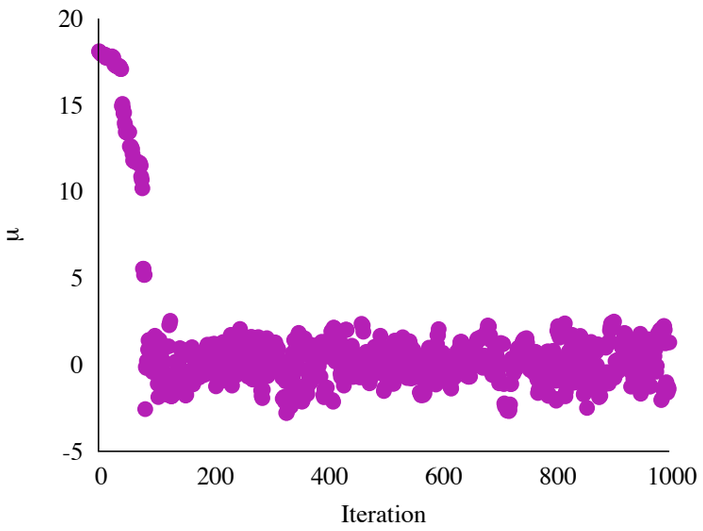
In practice, MCMC proceeds in three stages

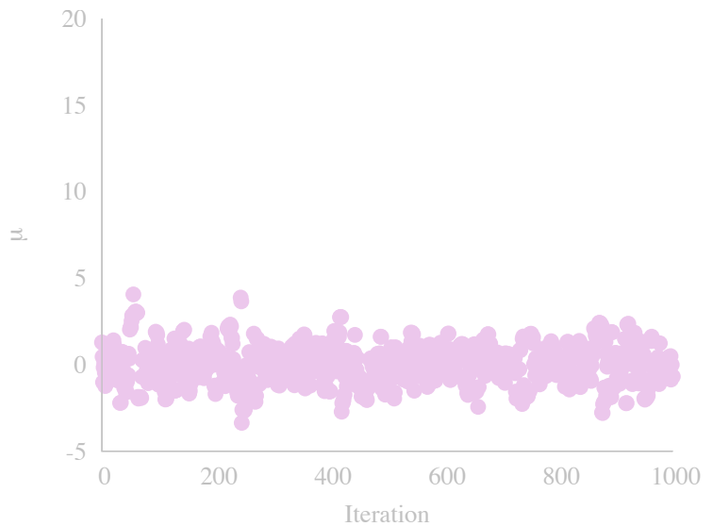


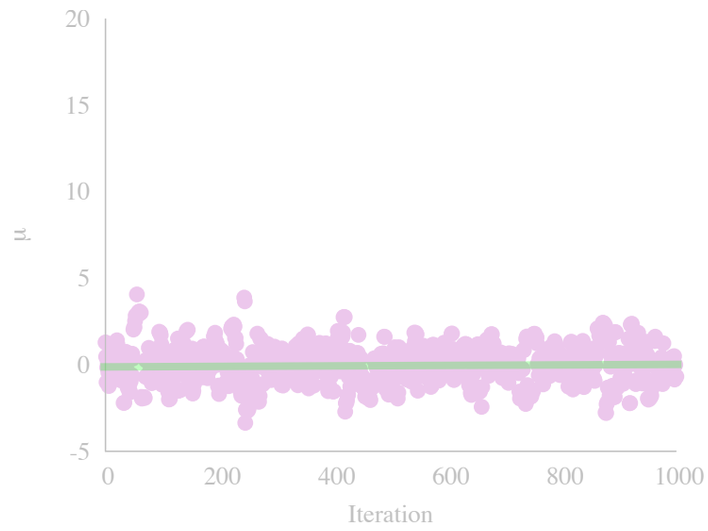




Warmup


μ
20
15
10
5
0
-5
0
200
400
600
800
1000
Iteration


μ
20
15
10
5
0
-5
0
200
400
600
800
1000
Iteration


μ
20
15
10
5
0
-5
0
200
400
600
800
1000
Iteration

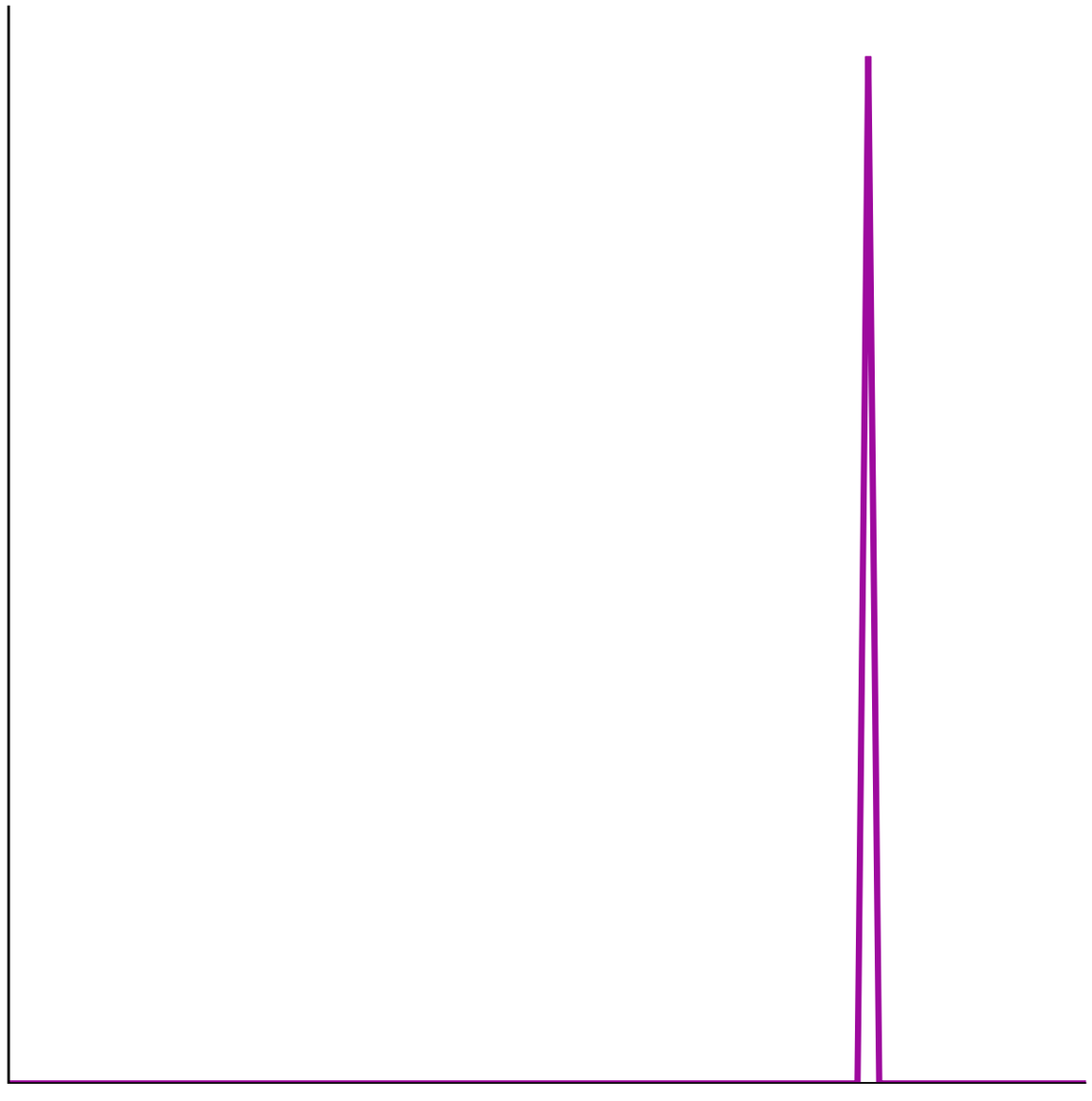
A Markov chain will preserve its stationary distribution...

$$\pi(\theta) = \int \mathrm{d}\theta \, T(\theta, \theta') \, \pi(\theta')$$

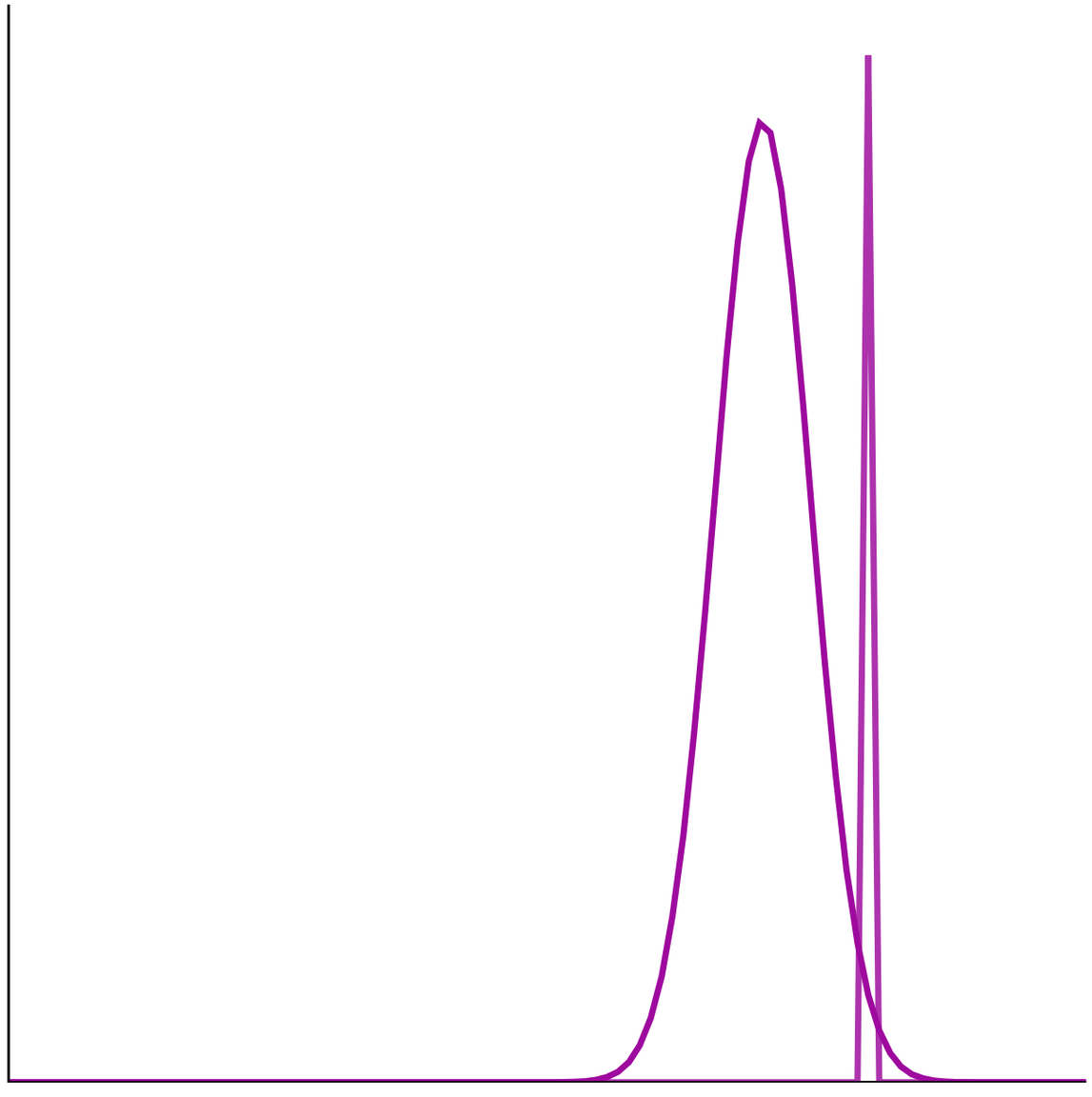
But in order to reach the stationary distribution
we have to "filter" our initial distribution

$$\pi(\theta) = \int \mathrm{d}\theta\, T(\theta, \theta') \ldots \int \mathrm{d}\theta'''\, T(\theta''', \theta'''')\, \pi(\theta'''')$$

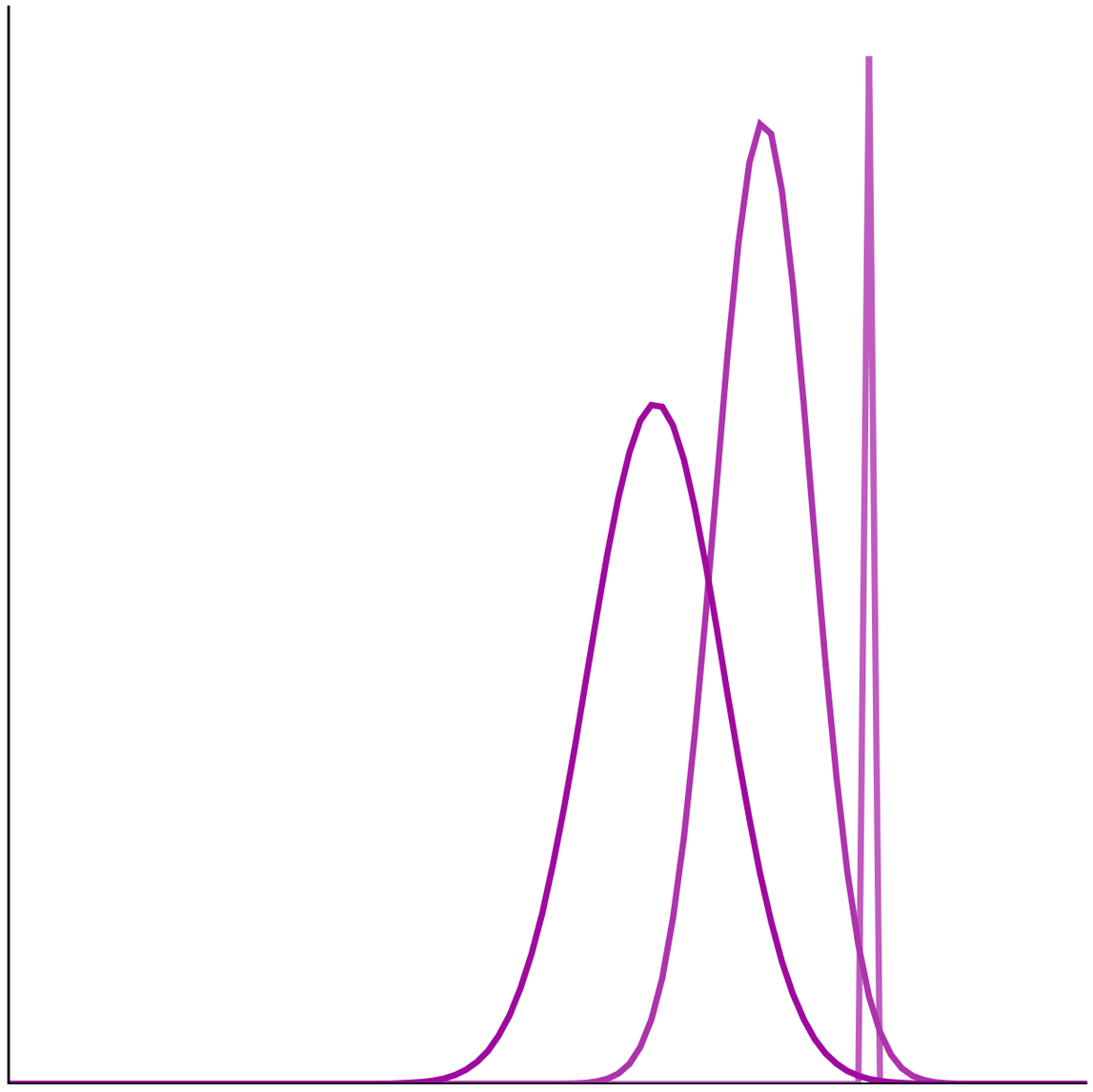
But in order to reach the stationary distribution
we have to “filter” our initial distribution



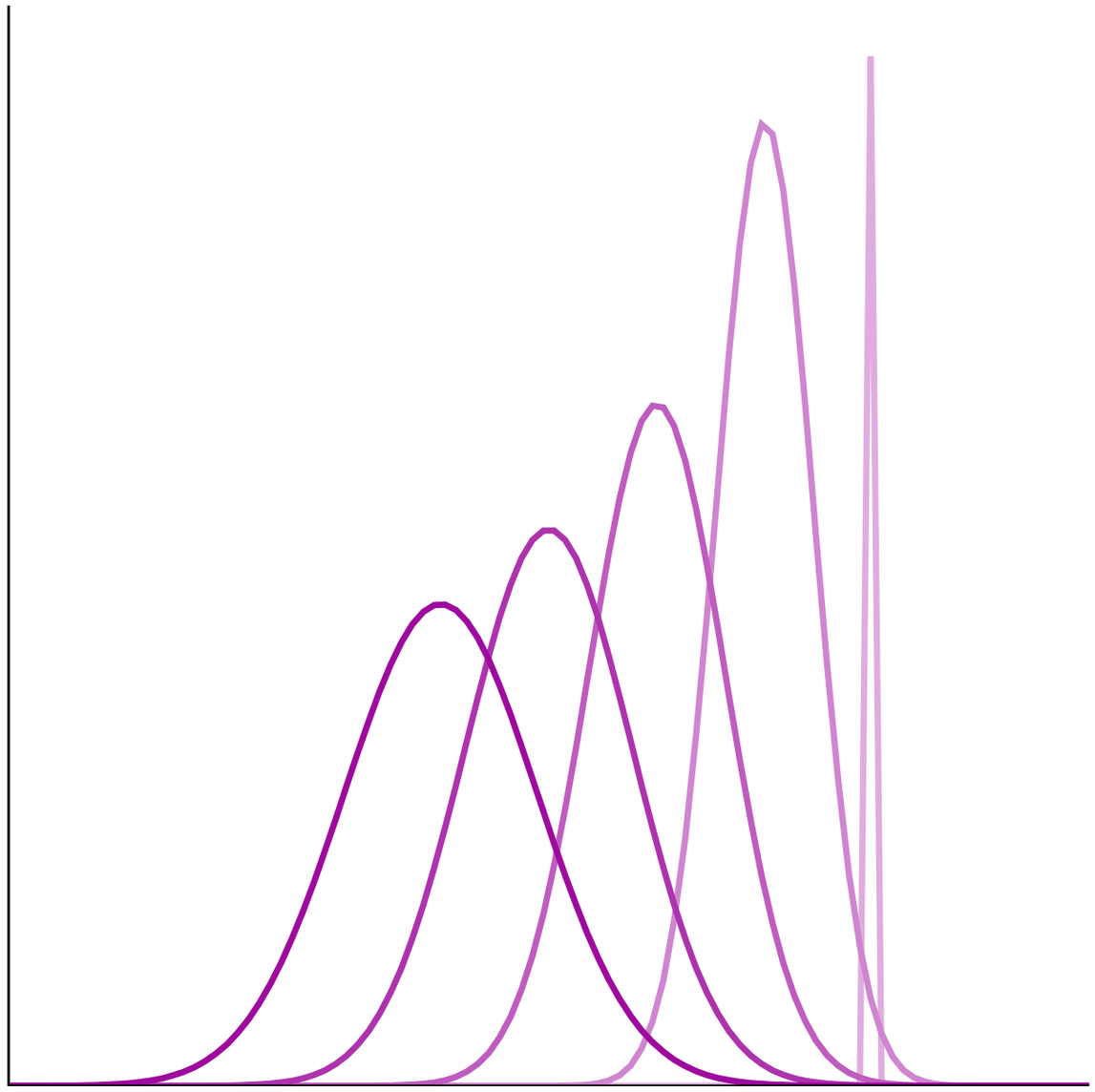
But in order to reach the stationary distribution
we have to “filter” our initial distribution



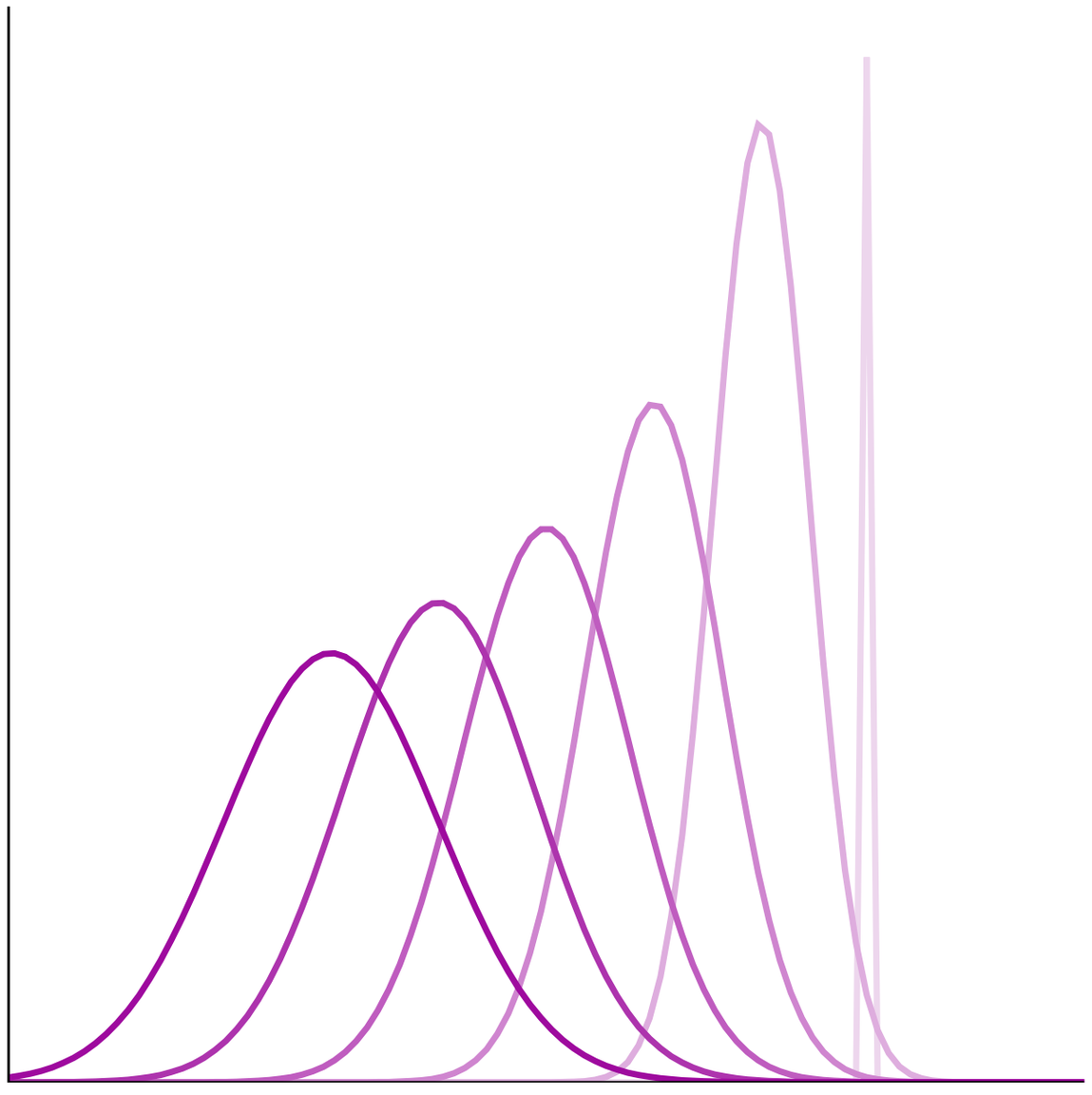
But in order to reach the stationary distribution
we have to “filter” our initial distribution



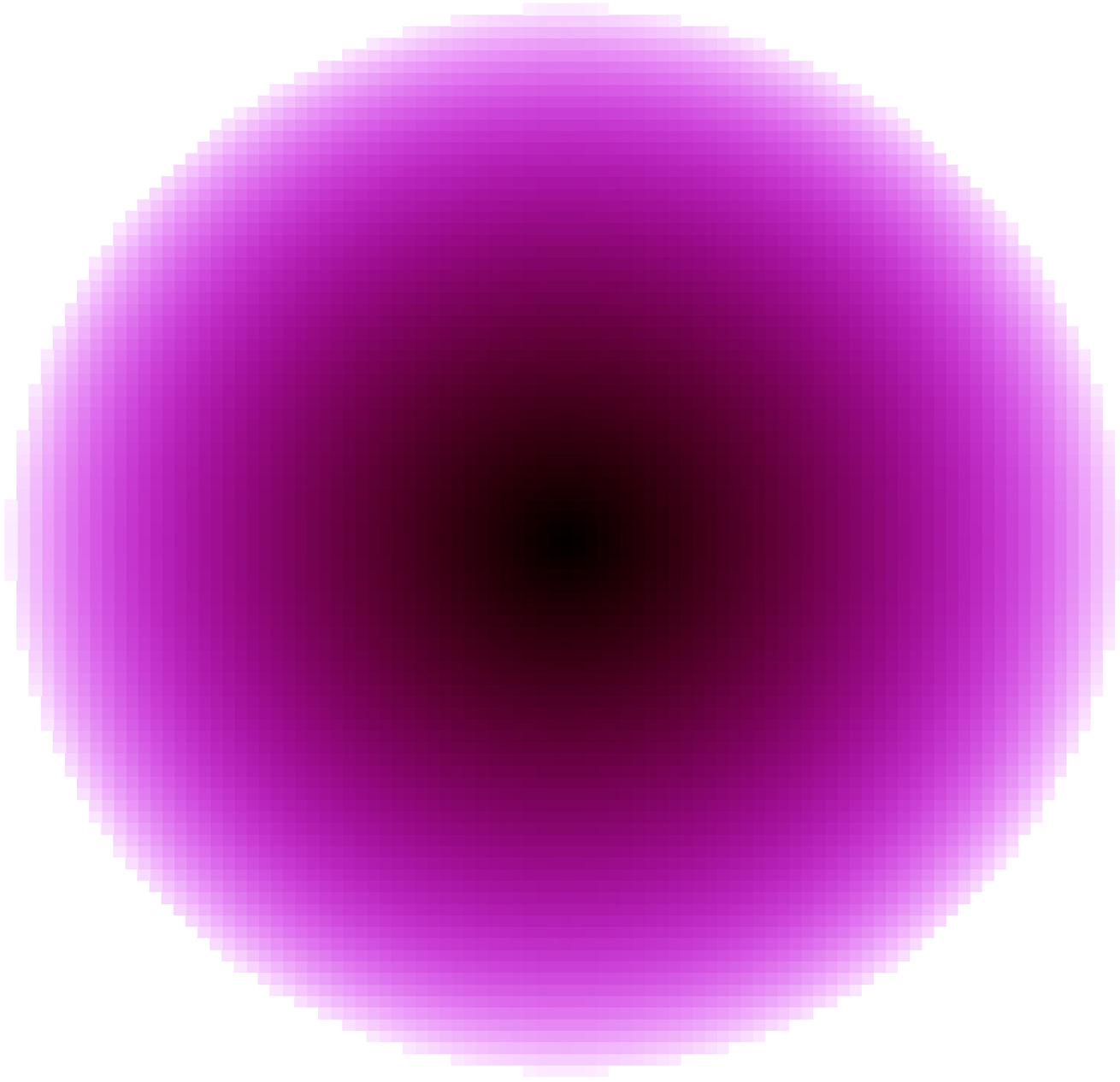
But in order to reach the stationary distribution
we have to “filter” our initial distribution



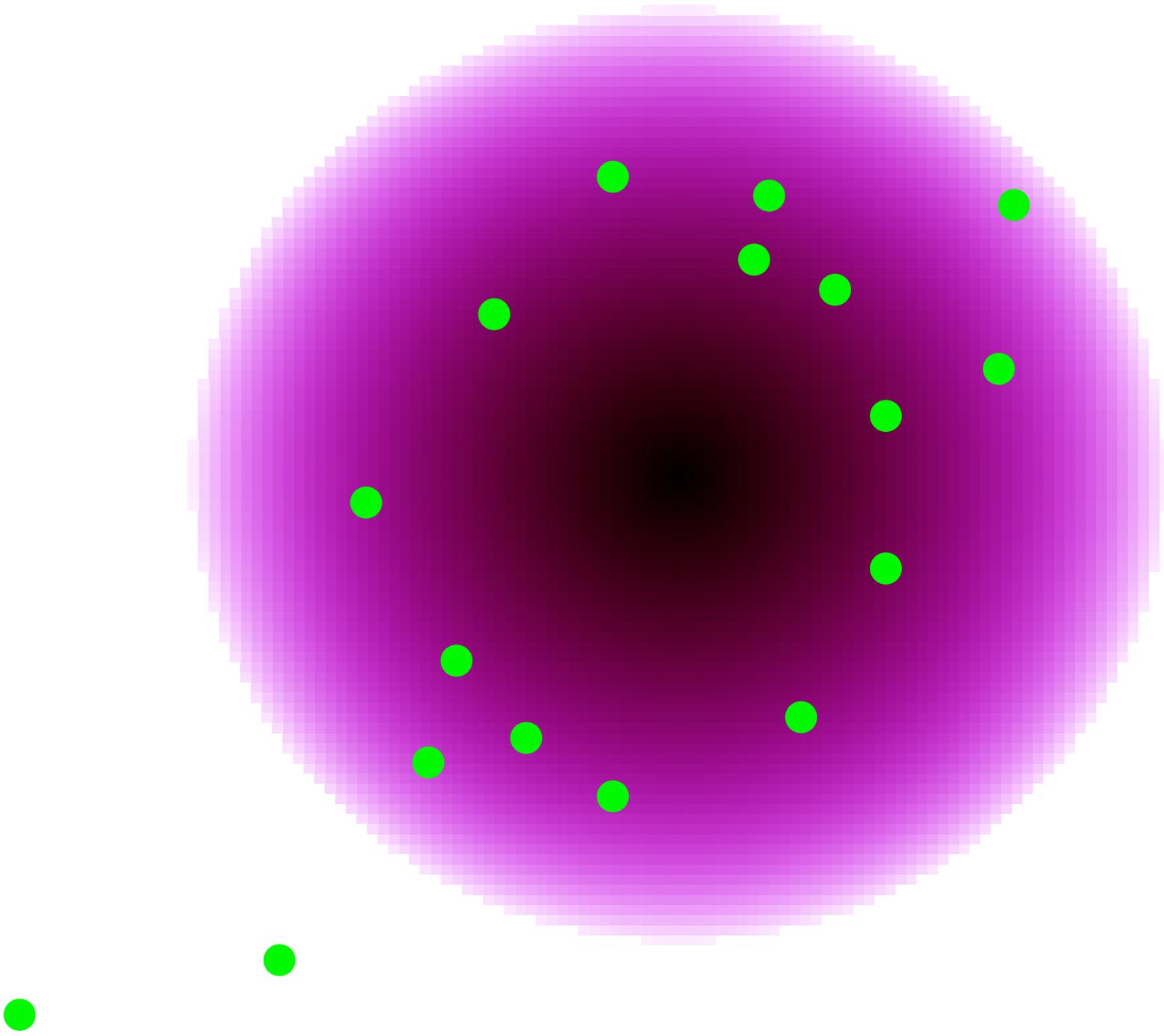
But in order to reach the stationary distribution
we have to “filter” our initial distribution



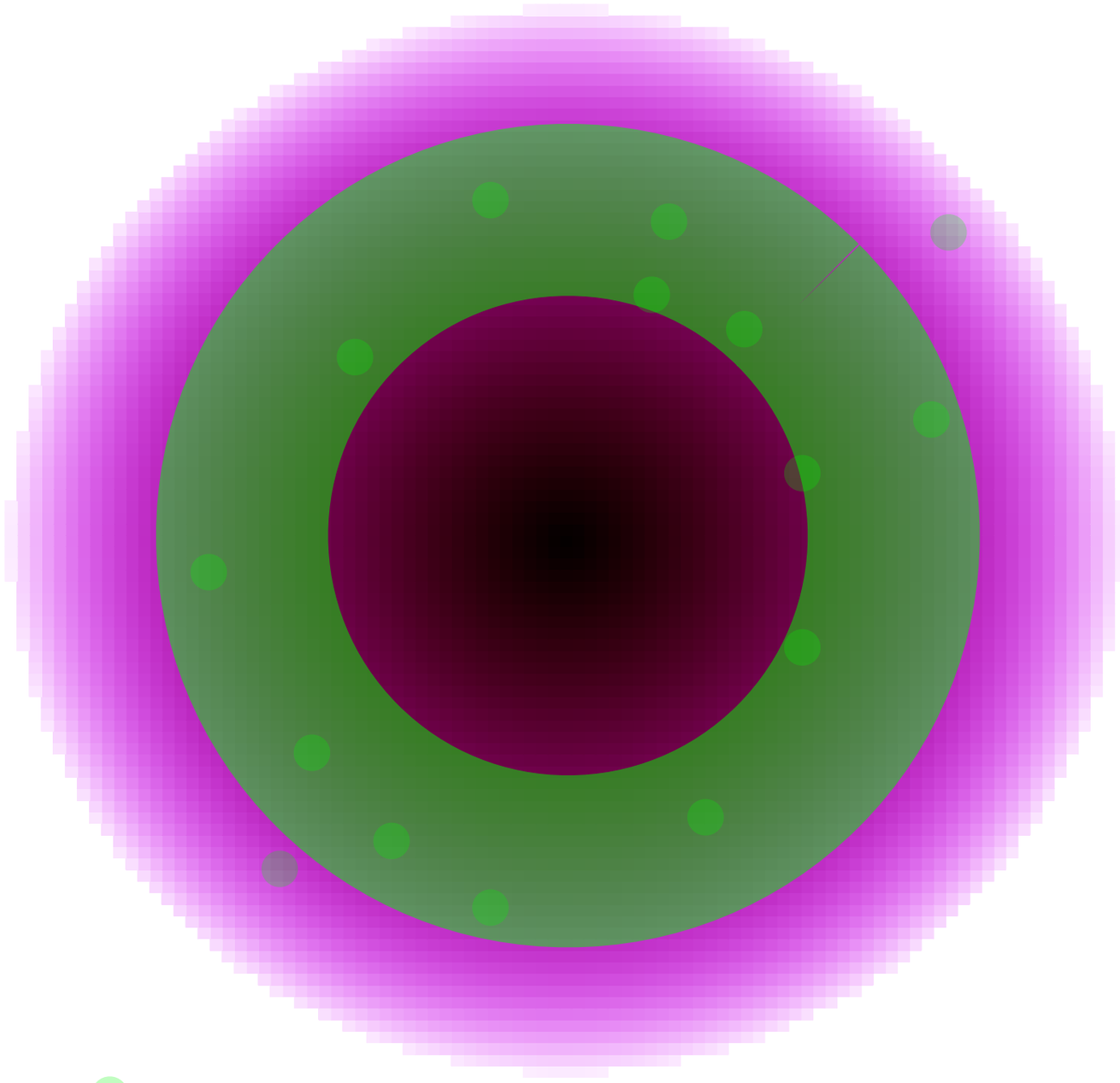
In practice it's easier to consider the state of the Markov chain relative to *the typical set*



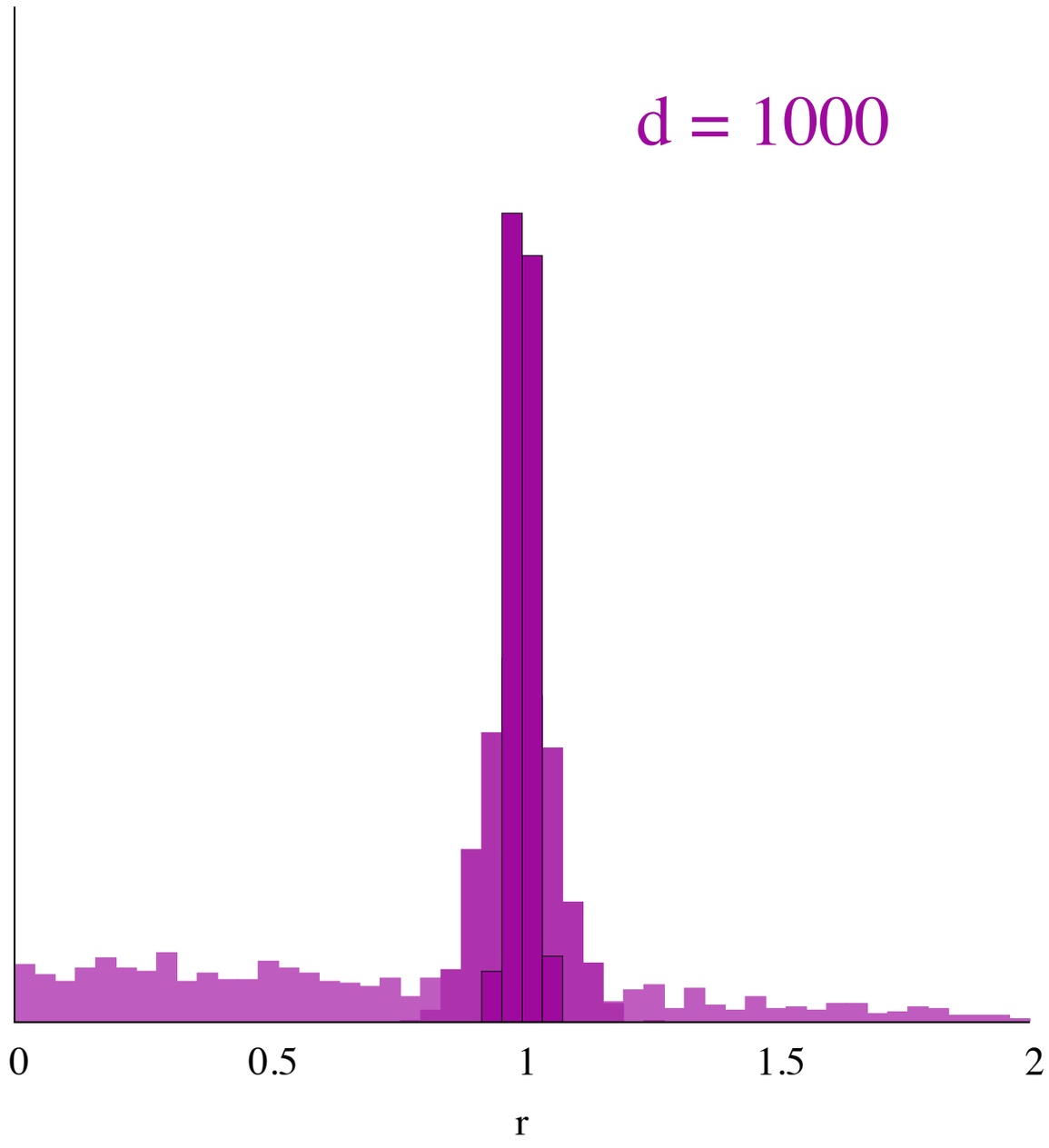
In practice it's easier to consider the state of the Markov chain relative to *the typical set*



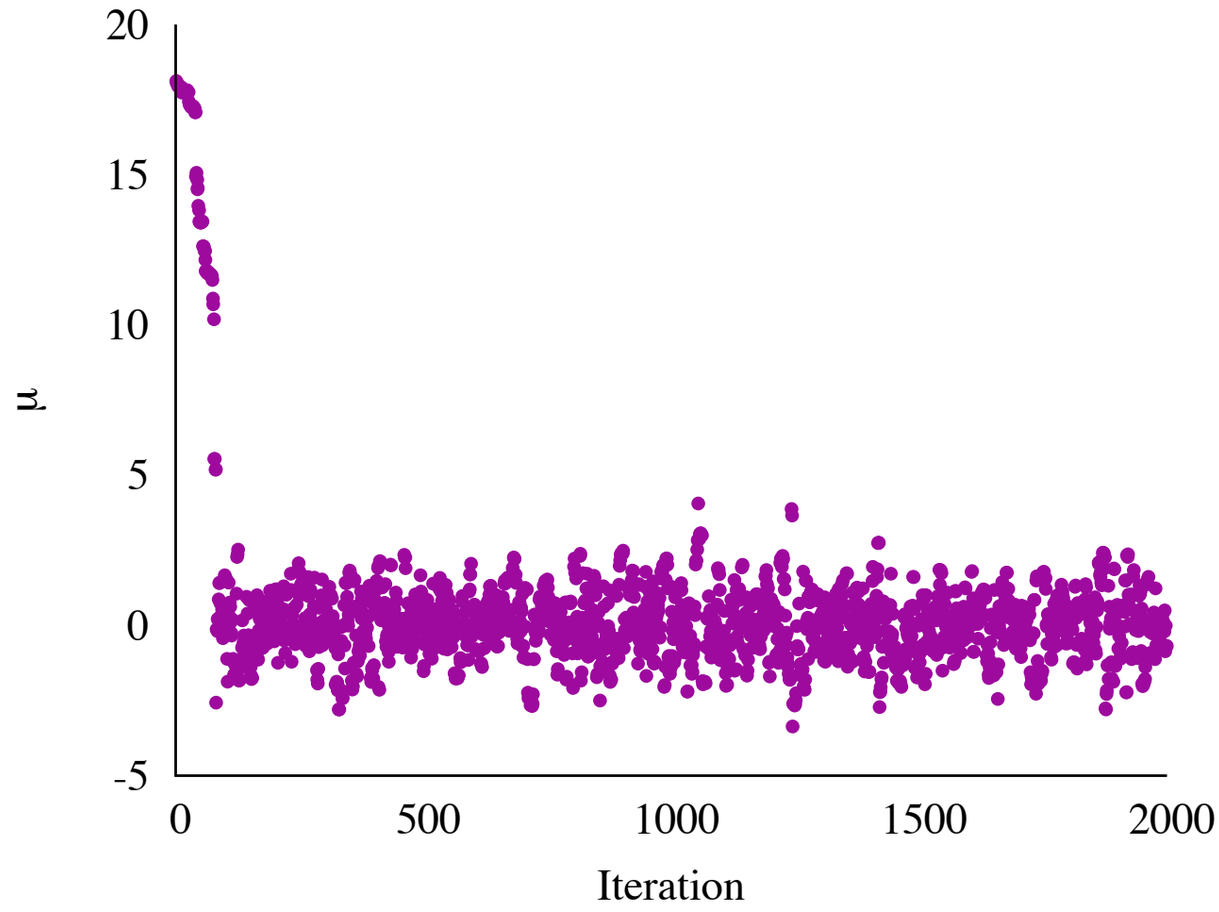
In practice it's easier to consider the state of the Markov chain relative to *the typical set*



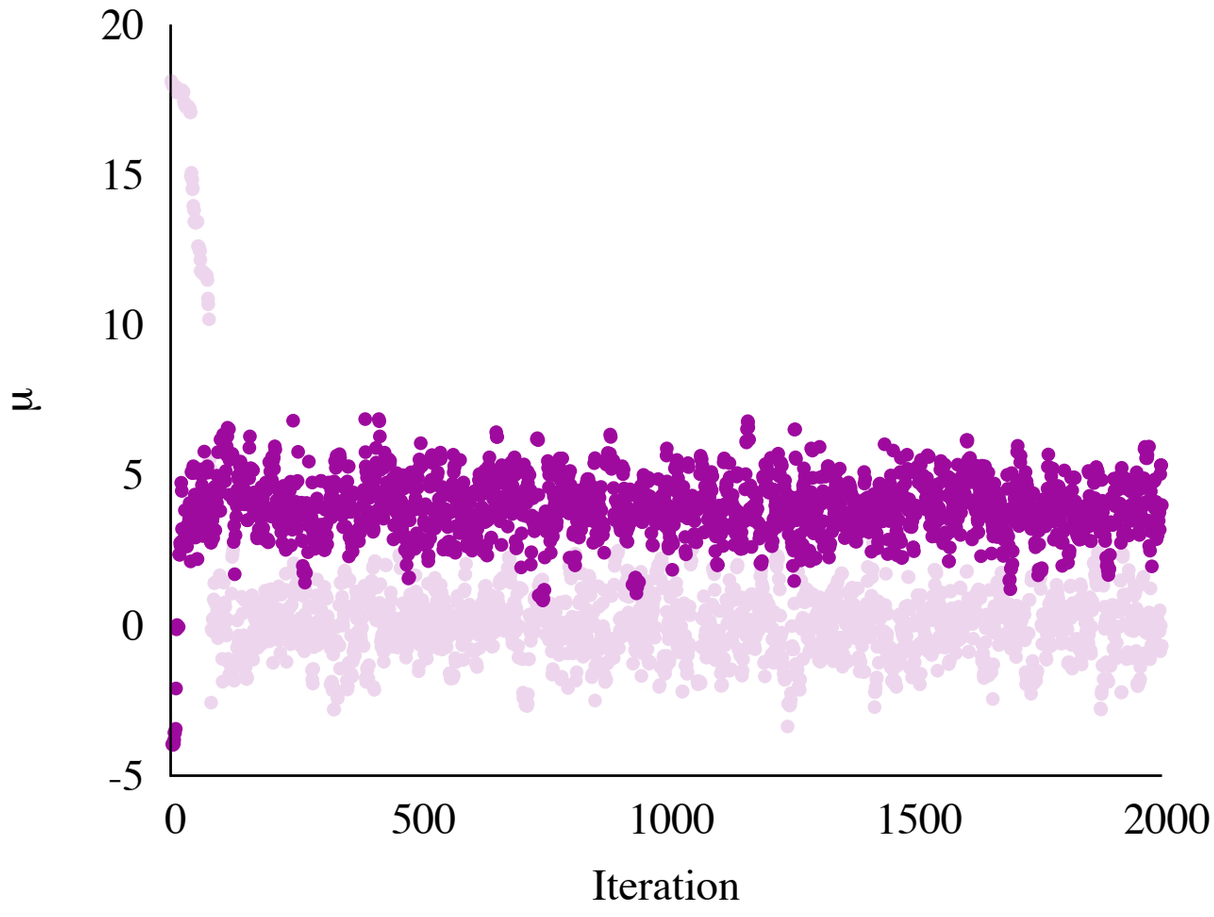
In high dimensions the typical set is often vary far from any MAP



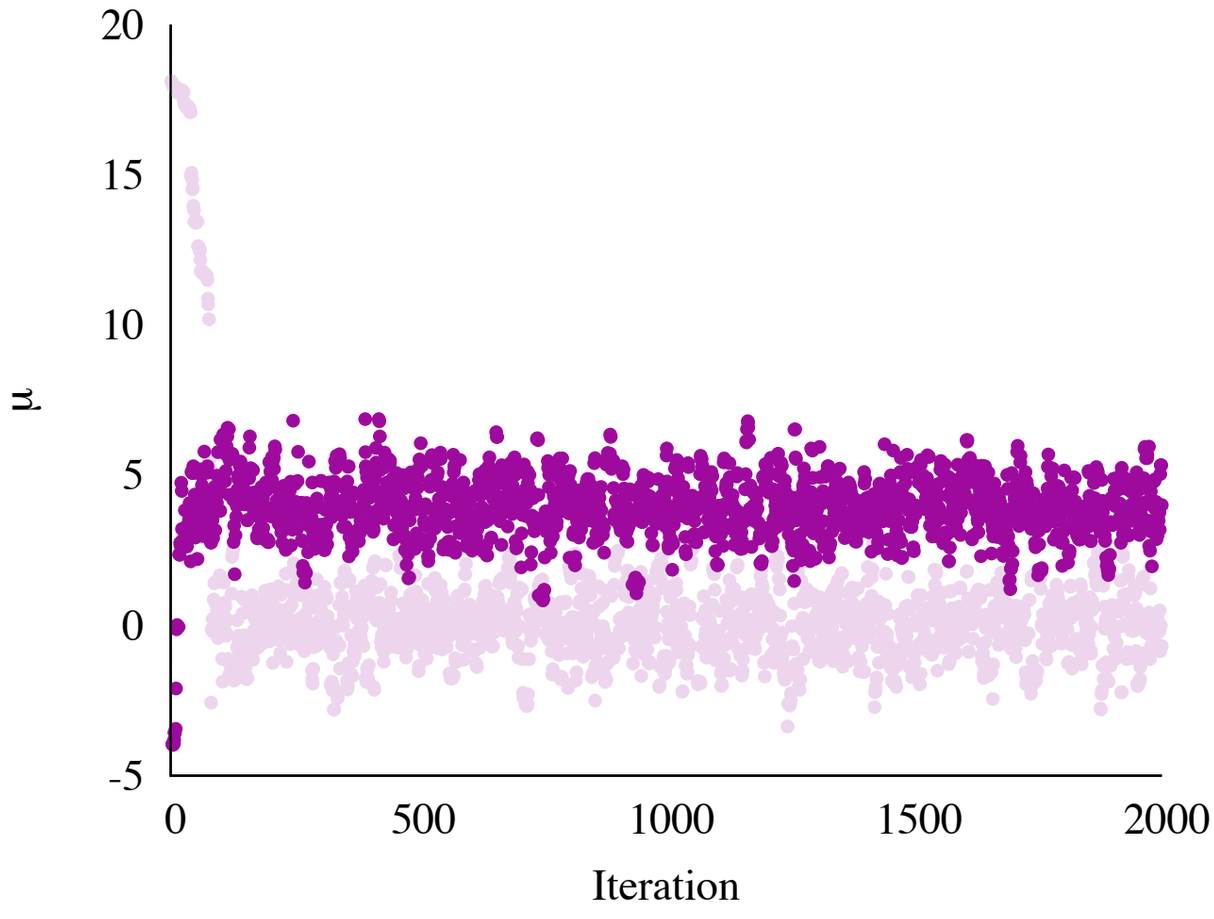
How do we know that we've converged?
Visual diagnostics are appealing...



But they can be misleading!



The best strategy is to run multiple chains from diffuse initializations and compare



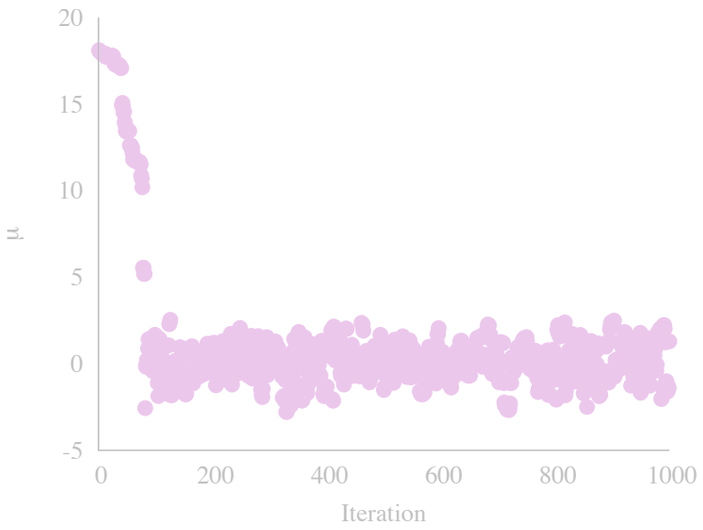
$$\hat{R} = \sqrt{\frac{N-1}{N} + \frac{1}{N}\frac{B}{W}}$$

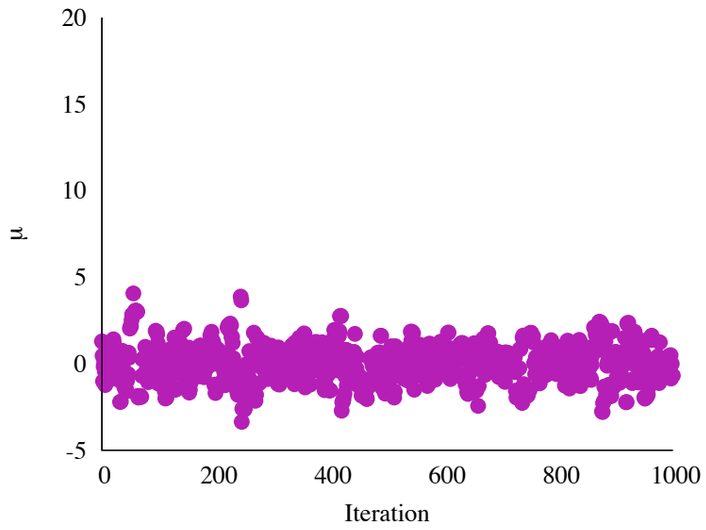
We can also learn sampler parameters during warmup, provided we've already converged

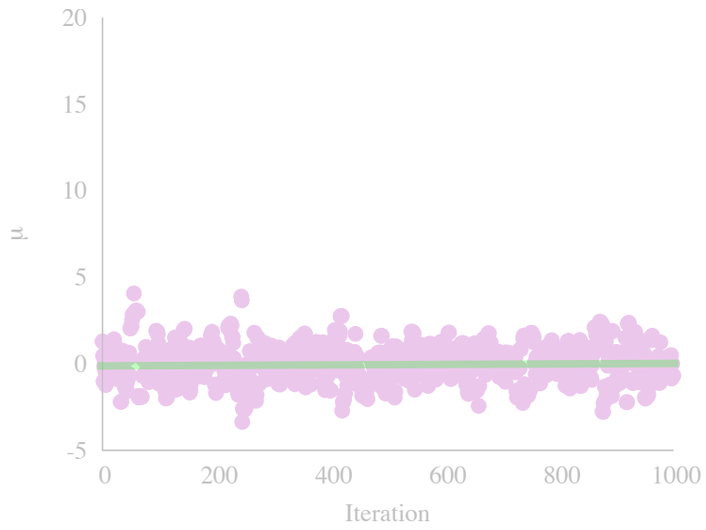
$$q \to q + \epsilon M^{-1} p$$

$$p \to p - \epsilon \frac{\partial V}{\partial q}$$

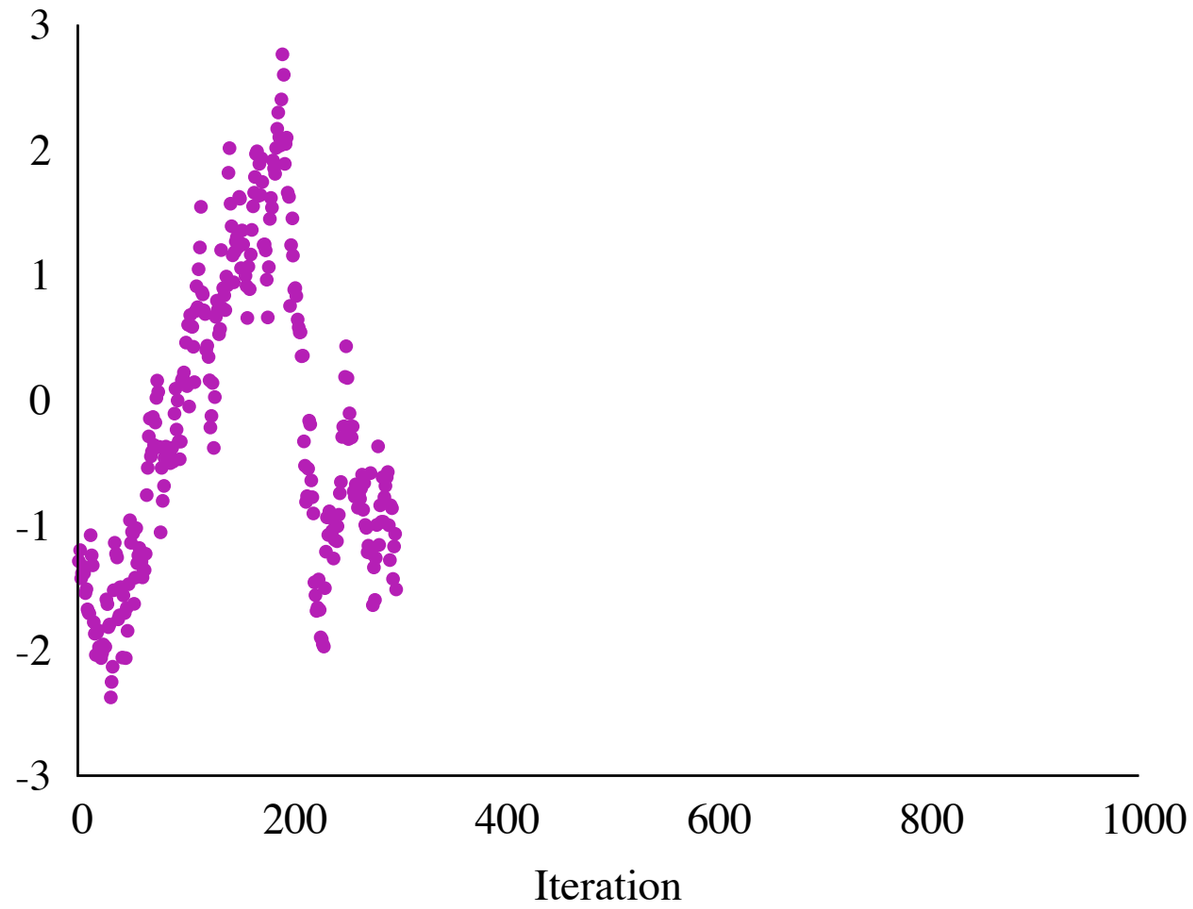
Sampling


20
15
10
5
0
-5
0
200
400
600
800
1000
Iteration

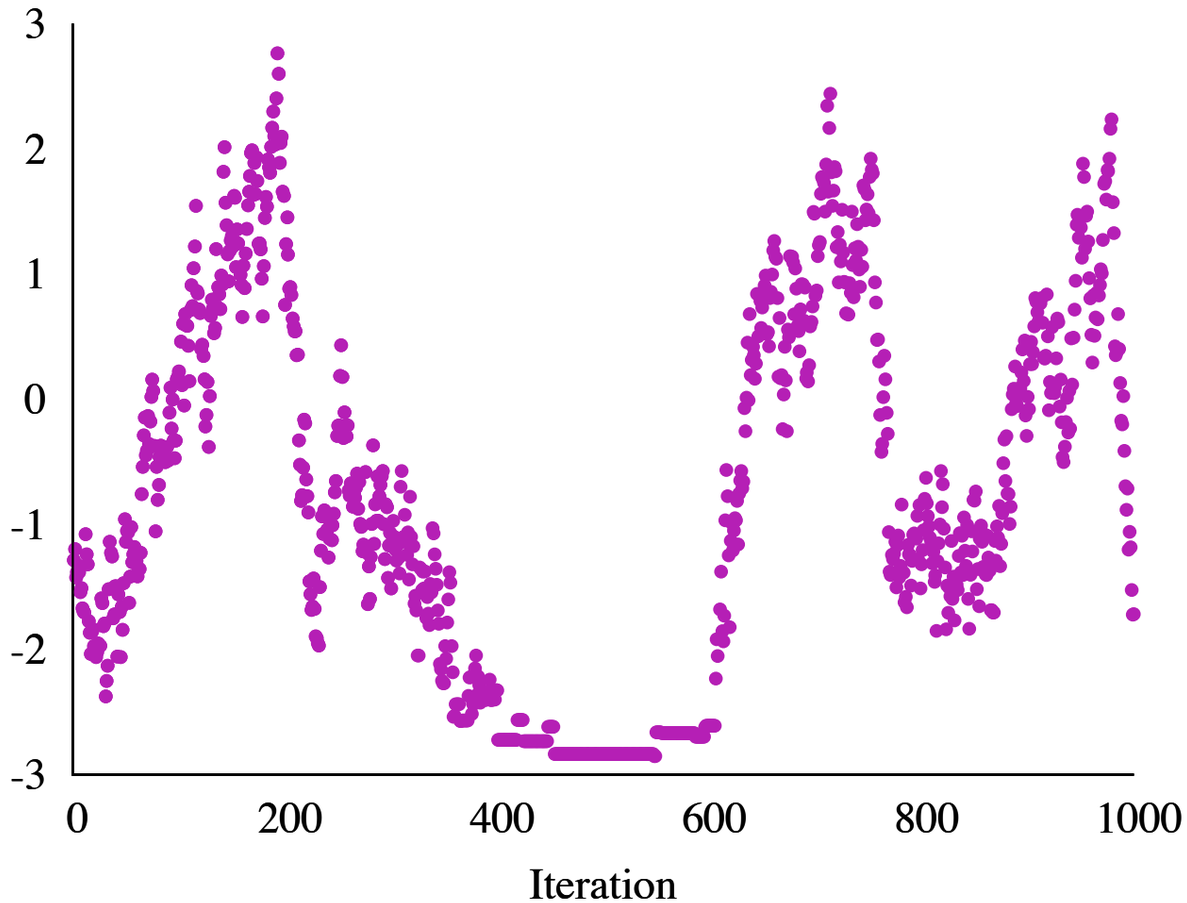

20
15
10
5
0
-5
0
200
400
600
800
1000
Iteration


20
15
10
5
0
-5
0
200
400
600
800
1000
Iteration

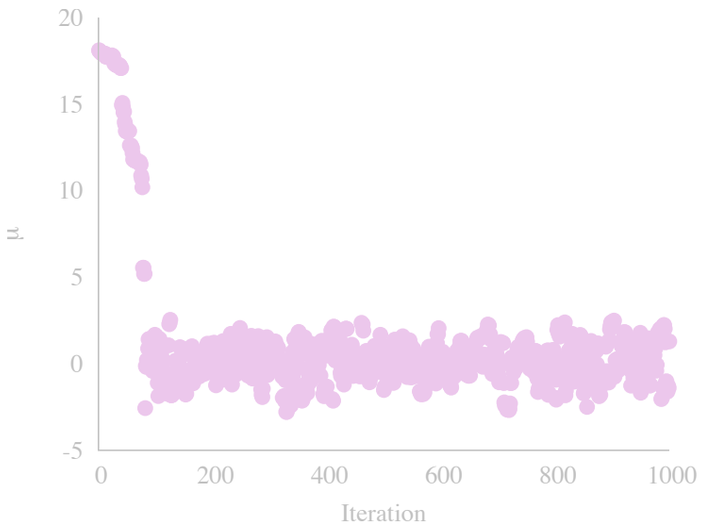
Sometimes chains get “stuck”

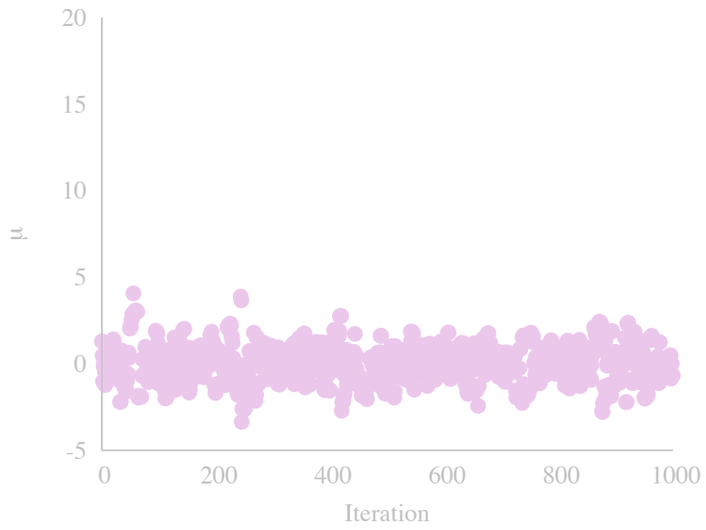


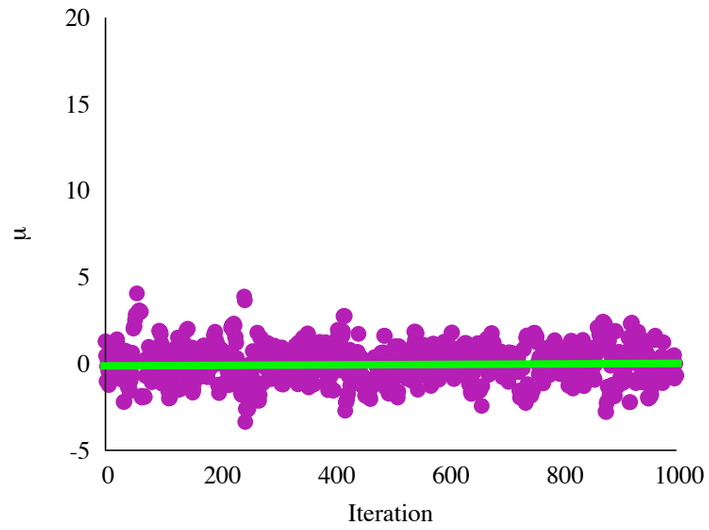
Sometimes chains get “stuck”



Analysis


20
15
10
5
0
-5
μ
0
200
400
600
800
1000
Iteration


20
15
10
5
0
-5
μ
0
200
400
600
800
1000
Iteration


20
15
10
5
0
-5
μ
0
200
400
600
800
1000
Iteration

It's time to calculate some expectations!

$$\hat{f} = \frac{1}{N} \sum_{n=1}^{N} f(\theta_i)$$

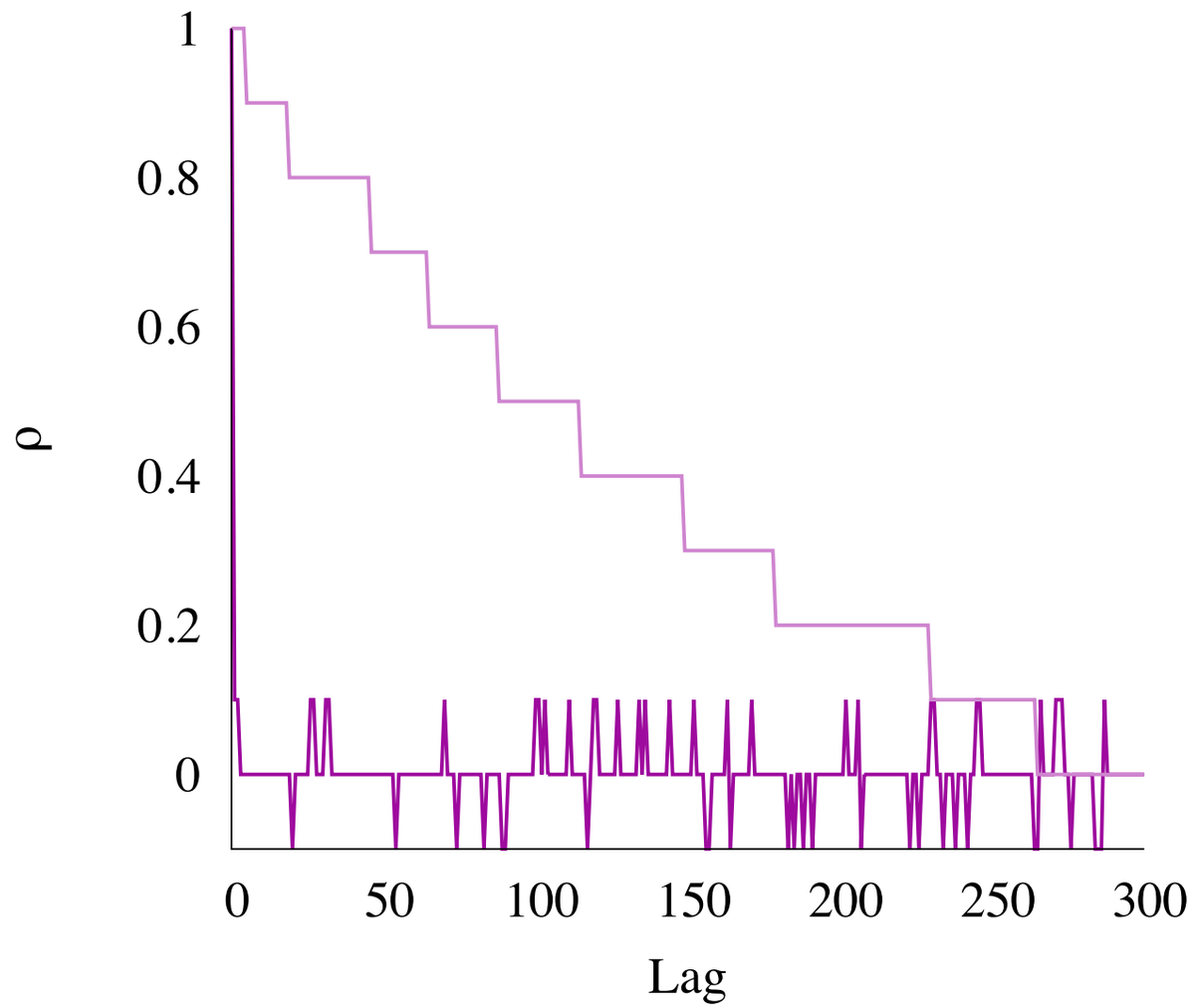
Under mild conditions, Monte Carlo expectations are distributed around the true value

$$\hat{f} \sim \mathcal{N}\left(\mathbb{E}[f]\,, \mathrm{MCSE}^2\right)$$

The *Monte Carlo Standard Error* measures the precision of the Monte Carlo estimate

$$\mathrm{MCSE}^2 = \frac{\mathrm{Var}(f)}{\mathrm{ESS}}$$

The *Effective Sample Size* is roughly the number of independent samples generated in the chain



$$\mathrm{ESS} = \frac{N}{1 + 2\sum_{n=1}^{N} \rho_n}$$

Careful inspection of Monte Carlo estimates is always a good idea

```
Inference for Stan model: example_model
1 chains: each with iter=(1000); warmup=(0); thin=(1); 1000 iterations saved.

Warmup took (0.0081) seconds, 0.0081 seconds total
Sampling took (0.012) seconds, 0.012 seconds total

                Mean     MCSE   StdDev    5%    50%        95%  N_Eff  N_Eff/s  R_hat
lp__           -0.53  3.3e-02  7.1e-01  -2.0  -0.25  -2.3e-03    460    36797   1.00
accept_stat__   0.85  6.7e-03  2.1e-01  0.36   0.95   1.0e+00   1000    80019   1.00
stepsize__       1.5  7.2e-15  5.1e-15   1.5    1.5   1.5e+00   0.50       40   1.00
treedepth__     0.48  1.8e-02  5.0e-01  0.00   0.00   1.0e+00    806    64458   1.00
mu               3.9  5.0e-02  1.0e+00   2.2    3.9   5.6e+00    419    33557    1.0
```

You can use MCMC to
validate your model as well

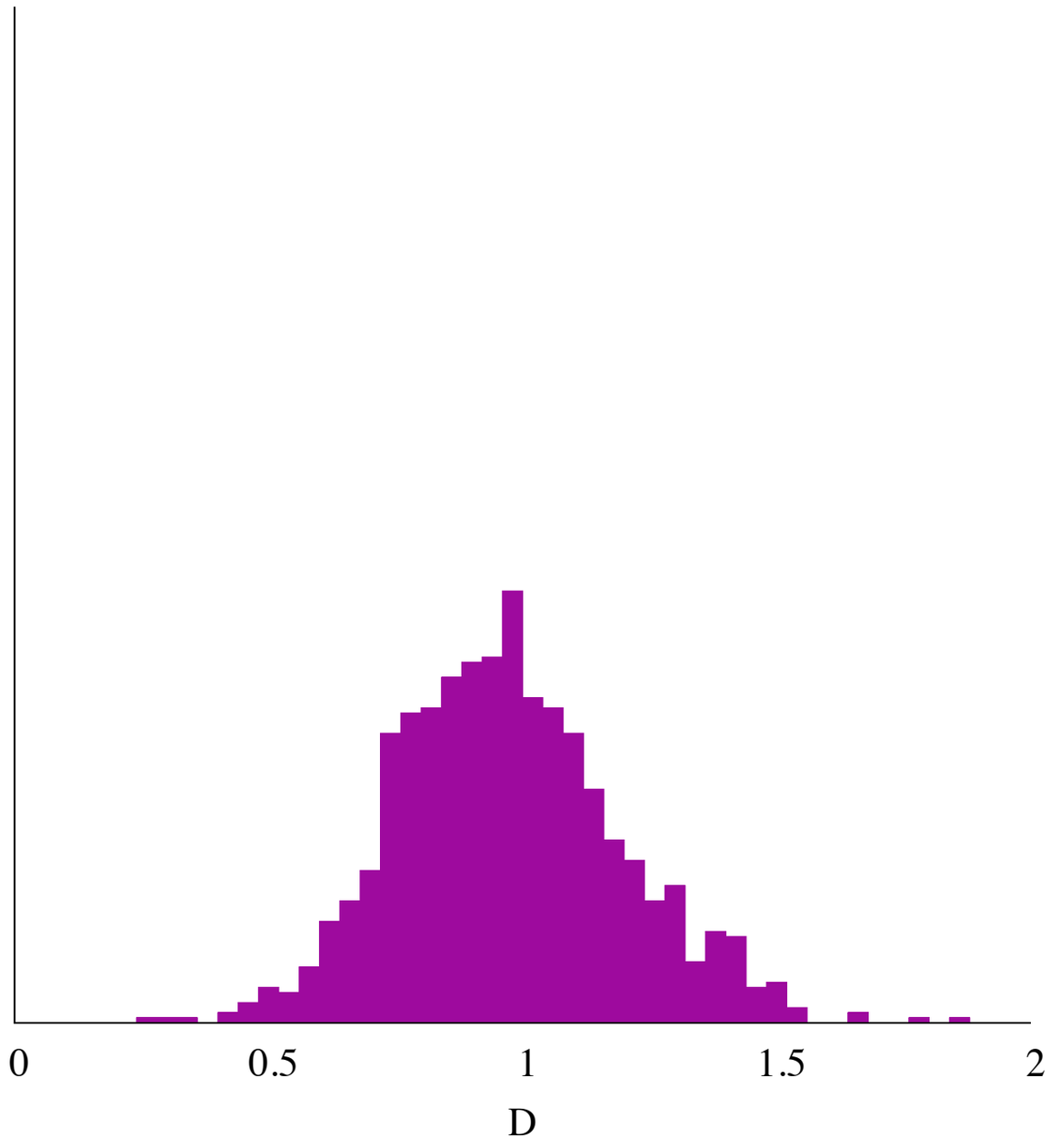
$$\pi\left(\tilde{\mathcal{D}}|\mathcal{D}\right) = \int \mathrm{d}\theta\, \pi\left(\tilde{\mathcal{D}}|\theta\right) \pi\Big(\theta|\mathcal{D}\Big)$$

You can use MCMC to
validate your model as well

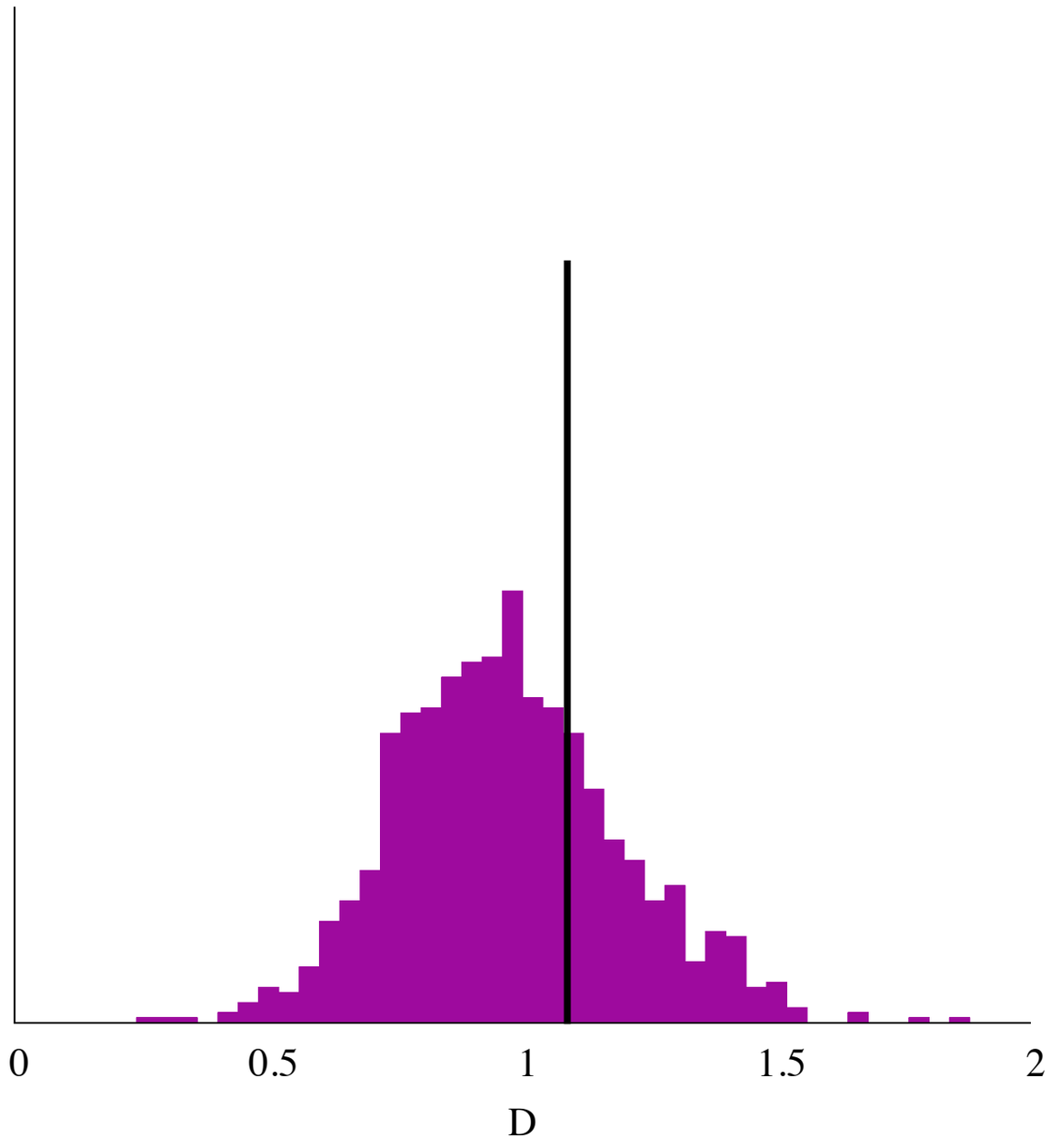
$$\pi\Big(\tilde{\mathcal{D}}|\mathcal{D}\Big) = \int \mathrm{d}\theta\, \pi\Big(\tilde{\mathcal{D}}|\theta\Big)\, \pi\Big(\theta|\mathcal{D}\Big)$$

$$\theta \sim \pi\Big(\theta|\mathcal{D}\Big) \qquad \tilde{\mathcal{D}} \sim \pi\Big(\tilde{\mathcal{D}}|\theta\Big)$$

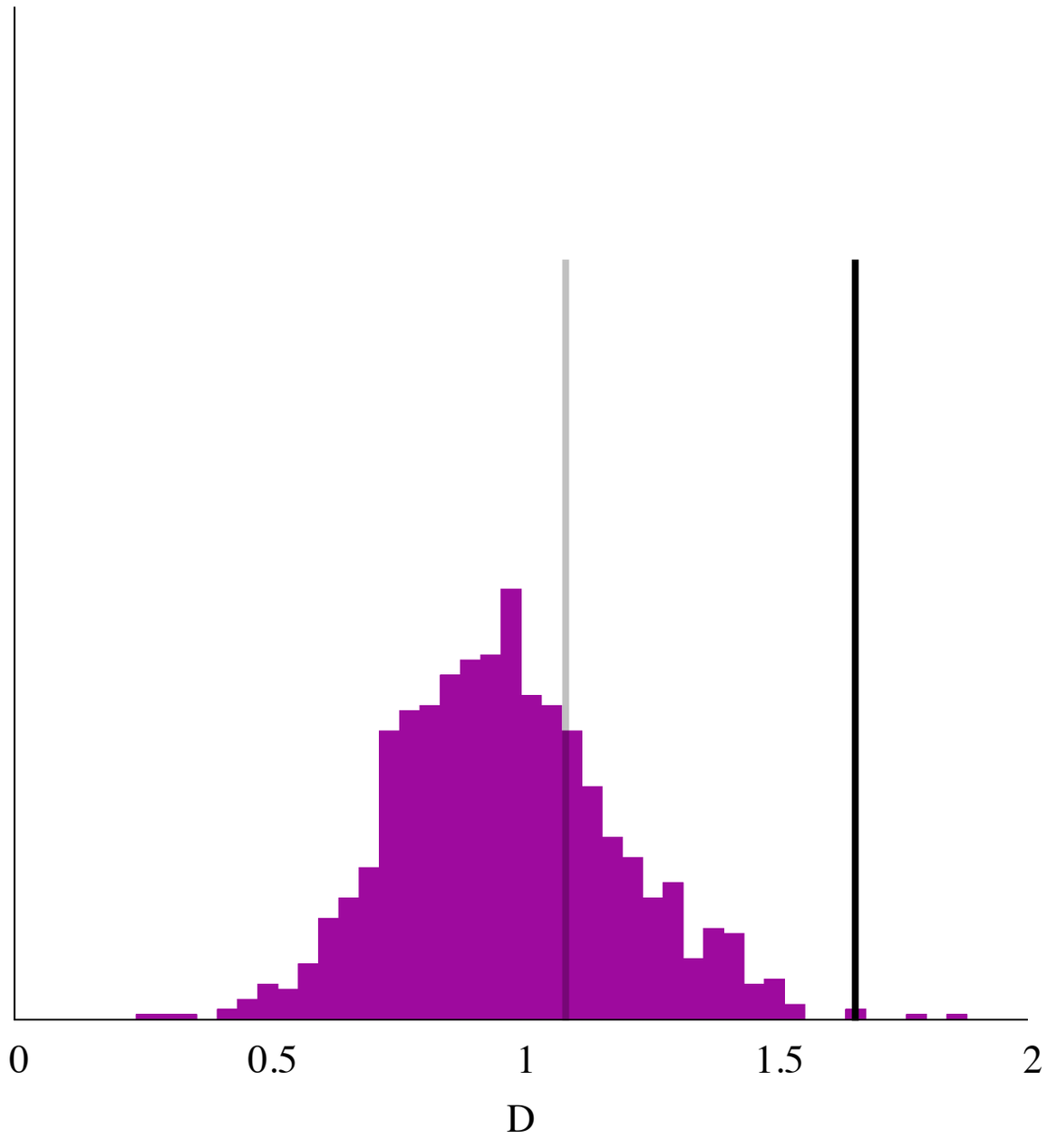
You can use MCMC to validate your model as well

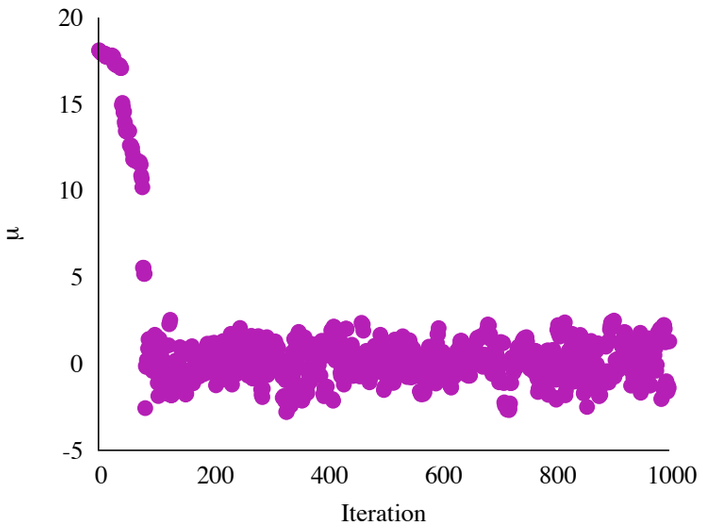


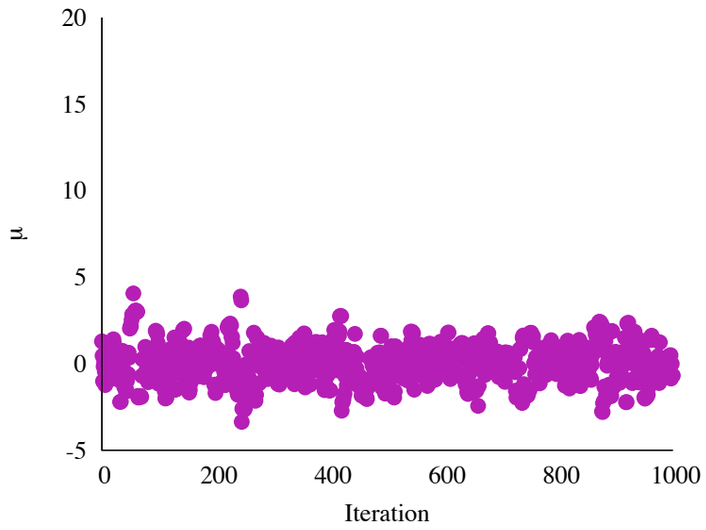
You can use MCMC to validate your model as well

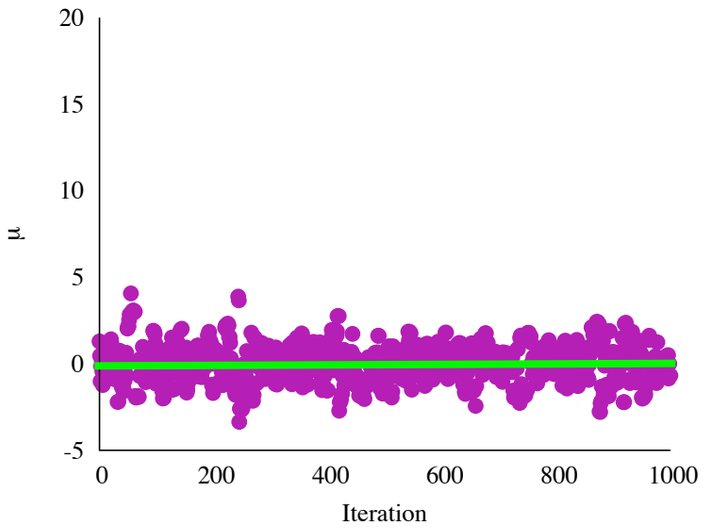


You can use MCMC to validate your model as well

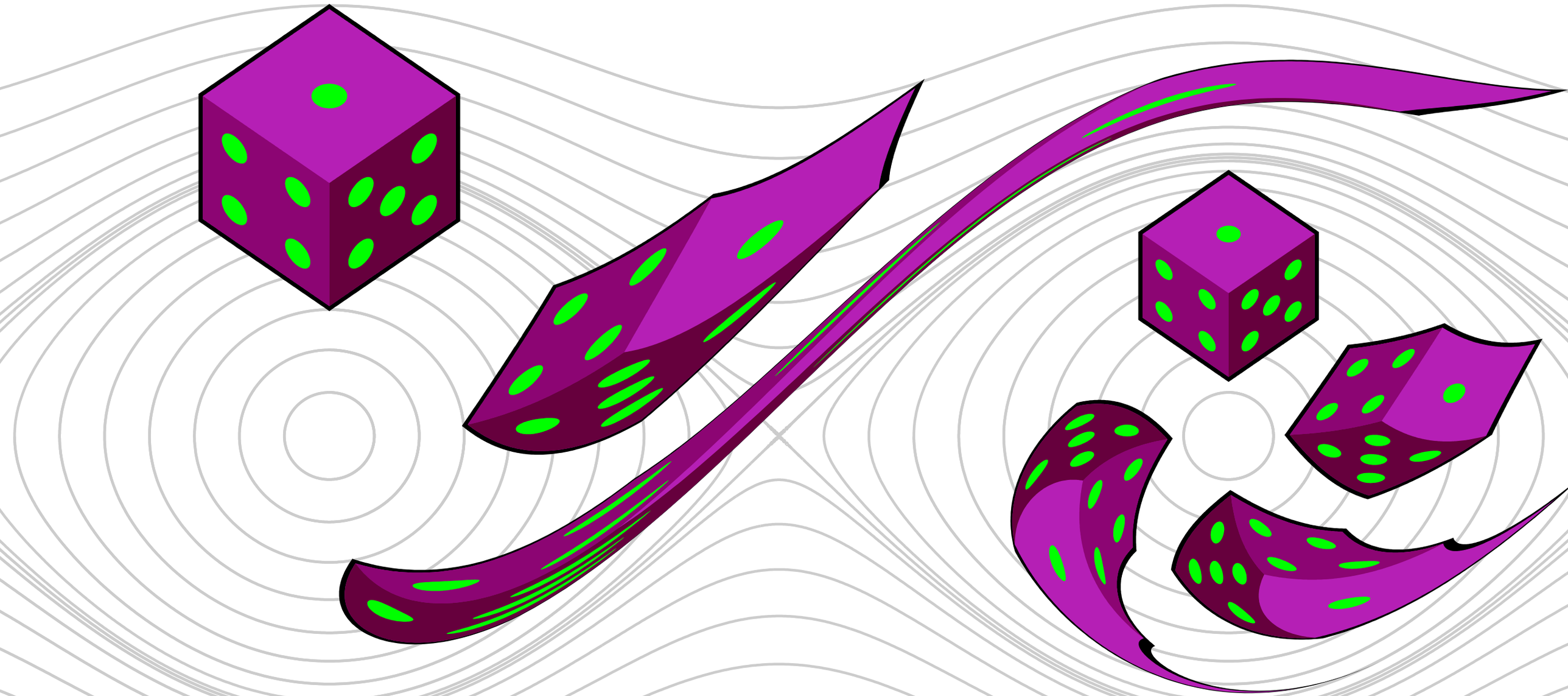



20
15
10
5
0
-5
μ
0
200
400
600
800
1000
Iteration


20
15
10
5
0
-5
μ
0
200
400
600
800
1000
Iteration


20
15
10
5
0
-5
μ
0
200
400
600
800
1000
Iteration

An Introduction to Hamiltonian Monte Carlo



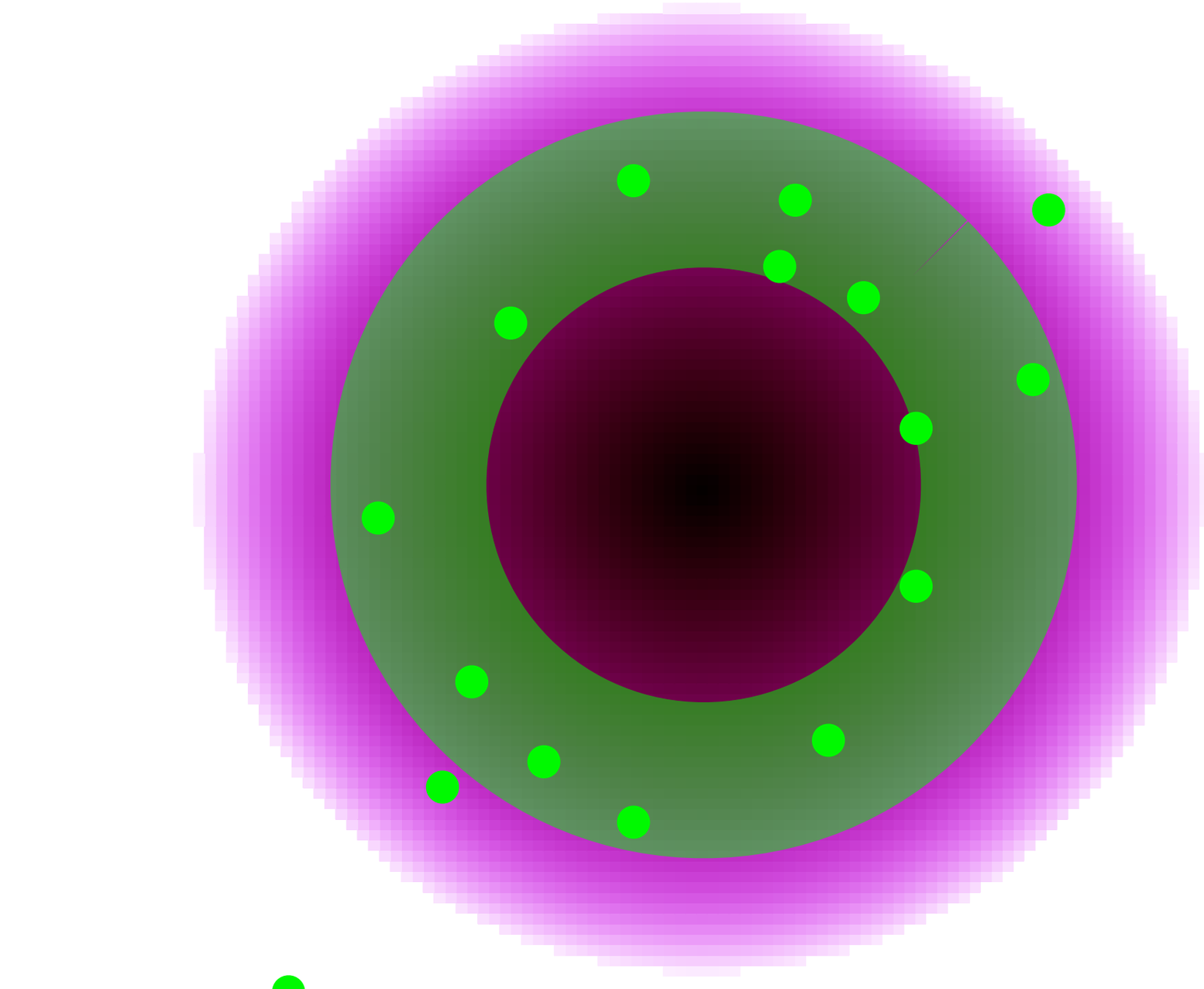
Random Walk Metropolis generates transitions with a "guided" diffusion

$$T(\theta, \theta') = \mathcal{N}\left(\theta' | \theta, \sigma^2\right) \min\left(1, \frac{\pi(\theta')}{\pi(\theta)}\right)$$

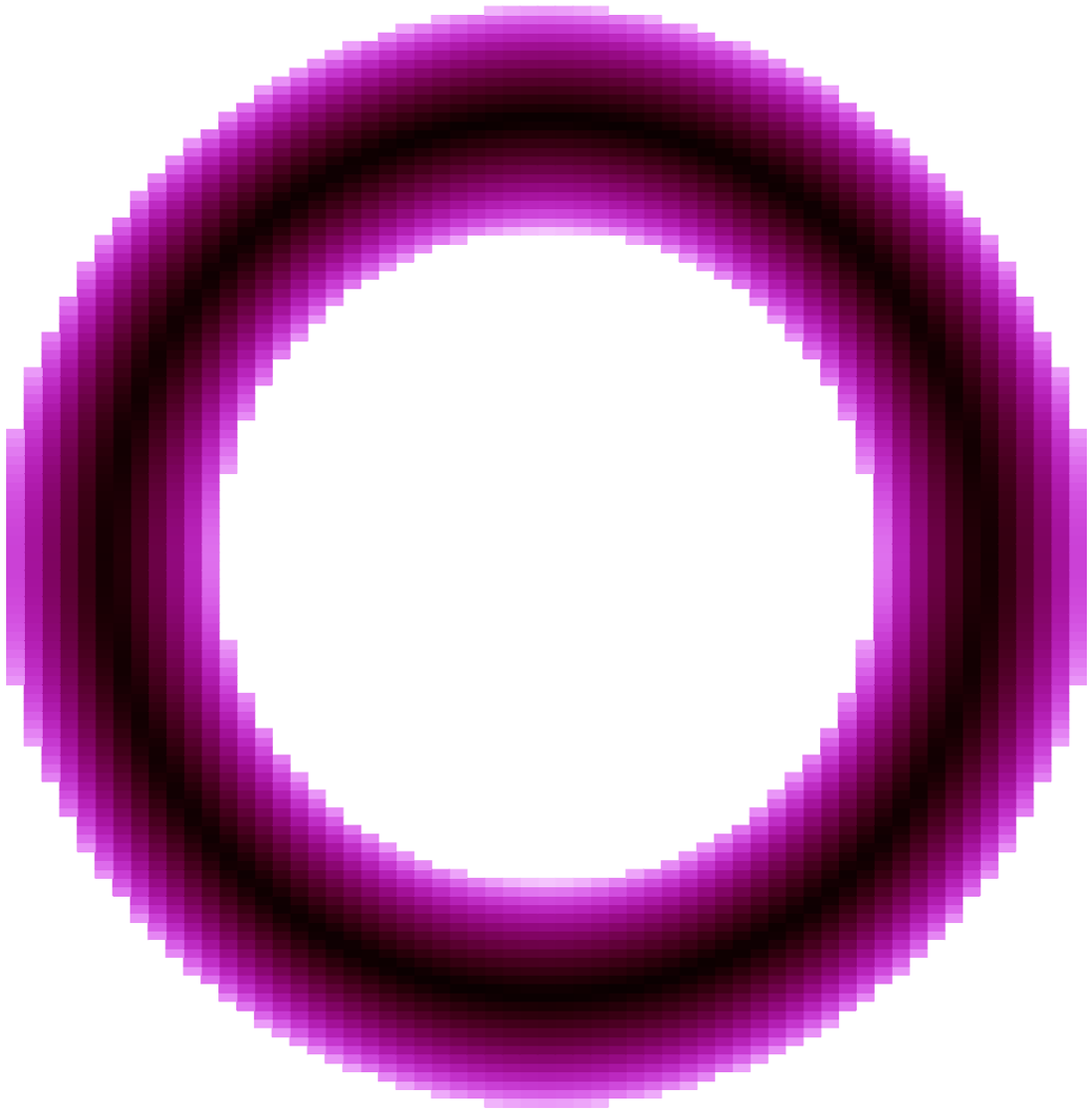
While the Gibbs sampler scans through conditional transitions

$$T(\theta, \theta') = \prod_i \pi\left(\theta_i' | \theta_{j \setminus i}\right)$$

In order to understand the efficacy of these transitions we have to consider the distribution of probability mass

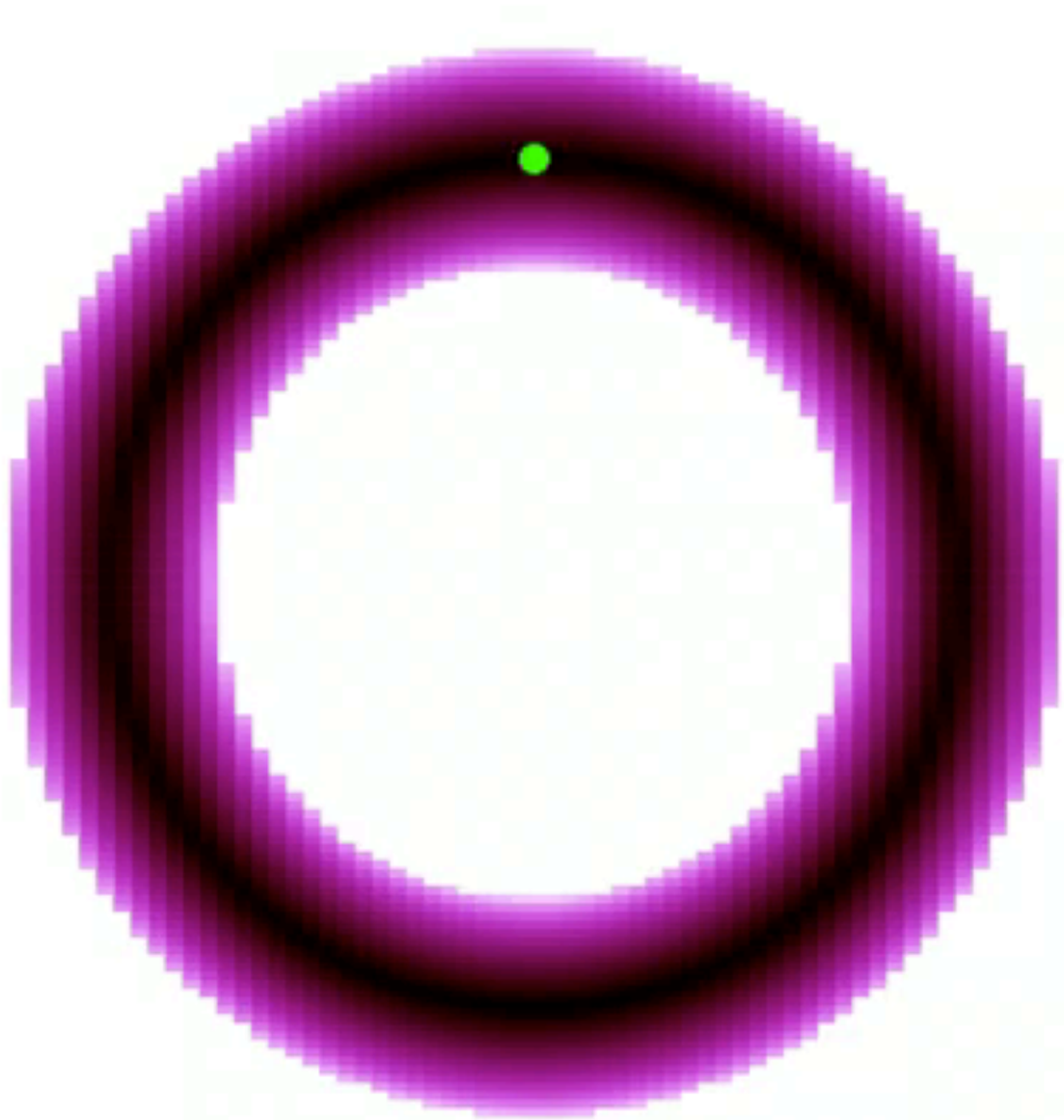


In practice, MCMC performance is limited by the complex distribution of posterior mass



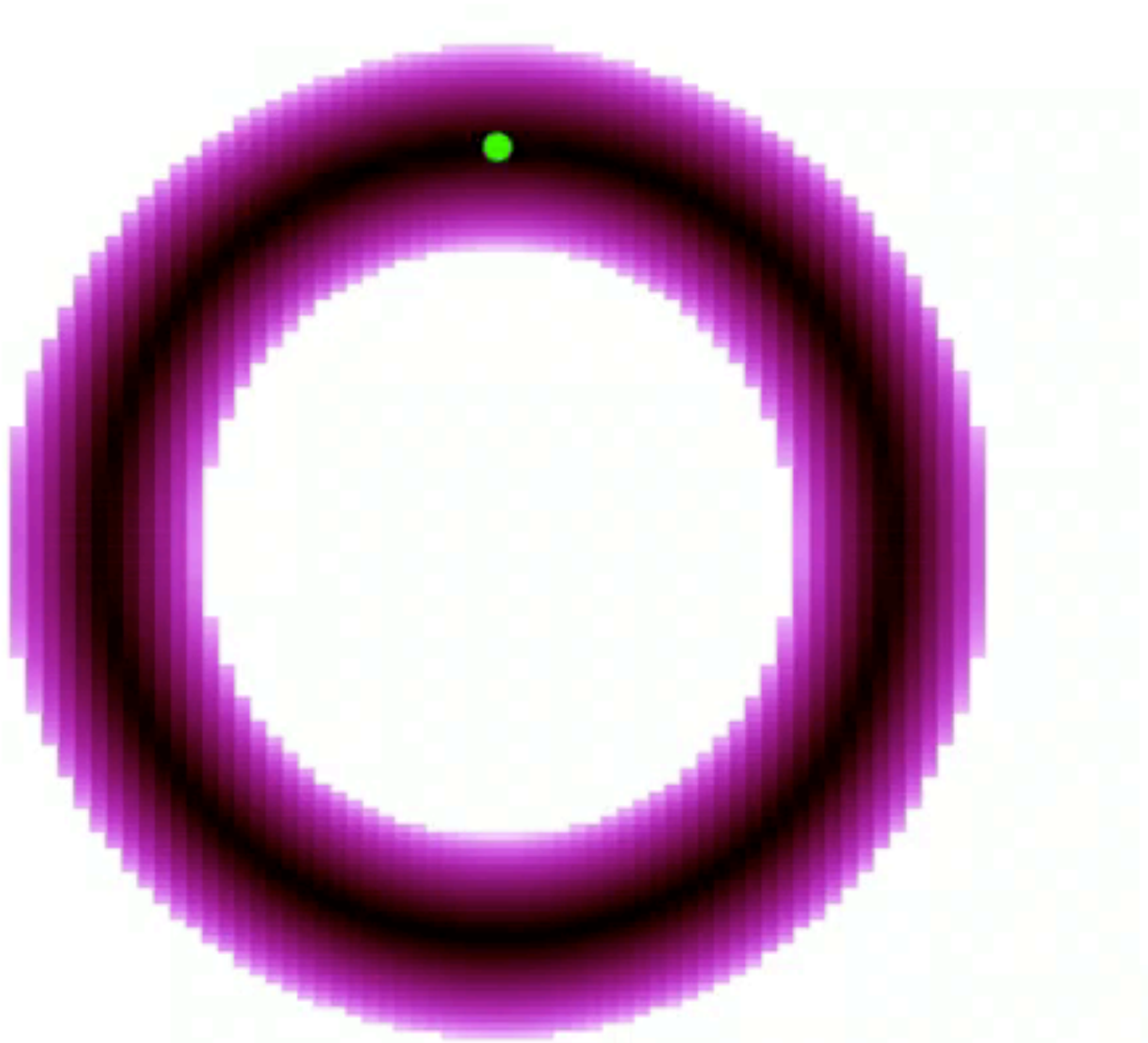
Random walk Metropolis sampling explores only slowly

Random walk Metropolis sampling explores only slowly

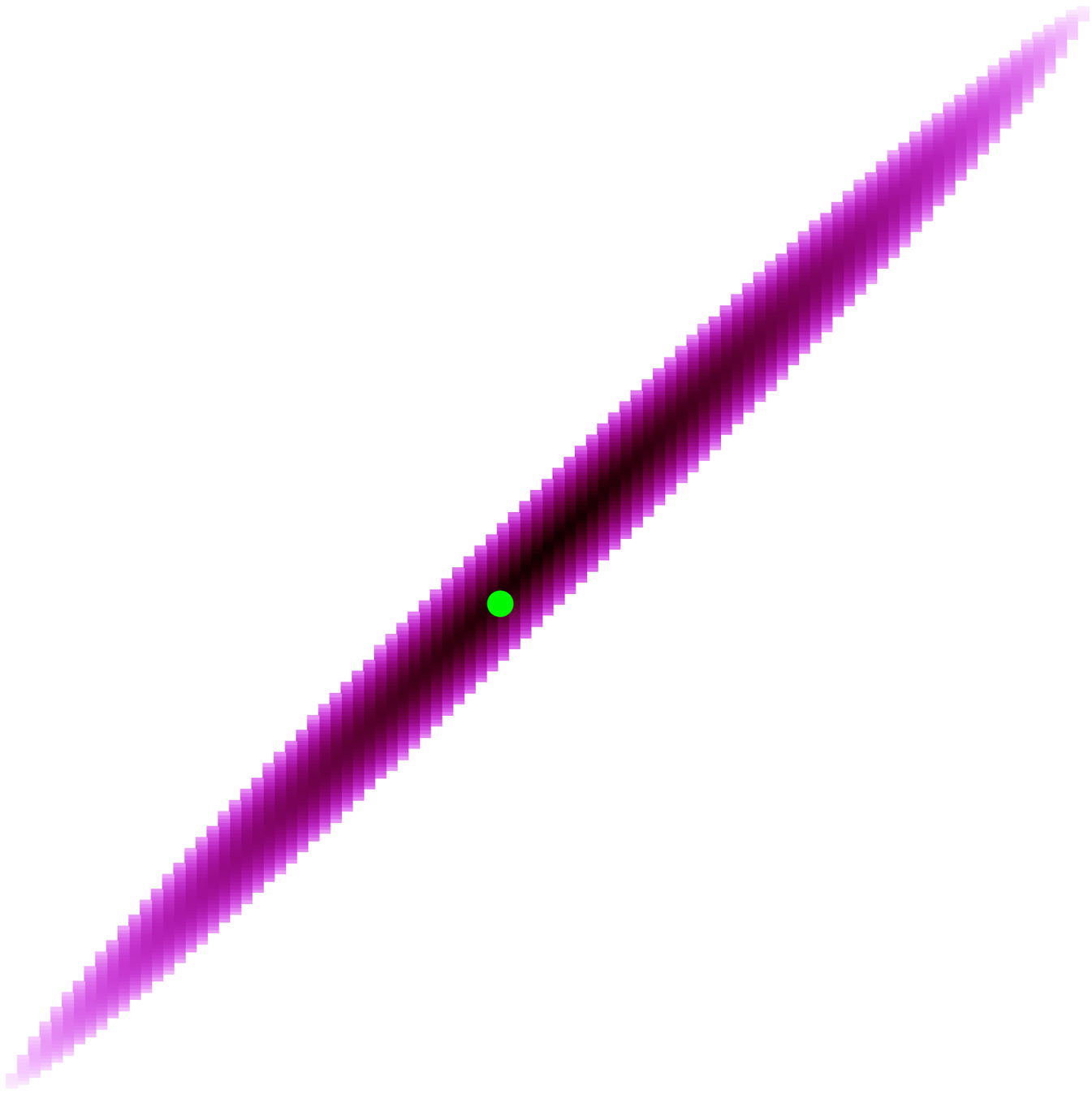


Gibbs sampling doesn't fare much better

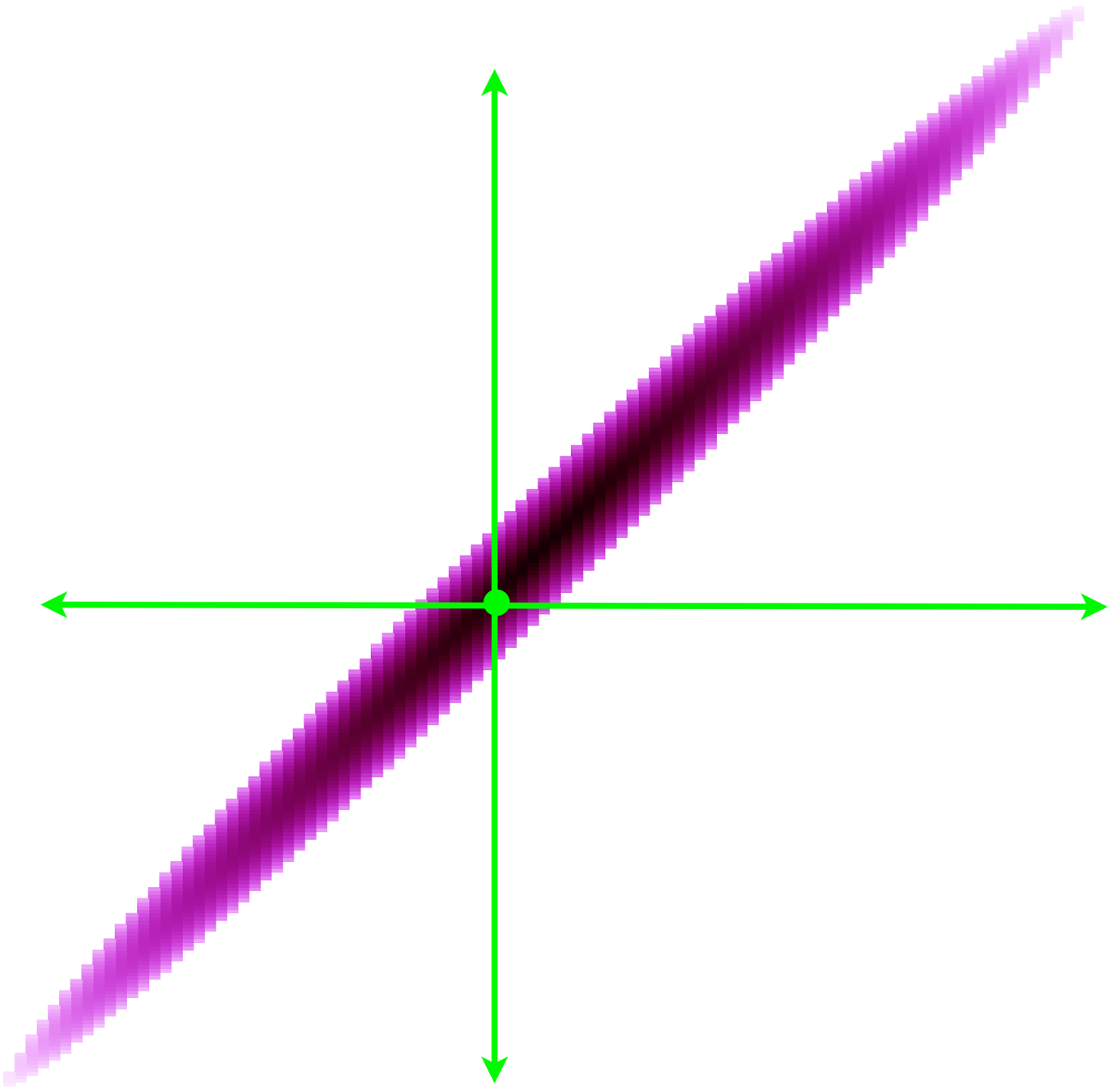
Gibbs sampling doesn't fare much better



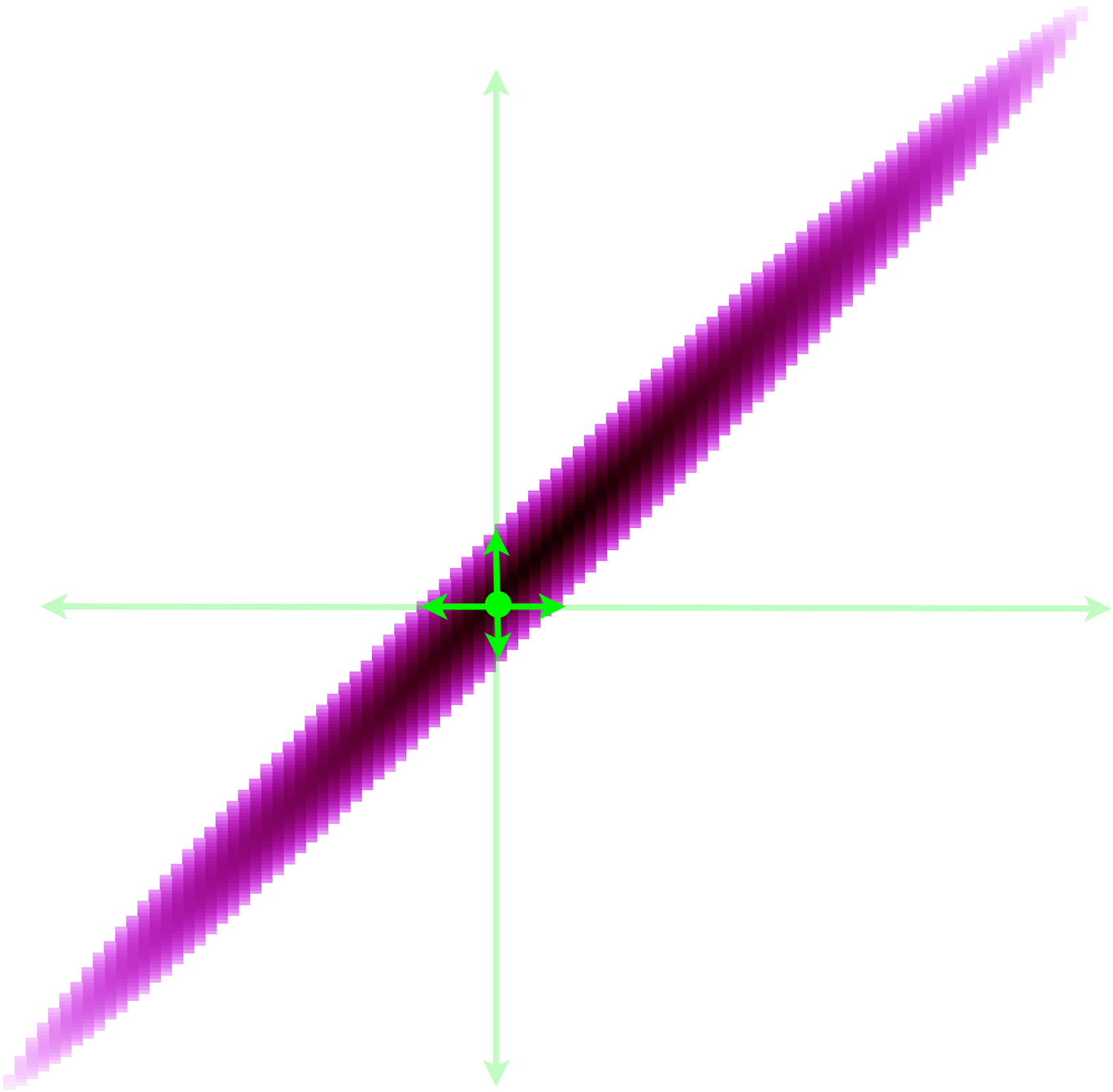
The problem is that RWM and Gibbs explore *incoherently*



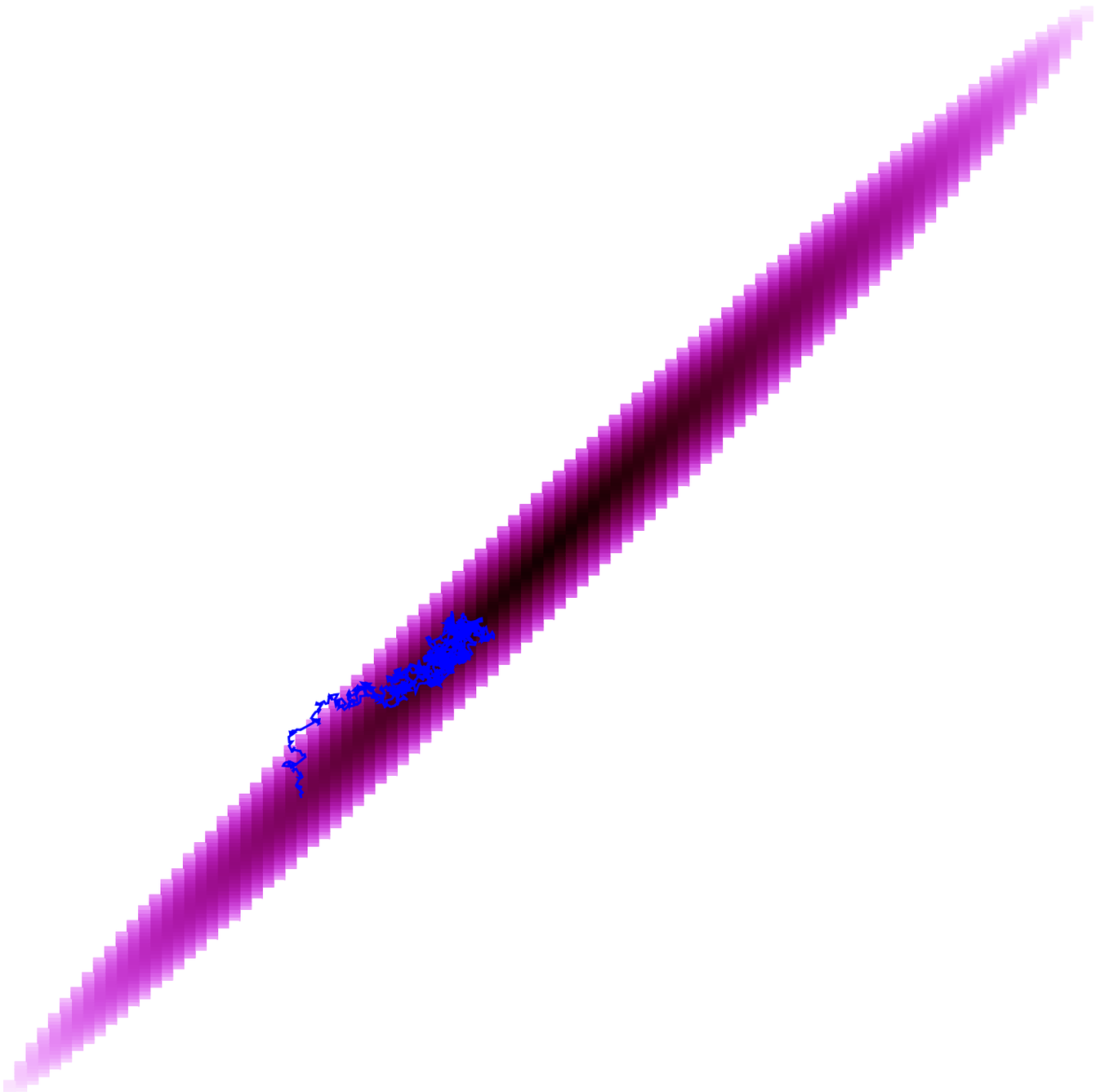
The problem is that RWM and Gibbs explore *incoherently*



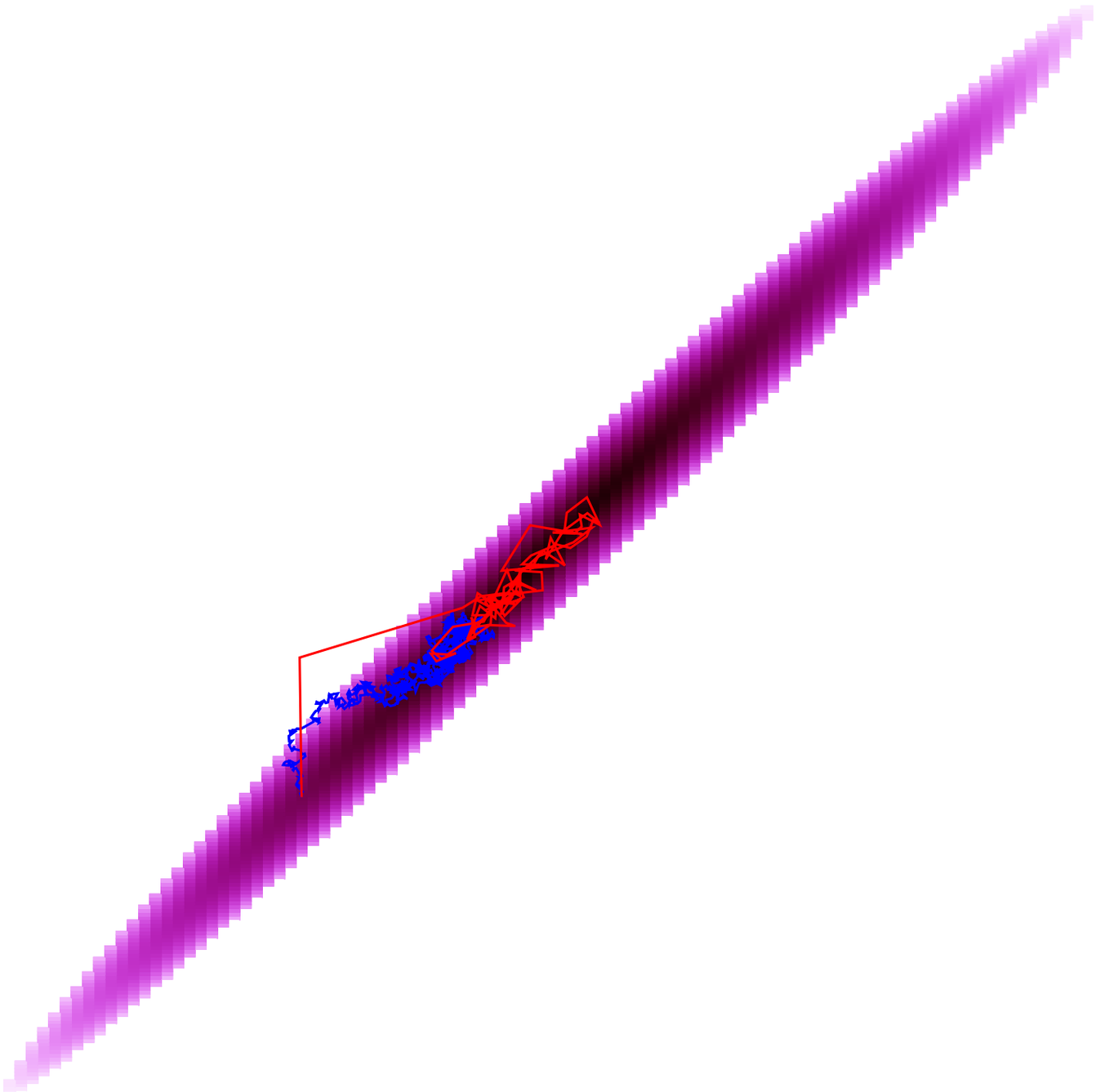
The problem is that RWM and Gibbs explore *incoherently*



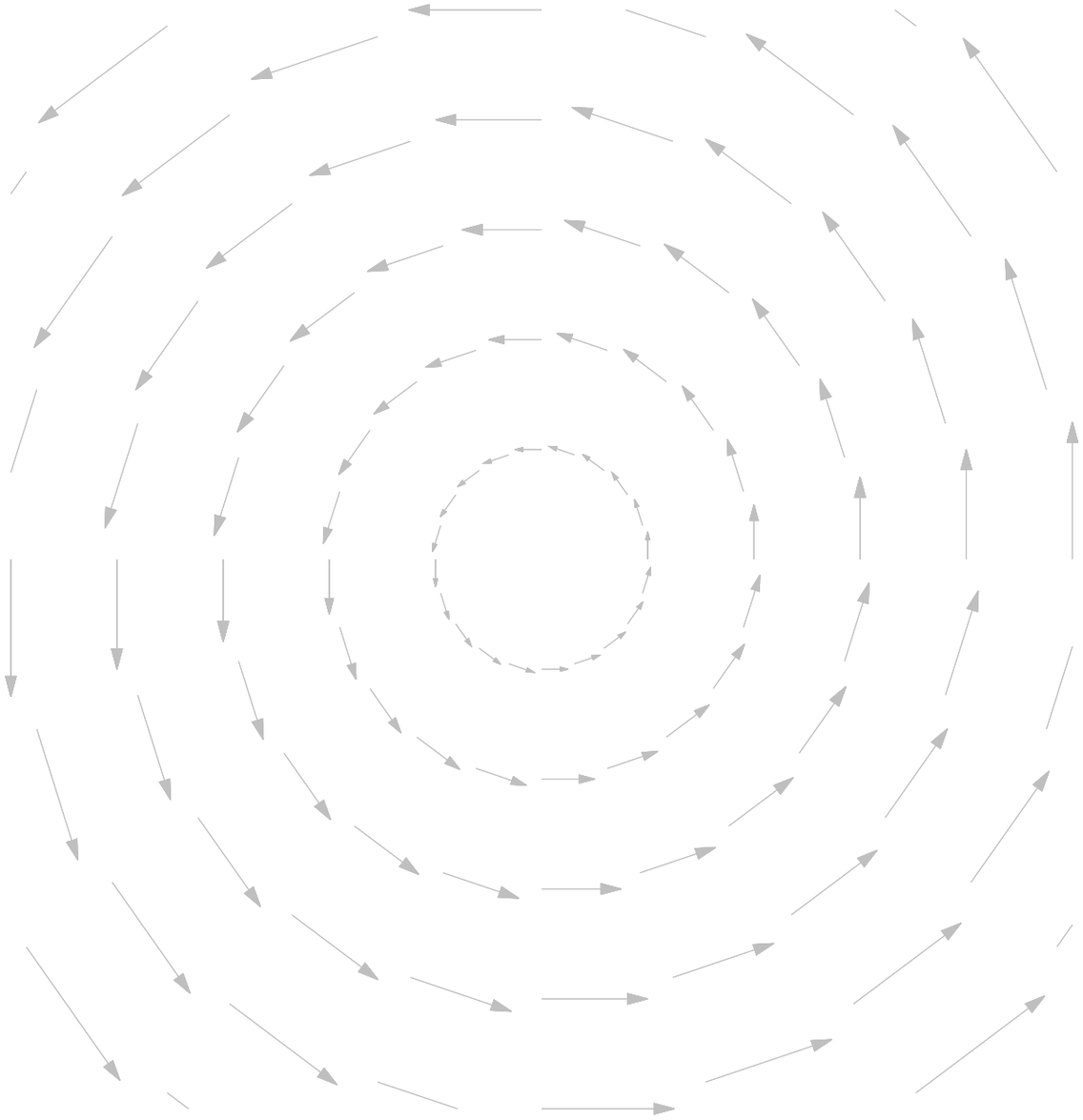
The problem is that RWM and Gibbs explore *incoherently*



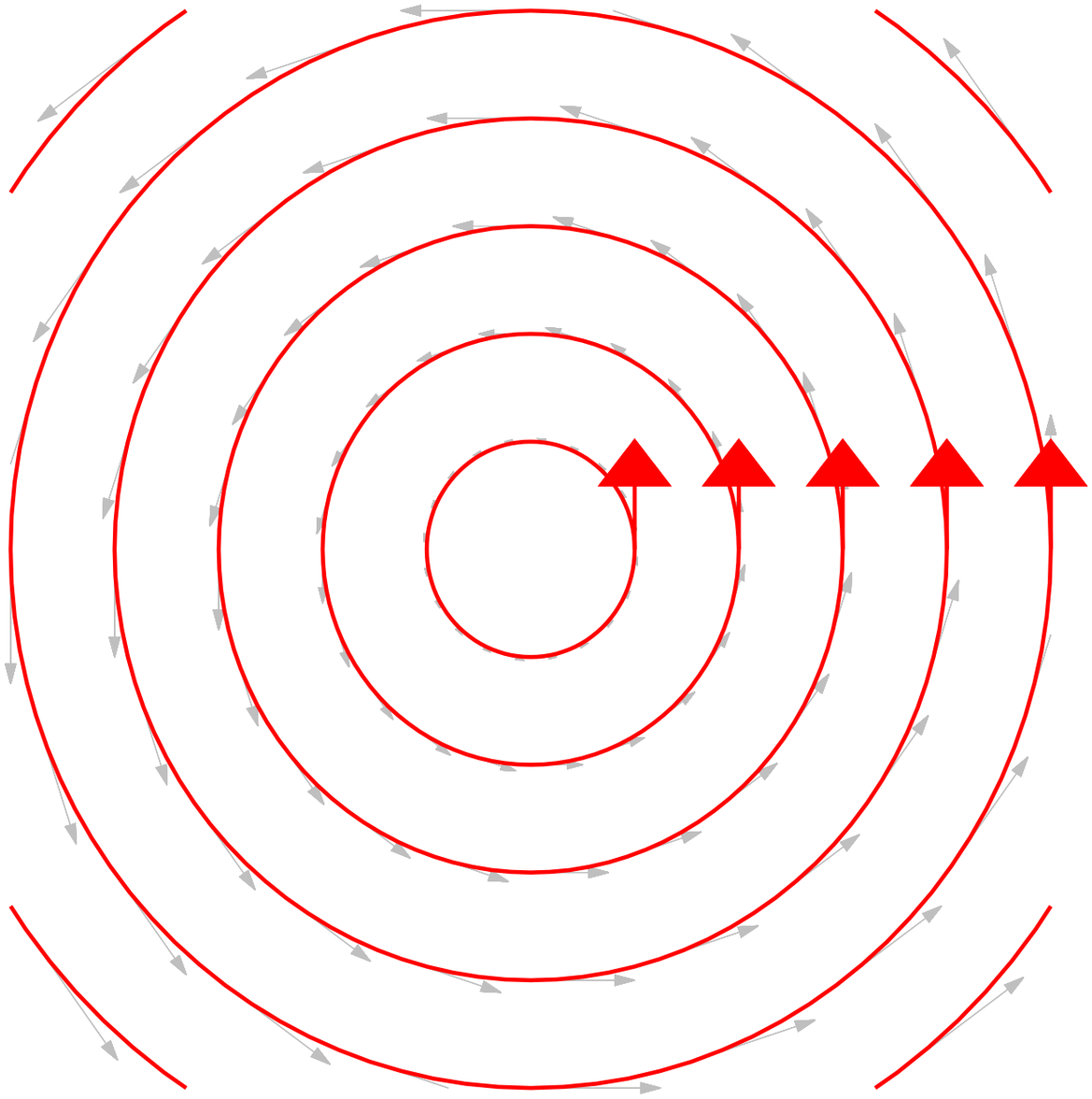
The problem is that RWM and Gibbs explore *incoherently*



How can we explore coherently?



How can we explore coherently?



Hamiltonian flow is a coherent,
measure-preserving map

$$q \to (p, q)$$

Hamiltonian flow is a coherent,
measure-preserving map

$$q \to (p, q)$$

$$H(p,q) \to e^{-H(p,q)} \mathrm{d}^n p \, \mathrm{d}^n q$$

Which is exactly what we need for a Markov transition

$$T(q', q) = \pi(p)\, \delta((p', q') - \phi_\tau(p, q))$$

We just need to define
the Hamiltonian appropriately

$$H(p, q) = -\log \pi(p, q)$$

We just need to define
the Hamiltonian appropriately

$$H(p, q) = -\log \pi(p, q)$$

$$-\log \pi(p|q)\, \pi(q)$$

We just need to define
the Hamiltonian appropriately

$$H(p, q) = -\log \pi(p, q)$$

$$-\log \pi(p|q)\, \pi(q)$$

$$-\log \pi(p|q) - \log \pi(q)$$

We just need to define
the Hamiltonian appropriately

$$H(p, q) = -\log \pi(p, q)$$

$$-\log \pi(p|q)\, \pi(q)$$

$$-\log \pi(p|q) - \log \pi(q)$$

$$T$$

We just need to define
the Hamiltonian appropriately

$$H(p, q) = -\log \pi(p, q)$$

$$-\log \pi(p|q)\, \pi(q)$$

$$-\log \pi(p|q) - \log \pi(q)$$

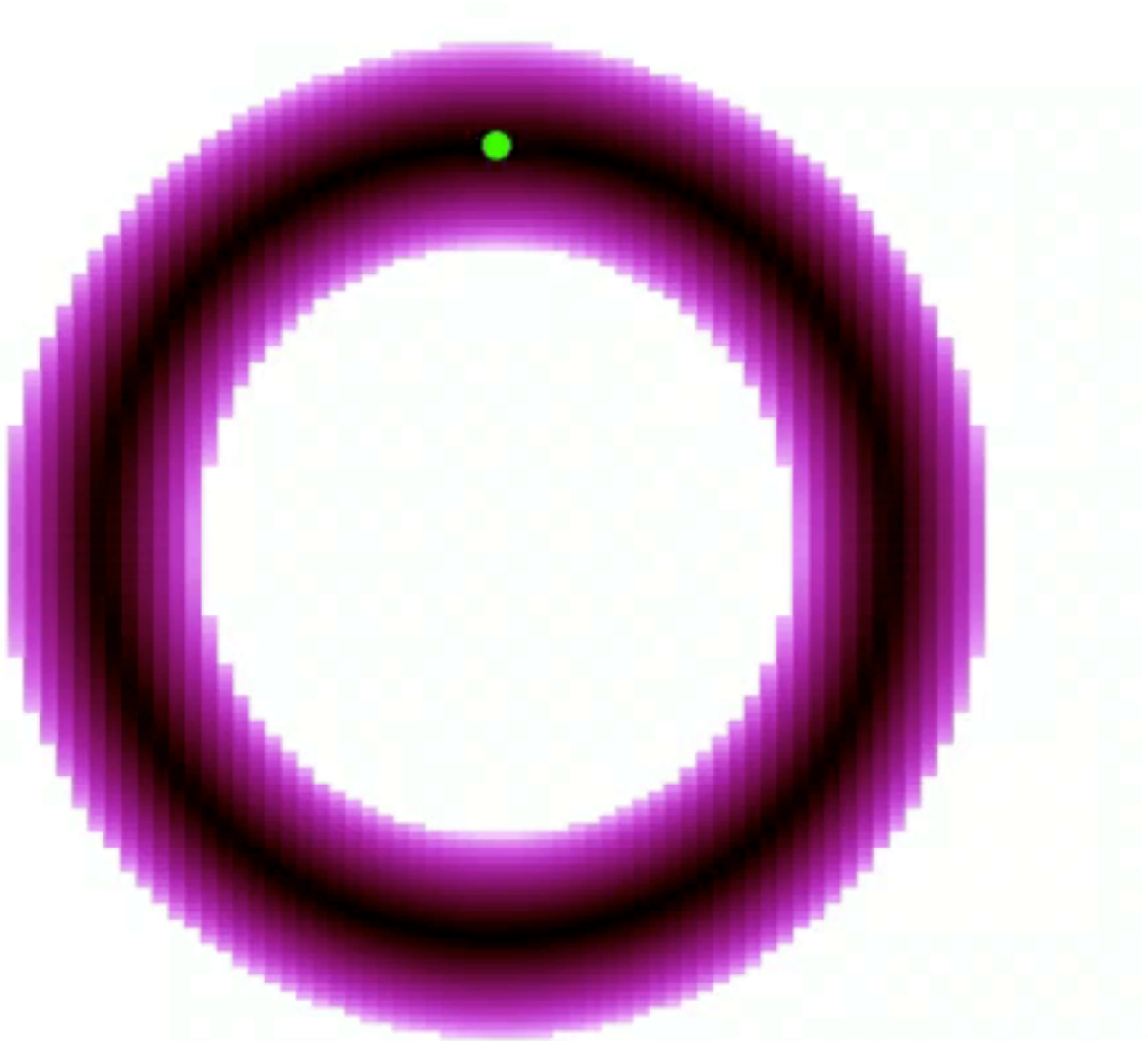
$$V$$

Quadratic kinetic energies with constant metrics emulate dynamics on a Euclidean manifold

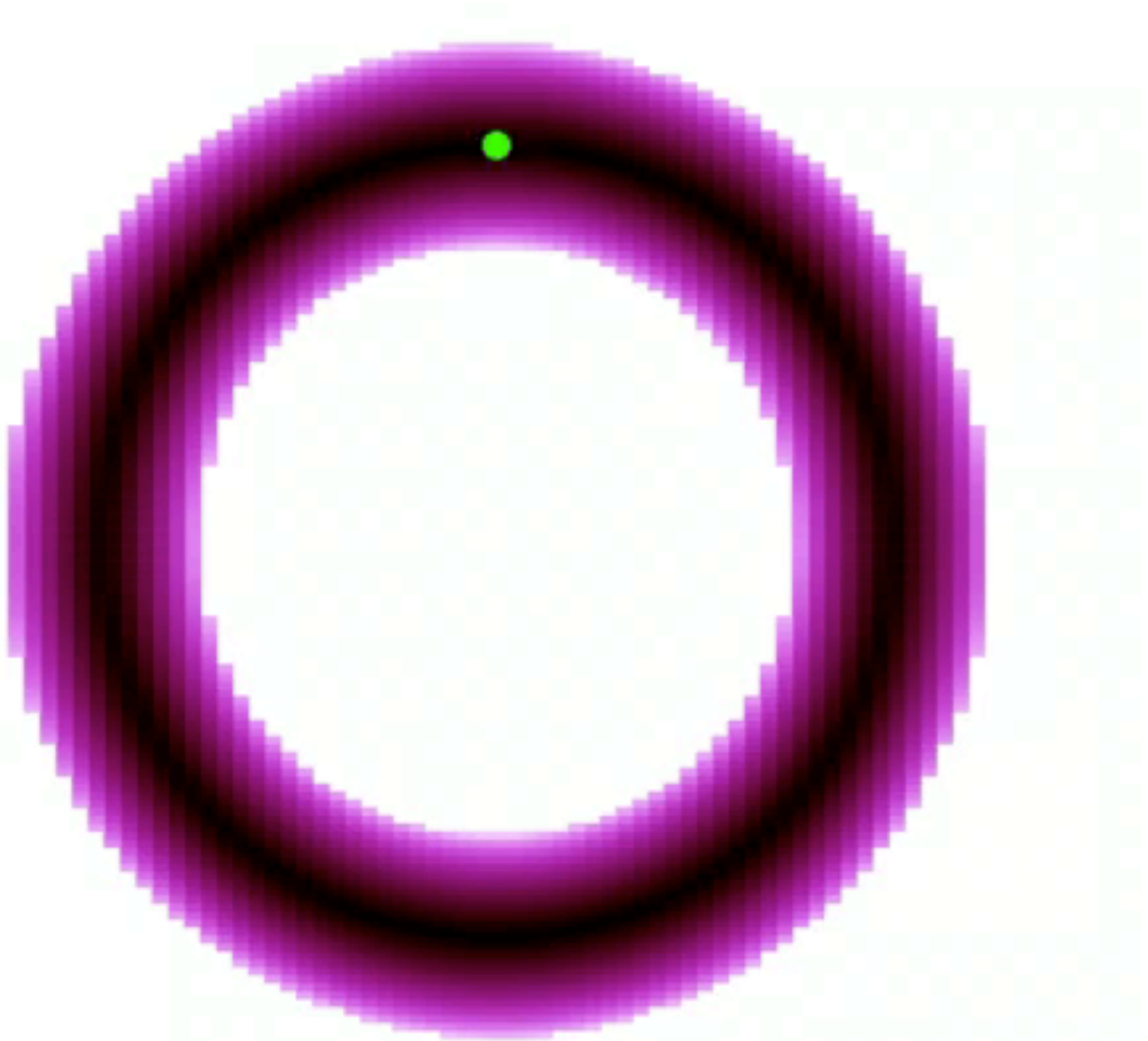
$$\pi(p|q) = \mathcal{N}(0, M)$$

$$T = \frac{1}{2} p_i p_j \left(M^{-1}\right)^{ij}$$

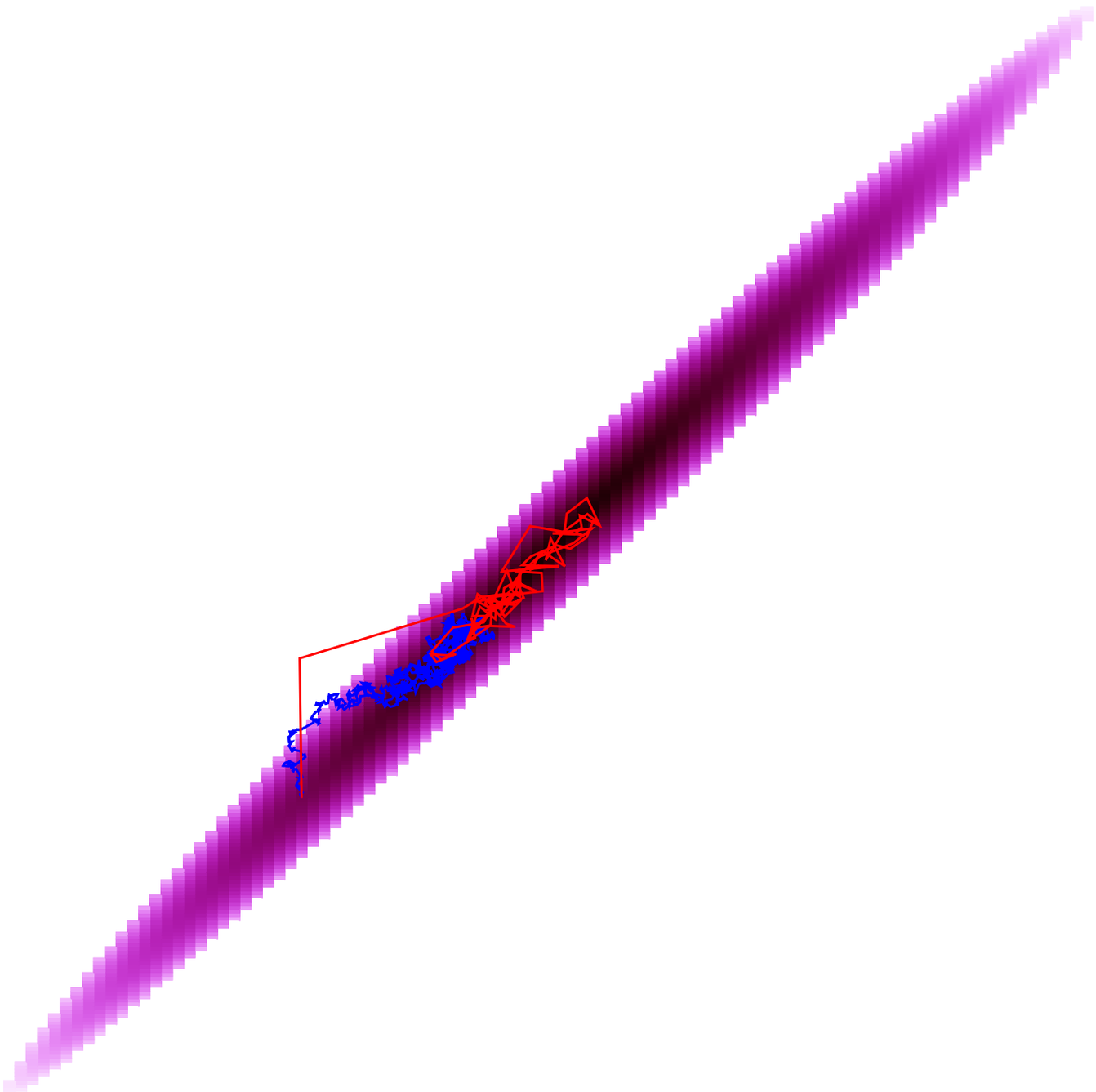
The coherent flow the Markov chain along the target distribution, avoiding random walk behavior



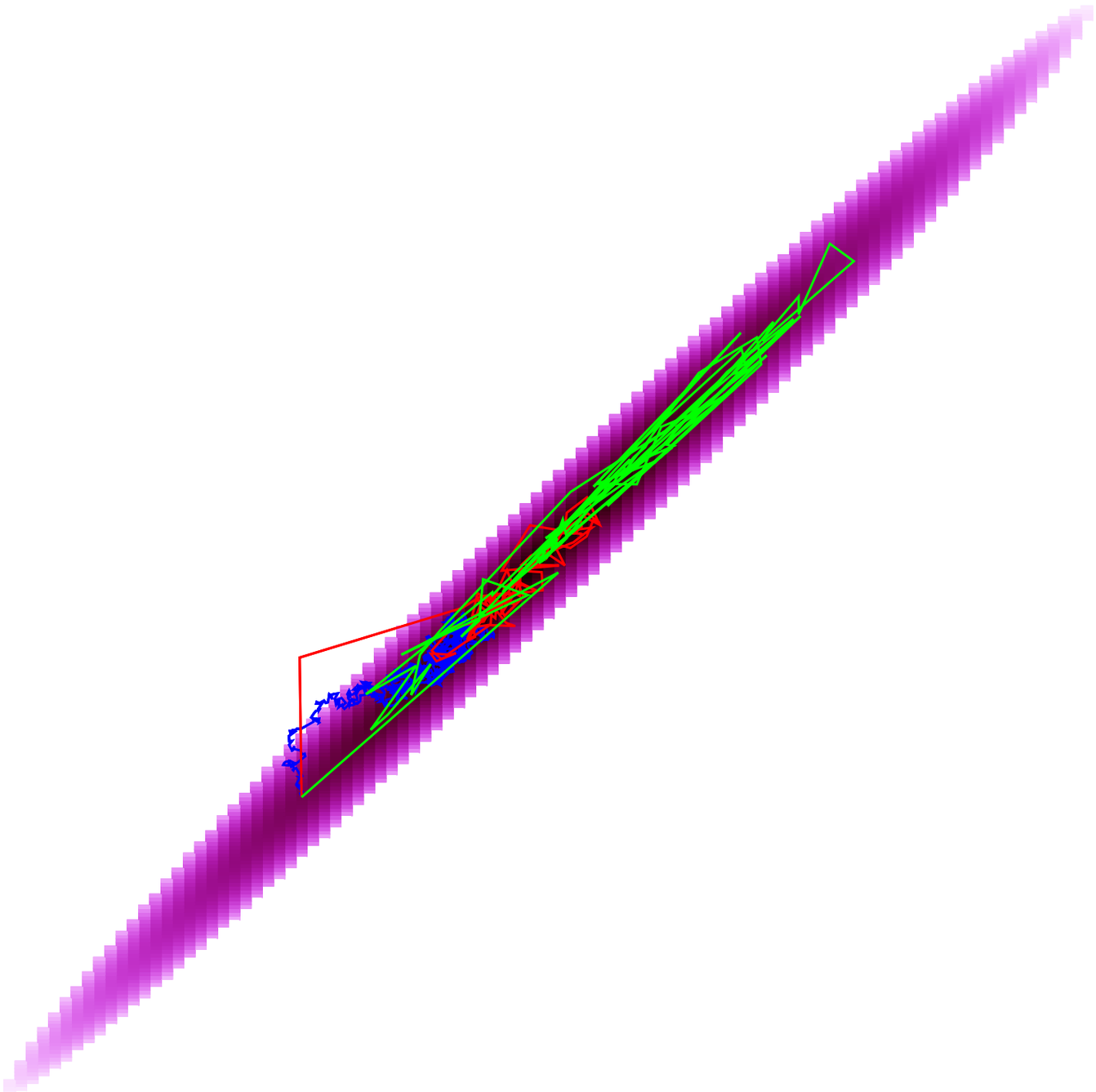
The coherent flow the Markov chain along the target distribution, avoiding random walk behavior



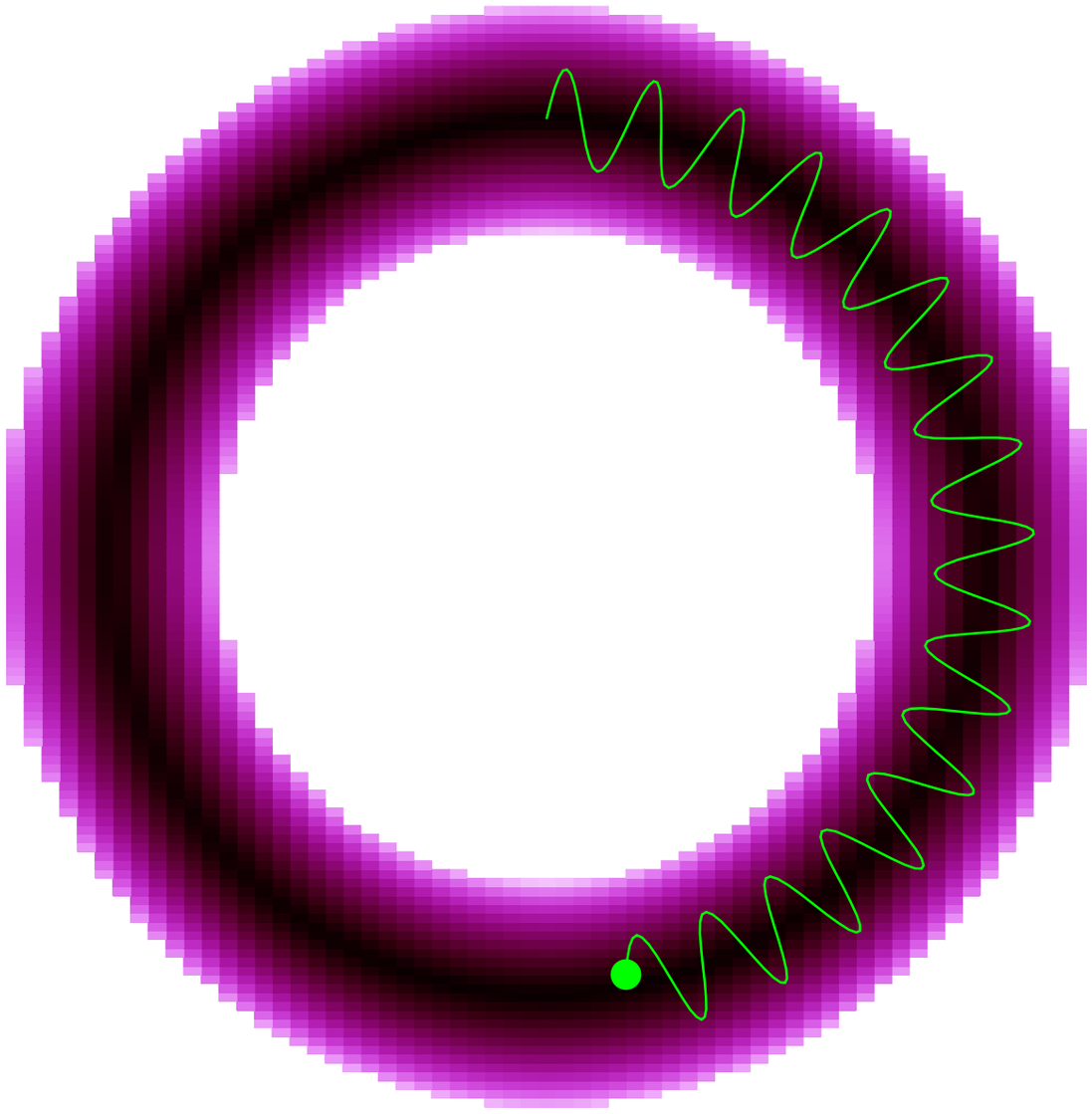
The coherent flow the Markov chain along the target distribution, avoiding random walk behavior



The coherent flow the Markov chain along the target distribution, avoiding random walk behavior



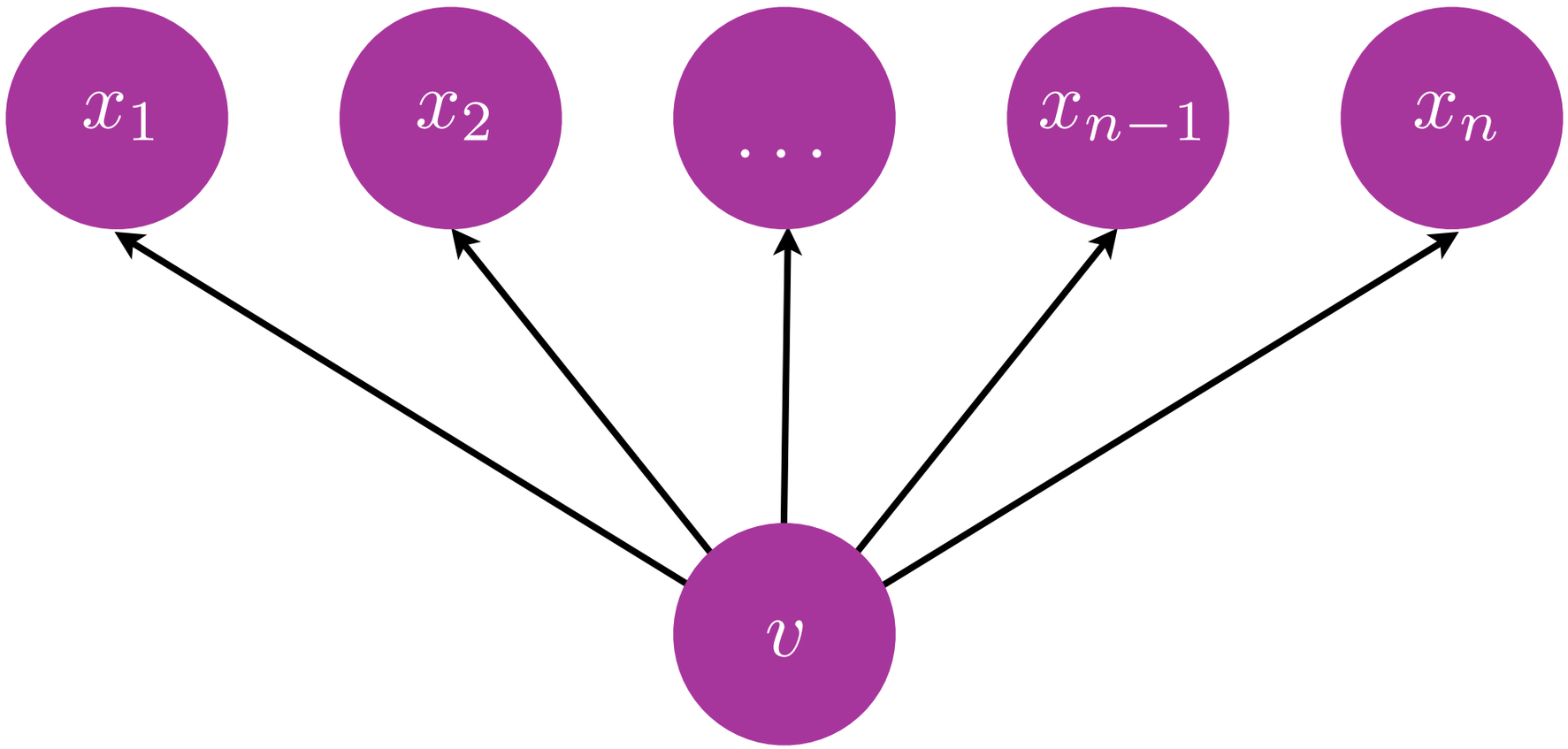
Unfortunately, Euclidean HMC is sensitive to large variations in curvature



As well as variations in the target density

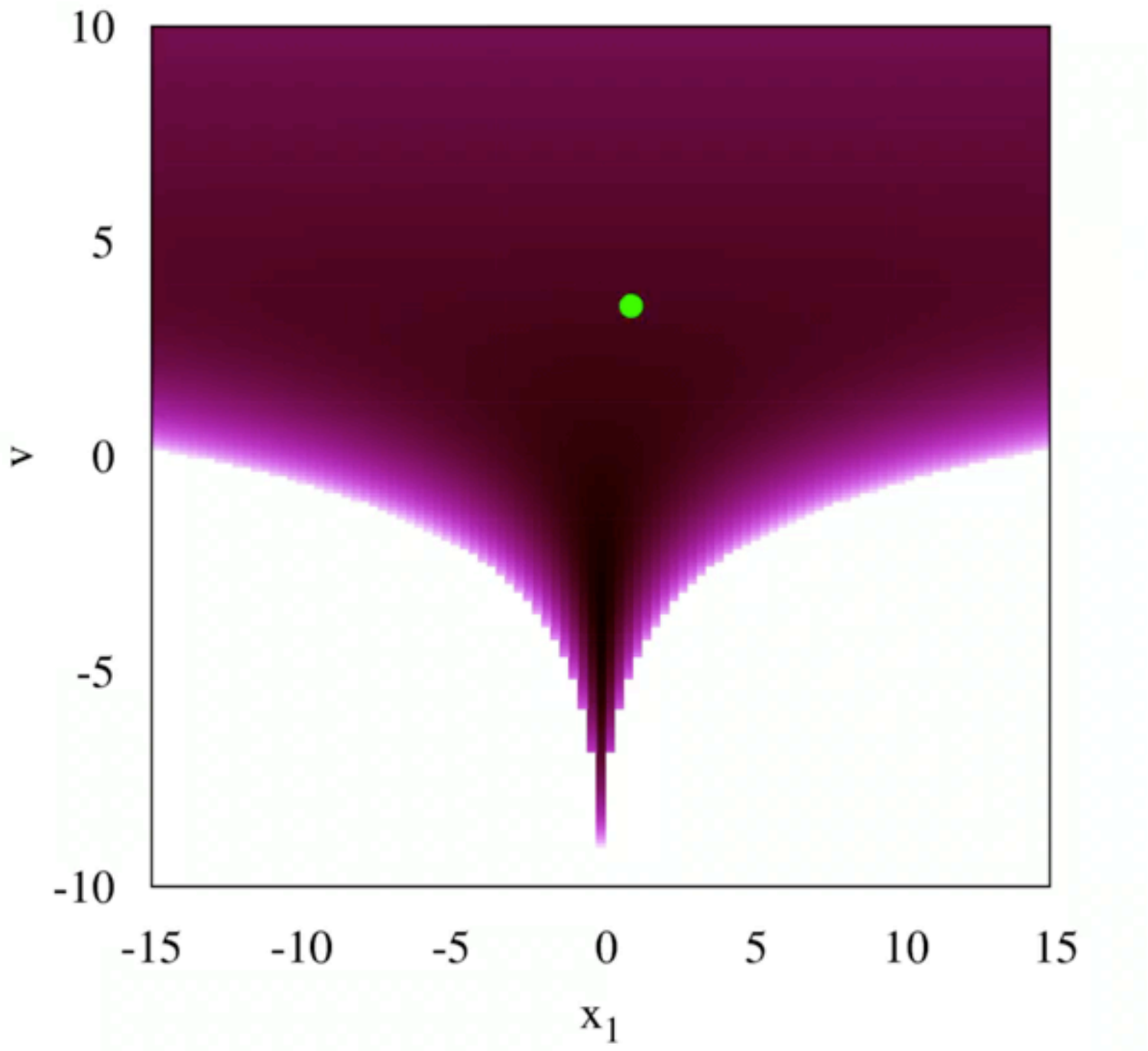
$$\Delta V = \Delta T = \frac{n}{2}$$

These weaknesses are particularly evident in hierarchical models

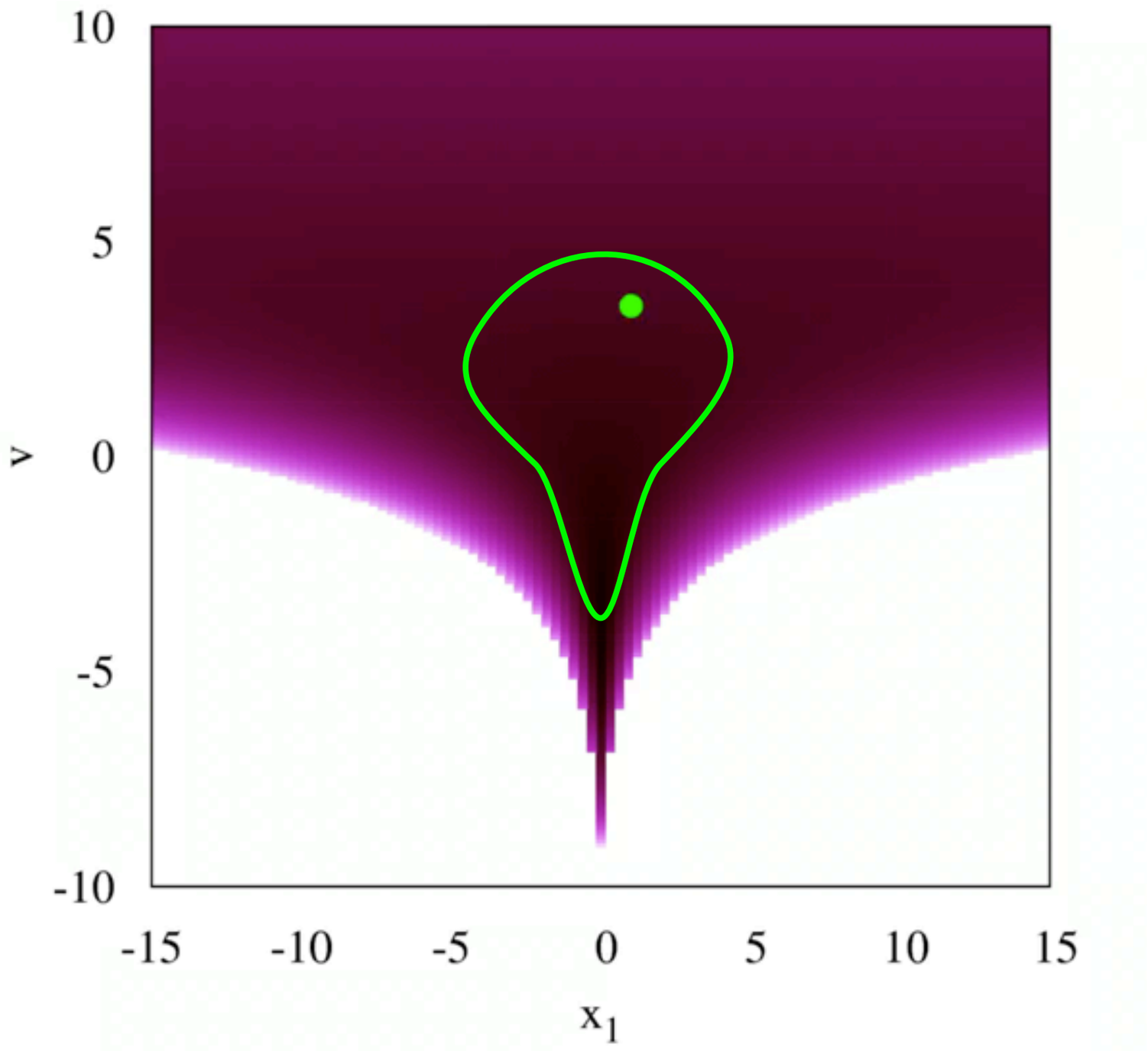


$$\pi(\mathbf{x}, v) = \prod_{i=1}^{n} \pi(x_i|v)\, \pi(v)$$

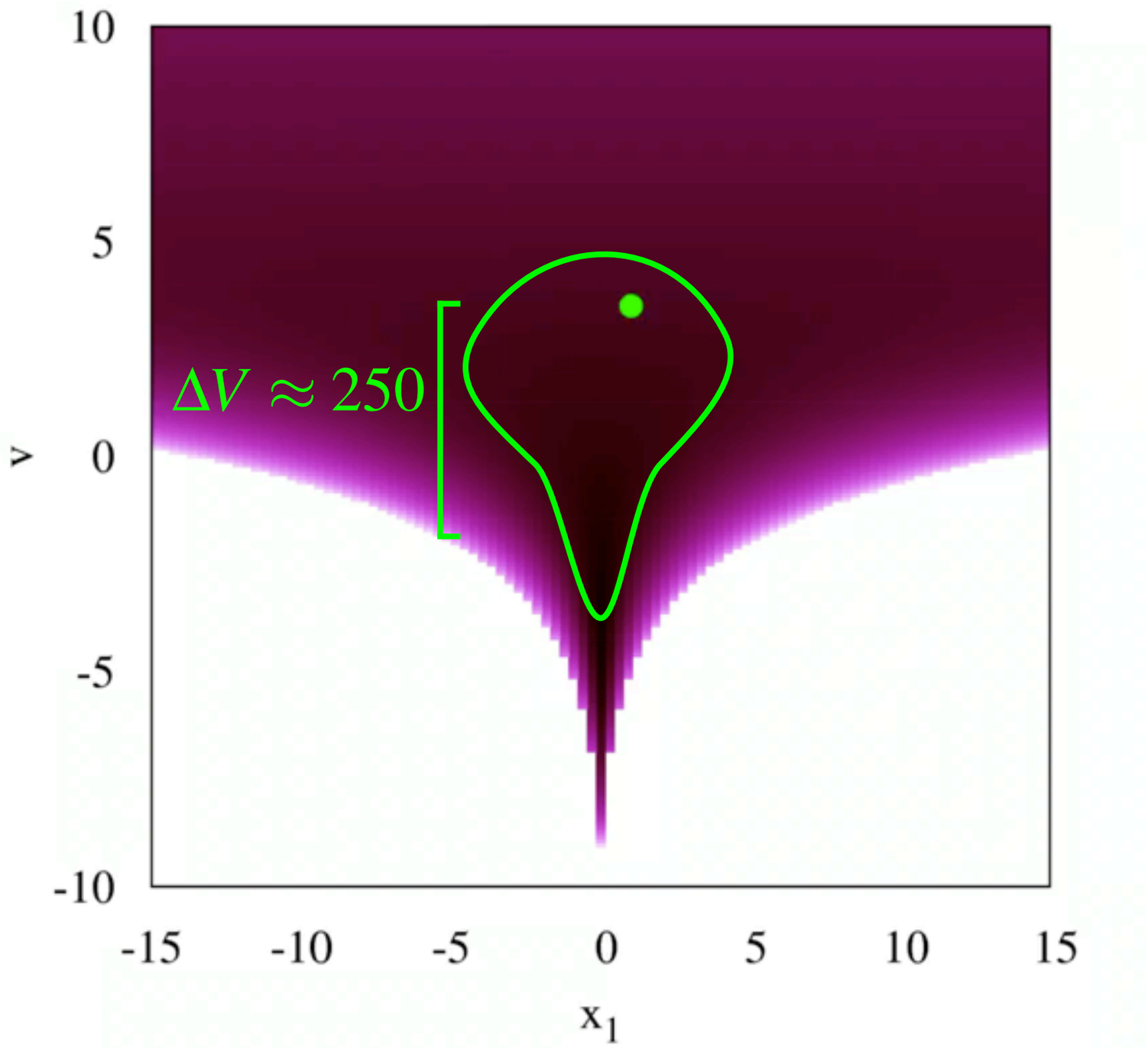
These weaknesses are particularly evident in hierarchical models



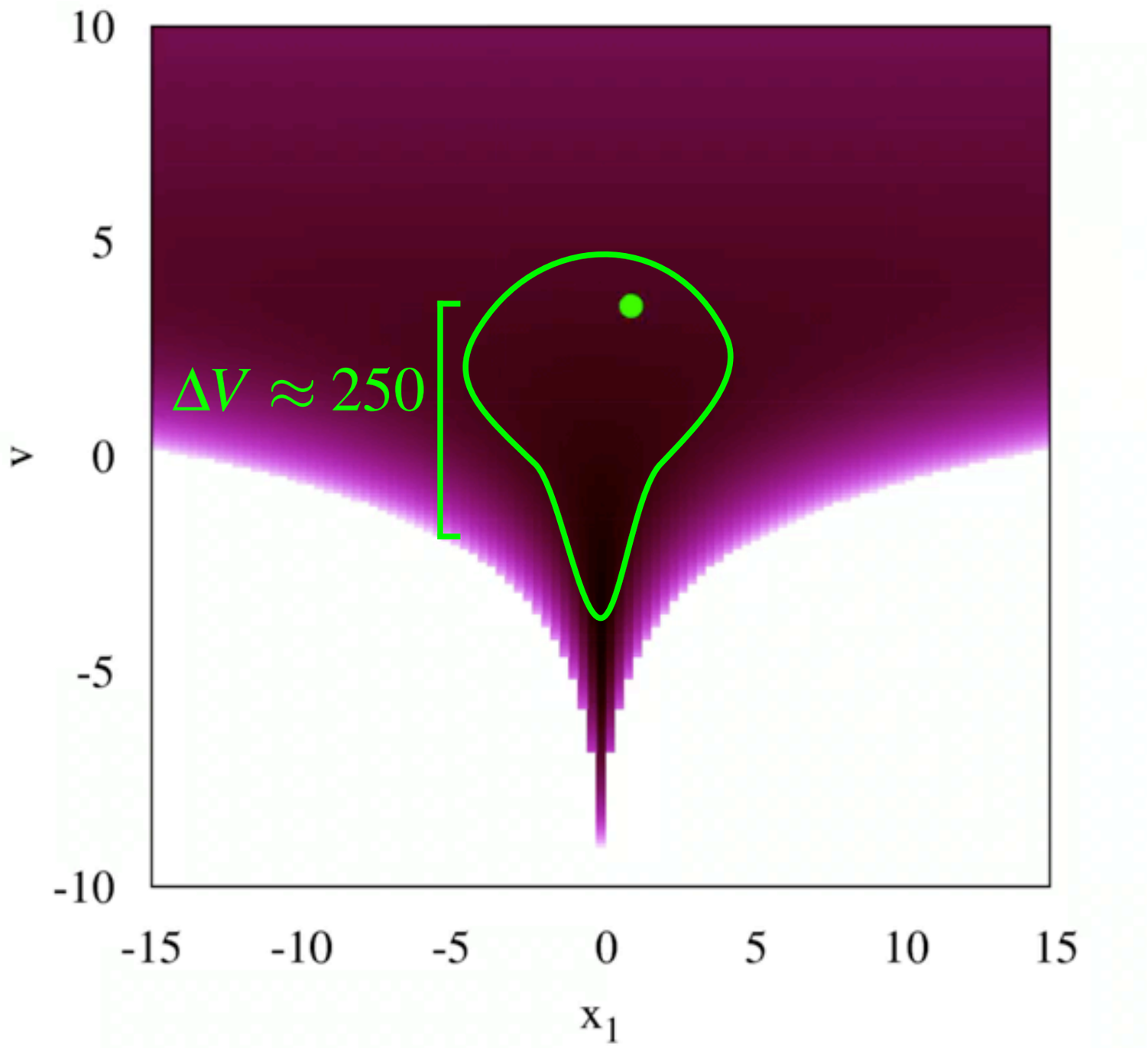
These weaknesses are particularly evident in hierarchical models



These weaknesses are particularly evident in hierarchical models



These weaknesses are particularly evident in hierarchical models

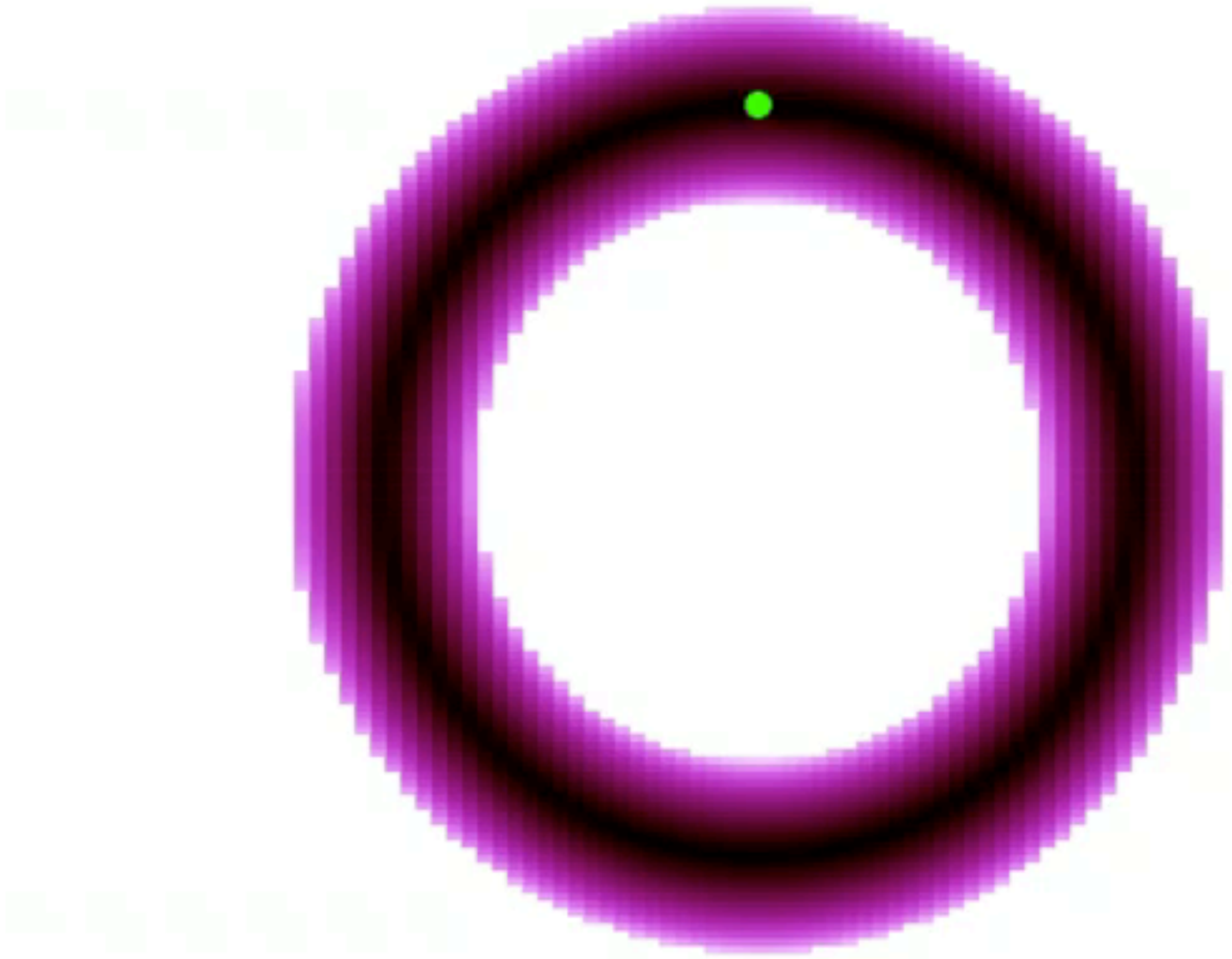


Quadratic kinetic energies with dynamic metrics emulate dynamics on a Riemannian manifold

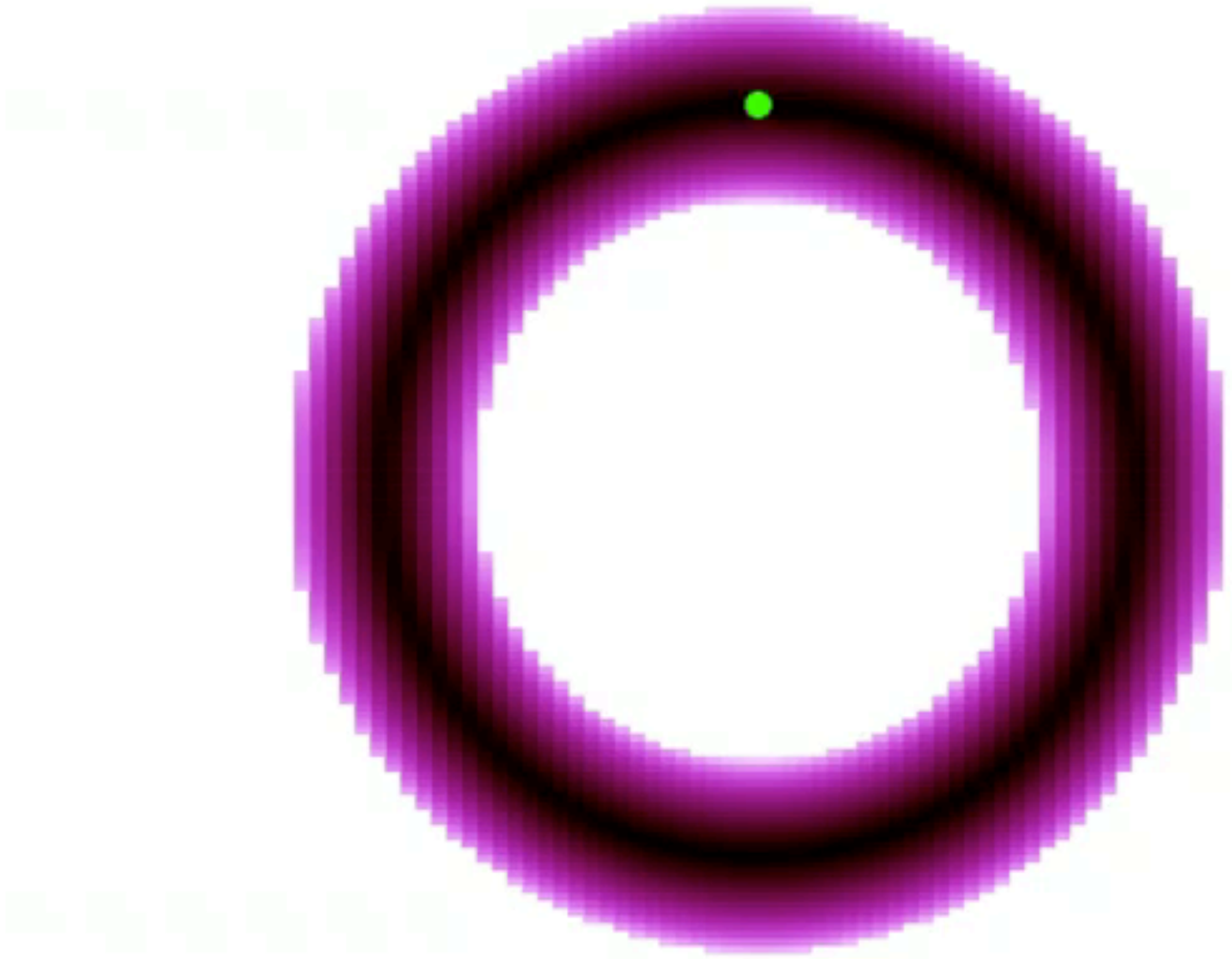
$$\pi(p|q) = \mathcal{N}(0, \Sigma(q))$$

$$T = \frac{1}{2} p_i p_j \left(\Sigma^{-1}(q)\right)^{ij} + \frac{1}{2} \log |\Sigma(q)|$$

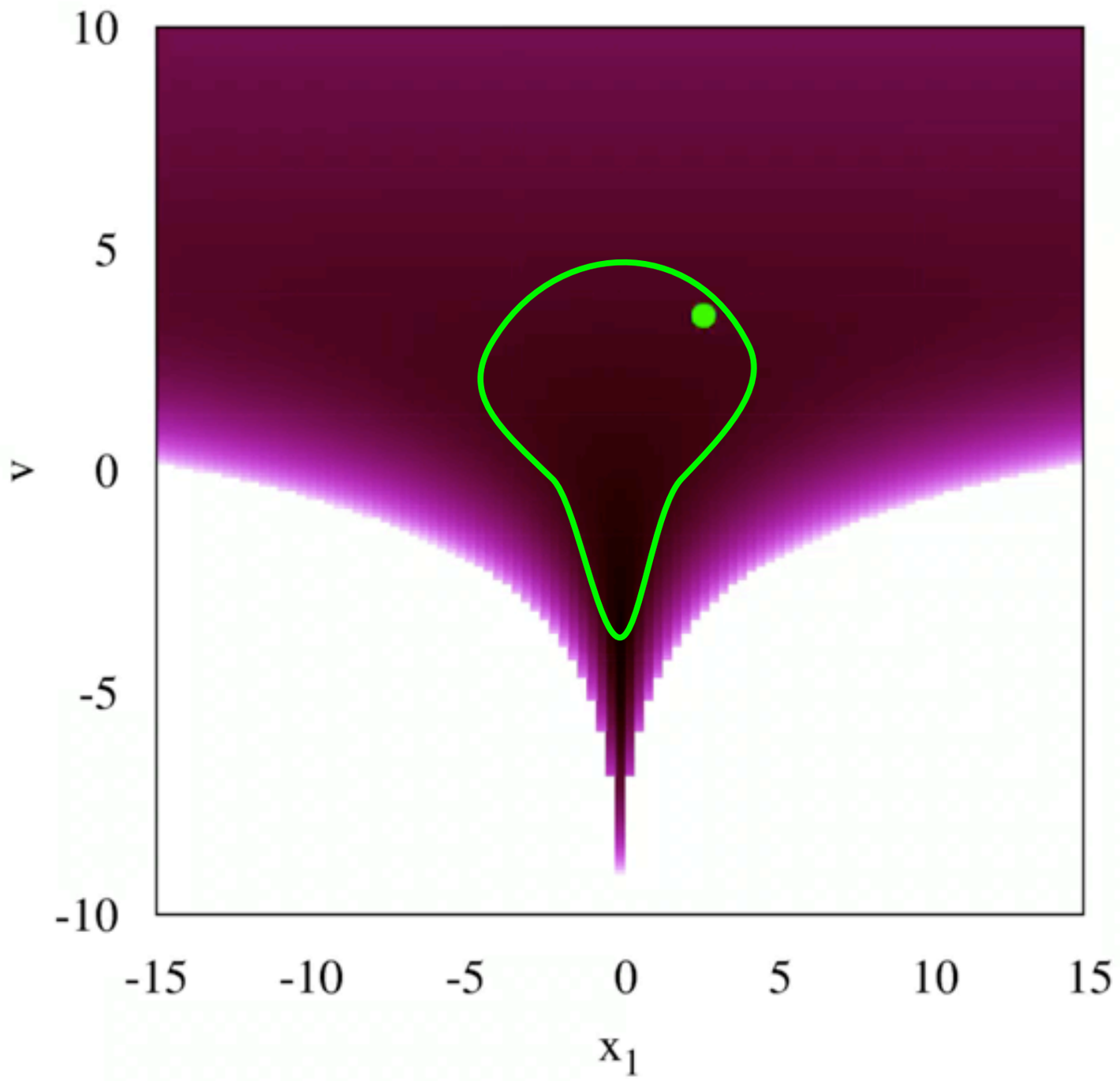
The Riemannian HMC locally standardizes the target distribution



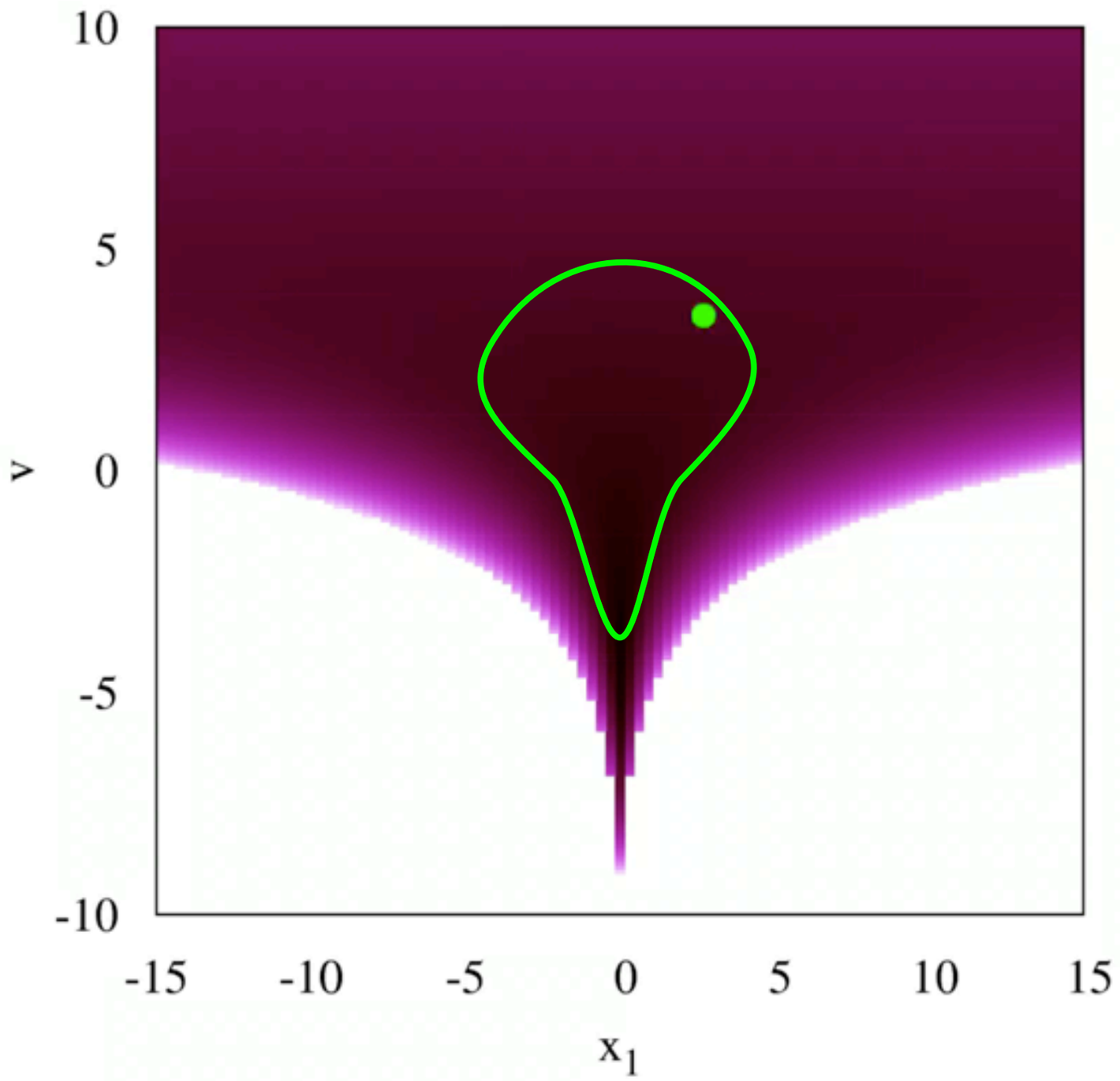
The Riemannian HMC locally standardizes the target distribution



And the log determinant admits full exploration of the funnel



And the log determinant admits full exploration of the funnel



Unfortunately, a naive implementation of HMC requires significant user input

$$\frac{dq}{dt} = +M^{-1}p$$

$$\frac{dp}{dt} = -\frac{\partial V}{\partial q}$$

Unfortunately, a naive implementation of HMC requires significant user input

$$q \to q + \epsilon M^{-1} p$$

$$p \to p - \epsilon \frac{\partial V}{\partial q}$$

Unfortunately, a naive implementation of HMC requires significant user input

$$q \to q + \epsilon M^{-1} p$$

$$p \to p - \epsilon \frac{\partial V}{\partial q}$$

$$\pi(\text{accept}) = \min\left(1, \frac{\pi(\Phi_\tau(p,q))}{\pi(p,q)}\right)$$

Unfortunately, a naive implementation of HMC requires significant user input

$$q \to q + \epsilon M^{-1} p$$

$$p \to p - \epsilon \frac{\partial V}{\partial q}$$

$$\pi(\text{accept}) = \min\left(1, \frac{\pi(\Phi_\tau(p,q))}{\pi(p,q)}\right)$$

Unfortunately, a naive implementation of HMC requires significant user input

$$q \to q + \epsilon M^{-1} p$$

$$p \to p - \epsilon \frac{\partial V}{\partial q}$$

$$\pi(\text{accept}) = \min\left(1, \frac{\pi(\Phi_\tau(p,q))}{\pi(p,q)}\right)$$

Unfortunately, a naive implementation of HMC requires significant user input

$$q \to q + \epsilon M^{-1} p$$

$$p \to p - \epsilon \frac{\partial V}{\partial q}$$

$$\pi(\text{accept}) = \min\left(1, \frac{\pi(\Phi_\tau(p,q))}{\pi(p,q)}\right)$$

Unfortunately, a naive implementation of HMC requires significant user input

$$q \to q + \epsilon M^{-1} p$$

$$p \to p - \epsilon \frac{\partial V}{\partial q}$$

$$\pi(\text{accept}) = \min\left(1, \frac{\pi(\Phi_\tau(p,q))}{\pi(p,q)}\right)$$

Unfortunately, a naive implementation of HMC requires significant user input

$$q \to q + \epsilon M^{-1} p$$

$$p \to p - \epsilon \frac{\partial V}{\partial q}$$

$$\pi(\text{accept}) = \min\left(1, \frac{\pi(\Phi_\tau(p,q))}{\pi(p,q)}\right)$$

Unfortunately, a naive implementation of HMC requires significant user input

$$q \to q + \epsilon M^{-1} p$$

$$p \to p - \epsilon \frac{\partial V}{\partial q}$$

$$\pi(\text{accept}) = \min\left(1, \frac{\pi(\Phi_{\tau}(p, q))}{\pi(p, q)}\right)$$

Stan

Hamiltonian Monte Carlo

Modeling Language

Automatic Differentiation

Adaptation

Stan

Hamiltonian Monte Carlo

Modeling Language

Automatic Differentiation

Adaptation

A strongly typed modeling language allows users to specify complex models with minimal effort

```
data {
  int<lower=1> N;
  real x[N];
}
transformed data {
  vector[N] mu;
  cov_matrix[N] Sigma;
  for (i in 1:N)
    mu[i] <- 0;
  for (i in 1:N)
    for (j in 1:N)
      Sigma[i,j] <- exp(-pow(x[i] - x[j],2))
                    + if_else(i==j, 0.1, 0.0);
}
parameters {
  vector[N] y;
}
```

Stan

Hamiltonian Monte Carlo

Modeling Language

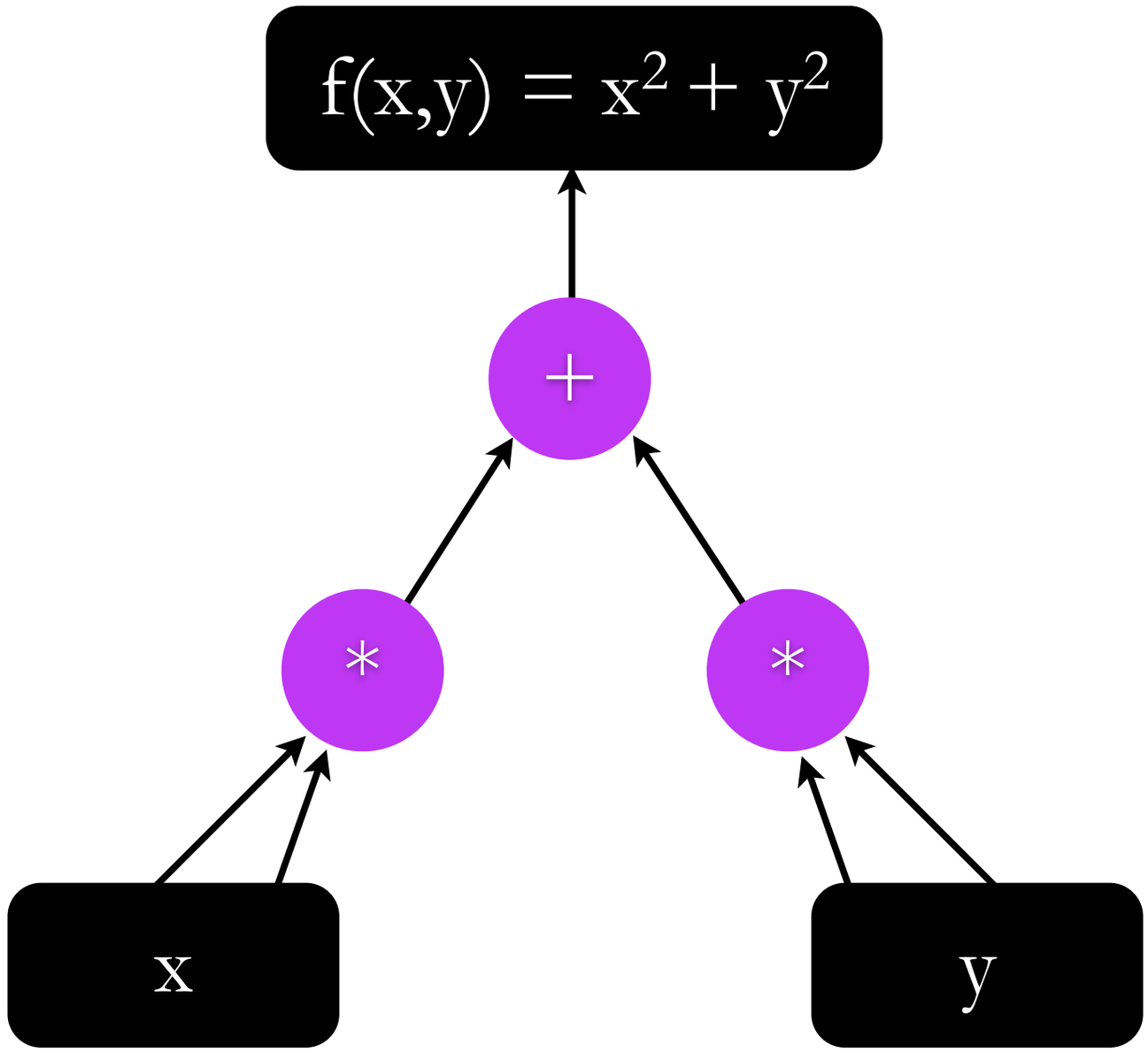
Automatic Differentiation

Adaptation

Automatic differentiation enables efficient, exact computation of the necessary gradients

$$f(x,y) = x^2 + y^2$$

Automatic differentiation enables efficient, exact computation of the necessary gradients



Stan

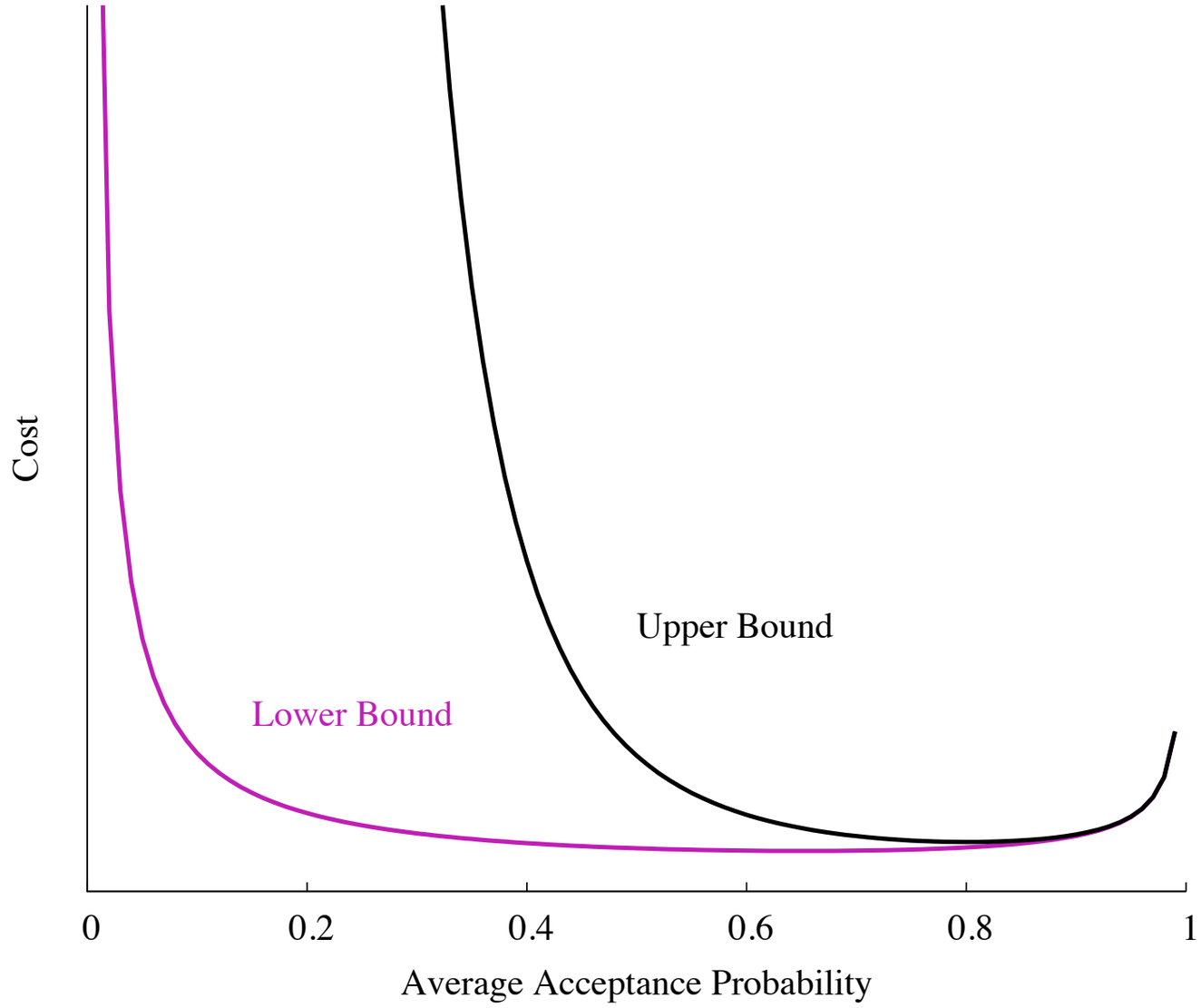
Hamiltonian Monte Carlo

Modeling Language

Automatic Differentiation

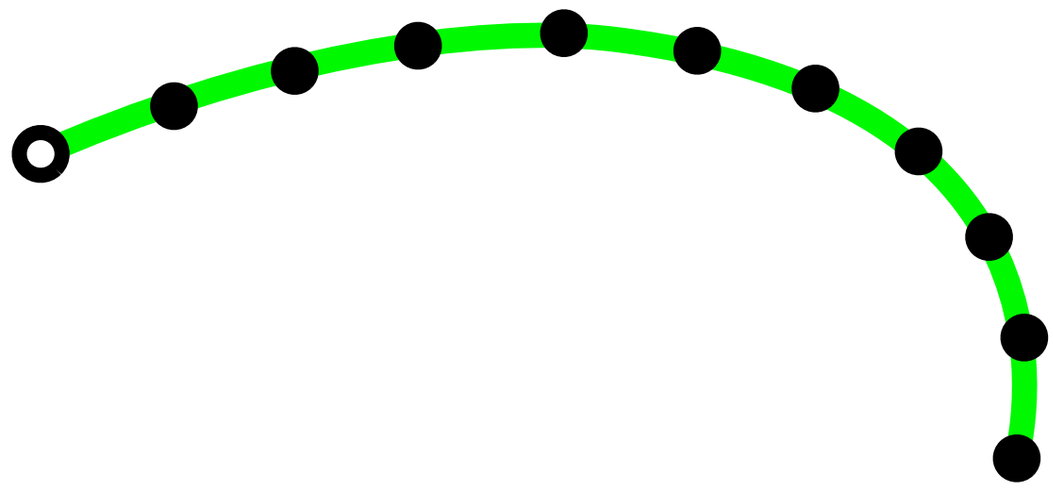
Adaptation

Free parameters, such as the step size,
can be adapted to each target distribution

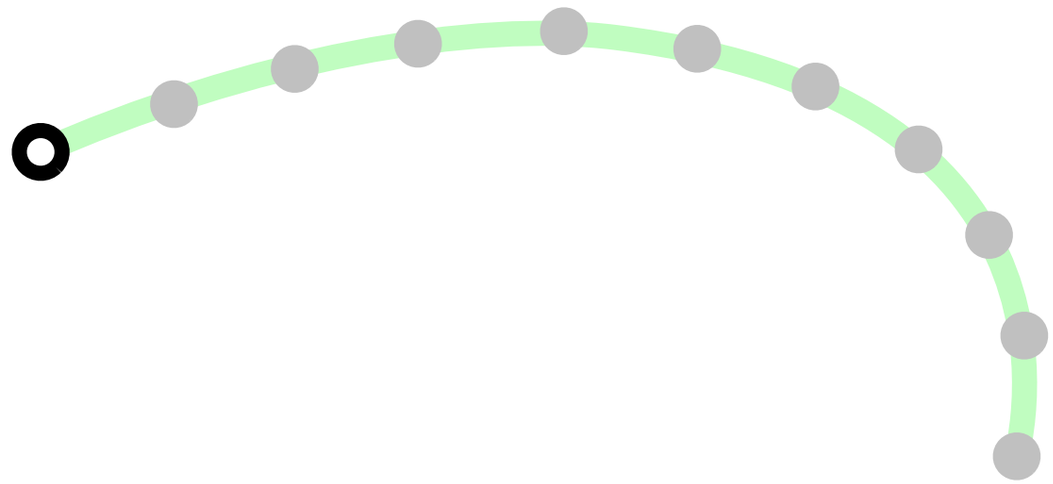


We can also adapt the integration time
using the No-U-Turn Sampler

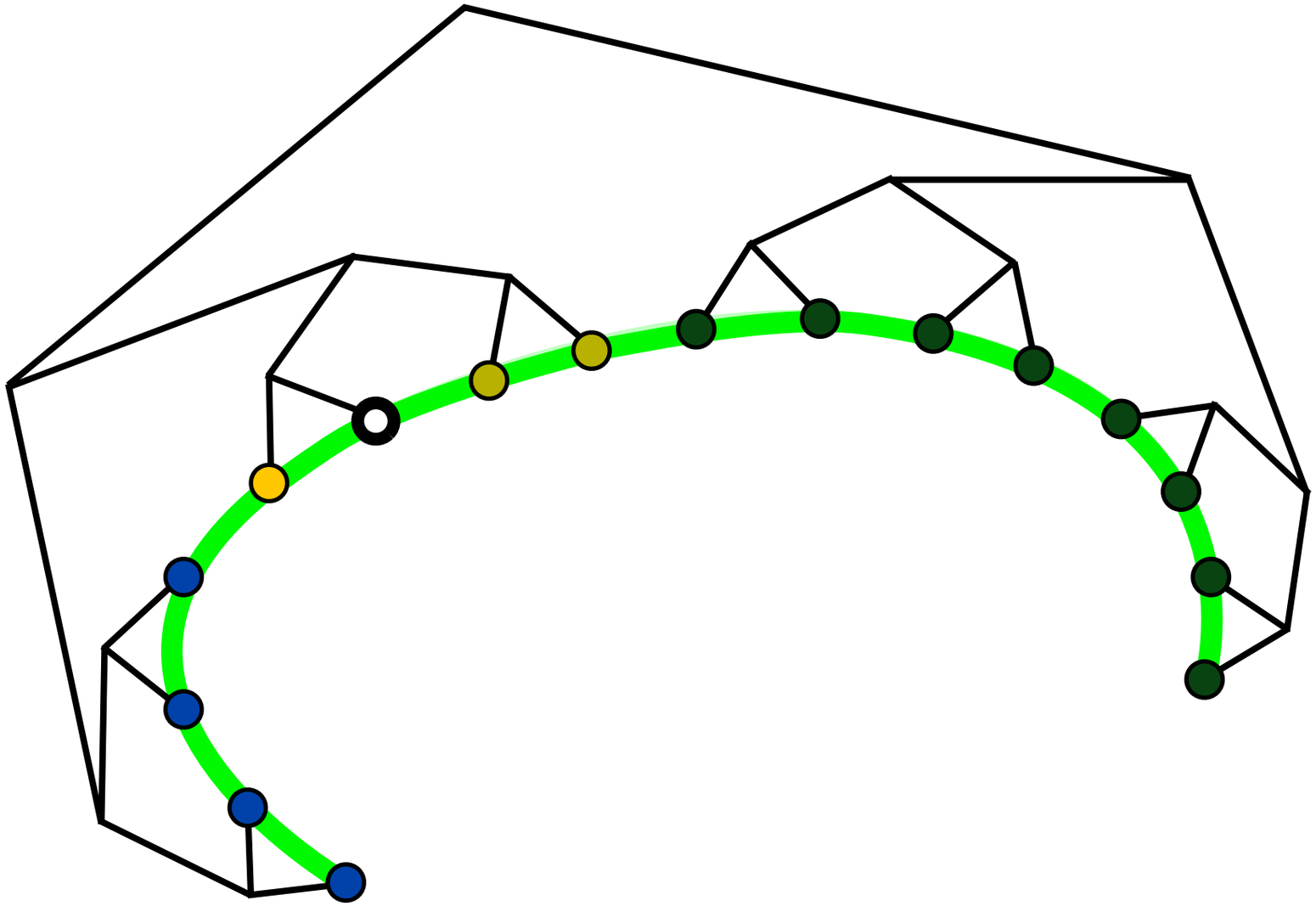
We can also adapt the integration time
using the No-U-Turn Sampler



We can also adapt the integration time
using the No-U-Turn Sampler



We can also adapt the integration time
using the No-U-Turn Sampler



The Stan user community is active and rapidly growing, coming from such diverse fields as

Influenza Epidemiology
Political Science / International Relations
Demography / Sociology
Cardiovascular and Substance-Abuse Epidemiology
Evolutionary Biology
Neuropsychopharmacology / Psychophysiology
Fish Population Dynamics
Evolutionary Anthropology
Exoplanet Astrophysics

Stan

Hamiltonian Monte Carlo

Modeling Language

Automatic Differentiation

Adaptation

Backups

Optimal numerical integration suggests using the Hessian, but the Hessian isn't positive-definite

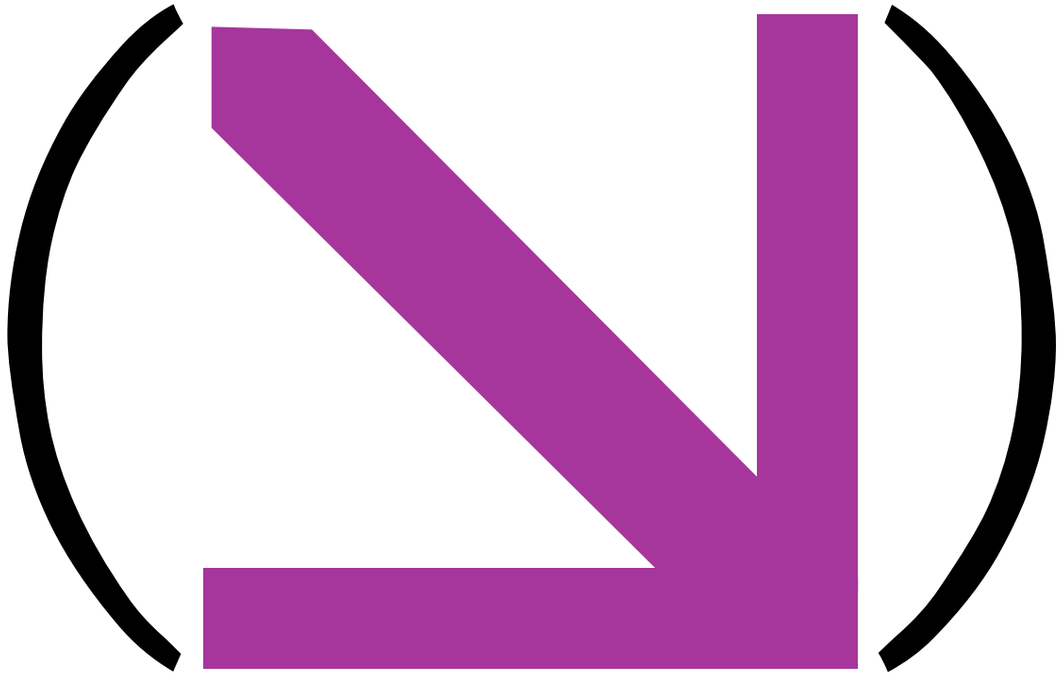
$$\Sigma(q)_{ij} \neq \partial_i \partial_j V(q)$$

Fisher-Rao is both impractical and ineffective

$$\Sigma(q)_{ij} = \mathbb{E}_{\mathcal{D}}\left[\partial_i \partial_j V(q|\mathcal{D})\right]$$

Fisher-Rao is both impractical and ineffective

$$\Sigma(q)_{ij} = \mathbb{E}_{\mathcal{D}}\left[\partial_i \partial_j V(q|\mathcal{D})\right]$$



$$\partial_i \partial_j V(q|\mathcal{D})$$

Fisher-Rao is both impractical and ineffective

$$\Sigma(q)_{ij} = \mathbb{E}_{\mathcal{D}}\left[\partial_i \partial_j V(q|\mathcal{D})\right]$$

$$\mathbb{E}_{\mathcal{D}}\left[\partial_i \partial_j V(q|\mathcal{D})\right]$$

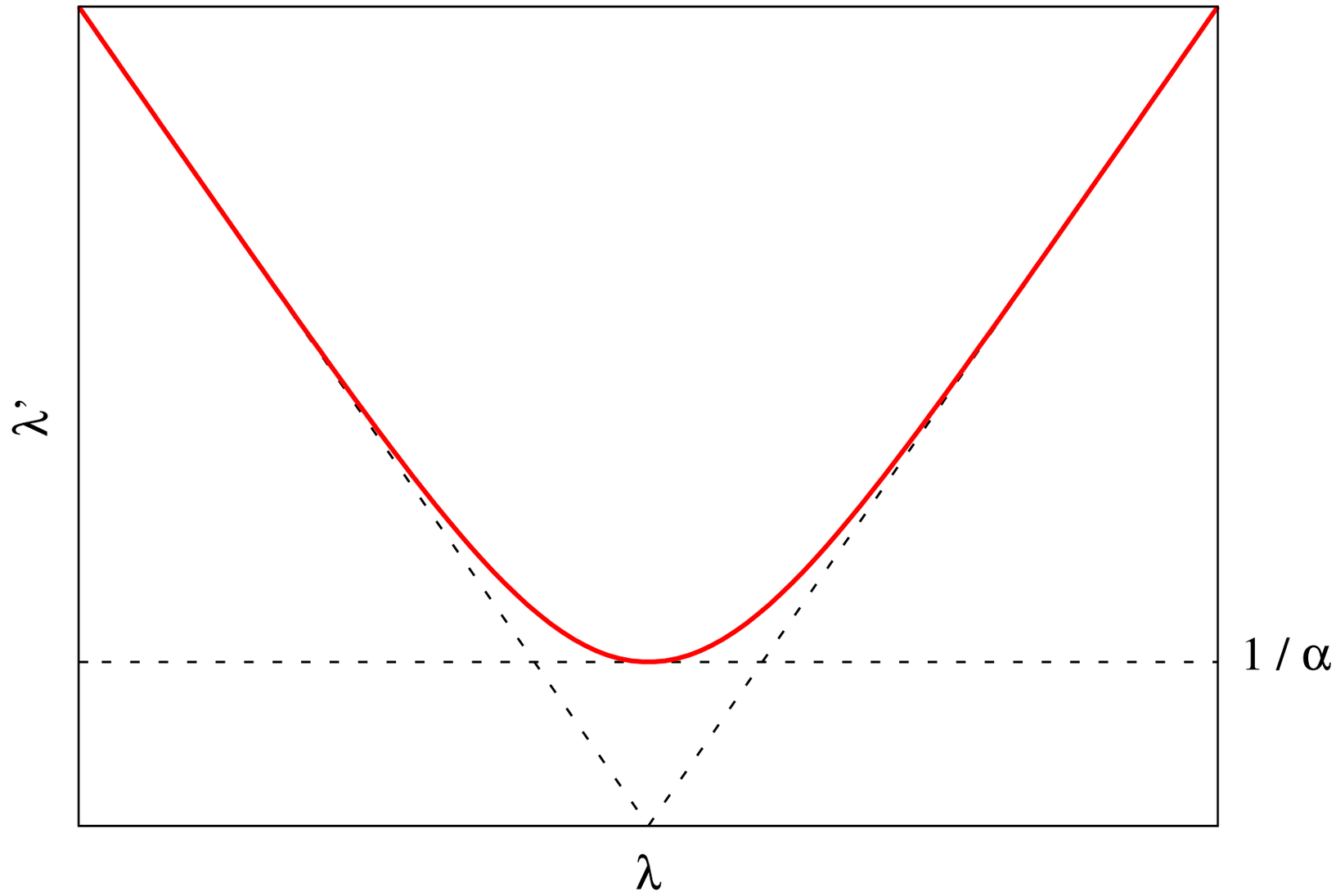
We can regularize without appealing to expectations

$$\Sigma_{ij}(q) = [\exp(\alpha H_{ik}) + \exp(-\alpha H_{ik})]$$

$$\cdot H_{kl} \cdot$$

$$[\exp(\alpha H_{lj}) - \exp(-\alpha H_{lj})]^{-1}$$

The “SoftAbs” metric serves as a differentiable absolute value of the Hessian



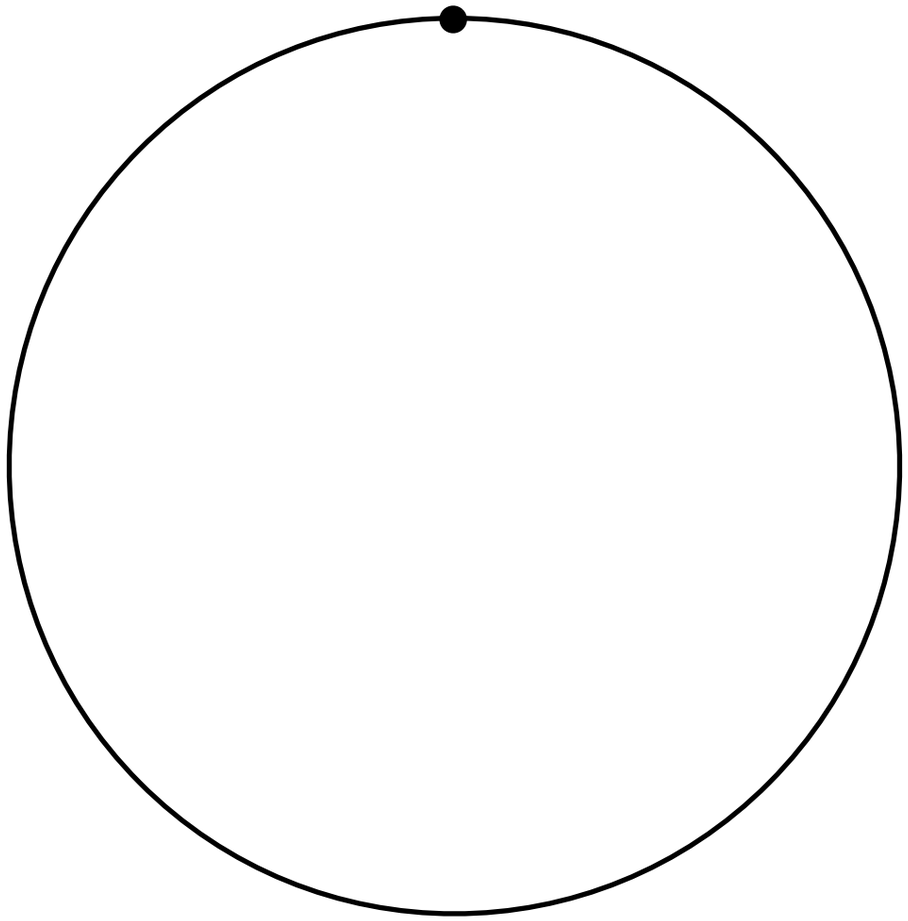
Free parameters, such as the step size,
can be adapted to each target distribution

$$\pi(\text{accept}) = \min\left(1, \frac{\pi(\phi_\tau(p,q))}{\pi(p,q)}\right)$$

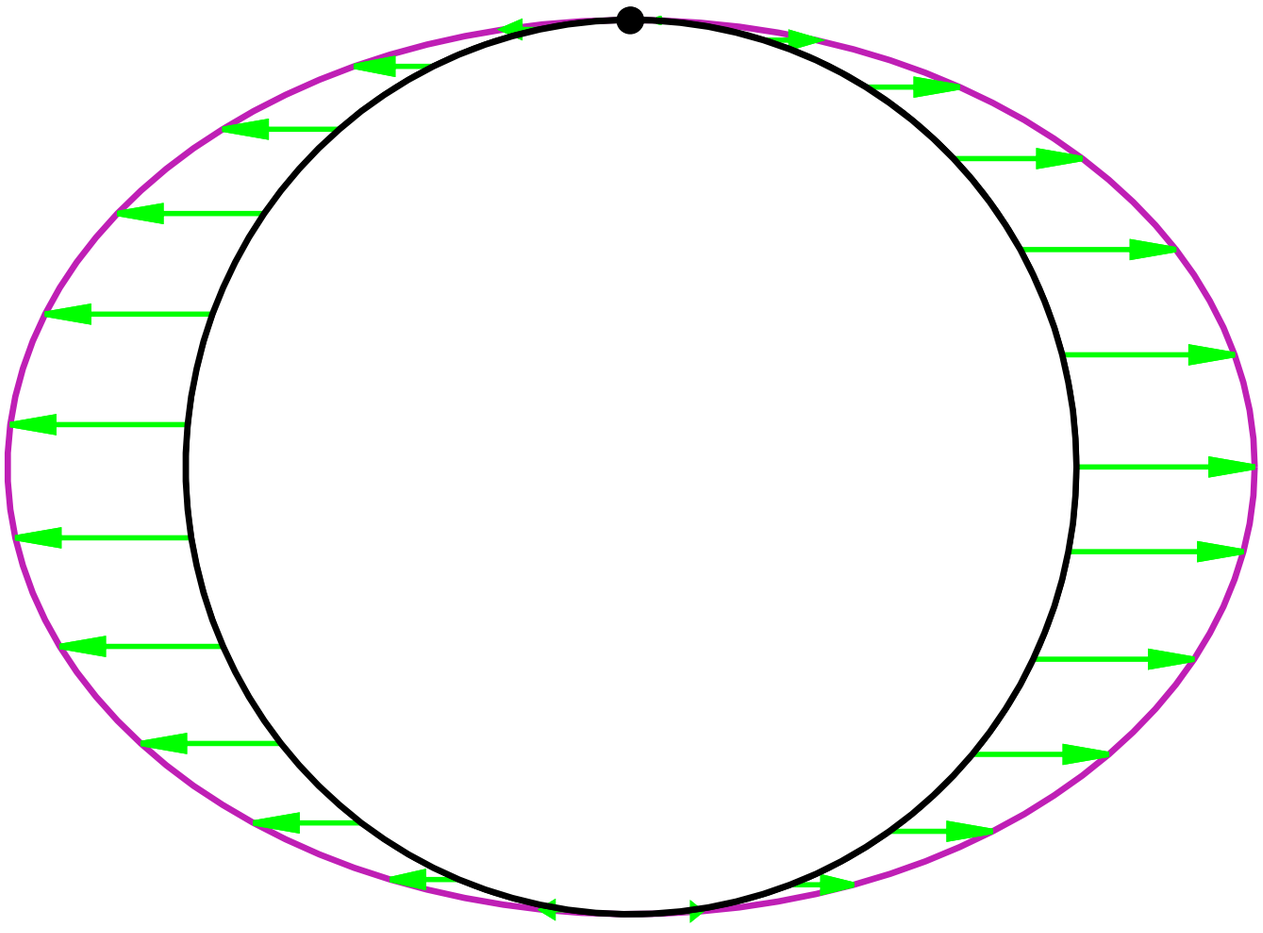
Free parameters, such as the step size,
can be adapted to each target distribution

$$\pi(\text{accept}) = \min\left(1, e^{H(p,q) - H(\phi_\tau(p,q))}\right)$$

Free parameters, such as the step size,
can be adapted to each target distribution



Free parameters, such as the step size,
can be adapted to each target distribution



Free parameters, such as the step size,
can be adapted to each target distribution