Monte Carlo Methods for Inference and Learning

Ryan Adams
University of Toronto

CIFAR NCAP Summer School
14 August 2010

http://www.cs.toronto.edu/~rpa

Thanks to: Iain Murray,
Marc'Aurelio Ranzato

CIFAR

Overview

- Monte Carlo basics
- Rejection and Importance sampling
- Markov chain Monte Carlo
- Metropolis-Hastings and Gibbs sampling
- Practical issues
- Slice sampling
- Hamiltonian Monte Carlo

Computing Expectations

We often like to use probabilistic models for data.

$$\text{data } v, \text{parameters } \theta$$

$$v \sim p(v \,|\, \theta)$$

$$\theta \,|\, v \sim p(\theta \,|\, v) = \frac{p(\theta \,|\, v)\, p(\theta)}{p(v)}$$

What is the mean of the posterior?

$$\hat{\theta} = \int p(\theta \,|\, v)\, \theta \, \mathrm{d}\theta$$

Computing Expectations

What is the predictive distribution?

$$p(v' \,|\, v) = \int p(v' \,|\, \theta)\, p(\theta \,|\, v)\, \mathrm{d}\theta$$

What is the marginal (integrated) likelihood?

$$p(v \,|\, \mathcal{H}) = \int p(v \,|\, \theta, \mathcal{H})\, p(\theta \,|\, \mathcal{H})\, \mathrm{d}\theta$$

Computing Expectations

Sometimes we prefer latent variable models.

$$\text{data } v, \text{parameters } \theta, \text{latent variables } h$$

Sometimes these joint models are intractable.

$$v, h \sim p(v, h \,|\, \theta) = \frac{1}{\mathcal{Z}}\, e^{-E(v,h\,;\,\theta)}$$

Maximize the marginal probability of data

$$\theta^\star = \arg\max_\theta \int p(v, h \,|\, \theta)\, \mathrm{d}h$$

The Monte Carlo Principle

Each of these examples has a shared form:

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

Any such expectation can be computed from samples:

$$\int f(x)\, \pi(x)\, \mathrm{d}x \;\approx\; \frac{1}{S}\sum_{s=1}^{S} f(x^{(s)}) \;\;\text{where}\;\; x^{(s)} \sim \pi(x)$$

The Monte Carlo Principle

Example: Computing a Bayesian predictive distribution

$$p(v' \,|\, v) = \int p(v' \,|\, \theta)\, p(\theta \,|\, v)\, \mathrm{d}\theta$$

$$x = \theta, \;\; \pi(x) = p(\theta \,|\, v), \;\; f(x) = p(v' \,|\, \theta)$$

We get a predictive mixture distribution:

$$p(v' \,|\, v) \;\approx\; \frac{1}{S} \sum_{s=1}^{S} p(v' \,|\, \theta^{(s)}) \;\; \text{where} \;\; \theta^{(s)} \sim p(\theta \,|\, v)$$

Properties of MC Estimators

$$\int f(x)\,\pi(x)\,\mathrm{d}x \approx \hat{f} \equiv \frac{1}{S}\sum_{s=1}^{S} f(x^{(s)}), \quad \text{where } x^{(s)} \sim \pi(x)$$

Monte Carlo estimates are unbiased.

$$\mathbb{E}_{\pi(\{x^{(s)}\})}[\,\hat{f}\,] = \frac{1}{S}\sum_{s=1}^{S}\mathbb{E}_{\pi(x)}[\,f(x)\,] = \mathbb{E}_{\pi(x)}[\,f(x)\,]$$

The variance of the estimator shrinks as $1/S$

$$\mathrm{Var}_{\pi(\{x^{(s)}\})}[\,\hat{f}\,] = \frac{1}{S^2}\sum_{s=1}^{S}\mathrm{Var}_{\pi(x)}[\,f(x)\,] = \mathrm{Var}_{\pi(x)}[\,f(x)\,]/S$$

The "error" of the estimator shrinks as $1/\sqrt{S}$

Why Monte Carlo?

"Monte Carlo is an extremely bad method; it should be used only when all alternative methods are worse."

Alan Sokal

Monte Carlo methods in statistical mechanics, 1996

The error is only shrinking as $1/\sqrt{S}$?!?!? Isn't that bad?

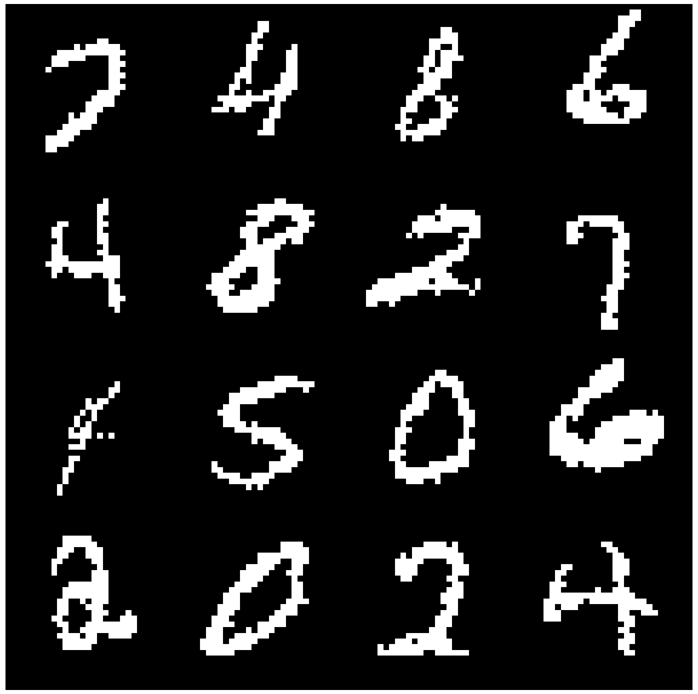
Heck, Simpson's Rule gives $1/S^{4/D}$!!!

How many dimensions do you have?

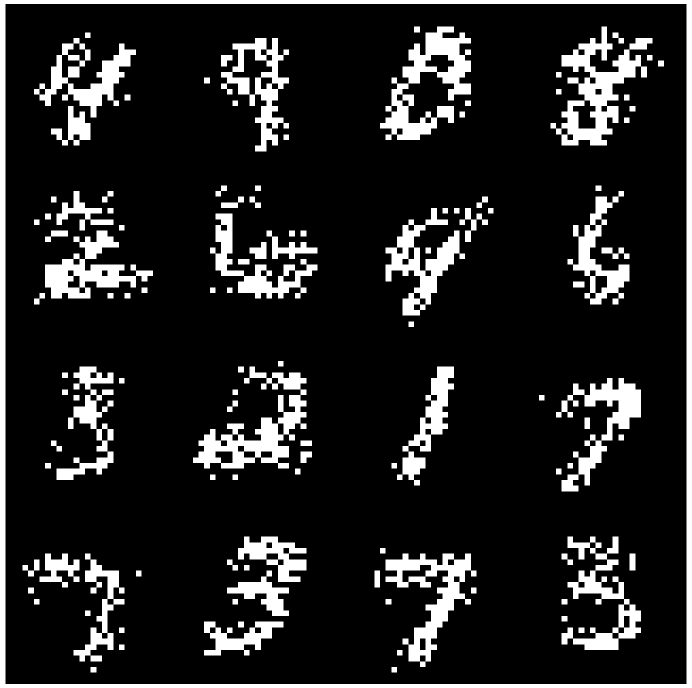
Why Monte Carlo?

If we have a generative model, we can *fantasize* data.

This helps us understand the properties of our model and know what we're learning from the true data.



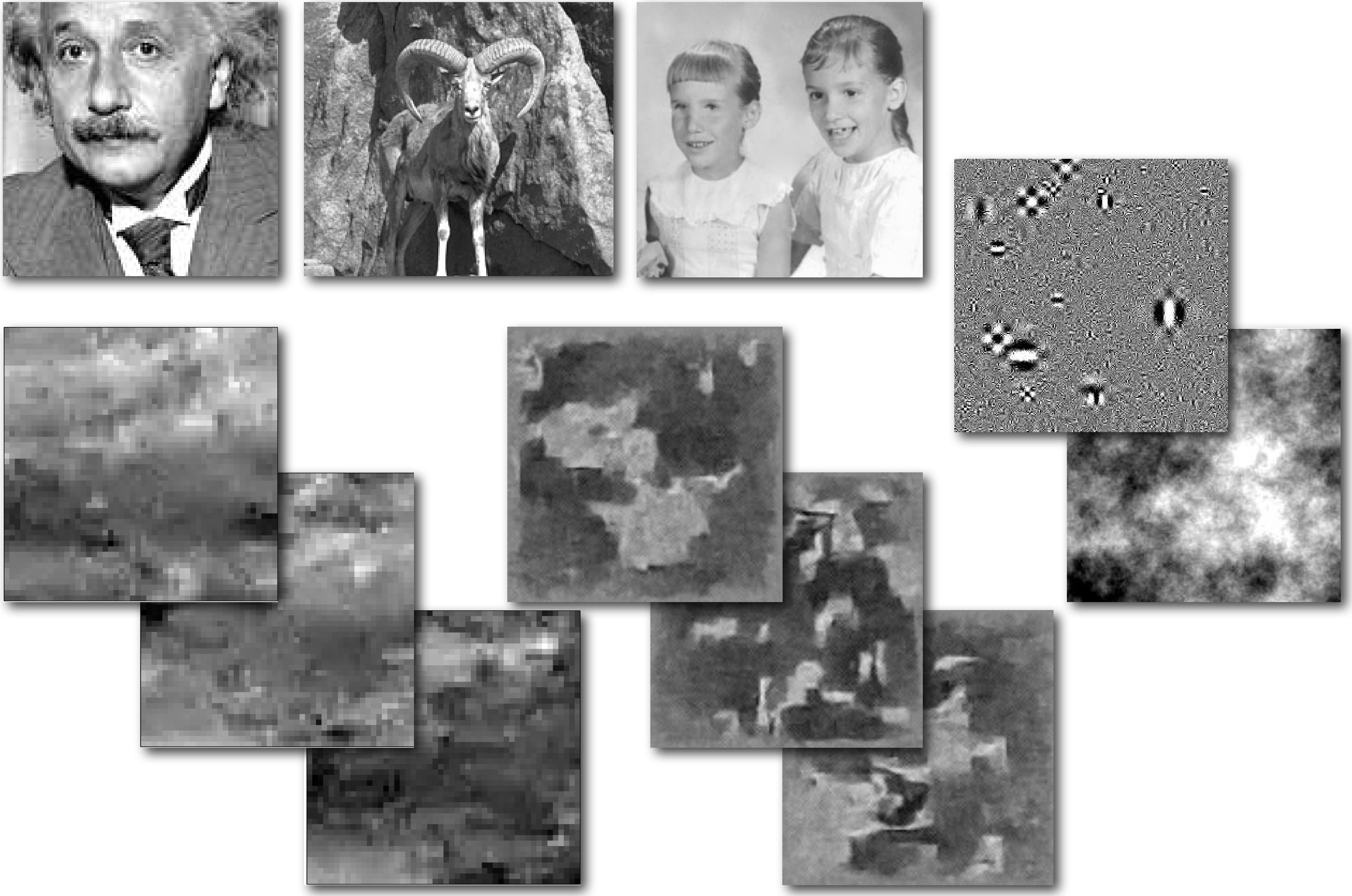
Data samples



MoB samples

RBM samples

Generating Fantasy Data



Sampling Basics

$$\int f(x)\,\pi(x)\,\mathrm{d}x \;\approx\; \frac{1}{S}\sum_{s=1}^{S} f(x^{(s)}) \;\;\text{where}\;\; x^{(s)} \sim \pi(x)$$

We need samples from $\pi(x)$. How to get them?

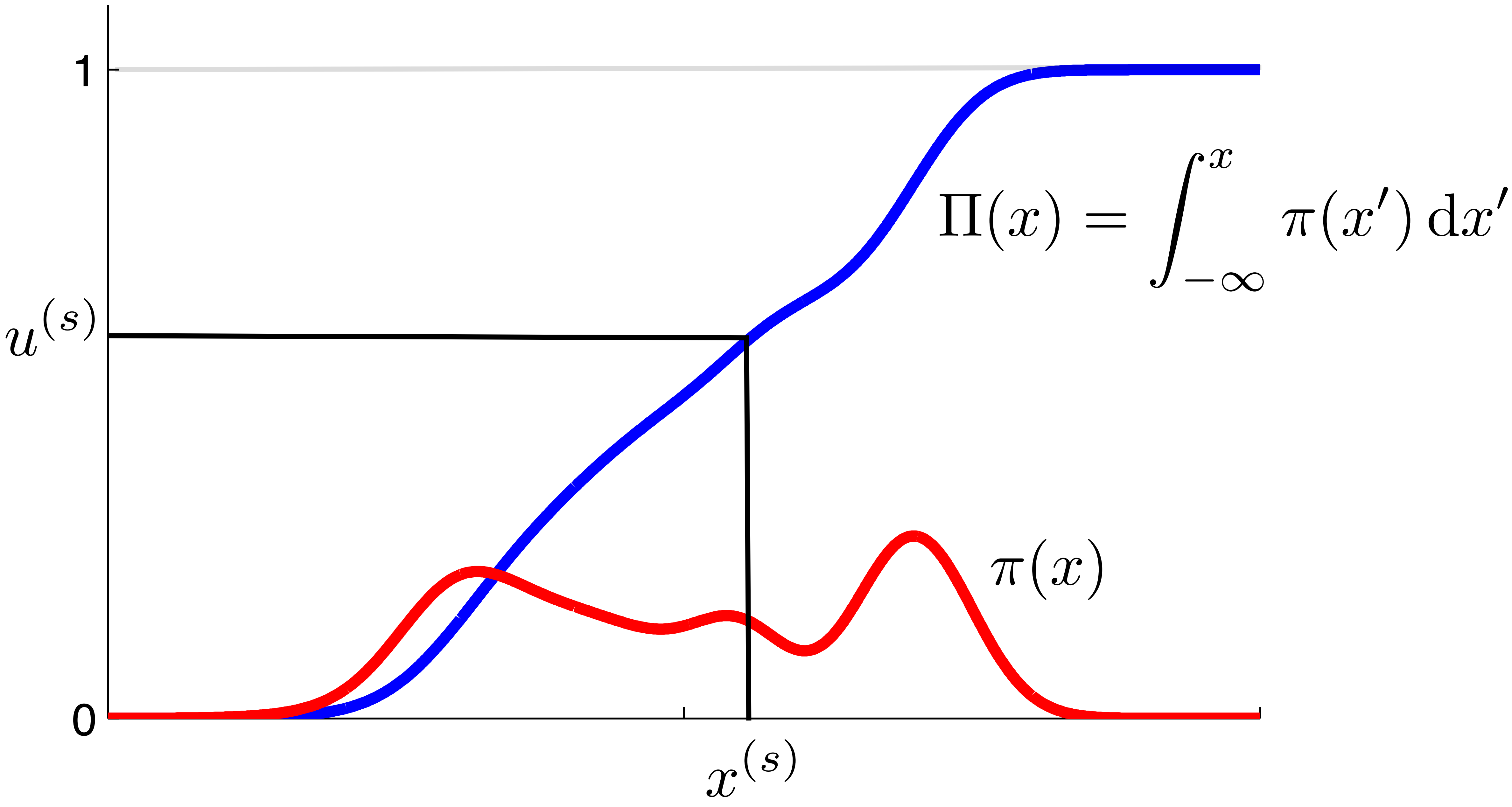
Most generally, your pseudo-random number generator is going to give you a sequence of integers from large range.

These you can easily turn into floats in $[0,1]$.

Probably you just call `rand()` in Matlab or Numpy.

Your $\pi(x)$ is probably more interesting than this.

Inversion Sampling



$u^{(s)} \sim \text{Uniform}(0, 1)$

$x^{(s)} = \Pi^{-1}(u^{(s)})$

Inversion Sampling

Good News:

Straightforward way to take your uniform (0,1) variate and turn it into something complicated.

Bad News:

We still had to do an integral.

Doesn't generalize easily to multiple dimensions.

The distribution had to be normalized.

The Big Picture

So, if generating samples is just as difficult as integration, what's the point of all this Monte Carlo stuff?

This entire tutorial is about the following idea:

Take samples from some simpler distribution $q(x)$ and turn them into samples from the complicated thing that we're actually interested in, $\pi(x)$.

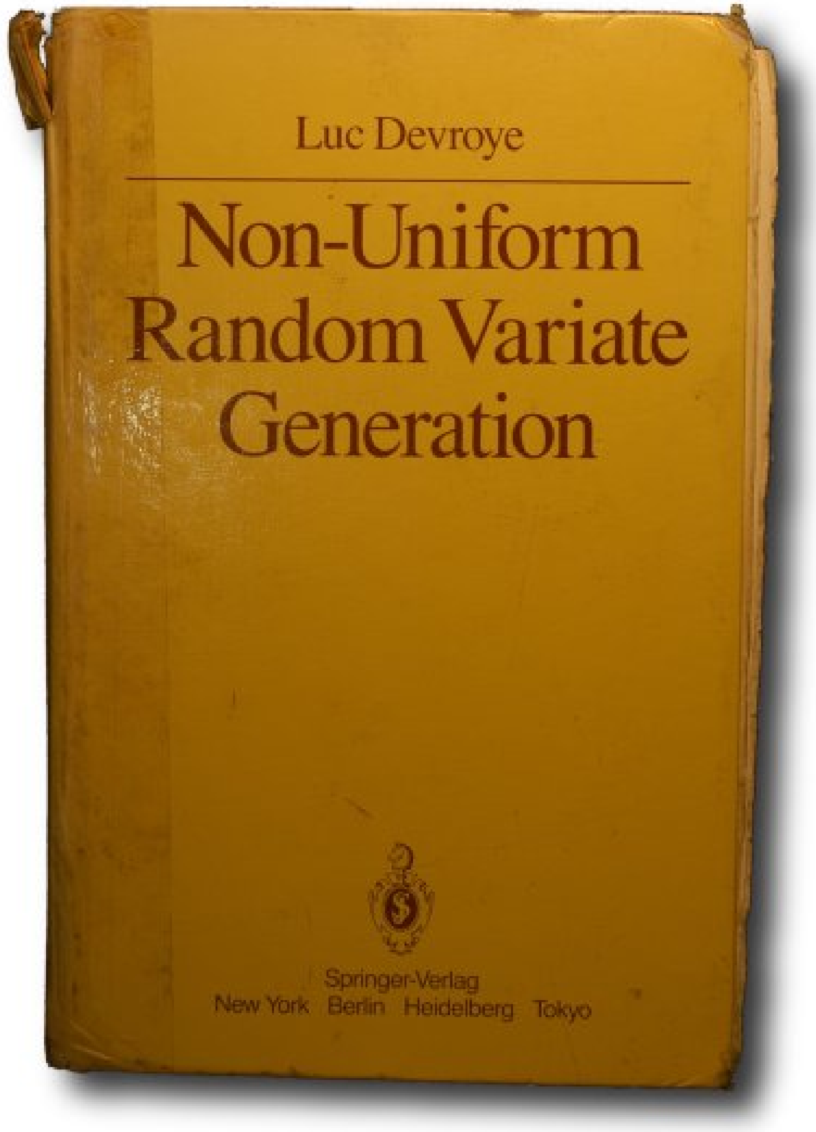
In general, I will assume that we only know $\pi(x)$ to within a constant and that we cannot integrate it.

Standard Random Variates

It’s worth pointing out that for lots of simple, standard univariate distributions, many tools will already exist for generating samples, e.g., `randn()`, `poissrnd()`, and `randg()` in Matlab for normal, Poisson and gamma distributions, respectively.

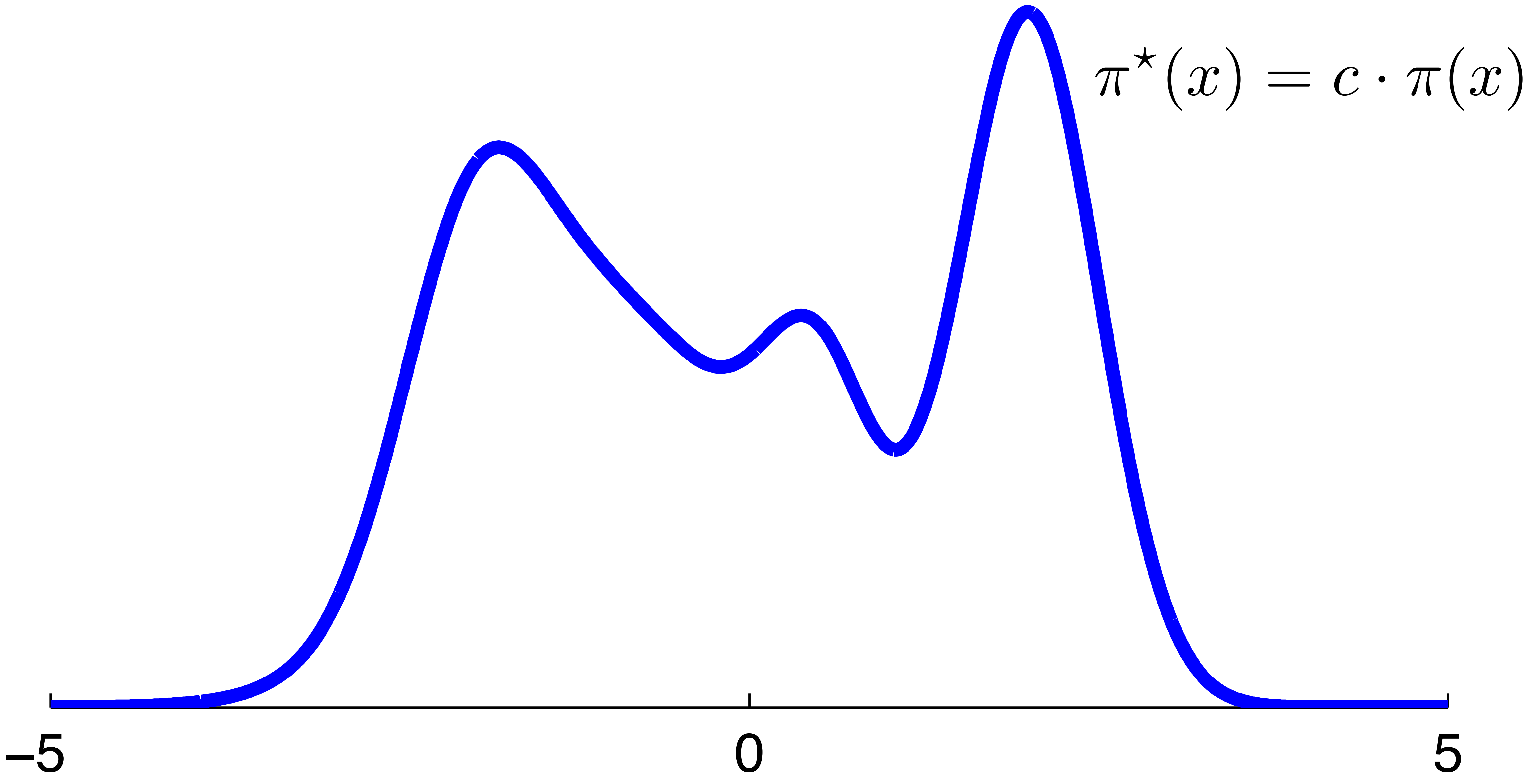
There is a great book online by Luc Devroye with recipes for lots of standard distributions.

http://cg.scs.carleton.ca/~luc/rnbookindex.html



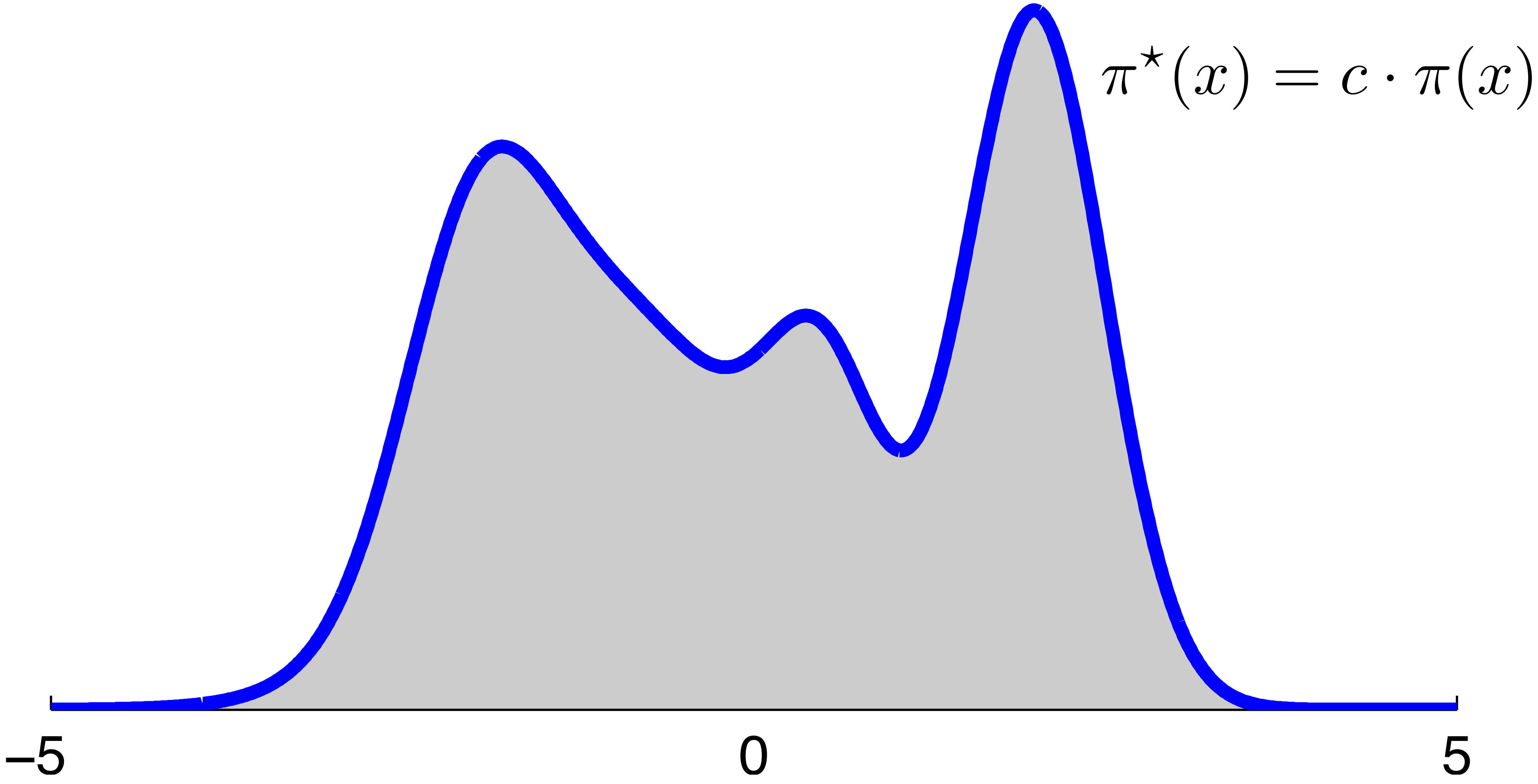
Rejection Sampling

One useful observation is that samples uniformly drawn from the volume beneath a (not necessarily normalized) PDF will have the correct marginal distribution.



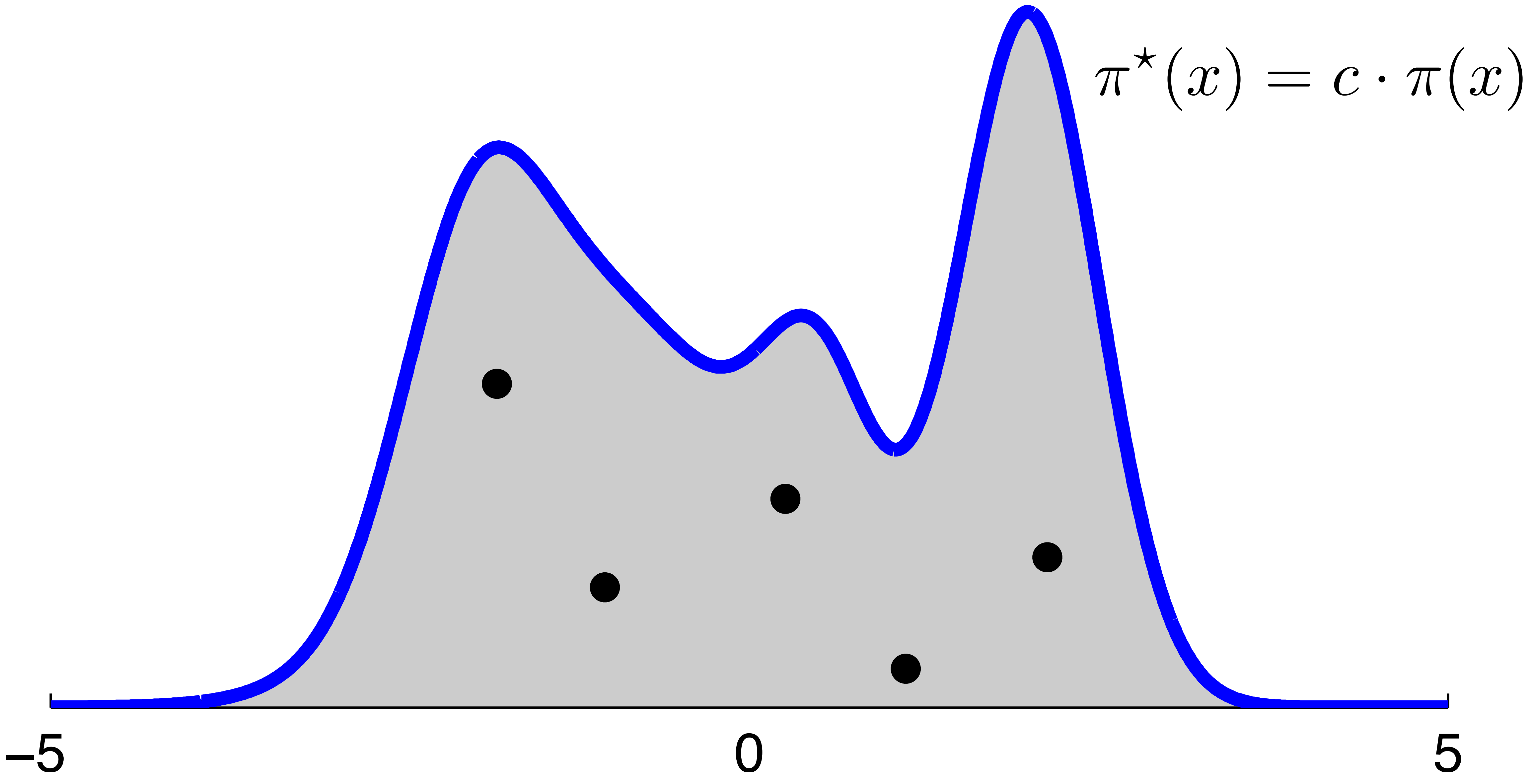
Rejection Sampling

One useful observation is that samples uniformly drawn from the volume beneath a (not necessarily normalized) PDF will have the correct marginal distribution.



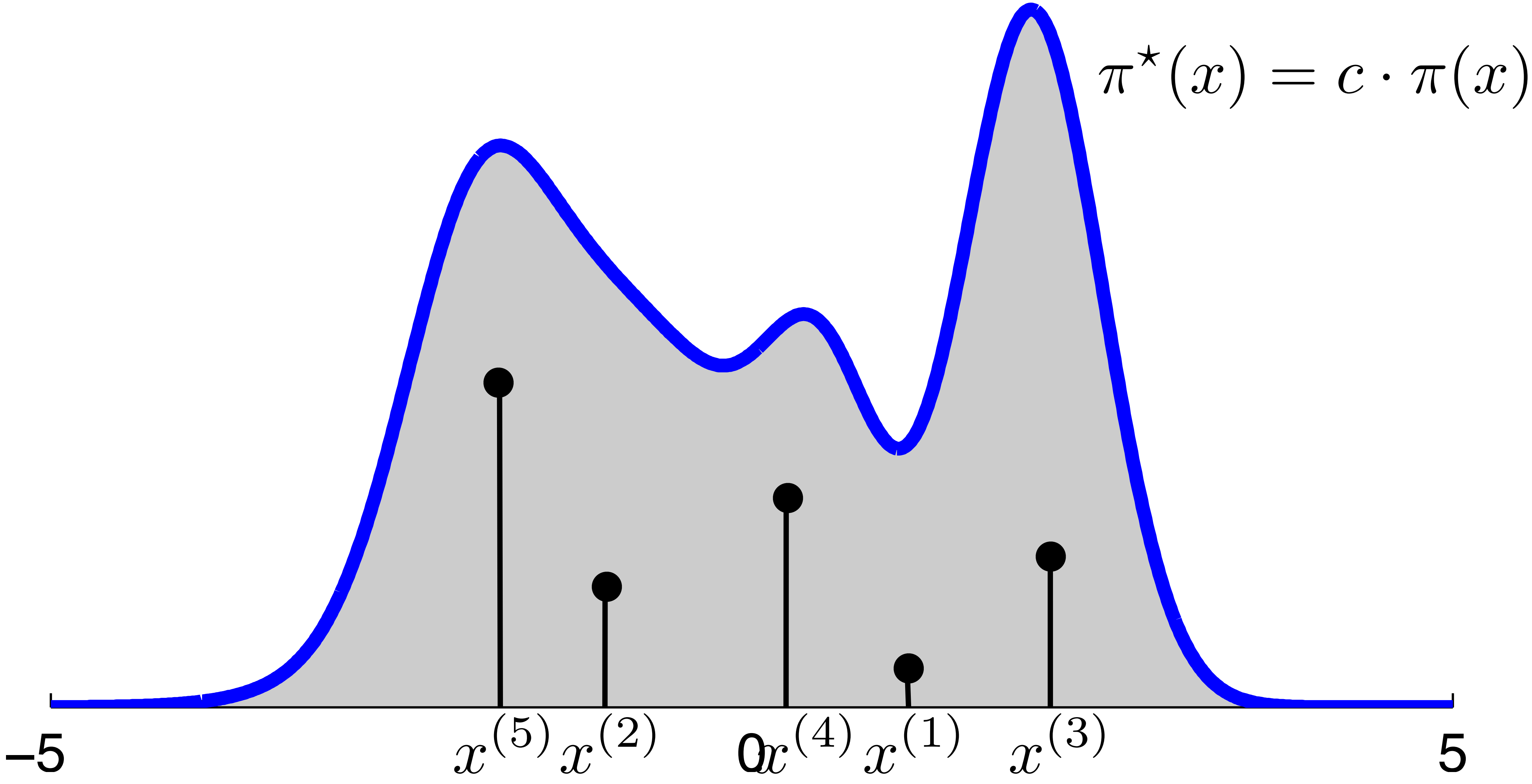
Rejection Sampling

One useful observation is that samples uniformly drawn from the volume beneath a (not necessarily normalized) PDF will have the correct marginal distribution.



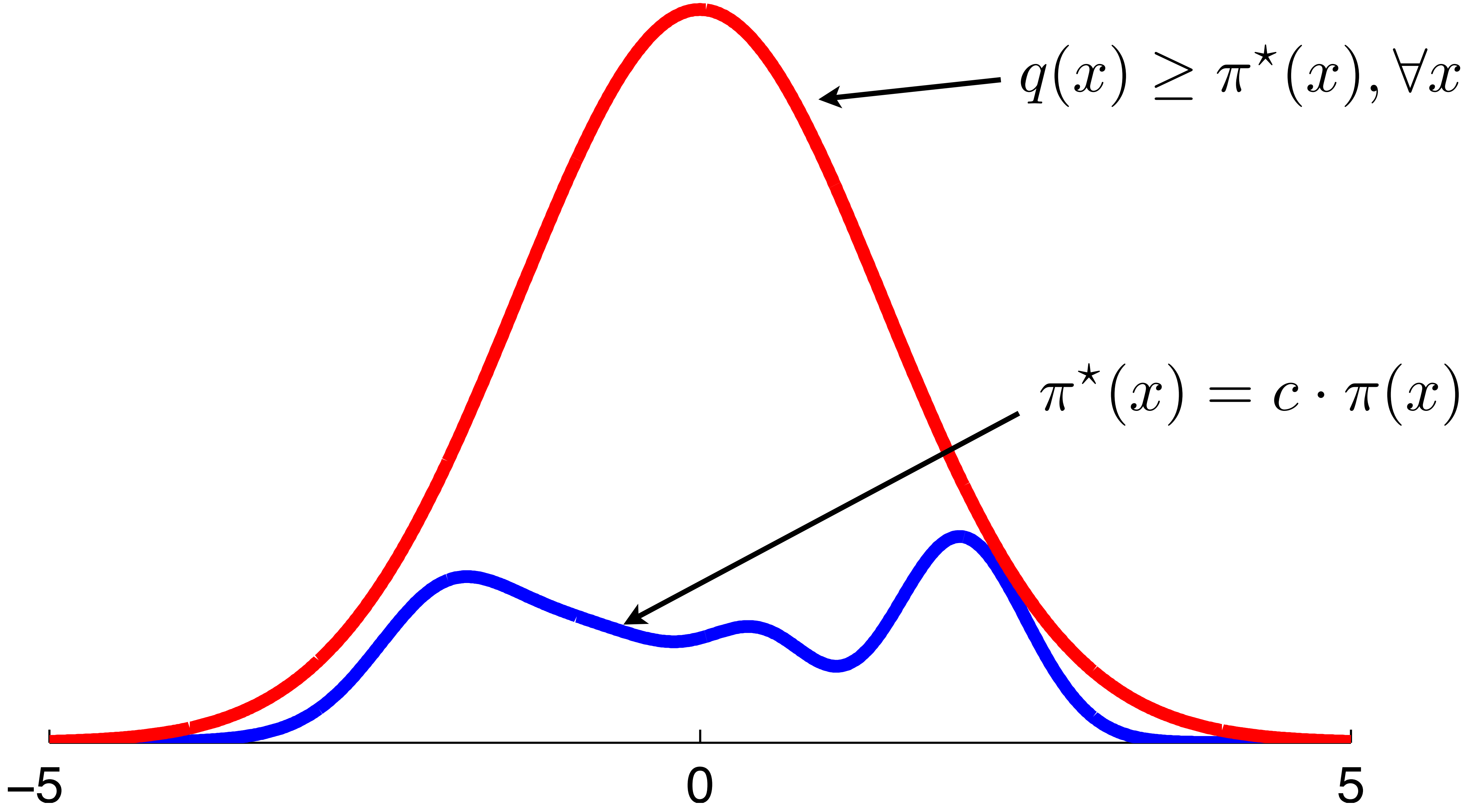
Rejection Sampling

One useful observation is that samples uniformly drawn from the volume beneath a (not necessarily normalized) PDF will have the correct marginal distribution.



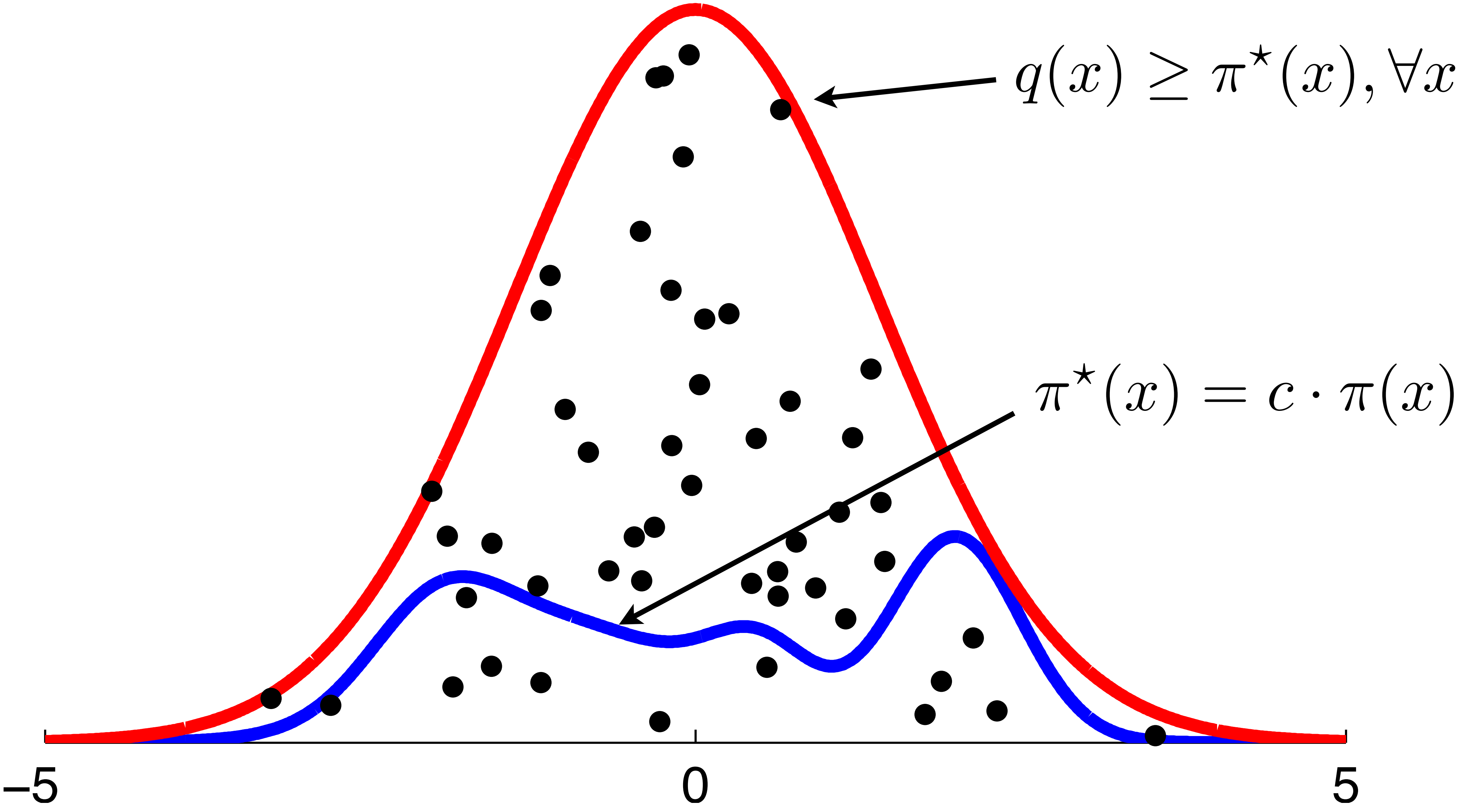
Rejection Sampling

How to get samples from the area? This is the first example, of sample from a simple $q(x)$ to get samples from a complicated $\pi(x)$.



Rejection Sampling

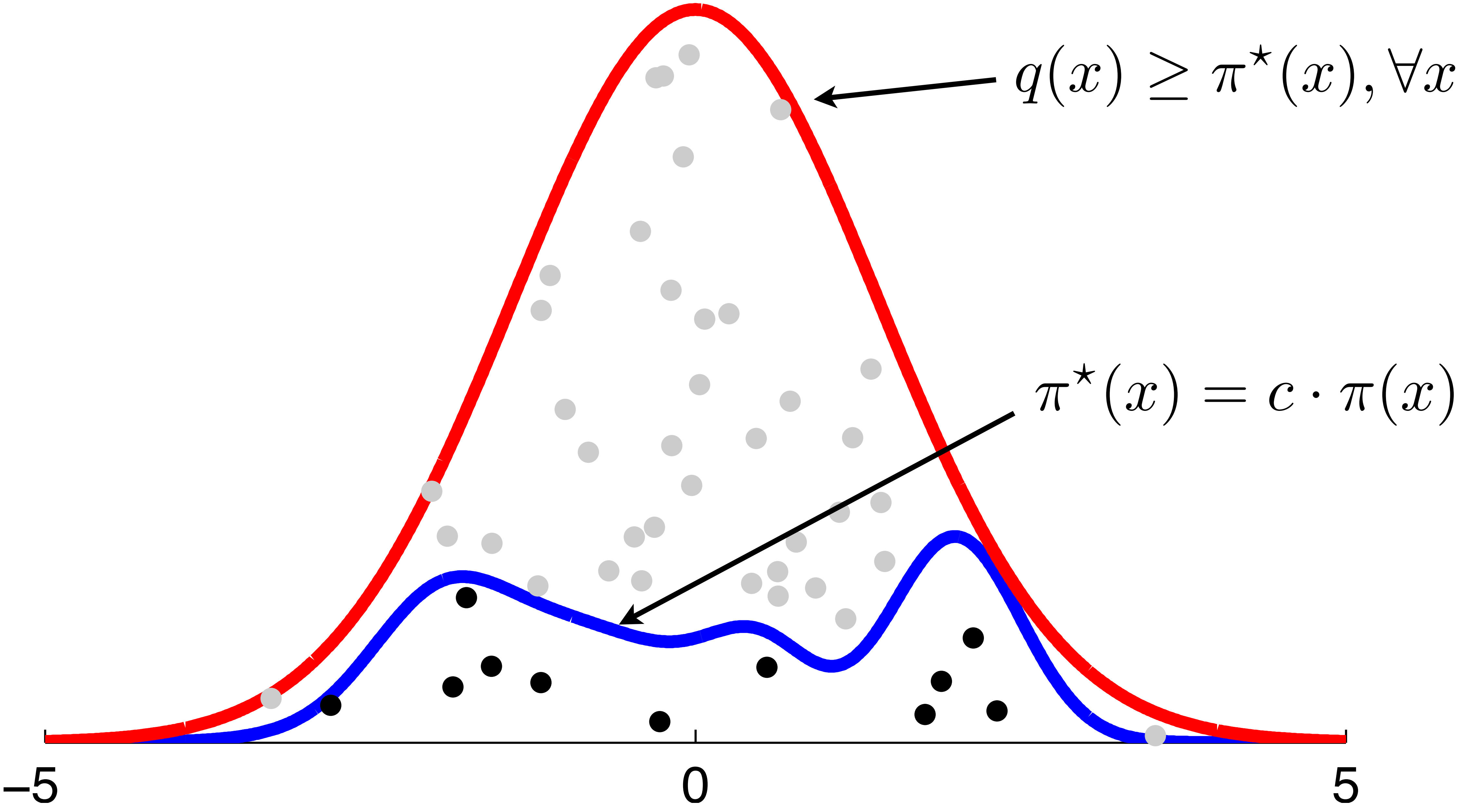
How to get samples from the area? This is the first example, of sample from a simple $q(x)$ to get samples from a complicated $\pi(x)$.


$q(x) \geq \pi^{\star}(x), \forall x$

$\pi^{\star}(x) = c \cdot \pi(x)$

Rejection Sampling

How to get samples from the area? This is the first example, of sample from a simple $q(x)$ to get samples from a complicated $\pi(x)$.



Rejection Sampling

1. Choose $q(x)$ and c so that $q(x) \geq \pi^\star(x) = c \cdot \pi(x), \forall x$
2. Sample $x^{(s)} \sim q(x)$
3. Sample $u^{(s)} \sim \text{Uniform}(0, q(x^{(s)}))$
4. If $u^{(s)} \leq \pi^\star(x^{(s)})$ keep $x^{(s)}$, else reject and goto 2.

If you accept, you get an unbiased sample from $\pi(x)$.

Isn't it wasteful to throw away all those proposals?

Importance Sampling

Recall that we're really just after an expectation.

$$\int f(x)\,\pi(x)\,\mathrm{d}x \;\approx\; \frac{1}{S}\sum_{s=1}^{S} f(x^{(s)}) \;\;\text{where}\;\; x^{(s)} \sim \pi(x)$$

We could write the above integral another way:

$$\int f(x)\,\pi(x)\,\mathrm{d}x = \int f(x)\,q(x)\,\frac{\pi(x)}{q(x)}\,\mathrm{d}x$$

$$\text{where } q(x) > 0 \text{ if } \pi(x) > 0$$

Importance Sampling

We can now write a Monte Carlo estimate that is also an expectation under the "easy" distribution $q(x)$

$$\int f(x)\,\pi(x)\,\mathrm{d}x = \int f(x)\,q(x)\,\frac{\pi(x)}{q(x)}\,\mathrm{d}x \approx \frac{1}{S}\sum_{s=1}^{S} f(x^{(s)})\,\frac{\pi(x^{(s)})}{q(x^{(s)})}$$

$$\text{where } x^{(s)} \sim q(x)$$

We don't get samples from $\pi(x)$, so no easy visualization of fantasy data, but we do get an unbiased estimator of whatever expectation we're interested in.

It's like we're "correcting" each sample with a weight.

Importance Sampling

As a side note, this trick also works with integrals that do not correspond to expectations.

$$\int f(x)\,\mathrm{d}x = \int q(x)\frac{f(x)}{q(x)}\,\mathrm{d}x \approx \frac{1}{S}\sum_{s=1}^{S}\frac{f(x^{(s)})}{q(x^{(s)})}$$

$$\text{where } x^{(s)} \sim q(x)$$

Scaling Up

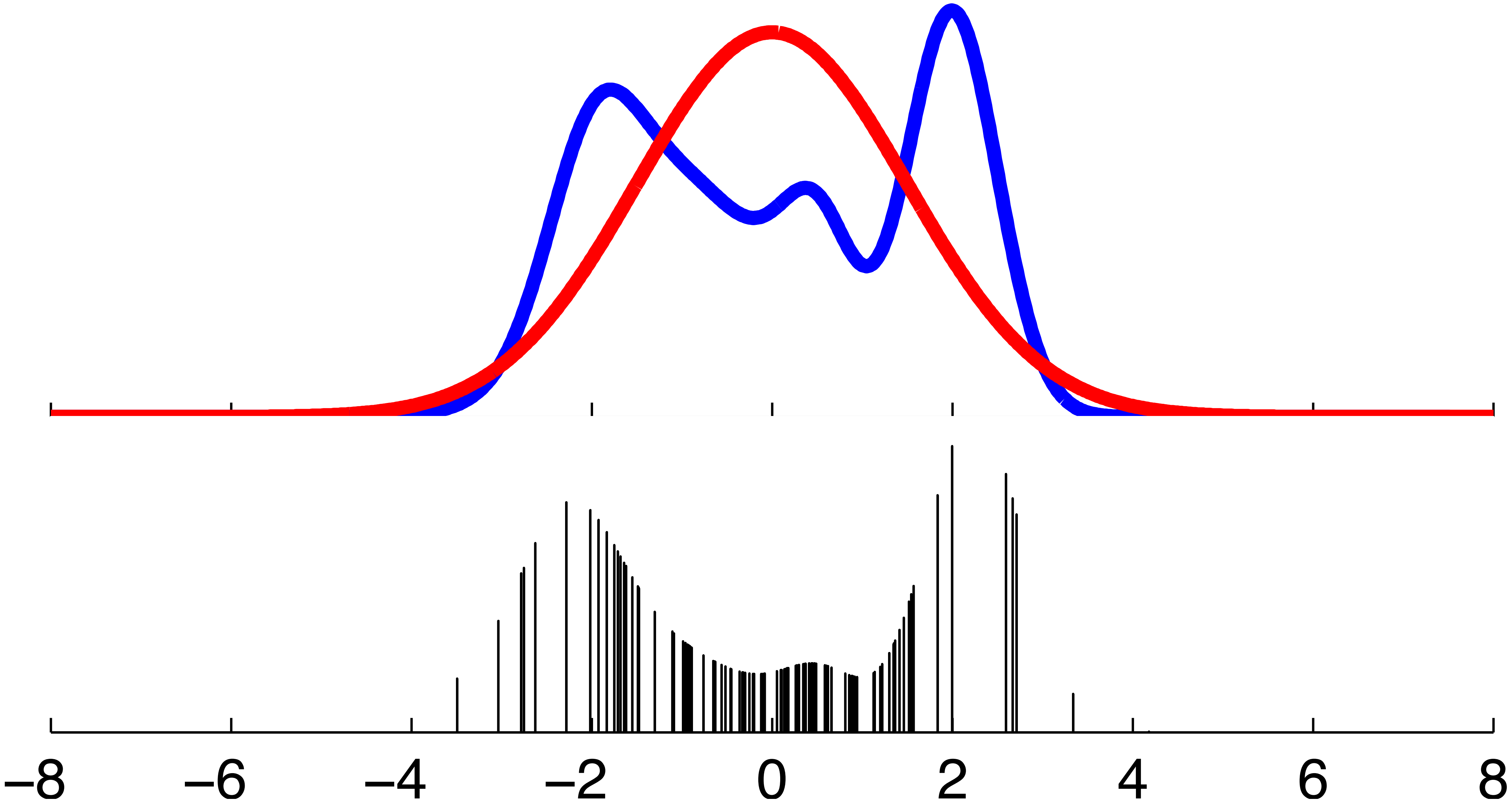
Both rejection and importance sampling depend heavily on having a $q(x)$ that is very similar to $\pi(x)$

In interesting high-dimensional problems, it is very hard to choose a $q(x)$ that is "easy" and also resembles the fancy distribution you're interested in.

The whole point is that you're trying to use a powerful model to capture, say, the statistics of natural images in a way that *isn't* captured by a simple distribution!

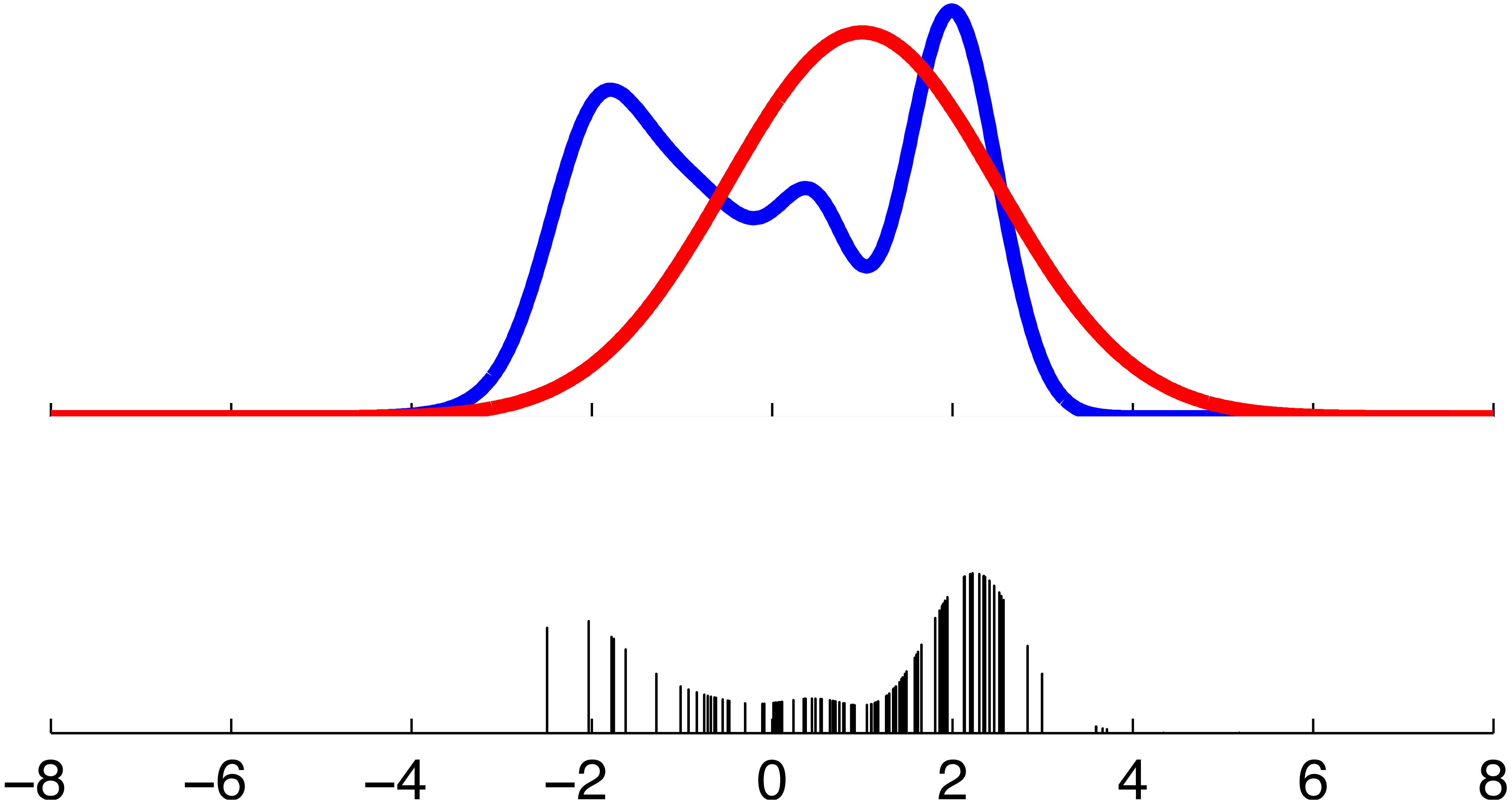
Exploding Importance Weights

Even without going into high dimensions, we can see how a mismatch between the distributions can cause a few importance weights to grow very large.



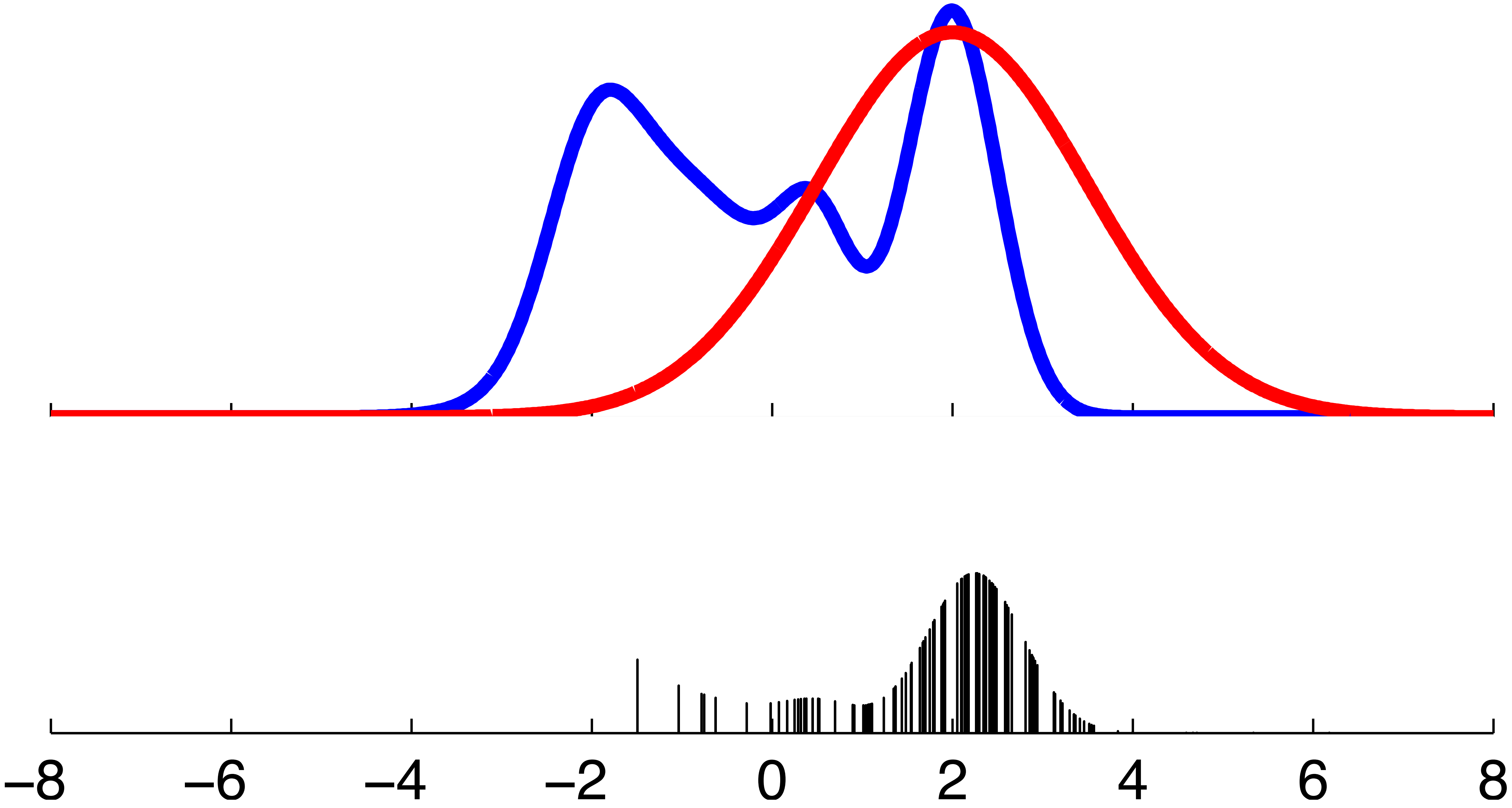
Exploding Importance Weights

Even without going into high dimensions, we can see how a mismatch between the distributions can cause a few importance weights to grow very large.



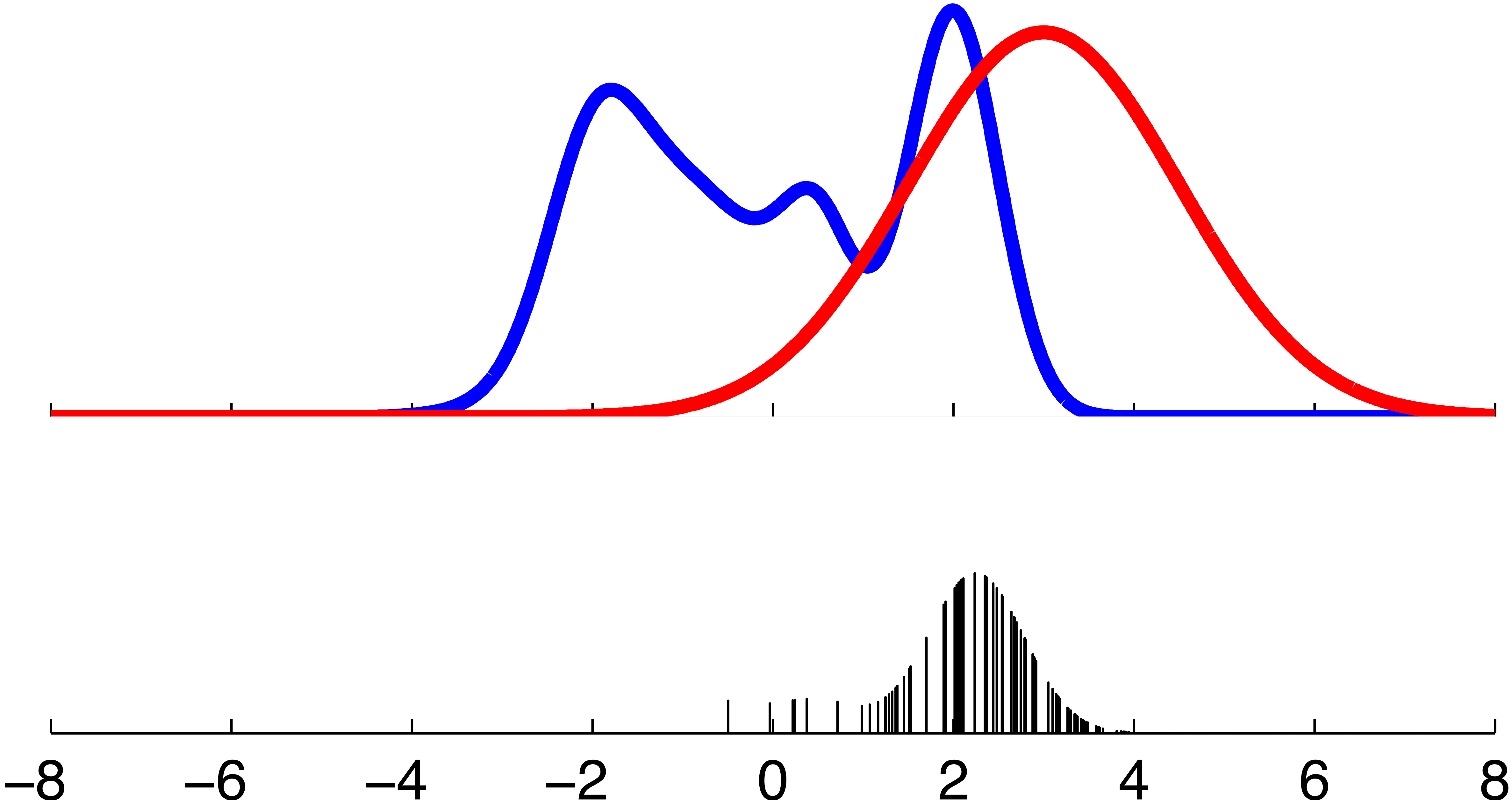
Exploding Importance Weights

Even without going into high dimensions, we can see how a mismatch between the distributions can cause a few importance weights to grow very large.



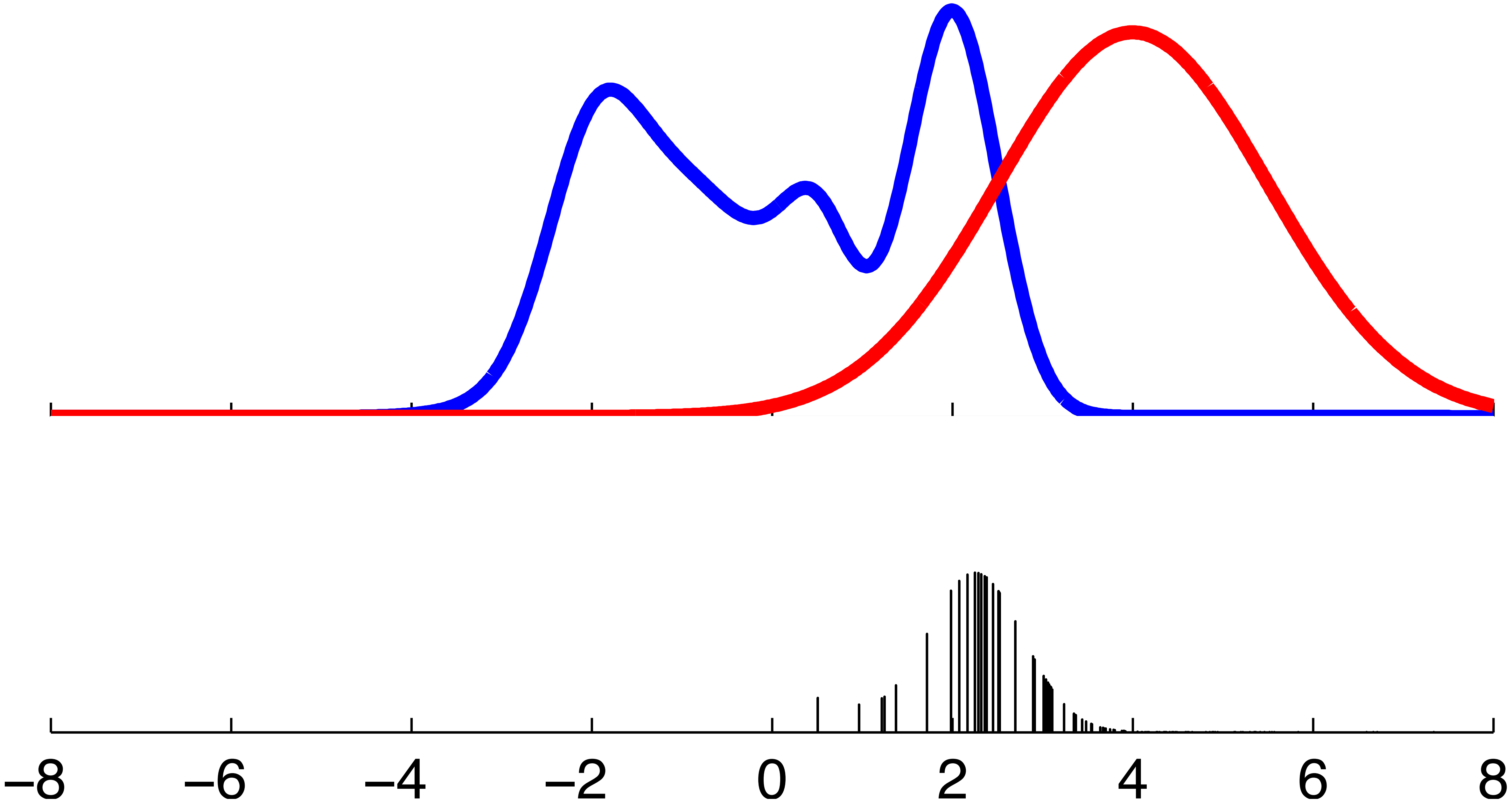
Exploding Importance Weights

Even without going into high dimensions, we can see how a mismatch between the distributions can cause a few importance weights to grow very large.



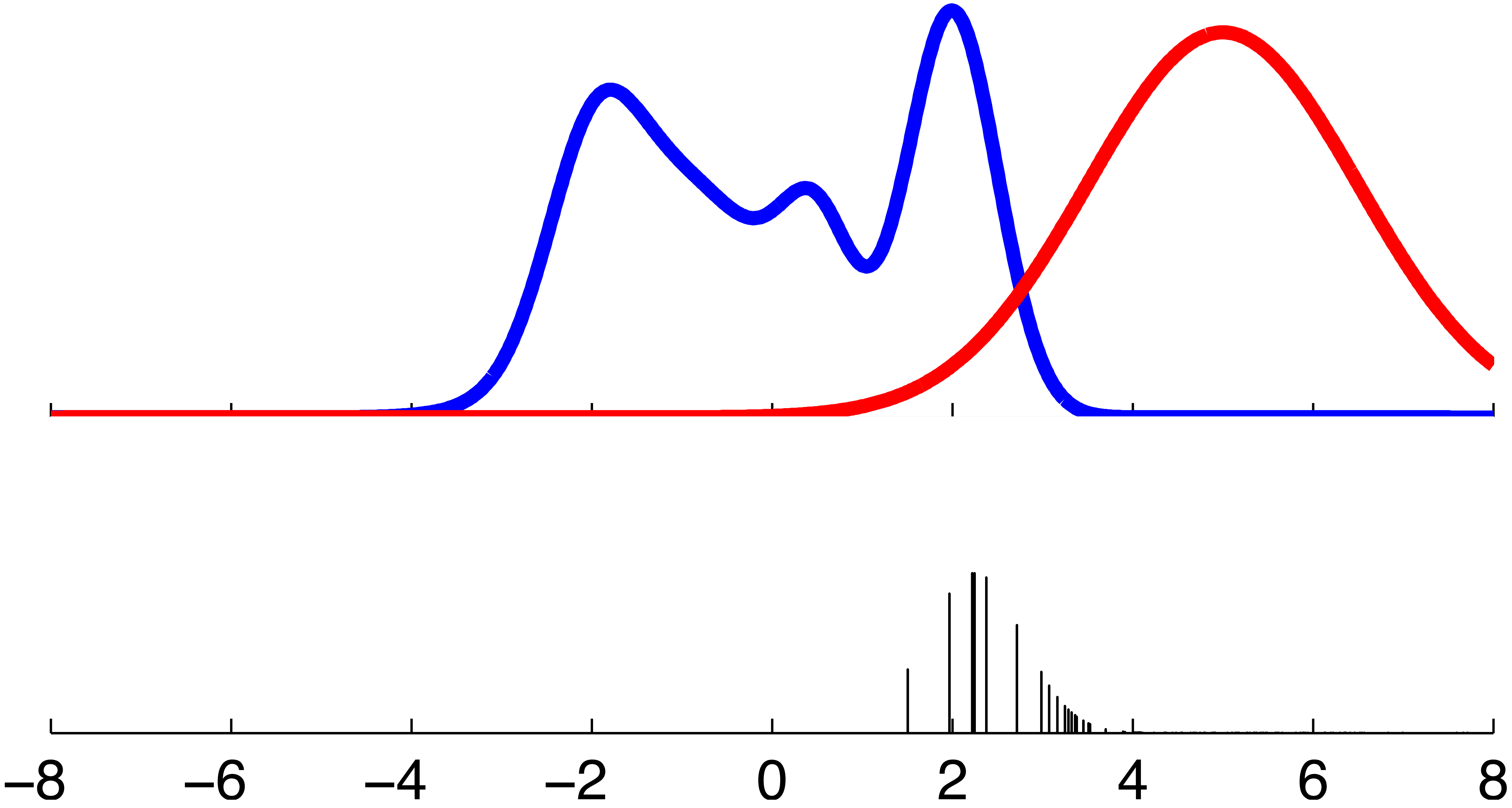
Exploding Importance Weights

Even without going into high dimensions, we can see how a mismatch between the distributions can cause a few importance weights to grow very large.



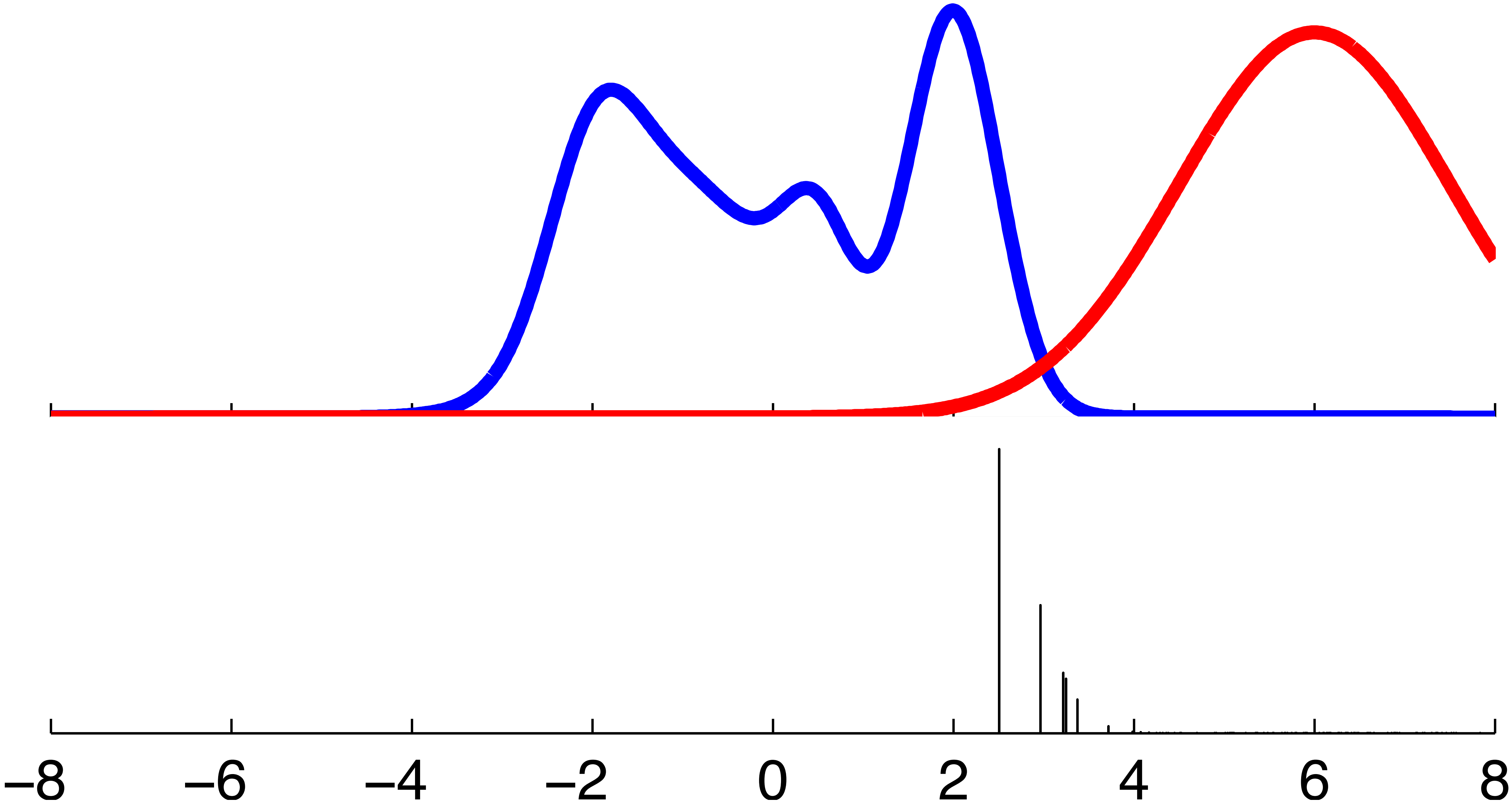
Exploding Importance Weights

Even without going into high dimensions, we can see how a mismatch between the distributions can cause a few importance weights to grow very large.



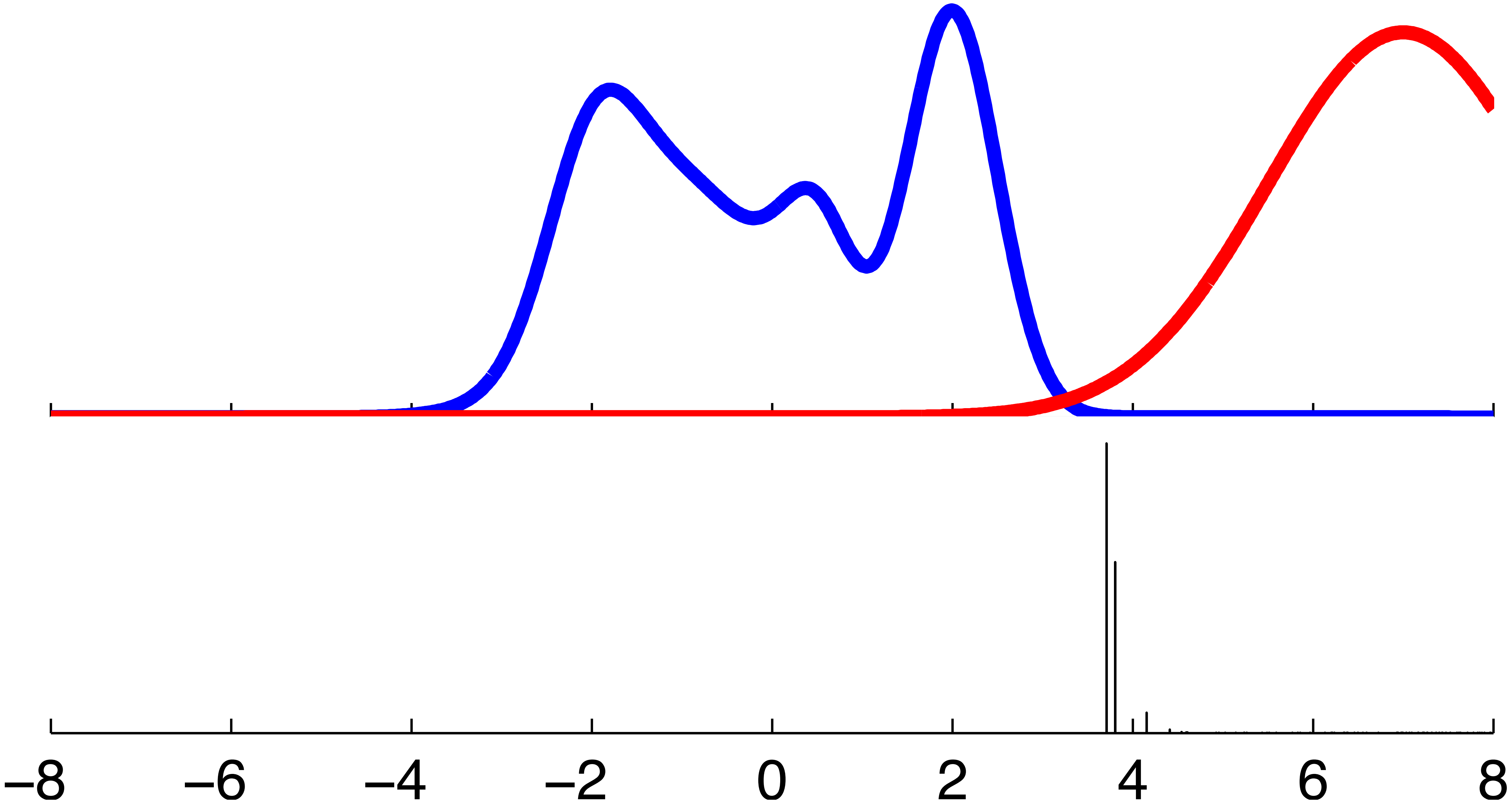
Exploding Importance Weights

Even without going into high dimensions, we can see how a mismatch between the distributions can cause a few importance weights to grow very large.



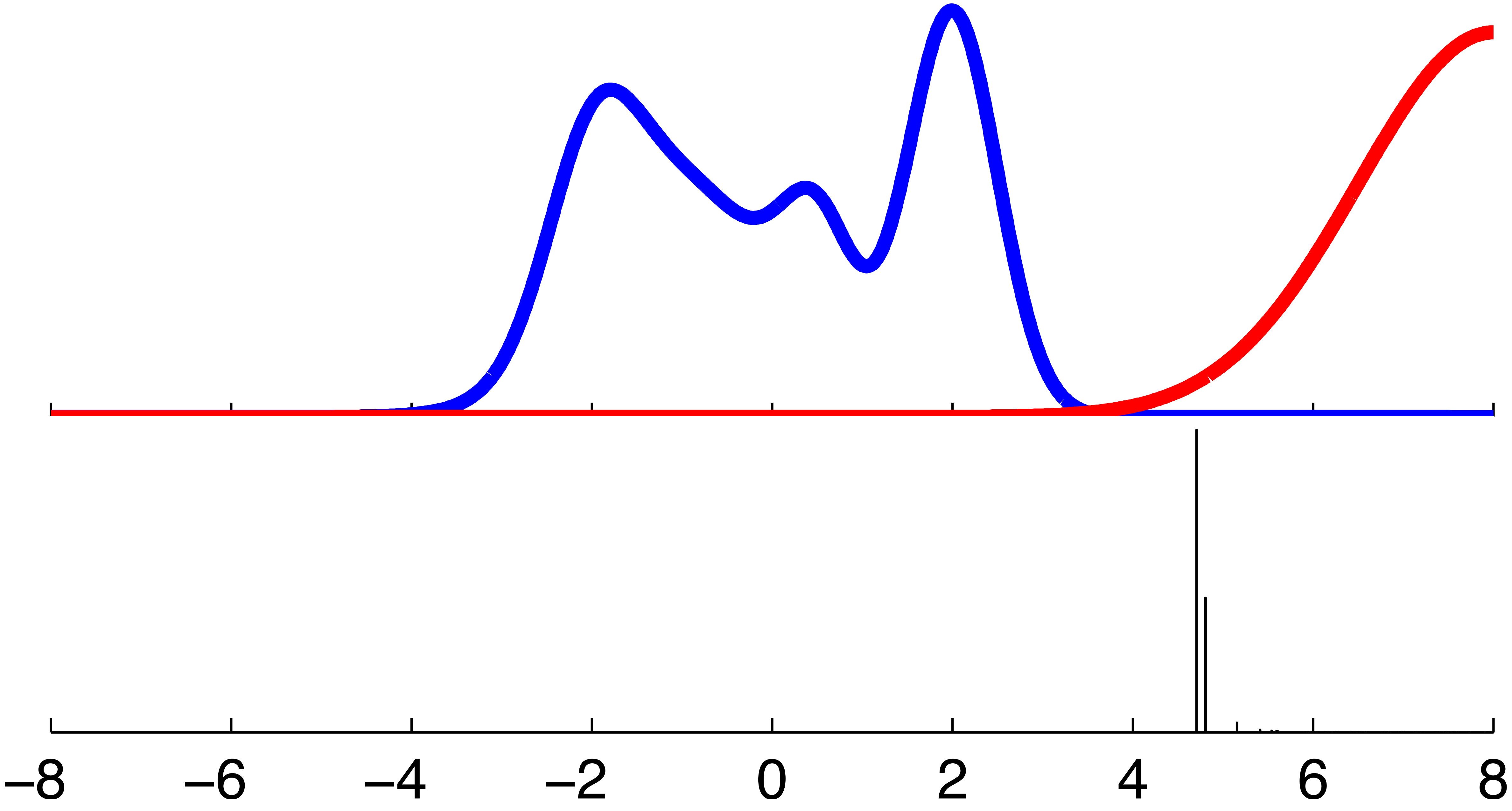
Exploding Importance Weights

Even without going into high dimensions, we can see how a mismatch between the distributions can cause a few importance weights to grow very large.



Exploding Importance Weights

Even without going into high dimensions, we can see how a mismatch between the distributions can cause a few importance weights to grow very large.



Scaling Up

In high dimensions, the mismatch between the proposal distribution and the true distribution can really ramp up quickly. Example:

$$\pi(x) = \mathcal{N}(0, \mathbb{I}) \text{ and } q(x) = \mathcal{N}(0, \sigma^2\mathbb{I})$$

Rejection sampling requires $\sigma \geq 1$ and accepts with probability σ^{-D}. For $\sigma = 1.1, D = 50$ the acceptance rate will be less than one percent.

The variance of the importance sampling weights will grow exponentially with dimension. That means that in high dimensions, the answer will be dominated by only a few of the samples.

Summary So Far

We would like to find statistics of our probabilistic models for inference, learning and prediction.

Computation of these quantities often involves difficult integrals or sums.

<u>Monte Carlo</u> approximates these with sample averages.

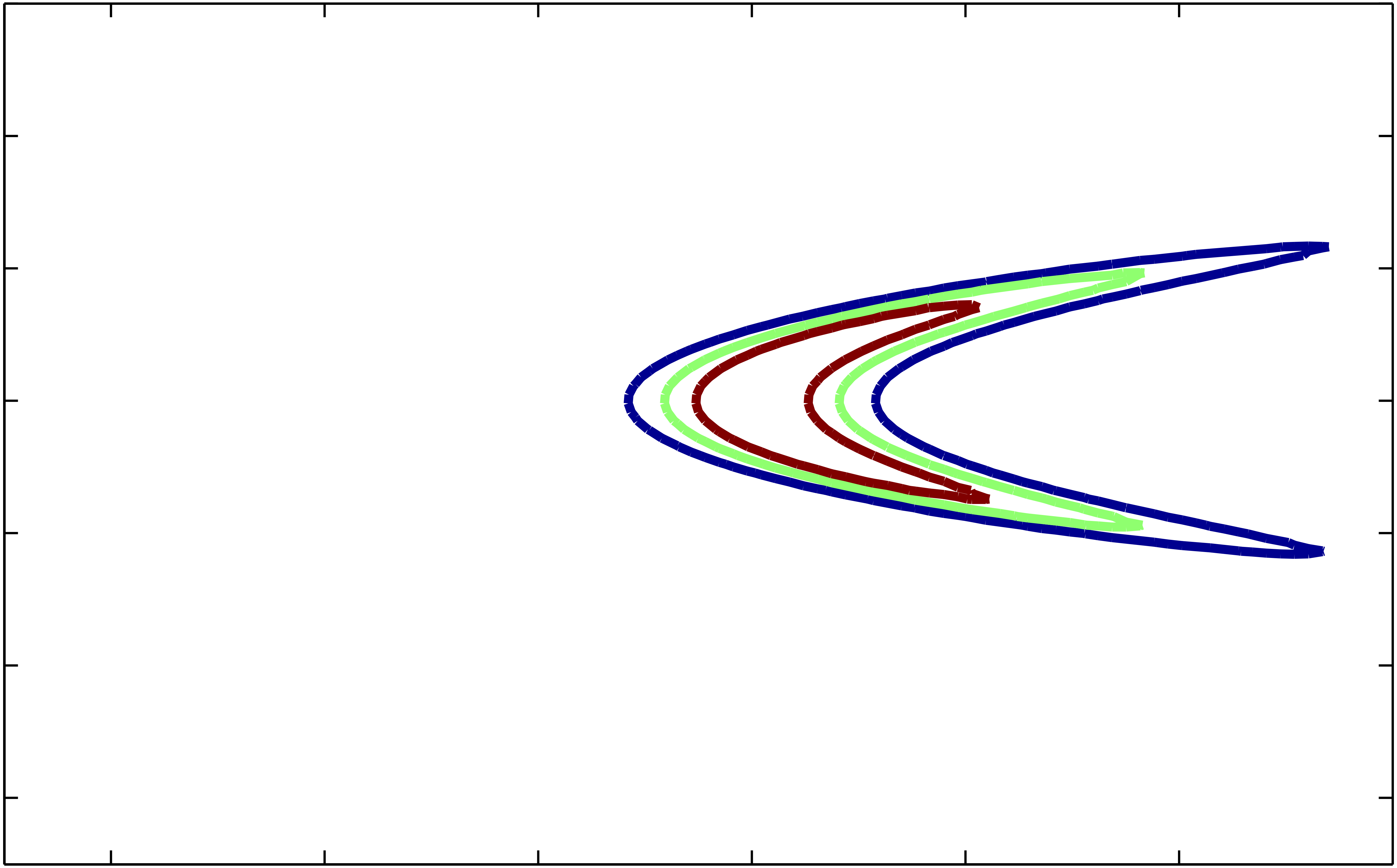
<u>Rejection sampling</u> provides unbiased samples from a complex distribution.

<u>Importance sampling</u> provides an unbiased estimator of a difficult expectation by “correcting” another expectation.

Neither of these methods scale well in high dimensions.

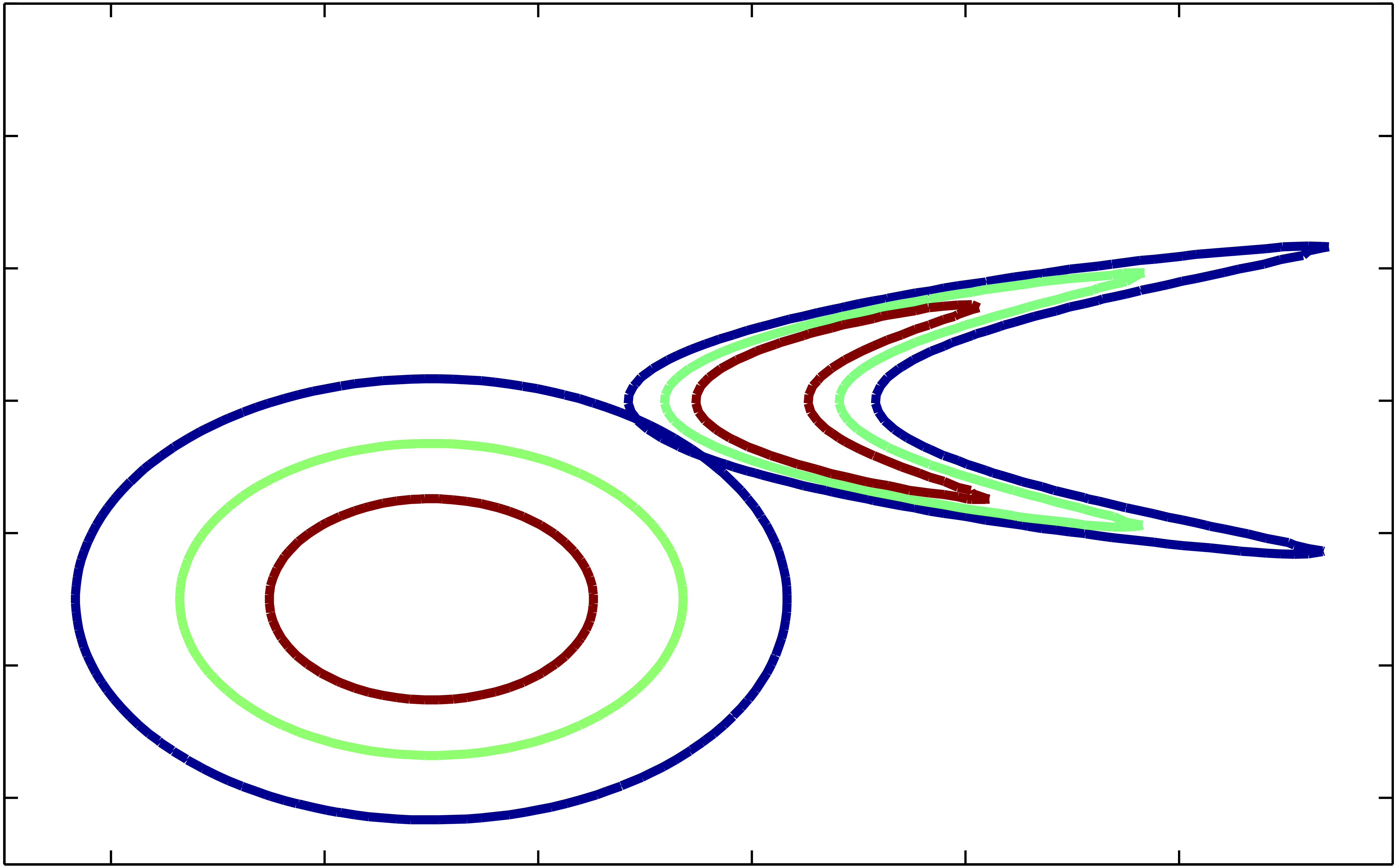
Revisiting Independence

It’s hard to find the mass of an unknown density!



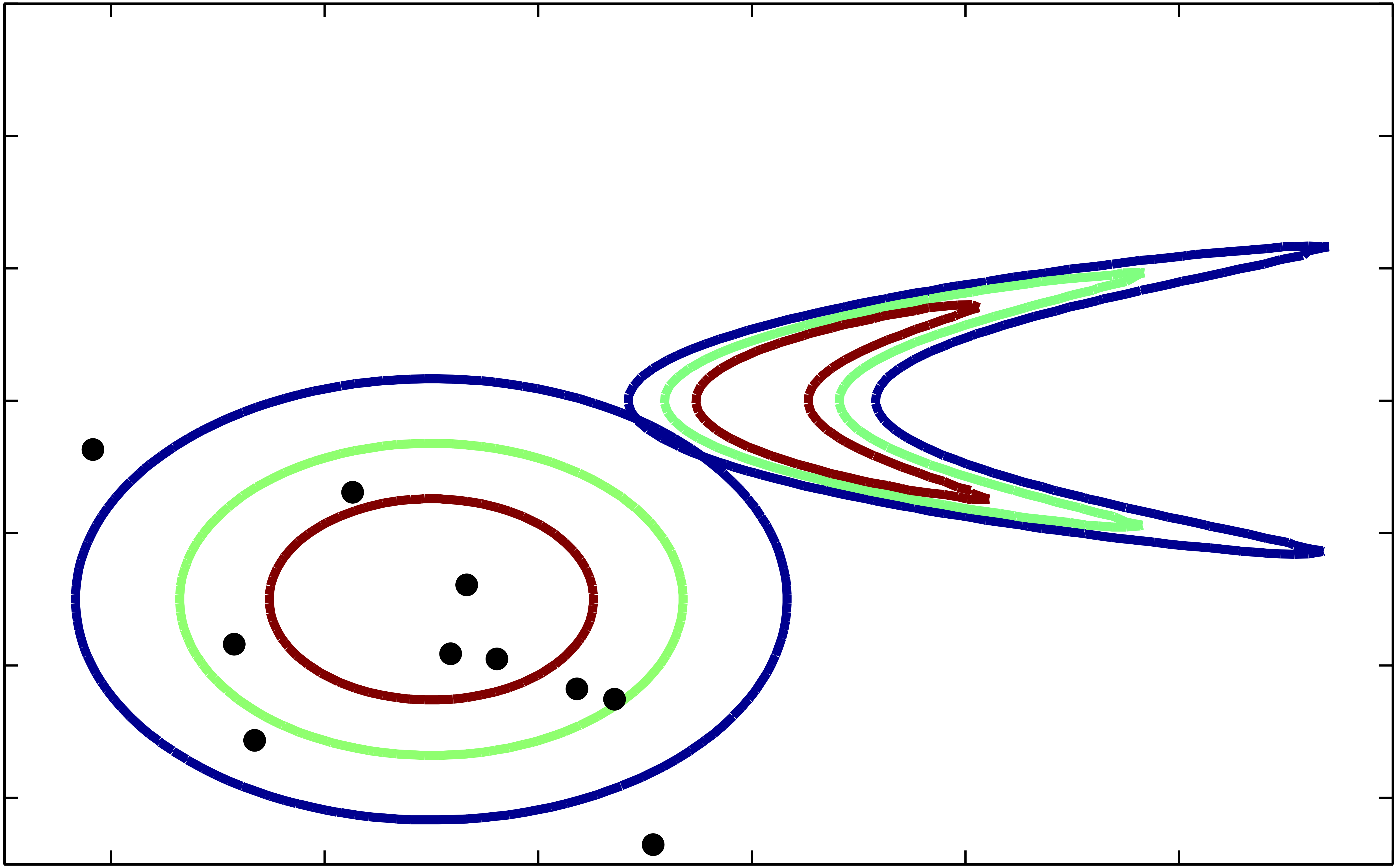
Revisiting Independence

It’s hard to find the mass of an unknown density!



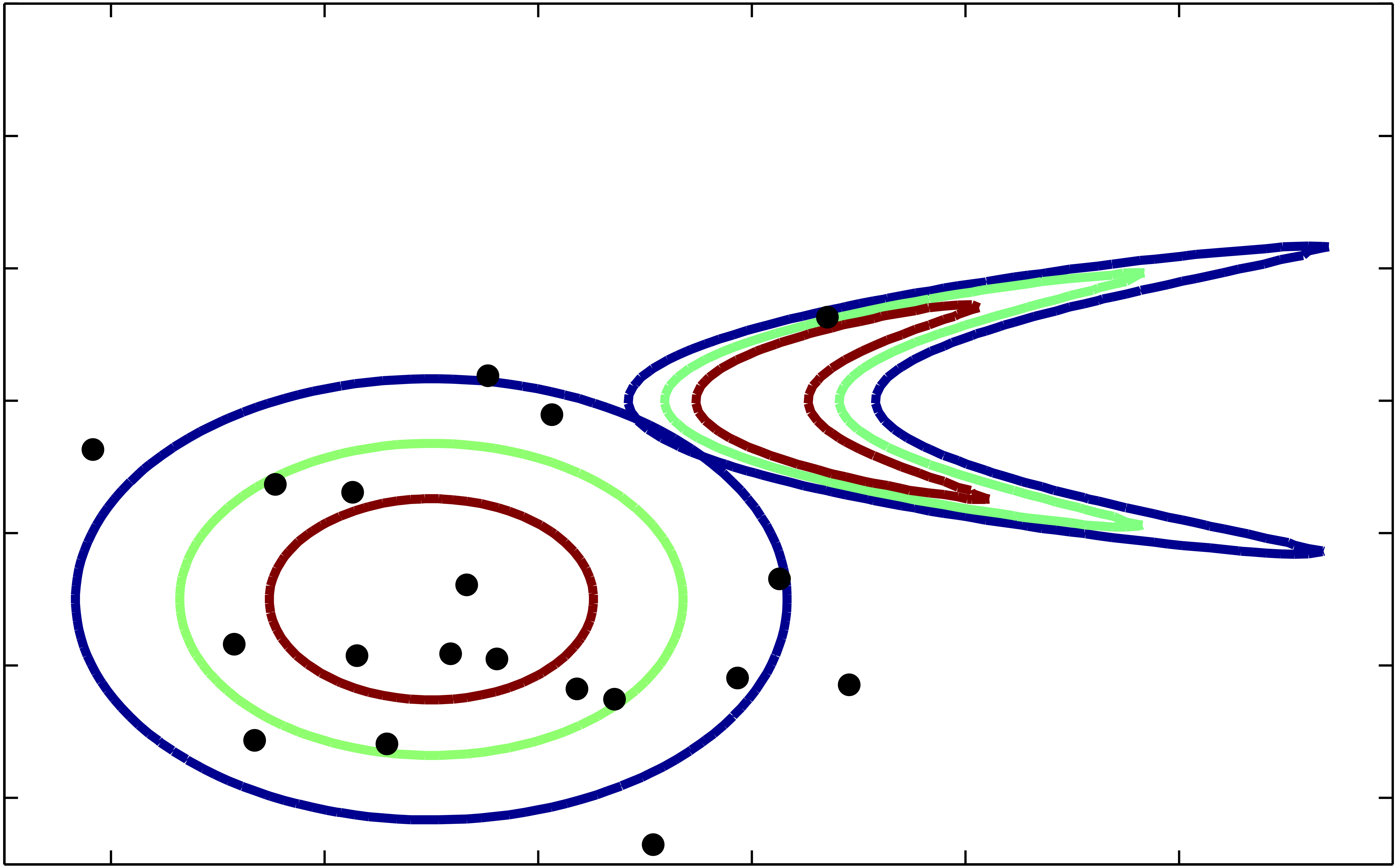
Revisiting Independence

It's hard to find the mass of an unknown density!



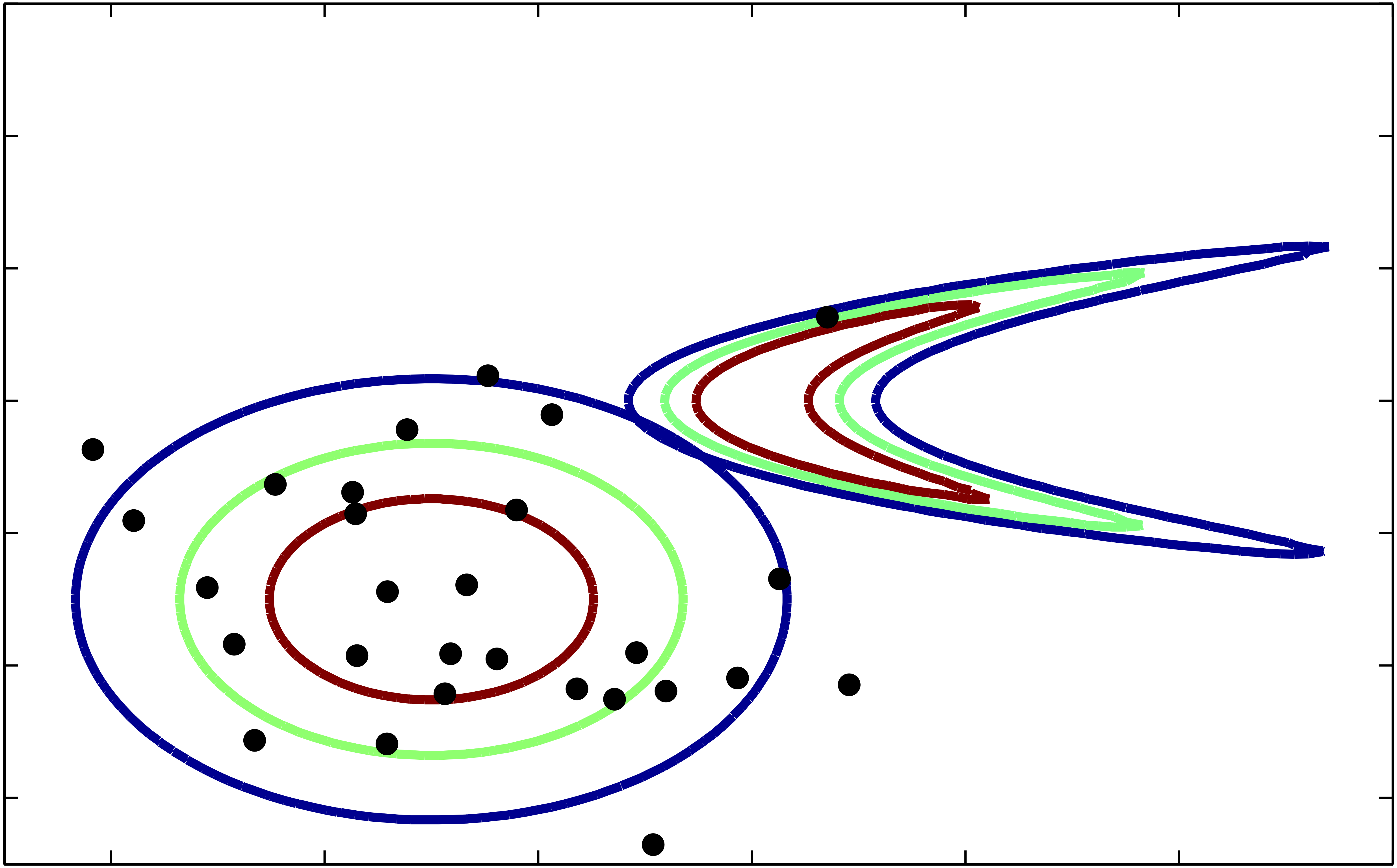
Revisiting Independence

It’s hard to find the mass of an unknown density!



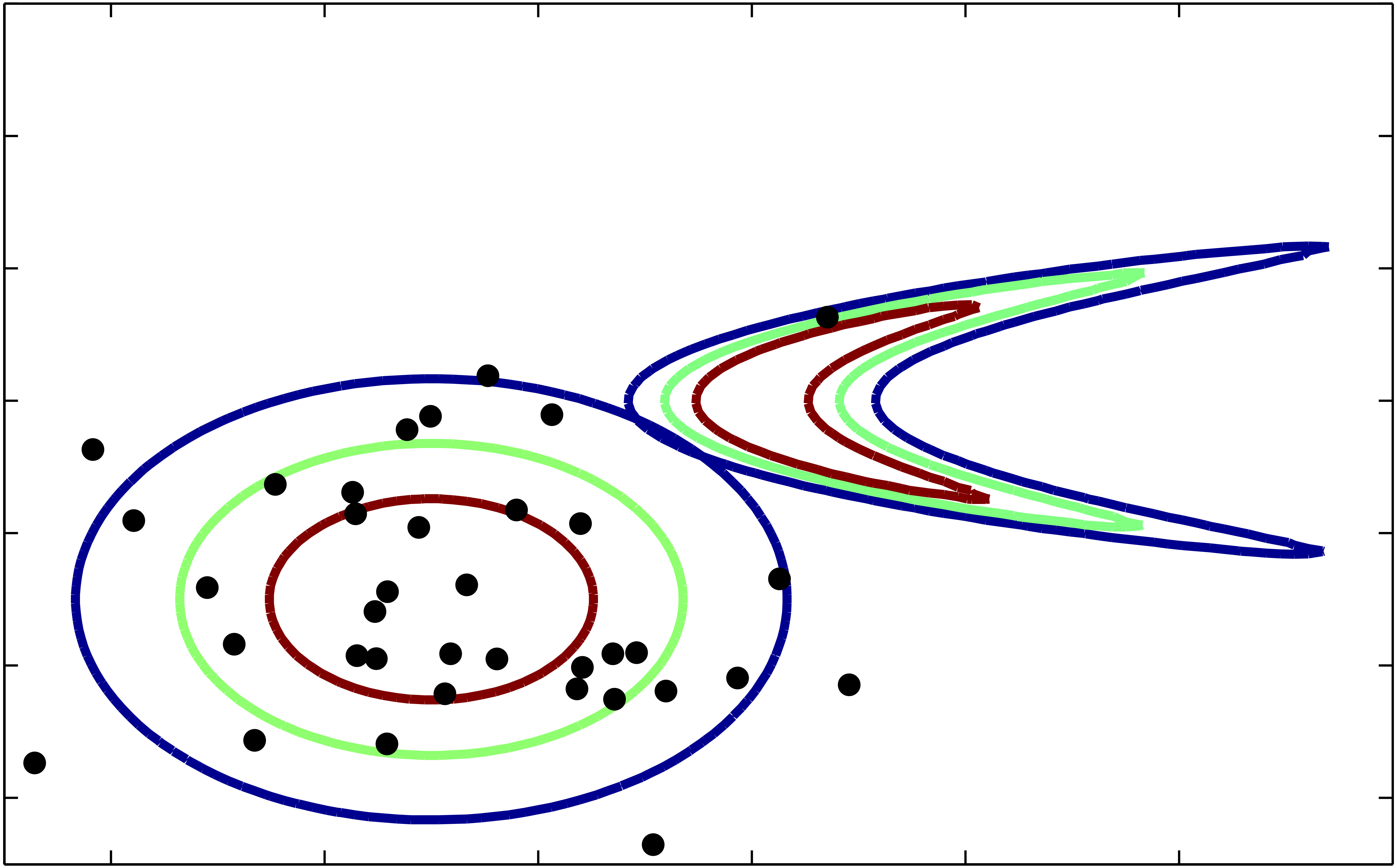
Revisiting Independence

It’s hard to find the mass of an unknown density!



Revisiting Independence

It’s hard to find the mass of an unknown density!



Revisiting Independence

Why should we immediately forget that we discovered a place with high density? Can we use that information?

Storing this information will mean that the sequence now has correlations in it. Does this matter?

Can we do this in a principled way so that we get good estimates of the expectations we're interested in?

Markov chain Monte Carlo

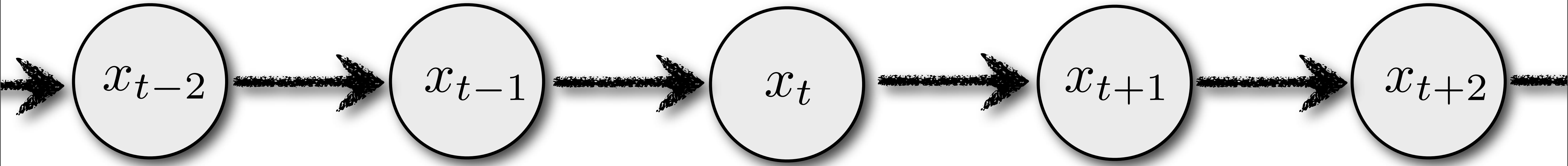
Markov chain Monte Carlo

As in rejection and importance sampling, in MCMC we have some kind of "easy" distribution that we use to compute something about our "hard" distribution $\pi(x)$.

The difference is that we're going to use the easy distribution to **update** our current state, rather than to draw a new one from scratch.

If the update depends only on the current state, then it is Markovian. Sequentially making these random updates will correspond to simulating a Markov chain.

Markov chain Monte Carlo

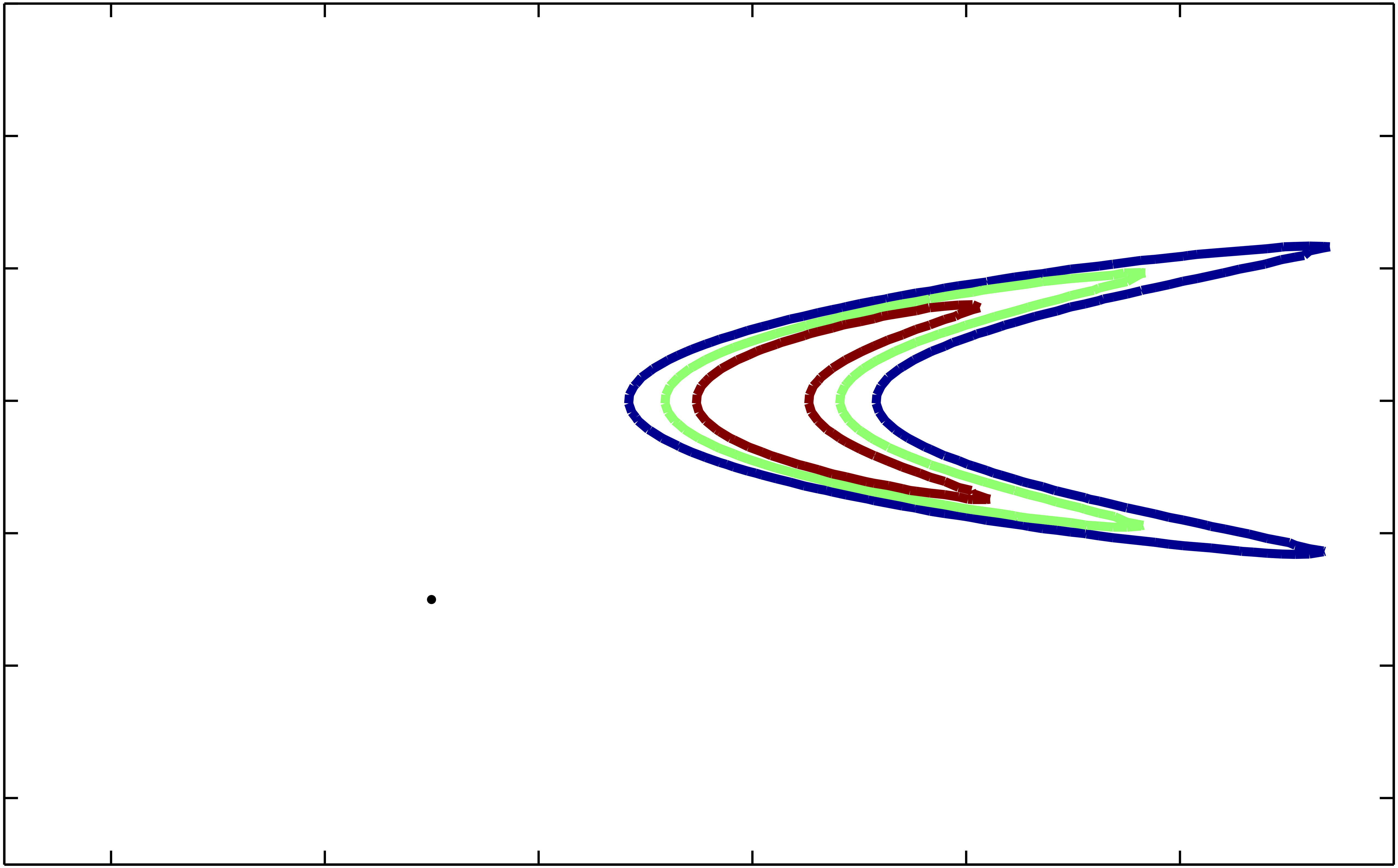


We define a Markov transition operator $T(x' \leftarrow x)$

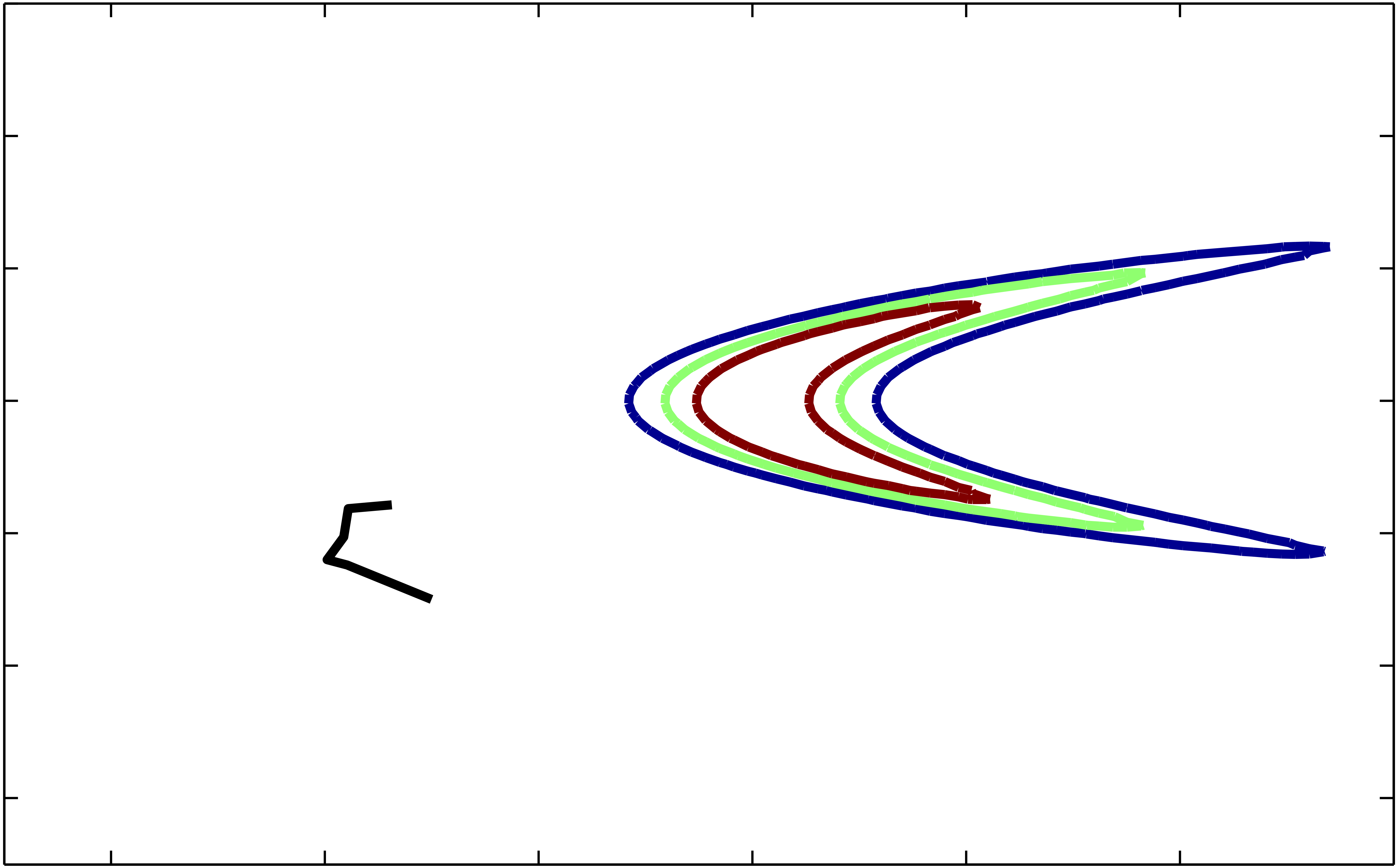
The trick is: if we choose the transition operator carefully, the marginal distribution over the state at any given instant can have our distribution $\pi(x)$

If the marginal distribution is correct, then our estimator for the expectation is unbiased.

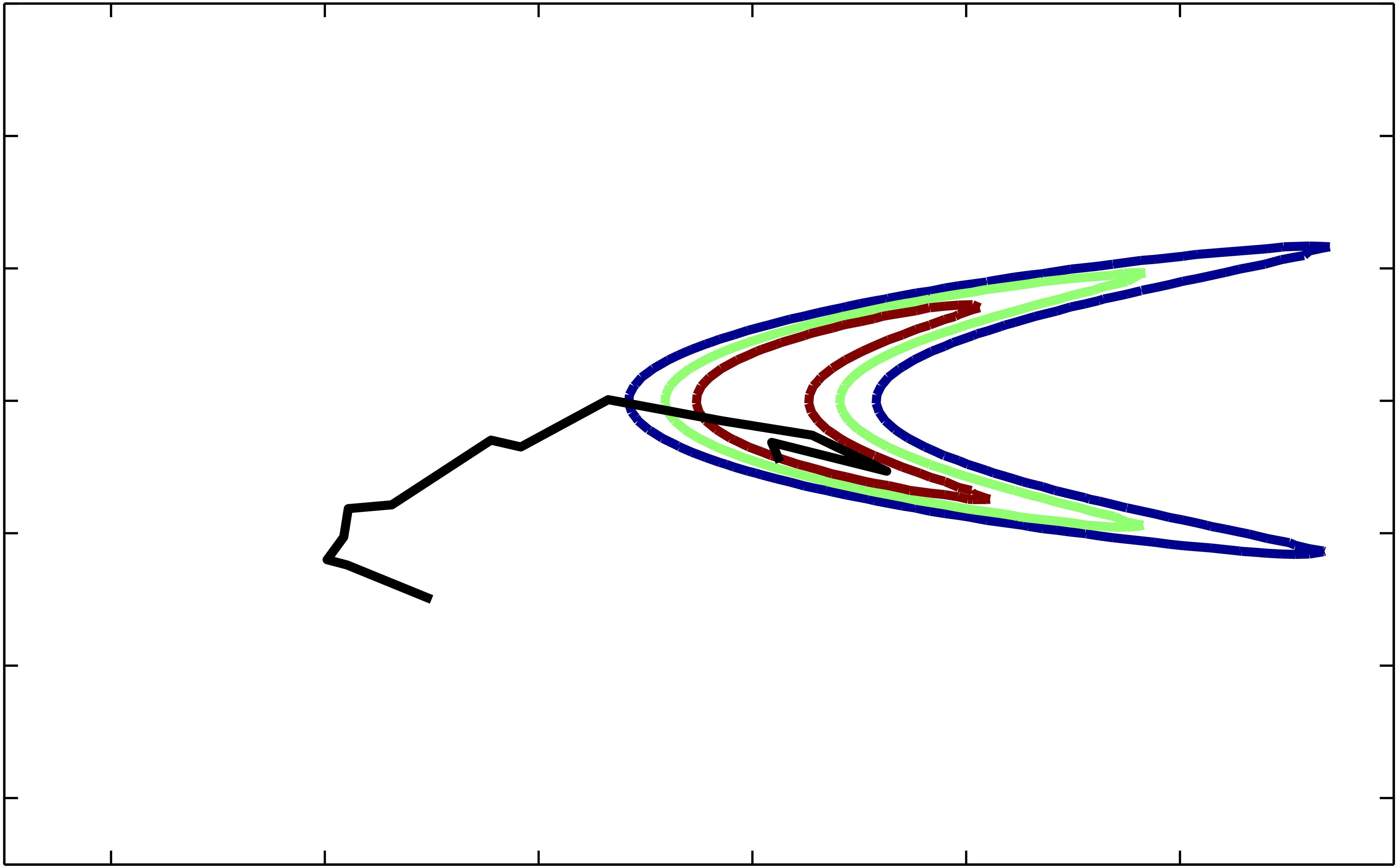
Markov chain Monte Carlo



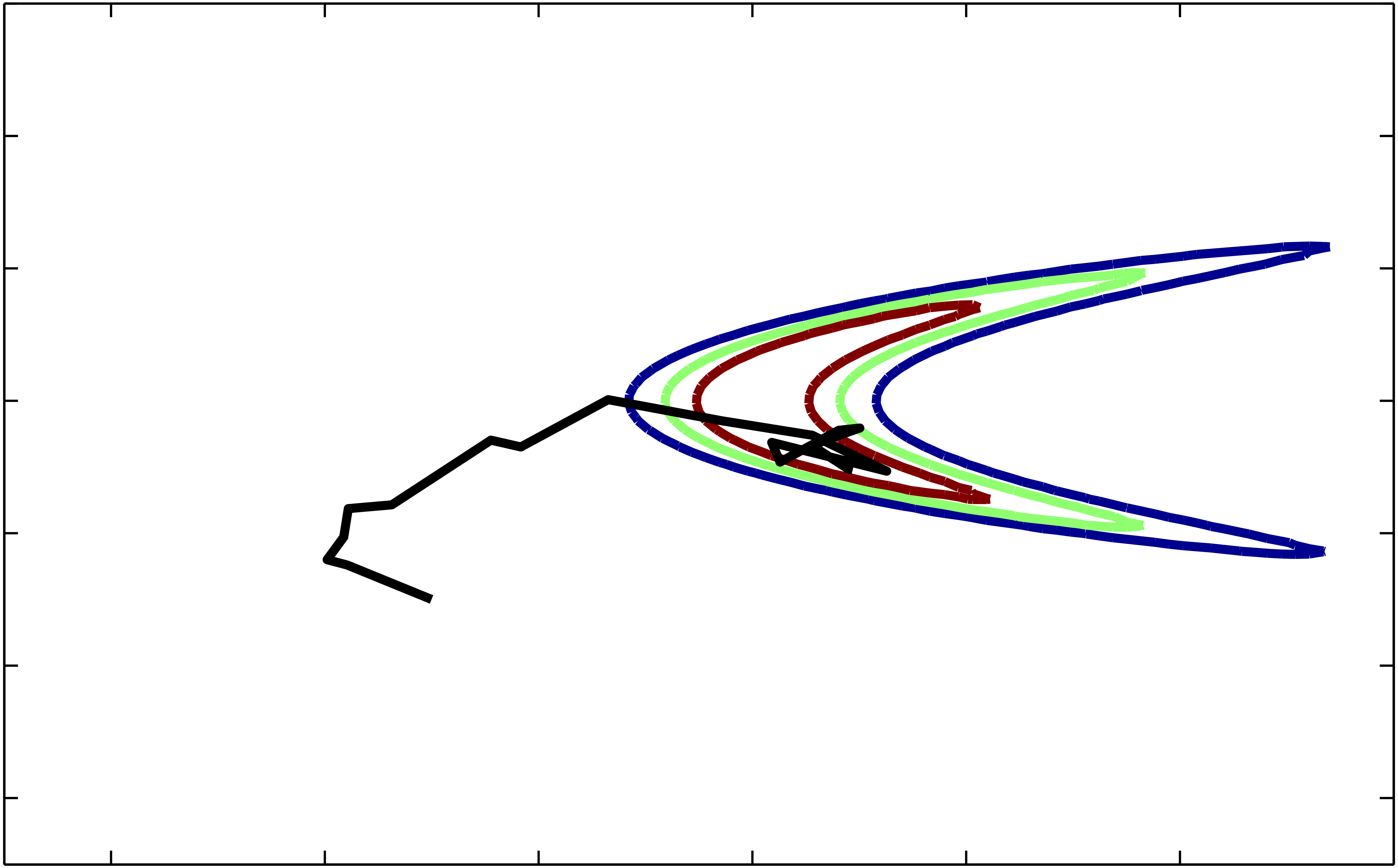
Markov chain Monte Carlo



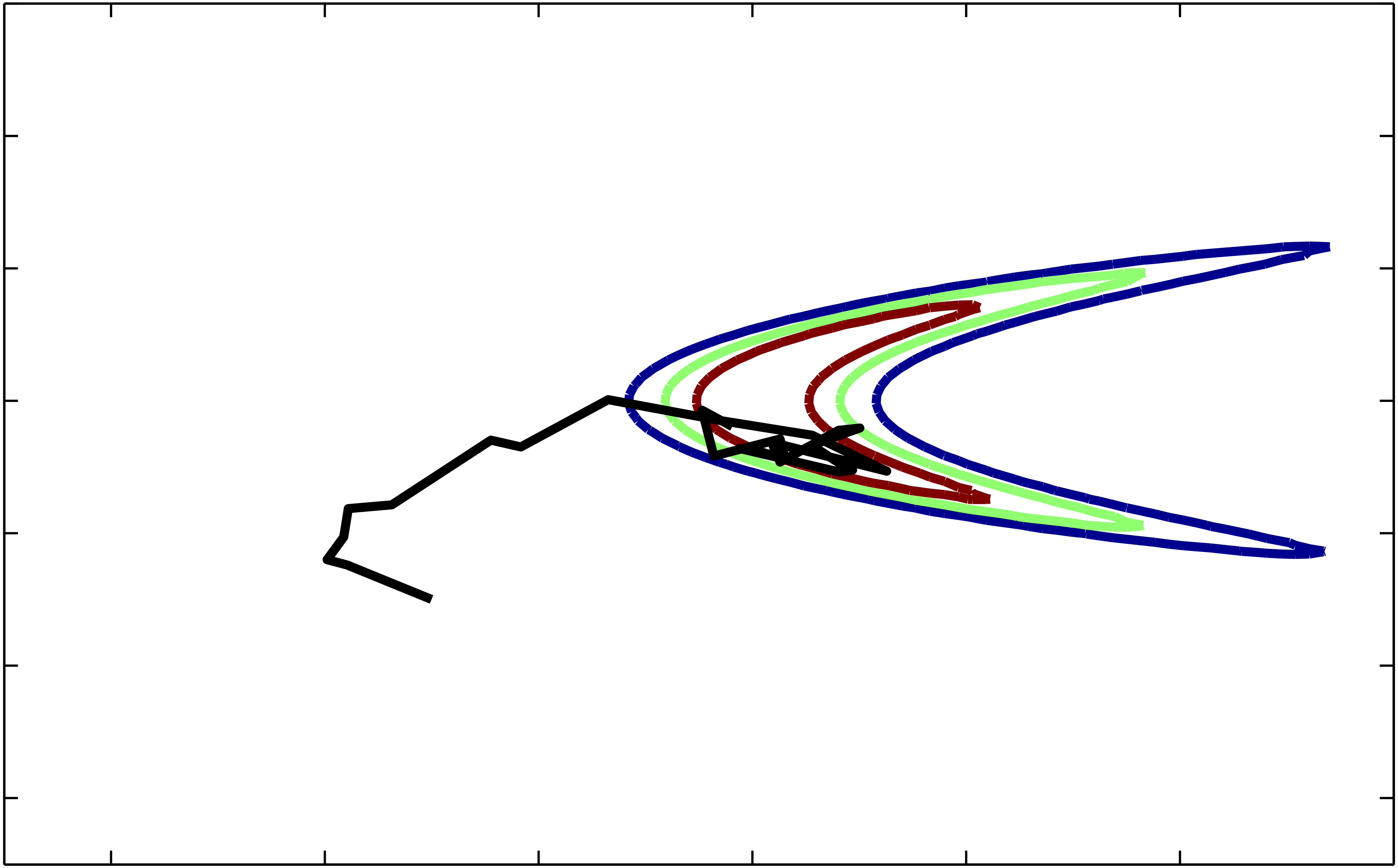
Markov chain Monte Carlo



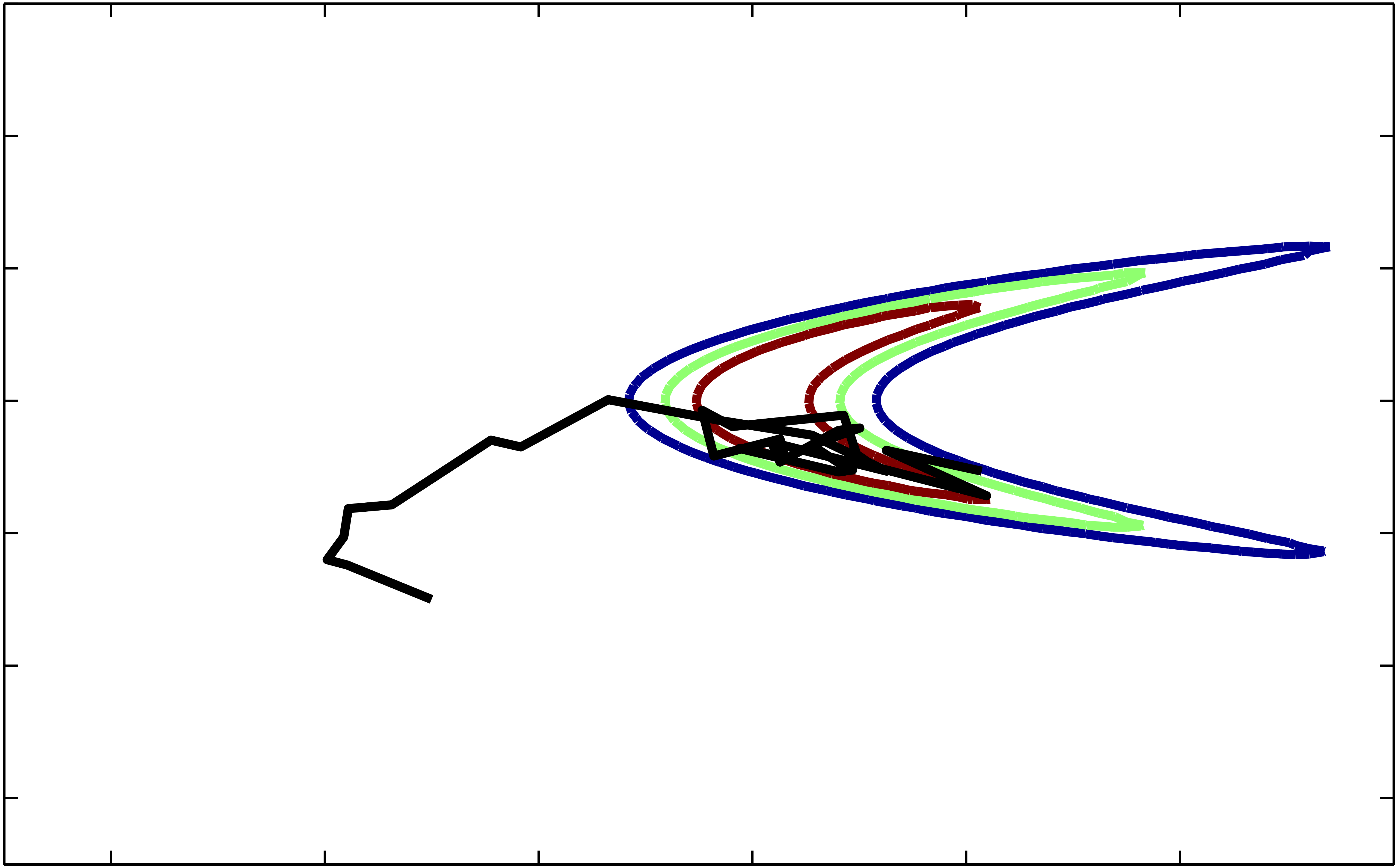
Markov chain Monte Carlo



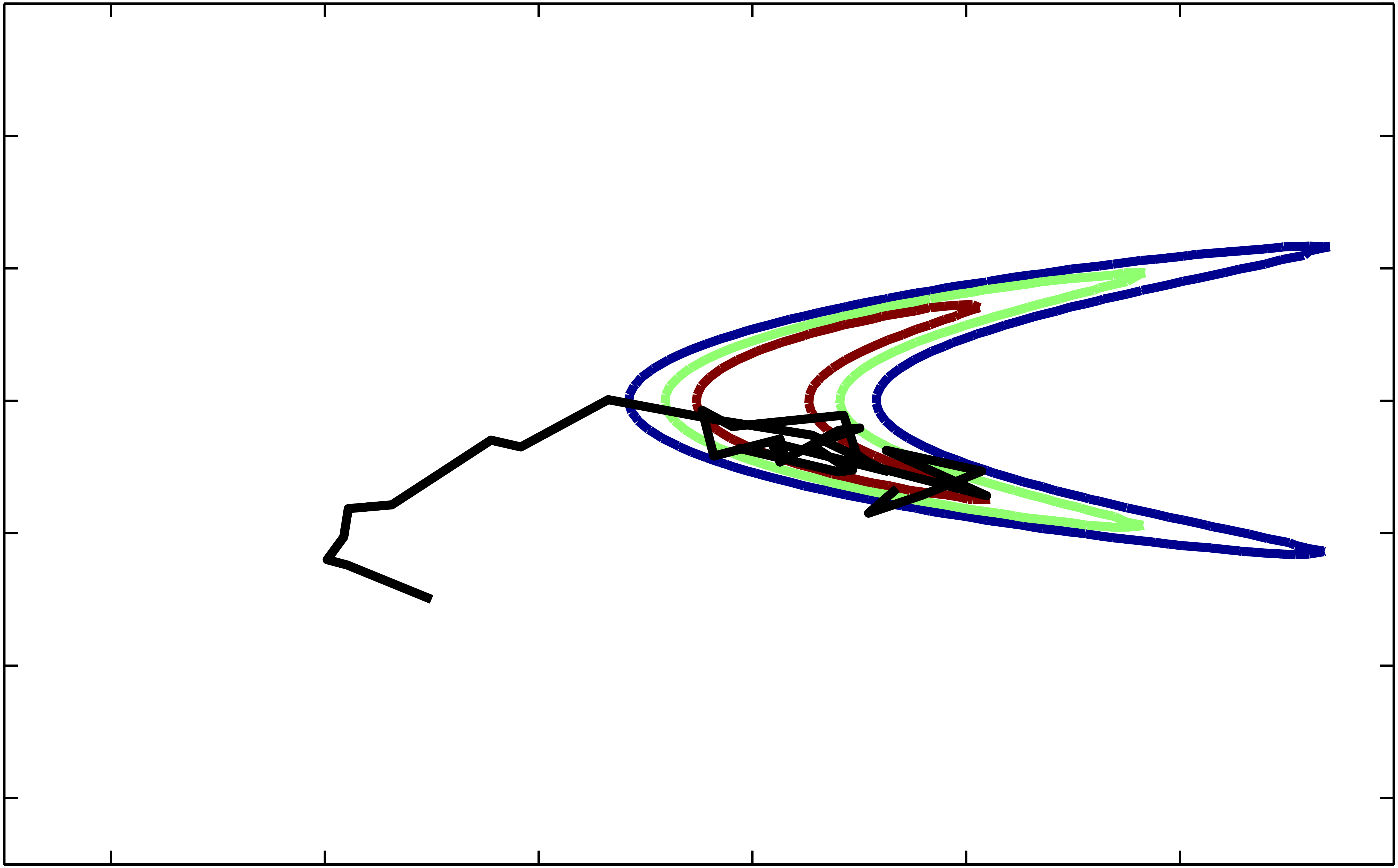
Markov chain Monte Carlo



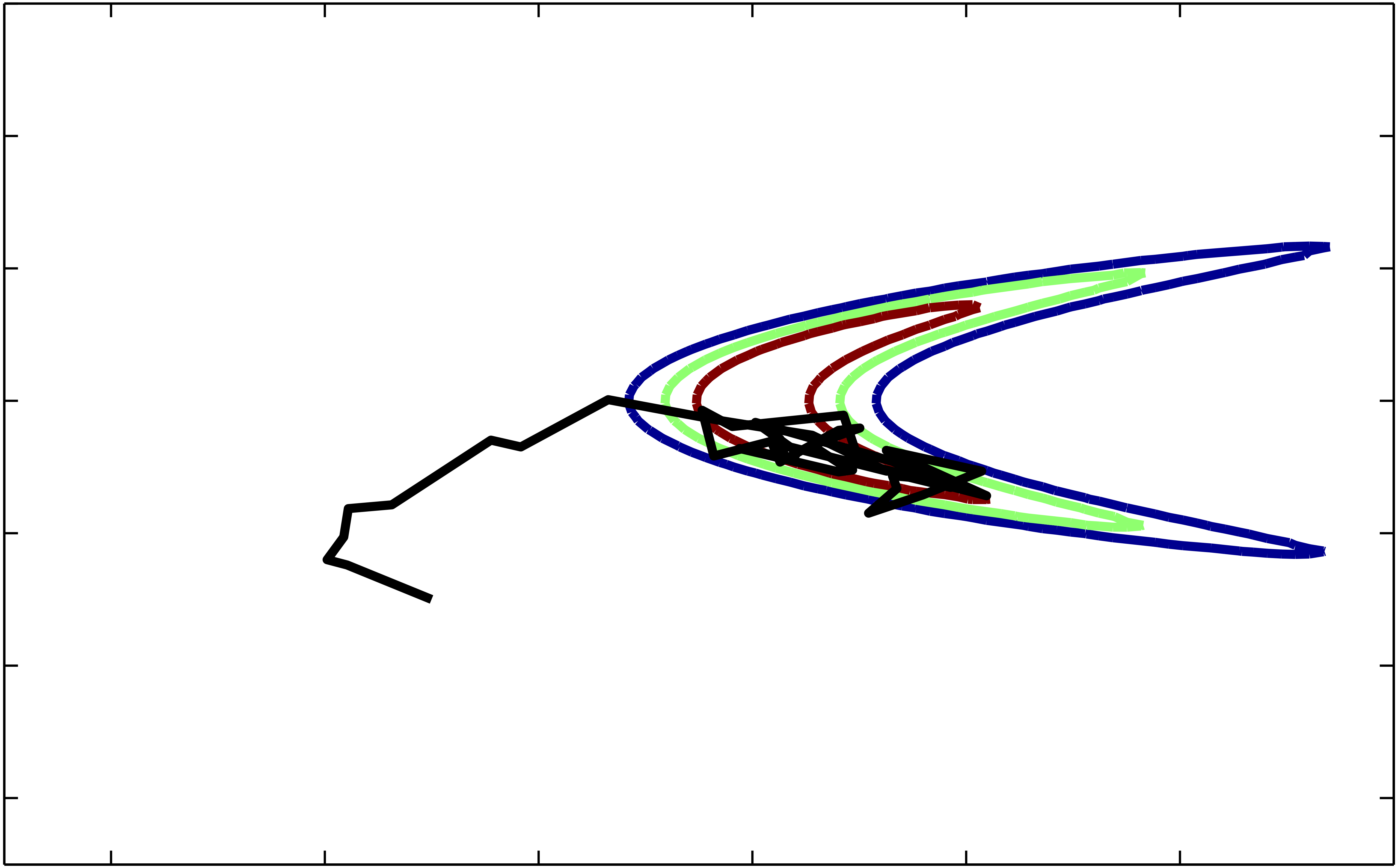
Markov chain Monte Carlo



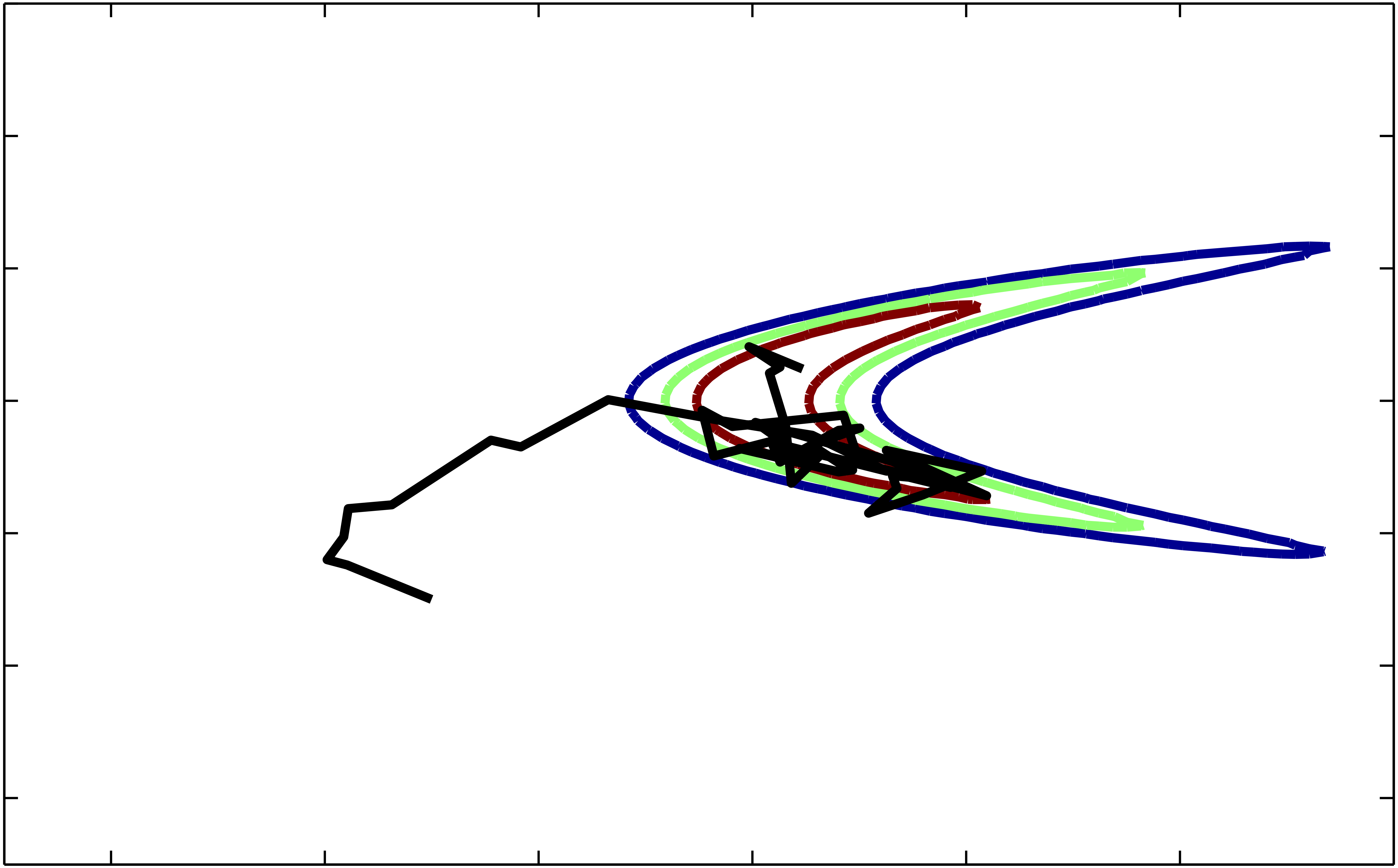
Markov chain Monte Carlo



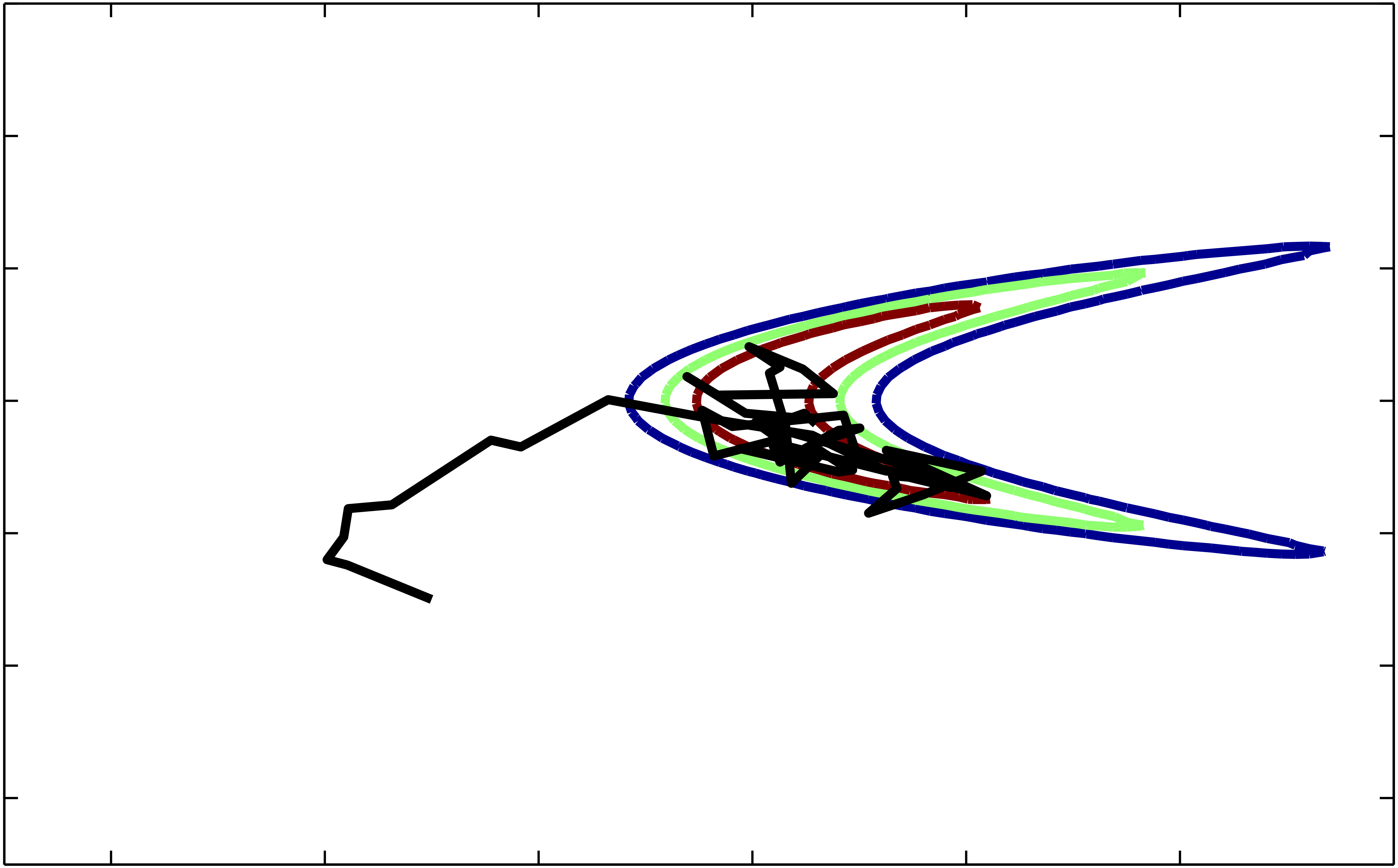
Markov chain Monte Carlo



Markov chain Monte Carlo



Markov chain Monte Carlo



A Discrete Transition Operator

$$\boldsymbol{\pi} = \begin{pmatrix} 3/5 \\ 1/5 \\ 1/5 \end{pmatrix} \qquad \boldsymbol{T} = \begin{pmatrix} 2/3 & 1/2 & 1/2 \\ 1/6 & 0 & 1/2 \\ 1/6 & 1/2 & 0 \end{pmatrix} \quad T(x_i \leftarrow x_j) = T_{ij}$$

$\boldsymbol{\pi}$ is an <u>invariant distribution</u> of $\boldsymbol{T}$, i.e. $\boldsymbol{T}\boldsymbol{\pi} = \boldsymbol{\pi}$

$$\sum_x T(x' \leftarrow x)\,\pi(x) = \pi(x')$$

$\boldsymbol{\pi}$ is the <u>equilibrium distribution</u> of $\boldsymbol{T}$, i.e.

$$\boldsymbol{T}^{100} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 3/5 \\ 1/5 \\ 1/5 \end{pmatrix} = \boldsymbol{\pi}$$

$\boldsymbol{T}$ is <u>ergodic</u>, i.e., for all $x' : \pi(x') > 0$ there exists a K such that $T^K(x' \leftarrow x) > 0$

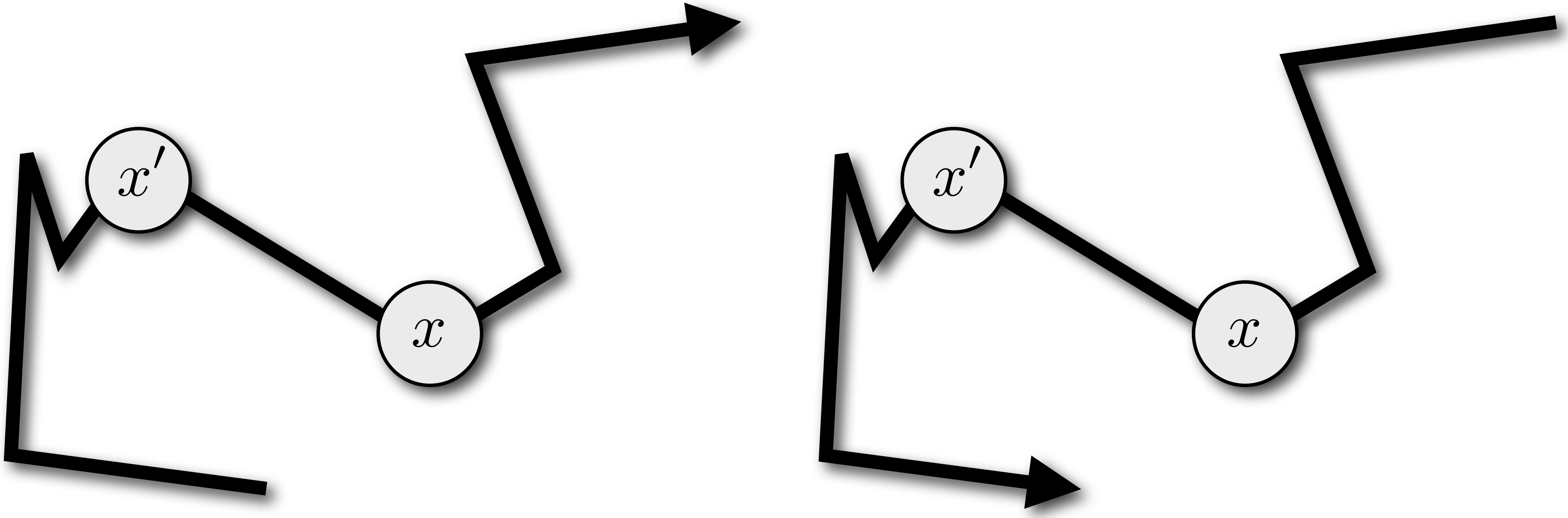
Detailed Balance

In practice, most MCMC transition operators satisfy detailed balance, which is stronger than invariance.

$$T(x' \leftarrow x)\pi(x) = T(x \leftarrow x')\pi(x')$$

$$\sum_x T(x' \leftarrow x)\pi(x) = \sum_x T(x \leftarrow x')\pi(x')$$

$$\sum_x T(x' \leftarrow x)\pi(x) = \pi(x')$$



Metropolis-Hastings

This is the sledgehammer of MCMC. Almost every other method can be seen as a special case of M-H.

Simulate the operator in two steps:

1) Draw a "proposal" from a distribution $q(x' \leftarrow x)$. This is typically something "easy" like $\mathcal{N}(x' \,|\, x, \sigma^2 \mathbb{I})$

2) Accept or reject this move with probability

$$\min\left(1, \frac{q(x \leftarrow x')\,\pi(x')}{q(x' \leftarrow x)\,\pi(x)}\right)$$

The actual transition operator is then

$$T(x' \leftarrow x) = q(x' \leftarrow x) \min\left(1, \frac{q(x \leftarrow x')\,\pi(x')}{q(x' \leftarrow x)\,\pi(x)}\right)$$

Metropolis-Hastings

Things to note:

1) If you reject, the new state is a copy of the current state. Unlike rejection sampling, the rejections count.

2) $\pi(x)$ only needs to be known to a constant.

3) The proposal $q(x' \leftarrow x)$ needs to allow ergodicity.

4) The operator satisfies detailed balance.

Metropolis-Hastings

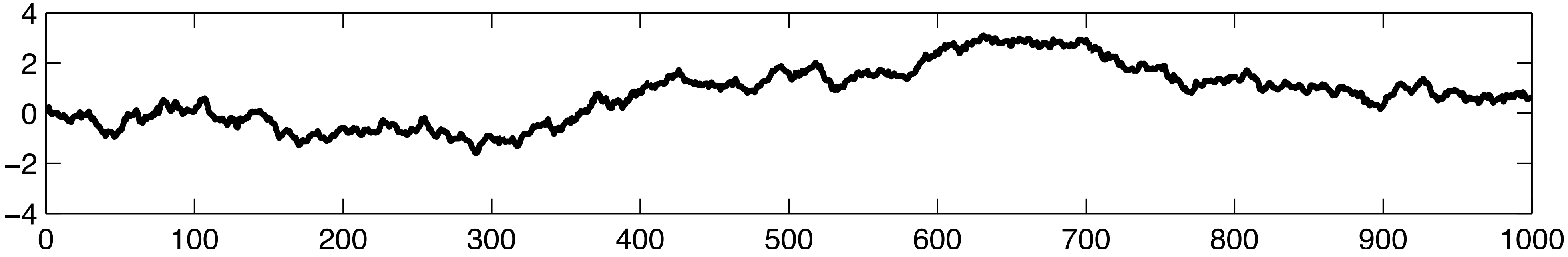
```
function samples = dumb_metropolis(init, log_ptilde, iters, sigma)

D = numel(init);
samples = zeros(D, iters);

state = init;
Lp_state = log_ptilde(state);
for ss = 1:iters
    % Propose
    prop = state + sigma*randn(size(state));
    Lp_prop = log_ptilde(prop);
    if log(rand) < (Lp_prop - Lp_state)
        % Accept
        state = prop;
        Lp_state = Lp_prop;
    end
    samples(:, ss) = state(:);
end
```

Effect of M-H Step Size

```
sigma = @(s) plot(dumb_metropolis(0, @(x) -0.5*x*x, 1e3, s));
```

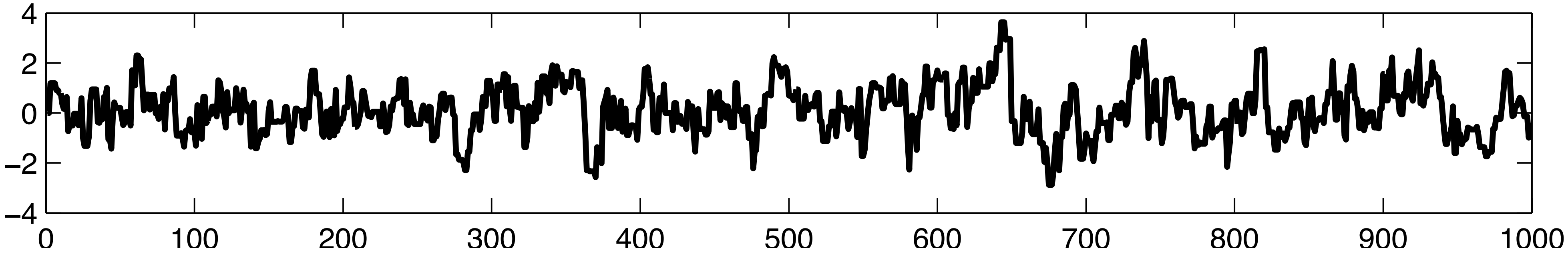
`sigma(0.1)`

99.8% accepts



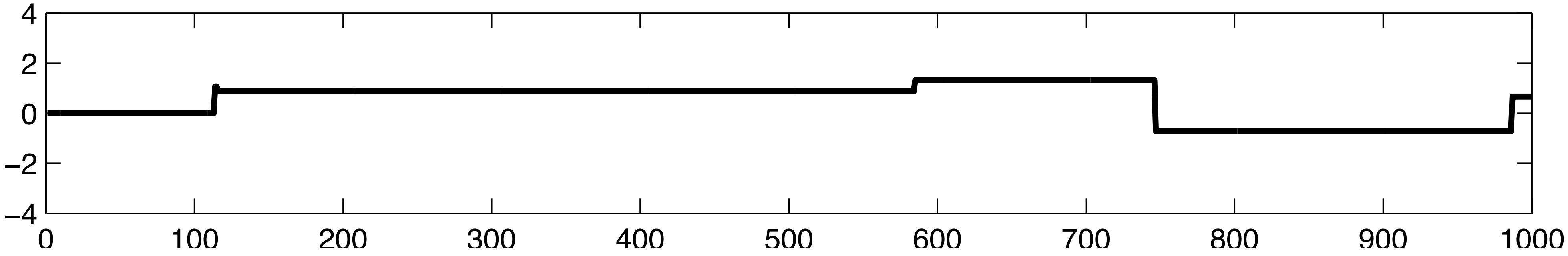
`sigma(1)`

68.4% accepts

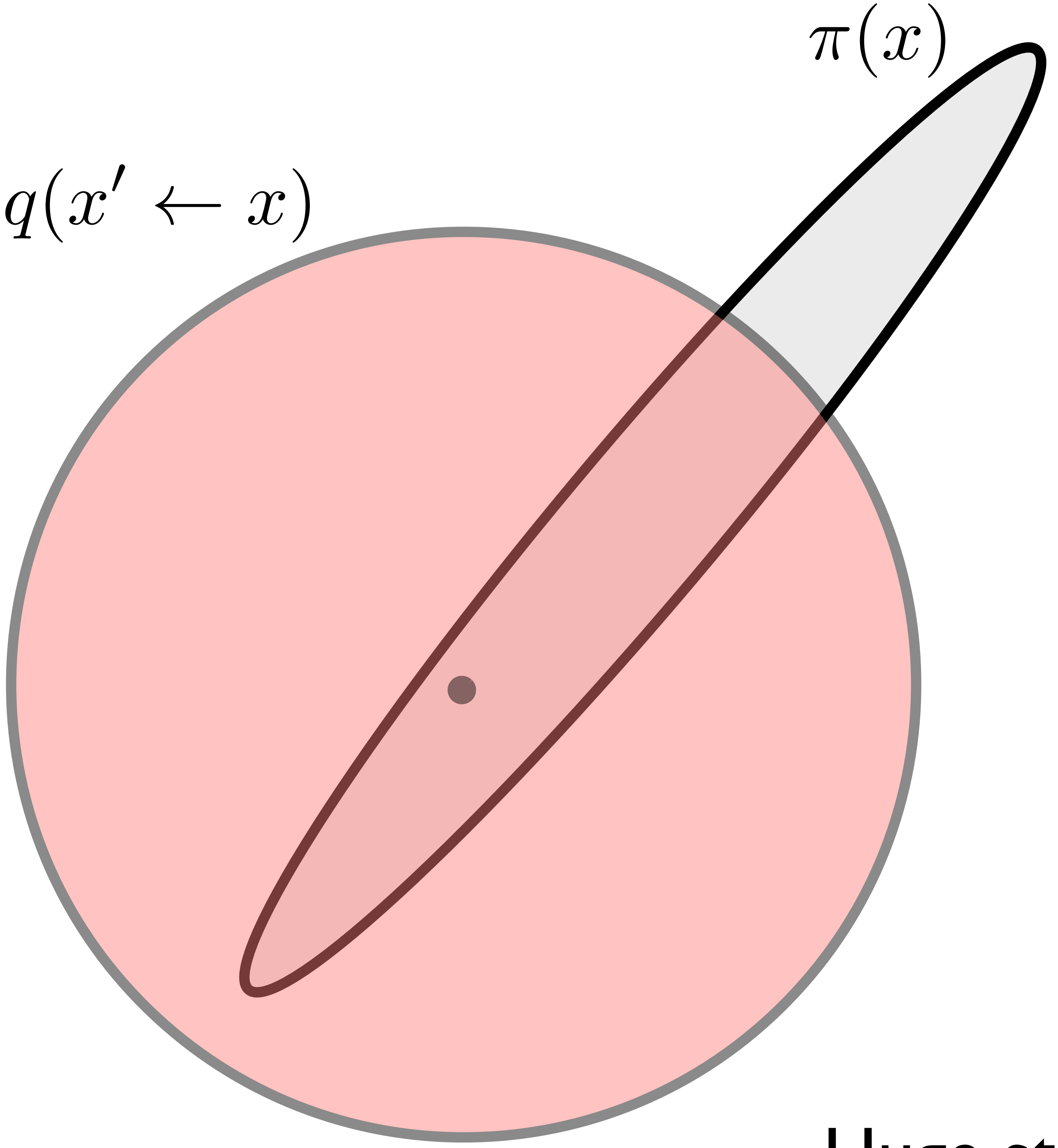


`sigma(100)`

0.5% accepts

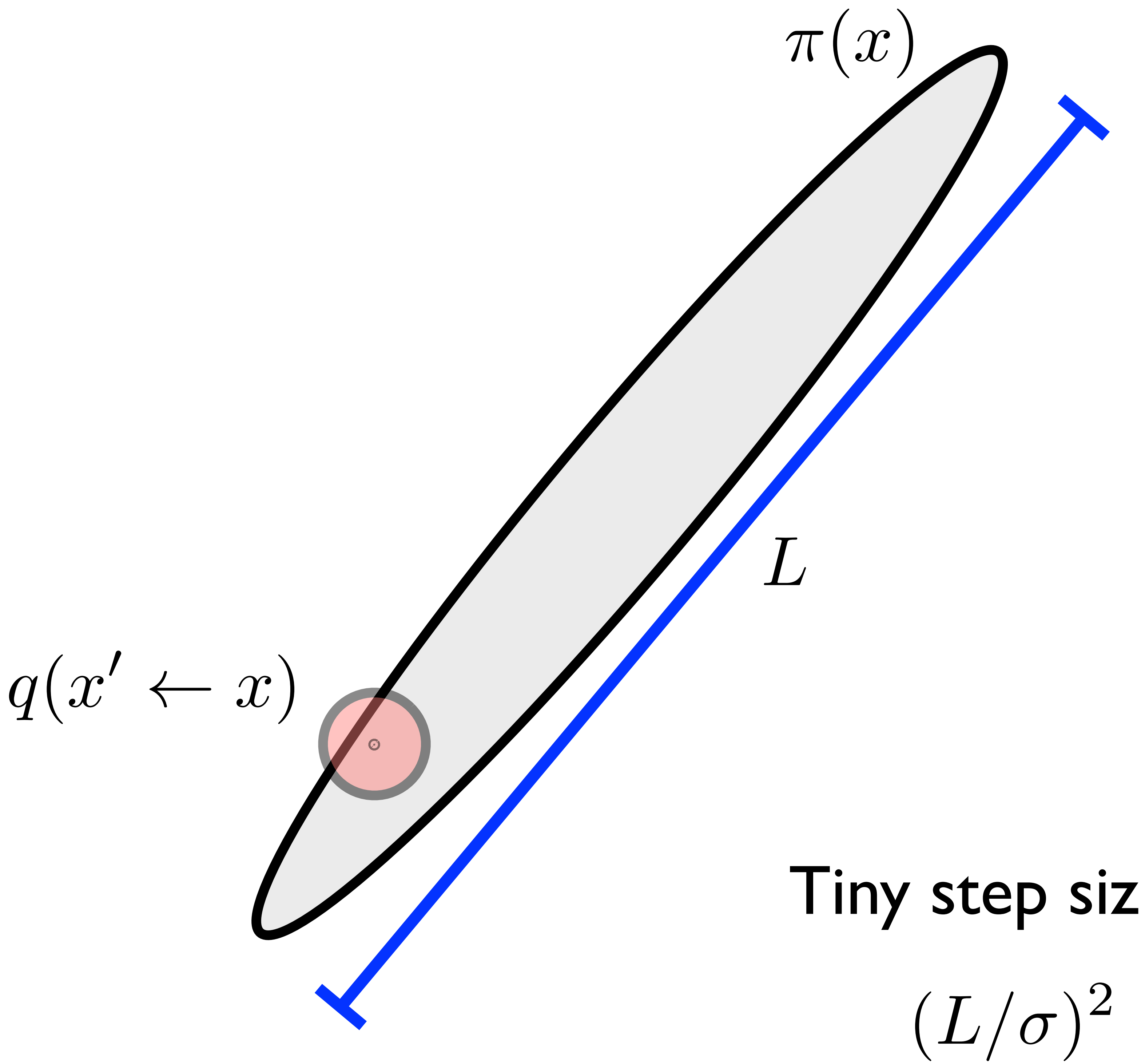


Effect of M-H Step Size



Huge step size = lots of rejections

Effect of M-H Step Size



Tiny step size = slow diffusion

$(L/\sigma)^2$ steps

Gibbs Sampling

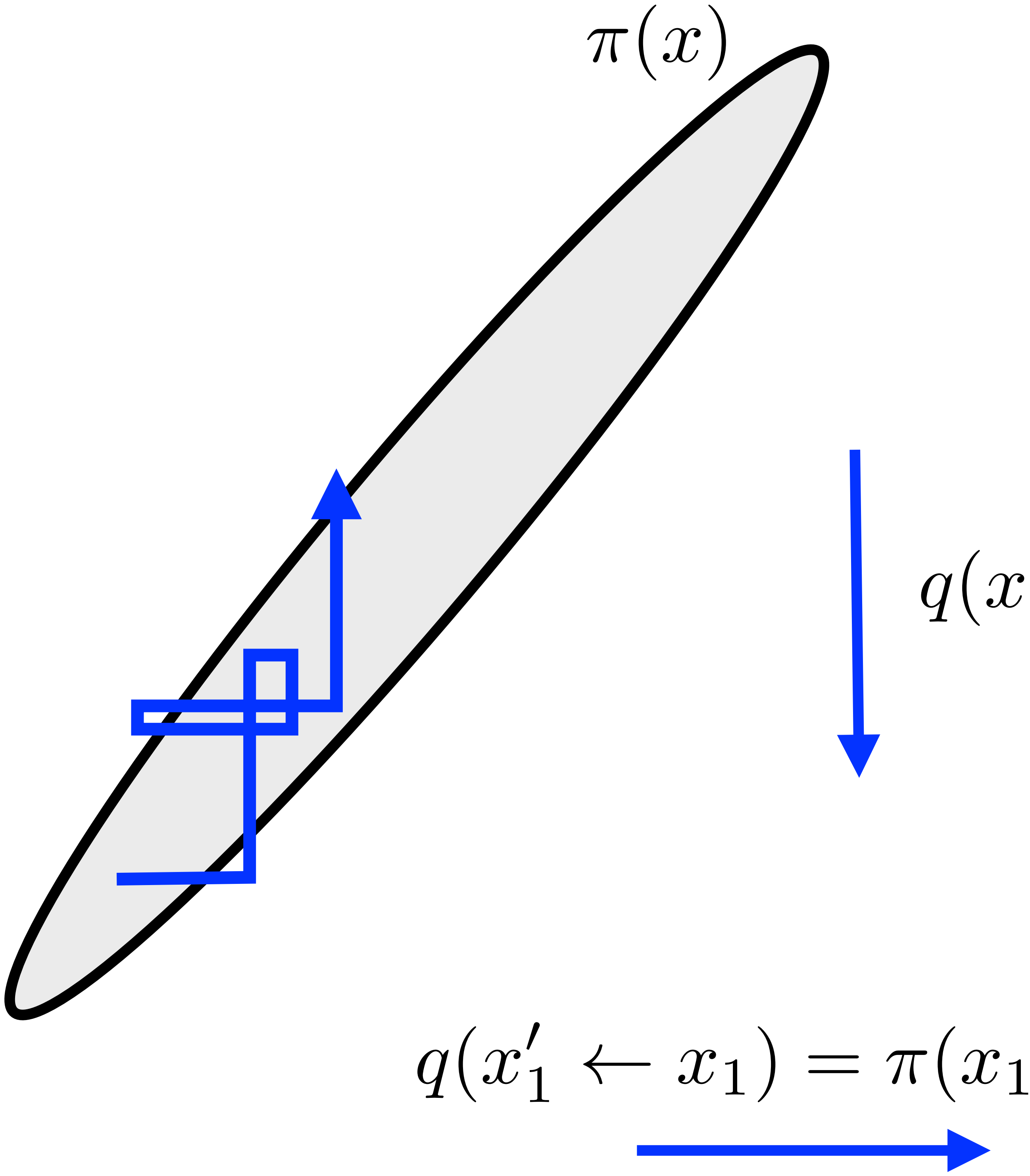
One special case of Metropolis-Hastings is very popular and does not require any choice of step size.

Gibbs sampling is the composition of a sequence of M-H transition operators, each of which acts upon a single component of the state space.

By themselves, these operators are not ergodic, but in aggregate they typically are.

Most commonly, the proposal distribution is taken to be the conditional distribution, given the rest of the state. This causes the acceptance ratio to always be one and is often easy because it is low-dimensional.

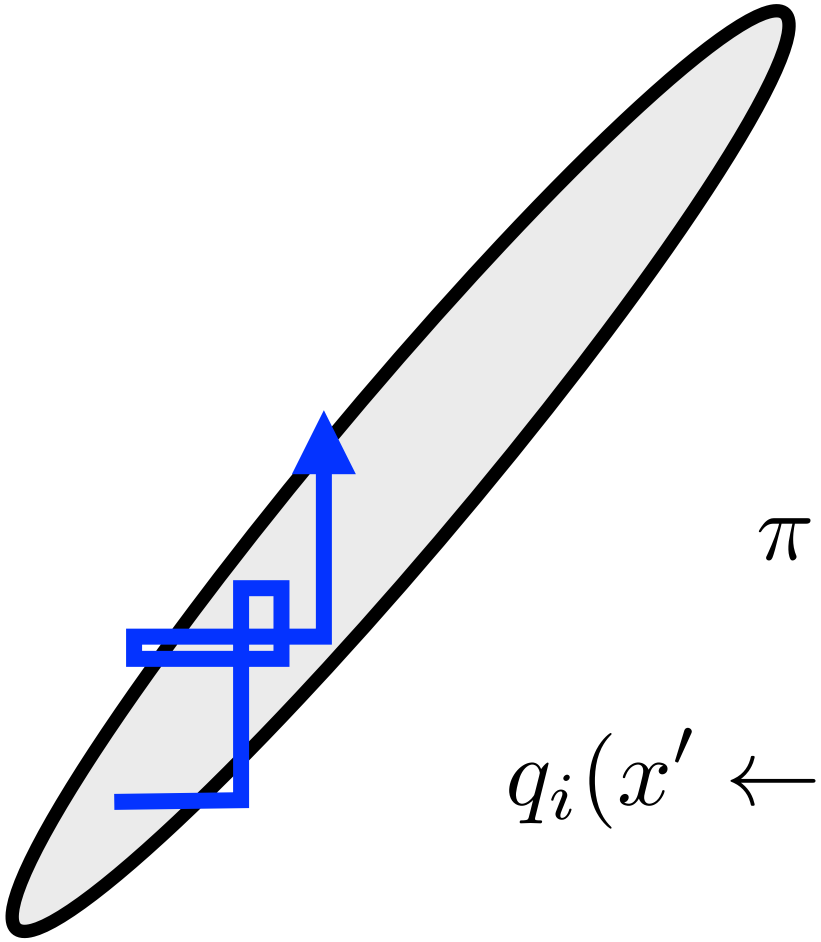
Gibbs Sampling



$$q(x_2' \leftarrow x_2) = \pi(x_2 \,|\, x_1)$$

$$q(x_1' \leftarrow x_1) = \pi(x_1 \,|\, x_2)$$

Gibbs Sampling



$$\pi(x) = \pi(x_i \,|\, x_{j\neq i})\pi(x_{j\neq i})$$

$$q_i(x' \leftarrow x) = \pi(x'_i \,|\, x_{j\neq i})\delta(x'_{j\neq i} - x_{j\neq i})$$

$$T_i(x' \leftarrow x) = q(x' \leftarrow x)\min\left(1, \frac{q(x \leftarrow x')\,\pi(x')}{q(x' \leftarrow x)\,\pi(x)}\right)$$

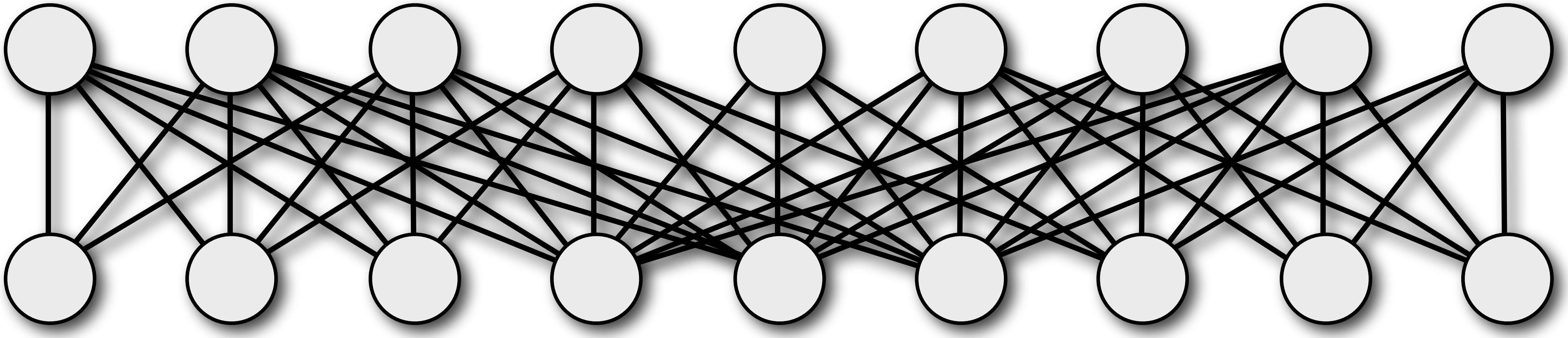
$$\frac{q(x \leftarrow x')\,\pi(x')}{q(x' \leftarrow x)\,\pi(x)} = \frac{\pi(x_i \,|\, x_{j\neq i})\pi(x'_i \,|\, x_{j\neq i})\pi(x_{j\neq i})}{\pi(x'_i \,|\, x_{j\neq i})\pi(x_i \,|\, x_{j\neq i})\pi(x_{j\neq i})} = 1$$

Gibbs Sampling

Sometimes, it’s really easy: if there are only a small number of possible states, they can be enumerated and normalized easily, e.g. binary hidden units in a restricted Boltzmann machine.

When groups of variables are jointly sampled given everything else, it is called “block-Gibbs” sampling.

Parallelization of Gibbs updates is possible if the conditional independence structure allows it. RBMs are a good example of this also.



Gibbs Sampling

Component-wise M-H moves are also allowed, inexplicably called "Metropolis-within-Gibbs". You will not call it this because you recognize that it is silly.

Despite our earlier criticisms of rejection sampling, it can work well in the inner loop of Gibbs sampling.

Off the shelf tools such as WinBUGS and OpenBUGS sometimes do this. If you have a simple model, you may just be able to use one of these tools directly.

If you go this route, look into something called "adaptive rejection sampling", which builds a tight hull around the distribution on the fly.

Summary So Far

We don’t have to start our sampler over every time!

We can use our “easy” distribution to get correlated samples from the “hard” distribution.

Even though correlated, they still have the correct marginal distribution, so we get the right estimator.

Designing an MCMC operator sounds harder than it is.

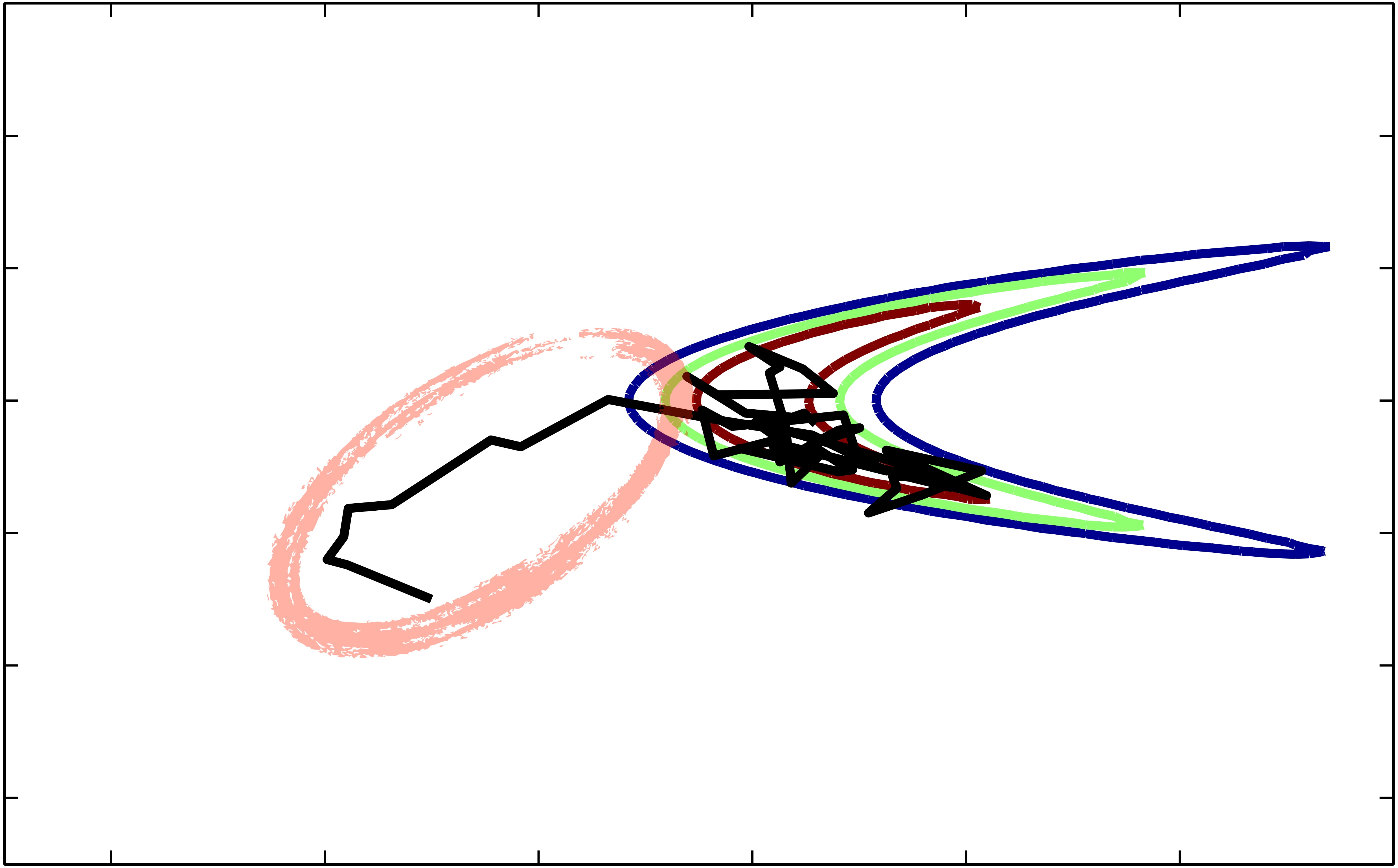
Metropolis-Hastings can require some tuning.

Gibbs sampling can be an easy version to implement.

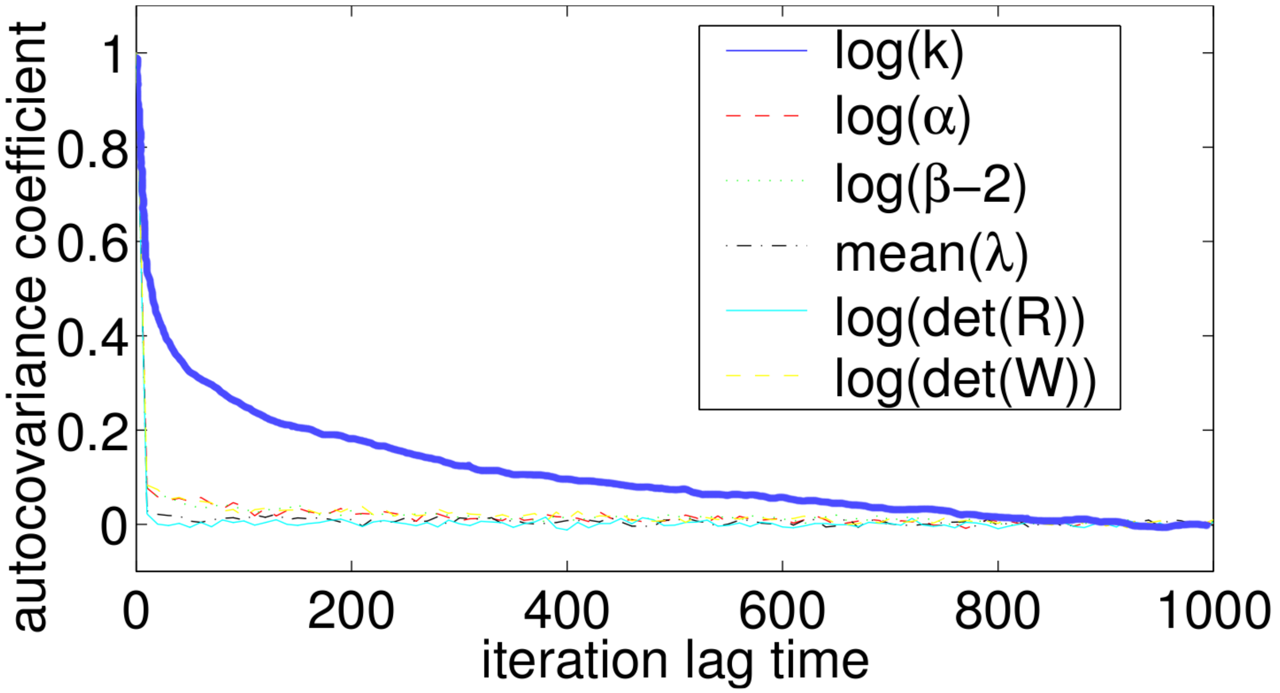
Frequently Asked Questions

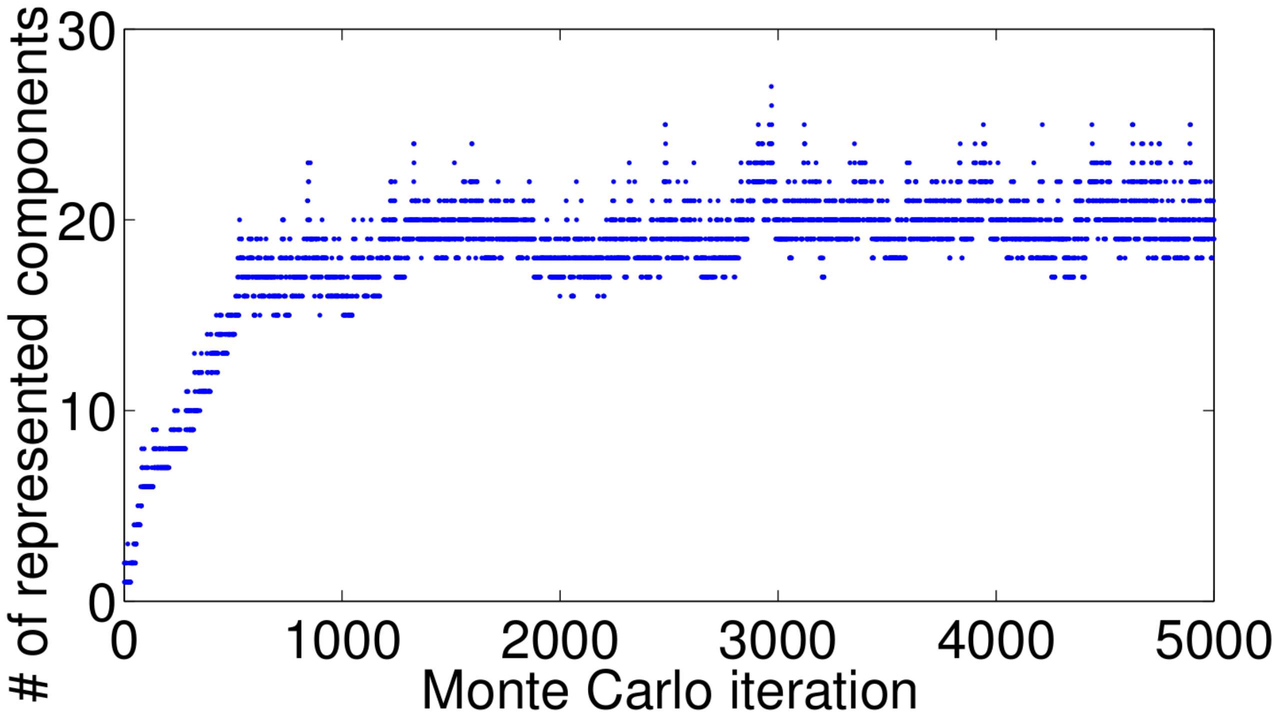
- Has my MCMC run long enough yet?
- How should I set M-H parameters?
- Can I adapt my step sizes?
- How many chains should I run?
- Should I use all of the samples?
- How do I diagnose bugs in my code?

Discarding the "Burn-In"



Heuristics for Mixing





Rasmussen (2000)

- Plot autocorrelations of scalar variables.
- Plot traces of variables in the model.
- Run several chains from different starting points.
- Examine the "effective number of samples" via R-CODA.

Charles J. Geyer, "Practical Markov chain Monte Carlo", *Statistical Science*, 7(4) 473-483, 1992.

Setting M-H Step Sizes

- Typically, this is done via preliminary runs.
- You want an acceptance rate of around 50%.
- You cannot adapt the step size according to the chain's history - it would no longer be Markovian.
- There are adaptive methods, but they are beyond the scope of this tutorial.
- Use slice sampling instead. More on this later.

How Many Chains?

- Multiple chains are useful for diagnosis.
- Multiple chains allow trivial parallelization.
- All else being equal, run longer.
- “Please tell me how many to use?” I tend to use either one or ten.

Do I Use All of the Samples?

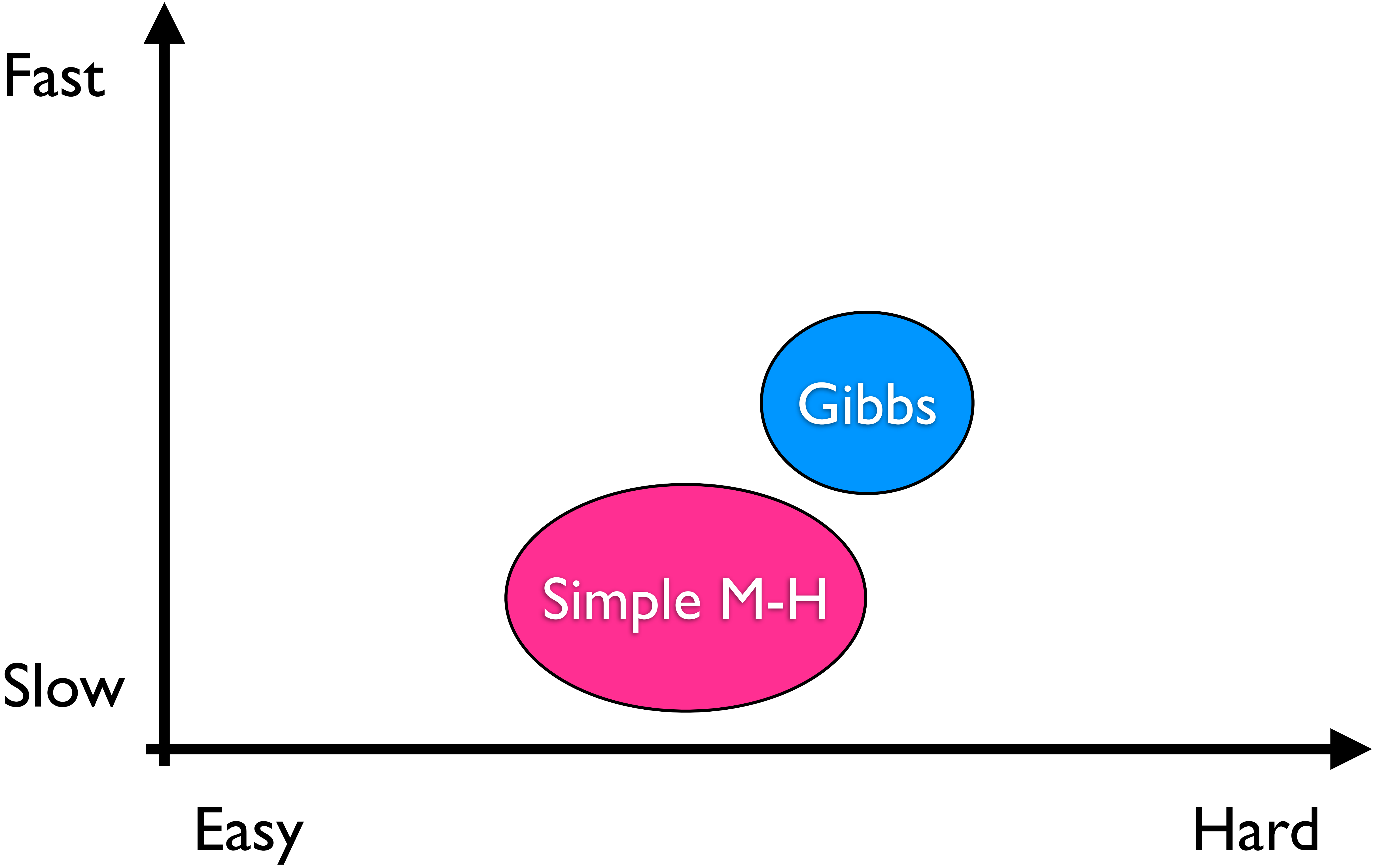
- Some authors recommend “thinning” and only taking every Nth sample from the Markov chain.
- It is true that these samples will be closer to being independently from the stationary distribution.
- It is also true that thinning strictly worsens your estimator.
- So, don’t thin unless the computational cost is dominated by the function whose expectation you are trying to evaluate, i.e., thin only if $f(x)$ costs vastly more than $\pi(x)$.

Diagnosing Bugs

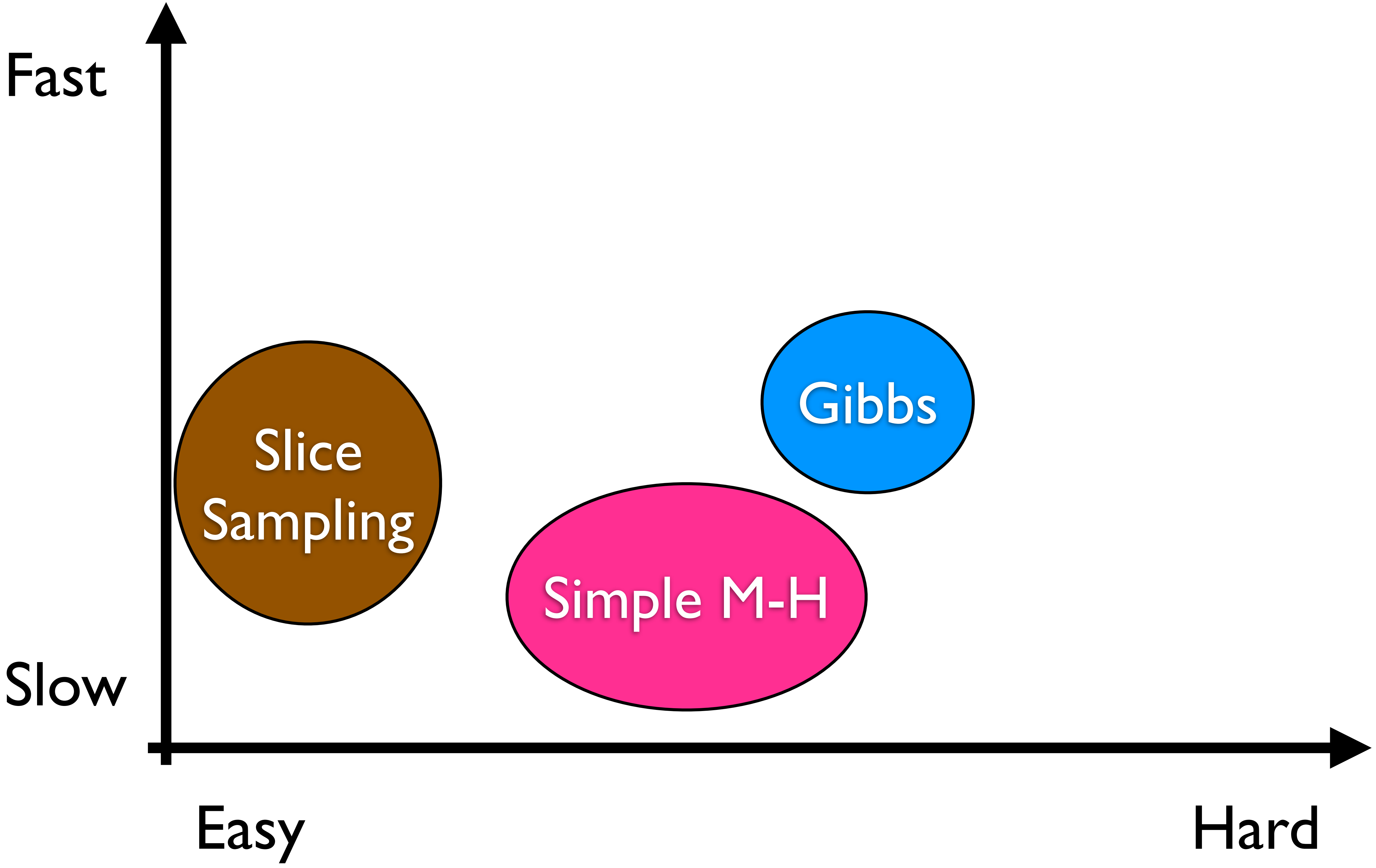
- Generate synthetic data (fantasies!) with known parameters and see if you can infer them.
- Even better: run the "Geweke Test". Generate fantasy data as part of your Markov chain and ensure that the histograms of your parameters match your priors.
- Think of this as "finite difference validation for MCMC".

John Geweke, "Getting it Right: joint distribution tests of posterior simulators", *JASA* 99(467), 799-804, 2004.

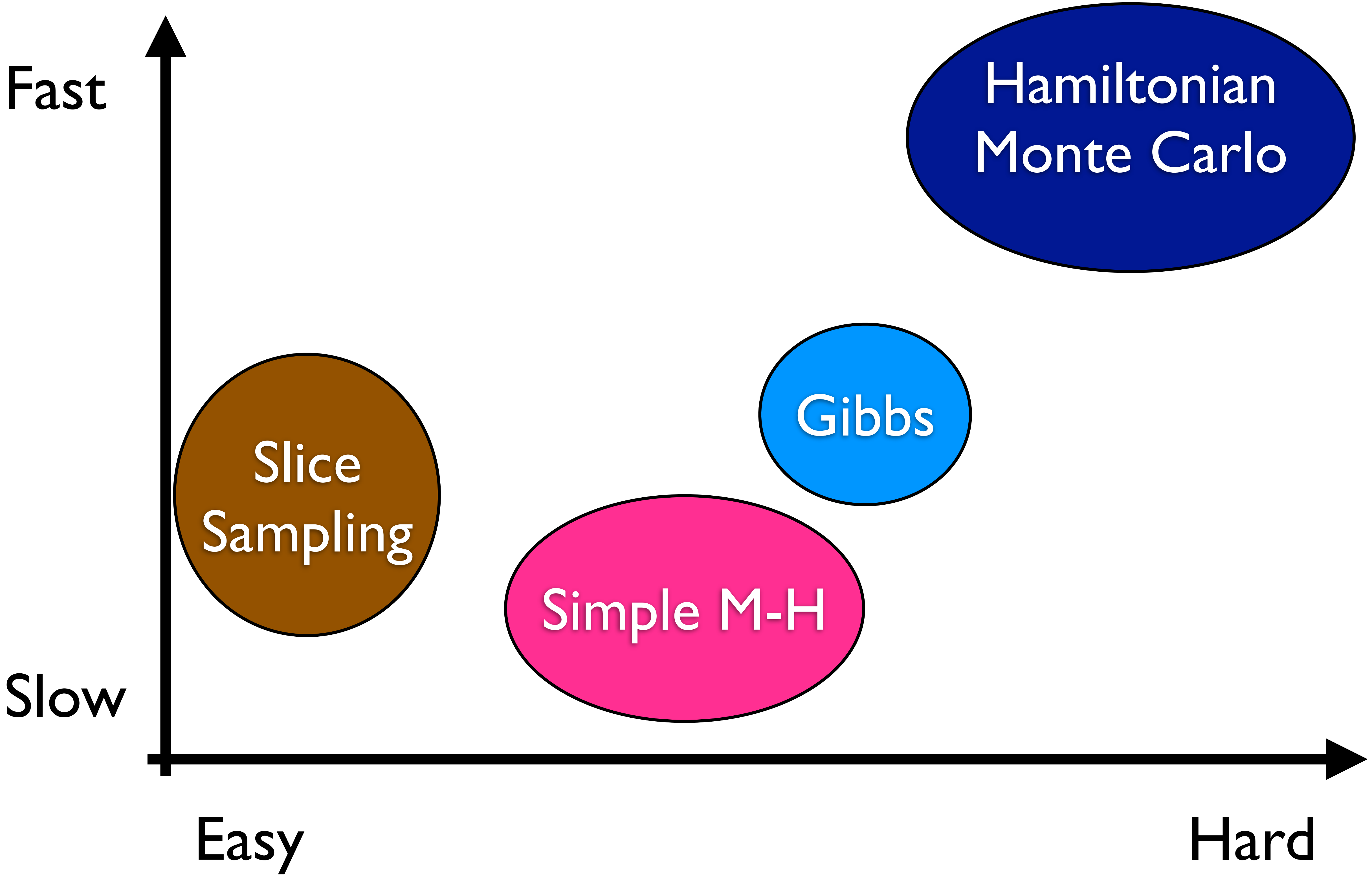
An MCMC Cartoon



An MCMC Cartoon



An MCMC Cartoon



Fast

Slow

Easy

Hard

Hamiltonian Monte Carlo

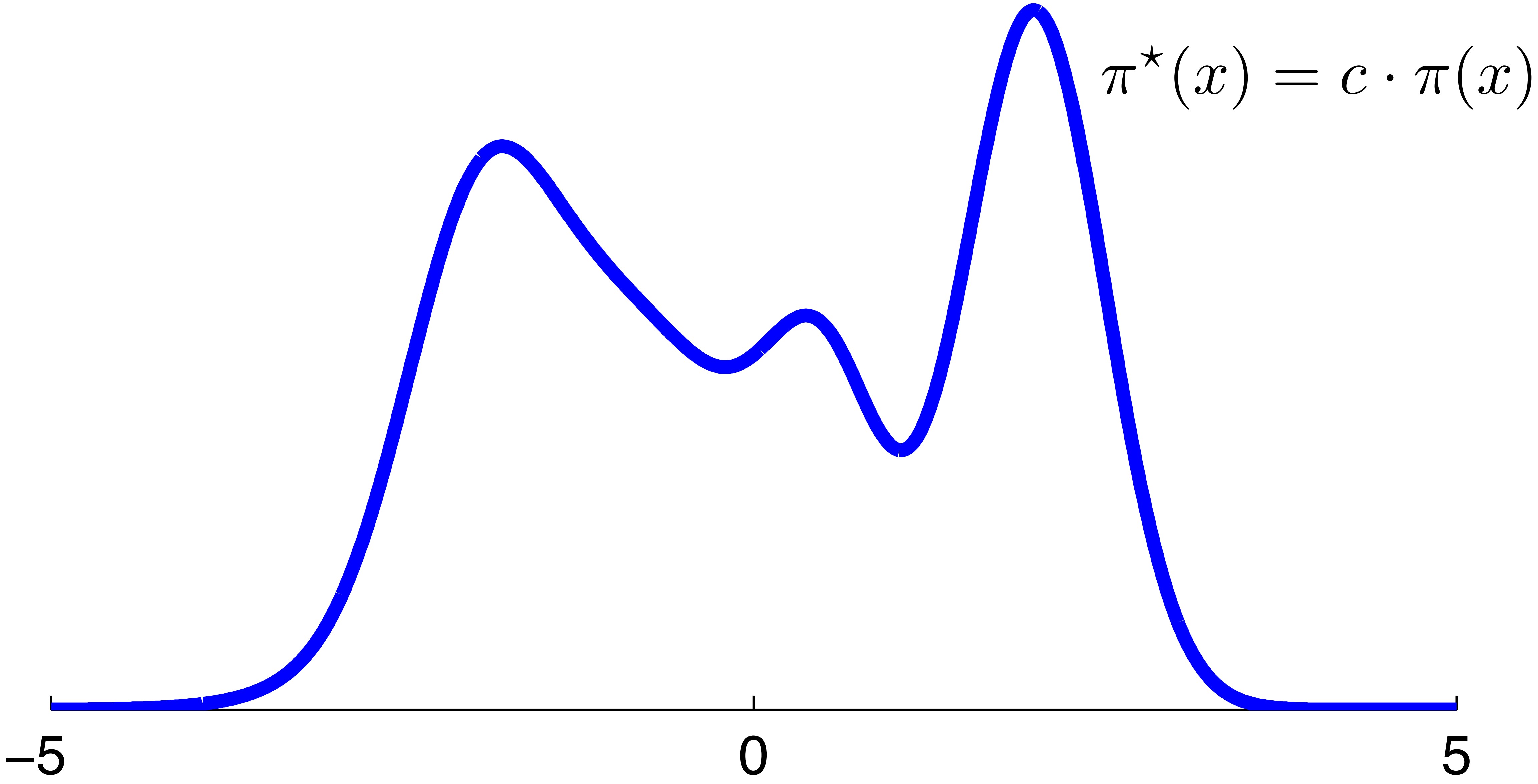
Gibbs

Slice Sampling

Simple M-H

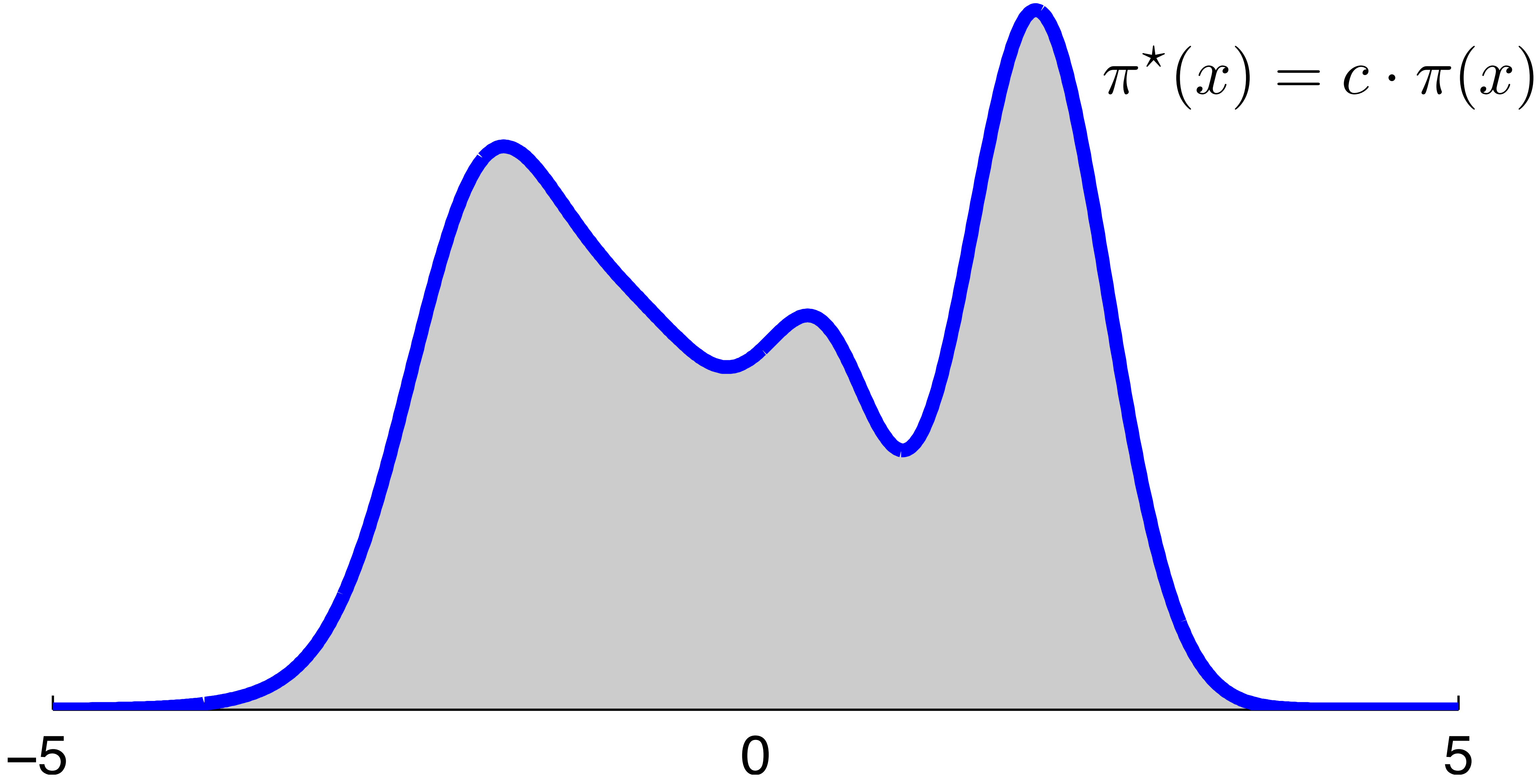
Slice Sampling

An auxiliary variable MCMC method that requires almost no tuning. Remember back to the beginning...



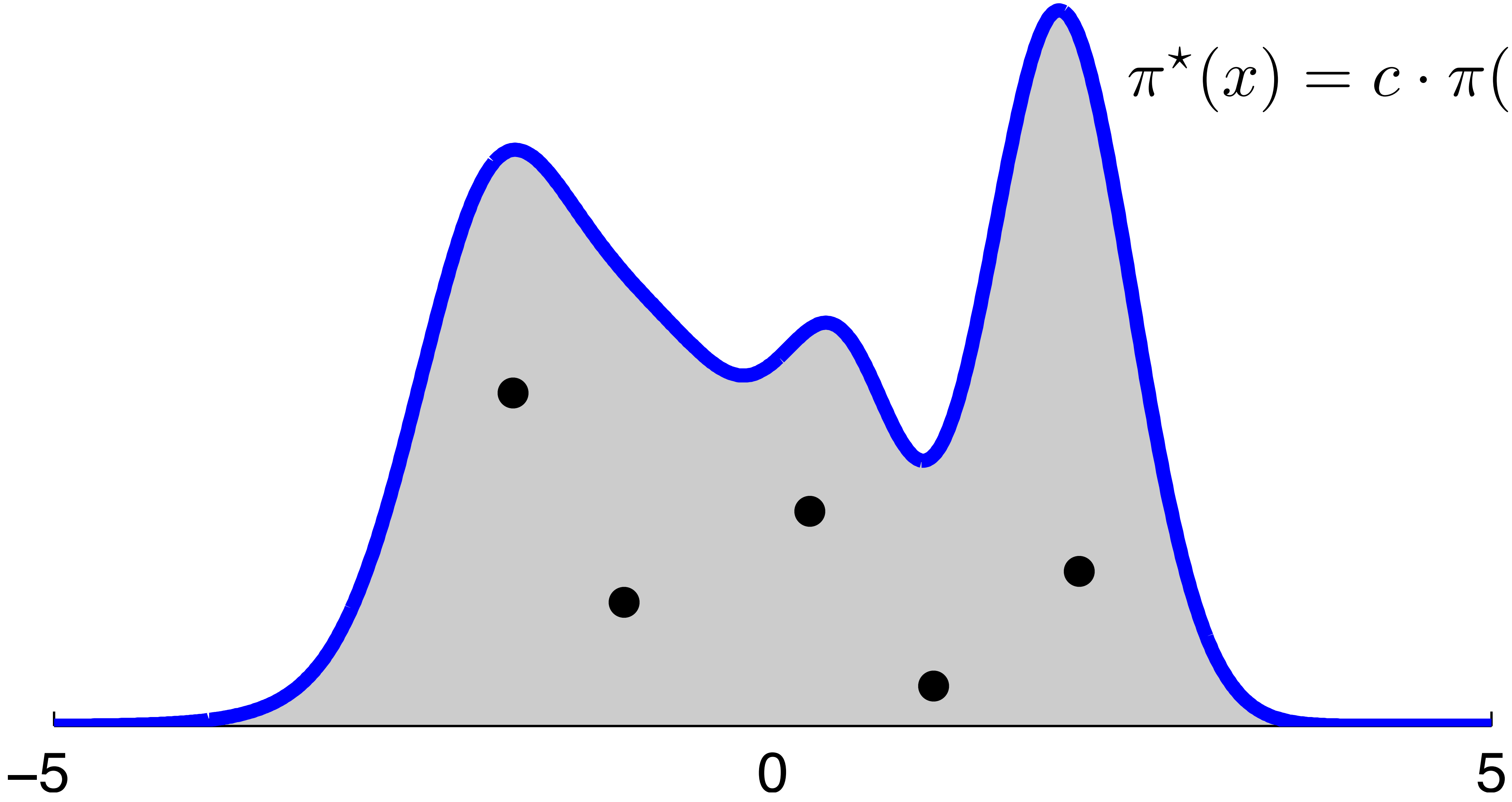
Slice Sampling

An auxiliary variable MCMC method that requires almost no tuning. Remember back to the beginning...



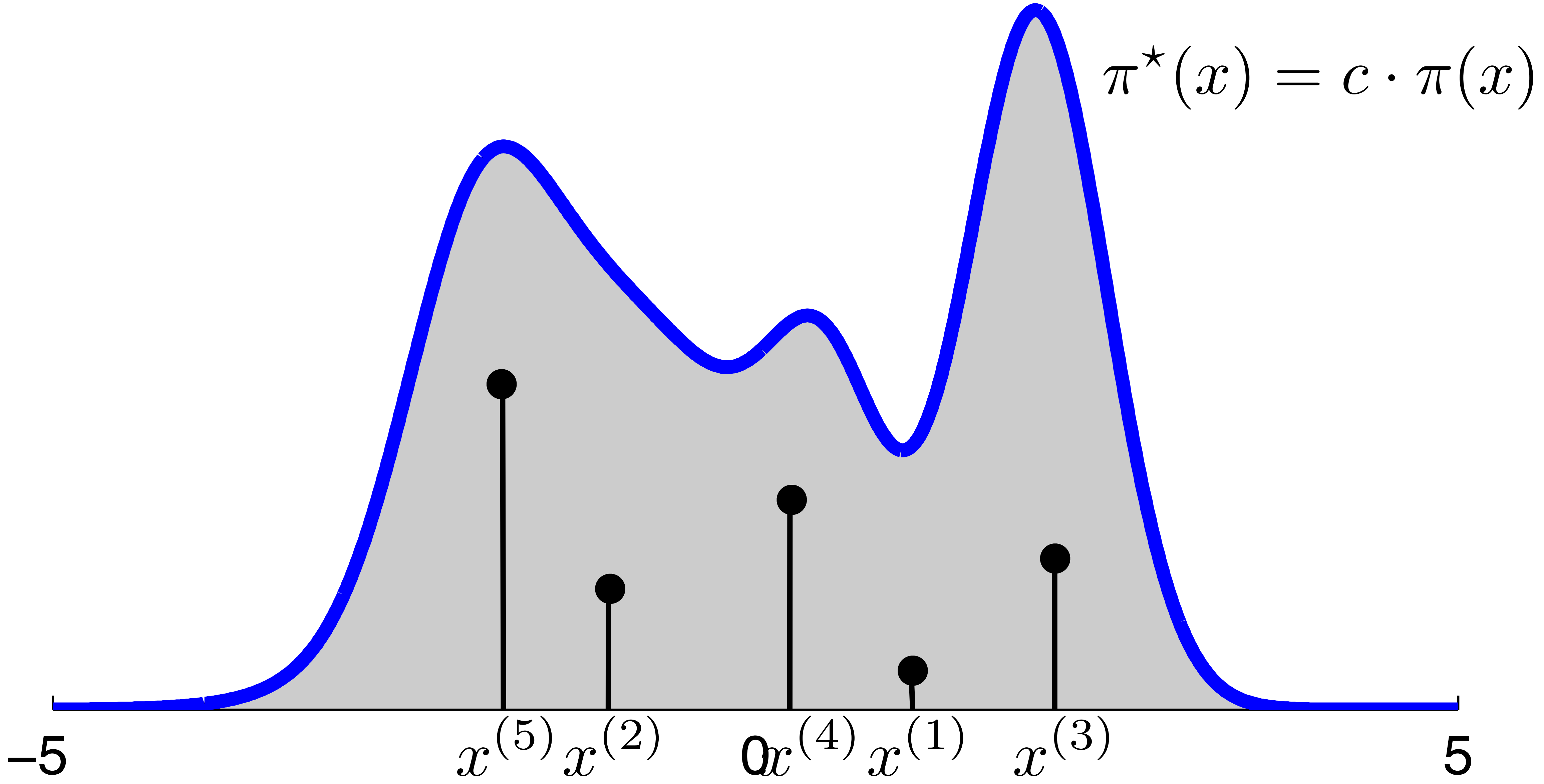
Slice Sampling

An auxiliary variable MCMC method that requires almost no tuning. Remember back to the beginning...



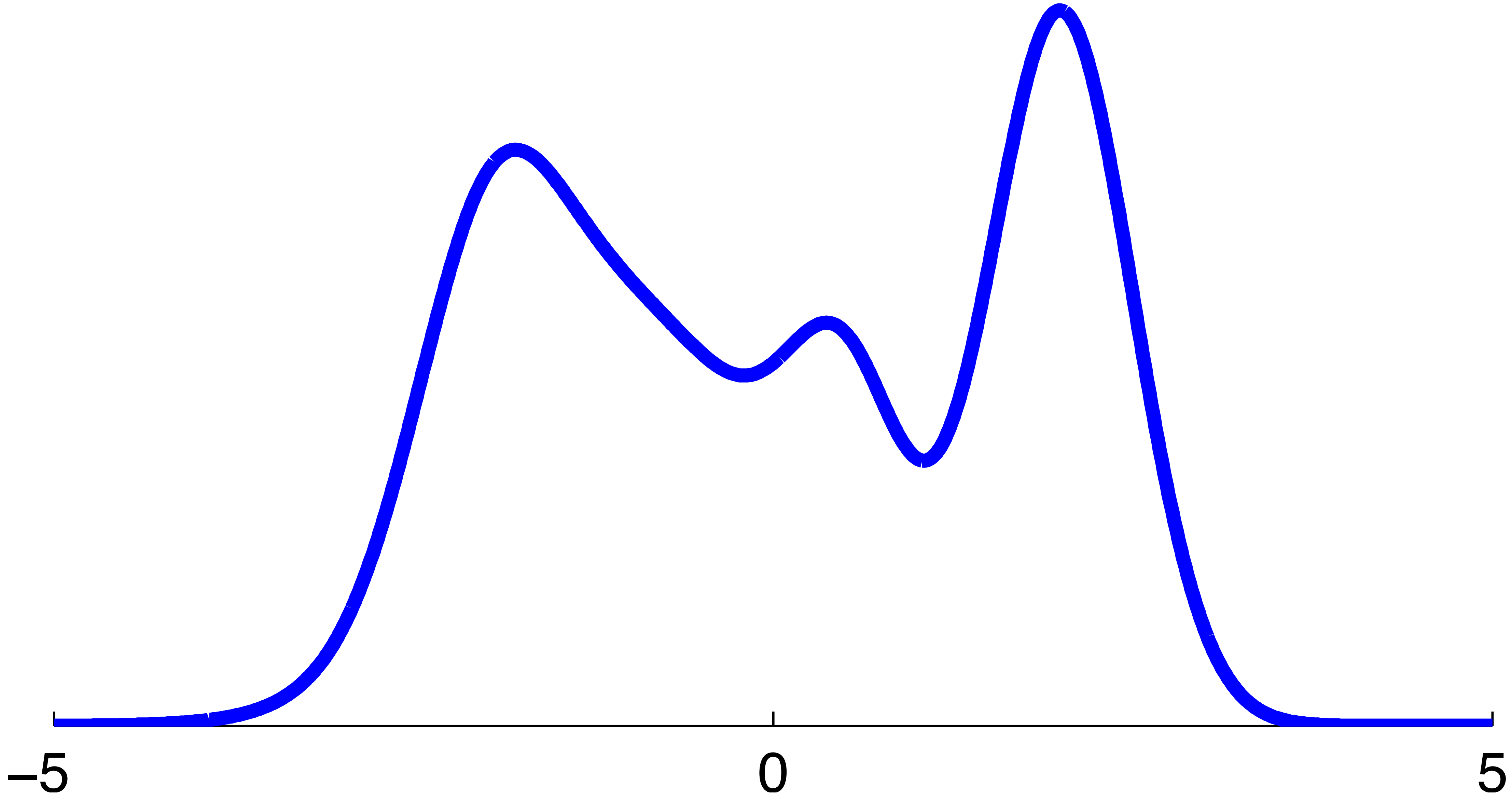
Slice Sampling

An auxiliary variable MCMC method that requires almost no tuning. Remember back to the beginning...



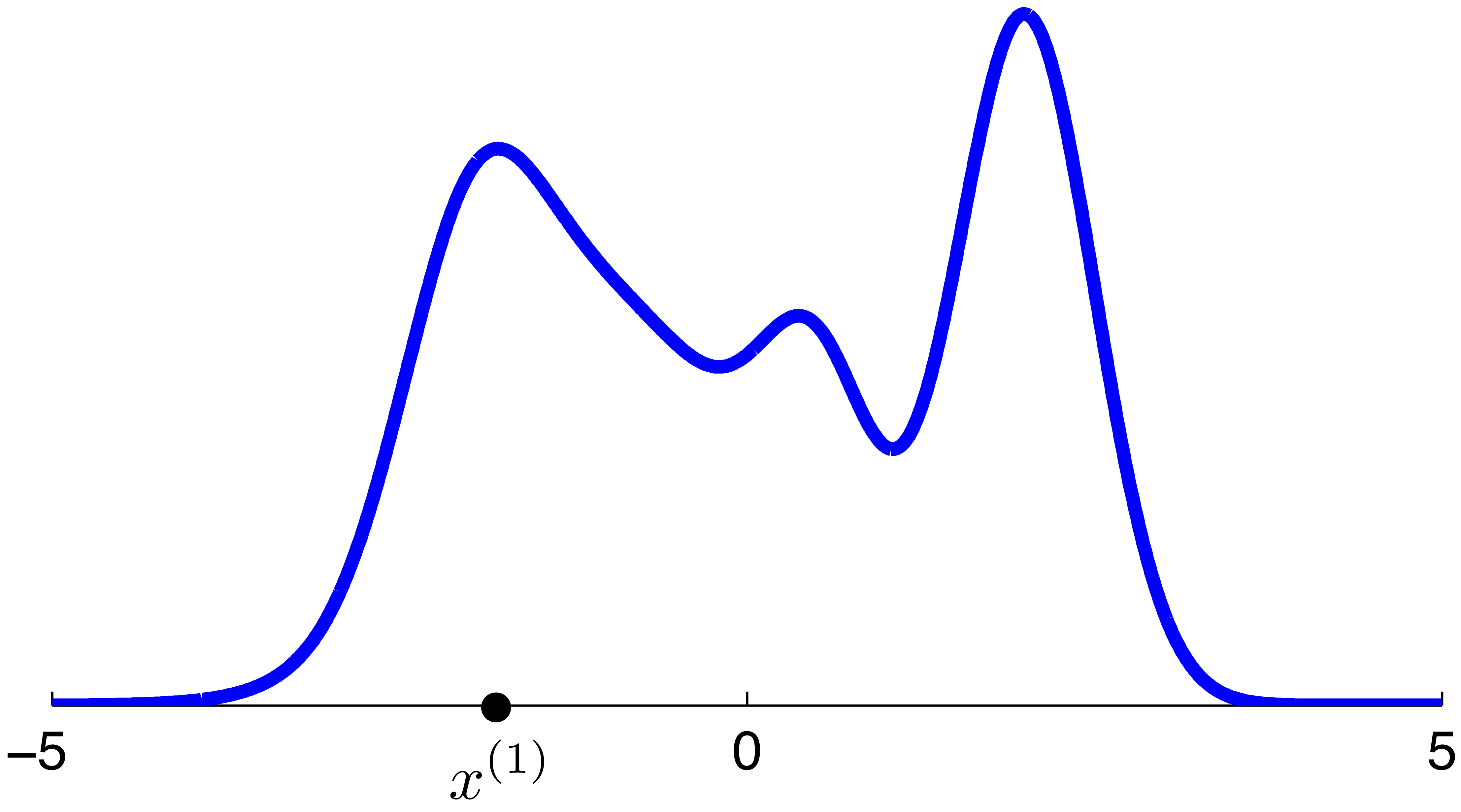
Slice Sampling

Define a Markov chain that samples uniformly from the area beneath the curve. This means that we need to introduce a “height” into the MCMC sampler.



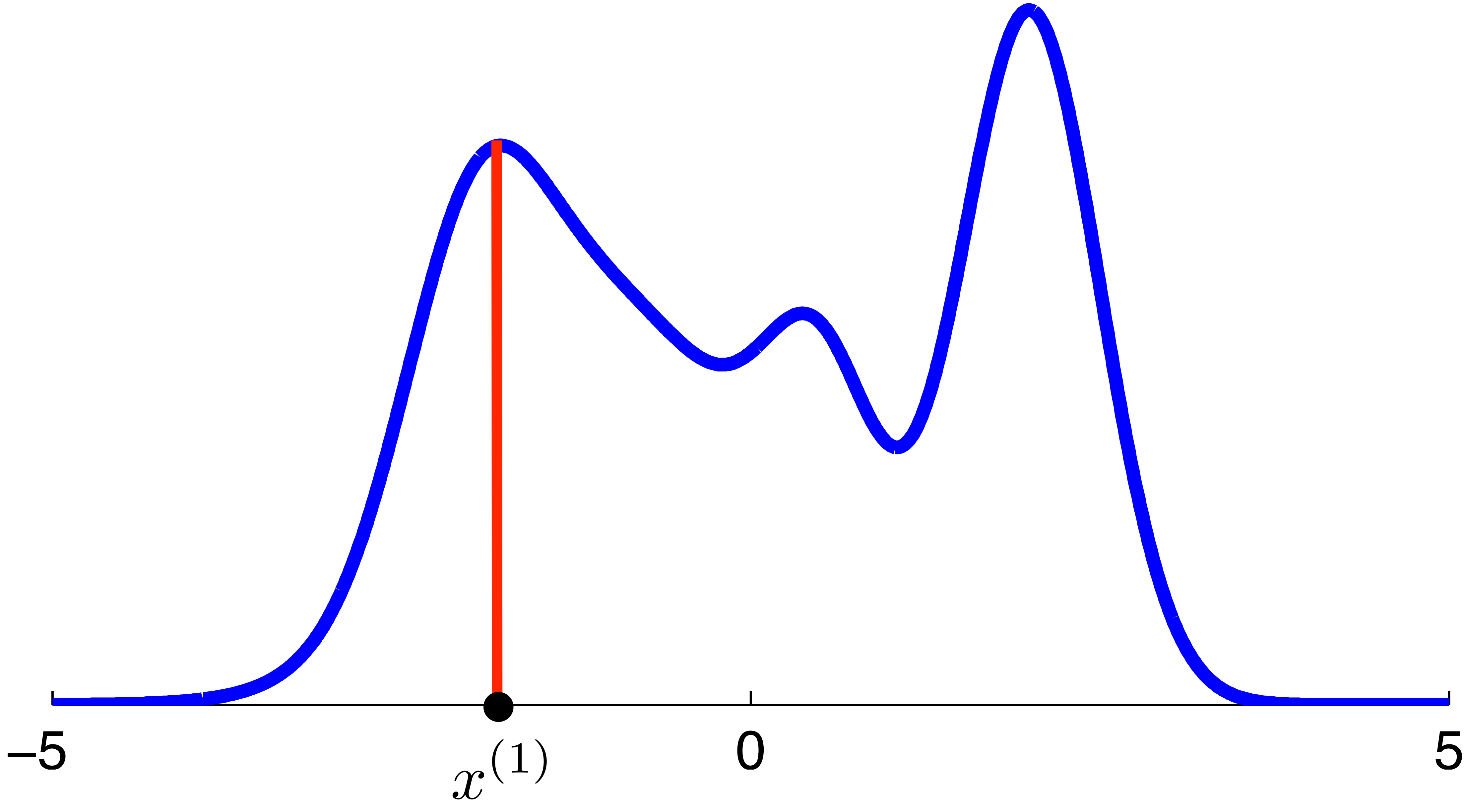
Slice Sampling

Define a Markov chain that samples uniformly from the area beneath the curve. This means that we need to introduce a “height” into the MCMC sampler.



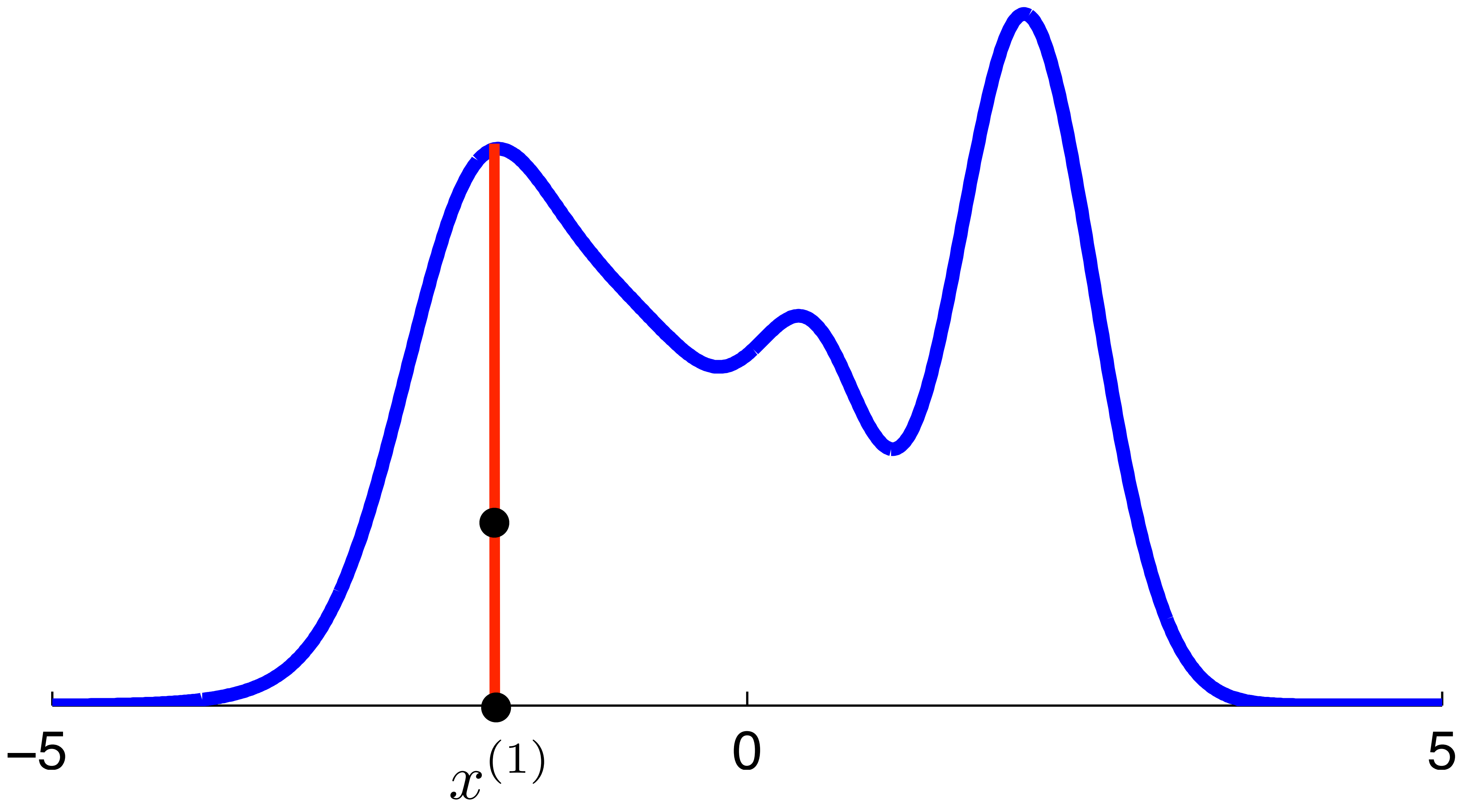
Slice Sampling

Define a Markov chain that samples uniformly from the area beneath the curve. This means that we need to introduce a "height" into the MCMC sampler.



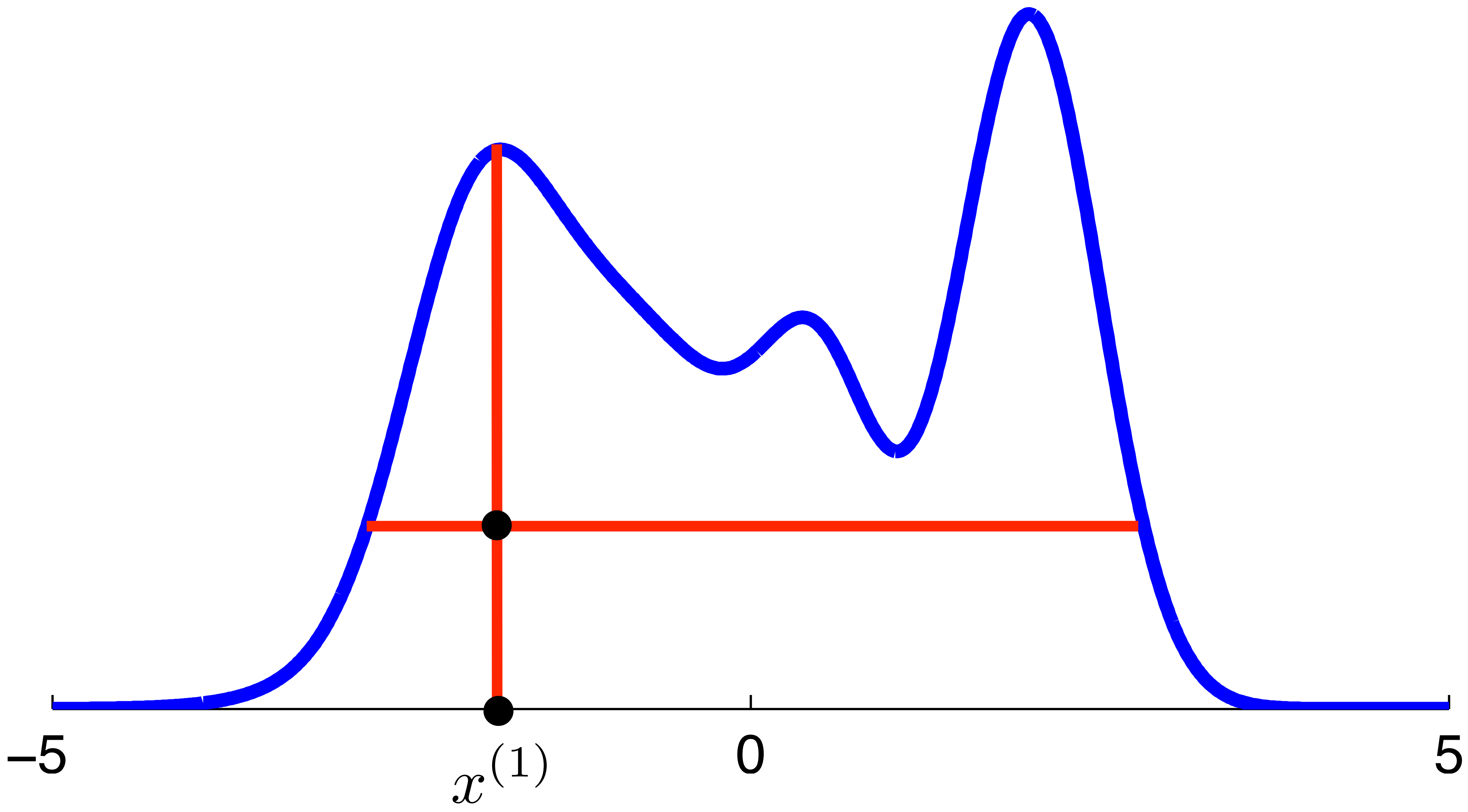
Slice Sampling

Define a Markov chain that samples uniformly from the area beneath the curve. This means that we need to introduce a “height” into the MCMC sampler.



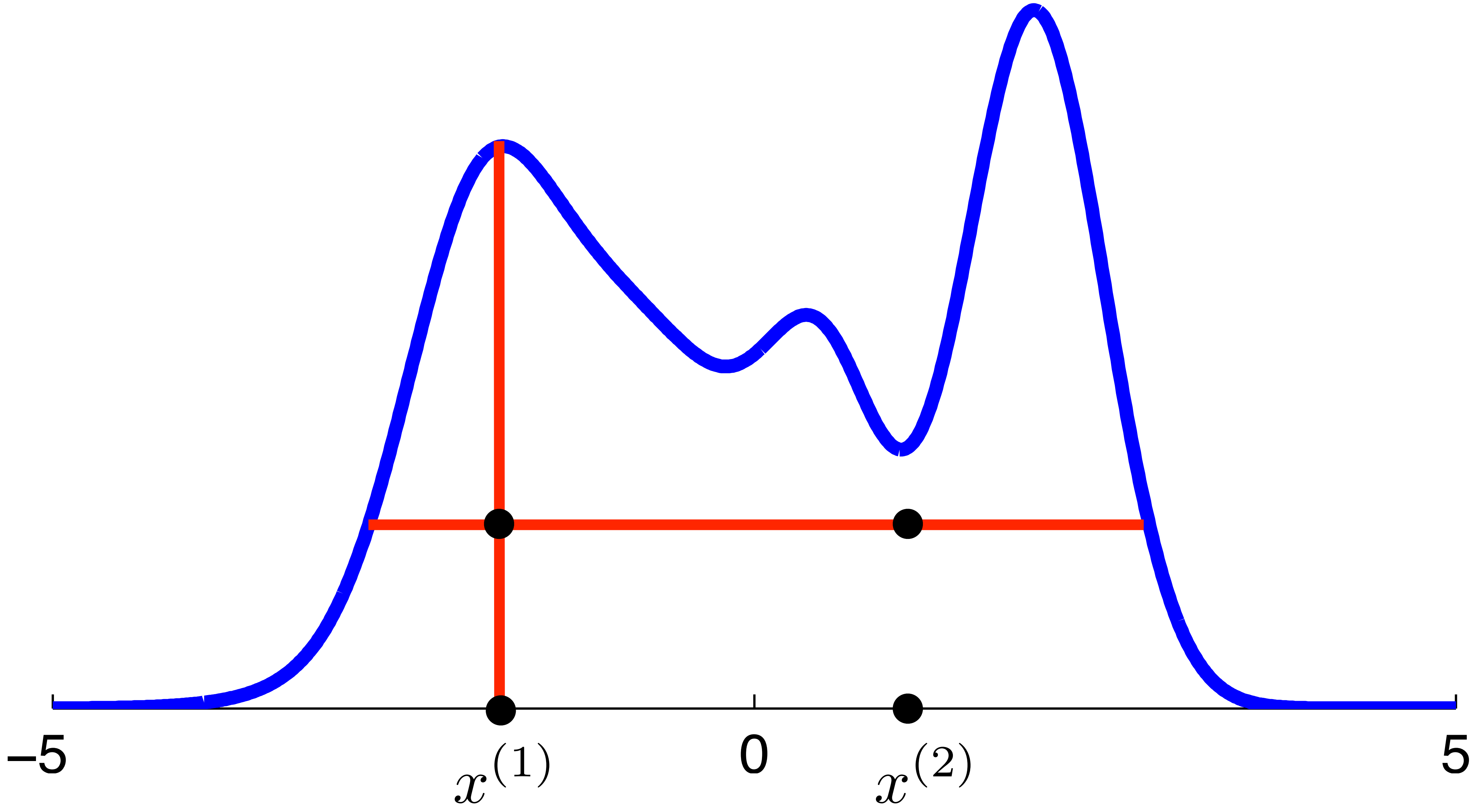
Slice Sampling

Define a Markov chain that samples uniformly from the area beneath the curve. This means that we need to introduce a “height” into the MCMC sampler.



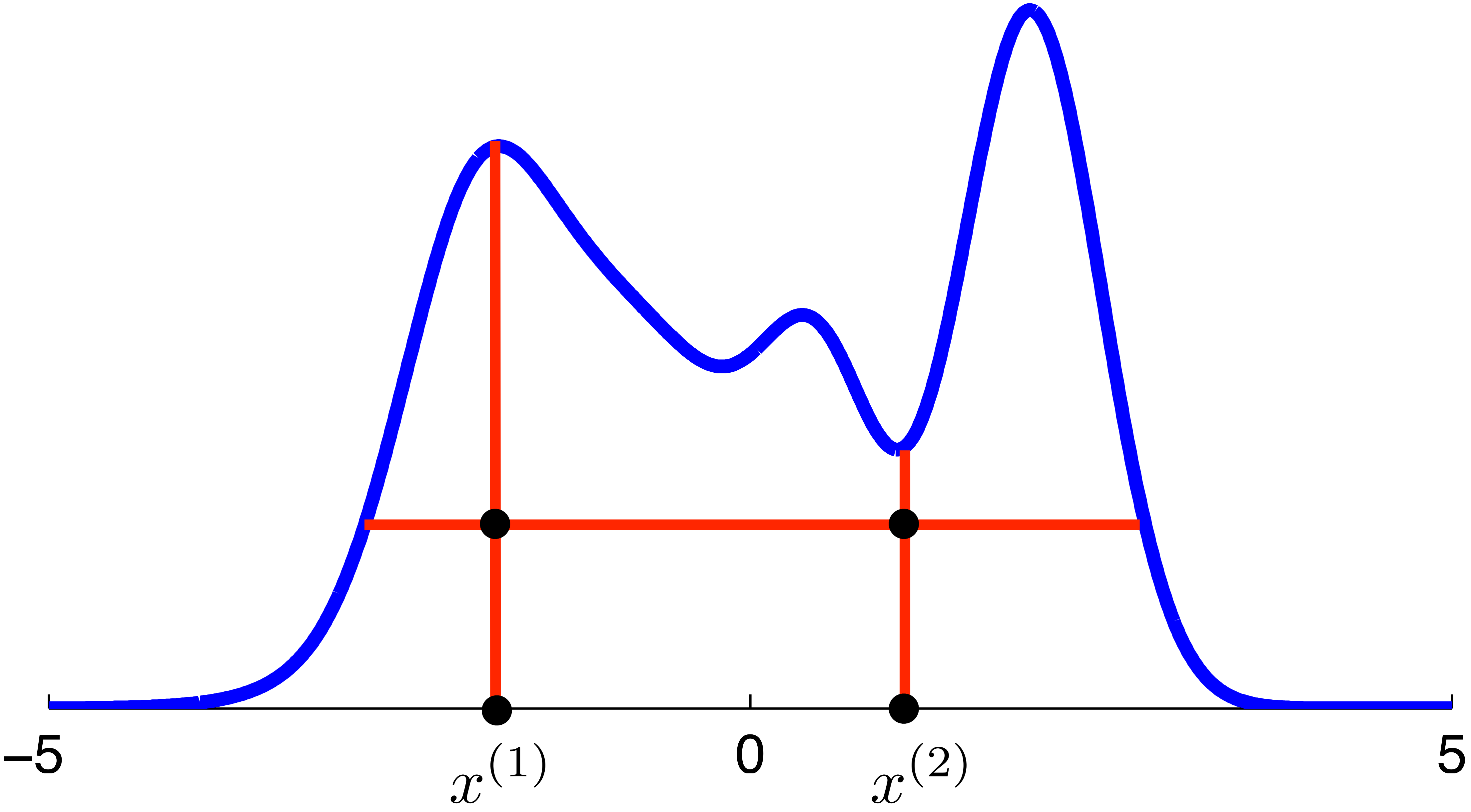
Slice Sampling

Define a Markov chain that samples uniformly from the area beneath the curve. This means that we need to introduce a "height" into the MCMC sampler.



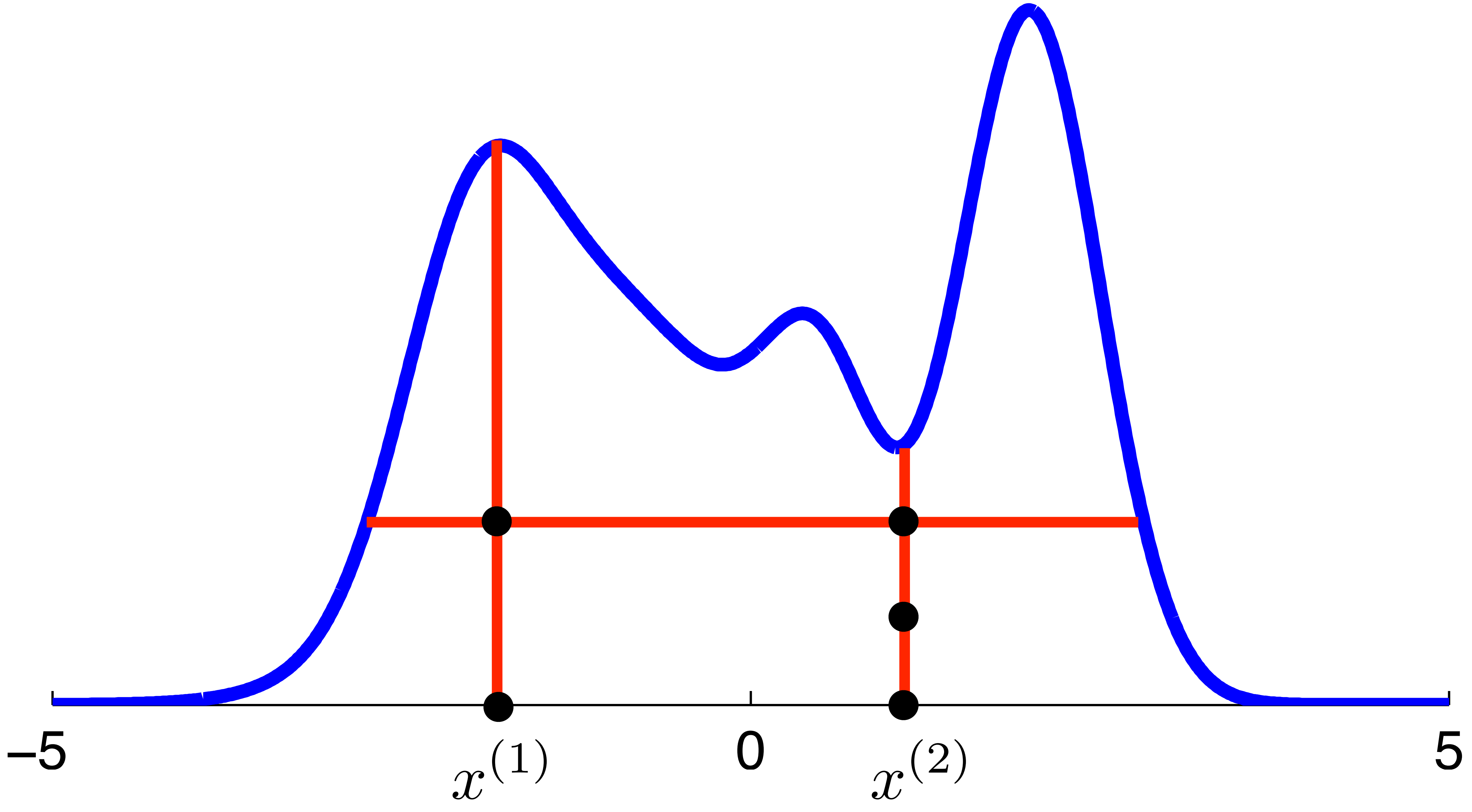
Slice Sampling

Define a Markov chain that samples uniformly from the area beneath the curve. This means that we need to introduce a "height" into the MCMC sampler.



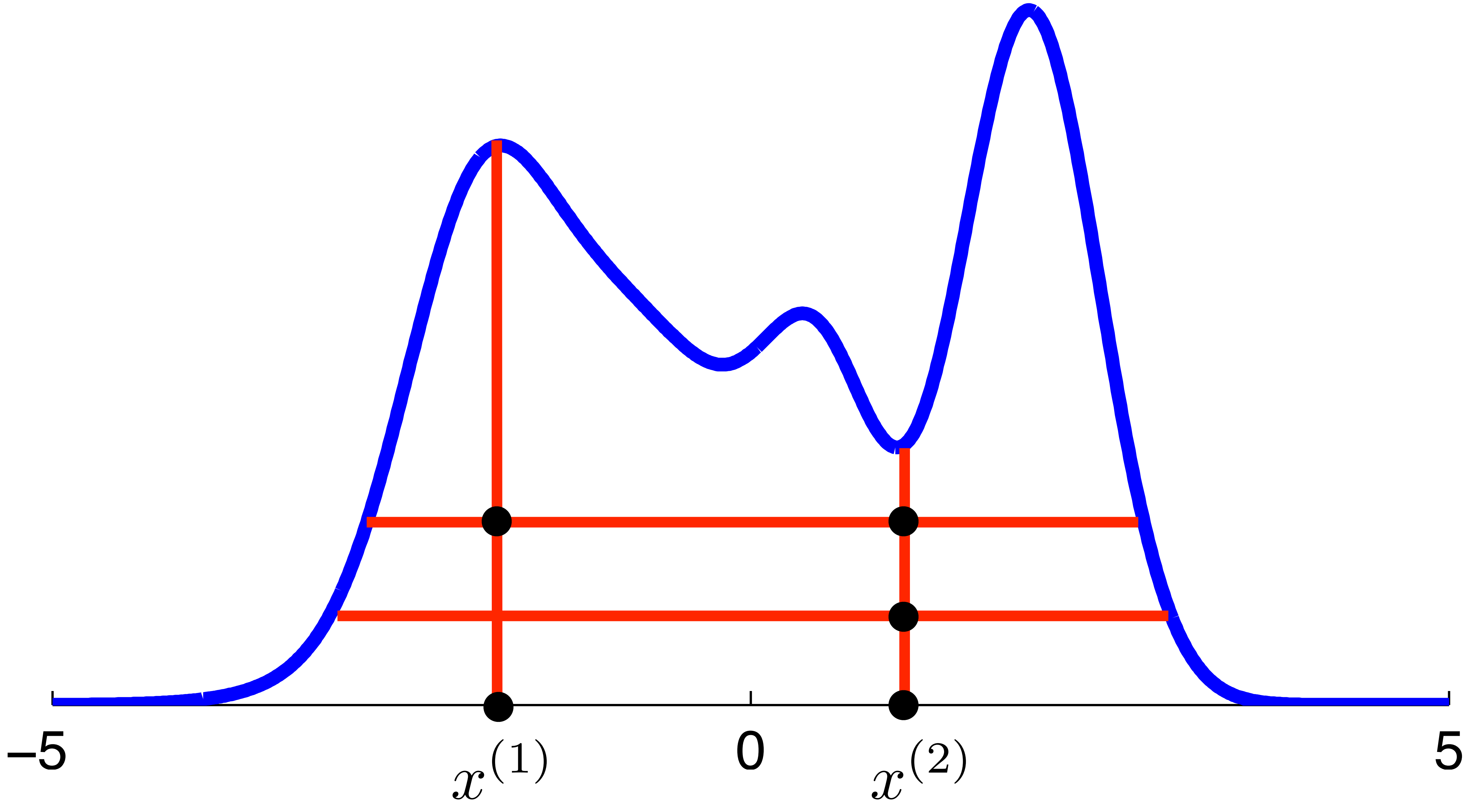
Slice Sampling

Define a Markov chain that samples uniformly from the area beneath the curve. This means that we need to introduce a "height" into the MCMC sampler.



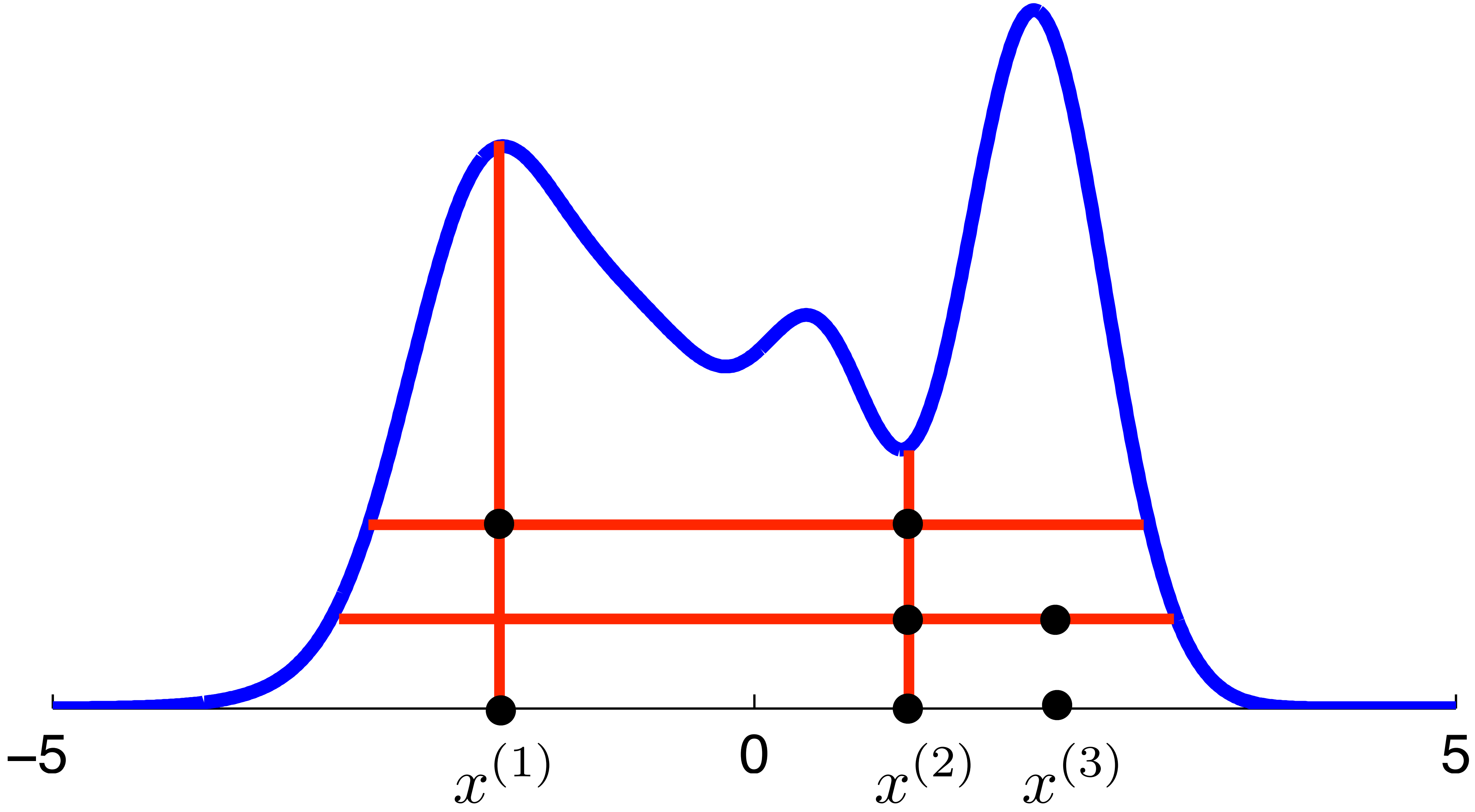
Slice Sampling

Define a Markov chain that samples uniformly from the area beneath the curve. This means that we need to introduce a “height” into the MCMC sampler.



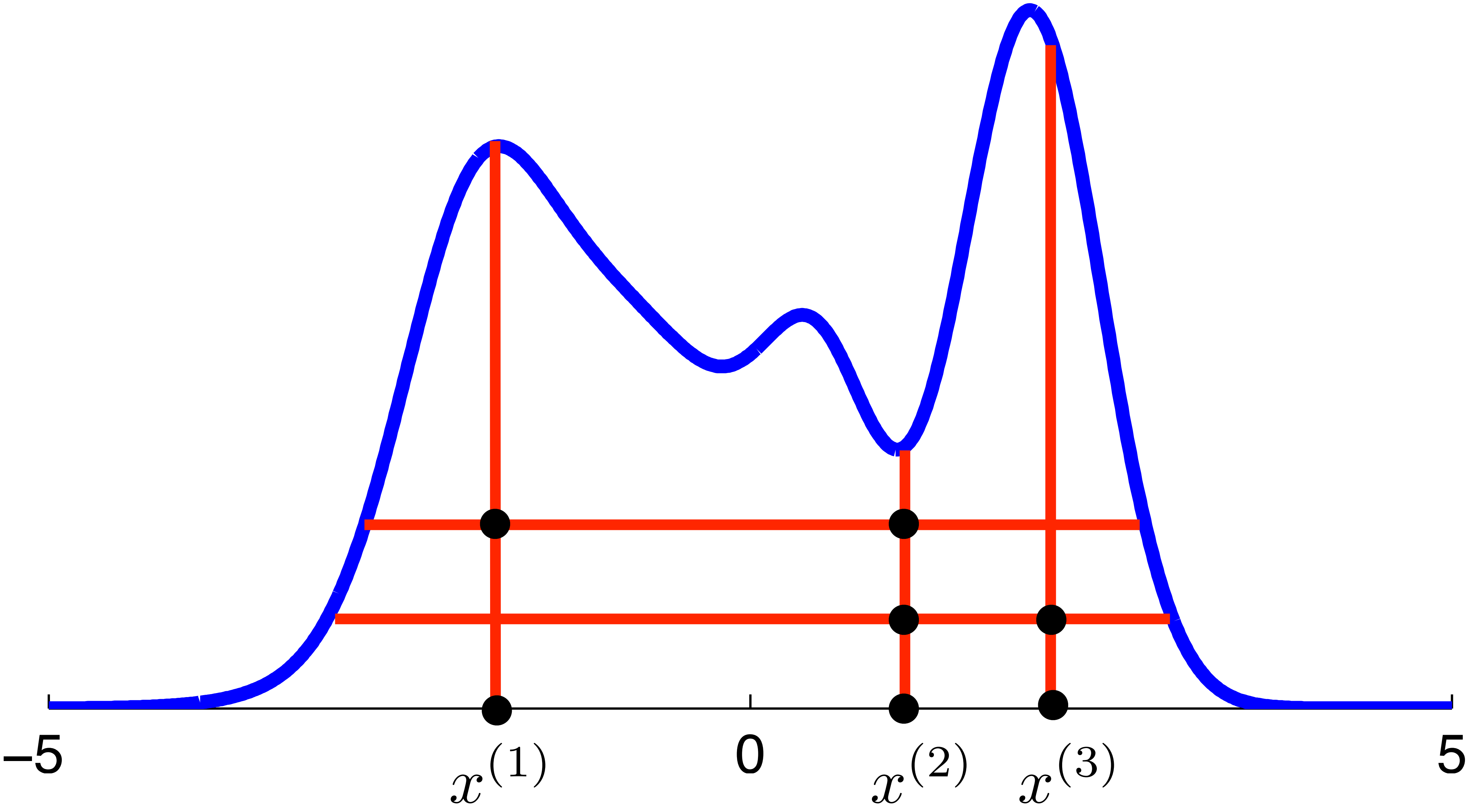
Slice Sampling

Define a Markov chain that samples uniformly from the area beneath the curve. This means that we need to introduce a "height" into the MCMC sampler.



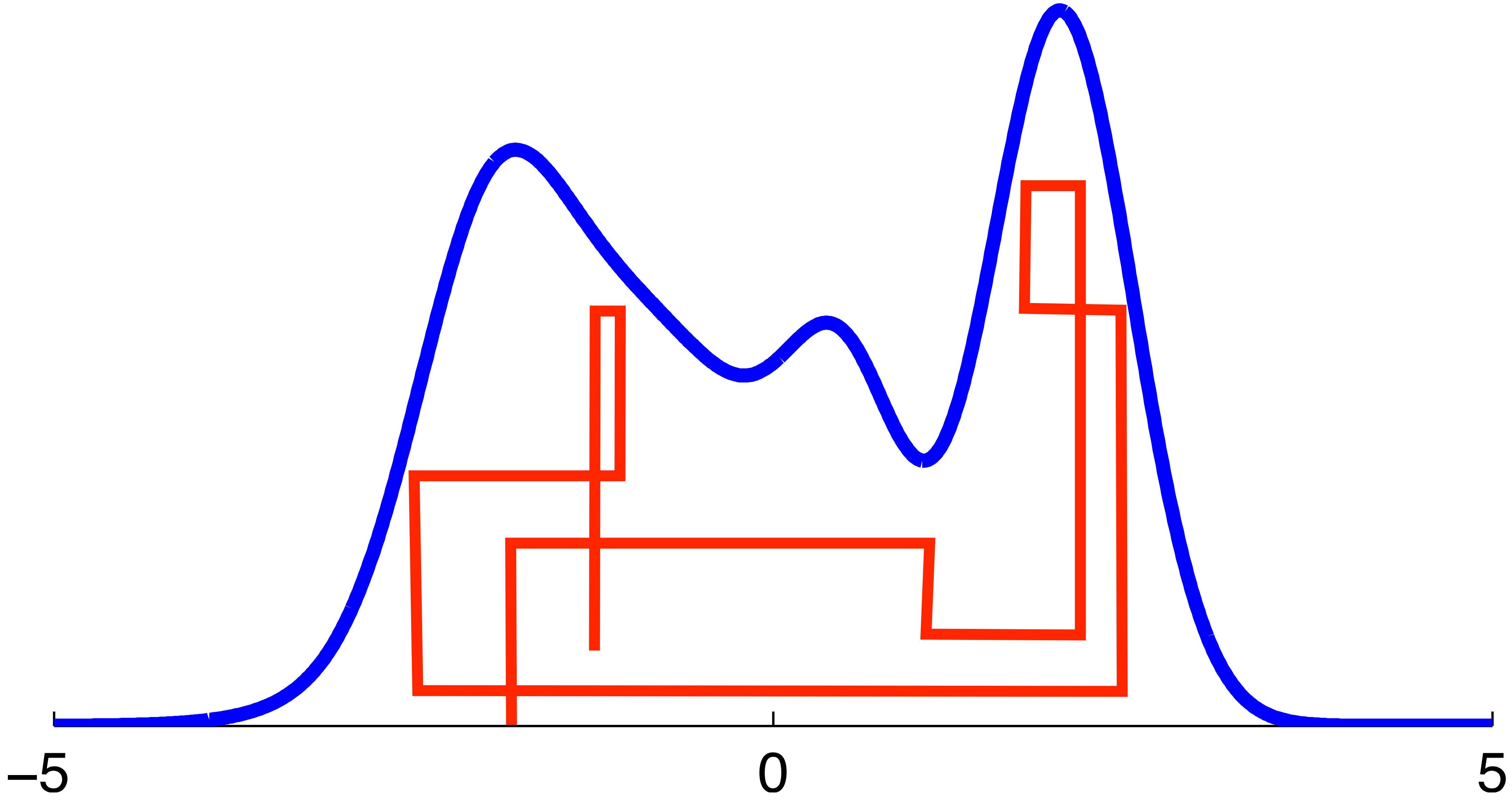
Slice Sampling

Define a Markov chain that samples uniformly from the area beneath the curve. This means that we need to introduce a "height" into the MCMC sampler.



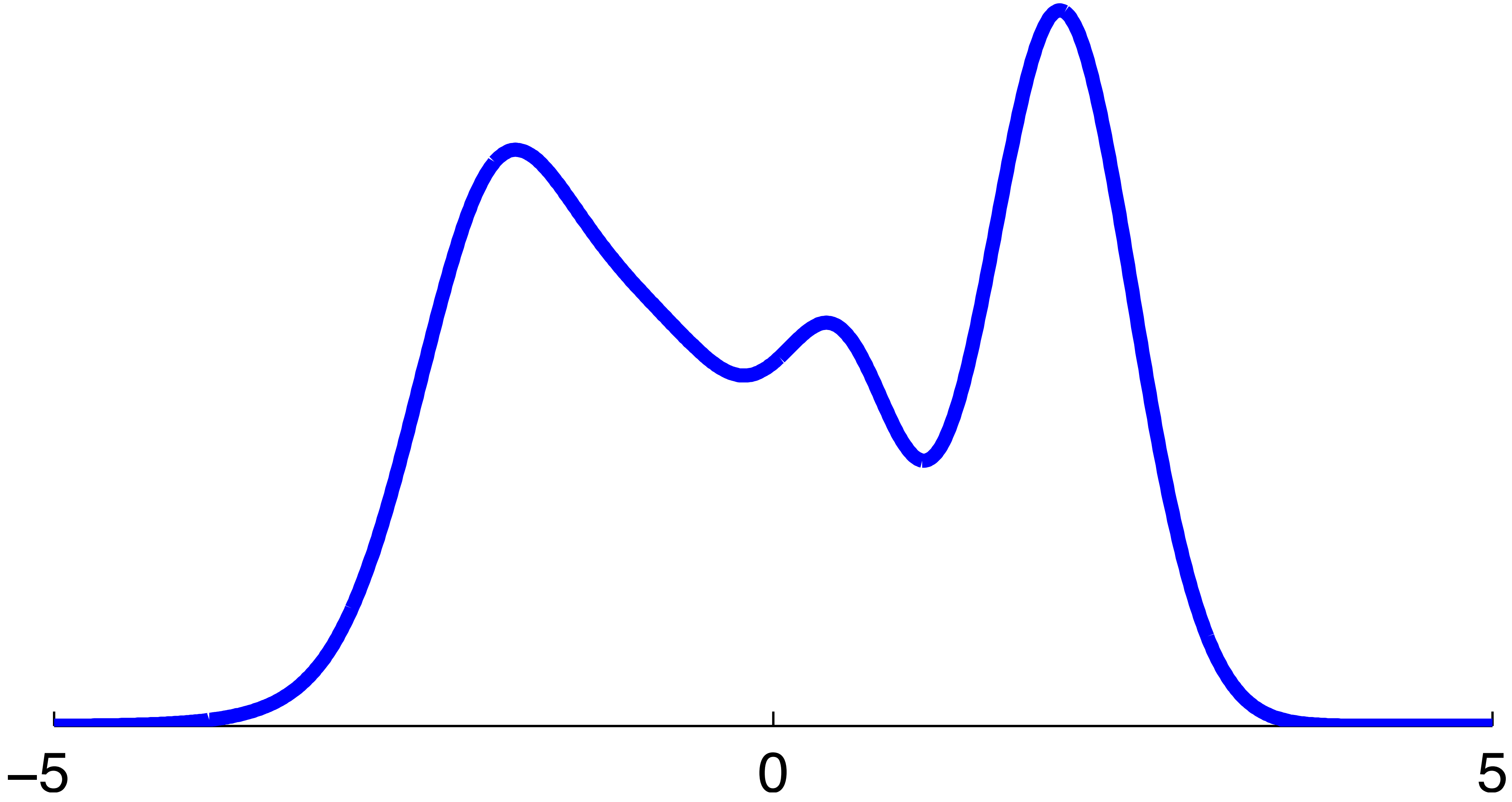
Slice Sampling

Define a Markov chain that samples uniformly from the area beneath the curve. This means that we need to introduce a "height" into the MCMC sampler.



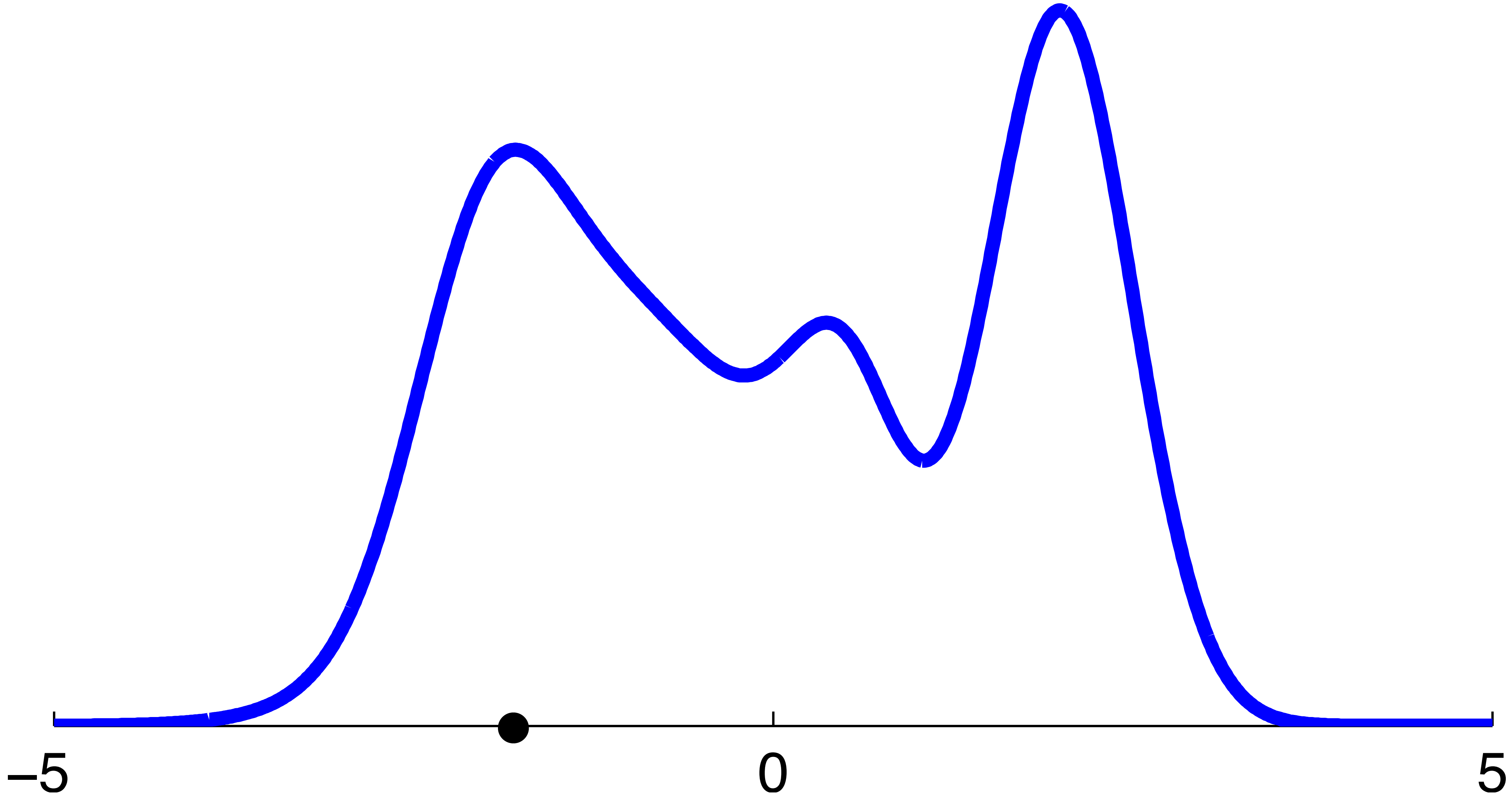
Slice Sampling

Sampling the height is easy: simulate a random variate uniformly between 0 and the height of your (perhaps unnormalized) density function.



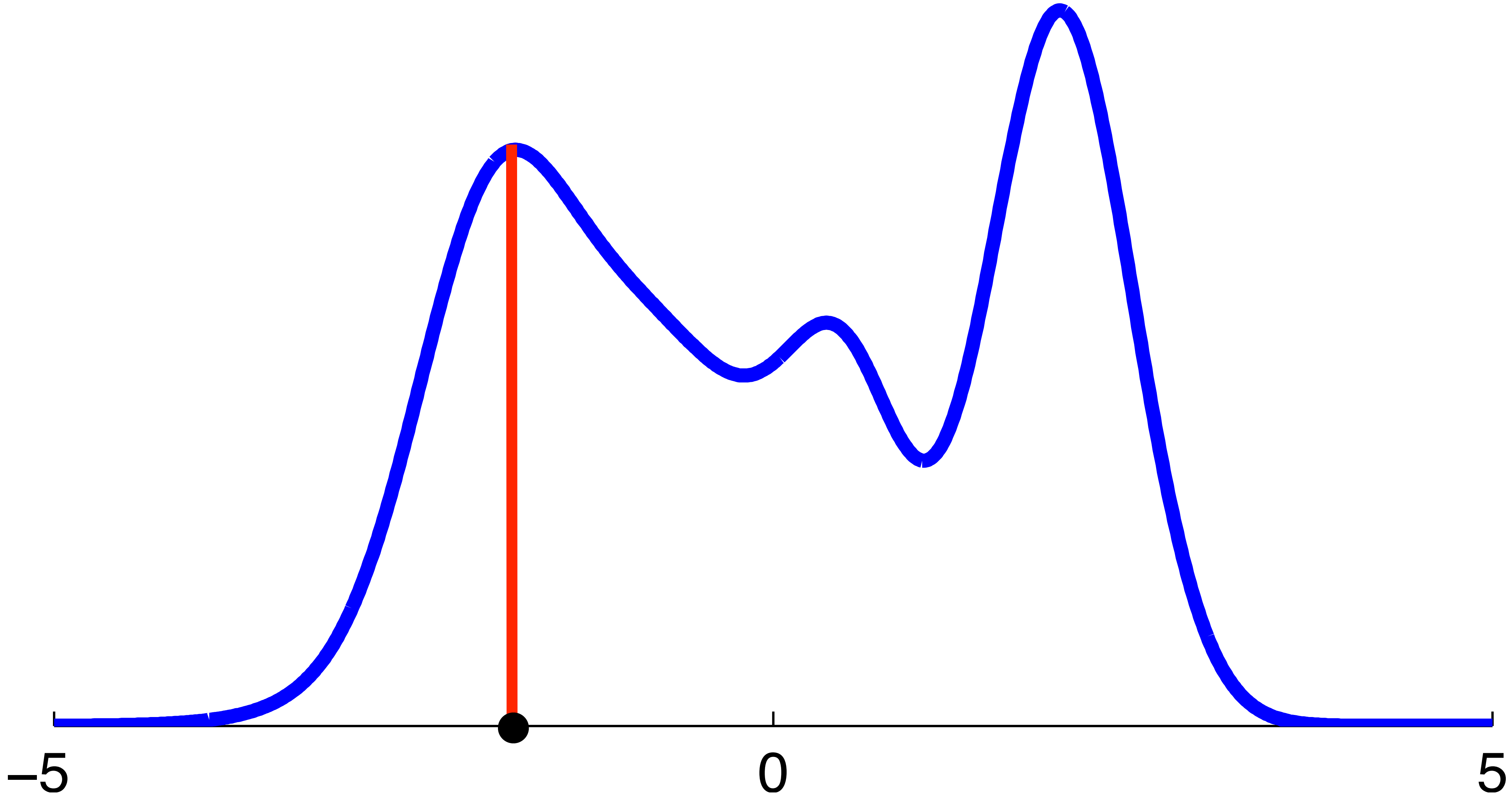
Slice Sampling

Sampling the height is easy: simulate a random variate uniformly between 0 and the height of your (perhaps unnormalized) density function.



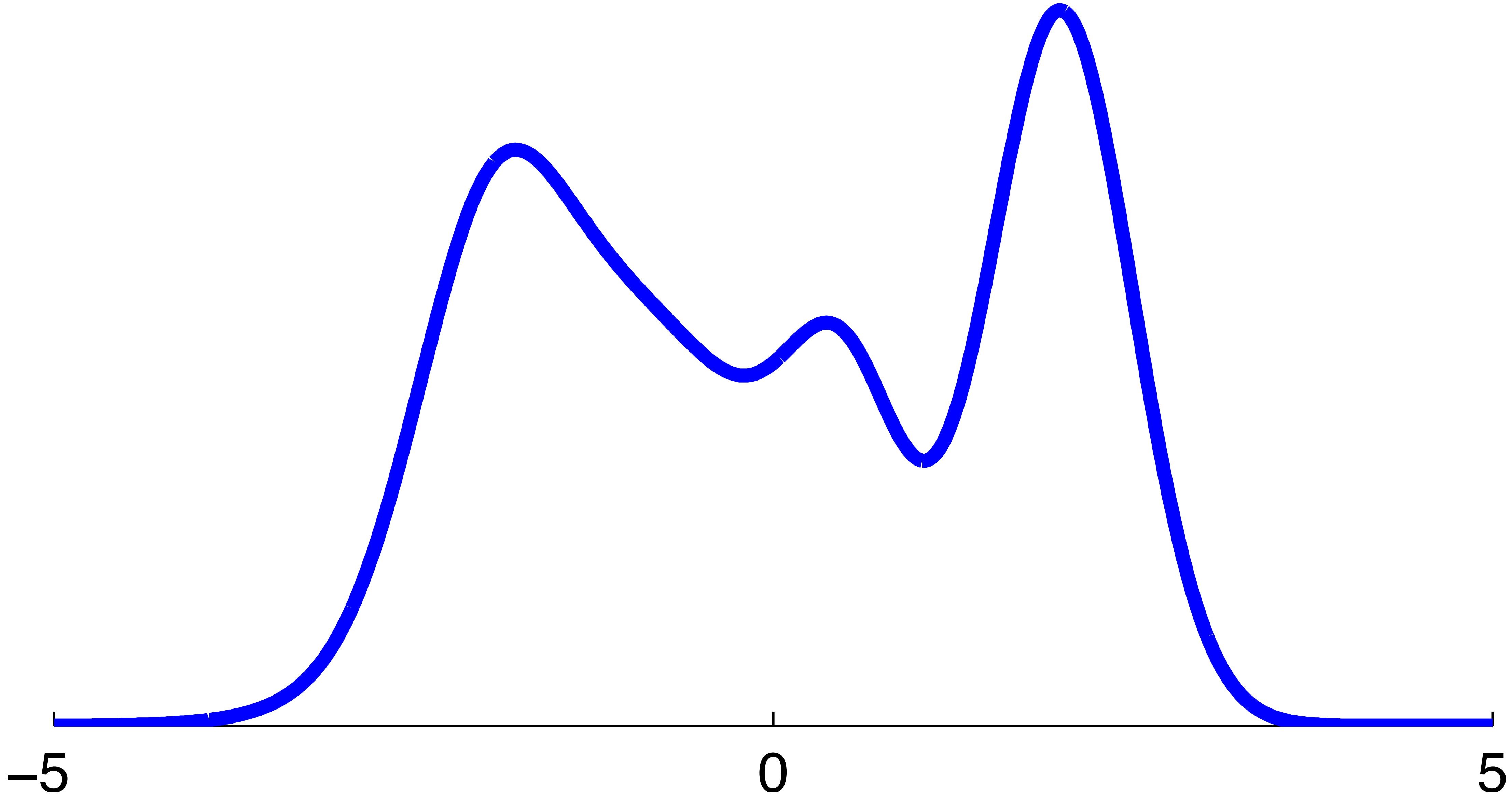
Slice Sampling

Sampling the height is easy: simulate a random variate uniformly between 0 and the height of your (perhaps unnormalized) density function.



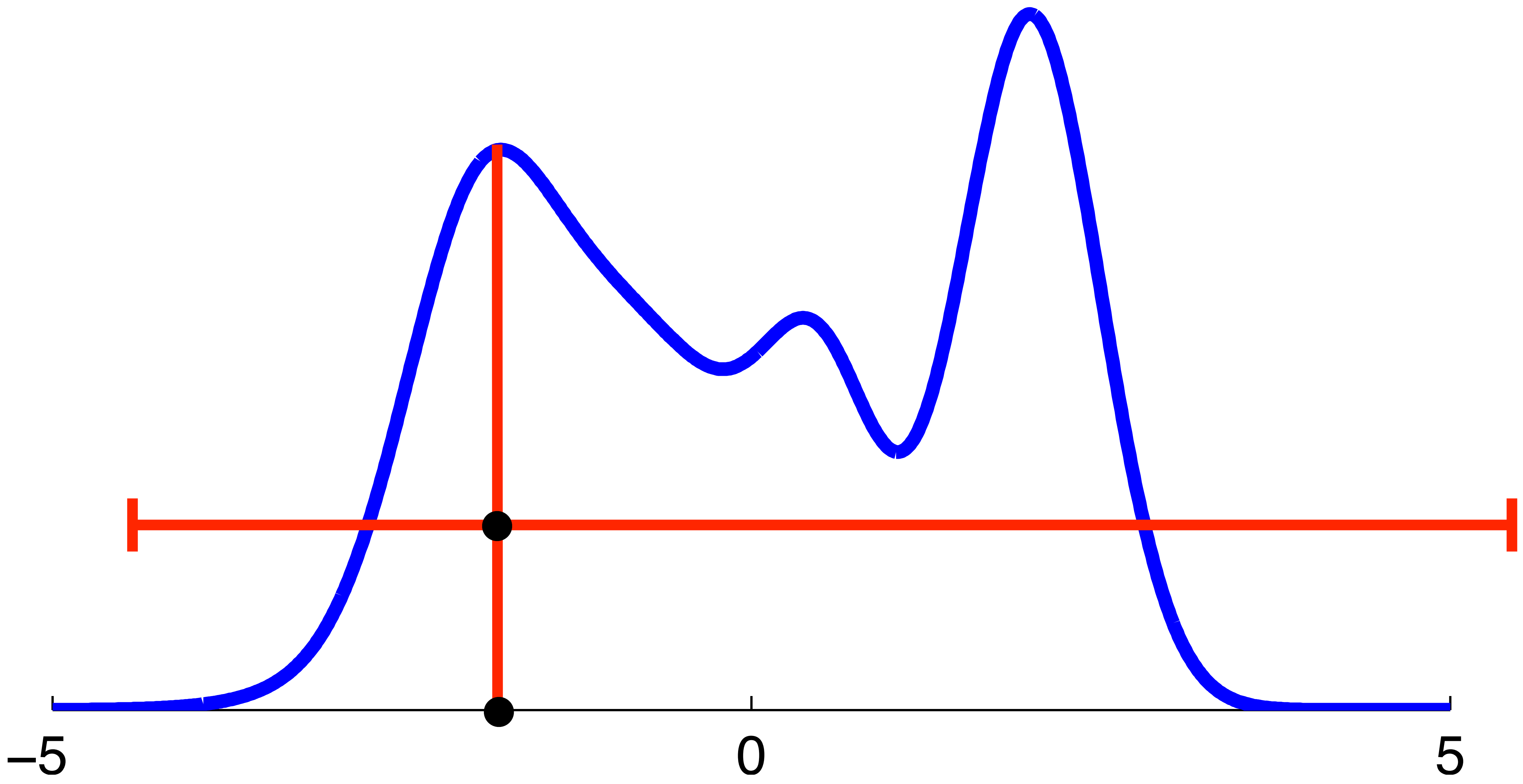
Slice Sampling

Sampling the height is easy: simulate a random variate uniformly between 0 and the height of your (perhaps unnormalized) density function.



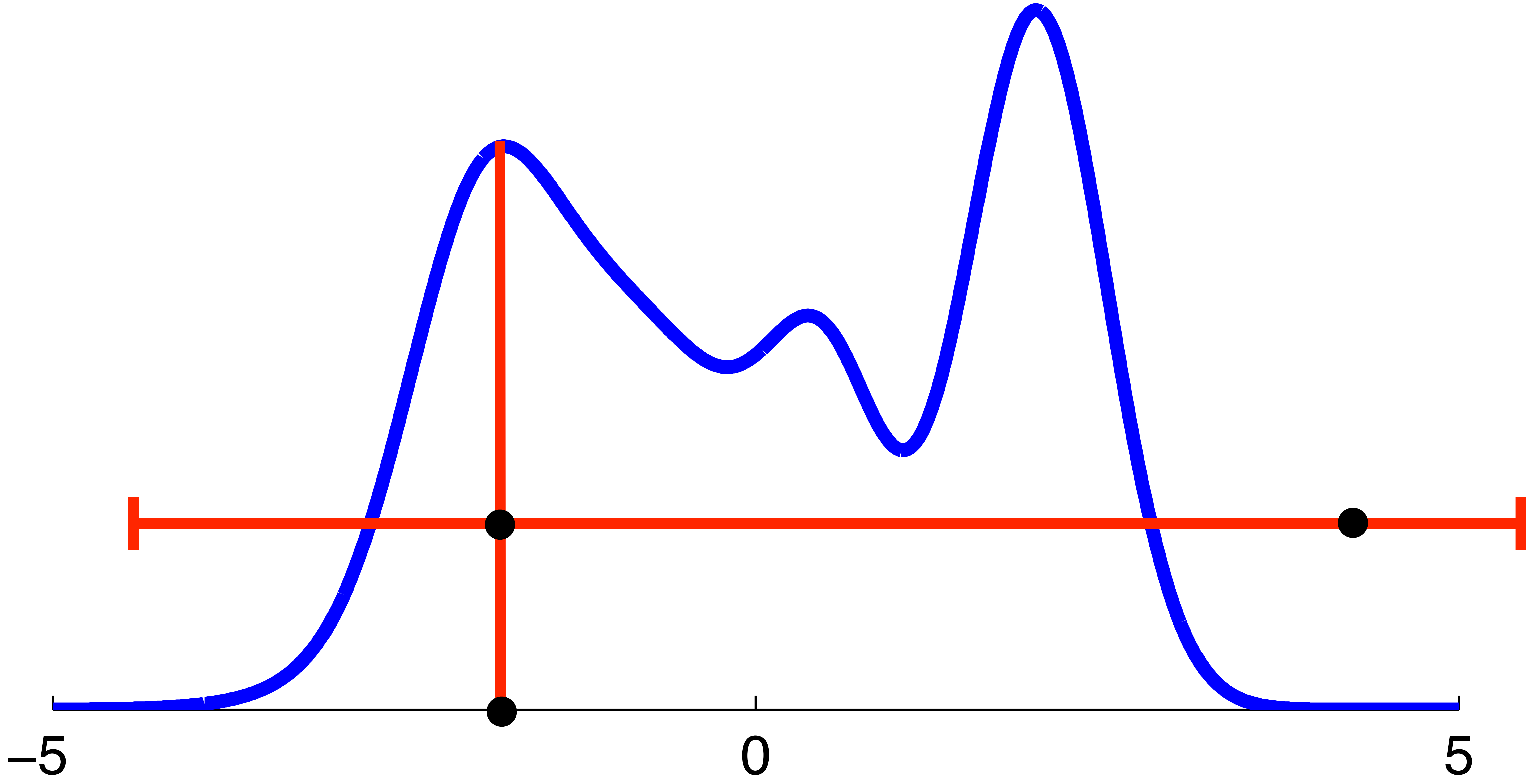
Slice Sampling

Sampling the horizontal slice is more complicated. Start with a big "bracket" and rejection sample, shrinking the bracket with rejections. Shrinks exponentially fast!



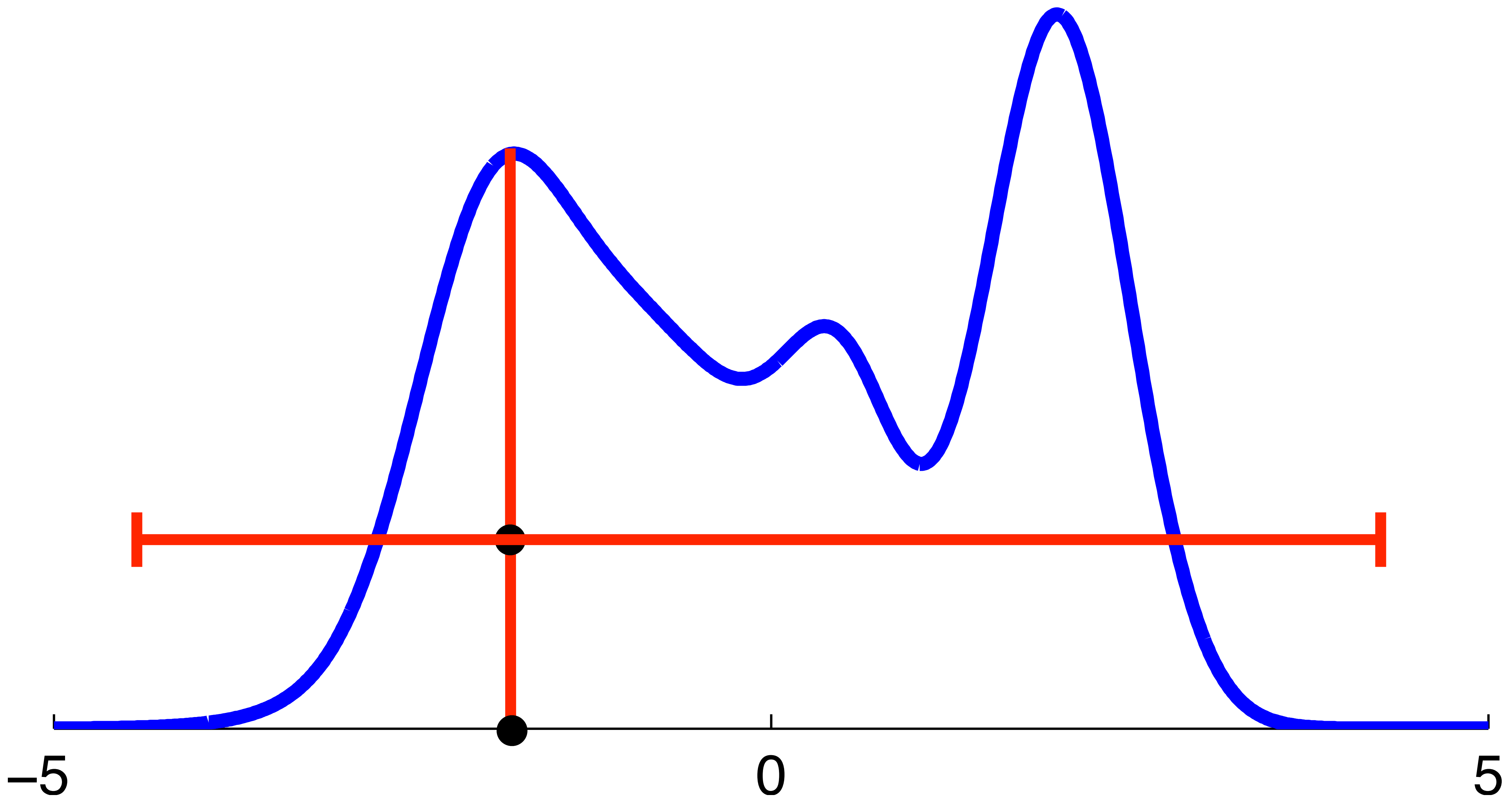
Slice Sampling

Sampling the horizontal slice is more complicated. Start with a big "bracket" and rejection sample, shrinking the bracket with rejections. Shrinks exponentially fast!



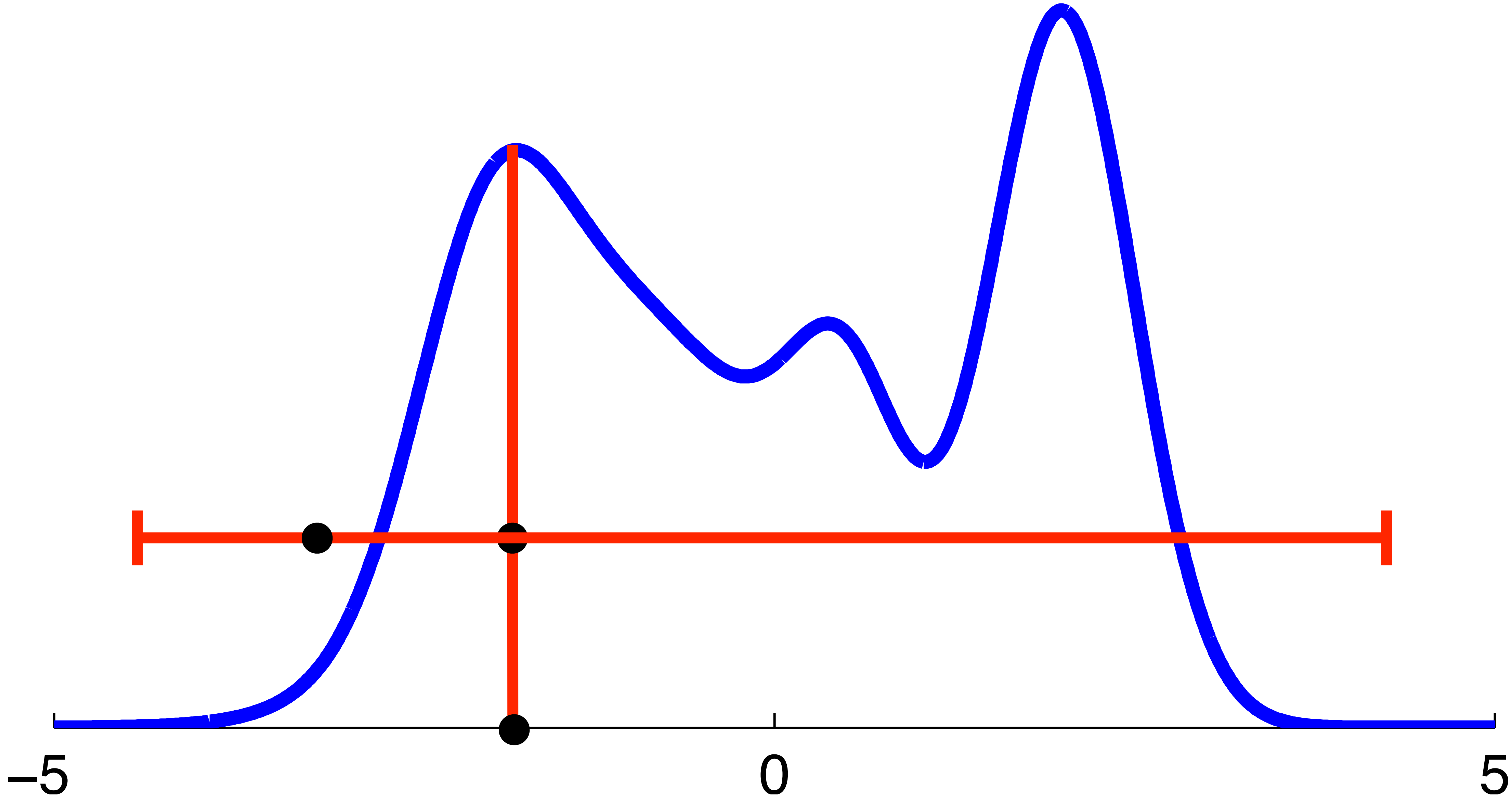
Slice Sampling

Sampling the horizontal slice is more complicated. Start with a big "bracket" and rejection sample, shrinking the bracket with rejections. Shrinks exponentially fast!



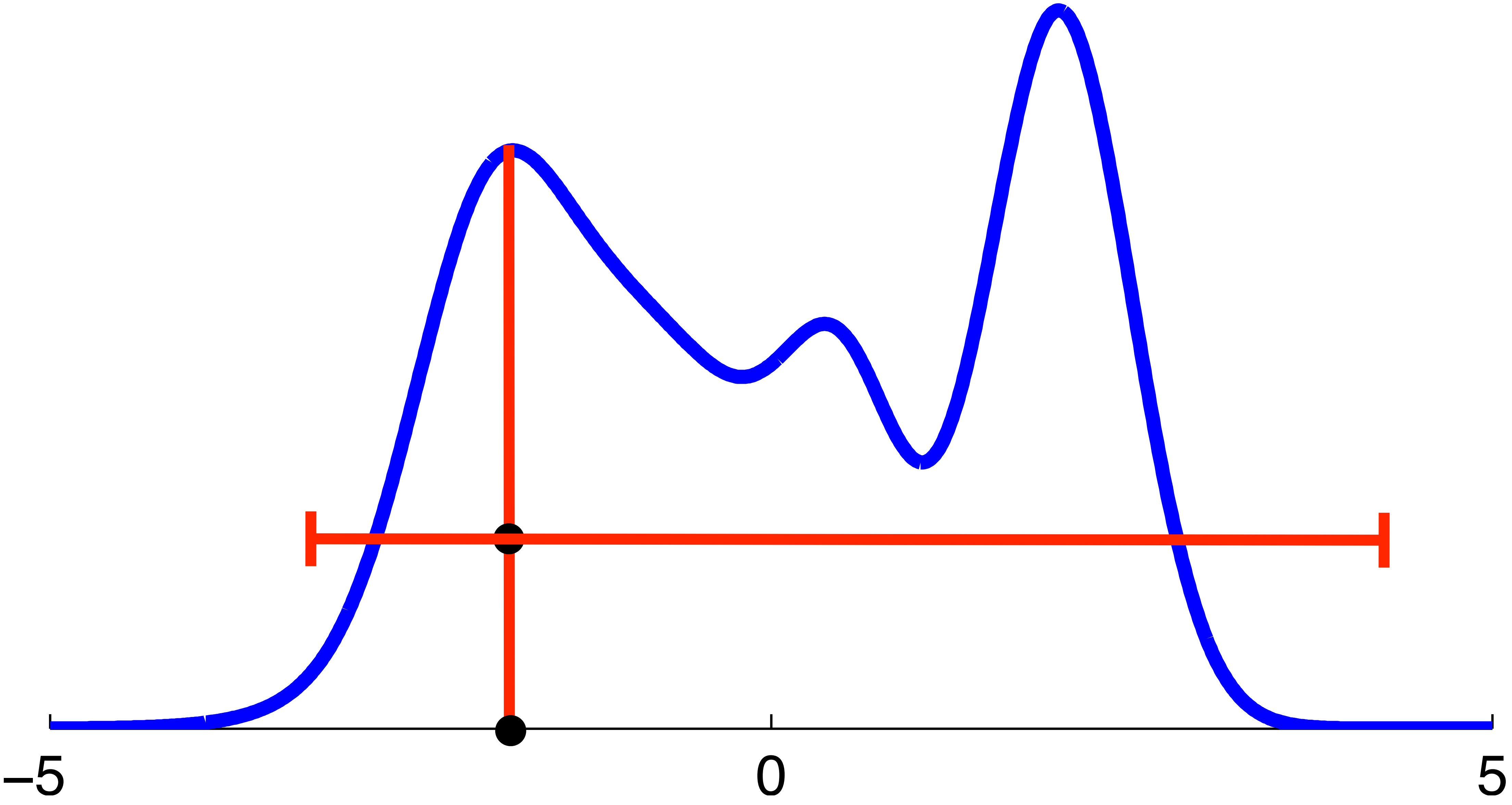
Slice Sampling

Sampling the horizontal slice is more complicated. Start with a big "bracket" and rejection sample, shrinking the bracket with rejections. Shrinks exponentially fast!



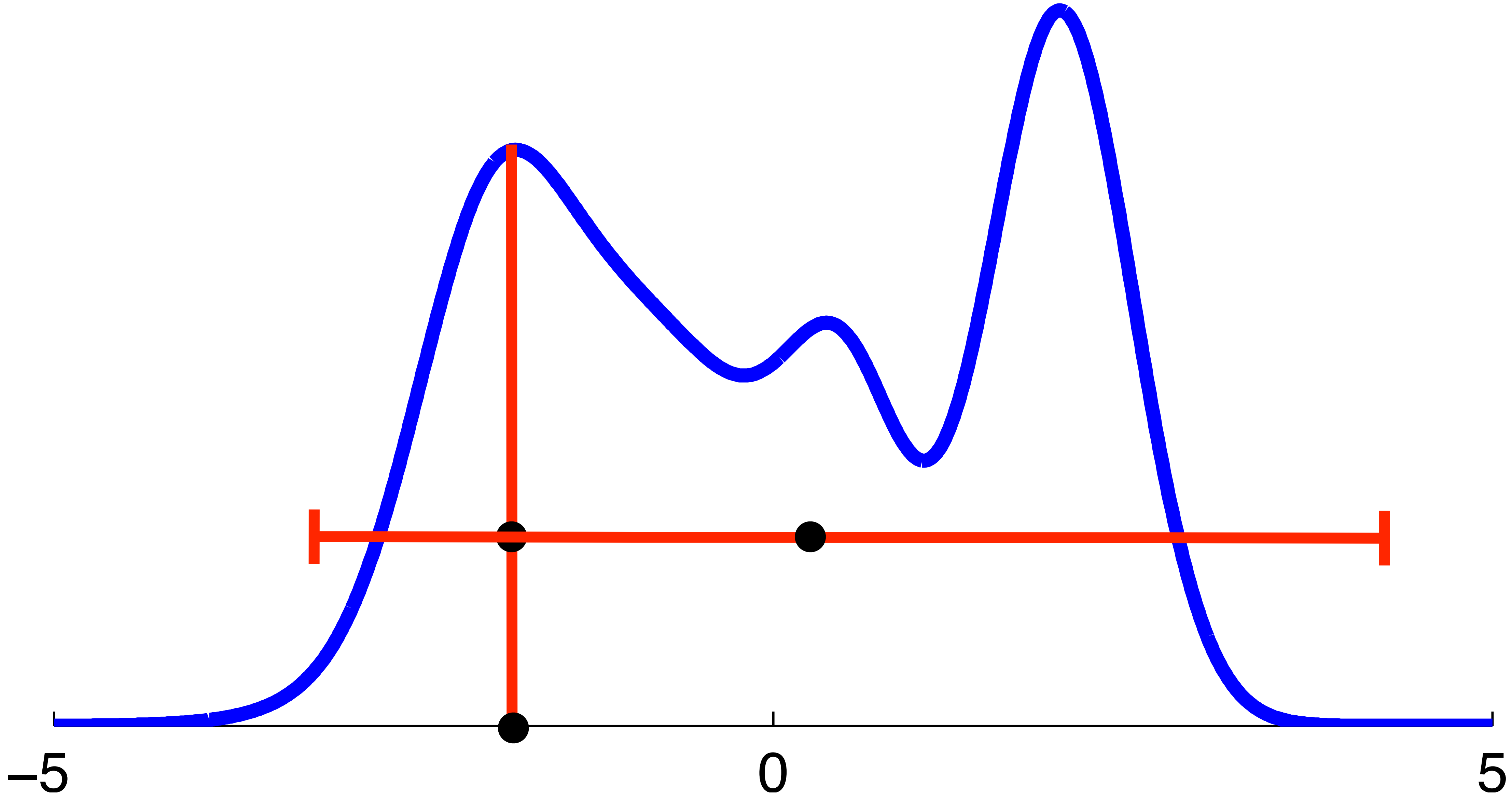
Slice Sampling

Sampling the horizontal slice is more complicated. Start with a big "bracket" and rejection sample, shrinking the bracket with rejections. Shrinks exponentially fast!



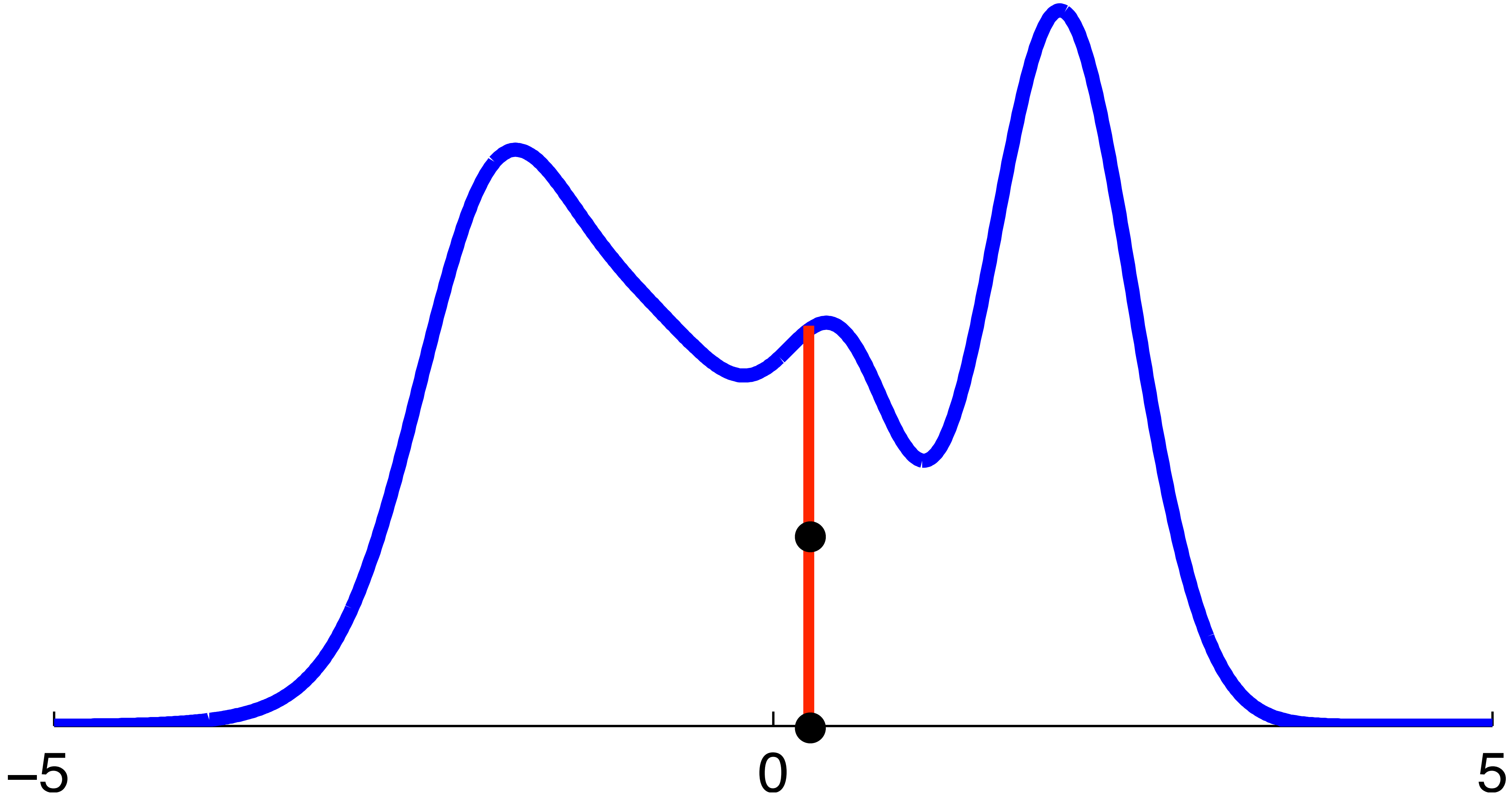
Slice Sampling

Sampling the horizontal slice is more complicated. Start with a big "bracket" and rejection sample, shrinking the bracket with rejections. Shrinks exponentially fast!



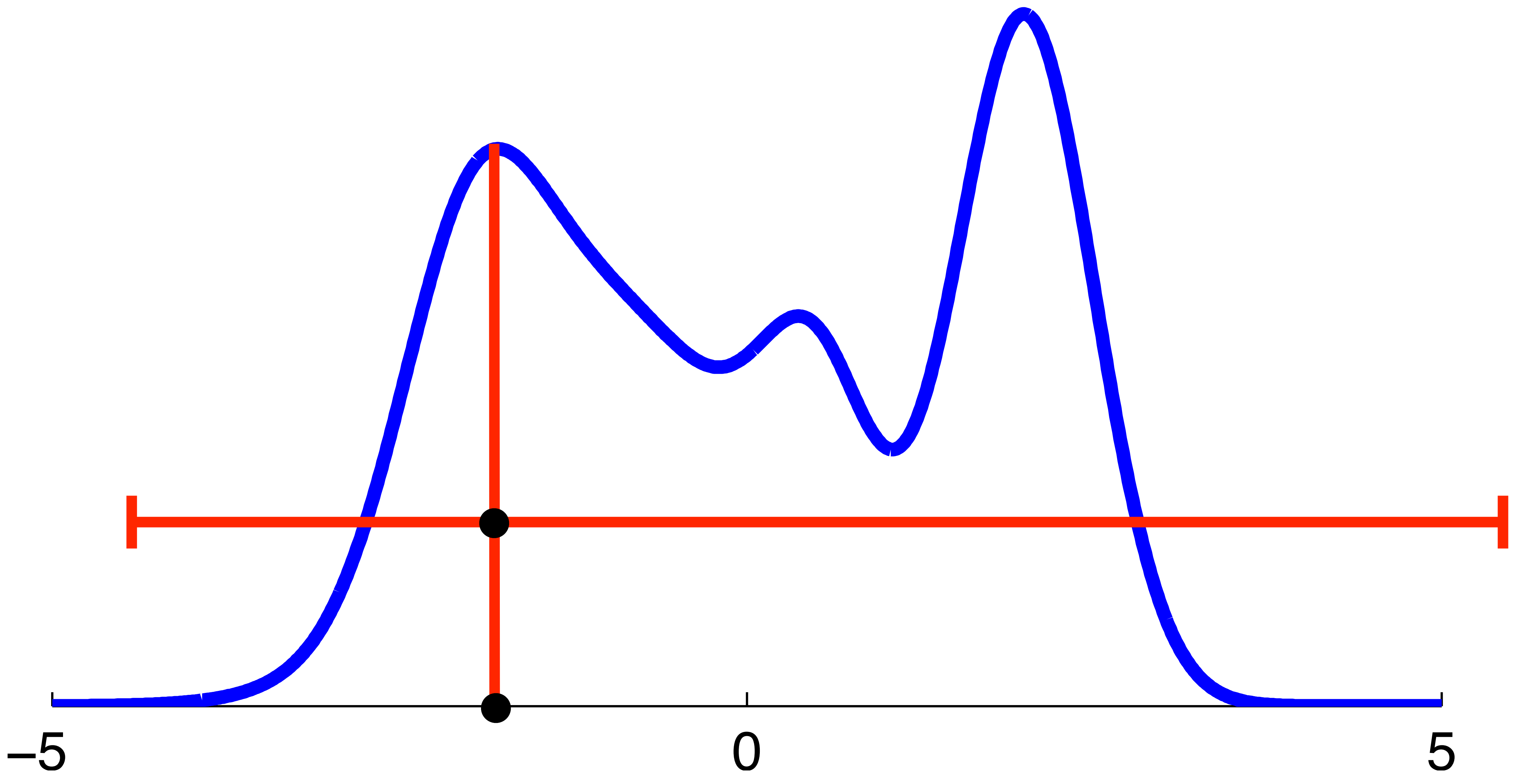
Slice Sampling

Sampling the horizontal slice is more complicated. Start with a big "bracket" and rejection sample, shrinking the bracket with rejections. Shrinks exponentially fast!



Slice Sampling

Unfortunately, you have to pick an initial bracket size. Exponential shrinkage means you can err on the side of being too large without too much additional cost.



Slice Sampling

There are also fancier versions that will automatically grow the bracket if it is too small. Radford Neal's paper discusses this and many other ideas.

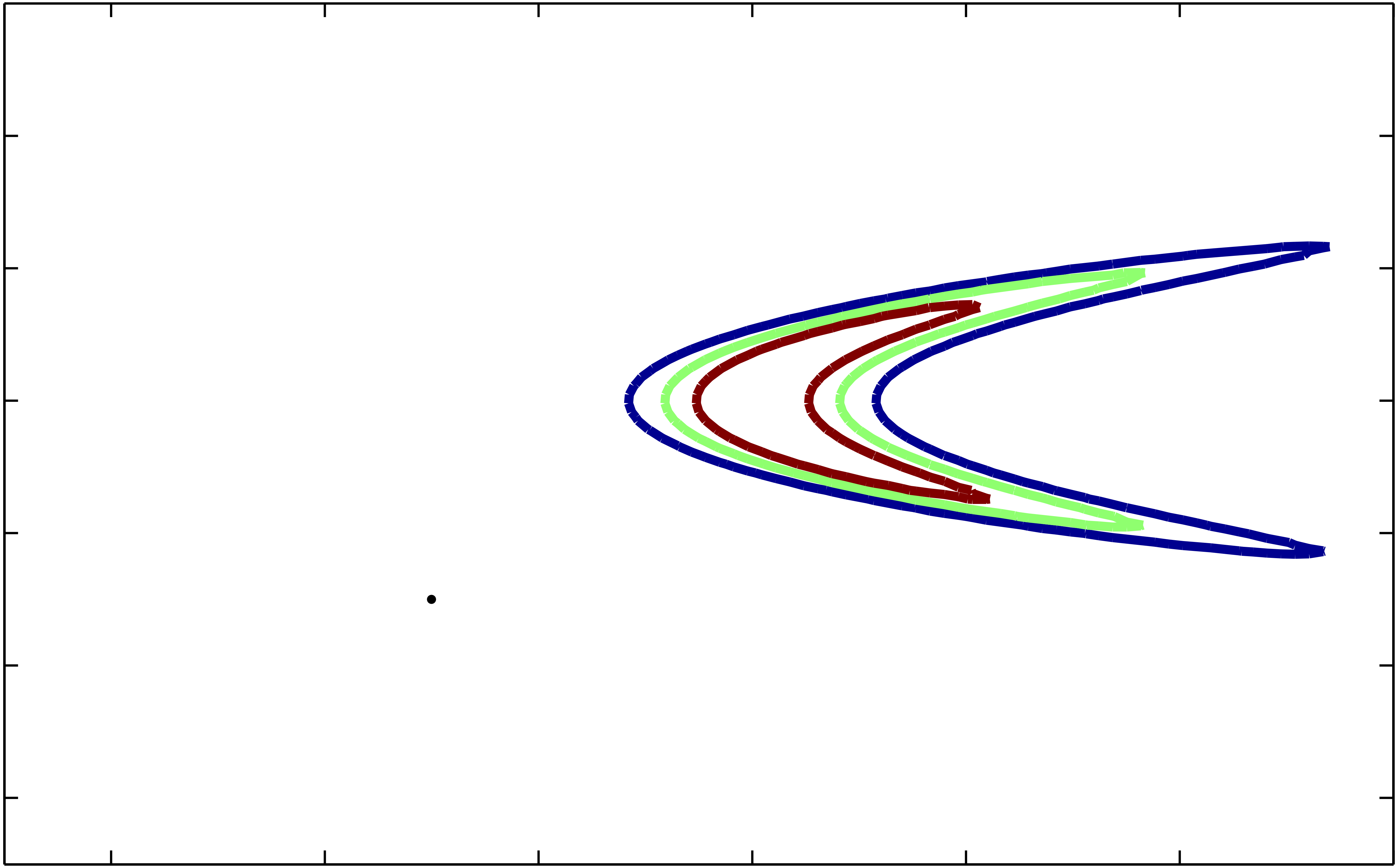
Radford M. Neal, "Slice Sampling", *Annals of Statistics* 31, 705-767, 2003.

Iain Murray has Matlab code on the web. I have Python code on the web also. The Matlab statistics toolbox includes a `slicesample()` function these days.

It is easy and requires almost no tuning. If you're currently solving a problem with Metropolis-Hastings, you should give this a try. Remember, the "best" M-H step size may vary, even with a single run!

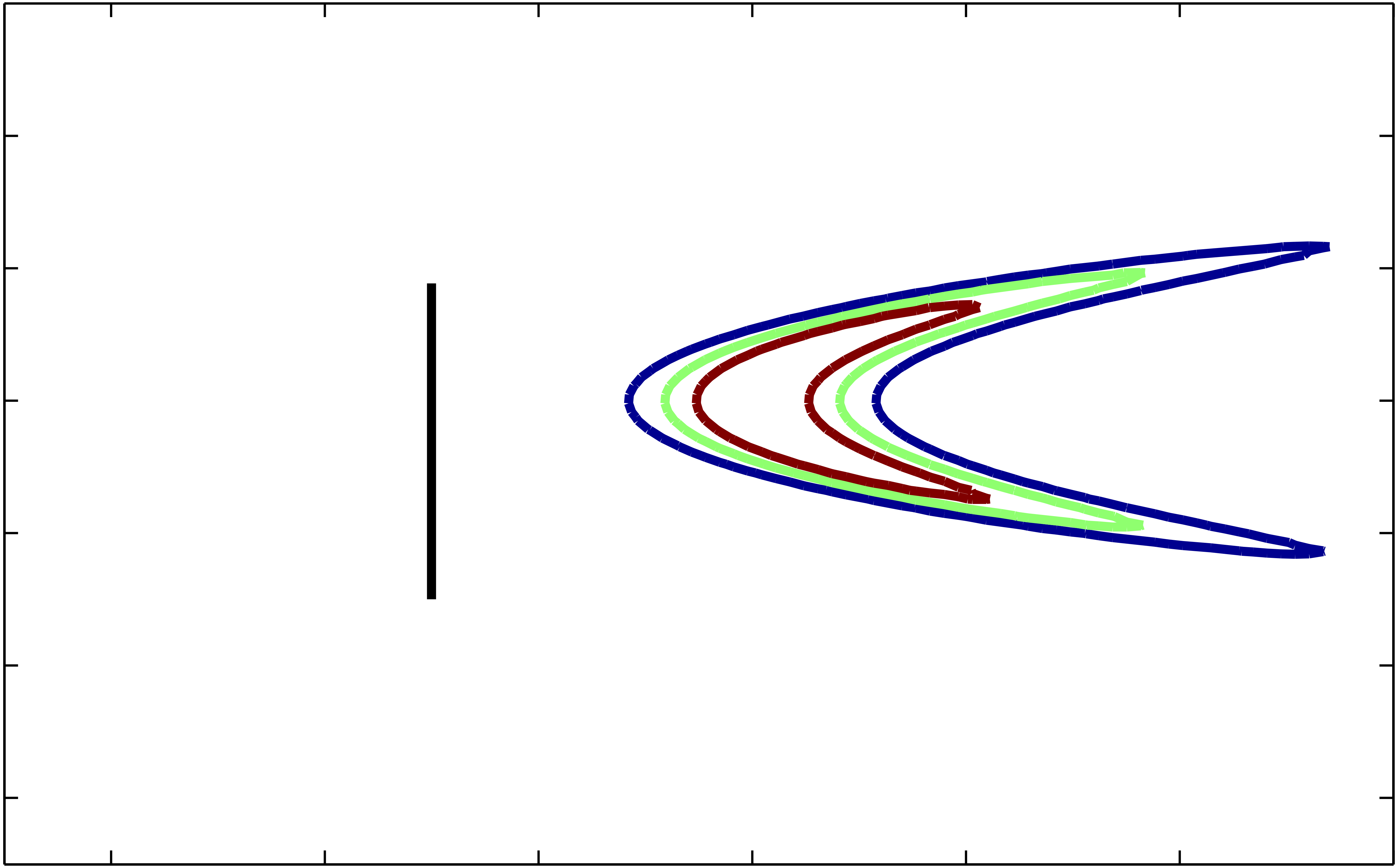
Multiple Dimensions

One Approach: Slice sample each dimension, as in Gibbs



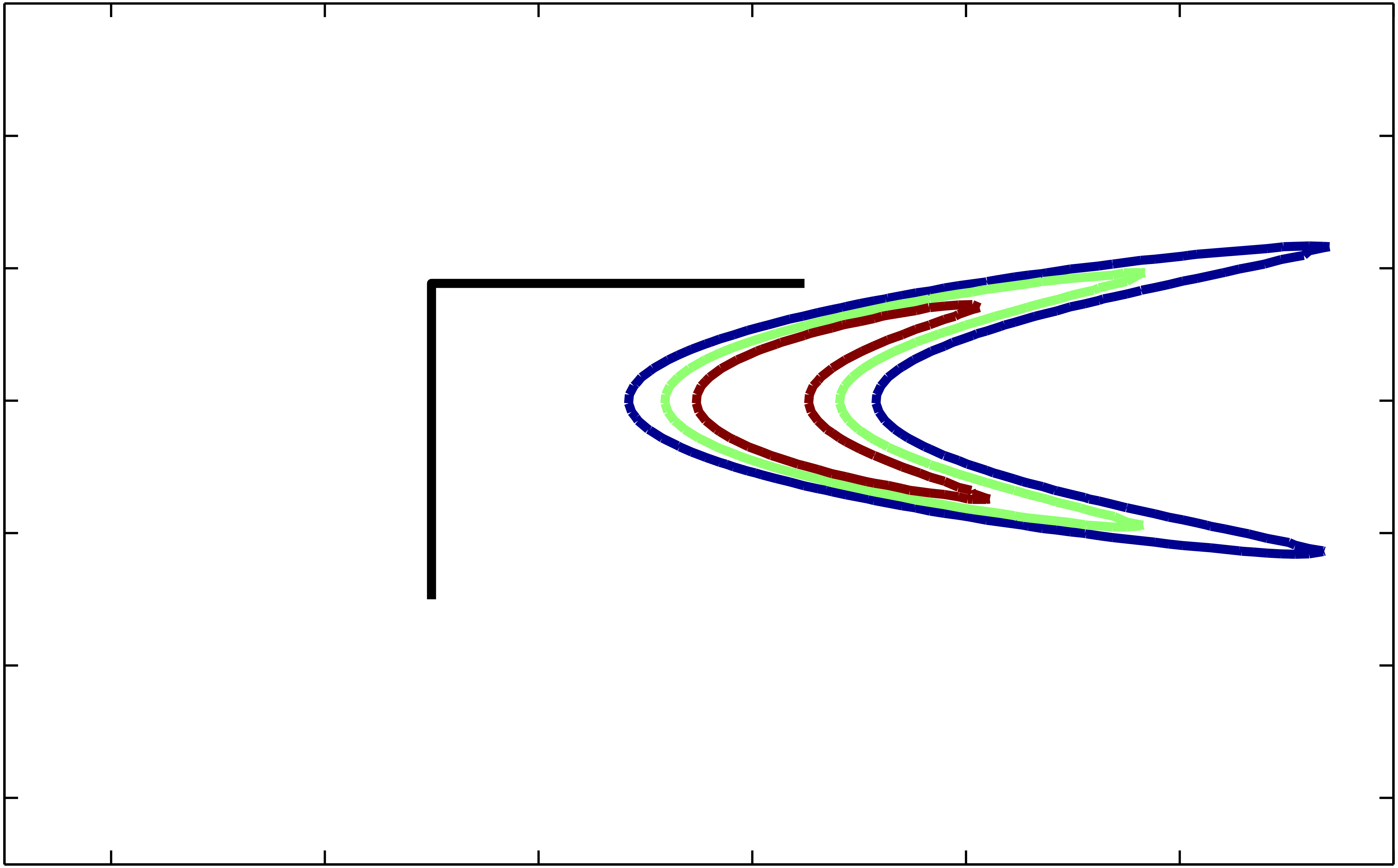
Multiple Dimensions

One Approach: Slice sample each dimension, as in Gibbs



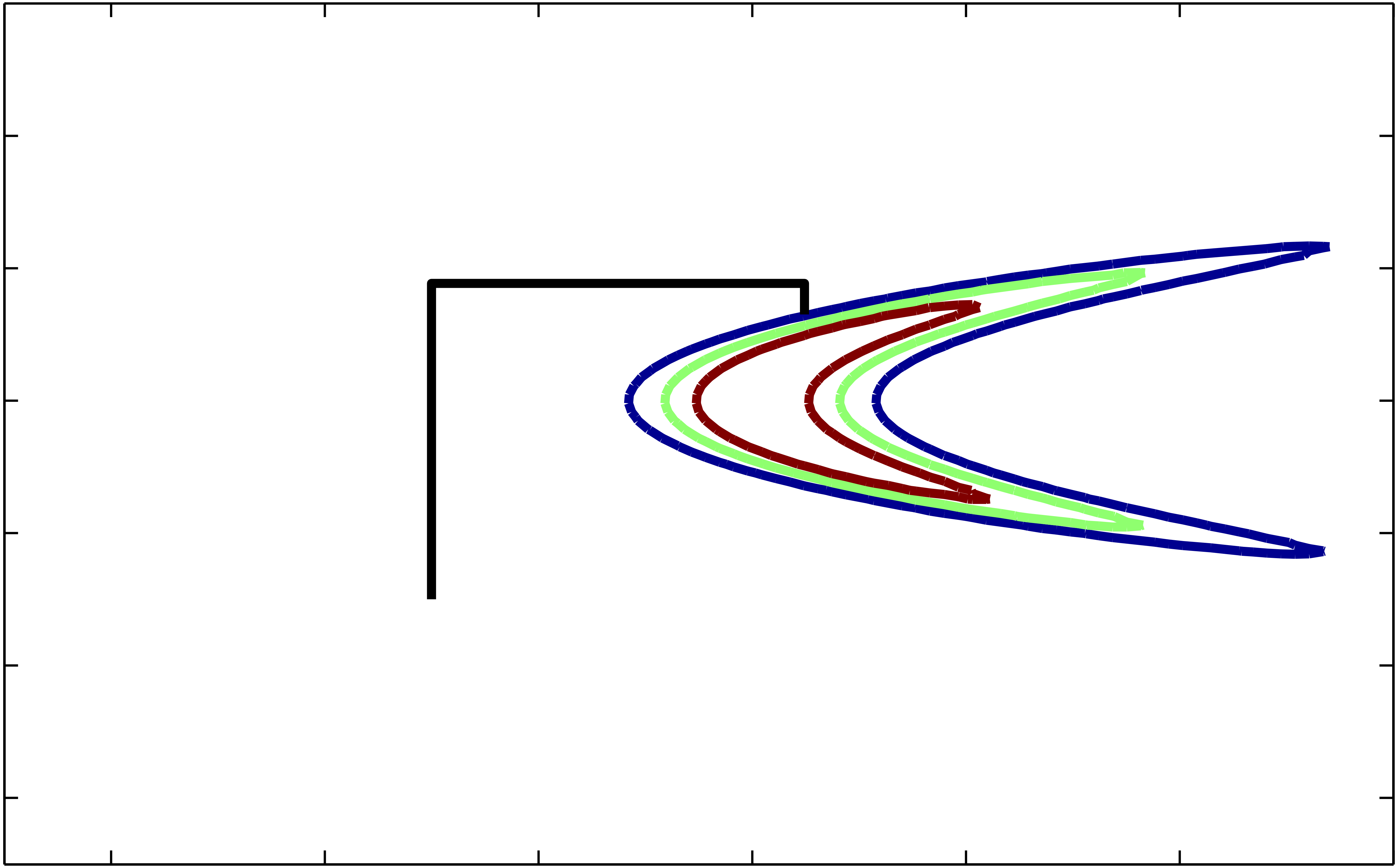
Multiple Dimensions

One Approach: Slice sample each dimension, as in Gibbs



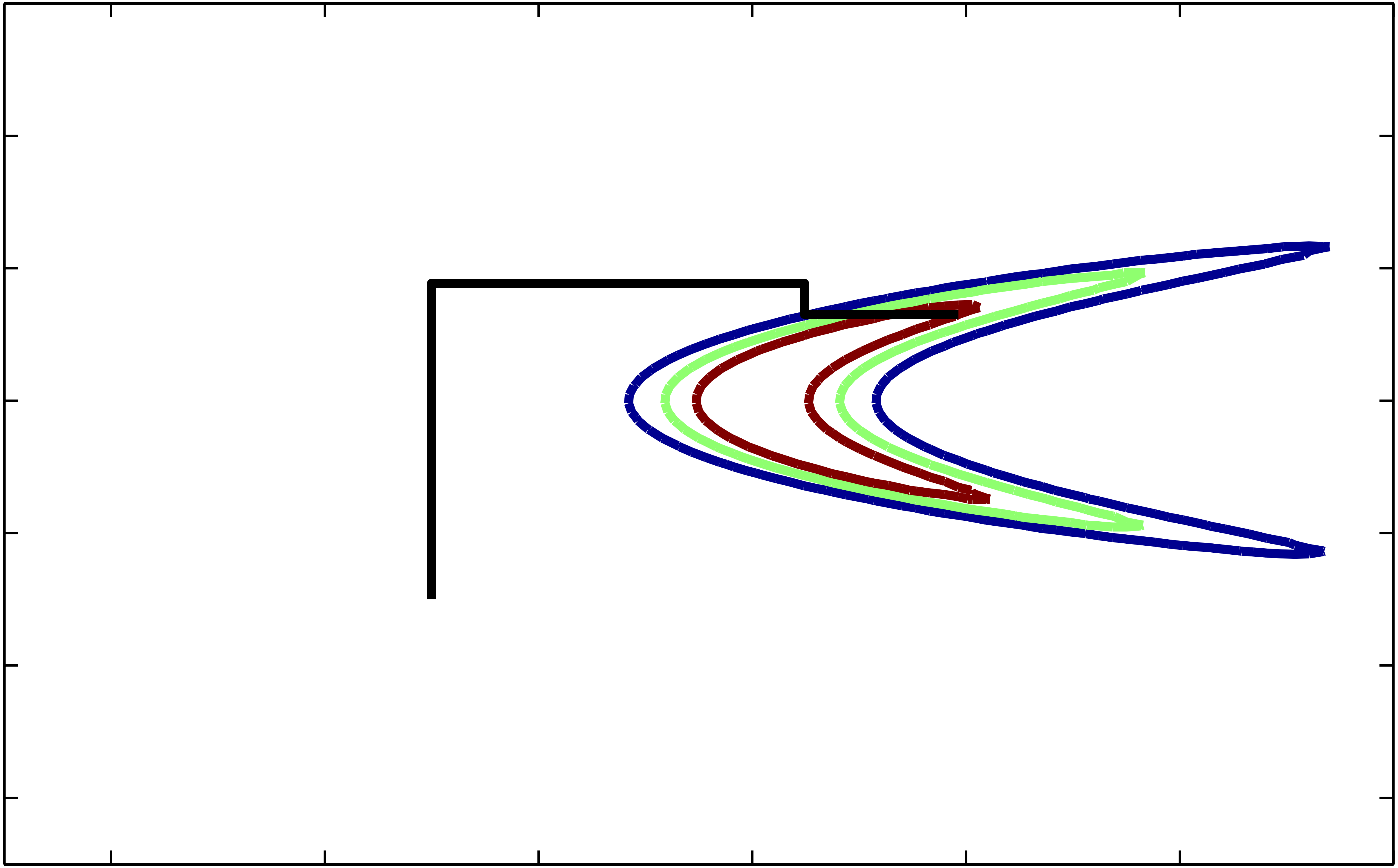
Multiple Dimensions

One Approach: Slice sample each dimension, as in Gibbs



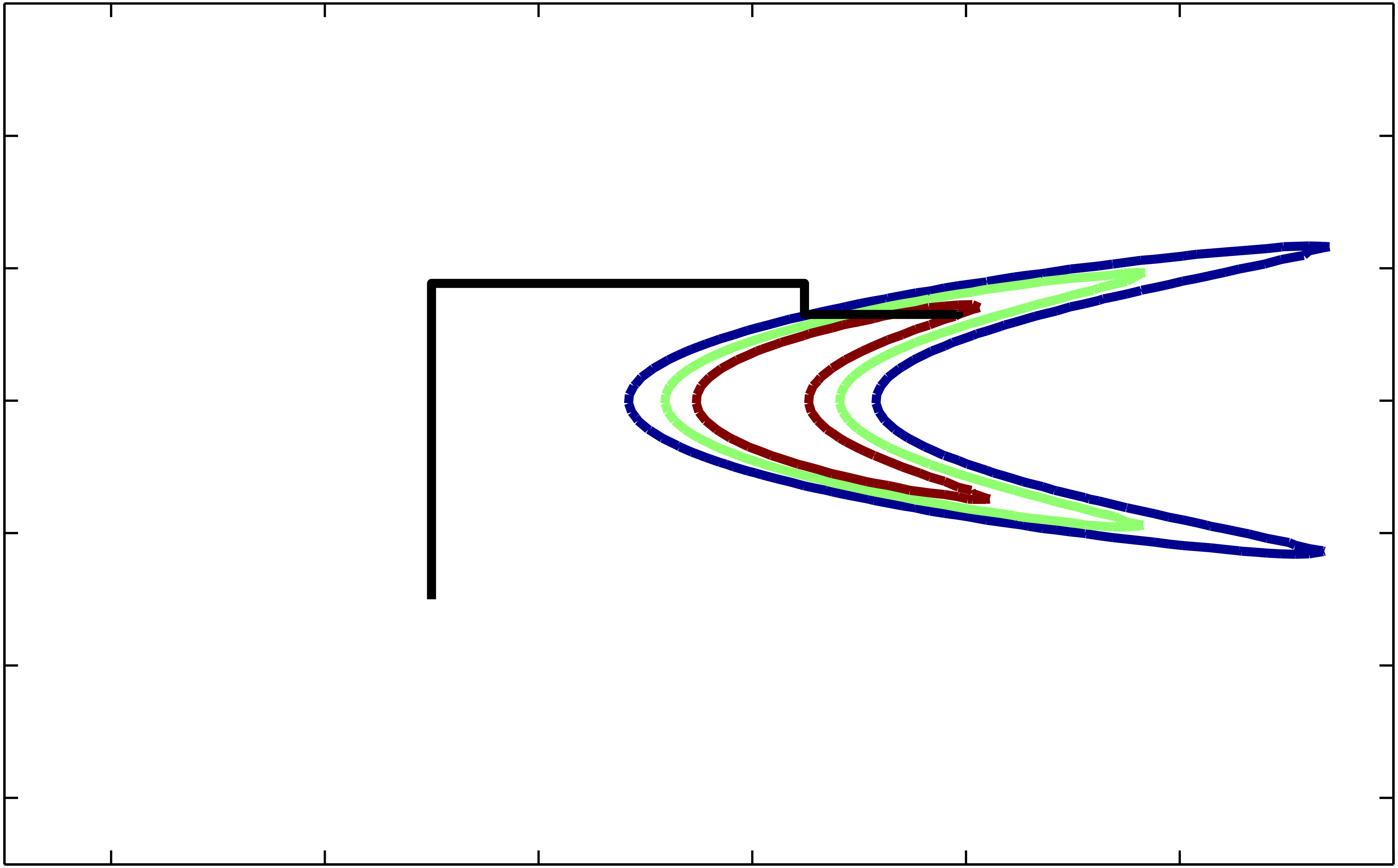
Multiple Dimensions

One Approach: Slice sample each dimension, as in Gibbs



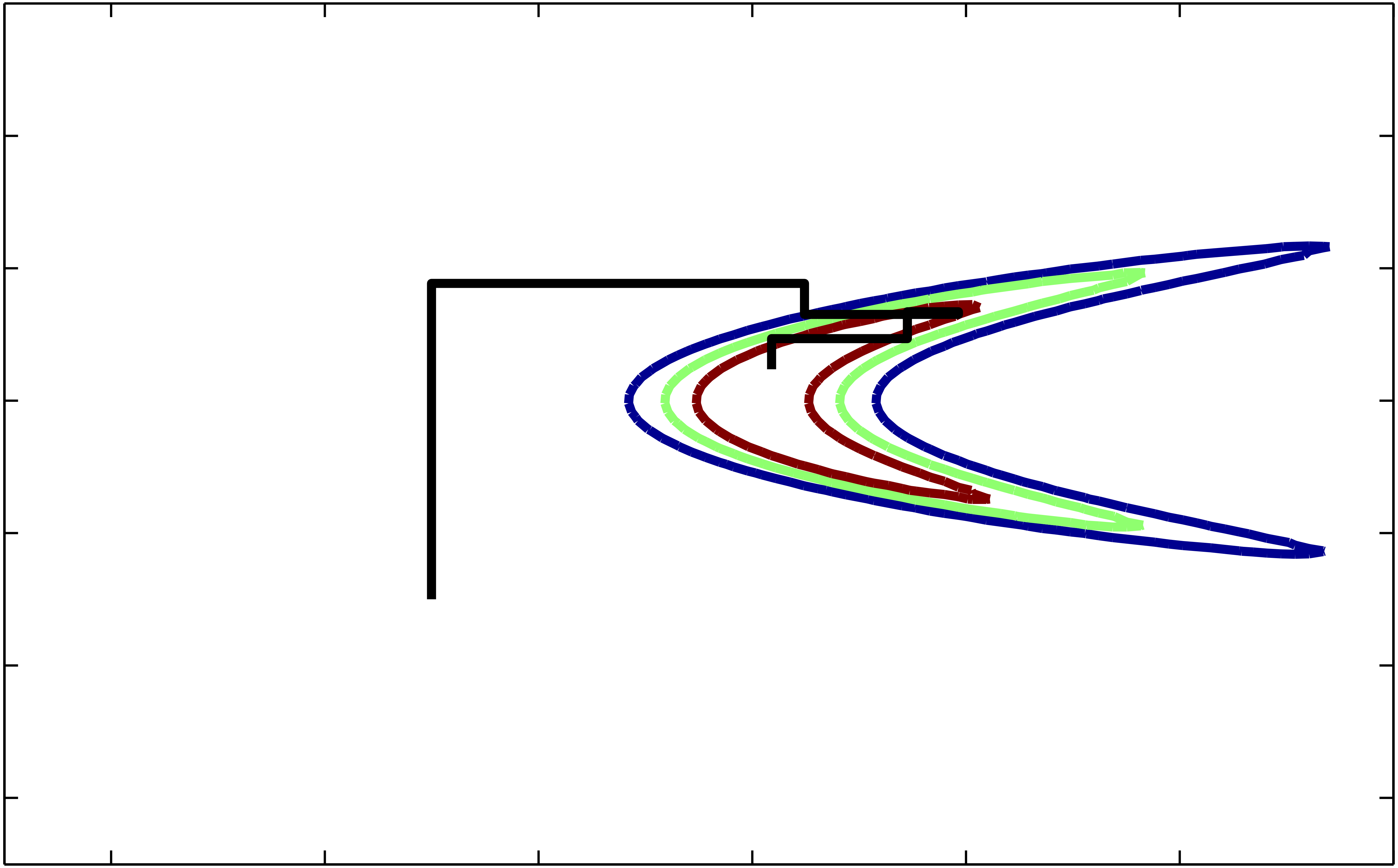
Multiple Dimensions

One Approach: Slice sample each dimension, as in Gibbs



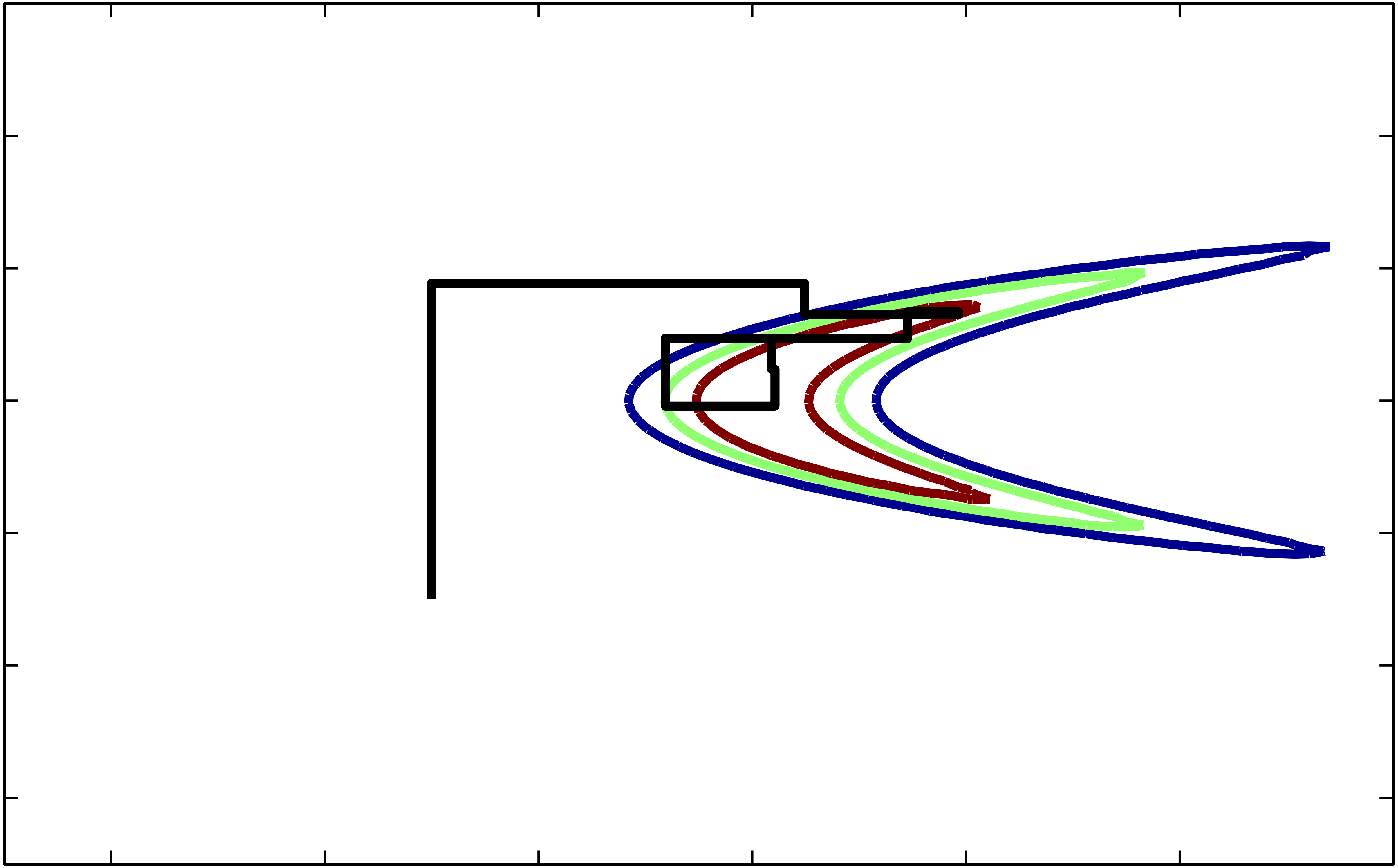
Multiple Dimensions

One Approach: Slice sample each dimension, as in Gibbs



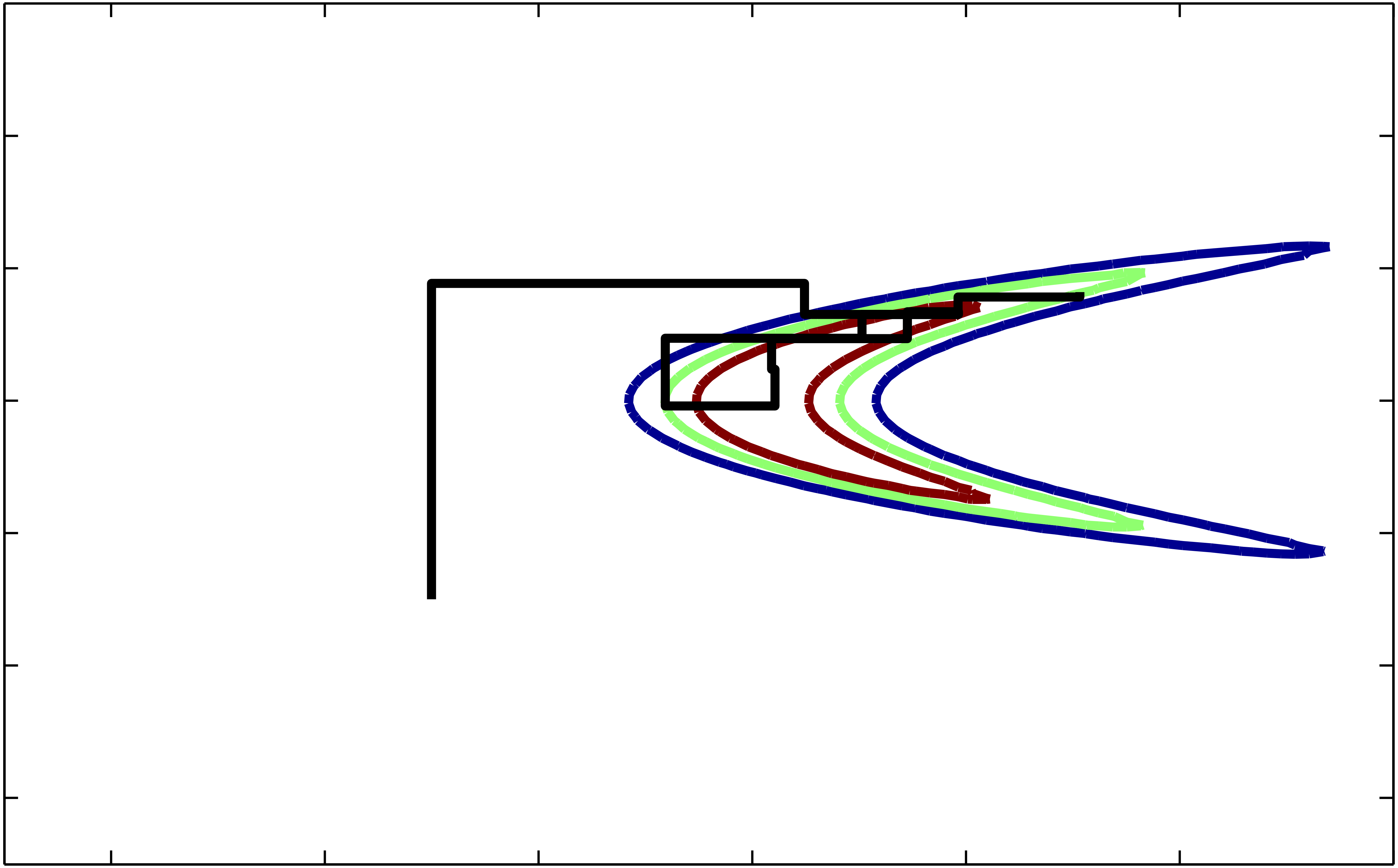
Multiple Dimensions

One Approach: Slice sample each dimension, as in Gibbs



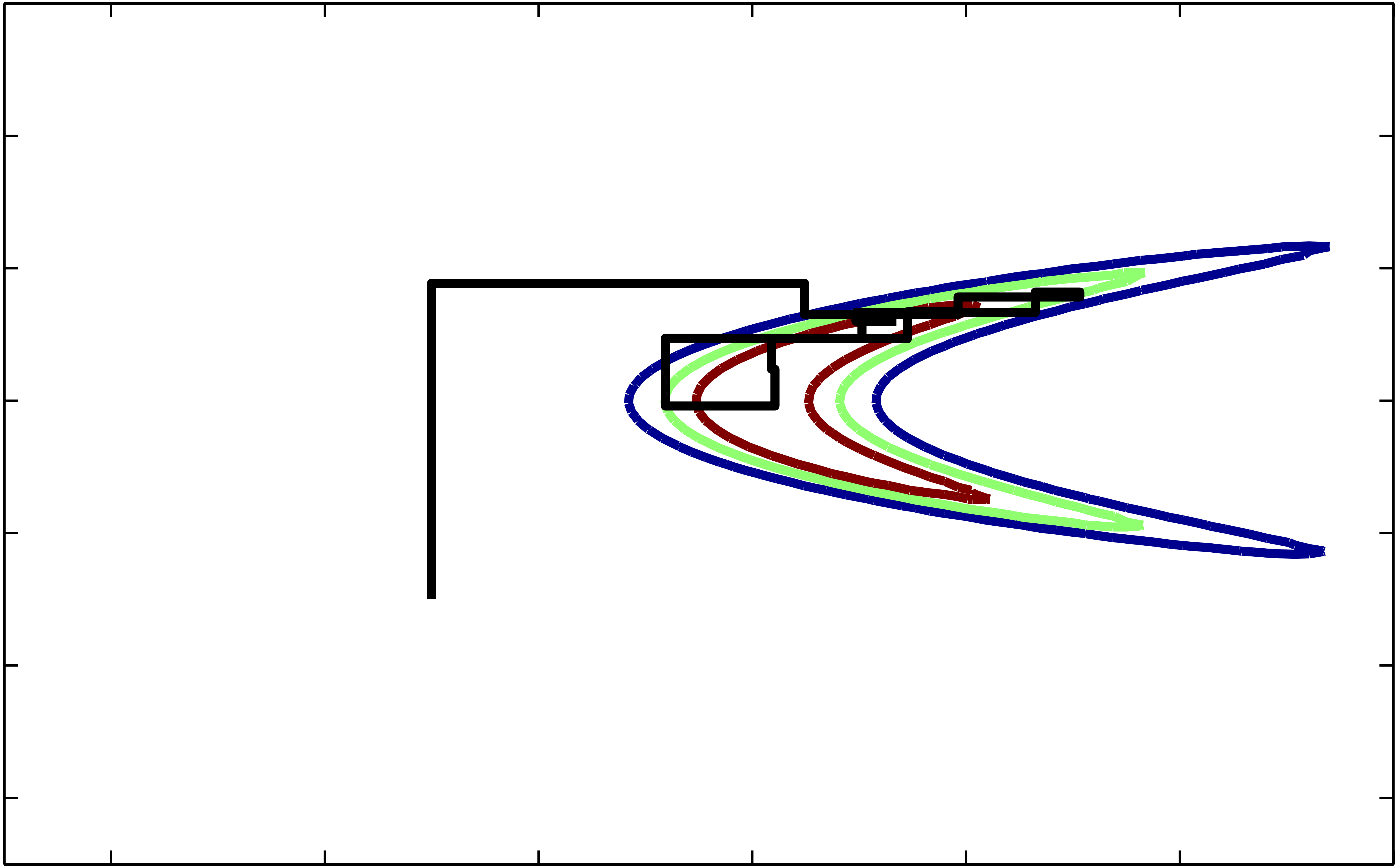
Multiple Dimensions

One Approach: Slice sample each dimension, as in Gibbs



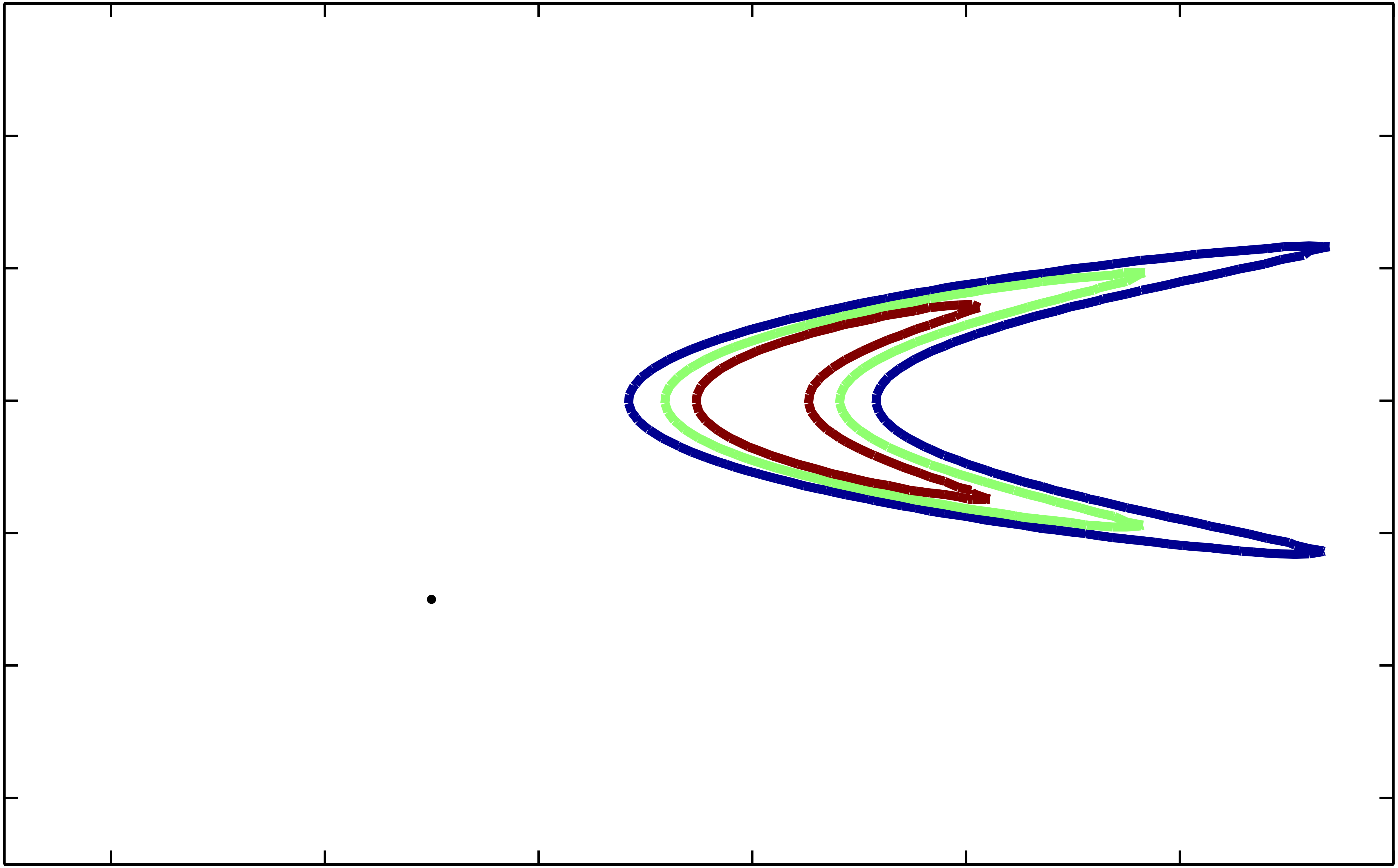
Multiple Dimensions

One Approach: Slice sample each dimension, as in Gibbs



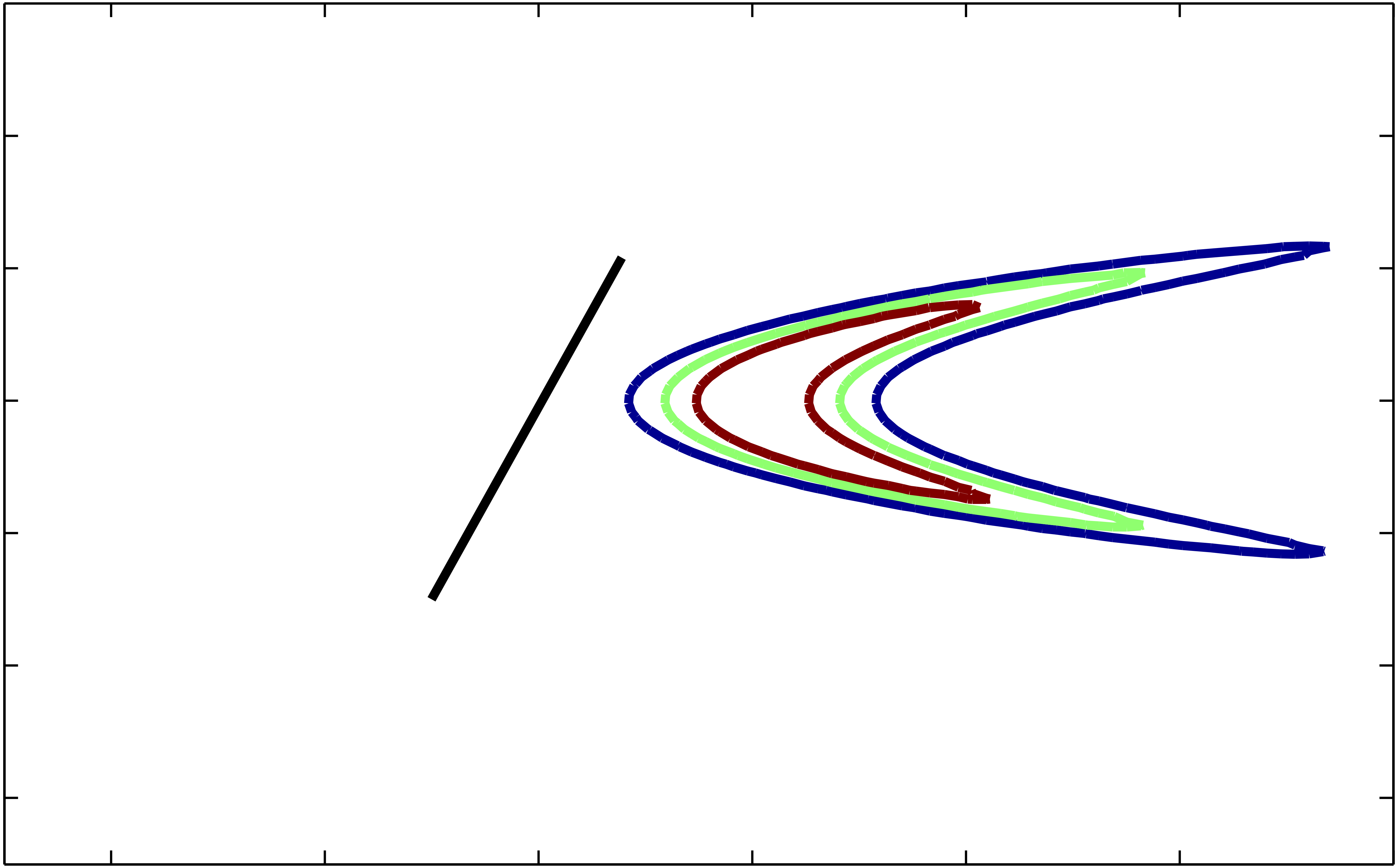
Multiple Dimensions

Another Approach: Slice sample in random directions



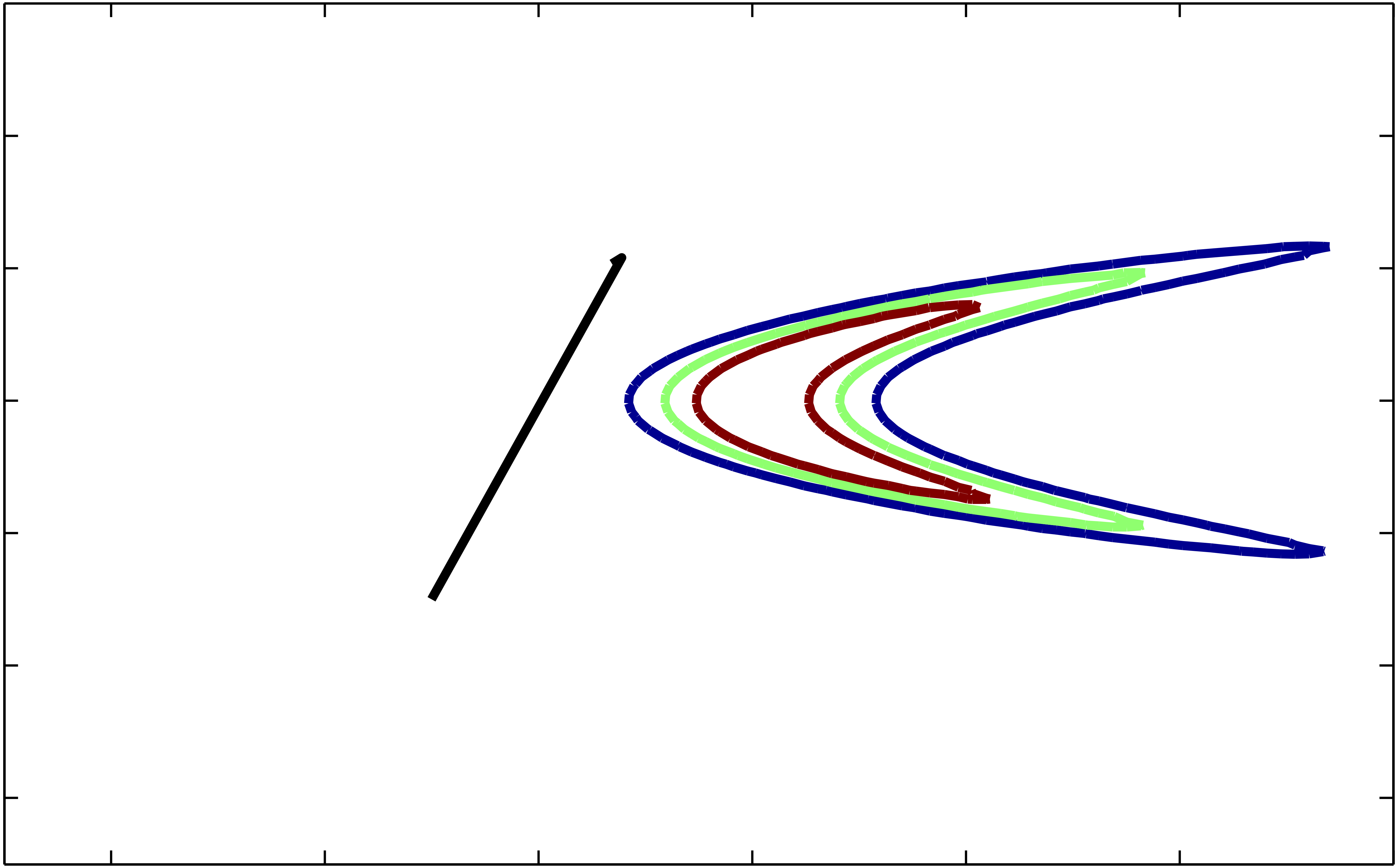
Multiple Dimensions

Another Approach: Slice sample in random directions



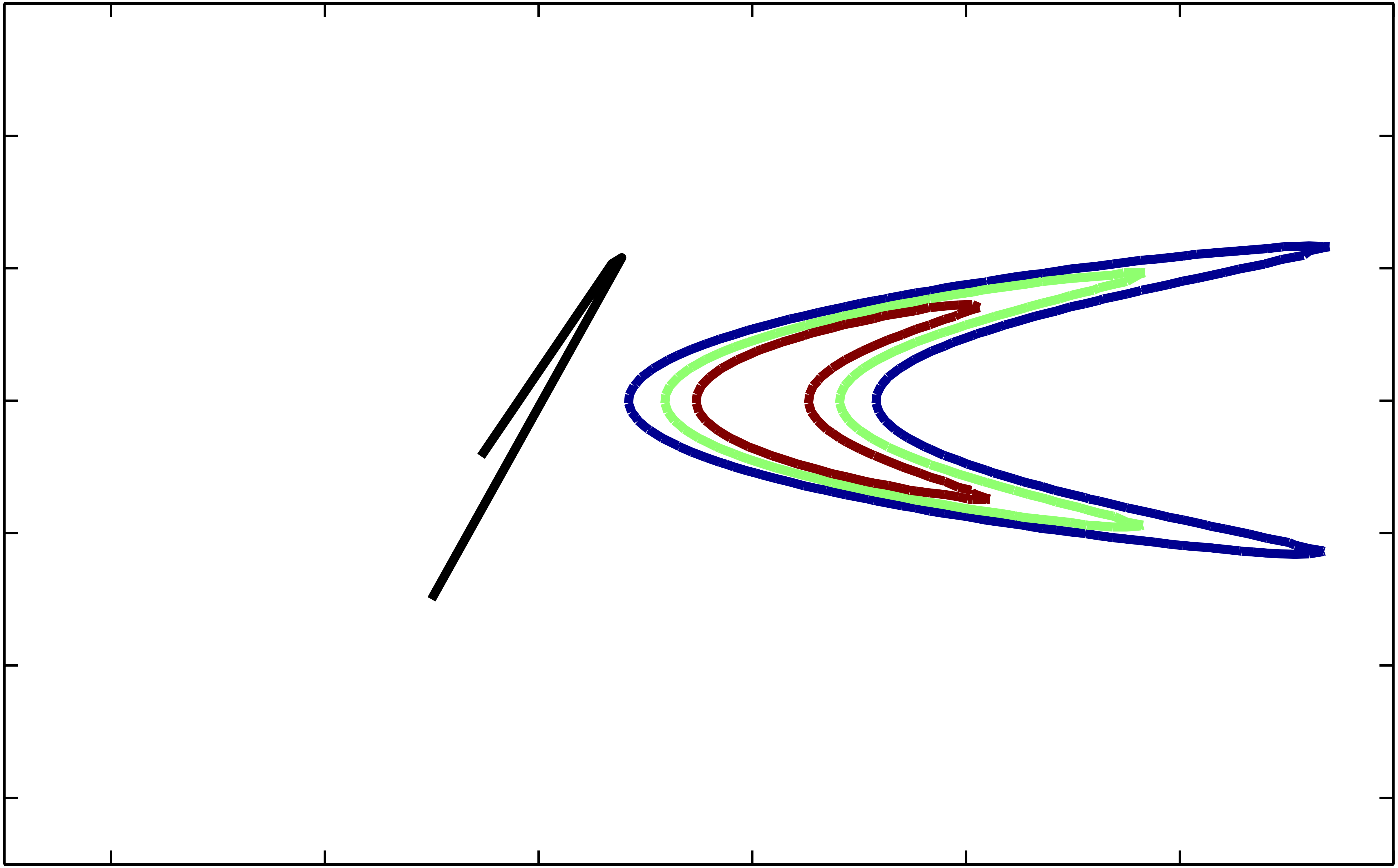
Multiple Dimensions

Another Approach: Slice sample in random directions



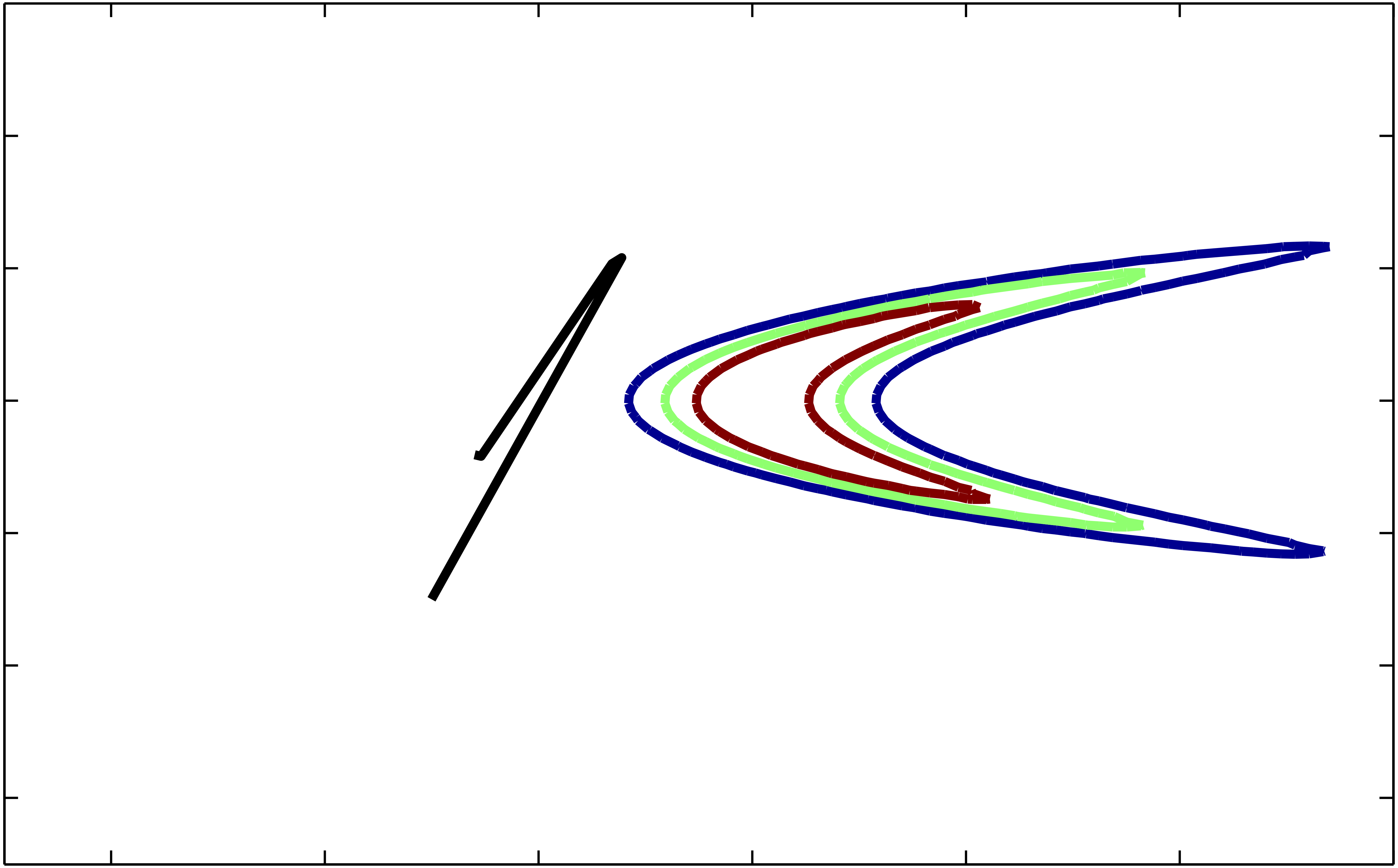
Multiple Dimensions

Another Approach: Slice sample in random directions



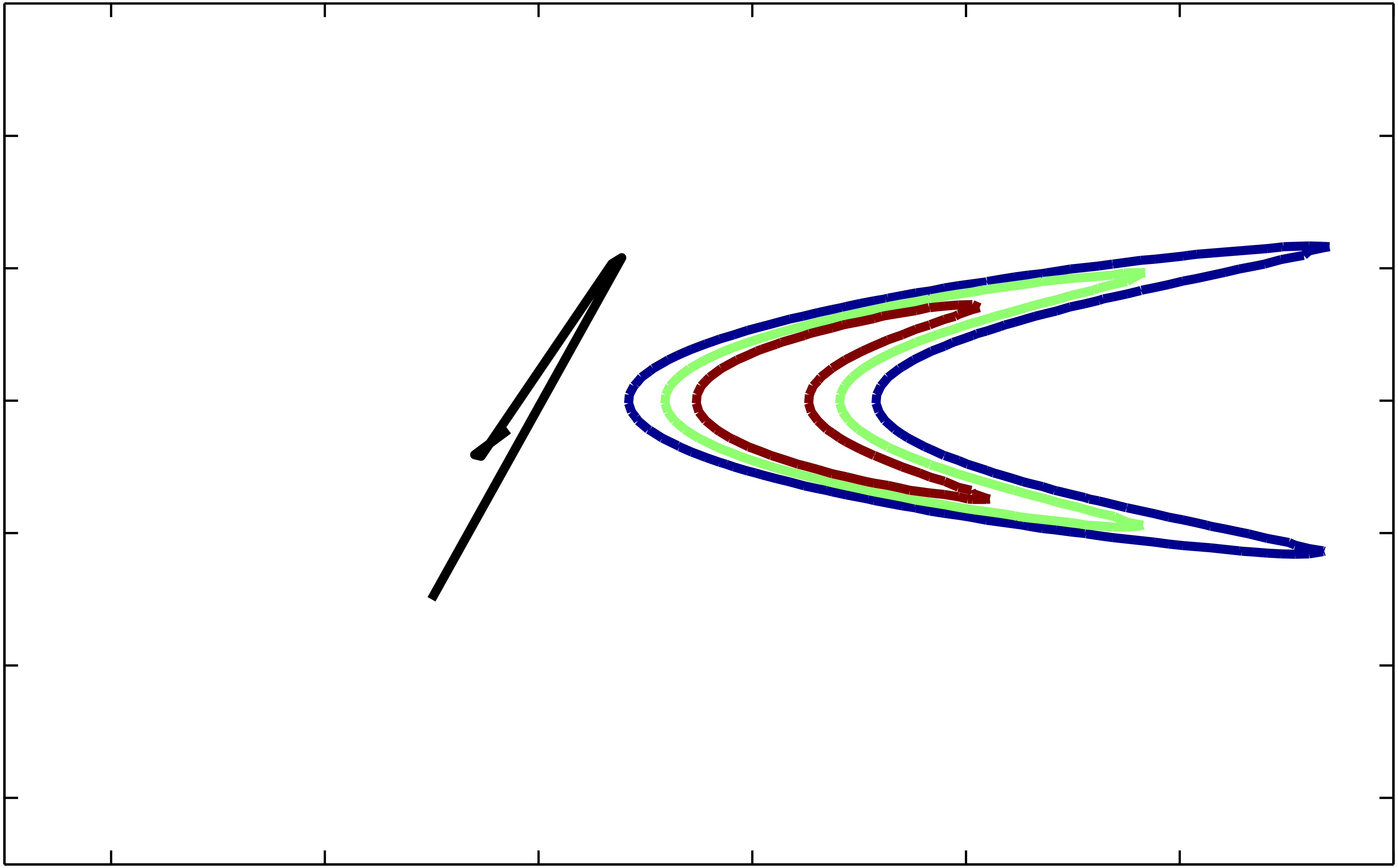
Multiple Dimensions

Another Approach: Slice sample in random directions



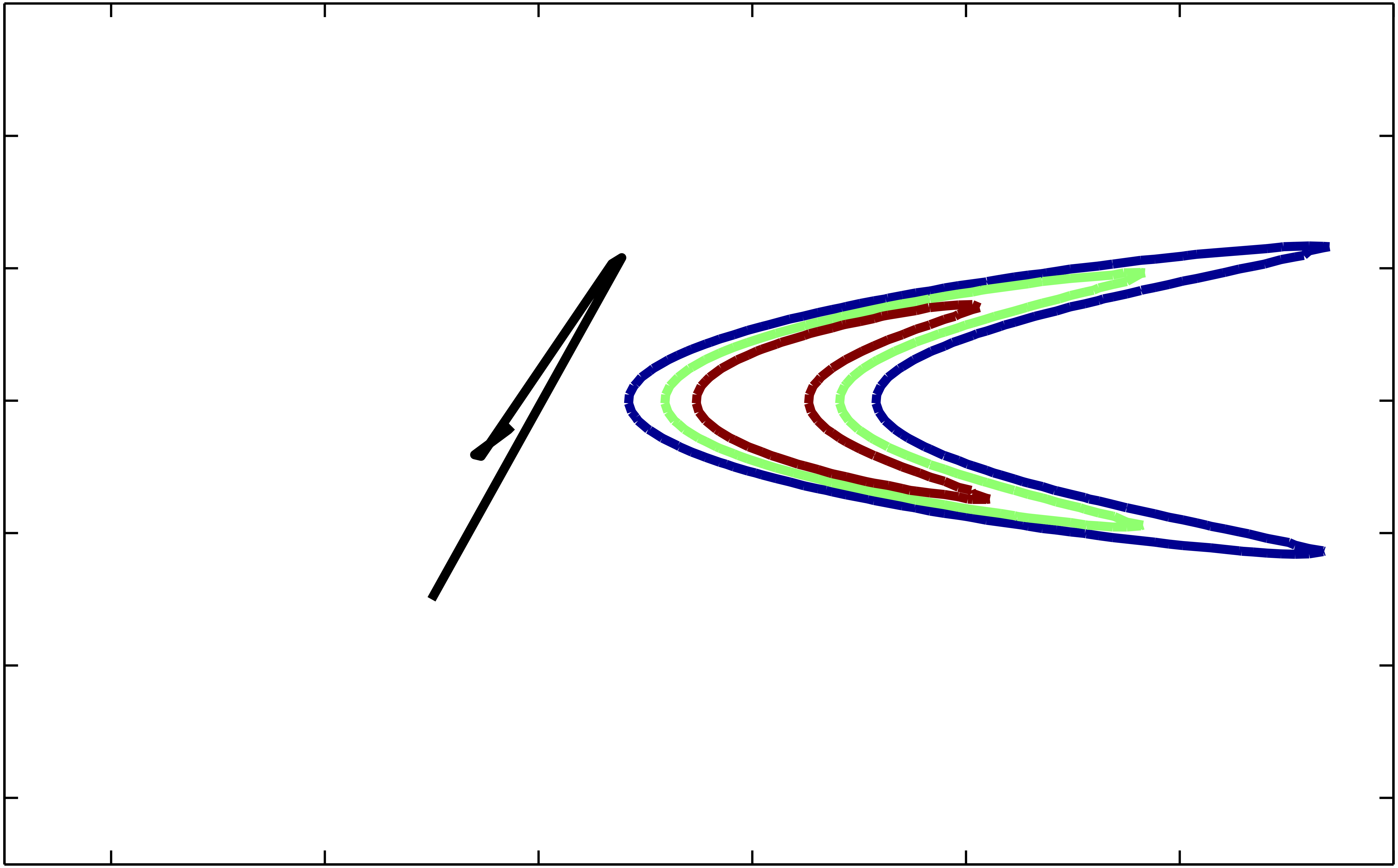
Multiple Dimensions

Another Approach: Slice sample in random directions



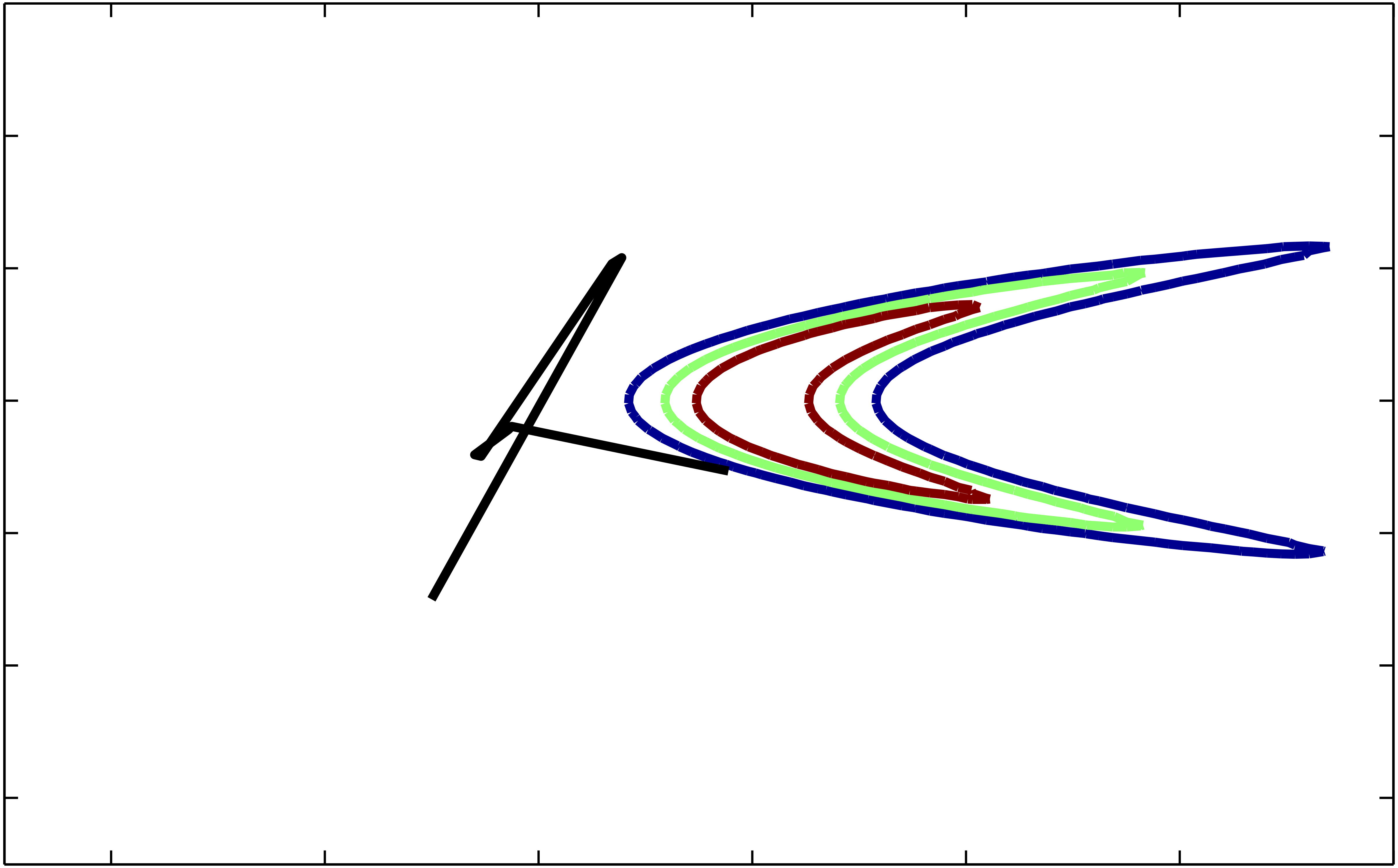
Multiple Dimensions

Another Approach: Slice sample in random directions



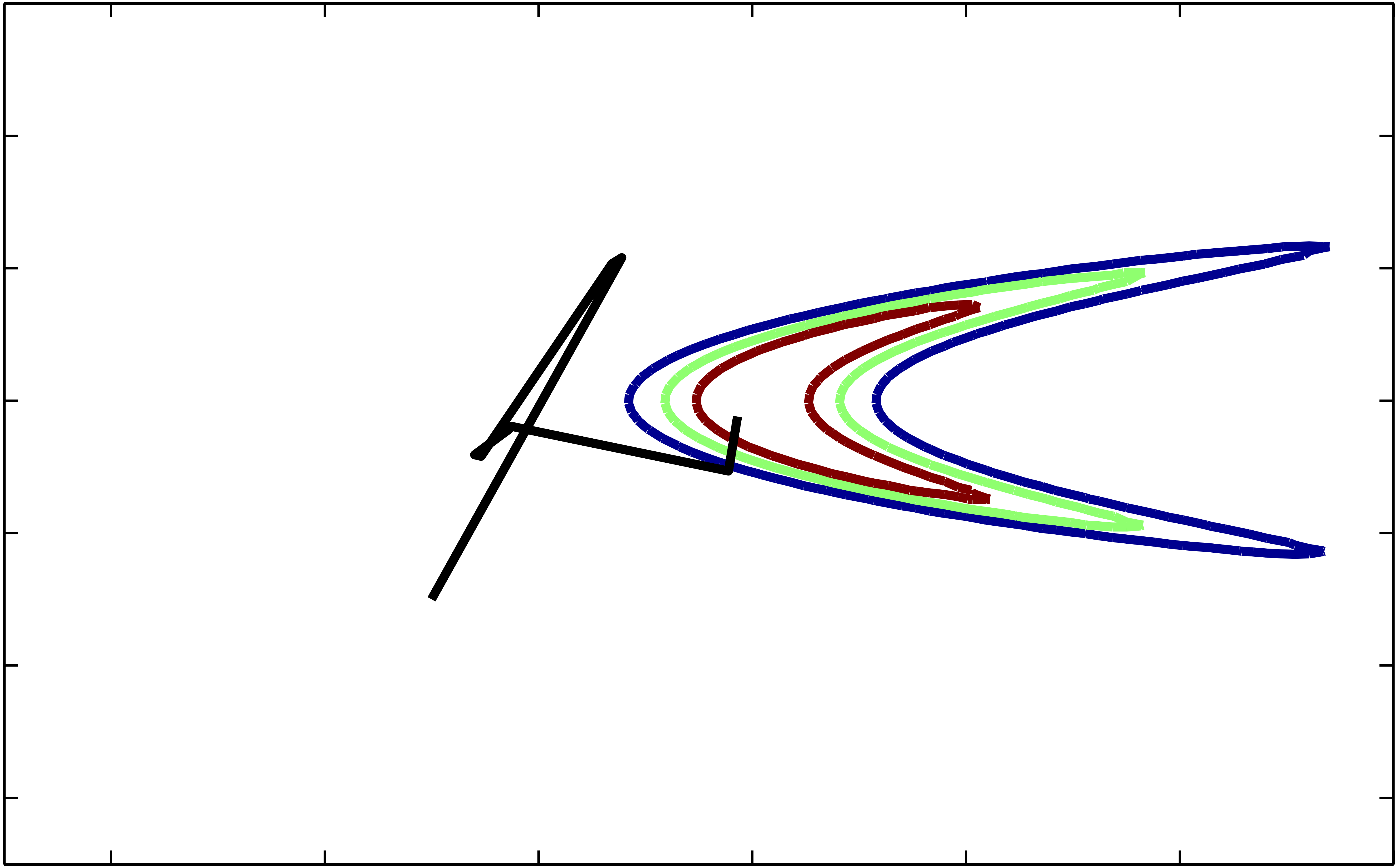
Multiple Dimensions

Another Approach: Slice sample in random directions



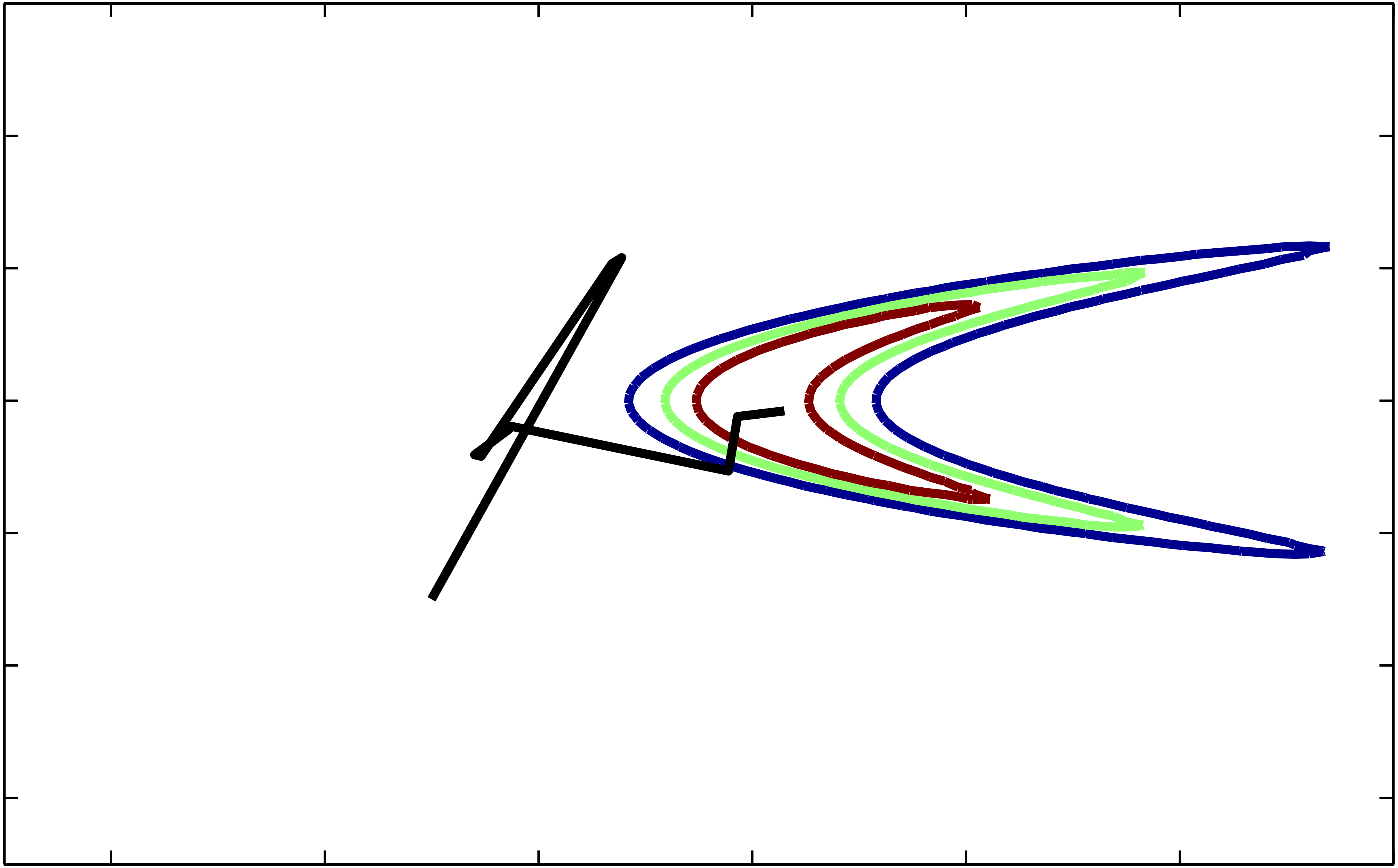
Multiple Dimensions

Another Approach: Slice sample in random directions



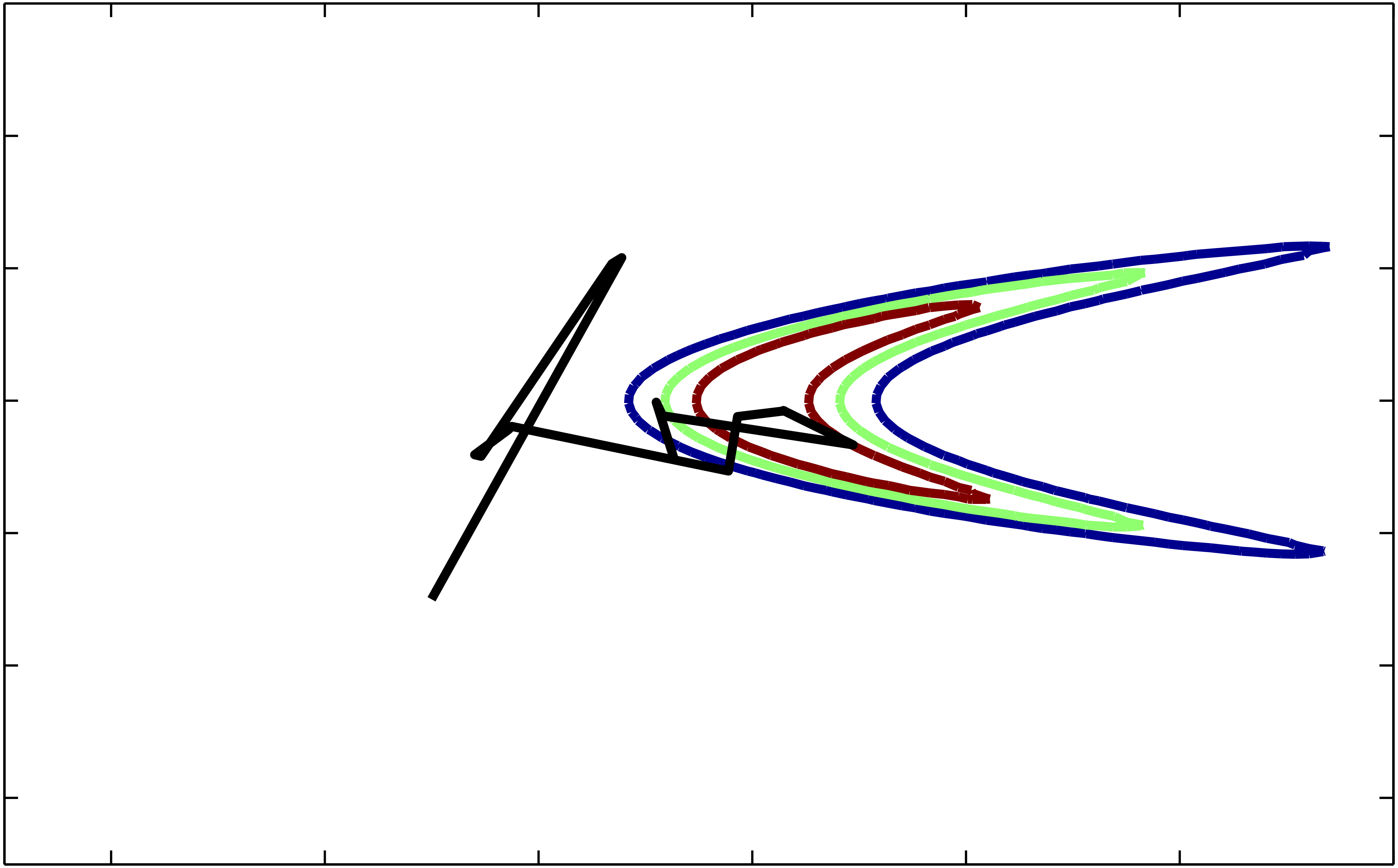
Multiple Dimensions

Another Approach: Slice sample in random directions



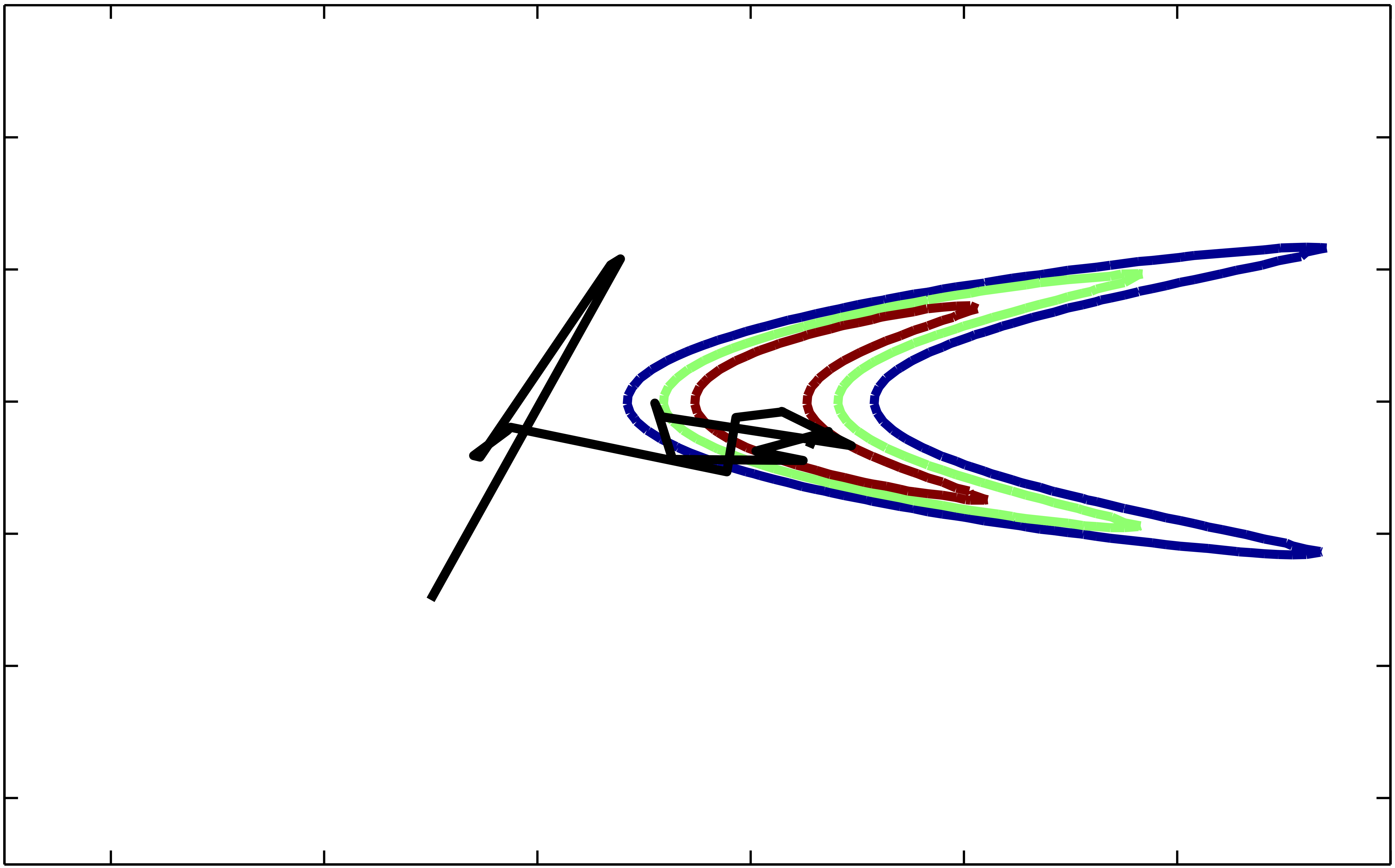
Multiple Dimensions

Another Approach: Slice sample in random directions



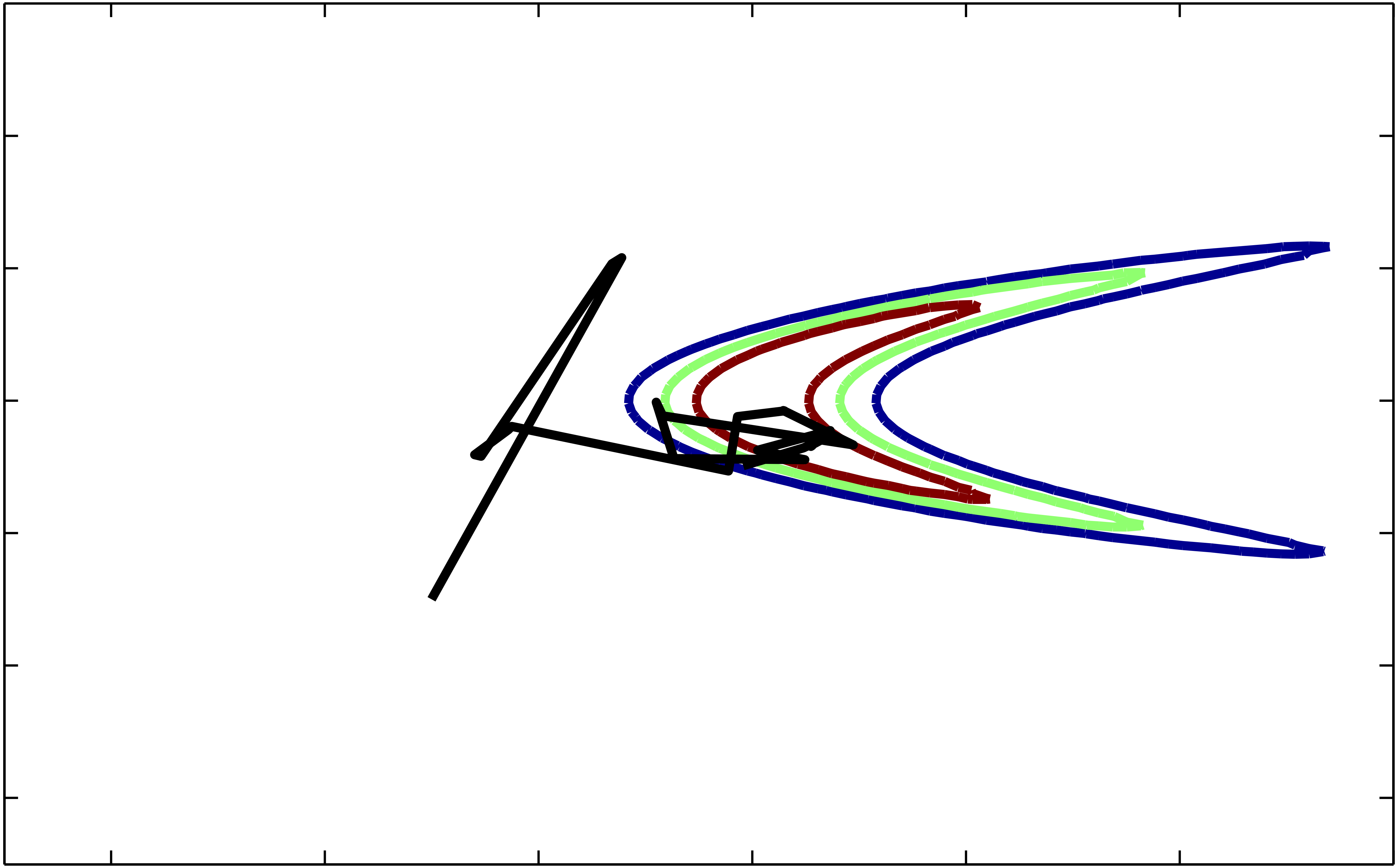
Multiple Dimensions

Another Approach: Slice sample in random directions



Multiple Dimensions

Another Approach: Slice sample in random directions



Auxiliary Variables

Slice sampling is an example of a very useful trick.

Getting marginal distributions in MCMC is easy: just throw away the things you're not interested in.

Sometimes it is easy to create an **expanded** joint distribution that is easier to sample from, but has the marginal distribution that you're interested in.

In slice sampling, this is the height variable.

$$p(x,u) = \pi(x)\,p(u\,|\,x)$$

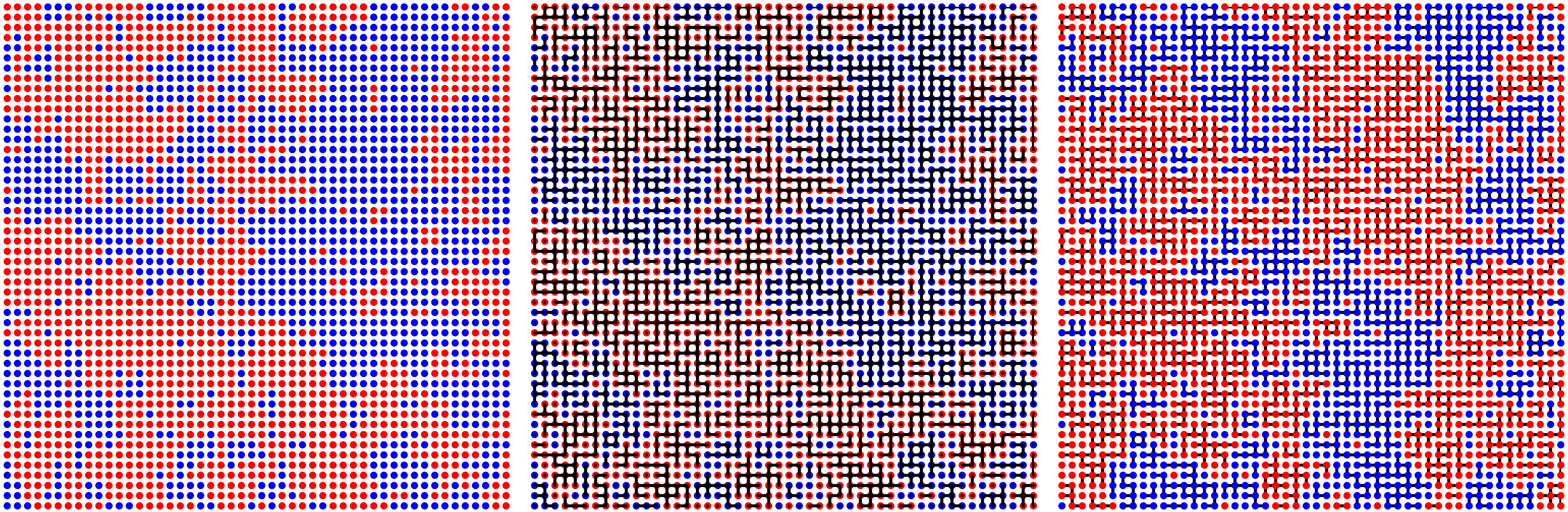
$$\pi(x) = \int p(x,u)\,\mathrm{d}u = \pi(x)\int p(u\,|\,x)\,\mathrm{d}u$$

Auxiliary Variables

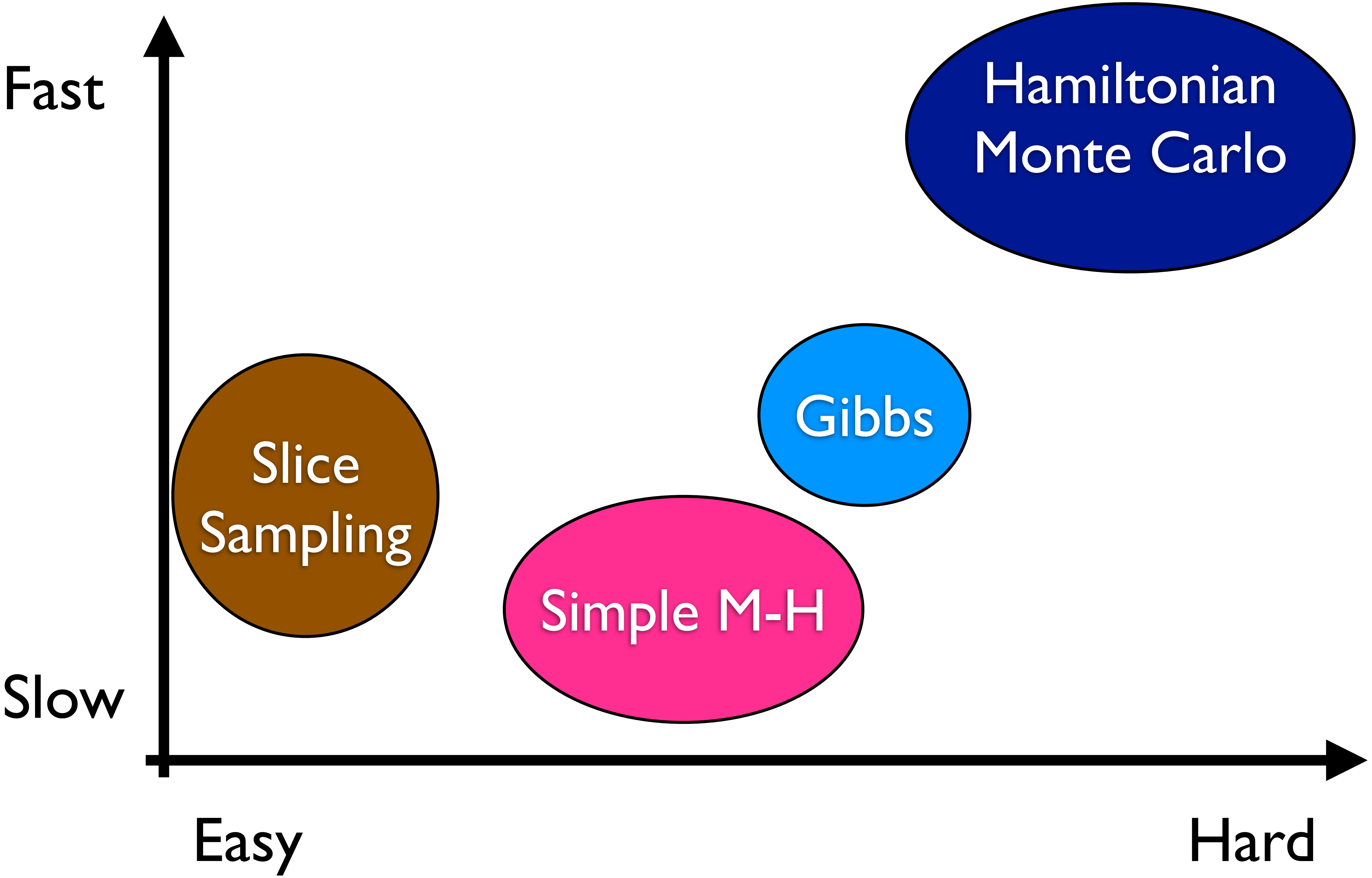
The auxiliary variable trick comes up all the time and is immensely useful in making many problems easier.

It's counterintuitive, however: we're **increasing** the dimensionality of our problem.

Such methods are a continuing area of active research.

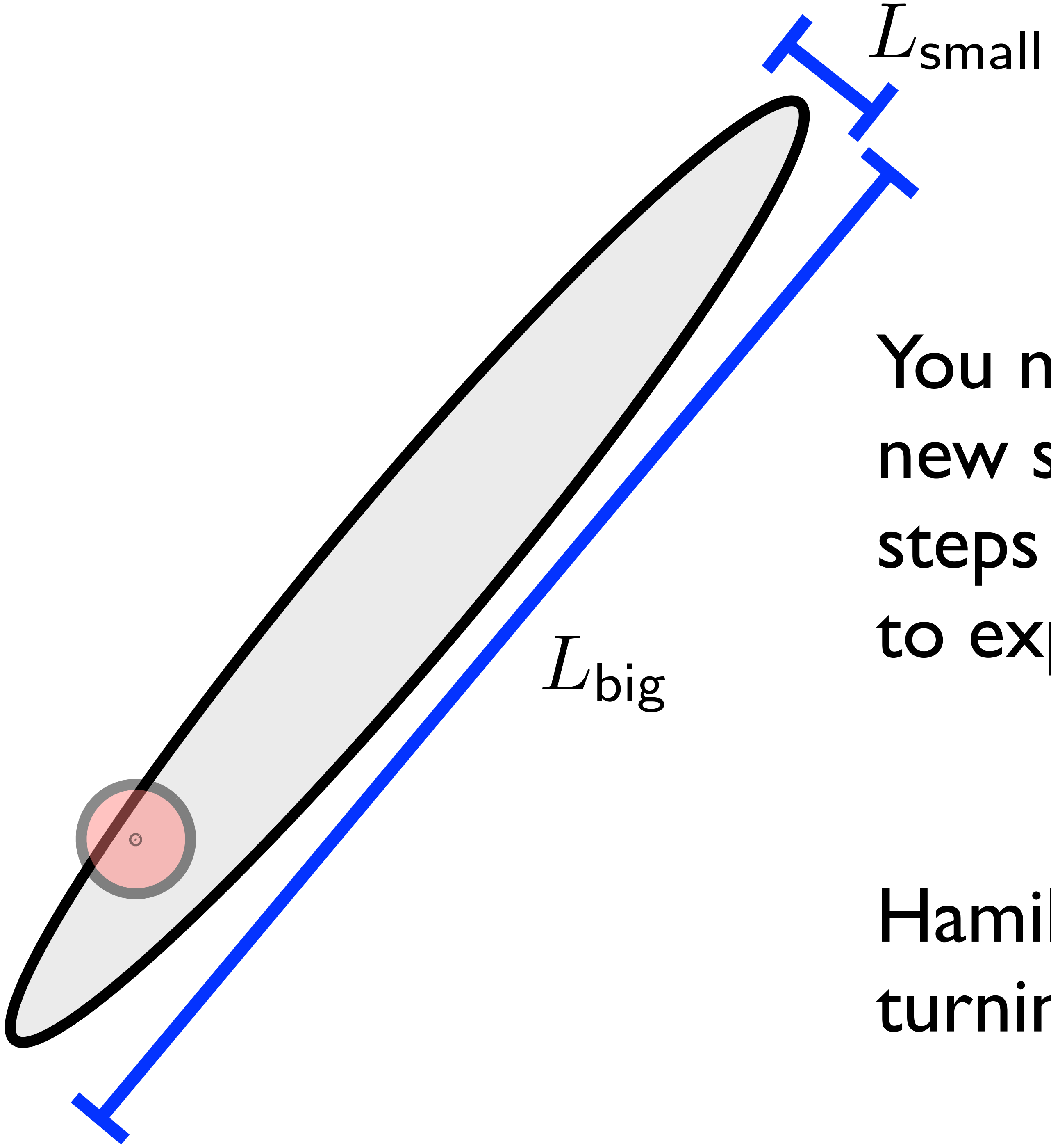


An MCMC Cartoon



Avoiding Random Walks

All of the MCMC methods I've talked about so far have been based on biased random walks.



You need to go about L_{big} to get a new sample, but you can only take steps around size L_{small} , so you have to expect it to take about

$$(L_{\text{big}}/L_{\text{small}})^2$$

Hamiltonian Monte Carlo is about turning this into $L_{\text{big}}/L_{\text{small}}$

Hamiltonian Monte Carlo

Hamiltonian (also "hybrid") Monte Carlo does MCMC by sampling from a fictitious dynamical system. It suppresses random walk behaviour via persistent motion.

Think of it as rolling a ball along a surface in such a way that the Markov chain has all of the properties we want.

Call the negative log probability an "energy".

$$\pi(x) = \frac{1}{\mathcal{Z}} e^{-E(x)}$$

Think of this as a "gravitational potential energy" for the rolling ball. The ball wants to roll downhill towards low energy (high probability) regions.

Hamiltonian Monte Carlo

Now, introduce auxiliary variables ρ (with the same dimensionality as our state space) that we will call "momenta".

Give these momenta a distribution and call the negative log probability of that the "kinetic energy". A convenient form is (not surprisingly) the unit-variance Gaussian.

$$p(\rho) = \frac{1}{\mathcal{Z}} e^{-K(\rho)}$$

$$K(\rho) = \frac{1}{2} \rho^{\mathsf{T}} \rho$$

$$p(x, \rho) \propto e^{-E(x) - K(\rho)}$$

As with other auxiliary variable methods, marginalizing out the momenta gives us back the distribution of interest.

Hamiltonian Monte Carlo

We can now simulate Hamiltonian dynamics, i.e., roll the ball around the surface. Even as the energy sloshes between potential and kinetic, the Hamiltonian is constant.

The corresponding joint distribution is invariant to this.

$$p(x, \rho) \propto e^{-E(x)-K(\rho)}$$

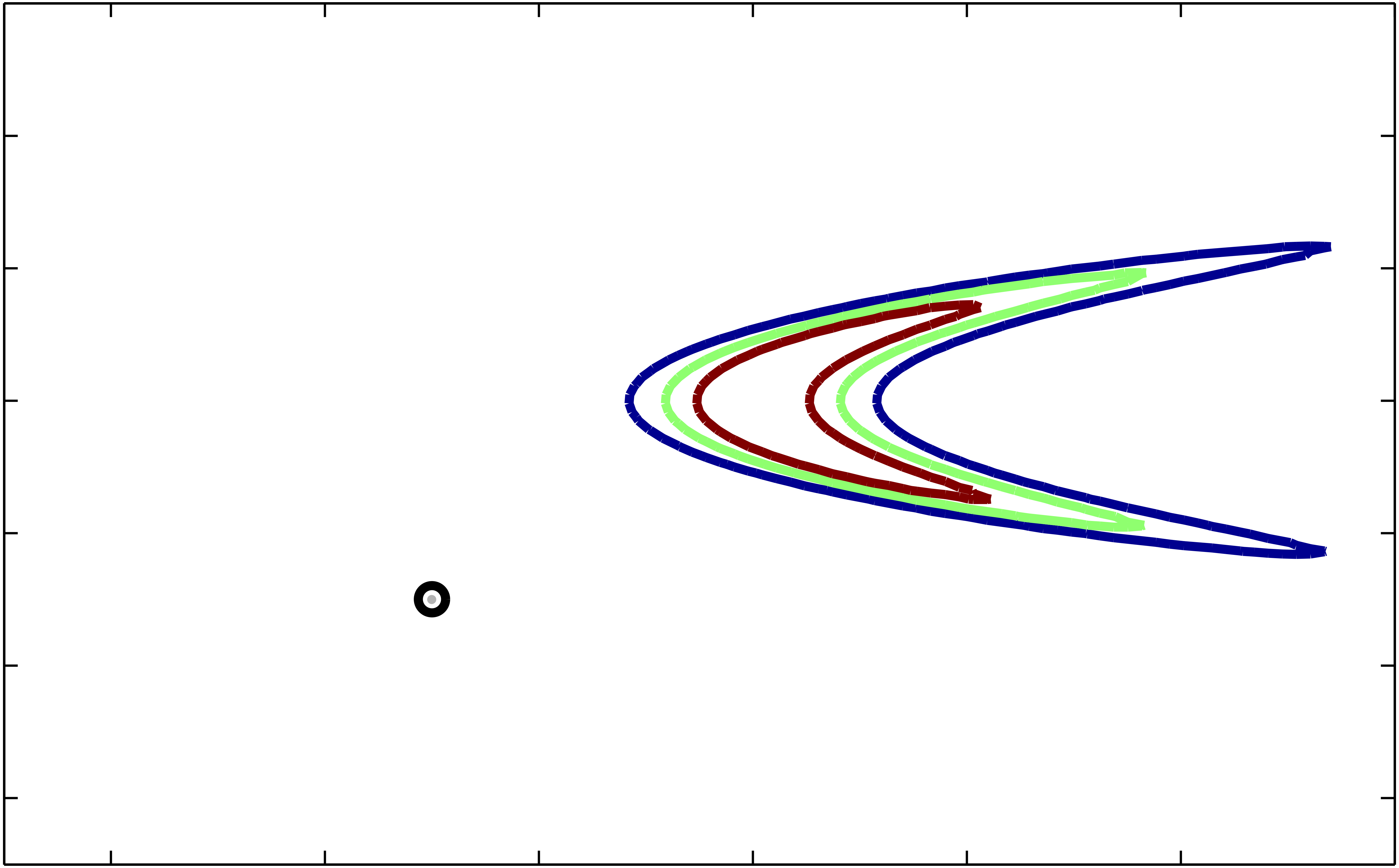
This is not ergodic, of course. This is usually resolved by randomizing the momenta, which is easy because they are independent and Gaussian.

So, HMC consists of two kind of MCMC moves:

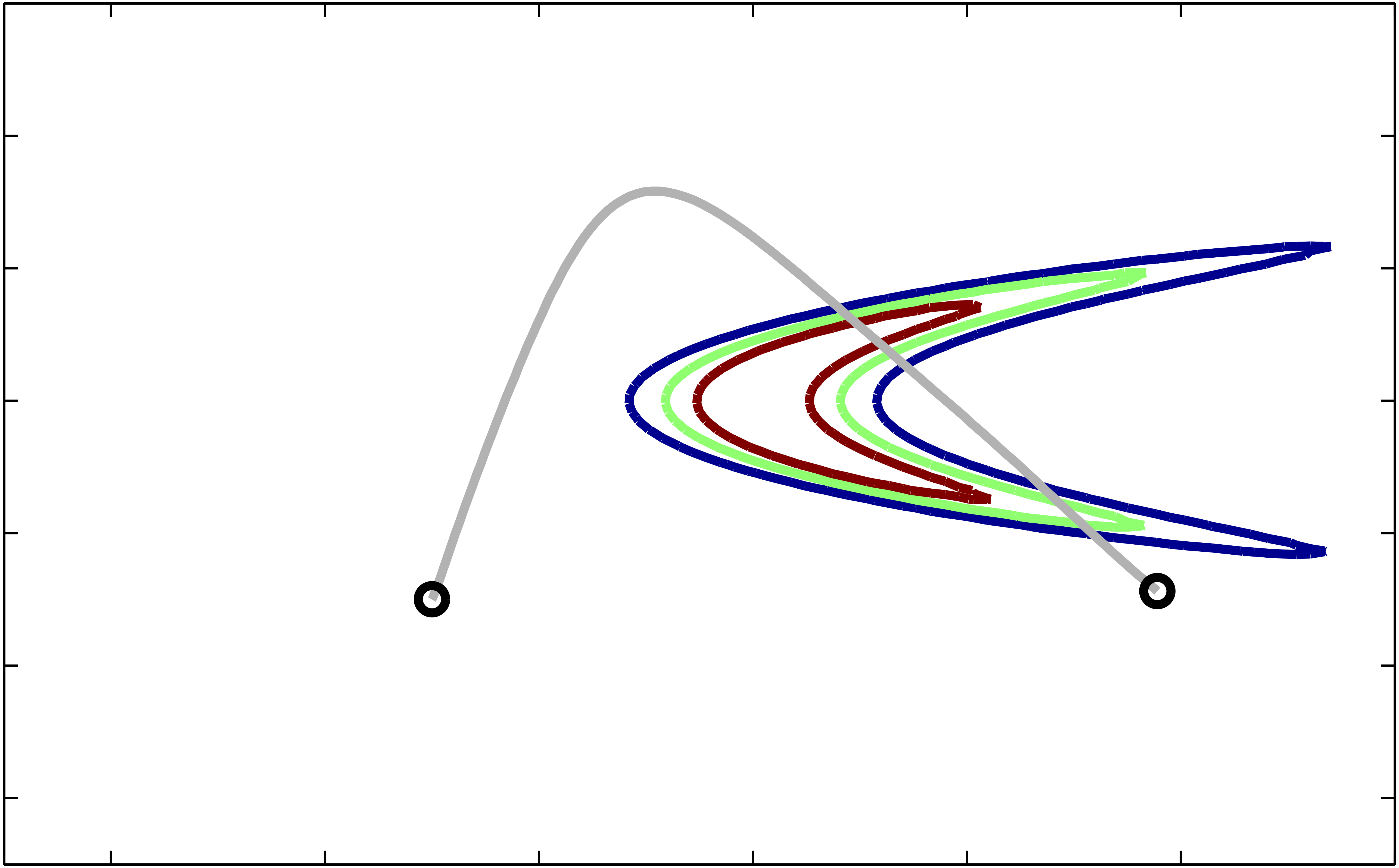
1) Randomize the momenta.

2) Simulate the dynamics, starting with these momenta.

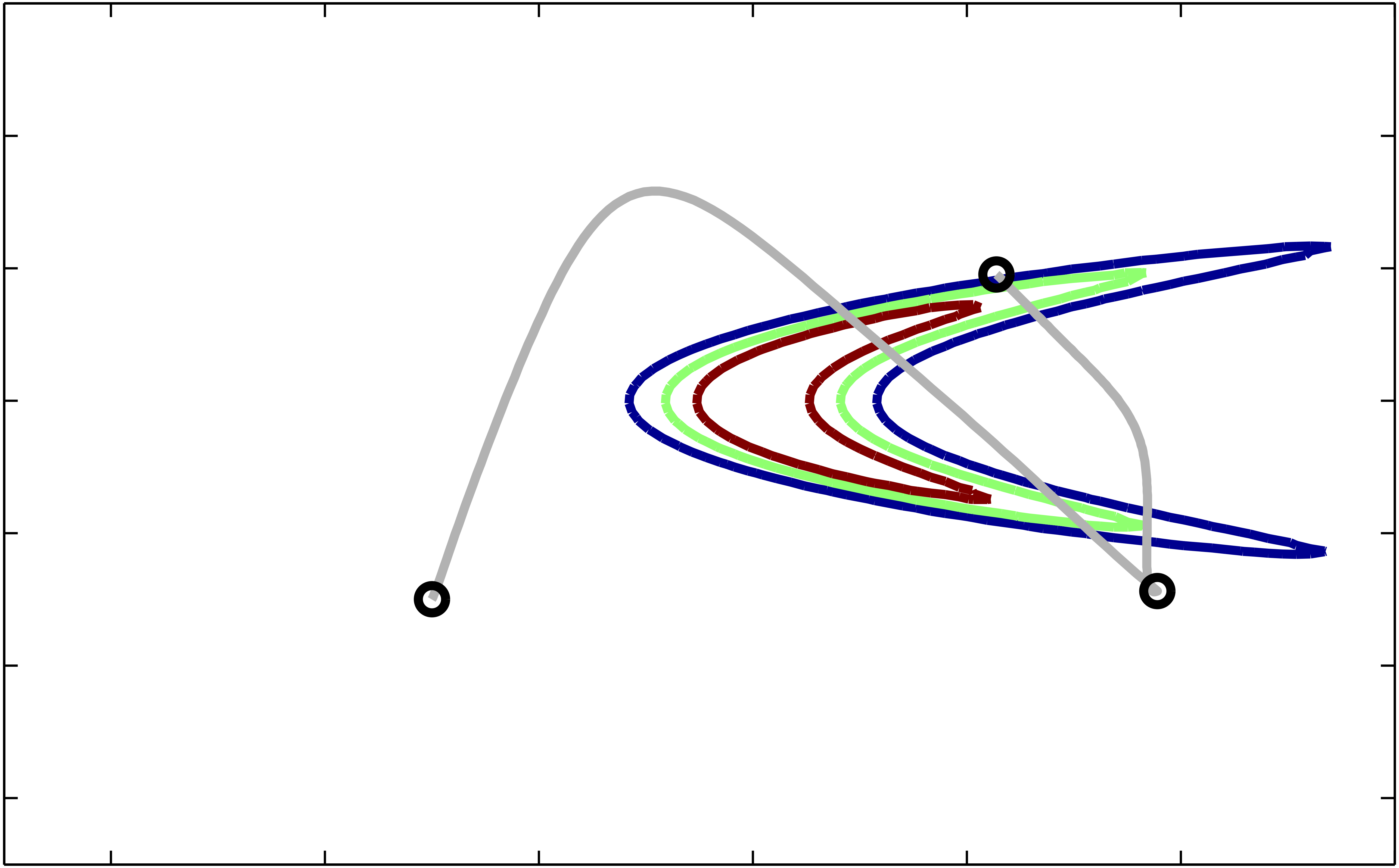
Alternating HMC



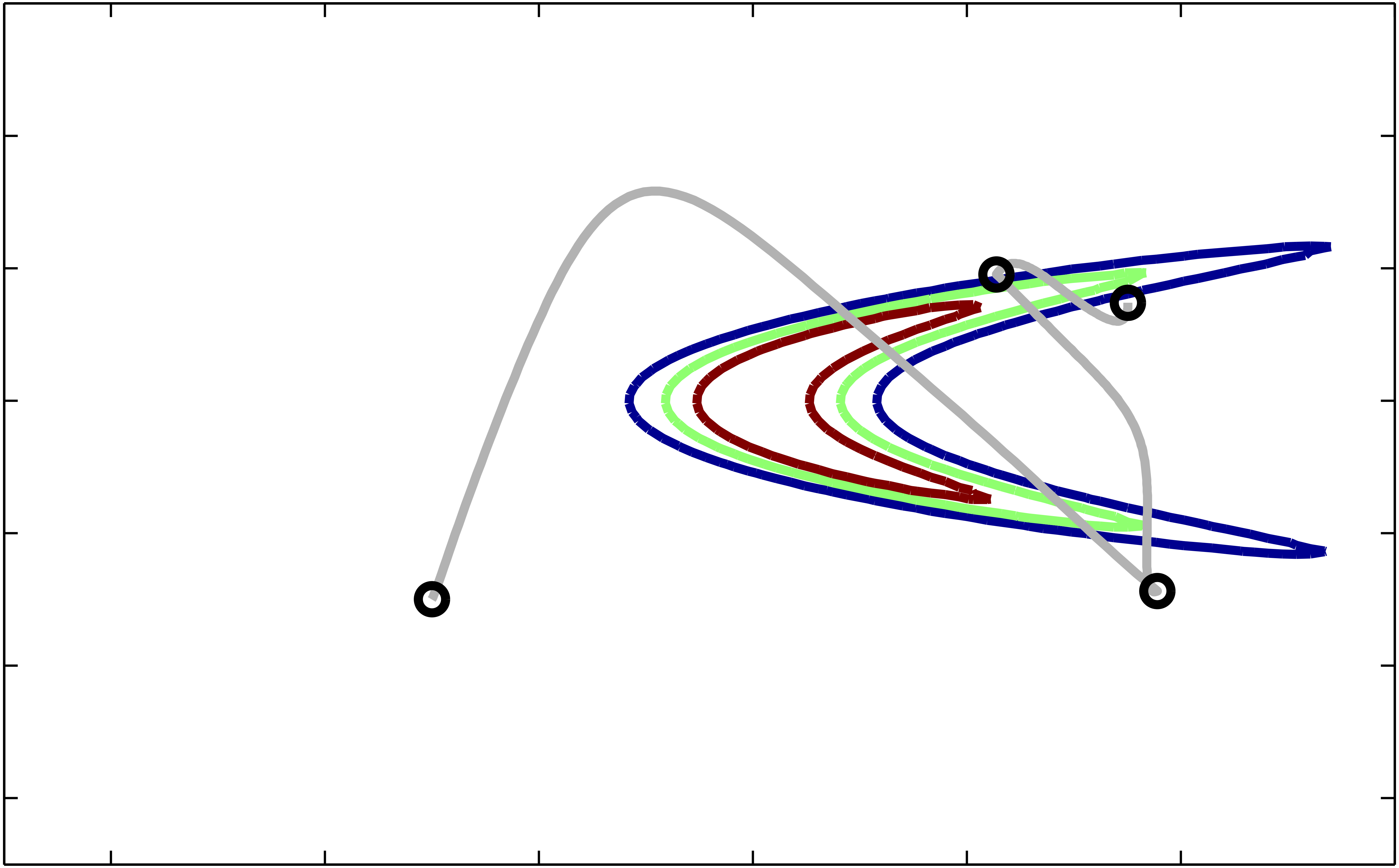
Alternating HMC



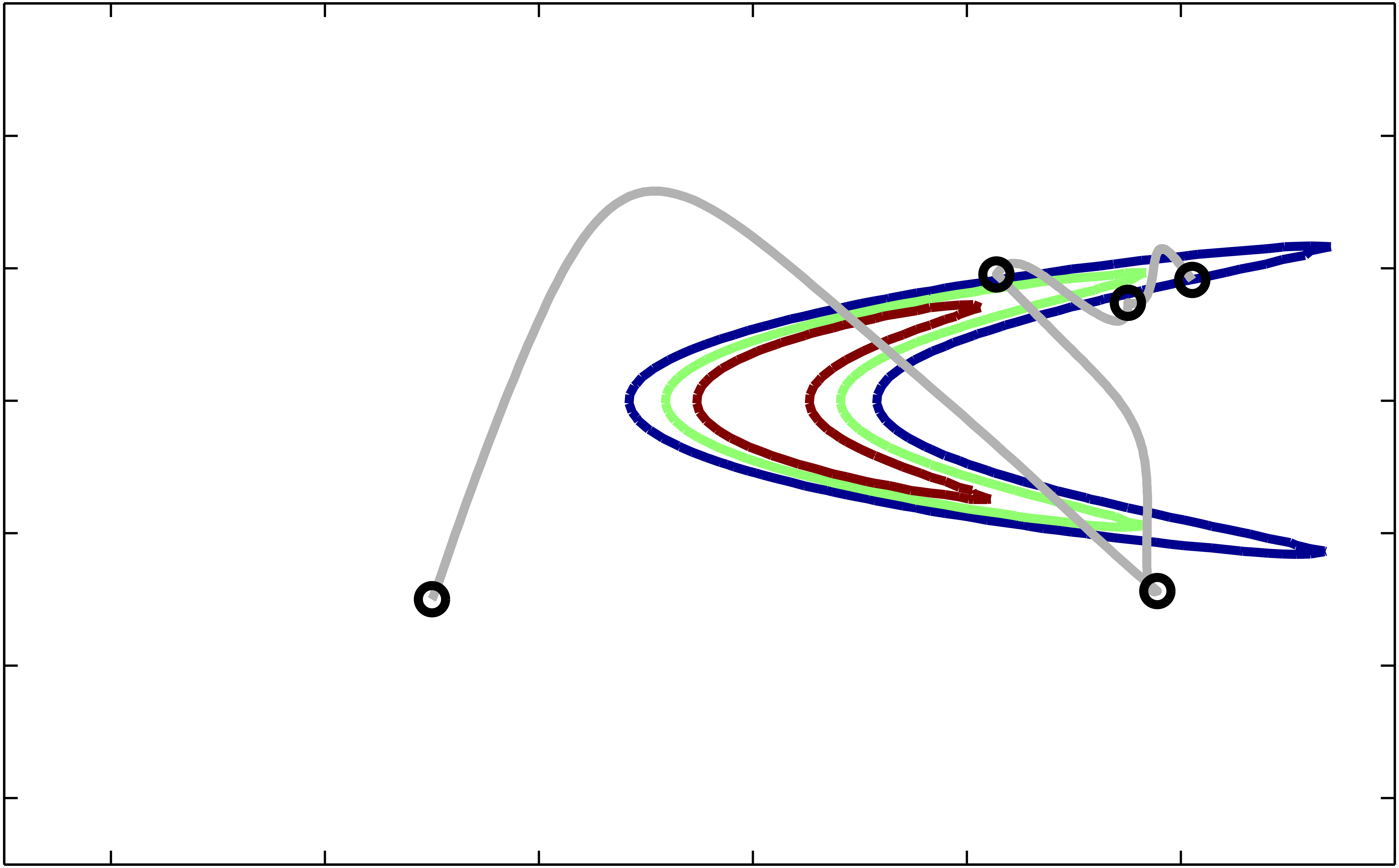
Alternating HMC



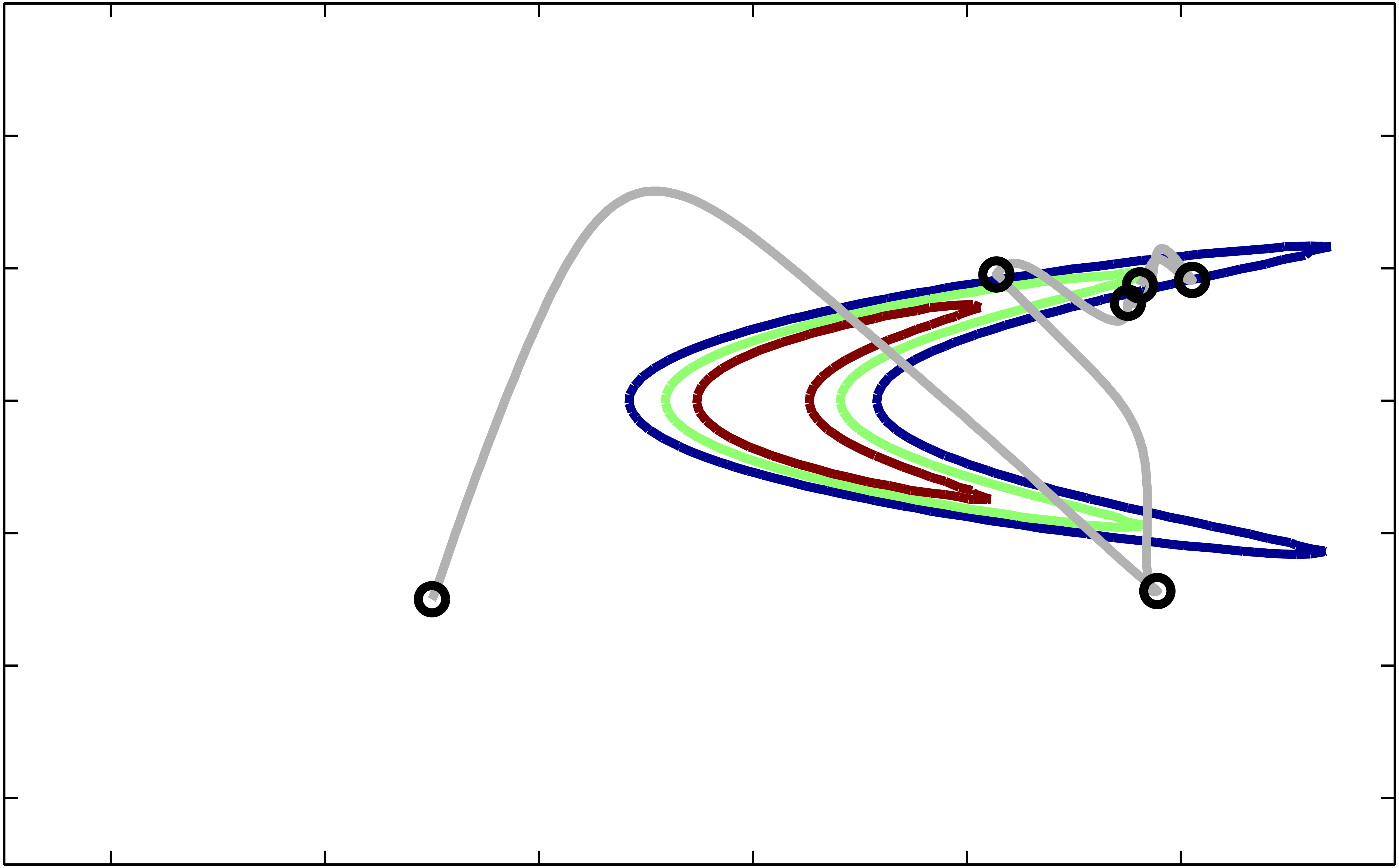
Alternating HMC



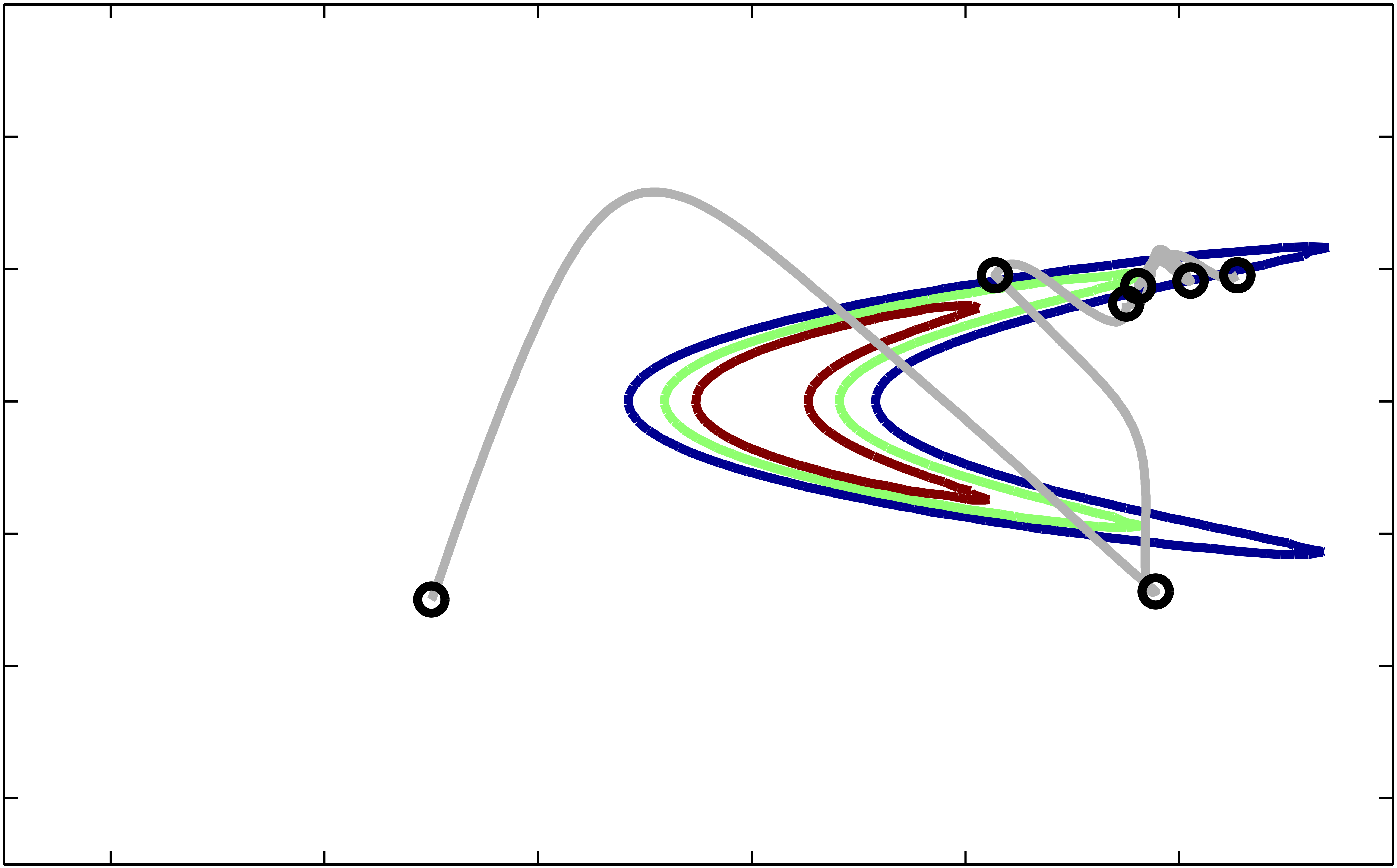
Alternating HMC



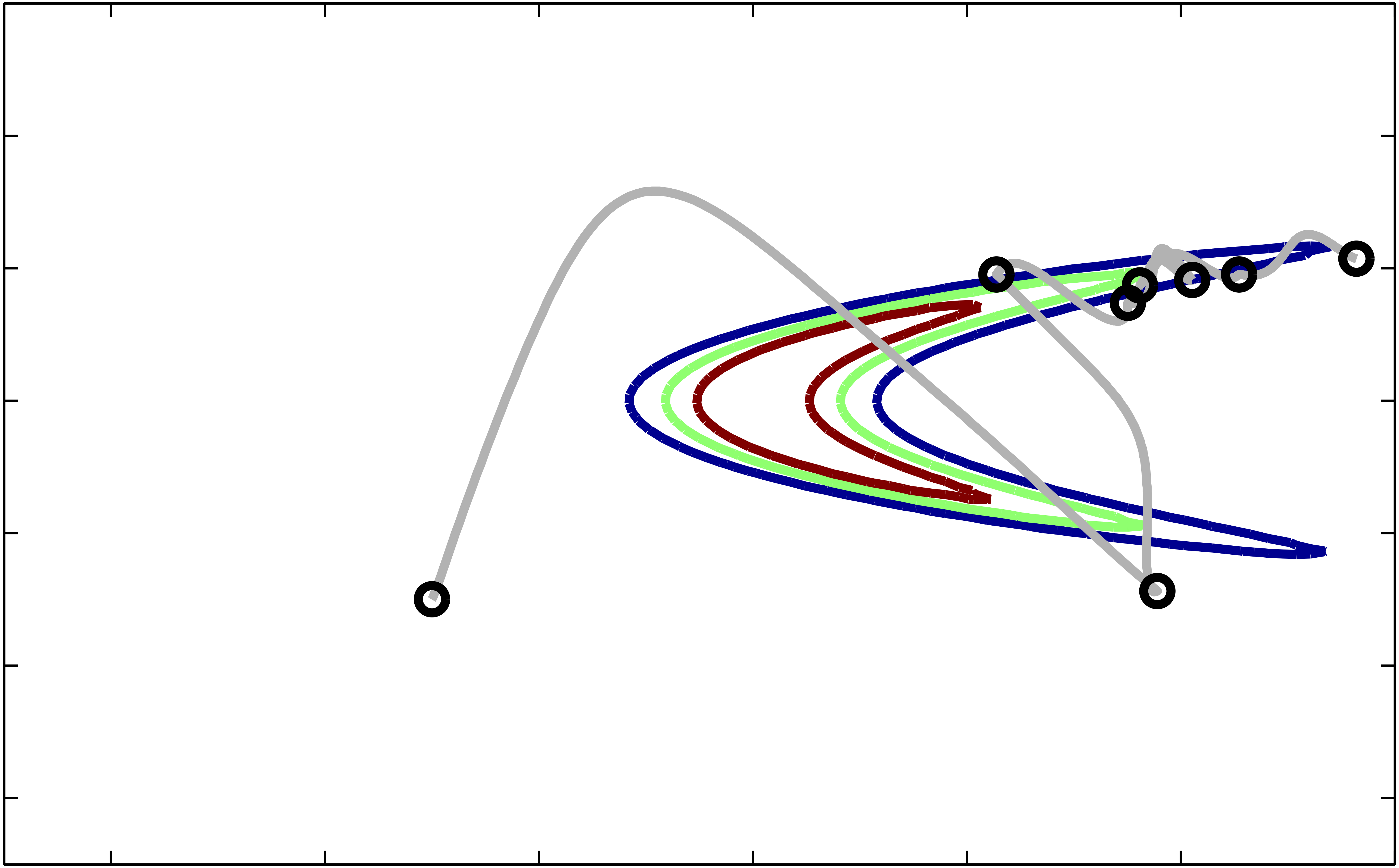
Alternating HMC



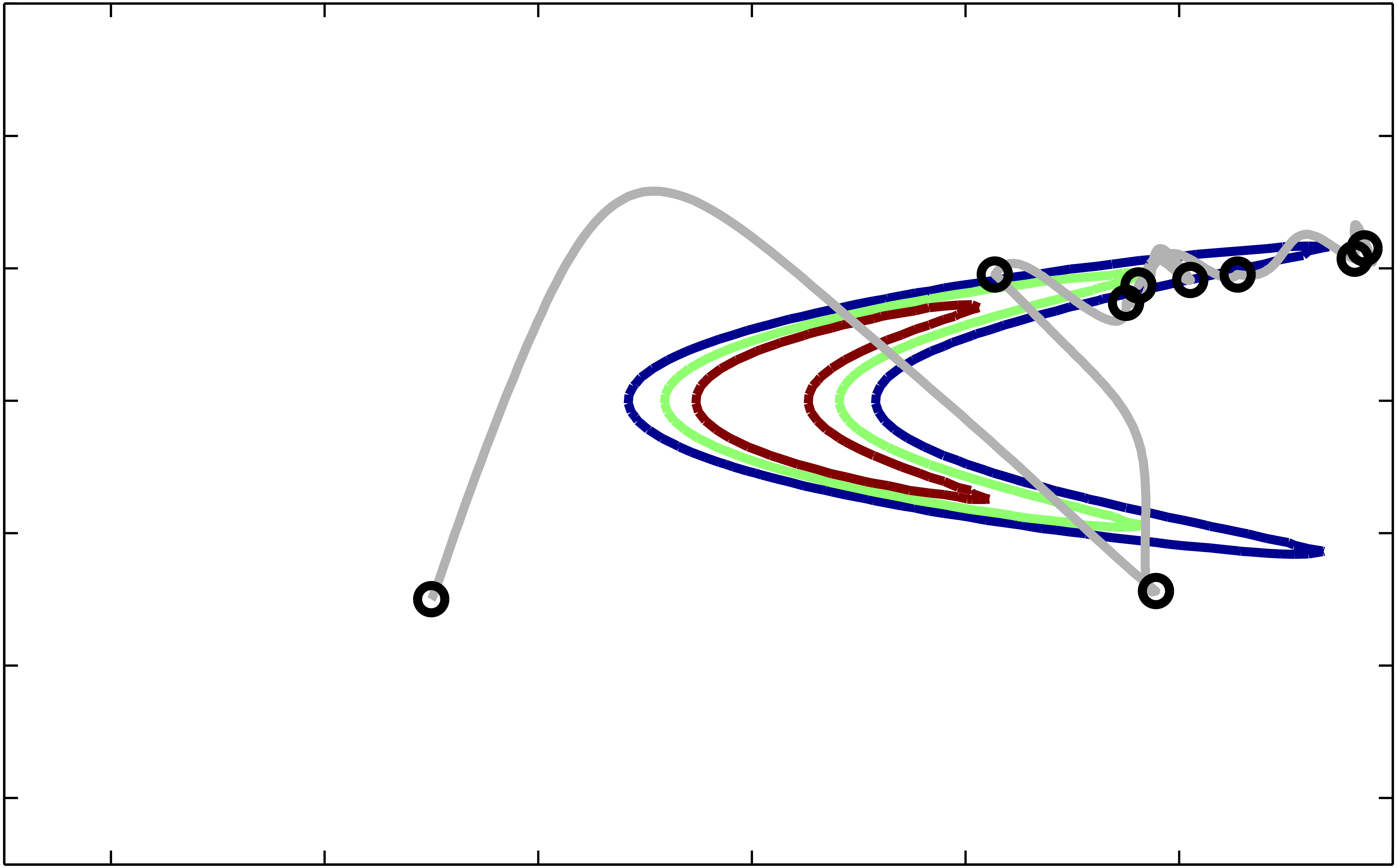
Alternating HMC



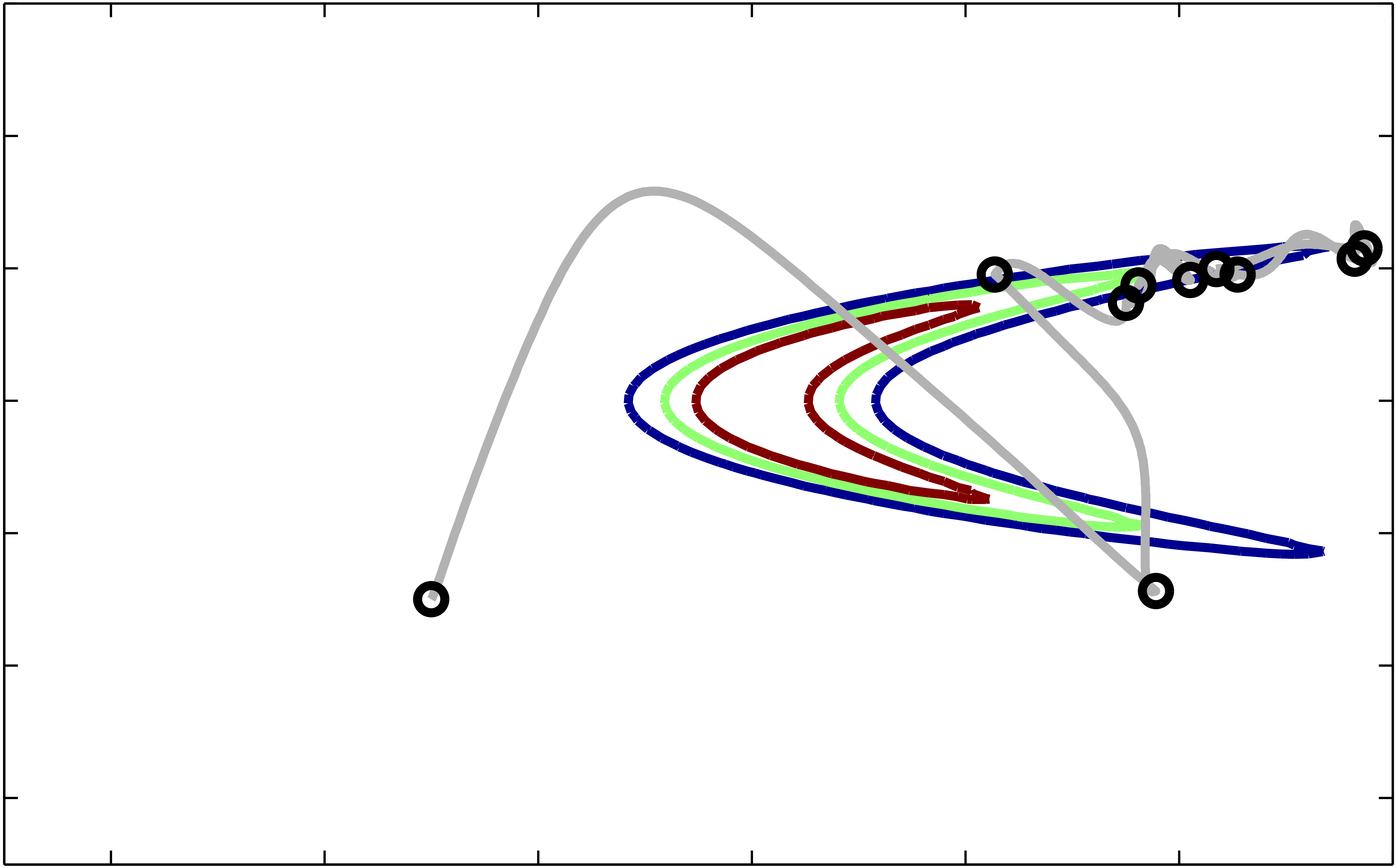
Alternating HMC



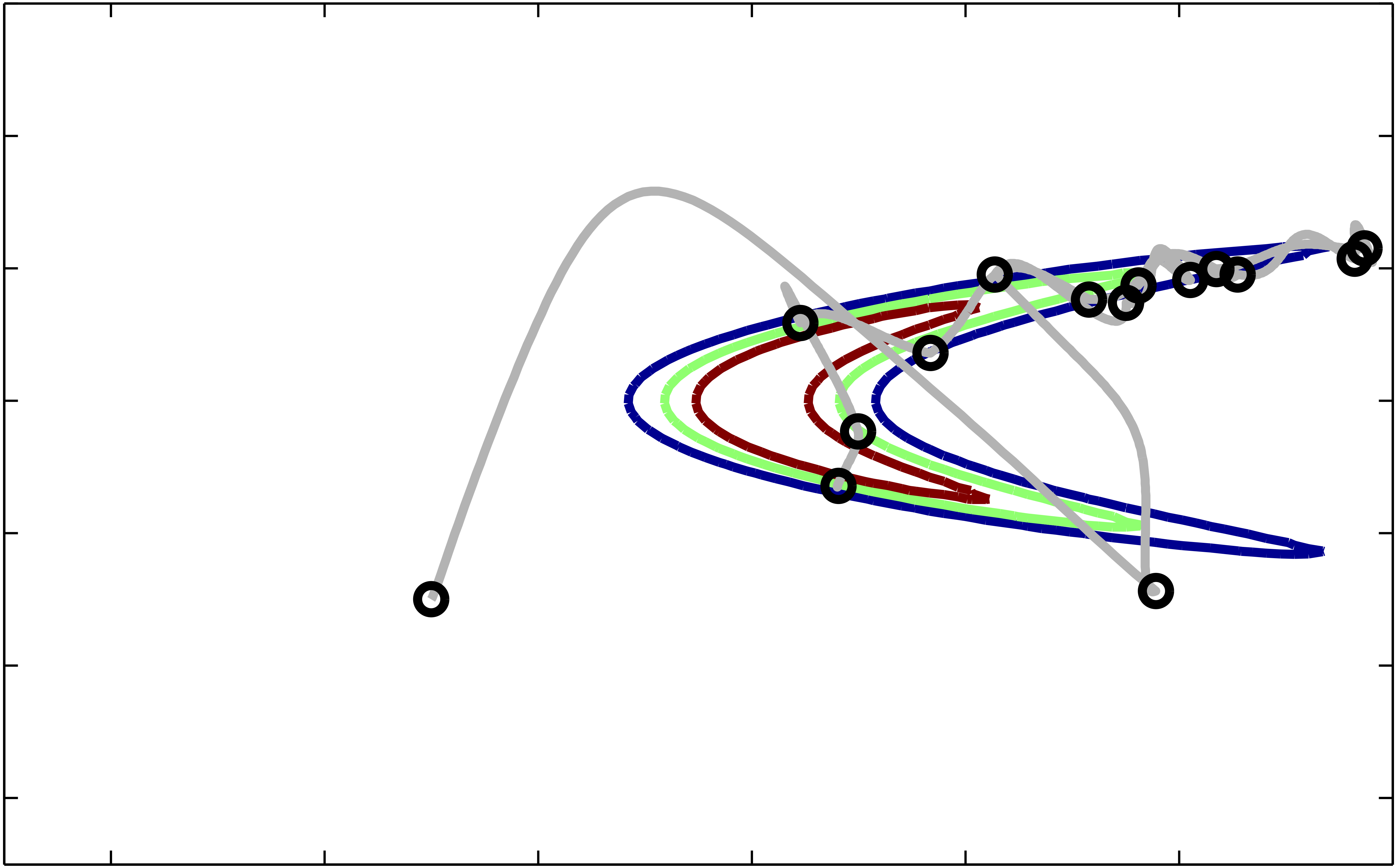
Alternating HMC



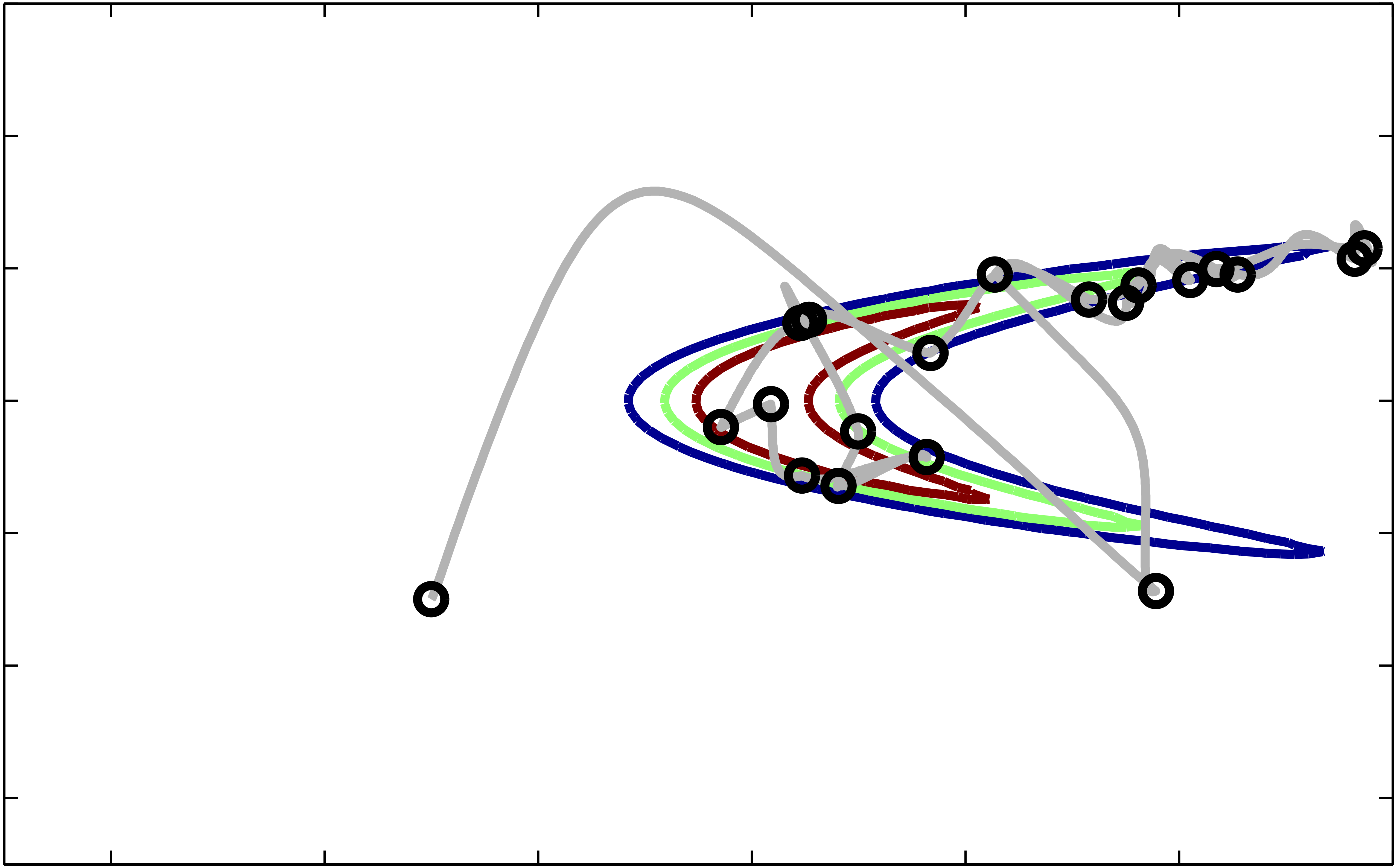
Alternating HMC



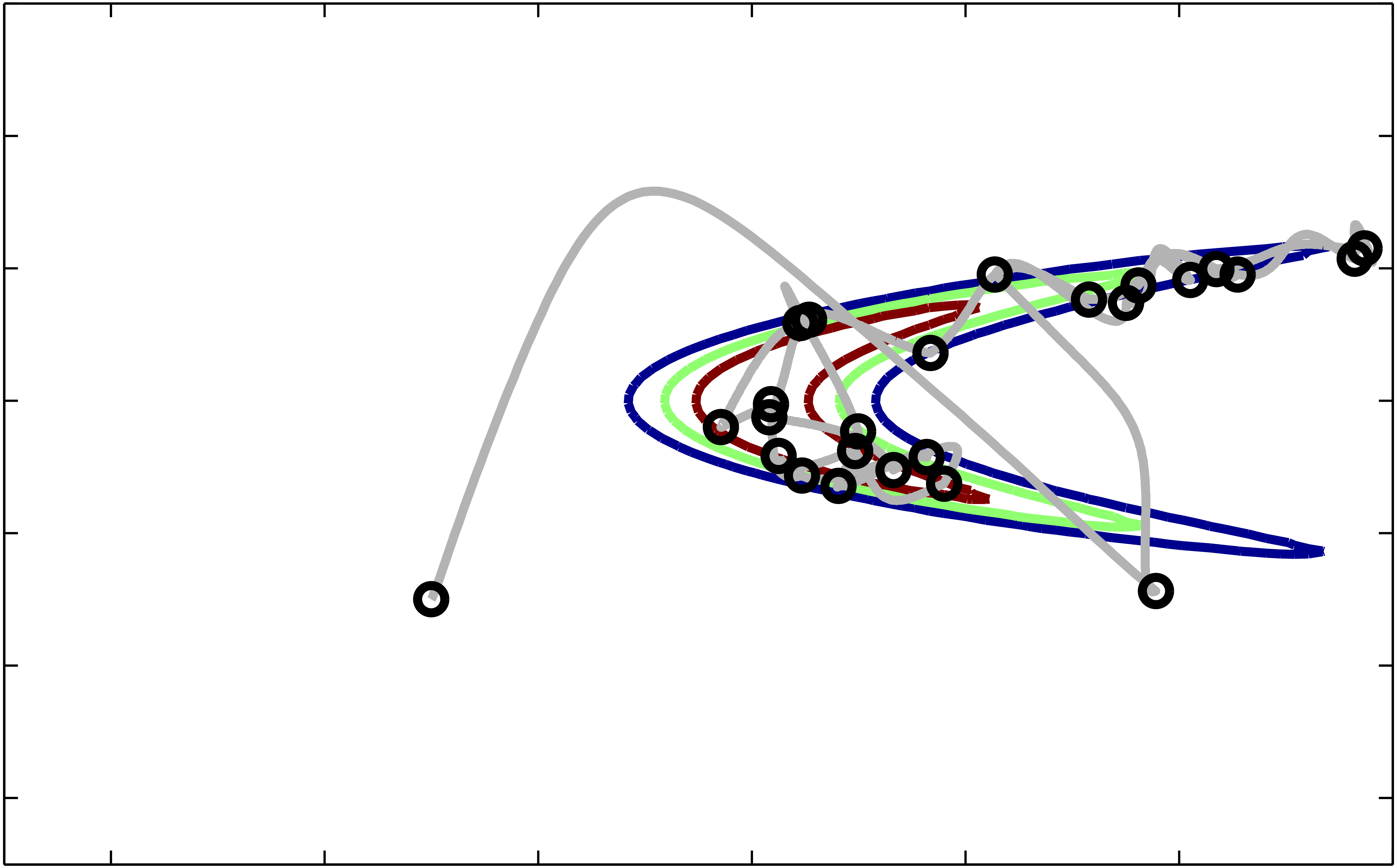
Alternating HMC



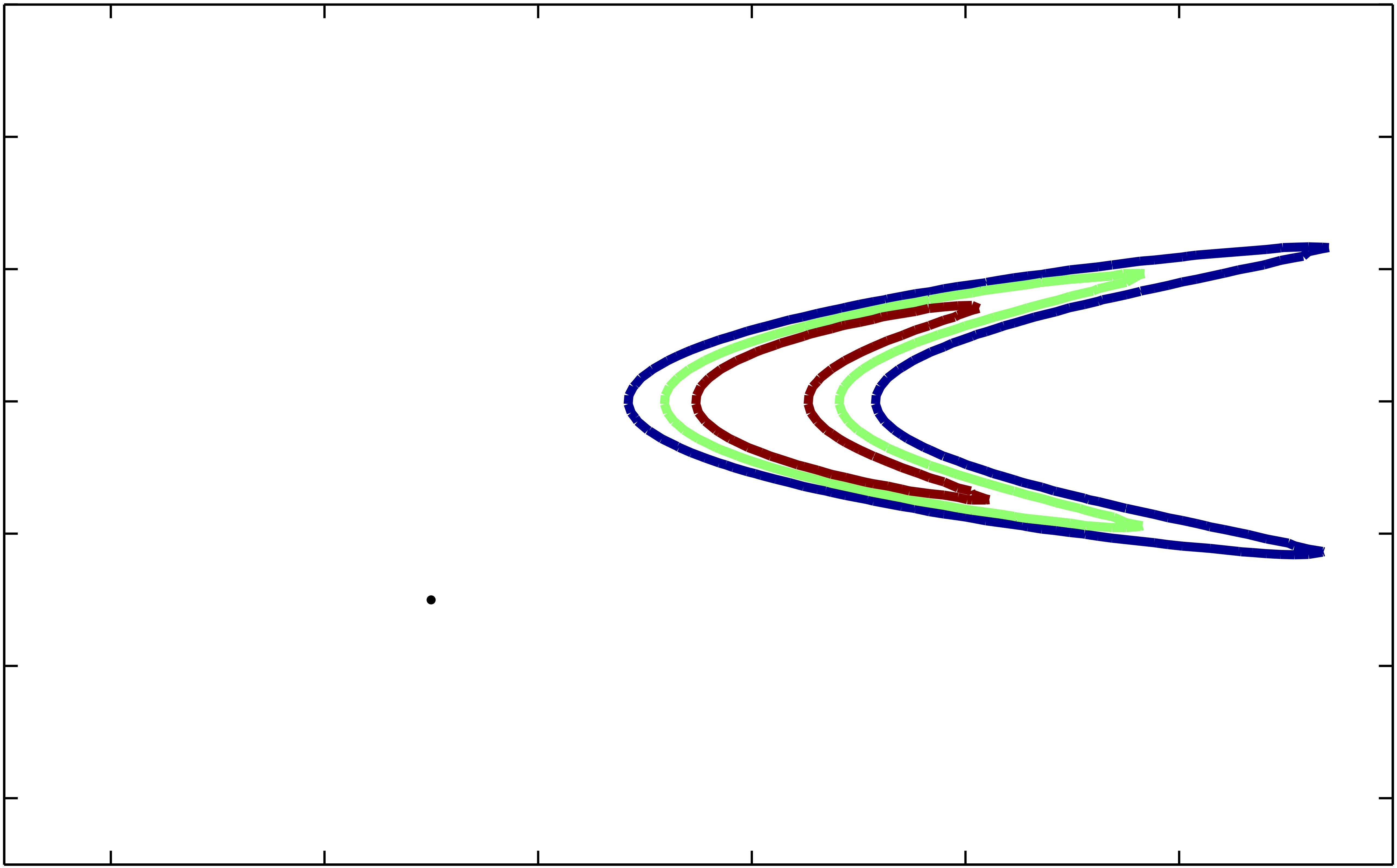
Alternating HMC



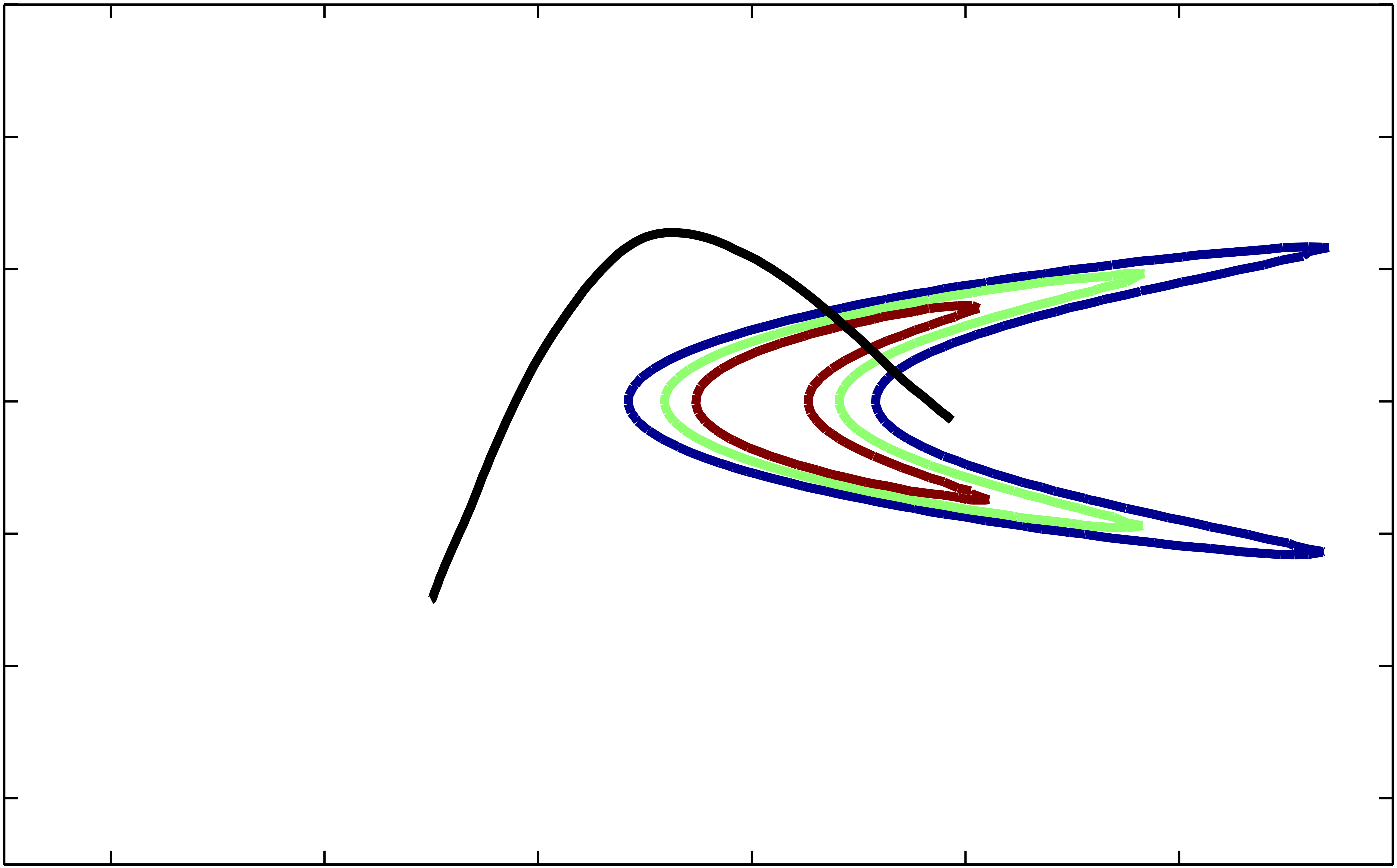
Alternating HMC



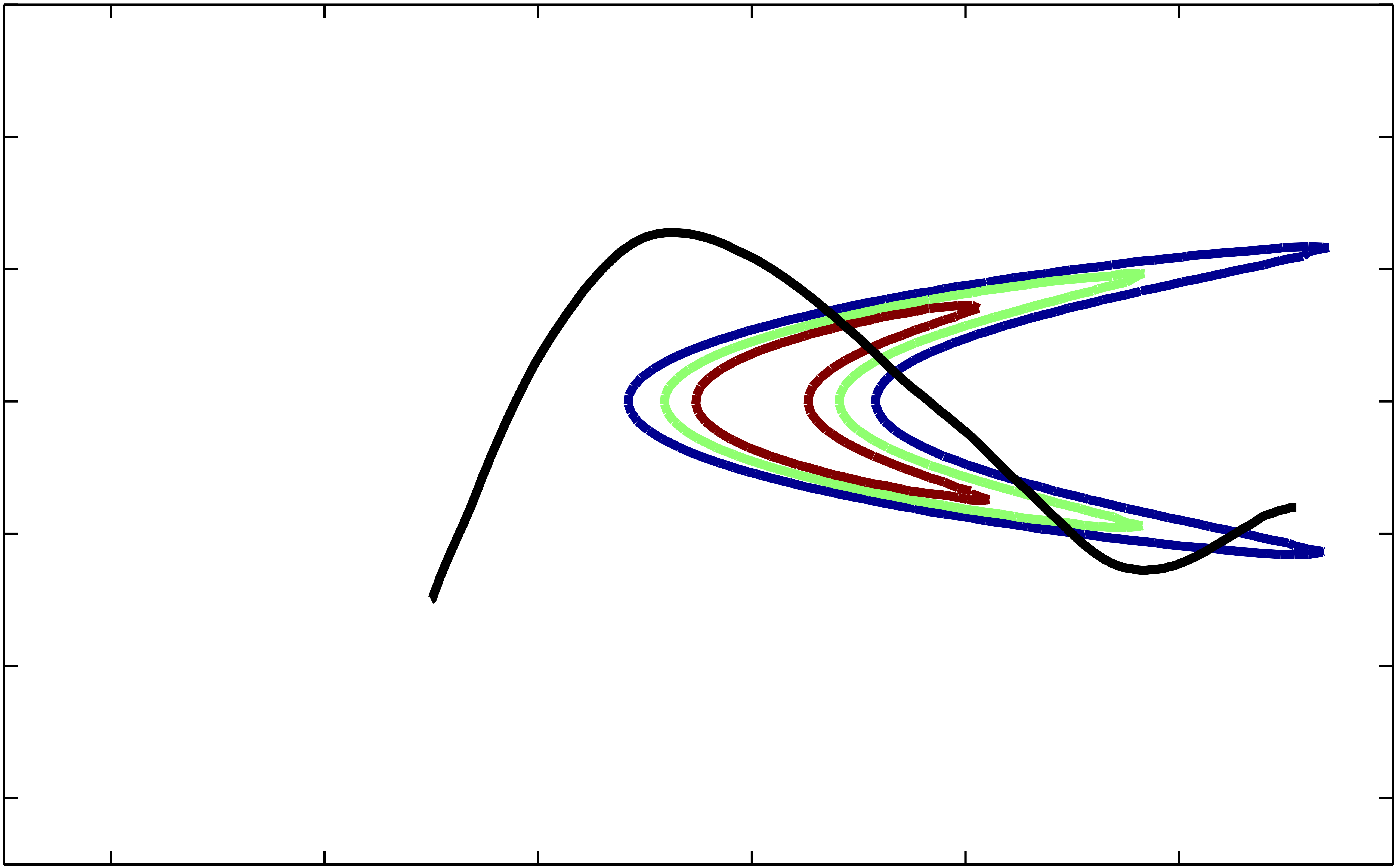
Perturbative HMC



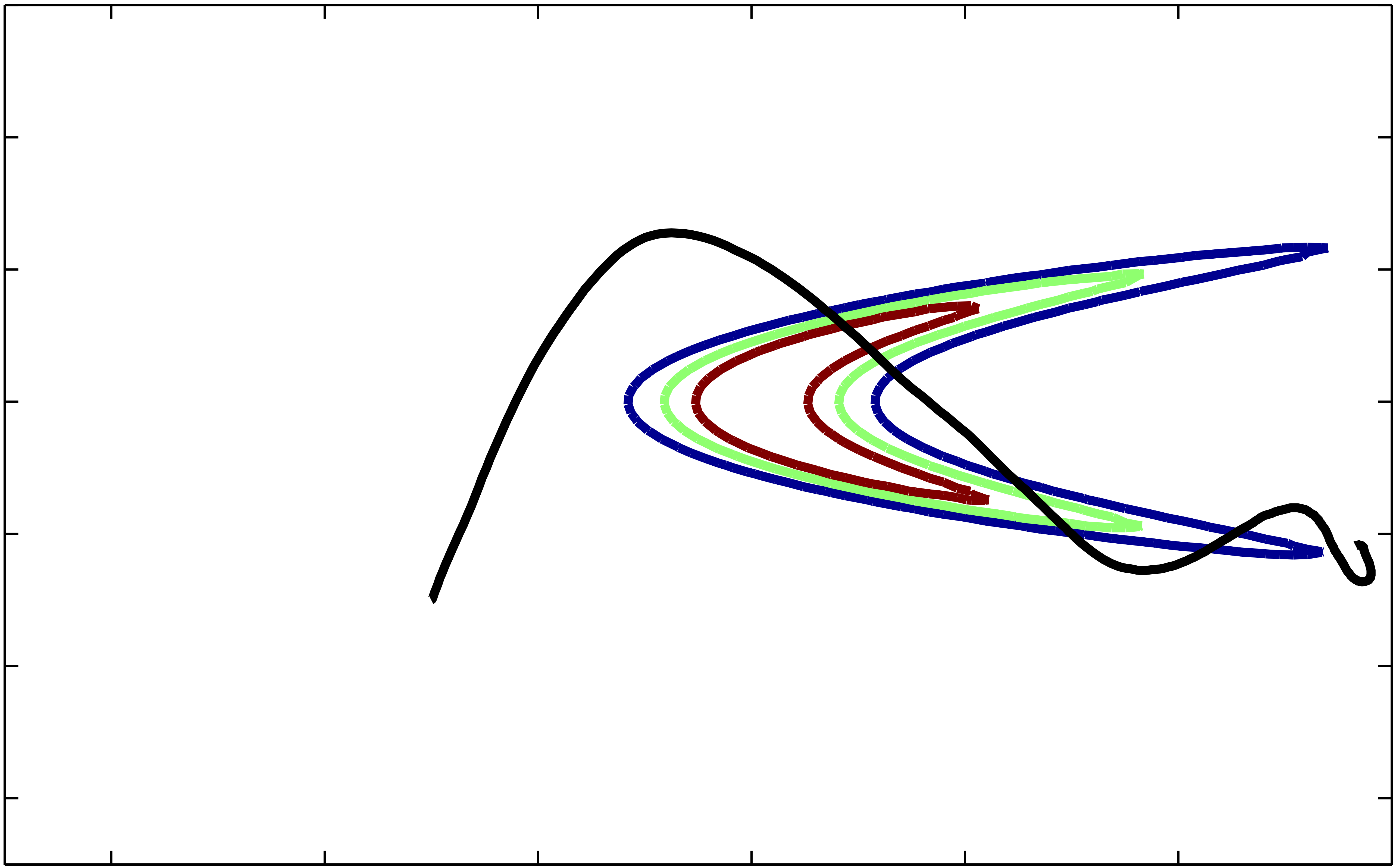
Perturbative HMC



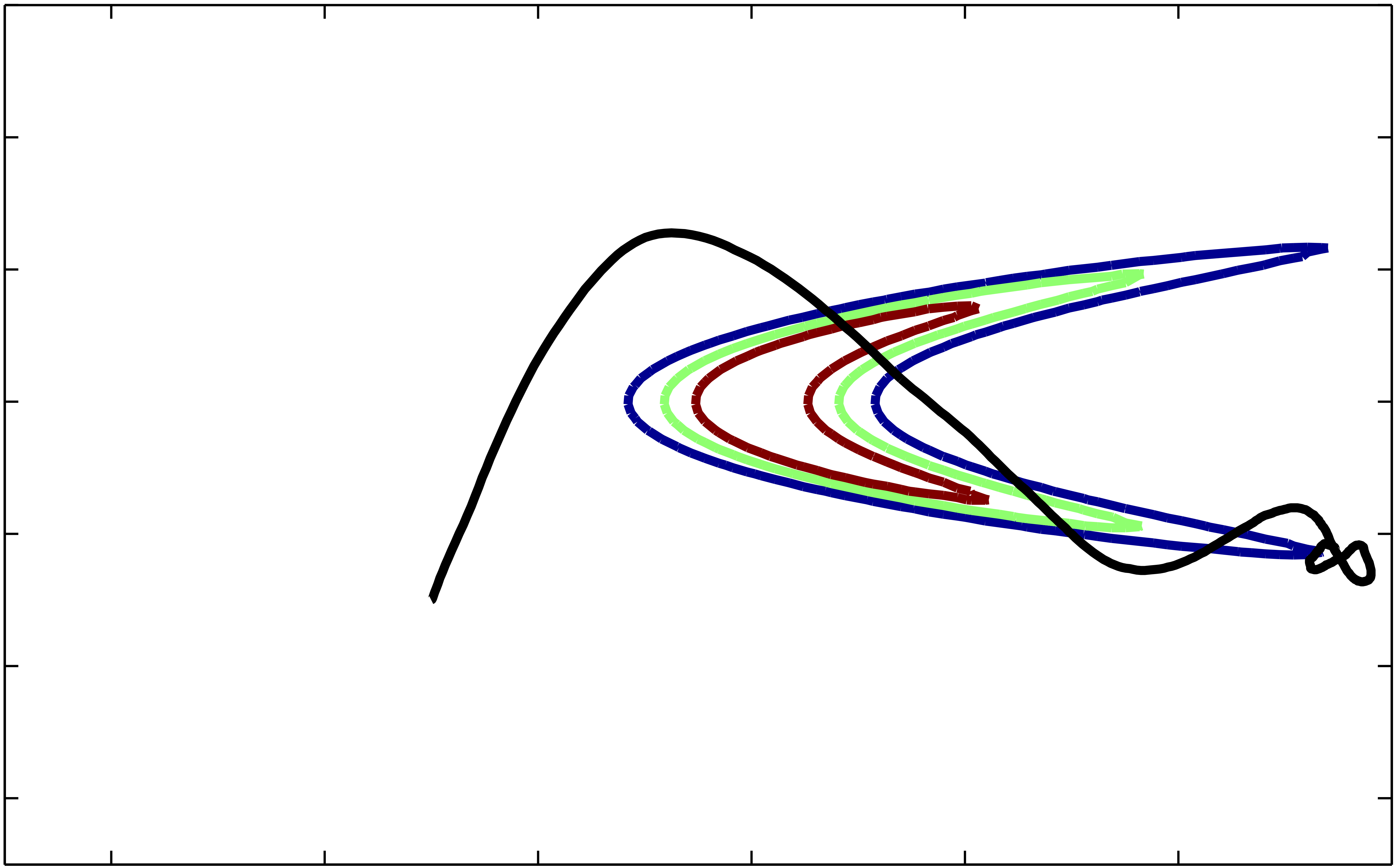
Perturbative HMC



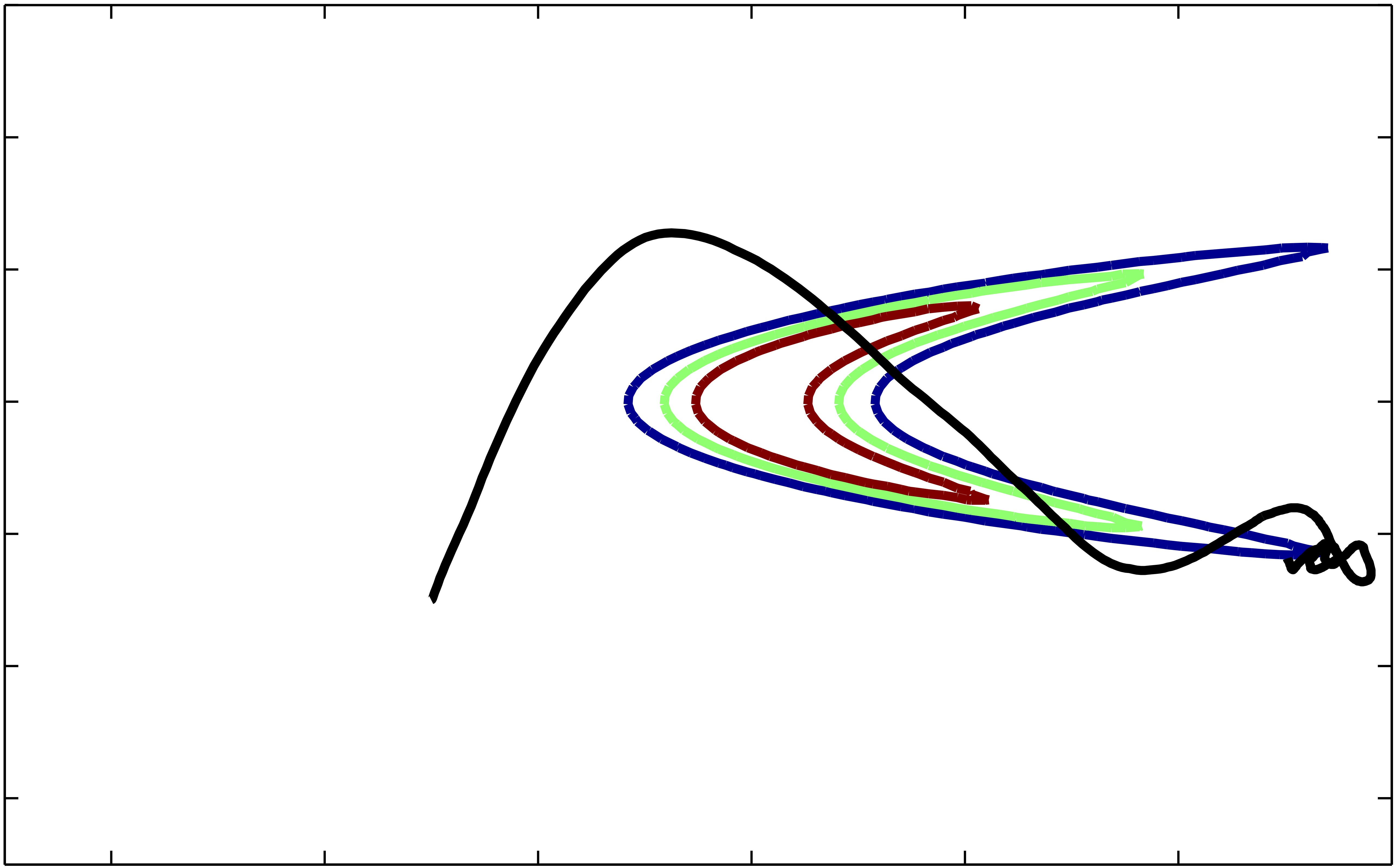
Perturbative HMC



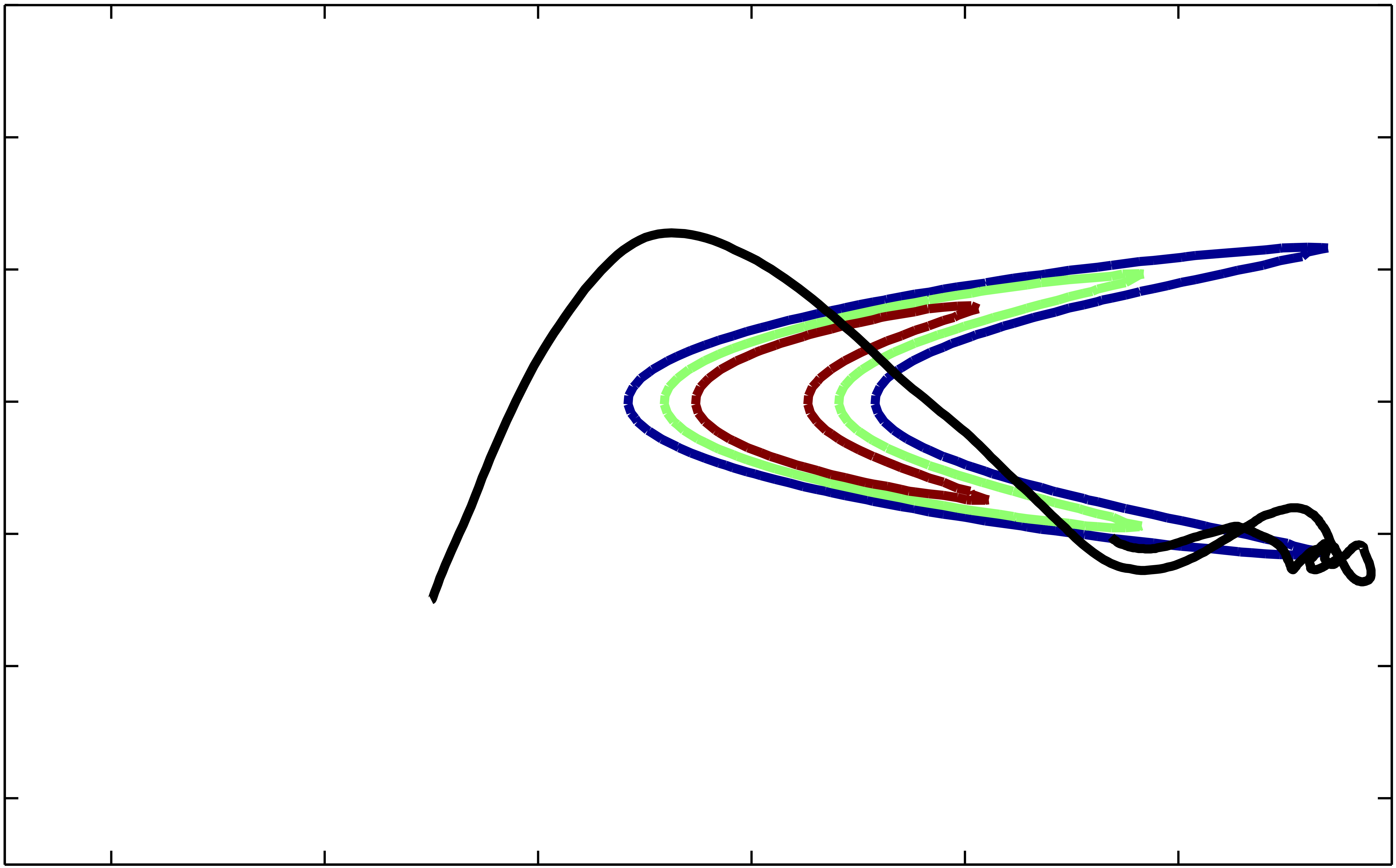
Perturbative HMC



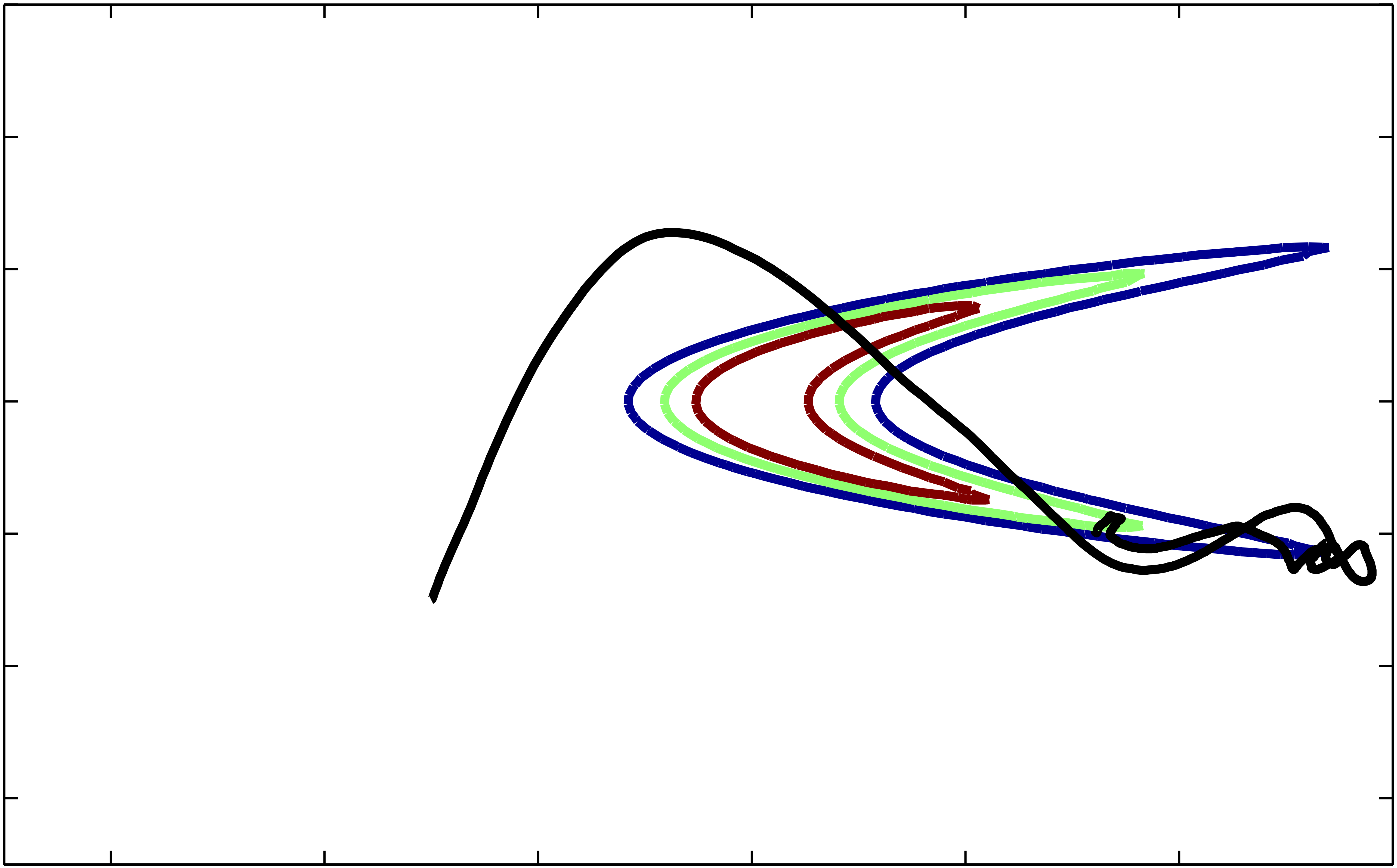
Perturbative HMC



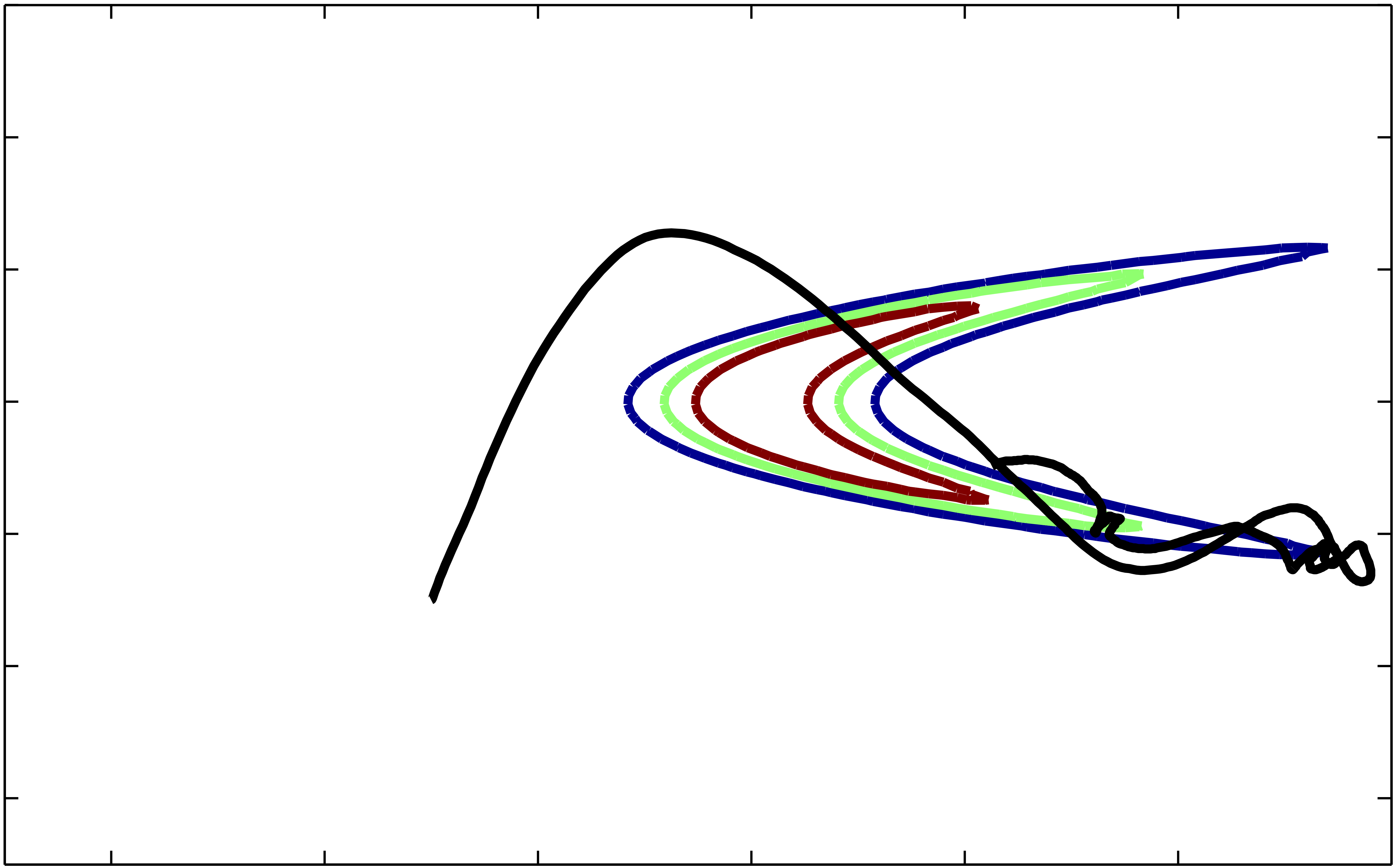
Perturbative HMC



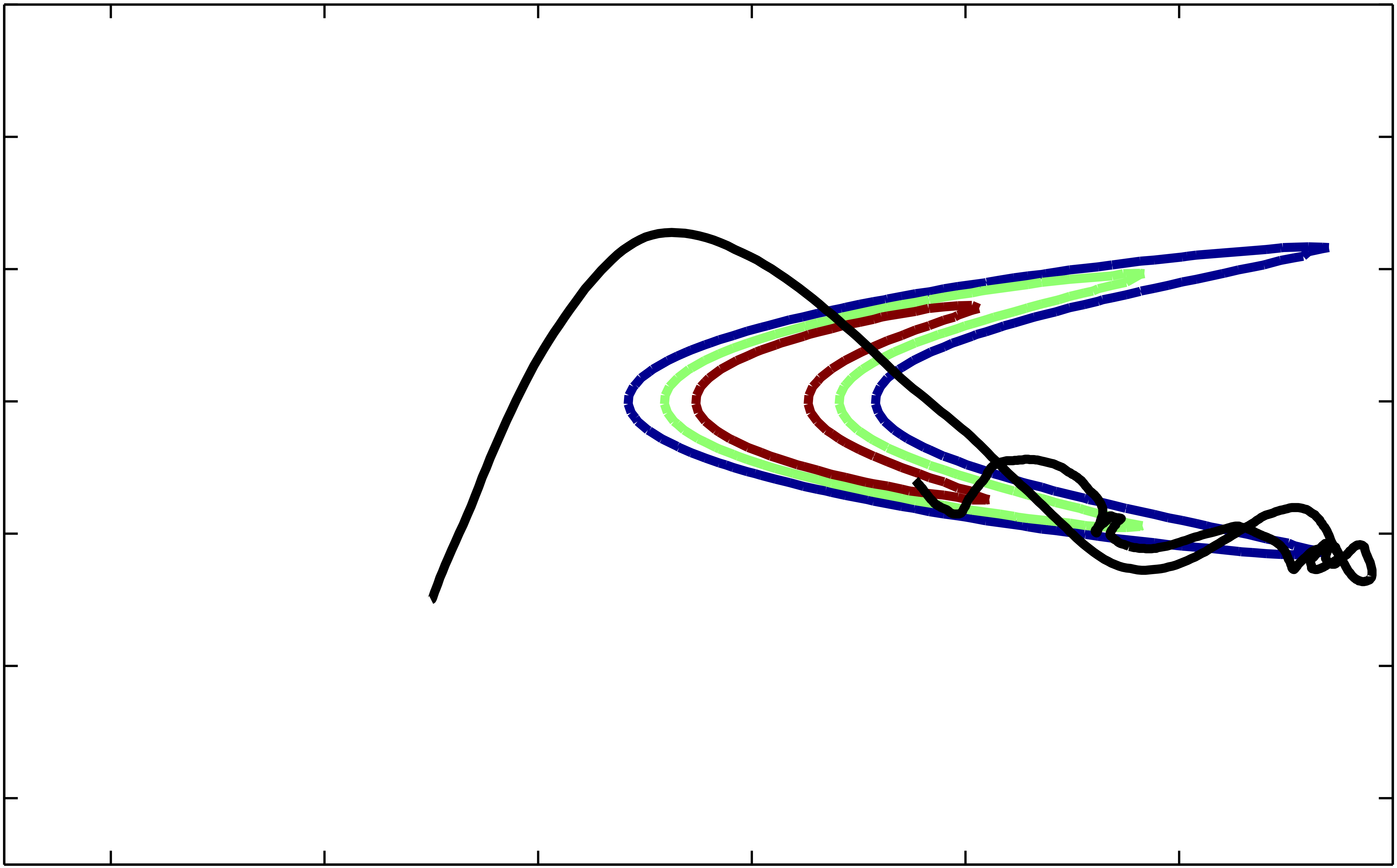
Perturbative HMC



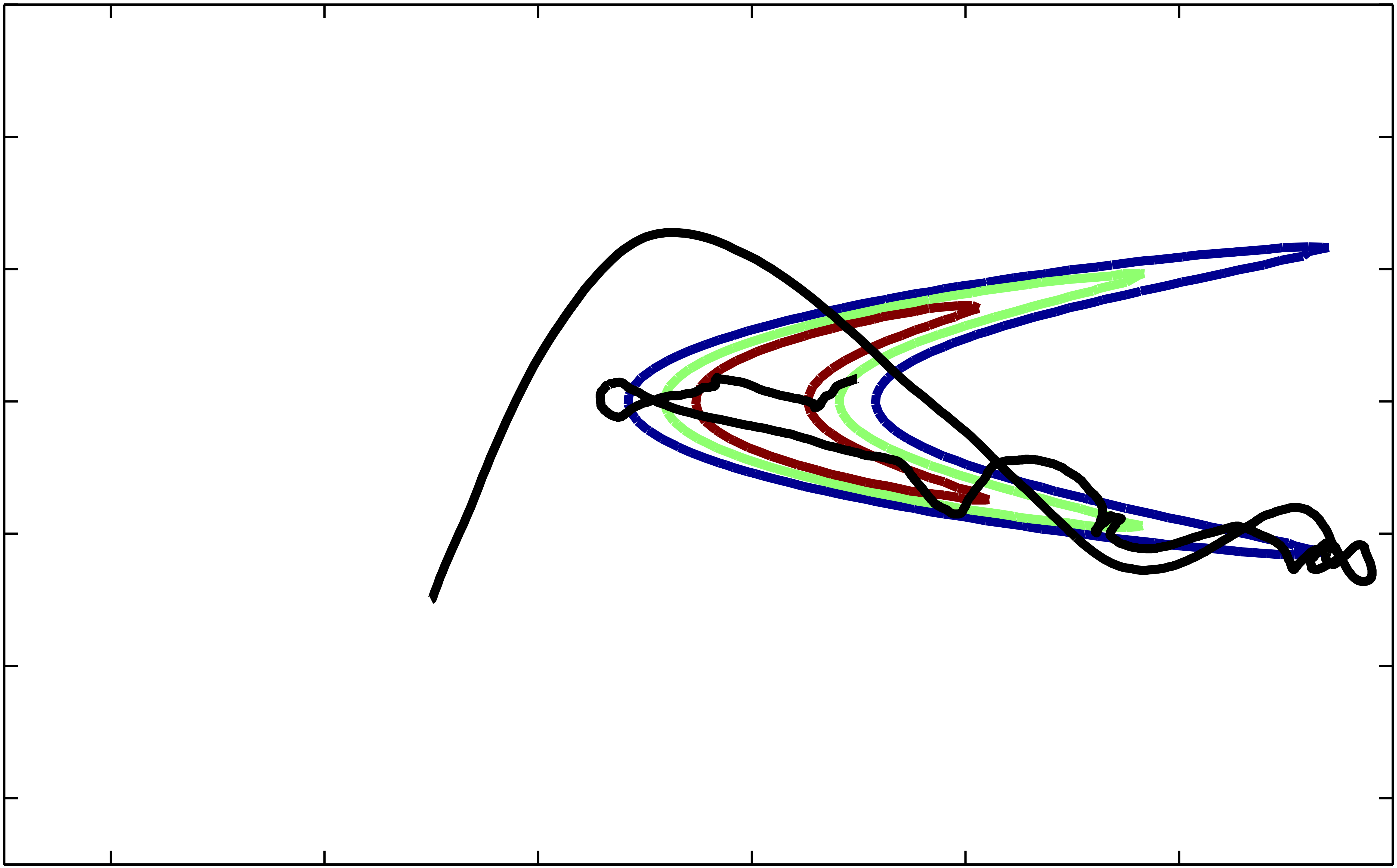
Perturbative HMC



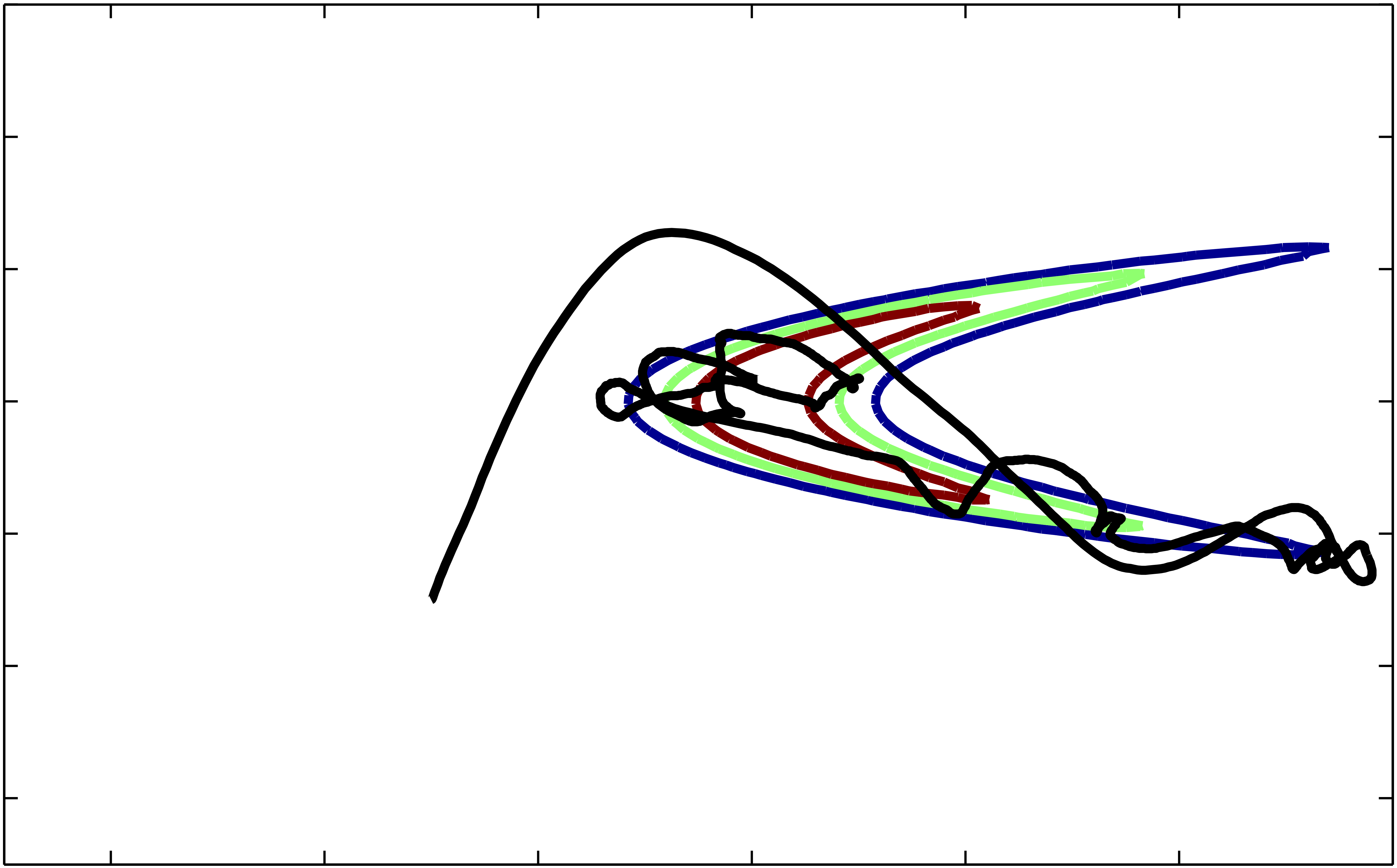
Perturbative HMC



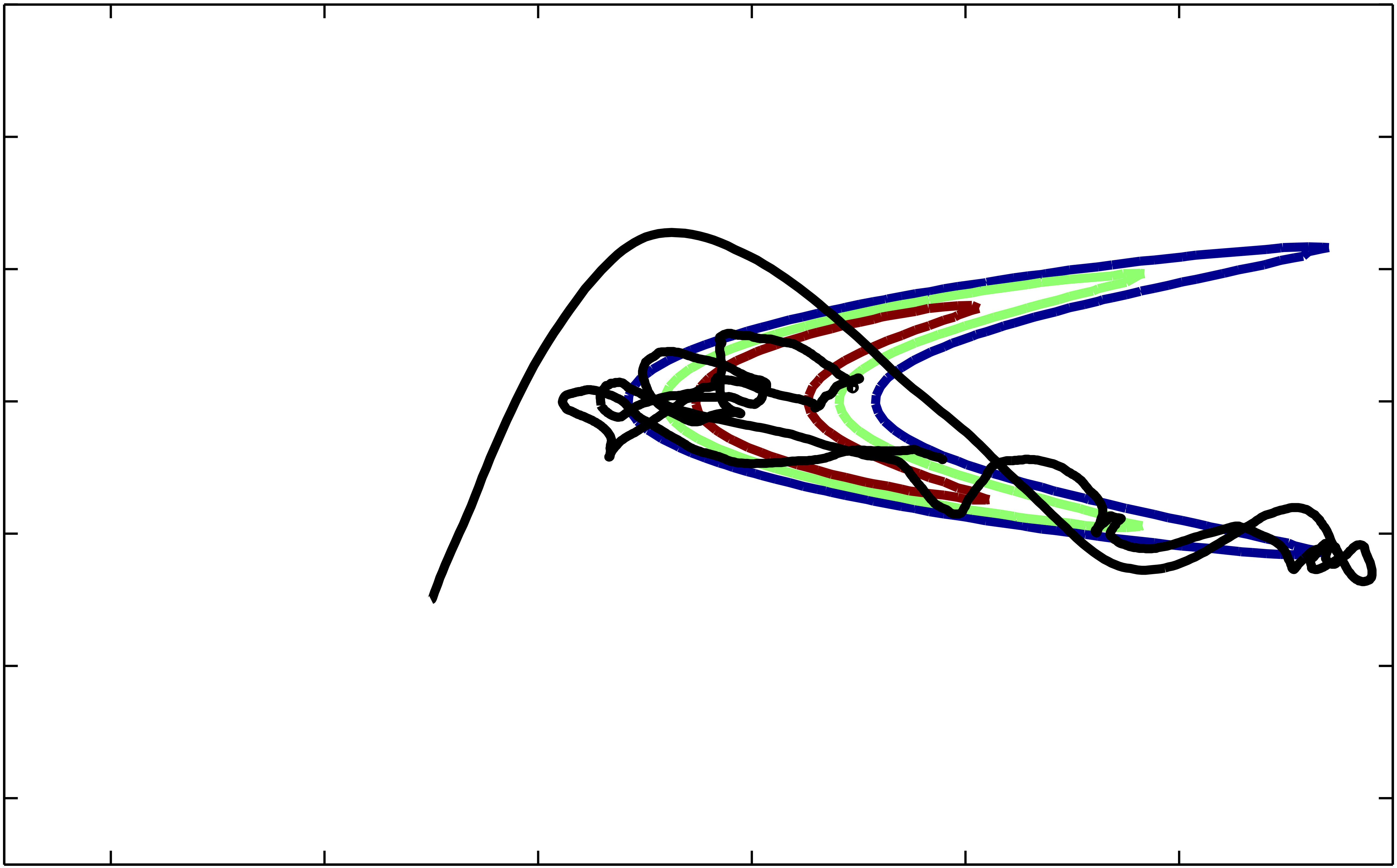
Perturbative HMC



Perturbative HMC



Perturbative HMC



HMC Leapfrog Integration

On a real computer, you can't actually simulate the true Hamiltonian dynamics, because you have to discretize.

To have a valid MCMC algorithm, the simulator needs to be reversible and satisfy the other requirements.

The easiest way to do this is with the "leapfrog method":

$$\rho_i(t+\epsilon/2) = \rho(t) - \frac{\epsilon}{2}\frac{\partial}{\partial x_i}E(x(t))$$

$$x_i(t+\epsilon) = x_i(t) + \epsilon\rho_i(t+\epsilon/2)$$

$$\rho_i(t+\epsilon) = \rho_i(t+\epsilon/2) - \frac{\epsilon}{2}\frac{\partial}{\partial x_i}E(x(t+\epsilon))$$

The Hamiltonian is not conserved, so you accept/reject via Metropolis-Hastings on the overall joint distribution.

HMC Practicalities

You have to decide:

1) How large the simulation steps should be.

2) Either: a) How many simulation steps to take between randomizing momenta, or b) How much to perturb the momenta after each step.

Tuning these quantities is difficult. You want the trajectories to be of the same order as the longest dimension of the distribution, but you need to discretize finely enough that you don't reject too often.

In general, the optimal acceptance rate will be higher than that for vanilla Metropolis-Hastings.

HMC Practicalities

My strategies for implementing HMC:

1) Try slice sampling first.

2) Get the gradients right with finite differences!

3) Tune the trajectory length by tracking distances.

4) Set step size for 80-90% acceptance rate.

5) Actually draw the number and size of steps from distributions centred on the results from tuning.

6) Mix in some slice sampling moves for good measure.

Your mileage may vary.

HMC Practicalities

Radford M. Neal. "MCMC using Hamiltonian Dynamics". To appear in *Handbook of Markov chain Monte Carlo*. 2010.

http://www.cs.toronto.edu/~radford/ham-mcmc.abstract.html

Advanced MCMC, Summarized

Slice Sampling

Hamiltonian Monte Carlo

Overall Summary

Monte Carlo allows you to estimate integrals that may be impossible for deterministic numerical methods.

Sampling from arbitrary distributions can be done pretty easily in low dimensions.

MCMC allows us to generate samples in high dimensions.

Metropolis-Hastings and Gibbs sampling are popular, but you should probably consider slice sampling instead.

If you have a difficult high-dimensional problem, Hamiltonian Monte Carlo may be for you.

Keywords to Google

If you want to ...

... prove that your MCMC has converged? “perfect simulation”

... get better estimates? “Rao-Blackwellization”

... estimate partition functions? “annealed importance sampling”

... sample from doubly-intractable models? “exchange sampling”

... importance sample from time series? “particle filtering”

... alter model dimensionality? “reversible jump Monte Carlo”

... do faster sampling in MRFs? “Swendsen-Wang”

... run MCMC on Gaussian processes? “elliptical slice sampling"

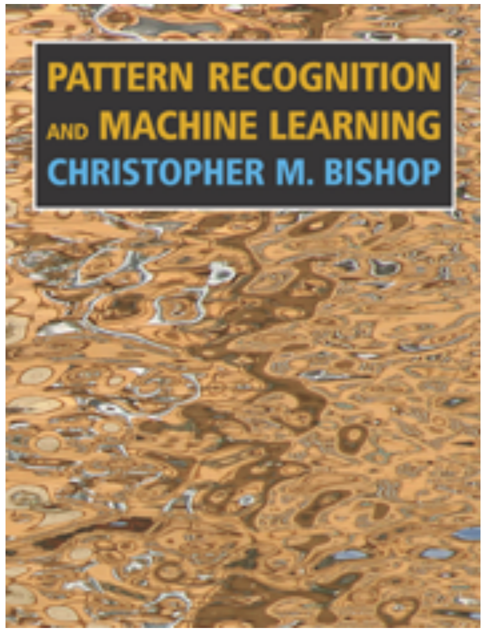
General References

Radford Neal's Review Article

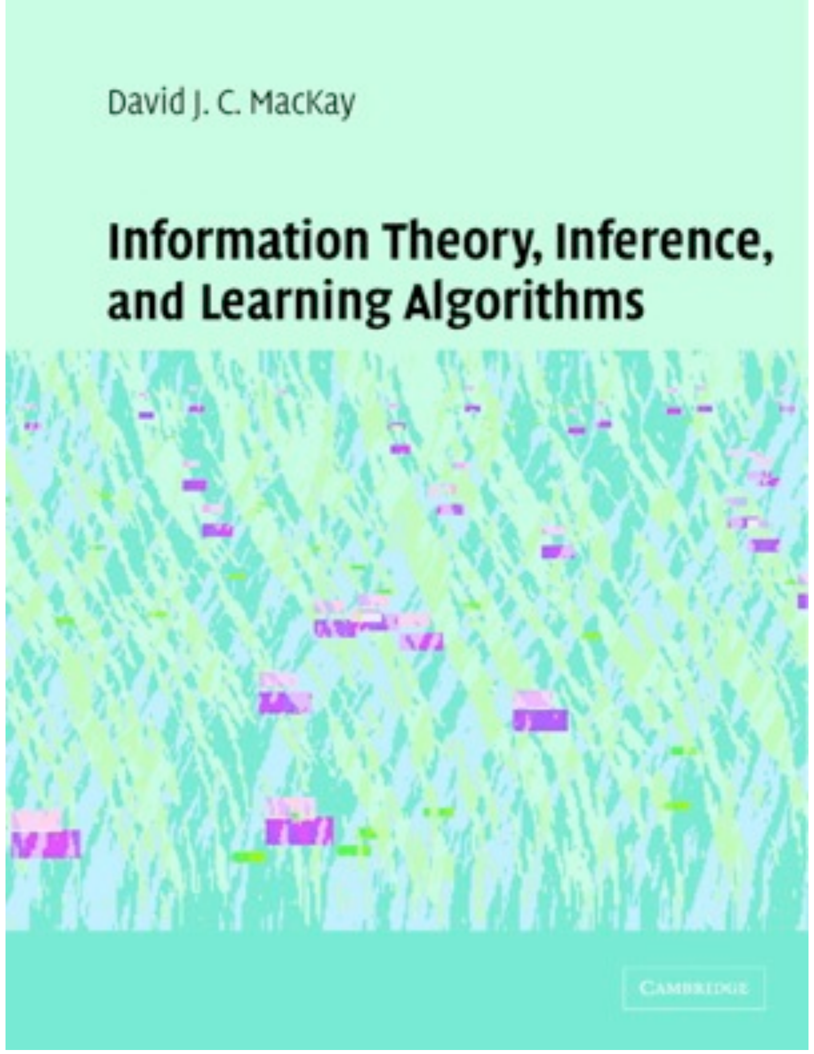
http://www.cs.toronto.edu/~radford/review.abstract.html

Monte Carlo Workshop at NIPS 2010

http://montecarlo.wikidot.com



Chris Bishop's Book



David MacKay's Book