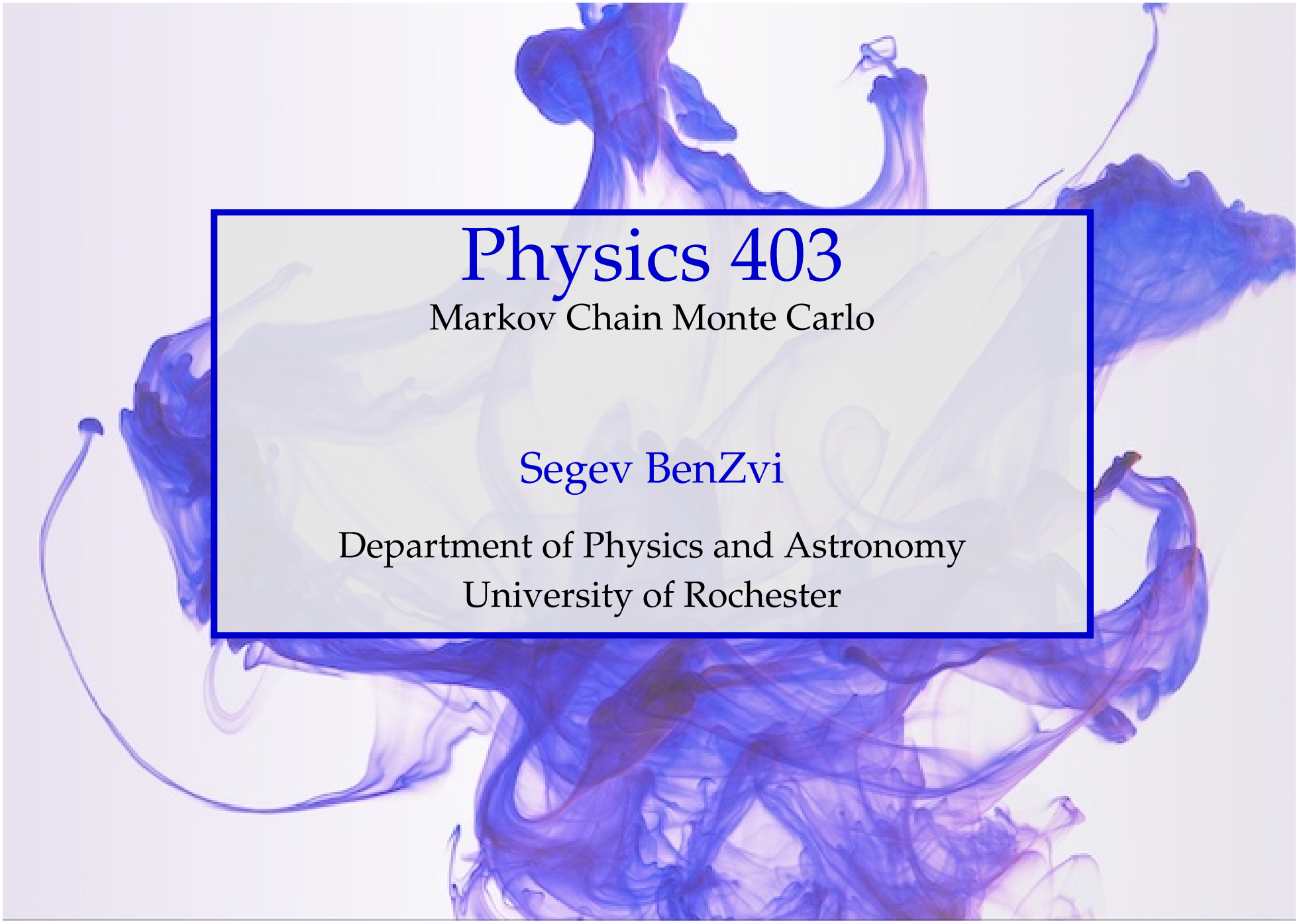


Physics 403

Markov Chain Monte Carlo

Segev BenZvi

Department of Physics and Astronomy
University of Rochester

Reading

- D. MacKey, *Information Theory, Inference, and Learning Algorithms* (Cambridge UP, 2003), Ch. 29.
  Available as an online PDF
- P. Gregory, *Bayesian Logical Data Analysis* (Cambridge UP, 2005), Ch. 12.
  Available in POA

Table of Contents

1. Sampling from PDFs in Many Dimensions
   - Markov Chain Monte Carlo
   - The Metropolis-Hastings Algorithm
2. Case Study: Sampling from a 1D Distribution
   - Burn-In
   - Autocorrelation
   - Efficiency and Detailed Balance
3. Sampling from a Joint Distribution
   - Parallel Tempering

Bayesian Calculations in Many Dimensions

- Imagine that we have a probability distribution for a set of parameters $\boldsymbol{\theta}$ given by $p(\boldsymbol{\theta}|D,I)$
- Often we have to marginalize over nuisance parameters $\boldsymbol{\nu}$, where the $\{\nu\}$ are uninteresting but necessary to complete the calculation:

$$p(\boldsymbol{\theta}|D,I) = \int d\boldsymbol{\nu}\; p(\boldsymbol{\theta},\boldsymbol{\nu}|D,I)$$

- If the set $\{\nu\}$ is large this integral can become very expensive
- **Recall**: we can integrate numerically using Monte Carlo sampling, but we waste time in regions of low probability

Example

If we spend a fraction 10^{-1} of our time in a region of high probability for nuisance parameter, then for m parameters the fraction falls to 10^{-m}

Markov Chain Monte Carlo

- The goal of Markov Chain Monte Carlo (MCMC) algorithms is to draw samples from the PDF

$$p(\boldsymbol{\theta}, \boldsymbol{\nu}|D, I) = \frac{1}{Z} p(D|\boldsymbol{\theta}, \boldsymbol{\nu}, I)\, p(\boldsymbol{\theta}, \boldsymbol{\nu}|I)$$

  where $Z = p(D|I)$ is the marginal evidence
- Since Z is independent of $\boldsymbol{\theta}$ and $\boldsymbol{\nu}$ we usually don't have to calculate it... which is good because it's expensive
- Once the samples produced by MCMC are available, the expectation value of a function of the model parameters $f(\boldsymbol{x})$ is

$$\langle f(\boldsymbol{\theta}) \rangle = \int p(\boldsymbol{\theta}|D, I)\, f(\boldsymbol{\theta})\, d\boldsymbol{\theta} \approx \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{x}_i)$$

- In MCMC, we randomly walk over positions $\boldsymbol{x}$ in the parameter space and draw samples $\boldsymbol{x}(t_i) = [\boldsymbol{\theta}_i, \boldsymbol{\nu}_i]$ from the distribution

Metropolis-Hastings Algorithm

At each point in a Markov chain, $\boldsymbol{x}(t_i)$depends only on the previous step $\boldsymbol{x}(t_{i-1})$ according to the transition probability $q(\boldsymbol{x}(t+1)|\boldsymbol{x}(t))$

The simplest MCMC algorithm is the Metropolis-Hastings method [1], which proceeds in two steps:

1. Given $\boldsymbol{x}(t)$ sample a proposal position $\boldsymbol{y}$ from $q(\boldsymbol{y}|\boldsymbol{x}(t))$
2. Accept this proposal with probability

$$\alpha(\boldsymbol{x}(t),\boldsymbol{y}) = \min(1,r) = \min\left(1, \frac{p(\boldsymbol{y}|D,I)}{p(\boldsymbol{x}(t)|D,I)}\frac{q(\boldsymbol{x}(t)|\boldsymbol{y})}{q(\boldsymbol{y}|\boldsymbol{x}(t))}\right)$$

In practice, what you do is:

1. Initialize $\boldsymbol{x}(0)$, set $t = 0$
2. Sample $\boldsymbol{y}$ from $q(\boldsymbol{y}|\boldsymbol{x}(t))$ and $u \sim \text{Uniform}(0,1)$
3. If $u \leq r$ then $\boldsymbol{x}(t+1) \to \boldsymbol{y}$; otherwise, $\boldsymbol{x}(t+1) \to \boldsymbol{x}(t)$

Properties Required of the Markov Chain

The Metropolis-Hastings algorithm samples x with a probability density that converges to $p(x|D,I)$, called the stationary distribution of the Markov Chain. For this to occur, the Markov chain must have these properties:

1. **Irreducibility**: from all starting points, the Markov Chain must be able to jump to all states in the target distribution with positive probability.
2. **Aperiodicity**: the chain does not oscillate between different states in a regular, periodic way.
3. **Positive Recurrence**: if an initial value x_0 is sampled from $p(x|D,I)$, then all subsequent iterates will be distributed according to $p(x|D,I)$.

Table of Contents

1. Sampling from PDFs in Many Dimensions
   - Markov Chain Monte Carlo
   - The Metropolis-Hastings Algorithm
2. Case Study: Sampling from a 1D Distribution
   - Burn-In
   - Autocorrelation
   - Efficiency and Detailed Balance
3. Sampling from a Joint Distribution
   - Parallel Tempering

Sampling from a Poisson Distribution

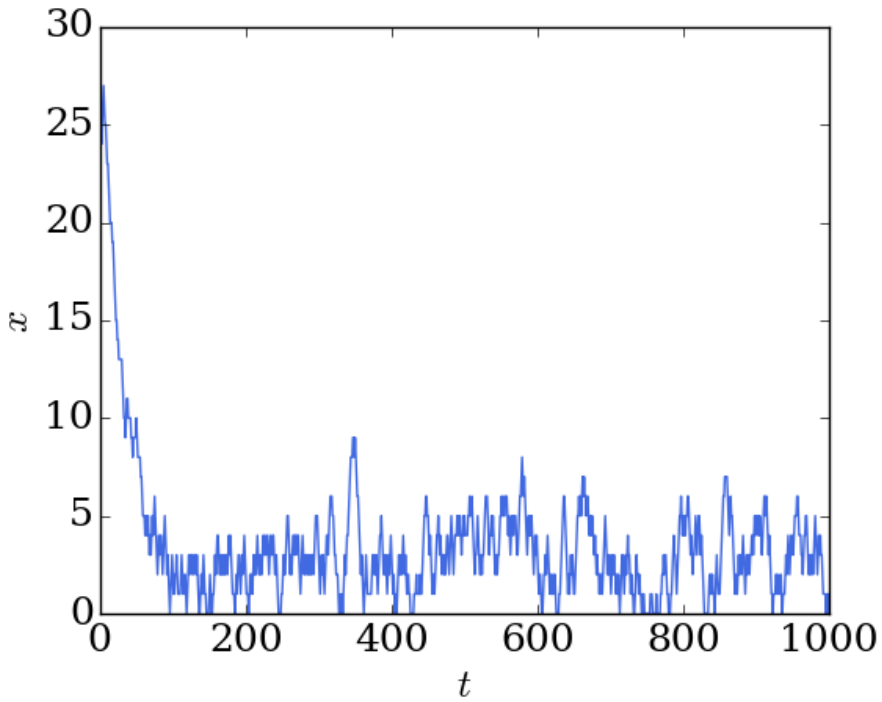
Example

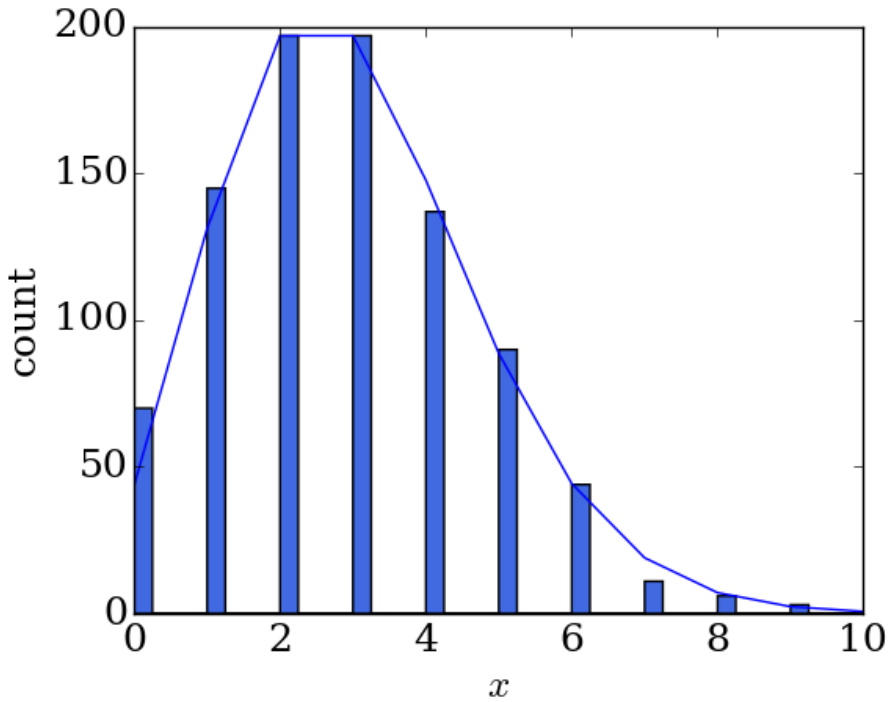
Suppose we want to sample from the 1D PDF $p(x|D,I) = \lambda^x e^{-\lambda}/x!$. Let's choose $q(y|x_t)$ to be a simple random walk defined by the uniform distribution [2].

1. Given x_t, pick a random number $u_1 \sim \text{Uniform}(0,1)$
2. If $u_1 > 0.5$:
   propose $y = x_t + 1$
   otherwise, $y = x_t - 1$
3. Compute the ratio $r = p(y|D,I)/p(x_t|D,I) = \lambda^{y-x} x!/y!$
4. Generate a second random number $u_2 \sim \text{Uniform}(0,1)$.
   If $u_2 \leq r$:
   accept $x_{t+1} = y$
   otherwise, $x_{t+1} = x_t$

MCMC for a Poisson Distribution

Results from MCMC simulation starting at $x_0 = 25$:

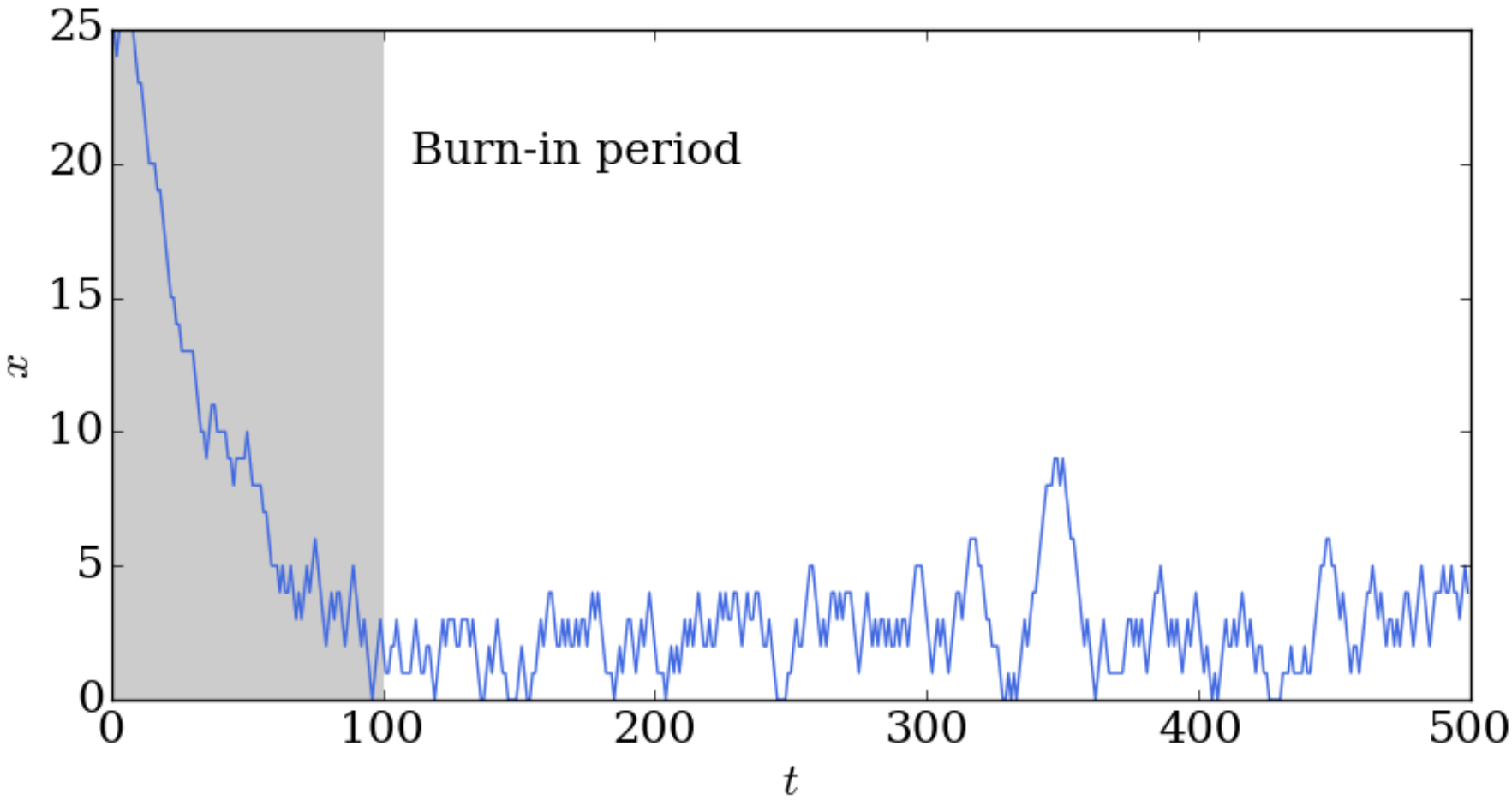




This shows a sequence of the first 1000 samples $\{x_t\}$ from the MCMC (left) and a histogram of the x_t for $t > 100$. We cut out the first 100 samples to allow the MCMC to "burn in"

The Burn-In Period

The MCMC requires a burn-in period before the transition probability becomes independent of t. The length of the burn-in depends on the starting values and proposal distribution



The Autocorrelation Function

- An optimized MCMC should give you a small autocorrelation in $\{x_t\}$
- The cross correlation of two time series $\{x_t\}$ and $\{y_t\}$ is

$$\rho_{xy}(h) = \mathrm{E}\left[(x_t - \mu_x)(y_{t+h} - \mu_y)\right] / (\sigma_{x_t}\sigma_{y_{t+h}})$$

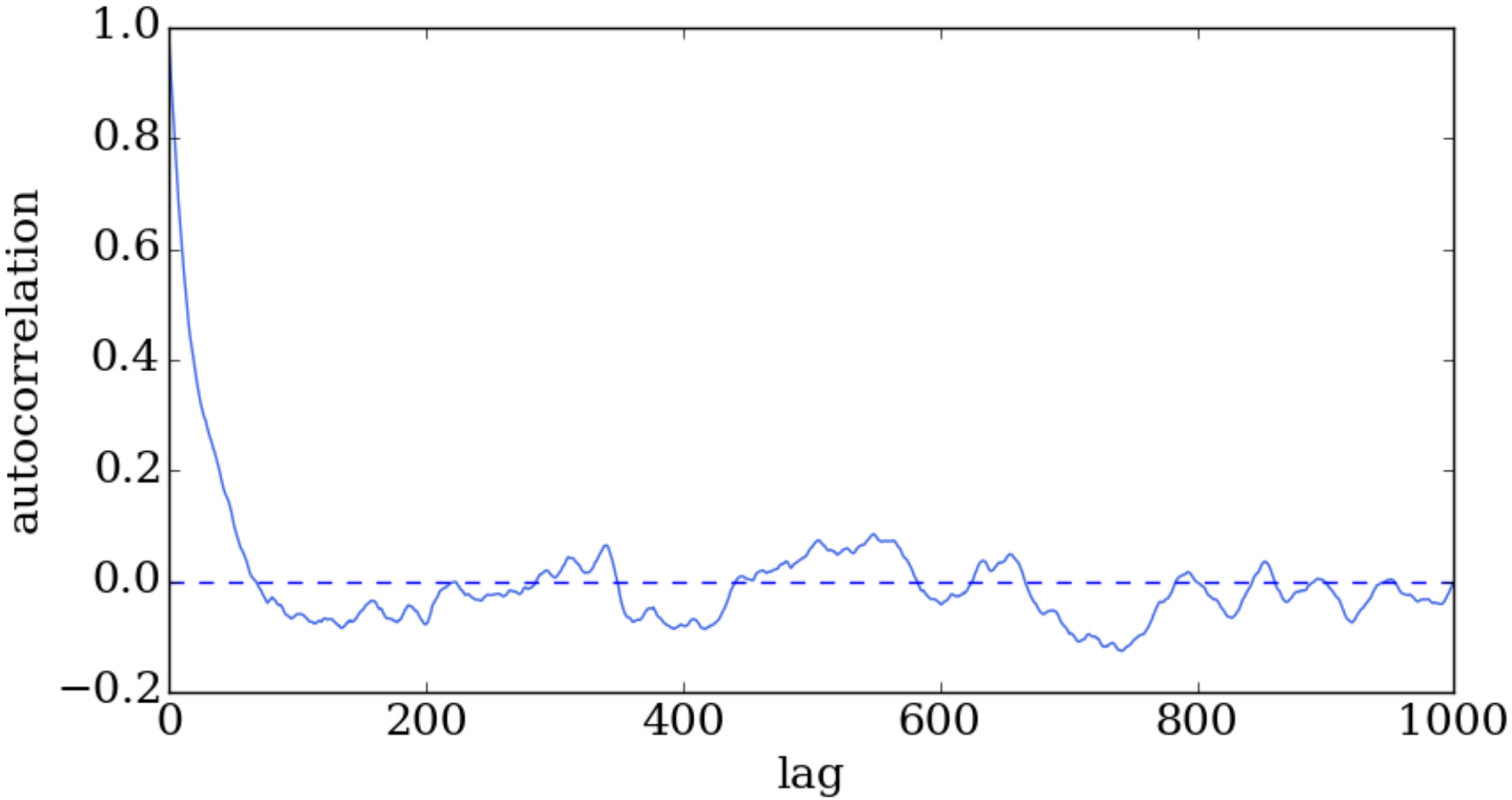
  where h is a lag or shift between the series. The expectation is calculated in the overlap between the series
- Autocorrelation is the lagged correlation of a series with itself:

$$\begin{aligned}\rho_{xx}(h) &= \mathrm{E}\left[(x_t - \mu_x)(x_{t+h} - \mu_x)\right] / (\sigma_{x_t}\sigma_{x_{t+h}}) \\ &= \frac{\sum_{\text{overlap}}\left[(x_t - \bar{x})(x_{t+h} - \bar{x})\right]}{\sqrt{\sum_{\text{overlap}}(x_t - \bar{x})^2}\sqrt{\sum_{\text{overlap}}(x_{t+h} - \bar{x})^2}}\end{aligned}$$

The Autocorrelation Function

ACF from the Poisson MCMC

Autocorrelation tells you how much each step in the time series depends on the value of previous steps:



Autocorrelation Time Constant

- If the transition probability is independent of t then the ACF should fluctuate around zero. This is what happens after the burn-in
- During the burn-in, the ACF is roughly exponential in shape,

$$\rho_{xx}(h) \sim \exp\left\{-\frac{h}{\tau}\right\}$$

  where τ is called the time constant
- Larger τ means that the MCMC takes longer to converge, so the goal is to choose a proposal distribution that minimizes τ
- Empirically, you can estimate τ from the data and start using the data when t is several multiples of τ
- For our Poisson example, $\tau \approx 23$ samples, so to be safe we've started using the data at $t = 4\tau \approx 100$

A Note on Implementation

When implementing a calculation, it is always better to use logarithms rather than actual values to avoid hitting numeric limits:

```
def poisson(lmda, x):
    logp = x*np.log(lmda) - lmda - gammaln(x+1.)
    return np.exp(logp)

def mhRatio(lmda, x, y):
    logr = (y-x)*np.log(lmda) + gammaln(x+1.) -
   gammaln(y+1.)
    return np.exp(logr)
```

If the actual PDF is needed we exponentiate at the end of the calculation. Note that we used the definition $n! = \Gamma(n+1)$ and called the function `scipy.special.gammaln` instead of using Stirling's approximation $\ln n! \approx n \ln n - n$

Detailed Balance

The Metropolis-Hastings algorithm works because it reaches an equilibrium state after the burn-in. In particular, the transition probabilities obey the detailed balance equation, which characterizes a Markov Chain:

$$
\begin{aligned}
p(x_t, x_{t+1}|D,I) &= p(x_t|D,I)\; p(x_{t+1}|x_t) \\
&= p(x_t|D,I)\; q(x_{t+1}|x_t)\; \alpha(x_t, x_{t+1}) \\
&= p(x_t|D,I)\; q(x_{t+1}|x_t)\; \min\left(1, \frac{p(x_{t+1}|D,I)}{p(x_t|D,I)} \frac{q(x_t|x_{t+1})}{q(x_{t+1}|x_t)}\right) \\
&= \min\left(p(x_t|D,I)\; q(x_{t+1}|x_t), p(x_{t+1}|D,I)\; q(x_t|x_{t+1})\right) \\
&= p(x_{t+1}|D,I)\; q(x_t|x_{t+1})\; \alpha(x_{t+1}, x_t) \\
&= p(x_{t+1}|D,I)\; p(x_t|x_{t+1})
\end{aligned}
$$

Therefore, $p(x_{t+1}|x_t)\; p(x_t|D,I) = p(x_t|x_{t+1})\; p(x_{t+1}|D,I)$; the rate of transitions $x_t \to x_{t+1}$ is the same as the rate of transitions $x_{t+1} \to x_t$

MCMC Efficiency

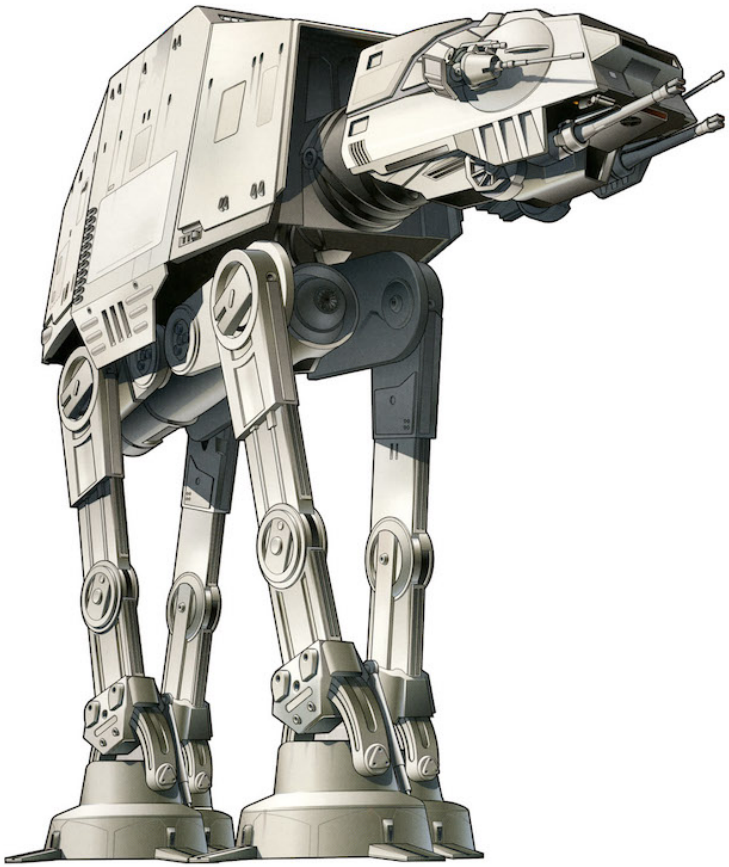
A number of issues have to be decided when running an MCMC:

1. What is the length of the burn-in period? I.e., when can we start trusting the data?
2. When do we stop the Markov Chain? I.e., how do we know if we've sufficiently sampled the parameter space?
3. How do we choose a suitable proposal distribution that gives a reasonable acceptance rate for transitions $x_t \to x_{t+1}$?

There is a large literature about optimizing Markov Chain Monte Carlo, as you might imagine [3]. An MCMC that takes forever to burn-in or which accepts few transitions isn't worth much

Current state of the art: affine-invariant samplers [4], which are implemented in the Python package `emcee` [5]

MCMC Parallelization



- There are various tricks to speed up MCMC and ensure that it explores as much of the parameter space as possible
- One common approach is to define multiple chains (or "walkers") that have different starting points and proceed independently
- If the sampled PDF is very peaked or multimodal, this might still not be enough to push explore all parts of the parameter space. We'll discuss how to deal with that after the next example

Table of Contents

1. Sampling from PDFs in Many Dimensions
   - Markov Chain Monte Carlo
   - The Metropolis-Hastings Algorithm
2. Case Study: Sampling from a 1D Distribution
   - Burn-In
   - Autocorrelation
   - Efficiency and Detailed Balance
3. Sampling from a Joint Distribution
   - Parallel Tempering

Sampling from a Joint Posterior

Example

Now consider sampling from a joint distribution $p(x_1, x_2|D, I)$ in two parameters x_1 and x_2. The PDF is the sum of two 2D Gaussians and has a double-peaked structure [2]:

$$p(x_1, x_2|D, I) = \frac{1}{2}\left[\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)\right]$$

where $\boldsymbol{\mu}_1 = (0, 0)$, $\boldsymbol{\mu}_2 = (4, 3)$, and

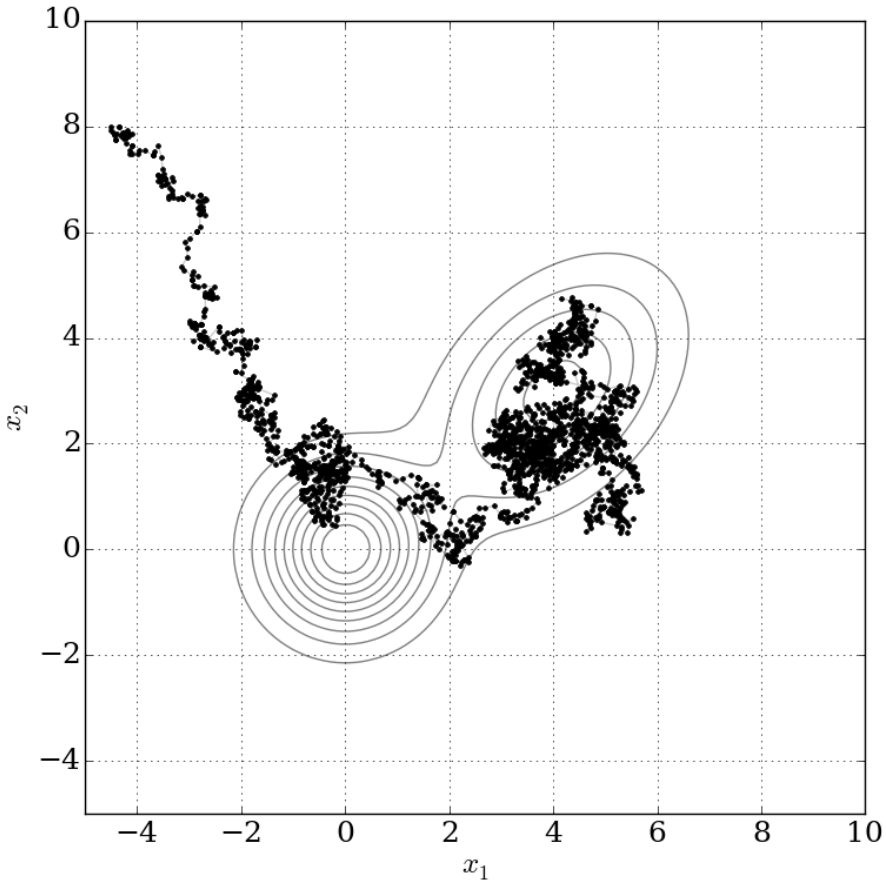
$$\boldsymbol{\Sigma}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \boldsymbol{\Sigma}_2 = \begin{pmatrix} 2 & 0.8 \\ 0.8 & 2 \end{pmatrix}$$

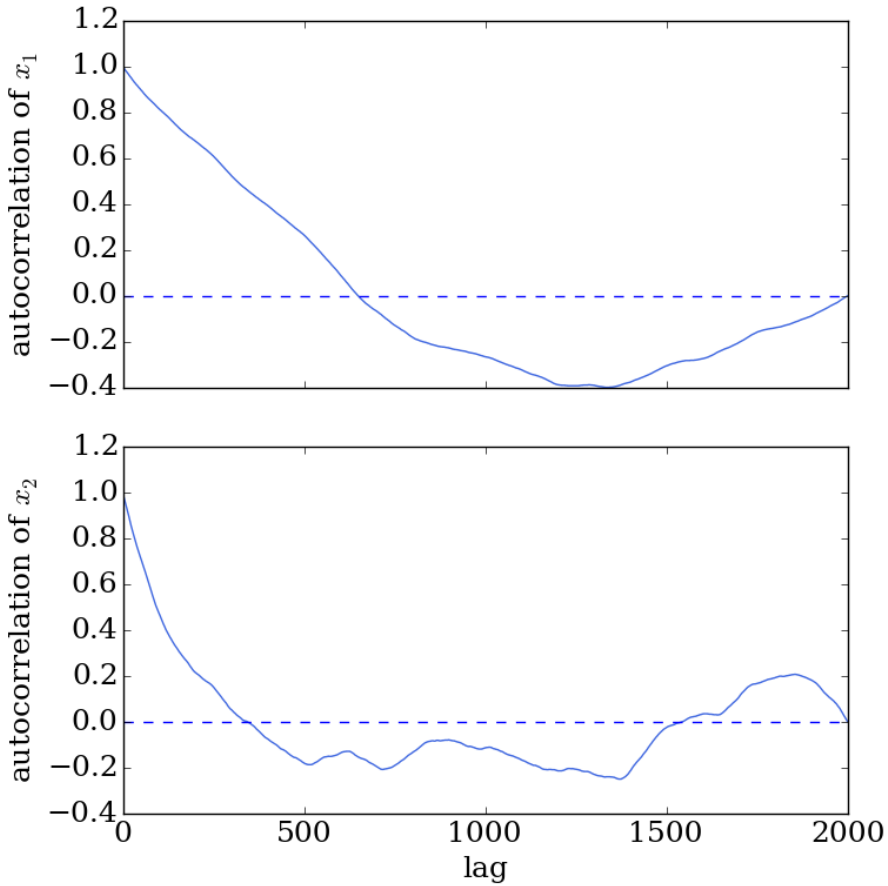
For the proposal density function q, use a unimodal 2D Gaussian:

$$q(\boldsymbol{y}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu} = \boldsymbol{x}, \boldsymbol{\Sigma}_q), \quad \boldsymbol{\Sigma}_q = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix}$$

Sampling Distribution with $\sigma = 0.1$

Start at $x_0 = (-4.5, 8)$ with the width of the proposal PDF set to $\sigma = 0.1$.

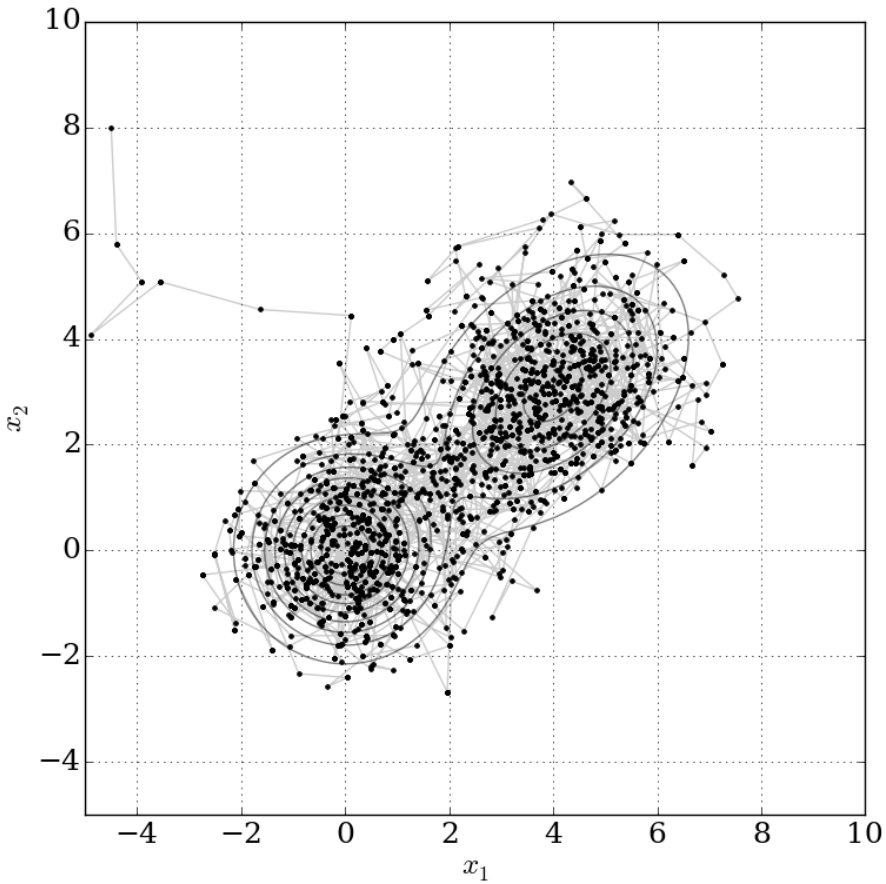


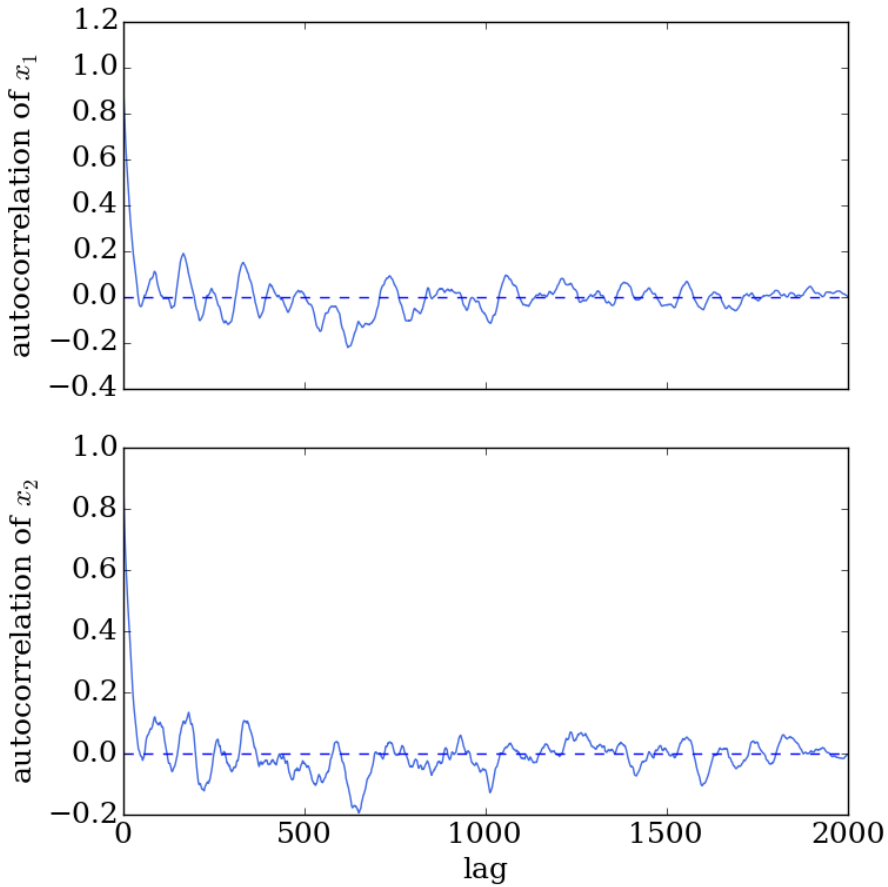


Notice the long autocorrelation time. Acceptance probability is $\sim 95\%$

Sampling Distribution with $\sigma = 1$

Start at $x_0 = (-4.5, 8)$ with the width of the proposal PDF set to $\sigma = 1$.

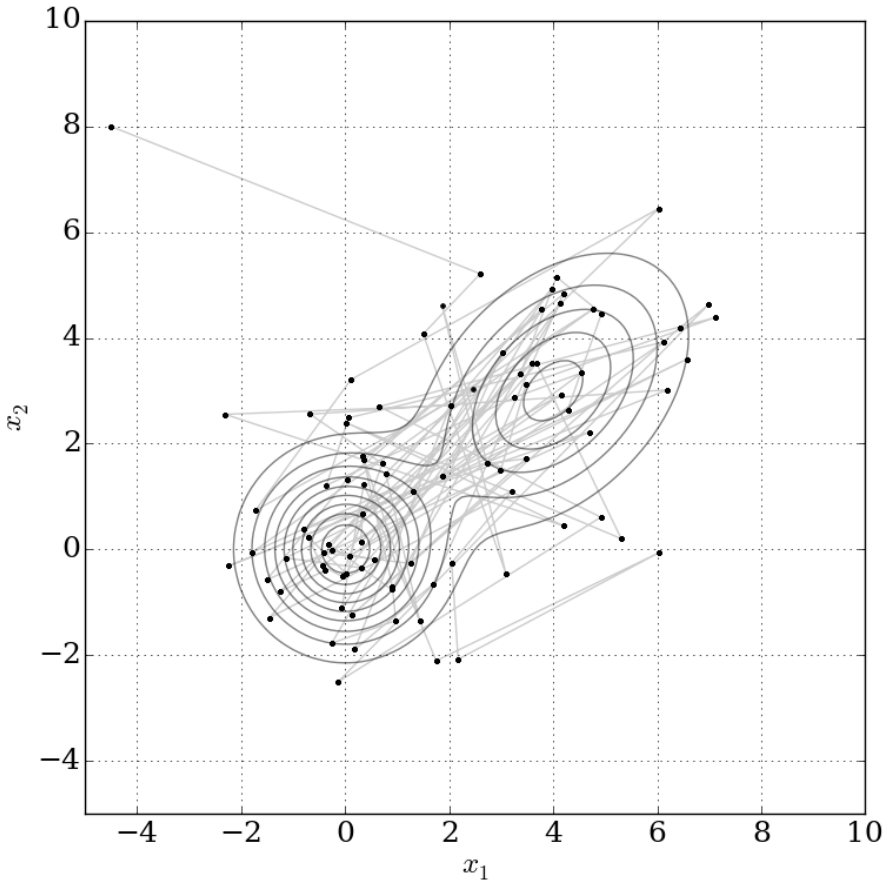


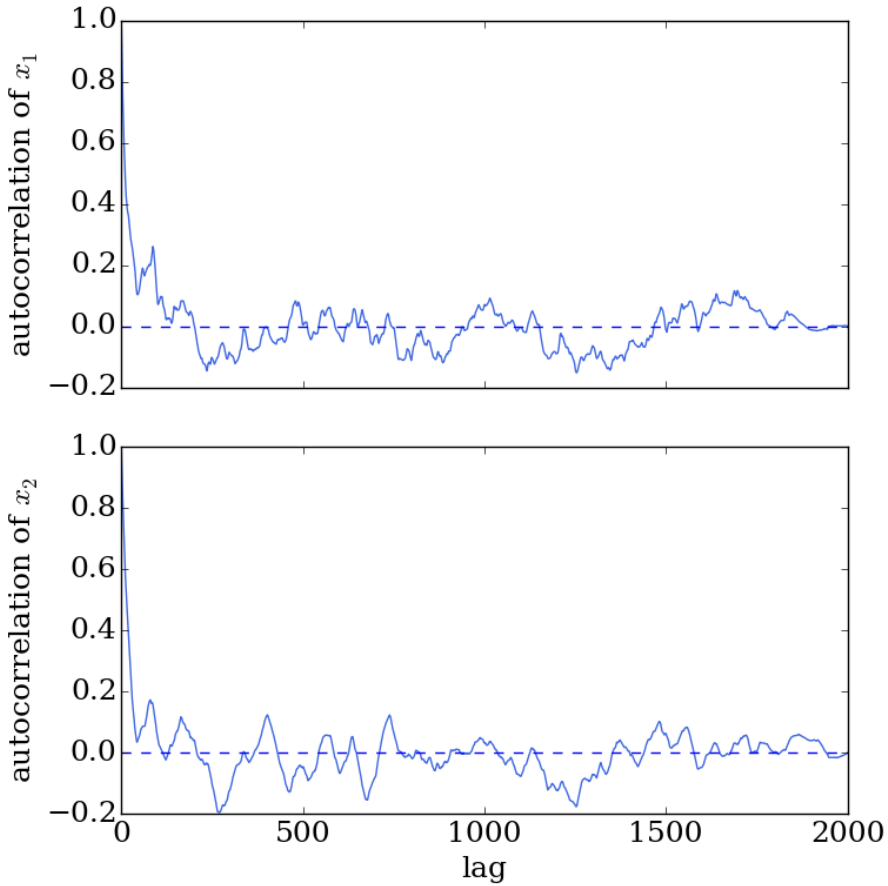


Faster convergence, with acceptance probability $\sim 60\%$

Sampling Distribution with $\sigma = 10$

Start at $x_0 = (-4.5, 8)$ with the width of the proposal PDF set to $\sigma = 10$.





Fast convergence, but acceptance probability is now $\sim 5\%$

Dealing with Multimodal Distributions

- The double-peaked Gaussian example showed that the MCMC can become stuck in a local mode
- **Recall**: this is similar to the situation in parameter estimation when a minimizer gets stuck in a local minimum
- The solution is to create a series of progressively "flatter" distributions using a temperature parameter T (or $\beta = 1/T$). As $T \to \infty$ and $\beta \to 0$, the distribution will flatten and more of the paramter space can be explored
- Given a posterior

$$p(\boldsymbol{x}|D, I) \propto p(\boldsymbol{x}|I)\, p(D|\boldsymbol{x}, I)$$

we can construct a flattened distribution using $\beta \in [0, 1]$:

$$\pi(\boldsymbol{x}|D, \beta, I) = p(\boldsymbol{x}|I)\, p(D|\boldsymbol{x}, I)^{\beta} = p(\boldsymbol{x}|I)\ \exp\left(\beta \ln\left[p(D|\boldsymbol{x}, I)\right]\right)$$

Parallel Tempering

- With $\pi(\boldsymbol{x}|D,\beta,I)$, we can use a set of discrete values $\beta = \{1,\beta_2,\ldots,\beta_m\}$ in parallel
- **Parallel Tempering**: multiple copies of the MCMC are run in parallel, each with a different temperature β_i
- As the simulations run, pairs of adjacent simulations on the temperature ladder are allowed to swap their parameter states with probability

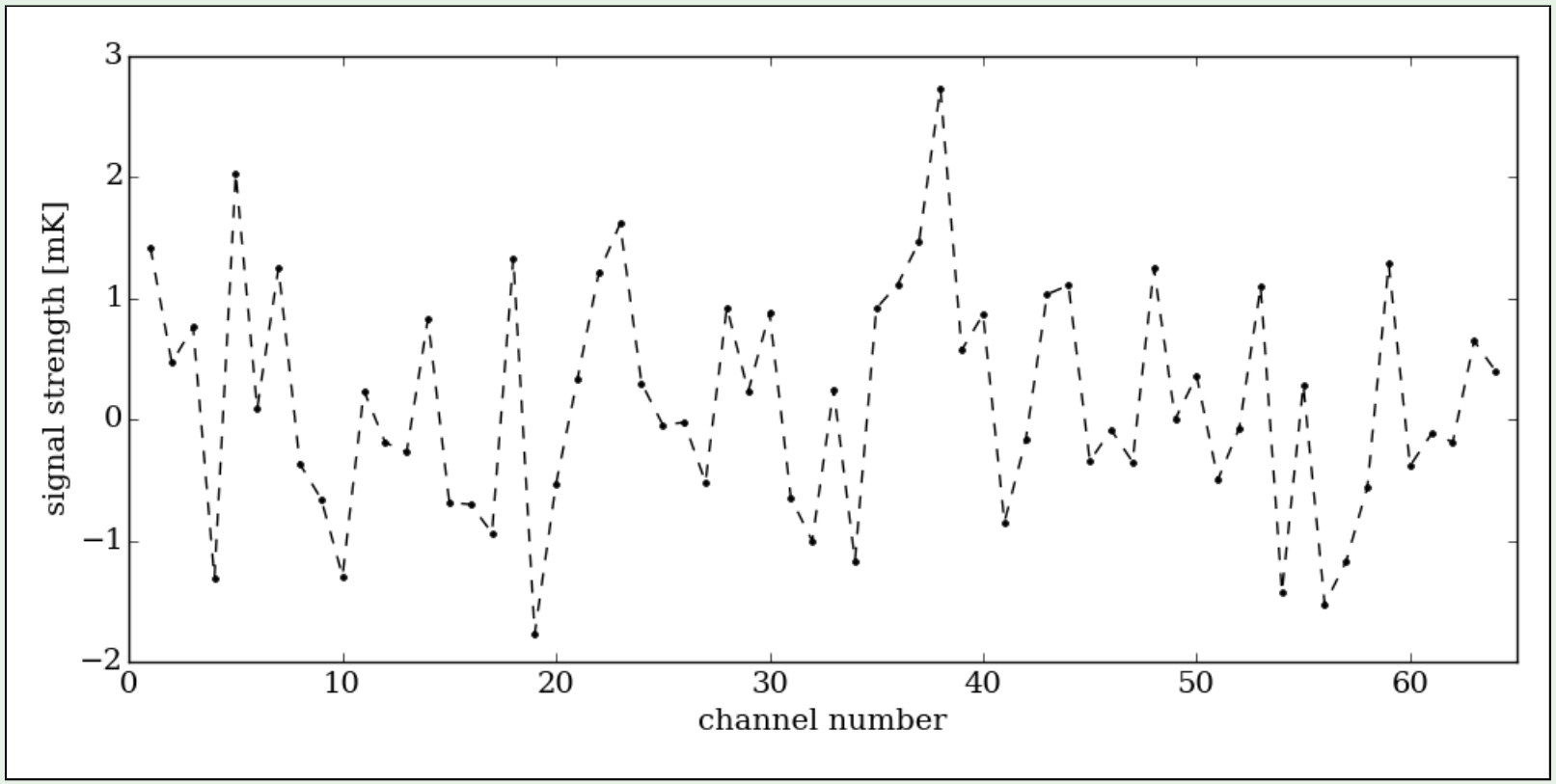
$$r = \min\left\{1, \frac{\pi(\boldsymbol{x}_{t,i+1}|D,\beta_i,I)\ \pi(\boldsymbol{x}_{t,i}|D,\beta_{i+1},I)}{\pi(\boldsymbol{x}_{t,i}|D,\beta_i,I)\ \pi(\boldsymbol{x}_{t,i+1}|D,\beta_{i+1},I)}\right\}$$

- Algorithm:
  1. Propose a swap every n_s iterations, and proceed with the swap if $u_1 \sim \text{Uniform}(0,1) \leq 1/n_s$
  2. Randomly pick simulation i to swap its state with simulation $i+1$
  3. Accept the swap if $u_2 \sim \text{Uniform}(0,1) \leq r$

Bump Finding

Example

We have a 64-channel spectrum from a radio spectrometer with an instrumental resolution of 2 channels and Gaussian noise of 1 mK per channel [2]. Is there a peak in the spectrum and what is its amplitude?



Bump Finding: Problem Setup

- If there is a bump in channel ν_0 with amplitude A, we want to calculate

$$p(A, \nu_0 | D, I) \propto p(D | A, \nu_0, I)\; p(A, \nu_0 | I)$$

- A is a scale parameter and ν_0 a location parameter, so it seems reasonable to choose

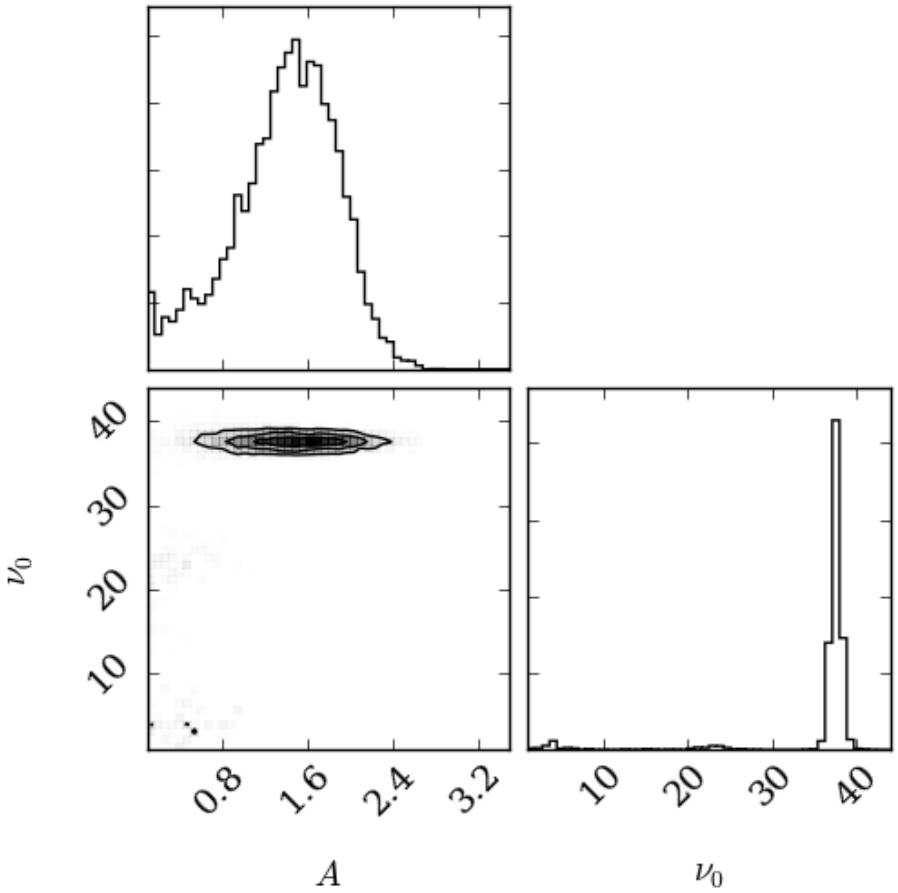
$$p(A, \nu_0 | I) = p(A | I) \times p(\nu_0 | I) = \frac{1}{A \ln\left(A_{\max}/A_{\min}\right)} \times \frac{1}{\nu_{\max} - \nu_{\min}}$$

- Meanwhile, the likelihood is given by a product of Gaussians:

$$p(D | A, \nu_0, I) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ -\frac{(d_i - A f_i)^2}{2\sigma^2} \right\}, \quad \sigma = 1\ \mathrm{mK}$$

$$f_i = \exp\left\{ -\frac{(\nu_i - \nu_0)^2}{2\sigma_L^2} \right\}, \quad \sigma_L = 2$$

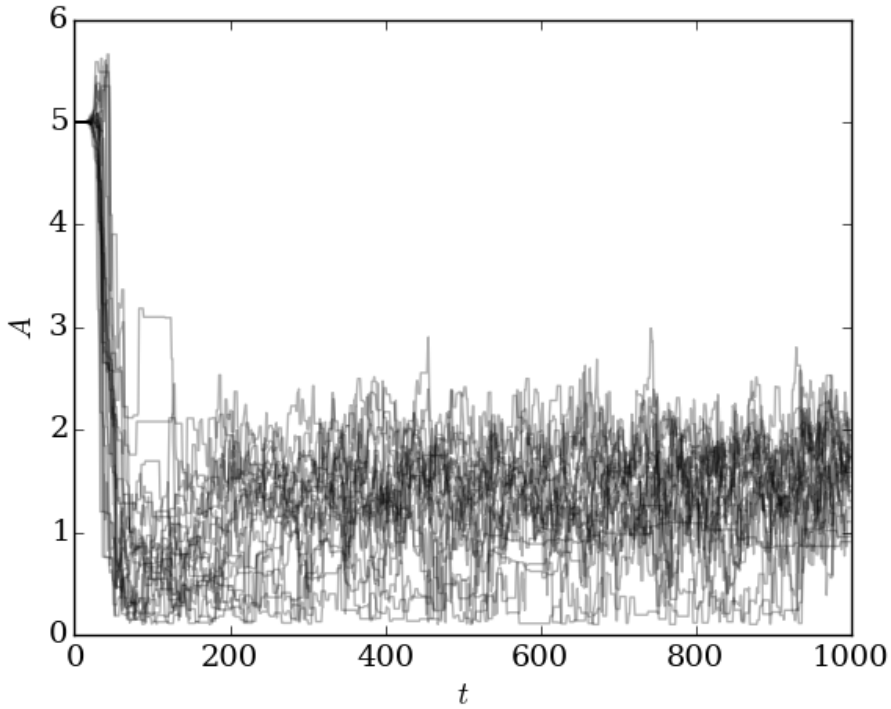
Results: Simple MCMC

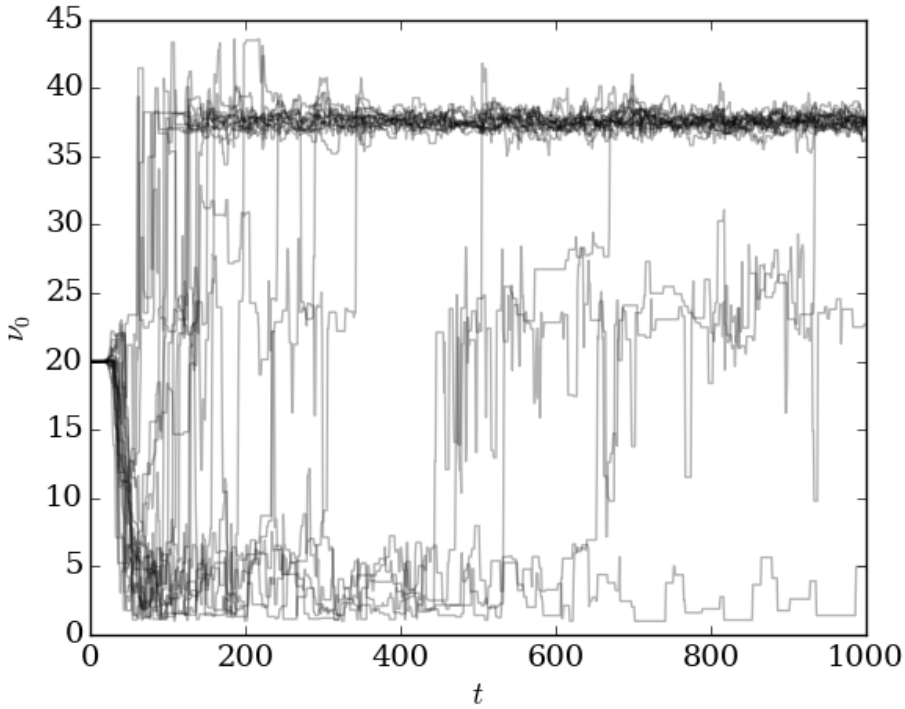


- Putting it all together, we draw random samples from $p(A, \nu_0|D, I)$ using the `emcee` package [5]
- Simulation parameters:
  1. 2 free parameters A, ν_0
  2. 20 MCMC "walkers"
  3. 1000 samples
- The posterior PDF is shown at left with the marginal distributions of A and ν_0
- Note that the first 100 samples from each walker were treated as burn-in data and ignored

Bump Finding: Burn-In

The time series of A and ν_0 indicate burn in after 100 samples

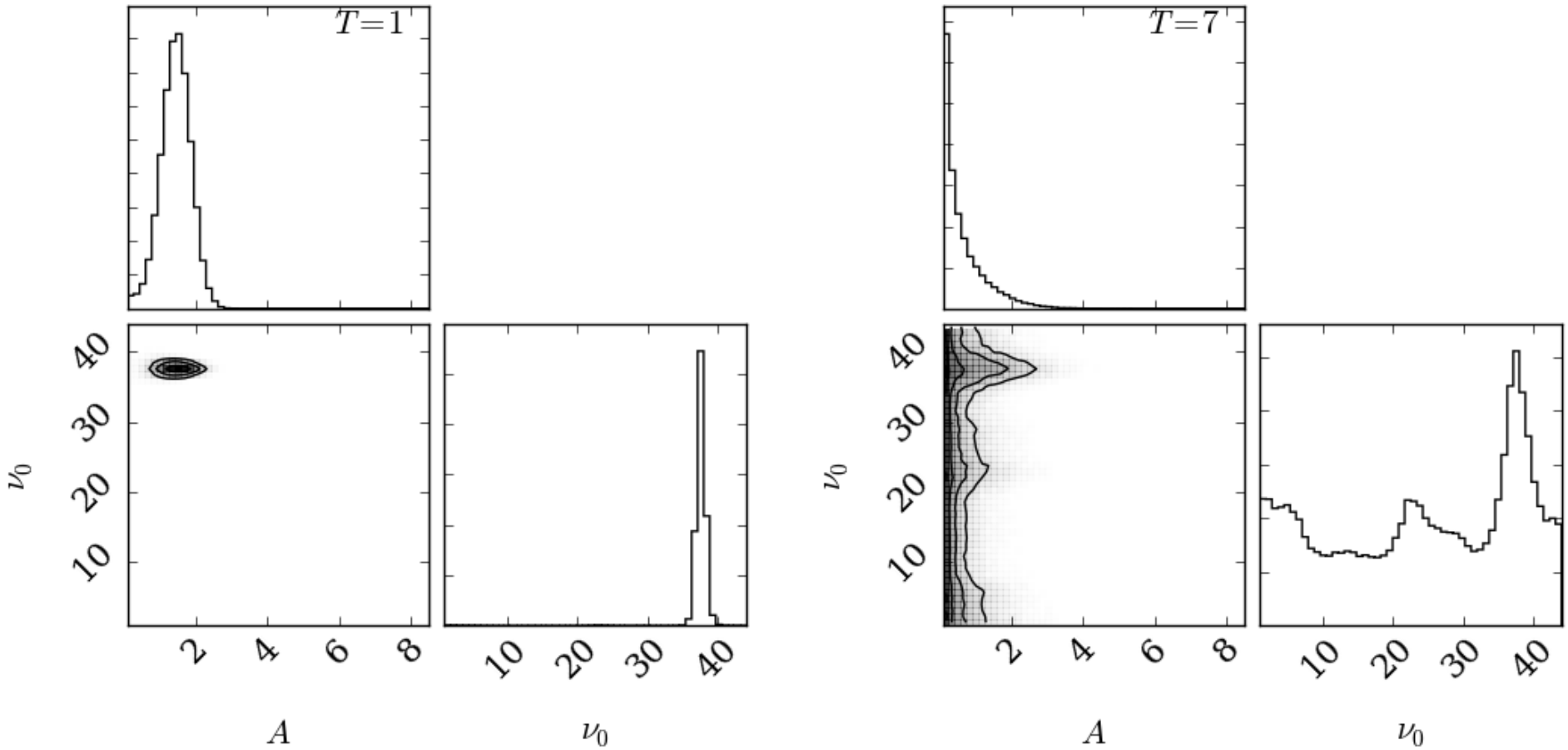




The distribution of ν_0 shows some multimodality; several walkers do not converge to $\nu_0 = 37$. We can explore this more with parallel tempering

Parallel Tempering: $T = 1, T = 7$

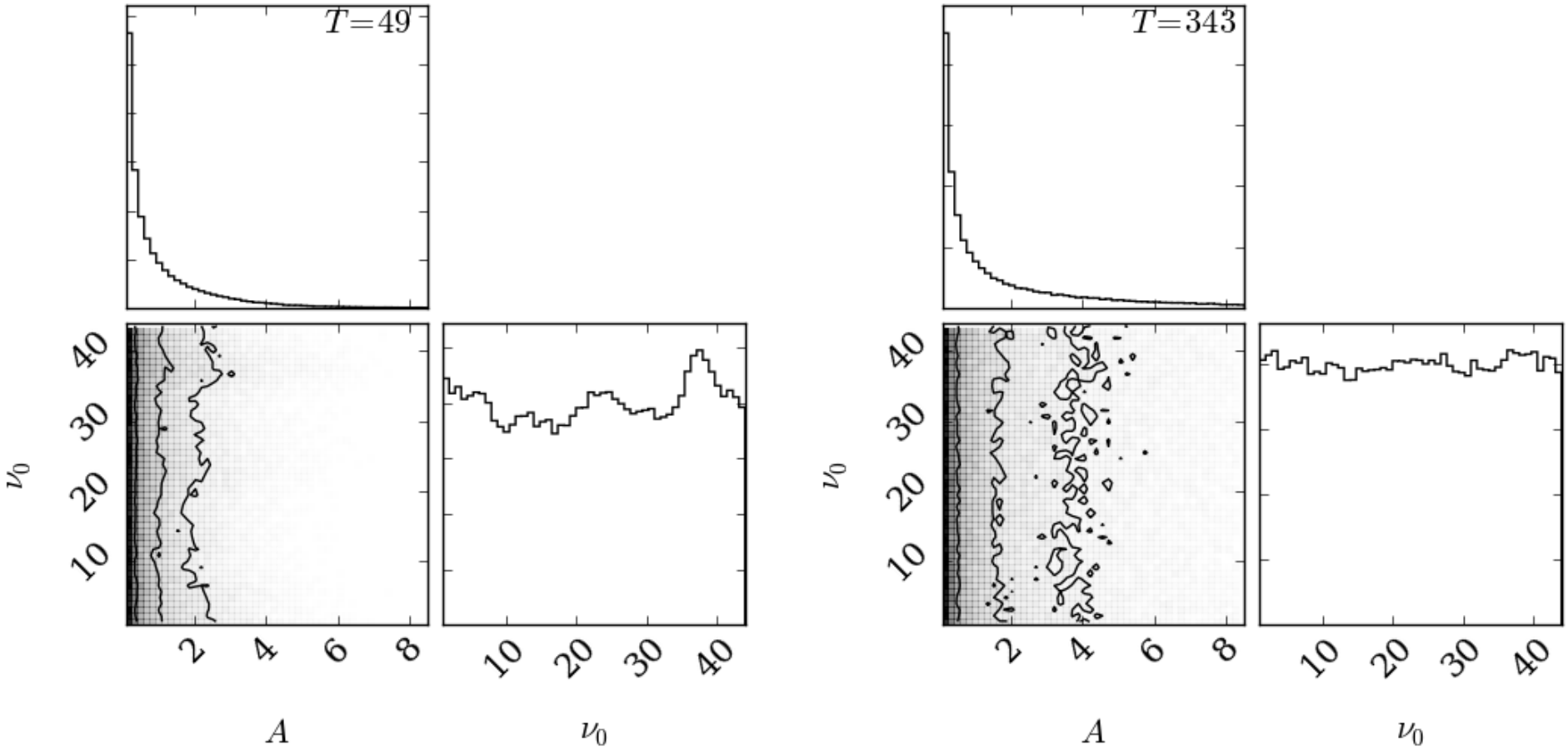
Let's try the same simulation with 4 parallel simulations, with T increasing in powers of 7 between each simulation



Notice how $p(A|D, I)$ loses its bump as T increases!

Parallel Tempering: $T = 49$, $T = 343$

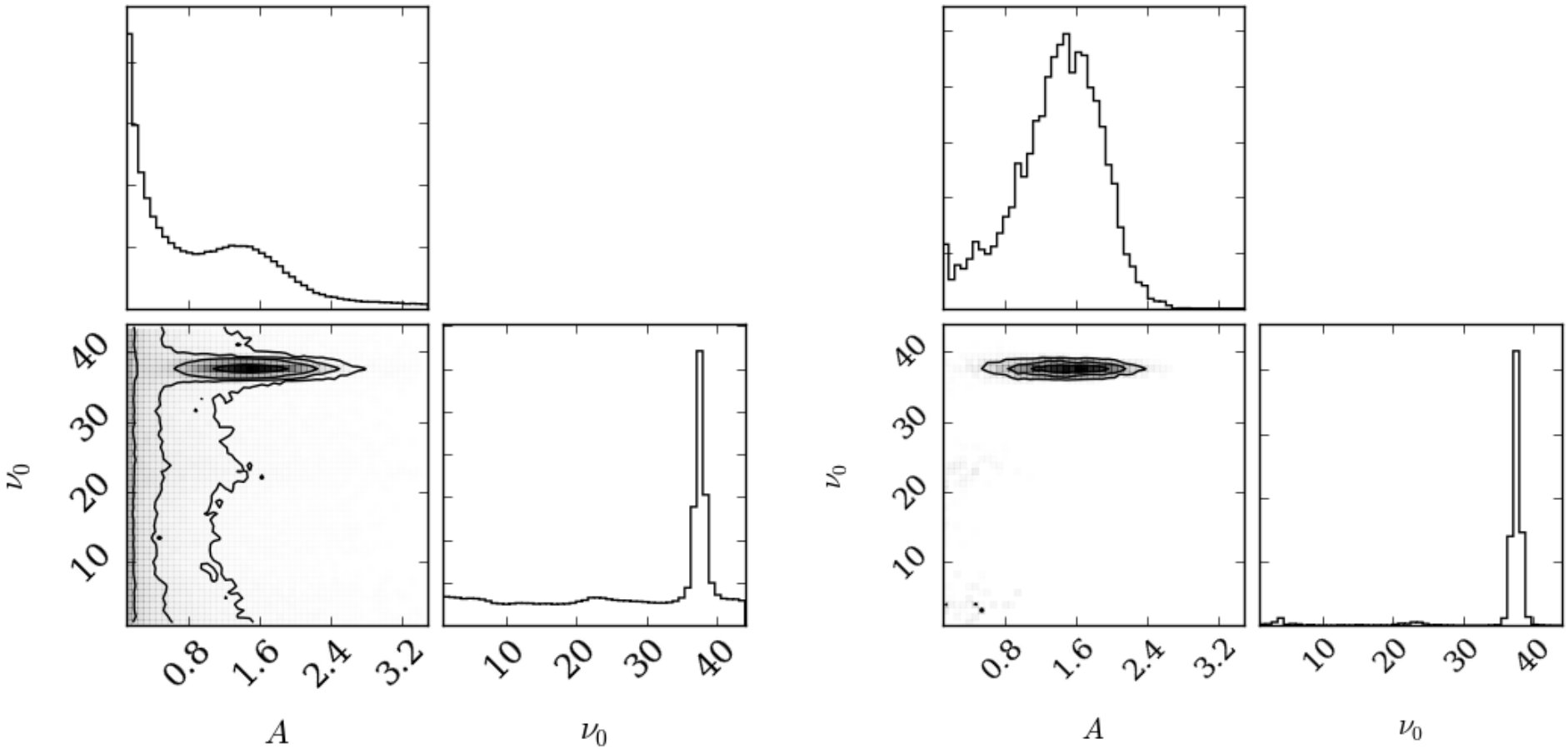
As T goes up, the features in the PDF are getting washed out



The sampling distribution is becoming increasingly flat and the MCMC is exploring the full parameter space

Parallel Tempering: Final Results

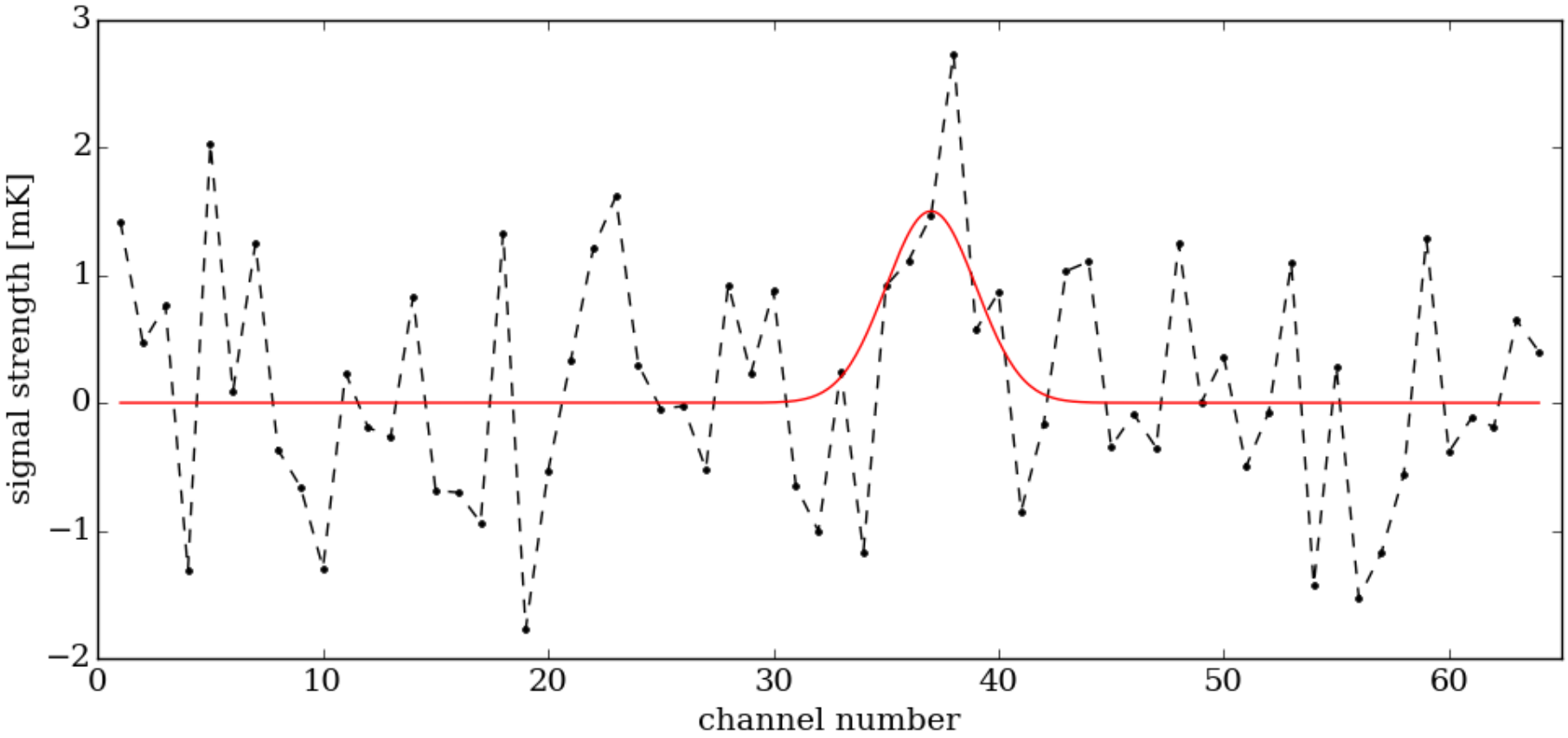
The combination of the four simulations shows multimodal behavior in the PDF



Compare the final results (left) to our original MCMC (right)

Bump Finding: Best Fit

Plugging in $\hat{A} \approx 1.5$ and $\hat{\nu}_0 \approx 37$ we get:



Next time: model comparison. We would like to estimate a Bayes factor comparing the bump model to a null hypothesis where there is no bump

Summary

- MCMC is a general technique for generating parameter samples from high-dimensional PDFs $p(\boldsymbol{\theta}|D, I)$
- Issues that affect MCMC calculations:
  1. Estimating the length of the burn-in period
  2. Deciding when to stop the Markov chain
  3. Choosing a suitable proposal distribution
- For a given problem these issues are usually addressed by trial and error. You tune $q(\boldsymbol{y}|\boldsymbol{x}_t)$ to get an acceptance rate of $25\% - 50\%$, play with the starting values $\boldsymbol{x}_0$, look at the results to assess burn-in, etc.
- There are also nice tricks like the use of walkers and parallel tempering that help ensure your MCMC explores the full parameter space
- Coding up a Metropolis-Hastings problem is not hard, but be aware of existing packages like `emcee` that make running MCMC much less of a grind [5]

References I

[1] N. Metropolis et al. "Equation of State Calculations by Fast Computing Machines". In: *J. Chem. Phys.* 21 (1953), p. 1087.

[2] P. Gregory. *Bayesian Logical Data Analysis for the Physical Sciences*. New York: Cambridge University Press, 2005.

[3] D. MacKey. *Information Theory, Inference, and Learning Algorithms*. New York: Cambridge University Press, 2003.

[4] J. Goodman and J. Weare. "Ensemble Samplers with Affine Invariance". In: *Comm. Appl. Math. Comp. Sci.* 5 (2010), 65–80.

[5] D. Foreman-Mackey et al. *emcee: The MCMC Hammer*. 2013. URL: http://dan.iel.fm/emcee/current/.