Lecture 17

Gibbs, Data Augmentation, and HMC

The idea of Gibbs

$$f(x) = \int f(x,y)dy = \int f(x|y)f(y)dy = \int dy f(x|y) \int dx' f(y|x')f(x')$$

Thus: $f(x) = \int h(x,x')f(x')dx'$ integral fixed point equation

where $h(x,x') = \int dy f(x|y)f(y|x')$.

Iterative scheme in which the "transition kernel" $h(x, x')$ is used to create a proposal for metropolis-hastings moves:

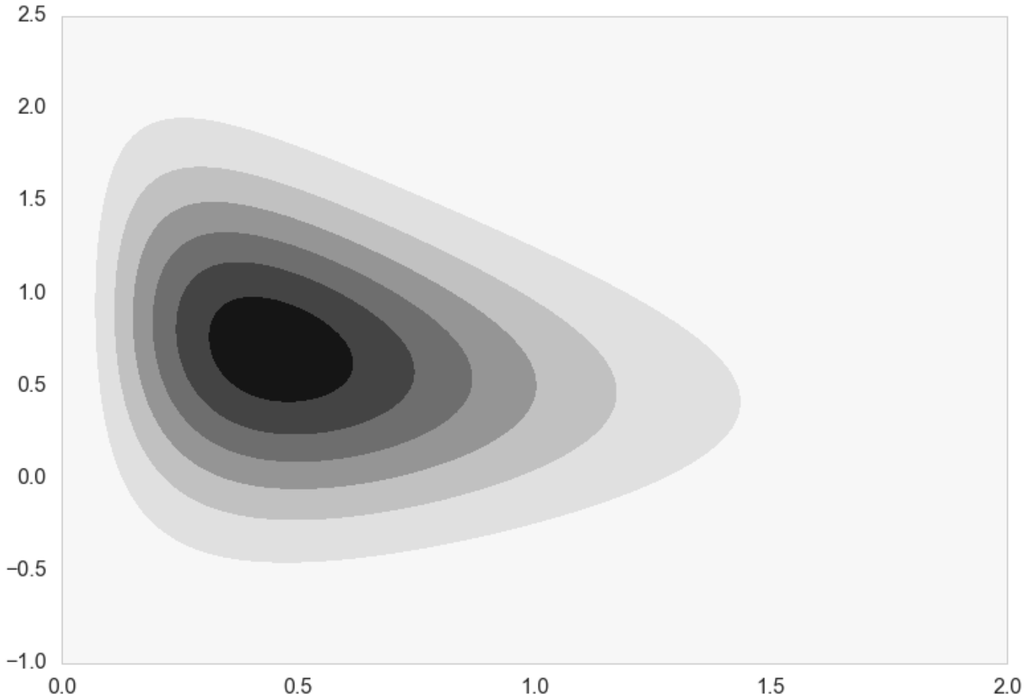
$$f(x_t) = \int h(x_t, x_{t-1}) f(x_{t-1}) dx_{t-1},$$ a Stationary distribution.

$$h(x, x') = \int dy f(x|y) f(y|x').$$: Sample alternately to get transitions.

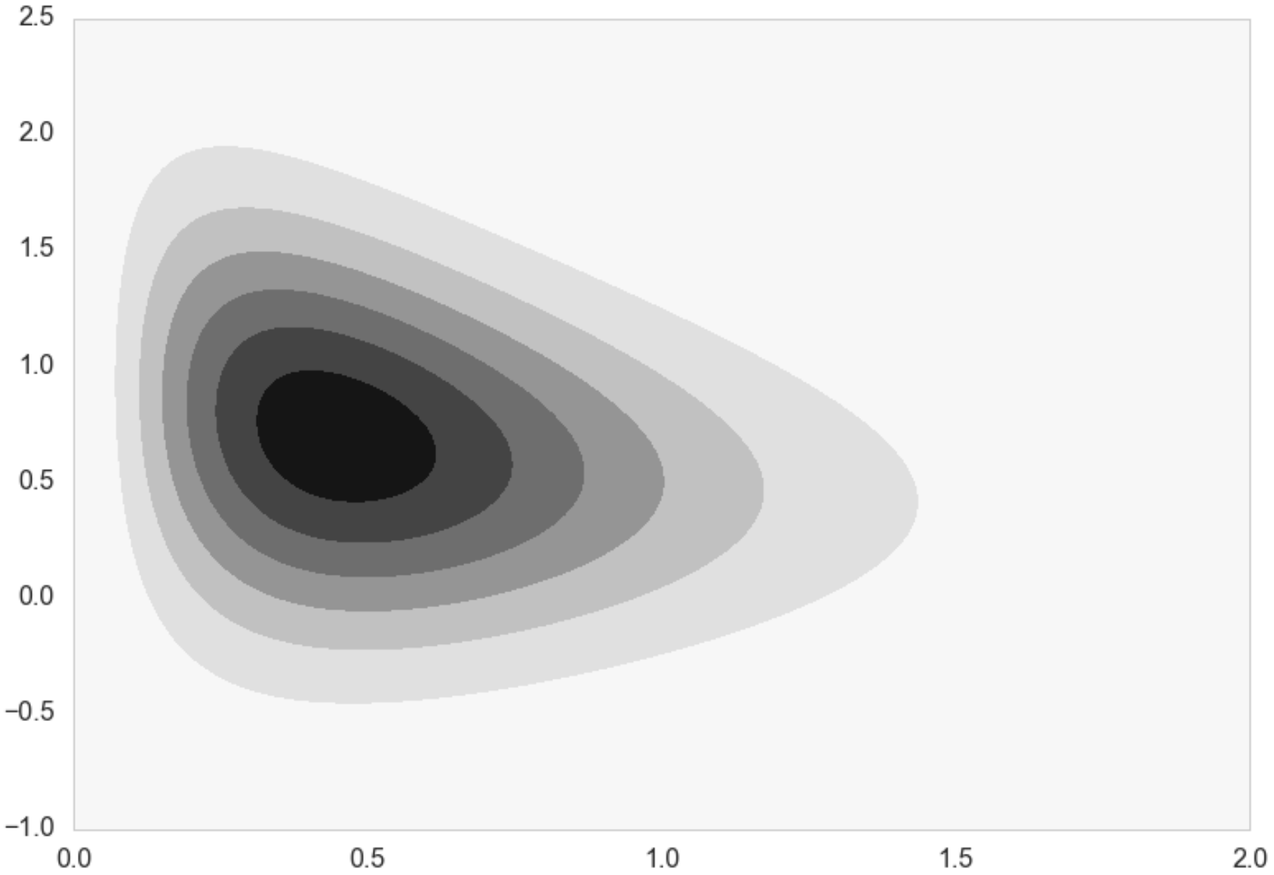
Can sample x marginal and $y|x$ so can sample the joint x, y.

Example

Sample from $f(x, y) = x^2 \exp[-xy^2 - y^2 + 2y - 4x]$



Example:



$$f(x,y) = x^2 \exp[-xy^2 - y^2 + 2y - 4x]$$

$$= x^2 \exp[-x(y^2+4)] \exp[-y^2 + 2y]$$

$$= g(y) \text{Gamma}(x, 3, y^2+4)$$

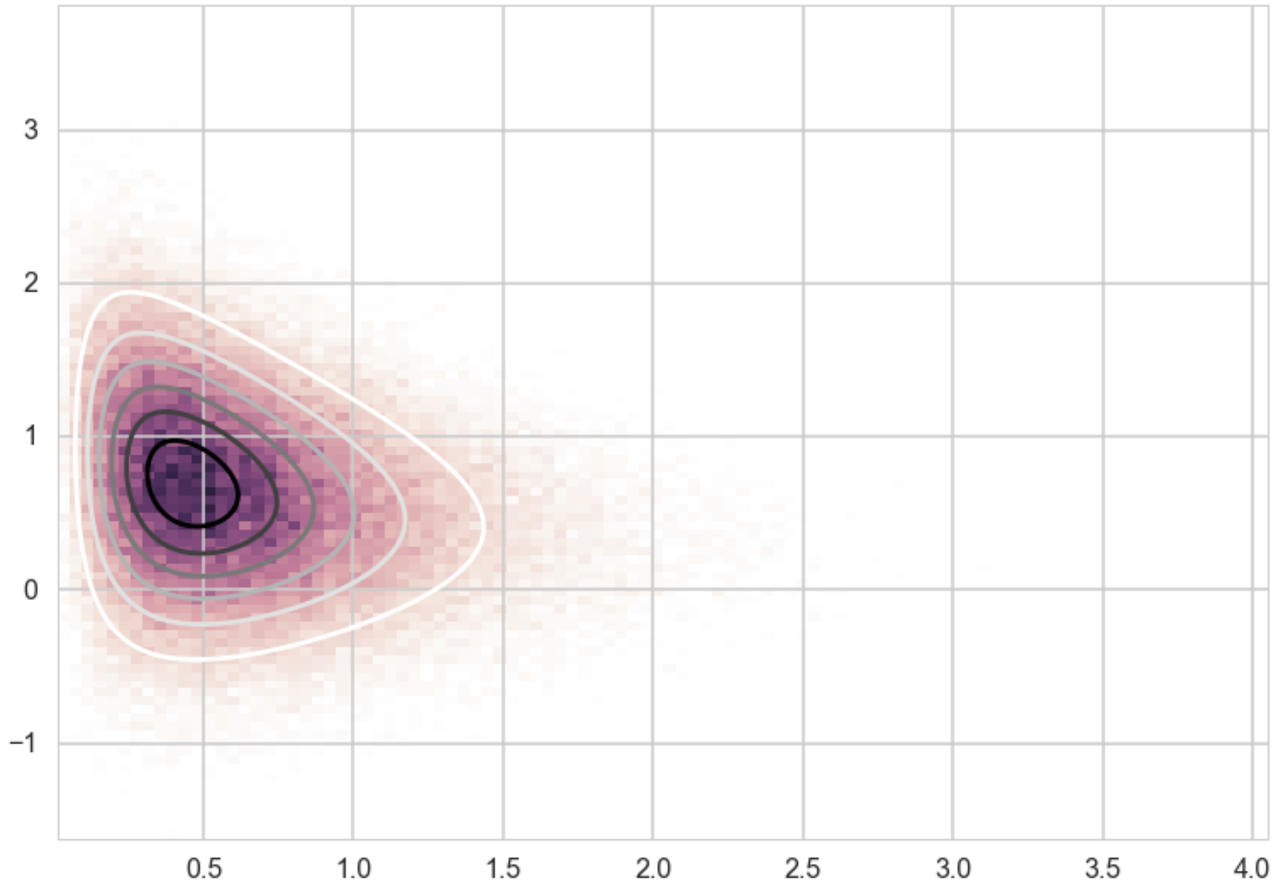
$$\implies f(x|y) \sim \text{Gamma}(3, y^2+4)$$

$$f(x,y) = x^2 \exp[-y^2(1+x) + 2y] \exp[-4x]$$

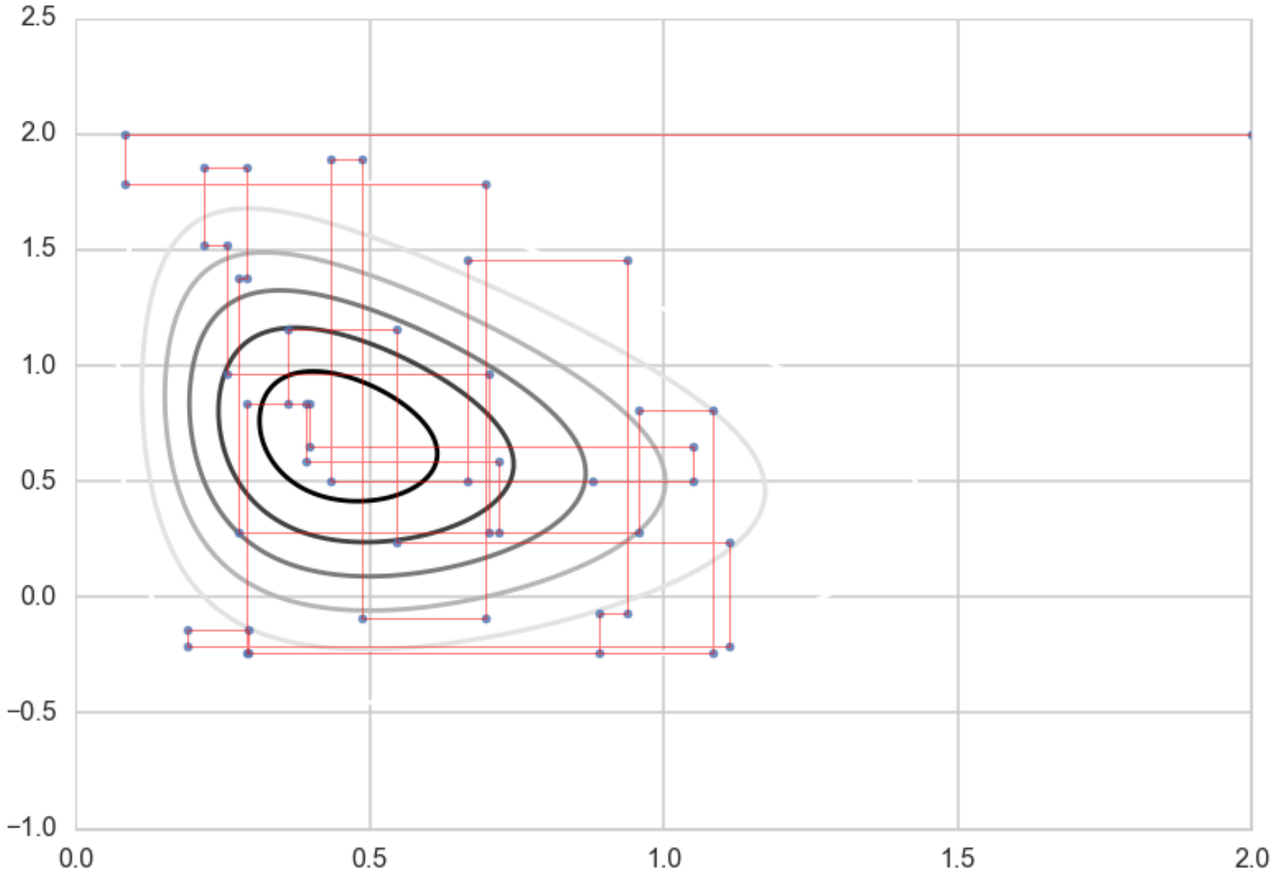
$$\implies f(y|x) \sim N(\frac{1}{1+x}, \frac{1}{\sqrt{(2(1+x))}})$$

Sampler

```
def xcond(y):
    return gamma.rvs(3, scale=1/(y*y + 4))
def ycond(x):
    return norm.rvs(1/(1+x), scale=1.0/np.sqrt(2*(x+1)))
def gibbs(xgiveny_sample, ygivenx_sample, N, start = [0,0]):
    x=start[0]
    y=start[1]
    samples=np.zeros((N+1, 2))
    samples[0,0]=x
    samples[0,1]=y
    for i in range(1,N,2):
        x=xgiveny_sample(y)
        samples[i,0]=x
        samples[i, 1]=y
        ######################
        y=ygivenx_sample(x)
        samples[i+1,0]=x
        samples[i+1,1]=y
    return samples
out=gibbs(xcond, ycond, 100000)
```



More about gibbs



- easiest is to know how to sample directly from conditionals: **no need for locality**
- moves one component (or one block) at a time
- all is not lost if thats not the case: can use a MH-step once stationarity has been reached
- this makes gibbs a very general idea

Fully Bayesian Rat tumors

Joint Posterior:

$$p(\Theta, \alpha, \beta | Y, \{n_i\}) \propto p(\alpha, \beta) \prod_{i=1}^{70} Beta(\theta_i, \alpha, \beta) \prod_{i=1}^{70} Binom(n_i, y_i, \theta_i)$$

Conditionals:

$$p(\theta_i | y_i, n_i, \alpha, \beta) = Beta(\alpha + y_i, \beta + n_i - y_i)$$

More Conditionals

$$p(\alpha|Y,\Theta,\beta) \propto p(\alpha,\beta)\left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)}\right)^N \prod_{i=1}^{N}\theta_i^{\alpha}$$

$$P(\beta|Y,\Theta,\alpha) \propto p(\alpha,\beta)\left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\beta)}\right)^N \prod_{i=1}^{N}(1-\theta_i)^{\beta}$$

These depend on Y and $\{n\}$ via the θ's

Sampling (sampler done in lab)

- Fix α and β, we have a Gibbs step for all of the θ_is
- For α and β, everything else fixed, use stationary metropolis step, as conditionals are not isolatable to simply sampled distributions
- when we sample for α, we will propose a new value using a normal proposal, while holding all the θs and β constant at the old value. ditto for β.

GENERAL GIBBS SAMPLER

- model is a DAG
- for eg, in a hierarchical model, we have observations at the bottom of a tree, next layer intermediate parameters, upper layers hyper-parameters
- sample conditionals. Might be direct due to conjugacy or explicit conditionals, or metropolis steps, or other samplers.

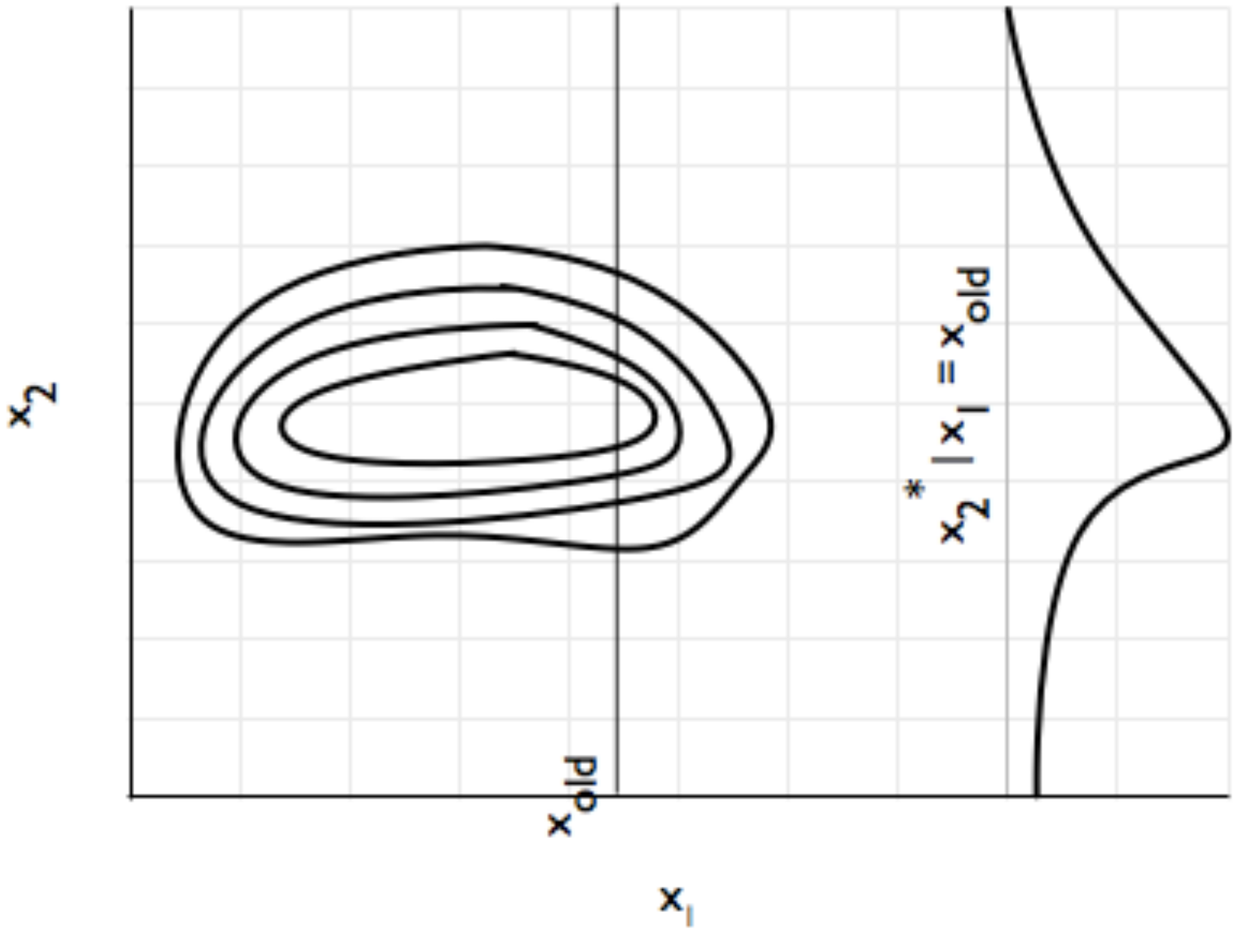
"sampler within gibbs"

More Gibbs Theory

The transition kernel corresponds to this proposal:

$$q_k(x^*|x^i) = \begin{cases} p(x^*_k|x^i_{-k}) & for\ x^*_{-k} = x^i_{-k}, \\ 0 & otherwise \end{cases}$$

where x^i_k is the kth component (or block) of x at ith step, while x^i_{-k} is all other components of x at the same step



Gibbs=MH with no rejection

$$A = min(1, \frac{p(x^*)}{p(x^i)} \frac{q_k(x^i|x^*)}{q_k(x^*|x^i)})$$

$$p(x^*) = p(x^*_{-k}, x^*_k) = p(x^*_k|x^*_{-k})p(x^*_{-k})$$

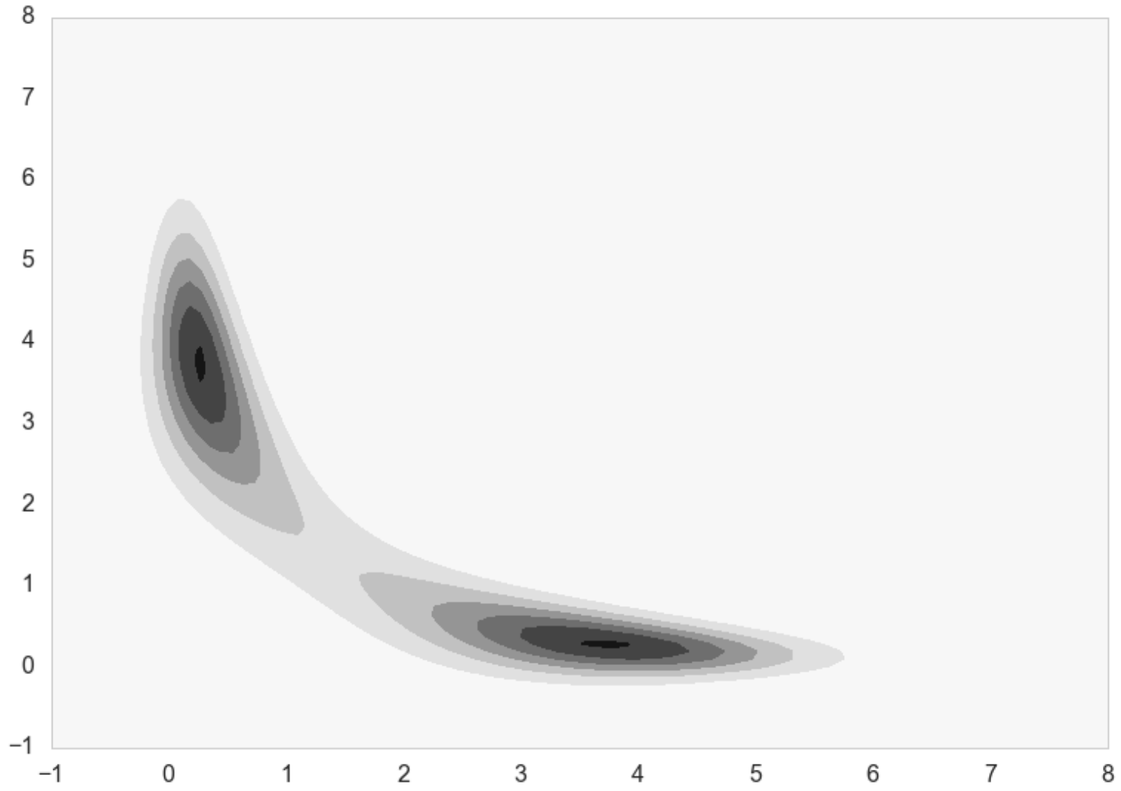
$$A = min(1, \frac{p(x^*_k|x^*_{-k})p(x^*_{-k})}{p(x^i_k|x^i_{-k})p(x^i_{-k})} \frac{q_k(x^i|x^*)}{q_k(x^*|x^i)}) = min(1, \frac{p(x^*_k|x^*_{-k})p(x^*_{-k})}{p(x^i_k|x^i_{-k})p(x^i_{-k})} \frac{p(x^i_k|x^*_{-k})}{p(x^*_k|x^i_{-k})})$$

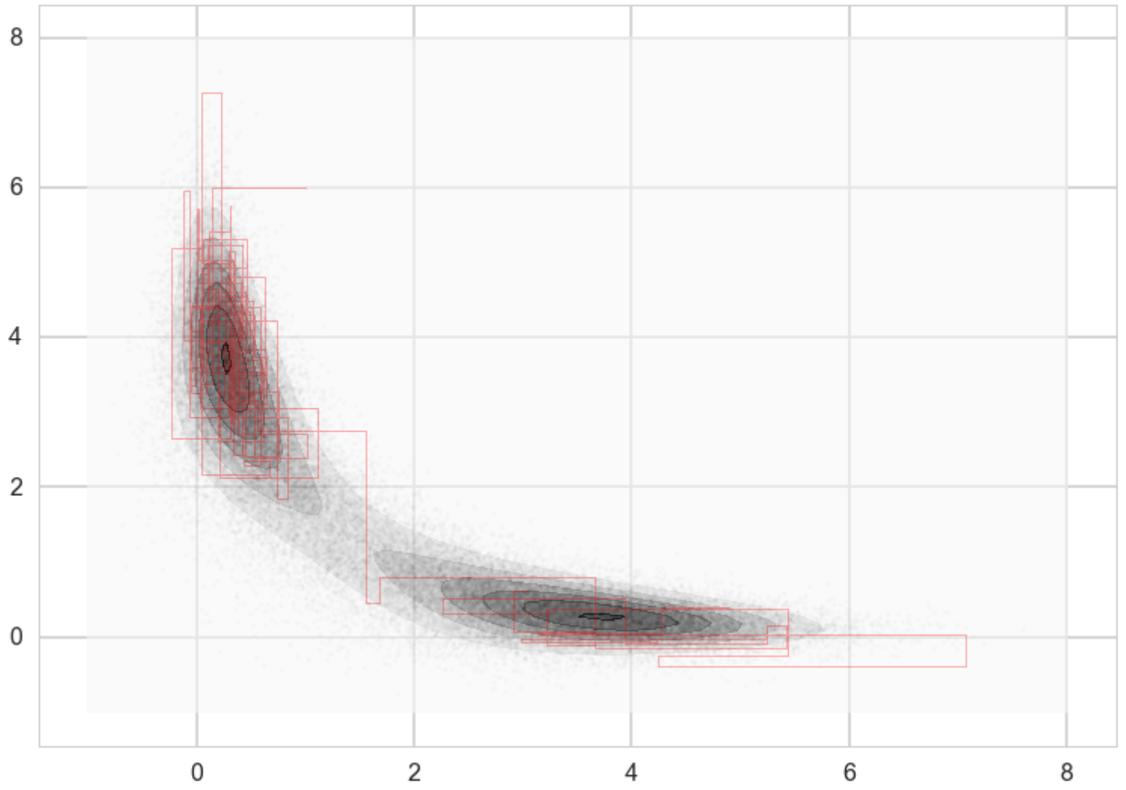
Componentwise update, $\implies x^*_{-k} = x^i_{-k}$ and A is 1!

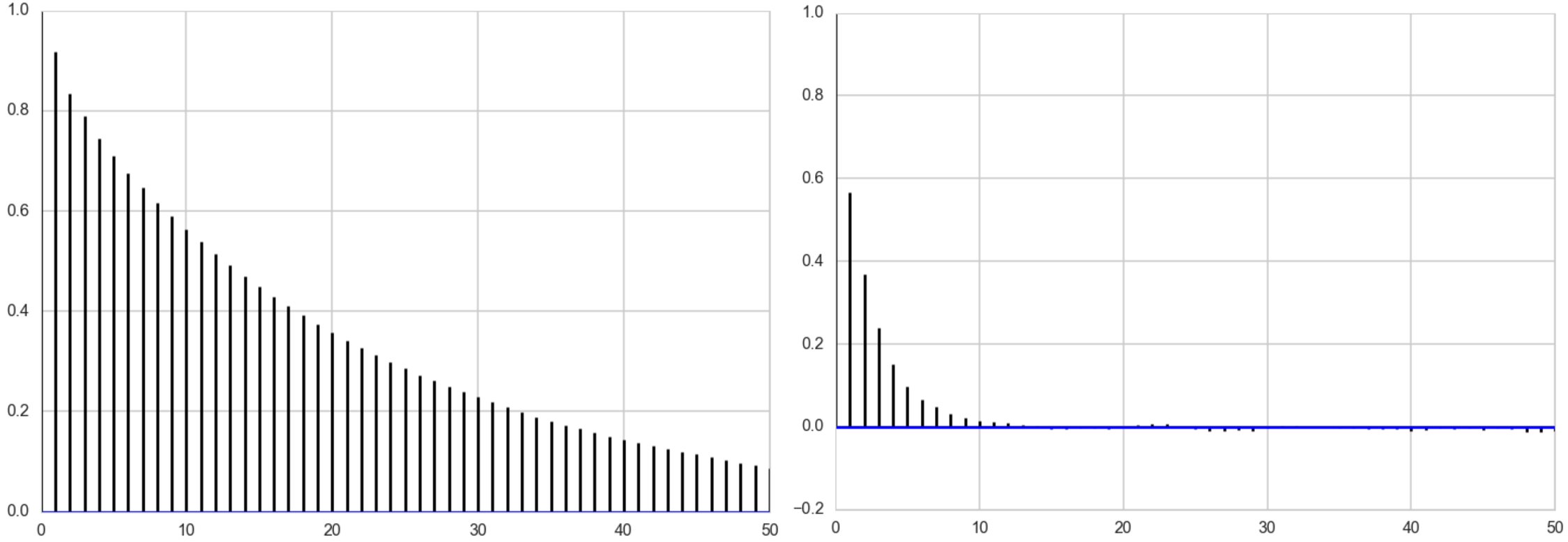
Blockwise Sampling vs Componentwise sampling

- the most general structure is to sample each parameter in turn. This can lead to higher autocorrelation and slower sampling
- may be worth sampling blocks of parameters together using a multivariate proposal
- mix and match as you see fit (see homework)

Tetchy Gibbs







xcorr on left, ycorr on right

Data Augmentation

want to sample a $p(x)$

The difference from Gibbs Sampling: the other variable, say y, is to be treated as latent.

The game is to construct a joint $p(x, y)$ such that we can sample from $p(x|y)$ and $p(y|x)$, and then find the marginal

$$p(x) = \int dy\, p(x, y).$$

Simplest form of a DA algo:

1. Draw $Y \sim p_{Y|X}(.|x)$ and call the observed value y

2. Draw $X_{n+1} \sim p_{X|Y}(.|y)$

3. Histo the X

Usual "Fake News" Example

Sample from $p(x) = e^{-x^2/2}/\sqrt{2\pi}$.

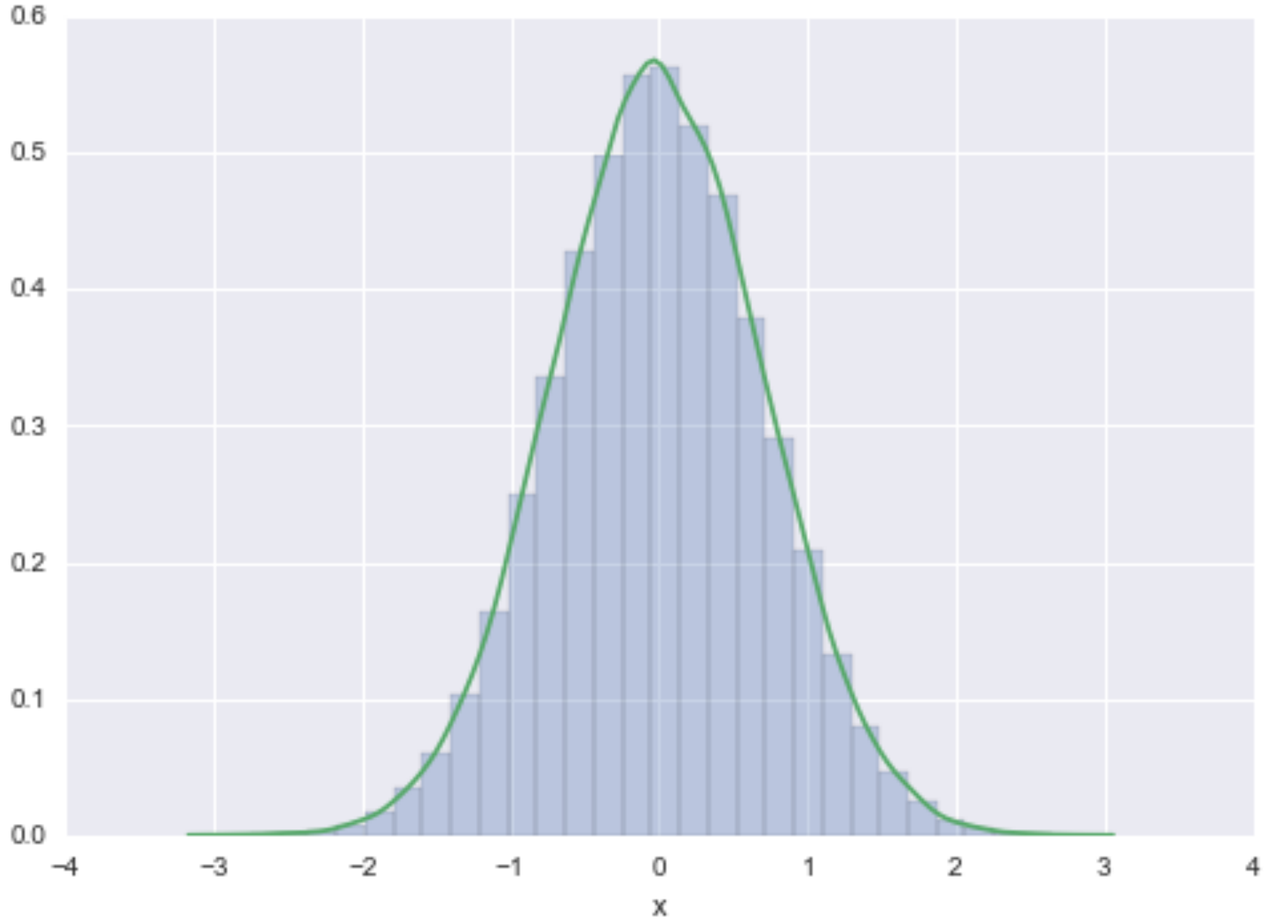
Take $p(x, y) = 1/(\sqrt{2\pi}) \exp\left\{-(x^2 - \sqrt{2}xy + y^2)\right\}$

$$Y|X = x \sim N(x/\sqrt{2}, 1/2) \text{ and } \mathrm{X}|\mathrm{Y} = \mathrm{y} \sim \mathrm{N}(\mathrm{y}/\sqrt{2}, 1/2)$$

The x-marginal is $\propto e^{-x^2/2} \int e^{-(y - x/\sqrt{2})^2} dy$

Example (contd)

```
N=100000
x=np.zeros(N)
x[0] = np.random.rand() # INITIALIZE
for i in np.arange(1,N):
    Y=sp.stats.norm.rvs(x[i-1]/np.sqrt(2), 0.5)
    x[i]=sp.stats.norm.rvs(Y/np.sqrt(2), 0.5)
```



Transition kernel

$h(x', x) = h(x'|x) = \int_Y p(x'|y)\, p(y|x)\, dy$ has stationarity by construction from Gibbs.

Its a probability: $\int h(x'|x)dx' = \int_X \int_Y p(x'|y)\, p(y|x)\, dy\, dx'$

$= \int_Y p(y|x) \left[\int_X p(x'|y)dx' \right] dy = \int_Y p(y|x)dy = 1$

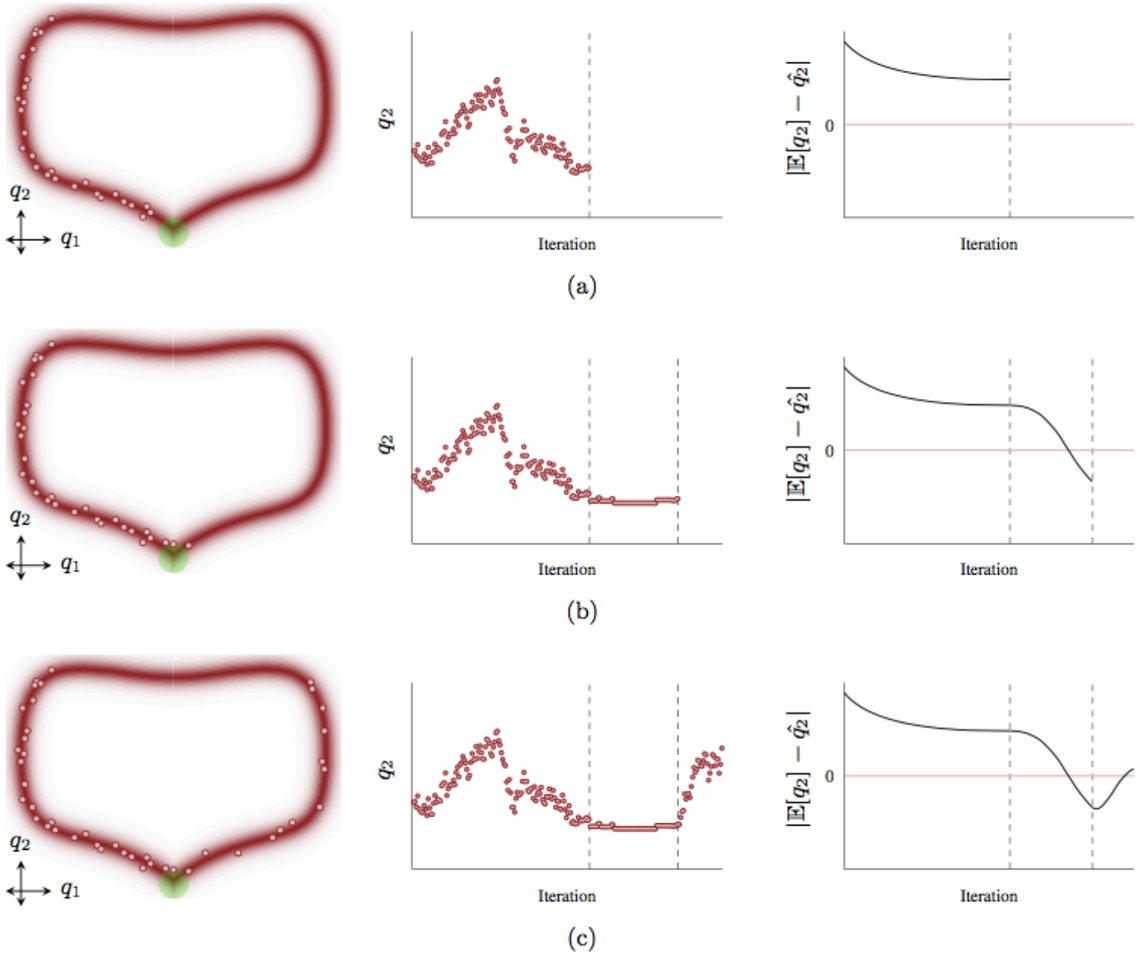
$h(x'|x)\,p(x)$ is symmetric in (x, x'):

$$h(x'|x)\,p(x) = p(x)\int_Y p(x'|y)\,p(y|x)\,dy = \int_Y \frac{p(x',y)\,p(x,y)}{p(y)}\,dy.$$

The rhs is symmetric in (x, x') and so is $h(x'|x)p(x)$.

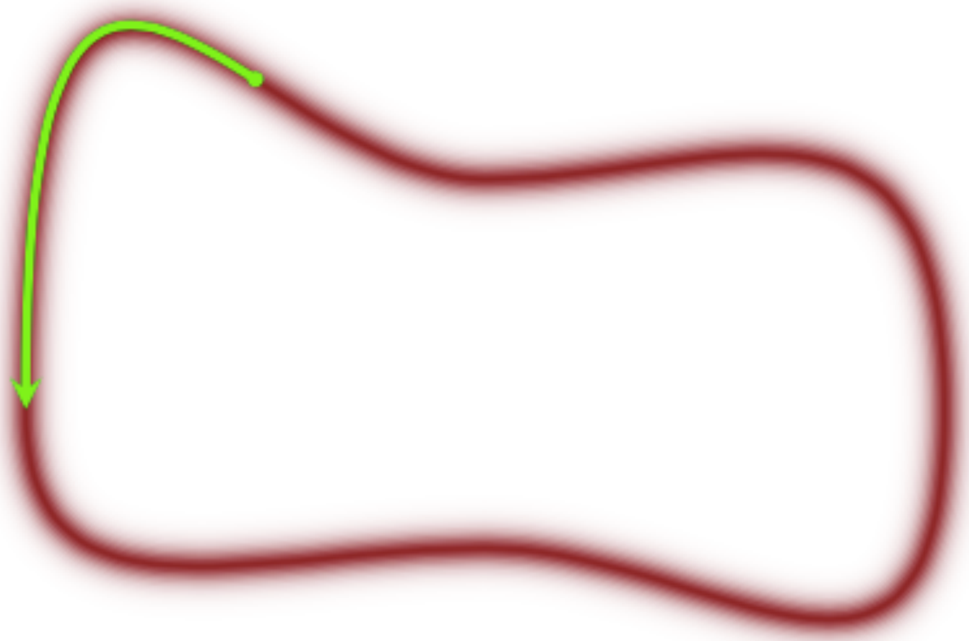
The Markov chain generated with transition probability $h(x'|x)$ is **REVERSIBLE** and thus supports detailed balance.

Problems with Metropolis MCMC



- overshoot and oscillate at pinches
- need to specify step step sizes
- large steps go outside typical set and are not accepted
- small steps accepted but go nowhere
- large correrlations

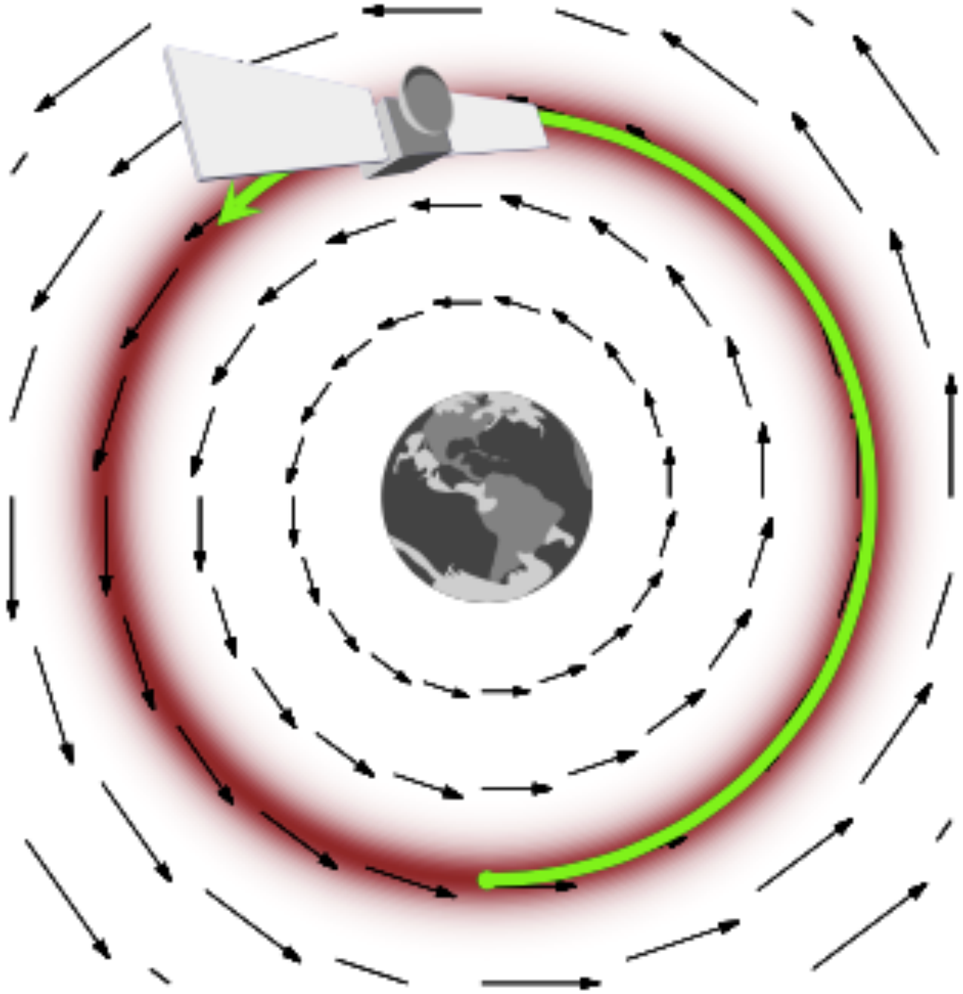
Hamiltonian Monte Carlo



Need a Coherent Glide

- want to cover on $p(q)$ better than a drunkard
- move smoothly on $p(q)$
- instead we will augment with a "momentum" variable p
- try to move smoothly on $p(p, q)$
- and then marginalize:

$$\int dp p(p, q) = \int dp p(p|q)p(q) = p(q)$$



Balance between gravity and momentum in a rocket provides it

Now, like in sampling, let
$p(p,q) = e^{-Energy}$

Carry out an augmentation with an additional momentum with the energy **Hamiltonian**

$$H(p,q) = \frac{p^2}{2m} + V(q)$$

$$p(p,q) = e^{-H(p,q)} = e^{-K(p,q)}e^{-V(q)} = p(p|q)p(q)$$

and thus: $H(p,q) = -log(p(p,q)) = -logp(p|q) - logp(q)$

The choice of a kinetic energy term then is the choice of a conditional probability distribution over the "augmented" momentum which ensures that

$$\int dp p(p,q) = \int dp p(p|q)p(q) = p(q)\int p(p|q)dp = p(q).$$

Canonical distribution

Distribution of a physical system in connection with a heat bath.

Its temperature if thus fixed.

$p(p, q)$ is our canonical distribution

$$p(p, q) = e^{-H(p,q)} = e^{-K(p,q)} e^{-V(q)} = p(p|q)p(q)$$

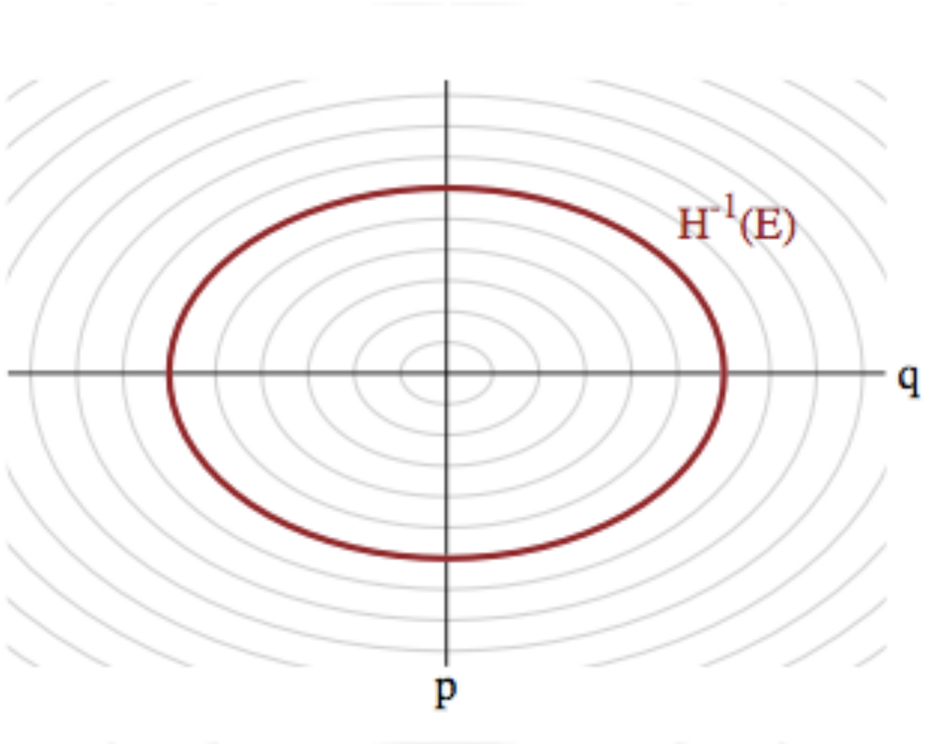
Phase Space level sets

Typical Set decomposes into level sets of constant probability(energy)

The energy **Hamiltonian**

$$H(p,q) = \frac{p^2}{2m} + V(q) = E_i\,,$$

with E_i constants (constant energies) for each level-set foliate and where the **potential energy** $V(q) = -log(p(q))$.



We are looking at level sets of the

Joint distribution

Why do it this way?

Because Hamiltonian flow conserves energy, leading naturally to using level sets and the

Microcanonical distribution

Microcanonical distribution: states for given energy.

Time implicit H: flows **constant energy, vol preserving, reversible**.

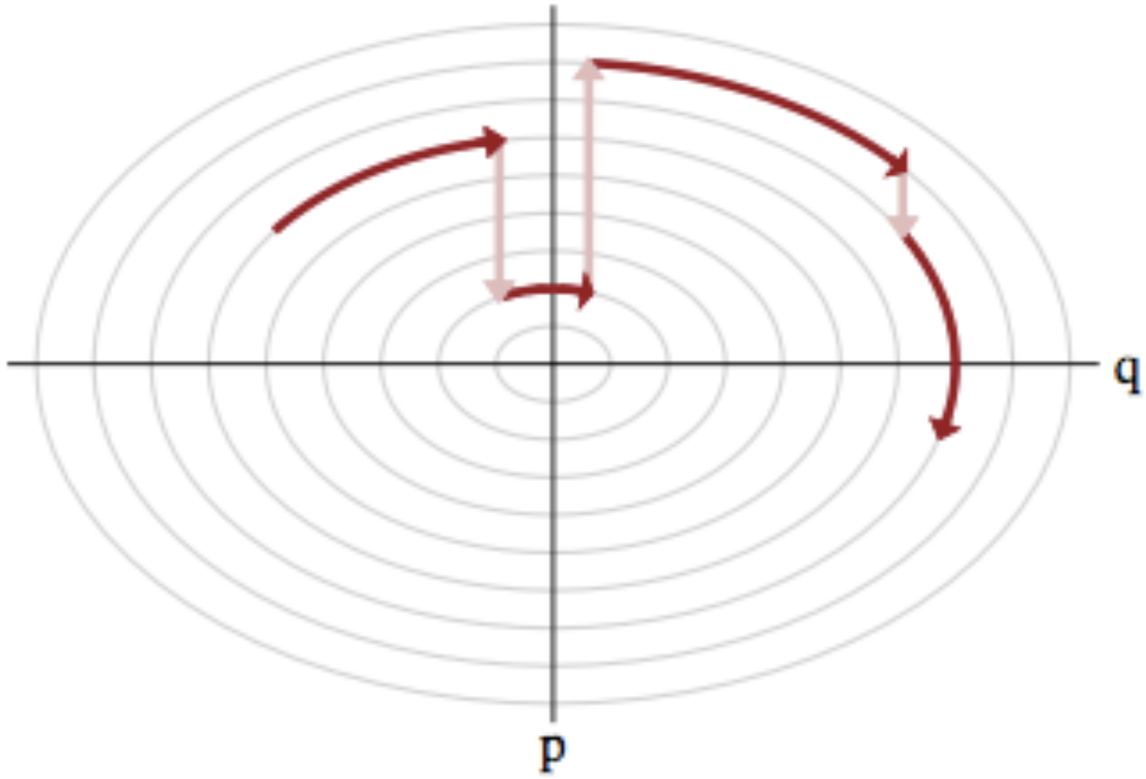
The canonical distribution can be written as a product of this microcanonical distribution and a **marginal energy distribution**:

$$p(q,p) = p(\theta_E|E)p(E)$$

where θ_E indexes the position on the level set.

Also need to sample **Marginal Energy Distrib**: probability of level set in the typical set.

Momentum resampling (thruster fire) moves us between level sets



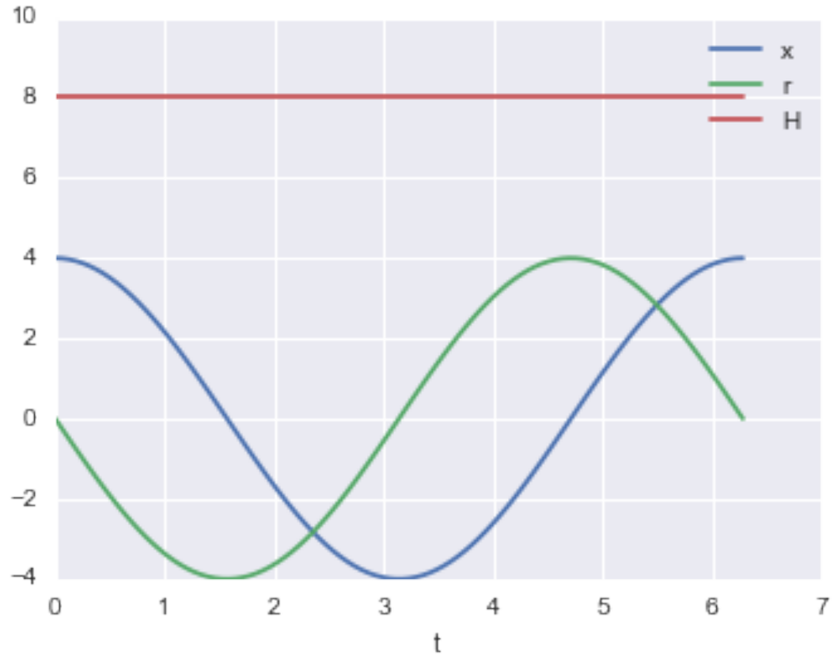
Traverse a level set: Hamiltonian Mechanics

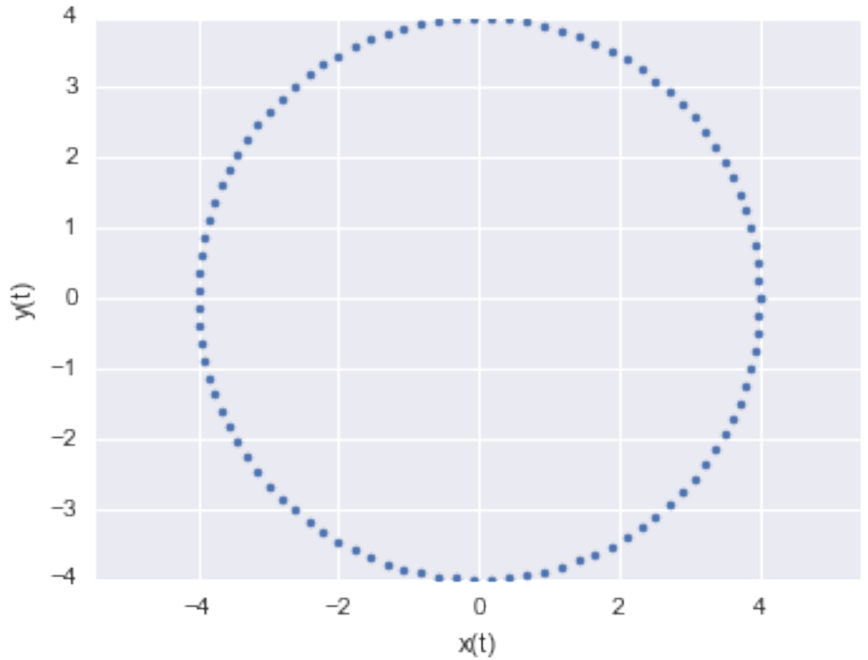
Physics equations of motion in the **Hamiltonian Formalism** set up the "glide" (over a level set).

$$\frac{dp}{dt} = -\frac{\partial H}{\partial q}$$
$$\frac{dq}{dt} = \frac{\partial H}{\partial p}$$

$H = p^2/2m + V(q)$, $\frac{dp}{dt} = -\frac{\partial H}{\partial q} = -\frac{\partial V}{\partial q} = Force$: Newton's law.

Oscillator: an EXACT solution!






```
q_t = lambda t: 4. * np.cos(t)
p_t = lambda t: -4. * np.sin(t)
```

Explicitly time-independent Hamiltonian is conserved

If the Hamiltonian H doesn't have a functional dependence on time we see that

$$\frac{dH}{dt} = \sum_i \left[\frac{\partial H}{\partial q_i} \frac{dq_i}{dt} + \frac{\partial H}{\partial p_i} \frac{dp_i}{dt} \right] + \frac{\partial H}{\partial t}$$

$$\frac{dH}{dt} = \sum_i \left[\frac{\partial H}{\partial q_i} \frac{\partial H}{\partial p_i} + (\frac{\partial H}{\partial p_i})(-\frac{\partial H}{\partial q_i}) \right] + \frac{\partial H}{\partial t}$$

if

$$\frac{\partial H}{\partial t} = 0, \frac{dH}{dt} = 0$$

Then

$$H(t + \Delta t) = H(t) \, \forall t.$$

This time independence is crucial to reversibility: cannot figure which direction equations are being run

Reversibility

T_s from $(q, p) \to (q', p')$ to a "later" time $t' = t + s$. Mapping is 1-1, inverse T_{-s}. This can be obtained by simply negating time:

$$\frac{dp}{d(-t)} = -\frac{\partial H}{\partial q}$$
$$\frac{dq}{d(-t)} = \frac{\partial H}{\partial p}$$

To get reversibility:

If we then transform $p \rightarrow -p$, we have the old equations back:

$$\frac{d(-p)}{d(-t)} = -\frac{\partial H}{\partial q}$$
$$\frac{dq}{d(-t)} = \frac{\partial H}{\partial(-p)}$$

Superman Transform

To reverse T_s, flip the momentum, run Hamiltonian equations T_s until you get back to the original position and momentum in phase space at time t, then flip the momentum again so it is pointing in the right direction.

Volume in phase space is conserved

T_s for small change $s = \delta$ can be written as:

$$T_\delta = \begin{pmatrix} q \\ p \end{pmatrix} + \delta \begin{pmatrix} \frac{dq}{dt} \\ \frac{dp}{dt} \end{pmatrix} + O(\delta^2)$$

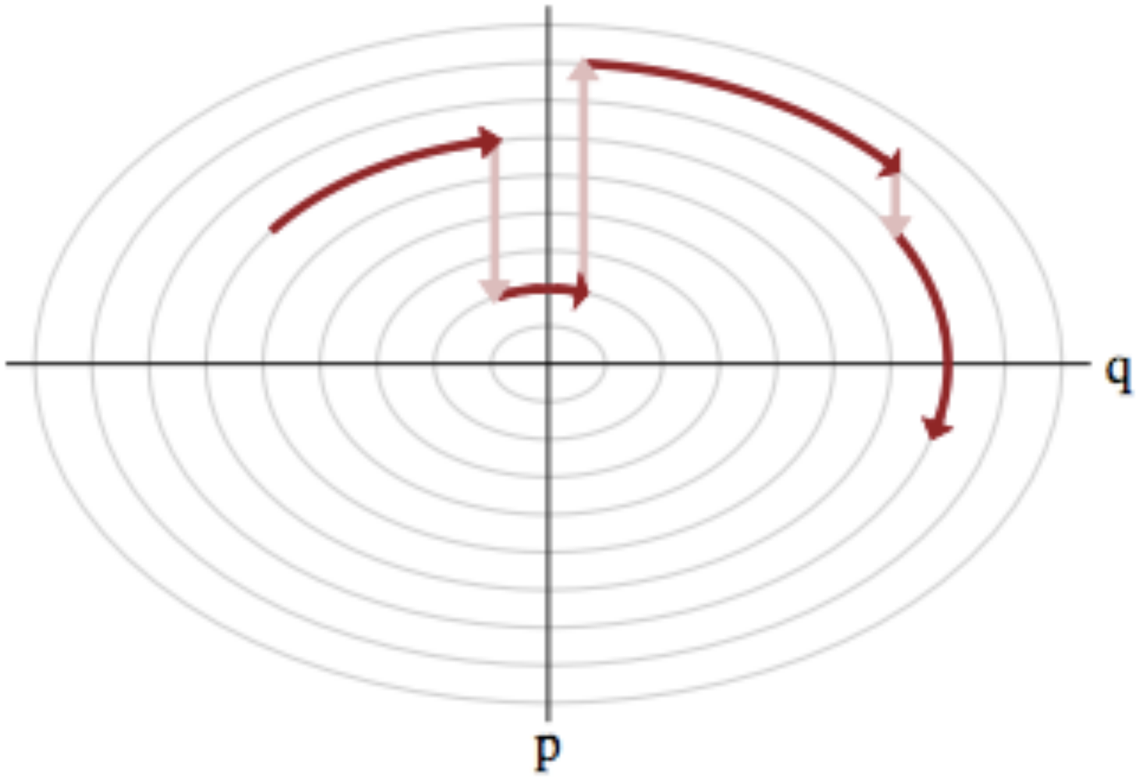
Jacobian:

$$\begin{bmatrix} 1 + \delta \frac{\partial^2 H}{\partial q \partial p} & \delta \frac{\partial^2 H}{\partial p^2} \\ \delta \frac{\partial^2 H}{\partial q^2} & 1 - \delta \frac{\partial^2 H}{\partial p \partial q} \end{bmatrix}$$ and thus the determinant is $1 + O(\delta^2)$.

Thus as our system evolves, any contraction or expansion in position space must be compensated by a respective expansion or compression in momentum space.

As a result of this, the momenta we augment our distribution with must be **dual** to our pdf's parameters, transforming in the opposite way so that phase space volumes are invariant.

Between level sets: Momentum resampling



Draw p from a distribution that is determined by the distribution of momentum, i.e. $p \sim N(0, \sqrt{M})$ for example, and attempt to explore the level sets.

Firing the thruster moves us between level sets!

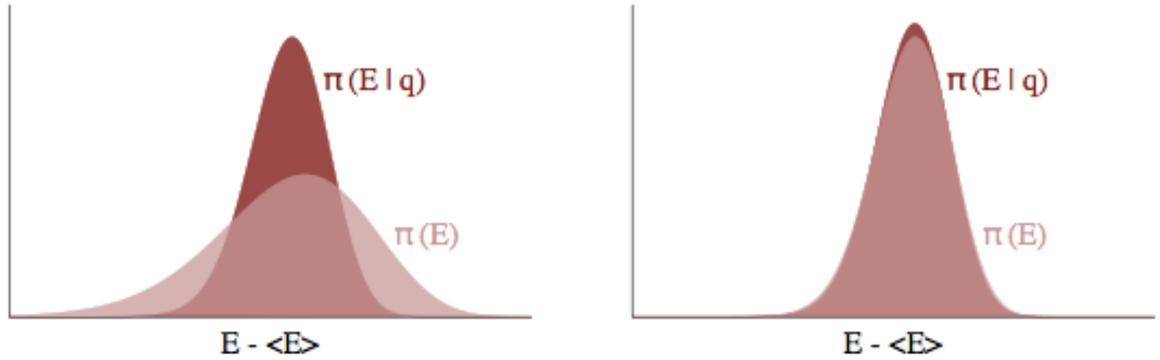
That is, we sample the marginal energy distribution.

Resampling Efficiency

Let $p(E|q)$ as the transition distribution of energies induced by a momentum resampling using $p(p|q) = -log\, K(p, q)$ at a given position q.

If $p(E|q)$ narrow compared to the marginal energy distribution $p(E)$: random walk amongst level sets proceeds slowly.

If $p(E|q)$ matches $p(E)$: independent samples generated from the marginal energy distribution very efficiently.



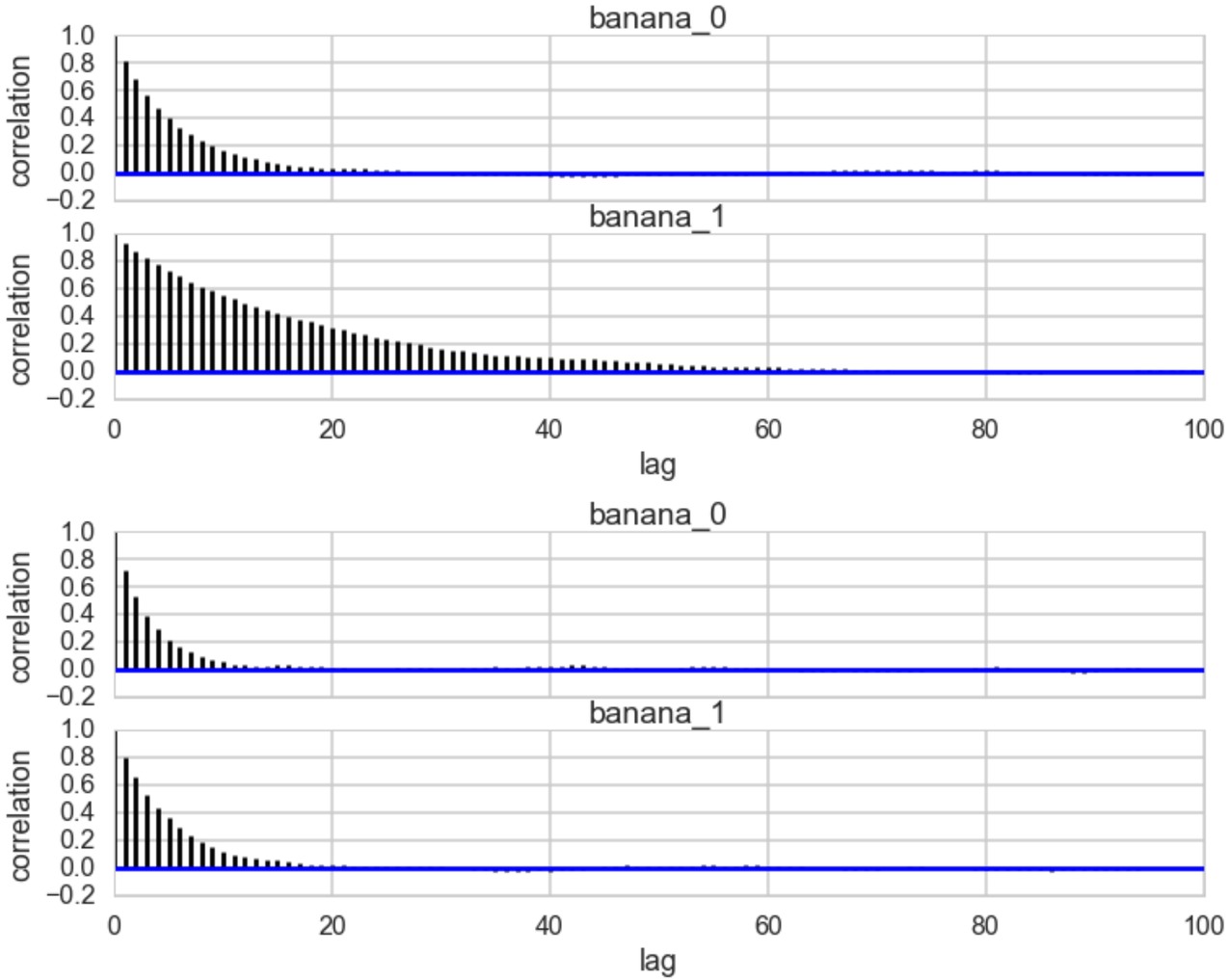
HMC/NUTS in pymc3

```
def clike2(value):
    x = value[0]
    y = value[1]
    val = -100 * (T.sqrt(y**2+x**2)-1)**2 + (x-1)**3 - y -5
    return (val)

with pm.Model() as model:
    banana = pm.DensityDist("custom", clike2, shape=2, testval=[1,1])

with model:
    start = pm.find_MAP()
    stepper=pm.Metropolis()
    trace=pm.sample(100000, step=stepper, start=start)
pm.autocorrplot(trace[20000::5])

with model:
    stepper_nuts=pm.NUTS()
    trace_nuts=pm.sample(100000, step=stepper_nuts)
pm.autocorrplot(trace_nuts[:16000])
```



Basic Idea

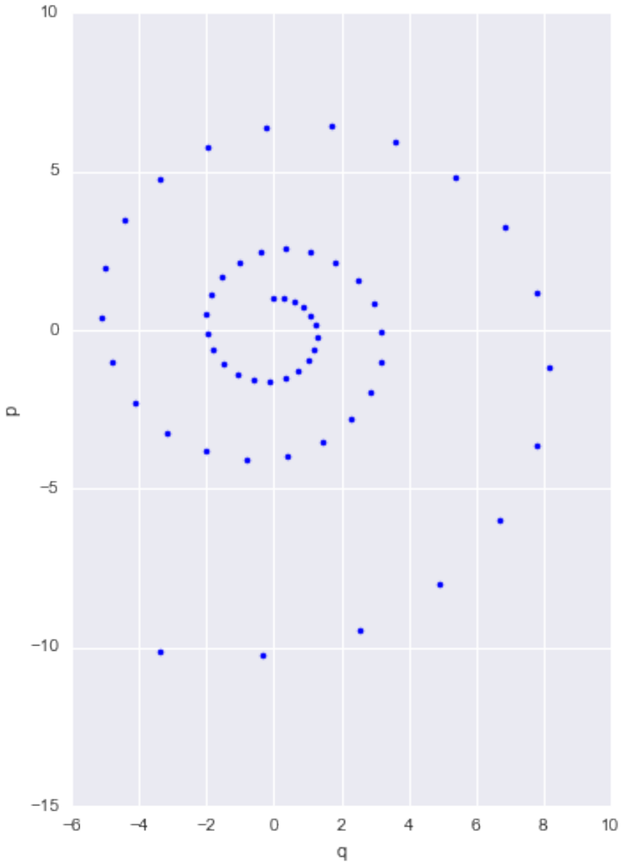
1. Move on a level set of the Hamiltonian $H(p, q)$. Pick up samples at will with acceptance probability 1
2. Fire thrusters, that is sample p, from kinetic energy distribution, to move to another level set
3. Repeat

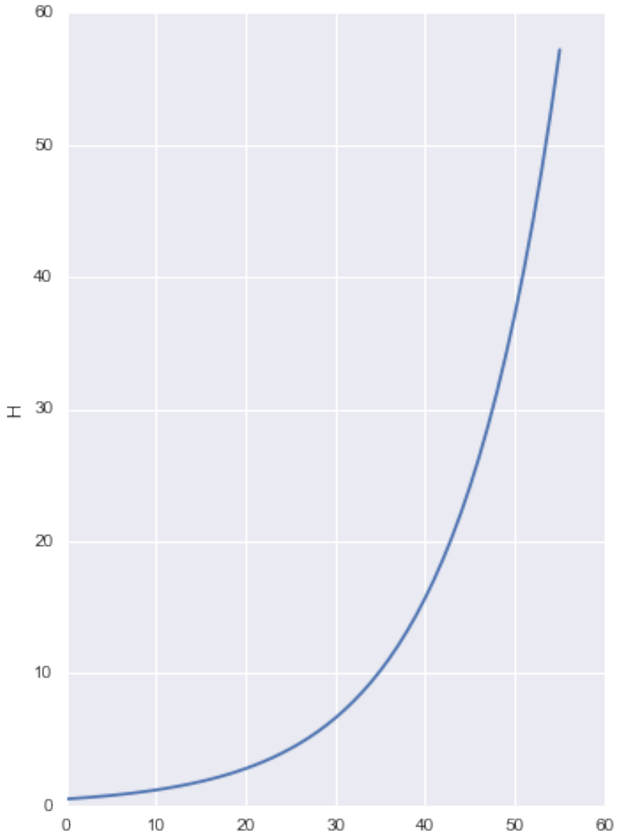
More Problems arise

- how long should we go?
- whats a good kinetic energy?
- discretization to solve differential equations and the need for symplecticity
- lack of reversibility even with symplecticity (we are marginally off the level set)
- this means that thar acceptance probability with no longer be 1

Practical implementation

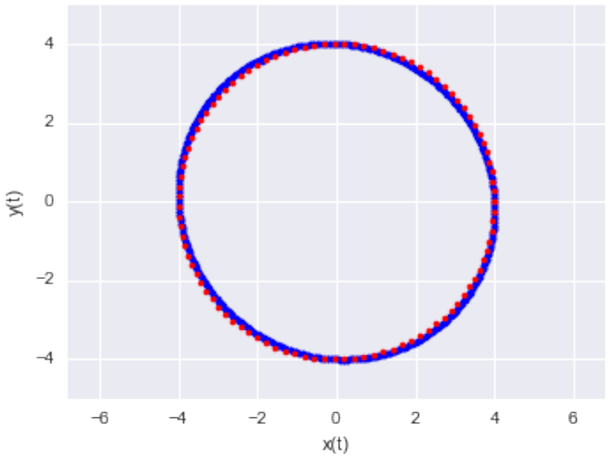
Discretization problems

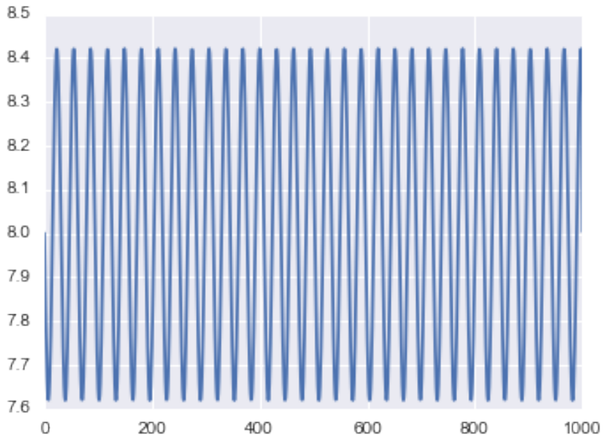




- $p_i(t+\epsilon) = p_i(t) - \epsilon \frac{\partial U}{\partial q_i}|_{q(t)}$
- $q_i(t+\epsilon) = q_i(t) + \epsilon \frac{p_i(t)}{m_i}$
- off-diagonal terms of size ϵ makes volume not preserved
- leads to drift over time
- use "leapfrog" instead

Sympletic Leapfrog





- Only *shear* transforms allowed, will preserve volume.
- $p_i(t+\frac{\epsilon}{2}) = p_i(t) - \frac{\epsilon}{2}\frac{\partial V}{\partial q_i}|_{q(t)}$
- $q_i(t+\epsilon) = q_i(t) + \epsilon\frac{p_i(t+\frac{\epsilon}{2})}{m_i}$
- $p_i(t+\epsilon) = p_i(t+\frac{\epsilon}{2}) - \frac{\epsilon}{2}\frac{\partial V}{\partial q_i}|_{q(t+\epsilon)}$
- still error exists, oscillatory, so reversibility not achieved
- use superman transform. Works even when we are off level set.

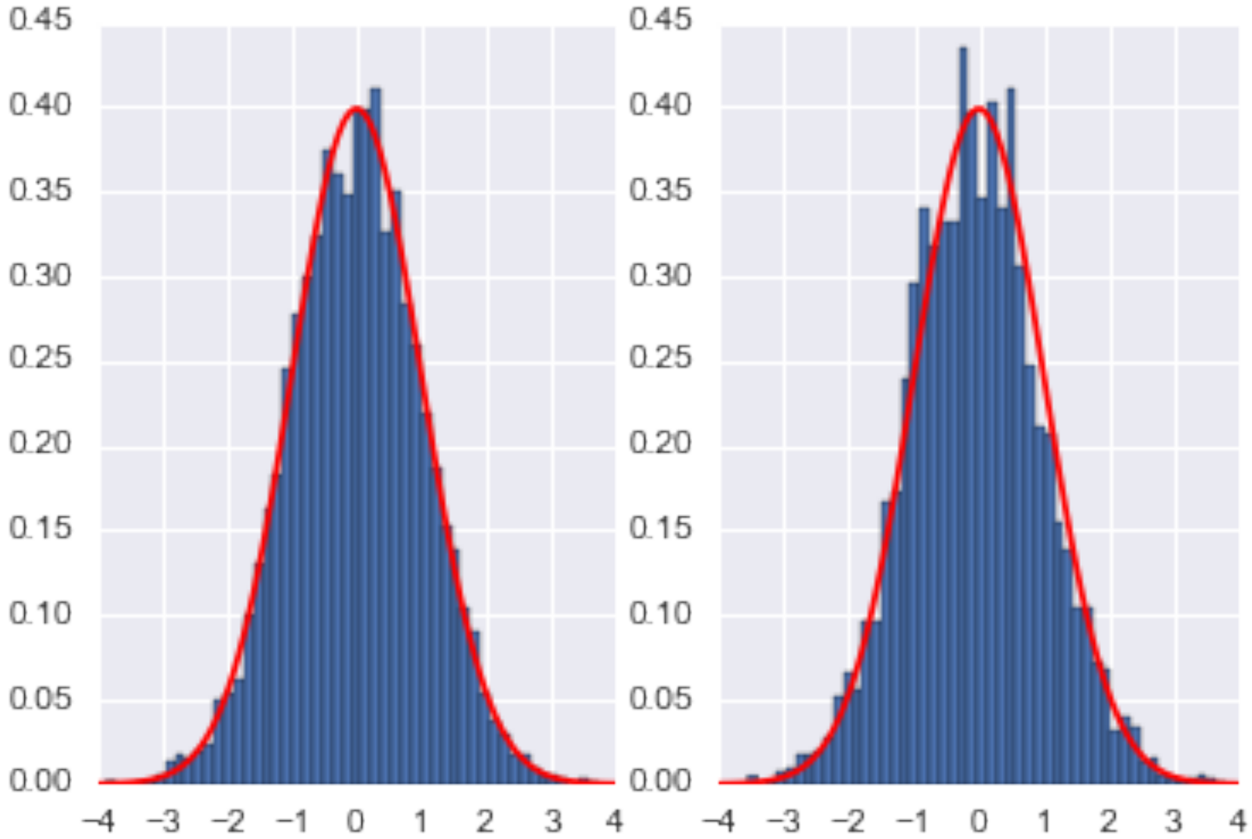
HMC Algorithm

- for i=1:N_samples
  - 1. Draw $p \sim N(0, M)$
  - 2. Set $q_c = q^{(i)}$ where the subscript c stands for current
  - 3. $p_c = p$
  - 4. Update momentum before going into LeapFrog stage: $p^* = p_c - \frac{\epsilon * \nabla U(q_c)}{2}$
  - 5. LeapFrog to get new proposals. For j=1:L (first/third steps together)
    - $q^* = q^* + \epsilon p$
    - if not the last step, $p = p - \epsilon \nabla U(q)$
  - 6. Complete leapfrog: $p = p - \frac{\epsilon \nabla U(q)}{2}$

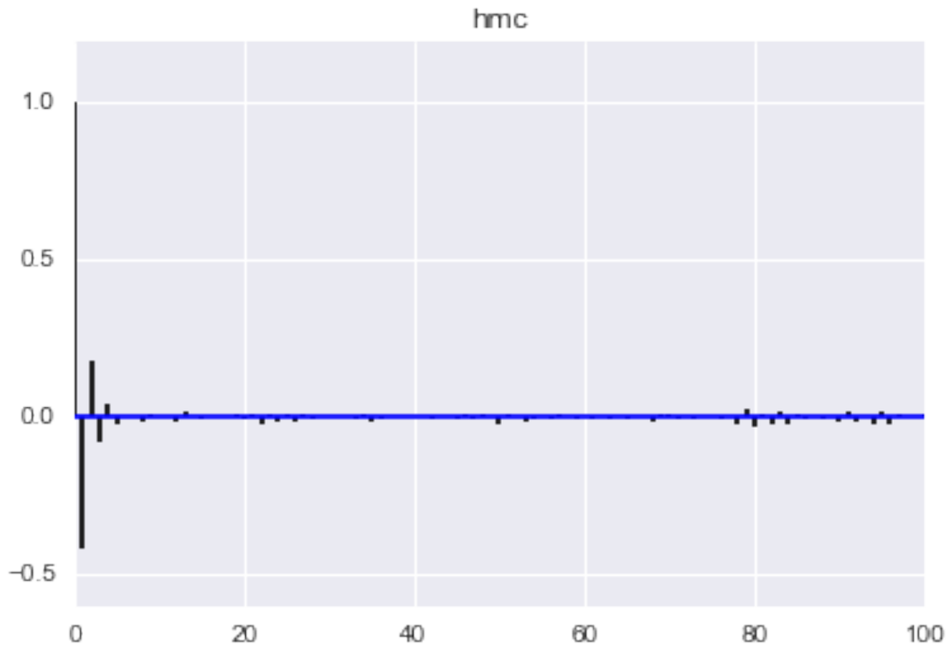
HMC (contd)

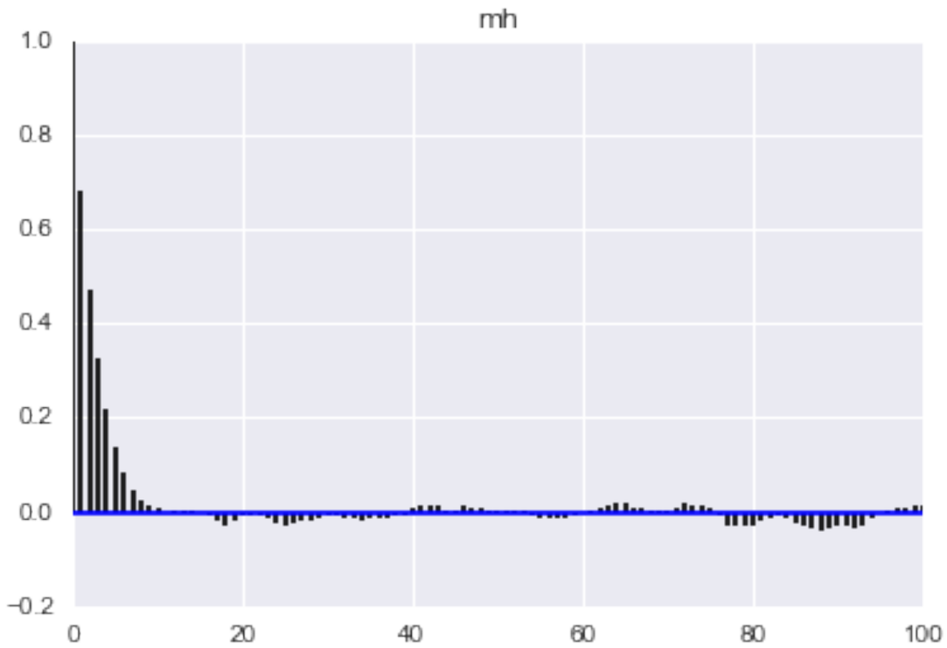
- for i=1:N_samples
  - 7. $p^* = -p$
  - 8. $V_c = V(q_c), \;\; K_c = \frac{p_c^\top M^{-1} p_c}{2}$
  - 9. $V^* = V(q^*), \;\; K^* = \frac{p^{\top *} M^{-1} p^*}{2}$
  - 10. $r \sim \mathbf{Unif}(0,1)$
  - 11. if $r < e^{(U_c - U^* + K_c - K^*)}$
    - accept $q_i = q^*$
    - otherwise reject

```python
def HMC(U,K,dUdq,N,q_0, p_0, epsilon=0.01, L=100):
    current_q = q_0
    current_p = p_0
    H = np.zeros(N)
    qall = np.zeros(N)
    accept=0
    for j in range(N):
        q = current_q
        p = current_p
        #draw a new p
        p = np.random.normal(0,1)
        current_p=p
        # leap frog
        # Make a half step for momentum at the beginning
        p = p - epsilon*dUdq(q)/2.0
        # alternate full steps for position and momentum
        for i in range(L):
            q = q + epsilon*p
            if (i != L-1):
                p = p - epsilon*dUdq(q)
        #make a half step at the end
        p = p - epsilon*dUdq(q)/2.
        # negate the momentum
        p= -p;
        current_U = U(current_q)
        current_K = K(current_p)
        proposed_U = U(q)
        proposed_K = K(p)
        A=np.exp( current_U-proposed_U+current_K-proposed_K)
        # accept/reject
        if np.random.rand() < A:
            current_q = q
            qall[j]=q
            accept+=1
        else:
            qall[j] = current_q
        H[j] = U(current_q)+K(current_p)
    print("accept=",accept/np.double(N))
    return H, qall
```



Autocorrelation: HMC vs MH






```
H, qall= HMC(U=U,K=K,dUdq=dUdq,N=10000,q_0=0, p_0=-4, epsilon=0.01, L=200)

samples_mh = MH_simple(p=P, n=10000, sig=4.0, x0=0)
```