UiO : Matematisk institutt
Det matematisk-naturvitenskapelige fakultet

STK-4051/9051 Computational Statistics Spring 2021
Markov Chain Monte Carlo part 3

Instructor: Odd Kolbjørnsen, oddkol@math.uio.no

Last time

- Code examples MCMC

Requirement for convergence

- Markov chain:
  - is Irreducible: you can visit all of parameter space
  - is Aperiodic : you do not go in loop
  - Is Recurrent : you will always return to a set
  - Has the correct stationary distribution

$$f(\mathbf{y}) = \int_{\mathbf{x}} f(\mathbf{x}) P(\mathbf{y}|\mathbf{x}) d\mathbf{x}$$

Detailed balance:

$$f(\boldsymbol{y}) P(\boldsymbol{x}|\boldsymbol{y}) = f(\boldsymbol{x}) P(\boldsymbol{y}|\boldsymbol{x})$$

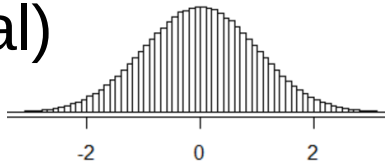
Sufficient for stationary distribution

No guarantee for the other three

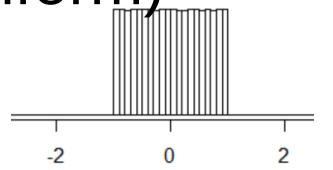
Error in independence sampler

Example 1: Independence sampler:

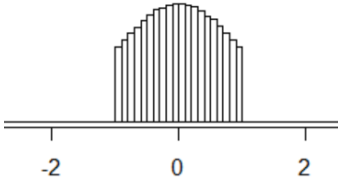
- Target: $f(x) = \phi(x; 0,1^2)$ (standard normal)



- Proposal: $g(x) = 0.5 \quad \text{for } -1 < x \leq 1$ (uniform)



- Result: $p_L(x) = \frac{\phi(x;0,1^2)}{\Phi(1)-\Phi(-1)}$ for $-1 < x \leq 1$ (truncated)



- Your proposal does not allow you to visit outside the interval: $-1 < x \leq 1$ **irreducible** fail

Gibbs sampler failure, not irreducible

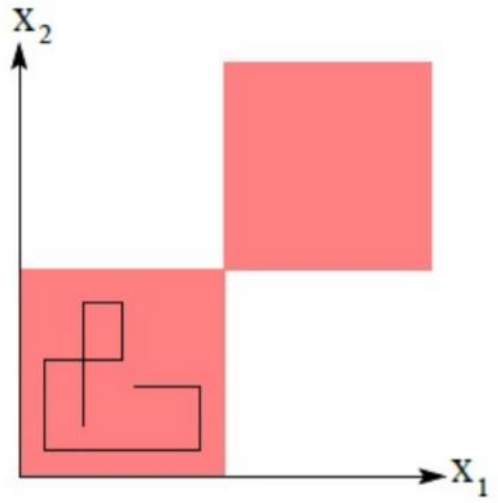
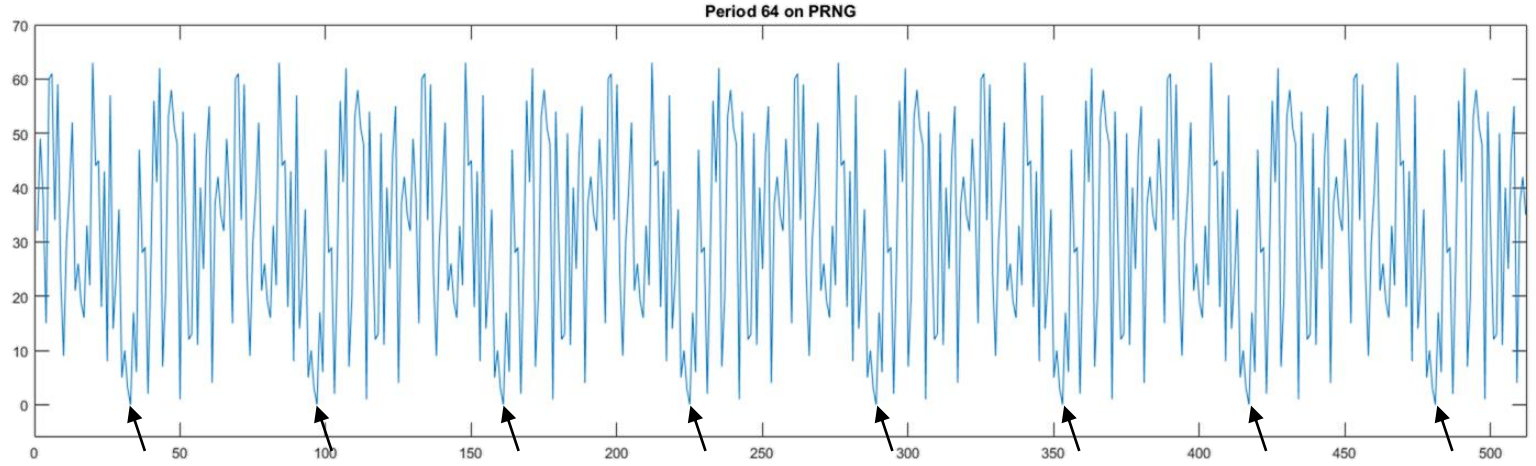


Figure 27.5 (Taken from Barber's *Bayesian Reasoning and Machine Learning*): A two dimensional distribution for which Gibbs sampling fails. The distribution has mass only in the shaded quadrants. Gibbs sampling proceeds from the l^{th} sample state (x_1^l, x_2^l) and then sampling from $p(x_2|x_1^l)$, which we write (x_1^{l+1}, x_2^{l+1}) where $x_1^{l+1} = x_1^l$. One then continues with a sample from $p(x_1|x_2 = x_2^{l+1})$, etc. If we start in the lower left quadrant and proceed this way, the upper right region is never explored.

MCMC and Bayesian Modeling(2017), Martin Haugh Columbia University (under resources on course page)

When does a MCMC fail periodicity?

- Rare in continuous chains, avoided by construction
- PRNG with a short period may cause a periodicity failure
  - $x_{n+1} = (ax_n + c) \bmod m$

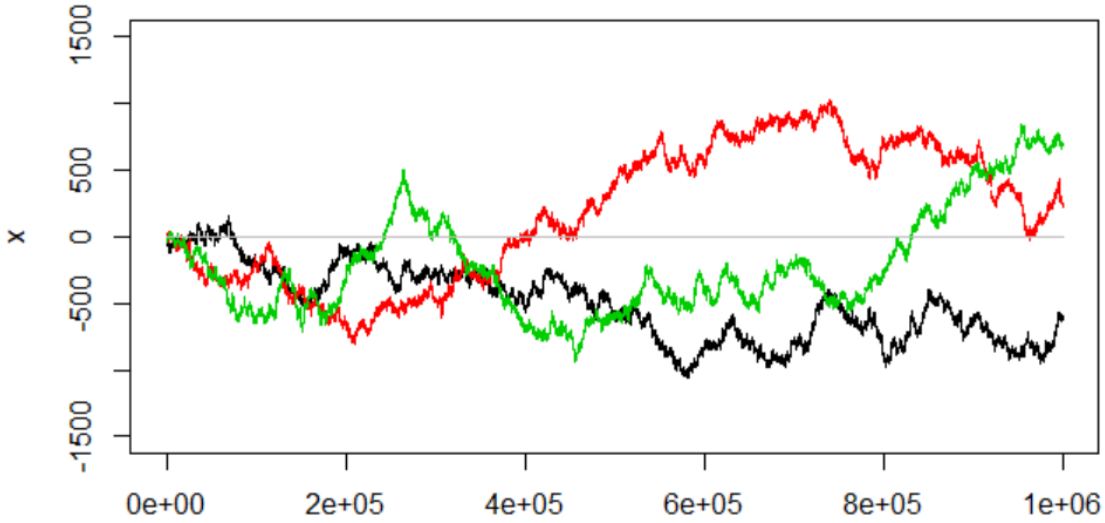


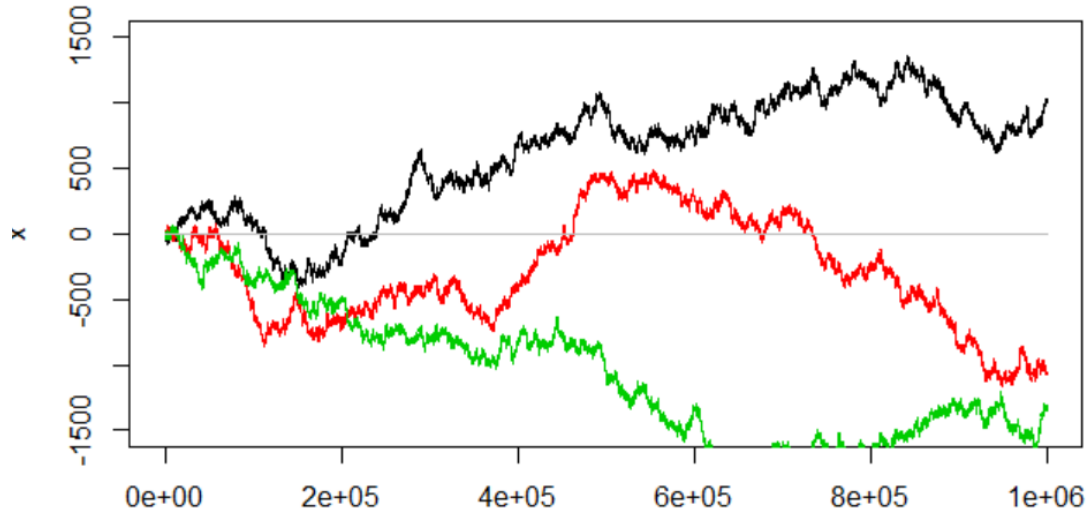
=> Use Mersenne Twister (or another modern PRNG)

Example recurrent fail : improper prior

- $f(\boldsymbol{x}) \propto 1, \boldsymbol{x} = (x_1, x_2, x_3) \in R^3$
- Random walk
- $p(\boldsymbol{x}^*|\boldsymbol{x}) = \phi(x_1^*; x_1, 1) \cdot \phi(x_2^*; x_2, 1) \cdot \phi(x_3^*; x_3, 1)$
- Irreducible? (possible to reach any point with a finite number of steps)
  - Yes, there is a positive probability for any set of non-zero measure in one step.
- Aperiodic?
  - Yes, any non zero set can be reached at any time
- Detailed balance?
  - Yes we have $p(\boldsymbol{x}^*|\boldsymbol{x})f(\boldsymbol{x})=p(\boldsymbol{x}|\boldsymbol{x}^*)f(\boldsymbol{x}^*)$
- So what could go wrong??
  - The chain is not recurrent

Example random walk in R^3





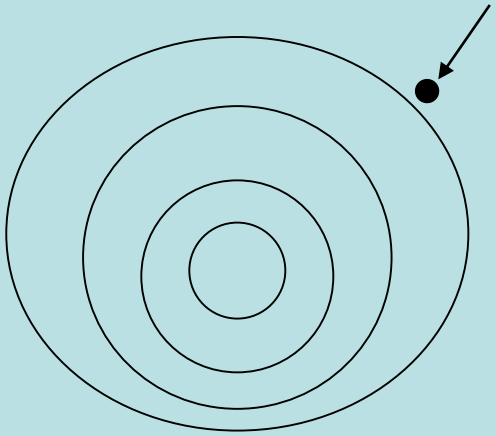
If you get sample paths like these, you might have a recurrence issue

Perhaps your target distribution is not a proper distribution
[not easy to tell upfront]

If you safe guard yourself against zero density regions by setting a minimum density value.
[you get into trouble]

Reccurent fail

- Since we often work with log density a probability of zero causes problems. A quick fix could be to allow the probability to be slightly positive everywhere.
  This is not a good solution
  - Having a small probability for everything gives problems ☹
    => Mc fail to be recurrent



If you get
out here and dimension
is larger than 2 chances
are that you will never
return to «central part»

Example where we it is easy to overlook detailed balance (and it matters)

- Target: $f(x) = 0.5 \quad \text{for } -1 < x \leq 1$ (uniform)
- Proposal: $g(x^*|x) = \phi(x^*; x, \sigma(x)^2)$

$$\sigma(x) = \max(1 - |x|, 0.1)$$

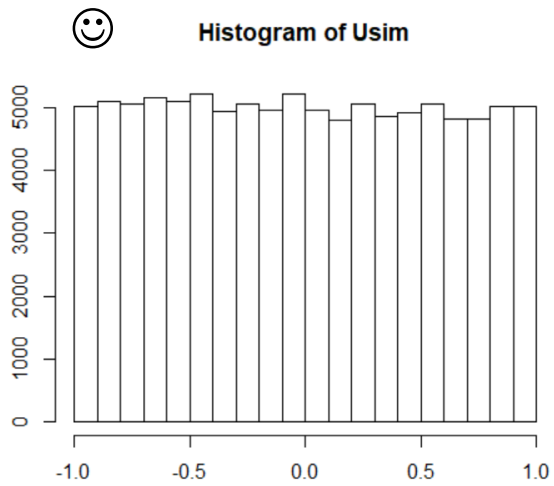
Want to avoid many proposals outside the interval

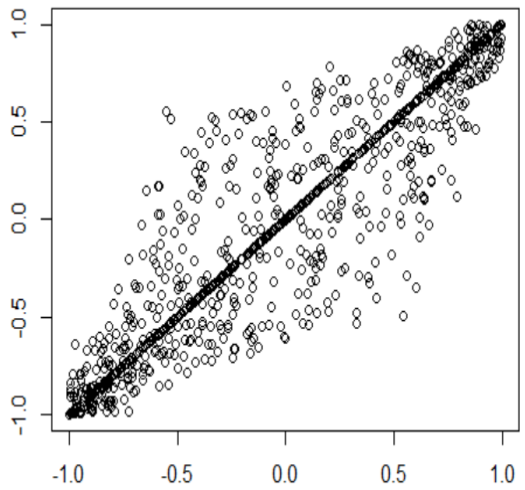
- MH-Ratio:

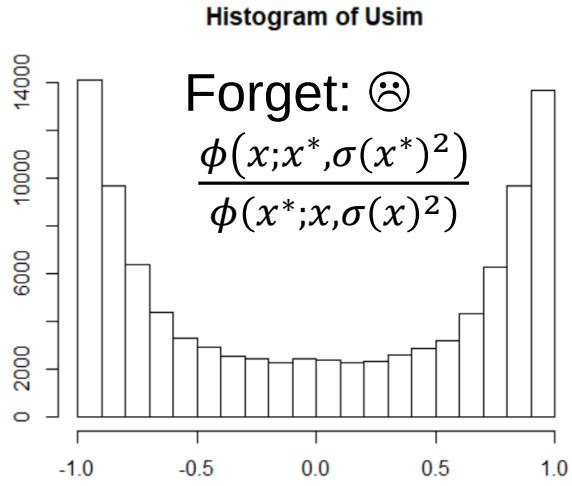
$$R(x^*|x) = \frac{f(x^*)\phi(x; x^*, \sigma(x^*)^2)}{f(x)\phi(x^*; x, \sigma(x)^2)}$$

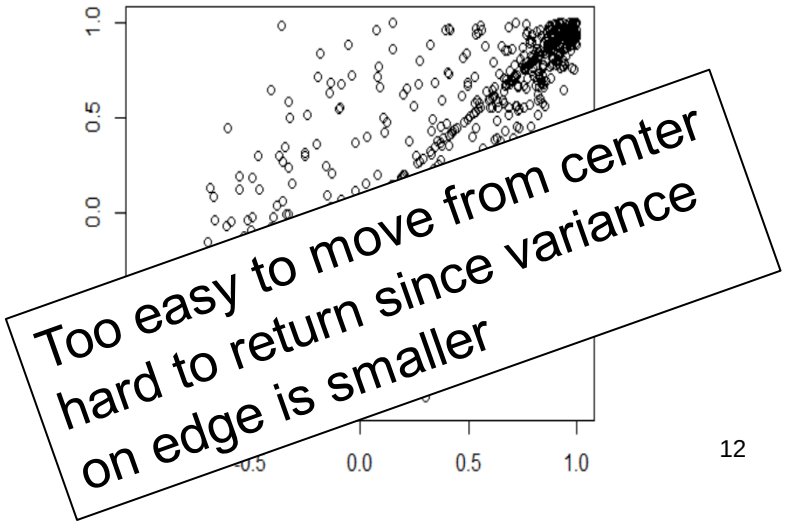
Classic mistake - forget: $\frac{\phi(x;x^*,\sigma(x^*)^2)}{\phi(x^*;x,\sigma(x)^2)}$

Results with and without error:









Advanced topics in MCMC

- Adaptive MCMC: Automatic tuning of proposal distributions (ex 36)
- Multiple-Try M-H (improve griddy Gibbs)
- Slice sampler
- Simulated tempering
- Reversible Jump MCMC (model selection)
- Langevin
- Hamiltonian [Central for STAN]

Adaptive MCMC

- **Want:** Automatic tuning of proposal distributions
  (To get a black box algorithm- universal inference)

- **Main challenge:** Specifying proposal based on history of chain breaks down the Markov property

- **Solution:** Reduce the amount of tuning as the number of iterations increase

- **Example:** Tune the variance in proposal distribution per component in random walk to get 44% acceptance rate

Multiple-Try M-H

- **Want:** A standard M-H approach gives one proposal (which might be bad), if we could propose many and «select» the best
- **Solution:**
  - Generate k proposals $X_1^*, \dots, X_k^*$ from $g(\cdot\,|x^{(t)})$
  - Select X_j^* with probability

$$w(x^{(t)}, X_j^*) = f(x^{(t)}) \cdot g(X_j^*\,|x^{(t)}) \cdot \lambda(x^{(t)}, X_j^*)$$

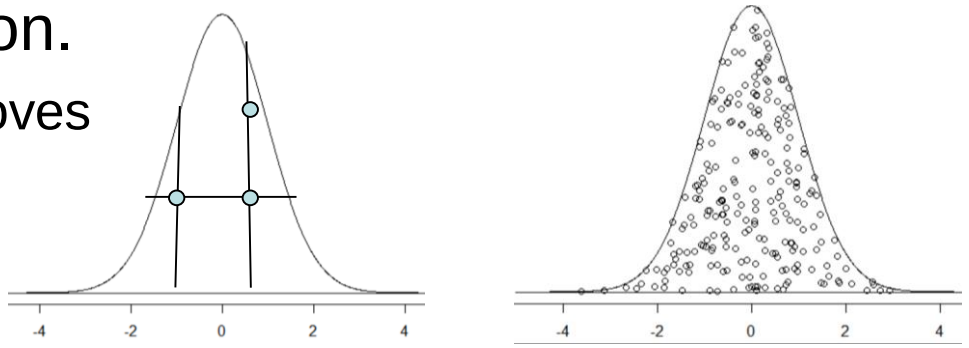
$\lambda(x, y)$ symmetric

  - Reverse:
  - Generate $k-1$ reverse $X_1^{**}, \dots, X_{k-1}^{**}$ from $g(\cdot\,|X_j^*)$,
  - Put $X_k^{**} = x^{(t)}$
  - Use Generalized M-H ratio

$$R_g = \frac{\sum_{i=1}^{k} w(x^{(t)}, X_i^*)}{\sum_{i=1}^{k} w(X_j^*, X_i^{**})}$$

Slice sampler

- **Want:** Show that sampling from any distribution can be reduced to sampling from a uniform distribution (on a restricted domain)
- **Solution:** Sample the uniform distribution obtained by the image of the distribution.
  - Alternate horizontal moves
  - Vertical moves



- Multivariate distributions using one-dimensional conditional
- Do not require sampling of the conditional distributions only need to evaluate them
- Can cope with multiple modes if not too far apart

Simulated tempering

- **Want:** to bypass "low probability" regions
- **Solution:**
  - Define $f^i(x) \propto f(x)^{1/\tau_i}$, $1 = \tau_1 < \tau_2 < \cdots \tau_m$
  - Simulate pair (X, I), where I changes distribution

$$(x, i) \propto f^i(x)p(i)$$

Chain: (x_k, i_k), $k = 1, \ldots, N$ with

$$f^i(x) \propto \exp\left(\frac{\log f(x)}{\tau_i}\right)$$

Parallell to simulated anealing, but we stop at $\tau_i = 1$, and can go back up

Inference:

1) Use subset where $i_k = 1$

2) Importance weigthing can be used to include all samples

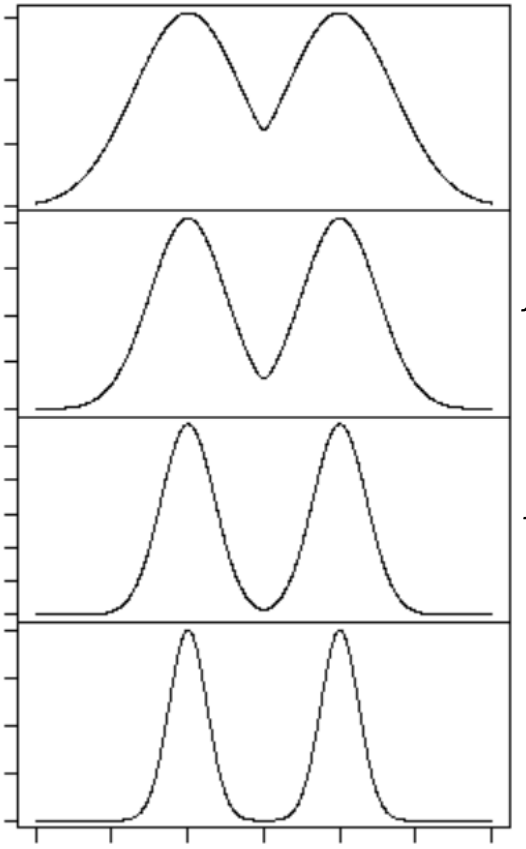
Simulated tempering

- Alternate changes to x and i
- Change x by standard M-H when currently at level i

$$R_i(x^*|x) = \frac{f^i(x^*)g(x|x^*)}{f^i(x)g(x^*|x)}$$

- Proposal for i (simple) $g(i^*|i)$:
  - On the edge move deterministic
  - In center 50-50 increase or decrease
- M-H Ratio

$$R_{ST}(i^*|i,x) = \frac{f^{i^*}(x)\cdot p(i^*)\cdot g(i|i^*)}{f^i(x)\cdot p(i)\cdot g(i^*|i)}$$



$$g(2|1) = 1$$
$$g(k \neq 2|1) = 0$$
$$g(M-1|M) = 1$$
$$g(k \neq M-1|M) = 0$$
$$g(|k-j| = 1|\, j) = \frac{1}{2}$$
$$g(|k-j| \neq 1|\, j) = 0$$

Simulated tempering

- Select $p(i)$ such that the algorithm spends equal time at each level
- Lowest level is the target distribution
- Highest level is the set such that we can move freely
- Number of levels:
  - Too high we use too much time climbing up and down
  - Too low we do not get acceptance moving between layers

Reversible Jump MCMC

- Assume several models $M_1, \ldots, M_K$
- Corresponding parameters $\theta_1, \ldots, \theta_K$
  off different dimensions and different interpretations!
- RJMCMC is a M-H method for moving between spaces of different dimensions
- **Want to:** Simulate pairs $X = (M, \theta_M)$

- **Main challenge:**
  - When changing $M \rightarrow M^*$, how to propose $\theta^*_{M^*}$,
  - construct a reversible chain using auxiliary variables

- **Example:** Bayesian model selection

Langevin M-H

- **Want:** To avoid symmetric proposal rules which propose moves into low probability areas and giving low acceptance rates

- **Solution:** Changes the proposal distribution of the MH algorithm to favor proposals in the direction of the maximum gradient of the target density. thus moving the chains towards the high density regions of the distribution

$$g(\boldsymbol{x}^*|\boldsymbol{x}) \sim N(\boldsymbol{x}^*, \boldsymbol{x} + \boldsymbol{d}_x, \sigma^2) \qquad \boldsymbol{d}_x = \left(\frac{\sigma^2}{2}\right) \frac{\partial \log f(\boldsymbol{u})}{\partial \boldsymbol{u}}\bigg|_{\boldsymbol{u}=\boldsymbol{x}}$$

- The proposal density depends on the location of the current sample and this is not symmetric

$$R(\boldsymbol{x}^*|\boldsymbol{x}) = \frac{f(\boldsymbol{x}^*) \cdot g(\boldsymbol{x}|\boldsymbol{x}^*)}{f(\boldsymbol{x}) \cdot g(\boldsymbol{x}^*|\boldsymbol{x})}$$

- When $g(\boldsymbol{x}^*|\boldsymbol{x})$ favors high probability the $g(\boldsymbol{x}|\boldsymbol{x}^*)$ tend to be low, so we do not get the full benefit [“reverse penalty”]

Hamiltonian MC

- **Want:** To avoid symmetric proposal rules which propose moves into low probability areas and giving low acceptance rates, but want to avoid the “reverse penalty” from Langevin

- **Solution:** Adding a momentum term **p** to each component of the target variable **q** (auxiliary variable = **p**) ,
  - Helps the chain move more rapidly though the target distribution
  - It favors successive proposals in the same direction, allowing the simulation to move rapidly through the space
  - Update target **q** and and moment **p** simultaneously .
  - We couple the proposal of the target variables **q** and the auxiliary variables **p** such that a low probability of the target variable is countered by a high probability of the auxiliary variables [avoid “reverse penalty”]
  - Interpreted as a trade off between potential and kinetic energy

Hamiltonian MC

- Common trick in Monte Carlo: Introduce auxiliary variables
- Hamiltonian MC (Neal et al., 2011):

$$\pi(\boldsymbol{q}) \propto \exp(-U(\boldsymbol{q}))$$

Distribution of interest

$$\pi(\boldsymbol{q}, \boldsymbol{p}) \propto \exp(-U(\boldsymbol{q}) - 0.5\boldsymbol{p}^T\boldsymbol{p})$$

Extended distribution

$$= \exp(-H(\boldsymbol{q}, \boldsymbol{p}))$$

$$H(\boldsymbol{q}, \boldsymbol{p}) = U(\boldsymbol{q}) + 0.5\boldsymbol{p}^T\boldsymbol{p}$$

«potential energy»

«kinetic energy»

Momentum

- Note
  - $\boldsymbol{q}$ and $\boldsymbol{p}$ are independent
  - $\boldsymbol{p} \sim N(\boldsymbol{0}, \boldsymbol{I})$.
  - Usually $\dim(\boldsymbol{p}) = \dim(\boldsymbol{q})$

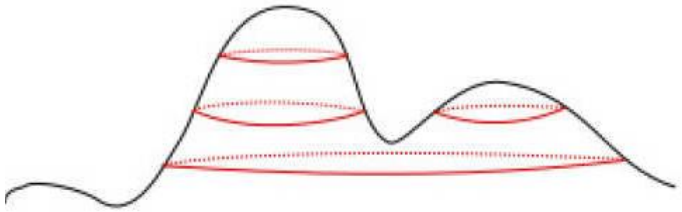


Figure from: https://www.imperial.ac.uk/inference-group/projects/monte-carlo-methods/hamiltonian-monte-carlo/

Hamiltonian MC Algorithm

- Algorithm ($\boldsymbol{q}$) current value
  1. Simulate $\boldsymbol{p} \sim N(\boldsymbol{0}, \boldsymbol{I})$
  2. Generate $(\boldsymbol{q}^*, \boldsymbol{p}^*)$ such that $H(\boldsymbol{q}^*, \boldsymbol{p}^*) \approx H(\boldsymbol{q}, \boldsymbol{p})$
  3. Accept $(\boldsymbol{q}^*, \boldsymbol{p}^*)$ by a Metropolis-Hastings step
- Main challenge: Generate $(\boldsymbol{q}^*, \boldsymbol{p}^*)$

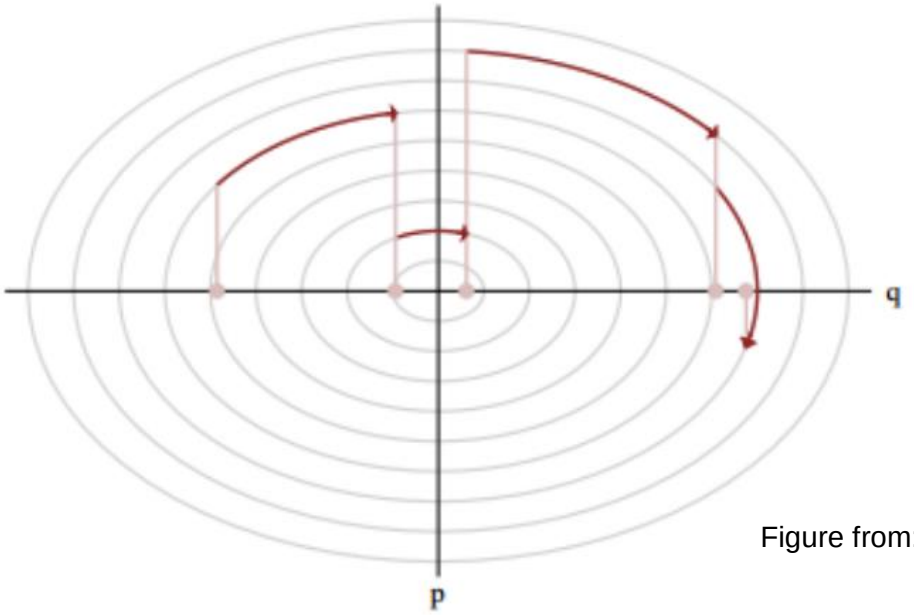


Figure from: https://mucrn.missouri.edu/assets/docs/stanintro_0.pdf

Hamiltonian dynamics preserve energy

- Consider $(\boldsymbol{q}, \boldsymbol{p})$ as a time-process $(\boldsymbol{q}(t), \boldsymbol{p}(t))$
- Hamiltonian dynamics: Change through

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}$$
$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}$$

«Conservation of energy»
Potential to kinetic

This gives

$$\frac{dH}{dt} = \sum_{i=1}^{d} \left[\frac{\partial H}{\partial q_i} \frac{dq_i}{dt} + \frac{\partial H}{\partial p_i} \frac{dp_i}{dt} \right]$$
$$= \sum_{i=1}^{d} \left[\frac{\partial H}{\partial q_i} \frac{\partial H}{\partial p_i} - \frac{\partial H}{\partial p_i} \frac{\partial H}{\partial q_i} \right] = 0$$

Need to solve equations
for the Hamiltonian Dynamics
by numerical scheeme

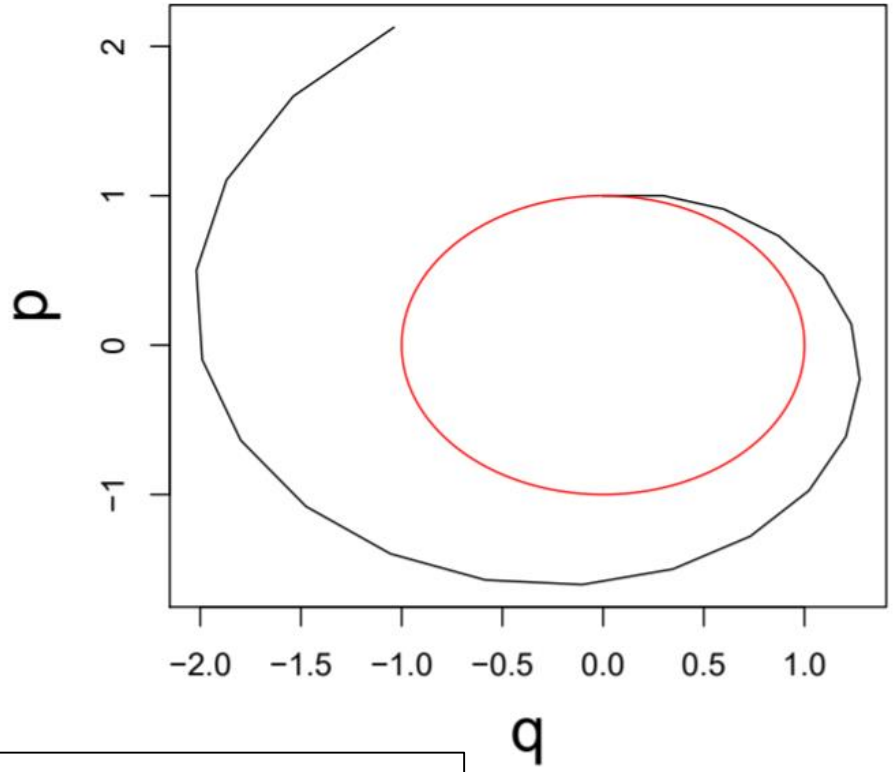
- If we can change $(\boldsymbol{q}, \boldsymbol{p})$ exactly by the Hamiltonian dynamics, H will not change!
- In practice, only possible to make numerical approximations

Hamiltonian dynamics - Eulers method

- Assume

$$\begin{aligned} p_i(t+\varepsilon) &= p_i(t) + \varepsilon \frac{dp_i}{dt}(t) \\ &= p_i(t) - \varepsilon \frac{\partial U}{\partial q_i}(q_i(t)) \\ q_i(t+\varepsilon) &= q_i(t) + \varepsilon \frac{dq_i}{dt}(t) \\ &= q_i(t) + \varepsilon p_i(t) \end{aligned}$$

- Note: Derivatives of $U(\boldsymbol{q})$ are used.
- However, not very exact.



$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}$$
$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}$$

$$H(\boldsymbol{q}, \boldsymbol{p}) = U(\boldsymbol{q}) + 0.5\boldsymbol{p}^T\boldsymbol{p}$$

Negative log-density

$$\pi(\boldsymbol{q}) \propto \exp(-U(\boldsymbol{q}))$$

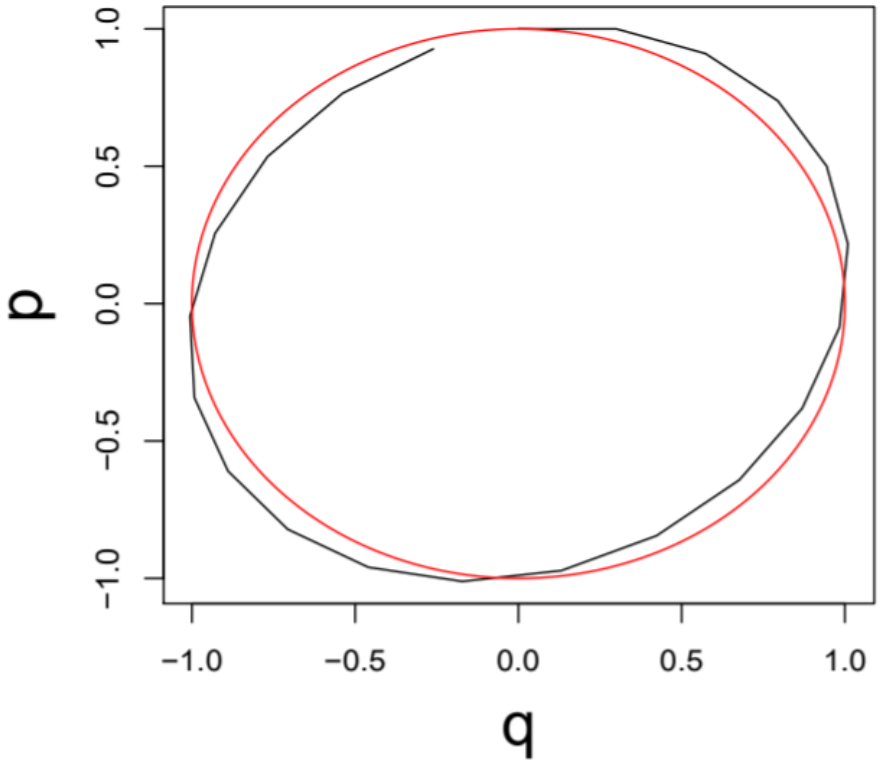
Hamiltonian dynamics - the modified Eulers method

- Assume

$$p_i(t+\varepsilon) = p_i(t) - \varepsilon \frac{\partial U}{\partial q_i}(q(t))$$

$$q_i(t+\varepsilon) = q_i(t) + \varepsilon p_i(t+\varepsilon)$$

- Better than Eulers method.



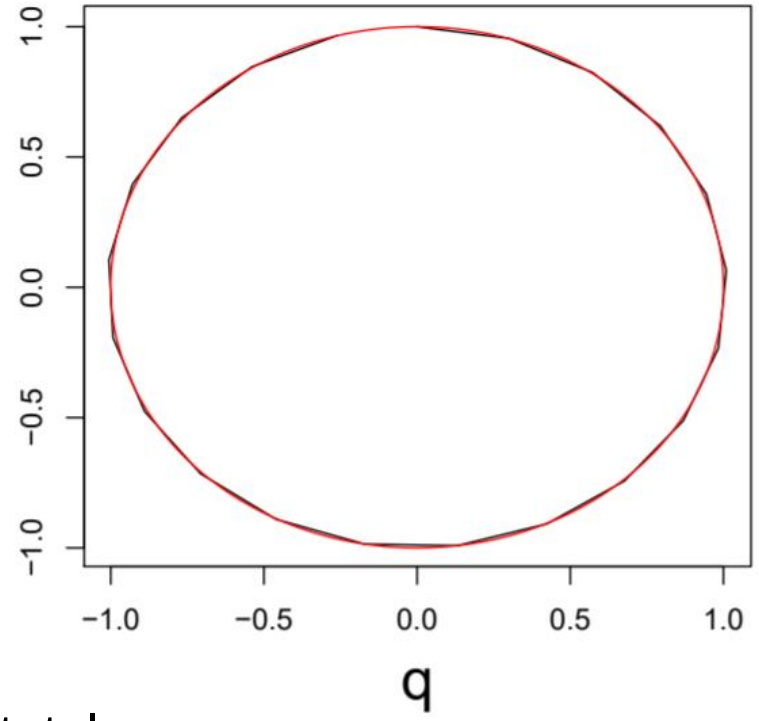
Hamiltonian dynamics – the Leapfrog method

- Assume

$$p_i(t+\tfrac{\varepsilon}{2}) = p_i(t) - \frac{\varepsilon}{2}\frac{\partial U}{\partial q_i}(q(t))$$

$$q_i(t+\varepsilon) = q_i(t) + \varepsilon p_i(t+\tfrac{\varepsilon}{2})$$

$$p_i(t+\varepsilon) = p_i(t+\tfrac{\varepsilon}{2}) - \frac{\varepsilon}{2}\frac{\partial U}{\partial q_i}(q_(t+\varepsilon))$$



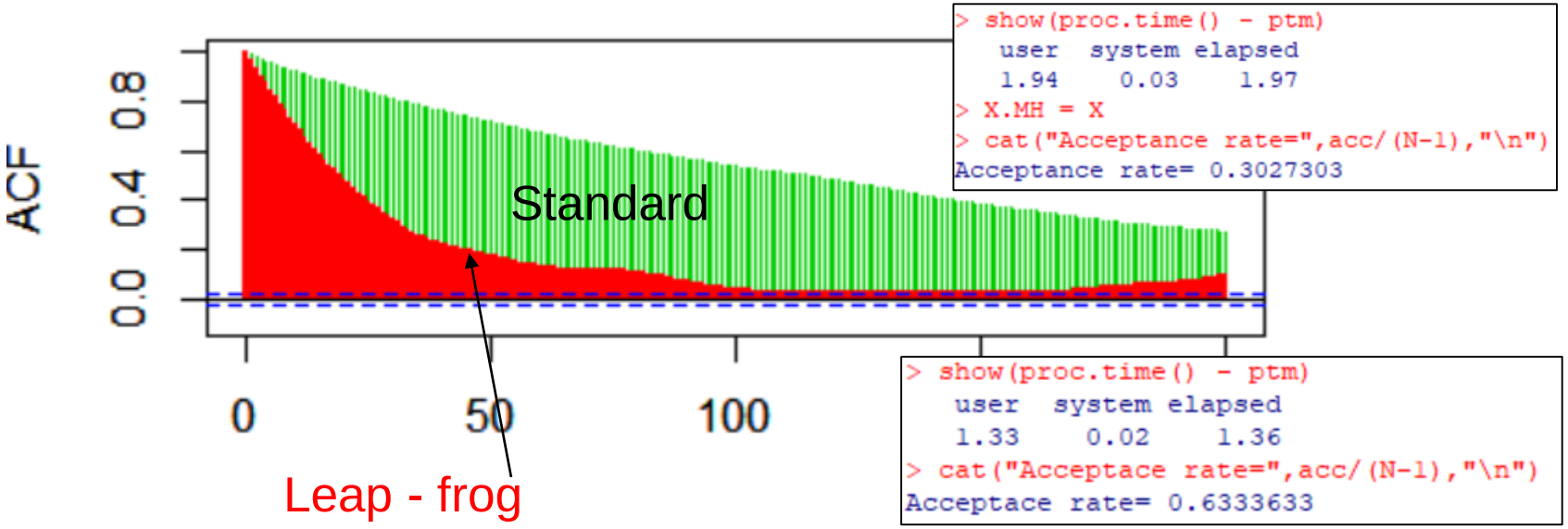
- Quite exact!
- Idea: Use this *L* steps

By solving this excact we get the inverse proposal to be identical and we keep the total energy (in practice still need to correct for approaximation)

$$R(\boldsymbol{x}^*|\boldsymbol{x}) = \frac{f(\boldsymbol{x}^*)\cdot g(\boldsymbol{x}|\boldsymbol{x}^*)}{f(\boldsymbol{x})\cdot g(\boldsymbol{x}^*|\boldsymbol{x})} = \frac{\exp(-H(\boldsymbol{q}^*,\boldsymbol{p}^*))g(\boldsymbol{q},\boldsymbol{p}|\boldsymbol{q}^*,\boldsymbol{p}^*)}{\exp(-H(\boldsymbol{q},\boldsymbol{p}))g(\boldsymbol{q}^*,\boldsymbol{p}^*|\boldsymbol{q},\boldsymbol{p})} = \frac{\exp(-H(\boldsymbol{q}^*,\boldsymbol{p}^*))}{\exp(-H(\boldsymbol{q},\boldsymbol{p}))}$$

Example - 2-dimensional Gaussian

- Assume $\boldsymbol{x} \sim N(\mathbf{0}, \boldsymbol{\Sigma})$, $\boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0.95 \\ 0.95 & 1 \end{pmatrix}$
- $H(\boldsymbol{x}, \boldsymbol{p}) = 0.5\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + 0.5\boldsymbol{p}^T\boldsymbol{p}$
- Use $L = 5$ leapfrog steps, with stepsize $\varepsilon = 0.1$
- `leapfrog_Gauss2.R`



```r
#Leapfrog
Hfunc = function(x,p)
{
  0.5*sum(x*solve(Sigma,x))+0.5*sum(p^2)
}

dU = function(x)
{
  solve(Sigma,x-mu)
}
```

```r
eps = 0.2
L = 5
N = 10000
X = matrix(nrow=N,ncol=2)
X[1,] = rnorm(2)
acc = 0
ptm <- proc.time()
for(i in 2:N)
{
  p.prop = rnorm(2)
  H = Hfunc(X[i-1,],p.prop)
  x.prop = X[i-1,]
  for(j in 1:L)
  {
    p.half = p.prop - 0.5*eps*dU(x.prop)
    x.prop = x.prop + eps*p.half
    p.prop = p.half - 0.5*eps*dU(x.prop)
  }
  H.prop = Hfunc(x.prop,p.prop)
  R = exp(H-H.prop)
  show(c(X[i-1,],x.prop,H,H.prop,R))
  if(runif(1)<R)
  {
    X[i,] = x.prop
    acc = acc + 1
  }
  else
    X[i,] = X[i-1,]
}
```

Sets the kinetic energy

Conservation of energy

Metropolis Hastings

Example - mixture Gaussians

- Assume

$$\pi(x) = pN(x; \mu_1, \sigma_1^2) + (1-p)N(x; \mu_2, \sigma_2^2)$$

- $H(\boldsymbol{x}, \boldsymbol{p}) = -log(\pi(x)) + 0.5\boldsymbol{p}^T\boldsymbol{p}$

```
distr = function(x)
{
  pr*dnorm(x,mu[1],sig[1]) + (1-pr)*dnorm(x,mu[2],sig[2])
}
```

```
dU = function(x)
{
  d1 = dnorm(x,mu[1],sig[1],log=TRUE)
  d2 = dnorm(x,mu[2],sig[2],log=TRUE)
  dr = exp(d2-d1)
  r = (pr*(x-mu[1])/sig[1]^2 + (1-pr)*dr*(x-mu[2])/sig[2]^2)/(pr+(1-pr)*dr)
}
```

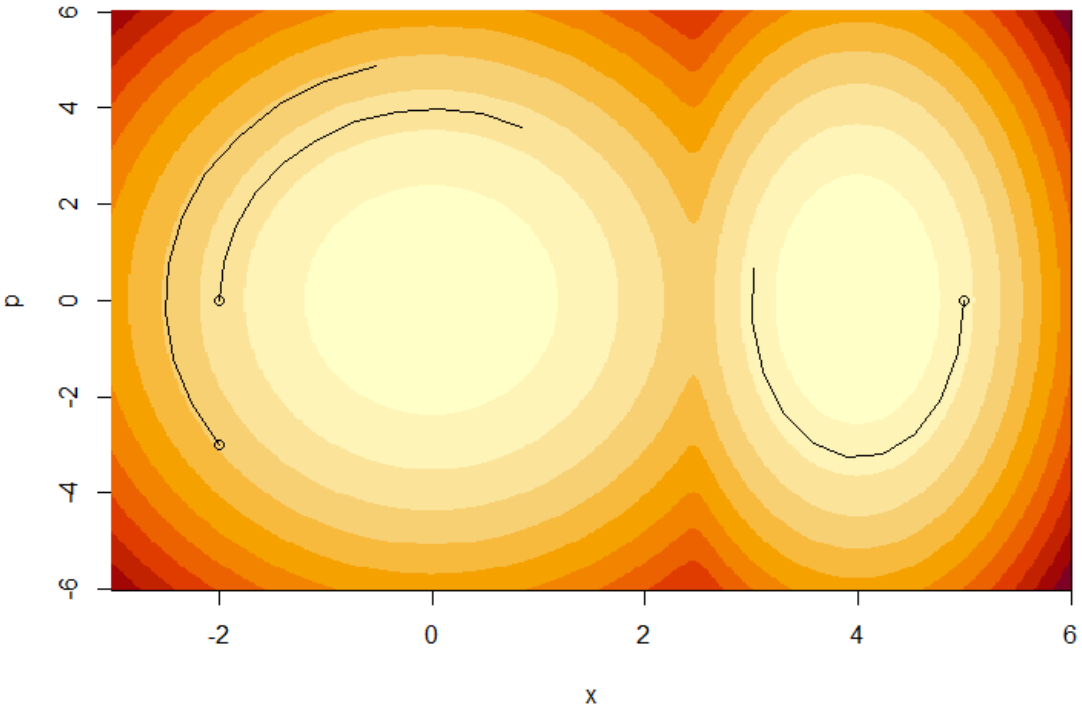
```
U = function(x)
{
  -log(distr(x))
}
Hfunc = function(x,p)
{
  U(x)+0.5*p^2/m
}
```

```
## image leapfrog
Nx=500
Np=401
HM=matrix(nrow = Nx, ncol = Np)
x = seq(-3,6,length=Nx)
p = seq(-sqrt(m)*6,sqrt(m)*6,length=Np)
for(i in 1:Nx)
   for(j in 1:Np)
     HM[i,j]=Hfunc(x[i],p[j])

par(mfrow=c(1,1))
image(x,p,HM)


xStore=rep(0,L+1)
pStore=rep(0,L+1)
L=10
eps=0.1;

x.prop =5
p.prop      = 0
xStore[1]=x.prop
pStore[1]=p.prop
points(x.prop,p.prop)
for(j in 1:L)
{
  p.half = p.prop - 0.5*eps*dU(x.prop)
  x.prop = x.prop + eps*p.half/m
  p.prop = p.half - 0.5*eps*dU(x.prop)
  xStore[j+1]=x.prop
  pStore[j+1]=p.prop
}
points(xStore[1],pStore[1])
lines(xStore,pStore)
```



Random step length

```
## run leap frog
eps = 0.1
L = 10
X = rep(NA,N)
X[1] = rnorm(1)
acc = 0
ptm <- proc.time()
for(i in 2:N)
{
  eps = runif(1,0.8,1.2)
  p.prop = rnorm(1,0,sqrt(m))
  H = Hfunc(X[i-1],p.prop)
  x.prop = X[i-1]
  for(j in 1:L)
  {
    p.half = p.prop - 0.5*eps*dU(x.prop)
    x.prop = x.prop + eps*p.half/m
    p.prop = p.half - 0.5*eps*dU(x.prop)
  }
  H.prop = Hfunc(x.prop,p.prop)

  R = exp(H-H.prop)
  show(c(X[i-1],x.prop,H,H.prop,R))
  if(runif(1)<R)
  {
    X[i] = x.prop
    acc = acc + 1
  }
  else
    X[i] = X[i-1]
}
```