STAT 576 Bayesian Analysis

Lecture 9: Hybrid Monte Carlo

Chencheng Cai

Washington State University

Hybrid Monte Carlo

- Hybrid Monte Carlo is a MCMC algorithm that combines the Metropolis algorithm with molecular dynamics (MD).
- The molecular dynamics (MD) is a **deterministic** algorithm that simulates the motion of particles in a physical system.
- We use the MD to propose a new state of the system, and then use the Metropolis algorithm to accept or reject the proposed state.
- Benefits:
  - It can explore the state space more efficiently than the random walk Metropolis.
  - It can handle high-dimensional state space.

Newton's Mechanics

Consider a single particle with mass m in a potential energy field $U(\boldsymbol{q})$.

- The particle's **position** is denoted by $\boldsymbol{q}$ and its **momentum** by $\boldsymbol{p}$. In **non-relativistic (NR)** mechanics, the momentum is given by

$$\boldsymbol{p} = m\boldsymbol{v},$$

  where $\boldsymbol{v} = \dot{\boldsymbol{q}} := d\boldsymbol{q}/dt$ is the velocity.

- The **kinetic energy** of the particle is

$$T(\boldsymbol{p}) = \frac{\|\boldsymbol{p}\|^2}{2m} \overset{\text{NR}}{=} \frac{1}{2}m\|\boldsymbol{v}\|^2.$$

Newton's Mechanics

The (NR) dynamics of this particle is determined by Newton's second law:

$$\boldsymbol{F} = m\boldsymbol{a},$$

where $\boldsymbol{F}$ is the force acting on the particle, and $\boldsymbol{a} = \ddot{\boldsymbol{q}} = d^2\boldsymbol{q}/dt^2$ is the acceleration.

- If the force is derived from a potential energy field $U(\boldsymbol{q})$, then the force is given by

$$\boldsymbol{F} = -\frac{\partial U(\boldsymbol{q})}{\partial \boldsymbol{q}}$$

- The $m\boldsymbol{a}$ is the time derivative of the momentum:

$$m\boldsymbol{a} = \dot{\boldsymbol{p}} = \frac{d\boldsymbol{p}}{dt}$$

- The Newton's second law in the **phase space** $(\boldsymbol{q}, \boldsymbol{p})$ is

$$-\frac{\partial U(\boldsymbol{q})}{\partial \boldsymbol{q}} = \dot{\boldsymbol{p}}.$$

Lagrangian Analytical Mechanics

We consider a (NR) particle in a system, whose dynamics is parametrized by $(t, \boldsymbol{q}, \dot{\boldsymbol{q}})$.

- We observe that
$$\frac{\partial T(\dot{\boldsymbol{q}})}{\partial \dot{\boldsymbol{q}}} = m\dot{\boldsymbol{q}}, \quad \boldsymbol{F} = -\frac{\partial U(\boldsymbol{q})}{\partial \boldsymbol{q}}$$

- By Newton's second law, we have
$$\frac{d}{dt}\frac{\partial T(\dot{\boldsymbol{q}})}{\partial \dot{\boldsymbol{q}}} + \frac{\partial U(\boldsymbol{q})}{\partial \boldsymbol{q}} = 0$$

- If we define the **Lagrangian** of the system by
$$L(t, \boldsymbol{q}, \dot{\boldsymbol{q}}) = T(\dot{\boldsymbol{q}}) - U(\boldsymbol{q}),$$
then the equation above is equivalent to the **Euler-Lagrange equation**:
$$\frac{\partial L}{\partial \boldsymbol{q}} - \frac{d}{dt}\frac{\partial L}{\partial \dot{\boldsymbol{q}}} = 0$$

Lagrangian Analytical Mechanics

We consider a particle in a system, whose dynamics is parametrized by $(t, \boldsymbol{q}, \boldsymbol{p})$.

- By Newton's second law, we have

$$-\frac{\partial U(\boldsymbol{q})}{\partial \boldsymbol{q}} = \dot{\boldsymbol{p}}.$$

- On the other hand, we have

$$\frac{\partial T(\boldsymbol{p})}{\partial \boldsymbol{p}} = \dot{\boldsymbol{q}}$$

- Combnining the two equations above, we have the **Hamilton's equations**:

$$\dot{\boldsymbol{q}} = \frac{\partial H(\boldsymbol{q}, \boldsymbol{p})}{\partial \boldsymbol{p}},$$
$$\dot{\boldsymbol{p}} = -\frac{\partial H(\boldsymbol{q}, \boldsymbol{p})}{\partial \boldsymbol{q}},$$

where $H(\boldsymbol{q}, \boldsymbol{p}) = T(\boldsymbol{p}) + U(\boldsymbol{q})$ is the **Hamiltonian** of the system.

Lagrangian Analytical Mechanics

The Hamiltonian's equations can be derived from the Euler-Lagrange equation directly.

- We define the momentum $\boldsymbol{p}$ as the conjugate variable of the position $\boldsymbol{q}$:

$$\boldsymbol{p} = \frac{\partial L}{\partial \dot{\boldsymbol{q}}}$$

- The Hamiltonian is defined as the Legendre transform of the Lagrangian:

$$H(\boldsymbol{q}, \boldsymbol{p}) = \boldsymbol{p} \cdot \dot{\boldsymbol{q}} - L(\boldsymbol{q}, \dot{\boldsymbol{q}})$$

- The Hamiltonian's equations are derived from the Euler-Lagrange equation:

$$\frac{\partial H}{\partial \boldsymbol{p}} = \dot{\boldsymbol{q}},$$
$$\frac{\partial H}{\partial \boldsymbol{q}} = -\frac{\partial L}{\partial \boldsymbol{q}} = -\frac{d}{dt}\frac{\partial L}{\partial \dot{\boldsymbol{q}}} = -\dot{\boldsymbol{p}}.$$

Molecular Dynamics Simulation

Consider simulating a system with Hamiltonian $H(\boldsymbol{q}, \boldsymbol{p})$.

- Use Taylor expansion:

$$\begin{aligned}
\boldsymbol{q}(t+dt) &= \boldsymbol{q}(t) + \dot{\boldsymbol{q}}(t)dt + \frac{1}{2}\ddot{\boldsymbol{q}}(t)(dt)^2 + \cdots, \\
&= \boldsymbol{q}(t) + \frac{\boldsymbol{p}(t)}{m}dt + \frac{1}{2}\frac{\dot{\boldsymbol{p}}(t)}{m}(dt)^2 + \cdots, \\
\boldsymbol{p}(t+dt) &= \boldsymbol{p}(t) + \dot{\boldsymbol{p}}(t)dt + \frac{1}{2}\ddot{\boldsymbol{p}}(t)(dt)^2 + \cdots.
\end{aligned}$$

- Furthermore, we have

$$\begin{aligned}
\boldsymbol{q}(t+dt) + \boldsymbol{q}(t-dt) &= 2\boldsymbol{q}(t) + \frac{\dot{\boldsymbol{p}}(t)}{m}(dt)^2 + O((dt)^4), \\
\boldsymbol{q}(t+dt) - \boldsymbol{q}(t-dt) &= 2\boldsymbol{p}(t)dt + O((dt)^3).
\end{aligned}$$

Molecular Dynamics Simulation

Størmer-Verlet algorithm:

- At time t with a time increment Δt.
- We update the position by:

$$\boldsymbol{q}(t+\Delta t) = 2\boldsymbol{q}(t) - \boldsymbol{q}(t-\Delta t) - \frac{(\Delta t)^2}{m}\frac{\partial H}{\partial \boldsymbol{q}}\bigg|_t$$

- We update the momentum by:

$$\boldsymbol{p}(t+\Delta t) = m\frac{\boldsymbol{q}(t+\Delta t) - \boldsymbol{q}(t-\Delta t)}{2\Delta t}$$

or

$$\boldsymbol{p}(t+\Delta t) = m\frac{\boldsymbol{q}(t+\Delta t) - \boldsymbol{q}(t)}{\Delta t}$$

Molecular Dynamics Simulation

Leap-frog algorithm:

- At time t with a time increment Δt.
- We update the position by:

$$\boldsymbol{q}(t+\Delta t) = \boldsymbol{q}(t) + \Delta t \frac{\boldsymbol{p}\left(t+\frac{1}{2}\Delta t\right)}{m}$$

- We update the momentum at the half-time interval by:

$$\boldsymbol{p}\left(t+\frac{1}{2}\Delta t\right) = \boldsymbol{p}\left(t-\frac{1}{2}\Delta t\right) - \left.\frac{\partial H}{\partial \boldsymbol{q}}\right|_t \Delta t$$

Molecular Dynamics Simulation

Leap-frog algorithm (alternative form):

- At time t with a time increment Δt.
- We update the momentum by the first half-time interval:

$$\boldsymbol{p}\left(t+\frac{1}{2}\Delta t\right)=\boldsymbol{p}\left(t\right)-\left.\frac{\partial H}{\partial \boldsymbol{q}}\right|_{t}\frac{\Delta t}{2}$$

- We update the position by:

$$\boldsymbol{q}(t+\Delta t)=\boldsymbol{q}(t)+\Delta t\frac{\boldsymbol{p}\left(t+\frac{1}{2}\Delta t\right)}{m}$$

- We update the momentum by the second half-time interval:

$$\boldsymbol{p}\left(t+\Delta t\right)=\boldsymbol{p}\left(t+\frac{1}{2}\Delta t\right)-\left.\frac{\partial H}{\partial \boldsymbol{q}}\right|_{t+\Delta}\frac{\Delta t}{2}$$

Molecular Dynamics Simulation

Størmer-Verlet algorithm and Leap-frog algorithm are identical algorithms.

- From the position update set in leap-frog algorithm, we have

$$\boldsymbol{q}(t+\Delta t) = \boldsymbol{q}(t) + \Delta t \frac{\boldsymbol{p}\left(t+\frac{1}{2}\Delta t\right)}{m}$$

$$\boldsymbol{q}(t) = \boldsymbol{q}(t-\Delta t) + \Delta t \frac{\boldsymbol{p}\left(t-\frac{1}{2}\Delta t\right)}{m}$$

- Substitute the second equation into the first equation, we have

$$\boldsymbol{q}(t+\Delta t) = 2\boldsymbol{q}(t) - \boldsymbol{q}(t-\Delta t) - \frac{(\Delta t)^2}{m}\frac{\partial H}{\partial \boldsymbol{q}}\bigg|_t$$

which is the same as the position update in Størmer-Verlet algorithm.

Molecular Dynamics Simulation

The Hamiltonian dynammics has the following preservation properties:

- The Hamiltonian $H(\boldsymbol{q}, \boldsymbol{p})$ is preserved by the dynamics.

$$\frac{dH}{dt} = \frac{\partial H}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + \frac{\partial H}{\partial \boldsymbol{p}}\dot{\boldsymbol{p}} = 0.$$

- The volume in the phase space is preserved by the dynamics.
  Let $V(t) = \{(\boldsymbol{q}(t), \boldsymbol{p}(t)) : (\boldsymbol{q}(0), \boldsymbol{p}(0)) \in V(0)\}$, then
  Louville's theorem: $V(t)$ has the same volume as $V(0)$.

The MD simulation is volume-preservation and time reversable, but not Hamiltonian-preserving.

- The MD simulation is volume-preserving because the Jacobian of the transformation at each step is 1.
- The MD simulation is time reversable because we can simulate the system backward by reversing the momentum.
- Hamiltonian is not preserved because of the dicretization error.

Example

Consider a dynamic system with

$$U(q) = q^2 - 2\log[\cosh(2q)] + 3, \quad T(p) = p^2/2$$

The Hamiltonian is

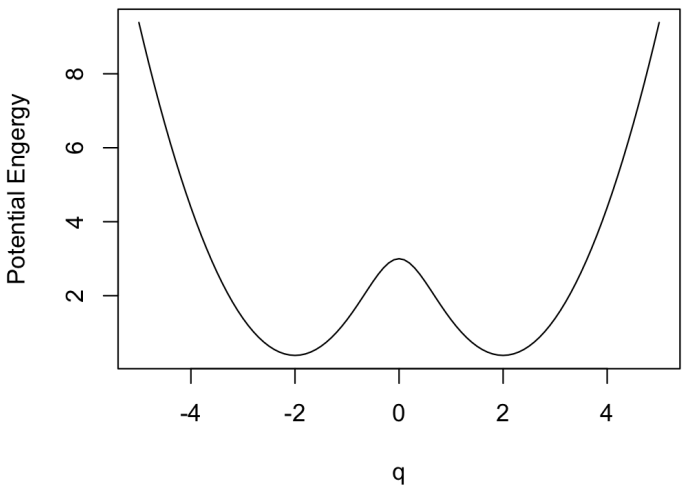
$$H(q,p) = U(q) + T(p) = q^2 + \frac{p^2}{2} - 2\log[\cosh(2q)]$$

The Hamiltonian's equations give

$$\dot{p} = -\frac{\partial H}{\partial q} = -2q + 4\tanh(2q)$$

We simulate the system using the leap-frog algorithm with $\Delta t = 0.1$, starting position $q(0) = 1$, and five different momentums $p(0) = 1, 2, 3, 4, 5$.

Example

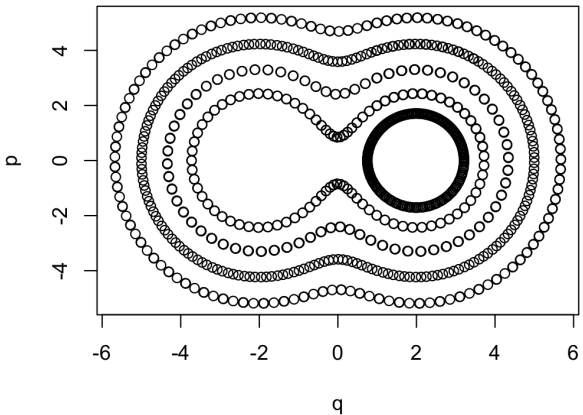


Example

```
n = 200
dt = 0.1

phase_sample = array(dim=c(5, 2, n+1))
phase_sample[,1,1] = 1
phase_sample[,2,1] = 1:5

pdot = function(q) {-2*q + 4*tanh(2*q)}

for(t in 1:n){
    p = phase_sample[,2,t] + 0.5 * pdot(phase_sample[,1,t]) * dt
    phase_sample[,1,t+1] = phase_sample[,1,t] + dt * p
    phase_sample[,2,t+1] = p + 0.5 * pdot(phase_sample[,1,t+1]) * dt
}
```

Example

Trajectories in the phase space of the system with different initial momentums.



Hamiltonian Monte Carlo

Suppose we have a target distribution $\pi(\boldsymbol{x}) \propto \exp\{-U(\boldsymbol{x})\}$, and we want to sample from it.

- We introduce an auxiliary momentum variable $\boldsymbol{p}$ with density $p(\boldsymbol{p}) = \exp\{-T(\boldsymbol{p})\}$.
- The joint distribution of $(\boldsymbol{x}, \boldsymbol{p})$ is

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

  where $H(\boldsymbol{x}, \boldsymbol{p}) = U(\boldsymbol{x}) + T(\boldsymbol{p})$ is the Hamiltonian.

Observations:

- If we can sample from the joint distribution $p(\boldsymbol{x}, \boldsymbol{p})$, then we can sample from the target distribution $\pi(\boldsymbol{x})$ by marginalizing out $\boldsymbol{p}$.
- The Hamiltonian dynamics is reversable, meaning that if $(\boldsymbol{x}(t), \boldsymbol{p}(t))$ results in $(\boldsymbol{x}(t + \Delta t), \boldsymbol{p}(t + \Delta t))$, then $(\boldsymbol{x}(t + \Delta t), -\boldsymbol{p}(t + \Delta t))$ results in $(\boldsymbol{x}(t), -\boldsymbol{p}(t))$.

Hamiltonian Monte Carlo

Suppose we have a target distribution $\pi(\boldsymbol{x}) \propto \exp\{-U(\boldsymbol{x})\}$. The Hamiltonian Monte Carlo algorithm is as follows: (for time $t+1$)

- Sample a momentum $\boldsymbol{p}$ from

$$p(\boldsymbol{p}) \propto \exp\{-T(\boldsymbol{p})\}$$

- Run the leap-frog algorithm for L steps with step size ϵ jumping from $(\boldsymbol{x}, \boldsymbol{p})$ to $(\boldsymbol{x}', \boldsymbol{p}')$.
- Accept the new state $(\boldsymbol{x}', \boldsymbol{p}')$ with probability

$$\alpha = \min \left\{1, \exp\{-H(\boldsymbol{x}', \boldsymbol{p}') + H(\boldsymbol{x}, \boldsymbol{p})\}\right\}$$

Remark: The leap-frog step can be replaced by any deterministic time-reversible and volume-preserving dynamics.

Hamiltonian Monte Carlo

Justification for correctness:

- (time-reversible) If the L leap-frog steps map $(\boldsymbol{x}, \boldsymbol{p})$ to $(\boldsymbol{x}', \boldsymbol{p}')$, then they also map $(\boldsymbol{x}', -\boldsymbol{p}')$ to $(\boldsymbol{x}, -\boldsymbol{p})$.
- (volume-preserving) The leap-frog algorithm is volume-preserving. Therefore $d\boldsymbol{x}d\boldsymbol{p} = d\boldsymbol{x}'d\boldsymbol{p}'$.
- The acceptance probability in a Metropolis-Hastings algorithm is

$$\alpha = \min\left\{1, \frac{\pi(\boldsymbol{x}')}{\pi(\boldsymbol{x})}\frac{p(\boldsymbol{x}', \boldsymbol{x})d\boldsymbol{x}'d\boldsymbol{p}'}{p(\boldsymbol{x}, \boldsymbol{x}')d\boldsymbol{x}d\boldsymbol{p}}\right\} = \min\left\{1, \frac{\pi(\boldsymbol{x}')}{\pi(\boldsymbol{x})}\frac{p(\boldsymbol{p}')}{p(\boldsymbol{p})}\right\} = \min\left\{1, \frac{e^{-H(\boldsymbol{x}', \boldsymbol{p})}}{e^{-H(\boldsymbol{x}, \boldsymbol{p})}}\right\}$$

Example

Consider sampling from the $\mathrm{Beta}(2, 2)$ distribution. The Hamiltonian is

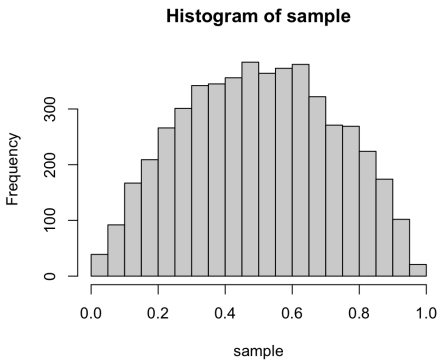
$$H(x, p) = -\log(x(1-x)) + \frac{p^2}{2}$$

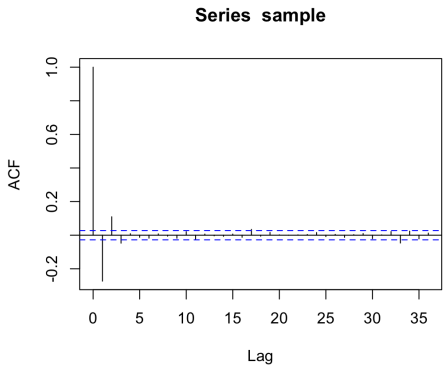
```
pdot = function(x){1/x-1/(1-x)}
h = function(x, p){
    if(x*(1-x) <= 0) return(Inf)
    else return(-log(x*(1-x)) + p**2/2)}

eps = 0.05
L = 10
n = 5000
burnin = 500
sample = rep(0, n+1)
x = 0.5
```

Example

```
for(i in 1:(n+burnin)){
    p = rnorm(1)
    h_old = h(x, p)
    y = x
    for(j in 1:L){
        p = p + 0.5 * pdot(y) * eps
        y = y + p * eps
        p = p + 0.5 * pdot(y) * eps
    }
    h_new = h(y, p)
    if(runif(1) <= exp(-h_new + h_old)) x = y
    if(i > burnin) sample[i-burnin] = x
}
```

Example


Histogram of sample
Frequency
0
100
200
300
0.0
0.2
0.4
0.6
0.8
1.0
sample


Series sample
ACF
-0.2
0.2
0.6
1.0
0
5
10
15
20
25
30
35
Lag

Langevin Dynamics

Recall the Taylor's expansion for the position:

$$\boldsymbol{q}(t+dt) = \boldsymbol{q}(t) + \frac{\boldsymbol{p}(t)}{m}dt + \frac{1}{2}\frac{\dot{\boldsymbol{p}}(t)}{m}(dt)^2 + \cdots$$

- With Hamiltonian's equations, we have

$$\boldsymbol{q}(t+dt) = \boldsymbol{q}(t) + \frac{\boldsymbol{p}(t)}{m}dt - \frac{1}{2}\frac{\partial H/\partial \boldsymbol{q}}{m}(dt)^2 + \cdots$$

- In the Hamiltonian Monte Carlo, we use the leap-frog algorithm to simulate the dynamics:
  - $m = 1$
  - $\boldsymbol{p}$ is drawn from a standard multivariate normal.
  - $H = U(\boldsymbol{q}) + \|\boldsymbol{p}\|^2$.

Langevin Dynamics

If the target distribution is $\boldsymbol{x} \sim \pi(\boldsymbol{x})$, one step of leap-frog is equivalent to the following Langevin dynamics:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \frac{1}{2}\frac{\partial \log \pi(\boldsymbol{x})}{\partial \boldsymbol{x}} h + \sqrt{h}\boldsymbol{Z}_t,$$

- Time indices are relabelled to integers.
- $\boldsymbol{Z}_t$ is standard multivariate normal. (random sampling for $\boldsymbol{p}$)
- h is the step size. ($\sqrt{dt} = h$).

The corresponding stochastic differential equation is

$$d\boldsymbol{x}_t = \frac{1}{2}\frac{\partial \log \pi(\boldsymbol{x})}{\partial \boldsymbol{x}} dt + d\boldsymbol{W}_t.$$

Langevin Dynamics

The **Langevin dynamics** is a continuous-time stochastic process for a particle in a potential field $U(\boldsymbol{x})$. The stochastic differential equation is

$$d\boldsymbol{x}_t = -\frac{1}{2}\frac{\partial U(\boldsymbol{x})}{\partial \boldsymbol{x}}dt + d\boldsymbol{W}_t,$$

where $\boldsymbol{W}_t$ is standard Brownian motion.

Let $p(\boldsymbol{x}, t)$ be the density of the particle at time t.
The **Fokker-Planck equation** (also known as the **Kolmogorov forward equation**) is

$$\frac{\partial p(\boldsymbol{x}, t)}{\partial t} = \frac{1}{2}\nabla \cdot \left[p(\boldsymbol{x}, t)\nabla U(\boldsymbol{x}) + \nabla p(\boldsymbol{x}, t)\right].$$

$p(\boldsymbol{x}, t) = \exp\{-U(\boldsymbol{x})\}$ is the stationary distribution such that

$$\frac{\partial p(\boldsymbol{x}, t)}{\partial t} = 0$$

Langevin Dynamics

To generate samples from the target distribution $\pi(\boldsymbol{x})$, we can simulate the Langevin dynamics with SDE:

$$d\boldsymbol{x}_t = \frac{1}{2}\nabla \log \pi(\boldsymbol{x})dt + d\boldsymbol{W}_t.$$

Its discretized version is

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \frac{1}{2}\nabla \log \pi(\boldsymbol{x}_t)h + \sqrt{h}\boldsymbol{Z}_t,$$

where $\boldsymbol{Z}_t$ is standard multivariate normal.
This step (without the auxiliary momentum) can replace the leap-frog step in the Hamiltonian Monte Carlo.

Metropolis-adjusted Langevin Algorithm

Suppose the target distribution is $\pi(\boldsymbol{x}) \propto \exp\{-U(\boldsymbol{x})\}$. The Metropolis-adjusted Langevin algorithm is as follows: (for time $t+1$)

- Run the Langevin dynamics with step size ϵ jumping from $\boldsymbol{x}$ to $\boldsymbol{x}'$.

$$\boldsymbol{x}' = \boldsymbol{x} - \frac{1}{2}\nabla U(\boldsymbol{x})h + \sqrt{h}\boldsymbol{Z}_t.$$

- Accept the new state $\boldsymbol{x}'$ with probability

$$\alpha = \min\left\{1, \frac{\pi(\boldsymbol{x}')\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{x}' - \frac{h}{2}\nabla U(\boldsymbol{x}'), h\boldsymbol{I})}{\pi(\boldsymbol{x})\mathcal{N}(\boldsymbol{x}' \mid \boldsymbol{x} - \frac{h}{2}\nabla U(\boldsymbol{x}), h\boldsymbol{I})}\right\}$$

Example

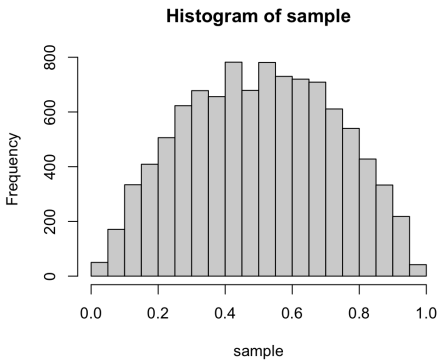
Draw samples from the $\mathrm{Beta}(2, 2)$ distribution using the Langevin dynamics.
We have

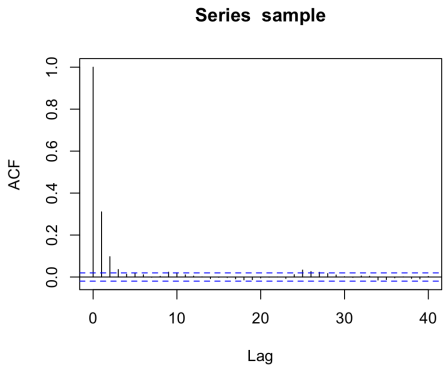
$$U(x) = -\log x(1-x) \text{ and } dU/dx = -\frac{1}{x} + \frac{1}{1-x}$$

Example

```
n = 10000
burnin = 1000
sample = rep(0, n)
x = 0.5
h = 0.1
hsqrt = sqrt(h)
for(i in 1:(n+burnin)){
    z = rnorm(1)
    y = x - 0.5*(2*x-1)/x/(1-x)*h + hsqrt*z
    if(y*(1-y)>0){
        logpy = log(y*(1-y)) - 0.5*(x-y+0.5*(2*y-1)/y/(1-y)*h)**2/h
        logpx =log(x*(1-x)) - 0.5*z**2
        if(runif(1) <= exp(logpy-logpx)) x = y
    }
    if(i > burnin) sample[i - burnin] = x
}
```

Example


Histogram of sample
Frequency
0
200
400
600
800
0.0
0.2
0.4
0.6
0.8
1.0
sample


Series sample
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
Lag

Langevin Dynamics

The Langevin dynamics can also be written as the following stochastic differential equation:

$$d\boldsymbol{x}_t = -\nabla U(\boldsymbol{x})dt + \sqrt{2}d\boldsymbol{W}_t,$$

where $\boldsymbol{W}_t$ is standard Brownian motion.
With the discretization as

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \nabla U(\boldsymbol{x}_t)h + \sqrt{2h}\boldsymbol{Z}_t.$$

This version (along with the previous version) is also called the **overdamped Langevin dynamics**.

Langevin Dynamics

The **underdamped Langevin dynamics** is a second-order stochastic differential equation:

$$d\boldsymbol{x}_t = \boldsymbol{v}_t dt, \quad d\boldsymbol{v}_t = -\nabla U(\boldsymbol{x}_t)dt - \gamma \boldsymbol{v}_t dt + \sqrt{2\gamma} d\boldsymbol{W}_t,$$

where γ is the friction coefficient, and the mass is assumed to be 1.

- When $\gamma \gg 1$, the dynamics is overdamped. When $\gamma \ll 1$, the dynamics is underdamped.
- The overdamped Langevin dynamics converges slower than the underdamped Langevin dynamics.

Example: the Eight Schools

Recall the eight schools example. The hierarchical model is

$$\begin{aligned} p(\mu, \tau) &\propto 1 \\ \alpha_j &\sim \mathcal{N}(\mu, \tau^2) \\ y_j &\sim \mathcal{N}(\theta_j, \sigma_j^2) \end{aligned}$$

The posterior distribution is $\theta = (\mu, \tau, \boldsymbol{\alpha})$.

$$p(\theta \mid y) \propto p(\mu, \tau) \prod_{j=1}^{8} p(\alpha_j \mid \mu, \tau) p(y_j \mid \alpha_j, \sigma_j) \propto \frac{1}{\tau^J} e^{-\frac{\sum(\alpha_j - \mu)^2}{2\tau^2}} e^{-\sum \frac{(\alpha_j - y_j)^2}{2\sigma_j^2}}$$

Example: the Eight Schools

In both HMC and LMC, we need to compute the gradient of the log posterior.

$$\frac{\partial \log p(\theta \mid y)}{\partial \mu} = -\frac{J}{\tau^2}(\mu - \bar{\alpha})$$

$$\frac{\partial \log p(\theta \mid y)}{\partial \tau} = -\frac{J}{\tau} + \frac{1}{\tau^3}\sum_{j=1}^{J}(\alpha_j - \mu)^2$$

$$\frac{\partial \log p(\theta \mid y)}{\partial \alpha_j} = -\frac{\alpha_j - \mu}{\tau^2} - \frac{\alpha_j - y_j}{\sigma_j^2}$$

The Hamiltonian is

$$H(\mu, \tau, \boldsymbol{\alpha}, p_\mu, p_\tau, \boldsymbol{p}_\alpha) = -\log p(\theta \mid y) + \frac{1}{2\sigma_p^2}\left(p_\mu^2 + p_\tau^2 + \|\boldsymbol{p}_\alpha\|^2\right)$$

Example: the Eight Schools

```
n = 1000
burnin = 100
J = 8

mu.sample = rep(0, n)
tau.sample = rep(0, n)
alpha.sample = array(dim=c(J, n))

mu = 0
tau = 5
alpha = rep(0, J)

L = 20
eps = 0.05
p.sigma = 0.1
```

Example: the Eight Schools

```
ham = function(mu, tau, alpha, p.mu, p.tau, p.alpha){
    if(tau <= 0)
        return(Inf)
    p1 = 0.5*sum((alpha-mu)**2)/tau**2
    p2 = 0.5*sum((alpha-y)**2/s**2)
    p3 = J*log(tau)
    p4 = p.mu**2/2/p.sigma**2
    p5 = p.tau**2/2/p.sigma**2
    p6 = sum(p.alpha**2)/2/p.sigma**2
    return(p1+p2+p3+p4+p5+p6)
}
```

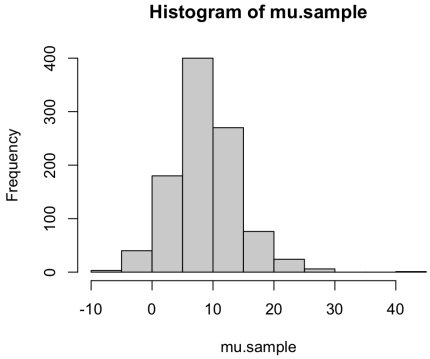
Example: the Eight Schools

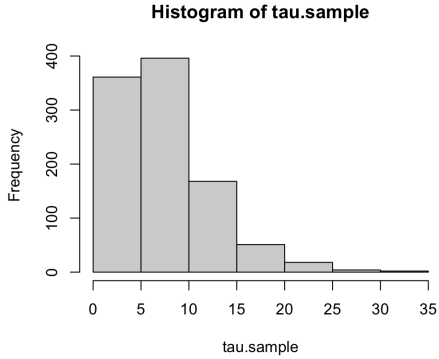
```
for(i in 1:(n+burnin)){
    p.mu = rnorm(1) * p.sigma
    p.tau = rnorm(1) * p.sigma
    p.alpha = rnorm(J) * p.sigma
    h_old = ham(mu, tau, alpha, p.mu, p.tau, p.alpha)
    n.mu = mu
    n.tau = tau
    n.alpha = alpha
    // leap-frog step here
    h_new = ham(n.mu, n.tau, n.alpha, p.mu, p.tau, p.alpha)
    if(runif(1) <= exp(h_old - h_new)){
        mu = n.mu
        tau = n.tau
        alpha = n.alpha
    }
    if(i > burnin){
        mu.sample[i-burnin] = mu
        tau.sample[i-burnin] = tau
        alpha.sample[,i-burnin] = alpha
    }
}
```

Example: the Eight Schools

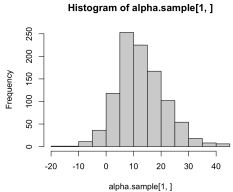
```
// leap-frog step
for(j in 1:L){
    p.mu = p.mu + 0.5*eps* (-J) * (n.mu - mean(n.alpha)) / n.tau ** 2
    p.tau = p.tau+0.5*eps* (-J/n.tau+sum((n.alpha-n.mu)**2)/n.tau**3)
    p.alpha = p.alpha+0.5*eps*(-(n.alpha-n.mu)/n.tau**2-(n.alpha-y)/s**2)

    n.mu = n.mu + p.mu * eps / p.sigma ** 2
    n.tau = n.tau + p.tau * eps / p.sigma ** 2
    n.alpha = n.alpha + p.alpha * eps / p.sigma ** 2

    p.mu = p.mu + 0.5*eps* (-J) * (n.mu - mean(n.alpha)) / n.tau ** 2
    p.tau = p.tau+0.5*eps* (-J/n.tau+sum((n.alpha-n.mu)**2)/n.tau**3)
    p.alpha = p.alpha+0.5*eps*(-(n.alpha-n.mu)/n.tau**2-(n.alpha-y)/s**2)
}
```

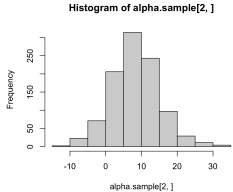
Example: the Eight Schools

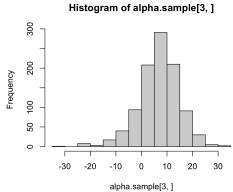


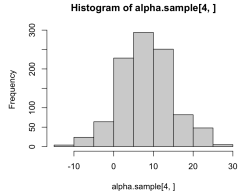


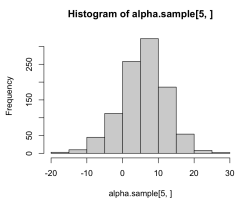
Example: the Eight Schools

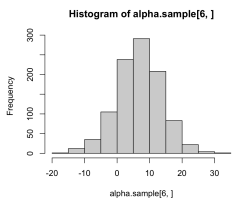


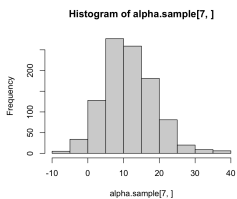


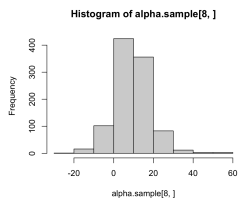












Example: the Eight Schools

We can also use the Langevin dynamics to sample from the posterior distribution.

```
h = 0.1
hsqrt = sqrt(h)
mu = 0
tau = 5
alpha = rep(0, J)

logp = function(mu, tau, alpha){
    if(tau <= 0) return(-Inf)
    p1 = 0.5*sum((alpha-mu)**2)/tau**2
    p2 = 0.5*sum((alpha-y)**2/s**2)
    p3 = J*log(tau)
    return(-(p1+p2+p3))
}
fun.dlogp = function(mu, tau, alpha){
    dmu = -J*(mu - mean(alpha))/tau ** 2
    dtau =  -J/tau+sum((alpha-mu)**2)/tau**3
    dalpha = -(alpha - mu)/tau**2 - (alpha-y)/s**2
    return(list(mu=dmu, tau=dtau, alpha=dalpha))
}
```

Example: the Eight Schools

```
for(i in 1:(n+burnin)){
    z.mu = rnorm(1)
    z.tau = rnorm(1)
    z.alpha = rnorm(J)
    dlogp = fun.dlogp(mu, tau, alpha)
    n.mu = mu + 0.5*h*dlogp$mu + hsqrt * z.mu
    n.tau = tau + 0.5*h*dlogp$tau + hsqrt*z.tau
    n.alpha = alpha + 0.5*h*dlogp$alpha +hsqrt*z.alpha
    dlogp = fun.dlogp(n.mu, n.tau, n.alpha)
    loga = logp(n.mu, n.tau, n.alpha) - logp(mu, tau, alpha)
    loga = loga - 0.5 * (mu - n.mu - 0.5*h*dlogp$mu)**2/h
    loga = loga - 0.5 * (tau - n.tau - 0.5*h*dlogp$tau)**2/h
    loga = loga - 0.5 * sum((alpha - n.alpha - 0.5*h*dlogp$alpha)**2/h)
    loga = loga + 0.5 * z.mu ** 2 + 0.5*z.tau**2 + 0.5 * sum(z.alpha**2)

    //Accept with prob exp(loga)
    //store the sample
}
```