MATH 598: TOPICS IN STATISTICS

HAMILTONIAN MARKOV CHAIN MONTE CARLO

Hamiltonian Monte Carlo appeals to the dynamics of physical systems to perform sampling from a target distribution. Consider two d-dimensional vectors that describe the motion of an object in $\mathbb{R}^d$.

- $x = (x_1, \ldots, x_d)$ denotes the *position* of the object
- $v = (v_1, \ldots, v_d)$ denotes the *momentum* (proportional to the *velocity*) of the object.

Let $z = (x, v)$. Hamiltonian dynamics is formulated via the *Hamiltonian*, denoted $H(z) \equiv H(x, v)$, which is a function of $z = (x, v)$ but *not* t, and which describes how z changes in time via the differential form

$$\frac{dz}{dt} = \begin{bmatrix} \mathbf{0}_d & \mathbf{I}_d \\ -\mathbf{I}_d & \mathbf{0}_d \end{bmatrix} \frac{\partial H(z)}{\partial z} = \mathbf{D} \frac{\partial H(z)}{\partial z}$$

say, that is

$$\frac{dx}{dt} = \frac{\partial H(x,v)}{\partial v} \qquad \frac{dv}{dt} = -\frac{\partial H(x,v)}{\partial x}.$$

Attention focusses on Hamiltonians that satisfy $H(x, v) = H(x, -v)$ and are assumed to be separable

$$H(x, v) = U(x) + K(v)$$

so that $K(v) = K(-v)$.

- $U(x)$ is termed the *potential energy*
- $K(v)$ is termed the *kinetic energy*, where typically

$$K(v) = \frac{1}{2} v^\top \mathbf{M}^{-1} v$$

  where $\mathbf{M}$ is a positive definite, symmetric matrix.

The Hamiltonian, and Hamilton's equations, define the evolution of the system in continuous time: the objective is to find the solution $z(t) = (x(t), v(t))$ to the equations for all t.

In a statistical problem, we consider joint pdf $\pi_{X,V}(x, v)$

$$\pi_{X,V}(x, v) \propto \exp\{-H(x, v)\} = \exp\{-U(x) - K(v)\}$$

corresponding to independence of corresponding random variables X and V

$$\pi_X(x) \propto \exp\{-U(x)\} \qquad \text{and} \qquad \pi_V(v) \propto \exp\{-K(v)\}.$$

If the marginal $\pi_X(x)$ is our true target, then V is merely an *auxiliary variable*. We are free to choose the distribution $\pi_V(v)$, and a typical choice is the multivariate Normal

$$\pi_V(v) \propto \exp\left\{-\frac{1}{2} v^\top \mathbf{M}^{-1} v\right\} = \exp\{-K(v)\}.$$

If $\mathbf{M} = \text{diag}(m_1, \ldots, m_d)$, then

$$K(v) = \frac{1}{2} \sum_{j=1}^{d} \frac{v_j^2}{m_j}$$

which corresponds to an assumption that $V_j \sim Normal(0, m_j)$ for $j = 1, \ldots, d$ are independent. Here

$$\dot{K}(v) = \mathbf{M}^{-1} v = \begin{bmatrix} \frac{v_1}{m_2} \\ \vdots \\ \frac{v_d}{m_d} \end{bmatrix}$$

Hamiltonian dynamics can be approximated using a discrete time approach. We consider $\{z_k \equiv z(k\delta), k = 1, 2, \ldots\}$ for time-step $\delta > 0$.

- **Euler method:** the dynamics equation becomes in an Euler approximation

$$z_{k+1} = z_k + \delta \mathbf{D} \frac{\partial H(z)}{\partial z}\bigg|_{z=z_k}.$$

That is, if $H(x,v) = U(x) + K(v)$

$$x_{k+1} = x_k + \delta \frac{\partial H(x,v)}{\partial v}\bigg|_{z=(x_k,v_k)} = x_k + \delta \dot{K}(v_k)$$

$$v_{k+1} = v_k - \delta \frac{\partial H(x,v)}{\partial x}\bigg|_{z=(x_k,v_k)} = v_k - \delta \dot{U}(x_k)$$

- **Improved Euler:** Euler's method can be improved by considering *sequential* updating:

$$x_{k+1} = x_k + \delta \dot{K}(v_k)$$

$$v_{k+1} = v_k - \delta \dot{U}(x_{k+1})$$

- **Leapfrog method:** The leapfrog method uses half-steps gives further improvement with the updates

$$v_k^* = v_k - \frac{\delta}{2}\dot{U}(x_k)$$

$$x_{k+1} = x_k + \delta \dot{K}(v_k^*)$$

$$v_{k+1} = v_k^* - \frac{\delta}{2}\dot{U}(x_{k+1})$$

where $v_k^* \equiv v((k+1/2)\delta)$.

The basic Hamiltonian MCMC algorithm proceeds using the following Metropolis accept/reject approach: we construct an MCMC move $(x_k, v_k) \longrightarrow (x_{k+1}, v_{k+1})$ as follows:

(I) Generate $v_1' \sim Normal_d(\mathbf{0}_d, \mathbf{M})$.

(II) Perform L dynamics updates with time-step δ: for example, for the leapfrog updates,

(i) set v_1', $x_1' = x_k$;

(ii) for $l = 1, \ldots, L-1$

$$v_l^* = v_l' - \frac{\delta}{2}\dot{U}(x_l')$$

$$x_{l+1}' = x_l' + \delta \dot{K}(v_l^*)$$

$$v_{l+1}' = v_l^* - \frac{\delta}{2}\dot{U}(x_{l+1}');$$

(iii) set $x_{k+1}^* = x_L'$ and $v_{k+1}^* = -v_L'$.

(III) Accept (x_{k+1}^*, v_{k+1}^*) with probability

$$\min\left\{1, \frac{\pi_{X,V}(x_{k+1}^*, v_{k+1}^*)}{\pi_{X,V}(x,v)}\right\}$$

where

$$\frac{\pi_{X,V}(x_{k+1}^*, v_{k+1}^*)}{\pi_{X,V}(x,v)} = \exp\{-H(x_{k+1}^*, v_{k+1}^*) + H(x_k, v_k)\}.$$

The proposal in step (II) is reversible by construction, so the proposal mechanism does not appear in the acceptance probability as it cancels in numerator and denominator of the ratio.

Example: Bivariate normal
Suppose that $\pi_X(x) \equiv Normal_2(\mathbf{0}_2, \Sigma)$ where

$$\Sigma = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$$

so that in the Hamiltonian notation

$$U(x) = \frac{1}{2} x^\top \Sigma^{-1} x \qquad\qquad \dot{U}(x) = \Sigma^{-1} x.$$

Suppose also that $\mathbf{M} = \mathbf{I}_2$, so that $\dot{K}(v) = v$. The basic version of Hamiltonian MCMC constructs a move $(x_k, v_k) \longrightarrow (x_{k+1}, v_{k+1})$ as follows:

(I) Generate $v_1' \sim Normal_2(\mathbf{0}_2, \mathbf{I}_2)$.

(II) For the leapfrog updates, for fixed δ and L

(i) set v_1', $x_1' = x_k$;
(ii) for $l = 1, \ldots, L-1$

$$v_l^* = v_l' - \frac{\delta}{2}\Sigma^{-1} x_l'$$

$$x_{l+1}' = x_l' + \delta v_l^*$$

$$v_{l+1}' = v_l^* - \frac{\delta}{2}\Sigma^{-1} x_{l+1}';$$

(iii) set $x_{k+1}^* = x_L'$ and $v_{k+1}^* = -v_L'$.

(III) Accept (x_{k+1}^*, v_{k+1}^*) with probability

$$\min\left\{1, \frac{\pi_{X,V}(x_{k+1}^*, v_{k+1}^*)}{\pi_{X,V}(x, v)}\right\}$$

where

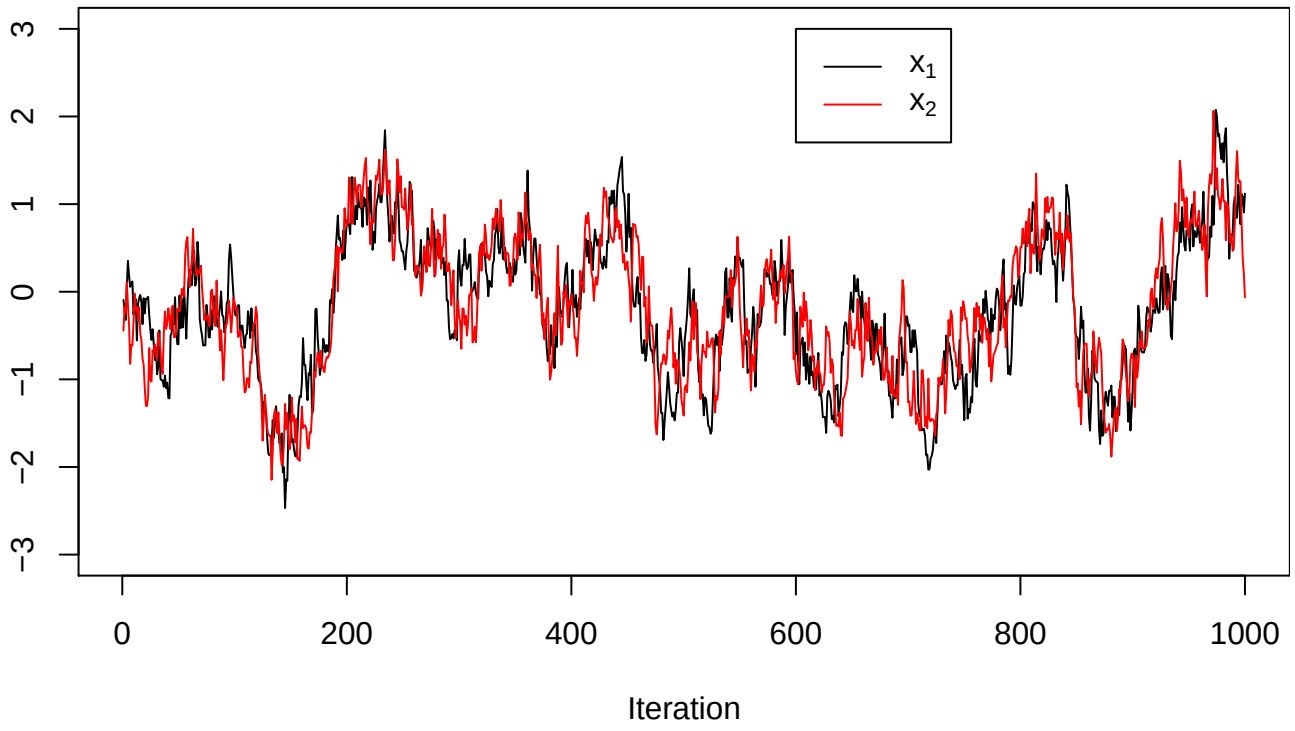
$$\frac{\pi_{X,V}(x_{k+1}^*, v_{k+1}^*)}{\pi_{X,V}(x, v)} = \exp\{-H(x_{k+1}^*, v_{k+1}^*) + H(x_k, v_k)\}.$$

We illustrate this with $\rho = 0.9$, $\delta = 0.1$ and $L = 10$.

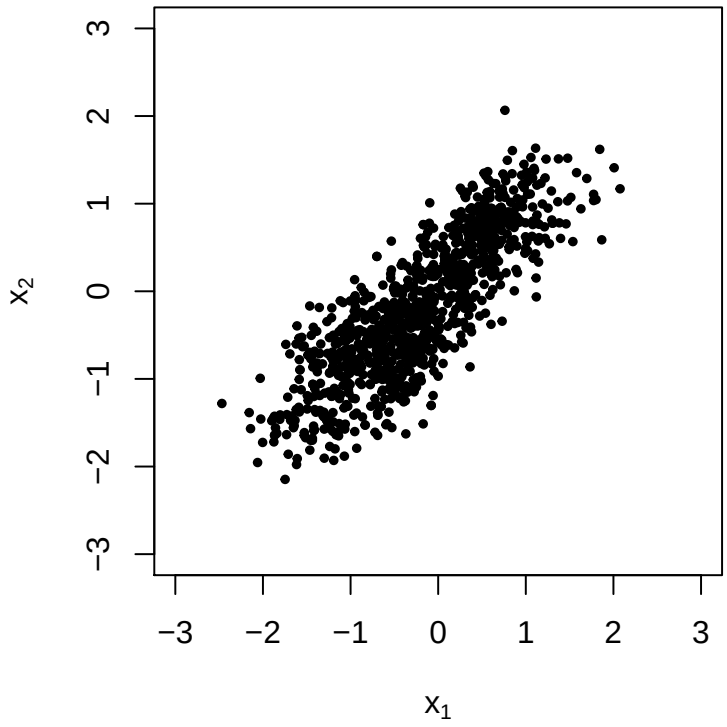
```
rho<-0.9
Sigma<-matrix(c(1,rho,rho,1),2,2)
SigInv<-solve(Sigma)

L<-25
del<-0.25

x<-c(0,0)
nits<-1000
xmat<-matrix(0,nrow=nits,ncol=2)
for(iter in 1:nits){

    #Initialize
    v<-rnorm(2)

    #Leapfrog
    for(l in 1:L){
        vstar<-v-0.5*del*SigInv %*% x
```

```
        xnew<-x+del*vstar
        vnew<-vstar-0.5*del*SigInv %*% xnew
    }

    #Negate
    vnew<--vnew

    #Metropolis

    lpi.old<--0.5*t(x) %*% (SigInv %*% x) - 0.5*sum(v^2)
    lpi.new<--0.5*t(xnew) %*% (SigInv %*% xnew) - 0.5*sum(vnew^2)

    if(log(runif(1)) < lpi.new-lpi.old){
        x<-xnew
        v<-vnew
    }
    xmat[iter,]<-x
}
```

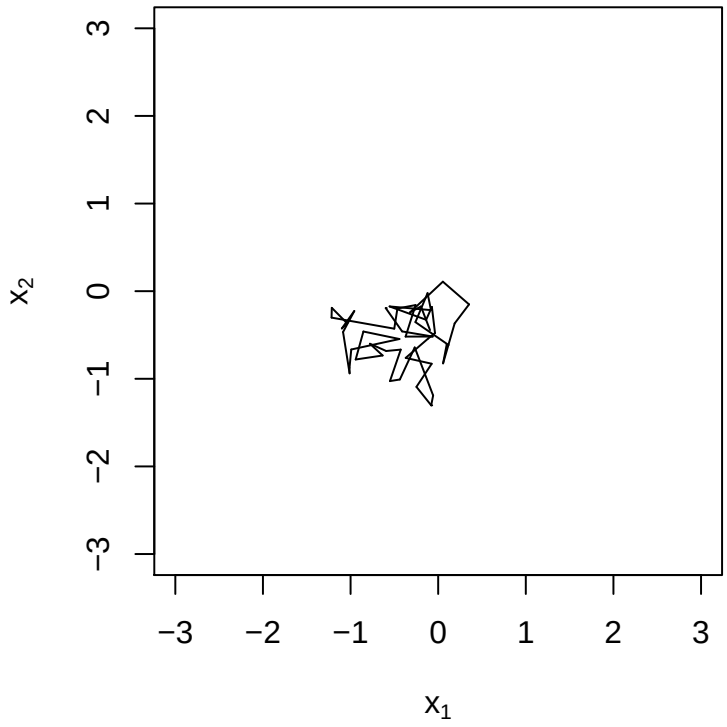
```
par(mar=c(4,3,1,0))
xl<-yl<-range(-3,3)
plot(xmat[,1],type='l',ylim=xl,xlab='Iteration')
lines(xmat[,2],col='red')
legend(600,3,c(expression(x[1]),expression(x[2])),lty=1,col=c('black','red'))
```




```
par(mar=c(4,3,1,0),pty='s')
plot(xmat,xlab=expression(x[1]),ylab=expression(x[2]),xlim=xl,ylim=yl,pch=20,cex=0.8)
```



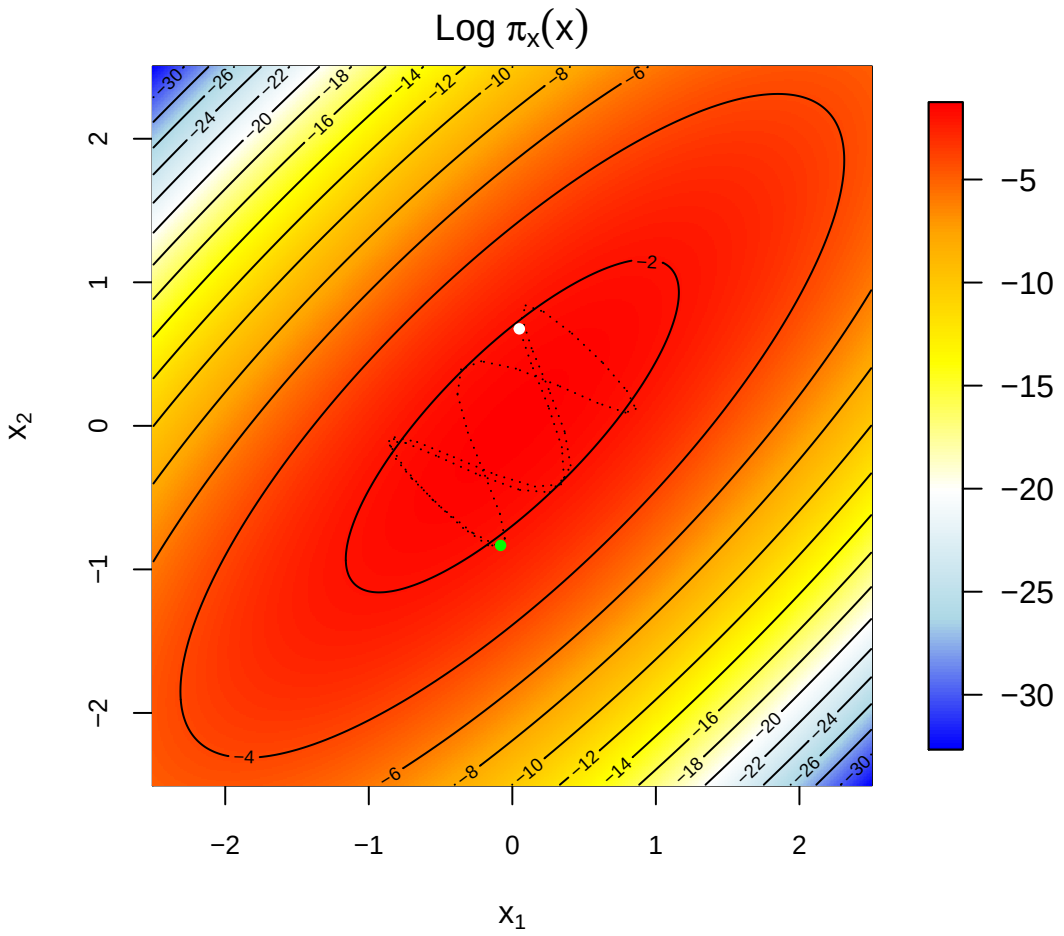
The first 50 steps of the MCMC run are plotted below



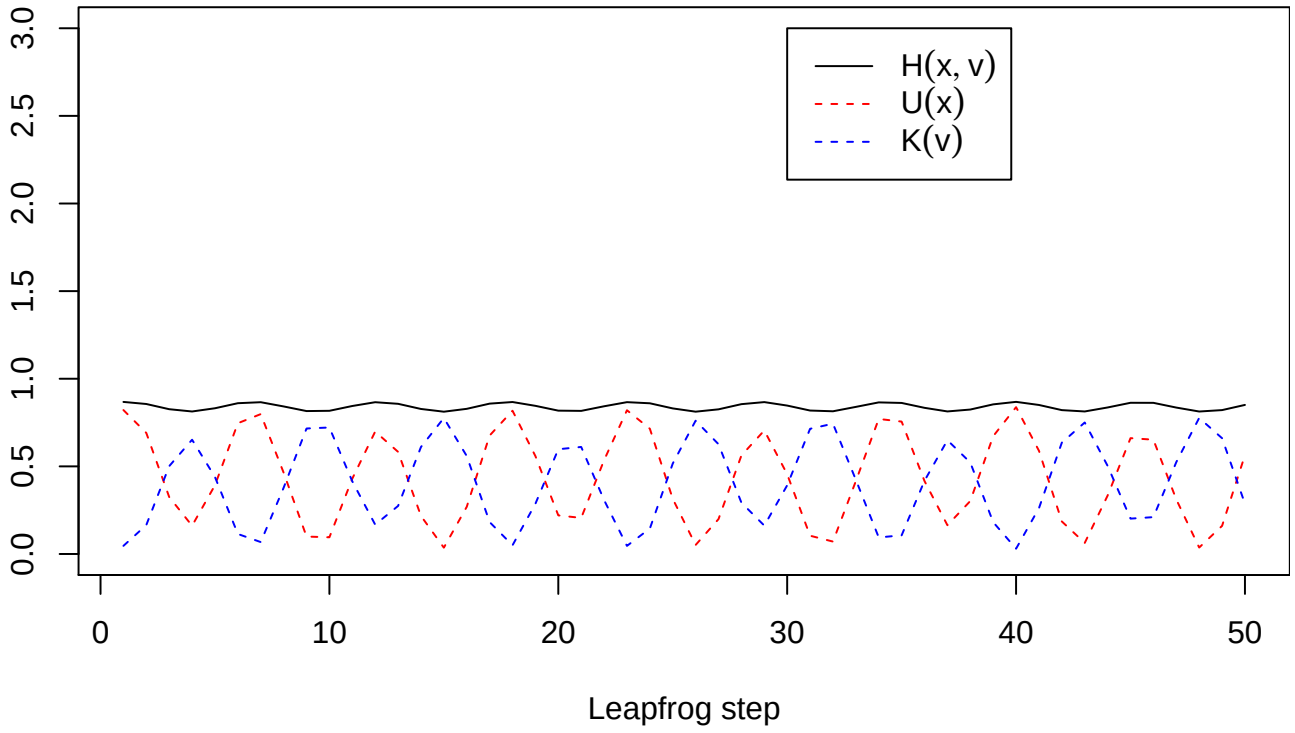
It is also instructive to look at the path of the leapfrog proposal mechanism. For $L = 50$ steps the generated path is as below: the green dot is the starting value of x, and the white dot is the ending position.

```
library(mvnfast)
xvec<-yvec<-seq(-2.5,2.5,by=0.01)
rho<-0.8
Sigma<-matrix(c(1,rho,rho,1),2,2)
SigInv<-solve(Sigma)
```

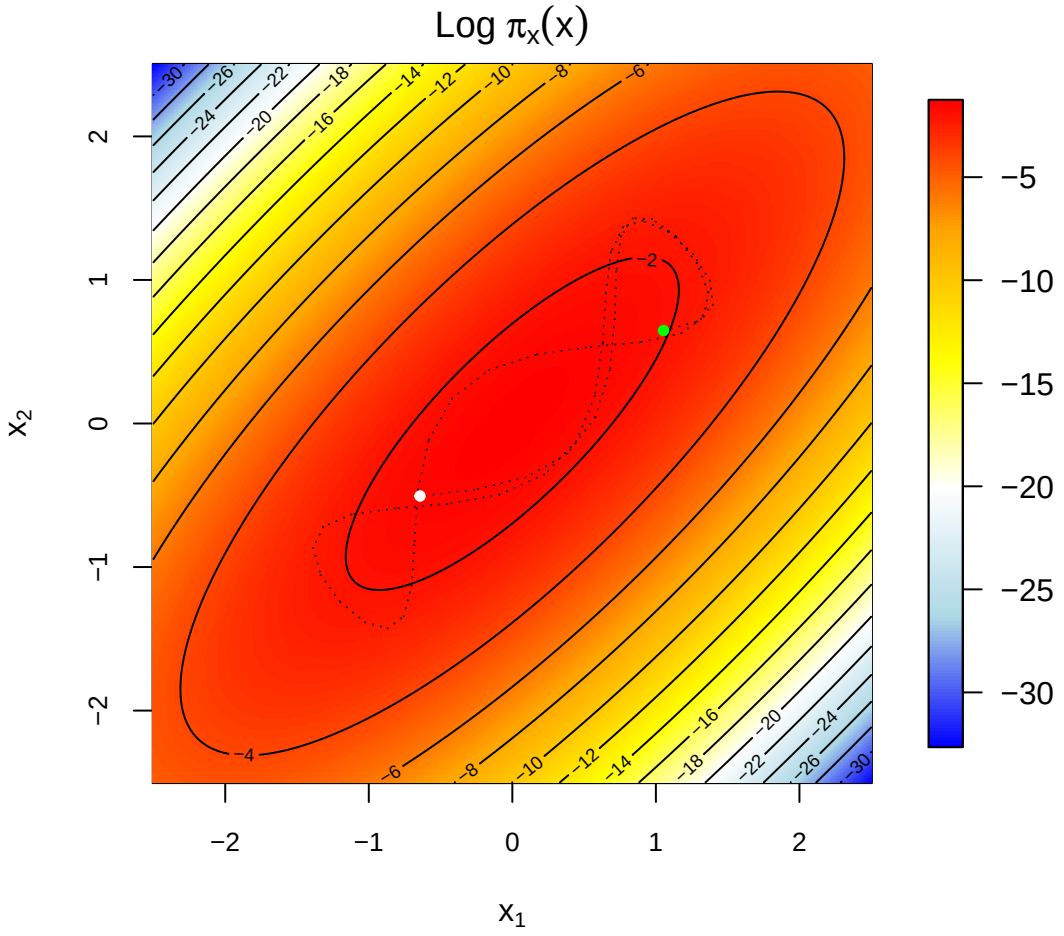
```
dfunc<-function(xv,yv,muv,sigv,lv=TRUE){
    dval<-dmvn(c(xv,yv),muv,sigv,log=lv)
}
f <- Vectorize(dfunc,vectorize.args=c("xv","yv"))
zmat<-outer(xvec,yvec,f,muv=c(0,0),sigv=Sigma)
library(fields,quietly=TRUE)
par(pty='s',mar=c(4,3,2,2))
colfunc <- colorRampPalette(c("blue","lightblue","white","yellow","orange","red"))
image.plot(xvec,yvec,zmat,col=colfunc(200),
           xlab=expression(x[1]),ylab=expression(x[2]),cex.axis=0.8)
contour(xvec,yvec,zmat,add=T,levels=seq(-30,0,by=2))
title(expression(paste('Log ',pi[x](x))))


#Leapfrog run from fixed starting position
L<-50
del<-0.25
x<-runif(2,-2,2)
v<-rnorm(2)
xpath<-matrix(0,nrow=L,ncol=2)
Ham<-U<-K<-rep(0,L)
xpath[1,]<-x
U[1]<-0.5*t(x) %*% (SigInv %*% x)
K[1]<-0.5*sum(v^2)
Ham[1]<-U[1]+K[1]
for(l in 1:(L-1)){
    vstar<-v-0.5*del*SigInv %*% x
    xnew<-x+del*vstar
    vnew<-vstar-0.5*del*SigInv %*% xnew
    x<-xnew
    v<-vnew
    xpath[l+1,]<-x
    U[l+1]<-0.5*t(x) %*% (SigInv %*% x)
    K[l+1]<-0.5*sum(v^2)
    Ham[l+1]<-U[l+1]+K[l+1]
}
for(i in 2:L){lines(xpath[c(i-1,i),1],xpath[c(i-1,i),2],lty=3)}
points(xpath[1,1],xpath[1,2],col='green',pch=20)
points(xpath[L,1],xpath[L,2],col='white',pch=20);
```

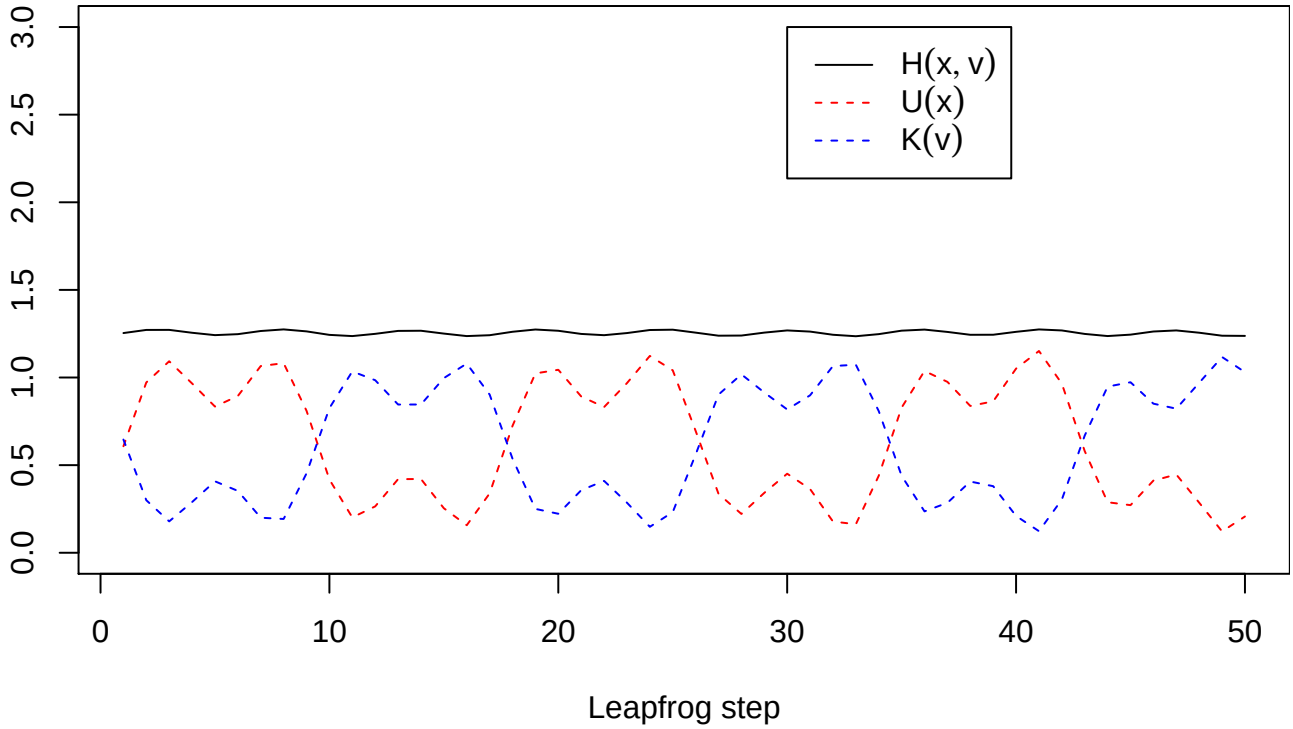


If we examine how the $H(x, v)$, $U(x)$ and $K(v)$ functions vary across the path, we see that $H(x, v)$ is almost constant, whereas the other two functions vary in opposition to each other.



For another realization





For a third realization