MATH 598: TOPICS IN STATISTICS

THE LANGEVIN ALGORITHM

Denote the target pdf $\pi(x)$. Consider the Langevin SDE, and the diffusion process $\{X_t, t > 0\}$

$$dX_t = \frac{1}{2}S(X_t)dt + dW_t$$

where $\{W_t\}$ is Brownian motion, and

$$S(x) = \frac{d}{dx}\log\pi(x) = \nabla \log\pi(x).$$

It can be shown that the invariant measure of this continuous time process is $\pi(x)$. A standard scheme for simulating this diffusion is given by an Euler approximation based on the time-discretization

$$\{X_{k\delta}, k = 1, 2, \ldots\}$$

for some time-step $\delta > 0$ where $X_0 = x_0$, and then for $k = 0, 1, 2, \ldots$

$$X_{k+1} = X_k + \frac{\delta}{2}S(X_k) + \sqrt{\delta}Z_k$$

where $\{Z_k, k = 1, 2, \ldots\}$ are an iid $Normal_d(0, \mathbf{I}_d)$ sequence. The approximation induced by time-discretization does not quite preserve $\pi(x)$ as the invariant distribution. This can be easily corrected using a Metropolis-Hastings step to yield the Metropolis-adjusted Langevin algorithm where the proposal

$$X^\star_{k+1} \sim Normal_d(X_k + \delta S(X_k)/2, \delta\mathbf{I}_d)$$

is *accepted* with probability

$$\min\left\{1, \frac{\pi(X^\star_{k+1})}{\pi(X_k)}\frac{q(X^\star_{k+1}, X_k)}{q(X_k, X^\star_{k+1})}\right\}$$

where

$$q(\mathbf{x}, \mathbf{y}) \propto \exp\left\{-\frac{1}{2\delta}(\mathbf{y} - \mathbf{x} - \delta S(\mathbf{x})/2)^\top(\mathbf{y} - \mathbf{x} - \delta S(\mathbf{x})/2)\right\}$$

Example: Suppose $Y \equiv Gamma(\alpha, 1)$, and consider the pdf of $X = \log Y$. Then for $x \in \mathbb{R}$

$$f_X(x) = e^x f_Y(e^x) = \frac{1}{\Gamma(\alpha)}\exp\{\alpha x - e^x\}.$$

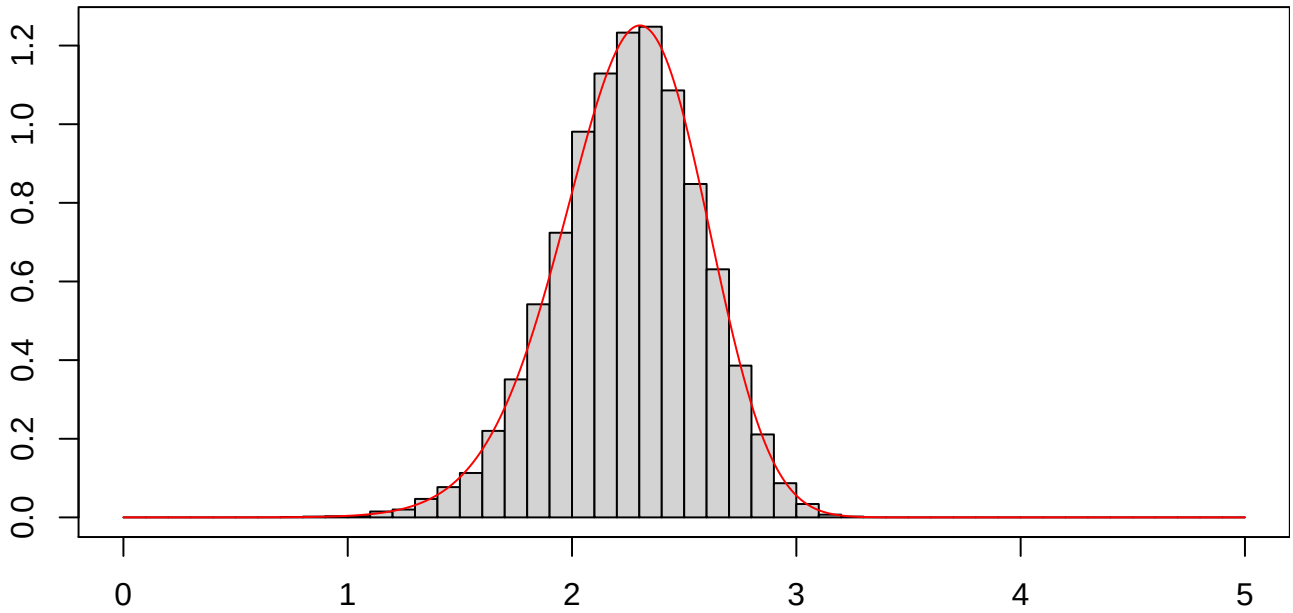
We set $\pi(x) \equiv f_X(x)$ so that

$$\log\pi(x) = \alpha x - e^x - \log\Gamma(\alpha)$$

$$S(x) = \frac{d}{dx}\log\pi(x) = \alpha - e^x$$

It is straightforward to simulate this pdf directly

```
set.seed(23984)
al<-10
N<-10001
Y<-rgamma(N,al,1)
X<-log(Y)
par(mar=c(3,3,3,0))
hist(X,freq=FALSE,breaks=seq(0,5,by=0.1),main='Direct sampling');box()
xvec<-seq(0,5,by=0.001)
yvec<-exp(al*xvec-exp(xvec))/gamma(al)
lines(xvec,yvec,col='red')
```



We may also simulate the Langevin dynamics directly using the Euler approximation. We use $N = 10000$ steps, with a time step of $\delta = 0.01$, taking $x_0 = 2$. We can compute the effective sample size for the sampling approach.

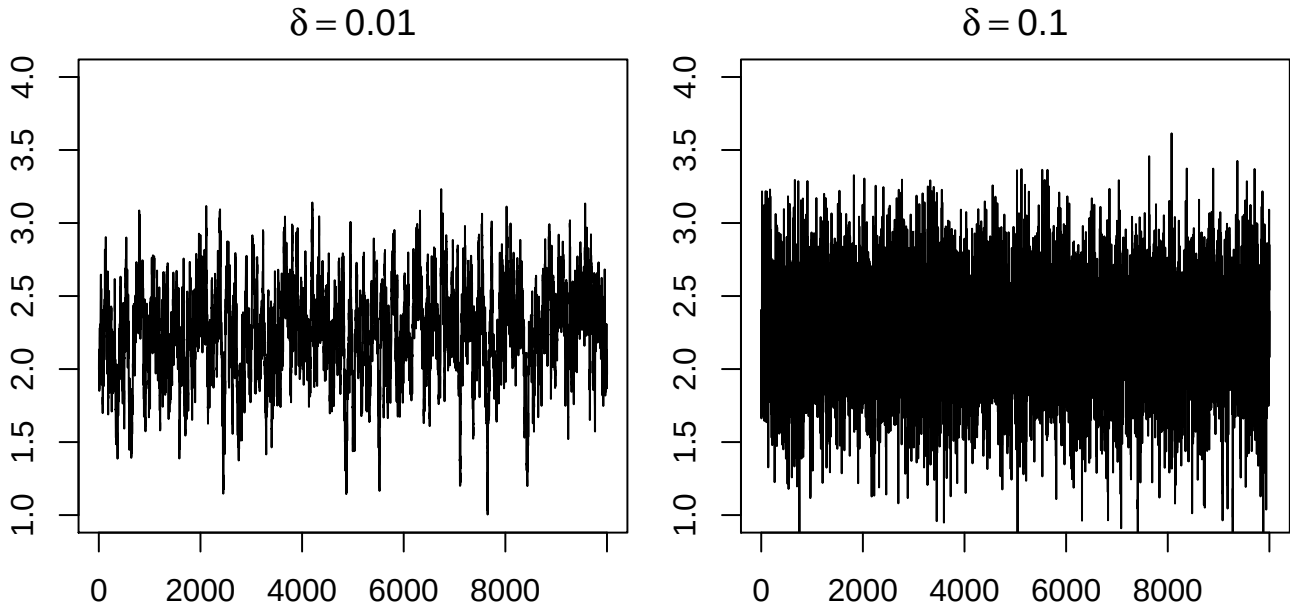
```
del<-0.01
x<-rep(0,N)
x[1]<-2
Z<-rnorm(N)
for(k in 1:N){
        x[k+1] <- x[k] + del*(al - exp(x[k]))/2 + sqrt(del)*Z[k]
}
library(coda)
ESS<-effectiveSize(x)
x.1<-x
```

The low effective sample size of 262.942 indicates a high degree of autocorrelation. The effective sample size can be increased by increasing δ to 0.1.

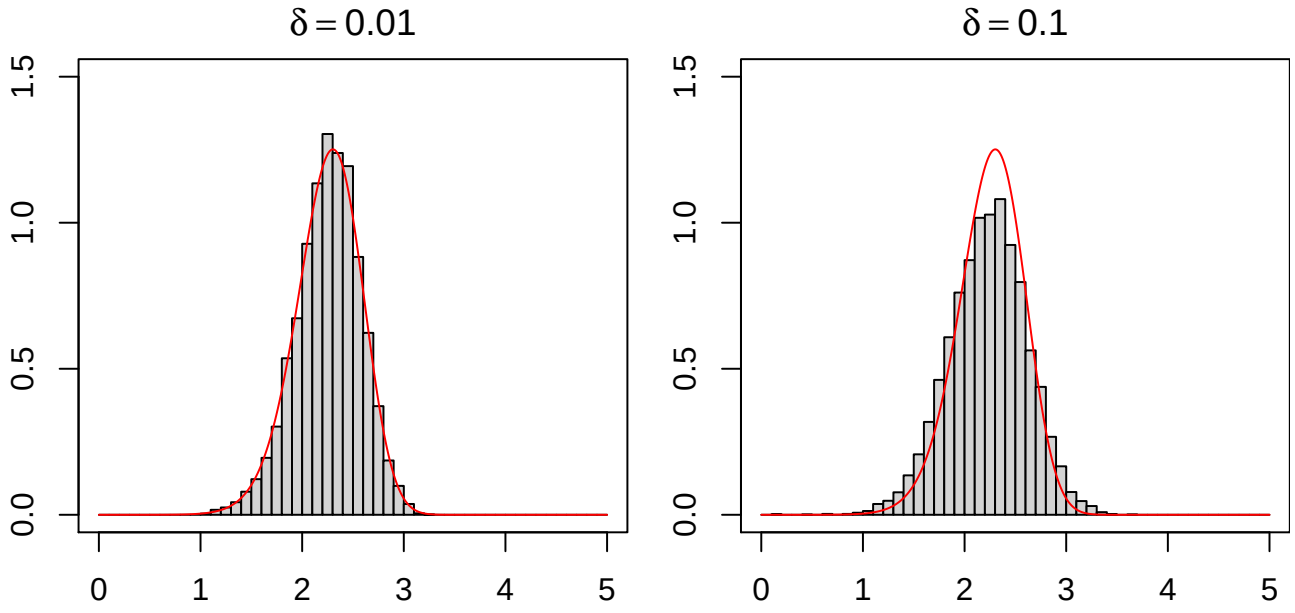
```
del<-0.1
x<-rep(0,N)
x[1]<-2
Z<-rnorm(N)
for(k in 1:N){
        x[k+1] <- x[k] + del*(al - exp(x[k]))/2 + sqrt(del)*Z[k]
}
(ESS<-effectiveSize(x))

+     var1
+ 2988.588

x.2<-x
```

The effective sample size is now 2988.588 so performance seems to be better. We now see how the two schemes with different δ values compare.

```
par(mar=c(3,3,2,0),mfrow=c(1,2))
plot(x.1,type='l',main=expression(delta==0.01),ylim=range(1,4))
plot(x.2,type='l',main=expression(delta==0.1),ylim=range(1,4))
```




```
hist(x.1,freq=FALSE,breaks=seq(0,5,by=0.1),main=expression(delta==0.01),ylim=range(0,1.5));box()
lines(xvec,yvec,col='red')
hist(x.2,freq=FALSE,breaks=seq(0,5,by=0.1),main=expression(delta==0.1),ylim=range(0,1.5));box()
lines(xvec,yvec,col='red')
```



The approximation is poorer when $\delta = 0.1$.

We now implement the MALA enhancement:

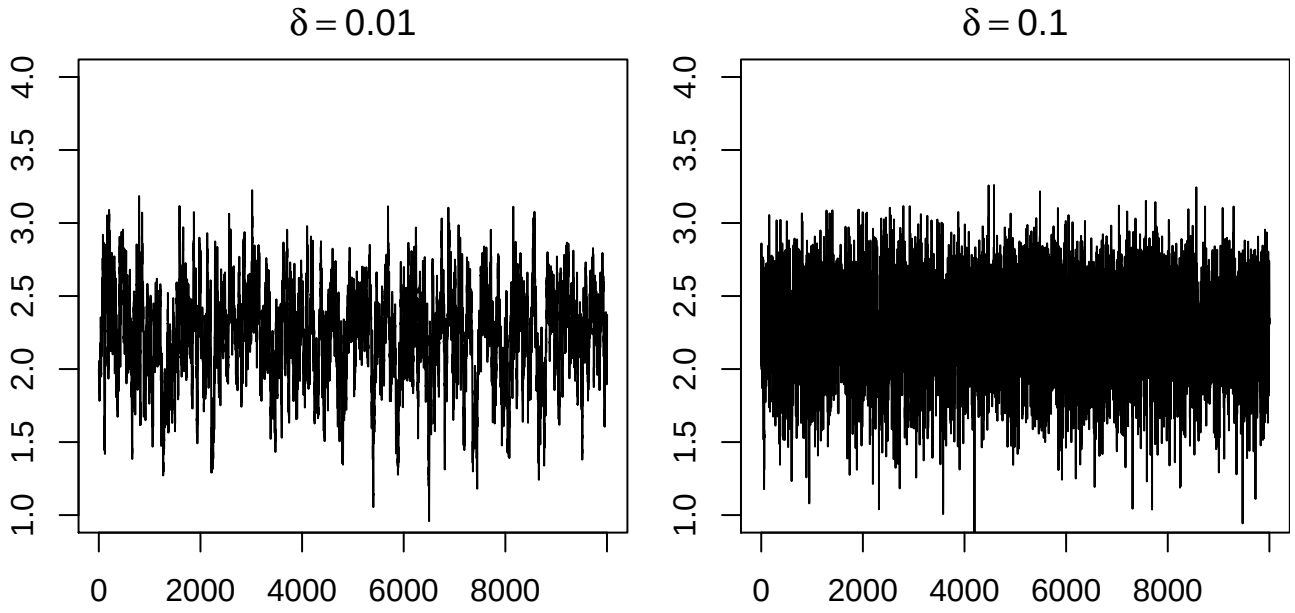
```
log.pi<-function(xv,av){
    return(av*xv - exp(xv))
}
dlog.pi<-function(xv,av){
    return(av- exp(xv))
}
q.func<-function(xv,yv,av,dv){
    m<-xv+0.5*dv*dlog.pi(xv,av)
    return(dnorm(yv,m,sqrt(dv),log=T))
}

del<-0.01
x<-rep(0,N)
x[1]<-2
Z<-rnorm(N)
irej<-0
for(k in 1:N){
        x[k+1] <- x[k] + del*(al - exp(x[k]))/2 + sqrt(del)*Z[k]
    acc<-log.pi(x[k+1],al)-log.pi(x[k],al) + q.func(x[k+1],x[k],al,del) - q.func(x[k],x[k+1],al,del)
    if(log(runif(1)) > acc){
        x[k+1]<-x[k]  #Reject !
        irej<-irej+1
    }
}
(ESS<-effectiveSize(x))

+    var1
+ 255.998

x.3<-x
acc.rate.3<-(N-irej)/N
```

The low effective sample size of 255.998 again indicates a high degree of autocorrelation. The effective sample size can again be increased by increasing δ to 0.1.

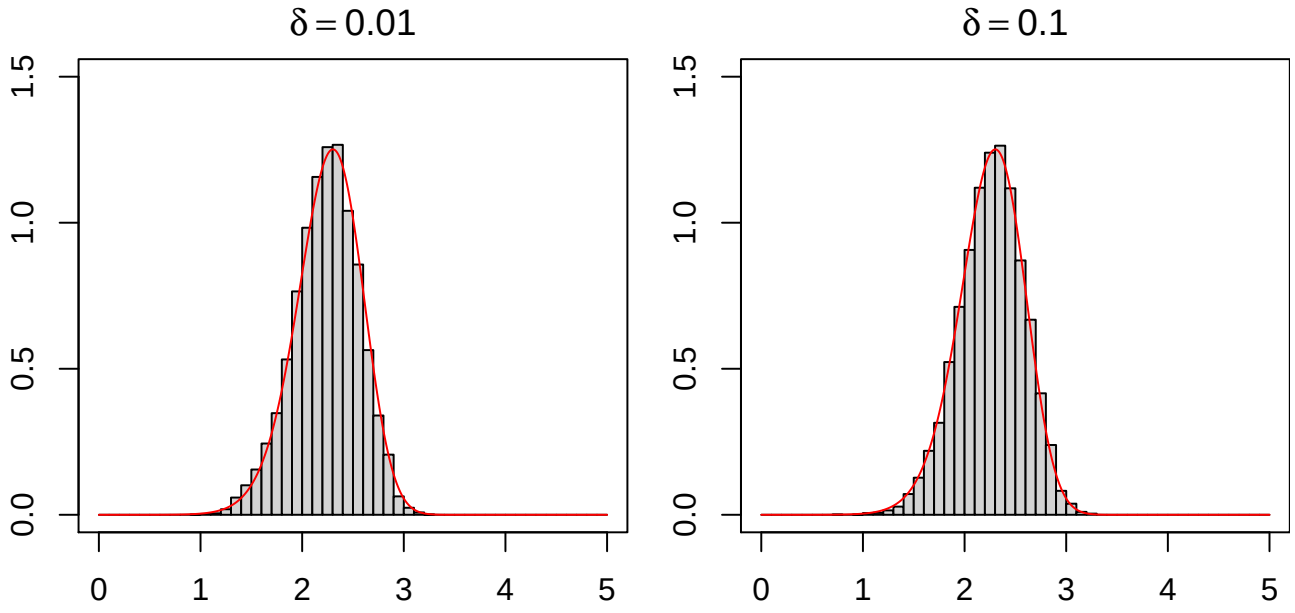
```
del<-0.1
x<-rep(0,N)
x[1]<-2
Z<-rnorm(N)
irej<-0
for(k in 1:N){
        x[k+1] <- x[k] + del*(al - exp(x[k]))/2 + sqrt(del)*Z[k]
    acc<-log.pi(x[k+1],al)-log.pi(x[k],al) + q.func(x[k+1],x[k],al,del) - q.func(x[k],x[k+1],al,del)
    if(log(runif(1)) > acc){
        x[k+1]<-x[k]  #Reject !
        irej<-irej+1
    }
}
(ESS<-effectiveSize(x))

+     var1
+ 3223.515

x.4<-x
acc.rate.4<-(N-irej)/N
print(c(acc.rate.3,acc.rate.4))

+ [1] 0.9976002 0.9200080
```

The effective sample size is now 3223.515. The acceptance rates for the Metropolis step are very high in both algorithms.

```
par(mar=c(3,3,2,0),mfrow=c(1,2))
plot(x.3,type='l',main=expression(delta==0.01),ylim=range(1,4))
plot(x.4,type='l',main=expression(delta==0.1),ylim=range(1,4))
```



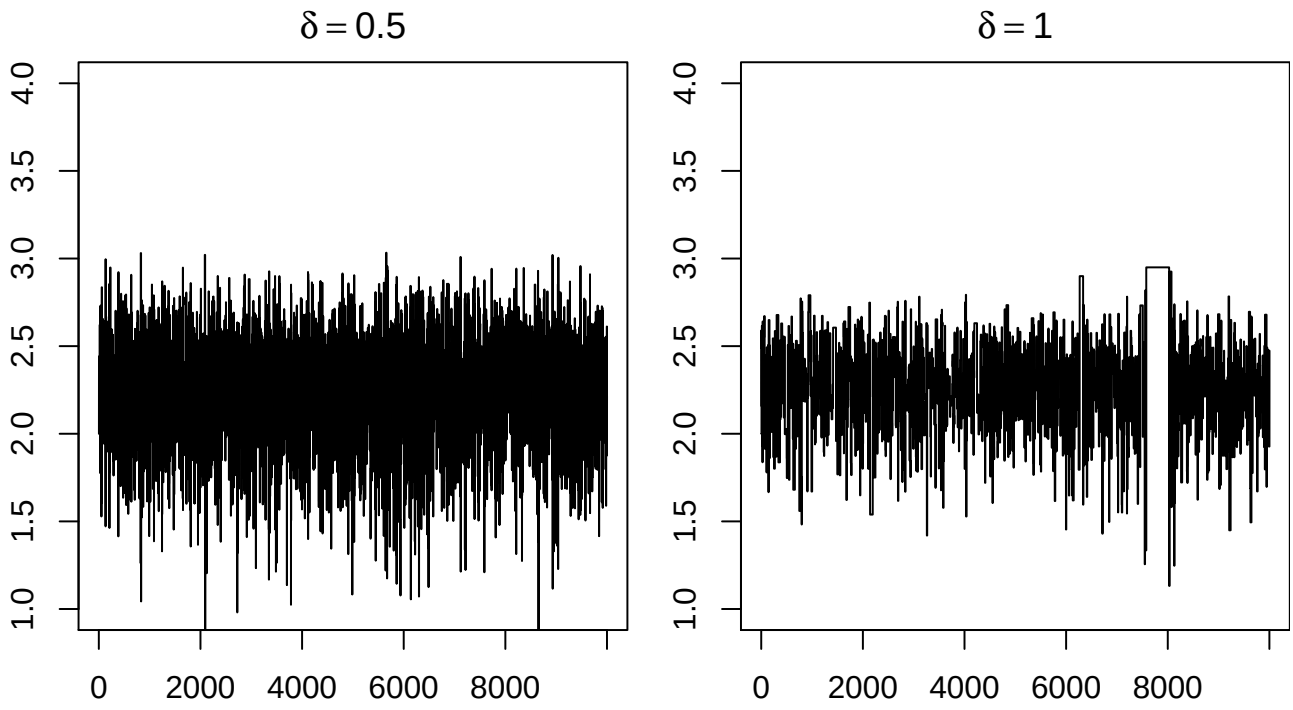

```
hist(x.3,freq=FALSE,breaks=seq(0,5,by=0.1),main=expression(delta==0.01),ylim=range(0,1.5));box()
lines(xvec,yvec,col='red')
hist(x.4,freq=FALSE,breaks=seq(0,5,by=0.1),main=expression(delta==0.1),ylim=range(0,1.5));box()
lines(xvec,yvec,col='red')
```



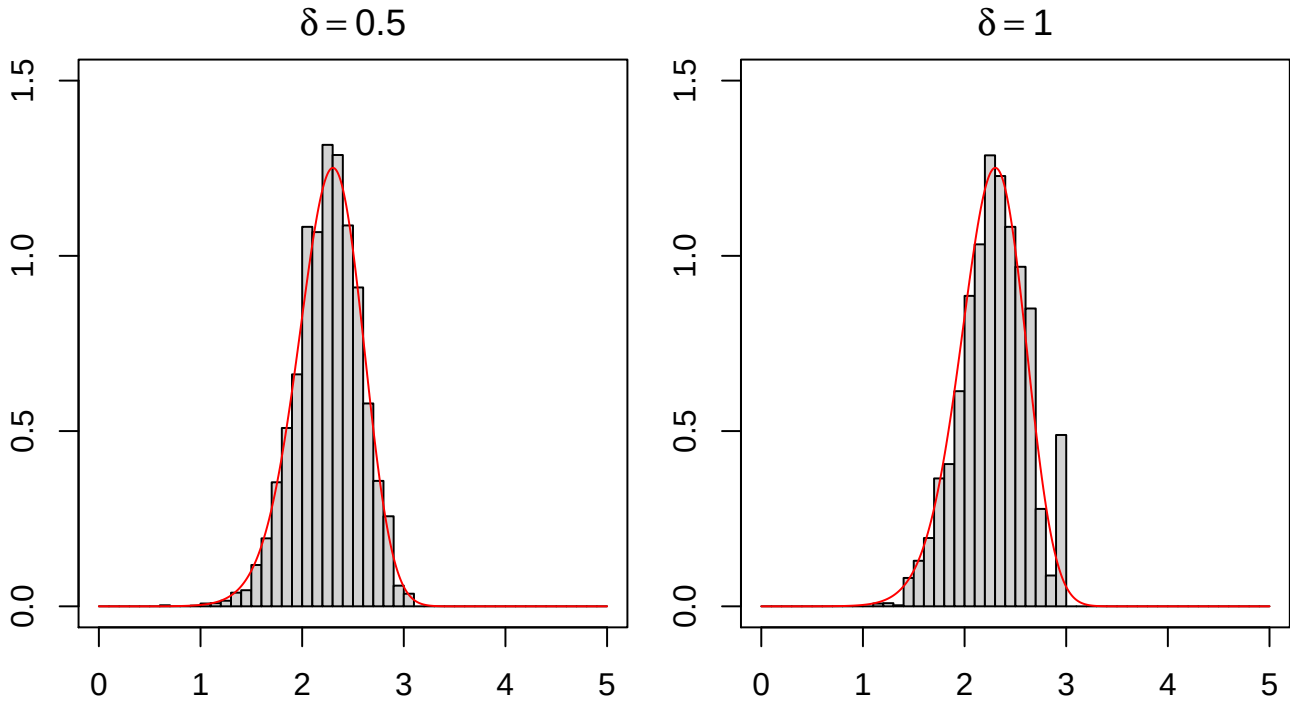
The sampling from the target is good in both cases, the inaccuracy for $\delta = 0.1$ has been corrected.

We now assess results when δ is increased to $\delta = 0.5$ and $\delta = 1.0$.

```
del<-0.5
x<-rep(0,N)
x[1]<-2
Z<-rnorm(N)
irej<-0
for(k in 1:N){
        x[k+1] <- x[k] + del*(al - exp(x[k]))/2 + sqrt(del)*Z[k]
    acc<-log.pi(x[k+1],al)-log.pi(x[k],al) + q.func(x[k+1],x[k],al,del) - q.func(x[k],x[k+1],al,del)
    if(log(runif(1)) > acc){
        x[k+1]<-x[k]  #Reject !
        irej<-irej+1
    }
}
(ESS<-effectiveSize(x))

+     var1
+ 3864.617

x.5<-x
acc.rate.5<-(N-irej)/N

del<-1
x<-rep(0,N)
x[1]<-2
Z<-rnorm(N)
irej<-0
for(k in 1:N){
        x[k+1] <- x[k] + del*(al - exp(x[k]))/2 + sqrt(del)*Z[k]
    acc<-log.pi(x[k+1],al)-log.pi(x[k],al) + q.func(x[k+1],x[k],al,del) - q.func(x[k],x[k+1],al,del)
    if(log(runif(1)) > acc){
        x[k+1]<-x[k]  #Reject !
        irej<-irej+1
    }
}
(ESS<-effectiveSize(x))

+     var1
+ 321.7487

x.6<-x
acc.rate.6<-(N-irej)/N
print(c(acc.rate.5,acc.rate.6))

+ [1] 0.4166583 0.1618838

par(mar=c(3,3,2,0),mfrow=c(1,2))
plot(x.5,type='l',main=expression(delta==0.5),ylim=range(1,4))
plot(x.6,type='l',main=expression(delta==1.0),ylim=range(1,4))
```




```
hist(x.5,freq=FALSE,breaks=seq(0,5,by=0.1),main=expression(delta==0.5),ylim=range(0,1.5));box()
lines(xvec,yvec,col='red')
hist(x.6,freq=FALSE,breaks=seq(0,5,by=0.1),main=expression(delta==1.0),ylim=range(0,1.5));box()
lines(xvec,yvec,col='red')
```



```
var(X)

+ [1] 0.1036018

apply(cbind(x.1,x.2,x.3,x.4,x.5,x.6),2,var)

+        x.1        x.2        x.3        x.4        x.5        x.6
+ 0.10031906 0.14562766 0.10238010 0.10187117 0.09821227 0.10574323
```

There is some theory to suggest that the optimal acceptance rate for the MALA algorithm is around 0.57. We examine the variation in the ESS and acceptance rate as δ varies from 0.1 to 0.5.

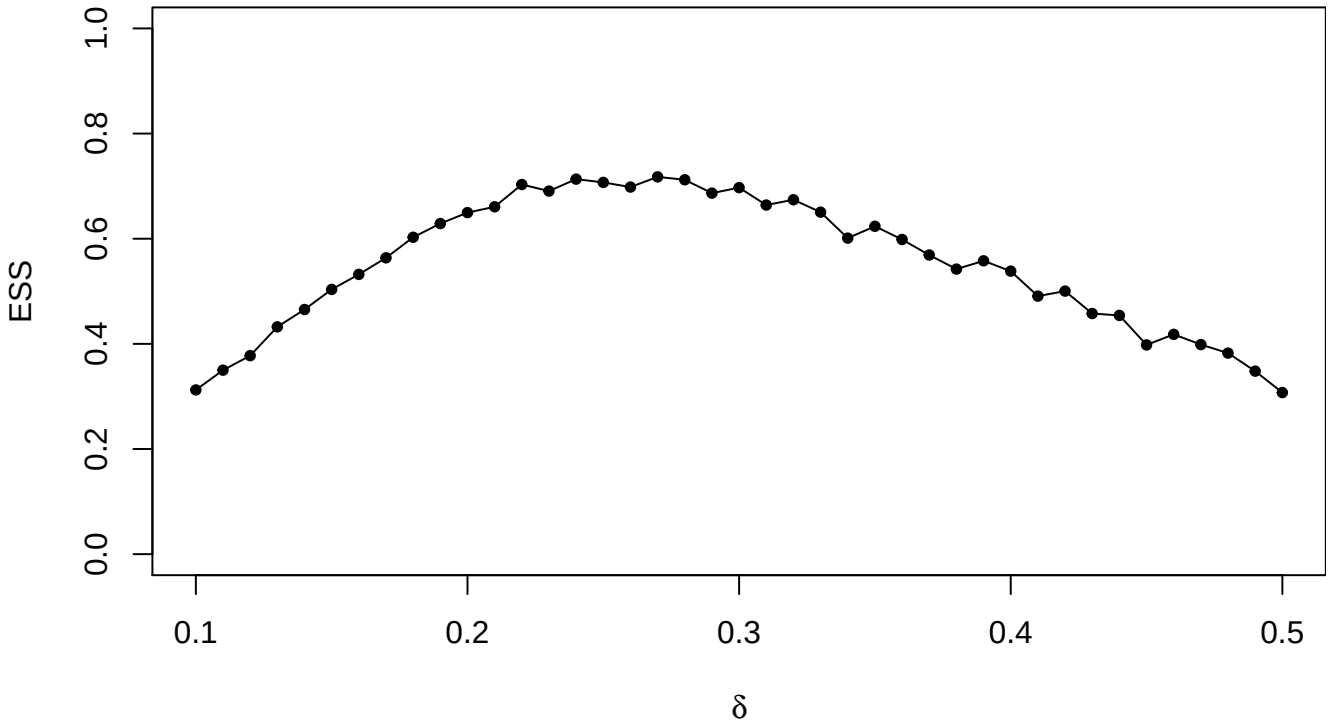
```
N<-100001
del.vec<-acc.vec<-seq(0.1,0.5,by=0.01)
xd<-matrix(0,nrow=N+1,ncol=length(del.vec))
x<-rep(0,N)
for(j in 1:length(del.vec)){
    del<-del.vec[j]
    x[1]<-2
    Z<-rnorm(N)
    irej<-0
    for(k in 1:N){
           x[k+1] <- x[k] + del*(al - exp(x[k]))/2 + sqrt(del)*Z[k]
        acc<-log.pi(x[k+1],al)-log.pi(x[k],al) + q.func(x[k+1],x[k],al,del) - q.func(x[k],x[k+1],al,del)
        if(log(runif(1)) > acc){
            x[k+1]<-x[k]  #Reject !
            irej<-irej+1
        }
    }
    xd[,j]<-x
    acc.vec[j]<-(N-irej)/N
}
ESSd<-effectiveSize(xd)
```

In the plots below, we compare

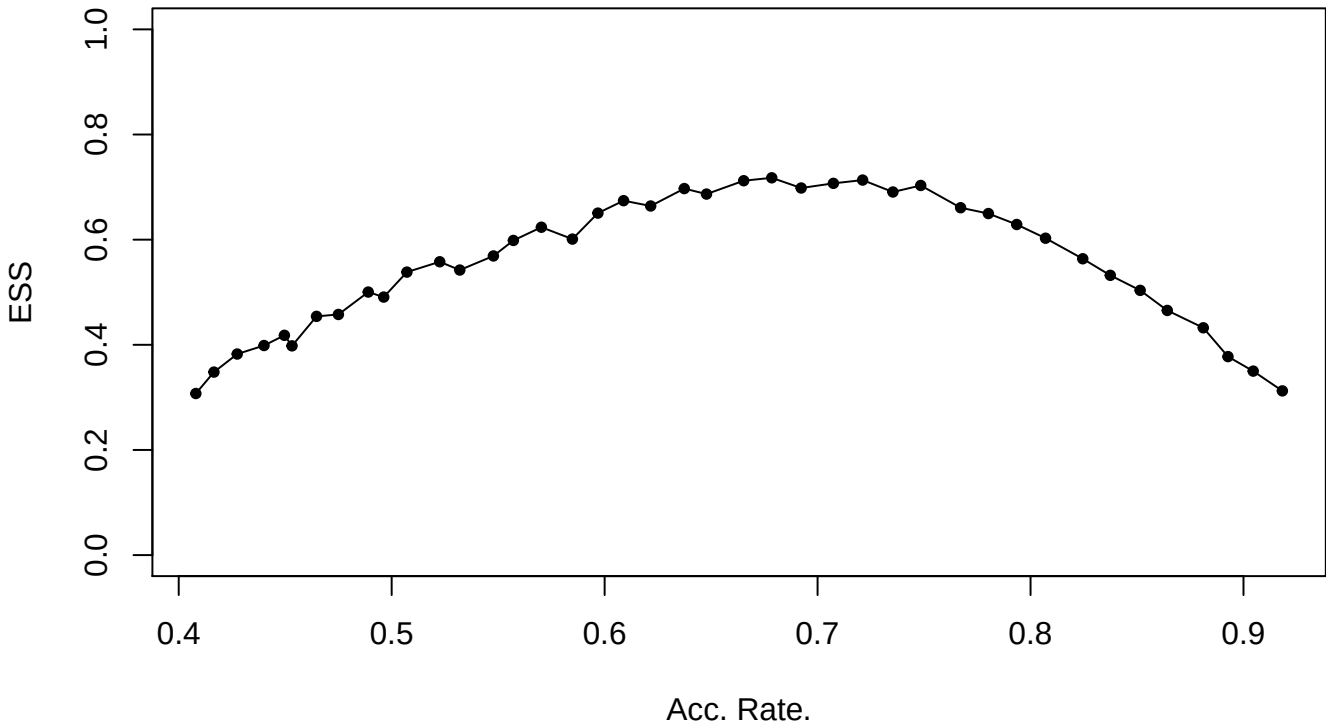
- Effective sample size vs δ;
- Effective sample size vs Metropolis acceptance probability.

The effective sample sizes are plotted as proportions of the run-length $N = 100001$.

```
#cbind(del.vec,ESSd,acc.vec)
par(mar=c(4,4,1,0))
plot(del.vec,ESSd/N,ylim=range(0,1),type='l',xlab=expression(delta),ylab='ESS')
points(del.vec,ESSd/N,pch=20)
```

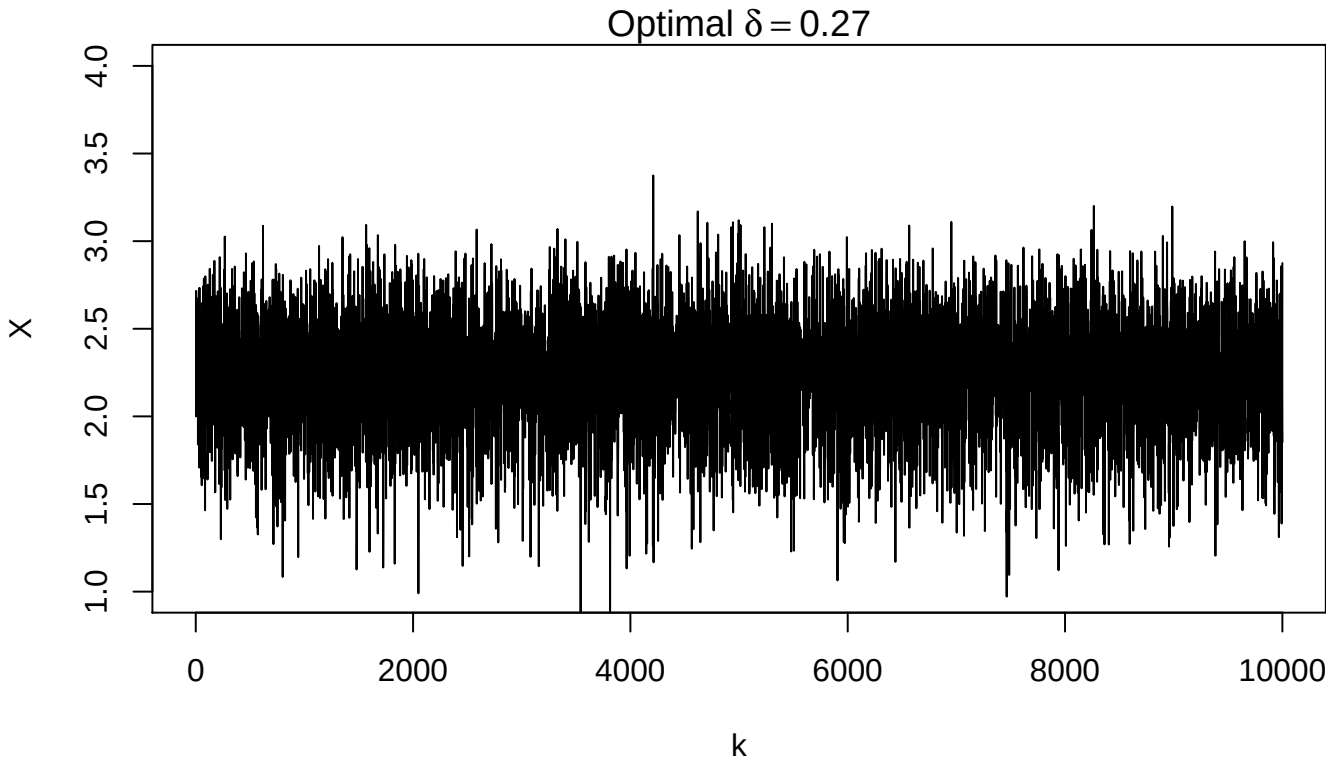



```
plot(acc.vec,ESSd/N,pch=20,ylim=range(0,1),type='l',xlab='Acc. Rate.',ylab='ESS')
points(acc.vec,ESSd/N,pch=20)
```



From the plots we deduce that the optimal δ in this problem is approximately 0.270, which returns an acceptance rate of 0.678 and an effective sample size of 71763.932.

```
par(mar=c(4,4,1,0))
tval<-substitute(paste('Optimal ',delta)==d,list(d = del.vec[which.max(ESSd)]))
plot(xd[1:10000,which.max(ESSd)],type='l',xlab='k',ylab=expression(X),
    main=tval,ylim=range(1,4))
```




```
hist(xd[1:10000,which.max(ESSd)],freq=FALSE,xlab=expression(X),breaks=seq(0,5,by=0.1),
    main=tval,ylim=range(0,1.5));box();lines(xvec,yvec,col='red')
```