MATH 598: TOPICS IN STATISTICS

SEQUENTIAL MONTE CARLO FOR NON-LINEAR STATE SPACE MODELS

For $t = 1, \ldots, n$, a state space model takes the form

$$\begin{aligned} y_t &= g(x_t; \theta) + \epsilon_t \\ x_{t+1} &= h(x_t; \theta) + \varepsilon_t \end{aligned}$$

where only the series $y_1, \ldots, y_T$, is observed. Sequences $\{\epsilon_t\}$ and $\{\varepsilon_t\}$ are independent zero mean random variables. If $y_{1:t} = (y_1, \ldots, y_t)$, $x_{1:t} = (x_1, \ldots, x_t)$ we must carry out for $t = 1, 2, \ldots, n$ the following calculations.

- **Filtering:** Compute $p(x_t|y_{1:t})$: By Bayes theorem

$$\pi(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:(t-1)})}{p(y_t|y_{1:(t-1)})} \propto p(y_t|x_t)p(x_t|y_{1:(t-1)})$$

  where the function $p(y_t|x_t)$ is usually a straightforward conditional density, but the function $p(x_t|y_{1:(t-1)})$ is the density computed at the previous time point **prediction** step.

- **Prediction:** Compute $p(x_{t+1}|y_{1:t})$: we have that

$$\pi(x_{t+1}|y_{1:t}) = \int p(x_{t+1}|x_t)p(x_t|y_{1:t})\, dx_t$$

  where $p(x_{t+1}|x_t)$ is again a conditional density derived from the state equation, and $p(x_t|y_{1:t})$ is the posterior density computed at the time point **filtering** step.

- **Smoothing:** Compute $p(x_t|y_{1:n})$

The first two operations are needed to carry out analysis of the state space models. We consider simulation-based versions of these calculations.

- In the **filtering** step, suppose we have a sample

$$z_t^{(1)}, \ldots, z_t^{(N)}$$

  from $p(x_t|y_{1:(t-1)})$ obtained at the previous prediction step. Then we can obtain a kernel density estimate of the target density

$$\widehat{\pi}(x|y_{1:t}) = \frac{1}{N}\sum_{i=1}^{N} p(y_t|x)\mathbb{1}_{z_t^{(i)}}(x).$$

  We can also produce a sample from the *particle approximation* to $\pi(x_t|y_{1:t})$

$$\widehat{\pi}(x_t|y_{1:t}) = \frac{1}{N}\sum_{i=1}^{N} p(y_t|z_t^{(i)})\delta_{z_t^{(i)}}$$

  by performing a *weighted resampling* from $(z_t^{(1)}, \ldots, z_t^{(N)})$ with weights proportional to $p(y_t|z_t^{(i)})$. Denote this resampled vector

$$v_t^{(1)}, \ldots, v_t^{(N)}$$

- For the **prediction** step, the particle approximation to $p(x_{t+1}|y_{1:t})$ is obtained using a similar Monte Carlo strategy

$$\widehat{\pi}(x_{t+1}|y_{1:t}) = \frac{1}{N}\sum_{i=1}^{N} p(x_{t+1}|v_t^{(i)})$$

  Also, a sample from this approximation can be obtained by propagating the samples $v_t^{(1)}, \ldots, v_t^{(N)}$ forward by sampling the transition densities $p(x_{t+1}|v_t^{(i)})$ for each i. This yields the sample

$$z_{t+1}^{(1)}, \ldots, z_{t+1}^{(N)}$$

  and the recursion continues.

In the state space model, to perform sequential importance sampling (SIS) for

$$\pi_n(x_{1:n}) \equiv \pi(x_{1:n}|y_{1:n}) = \pi_1(x_1)\prod_{j=2}^{n}\widetilde{\pi}_j(x_j|x_{1:(j-1)})$$

we might use the IS density

$$p_n(x_{1:n}) = p_1(x_1)\prod_{j=2}^{n}\widetilde{p}_j(x_j|x_{1:(j-1)})$$

with weights given by $w_1(x_1) = \pi_1(x_1)/p_1(x_1)$ and

$$\begin{aligned} w_j(x_{1:j}) &= \pi_1(x_1)\prod_{k=2}^{j}\widetilde{\pi}_k(x_k|x_{1:(k-1)}) \\ &= w_{j-1}(x_{1:(j-1)})\frac{\widetilde{\pi}_j(x_j|x_{1:(j-1)})}{\widetilde{p}_j(x_j|x_{1:(j-1)})} \\ &= w_{j-1}(x_{1:(j-1)})\frac{\pi_j(x_{1:j})}{\pi_{j-1}(x_{1:(j-1)})\widetilde{p}_j(x_j|x_{1:(j-1)})}. \end{aligned}$$

If

$$\pi_j(x_{1:j}) = \pi(x_{1:j}|y_{1:j}) = \frac{g_j(x_{(1:j)})}{Z_j}$$

the normalizing constants Z_j for the π_j for $j = 1, \ldots, n$ are unknown, these weights can be replaced by $w_1(x_1) = g_1(x_1)/p_1(x_1)$ and

$$w_j(x_{1:j}) = w_{j-1}(x_{1:(j-1)})\frac{g_j(x_{1:j})}{g_{j-1}(x_{1:(j-1)})\widetilde{p}_j(x_j|x_{1:(j-1)})} = w_{j-1}(x_{1:(j-1)})\alpha_j(x_{1:j})$$

say. The Monte Carlo computation is carried out by sampling

$$x_1^{(1)}, \ldots, x_1^{(N)}$$

from p_1, then at step $j \geq 2$ sampling for $i = 1, \ldots, N$

$$x_j^{(1)}, \ldots, x_j^{(N)} \quad \text{from} \quad \widetilde{p}_j(x_j|x_{1:(j-1)}^{(i)})$$

The importance sampling weights are $w_1(x_1^{(i)})$ and, for $j \geq 2$,

$$w_j(x_{1:j}^{(i)}) = w_{j-1}(x_{1:(j-1)}^{(i)})\alpha_j(x_{1:j}^{(i)})$$

We then have ultimately

$$\widehat{\pi}_n(\mathbf{x}) = \sum_{i=1}^{N} W_n^{(i)}\delta_{x_{1:n}^{(i)}}(\mathbf{x})$$

where

$$W_n^{(i)} = \frac{w_n(x_{1:n}^{(i)})}{N\widehat{Z}_n} \qquad \widehat{Z}_n = \frac{1}{N}\sum_{i=1}^{N} w_n(x_{1:n}^{(i)})$$

in the usual way.

In a simple state-space model, write

$$\text{Observation equation} \quad : \quad p_{Y|X}(y_j|x_j)$$

$$\text{State equation} \quad : \quad p_X(x_j|x_{j-1})$$

with $X_1 \sim p_{X_1}$. In light of the observed data $y_{1:n} = (y_1, \ldots, y_n)$, the posterior of interest is

$$\pi_n(x_{1:n}) = p(x_{1:n}|y_{1:n}) = \frac{p(x_{1:n}, y_{1:n})}{p(y_{1:n})} = \frac{g_n(x_{1:n}, y_{1:n})}{Z_n(y_{1:n})}$$

We have

$$g_n(x_{1:n}, y_{1:n}) = p_{X_1}(x_1) \left\{ \prod_{j=2}^{n} p_X(x_j|x_{j-1}) \right\} \left\{ \prod_{j=1}^{n} p_{Y|X}(y_j|x_j) \right\}$$

and

$$Z_n(y_{1:n}) = \int g_n(x_{1:n}, y_{1:n}) \, dx_{1:n}$$

Now, for SMC, the optimal choice of IS density $\widetilde{p}_j(x_j|x_{1:(j-1)})$ is $\widetilde{\pi}_j(x_j|x_{1:(j-1)})$ which in the simple state-space model is given by

$$\begin{aligned}
\widetilde{\pi}_j(x_j|x_{1:(j-1)}) \equiv \widetilde{\pi}_j(x_j|y_{1:j}, x_{1:(j-1)}) &= \frac{p_{Y|X}(y_j|x_{1:j}) p_X(x_j|x_{1:(j-1)})}{p(y_j|x_{1:(j-1)})} \\
&= \frac{p_{Y|X}(y_j|x_j) p_X(x_j|x_{j-1})}{p(y_j|x_{j-1})} \\
&= \frac{p_{Y|X}(y_j|x_j) p_X(x_j|x_{j-1})}{\displaystyle\int p_{Y|X}(y_j|x) p_X(x|x_{j-1}) \, dx}
\end{aligned}$$

as, conditional on $x_{1:(j-1)}$, x_j is independent of $y_{1:(j-1)}$. The optimal choice is typically not feasible: recall that we need successive samples from the IS densities

$$\widetilde{p}_j(x_j|x_{1:(j-1)}) = \widetilde{\pi}_j(x_j|y_{1:j}, x_{1:(j-1)}) = \widetilde{\pi}_j(x_j|y_j, x_{(j-1)})$$

so instead we might choose

$$\widetilde{p}_j(x_j|x_{1:(j-1)}) = p_X(x_j|x_{1:(j-1)})$$

which typically is straightforward to sample from, and yields the recursive weight calculation

$$w_j(x_{1:j}) = w_{j-1}(x_{1:(j-1)}) p_{Y|X}(y_j|x_j)$$

Stochastic Volatility Model: We aim to carry out sequential Monte Carlo (SMC) for a

$$\text{Observation equation} \quad : \quad p_{Y|X}(y_j|x_j) \equiv \mathcal{N}(0, \exp\{2x_j\})$$

$$\text{State equation} \quad : \quad p_X(x_j|x_{j-1}) \equiv \mathcal{N}(\mu + \phi(x_{j-1} - \mu), \sigma^2)$$

with $p_{X_1}(x) \equiv \mathcal{N}(\mu, \sigma^2/(1-\phi^2))$. We regard μ, ϕ and σ as known constants. In the simulation below, $\mu = -2, \phi = 0.95$ an $\sigma = 0.3$.

This model is used in the analysis of financial time series: it is a non-linear state-space model, due to the non-linear dependence of the distribution of y_j on x_j. Here, for $j \geq 2$,

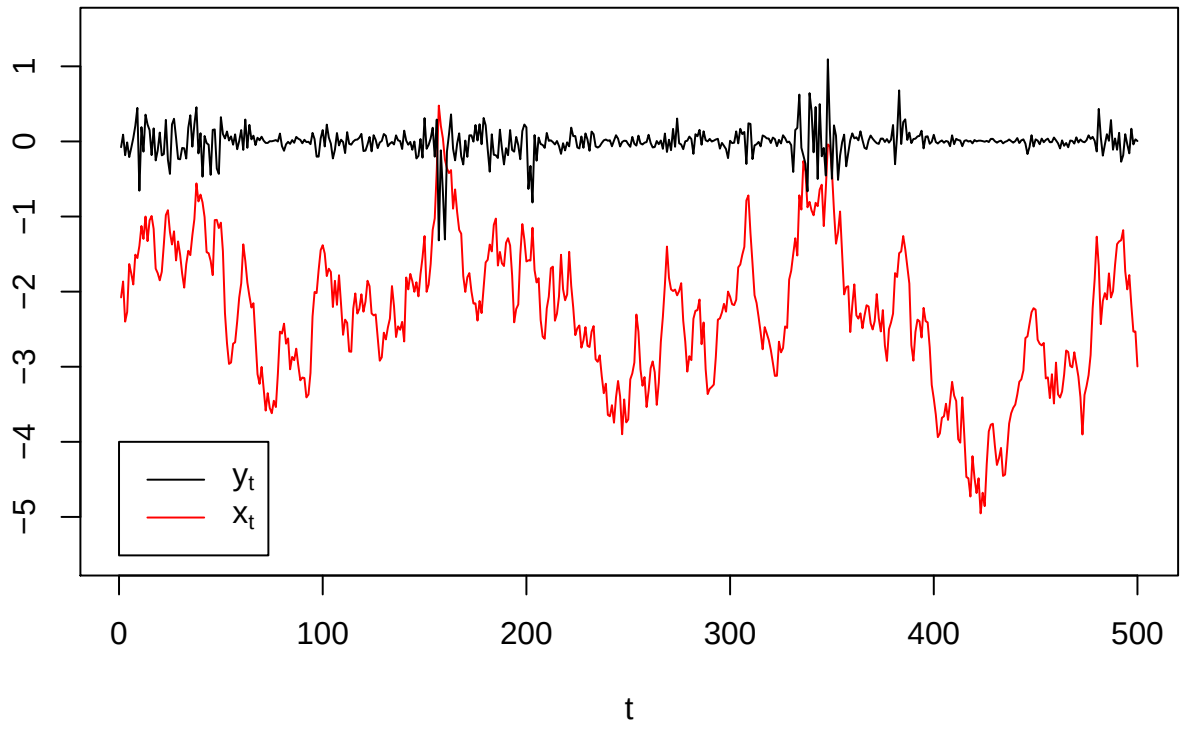
$$p_{Y|X}(y_j|x_j) = \left(\frac{e^{-2x_j}}{2\pi} \right)^{1/2} \exp\{-e^{-2x_j} y_j^2/2\}$$

and the recursive weight calculation is based on

$$w_j(x_{1:j}) = \prod_{k=1}^{j} \frac{\widetilde{\pi}_k(x_k|x_{1:(k-1)})}{\widetilde{p}_k(x_k|x_{1:(k-1)})} = \prod_{k=1}^{j} p_{Y|X}(y_k|x_k)$$

that is, the likelihood up to observation j.

```r
set.seed(1231)
n<-500
mu<--2; phi<-0.95; sig<-0.3
xvec<-yvec<-rep(0,n)
xvec[1]<-rnorm(1,mu,sig/sqrt(1-phi^2))
for(j in 2:n){xvec[j]<-rnorm(1,mu+phi*(xvec[j-1]-mu),sig)}
yvec<-rnorm(n)*exp(xvec)
par(mar=c(4,4,1,2))
plot(1:n,xvec,type="l",xlab="t",col='red',ylab=' ',ylim=range(-5.5,1.5))
lines(1:n,yvec,pch=19,cex=0.5)
legend(0,-4,c(expression(y[t]),expression(x[t])),col=c('black','red'),lty=1)
```



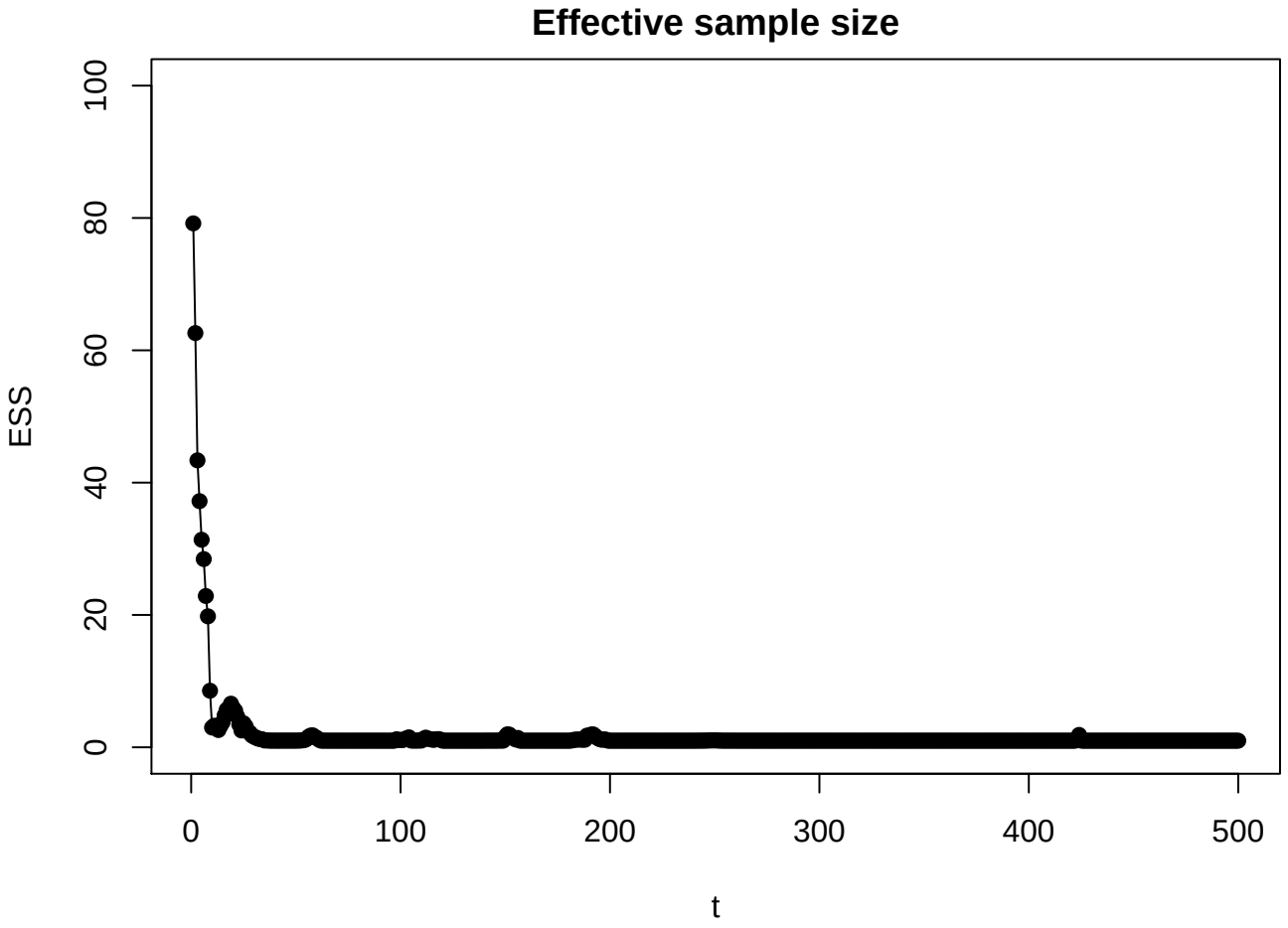

```r
N<-100
Xmat<-wmat<-Wmat<-matrix(0,nrow=n,ncol=N)
Xmat[1,]<-rnorm(N,mu,sig/sqrt(1-phi^2))
wmat[1,]<-dnorm(yvec[1],0,exp(Xmat[1,]))
Wmat[1,]<-wmat[1,]/sum(wmat[1,])
ESS.1<-rep(0,n)
ESS.1[1]<-1/sum(Wmat[1,]^2)
for(j in 2:n){
        Xmat[j,]<-rnorm(N)*sig+(mu+phi*(Xmat[j-1,]-mu))
        wmat[j,]<-wmat[j-1,]*dnorm(yvec[j],0,exp(Xmat[j,]))
        Wmat[j,]<-wmat[j,]/sum(wmat[j,])
    ESS.1[j]<-1/sum(Wmat[j,]^2)
}
```

In order to assess the effectiveness of the SMC procedure we may track the *effective sample size* (ESS) where

$$\text{ESS} = \left(\sum_{i=1}^{N} \left\{ W_j^{(i)} \right\}^2 \right)^{-1}$$

We have $1 \leq \text{ESS} \leq N$, and large values of the ESS statistic indicate more effective performance. Using this specific SIS proposal, we observe quite poor performance.

```
par(mar=c(4,4,2,1))
plot(c(1:n),ESS.1[1:n],type="l",xlab='t',ylab="ESS",ylim=range(0,N))
points(c(1:n),ESS.1[1:n],pch=19)
title('Effective sample size')
```



We may also estimate the unknown state by $\widehat{x}_t$ computed from the sampled points. We compute the estimated (filtered) state by using the kernel density

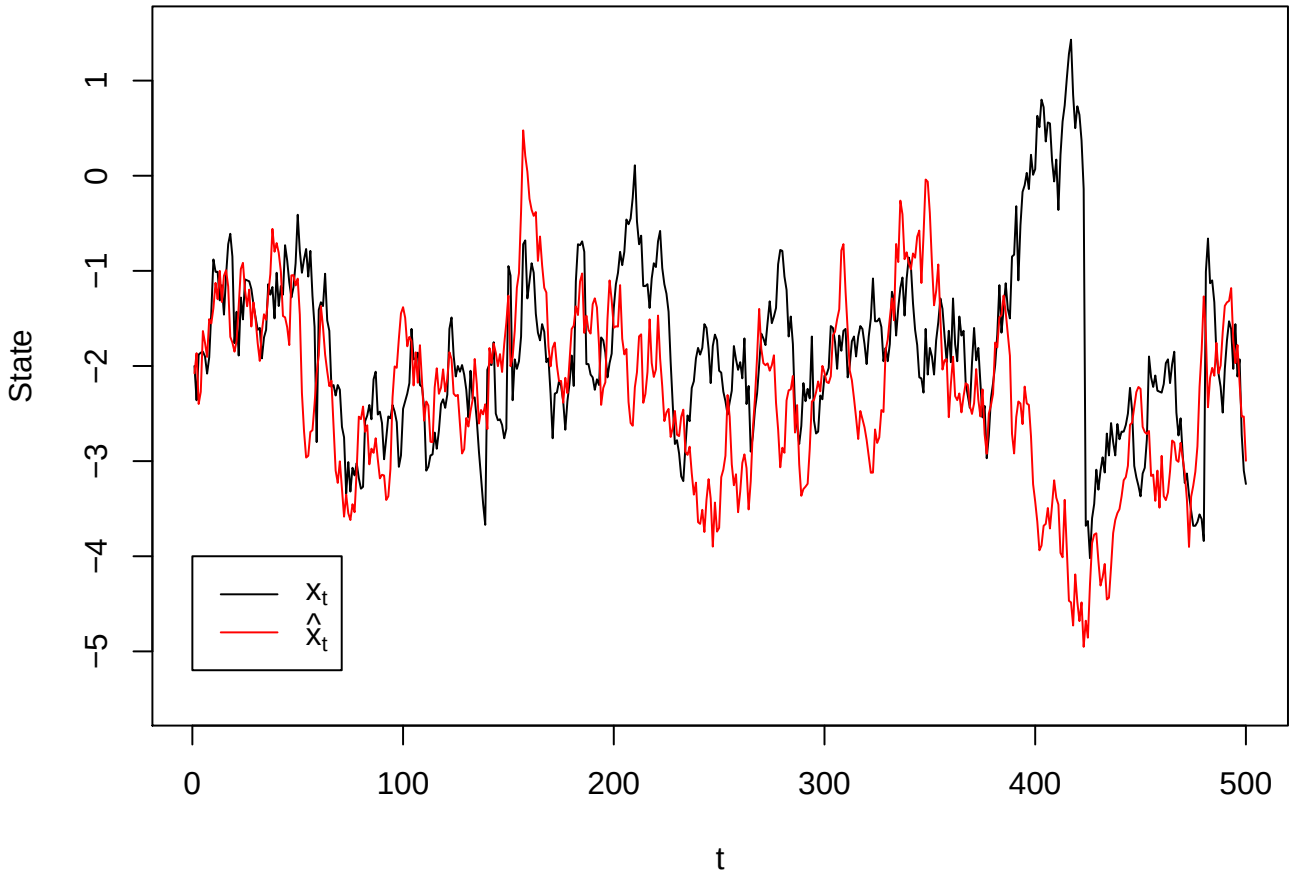
$$\widehat{\pi}(x|y_{1:t}) = \frac{1}{N}\sum_{i=1}^{N}\phi(y_t, 0, \exp\{2x\})\mathbb{1}_{z_t^{(i)}}(x).$$

where $\phi(x;\mu,\sigma^2)$ is the Normal pdf with parameters μ and σ^2, or the mean estimate derived from the particles

$$\widehat{x}_t = \frac{1}{N}\sum_{i=1}^{N} W_t^{(i)} z_t^{(i)}.$$

```
xv<-yv<-seq(-5,5,by=0.01)
dnsv<-function(yval,xval){
        return(dnorm(yval,0,exp(xval)))
}
dns0<-function(x1,x0){
        return(dnorm(x1,mu,sig/sqrt(1-phi^2)))
}
dns<-function(x1,x0){
        return(dnorm(x1,mu+phi*(x0-mu),sig))
}
```

```
Xhat<-matrix(0,nrow=length(yv),ncol=n)  #Filtering
Yhat<-matrix(0,nrow=length(yv),ncol=n)  #Prediction for y
for(j in 1:n){
        if(j == 1){
                fmat<-outer(xv,Xmat[1,],dns0)
                Xhat[,j]<-apply(fmat,1,function(x){return(sum(x*Wmat[j,]))})
        }else{
                fmat<-outer(xv,Xmat[j-1,],dns)
                Xhat[,j]<-apply(fmat,1,function(x){return(sum(x*Wmat[j,]))})
        }
        fmat<-outer(yv,exp(Xmat[j,]),dnsv)
        Yhat[,j]<-apply(fmat,1,function(x){return(sum(x*Wmat[j,]))})
}
par(mar=c(4,4,2,1))
plot(xv[apply(Xhat,2,which.max)],xlab='t',type='l',ylab='State',ylim=range(-5.5,1.5))
lines(xvec,col='red')
legend(0,-4,c(expression(x[t]),expression(hat(x)[t])),col=c('black','red'),lty=1)
```



Particle Degeneracy: A problem with this strategy is that the weights start to **degenerate** as j increases;

- the weights that define the Monte Carlo estimate $\widehat{\pi}_n(\mathbf{x})$ are very small for most i, with only a few large weights.
- the empirical variance of the collection of unnormalized weights $\{w_j(x_j^{(i)})\}$ for $i = 1, \ldots, N$ increases
- this is mitigated, but not resolved by a better choice of $\widetilde{p}_j(x_j|x_{1:(j-1)})$

- could try a normal approximation of

$$\begin{aligned} p_j(x_j|y_{1:j}, x_{1:(j-1)}) &\propto p_{Y|X}(y_j|x_j)p_X(x_j|x_{j-1}) \\ &= \exp\left\{-x_j - \frac{y_j^2}{2e^{2x_j}} - \frac{1}{2\sigma^2}(x_j - \mu_j)^2\right\} \end{aligned}$$

  where $\mu_j = \mu + \phi(x_{j-1} - \mu)$.

Here the mode of

$$g_j(x|y_j, x_{j-1}) = \exp\left\{-x - \frac{y_j^2}{2e^{2x}} - \frac{1}{2\sigma^2}(x - \mu_j)^2\right\}$$

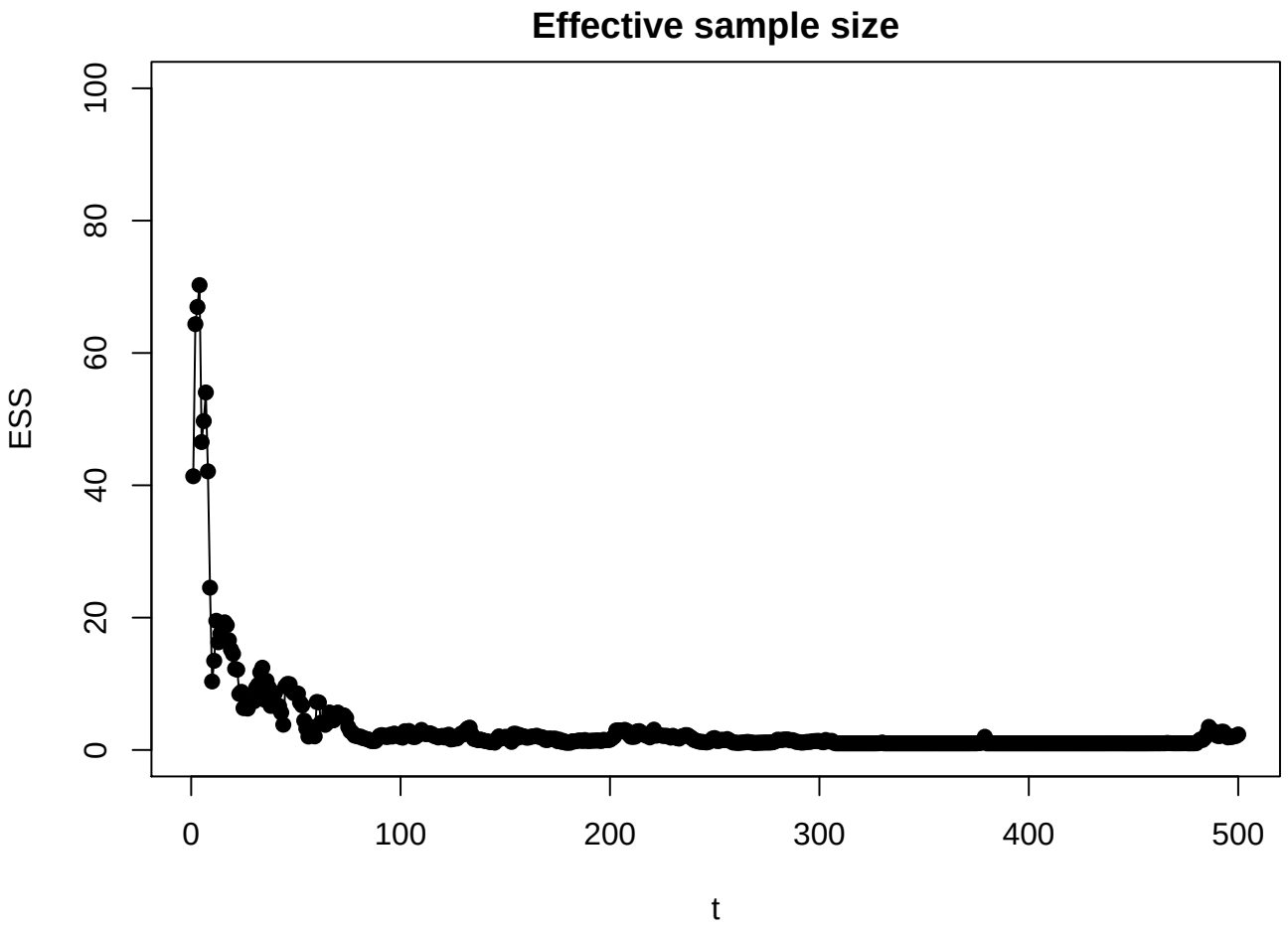
can be computed numerically, and the variance of the normal approximation can be computed by inspecting the curvature at the mode, given by

$$\left[-\frac{\partial^2 \log p(x|y_j, x_{j-1})}{\partial x^2}\right]^{-1} = \left[2y_j^2 e^{-2x} + 1/\sigma^2\right]^{-1}$$

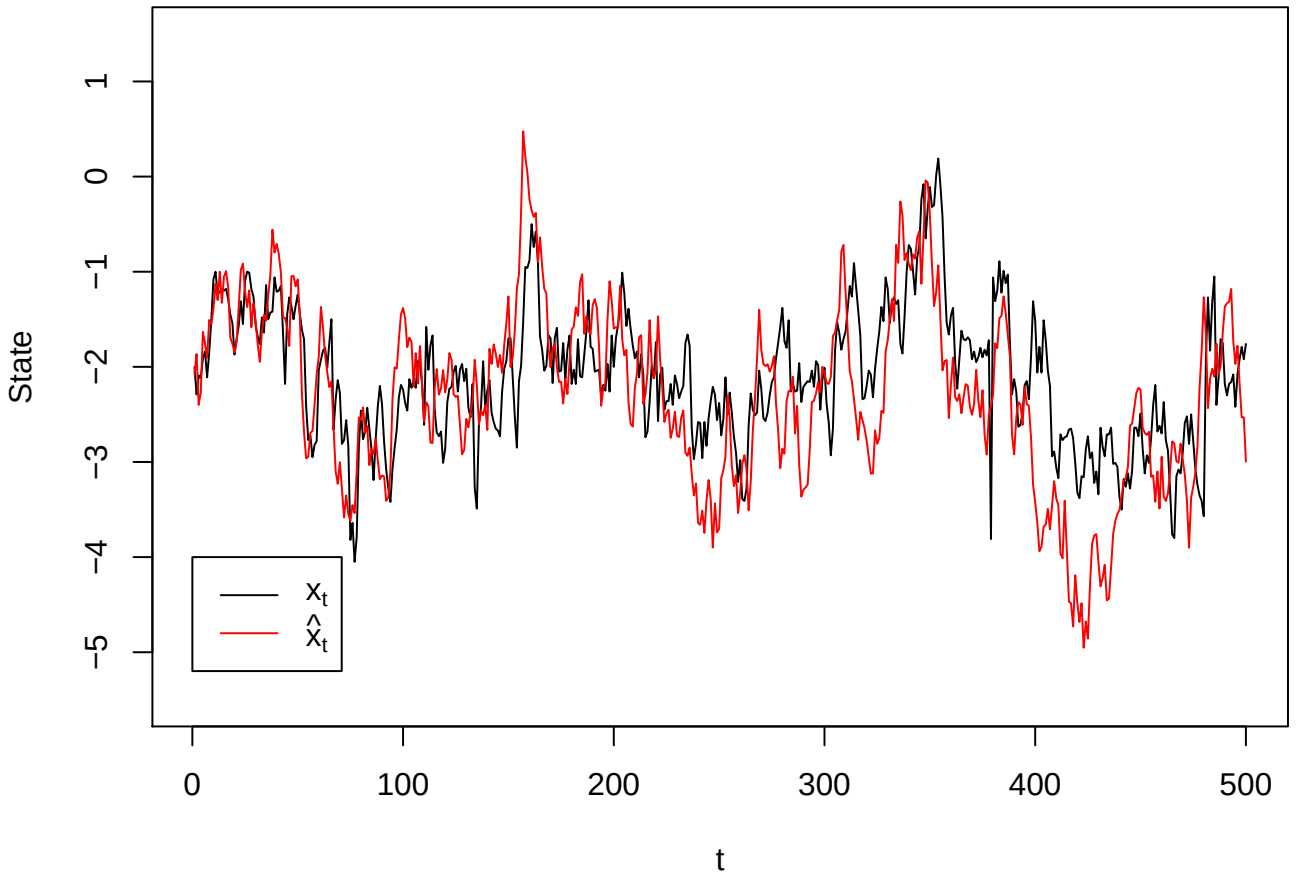
```
N<-100
Xmat<-wmat<-Wmat<-matrix(0,nrow=n,ncol=N)
pfunc<-function(x,yv,x0,muv,phiv,sigv){
        muj<-muv+phiv*(x0-muv)
        return(-x-0.5*yv^2*exp(-2*x)-0.5*(x-muj)^2/sigv^2)
}

f1<-optimize(pfunc,yv=yvec[1],x0=mu,muv=mu,phiv=phi,sigv=sig/sqrt(1-phi^2),
             interval=c(-10,10),maximum=TRUE)
mu0<-f1$maximum
sig0<-1/sqrt(2*yvec[1]^2*exp(-2*mu0)+1/sig^2)
Xmat[1,]<-rnorm(N,mu0,sig0)
wmat[1,]<-dnorm(yvec[1],0,exp(Xmat[1,]))*dnorm(Xmat[1,],
                mu,sig/sqrt(1-phi^2))/dnorm(Xmat[1,],mu0,sig0)
Wmat[1,]<-wmat[1,]/sum(wmat[1,])
ESS.2<-rep(0,n)
ESS.2[1]<-1/sum(Wmat[1,]^2)
for(j in 2:n){
        for(i in 1:N){
                f1<-optimize(pfunc,yv=yvec[j],x0=Xmat[j-1,i],
            muv=mu,phiv=phi,sigv=sig,interval=c(-10,10),maximum=TRUE)
                mu0<-f1$maximum
                sig0<-1/sqrt(2*yvec[j]^2*exp(-2*mu0)+1/sig^2)
                x<-rnorm(1,mu0,sig0)
                Xmat[j,i]<-x
                wmat[j,i]<-wmat[j-1,i]*dnorm(yvec[j],0,exp(x))*
                                dnorm(x,mu+phi*(Xmat[j-1,i]-mu),sig)/
                                dnorm(x,mu0,sig0)
                Wmat[j,]<-wmat[j,]/sum(wmat[j,])
        }
        ESS.2[j]<-1/sum(Wmat[j,]^2)
}
```

```
par(mar=c(4,4,2,1))
plot(c(1:n),ESS.2[1:n],type="l",xlab='t',ylab="ESS",ylim=range(0,N))
points(c(1:n),ESS.2[1:n],pch=19)
title('Effective sample size')
```




```
Xhat<-matrix(0,nrow=length(yv),ncol=n)
Yhat<-matrix(0,nrow=length(yv),ncol=n)
for(j in 1:n){
        if(j == 1){
                fmat<-outer(xv,Xmat[1,],dns0)
                Xhat[,j]<-apply(fmat,1,function(x){return(sum(x*Wmat[j,]))})
        }else{
                fmat<-outer(xv,Xmat[j-1,],dns)
                Xhat[,j]<-apply(fmat,1,function(x){return(sum(x*Wmat[j,]))})
        }
        fmat<-outer(yv,exp(Xmat[j,]),dnsv)
        Yhat[,j]<-apply(fmat,1,function(x){return(sum(x*Wmat[j,]))})
}
par(mar=c(4,4,2,1))
plot(xv[apply(Xhat,2,which.max)],xlab='t',type='l',ylab='State',ylim=range(-5.5,1.5))
lines(xvec,col='red')
legend(0,-4,c(expression(x[t]),expression(hat(x)[t])),col=c('black','red'),lty=1)
```



Resampling: Particle degeneracy can be overcome by *resampling* that removes the particles with low weights. The commonest form of resampling is to use a non-parametric bootstrap, with multinomial sampling. At step j, the Monte Carlo estimate of π_j is

$$\widehat{\pi}_j(\mathbf{x}) = \sum_{i=1}^{N} W_j(x_{1:j}^{(i)}) \delta_{x_{1:j}^{(i)}}(\mathbf{x})$$

This discrete distribution on

$$x_{1:j}^{(1)}, \ldots, x_{1:j}^{(N)}$$

is resampled N times according to the weights

$$W_j(x_{1:j}^{(1)}), \ldots, W_j(x_{1:j}^{(N)}).$$

After resampling, the new resampled values replace the

$$x_{1:j}^{(1)}, \ldots, x_{1:j}^{(N)}$$

collection, and each resampled point carries with it its normalized resampling weight $W_j(x_{1:j}^{(i)})$.

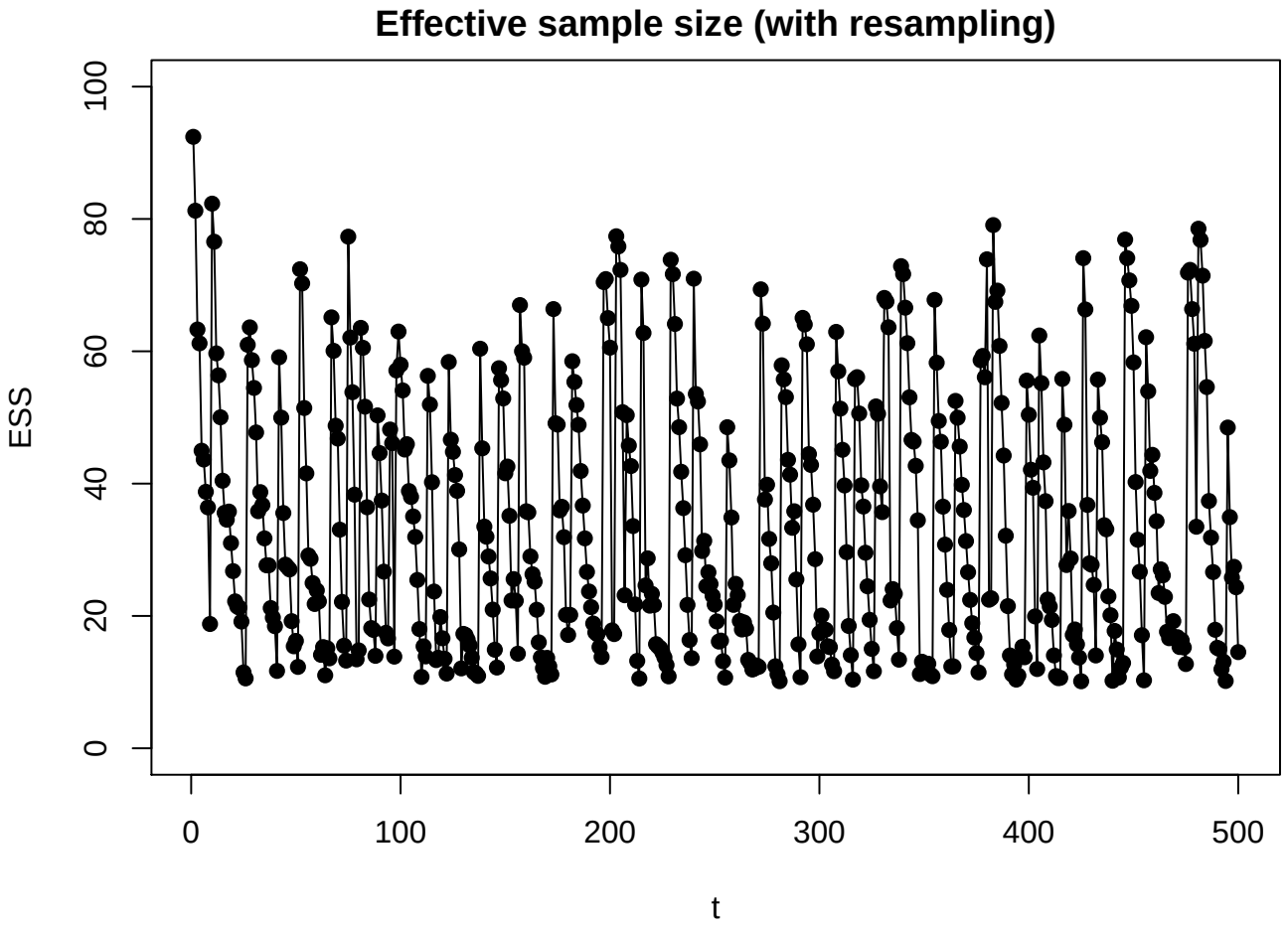
Each original value $x_{1:j}^{(i)}$ may appear in the resampled collection more than once. We perform resampling to preserve particle diversity, but should not resample too often, as that introduces excess variability.

A common strategy is to resample only when the observed variance of the weights exceeds some threshold.

```
N<-100
index<-c(1:N)
Xmat<-wmat<-Wmat<-Imat<-matrix(0,nrow=n,ncol=N)
```

```
Xmat[1,]<-rnorm(N,mu,sig/sqrt(1-phi^2))
wmat[1,]<-dnorm(yvec[1],0,exp(Xmat[1,]))
Wmat[1,]<-wmat[1,]/sum(wmat[1,])
Imat[1,]<-sample(index,prob=Wmat[1,],size=N,rep=T)
Xmat[1,]<-Xmat[1,Imat[1,]]
wmat[1,]<-Wmat[1,Imat[1,]]
Wmat[1,]<-wmat[1,]/sum(wmat[1,])
ESS.3<-rep(0,n)
ESS.3[1]<-1/sum(Wmat[1,]^2)
for(j in 2:n){
        Xmat[j,]<-rnorm(N)*sig+(mu+phi*(Xmat[j-1,]-mu))
        wmat[j,]<-wmat[j-1,]*dnorm(yvec[j],0,exp(Xmat[j,]))
        Wmat[j,]<-wmat[j,]/sum(wmat[j,])
        ESS.3[j]<-1/sum(Wmat[j,]^2)
        if(ESS.3[j]<10){  #Resample
                Imat[j,]<-sample(index,prob=Wmat[j,],size=N,rep=T)
                Xmat[j,]<-Xmat[j,Imat[j,]]
                wmat[j,]<-Wmat[j,Imat[j,]]
                Wmat[j,]<-wmat[j,]/sum(wmat[j,])
                ESS.3[j]<-1/sum(Wmat[j,]^2)
        }
}
```

```
par(mar=c(4,4,2,1))
plot(c(1:n),ESS.3[1:n],type="l",xlab='t',ylab="ESS",ylim=range(0,N))
points(c(1:n),ESS.3[1:n],pch=19)
title('Effective sample size (with resampling)')
```



```
Xhat<-matrix(0,nrow=length(yv),ncol=n)
Yhat<-matrix(0,nrow=length(yv),ncol=n)
for(j in 1:n){
        if(j == 1){
                fmat<-outer(xv,Xmat[1,],dns0)
                Xhat[,j]<-apply(fmat,1,function(x){return(sum(x*Wmat[j,]))})
        }else{
                fmat<-outer(xv,Xmat[j-1,],dns)
                Xhat[,j]<-apply(fmat,1,function(x){return(sum(x*Wmat[j,]))})
        }
        fmat<-outer(yv,exp(Xmat[j,]),dnsv)
        Yhat[,j]<-apply(fmat,1,function(x){return(sum(x*Wmat[j,]))})
}
par(mar=c(4,4,2,1))
plot(xv[apply(Xhat,2,which.max)],xlab='t',type='l',ylab='State',ylim=range(-5.5,1.5))
lines(xvec,col='red')
legend(0,-4,c(expression(x[t]),expression(hat(x)[t])),col=c('black','red'),lty=1)
```