DTU Compute

02425
December 4, 2024
UHT/uht

Exercise 10: Filtering in a non-linear equation

This exercise reproduces the running example in chapter 10, with a few minor deviations. A bacterial population evolves according to the Cox-Ingersoll-Ross model

$$dX_t = \lambda \cdot (\xi - X_t)\, dt + \gamma\sqrt{X_t}\, dB_t \qquad \text{with } \lambda = \xi = \gamma = 1.$$

Here X_t is a measure of the concentration of the bacteria (cells per volume) at time t. We do not specify the volume unit.

Question 1 **The forward Kolmogorov equation:** Write up the forward Kolmogorov equation for the transition probabilities, in advection-diffusion form, and discretize the generator. Use a uniform grid on $[0, 5]$. Compute the transition probabilities over a time step of 0.1 and plot them.

Solution: The forward Kolmogorov equation is

$$\dot{\phi} = -(u\phi - D\phi')'$$

where

$$D(x) = \frac{1}{2}g^2(x) = \frac{1}{2}\gamma^2 x$$

and

$$u(x) = f(x) - D'(x) = \lambda(\xi - x) - \frac{1}{2}\gamma^2 \quad .$$

We truncate the grid $x \in [0, 5]$ and discretize it, using a uniform grid and reflective boundary conditions:

```
require(SDEtools)
```

```
## Loading required package: SDEtools
```

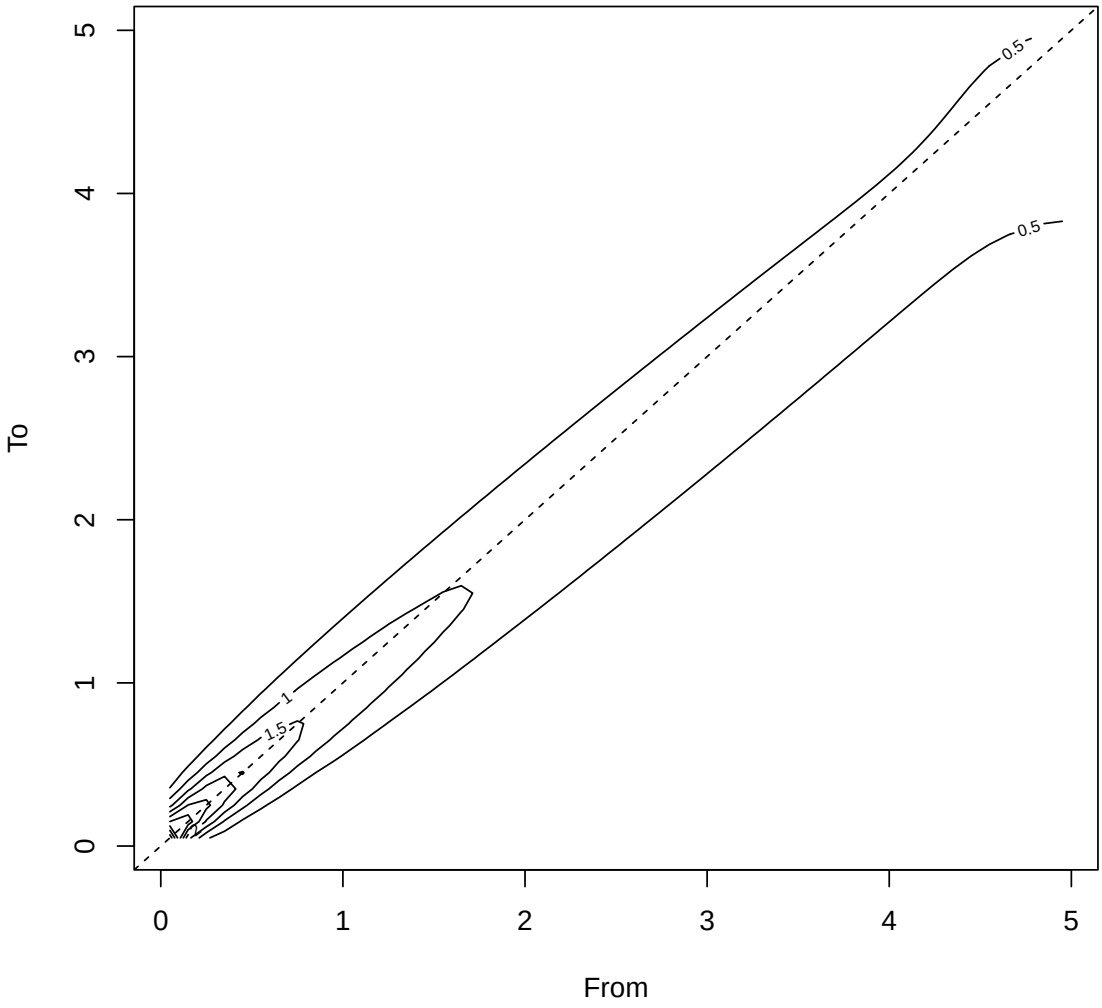
```
lambda <- 1
xi <- 1
gamma <- 1

# Define model.
f = function(x) lambda*(xi-x);
g = function(x) gamma*sqrt(x);

# Advection-diffusion form
D = function(x) 0.5*gamma^2*x;
u = function(x) f(x) - 0.5*gamma^2;

## Generator

## Define grid
xii = seq(0,5,0.1)                                  # Cell Interfaces
dx <- diff(xii)
xc = 0.5*(tail(xii,-1) + head(xii,-1))    # Cell centers
nc <- length(xc)

G = fvade(u,D,xii,'r');
```

```
## Loading required package: Matrix
```

```
tsample <- 0.1
P <- as.matrix(expm(G*tsample))
```

To visualize it, we plot the contour lines of the the transition densities. The following code takes into account the possibility that the grid is non-uniform, even if that is not needed here:

```
contour(xc,xc,P/rep(dx,rep(nc,nc)),xlab="From",ylab="To")
abline(0,1,lty="dashed")
```

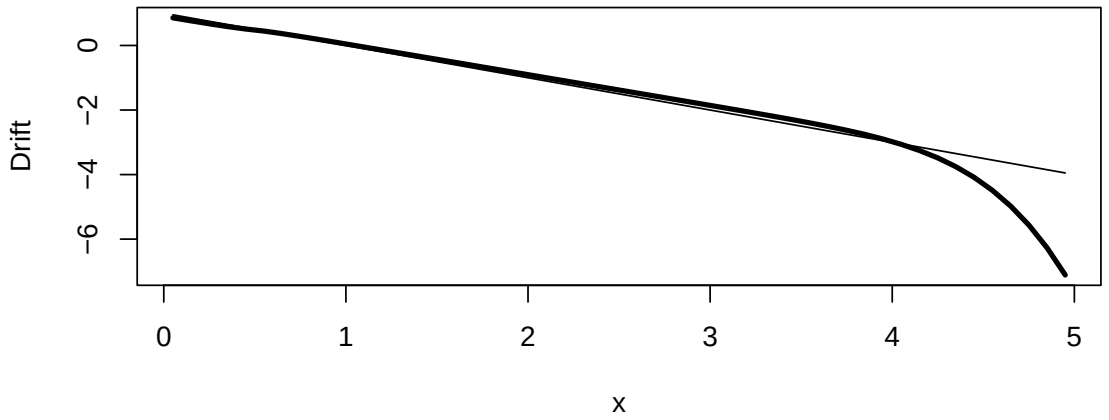


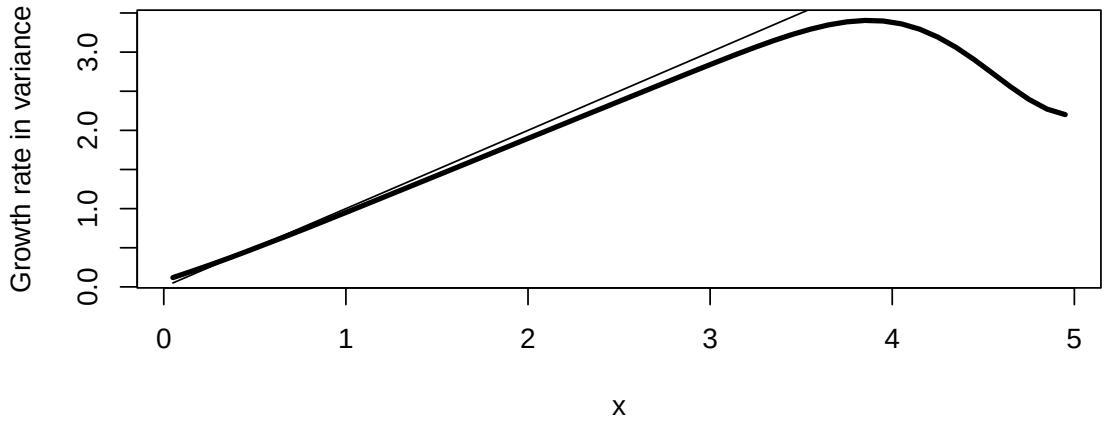
We see that the densities are roughly concentrated on the diagonal, and that the variance seems to grow with the initial state.

Note: It is very easy to confuse the axis. Each row in P corresponds to a "from" state while each column corresponds to a "to" state. The `contour` command views each row as an x-coordinate and each row as a y-coordinate. These things depend on the language and the commands. An advice is to also do the analysis with a much larger time step, where we know that the density should be almost independent of the "from" state, to ensure that we don't confuse the axis.

To verify the transition probabilities, we could compute the incremental mean and variance and compare it with the Euler approximation:

```
mu <- P %*% xc
dmudt <- (mu - xc) / tsample
V  <- P %*% (xc^2) - mu^2
dVdt <- V/tsample

par(mfrow=c(2,1))
plot(xc,dmudt,type="l",lwd=3,xlab="x",ylab="Drift")
lines(xc,f(xc))

plot(xc,dVdt,type="l",lwd=3,xlab="x",ylab="Growth rate in variance")
lines(xc,g(xc)^2)
```





We see that there is a reasonable agreement. The differences are due to three elements:

- The Euler discretization of time.

- The discretization of space.
- The imposed reflecting boundary conditions.

The effect of boundary conditions is very visible for large x; it is more difficult to tease apart the effect of discretizing time vs. space. Such an analysis would involve generator G, since that does not involve time discretization.

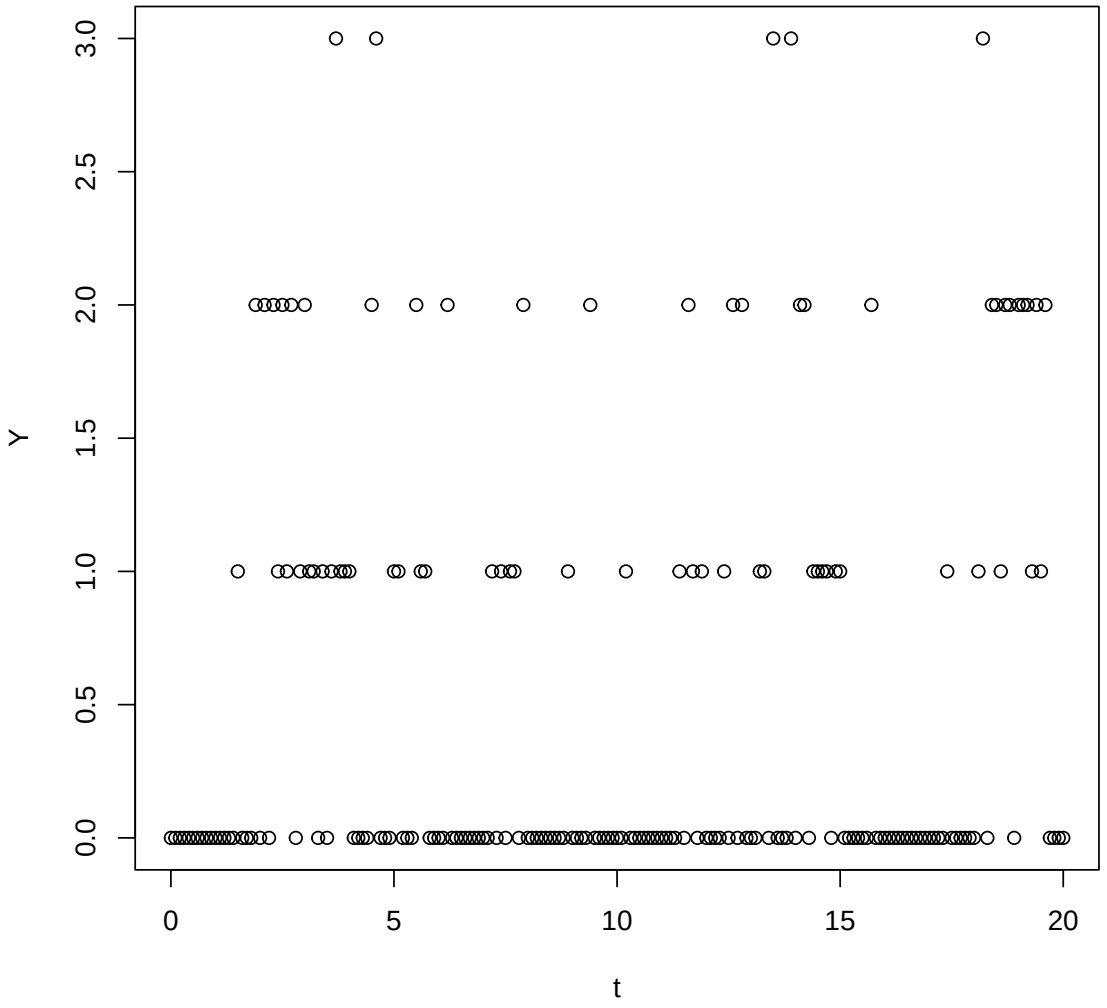
An observer measures the population abundance, i.e. the state X_t, in the following way: A small volume $v = 0.5$ is sampled and all bacteria in the volume are counted. The count data is Poisson distributed with (conditional) mean $\mathbf{E}\{Y_t|X_t\} = v \cdot X_t$, i.e.

$$Y_t|X_t \sim \text{Poisson}(\text{mean} = v \cdot X_t)$$

Question 2 **Measurements:** Import the measurements from the file `hmm-obs.txt` and plot them.

Solution: Ah, finally an easy question:

```
obs <- read.table("hmm-obs.txt",header=TRUE)

plot(Y ~t, data=obs)
```



Question 3 **The state likelihood:** Write up the "state likelihood function" of the state $X_t = x$, for a given measurement $y = Y_{t_i}(\omega)$, i.e. identify

$$\mathbb{P}\{Y_{t_i} = y | X_{t_i} = x\}$$

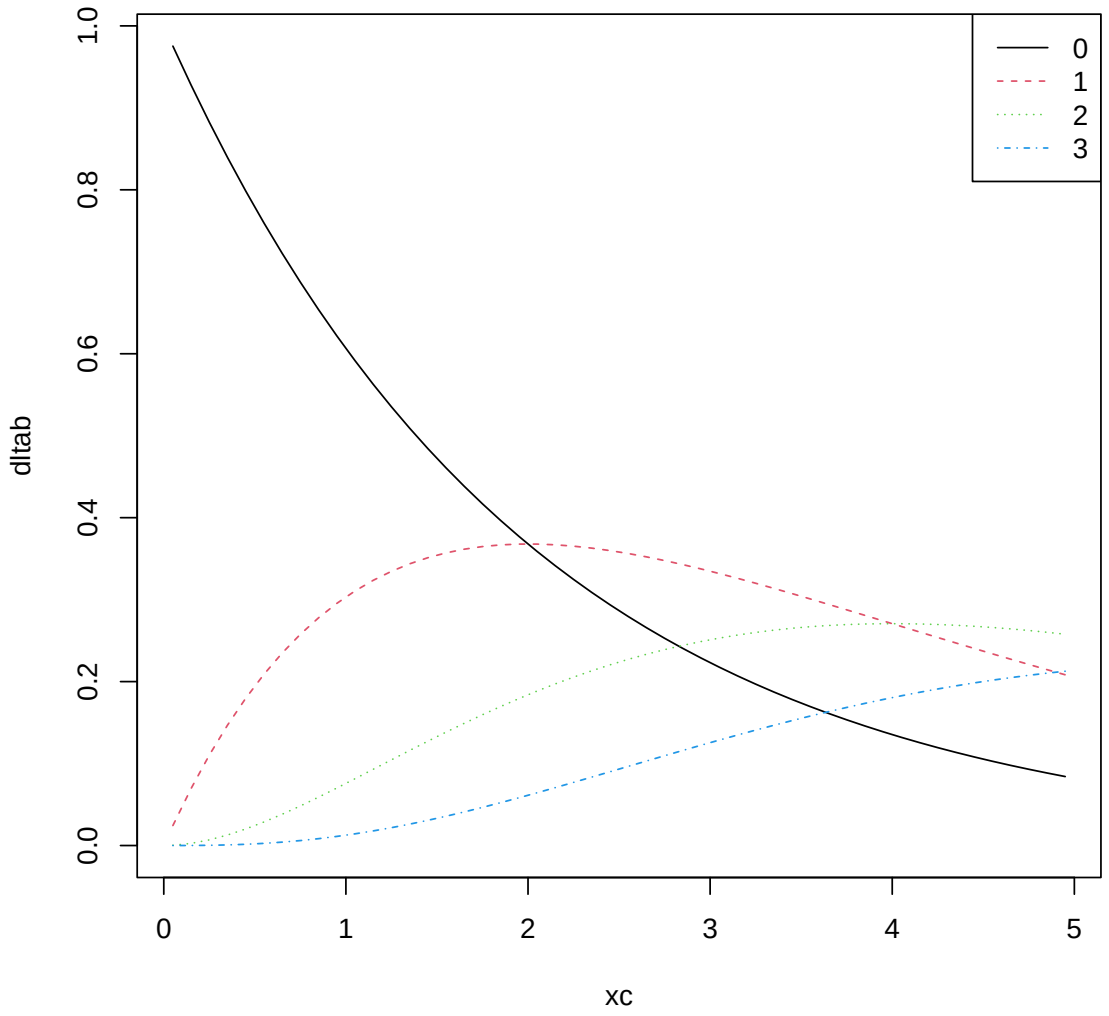
as a function of x and y. Plot the state likelihood function as a function of x, for $y = 0, \ldots, 5$.

Solution: Conditional on $X_{t_i} = x$, we have that Y_i is Poisson distributed with mean vx. Hence

$$\mathbb{P}\{Y_{t_i} = y | X_{t_i} = x\} = e^{-xv} \frac{(vx)^y}{y!}$$

These probabilties exist in the `R`-function `dpois`. So:

```
vsample <- 0.5
dl <- function(x,y) dpois(y,lambda=vsample*x)
maxy <- max(obs$Y)
ys <- 0:maxy
dltab = outer(xc,ys,dl);

matplot(xc,dltab,type="l")
legend("topright",lty=1+ys,legend=ys,col=1+ys)
```

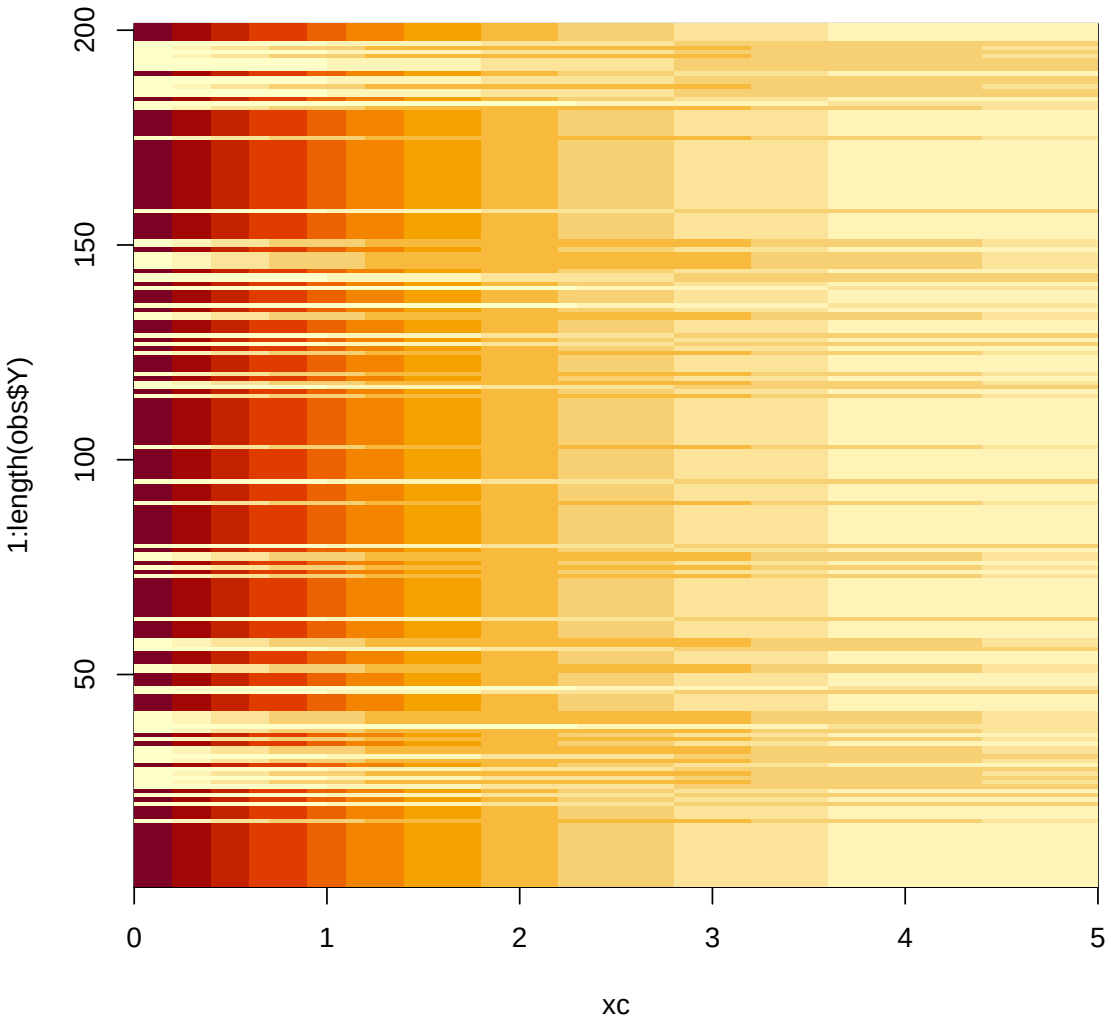


We see that, e.g., when $y = 0$, the likelihood function of x is an exponentially decaying function with rate v. When $y = 1$, the likelihood-maximizing x is $1/v = 2$.

Question 4: Then construct a table which, for each sampling time $t = t_i$ and each possible value of the state $x = X_t$, holds the state likelihood function $l_i(x)$. Plot this table as a pseudocolor image.

Solution:

```
ltab <- dltab[,obs$Y+1]
image(xc,1:length(obs$Y),ltab)
```



We see that the image is quite unclear. This is because the single measurement holds very little information about the current state.

Question 5 **Implementing the filter:** Implement the HMM filter to estimate the state, based on iterating time update and data update.

Solution: The following code implements the filter. It takes the generator as an argument, because later in the exercise we will do this for different modifications of the generator.

```
## Filter
hmmfilter <- function(G)
{
    phi <- Matrix(array(0,c(length(obs$t),length(xc))))
    psi <- phi

    ## Initialize with the stationary distribution
    mu <- StationaryDistribution(G)
    mu <- mu/sum(mu)

    ## Compute transition probabilities
    P <- expm(G*tsample)
    const <- numeric(length(obs$t))

    phi[1,] <- mu

    ## Include the first data update
    psi[1,] <- phi[1,] * ltab[,1]
    const[1] <- sum(psi[,1] )
    psi[,1] <- psi[,1] / const[1]

    ## Main time loop over the remaining time steps
    for(i in 2:length(obs$t))
    {
        phi[i,] = psi[i-1,] %*% P    # Time update
        psi[i,] = phi[i,] * ltab[,i] # Data update
        const[i] <- sum(psi[i,])     # Normalization
        psi[i,] = psi[i,] / const[i]
    }

    return(list(c=const,phi=as,matrix(phi),psi=as.matrix(psi),loglik=sum(log(const))))
}

## Run the filter with default values for parameters
est <- hmmfilter(G)
```

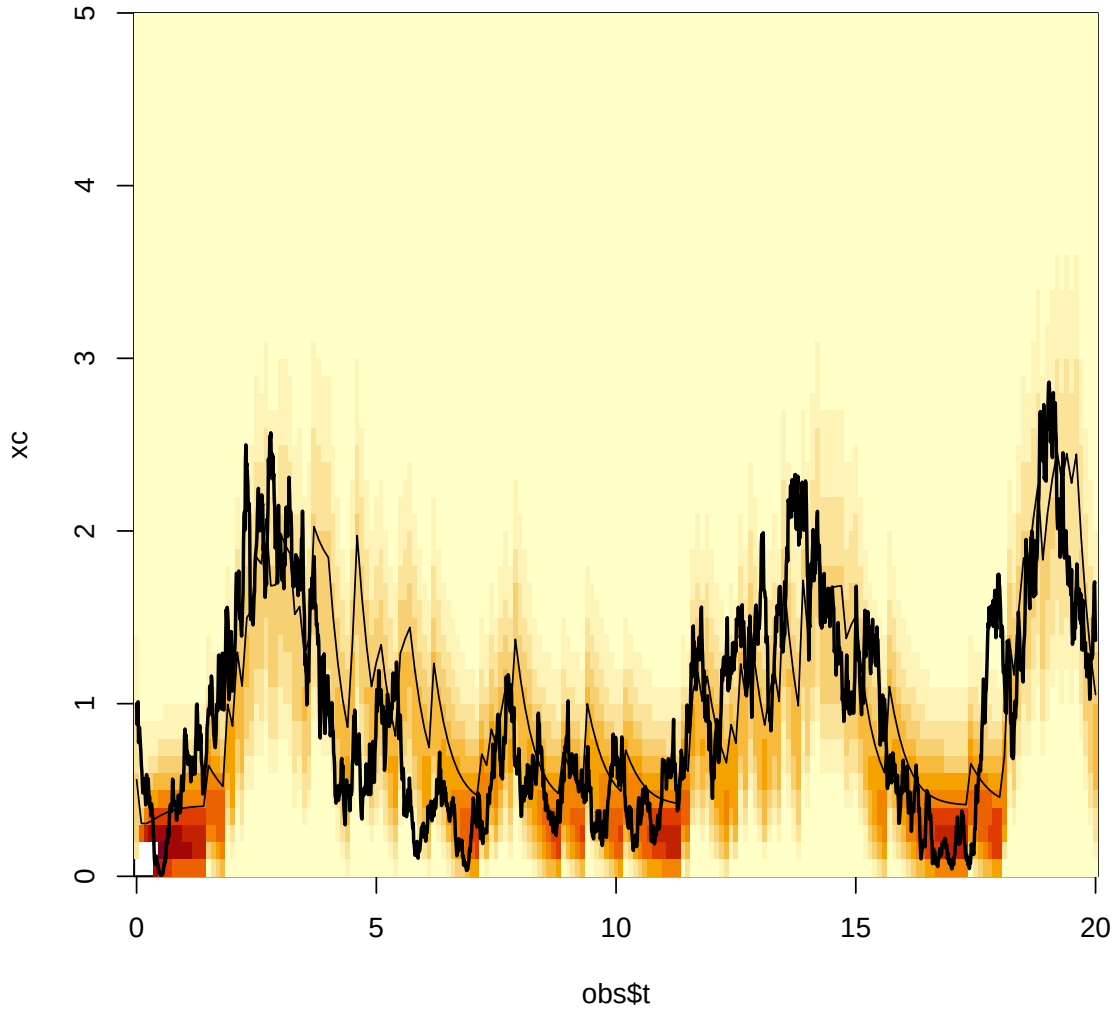
Note that this has also been implemented in the `HMMfilterSDE` function in the `HMMfilterSDE` toolbox (with a few more bells and whistles).

Question 6: Make a color plot of the estimated state distribution. Add to the plot time series of the mean (alternatively, median or mode) in the estimated distribution.

Download the file `hmm-states.txt`, which holds the true states, and add this to the plot

Solution: We fiddle around a bit with the color code to get the clearest image:

```
## Show posterior probabilities, and posterior mean
image(obs$t,xc,est$psi,zlim=c(0,0.2))
lines(obs$t,apply(est$psi,1,function(p)sum(p*xc)))

## Add true state
sim <- read.table("hmm-states.txt",header=TRUE)
lines(sim$t,sim$X,lwd=2)
```



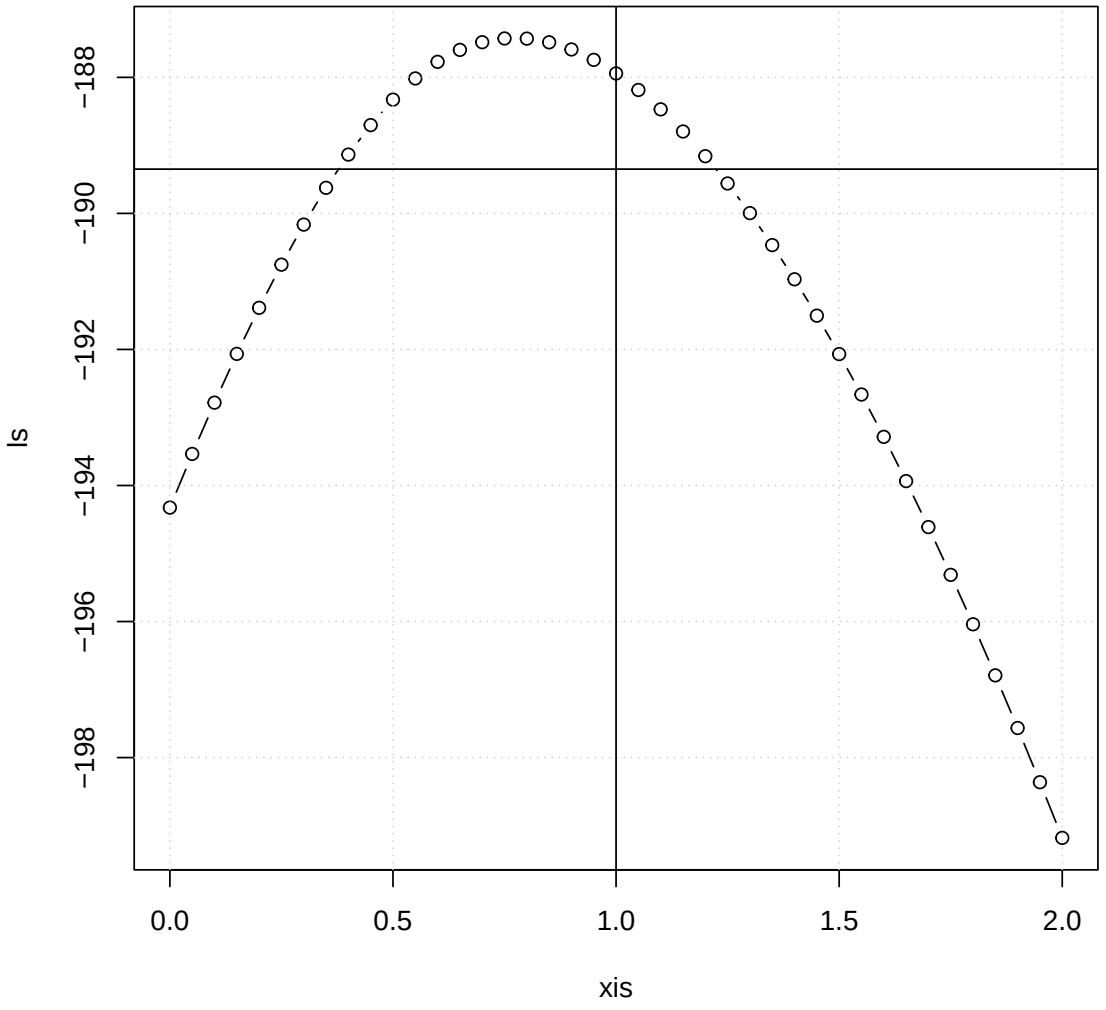
There are several things to notice:

- First, the estimated mean is largely consistent with the "true" states. The implementation works!
- The estimated mean increases in jumps and then relaxes exponentially. This come from individual counts surrounded by zeros.
- The estimate lags a bit behind the "true" state. This is because the filter uses *past* measurements.

Question 7 **Maximum likelihood estimation:** We now consider the parameter ξ unknown. Run the filter in the previous with values of ξ ranging from 0 to 2, in steps of 0.1. For each value of ξ, compute the likelihood. Plot the likelihood function of ξ and commment - the data was simulated with a value of $\xi = 1$.

Solution: Here, we use a bit finer grid than was asked for:

```
## Likelihood estimation of xi
xis <- seq(0,2,0.05)   # Try these values of xi

loglik <- function(xi)    # Likelihood function of one value of xi
{
    f <- function(x) lambda*(xi-x);
    D <- function(x) 0.5*gamma^2*x;
    u <- function(x) f(x) - 0.5*gamma^2;


    G = fvade(u,D,xii,'r');

    return(hmmfilter(G)$loglik)
}

## Tabulate likelihood function
ls <- sapply(xis,loglik)

plot(xis,ls,type="b")
grid()
abline(v=xi) ## Include the true value used in the simulation
abline(h=max(ls)-0.5*qchisq(0.95,df=1)) ## Include confidence interval
```



We see that the likelihood function is maximized around $\xi = 0.8$. The confidence area is fairly wide, reflecting that it is difficult to estimate ξ based on this data set. Alternatively, we can be relieved to learn that the filter operates well in predicting next measurements, even if we use a "wrong" value for ξ.