Inverse Problem of Diffusion

Martin Outzen Berild

September 12, 2019

We consider an inverse problem with a basis in the following differential equation

$$\frac{du(x,t)}{dt} = \frac{d^2u(x,t)}{dx^2}, \;\; u(x,0) = h_0(x), \;\; x \in (0,1), t \geq 0. \tag{1}$$

Data is $u(x,t) = h_t(x)$ for a given time $t > 0$. The aim of the inverse problem is $h_0(x)$.

The forward model can be written as

$$u(x,t) = h_t(x) = \frac{1}{\sqrt{4\pi t}} \int e^{-(x-y)^2/(4t)} h_0(y)dy, \;\; t \geq 0. \tag{2}$$

Using discretization we get

$$\mathbf{h_t} = \begin{bmatrix} h_t(x_1) \\ h_t x_2 \\ \vdots \\ h_t x_N \end{bmatrix} = \mathbf{A} \begin{bmatrix} h_0 x_1 \\ h_0 x_2 \\ \vdots \\ h_0 x_N \end{bmatrix} = \mathbf{A h_0}, \tag{3}$$

where a regular grid of N = 100 points is used, such that $x_1 = 0$, $x_2 = 0.01$, $\ldots$, $x_N = 0.99$. The sequence x is created in **R** by the the code

```
x = seq(from = 0, to = 0.99, by = 0.01)
```

The interval (0,1) is made into a circle, i.e. 1 corresponds to 0. The matrix A has elements

$$A(i,j) = \frac{0.01}{\sqrt{4\pi t}} e^{-|x_i - x_j|^2/(4t)}. \tag{4}$$

The distance $|x_i - x_j|$ is modular on the circle (0,1). The *createA* function in below calculates the matrix A for a given position x and time t

```
createA <- function(x,t){
  A = diag(0.01/sqrt(4*pi*t), nrow = length(x), ncol = length(x))
  for (i in seq(1,length(x)-1)){
    for (j in seq(i+1,length(x))){
      A[i,j] = 0.01/sqrt(4*pi*t)*exp(-(x[i]-x[j])^2/(4*t))
      A[j,i] = A[i,j]
    }
  }
  A
}
```

Measurements $\mathbf{y} = (\mathbf{y_1}, ..., \mathbf{y_N})'$ are acquired at time t = 0.001 (1ms):

$$y_i = h_t(x_i) + \epsilon_i, \;\; \epsilon_i \sim \mathcal{N}(0, 0.025^2), \;\; \text{iid}. \tag{5}$$

The observations y are downloaded, imported into **R** and converted to vector form.

```
y = read.delim2(file = "OppgA.txt", header = F, sep = "\n", dec = ".")[[1]]
```

The observations are presented in Figure 1

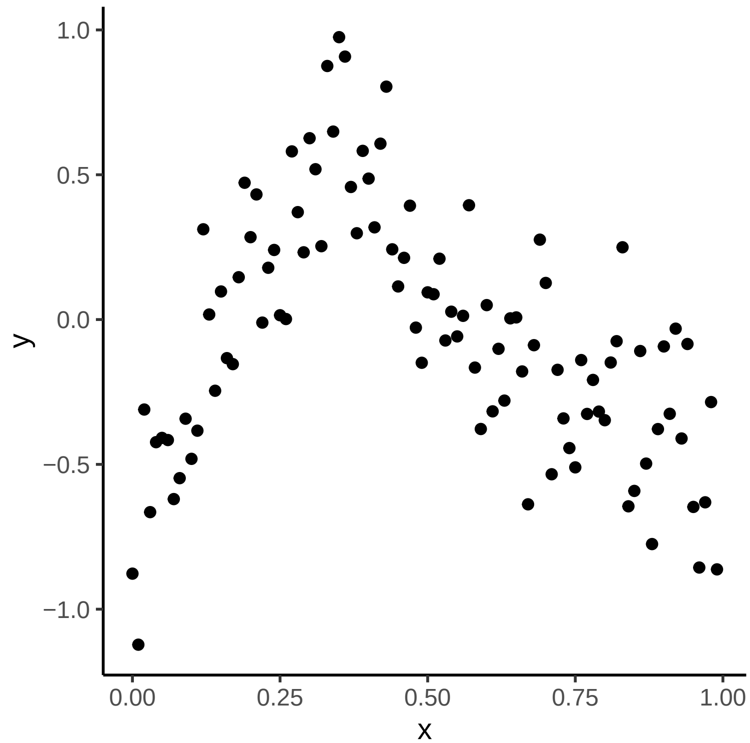


Figure 1: Observations $(y_1, ...y_{100})'$ that are informative of the latent process $h_t(x)$ at time $t = 1\text{ms}$.

Exercise a

We want to solve the inverse problem directly by $A^{-1}\mathbf{y}$. First we compute the eigenvalues of the matrix. The observations y are collected at time $t = 1\text{ms}$, and we firstly initialize the matrix A.

```
A = createA(x,t = 0.001)
```

The eigenvalues of A can easily be calculated in **R** and are shown in Figure 2.

```
S = eigen(A)[[1]]
```

The singular value decomposition can be found by finding the eigenvectors of A^TA and AA^T. Then since our matrix A is square we can use its eigenvalues in the formula

$$A = USV^T, \tag{6}$$

where U contains the eigenvectors of AA^T, V the eigenvectors of A^TA and S the eigenvalues of A.

```
U = eigen(A%*%t(A))[[2]]
V = eigen(t(A)%*%A)[[2]]
```

We want to approximate this solution using a filter. The approximation is given by

$$\mathbf{\hat{h}_0} = \sum_{\{\mathbf{i}:\sigma_\mathbf{i}>\mathbf{0}\}} \phi_\mathbf{i}(\alpha)\frac{<\mathbf{u_i}, \mathbf{y}>}{\sigma_\mathbf{i}}\mathbf{v_i}, \tag{7}$$

where $\phi_i(\alpha)$ is the filter applied. In our case we want to truncate the small eigenvalue of A, and this is done by the truncated singular value expansion which uses the filter $\phi_i(\alpha) = I\{\sigma_i > \alpha\}$. The choice of α which yields the best solution is not known however.

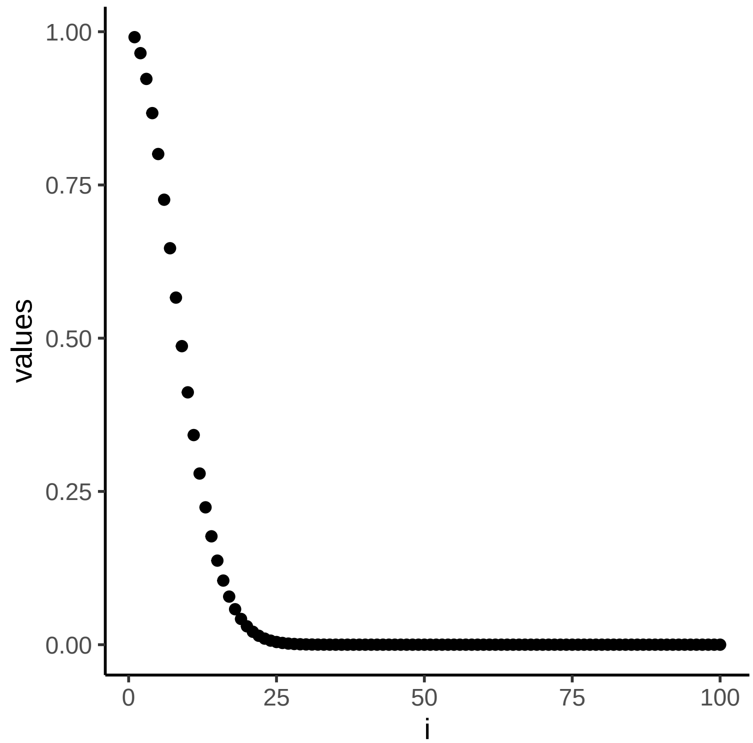
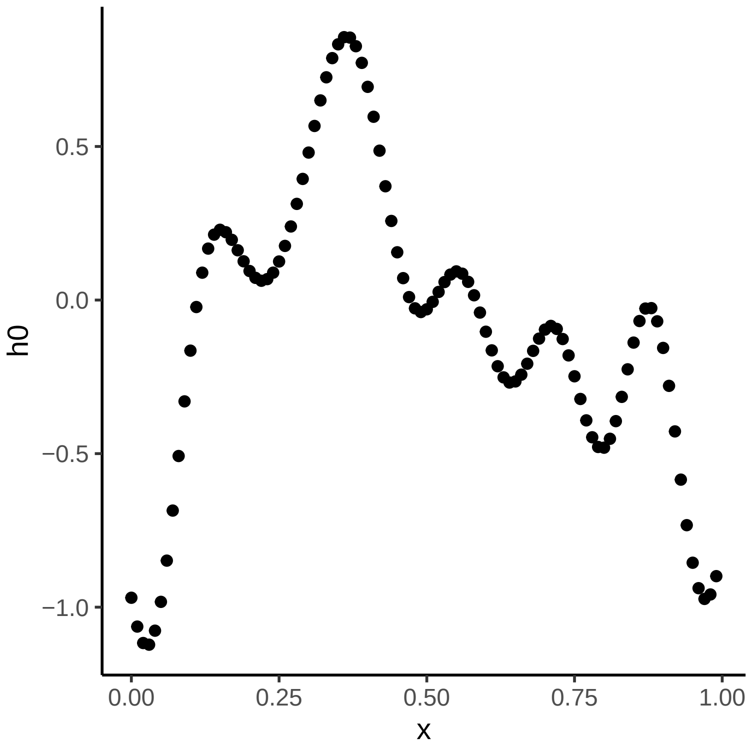


Figure 2: Eigenvalues of A at time $t = 1\text{ms}$

```
trunc.svd <- function(alpha,y,U,S,V){
  res = numeric(length(S))
  for (i in seq(length(S))){
    if(S[i]>alpha){
      res = res + (((U[,i]%*%y)[[1]])/S[i])*V[,i]
    }
  }
  res
}

h0 <- trunc.svd(alpha = 0.2, y,U,S,V)
```




```
# {r bestalpha,echo = F, fig.width=4,fig.height=4,
the optimal value of \alpha by iteration. Optimal solution is $0.03$, truncating eigenvalues
of A that are less."} # find_alpha<-function(y,U,S,V,A){ #   alpha = seq(0.00000000001,1,0.00001)
#   error = c() #   for (i in alpha){ #     h0 = trunc.svd(i,y,U,S,V) #     error =
c(error, mean((y - A%*%h0)^2)) #   } #   data.frame(alpha = alpha, error = error) # } #
res = find_alpha(y,U,S,V,A) # best_alpha = res$alpha[which.min(res$error)] # ggplot(res,
aes(x = alpha, y = error)) + #   geom_point() + #   geom_vline(xintercept = best_alpha,
color = "red") #
```

Exercise b

Assume that we now add prior information to $\mathbf{h_0}$ in the form of a Gaussian prior, $\mathbf{h_0} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. We want to find the posterior expectation $E(\mathbf{h_0}|\mathbf{y})$. We also have knowledge that the error $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{0.25^2 I})$. We have the linear relationship $\mathbf{y} = \mathbf{A h_0} + \epsilon$ and we have the posterior distribution given by

$$p(\mathbf{h_0}|\mathbf{y}) = \frac{\mathbf{p(y, h_0)}}{\mathbf{p(y)}} = \frac{\mathbf{p(y|h_0)p(h_0)}}{\mathbf{p(y)}}.$$

It is hard to anytically find the posterior mean, but if we represent the gaussian random function H_0 and ϵ by the Karhunen-Loève expansion given by

$$\begin{aligned} H_0 &= \sum_{i=1}^{\infty} H_{0,i}\mathbf{v_i} \\ \epsilon &= \sum_{i=1}^{\infty} \epsilon_i \mathbf{v_i}, \end{aligned}$$

this will make the estimation easier. This makes $\{H_0\}_{i=1}^{\infty}$ be independent Gaussian random variable with mean μ_i and variance γ_i^2, and $\{\epsilon_i\}_{i=1}^{\infty}$ becomes gaussian random variable with variance λ_i^2. The matrix A still is represented by the singular system $\{\sigma_i^2, v_i, u_i\}_{i=1}^{\infty}$, with σ_i, u_i and v_i being from S, U and V respectivly. The posterior random function can also be represented by this Karhunen-Loève expansion given by

$$(H_0|Y = y) = \sum_{i=1}^{\infty} (H_{0,i}|Y_i = y_i)\mathbf{v_i},$$

with the set $\{(H_{0,i}|Y_i = y_i)\}_{i=1}^{\infty}$ being independent Gaussian random variables with expected value $y_i\sigma_i/(\sigma_i^2 + \lambda_i^2/\gamma_i^2)$ and variance $\gamma_i^2[1 - \sigma_i^2/(\sigma_i^2 + \lambda_i^2/\gamma_i^2)]$. This yields the joint posterior expectation

$$E(H_0|Y = y) = \sum_{i=1}^{100} \frac{y_i\sigma_i}{\sigma_i^2 + \frac{\lambda_i^2}{\gamma_i^2}}.$$

```
post.expect <- function(lambda,gamma,y,S){
  res = numeric(length(y))
  for (i in seq(1,length(y))){
    res[i] = (y[i]*S[i])/(S[i]^2 + lambda^2/gamma^2)
  }
  res
}
# lambda = epsilon, gamma = h_0
e_post <- post.expect(lambda = 0.25, gamma = 1, y, S)
```

Since we have independent random variables they are uncorrelated and therby yielding the join posterior variance

$$\mathrm{Var}(\sum_{i=1}^{100}(H_{0,i}|Y_i = y_i)) = \sum_{i=1}^{100} \mathrm{Var}(H_{0,i}|Y_i = y_i) = \sum_{i=1}^{100} \gamma_i^2 \left[1 - \frac{\sigma_i^2}{\sigma_i^2 + \frac{\lambda_i^2}{\gamma_i^2}}\right]$$

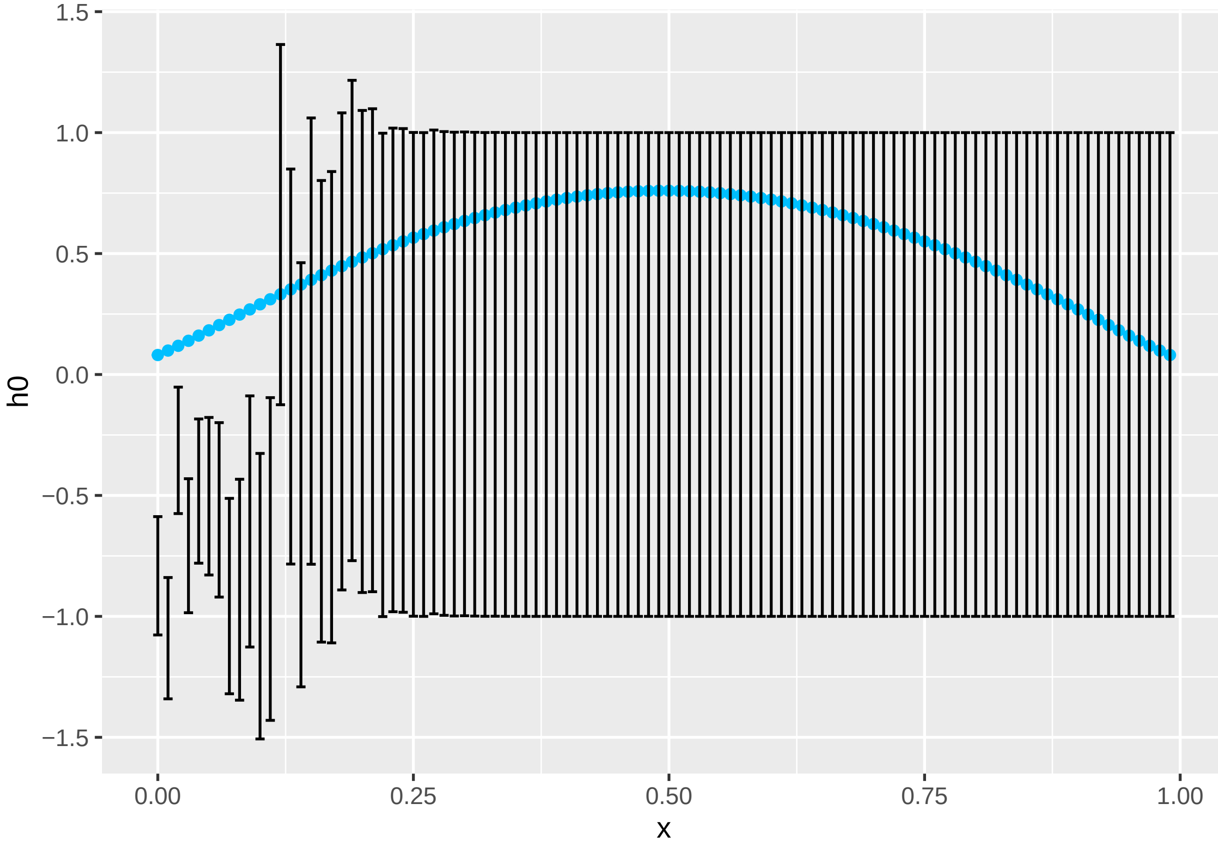
.

```
post.var <- function(lambda,gamma,y,S){
  res = numeric(length(y))
  for (i in seq(1,length(y))){
    res[i] = gamma^2*(1-S[i]^2/(S[i]^2 + lambda^2/gamma^2))
  }
  res
}
var_post <- post.var(lambda = 0.25, gamma = 1, y, S)
```

The optimal estimator is the represented by the following equation

$$\hat{\mathbf{h}}_0 = \sum_{i=1}^{100} \frac{\sigma_i^2}{\sigma_i^2 + \frac{\lambda_i^2}{\gamma_i^2}} \frac{< \mathbf{u_i}, \mathbf{y} >}{\sigma_i} \mathbf{v_i}$$

```
bay_sol <- function(lambda,gamma,y,U,S,V){
  res = numeric(length(y))
  for (i in seq(length(y))){
    res = res + (S[i]^2/(S[i]^2 + lambda^2/gamma^2)*((U[,1]%*%y)[[1]])/S[1])*V[,1]
  }
  res
}
h0 <- bay_sol(lambda = 0.25, gamma = 1, y, U, S, V)
```



Exercise c