Inverse Problem of Diffusion

Martin Outzen Berild

September 16, 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

$$\mathbf{h}_t = \begin{bmatrix} h_t(x_1) \\ h_t x_2 \\ \vdots \\ h_t x_N \end{bmatrix} = A \begin{bmatrix} h_0 x_1 \\ h_0 x_2 \\ \vdots \\ h_0 x_N \end{bmatrix} = A\mathbf{h}_0, \tag{3}$$

where a regular grid of N = 100 points is used, such that $x_1 = 0$, $x_2 = 0.01$, ..., $x_N = 0.99$. 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 = matrix(NA, nrow = length(x), ncol = length(x))
  for (i in seq(length(x))){
    for (j in seq(length(x))){
      d = min(abs(x[i]-x[j]),1-abs(x[i]-x[j]))
      A[i,j] = 0.01/sqrt(4*pi*t)*exp(-d^2/(4*t))
    }
  }
  A
}
```

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

$$y_i = h_t(x_i) + \epsilon_i, \;\; \epsilon_i \sim \mathcal{N}(0, 0.25^2), \;\; \text{iid.} \tag{5}$$

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

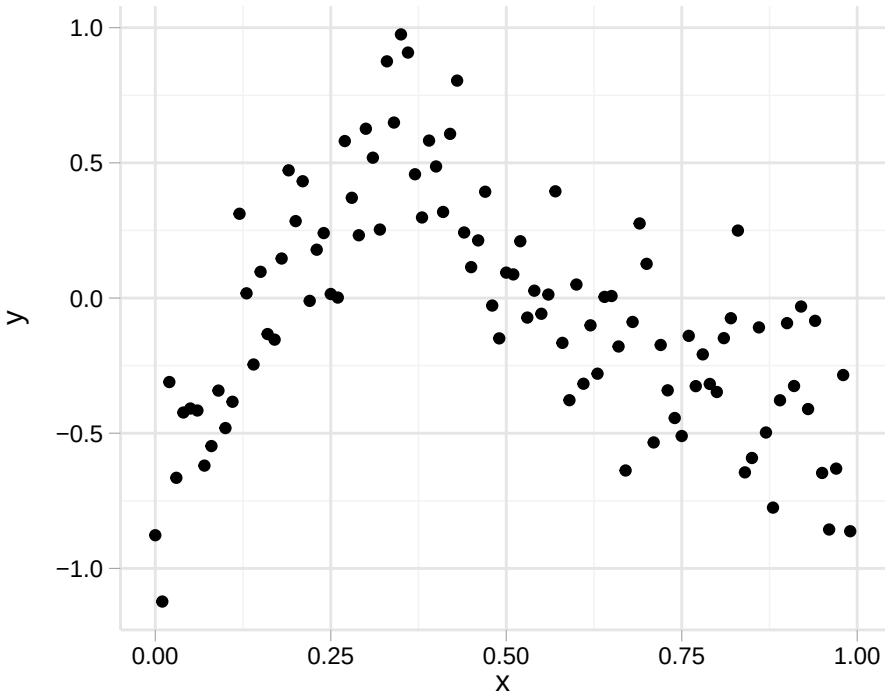

Figure 1: Observations $(y_1, ...y_{100})'$ that are informative of the latent process $h_t(x)$ at time $t = 1$ms.

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 $t = 1$ms, and we firstly initialize the matrix A.

The eigenvalues of A can easily be calculated in **R** and are shown in Figure 3.

```
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

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

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.

```
tsvd <- function(k,y,U,S,V){
  res = numeric(length(S))
  for (i in seq(k)){
      res = res + (((U[,i]%*%y)[[1]])/S[i])*V[,i]
```

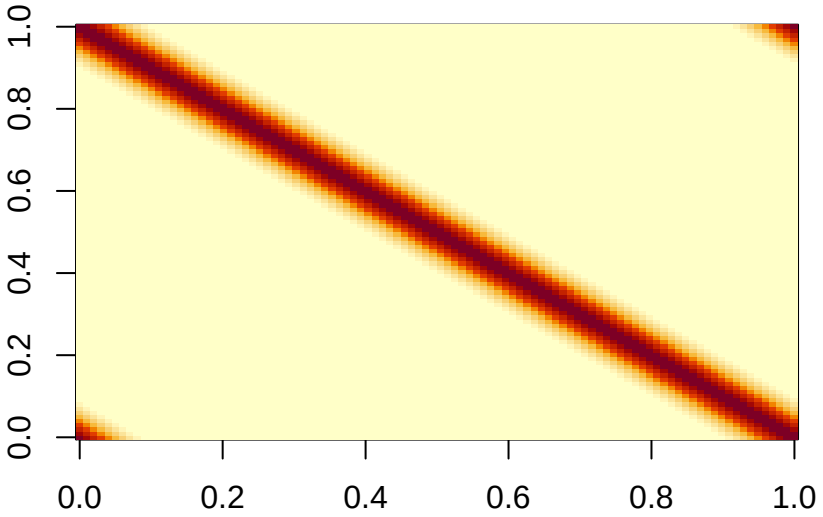

Figure 2: A visual representation of the values in A.

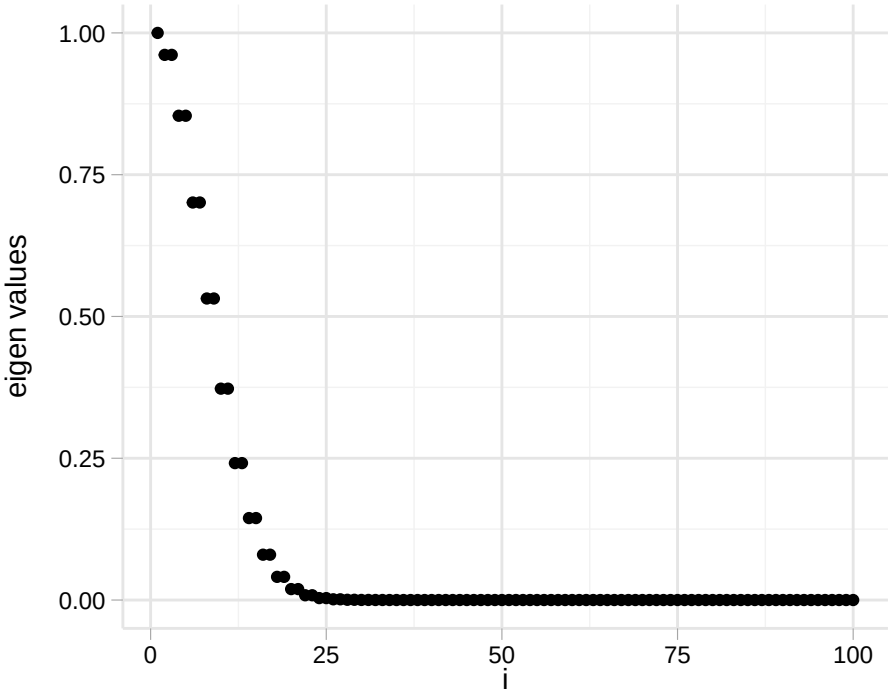

Figure 3: The eigenvalues of matrix A.

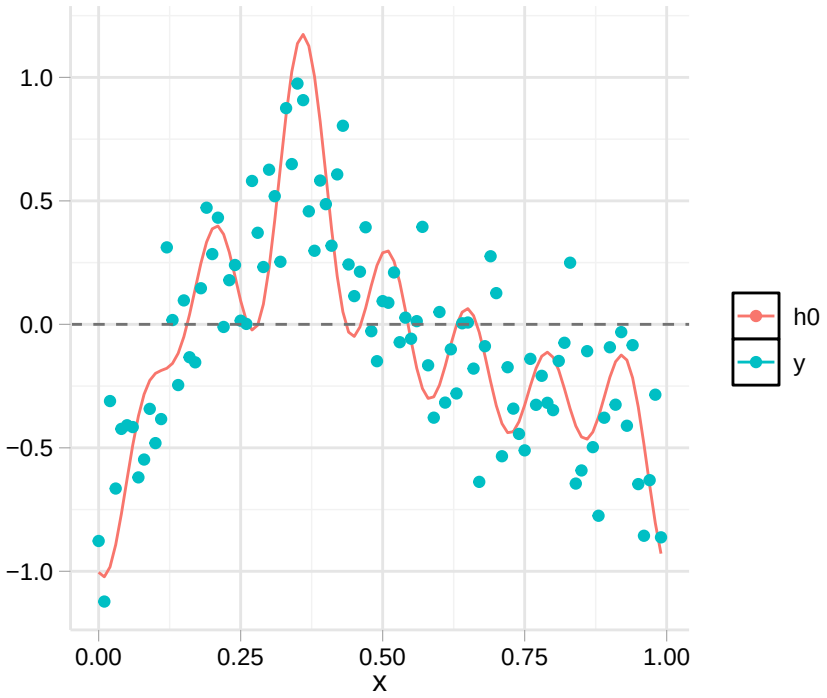

Figure 4: The solution h_0 using $k = 15$ largest eigenvalues of A.

```
  }
  res
}

h0 <- tsvd(15,y,U,S,V)
```

Exercise b

We now add prior information to $\mathbf{h}_0$ in the form of a Gaussion prior $\mathbf{h}_0 \sim \mathcal{N}(\mathbf{0}, Q_{\mathbf{h}_0})$, where $Q_{\mathbf{h}_0} = I$ is the precision matrix. We know from earlier that $\mathbf{y}|\mathbf{h}_0 \sim \mathcal{N}(A\mathbf{h}_0, Q_\epsilon)$, where Q_ϵ is the precision matrix, which yields the posterior distribution

$$\begin{aligned} p(\mathbf{h}_0|\mathbf{y}) &= \tfrac{p(\mathbf{h}_0,\mathbf{y})}{p(\mathbf{y})} \\ &\propto p(\mathbf{y}|\mathbf{h}_0)p(\mathbf{h}_0) \\ &\propto \exp\{-\tfrac{1}{2}((\mathbf{y}-A\mathbf{h})^T Q_\epsilon(\mathbf{y}-A\mathbf{h}_0) + \mathbf{h}_0^T Q_{\mathbf{h}_0}\mathbf{h}_0)\}. \end{aligned}$$

We want to find the Maximum Aposterior Prediction(MAP) of the $\mathbf{h}_0$, which is found by

$$\hat{\mathbf{h}}_0 = \underset{\mathbf{h}_0}{\operatorname{argmin}}\{\mathbf{y}^T Q_\epsilon \mathbf{y} - \mathbf{y}^T Q_\epsilon(A\mathbf{h}_0) - (A\mathbf{h}_0)^T Q_\epsilon \mathbf{y} + (A\mathbf{h}_0)^T Q_\epsilon(A\mathbf{h}_0) + \mathbf{h}_0^T Q_{\mathbf{h}_0}\mathbf{h}_0\}$$

$$0 = -2\mathbf{y}^T Q_\epsilon(A\mathbf{1}) + 2(A\mathbf{1})^T Q_\epsilon(A\mathbf{h}_0) + 2\cdot \mathbf{1}^T Q_{\mathbf{h}_0}\mathbf{h}_0$$

$$\hat{\mathbf{h}}_0 = \left[A^T Q_\epsilon A + Q_{\mathbf{h}_0}\right]^{-1}\mathbf{y}^T Q_\epsilon A, \qquad Q_{\hat{\mathbf{h}}_0} = A^T Q_\epsilon A + Q_{\mathbf{h}_0}.$$

The function *post.h0()* calculates the prediction $\hat{\mathbf{h}}_0$ and the standard deviation.

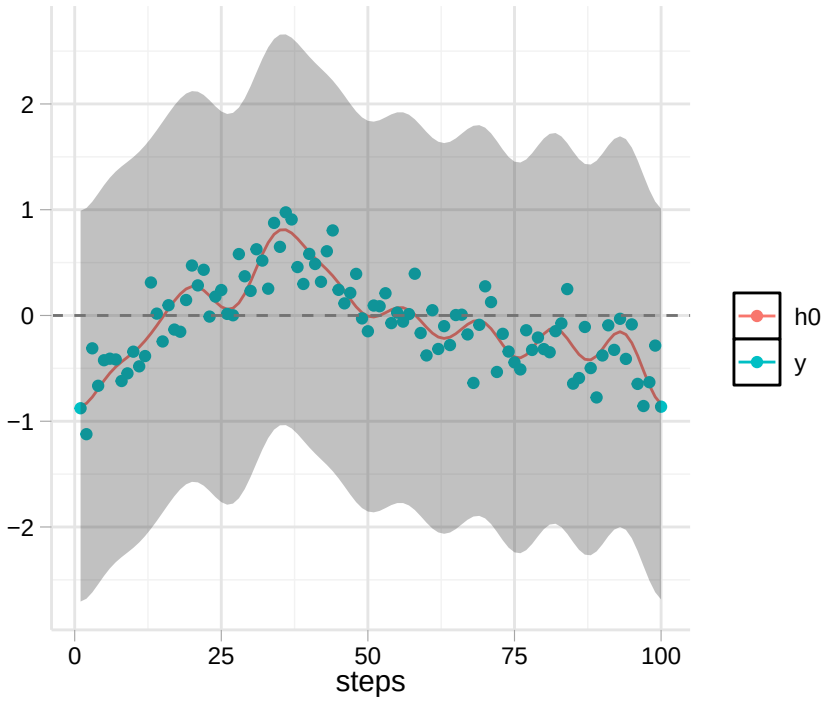

Figure 5: Maximum Aposterior Prediction of h_0.

```
Q_e = 4^2*diag(length(y))
Q_h0 = diag(length(y))
post.h0 <- function(y,A,Q_e, Q_h0){
  Q_post = (t(A)%*%Q_e)%*%A + Q_h0
  covar_post = solve(Q_post)
  data.frame(steps = seq(length(y)),
             y = y,
             mu = covar_post%*%(t((t(y)%*%Q_e)%*%A)),
             stdev = sqrt(diag(covar_post)))

}
h0.res = post.h0(y,A,Q_e,Q_h0)
```

Uncertainty bounds are found with a 95% confidence interterval, which means that $\hat{\mathbf{h}}_0 \pm 1.96\text{sd}(\hat{\mathbf{h}}_0)$. The result of the MAP is presented in figure 5 with uncertainty bounds. In this plot it is shown that the uncertainty bounds is relativly large, which means that our model isn't a very accurate predictor.

Exercise c

We will use different prior information $\mathbf{h}_0 \sim \mathcal{N}(\mathbf{0}, \Sigma)$, where the covariance matrix is given by the equation

$$\Sigma(i,j) = \exp\{-|x_i - x_j|/0.1\},$$

where $|x_i - x_j|$ is modular on the circle, and calculated like previously.

```
createQh0 <- function(x){
  Q_h0 = matrix(NA, nrow = length(x), ncol = length(x))
  for (i in seq(length(x))){
    for (j in seq(length(y))){
```

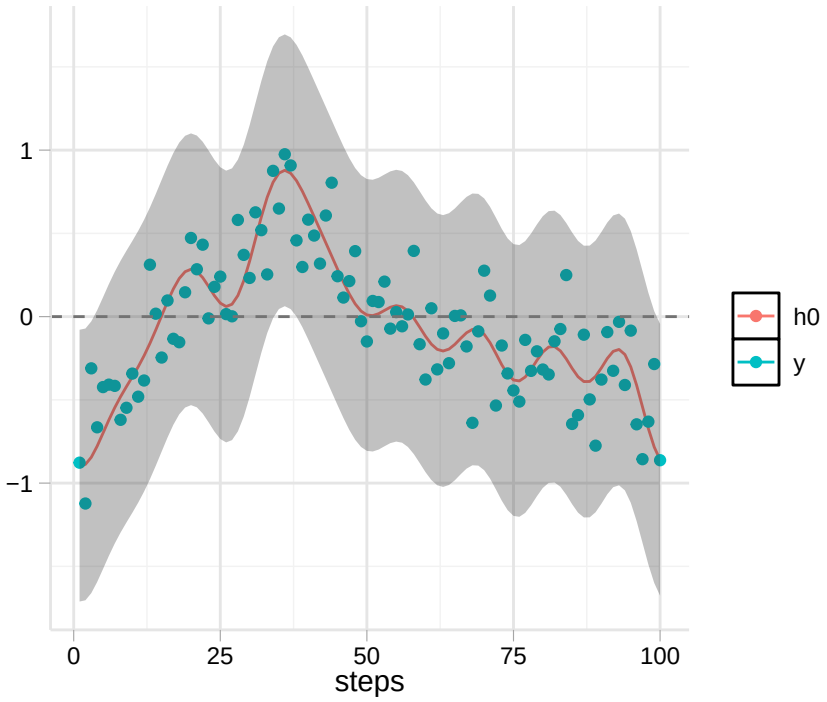

Figure 6: Maximum Aposterior Prediction of h_0.

```
      d = min(abs(x[i]-x[j]),1-abs(x[i]-x[j]))
      Q_h0[i,j] = exp(-d/0.1)
    }
  }
  solve(Q_h0)
}
Q_h0 = createQh0(x)
```

We still use the function *post.h0* from Exercise b to find the expectation and standard deviation.

```
h0.res2 = post.h0(y,A,Q_e,Q_h0)
```

In Figure 6 the result of the MAP of $\mathbf{h}_0$ is presented, and we observe that the uncertainty bounds are smaller than for the prediction in Figure 5. As for the estimation shown in Figure 4 where we used the truncated singular value expansion we lack the uncertainty bounds. A comparison between the models is shown in Figure 7, where we can see that the MAP predictions are close to similar, but with larger uncertainty bounds on the model from exercise b. We also observe that the TSVD solution has some similarity with the other solution, but differs alot at some points.

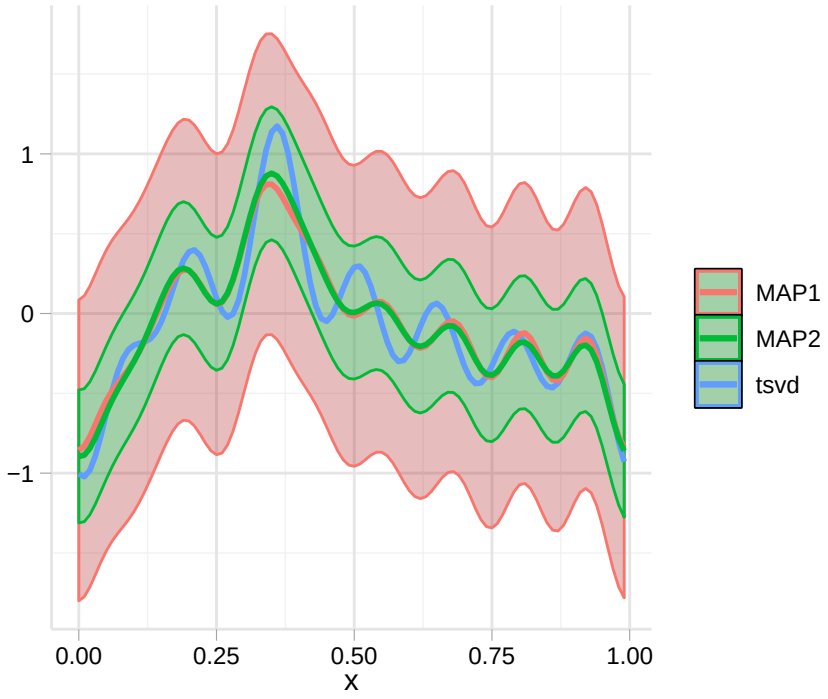

Figure 7: All prediction models presented together. The model from Exercise b is presented in *red* with corresponding uncertainty bounds, the model from Exercise c is green with corresponding uncertainty bounds, and TSVD solution is presented in blue.