Time Series Analysis

Assignment 4

Authors

Elli Georgiou (s223408)
Dimitris Voukatas (s230148)
Dimitris Sousounis (s230118)

May 3, 2023

Question 4.1: Presenting data

From the salinity and dissolved oxygen plots we observe different trends and modifications in the observations over time. The salinity data illustrate a rather steady range of 17 to 20 PSU, with occasional singular outliers. These reductions could be caused by changes in water composition or other environmental conditions. Furthermore, there are missing data points in both plots, which might be due to sensor or data recording equipment failures. Dissolved oxygen measurements exhibit a similar pattern of rather continuous fluctuations, with a range of 6 to 11 PSU with occasional spikes and dips. Lastly, dissolved oxygen levels rise sharply in the second part of the data.

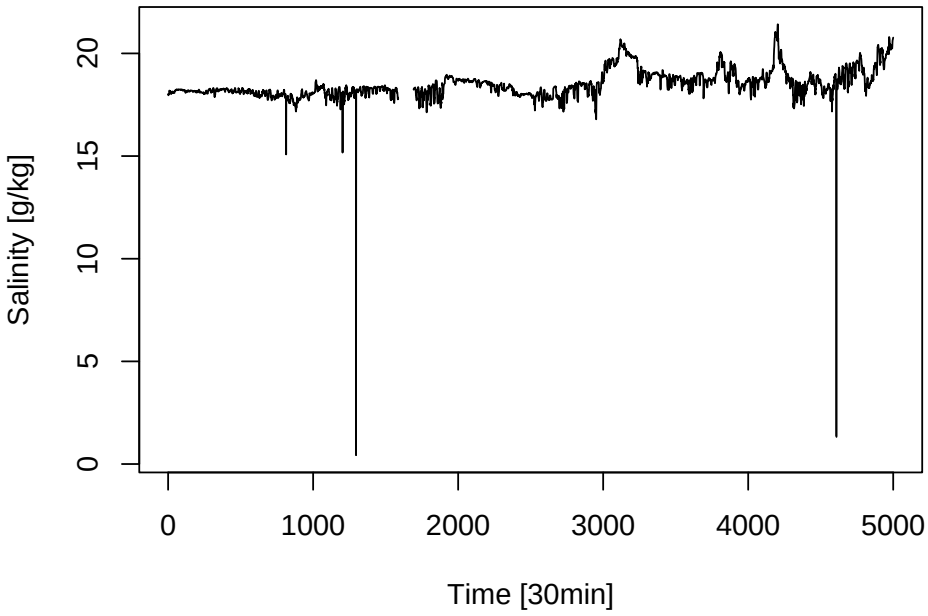

Figure 1: Salinity plot

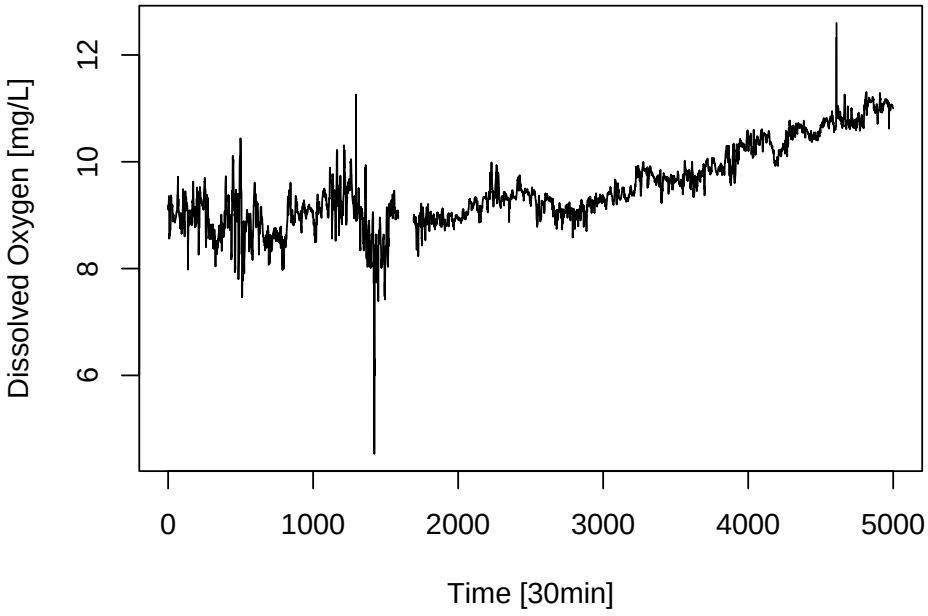

Figure 2: Dissolved oxygen plot

Question 4.2: Random walk state-space model of salinity

The salinity can be described by a random walk process that is observed at equidistant time points so it can be written as:

$$X_t = X_{t-1} + \varepsilon_t \tag{1}$$

where ε_t is the noise term. It is clear, that the process is an AR(1) model thus the size of matrix A is d=max(p,q+1)=max(1,1)=1 and the coefficient ϕ_1 is equal to -1. Furthermore, the salinity at time t is only reliant on its previous value at time t-1, so there is no input. From all the above, we can conclude that the state space model has only one state and the matrices that define it are A=[1], B=[0] and C=[1]. Similarly, since we have only one state and one observable output, the covariance matrices of the system and measurement noise are also 1-dimensional ($\Sigma_{1,t} = [\sigma_1^2], \Sigma_{2,t} = [\sigma_2^2]$). In conclusion, the state space model can be defined by the following equations

$$X_t = X_{t-1} + \varepsilon_{1,t} \tag{2}$$

$$Y_t = X_t + \varepsilon_{2,t} \tag{3}$$

the first being the system equation and the second one the observation equation.

Question 4.3: Pure Kalman filter

One step predictions along the data

In figure 3, the one step predictions along the data and 95% prediction intervals are shown. The presence of large outliers significantly affects the predictions and causes them to deviate from the rest of the data. Also it can be noticed, that due to the missing data, the variance of the predictions increase dramatically as do the prediction intervals.

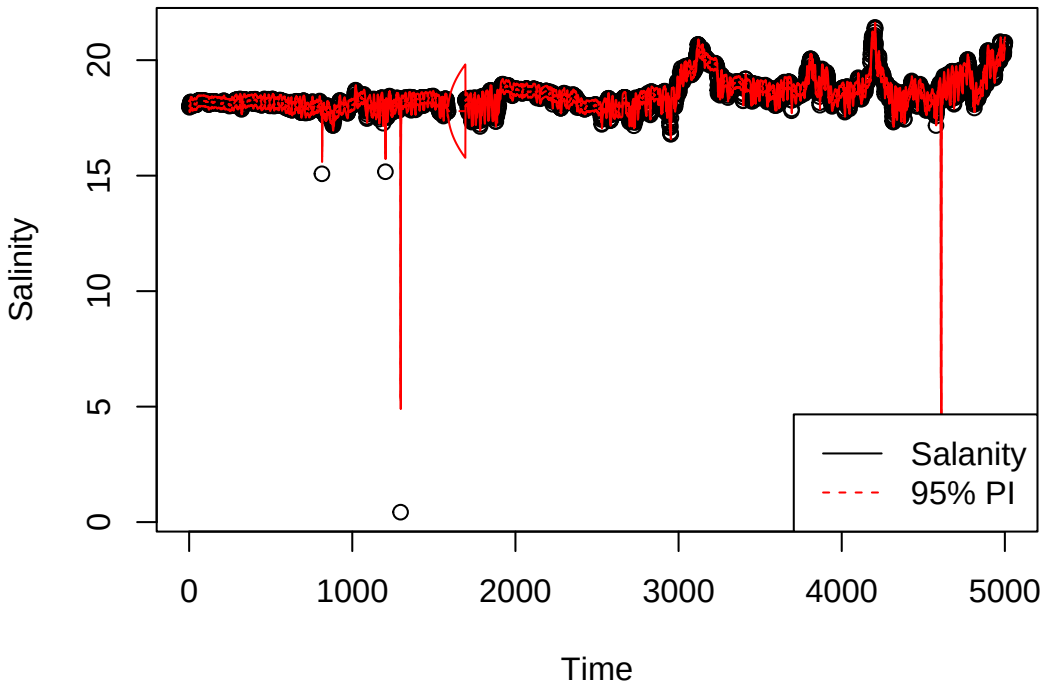

Figure 3: Salinity with 95% Prediction interval

Standardized one step prediction errors

Figure 4 displays the standardized one-step prediction errors. What can be observed is that the residuals corresponding to the outliers are significantly larger than the rest. The immediately next residual is also large but not as much and has an opposite sign.

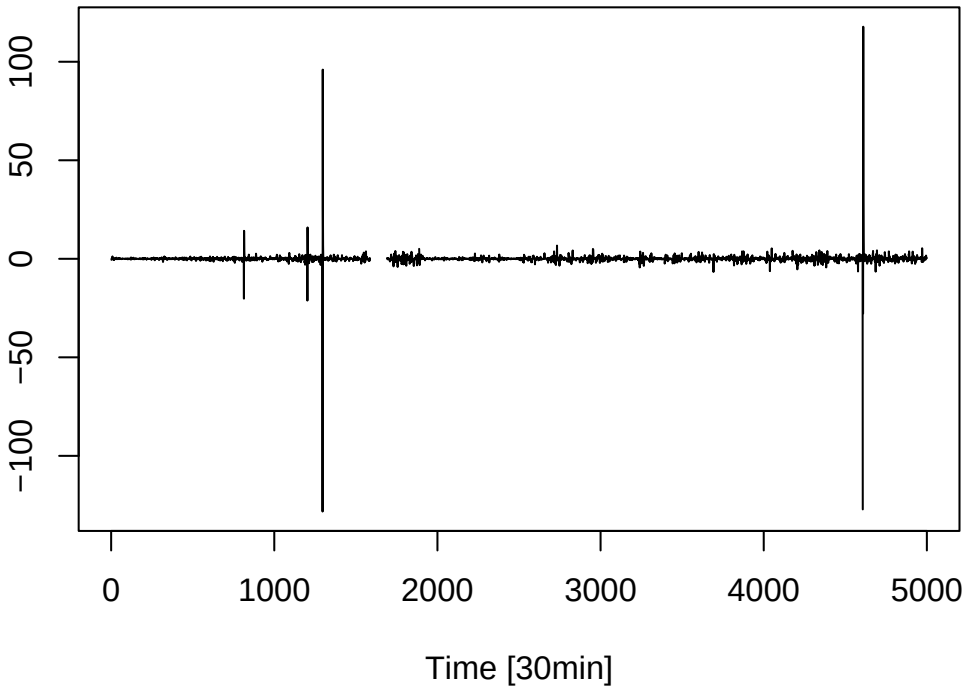

Figure 4: Standardized one-step prediction errors

Zooming into observations 800 to 950

We observe the same characteristics and trends in the zoomed in plots. What can also be observed in figure 5, is that the outlier at time t affects the prediction t+1 and not the prediction t. That is evident if we observe the equations of the Kalman filter:

$$\hat{X}_{t|t} = \hat{X}_{t|t-1} + K_t(Y_t - C\hat{X}_{t|t-1}) \tag{4}$$

$$\hat{X}_{t+1|t} = A\hat{X}_{t|t} + Bu_t \tag{5}$$

If Y_t is an outlier then the reconstructed value $\hat{X}_{t|t}$ and the prediction $\hat{X}_{t+1|t}$ is going to deviate greatly from the previous ones. Consequently, the error $\tilde{Y}_t = Y_t - C\hat{X}_{t|t-1}$ is also going to deviate. In our case the outlier is much smaller than the prediction $\hat{X}_{t|t-1}$ (which is calculated from the previous non-outlier observation) thus the prediction error is large and negative. In the next time step, the observation has a normal value while the prediction (calculated with the outlier) is large thus the prediction error is large but positive.

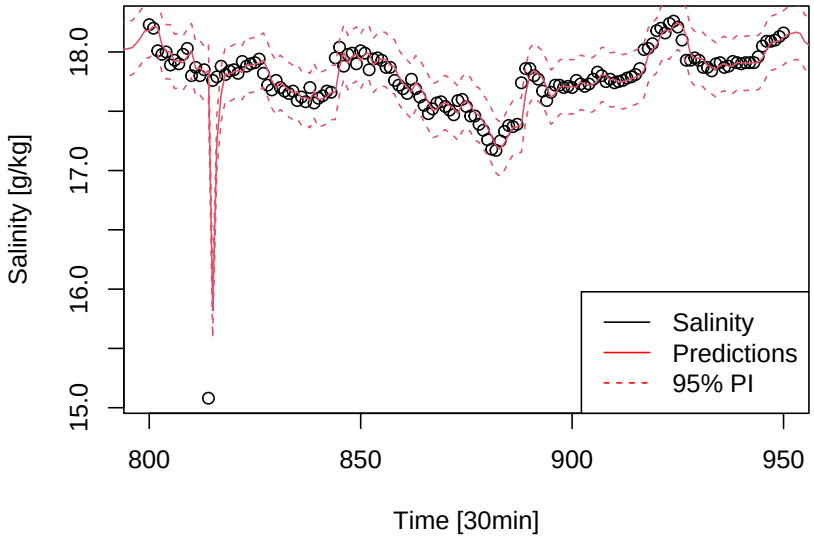

Figure 5: Zoomed in Salinity

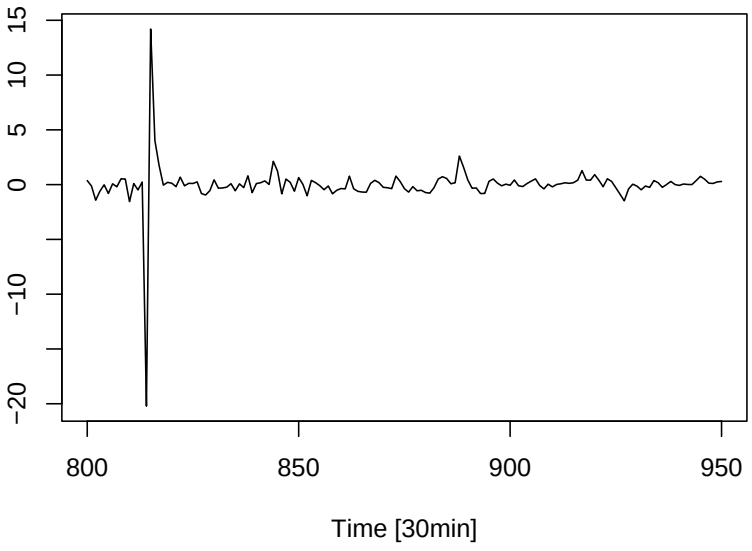

Figure 6: Zoomed in standardized one-step prediction errors

Defining Values

The final state of the filter is defined by the equation 4 and the equation

$$\Sigma_{t|t}^{xx} = \Sigma_{t|t-1}^{xx} - K_t C \Sigma_{t|t-1}^{xx} \tag{6}$$

For t=5000 these equations produce 20.75815 and 0.003660254 respectively.

Question 4.4: Skipping outliers when filtering

One-step predictions for index 800 to 950

As expected, in figure 7, the outlier does not influence the prediction. However, in figure 8 the large negative error remains while the positive one is gone. The indexes of the first five detected outliers are 814, 1203, 1296, 2733, 3692 and the number of observations that are skipped is 10.

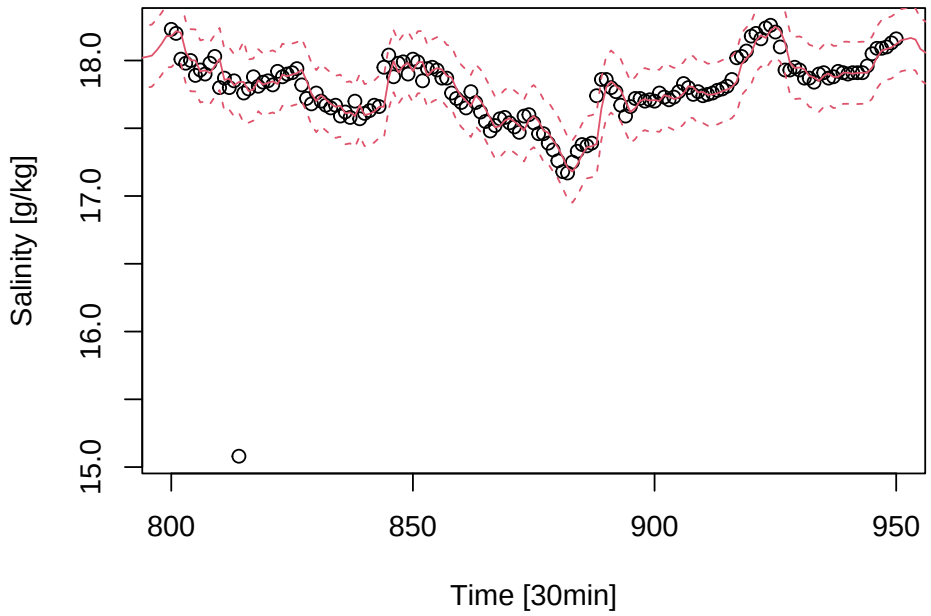

Figure 7: Zoomed in Salinity

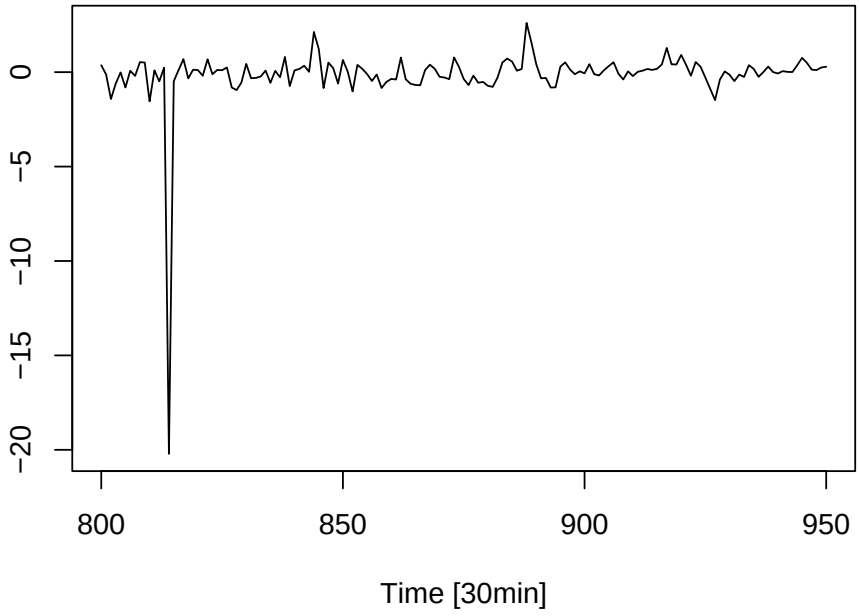

Figure 8: Zoomed in standardized one-step prediction errors

Defining Values

In this case, the final reconstructed state is $\hat{X}_{t|t} = 20.75815$ and the its variance is $\Sigma^{xx}_{t|t} = 0.003660254$. It can be observed that they are exactly the same whether we skip the outliers or not.

Question 4.5: Optimizing the variances

Sensible lower bound for the observation variance

The uncertainty of the observations should be indicated by the lower bound. Since the sensor is not perfect (assumably) and has an accuracy of two decimal digits, any value that is recorded is actually the midpoint of an interval with a width of 0.01. If we assume that the error is normal distributed then the variable of salinity (under the assumption that it is described by a random walk process) is also normal distributed. Therefore, if the standard deviation is 0.005 then a sensible lower bound for the observation variance would be 0.005^2.

ML estimates of the two parameters using the first 800 observations

The two maximum likelihood estimates for the system variance and observation variance using the first 800 observations are 0.001838911 and 0.000025 respectively.

Zooming into observations 800 to 950

With much lower system and observation variance the prediction intervals are much narrower. The prediction made by the pure Kalman filter seem to be closer to the outlier. In the case of the filter that skips the outliers, it can also be observed that many observations seem to be treated as such and lie outside the prediction intervals.

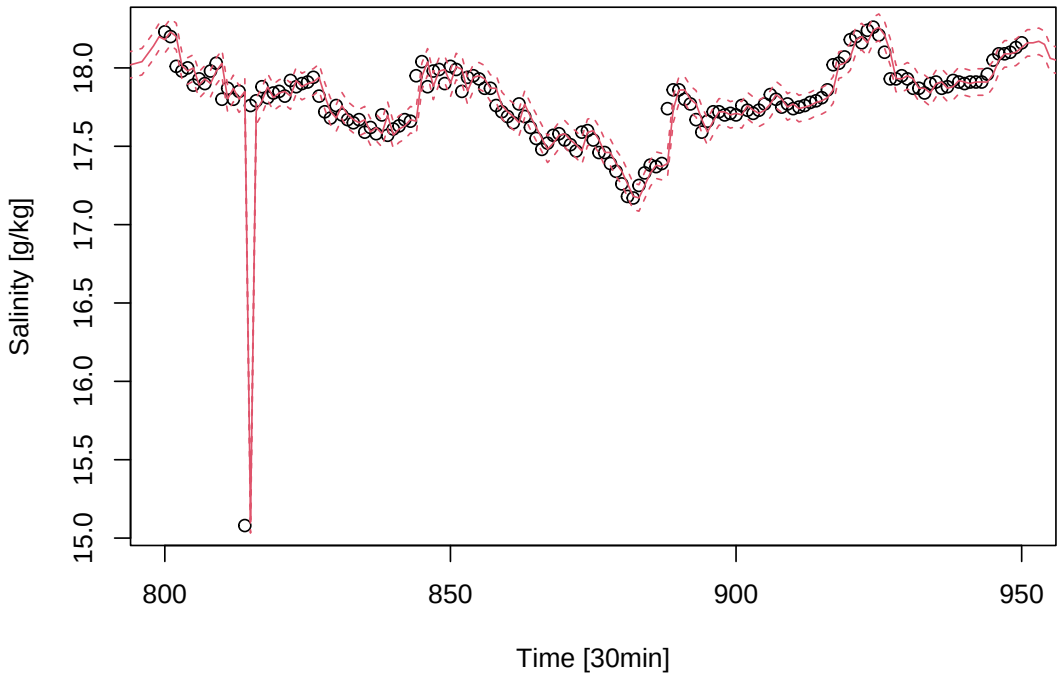

Figure 9: Zoomed in one-step predictions with 95% P.I. along the data

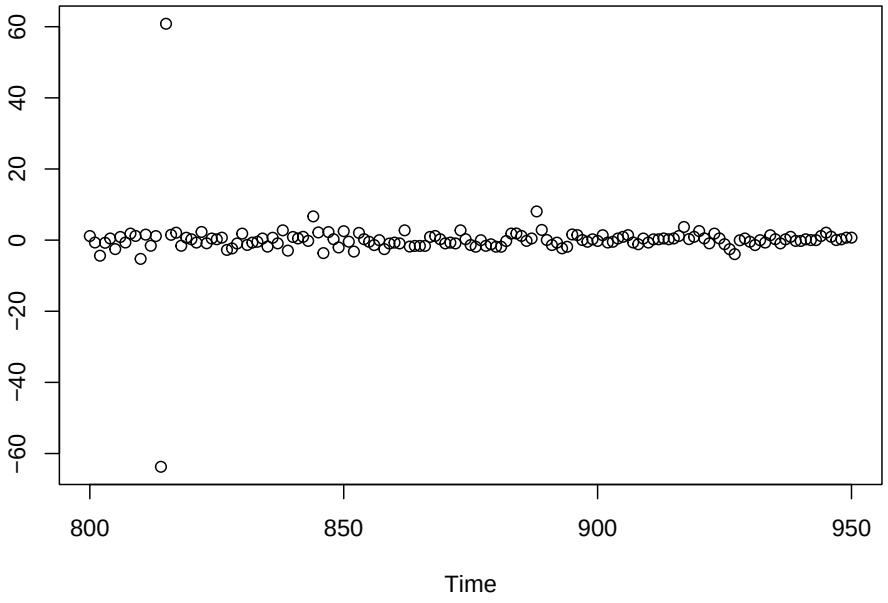

Figure 10: Zoomed in standardized one-step prediction errors

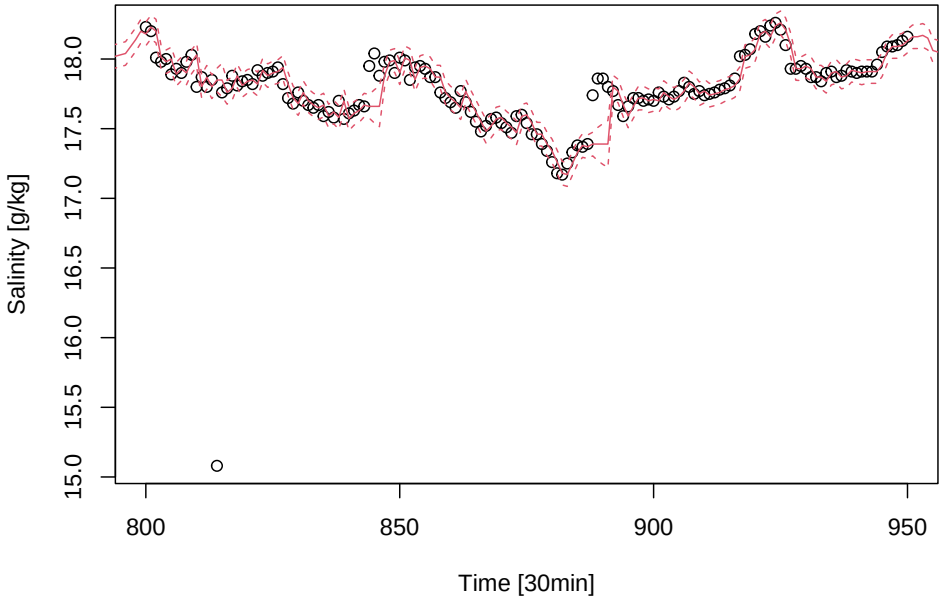

Figure 11: Zoomed in one-step predictions with 95% P.I. along the data - skipped outliers

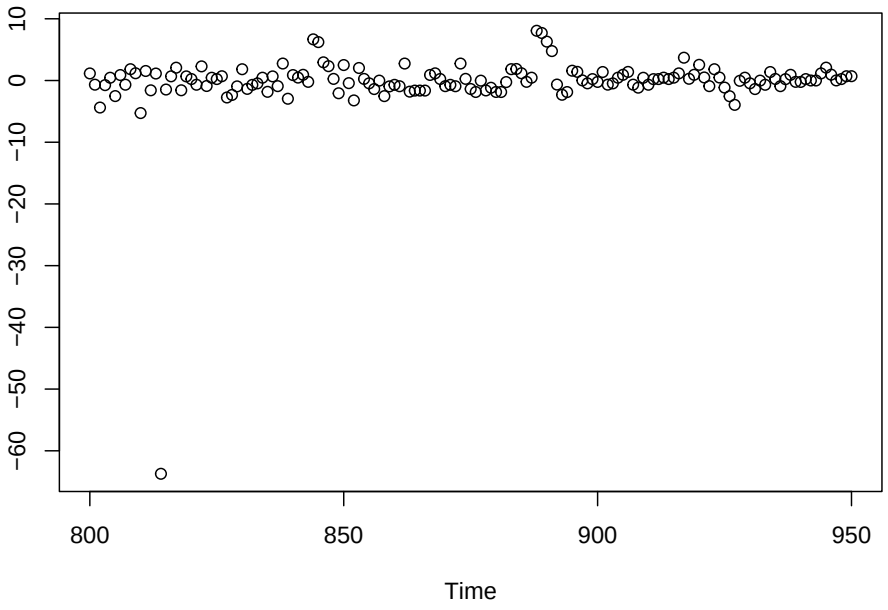

Figure 12: Zoomed in standardized one-step prediction errors - skipped outliers

Defining Values

In this case, the final reconstructed state is $\hat{X}_{t|t} = 20.7696$ and the its variance $\Sigma^{xx}_{t|t} = 2.466906 \times 10^{-5}$. The results are the same whether we use the pure Kalman filter or the one that skips the outliers. Compared with the results with the non-optimal variances, there is a small difference between the final reconstructed states while the difference between the variances is much larger.

Question 4.6: Model for dissolved oxygen

The concentration of dissolved oxygen is dependent on a number of factors. Its current value can be given as such

$$DO_t = \alpha DO_{t-1} + \beta J_{t-1} + \gamma I_{t-1} + \delta R_{t-1} + \varepsilon_t \tag{7}$$

where J_t is the rate of oxygen exchange with the atmosphere. Fick's Law of Diffusion states that the rate of exchange J is proportional to the concentration gradient ∇C and the diffusion coefficient D and the concentration gradient is proportional (says the internet) to

the difference of the saturation concentration of dissolved oxygen and the concentration of dissolved oxygen

$$J = -D \cdot \nabla C = -D \cdot (DO_{sat} - DO)/\Delta \tag{8}$$

If we define a new constant $\beta' = -\beta D/\Delta$ then equation 7 becomes

$$DO_t = \alpha DO_{t-1} + \beta'(DO_{sat_{t-1}} - DO_{t-1}) + \gamma I_{t-1} + \delta R_{t-1} + \varepsilon_t \tag{9}$$

Furthermore, the question specifies that the dissolved oxygen is a function of sunlight intensity and the oxygen exchange with the atmosphere, thus I_t and DO_{sat_t} are inputs. Additionally, in an environment without any factors as the aforementioned, the concentration of dissolved oxygen would remain the same, thus $\alpha = 1$. Finally, with the inclusion of respiration as a random walk, the state space model can be written

$$\begin{bmatrix} DO_t \\ R_t \end{bmatrix} = \begin{bmatrix} 1-\beta' & \delta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} DO_{t-1} \\ R_{t-1} \end{bmatrix} + \begin{bmatrix} \beta' & \gamma \\ 0 & 0 \end{bmatrix} \begin{bmatrix} DO_{sat_{t-1}} \\ I_{t-1} \end{bmatrix} + \varepsilon_{1,t} \tag{10}$$

$$Y_t = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} DO_t \\ R_t \end{bmatrix} + \varepsilon_{2,t} \tag{11}$$

R code

```r
rm(list=ls())

library(FKF)

data <- read.csv("A4_Kulhuse.csv", header=TRUE, fill = TRUE)

# Plot the data
plot(data$Sal, type="l", xlab = "Time [30min]", ylab = "Salinity [g/kg]")
plot(data$ODO, type="l", xlab = "Time [30min]", ylab = "Dissolved Oxygen [mg/L]")


# find model
A=matrix(1,nrow=1)
B=matrix(0,nrow=1)
C=matrix(1,nrow=1)
Sigma.1=matrix(.01,nrow=1)
Sigma.2=matrix(.005,nrow=1)

# pure kalman filter
kf1 <- fkf(a0=data$Sal[1], P0 = Sigma.1, dt = matrix(0, nrow = 1), Tt = A, ct=0,
           Zt = C, HHt = Sigma.1, GGt = Sigma.2, yt =
              matrix(data$Sal, nrow = 1))
```

```r
plot(data$Sal, type = "l")
with(kf1, matlines((at[1,]) +
   cbind(0,-1.96*sqrt(Pt[1,1,]),1.96*sqrt(Pt[1,1,])),
                   type="l", lty=c(1,2,2), col=2))
spe1 <- kf1$vt[1,]/sqrt(kf1$Ft[1,1,])
plot(spe1, type="l")
plot(800:950, data$Sal[800:950])
with(kf1, matlines((at[1,]) +
   cbind(0,-1.96*sqrt(Pt[1,1,]),1.96*sqrt(Pt[1,1,])),
                   type="l", lty=c(1,2,2), col=2))
plot(800:950, spe1[800:950], type="l")
kf1$at[length(kf1$at)]


####New plot


 # Plot the data
      plot(data$Sal, type = "p", xlab = "Time", ylab = "Salinity")

      # Plot the prediction interval
      matlines(kf1$at[1, ] + cbind(0, -1.96 * sqrt(kf1$Pt[1, 1, ]),
         1.96 * sqrt(kf1$Pt[1, 1, ])),
               type = "l", lty = c(2, 1, 1), col = "red")

      # Add a legend
      legend("bottomright", legend = c("Salanity", "95% PI"), col =
         c("black", "red"), lty = c(1, 2))

      # Plot a subset of the data
      plot(data$Sal[800:950], type = "p", xlab = "Time", ylab =
         "Salinity")

      # Plot the prediction interval
      matlines(kf1$at[1, 800:950] + cbind(0, -1.96 * sqrt(kf1$Pt[1,
         1, 800:950]), 1.96 * sqrt(kf1$Pt[1, 1, 800:950])),
               type = "l", lty = c(2, 1, 1), col = "red")

      # Add a legend
      legend("bottomright", legend = c("Subset of Salinity", "95%
         PI"), col = c("black", "red"), lty = c(1, 2))

      plot(800:950, data$Sal[800:950], xlab = "Time [30min]", ylab =
         "Salinity [g/kg]")

      with(kf2, matlines(pred[,1] + cbind(0,
         -1.96*sqrt(Sigma.xx.pred[1,1,]),
```

```
                                                1.96*sqrt(Sigma.xx.pred[1,1,])),
                                                   type = "l",
                           lty = c(1,2,2), col = 2))

      legend("bottomright", legend = c("Salinity", "Predictions" ,
         "95% PI"), col = c("black","red","red"), lty = c(1,1, 2))









source("kalman.R")
kf2 <- kalman(Y=c(data$Sal), A=A, B=B,
   u=matrix(rep(0,length(c(data$Sal)))), C=C
               , Sigma.1=Sigma.1, Sigma.2=Sigma.2, Xhat0=data$Sal[1],
                  V0=Sigma.1,
               verbose=TRUE)
plot(data$Sal, type = "l", xlab = "Time [30min]", ylab = "Salinity
   [g/kg]")
with(kf2, matlines(pred[,1] + cbind(0,
   -1.96*sqrt(Sigma.xx.pred[1,1,]),
                                    1.96*sqrt(Sigma.xx.pred[1,1,])),
                                       type = "l",
                                    lty = c(1,2,2), col = 2))
spe2 <- (data$Sal-kf2$pred[-length(kf2$pred)])/
                           sqrt(kf2$Sigma.yy.pred[1,1,-length(kf2$Sigma.yy.pred)])
plot(spe2, type = "l",xlab= "Time [30min]", ylab = "")
plot(800:950, data$Sal[800:950], xlab = "Time [30min]", ylab =
   "Salinity [g/kg]")
with(kf2, matlines(pred[,1] + cbind(0,
   -1.96*sqrt(Sigma.xx.pred[1,1,]),
                                    1.96*sqrt(Sigma.xx.pred[1,1,])),
                                       type = "l",
                                    lty = c(1,2,2), col = 2))

plot(800:950, spe2[800:950], type="l",xlab= "Time [30min]", ylab =
   "")

kf2$rec[length(kf2$rec)-1]
kf2$pred[length(kf2$pred)]
kf2$Sigma.xx.rec[length(kf2$Sigma.xx.rec)]
kf2$Sigma.xx.pred[length(kf2$Sigma.xx.pred)]
kf2$Sigma.yy.rec[length(kf2$Sigma.yy.rec)]
kf2$Sigma.yy.pred[length(kf2$Sigma.yy.pred)]
kf2$K[,,length(kf2$K)]

# skip outliers
```

```
source("kalmanADJ(new).R")
kf3 <-
   kalmanADJ(Y=c(data$Sal),A=A,B=B,u=matrix(rep(0,length(c(data$Sal)))),
   C=C
               , Sigma.1=Sigma.1, Sigma.2=Sigma.2, Xhat0=data$Sal[1],
                  V0=Sigma.1,
               verbose=TRUE)
plot(800:950, data$Sal[800:950], xlab = "Time [30min]", ylab =
   "Salinity [g/kg]")
with(kf3, matlines(pred[,1] + cbind(0,
   -1.96*sqrt(Sigma.xx.pred[1,1,]),
                                    1.96*sqrt(Sigma.xx.pred[1,1,])),
                                       type = "l",
                                    lty = c(1,2,2), col = 2))
spe3 <- (data$Sal-kf3$pred[-length(kf3$pred)])/
  sqrt(kf3$Sigma.yy.pred[1,1,-length(kf3$Sigma.yy.pred)])
plot(800:950, spe3[800:950], type="l",xlab= "Time [30min]", ylab =
   "")

kf3$skindex
length(kf3$skindex)

kf3$rec[length(kf3$rec)-1]
kf3$pred[length(kf3$pred)]
kf3$Sigma.xx.rec[length(kf3$Sigma.xx.rec)]
kf3$Sigma.xx.pred[length(kf3$Sigma.xx.pred)]
kf3$Sigma.yy.rec[length(kf3$Sigma.yy.rec)]
kf3$Sigma.yy.pred[length(kf3$Sigma.yy.pred)]
kf3$K[,,length(kf3$K)]

# ml estimates
sal <- data$Sal[1:800]

loglik1 <- function(theta){
  S1 <- matrix(theta[1], nrow = 1)
  S2 <- matrix(theta[2], nrow = 1)
  kf4 <- fkf(a0=sal[1], P0 = S1, dt = matrix(0, nrow = 1), Tt = A,
     ct=0,
              Zt = C, HHt = S1, GGt = S2, yt = matrix(sal, nrow = 1))
  ll <- -0.5 * sum(log(kf4$Pt[1,1,1:800]+S2))-0.5 * sum((sal[1:800]
        -kf4$at[1,1:800])^2/(kf4$Pt[1, 1, 1:800]+S2))
  return(ll)
}
(res1 <- optim(c(.01,.005), loglik1, control = list(fnscale = -1),
                method = "L-BFGS-B", lower = c(1e-4,0.005^2)))
# similarly to the assumed values in 4.3, the lower bound of system
   variance was
```

```r
# set slightly higher than the lower bound of the observation variance

loglik2 <- function(theta){
  S1 <- matrix(theta[1], nrow = 1)
  S2 <- matrix(theta[2], nrow = 1)
  kf4 <- kalmanADJ(sal, A=A, B=B, u=matrix(rep(0,length(c(sal)))), C=C
                   , Sigma.1=S1, Sigma.2=S2, Xhat0=sal[1], V0=S1,
                   verbose=TRUE)
  ll <- -0.5 * sum(log(kf4$Sigma.yy.pred[1,1,2:800])
                   +(sal[-1]-kf4$pred[2:800,1])^2 /
                      kf4$Sigma.yy.pred[1,1,2:800])
  return(ll)
}
(res2 <- optim(c(0.01,0.005), loglik2, control = list(fnscale = -1),
               method = "L-BFGS-B", lower = c(1e-4,0.005^2))) # not converging

loglik3 <- function(theta){
  S1 <- matrix(exp(theta[1]), nrow = 1)
  S2 <- matrix(exp(theta[2]), nrow = 1)
  kf4 <- kalmanADJ(sal, A=A, B=B, u=matrix(rep(0,length(c(sal)))), C=C
                   , Sigma.1=S1, Sigma.2=S2, Xhat0=sal[1], V0=S1,
                   verbose=TRUE)
  ll <- -0.5 * sum(log(kf4$Sigma.yy.pred[1,1,2:800])
                   +(sal[-1]-kf4$pred[2:800,1])^2 /
                      kf4$Sigma.yy.pred[1,1,2:800])
  return(ll)
}
(res3 <- optim(c(log(0.01),log(0.005)), loglik3, control = list(fnscale = -1),
               method = "L-BFGS-B", lower =
                  c(log(1e-4),log(0.005^2)))) # conv

exp(res3$par)

Sigma.1.est <- matrix(exp(res3$par[1]), nrow = 1)
Sigma.2.est <- matrix(exp(res3$par[2]), nrow = 1)

kf4 <- kalmanADJ(Y=c(data$Sal), A=A, B=B,
   u=matrix(rep(0,length(c(data$Sal)))), C=C
               , Sigma.1=Sigma.1.est, Sigma.2=Sigma.2.est,
                  Xhat0=data$Sal[1],
               V0=Sigma.1, verbose=TRUE)
```

```r
plot(800:950, data$Sal[800:950], xlab = "Time [30min]", ylab =
   "Salinity [g/kg]")
with(kf4, matlines(pred[,1] + cbind(0,
   -1.96*sqrt(Sigma.xx.pred[1,1,]),
                   1.96*sqrt(Sigma.xx.pred[1,1,])), type = "l", lty
                      = c(1,2,2),
                   col = 2))
kf4$rec[length(kf5$rec)-1]
kf4$Sigma.xx.rec[length(kf5$Sigma.xx.rec)]
kf5 <- kalman(Y=c(data$Sal), A=A, B=B,
   u=matrix(rep(0,length(c(data$Sal)))), C=C
                  , Sigma.1=Sigma.1.est, Sigma.2=Sigma.2.est,
                    Xhat0=data$Sal[1],
                  V0=Sigma.1, verbose=TRUE)
plot(800:950, data$Sal[800:950], xlab = "Time [30min]", ylab =
   "Salinity [g/kg]")
with(kf5, matlines(pred[,1] + cbind(0,
   -1.96*sqrt(Sigma.xx.pred[1,1,]),
                   1.96*sqrt(Sigma.xx.pred[1,1,])), type = "l", lty =
                     c(1,2,2),
                   col = 2))
kf5$rec[length(kf5$rec)-1]
kf5$Sigma.xx.rec[length(kf5$Sigma.xx.rec)]
```

Listing 1: My R code

Kalman Adjusted code

```r
kalmanADJ <-
   function(Y,A,B=NULL,u=NULL,C,Sigma.1=NULL,Sigma.2=NULL,debug=FALSE,
                      V0=Sigma.1,Xhat0=NULL,n.ahead=1,skip=0,verbose=FALSE){

  ## predictions through data are one-step predictions. n.ahead means
  ## how long we must keep predict after data. These are of course
  ## predictions of different step lengths.

  ## Y has to be columns.
  if(class(Y)=="numeric"){
    dim.Y <- c(length(Y),1)
    Y <- matrix(Y,ncol=1)
   } ## else {
  dim.Y <- dim(Y)
  ##  }

  ## Definition of default variables
  ## A and C must be supplied
```

```
  nstates <- dim(A)[1]

  ## these default values don't make much sense
  if(is.null(Sigma.1)){
    Sigma.1 <- diag(rep(1,nstates))
  }
  if(is.null(Sigma.2)){
    Sigma.2 <- diag(rep(1,dim.Y[2]))
  }

  if(is.null(B)){
    B <- matrix(rep(0,nstates),ncol=1)
  }
  if(is.null(u)){
    u <- matrix(rep(0,dim.Y[1]+n.ahead),ncol=1)
  }
  if(is.null(V0)){
    V0 <- Sigma.1
  }
  if(is.null(Xhat0)){
    Xhat0 <- matrix(rep(0,nstates),ncol=1)
  }


  ## i stedet for (10.79)
  X.hat <- Xhat0
  ## (10.80)
  Sigma.xx <- V0
  ## (10.78) (8.78)
  Sigma.yy <- C%*%Sigma.xx%*%t(C)+Sigma.2

  ## for saving reconstruction
  X.rec <- array(dim=c(dim.Y[1]+n.ahead,nstates))
  X.pred <- array(dim=c(dim.Y[1]+n.ahead,nstates))

  ## for saving K, Sigmas.
  if(verbose){
      K.out <-
         array(dim=c(dim(Sigma.xx%*%t(C)%*%solve(Sigma.yy)),dim.Y[1]))
      Sigma.xx.rec <- array(dim=c(dim(Sigma.xx),dim.Y[1]))
      Sigma.yy.rec <- array(dim=c(dim(Sigma.yy),dim.Y[1]))
      Sigma.xx.pred <- array(dim=c(dim(Sigma.xx),dim.Y[1]+n.ahead))
      Sigma.yy.pred <- array(dim=c(dim(Sigma.yy),dim.Y[1]+n.ahead))

  }
  #index <- c(dim.Y[1]+n.ahead) # or array(dim=c(dim.Y[1]+n.ahead))
  index <- vector()
```

```
  for(tt in (skip+1):dim.Y[1]){
    ## (10.75) (8.75)
    K <- Sigma.xx%*%t(C)%*%solve(Sigma.yy)

    ## (10.73) (8.73) - reconstruction
    if(!any(is.na(Y[tt,])) && abs(Y[tt,]-X.hat)<6*sqrt(Sigma.yy)){
      # At first everything is thrown away if one is missing
    X.hat <- X.hat+K%*%(t(Y[tt,])-C %*% as.matrix(X.hat))
    X.rec[tt,] <- X.hat
    ## (10.74) (8.74)
    Sigma.xx <- Sigma.xx-K%*%C%*%Sigma.xx
    } else if (!any(is.na(Y[tt,])) &&
       abs(Y[tt,]-X.hat)>6*sqrt(Sigma.yy)) {
      #index[tt] <- tt
      index <- c(index,tt)
    } else if (any(is.na(Y[tt,]))) {
      X.rec[tt,] <- X.hat
    }

    if(verbose){
        Sigma.xx.rec[,,tt] <- Sigma.xx
        Sigma.yy.rec[,,tt] <- Sigma.yy
    }

    ##(10.76) (8.76) - prediction
    X.hat <- A%*%X.hat + B%*%t(matrix(as.numeric(u[tt,]),nrow=1))
    X.pred[tt+1,] <- X.hat

    ##(10.77) (8.77)
    Sigma.xx <- A%*%Sigma.xx%*%t(A)+Sigma.1
    ##(10.78) (8.78)
    Sigma.yy <- C%*%Sigma.xx%*%t(C)+Sigma.2

    if(verbose){
        K.out[,,tt] <- K
#### these are the prediction error variance-covariances
        Sigma.xx.pred[,,tt+1] <- Sigma.xx
        Sigma.yy.pred[,,tt+1] <- Sigma.yy
    }

  }

if(n.ahead>1){
    for(tt in dim.Y[1]+(1:(n.ahead-1))){
      X.hat <- A%*%X.hat + B%*%t(matrix(u[tt,],nrow=1))
      X.pred[tt+1,] <- X.hat
      Sigma.xx <- A%*%Sigma.xx%*%t(A)+Sigma.1
```

```
      Sigma.xx.pred[,,tt+1] <- Sigma.xx
      Sigma.yy.pred[,,tt+1] <- C%*%Sigma.xx%*%t(C)+Sigma.2
    }
  }
  if(verbose){
      out <- list(rec=X.rec,pred=X.pred,
                  K=K.out,Sigma.xx.rec=Sigma.xx.rec,Sigma.yy.rec=Sigma.yy.rec,
                  Sigma.xx.pred=Sigma.xx.pred,Sigma.yy.pred=Sigma.yy.pred,
                  skindex=index)
  } else {
      out <- list(rec=X.rec,pred=X.pred, skindex=index)
  }
  return(out)
}
```

Listing 2: Kalman adjusted code