Time Series Analysis
Assignment 4

Author

Ghassen Lassoued - s196609

May 15, 2020

Contents

1 Question 4.1

The assignment will only look into the salinity and the dissolved oxygen. Two time series are defined then :
S_t: salinity
DO_t: dissolved oxygen

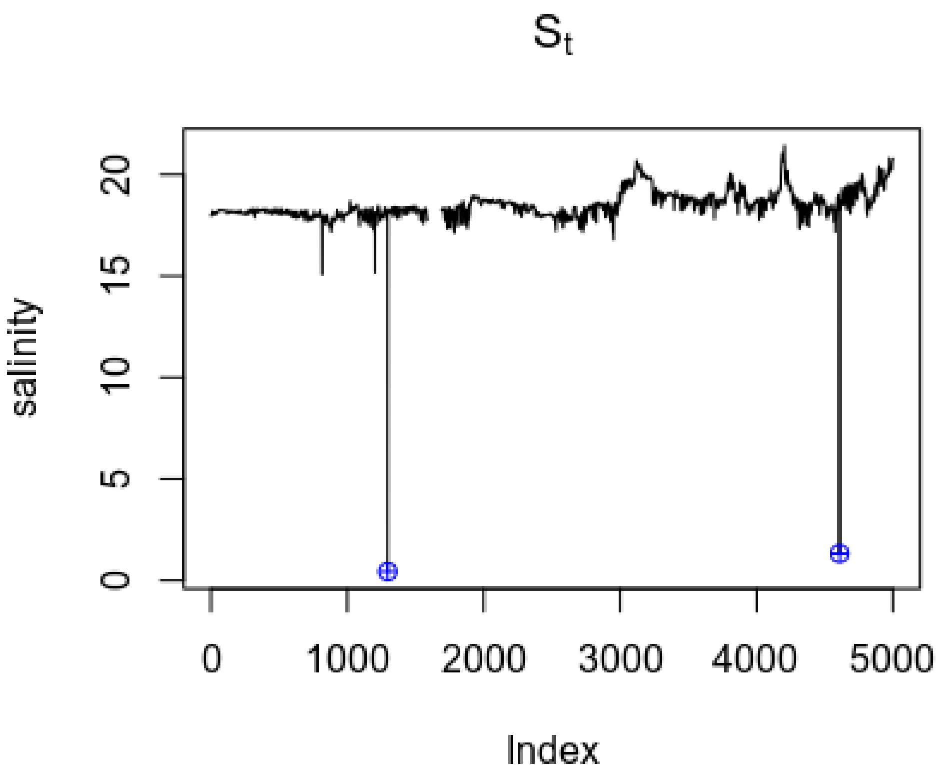

Figure 1: plotting data

It can be seen that the salinity data has some outliers. One can ask whether the first 2 points that are not following the same trend around index 1000 are outliers or no. If one considers only the first part of the plot (till index 2500), one will say that they are outliers. Considering the second part of plot, one might consider them normal because there are several points having more salinity than the rest majority. Still more likely at this point, these two first points are most likely outliers.
It can also be seen the missing data around index 1500

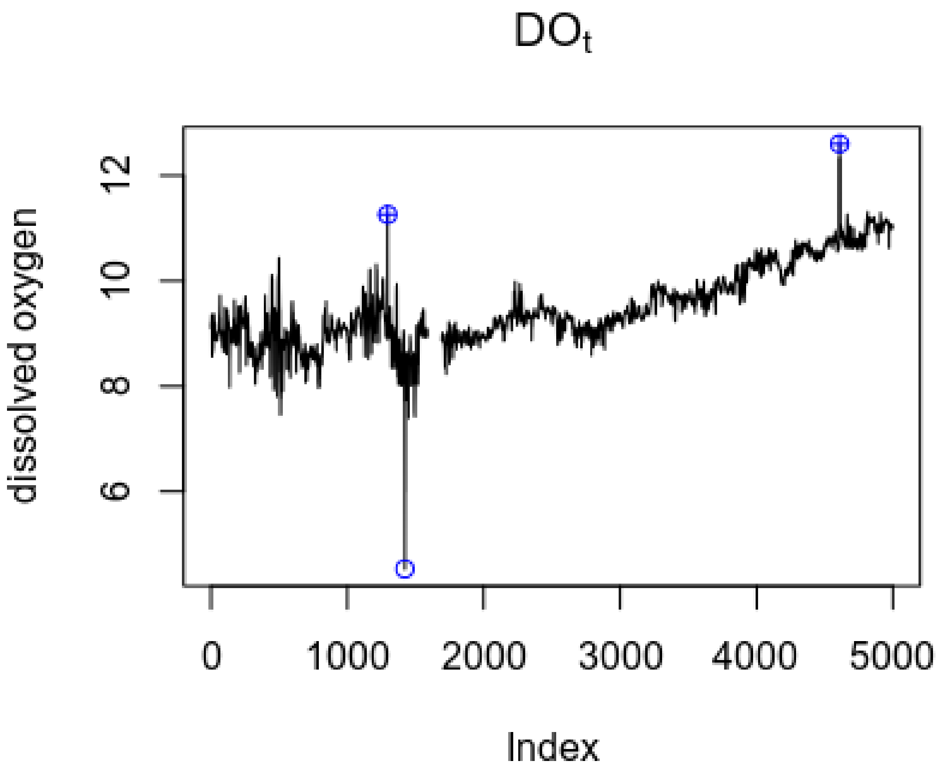

Figure 2: plotting data

There are some outliers also in the second plot. There are also missing data around index 1500. Going back to the data in csv form, it can be said that the two times series have the gap at exact same moments. In fact, from index 1588 to 1692 S_t and DO_t are NAs. What is new in the second plot is a linear increase from around index 3000.

2 Question 4.2

According to 10.1 from the book p284:

$$X_t = \mathbf{A}_t X_{t-1} + \mathbf{B}_t u_{t-1} + e_t \tag{1}$$
$$S_t = \mathbf{C}_t X_t + m_t \tag{2}$$

such that X_t: state of salinity
A_t=1, B_t=1 because there is no input, the latent noise is

$$e_t \sim \mathcal{N}(0, \Sigma_1) \tag{3}$$

S_t observed (i.e measured) salinity (denotes the observations)
C_t=1 and the noise of the measurable output S_t is

$$m_t \sim \mathcal{N}(0, \Sigma_1) \tag{4}$$

e_t is white noise. m_t is white noise. They are uncorrelated.

It should be mentioned that in the model, the state space and observations are univariates.

3 Question 4.3

To plot the one step prediction error, the missing data is substituted by the corresponding predictions.
To calculate the 95% prediction interval, the following formula is used

$$X_{t+1|t} \pm 1.96 \cdot \sqrt{\Sigma_{t+1|t}^{ss}} \tag{5}$$

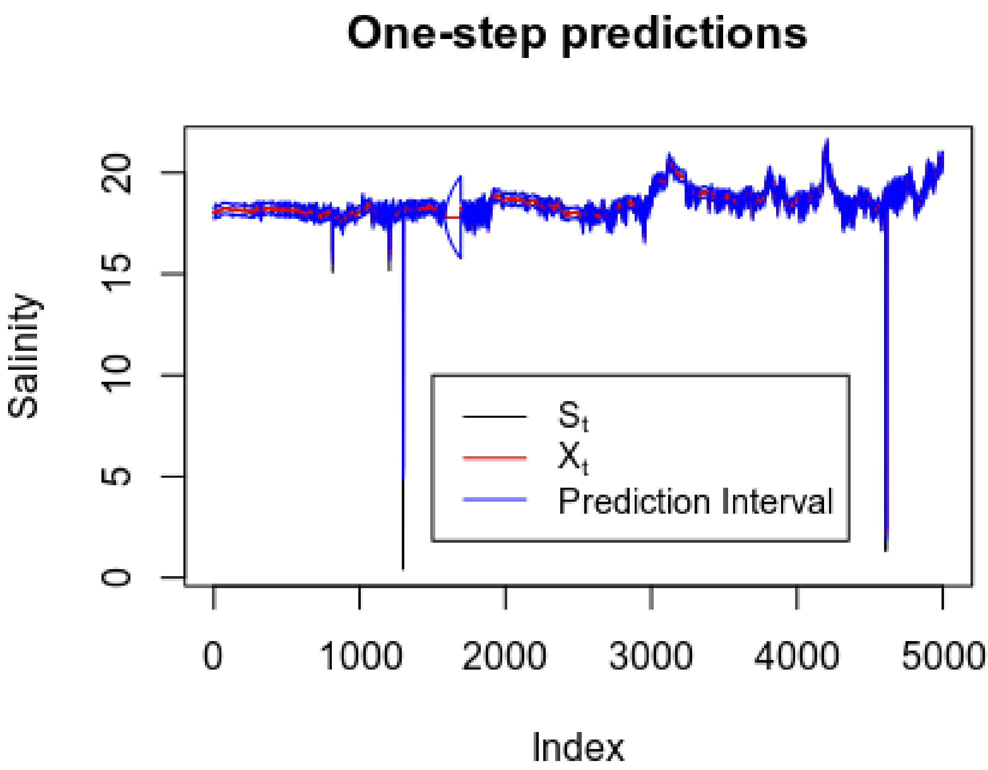

Figure 3

1)
The predicted values are within the confidence interval. It is hard too see it from this plot. It is much easier to refer to the zoom plot next to see this.
One can observe that the variance of the prediction interval increased around index 1500. This is due to the missing data there.
It is seen that the prediction intervals goes with the outliers i.e they follow them.
2)
$\hat{X}_{t+1|t} = E[X_{t+1}]$ is the one-step predicion of X_t
The definition of one step prediction error is:

$$\tilde{S}_{t+1|t} = S_{t+1} - \hat{X}_{t+1|t} \tag{6}$$

its variance:

$$\mathrm{Var}[S_{t+1}|S_0, ..., S_t] = \mathrm{Var}[\tilde{S}_{t+1}] = \Sigma^{ss}_{t+1|t} \tag{7}$$

So the standardized one step prediction errors are defined as:

$$\tilde{S}_{t+1|t} = \frac{S_{t+1} - \hat{X}_{t+1|t}}{\sqrt{\Sigma^{ss}_{t+1|t}}} \tag{8}$$

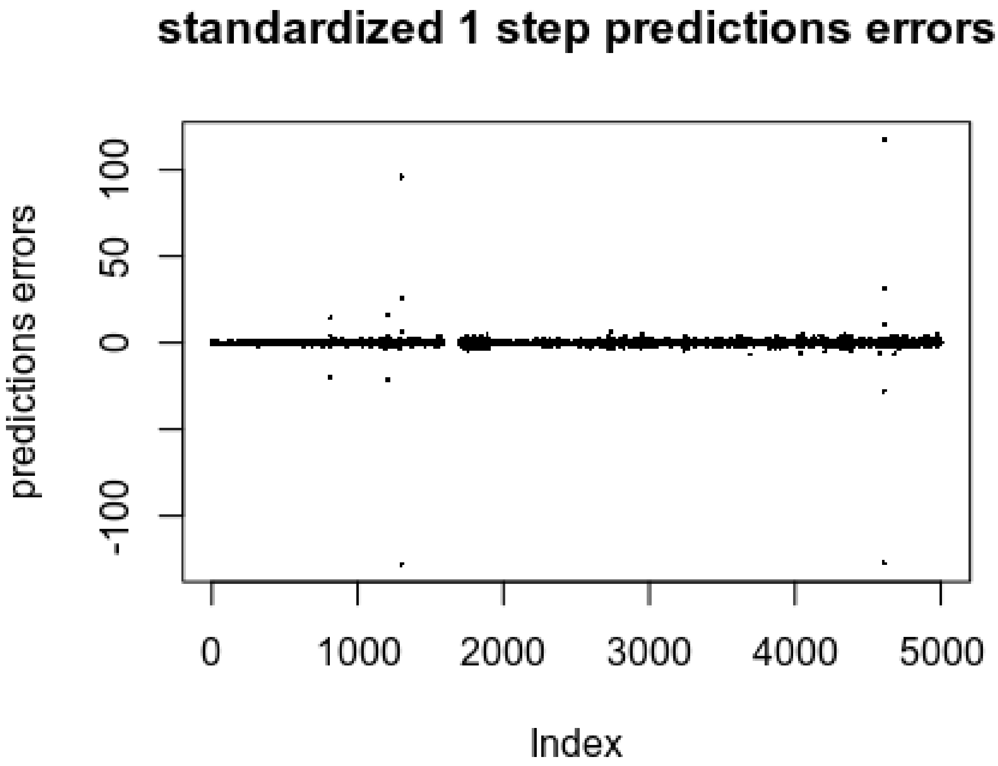

Figure 4

Again, the gap in data caused a gap in the one step prediction errors plot.
One can also notice that each time there is an outlier in the salinity plot, there are 2 points in predictions errors that do not follow the rest of points which are in 0. It seems that the absolute value of the positive point is smaller than the negative one.

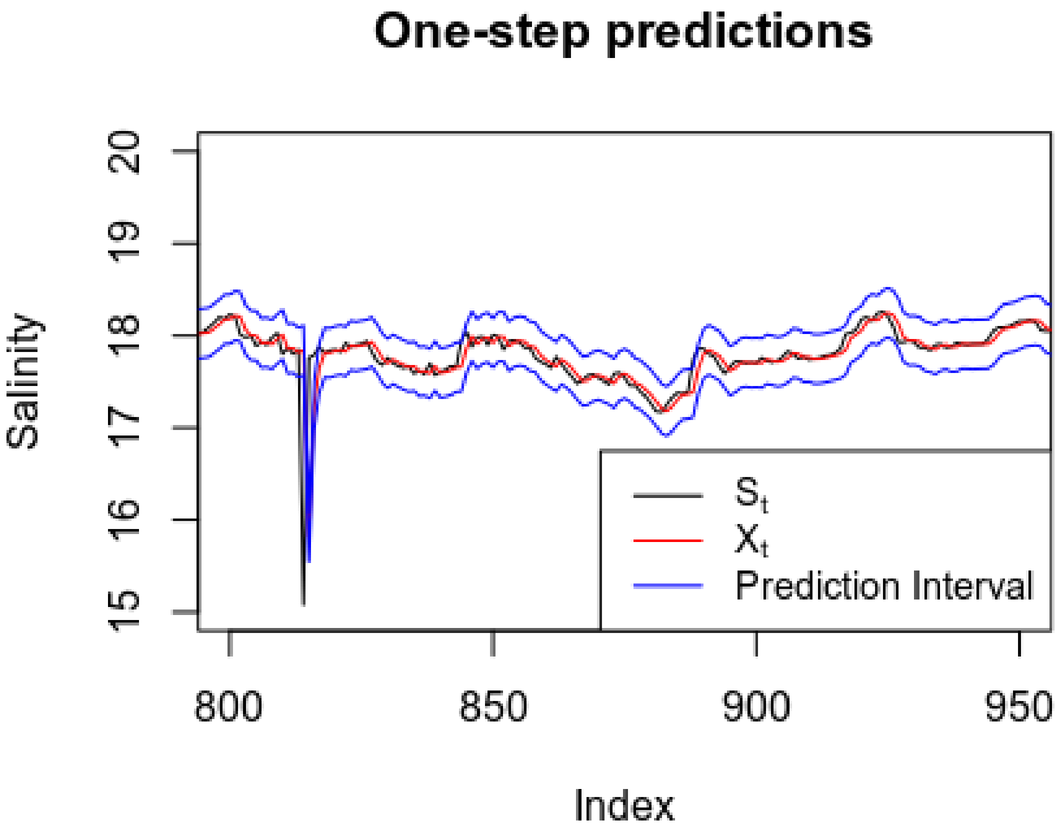
One-step predictions
Salinity
15
16
17
18
19
20
800
850
900
950
Index
S_t
X_t
Prediction Interval

Figure 5

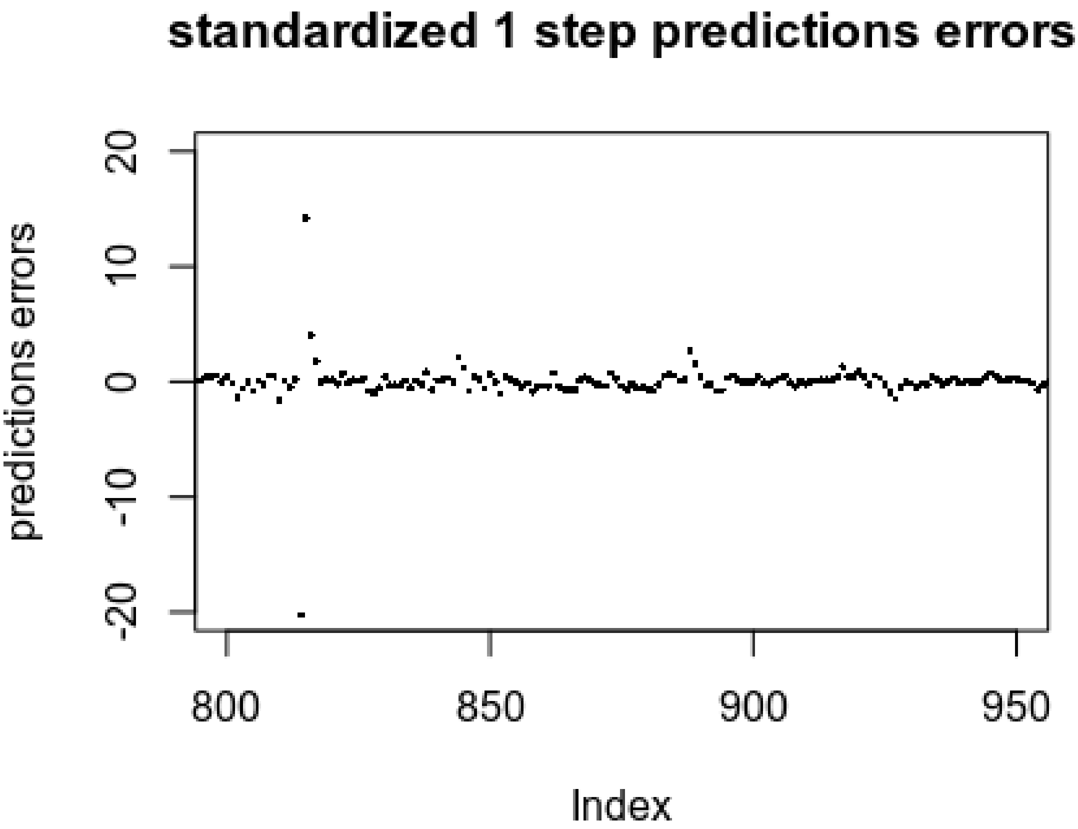

Figure 6

3)
With the zoom, one can see the good fit of confidence interval i.e inside it there is the predictions except for outliers. In fact it can be seen how the prediction interval follows the outlier down.
The outlier in black in S_t cause the next prediction and prediction interval to go down. As a result, the prediction error goes down also (negative point), then it goes up (positive point). The absolute negative value is greater than the positive one.
4)
The final state of filter is given by:

$$\hat{X}_{5000} = 20.75815 \tag{9}$$

and

$$\Sigma^{xx}_{5000|5000} = 0.003660254 \tag{10}$$

4 Question 4.4

Please refers to the function "Kalman_filter" in appendix in code.

1)

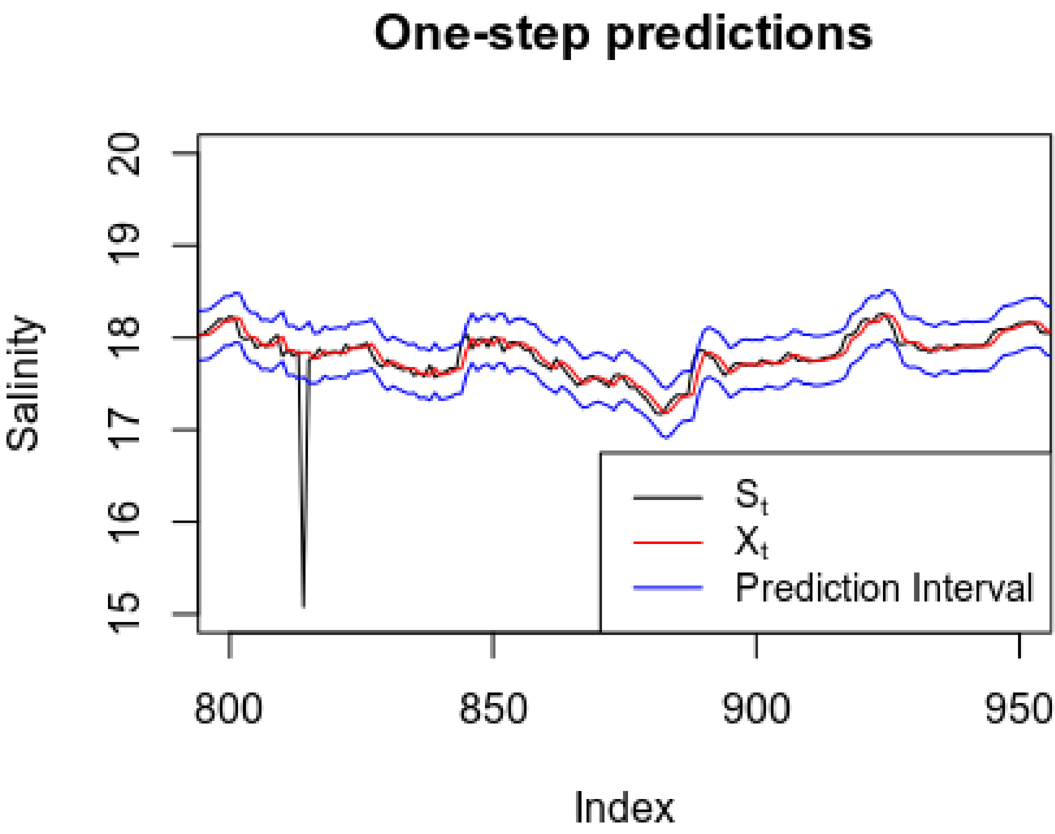

Figure 7

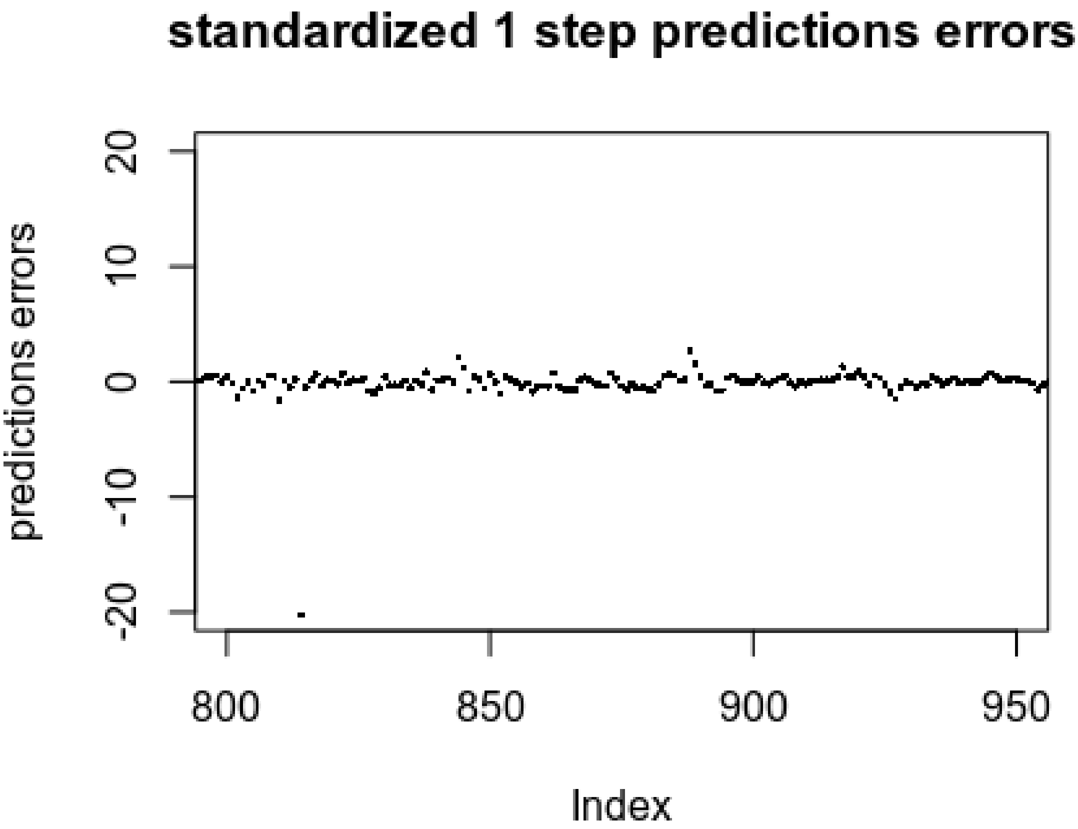

Figure 8

1)
It can be seen that there is no more down-up : the negative point followed by positive point in predictions errors. This confirms that that was due to outliers. 2)
The first five detected outliers have the following indexes:
814, 1203, 1296, 2733, 3692
3)
The number of skipped observation are 13. Note that this number does not include the NAs in data which are 111. Hence, if it is wanted to consider those too, the answer will be 13+111=124
4)
The final state of filter is given by:

$$\hat{X}_{5000} = 20375815 \tag{11}$$

and

$$\Sigma^{xx}_{5000|5000} = 0.003660254 \tag{12}$$

It is the same final state as found in question 4.3) This question is remained without answer, in other words does the outliers not affect the final state of filter ?

5 Question 4.5

6 Question 4.6

The physical models where there are production, consumption of a quantity are numerous. A famous example is the model of temperature in a house. In this section,the results are basically obtained by analogy to that model.
1 and 2)

$$DO_t = DO_{t-1} + \alpha_{prod} I_{t-1} + \alpha_{cons} R_{t-1} + \alpha_{exch}(DO_{sat,t-1} - DO_{t-1}) + \epsilon_{DO,t} \tag{13}$$
$$R_t = R_{t-1} + \epsilon_{R,t} \tag{14}$$

3)
The state space model for dissolved oxygen defined by :
System equation:

$$X_t = \mathbf{A}_t X_{t-1} + \mathbf{B}_t u_{t-1} + e_t \tag{15}$$

where the state is $X_t = \begin{bmatrix} DO_t \\ R_t \end{bmatrix}$,this gives:

$$\mathbf{X}_t = \begin{bmatrix} DO_t \\ R_t \end{bmatrix} = \begin{bmatrix} 1-\alpha_{exch} & \alpha_{cons} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} DO_{t-1} \\ R_{t-1} \end{bmatrix} + \begin{bmatrix} \alpha_{prod} & \alpha_{exch} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} I_{t-1} \\ DO_{sat,t-1} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} e_{DO,t} \\ e_{R,t} \end{bmatrix} \tag{16}$$

Observation equation:

$$Y_t = \mathbf{C}_t X_t + \epsilon_{Y,t} \tag{17}$$
$$\tag{18}$$

$$Y_t = [1,0]\mathbf{X_t} + \epsilon_{Y,t} = [1,0] \begin{bmatrix} DO_t \\ R_t \end{bmatrix} + \epsilon_{Y,t} = DO_t + \epsilon_{Y,t} \tag{19}$$

$$\Sigma_1 = \Sigma_{DO,R} = \begin{bmatrix} \sigma^2_{DO} & 0 \\ 0 & \sigma^2_R \end{bmatrix} \tag{20}$$

$$\Sigma_2 = \Sigma_Y = \sigma^2_Y \tag{21}$$

The assumptions made here are :
$\epsilon_{DO,t}$ is white noise. $\epsilon_{R,t}$ is white noise. They are uncorrelated.
$\epsilon_{Y,t}$ is white noise. It is uncorrelated with $\epsilon_{DO,t}$ and $\epsilon_{R,t}$

Explanation of the procedure by analogy in eq(13):
a) DO_t is the dissolved oxygen at time t, equivalent to the temperature inside the house $T_{i,t}$

b) $\alpha_{prod} I_{t-1}$ this is the production of oxygen from chlorophyll and sunlight i.e photosynthesis. This is equivalent to the production of temperature via radiator in house. As a result I_{t-1} is considered as the first input to the model and α_{prod} denotes the amount of production.

c) $\alpha_{cons} R_{t-1}$ is the consumption of oxygen i.e respiration. α denotes its quantity and should be negative to denote that it is consumption i.e a quantity going out from the system. This is similar to any cooling event in the house, for instance if someone opens the door for seconds. R_{t-1} is modelled as random walk.

d) $\alpha_{exch}(DO_{sat,t-1} - DO_{t-1})$ is the exchange of oxygen with the atmosphere so that the dissolved oxygen concentration approaches the saturation concentration. $(DO_{sat,t-1}$ is the second input. This is similar to $\alpha_{exch}(T_{atm,t-1} - T_{t-1})$ where the atmosphere temperature is the second input and denoted $T_{atm,t-1}$ equivalent to $(DO_{sat,t-1}$ here.

4)
It is believed that the answer to this question is probably obtained when coding this question. One should verify the assumption made above abour errors . Sadly and due to lack of time, I did not do it.

7 Code

```

rm(list=ls())

library(data.table)
library("dlm")
library("FKF")
library(scales)
library("plot3D")
library(plotly)
library(numDeriv)
library(MASS)
library(mvtnorm)
require(graphics)

#Question1
    ##########################################################################################
```

```
data = read.csv(file = "/home/ghassen97/Desktop/S8/time series analysis/
   assignment/assignment 4/A4_Kulhuse.csv", fill = TRUE, header = TRUE)

S = data$Sal
DO = data$ODO
DT = data$DateTime
i = 1:length(S) # Time

plot(S, type = 'l', main = expression(S[t]), ylab='salinity')
points(c(1296, 4609),c(0.43,1.32), col = 'blue', pch = 10)
plot(DO, type = 'l', main = expression(DO[t]), col = 'black',ylab='dissolved
    oxygen')

points(c(1422), c(4.53), col = 'blue', type = 'p')
points(c(4609, 1296),c(12.60,     11.26), col = 'blue', pch = 10)

#Question2 see report
   ###############################################################################

#Question3
   ###############################################################################

Kalman_filter = function(obs_condition, outlier_condition) {
  S = data$Sal # S_t
  N = length(S)
  X = vector(length = N) # State of salinity (state vector/number)
  X_pred = vector(length = N) # One-step predictions of X
  S_xx = vector(length = N) # This is Conditional covariance of X given all
     previous S
  S_yy = vector(length = N) # This is Variance of S given all previous S
  S_xx_pred = vector(length = N) #  one step prediction variance
  S_yy_pred = vector(length = N) #  one step prediction variance
  K    = vector(length = N) # Kalman gain
  outlier_ind    = vector(length = N) # Outlier index
  S_t  = vector(length = N)
  error1  = vector(length = N)
  i = 1:N # Time
  V_e = 0.01 #system variance
  V_m = 0.005 #observation variance

  # Initial Condition
```

```
X[1] = S[1]
X_pred[1] = S[1]
S_xx[1] = V_e
S_yy[1] = V_e + V_m
S_xx_pred[1] = V_e
S_yy_pred[1] = V_e + V_m

# Kalman Filter
for (t in i) {

  error1[t] = S[t] - X_pred[t]

  # fix missing observations (if obs_condition = 1) fix outliers (if
      outlier_condition = 1). This if statement skips the reconstruction
      step
  # if obs_condition == TRUE then fix missing obersvations, if
      outlier_condition == TRUE then skip outliers.
  # There is no reconstruction in this if below
  if((S[t] %in% NA  && obs_condition == TRUE ) | (abs(error1[t]) > 6*sqrt(
      S_xx[t]) && outlier_condition == TRUE)){
    outlier_ind[t] = t
    X[t+1] = X[t]
    X_pred[t+1] = X[t+1]
    S_xx[t+1] = S_xx[t] + V_e
    S_yy[t+1] = S_xx[t+1] + V_m
    S_xx_pred[t+1] = S_xx[t]
    S_yy_pred[t+1] = S_xx[t+1]
  }else{


    K[t] = S_xx[t]/S_yy[t] #Kalman gain

    #Reconstruction , updating
    X[t] = X_pred[t] + K[t]*(S[t] - X_pred[t]) # X_t|t
    S_xx[t] = S_xx[t] - K[t]*S_xx[t]

    #Prediction
    X[t+1] = X[t]
    X_pred[t+1] = X[t+1]
    S_xx[t+1] = S_xx[t] + V_e
    S_yy[t+1] = S_xx[t+1] + V_m
    S_xx_pred[t+1] = S_xx[t]
    S_yy_pred[t+1] = S_xx[t+1]
  }
```

```
  }

  CI_upper = X_pred + 1.96*sqrt((S_yy)) # 95% confidence interval upper
  CI_lower = X_pred - 1.96*sqrt((S_yy)) # 95% Confidence interval lower

  outlier_ind = outlier_ind[!is.na(S)]
  outlier_ind = outlier_ind[outlier_ind != 0]
  # with fkf implementation, check if Kalman_filter is ok
  mod = fkf(a0 = c(S[1]), P0 = matrix(V_e), dt = matrix(0), ct = matrix(0),
     Tt = matrix(1), Zt = matrix(1),
            HHt = matrix(V_e), GGt = matrix(V_m), yt = matrix(t(S), nrow =
               1, ncol = length(S)), check.input = TRUE)
  check_fkf_1 = sum(X_pred - mod$at) #check_fkf_1=5,329071e-15
  #print(check_fkf_1)
  check_fkf_2 = sum(X[i] - mod$att)#check_fkf_2=5,329071e-15
  print(check_fkf_2)
  error2 = S - X_pred[i]

  # Return Values
  newlist = list("X" = X, "X_pred" = X_pred, "S_xx" = S_xx, "S_yy" = S_yy,
                 "CI_upper" = CI_upper, "CI_lower" = CI_lower, "fkf" = mod,
                    "check_fkf_1" = check_fkf_1,
                 "check_fkf_2" = check_fkf_2, "S_xx_pred" = S_xx_pred, "
                    S_yy_pred" = S_yy_pred,
                 "error1" = error1, "error2" = error2, 'outlier_ind' =
                    outlier_ind, "K" = K)
  return(newlist)

}
#1)######################################################
K_filter_1 = Kalman_filter(TRUE, FALSE)
plot(S, type = 'l', col = 'black', main = "One-step predictions", ylab = "
   Salinity")
lines(K_filter_1$X_pred, type = 'l', col = 'red')
lines(K_filter_1$CI_upper, type = 'l', col = 'blue')
lines(K_filter_1$CI_lower, type = 'l', col = 'blue')
legend(1500, 10, legend = c(expression(S[t]), expression(X[t]), "Prediction
   Interval"), col = c("black", "red", "blue"), lty=1:1, cex = 0.9)
#2)################

i = 1:length(S)

S_pred_error = (S - K_filter_1$X_pred[i])/sqrt(K_filter_1$S_yy[i])
```

```
plot(S_pred_error, type = 'p', ylab = 'predictions errors', main = "standardized 1 step predictions errors",pch=16,cex=0.3)

#3)########################

K_filter_1 = Kalman_filter(TRUE, FALSE)

plot(S, type = 'l', col = 'black', main = "One-step predictions", ylab = "Salinity",xlim = c(800,950), ylim = c(15,20))
lines(K_filter_1$X_pred, type = 'l', col = 'red')
lines(K_filter_1$CI_upper, type = 'l', col = 'blue')
lines(K_filter_1$CI_lower, type = 'l', col = 'blue')
legend("bottomright", legend = c(expression(S[t]), expression(X[t]), "Prediction Interval"), col = c("black", "red", "blue"), lty=1:1, cex = 0.9)


S_pred_error = (S - K_filter_1$X_pred[i])/sqrt(K_filter_1$S_yy[i])
plot(S_pred_error, type = 'p', col = 'black', xlim = c(800,950), ylim = c(-20,20), ylab = 'predictions errors', main = "standardized 1 step predictions errors",pch=16,cex=0.4)

#4)#######################
#using function to get X
K_filter_1$X[5000]
K_filter_1$S_xx[5000]

# using library FKF
K_filter_1$fkf$att[5000]  #same values
K_filter_1$fkf$Ptt[5000]

#Question4 ####################################################################################

#1)################
K_filter_no_outlier = Kalman_filter(TRUE, TRUE)

plot(S, type = 'l', col = 'black', main = "One-step predictions", ylab = "Salinity",xlim = c(800,950), ylim = c(15,20))
lines(K_filter_no_outlier$X_pred, type = 'l', col = 'red')
lines(K_filter_no_outlier$CI_upper, type = 'l', col = 'blue')
lines(K_filter_no_outlier$CI_lower, type = 'l', col = 'blue')
legend("bottomright", legend = c(expression(S[t]), expression(X[t]), "Prediction Interval"), col = c("black", "red", "blue"), lty=1:1, cex =
```

```
    0.9)

S_pred_error = (S - K_filter_no_outlier$X_pred[i])/sqrt(K_filter_no_outlier$
    S_yy[i])
plot(S_pred_error, type = 'p', col = 'black', xlim = c(800,950), ylim = c
    (-20,20), ylab = 'predictions errors', main = "standardized 1 step
    predictions errors",pch=16,cex=0.4)

#2)################
K_filter_no_outlier$outlier_ind[1:5] #Outliers from the output of tje kalman
     filter K_filter_no_outlier = Kalman_filter(1,0,1)
#3)################
length(K_filter_no_outlier$outlier_ind) #Number of skipped observations
    excluding the missing observations (NA)
#4)###############
K_filter_no_outlier$X[5000]
K_filter_no_outlier$S_xx[5000]

#Question5
    ####################################################################################


#Question6
    ####################################################################################

#see report
```

List of Figures