Technical University of Denmark

Time Series Analysis

02417

Assignment 4

Authors: Jiayi Han (s164229)

December 3, 2019

Contents

1 Presenting data

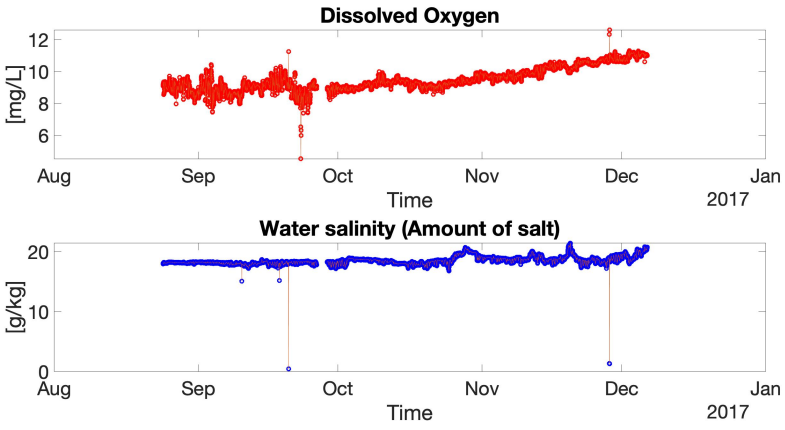

Figure 1: Displaying the given data over time.

Comment: The data is presented in figure 1. The dissolved oxygen and water salinity is the main focus in this report. The concentration of dissolved oxygen is oscillating around 8 to 10 $[mg/L]$. The water salinity is very stable and could be considered as a constant value throughout the observation time, where it is obvious there are three outliers. Besides the outliers, there is discontinuity in the plot, meaning there's some missing data.

2 Random walk state-space model

$$X_t = A \cdot X_{t-1} + B \cdot u_{t_1} + \epsilon_{1,t-1} \tag{1}$$

And the observation equation is given by:

$$Y_t = C \cdot X_t + \epsilon_{2,t-1} \tag{2}$$

Where at this case we observe Y_t which is the measured data. and X_t is what we want to measure, which is the whole inlet (Roskilde fjord). Since we do only have concentration we want to estimate and can measure we have:

$$C = [1] \tag{3}$$

And we assume that there's no input into the system- It does not make sense if there's someone adding anything to contribute to further increase in concentration:

$$B \cdot u_{t-1} = [0] \tag{4}$$

And based on the data, it seems the salt concentration is constant, which leads to

$$A = [1] \tag{5}$$

3 Pure Kalman filter

Kalman filter is used for finding the best prediction of an indirect measurement. For example, it is not possible to correctly measure the whole lake and determine the salt concentration. Instead by taking the one measurement from the sensor and estimate concentration for the whole lake.

The state variance:

$$\Sigma_1 = 0.01 \tag{6}$$

The observation variance:

$$\Sigma_2 = 0.005 \tag{7}$$

The estimate for the next time stamp will have an increased variance. As the variance of the measurement is quite small, this means that the correction term is weighted heavier to the y_t. The Kalman gain determines how heavily the prior state estimate and measurements influence in the estimation of $\hat{x}_t$. If measurement error (variance) is small, then we trust the measurement more.
Initialization Before running the data through Kalman filter there are some initial values that need to be defined first:

$$X_{1|0} = E[X_1] = \mu_0 = 18.0300 \tag{8}$$

$$\Sigma_{1|0}^{x|1} = V[X_1] = V_0 = \Sigma_1 = 0.01 \tag{9}$$

$$\Sigma_{1|0}^{\overline{yy}} = C\Sigma_{1|0}^{xx}C^T + \Sigma_2 = 0.015 \tag{10}$$

The rest of the algorithm is composed of two steps namely reconstruction and prediction where the one-step prediction is performed. When there's missing values, parts of reconstruction is skipped (marked with red). Thereby the variance will not be reduced at time points where data is NaN.

$$\boldsymbol{K}_t = \Sigma_{t|t-1}^{xx}\boldsymbol{C}^T\left(\Sigma_{t|t-1}^{yy}\right)^{-1} \tag{11}$$

$$\widehat{X}_{t|t} = \widehat{X}_{t|t-1} + K_t\left(Y_t - C\widehat{X}_{t|t-1}\right) \tag{12}$$

$$\Sigma_{t|t}^{xx} = \Sigma_{t|t-1}^{xx} - K_t\Sigma_{t|t-1}^{yy}K_t^T \tag{13}$$

In the prediction step (for determining one step prediction):

$$\widehat{X}_{t+1|t} = A\widehat{X}_{t|t} + Bu_t \tag{14}$$

$$\Sigma_{t+1|t}^{xx} = A\Sigma_{t|t}^{xx}A^T + \Sigma_1 \tag{15}$$

$$\Sigma_{t+1|t}^{yy} = C\Sigma_{t+1|t}^{xx}C^T + \Sigma_2 \tag{16}$$

And then getting the observation prediction:

$$\widehat{Y}_{t+1|t} = C\widehat{X}_{t+1|t} = \widehat{X}_{t+1|t} \tag{17}$$

Where C=[1]

The algorithm is implemented in MATLAB and attached in section 7.3.

3.1 Plot the one step predictions along the data. Do include 95% PI.

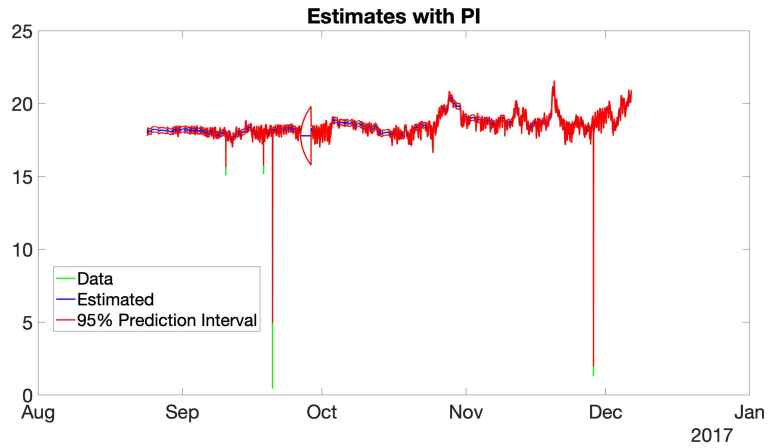

Figure 2: The data, the prediction and the prediction interval.

The variance will normally reduce during the reconstruction phase, but when having no observation (NaN) then the variance is not reduced therefore the triangular shaped prediction interval.

3.2 Plot the standardized one step prediction errors.

The prediction errors are large at the points where the possible outliers are. This is due to outliers has a larger distance to the previous data.

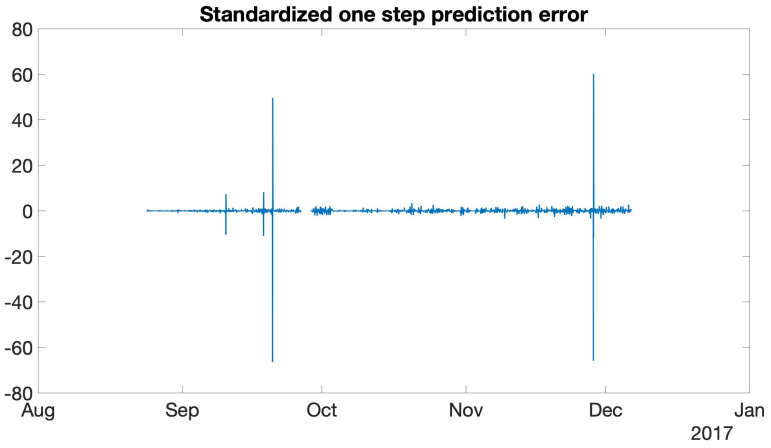

Figure 3: Standardized one step prediction error.

3.3 Repeat the two plots with the same content but zooming into observations 800 to 950.

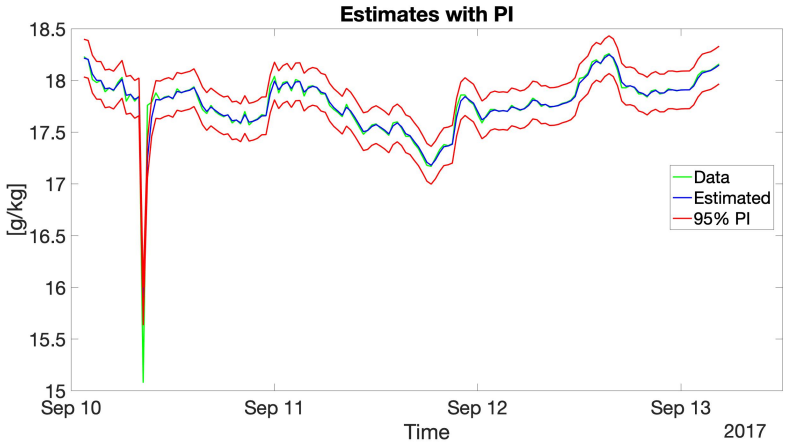

Figure 4: One step predictions zoomed into observation 800 to 950

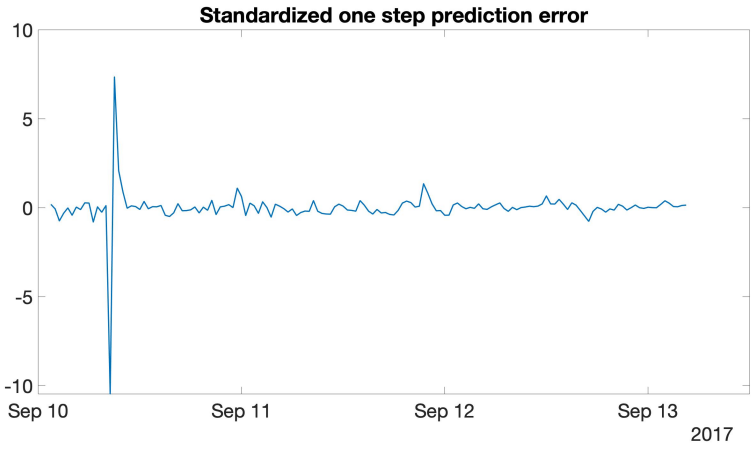

Figure 5: The standardized one step prediction error has some larger fluctuations at the same location where the outlier is located. This is due to the fact that the sudden lowering in the concentration is far away from prediction. Otherwise the error is quite stable along time points.

3.4 Report the values that defines the final state of the filter (at observation 5000).

K_t	$\hat{X}_{t\|t}$	$\Sigma^{xx}_{t\|t}$	$\hat{X}_{t+1\|t}$	$\Sigma^{xx}_{t+1\|t}$	$\Sigma^{yy}_{t+1\|t}$
0.73205	20.7582	0.0036603	20.7582	0.01366	0.01866

Where t=5000

4 Skipping outliers when filtering

4.1 Present the 1-step predictions for index 800 to 950

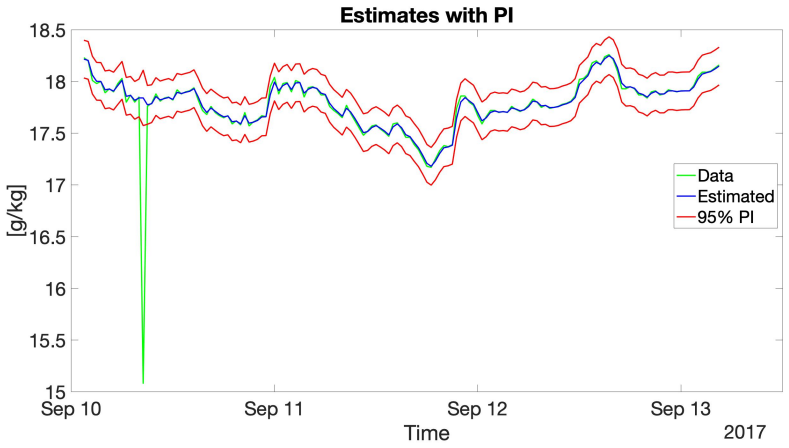

Figure 6: Plot for index 800 to 950. The data measurements, estimates and prediction intervals are shown.

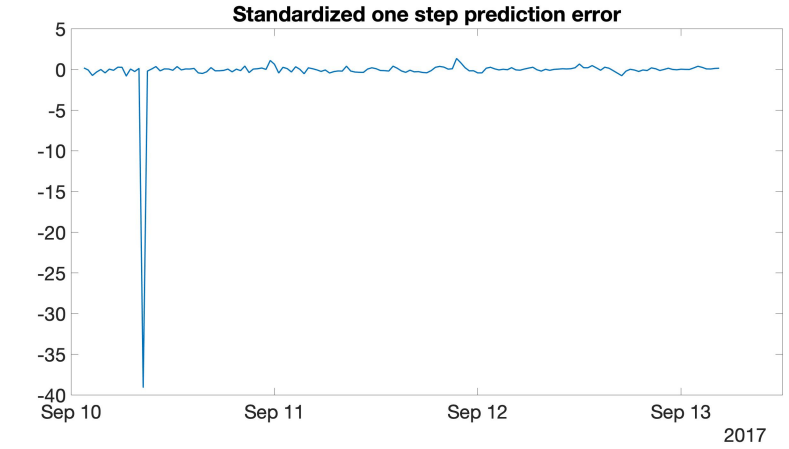

Figure 7: Naturally it is expected that the error has a high magnitude at the location of outlier since it is an outlier and the filter treated the outlier as NaN.

4.2 Report the indexes for the first five detected outliers (observations that are skipped because they are more than six standard deviations away).

In the following table contains all the indexes of the skipped observations, the values Sal(Index) is the salt concentration for which they are considered as outliers based on the criteria given in this exercise.

Index	**Sal(Index)**
814	15.08
1203	15.17
1296	0.43
2733	18.24
3692	17.81
4038	17.95
4578	17.17
4607	1.37
4609	1.32
4686	18.13

This table shows all the values that are skipped

4.3 Do also report the number of observations that are skipped.

In total skipping **10 observations**, the index of which observations is shown in table given above. It is expected that the values that are obvious far away from the overall mean of the data which is around 18 such as 0.43.

4.4 Report the values that defines the final state of the filter (at observation 5000).

K_t	$\hat{X}_{t\|t}$	$\Sigma^{xx}_{t\|t}$	$\hat{X}_{t+1\|t}$	$\Sigma^{xx}_{t+1\|t}$	$\Sigma^{yy}_{t+1\|t}$
0.73205	20.7582	0.0036603	20.7582	0.01366	0.01866

Where t=5000

As it can be seen that the values in this table is identical to the previous ones where the outliers are not removed, this is due to the outliers that are located early in the data set does not influence on the last data outputs.

4.5 Own code

The outlier removal is located at the last part of the code line 42 to 49. The requirement of data to be considered as outlier is if the point is over 6 standard deviation away from prediction

$$Distance = |Y_t - \hat{Y}_{t+1|t}| \tag{18}$$

$$Distance > 6 * \sqrt{\Sigma_{t+1|t}^{yy}} \tag{19}$$

```
function [Skipped XTPlus1T SigmaxxTPlusT SigmayyTPlusT
    XTT Kt SigmaXXTT]=KalmanRemoveOutlier(Sigma1,
   Sigma2, Ydata, initialSigmaXX, A, B, C, Xinitial,U)
% This function works only for univariate, since the
   indexing method does
% not take matrix nor vectors into account.
% This function is done for making predictions with
   procedures decribed in
% Time series analysis F19
% This code is made by s164229
numberofsteps=length(Ydata); % defining number of
   steps
%% Initialization
XTTminus1(1)=Xinitial;
   %The initial value is defined outside of this
   function
SigmaxxTTMinus1=initialSigmaXX;
   %Variance of the first value
SigmayyTTMinus1=C*initialSigmaXX*transpose(C)+Sigma2;
   % calculate the SigmaYY:
% Running for-loop for repeating the Kalman
   reconstruction and prediction
% algorthm:
kk=1;
for k=1:numberofsteps
    %% Reconstruction:
    if isnan(Ydata(k))    % When observation is
       missing then:
    XTT(k)=XTTminus1(k);
    SigmaXXTT(k)=SigmaxxTTMinus1(k);
    else
    %% Reconstruction:
    Kt(k)=SigmaxxTTMinus1(k)*transpose(C)*inv(
       SigmayyTTMinus1(k));  % kalman gain
    XTT(k)=XTTminus1(k)+Kt(k)*(Ydata(k)-C*XTTminus1(k)
       );
    SigmaXXTT(k)=SigmaxxTTMinus1(k)-Kt(k)*
       SigmayyTTMinus1(k)*transpose(Kt(k));
    end
    %% Prediction:
    XTPlus1T(k)=A*XTT(k)+B*U;
    SigmaxxTPlusT(k)=A*SigmaXXTT(k)*transpose(A)+
       Sigma1;
    SigmayyTPlusT(k)=C*SigmaxxTPlusT(k)*transpose(C)+
```

```
        Sigma2;
30
31      %% Re-naming the coefficients for next iteration:
32      XTTminus1(k+1)=XTPlus1T(k);
33      SigmayyTTMinus1(k+1)=SigmayyTPlusT(k);
34      SigmaxxTTMinus1(k+1)=SigmaxxTPlusT(k);
35
36      %% After prediction
37      % Check if the predicted value is far away from
            the true value in ydata
38      % the predicted value
39      % It is known that the Y_hat(t+1|t)=C*Y_hat(t+1|t)
40      % so  by comparing the distance:
41
42      if k<numberofsteps % indexing does not exceed the
            data length
43      distance_pred=abs(Ydata(k+1)-XTPlus1T(k));
44      if distance_pred>6*sqrt(SigmayyTPlusT(k))
45          Ydata(k+1)=NaN; %
46          Skipped(kk)=k+1;
47          kk=kk+1;
48      end
49      end
50  end
51
52  end
```

Listing 1: KalmanRemoveOutlier.m

5 Optimizing the variances

5.1 What is a sensible lower bound for the observation variance?

For the observation variance, if we consider the system as a pure ARMA model then the observation noise $\Sigma_2 = 0$. *(which can be considered as a lower bound fir the observation noise)* Then for the system noise Σ_1 it is different from zero since it is a part of ARMA model.

5.2 Find the ML estimates of the two parameters using the first 800 observations.

By using both optim (R build-in function) and MATLAB fminsearch to estimate the optimal parameter values for Σ_1 and Σ_2:

	Σ_1	$\Sigma 2$
R Optim	1.884394e-03	4.578280e-09
MATLAB fminsearch	0.5563-308	0.

As it can be noticed, Σ_1 is larger than Σ_2 which verifies the expectation about the lower bound for the observation variance. Especially for fminsearch, clearly Σ_1 is different from zero in both cases.

5.3 Filter the data with the optimal parameters.

With the new optimal values from the MLE, the kalman filter has been run again with the same algorithm as in section 4.3.

5.4 Plot as in Q4.3.4 for observations 800 through 950.

As it is shown in the plot in figure 8, as the variances are found to become very small, converging towards zero when optimizing the model with Maximum log likelihood.

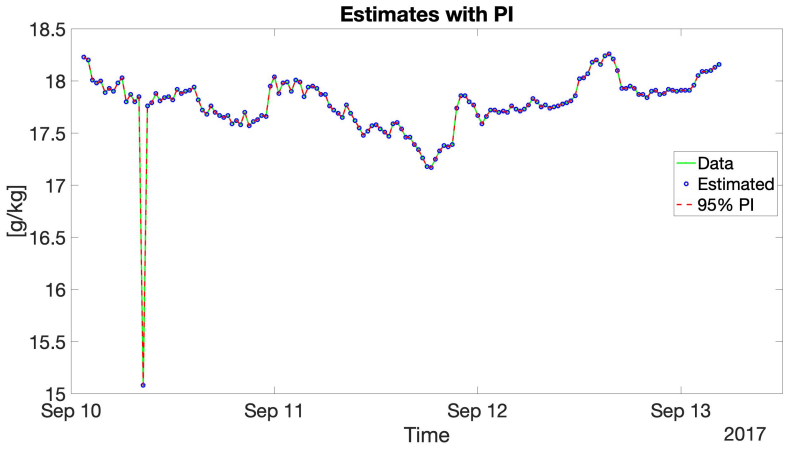

Figure 8: As the variance becomes smaller, the estimates, prediction interval converges toward the data / salt concentration measurements

5.5 Report the values that defines the final state of the filter (at observation 5000)

K_t	$\hat{X}_{t\|t}$	$\Sigma^{xx}_{t\|t}$	$\hat{X}_{t+1\|t}$	$\Sigma^{xx}_{t+1\|t}$	$\Sigma^{yy}_{t+1\|t}$
1	20.77	4.5783e-09	20.77	0.0018844	0.0018844

Where t=5000

As it can be seen here, the variance of the observation and system are very small compared to previous tables.

6 Model for dissolved oxygen

* Not done

7 Appendix

7.1 Main code

```
%% Header
% Assignment 4
% Time series analysis Kalman filter
% This data contains measurements from Roskilde fjord
%% Read the file
%Load the data and present it by making some plots for
    salinity and dissolved oxygen.
```

```
% "Temp","Sal","Depth","pH","Chl","ODOsat","ODO","Battery","DateTime"
% C = textscan(fopen('A4_Kulhuse.csv'), '%f %f %f %f %f %f %f %{yyyy-MM-dd HH:mm:ss}D','HeaderLines',1,'delimiter',',');
% AttributeNames = {'Temp','Sal','Deph','pH','Chl','ODOsat','ODO','Battery','DateTime'};
% Data = cell2struct(C,AttributeNames,2);
%% Load file
Data = readtable('A4_Kulhuse.csv');
t = Data.DateTime;
y = Data.Sal;
y2= Data.ODO;
y(strcmp(y,'NA'))={'NaN'};
y2(strcmp(y2,'NA'))={'NaN'};
yydata=[];
yydataODO=[];
for kk=[1:5000]
    yydata(kk)=str2num(y{kk,1});
    yydataODO(kk)=str2num(y2{kk,1});
end
%%
figure('WindowState','maximized')
subplot(2,1,1)
plot(t,yydata,'o'); hold on
plot(t,yydata)
subplot(2,1,2)
plot(t,yydataODO,'o'); hold on
plot(t,yydataODO)
%% Plot
% Load the data and present it by making some plots for salinity and dissolved oxygen.
figure('WindowState','maximized')
subplot(2,1,1)
plot(t,yydataODO,'ro','LineWidth',2);
 hold on
plot(t,yydataODO)
set(gca, 'fontsize',30);
ylabel('[mg/L]')
xlabel('Time')
title('Dissolved Oxygen')
subplot(2,1,2)
plot(t,yydata,'bo','LineWidth',2);
hold on
plot(t,yydata)
set(gca, 'fontsize',30);
```

```
ylabel('[g/kg]')
xlabel('Time')
title('Water salinity (Amount of salt)')
% salinity is quite stable with obvious outliers
% the dissolved oxygen is oscillating and not as stable as the salinity
% over time, there's also some spread points.
%% filter with kalman filter and fixed state variance
Sigma1=0.01;            % e1
Sigma2=0.005;           % e2
initialSigmaXX=Sigma1;  % slipper for at estimere og optimere en parameter
Input=0;
A=1;
B=0;
C=1;
Xinitial=yydata(1); % the initialization the start
U=0;
[XTPlus1T SigmaxxTPlusT SigmayyTPlusT XTT Kt SigmaXXTT]...
    =HomeMadeKalmanPred(Sigma1, Sigma2, yydata, initialSigmaXX, A, B, C, Xinitial,U);
EExport=[Kt(end) XTT(end) SigmaXXTT(end) XTPlus1T(end) SigmaxxTPlusT(end) SigmayyTPlusT(end)];
columnLabels2 = {'$K_t$','$\hat{X}_{t|t}$','$\Sigma^{xx}_{t|t}$','$\hat{X}_{t+1|t}$','$\Sigma^{xx}_{t+1|t}$','$\Sigma^{yy}_{t+1|t}$'};
matrix2latex(EExport, 'Filt5000.tex', 'columnLabels', columnLabels2)
%% test plot of output
figure('WindowState','Maximized')
plot(t(1:5000),XTT,'ro','LineWidth',3); hold on
plot(t(1:5000),yydata(1:5000),'b-','LineWidth',2)
legend('Predicted','Observed')
title('Test the output')
set(gca, 'fontsize',30);

%% plot of the one step error
figure('WindowState','Maximized')
OneStepPredError=(yydata-XTT)/sqrt(0.005); % according to equation (3.78) in textbook
plot(t,OneStepPredError,'Linewidth',2)
title('Standardized one step prediction error')
set(gca, 'fontsize',30);
%% Plot full estimated value
figure('WindowState','Maximized')
```

```
SigmaYY=SigmaXXTT+Sigma2;
CI_up=XTT+sqrt(SigmaYY)*[1.96];
CI_low=XTT+sqrt(SigmaYY)*[-1.96];
plot(t(1:5000),yydata,'g','LineWidth',2); hold on
plot(t(1:5000),XTT,'b','LineWidth',2);
plot(t,CI_up,'r','LineWidth',2)
plot(t,CI_low,'r','LineWidth',2)
title('Estimates with PI')
legend('Data','Estimated','95% Prediction Interval')
set(gca, 'fontsize',30);
%% zoom and plot
figure('WindowState','Maximized')
OneStepPredError=(yydata-XTT)/sqrt(0.005); % according
    to equation (3.78) in textbook
plot(t(800:950),OneStepPredError(800:950),'LineWidth'
   ,2)
title('Standardized one step prediction error')
set(gca, 'fontsize',30);
%% Zoom Plot full estimated value
figure('WindowState','Maximized')
SigmaYY=SigmaXXTT+Sigma2;
CI_up=XTT+sqrt(SigmaYY)*[1.96];
CI_low=XTT+sqrt(SigmaYY)*[-1.96];
plot(t(800:950),yydata(800:950),'g','LineWidth',2);
   hold on
plot(t(800:950),XTT(800:950),'b','Linewidth',2);
plot(t(800:950),CI_up(800:950),'r','Linewidth',2)
plot(t(800:950),CI_low(800:950),'r','Linewidth',2)
ylabel('[g/kg]')
xlabel('Time')
legend('Data','Estimated','95% PI')
title('Estimates with PI')
set(gca, 'fontsize',30);
%% Running the algorithm 4.4
Sigma1=0.01;            % e1
Sigma2=0.005;           % e2
initialSigmaXX=Sigma1;  % slipper for at estimere og
   optimere en parameter
Input=0;
A=1;
B=0;
C=1;
Xinitial=yydata(1); % the initialization the start
U=0;
[Skipped XTPlus1T SigmaxxTPlusT SigmayyTPlusT XTT Kt
   SigmaXXTT]= ...
```

```
126     KalmanRemoveOutlier(Sigma1, Sigma2, yydata,
           initialSigmaXX, A, B, C, Xinitial,U);
127 %%
128 EExport=[Kt(end) XTT(end) SigmaXXTT(end) XTPlus1T(end)
     SigmaxxTPlusT(end) SigmayyTPlusT(end)];
129 columnLabels2 = {'$K_t$','$\hat{X}_{t|t}$','$\Sigma^{
    xx}_{t|t}$','$\hat{X}_{t+1|t}$','$\Sigma^{xx}_{t+1|
    t}$','$\Sigma^{yy}_{t+1|t}$'};
130 matrix2latex(EExport, 'outlier5000.tex', 'columnLabels
    ', columnLabels2)
131
132 %% test plot of output
133 figure('WindowState','Maximized')
134 plot(t(1:5000),XTT,'ro','LineWidth',3); hold on
135 plot(t(1:5000),yydata(1:5000),'b-','LineWidth',2)
136 plot(t,CI_up,'r','LineWidth',2)
137 plot(t,CI_low,'r','LineWidth',2)
138 legend('Predicted','Observed')
139 title('Test the output')
140 set(gca, 'fontsize',30);
141 %% Zoom Plot full estimated value
142 figure('WindowState','Maximized')
143 SigmaYY=SigmaXXTT+Sigma2;
144 CI_up=XTT+sqrt(SigmaYY)*[1.96];
145 CI_low=XTT+sqrt(SigmaYY)*[-1.96];
146 plot(t(800:950),yydata(800:950),'g','LineWidth',2);
    hold on
147 plot(t(800:950),XTT(800:950),'b','Linewidth',2);
148 plot(t(800:950),CI_up(800:950),'r','Linewidth',2)
149 plot(t(800:950),CI_low(800:950),'r','Linewidth',2)
150 ylabel('[g/kg]')
151 xlabel('Time')
152 legend('Data','Estimated','95% PI')
153 title('Estimates with PI')
154 set(gca, 'fontsize',30);
155
156 %%
157 figure('WindowState','Maximized')
158 OneStepPredError=(yydata-XTT)/sqrt(0.005); % according
     to equation (3.78) in textbook
159 plot(t(800:950),OneStepPredError(800:950),'LineWidth'
    ,2)
160 title('Standardized one step prediction error')
161 set(gca, 'fontsize',30);
162 %% export file of the skipped values:
163 columnLabels = {'Index','Sal(Index)'};
```

```
exportmatrix=[Skipped' yydata(Skipped)'];
matrix2latex(exportmatrix, 'IndexAndValues45.tex', '
    columnLabels', columnLabels)
%% LogLikelihood estimate

initial=log([0.01 0.005]);
%options = optimset('MaxFunEvals',200*length(initial))
    ;
% x = fminsearch(@MyMethod,initial);
% [xx,fval,exitflag,output] = fminsearchbnd(@MyMethod,
    initial,log([0 0]),[Inf Inf]);
options = optimoptions(@fminunc,'Algorithm','quasi-
    newton');
xxx = fminunc(@MyMethod,initial,options);
% second run with different initialization:
%%
columnLab = {'$\Sigma_1$','$\Sigma2$'};
matrix2latex(exp(x), 'parm.tex', 'columnLabels',
    columnLab)
%%
NN=[1.884394e-03 4.578280e-09];
Sigma1=NN(1);           % e1
Sigma2=NN(2);           % e2
initialSigmaXX=Sigma1;  % slipper for at estimere og
    optimere en parameter
Input=0;
A=1;
B=0;
C=1;
Xinitial=yydata(1); % the initialization the start
U=0;
[XTPlus1T SigmaxxTPlusT SigmayyTPlusT XTT Kt SigmaXXTT
    ]...
     =HomeMadeKalmanPred(Sigma1, Sigma2, yydata,
        initialSigmaXX, A, B, C, Xinitial,U);
EExport=[Kt(end) XTT(end) SigmaXXTT(end) XTPlus1T(end)
     SigmaxxTPlusT(end) SigmayyTPlusT(end)];
%%
EExport=[Kt(end) XTT(end) SigmaXXTT(end) XTPlus1T(end)
     SigmaxxTPlusT(end) SigmayyTPlusT(end)];
columnLabels2 = {'$K_t$','$\hat{X}_{t|t}$','$\Sigma^{
    xx}_{t|t}$','$\hat{X}_{t+1|t}$','$\Sigma^{xx}_{t+1|
    t}$','$\Sigma^{yy}_{t+1|t}$'};
matrix2latex(EExport, 'F_filoptim.tex', 'columnLabels'
    , columnLabels2)
%%
```

```
figure('WindowState','Maximized')
SigmaYY=SigmaXXTT+Sigma2;
CI_up=XTT+sqrt(SigmaYY)*[1.96];
CI_low=XTT+sqrt(SigmaYY)*[-1.96];
plot(t(800:950),yydata(800:950),'.-g','LineWidth',2);
    hold on
plot(t(800:950),XTT(800:950),'ob','Linewidth',2);
plot(t(800:950),CI_up(800:950),'--r','Linewidth',2)
plot(t(800:950),CI_low(800:950),'--r','Linewidth',2)
ylabel('[g/kg]')
xlabel('Time')
legend('Data','Estimated','95% PI')
set(gca, 'fontsize',30);
title('Estimates with PI')
%%
y_tilte=(yydata(1:800)-XTT(1:800));
Ri=SigmayyTPlusT;
neps=y_tilte.^2./Ri(1:800);
Run=(-0.5 * sum(neps + log(Ri(1:800)))); % maximize by
    minimize the -ML
%%
function Run=MyMethod(initial)
Data = readtable('A4_Kulhuse.csv');
y = Data.Sal;
y(strcmp(y,'NA'))={'NaN'};
yydata=[];
for kk=[1:800]
    yydata(kk)=str2num(y{kk,1});

end
Sigma1=exp(initial(1));
Sigma2=exp(initial(2));
Input=0;
A=1;
B=0;
C=1;
Xinitial=yydata(1); % the initialization the start
U=0;
% run the filter:
[XTPlus1T, SigmaxxTPlusT, SigmayyTPlusT, XTT, Kt,
   SigmaXXTT]...
    =HomeMadeKalmanPred(Sigma1, Sigma2, yydata, Sigma1
       , A, B, C, Xinitial,U);
% calculate
y_tilte=(yydata(1:800)-XTT(1:800));
Ri=SigmayyTPlusT;
```

```
neps=y_tilte.^2./Ri(1:800);
Run=-(-0.5 * sum(neps + log(Ri(1:800)))+c); % maximize
    by minimize the -ML
end
```

Listing 2: Main code

7.2 R code - exercise 4.5

```
library("FKF")
setwd("~/Desktop/7.semester/Introduction to Time Series Analysis/
    Assignment4")
data = read.csv(file="A4_Kulhuse.csv", header=TRUE, sep=",")
###### Find the optimal Sigma1 and Sigma2 ######
plot(data$DateTime,data$Sal)
A <- matrix(1)
B <- matrix(0)
C <- matrix(1)
Sigma1 <- matrix(0.01)
Sigma2 <- matrix(0.005)
#Initial values
my.obj <- function(par){
  kf2 <- fkf(a0=c(data$Sal[1]), P0 = matrix(c(exp(par[1]))), dt =
      matrix(0), Tt = A, ct=0, Zt = C, HHt = matrix(c(exp(par[1])))
      ,
             GGt = matrix(exp(par[2])), yt = matrix(data$Sal
                 [1:800],nrow=1))
  return(-(kf2$logLik))
}
First = log(c(Sigma1,Sigma2))
#Optimizing
Opt = optim(par = First, fn = my.obj, method = "L-BFGS-B")
(parOpt = exp(Opt$par))
```

files/Assignment4.R

7.3 Kalman filter

```
function [XTPlus1T SigmaxxTPlusT SigmayyTPlusT XTT Kt
    SigmaXXTT]=HomeMadeKalmanPred(Sigma1, Sigma2, Ydata
    , initialSigmaXX, A, B, C, Xinitial,U)
% This function works only for univariate, since the
    indexing method does
% not take matrix nor vectors into account.
% This function is done for making predictions with
    procedures decribed in
% Time series analysis F19
% This code is made by s164229
numberofsteps=length(Ydata); % defining number of
    steps
```

```
%% Initialization
XTTminus1(1)=Xinitial;
SigmaxxTTMinus1=initialSigmaXX;
SigmayyTTMinus1=C*initialSigmaXX*transpose(C)+Sigma2;

for k=1:numberofsteps
    %% Reconstruction:
    if isnan(Ydata(k))
    % When observation is missing then:
    XTT(k)=XTTminus1(k);
    SigmaXXTT(k)=SigmaxxTTMinus1(k);
    else
    %% Reconstruction:
    Kt(k)=SigmaxxTTMinus1(k)*transpose(C)*inv(
        SigmayyTTMinus1(k));  % kalman gain
    XTT(k)=XTTminus1(k)+Kt(k)*(Ydata(k)-C*XTTminus1(k)
        );
    SigmaXXTT(k)=SigmaxxTTMinus1(k)-Kt(k)*
        SigmayyTTMinus1(k)*transpose(Kt(k));
    end
    %% Prediction:
    XTPlus1T(k)=A*XTT(k)+B*U;
    SigmaxxTPlusT(k)=A*SigmaXXTT(k)*transpose(A)+
        Sigma1;
    SigmayyTPlusT(k)=C*SigmaxxTPlusT(k)*transpose(C)+
        Sigma2;
    % Re-naming the coefficients:
    XTTminus1(k+1)=XTPlus1T(k);
    SigmayyTTMinus1(k+1)=SigmayyTPlusT(k);
    SigmaxxTTMinus1(k+1)=SigmaxxTPlusT(k);
end

end
```

Listing 3: HomeMadeKalmanPred.m

7.4 Kalman filter with outlier removal algorithm (same as in exercise 4.4)

```
function [Skipped XTPlus1T SigmaxxTPlusT SigmayyTPlusT
    XTT Kt SigmaXXTT]=KalmanRemoveOutlier(Sigma1,
   Sigma2, Ydata, initialSigmaXX, A, B, C, Xinitial,U)
% This function works only for univariate, since the
   indexing method does
% not take matrix nor vectors into account.
```

```
% This function is done for making predictions with procedures decribed in
% Time series analysis F19
% This code is made by s164229
numberofsteps=length(Ydata); % defining number of steps
%% Initialization
XTTminus1(1)=Xinitial;
    %The initial value is defined outside of this function
SigmaxxTTMinus1=initialSigmaXX;
    %Variance of the first value
SigmayyTTMinus1=C*initialSigmaXX*transpose(C)+Sigma2;
    % calculate the SigmaYY:
% Running for-loop for repeating the Kalman reconstruction and prediction
% algorthm:
kk=1;
for k=1:numberofsteps
    %% Reconstruction:
    if isnan(Ydata(k))    % When observation is missing then:
    XTT(k)=XTTminus1(k);
    SigmaXXTT(k)=SigmaxxTTMinus1(k);
    else
    %% Reconstruction:
    Kt(k)=SigmaxxTTMinus1(k)*transpose(C)*inv(SigmayyTTMinus1(k));  % kalman gain
    XTT(k)=XTTminus1(k)+Kt(k)*(Ydata(k)-C*XTTminus1(k));
    SigmaXXTT(k)=SigmaxxTTMinus1(k)-Kt(k)*SigmayyTTMinus1(k)*transpose(Kt(k));
    end
    %% Prediction:
    XTPlus1T(k)=A*XTT(k)+B*U;
    SigmaxxTPlusT(k)=A*SigmaXXTT(k)*transpose(A)+Sigma1;
    SigmayyTPlusT(k)=C*SigmaxxTPlusT(k)*transpose(C)+Sigma2;

    %% Re-naming the coefficients for next iteration:
    XTTminus1(k+1)=XTPlus1T(k);
    SigmayyTTMinus1(k+1)=SigmayyTPlusT(k);
    SigmaxxTTMinus1(k+1)=SigmaxxTPlusT(k);

    %% After prediction
```

```
37      % Check if the predicted value is far away from
            the true value in ydata
38      % the predicted value
39      % It is known that the Y_hat(t+1|t)=C*Y_hat(t+1|t)
40      % so  by comparing the distance:
41
42      if k<numberofsteps % indexing does not exceed the
            data length
43      distance_pred=abs(Ydata(k+1)-XTPlus1T(k));
44      if distance_pred>6*sqrt(SigmayyTPlusT(k))
45          Ydata(k+1)=NaN; %
46          Skipped(kk)=k+1;
47          kk=kk+1;
48      end
49      end
50  end
51
52  end
```

Listing 4: KalmanRemoveOutlier.m