Time Series Analysis - Assignment 4

Lauge Thode Hermansen
s204111

Tymoteusz Barcinski
s221937

Vincent Olesen
s184017

May 2023

1 Introduction

This report aims to model and filter water salinity for Roskilde Fjord using state-space models.

The data used in the report was collected from sensors on a bouy located in the water near Kulhuse - in the opening of Roskilde Fjord. The measurements consist of water salinity and dissolved oxygen, which we will focus on, and seven other sensor measurements. The data was collected from 2017-08-24 to 2017-12-06 with a sampling rate of 30 minutes. The data has missing values; notably, there is a missing observation period from 2017-09-26 at 12:00:00 to 2017-09-28 at 16:00:00, but other missing observations exist, notably in September and at the end of November.

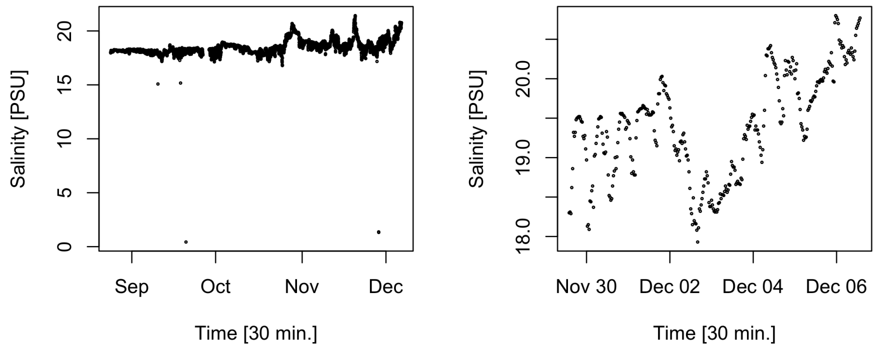

Figure 1: Plot of salinity for the whole period (left) and the last week (right)

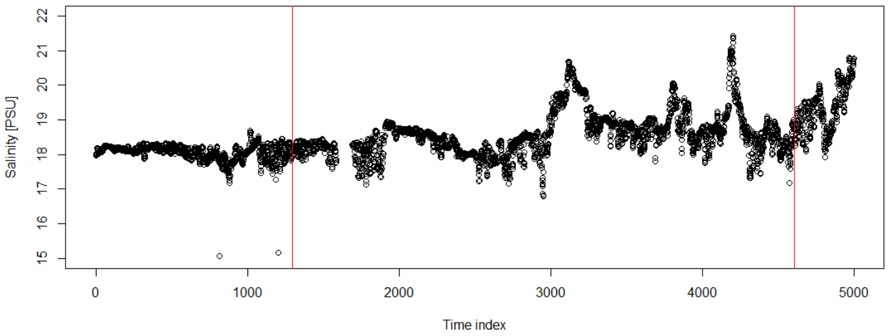

Figure 2: Plot of salinity for the whole period but with the y-range set to [15,22] for better viewing of the variation, The red lines, represent the time index of the observations not shown on the plot, because they are too low.

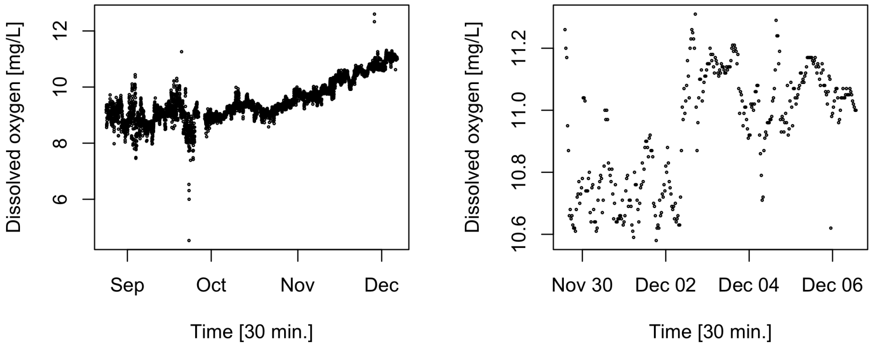

Figure 3: Plot of dissolved oxygen for the whole period (left) and the last week (right)

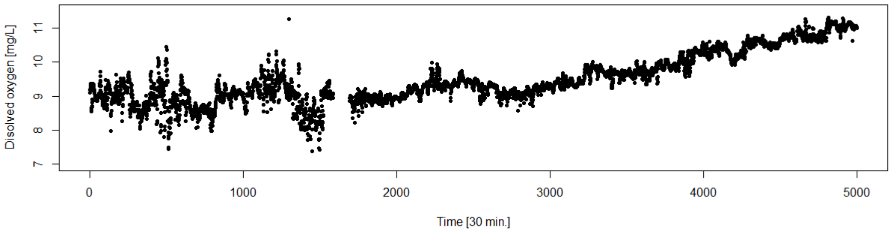

Figure 4: Plot of dissolved oxygen for the whole period

In figs. 1 and 3, we have the entire period (left) and the last week (right) to highlight daily trends.

Considering the entire period, there are outliers, notably in September and at the end of November.

We see a higher variation of the dissolved oxygen before the end of October (the period of the missing values) compared to the rest of the period, possibly due to a sensor replacement. We see a linear trend in the measured dissolved oxygen from the end of October (the period of the missing values).

Interestingly, the salinity has lower variance before the missing period, and a higher variance after the sensor replacement.

2 Random walk state-space model of salinity

We want to model the salinity X_t at time-step t, as a random walk process that is observed at equidistant time points. We take a similar approach as in [1, Ex. 10.6]. Our model for the state is

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

And when we model our observations as just the state with some measurement noise, we get the state-space model

$$\begin{cases} \boldsymbol{X}_t = \boldsymbol{A}\boldsymbol{X}_{t-1} + \boldsymbol{\varepsilon}_{1,t} \quad \text{(No input)} \\ \boldsymbol{Y}_t = \boldsymbol{C}\boldsymbol{X}_t + \boldsymbol{\varepsilon}_{2,t} \\ \boldsymbol{A} = \begin{pmatrix} 1 \end{pmatrix} \\ \boldsymbol{C} = \begin{pmatrix} 1 \end{pmatrix} \\ \boldsymbol{\Sigma}_1 = \begin{pmatrix} \sigma_1^2 \end{pmatrix} \\ \boldsymbol{\Sigma}_2 = \begin{pmatrix} \sigma_2^2 \end{pmatrix} \end{cases} \tag{1}$$

Where σ_1^2 is the system variance and σ_2^2 is the observation variance. In order to compute prediction intervals later on, we assume that the system and measurement errors are normally distributed with mean zero.

Note that the matrices are time-invariant and therefore the subscripts have been omitted. As C has full rank, the system is observable.

3 Pure Kalman filter

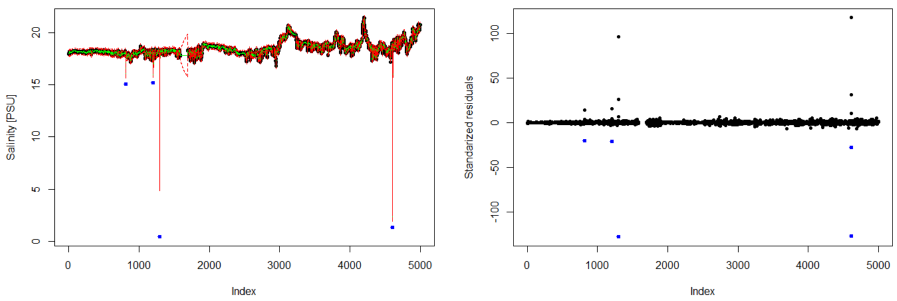

Figure 5: Predictions of the Pure Kalman Filter along with the prediction intervals and standardized one-step predictions errors. The blue points are points we deem to be outliers, and have been marked such that their impact upon the Kalman filter can be more readily be seen.

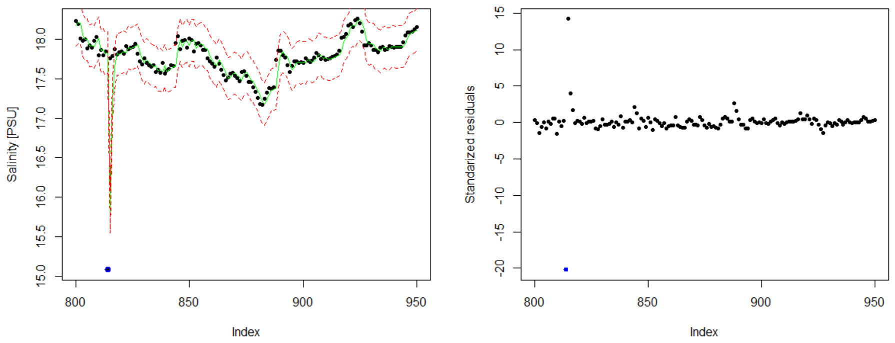

Figure 6: Predictions of the Pure Kalman Filter along with the prediction intervals and standardized one-step predictions errors in the range $[800, 950]$. The blue points are points we deem to be outliers, and have been marked such that their impact upon the Kalman filter can be more readily be seen.

When looking at the plots we see that the predictions follow the observations quite well. The empirical coverage (probability of being inside the confidence interval) is 95.52% which is actually pretty close to the desired 95%.

Considering the standardized errors, we see that the outliers seem to throw off the filter. Usually it takes 3-4 observations until the predictions are reasonable and the variance estimates are well calibrated again.

Upon close examination of the interval from 800 to 950 in fig. 6 we see that the prediction (the green line) has a tendency to be just the previous observation. This can be explained by looking at the prediction step in the Kalman filter, where we simplify the equations, as we don't have any control input.

Predicting the state, X_{t+1}, is based on the reconstructed state.

$$\widehat{X}_{t+1|t} = A\widehat{X}_{t|t} \tag{2}$$

Predicting the observation Y_{t+1} is done by

$$\widehat{Y}_{t+1|t} = C\widehat{X}_{t+1|t} \tag{3}$$

Since in this case, $A = C = 1$, we get

$$\widehat{Y}_{t+1|t} = CA\widehat{X}_{t|t} = \widehat{X}_{t|t} \tag{4}$$

So the prediction of the next observation is actually just the reconstruction of the previous state. Therefore, if our reconstructions are good estimates of the true states, the results in the plots are actually exactly what we would expect.

Empirically, the kalman gain, K, quickly converges to $K \approx 0.7320508$, when there are no missing values, which means that

$$\begin{aligned}\widehat{Y}_{t+1|t} &= \widehat{X}_{t|t}\\ &= \widehat{X}_{t|t-1} + K_t(Y_t - C\widehat{X}_{t|t-1})\\ &= \widehat{X}_{t|t-1} + 0.7320508(Y_t - 1\widehat{X}_{t|t-1})\\ &= 0.2679492\widehat{X}_{t|t-1} + 0.7320508Y_t\end{aligned}$$

So the one step prediction is dominated by the previous observation, which aligns well with what we see in the plot. Since the salinity dynamics are modelled as a random walk, using the Kalman filter for prediction does not say much more about the salinity level than the previous observation. The uncertainty of the prediction may, however, be useful as a gauge for how rapidly the system may change.

State	$\hat{X}_{5000\|5000}$	$\Sigma^{xx}_{5000\|5000}$	$\hat{X}_{5001\|5000}$	$\Sigma^{xx}_{5001\|5000}$	$\Sigma^{yy}_{5001\|5000}$
Value	20.76	$3.660 \cdot 10^{-3}$	20.76	$1.366 \cdot 10^{-2}$	$1.866 \cdot 10^{-2}$

Table 1: Final state of filter at observation 5000, where all observations have been included.

4 Skipping outliers when filtering

Observations are deemed outliers if they are 6 standard deviations away from a prediction[1]. Outliers are treated the same as missing observations. Formally, outliers are defined as

$$\mathcal{Y}_{\text{obs}} = \{Y_t\}_{t=1}^{5000}$$
$$\mathcal{Y}_{\text{outlier}} = \left\{ Y_t \in \mathcal{Y}_{\text{obs}} \;\middle|\; \frac{\left| Y_t - C\widehat{X}_{t|t-1} \right|}{\Sigma_{t|t-1}^{yy}} > 6 \right\}$$

There are 10 outliers and 111 missing observations, which means in total 121 observations are skipped. The missing observations are usually found in clusters. Skipped observations are handled by setting the associated reconstruction to the same time point's prediction, i.e. if the observation at time t was skipped,

$$\widehat{X}_{t|t} = \widehat{X}_{t|t-1}$$
$$\Sigma_{t|t}^{xx} = \Sigma_{t|t-1}^{xx}.$$

t	814	1203	1296	2733	3692	4038	4578	4607	4609	4686
$\mathbf{Y_t}$	15.08	15.17	0.43	18.24	17.81	17.95	17.17	1.37	1.32	18.13

Table 2: All outlier indexes and their values.

As expected, the outliers appear to form at time points where the salinity suddenly changes rapidly compared to the variance of the local past. Afterwards, the filter appears to adapt to this increase in variance, such that subsequent nearby rapidly changing salinity values are not marked as outliers.

However, based on the previous plots with clear outliers, the outlier detection may be overzealous as it appears to have perfect recall but mediocre precision, meaning that it classifies as outliers observations that have high variance due to the dynamics of a system, instead of being outliers. The threshold should therefore perhaps be increased from 6.

[1] In the multivariate case, if the noise is Gaussian, the Mahalanobis distance can maybe be used to decide whether a prediction is an outlier.

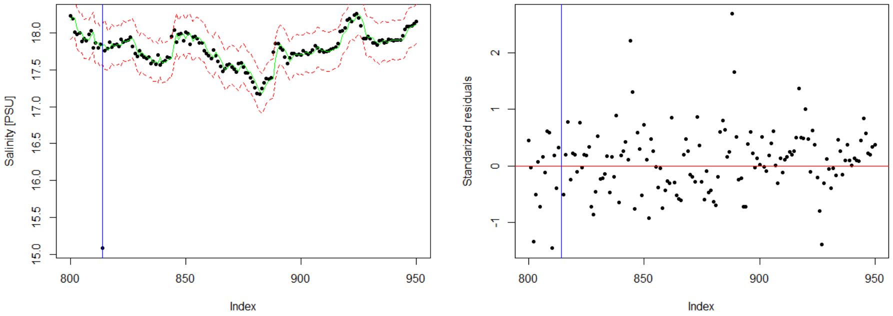

Figure 7: Predictions of the Kalman Filter with the skipping outliers mechanism along with the prediction intervals and standardized one-step prediction errors in the range $[800, 950]$. the blue lines represent the index of the skipped values

Compared to fig. 6, the Kalman Filter is no longer thrown off temporarily by the large spike in variance caused by the outlier. This is evident in the residuals plot. Other than that, things look very similar.

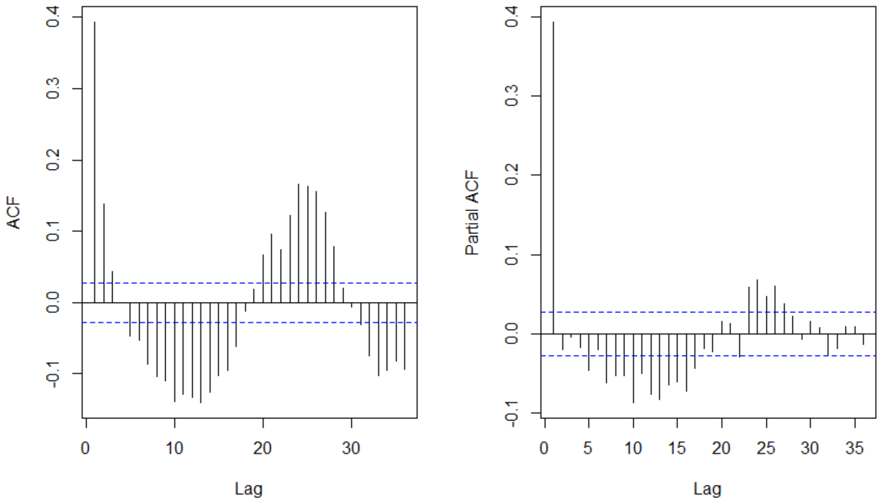

Figure 8: ACF and PACF plots for the standardized one-step prediction residuals for the entire data set.

State	$\hat{X}_{5000\|5000}$	$\Sigma^{xx}_{5000\|5000}$	$\hat{X}_{5001\|5000}$	$\Sigma^{xx}_{5001\|5000}$	$\Sigma^{yy}_{5001\|5000}$
Value	20.76	$3.660 \cdot 10^{-3}$	20.76	$1.366 \cdot 10^{-2}$	$1.866 \cdot 10^{-2}$

Table 3: Final state of filter at observation 5000, where the outliers have been skipped.

Comparing table 3 to table 1, there appears to be no difference when it comes to the final state of the filter. This is likely because the outliers are relatively few and far away from the final state.

Figure 8 presents the ACF and PACF plots for the standardised residuals for the entire dataset. We see that there are some dynamics which are not captured by our model due to its simplicity. Hence, the assumptions of the iid residuals of the model are violated.

5 Optimizing the variances

Variance is a non-negative quantity, so naturally, both system and observation noise variance are lower bounded by zero.

Now, let's see if we can derive some initial estimates of the variances of the system and the observation noise.

We will begin with the observation noise variance. First, notice that salinity is measured in steps of size 0.01. Therefore, let's assume that the observation of salinity is just a rounding to the nearest 0.01, and that this is the *only* variance in the observation. Formally, this means that,

$$\begin{aligned} \boldsymbol{Y}_t &= \text{round}(\boldsymbol{X}_t) \\ &= \boldsymbol{X}_t + \text{round}(\boldsymbol{X}_t) - \boldsymbol{X}_t \\ &= \boldsymbol{X}_t + \varepsilon_{2,t}, \end{aligned}$$

where $\varepsilon_{2,t} = \text{round}(\boldsymbol{X}_t) - \boldsymbol{X}_t$, and round$(\cdot)$ maps a number to the nearest element in $\mathbb{Z}10^{-2}$ (i.e. to nearest 0.01) with round to even for ties.

Since salinity is a physical quantity on a larger scale than atomic, it is fair to assume that its distribution has a continuous density. Now, we realize that from the definition of continuity, we can split the salinity scale into bins (closed intervals), such that the difference between max and min are as small as we like, just by making the bins small enough. This means that we can approximate its density function arbitrarily well by a weighted sum of *uniform* distributions.

Now, we notice that the variation of salinity is many orders of magnitude greater than 0.01, and therefore, we crudely assume that $\boldsymbol{X}_t$ can be approximated well by a weighted sum of uniform distributions over intervals of type

$$\left[10^{-2}\left(n - \frac{1}{2}\right), 10^{-2}\left(n + \frac{1}{2}\right)\right], \qquad n \in \mathbb{Z}$$

This means that, given one observation, $\boldsymbol{Y}_t$, if we assume that the only observation error is introduced by rounding, we know that $\boldsymbol{X}_t \in [Y_t - 0.005, Y_t + 0.005]$, and it is uniformly distributed within this interval. This corresponds to

$$\varepsilon_{2,t} \sim \text{Uniform}\left(\frac{-10^{-2}}{2}, \frac{10^{-2}}{2}\right),$$

which is known to have variance

$$Var(\varepsilon_{2,t}) = \frac{1}{12}\left(10^{-2}\right)^2 \approx 8.333 \cdot 10^{-6}.$$

Thus, under the assumption that the only error in $\boldsymbol{Y}_t$ is the one introduced by rounding, and that the precision is high enough for the piecewise uniform approximation to be good enough; the observation noise variance is approximately $8.333 \cdot 10^{-6}$.

Now, we'll estimate an initial guess of system noise variance. First recall that $\boldsymbol{Y}_t = \boldsymbol{X}_t + \varepsilon_{2,t}$, and since $\boldsymbol{X}_t$ is a random walk, $\boldsymbol{X}_t = \boldsymbol{X}_{t-1} + \varepsilon_{1,t} \Rightarrow \varepsilon_{1,t} = \boldsymbol{X}_t - \boldsymbol{X}_{t-1}$

$$\begin{aligned}\boldsymbol{Y}_t - \boldsymbol{Y}_{t-1} &= \boldsymbol{X}_t - \boldsymbol{X}_{t-1} + \varepsilon_{2,t} - \varepsilon_{2,t-1} \\ &= \varepsilon_{1,t} + \varepsilon_{2,t} - \varepsilon_{2,t-1} \Rightarrow \\ \varepsilon_{1,t} &= \boldsymbol{Y}_t - \boldsymbol{Y}_{t-1} - \varepsilon_{2,t} + \varepsilon_{2,t-1}\end{aligned}$$

Since we just saw that $Var(\varepsilon_{2,t}) \approx \frac{10^{-4}}{12}$, we get

$$\begin{aligned}Var(\varepsilon_{1,t}) &= Var(\boldsymbol{Y}_t - \boldsymbol{Y}_{t-1}) + Var(\varepsilon_{2,t}) + Var(\varepsilon_{2,t-1}) \\ &\approx Var(Y_t - Y_{t-1}) + \frac{10^{-4}}{6}\end{aligned}$$

There are no missing values in the first 800 observations, so we use these to estimate $Var(Y_t - Y_{t-1}) \approx 1.88891 \cdot 10^{-3}$. Noticing that this is multiple orders of magnitude greater than $2Var(\varepsilon_{2,t})$, we get

$$\begin{aligned}Var(\varepsilon_{1,t}) &\approx 1.88891 \cdot 10^{-3} + 8.333 \cdot 10^{-6} \\ &\approx 1.88891 \cdot 10^{-3}\end{aligned}$$

Thus our initial guess of system noise variance is $1.88891 \cdot 10^{-3}$.

Now, the maximum likelihood estimates of the two variances are obtained by minimizing [1, Eq. 10.149] according to the procedure described in [1, Chap. 10.6] with the log-transformed system and observation noise variance as θ. The minimization is performed via the R-function "optim" using the "SANN" method[2] with initial parameters $\log 0.01$ for both variances[3]. The results are found in table 4, and as we see, our initial guesses are actually very, very close to the ML-estimates. This indicates that the the vast majority of the error stems from the model being wrong. This makes good sense, as it is a very

[2] The other methods seemed to push the observation noise variance to zero which is likely wrong because of the rounding.

[3] Our lecturer suggested to use/test with these values.

simple model. Furthermore, it indicates that the sensor is actually very precise, and the error is dominated by the rounding.

As mentioned earlier, we did experience some optimizers setting $\sigma_2 = 0$, which doesn't seem to make sense in light of the above contemplations about the variances. So let's elaborate a bit on what may have happened.

First, note that since our model is a random walk, it can be rewritten in the form:

$$\boldsymbol{Y}_t = \sum_{i=1}^{t} \varepsilon_{1,i} + \varepsilon_{2,t}$$

This means that

$$\begin{aligned} Var(\boldsymbol{Y}_t) &= Var\left(\sum_{i=1}^{t} \varepsilon_{1,i} + \varepsilon_{2,t}\right) \\ &= t\sigma_1 + \sigma_2 \end{aligned}$$

Therefore, since σ_2 is already much, much lower than σ_1, the variance of $\boldsymbol{Y}_t$ is by far dominated by the variance of the state. Thus, the optimizer may overfit to the data and just set it to zero while compensating with σ_1 - even though, clearly this does not give the same model.

Parameter	ML	Guess
$\hat{\sigma}_1$	$1.885 \cdot 10^{-3}$	$1.889 \cdot 10^{-3}$
$\hat{\sigma}_2$	$6.765 \cdot 10^{-6}$	$8.333 \cdot 10^{-6}$

Table 4: ML-estimates and initial guesses of system and observation noise variance based upon first 800 observations.

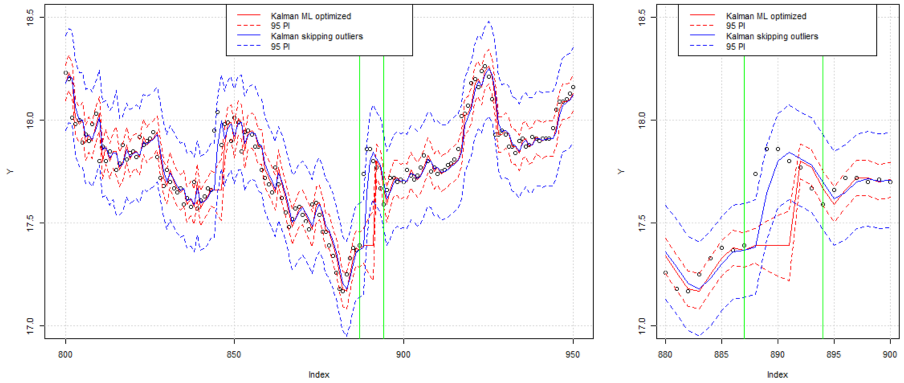

Figure 9: Predictions of the Kalman Filter with the skipping outliers mechanism and ML-estimated parameters, along with the prediction intervals in the range $[800, 950]$. The green lines are there to help orient readers in the zoomed in region.

State	$\hat{X}_{5000\|5000}$	$\Sigma^{xx}_{5000\|5000}$	$\hat{X}_{5001\|5000}$	$\Sigma^{xx}_{5001\|5000}$	$\Sigma^{yy}_{5001\|5000}$
Value	20.77	$6.741 \cdot 10^{-6}$	20.77	$1.891 \cdot 10^{-3}$	$1.898 \cdot 10^{-3}$

Table 5: Final state of filter at observation 5000. The previously found outliers have been skipped.

On table 5 it can be seen that compared to table 3, the variance of the reconstruction and prediction is orders of magnitude lower. The prediction, however, remains mostly the same - only difference is the last few decimals.

6 Model for dissolved oxygen

To build the model for DO_t, we start by modeling it in continuous time with a differential equation. Then, to get a discrete linear model, we use the first-order Taylor expansion, which is equivalent to the Euler's method.

The differential equation is the following:

$$\frac{d}{dt} DO(t) = \text{Production}(t) - \text{Consumption}(t),$$

where obviously Production(t) is the rate of oxygen production at time t and Consumption(t) is the oxygen consumption at time t.

The oxygen production (the primary production) is known to be a linear function of the sun intensity, $I(t)$, and the oxygen exchange with the atmosphere.

Before specifying this linear model, we note that the rate of oxygen exchange with the atmosphere, is known to behave such that $DO(t)$ approaches $DOsat(t)$. We assume that the rate is proportional to the negative signed difference, i.e.

$$\text{Exchange}(t) \propto DOsat(t) - DO(t)$$

Now we use this to get

$$\begin{aligned} \text{Production}(t) &= \alpha I(t) + \beta \text{Exchange}(t) \\ &= \alpha I(t) + \beta\big(DOsat(t) - DO(t)\big) \end{aligned}$$

where we have included the proportion constant in the parameter β. The oxygen consumption is just the respiration, $R(t)$, so now we can write the final differential equation:

$$\begin{aligned} \frac{d}{dt} DO(t) &= \text{Production}(t) - \text{Consumption}(t) \\ &= \underbrace{\alpha I(t) + \beta(DOsat(t) - DO(t))}_{\text{Production}} - \underbrace{R(t)}_{\text{Consumption}} \end{aligned}$$

Now we linearize using Taylor expansion, and discretize by setting $\Delta t = 1$.

$$\begin{aligned} DO(t+\Delta t) &= DO(t) + \Delta t \left(\frac{d}{dt} DO(t)\right) + \xi \Rightarrow \\ DO_{t+1} &= DO_t + \alpha I_t + \beta\big(DOsat_t - DO_t\big) - R_t + \xi \\ &= DO_t(1-\beta) + \alpha I_t + \beta DOsat_t - R_t + \xi, \end{aligned}$$

where ξ is the error term. Note that the use of ξ is a bit sloppy, as it will later just be included in the noise term of the model. We include respiration as a random walk. That is, we have

$$\begin{cases} DO_t = DO_{t-1}(1-\beta) - R_{t-1} + \beta DOsat_{t-1} + \alpha I_{t-1} + \boldsymbol{\varepsilon}_{(1,1),t} \\ R_t = R_{t-1} + \boldsymbol{\varepsilon}_{(1,2),t} \end{cases}$$

Equivalently, we can formulate it as the following state-space model *with* model error, $\boldsymbol{\varepsilon}_{1,t} := [\boldsymbol{\varepsilon}_{(1,1),t}, \boldsymbol{\varepsilon}_{(1,2),t}]^T \in \mathbb{R}^2$ and measurement error $\boldsymbol{\varepsilon}_{2,t} \in \mathbb{R}$:

$$\underbrace{\begin{bmatrix} DO_t \\ R_t \end{bmatrix}}_{\boldsymbol{X}_t} = \underbrace{\begin{bmatrix} 1-\beta & -1 \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} DO_{t-1} \\ R_{t-1} \end{bmatrix}}_{\boldsymbol{X}_{t-1}} + \underbrace{\begin{bmatrix} \beta & \alpha \\ 0 & 0 \end{bmatrix}}_{\boldsymbol{B}} \underbrace{\begin{bmatrix} DOsat_{t-1} \\ I_{t-1} \end{bmatrix}}_{\boldsymbol{u}_{t-1}} + \boldsymbol{\varepsilon}_{1,t}$$

$$\boldsymbol{Y}_t = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{\boldsymbol{C}} \underbrace{\begin{bmatrix} DO_t \\ R_t \end{bmatrix}}_{\boldsymbol{X}_t} + \boldsymbol{\varepsilon}_{2,t}$$

References

[1] Henrik Madsen. *Time Series Analysis*. Oct. 2008. ISBN: 978-1-4200-5967-0. DOI: 10.1201/9781420059687.

7 Code

7.1 A

~~~
---
title: Assignment 4
author: sarphiv
output:
  bookdown::html_document2: default
  bookdown::pdf_document2: default
---
```{r include = FALSE}
# Disable inclusion of code chunks in the output
knitr::opts_chunk$set(echo = FALSE)
library("stats")
library("dplyr")
library("r2r")

set.seed(1337)


# Function to reset the default parameters
par_defaults <- par(no.readonly = TRUE)
par_reset <- function() {
  par(par_defaults)
}
```

```{R}
# Load the data
data <- read.csv("A4_Kulhuse.csv", header = TRUE, sep =
    ",")
oxygen <- data$ODO
salinity <- data$Sal
time <- as.POSIXct(dplyr::pull(data, DateTime))
time_rel <- as.numeric(as.POSIXct(dplyr::pull(data,
    DateTime)))
time_rel <- (time_rel - time_rel[1])
time_delta <- mean(diff(time_rel), na.rm = T)
time_rel <- time_rel / time_delta
~~~

~~~

Length
n <- length(time_rel)
```

#
```{R}
par(mfrow = c(2, 2))
plot(time, oxygen, type='l')
plot(time, salinity, type='l')
plot(oxygen, salinity)
boxplot(oxygen, salinity)
```

#
Example 10.6 may be relevant for further detail.

Assuming stationary dynamics.
$$
X_t = X_{t-1} + \epsilon_t
$$

Must find variance of $\epsilon_t \sim \mathcal{N}(0, \sigma)$.

It is assumed that the salinity is measured directly, but with independent measurement noise.
$$
Y_t = X_t + \eta_t
$$

According to section 10.1, the matrices are given as
$$
A = I\\
B = 0\\
C = I
$$

#
```{R}
kalman <- function(A, C, X.var, Y.var, X0.pred, X0.pred.var, Y, n.ahead=1, B=NULL, u=NULL, outlier.sd=NULL, outlier.idx=NULL, verbose=NULL) {
  # NOTE: Implementation is according to theorem 10.2
  #  Assuming stationary dynamics.
~~~

```r

# Initialize state and variance variables
n <- length(Y) + n.ahead

X.dim <- dim(X0.pred)[1]
Y.dim <- dim(C)[1]

X.recon     <- lapply(1:n, function(x) rep(NA, X.dim))
X.recon.var <- lapply(1:n, function(x) matrix(NA, nrow = X.dim, ncol = X.dim))

X.pred      <- lapply(1:n, function(x) rep(NA, X.dim))
X.pred.var  <- lapply(1:n, function(x) matrix(NA, nrow = X.dim, ncol=X.dim))

Y.pred.var  <- lapply(1:n, function(x) matrix(NA, nrow = Y.dim, ncol = Y.dim))


# Initialize optional arguments
if (is.null(u)) {
  u <- matrix(0, nrow = 1, ncol = n)
}
if (is.null(B)) {
  B <- matrix(0, nrow = X.dim, ncol = dim(u)[1])
}

# Outlier handling
if (is.null(outlier.idx)) {
  outlier.idx <- hashset()
}

outlier.idx.detected <- hashset()


# Helper functions
recon.calc <- function(i) {
  # If missing observation, set reconstruction to prediction
  if (i > length(Y) || any(is.na(Y[[i]]))) {
    X.recon[[i]] <<- X.pred[[i]]
    X.recon.var[[i]] <<- X.pred.var[[i]]

    if ("missing" %in% verbose) {
      cat("Missing:", i, "\n")
    }

    return()
  }


  # Mahalanobis distance for outlier detection
  Y.error <- abs(Y[[i]] - C %*% X.pred[[i]])
  Y.error.sd <- sqrt((t(Y.error) %*% solve(Y.pred.var[[i]]) %*% Y.error) / Y.dim)

  # Outlier and outlier_sd is set
  outlier_sd_detected <- !is.null(outlier.sd) && (Y.error.sd > outlier.sd)
  # Outlier index is marked
  outlier_marked <- outlier.idx[[i]]
  # If outlier, set reconstruction to prediction
  if (outlier_sd_detected || outlier_marked) {
    X.recon[[i]] <<- X.pred[[i]]
    X.recon.var[[i]] <<- X.pred.var[[i]]

    if (outlier_sd_detected) {
      outlier.idx.detected[[i]] <<- TRUE
    }

    if ("outlier" %in% verbose) {
      cat("Outlier:", i, "Sd:", Y.error.sd, "\n")
    }
  }
  # Else, reconstruct
  else {
    K <- X.pred.var[[i]] %*% t(C) %*% solve(Y.pred.var[[i]])

    X.recon[[i]] <<- X.pred[[i]] + K %*% (Y[[i]] - C %*% X.pred[[i]])
    X.recon.var[[i]] <<- X.pred.var[[i]] - K %*% C %*% X.pred.var[[i]]
  }

}

pred.calc <- function(i) {
  X.pred[[i]] <<- A %*% X.recon[[i-1]] + B %*% u[[i-1]]
  X.pred.var[[i]] <<- A %*% X.recon.var[[i-1]] %*% t(A) + X.var
  Y.pred.var[[i]] <<- C %*% X.pred.var[[i]] %*% t(C) +
    Y.var
  }


  # NOTE: Initial predictions set according to theorem 10.2.
  #  Not setting reconstruction directly as in lecture notes,
  #  as it would not incorporate dynamics and the first observation.
  X.pred[[1]] <- X0.pred
  X.pred.var[[1]] <- X0.pred.var

  Y.pred.var[[1]] <- C %*% X.pred.var[[1]] %*% t(C) + Y.var

  # Initial reconstruction
  recon.calc(1)


  # Kalman filter iterative implementation of recursion
  for (i in 2:n) {
    pred.calc(i)
    recon.calc(i)
  }


  # Return reconstructed and predicted values
  return(list(
    X.recon     = X.recon,
    X.recon.var = X.recon.var,
    X.pred      = X.pred,
    X.pred.var  = X.pred.var,
    Y.pred.var  = Y.pred.var,
    outlier.idx = outlier.idx.detected
  ))
}


results.pure <- kalman(
  A=matrix(1, nrow = 1, ncol = 1),
  C=matrix(1, nrow = 1, ncol = 1),
  X.var=matrix(0.01, nrow = 1, ncol = 1),
  Y.var=matrix(0.005, nrow = 1, ncol = 1),
  X0.pred=matrix(salinity[1], nrow = 1, ncol = 1),
  X0.pred.var=matrix(var(salinity, na.rm = T), nrow = 1,
```

~~~~
    ncol = 1),
  Y = as.list(salinity)
)

```

```{R}
par_reset()
plot(time, salinity, type = "l")
lines(time, results.pure$X.pred[-(n+1)], col = "red")
```

```{R}
par_reset()
plot(time[800:950], salinity[800:950], type = "l")
lines(time[800:950], results.pure$X.pred[800:950], col = "red")
```

```{R}
par_reset()
plot(time[800:950], (salinity[800:950] - vapply(results.pure$X.pred[800:950], function(x) x[1, 1], c(1))) / sqrt(vapply(results.pure$X.pred.var[800:950], function(x) x[1, 1], c(1))), type = "p")
```

```{R}
print(results.pure$X.recon[[5000]])
print(results.pure$X.recon.var[[5000]])
print(results.pure$X.pred[[5001]])
print(results.pure$X.pred.var[[5001]])
print(results.pure$Y.pred.var[[5001]])
```

#
```{R}
results.skipped <- kalman(
  A=matrix(1, nrow = 1, ncol = 1),
  C=matrix(1, nrow = 1, ncol = 1),
  X.var=matrix(0.01, nrow = 1, ncol = 1),
  Y.var=matrix(0.005, nrow = 1, ncol = 1),
  X0.pred=matrix(salinity[1], nrow = 1, ncol = 1),
  X0.pred.var=matrix(var(salinity, na.rm = T), nrow = 1, ncol = 1),
~~~~

~~~
  Y=as.list(salinity),
  outlier.sd=6,
  verbose=c("outlier")
)

outlier.idx <- sort(unlist(keys(results.skipped$outlier.idx)))
```

```{R}
par_reset()
plot(time, salinity, type = "l")
for (i in outlier.idx) {
  abline(v = time[i], col = "blue")
}
abline(v=time[is.na(salinity)], col="#00ff002c")
lines(time, results.skipped$X.pred[-(n + 1)], col = "red")
```

```{R}
par_reset()
plot(time[-outlier.idx], salinity[-outlier.idx], type = "l")
for (i in outlier.idx) {
  abline(v = time[i], col = "blue")
}
abline(v=time[is.na(salinity)], col="#00ff002c")
lines(time[-outlier.idx], results.skipped$X.pred[-(n + 1)][-outlier.idx], col = "red")
```

```{R}
par_reset()
plot(time[800:950], salinity[800:950], type = "l")
lines(time[800:950], results.skipped$X.pred[800:950], col = "red")
```

```{R}
par_reset()
plot(time[800:950], (salinity[800:950] - vapply(results.skipped$X.pred[800:950], function(x) x[1, 1], c(1))) / sqrt(vapply(results.skipped$X.pred.var[800:950], function(x) x[1, 1], c(1))), type = "p")
~~~

~~~
```

```{R}
print(results.skipped$X.recon[[5000]])
print(results.skipped$X.recon.var[[5000]])
print(results.skipped$X.pred[[5001]])
print(results.skipped$X.pred.var[[5001]])
print(results.skipped$Y.pred.var[[5001]])
```

#

## Non-parametric bootstrap of observation rounding
```{R}
obs <- runif(1000000)
obs.round <- round(obs, 2)
obs.error <- obs.round - obs

par(mfrow = c(1, 2))
hist(obs, breaks = 100, main = "Histogram of original decimals", xlab = "Decimal")
hist(obs.error, breaks = 100, main = "Histogram of error", xlab = "Error")

print(var(obs.error))
print(0.01^2 / 12)

print(var(salinity[1:800]))
print(var(diff(salinity[1:800])))
```

Seems like the error is uniformly distributed between -0.005 and 0.005.
The lower bound is therefore 0.

A crude estimate of the system noise order of magnitude is also given.

```{R}
log.likelihood <- function(vars) {
  # WARN: Assuming no observations are missing
  A <- matrix(1, nrow = 1, ncol = 1)
  C <- matrix(1, nrow = 1, ncol = 1)
  Y.var <- matrix(exp(vars[2]), nrow = 1, ncol = 1)

   results <- kalman(
~~~

```
    A = A,
    C = C,
    X.var = matrix(exp(vars[1]), nrow = 1, ncol = 1),
    Y.var = Y.var,
    X0.pred = matrix(salinity[1], nrow = 1, ncol = 1),
    X0.pred.var = matrix(var(salinity, na.rm = T), nrow = 1, ncol = 1),
    Y = as.list(salinity[1:800]),
    n.ahead = 0,
    outlier.idx = results.skipped$outlier.idx,
    verbose = c(”outlier”, ”missing”)
  )

  # Implementation according to book
  # R <- lapply(results$X.pred.var, function(x) C %*% x %*% t(C) + Y.var)
  # # NOTE: Using 801 observations, since filtering with 1:800, but log likelihood with 2:801
  # error <- lapply(1:800, function(i) salinity[i+1] - C %*% A %*% results$X.recon[[i]])
  # # NOTE: Swapped sign to maximize via optim, which minimizes
  # return(sum(vapply(1:799, function(i) log(det(R[[i+1]])) + t(error[[i]]) %*% solve(R[[i+1]]) %*% error[[i]], c(1))))


  # Implementation according to slides
  # error <- lapply(1:799, function(i) salinity[i+1] - C %*% A %*% results$X.recon[[i]])
  # NOTE: Swapped sign to maximize via optim, which minimizes
  # return(sum(vapply(2:800, function(i) log(det(results$Y.pred.var[[i]])) + t(error[[i-1]]) %*% solve(results$Y.pred.var[[i]]) %*% error[[i-1]], c(1))))


  # Implementation simplified
  error <- lapply(1:800, function(i) salinity[i] - C %*% results$X.pred[[i]])
  # NOTE: Swapped sign to maximize via optim, which minimizes
  return(sum(vapply(1:800, function(i) log(det(results$Y.pred.var[[i]])) + t(error[[i]]) %*% solve(results$Y.pred.var[[i]]) %*% error[[i]], c(1))))
```

~~~
}

# print(nlminb(c(log(0.01), log(0.01)), log.likelihood))
# print(nlm(log.likelihood, c(log(0.01), log(0.01))))
# print(optim(c(log(0.01), log(0.01)), log.likelihood, control = list(trace = 1, REPORT = 1)))
optim.results <- optim(c(log(0.01), log(0.01)), log.likelihood, method = "SANN", control = list(maxit = 1000, tmax = 10, trace = 1, REPORT = 10))

print(optim.results$par)
print(exp(optim.results$par))
print(optim.results$value)
```


```{R}
results.optimal <- kalman(
  A=matrix(1, nrow = 1, ncol = 1),
  C=matrix(1, nrow = 1, ncol = 1),
  X.var=matrix(exp(optim.results$par[1]), nrow = 1, ncol = 1),
  Y.var=matrix(exp(optim.results$par[2]), nrow = 1, ncol = 1),
  X0.pred=matrix(salinity[1], nrow = 1, ncol = 1),
  X0.pred.var=matrix(var(salinity, na.rm = T), nrow = 1, ncol = 1),
  Y=as.list(salinity),
  outlier.idx = results.skipped$outlier.idx,
  verbose = c("outlier")
)
```

```{R}
par_reset()
plot(time, salinity, type = "l")
lines(time, results.optimal$X.pred[-(n+1)], col = "red")
```

```{R}
par_reset()
plot(time[800:950], salinity[800:950], type = "l")
lines(time[800:950], results.optimal$X.pred[800:950], col = "red")
```

~~~

~~~
```{R}
par_reset()
plot(time[800:950], (salinity[800:950] -
    vapply(results.optimal$X.pred[800:950], function(x)
    x[1, 1], c(1))) /
    sqrt(vapply(results.optimal$X.pred.var[800:950],
    function(x) x[1, 1], c(1))), type = ”p”)
```

```{R}
print(results.optimal$X.recon[[5000]])
print(results.optimal$X.recon.var[[5000]])
print(results.optimal$X.pred[[5001]])
print(results.optimal$X.pred.var[[5001]])
print(results.optimal$Y.pred.var[[5001]])
```


#
```{R}

```
~~~