Bayesian estimation in a bidimensional Ornstein-Uhlenbeck process

Bayesiaanse schatting in een tweedimensionaal Ornstein-Uhlenbeckproces

Jeffrey van der Voort (4709101)

Bachelor thesis
Applied Mathematics
TU Delft

Supervisor: Frank van der Meulen

Date: July 2020

blank page

Abstract

The goal of this thesis is to estimate parameters in a bidimensional Ornstein-Uhlenbeck process, namely a diffusion model which can be found in Favetto and Samson (2010), which considers plasma and interstitium concentrations. We first look at a general linear stochastic differential equation and some properties. Then we simulate possible paths and observations based on the diffusion model, and derive the underlying state space model. We consider state estimation, where we use the Kalman filter and smoother algorithm. It turns out that the smoother outperforms the Kalman filter. Next, we apply and derive the Liu and West filter for state estimation. We see that the performance for the Liu and West filter is close to the Kalman filter. Furthermore, we look at the parameter estimation. We again use and derive the Liu and West filter, but now for parameter estimation. We first apply this filter to a linear AR(1) model, which gives good results. Then we start with estimating one parameter in the diffusion model and finally we estimate all 7 parameters in the model, using priors concentrated around the true value and 10000 particles. For one parameter, we obtain good results. For all 7 parameters, results are satisfactory, with for fully observed data better results than for partially observed data. We also look at the influence of the number of particles: for 5000 particles estimation results are somewhat worse than for 10000 particles. Furthermore, we change some priors such that they are no longer concentrated around the true value. From this, we see that the Liu and West filter does not seem to perform as well for certain choices of priors. Next, we look at state and parameter estimation in a non-linear AR(1) model using the Liu and West filter, which gives rather good results. Finally, we apply the Liu and West filter once more, but now with a real data set. In this case we are able to obtain parameter values that provide a reasonable estimate for the sum of the concentrations, but for the interstitium concentration it leaves much uncertainty.

Contents

1 Introduction

One of the many disadvantages of a tumor is that it can cause angiogenesis (see Jaszai and Schmidt (2019)). Angiogenesis is the excessive growth of arteries due to certain factors that are produced by the tumor. These arteries split from the main blood vessel and if they reach the tumor, it can become much bigger and cause more damage. Certain treatments exist to prevent this from happening, and to study the effectiveness of such treatments, we have to look at the in and out flow of particles through the boundary of the artery. This can be modelled by a so-called Ornstein-Uhlenbeck process (see Favetto and Samson (2010)). The parameters of this diffusion model can tell us something about the in and out flow of particles and thus about the effectiveness of certain treatments. The goal of this report is to estimate the model parameters using Bayesian techniques in which we combine the knowledge of the diffusion model with observations. The observations are obtained by injection of a contrast agent into the body and recording a sequence of medical images from which we can measure the evolution of the contrast agent concentration. The main research question will be: How can we use Bayesian estimation in a bidimensional Ornstein-Uhlenbeck process?

We will first look at a general linear stochastic differential equation and its solution in section (2). Next, we apply this to the diffusion model of Favetto and Samson (2010) and look at a simulated path in section (3). We also simulate the observations in this section and derive the state space model. In section (4) we look at the Kalman filter and how it can be applied in our model. We also introduce the Smoother algorithm and compare it to the Kalman filter. The Liu and West filter for state estimation is described and derived in section (5). In this section we also look at the effective sample size and we apply the Liu and West filter for state estimation to the diffusion model. In section (6) we look at the Liu and West filter for parameter estimation. We derive this filter and then apply it to an AR(1) model. Eventually we apply it to the diffusion model, starting with the estimation of one parameter. Next, we estimate all parameters and shortly look at the influence of the number of particles and the choice of prior distributions. In section (7) we test the Liu and West filter in a non-linear AR(1) model for both state and parameter estimation. We look at parameter estimation in the case of real data in section (8). In section (9) we draw conclusions and give some suggestions for further research. References can be found in section (10), and the majority of the MATLAB code can be found in the Appendix, section (11).

2 A general linear Stochastic Differential Equation

We look at a linear SDE of the following form:

$$dX_t = (BX_t + b(t))dt + \sigma dW_t, \quad X_0 = \eta \tag{1}$$

with X_t the state vector of our system at time t, η the initial state, $b(t)$ a given vector valued function of t, B and σ constant matrices and W_t a multidimensional Wiener process at time t. The unique solution of (1) is given by:

$$X_t = e^{Bt}\left(X_0 + \int_0^t e^{-Bs}b(s)ds + \int_0^t e^{-Bs}\sigma dW_s\right) \tag{2}$$

Furthermore, we have that:

$$X_t \mid X_0 \sim N\left(e^{Bt}\left(X_0 + \int_0^t e^{-Bs}b(s)ds\right)\ ,\ e^{Bt}\left(\int_0^t e^{-Bs}\sigma\sigma^T(e^{-Bs})^T ds\right)(e^{Bt})^T\right) \tag{3}$$

The proof that (2) is the unique solution of equation (1) can be found in Liptser and Shiryaev (2001). They consider a more general SDE, where B and σ can depend on time. We will follow the same steps that they use in this section and we also show the conditional normality.

2.1 Existence of the solution

In order to verify that a solution of (1) exists, we show that expression (2) satisfies equation (1). First, define $Y_t = X_0 + \int_0^t e^{-Bs}b(s)ds + \int_0^t e^{-Bs}\sigma dW_s$. We use Ito's formula (see e.g. Oksendal (2000)) to rewrite dX_t:

$$\begin{aligned}
dX_t = d(e^{Bt}Y_t) &= Be^{Bt}Y_t dt + e^{Bt}dY_t = BX_t dt + e^{Bt}dY_t \\
&= BX_t dt + e^{Bt}d(X_0 + \int_0^t e^{-Bs}b(s)ds + \int_0^t e^{-Bs}\sigma dW_s) \\
&= BX_t dt + e^{Bt}d\left(\int_0^t e^{-Bs}b(s)ds\right) + e^{Bt}d\left(\int_0^t e^{-Bs}\sigma dW_s\right) \\
&= BX_t dt + e^{Bt}(e^{-Bt}b(t)dt) + e^{Bt}(e^{-Bt}\sigma dW_t) \\
&= BX_t dt + b(t)dt + \sigma dW_t = (BX_t + b(t))dt + \sigma dW_t
\end{aligned}$$

So we see that expression (2) is a solution of equation (1). Now we still need to prove that this is the unique solution.

2.2 Uniqueness of the solution

Assume both X_t and $\widetilde{X}_t$, driven by the same Wiener process W_t, satisfy (1). We look at the difference $\Delta_t = X_t - \widetilde{X}_t$. This difference satisfies the following equation:

$$d\Delta_t = dX_t - d\widetilde{X}_t = (BX_t + b(t))dt + \sigma dW_t - (B\widetilde{X}_t + b(t))dt - \sigma dW_t = B(\widetilde{X}_t - X_t)dt = B\Delta_t dt$$

$$d\Delta_t = B\Delta_t dt$$

This equation we can easily solve and its solution is:

$$\Delta_t = e^{Bt}\Delta_0 = e^{Bt}(X_0 - \widetilde{X}_0) = e^{Bt}(\eta - \eta) = \mathbf{0}$$

So we see that $\widetilde{X}_t = X_t$ and thus we have a unique solution.

2.3 Conditionally normal distribution of X_t

We also need to prove claim (3) that X_t, given X_0, is normally distributed. First, define

$$C_t := \int_0^t e^{-Bs} b(s) ds$$

$$Z_t := \int_0^t e^{-Bs} \sigma dW_s$$

Note that C_t is deterministic and Z_t is a random variable. Substituting these definitions in (2), we get:

$$X_t = e^{Bt}(X_0 + C_t + Z_t)$$

Now, given X_0, the distribution of X_t is the same as the distribution of Z_t, only scaled and shifted. Using the definition of an Ito integral (see e.g. Oksendal (2000)), we can write:

$$Z_t = \int_0^t e^{-Bs} \sigma dW_s = \lim_{\Delta t_i \to 0} \sum_{i=1}^{n} e^{-Bt_{i-1}} \sigma (W_{t_i} - W_{t_{i-1}})$$

with $\Delta t_i = t_i - t_{i-1}$, $t_0 = 0$ and $t_n = t$. Brownian motion has normal increments (see for example Klebaner (2005)), so we have that $W_{t_i} - W_{t_{i-1}}$ is normally distributed. So Z_t is actually the (limit of) the sum of normally distributed random variables, which means Z_t itself is also normally distributed. For the expectation of Z_t we have:

$$\mathrm{E}(Z_t) = \mathrm{E}\left(\lim_{\Delta t_i \to 0} \sum_{i=1}^{n} e^{-Bt_{i-1}} \sigma (W_{t_i} - W_{t_{i-1}}) \right)$$

$$= \lim_{\Delta t_i \to 0} \sum_{i=1}^{n} e^{-Bt_{i-1}} \sigma \mathrm{E}(W_{t_i} - W_{t_{i-1}}) = \mathbf{0}$$

For the expectation of X_t we now have:

$$\mathrm{E}(X_t \mid X_0) = \mathrm{E}(e^{Bt}(X_0 + C_t + Z_t) \mid X_0) = e^{Bt}(X_0 + C_t) + e^{Bt}\mathrm{E}(Z_t \mid X_0)$$

$$= e^{Bt}(X_0 + C_t) + e^{Bt}\mathrm{E}(Z_t) = e^{Bt}(X_0 + C_t)$$

where we used that C_t and X_0 (since X_0 is given) are deterministic. Next, we look at the covariance matrix of Z_t. We have:

$$\mathrm{Cov}(Z_t) = \mathrm{E}(Z_t Z_t^T) - \mathrm{E}(Z_t)\mathrm{E}(Z_t)^T = \mathrm{E}(Z_t Z_t^T)$$

Using Ito's isometry property (see Klebaner (2005)), we can write:

$$\mathrm{Cov}(Z_t) = \mathrm{E}(\int_0^t e^{-Bs} \sigma dW_s (\int_0^t e^{-Bs} \sigma dW_s)^T) = \mathrm{E}(\int_0^t e^{-Bs} \sigma (e^{-Bs} \sigma)^T ds)$$

$$= \int_0^t e^{-Bs} \sigma (e^{-Bs} \sigma)^T ds = \int_0^t e^{-Bs} \sigma \sigma^T (e^{-Bs})^T ds$$

So for the covariance matrix of X_t we find:

$$\mathrm{Cov}(X_t \mid X_0) = \mathrm{Cov}(e^{Bt}(X_0 + C_t + Z_t) \mid X_0) = e^{Bt}\mathrm{Cov}(Z_t)(e^{Bt})^T$$

where we again used that C_t and X_0 are deterministic. So we can now conclude that:

$$X_t \mid X_0 \sim N\left(e^{Bt}\left(X_0 + \int_0^t e^{-Bs} b(s) ds \right) \ , \ e^{Bt} \left(\int_0^t e^{-Bs} \sigma \sigma^T (e^{-Bs})^T ds \right) (e^{Bt})^T \right)$$

3 Diffusion of contrast agent particles

Favetto & Samson (2010) consider the following model:

$$dP_t = (\alpha(t) - (\lambda + \beta)P_t + (\kappa - \lambda)I_t)dt + \sigma_1 dW_{1,t} \tag{4}$$

$$dI_t = (\lambda P_t - (\kappa - \lambda)I_t)dt + \sigma_2 dW_{2,t} \tag{5}$$

This model describes the diffusion of contrast agent particles between 2 compartments: the blood plasma, with concentration P_t, and the interstitial fluid or interstitium, with concentration I_t. Furthermore, we have a known function $\alpha(t)$, which describes the injected contrast agent concentration in the artery, and $W_{1,t}$ and $W_{2,t}$ are two independent Wiener processes on $\mathbb{R}$. The unknown parameters are $\alpha(t), \beta, \lambda, \kappa, \sigma_1$ and σ_2. In order for the parameters to make sense biologically speaking, they have to be positive and $\lambda < \kappa$. An overview of the diffusion model can be seen in Figure 1.

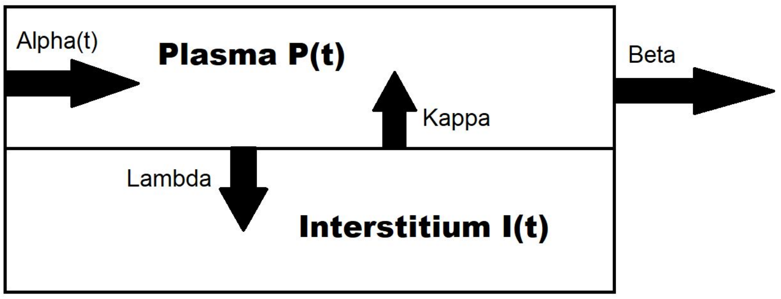


Figure 1: Simplified overview of the diffusion of contrast agent particles between the two compartments with concentrations $P(t)$ and $I(t)$

The diffusion model described by equations (4) and (5) is a particular case of the stochastic differential equation in (1), with:

$$X_t = \begin{bmatrix} P_t \\ I_t \end{bmatrix},\ b(t) = \begin{bmatrix} \alpha(t) \\ 0 \end{bmatrix},\ B = \begin{bmatrix} -\lambda - \beta & \kappa - \lambda \\ \lambda & \lambda - \kappa \end{bmatrix},\ \sigma = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix},\ dW_t = \begin{bmatrix} dW_{1,t} \\ dW_{2,t} \end{bmatrix}$$

3.1 Mean reversion property

We look at the eigenvalues of the matrix B. The characteristic equation of B is:

$$\mu^2 + \mu(\beta + \kappa) + \beta(\kappa - \lambda) = 0$$

with discriminant $d = (\beta + \kappa)^2 - 4\beta(\kappa - \lambda) = (\beta - \kappa)^2 + 4\beta\lambda > 0$. So the eigenvalues are:

$$\mu_1 = \frac{-(\beta + \kappa) - \sqrt{d}}{2},\ \mu_2 = \frac{-(\beta + \kappa) + \sqrt{d}}{2}$$

Clearly, $\mu_1 < 0$. For μ_2 to be negative, we need $-\beta - \kappa + \sqrt{(\beta - \kappa)^2 + 4\beta\lambda} < 0$. This is equivalent to $(\sqrt{(\beta - \kappa)^2 + 4\beta\lambda})^2 < (\beta + \kappa)^2$. If we work this out, we get: $\beta^2 - 2\kappa\beta + \kappa^2 + 4\beta\lambda < \beta^2 + 2\kappa\beta + \kappa^2$. This implies that $4\beta\lambda < 4\beta\kappa$. This only holds if $\lambda < \kappa$, which is indeed the case. So we have two negative eigenvalues. Recall that we have the following SDE:

$$dX_t = (BX_t + b(t))dt + \sigma dW_t$$

Since B has two distinct negative eigenvalues, we can write $B = PDP^{-1}$, with D the matrix of eigenvalues and P the matrix of eigenvectors. If we plug this in, we get:

$$dX_t = (PDP^{-1}X_t + b(t))dt + \sigma dW_t$$

Multiply by P^{-1}:

$$P^{-1}dX_t = (DP^{-1}X_t + P^{-1}b(t))dt + P^{-1}\sigma dW_t$$

Set $Y_t = P^{-1}X_t$, then we get:

$$dY_t = (DY_t + P^{-1}b(t))dt + P^{-1}\sigma dW_t$$

Since B has negative eigenvalues we can write $D = -A$, where A has entries $-\mu_1 > 0$ and $-\mu_2 > 0$ on the diagonal. So we have:

$$dY_t = (\widetilde{b}(t) - AY_t)dt + \widetilde{\sigma}dW_t$$

with $\widetilde{b}(t) = P^{-1}b(t)$ and $\widetilde{\sigma} = P^{-1}\sigma$. We now have a mean-reverting model (see e.g. Wiersema (2008)). This means that if the vector AY_t falls below $\widetilde{b}(t)$, then the term $\widetilde{b}(t) - AY_t$ will have positive entries, and dY_t is more likely to be positive. Conversely, if AY_t is above $\widetilde{b}(t)$, then the term $\widetilde{b}(t) - AY_t$ will have negative entries and dY_t is more likely to be negative. So Y_t will always revert back to some mean value.

3.2 Simulating paths using Euler discretisation

To get an idea of the behaviour of the contrast agent concentration over time, we can use the parameter values below from the simulation study by Favetto and Samson (2010), which they obtain using a step size of $h = 0.04$ and $n = 1000$ grid points.

$$\theta_1 = 0.91,\ \theta_2 = 0.99,\ \theta_3 = 0.2,\ \theta_4 = 0.03,\ \theta_5 = 0.01,\ \theta_6 = 20$$

However, those parameters are not the original ones, so we need to follow their parameter transformation to get back the original model parameters. The MATLAB code to do this can be found in appendix A. We then find the following parameter values:

$$\beta = 2.23,\ \sigma_1 = 2.49,\ \sigma_2 = 0.77,\ \kappa = 0.37,\ \lambda = 0.11,\ \alpha = 31.60$$

Note that Favetto and Samson take α constant, which is somewhat unnatural, but for simplicity we will follow this assumption as well. Now that we know the parameters, we can simulate a possible path for the contrast agent concentration. To do this, we will first discretise equations (4) and (5) using the Euler forward discretisation. A short description of how to do this can be found in the lecture notes by Haugh (2017). Since the parameters were obtained using $h = 0.04$ and $n = 1000$, we will use those values as well. The discretization yields:

$$P_j - P_{j-1} = (\alpha - (\lambda + \beta)P_{j-1} + (\kappa - \lambda)I_{j-1})h + \sigma_1(W_{1,j} - W_{1,j-1}) \tag{6}$$
$$I_j - I_{j-1} = (\lambda P_{j-1} - (\kappa - \lambda)I_{j-1})h + \sigma_2(W_{2,j} - W_{2,j-1}) \tag{7}$$

Using the time-homogeneity property of Brownian motion (see e.g. Klebaner (2005)), we have:

$$W_j - W_{j-1} = W(t_j) - W(t_j - h) \overset{\mathrm{d}}{=} W(h) - W(0)$$

Since Brownian motion has normal increments with mean 0 and variance equal to the step size, we have that:

$$W_j - W_{j-1} = Z_j \sim N(0, h)$$

We can now rewrite equations (6) and (7) as:

$$P_j = P_{j-1} + [\alpha - (\lambda + \beta)P_{j-1} + (\kappa - \lambda)I_{j-1}]h + \sigma_1\sqrt{h}Z_{1,j} \tag{8}$$
$$I_j = I_{j-1} + [\lambda P_{j-1} - (\kappa - \lambda)I_{j-1}]h + \sigma_2\sqrt{h}Z_{2,j} \tag{9}$$

where $Z_{1,j}$ and $Z_{2,j}$ are independent and standard normal distributed. The initial condition is $P(0) = I(0) = 0$, so we set $P_0 = I_0 = 0$. We implement this iteration scheme in MATLAB, see Appendix B, using the parameter values that we found before. Since the $Z_{1,j}$'s and the $Z_{2,j}$'s are random, we get different realisations of the stochastic process. For reproducibility, we set the seed of the default random number generator in MATLAB to 0. The plasma and interstitium concentrations now follow from the iteration scheme and the total concentration S_t is just the sum of the two concentrations. Figure 2 shows a realization of the process obtained by the Euler forward discretisation.

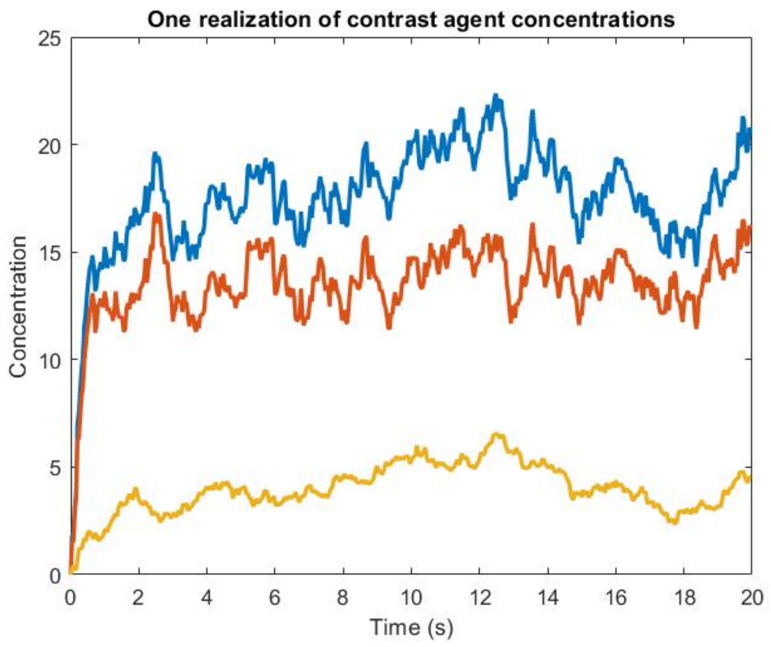


Figure 2: One realization of the contrast agent concentrations over time, displaying the plasma concentration (red line), the interstitium concentration (yellow line) and the sum of the concentrations (blue line).

We see that there is more variation in the plasma concentration P_t than in the interstitial concentration I_t, which is what we expect: σ_1 was taken higher than σ_2. Also, it seems that the contrast agent concentrations first increase, due to the injection of contrast agent. Then each of them fluctuates around some constant value, which means that the concentration has reached some sort of equilibrium. Random collisions of contrast agent particles make it so that random fluctuations around equilibrium occur.

3.3 Simulating observations

For future sections we will need some observations. For now, we simulate the observations, but in section 8 we will look at a real data set. We assume that the observations correspond to times in our grid, so we get observations $y_1, \ldots, y_n$, with $n = 1000$, which are realisations of the observed time series Y_t. We use the realization of Figure 2 as the basis and then manually add some noise to the observations. So we have an observation model of the form:

$$y_i = HX_i + \eta_i \tag{10}$$

with $y_i = \begin{bmatrix} \widetilde{P}_i \\ \widetilde{I}_i \end{bmatrix}$ the noisy observation at time t_i, $H = I$ the observation matrix, $X_i = \begin{bmatrix} P_i \\ I_i \end{bmatrix}$ the state of our system at time t_i and η_i the noise in the observation, which we assume is $N(0, R)$ distributed, with $R = \sigma I$. We will assume standard Gaussian noise, so we choose $\sigma = 1$. The simulated data is displayed in Figure 3.

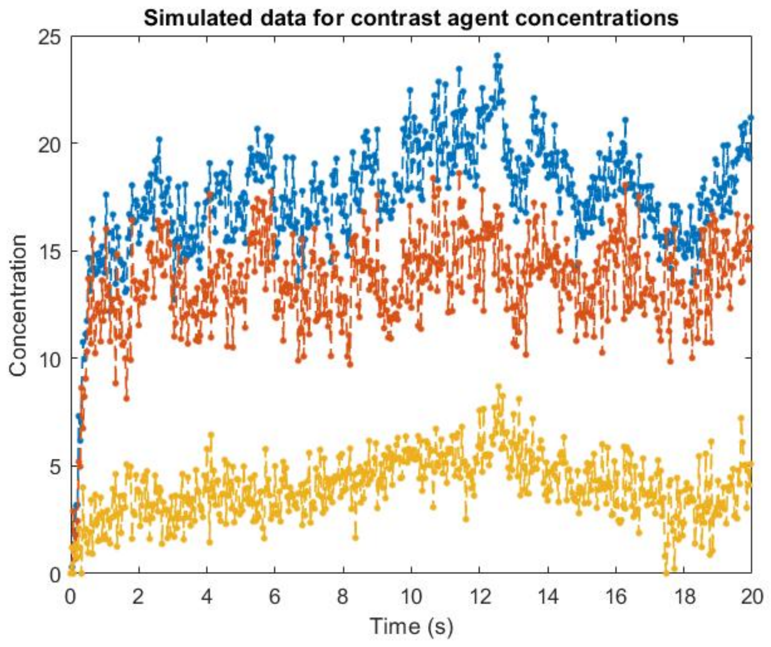


Figure 3: Simulated data for the contrast agent concentrations over time, displaying the plasma concentration (red line), the interstitium concentration (yellow line) and the sum of the concentrations (blue line).

In practice, the contrast agent concentration is measured at discrete times by taking a sequence of medical images of a portion of the vascular system. The intensity in the medical image can give us an indication about the total contrast agent concentration, but we cannot distinguish between the two compartments, so only the sum of the contrast agent concentrations, S_t, can be measured. In this case of partially observed data, we need to change the measurement model to:

$$\widetilde{y}_i = \widetilde{H} X_i + \widetilde{\eta}_i \tag{11}$$

with $\widetilde{y}_i = \widetilde{P}_i + \widetilde{I}_i = \widetilde{S}_i$, $\widetilde{H} = \begin{bmatrix} 1 & 1 \end{bmatrix}$, $X_i = \begin{bmatrix} P_i \\ I_i \end{bmatrix}$ and $\widetilde{\eta}_i \sim N(0, \widetilde{R})$. Note that the data is now one-dimensional, so we have $\widetilde{R} = \sigma^2 = 1$, assuming standard Gaussian noise.

3.4 From SDE to state space model

In order to be able to apply the Kalman filter (see section 4.1), we have to rewrite our SDE as a state space model. First we will do this for the general case of equation (1). We already know the solution at any time t, given by (2). So the solution at time $t + h$ is:

$$X_{t+h} = e^{B(t+h)} \left(X_0 + \int_0^{t+h} e^{-Bs} b(s) ds + \int_0^{t+h} e^{-Bs} \sigma dW_s \right)$$

To get to the state space form, we have to relate X_{t+h} to X_t. We have:

$$X_{t+h} = e^{B(t+h)} \left(X_0 + \int_0^{t+h} e^{-Bs} b(s) ds + \int_0^{t+h} e^{-Bs} \sigma dW_s \right)$$

$$= e^{B(t+h)} \left(X_0 + \int_0^{t} e^{-Bs} b(s) ds + \int_0^{t} e^{-Bs} \sigma dW_s + \int_t^{t+h} e^{-Bs} b(s) ds + \int_t^{t+h} e^{-Bs} \sigma dW_s \right)$$

$$= e^{Bh} e^{Bt} \left(X_0 + \int_0^{t} e^{-Bs} b(s) ds + \int_0^{t} e^{-Bs} \sigma dW_s \right) + e^{B(t+h)} \left(\int_t^{t+h} e^{-Bs} b(s) ds + \int_t^{t+h} e^{-Bs} \sigma dW_s \right)$$

Substituting (2) we find:

$$\begin{aligned} X_{t+h} &= e^{Bh}X_t + e^{B(t+h)}\int_t^{t+h} e^{-Bs}b(s)ds + e^{B(t+h)}\int_t^{t+h} e^{-Bs}\sigma dW_s \\ &= AX_t + C_t + \xi_t \end{aligned}$$

with

$$A = e^{Bh},\ C_t = e^{B(t+h)}\int_t^{t+h} e^{-Bs}b(s)ds,\ \xi_t = e^{B(t+h)}\int_t^{t+h} e^{-Bs}\sigma dW_s$$

We assume an equidistant grid, so we can set $t_{i-1} = t$ and $t_i = t+h$. Then we have:

$$X_i = A_i X_{i-1} + C_i + \xi_i \tag{12}$$

and

$$A_i = e^{B(t_i - t_{i-1})},\ C_i = e^{Bt_i}\int_{t_{i-1}}^{t_i} e^{-Bs}b(s)ds,\ \xi_i = e^{Bt_i}\int_{t_{i-1}}^{t_i} e^{-Bs}\sigma dW_s$$

Equation (12) is called the state equation or process model (Kim and Bang (2018)). In section 2.3 we have seen the following:

$$\int_0^t e^{-Bs}\sigma dW_s \sim N\left(\mathbf{0}, \int_0^t e^{-Bs}\sigma\sigma^T(e^{-Bs})^T ds\right)$$

In a similar way as in section 2.3 we can now derive the distribution of ξ_i. We find:

$$\xi_i \sim N(\mathbf{0}, Q_i)$$

with $Q_i = e^{Bt_i}\left(\int_{t_{i-1}}^{t_i} e^{-Bs}\sigma\sigma^T(e^{-Bs})^T ds\right)(e^{Bt_i})^T$. Now we go back to the diffusion model of Favetto and Samson (2010):

$$b(t) = \begin{bmatrix}\alpha \\ 0\end{bmatrix},\ B = \begin{bmatrix}-\lambda-\beta & \kappa-\lambda \\ \lambda & \lambda-\kappa\end{bmatrix},\ \sigma = \begin{bmatrix}\sigma_1 & 0 \\ 0 & \sigma_2\end{bmatrix}$$

Note that the vector $b(t)$ is actually independent of t. This allows us to explicitly compute the integral C_i:

$$C_i = e^{Bt_i}\int_{t_{i-1}}^{t_i} e^{-Bs}b ds = \left(\int_{t_{i-1}}^{t_i} e^{B(t_i-s)}ds\right)b = -B^{-1}(I - e^{B(t_i-t_{i-1})})b = B^{-1}(e^{Bh}-I)b$$

We can also somewhat simplify Q_i by noting that for a diagonal matrix σ we have $\sigma\sigma^T = \sigma^2$. Now, for our diffusion model we find the following process model:

$$X_i = FX_{i-1} + C + \xi_i \tag{13}$$

where

$$F = e^{Bh}\ ,\ C = B^{-1}(e^{Bh}-I)b\ ,\ \xi_i \sim N(\mathbf{0}, Q_i)$$

and

$$Q_i = e^{Bt_i}\left(\int_{t_{i-1}}^{t_i} e^{-Bs}\sigma^2(e^{-Bs})^T ds\right)(e^{Bt_i})^T$$

Similar to the derivation in section 2.3, we can now also derive the transition distribution:

$$X_i \mid (X_{i-1} = x_{i-1}) \sim N(Fx_{i-1} + C, Q_i)$$

with initial state $X_0 = \begin{bmatrix} P_0 \\ I_0 \end{bmatrix} = \mathbf{0}$. Now that we have the process model, we need to look at the measurement model. We have already derived this for fully observed data in equation (10) and for partially observed data in equation (11). Recall that it is of the following form:

$$Y_i = HX_i + \eta_i$$

with $\eta_i \sim N(0, R)$. Since the noise term is Gaussian, we have that:

$$Y_i \mid (X_i = x_i) \sim N(Hx_i, R)$$

The process model together with the measurement model form the underlying state space model of the SDE in equations (4) and (5).

4 State estimation using Kalman filtering and smoothing

4.1 Kalman filter

The Kalman filter (1960), named after Rudolf Kalman, is nowadays used in many areas. It is used to estimate state variables in a linear dynamical system to combine physical knowledge with data input. We will shortly discuss how this algorithm works, based on descriptions by Kim & Bang (2018) and Miller (2016). The model which is used in the Kalman filter considers observations $y_1, \ldots, y_n$ and underlying hidden states $x_1, \ldots, x_n$ which form a Hidden Markov Model (i.e. the future state only depends on the current state and not on the past). The observations and states are assumed to satisfy the following factorisation:

$$p(y_{1:n}, x_{1:n}) = p(x_1)p(y_1 \mid x_1) \prod_{i=2}^{n} p(x_i \mid y_{i-1})p(y_i \mid x_i)$$

where we use Bayesian notation: $p(y_i \mid x_i) = f_{Y_i|X_i}(y_i \mid x_i)$ is the probability density function $f(.)$ of the random variable $Y_i \mid X_i$ evaluated in $y_i \mid x_i$. The factorisation states that the joint density of the observations and states is the product of the initial density and conditional densities, more precisely:

- $p(x_1) = \phi(x_1; \mu_0, P_0)$ is the initial density, where $\phi(x; \mu, \sigma^2)$ denotes the density of the $N(\mu, \sigma^2)$ distribution evaluated at x, with μ_0 the initial state and P_0 the initial covariance matrix which captures the uncertainty in the initial state.

- $p(x_i \mid x_{i-1}) = \phi(x_i; Fx_{i-1}, Q_i)$ is the transition density, which describes how to get from state $i-1$ to state i using the physical model behind it. Here F is the transition matrix, which is derived from the physical model, and Q_i describes the noise/error that is not captured by the physical model.

- $p(y_i \mid x_i) = \phi(y_i; Hx_i, R)$ is the measurement density, with H the measurement matrix that relates the measurements to the states and R the noise/error in the measurements.

In order to be able to use the Kalman filter, the system must be linear and have normally distributed noise terms. Clearly, equations (4) and (5) are linear. Furthermore, we have seen in section 3.4 that the noise terms for the state space model are Gaussian. Since our initial state is exactly known, $X_0 = \mathbf{0}$, we have that $\mu_0 = \mathbf{0}$ and $P_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$. Next, we will derive the Kalman filter equations.

4.1.1 Derivation of Kalman filter

We assume that we have a measurement model of the form:

$$Y_{t+1} = LX_{t+1} + Z_{t+1}$$

with $Z_{t+1} \sim N(0, \Sigma_0)$. We also assume that the transition distribution is Gaussian, which means that $X_{t+1} \mid (X_t = x_t) \sim N(\mu_t, \Sigma_t)$, where μ_t depends on x_t. Now we look at the vector with components X_{t+1} and Y_{t+1}:

$$\begin{bmatrix} X_{t+1} \\ Y_{t+1} \end{bmatrix} = \begin{bmatrix} X_{t+1} \\ LX_{t+1} + Z_{t+1} \end{bmatrix} = \begin{bmatrix} I \\ L \end{bmatrix} X_{t+1} + \begin{bmatrix} 0 \\ I \end{bmatrix} Z_{t+1}$$

If we condition on X_t, and use the independence of X_t and Z_{t+1}, then by the calculation rules for multivariate normal distributions (see e.g. Bijma et al. (2017)), we obtain:

$$\begin{bmatrix} I \\ L \end{bmatrix} X_{t+1} \mid X_t \sim N\left(\begin{bmatrix} I \\ L \end{bmatrix} \mu_t, \begin{bmatrix} I \\ L \end{bmatrix} \Sigma_t \begin{bmatrix} I \\ L \end{bmatrix}^T \right)$$

and

$$\begin{bmatrix} 0 \\ I \end{bmatrix} Z_{t+1} \sim N\left(0, \begin{bmatrix} 0 \\ I \end{bmatrix} \Sigma_0 \begin{bmatrix} 0 \\ I \end{bmatrix}^T \right)$$

Because of independence, we can just add up the means and the covariances:

$$\begin{bmatrix} X_{t+1} \\ Y_{t+1} \end{bmatrix} \mid X_t \sim N\left(\begin{bmatrix} I \\ L \end{bmatrix} \mu_t, \begin{bmatrix} I \\ L \end{bmatrix} \Sigma_t \begin{bmatrix} I \\ L \end{bmatrix}^T + \begin{bmatrix} 0 \\ I \end{bmatrix} \Sigma_0 \begin{bmatrix} 0 \\ I \end{bmatrix}^T \right) = N\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \right)$$

One of the properties of multivariate normal distributions is that now $X_{t+1} \mid (Y_{t+1} = y_{t+1}, X_t = x_t)$ also has a multivariate normal distribution (see Majumdar and Majumdar (2017)), $N(\mu_{t+1}, \Sigma_{t+1})$, with $\mu_{t+1} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(y_{t+1} - \mu_2)$ and $\Sigma_{t+1} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$. We have to identify $\mu_1, \mu_2, \Sigma_{11}, \Sigma_{12}, \Sigma_{21}$ and Σ_{22}. Clearly, we have $\mu_1 = \mu_t$ and $\mu_2 = L\mu_t$. For the components of Σ we need to do some calculations:

$$\begin{bmatrix} I \\ L \end{bmatrix} \Sigma_t \begin{bmatrix} I \\ L \end{bmatrix}^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ L_{11} & L_{12} \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} \Sigma_{11}^t & \Sigma_{12}^t \\ \Sigma_{21}^t & \Sigma_{22}^t \end{bmatrix} \begin{bmatrix} 1 & 0 & L_{11} & L_{21} \\ 0 & 1 & L_{12} & L_{22} \end{bmatrix}$$

$$= \begin{bmatrix} \Sigma_{11}^t & \Sigma_{12}^t \\ \Sigma_{21}^t & \Sigma_{22}^t \\ L_{11}\Sigma_{11}^t + L_{12}\Sigma_{21}^t & L_{11}\Sigma_{12}^t + L_{12}\Sigma_{22}^t \\ L_{21}\Sigma_{11}^t + L_{22}\Sigma_{21}^t & L_{21}\Sigma_{12}^t + L_{22}\Sigma_{22}^t \end{bmatrix} \begin{bmatrix} 1 & 0 & L_{11} & L_{21} \\ 0 & 1 & L_{12} & L_{22} \end{bmatrix}$$

$$= \begin{bmatrix} \Sigma_{11}^t & \Sigma_{12}^t & L_{11}\Sigma_{11}^t + L_{12}\Sigma_{12}^t & L_{21}\Sigma_{11}^t + L_{22}\Sigma_{12}^t \\ \Sigma_{21}^t & \Sigma_{22}^t & L_{11}\Sigma_{21}^t + L_{12}\Sigma_{22}^t & L_{21}\Sigma_{21}^t + L_{22}\Sigma_{22}^t \\ L_{11}\Sigma_{11}^t + L_{12}\Sigma_{21}^t & L_{11}\Sigma_{12}^t + L_{12}\Sigma_{22}^t & L_{11}(L_{11}\Sigma_{11}^t + L_{12}\Sigma_{21}^t) + L_{12}(L_{11}\Sigma_{12}^t + L_{12}\Sigma_{22}^t) & L_{21}(L_{11}\Sigma_{11}^t + L_{12}\Sigma_{21}^t) + L_{22}(L_{11}\Sigma_{12}^t + L_{12}\Sigma_{22}^t) \\ L_{21}\Sigma_{11}^t + L_{22}\Sigma_{21}^t & L_{21}\Sigma_{12}^t + L_{22}\Sigma_{22}^t & L_{11}(L_{21}\Sigma_{11}^t + L_{22}\Sigma_{21}^t) + L_{12}(L_{21}\Sigma_{12}^t + L_{22}\Sigma_{22}^t) & L_{21}(L_{21}\Sigma_{11}^t + L_{22}\Sigma_{21}^t) + L_{22}(L_{21}\Sigma_{12}^t + L_{22}\Sigma_{22}^t) \end{bmatrix}$$

Since Σ_t is a covariance matrix, it is symmetric. So $\Sigma_{21}^t = \Sigma_{12}^t$. So we can simplify the above as:

$$= \begin{bmatrix} \Sigma_{11}^t & \Sigma_{12}^t & L_{11}\Sigma_{11}^t + L_{12}\Sigma_{12}^t & L_{21}\Sigma_{11}^t + L_{22}\Sigma_{12}^t \\ \Sigma_{12}^t & \Sigma_{22}^t & L_{11}\Sigma_{12}^t + L_{12}\Sigma_{22}^t & L_{21}\Sigma_{12}^t + L_{22}\Sigma_{22}^t \\ L_{11}\Sigma_{11}^t + L_{12}\Sigma_{12}^t & L_{11}\Sigma_{12}^t + L_{12}\Sigma_{22}^t & L_{11}^2\Sigma_{11}^t + 2L_{11}L_{12}\Sigma_{12}^t + L_{12}^2\Sigma_{22}^t & L_{21}(L_{11}\Sigma_{11}^t + L_{12}\Sigma_{12}^t) + L_{22}(L_{11}\Sigma_{12}^t + L_{12}\Sigma_{22}^t) \\ L_{21}\Sigma_{11}^t + L_{22}\Sigma_{12}^t & L_{21}\Sigma_{12}^t + L_{22}\Sigma_{22}^t & L_{11}(L_{21}\Sigma_{11}^t + L_{22}\Sigma_{12}^t) + L_{12}(L_{21}\Sigma_{12}^t + L_{22}\Sigma_{22}^t) & L_{21}^2\Sigma_{11}^t + 2L_{21}L_{22}\Sigma_{12}^t + L_{22}^2\Sigma_{22}^t \end{bmatrix}$$

and

$$\begin{bmatrix} 0 \\ I \end{bmatrix} \Sigma_0 \begin{bmatrix} 0 \\ I \end{bmatrix}^T = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Sigma_{11}^0 & \Sigma_{12}^0 \\ \Sigma_{21}^0 & \Sigma_{22}^0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \Sigma^0_{11} & \Sigma^0_{12} \\ \Sigma^0_{21} & \Sigma^0_{22} \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \Sigma^0_{11} & \Sigma^0_{12} \\ 0 & 0 & \Sigma^0_{21} & \Sigma^0_{22} \end{bmatrix}$$

So we can now identify $\Sigma_{11}, \Sigma_{12}, \Sigma_{21}$ and Σ_{22}:

$$\Sigma_{11} = \Sigma_t$$

$$\Sigma_{12} = \Sigma_t L^T$$

$$\Sigma_{21} = L\Sigma_t$$

$$\Sigma_{22} = L\Sigma_t L^T + \Sigma_0$$

Because of the conditional normality, we get:

$$X_{t+1} \mid (Y_{t+1} = y_{t+1}, X_t = x_t) \sim N(\mu_{t+1}, \Sigma_{t+1})$$

with

$$\mu_{t+1} = \mu_t + \Sigma_t L^T (L\Sigma_t L^T + \Sigma_0)^{-1}(y_{t+1} - L\mu_t)$$
$$\Sigma_{t+1} = \Sigma_t - \Sigma_t L^T (L\Sigma_t L^T + \Sigma_0)^{-1} L\Sigma_t$$

Define $K_t = \Sigma_t L^T (L\Sigma_t L^T + \Sigma_0)^{-1}$, which is the Kalman gain. Then we have:

$$\mu_{t+1} = \mu_t + K_t(y_{t+1} - L\mu_t)$$

$$\Sigma_{t+1} = \Sigma_t - K_t L\Sigma_t = (I - K_t L)\Sigma_t$$

These equations are precisely the Kalman filter equations.

4.1.2 Kalman filter algorithm

We will use the version of the Kalman filter algorithm as described by Kim & Bang (2018).

Algorithm 1: Kalman filter algorithm

```
Input: Observations $y_1, \ldots, y_n$, initial values $\mu_0$, $P_0$ and $F, C$ and $Q$ from the state
  equation and $H$ and $R$ from the observation equation
Output: Mean $\mu$ and covariance matrix $P$ of the filtering distribution
initialization: $\mu = \mu_0$ and $P = P_0$ ;
for $i = 1, \ldots, n$ do
    Prediction step:
    $\mu_i^- = F\mu_{i-1} + C$ ;
    $P_i^- = FP_{i-1}F^T + Q$ ;
    Update step:
    $K_i = P_i^- H^T (R + HP_i^- H^T)^{-1}$ ;
    $\mu_i^+ = \mu_i^- + K_i(y_i - H\mu_i^-)$ ;
    $P_i^+ = (I - K_i H)P_i^-$ ;
end
```

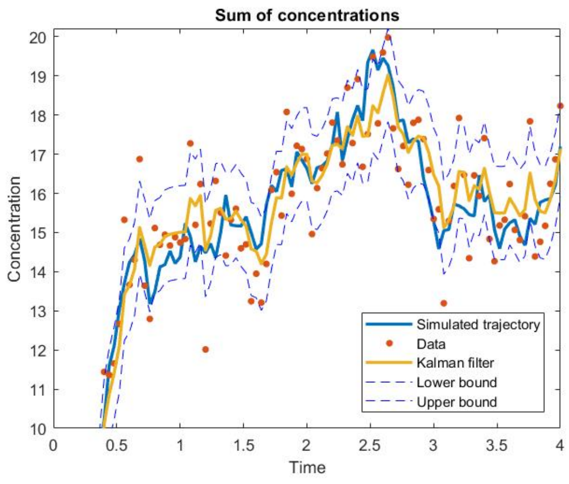
We write the superscript $-$ for the predicted estimate, prior to observing the data, and the superscript $+$ for the updated estimate, after observing the data. Note that the update equations are the same equations we found in the derivation in the previous section. The factor K_i is called the Kalman gain and tells us how much we need to change an estimate now that we have a new measurement. The filtering distribution is $N(\mu_i^+, P_i^+)$ and its mean, μ_i^+, is the filtered estimate for the state variable at time t_i.

4.1.3 A note on the implementation

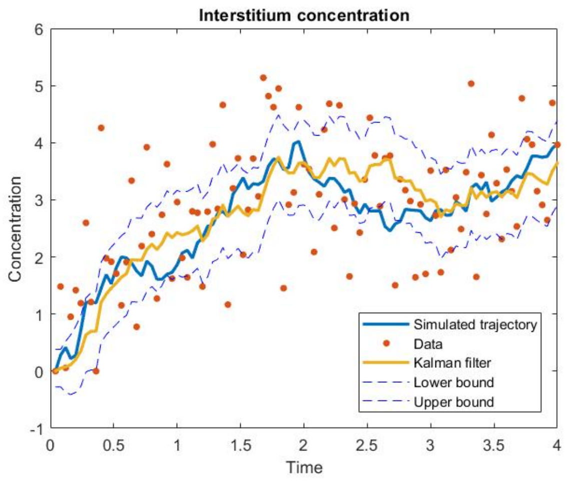
The MATLAB code for the Kalman filter algorithm can be found in appendix C. For efficient computations we follow Favetto and Samson (2010). They consider the system for S_t and I_t, so the state is $X_t = [S_t, I_t]^T$. Next, they multiply this state vector with the inverse of the matrix of eigenvectors of the system, which they call P^{-1}, and set $U_t = P^{-1}X_t$. They then derive the process and observation model for U_t. The advantage of this is that we now have that the matrix B in the process model in equation (13) is diagonal. This makes computations easier and allows us to compute the matrix Q exactly. However, we need to be careful here: after performing the Kalman filter we have to transform the state back to its original form by multiplying with P, the matrix of eigenvectors.

4.1.4 Results of Kalman filter

We will apply the Kalman filter, Algorithm 1, in the case of fully observed data and for partially observed data. We use the mean of the filtering distribution as the estimate for the state variable. From the covariance matrix P we use the diagonal entries, which represent the variance in the estimate for S and for I, to calculate approximate confidence bounds. For the fully observed data we use the measurement model from equation (10). The Kalman filter estimates up to time $t = 4$, including confidence bounds, can be found in Figure 4.



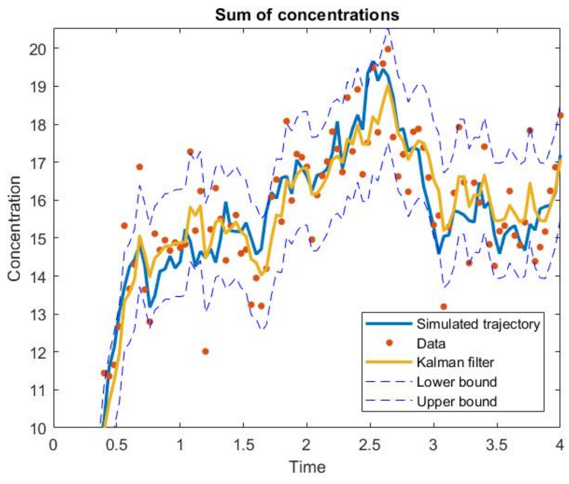
(a)



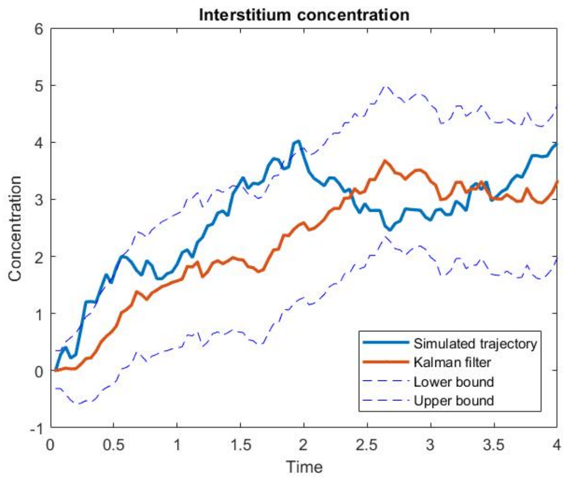
(b)

Figure 4: The sum of contrast agent concentrations (left) and the interstitium concentration (right) with the Simulated trajectory (blue line), the Data (red points) and the Kalman filter estimate (yellow line), including approximate confidence bounds

In both graphs we see that the Kalman filter estimate is closer to the simulated trajectory than the data points are, especially for the interstitium concentration. We also see that the simulated trajectory is mostly contained in the 95% confidence bounds. For partially observed data we use the measurement model from equation (11). We then find the following two graphs, which can be seen in Figure 5.



(a)



(b)

Figure 5: The sum of contrast agent concentrations (left) and the interstitium concentration (right) with in (a) the Simulated trajectory (blue line), the Partially Observed Data (red points) and the Kalman filter estimate (yellow line) and in (b) the Simulated trajectory (blue line) and Kalman filter estimate (red line), including approximate confidence bounds

One of the strengths of the Kalman filter is that we now have an available estimate of the interstitium concentration, although no data was available for this. We see, however, that the Kalman filter estimate is quite different from the simulated interstitium concentration. In order to get a better estimate, we can use a different algorithm, namely the RTS smoother algorithm.

4.2 Smoother algorithm

The RTS smoother algorithm (see e.g. Miller (2016)) can be used to retrospectively improve the estimate of the state variable at time i given by the Kalman filter, where we make use of the additional data at steps $i+1,\dots,n$. We will use the algorithm as described in the lecture notes by Sarkka (2011).

Algorithm 2: RTS smoother algorithm

```
**Input:** $\mu_i$ and $P_i$ for all $i = 1,\dots,n$, as computed by the Kalman filter algorithm
**Output:** Mean $\mu^S$ and covariance matrix $P^S$ of the smoothing distribution
initialization: $\mu^S = \mu_n$ and $P^S = P_n$ ;
**for** $i = n-1,\dots,1$ **do**
    $\mu_{i+1}^{-} = F\mu_i + C$ ;
    $P_{i+1}^{-} = FP_iF^T + Q$ ;
    $G_i = P_iF^TP_{i+1}^{-1}$ ;
    $\mu_i^S = \mu_i + G_i(\mu_{i+1}^S - \mu_{i+1}^{-})$ ;
    $P_i^S = P_i + G_i(P_{i+1}^S - P_{i+1}^{-})G_i^T$ ;
**end**
```

The superscript S indicates that it is the smoothed version. In this smoother algorithm we improve the Kalman filter estimate μ_i by adding the term $G_i(\mu_{i+1}^S - \mu_{i+1}^{-})$, which is some factor G_i multiplied by a correction term $\mu_{i+1}^S - \mu_{i+1}^{-}$. The MATLAB code for this algorithm can be found in Appendix D. After performing the RTS smoother algorithm, we obtain the smoothing distribution $N(\mu_i^S, P_i^S)$, where the mean μ_i^S is the smoothed estimate of the state variable.

4.2.1 Comparing Kalman filter and Smoother algorithm

In order to compare the Kalman filter and the Smoother algorithm, we perform a Monte Carlo study. We repeat the following steps $N = 1000$ times. First, we generate discrete time observations from the model. Next, we apply the Kalman filter and smoother algorithm and compute the mean of both the filtering and the smoothing distribution at all observation times. In the i-th repetition of these steps we obtain a vector $\mu^{F,i} = [\mu_1^{F,i} \dots \mu_n^{F,i}]$, where $\mu_j^{F,i}$ is the mean of the filtering distribution at time t_j. Similarly for the smoothing distribution we obtain a vector $\mu^{S,i} = [\mu_1^{S,i} \dots \mu_n^{S,i}]$. For both estimates we calculate the RMSE:

$$RMSE^{F,i} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(x_j - \mu_j^{F,i})^2}$$

$$RMSE^{S,i} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(x_j - \mu_j^{S,i})^2}$$

with x_j the real value of the state variable at time j from the simulation trajectory in Figure 2. The Monte Carlo study provides us with vectors $RMSE^F = [RMSE^{F,1} \dots RMSE^{F,N}]$ and $RMSE^S = [RMSE^{S,1} \dots RMSE^{S,N}]$. The mean of each of those vectors can be found in Table 1 and 2 below.

fully observed data	Kalman filter	RTS smoother
$S(t)$	0.618	0.514
$I(t)$	0.350	0.268

Table 1: Root Mean Squared Error for fully observed data after MC simulation

partially observed	Kalman filter	RTS smoother
$S(t)$	0.626	0.516
$I(t)$	0.591	0.572

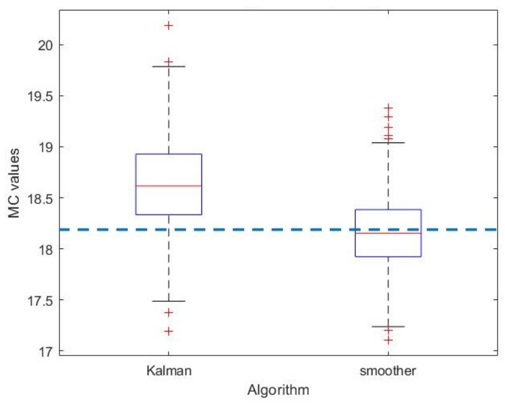
Table 2: Root Mean Squared Error for partially observed data after MC simulation

What we can see from this, is that for partially observed data the RMSE values are all higher than for fully observed data. We also see that in all cases the RTS smoother performs better than the Kalman filter, which is what we expect. It may also be interesting to see how many times the RTS smoother outperforms the Kalman filter in our Monte Carlo simulation. We do this by simply counting the number of times that the RMSE for the Kalman filter is higher than the RMSE for the RTS smoother. This gives the following results:

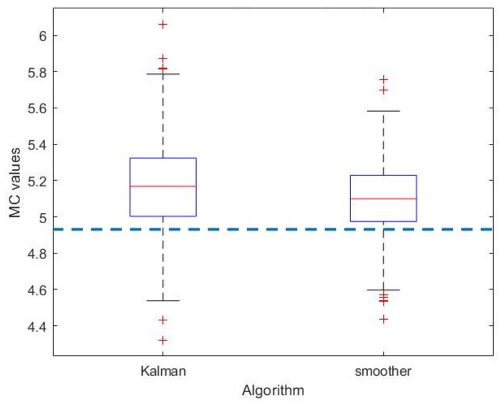
	fully observed	partially observed
$S(t)$	100 %	100 %
$I(t)$	100 %	97.2%

Table 3: Percentages for fully and partially observed data

Since the smoother algorithm considers more observations, we expect it to always outperform the Kalman filter. This is the case for $S(t)$ and also when we have fully observed data. For $I(t)$ and partially observed data, the RTS Smoother outperforms the Kalman filter almost all of the time. The reason that this is not 100% is because of the Monte Carlo error. Another way to compare the Kalman filter and smoother algorithm is to look at a box plot. We make a box plot to show the distribution of the Monte Carlo values for the estimate of the state variable at time $t = 32$ for both the Kalman and smoother algorithm. In each iteration of this Monte Carlo study, we generate a new set of observations and calculate the filtered and smoothed estimate for the state variable at time $t = 32$, which are the means of the filtering and smoothing distribution. The results can be found in Figure 6 and 7 below.

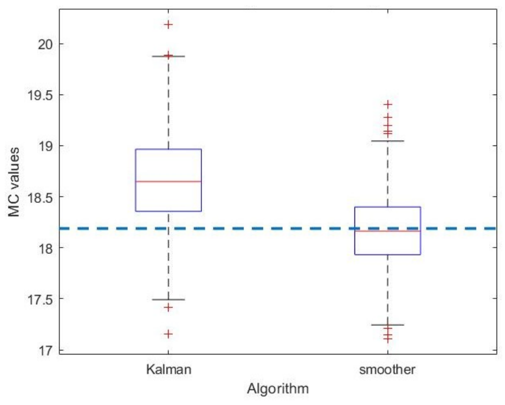


(a)

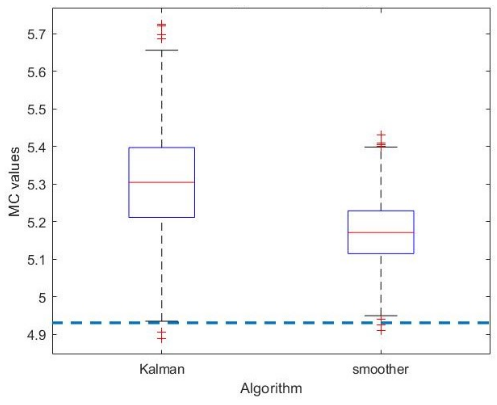


(b)

Figure 6: Monte Carlo values for the state variable at time $t = 32$ for both algorithms and fully observed data. The dotted blue line is the true value of the state variable at $t = 32$ from the simulated trajectory



(a)



(b)

Figure 7: Monte Carlo values for the state variable at time $t = 32$ for both algorithms and partially observed data. The dotted blue line is the true value of the state variable at $t = 32$ from the simulated trajectory

From the figures above we can see that the Monte Carlo values for the smoother algorithm are more concentrated around the median. So the smoothed estimates as computed by the smoother algorithm have a lower variance than the filtered estimates from the Kalman filter. Also, the median of the smoothed estimates is closer to the true simulated value of the state variable than the median of the filtered estimates. All in all, from the tables 1 up to 3 and figures 7 and 8, we see that the smoother algorithm outperforms the Kalman filter.

5 State estimation using particle filtering

Another option for state estimation is by using a particle filter. In a particle filter (see e.g. Carvalho et al. (2010)) we do not need normality of the noise terms, in contrast to the Kalman filter, but instead we approximate the filtering distribution by a discrete distribution, which consists of a sample of N state variables $\{x_t^{(i)}\}_{i=1}^N$, which we call particles, and corresponding weights (probabilities) $\{w_t^{(i)}\}_{i=1}^N$. A classical example is the Bootstrap Filter, which follows a propagate-resample framework: first we propagate the state variable $\{x_t^{(i)}\}_{i=1}^N$ to $\{x_{t+1}^{(i)}\}_{i=1}^N$ via the state equation and then we resample based on weights $w_{t+1}^{(i)} \propto p(y_{t+1} \mid x_{t+1}^{(i)})$. The Liu and West filter (Liu and West (2001)) is another example of a particle filter, which follows a resample-propagate framework. More recent descriptions of the Liu and West filter algorithm can be found in Carvalho et al. (2010) and Nemeth et al. (2015). We describe the Liu and West filter in the algorithm below.

Algorithm 3: Liu and West filter for state estimation

```
Input: Observations $y_1, \dots, y_n$
Output: Unweighted sample $\{x_n^{(i)}\}_{i=1}^N$ of particles which approximates the filtering
  distribution at time $t_n$
initialization: $x_0^{(i)} = x_0$ and $w_0^{(i)} = \frac{1}{N}$ for all $i = 1, \dots, N$ ;
for  $t = 1, \dots, n$ do
    Resample $\{\widetilde{x}_t^{(i)}\}_{i=1}^N$ from $\{x_t^{(i)}\}_{i=1}^N$ with weights $g_{t+1}^{(i)} \propto p(y_{t+1} \mid \mu_{t+1}^{(i)})$ ;
    Propagate $\{\widetilde{x}_t^{(i)}\}_{i=1}^N$ to $\{\widehat{x}_{t+1}^{(i)}\}_{i=1}^N$ by sampling from $p(x_{t+1} \mid \widetilde{x}_t^{(i)})$ ;
    Resample $\{x_{t+1}^{(i)}\}_{i=1}^N$ from $\{\widehat{x}_{t+1}^{(i)}\}_{i=1}^N$ with weights $w_{t+1}^{(i)} \propto \frac{p(y_{t+1}|\widehat{x}_{t+1}^{(i)})}{p(y_{t+1}|\widetilde{\mu}_{t+1}^{(i)})}$ ;
end
```

where we use

$$\mu_{t+1}^{(i)} = \mathrm{E}(x_{t+1} \mid x_t^{(i)}) \ , \ \widetilde{\mu}_{t+1}^{(i)} = \mathrm{E}(x_{t+1} \mid \widetilde{x}_t^{(i)})$$

The quantities $\mu_{t+1}^{(i)}$ and $\widetilde{\mu}_{t+1}^{(i)}$ follow from the state equation. The steps of Algorithm 3 are derived in section 5.2. Note that we use the following Bayesian notation: $p(y_{t+1} \mid \mu_{t+1}^{(i)}) = f_{Y_{t+1}|X_{t+1}}(y_{t+1} \mid \mu_{t+1}^{(i)})$, which is the probability density function of Y_{t+1}, given $X_{t+1} = x_{t+1}$, evaluated at y_{t+1} given $\mu_{t+1}^{(i)}$. A common problem in particle filters is weight degeneracy, which means that as time passes, less and less particles have a positive weight and the variance of the weights increases with every iteration. The resampling steps in the Liu and West filter help solve this problem, since after resampling we will end up with particles that have relatively high weights and abandon any zero-weight particles.

5.1 Two implementations

There are two possible implementations of the Liu and West algorithm, which are equivalent. Both implementations show up in the literature. One implementation is where we follow the algorithm above. The other implementation follows a slightly different algorithm, which can be found in Algorithm 4 below.

```
Algorithm 4: Equivalent Liu and West filter algorithm for state estimation
  Input: Observations $y_1, \dots, y_n$
  Output: Sample $\{x_n^{(i)}, w_n^{(i)}\}_{i=1}^N$ of particles and corresponding weights which approximates
    the filtering distribution at time $t_n$
  initialization: $x_0^{(i)} = x_0$ and $w_0^{(i)} = \frac{1}{N}$ for all $i = 1, \dots, N$ ;
  for $t = 1, \dots, n$ do
      Resample $\{\widetilde{x}_t^{(i)}\}_{i=1}^N$ from $\{x_t^{(i)}\}_{i=1}^N$ with weights $h_{t+1}^{(i)} \propto w_t^{(i)} p(y_{t+1} \mid \mu_{t+1}^{(i)})$ ;
      Propagate $\{\widetilde{x}_t^{(i)}\}_{i=1}^N$ to $\{\widehat{x}_{t+1}^{(i)}\}_{i=1}^N$ by sampling from $p(x_{t+1} \mid \widetilde{x}_t^{(i)})$ ;
      Calculate corresponding weights $w_{t+1}^{(i)} \propto \frac{p(y_{t+1} \mid \widehat{x}_{t+1}^{(i)})}{p(y_{t+1} \mid \widetilde{\mu}_{t+1}^{(i)})}$ ;
  end
```

In Algorithm 4 we obtain a weighted sample after the final step, so a sample of states with corresponding weights $\{\widehat{x}_{t+1}^{(i)}, w_{t+1}^{(i)}\}_{i=1}^N$. However, in Algorithm 3 we obtain an unweighted sample $\{x_{t+1}^{(i)}\}_{i=1}^N$. We can see that those algorithms are equivalent as follows. Suppose we have a weighted sample $\{\widehat{x}_T^{(i)}, w_T^{(i)}\}_{i=1}^N$, which was produced by Algorithm 4. We apply Algorithm 4 again to find an approximation for the posterior distribution of x_{T+1}. We calculate weights $h_{T+1}^{(i)} \propto w_T^{(i)} p(y_{T+1} \mid \mu_{T+1}^{(i)})$ and resample based on those weights. This is, however, the same as first resampling based on the weights $w_T^{(i)}$, which yields an unweighted sample $\{x_T\}_{i=1}^N$, and then resampling based on weights $g_{T+1}^{(i)} \propto p(y_{T+1} \mid \mu_{T+1}^{(i)})$, since the resampling steps are independent of each other. This is exactly what happens in Algorithm 3: in the final step of the previous iteration we resampled based on weights $w_T^{(i)}$ and in the first step of the current iteration we resample based on weights $g_{T+1}^{(i)} \propto p(y_{T+1} \mid \mu_{T+1}^{(i)})$. In our implementation we will follow Algorithm 3. Since we have to resample twice in this algorithm, it will be slower in computational time compared to Algorithm 4, but the advantage is that we will end up with an unweighted sample after each iteration, which means that for calculating characteristics of interest based on this sample we do not have to take into account the weights.

5.2 Derivation of Liu and West filter for state estimation

Assume we have state variables $\{x_t\}$ and observations $\{y_t\}$, which form a hidden Markov model (HMM). Then for the posterior density of the state variable at time $t+1$ we have (see also Liu and West (2001)) :

$$p(x_{t+1} \mid y_{1:t+1}) = p(x_{t+1} \mid y_{t+1}, y_{1:t}) \propto p(y_{t+1} \mid x_{t+1}, y_{1:t}) p(x_{t+1} \mid y_{1:t})$$

$$= p(y_{t+1} \mid x_{t+1}) p(x_{t+1} \mid y_{1:t}) = p(y_{t+1} \mid x_{t+1}) \int p(x_{t+1} \mid y_{1:t}, x_t) p(x_t \mid y_{1:t}) dx_t$$

$$= \int p(y_{t+1} \mid x_{t+1}) p(x_{t+1} \mid x_t) p(x_t \mid y_{1:t}) dx_t$$

We assume that we have a sample $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^N$ of state variables and corresponding weights such that:

$$p(x_t \mid y_{1:t}) = \sum_{i=1}^N w_t^{(i)} \delta(x_t^{(i)})$$

with $\delta(.)$ the distribution which places all of its mass at $x_t^{(i)}$. So the integral over all values of x_t turns into a sum over the particles $x_t^{(i)}$ and we can now rewrite our posterior density as:

$$p(x_{t+1} \mid y_{1:t+1}) \propto \int p(y_{t+1} \mid x_{t+1}) p(x_{t+1} \mid x_t) p(x_t \mid y_{1:t}) dx_t$$

$$= \sum_{i=1}^{N} p(y_{t+1} \mid x_{t+1}) p(x_{t+1} \mid x_t^{(i)}) w_t^{(i)}$$

The transition density $p(x_{t+1} \mid x_t^{(i)})$ follows from the state equation. The only unknown is the likelihood $p(y_{t+1} \mid x_{t+1})$, since x_{t+1} is unknown. Let $\mu_{t+1}^{(i)} = \mathrm{E}(x_{t+1} \mid x_t^{(i)})$ be the estimate for $x_{t+1}^{(i)}$. Then we get:

$$\begin{aligned} p(x_{t+1} \mid y_{1:t+1}) &\propto \sum_{i=1}^{N} p(y_{t+1} \mid x_{t+1}) p(x_{t+1} \mid x_t^{(i)}) w_t^{(i)} \\ &= \sum_{i=1}^{N} \frac{p(y_{t+1} \mid x_{t+1}^{(i)})}{p(y_{t+1} \mid \mu_{t+1}^{(i)})} p(y_{t+1} \mid \mu_{t+1}^{(i)}) p(x_{t+1} \mid x_t^{(i)}) w_t^{(i)} \\ &= \sum_{i=1}^{N} \frac{p(y_{t+1} \mid x_{t+1}^{(i)})}{p(y_{t+1} \mid \mu_{t+1}^{(i)})} p(x_{t+1} \mid x_t^{(i)}) g_{t+1}^{(i)} \end{aligned}$$

with $g_{t+1}^{(i)} \propto w_t^{(i)} p(y_{t+1} \mid \mu_{t+1}^{(i)})$, which are the initial weights based on which we will resample for the first time. We resample the $x_t^{(i)}$'s with probabilities proportional to $g_{t+1}^{(i)}$ and propagate $x_t^{(i)}$ to $x_{t+1}^{(i)}$ via $p(x_{t+1} \mid x_t^{(i)})$. Next, we use importance sampling (see Johansen and Evers, 2007), which works as follows. We fix $i \in \{1, \ldots, N\}$. Let $\pi_i = p(x_{t+1} \mid y_{1:t+1}) \propto p(y_{t+1} \mid x_{t+1}^{(i)}) p(x_{t+1} \mid x_t^{(i)}) w_t^{(i)}$ be the target density and $\eta_i = q(x_{t+1} \mid y_{1:t+1}) \propto p(y_{t+1} \mid \mu_{t+1}^{(i)}) p(x_{t+1} \mid x_t^{(i)}) w_t^{(i)}$ the importance density. Let h be an integrable function and X a random variable with density π_i. We want to estimate $h(X)$, in our case $h(X) = X$. Then:

$$\mathrm{E}_{\pi_i}(h(X)) = \int \pi_i(x) h(x) dx = \int \eta_i(x) \frac{\pi_i(x)}{\eta_i(x)} h(x) dx = \int \eta_i(x) w(x) h(x) dx = \mathrm{E}_{\eta_i}(w(X) h(X))$$

with $w_i(x) = \frac{\pi_i(x)}{\eta_i(x)}$ the importance weights. This means that the expectation of $h(X)$ under the density π_i is equal to the expectation of $w_i(X)h(X)$ under the density η_i. So, if we resample the $x_{t+1}^{(i)}$'s based on the importance weights, we get a sample which approximates the target density, which is also the posterior density. The importance weights are then:

$$w_{t+1}^{(i)} = \frac{\pi_i}{\eta_i} = \frac{p(x_{t+1} \mid y_{1:t+1})}{q(x_{t+1} \mid y_{1:t+1})} \propto \frac{p(y_{t+1} \mid x_{t+1}^{(i)}) p(x_{t+1} \mid x_t^{(i)}) w_t^{(i)}}{p(y_{t+1} \mid \mu_{t+1}^{(i)}) p(x_{t+1} \mid x_t^{(i)}) w_t^{(i)}} = \frac{p(y_{t+1} \mid x_{t+1}^{(i)})}{p(y_{t+1} \mid \mu_{t+1}^{(i)})}$$

5.3 Effective sample size for weighted samples

Johansen and Evers (2007) suggest using the effective sample size to quantify the quality of a weighted sample. Let $\{Y_i\}_{i=1}^N$ be an unweighted sample from the target (= filtering) distribution and $\{X_i\}_{i=1}^N$ be a weighted sample, with normalized weights $\{W_i\}_{i=1}^N$, which approximates the target distribution. If we want to estimate the mean of the target distribution, then the obvious estimators for this are:

$$\widehat{\mu}_Y = \sum_{i=1}^{N} \frac{1}{N} Y_i$$

$$\widehat{\mu}_X = \sum_{i=1}^{N} W_i X_i$$

The effective sample size is then defined as:

$$N_{ESS} = N \frac{\mathrm{Var}(\widehat{\mu}_Y)}{\mathrm{Var}(\widehat{\mu}_X)}$$

The effective sample size can be interpreted as the number of unweighted samples from the target distribution we need to obtain an estimator with the same variance as $\widehat{\mu}_X$. Since we are unable to draw samples from the unknown target distribution, we have to estimate N_{ESS}. Johansen and Evers (2007) use the following approximation:

$$N_{ESS} \approx \frac{\left(\sum_{i=1}^N W_i\right)^2}{\sum_{i=1}^N W_i^2} = \frac{1}{\sum_{i=1}^N W_i^2}$$

The quality of the weights is good if N_{ESS} is close to N and bad if N_{ESS} is close to 1, which may be a sign of weight degeneracy. Throughout this thesis we will often check the effective sample size to see if the quality of the weights is good.

5.4 Applying Liu and West filter for state estimation

In this section we apply the Liu and West filter for state estimation to the diffusion model from equations (4) and (5). For this we use Algorithm 4, with initial weights

$$g_{t+1}^{(i)} \propto \exp(-\frac{1}{2}(y_{t+1} - H\mu_{t+1}^{(i)})^2) \ ,$$

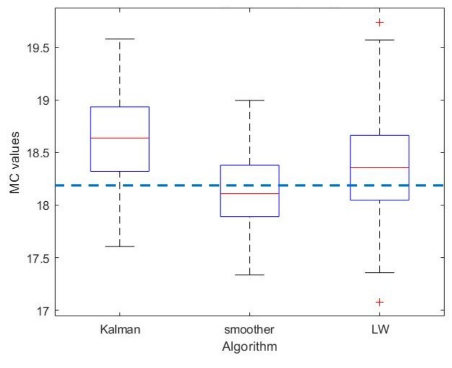
transition distribution

$$N(F\widetilde{x}_t^{(i)} + C, Q_t) \ ,$$

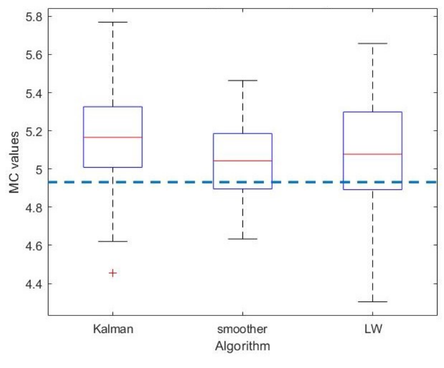
importance weights

$$w_{t+1}^{(i)} \propto \frac{\exp(-\frac{1}{2}(y_{t+1} - H\widehat{x}_{t+1}^{(i)})^2)}{\exp(-\frac{1}{2}(y_{t+1} - H\widetilde{\mu}_{t+1}^{(i)})^2)} \ ,$$

and $\mu_{t+1}^{(i)} = Fx_t^{(i)} + C$, $\widetilde{\mu}_{t+1}^{(i)} = F\widetilde{x}_t^{(i)} + C$, which follow from the state equation. The estimate for the state variable at time t will be the unweighted mean of the particles, i.e. $\widehat{x}_t = \sum_{i=1}^N \frac{1}{N} x_t^{(i)}$. The MATLAB code for this algorithm can be found in Appendix *E*. We will look at box plots at time $t = 32$, for both partially and fully observed data, and both $S(t)$ and $I(t)$, similar to section 4.2.1. We perform a Monte Carlo study. In each iteration $i \in \{1, \ldots, M\}$ we generate a new set of observations from the diffusion model. On this set of observations we apply the Kalman filter and smoother algorithm, and calculate the means of the filtering and smoothing distribution at time t, which we call $\mu_t^{F,i}$ and $\mu_t^{S,i}$. Next, we apply the Liu and West filter on the set of observations and calculate the unweighted mean of the particles at time t, which we call $\mu_t^{LW,i}$. The result is that we have vectors $\mu_t^F = [\mu_t^{F,1}, \ldots, \mu_t^{F,M}]$, $\mu_t^S = [\mu_t^{S,1}, \ldots, \mu_t^{S,M}]$ and $\mu_t^{LW} = [\mu_t^{LW,1}, \ldots, \mu_t^{LW,M}]$. For those vectors we make a box plot. Because of the large computational time of the Liu and West filter, we take the number of simulations in the Monte Carlo study $M = 100$ and the number of particles in the Liu and West filter $N = 100$.

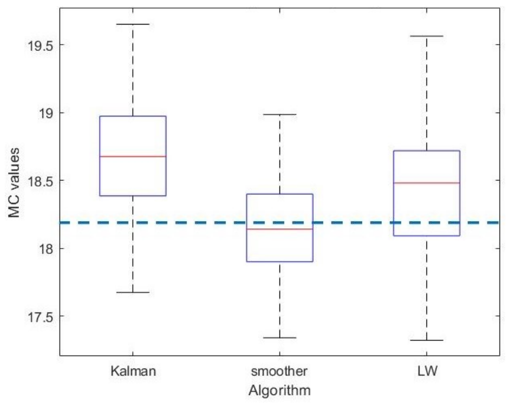


(a)

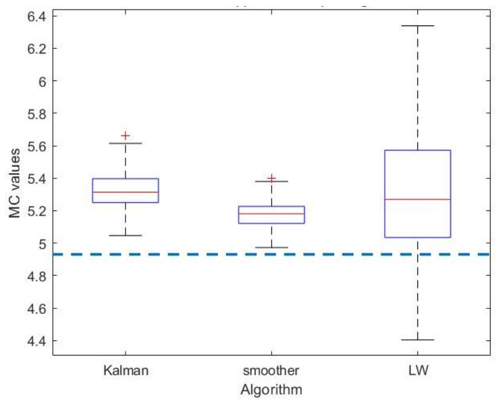


(b)

Figure 8: Monte Carlo values for the state variable at time $t = 32$ for both algorithms and fully observed data. The dotted blue line is the value of the state variable at $t = 32$ from the simulated trajectory



(a)



(b)

Figure 9: Monte Carlo values for the state variable at time $t = 32$ for both algorithms and partially observed data. The dotted blue line is the value of the state variable at $t = 32$ from the simulated trajectory

Regarding the smoother algorithm, the results are as expected: the median of the smoother estimates is in all cases closest to the true value of the state variable. For a linear Gaussian state space model, we expect the Kalman filter to be preferable. Due to the Monte Carlo error (recall we only used $M = 100$) the Liu and West filter seems to perform slightly better, but the values are more spread out than for the Kalman filter. This indicates that the Liu and West filter seems to perform well and is competitive with the Kalman filter. Furthermore, we find effective sample sizes of 96 for partially observed case and 92 for the fully observed case, so there is no sign of weight degeneracy.

6 Parameter estimation

6.1 Liu and West filter for parameter estimation

So far, we have assumed that there are no unknown parameters in the model. In this section we consider the case of unknown parameters. The Liu and West filter is able to simultaneously estimate state variables and parameters. A description of this filter can be found in Carvalho et al. (2010) and Nemeth et al. (2015). First, choose the so-called shrinkage coefficient $a \in (0, 1)$. The algorithm then works as follows.

Algorithm 5: Liu and West filter algorithm for parameter estimation

```
Input: Observations $y_1, \dots, y_n$
Output: Unweighted sample $\{x_n^{(i)}, \theta_n^{(i)}\}_{i=1}^N$ of particles, in which $\{x_n^{(i)}\}_{i=1}^N$ approximates
  the filtering distribution at time $t_n$ and $\{\theta_n^{(i)}\}_{i=1}^N$ approximates the unknown parameter
initialization: $x_0^{(i)} = x_0$, $w_0^{(i)} = \frac{1}{N}$ for all $i = 1, \dots, N$, $\{\theta_0^{(i)}\}_{i=1}^N$ is a sample simulated from
  a specified prior distribution ;
for $t = 1, \dots, n$ do
    Resample $\{\widetilde{x}_t^{(i)}, \widetilde{\theta}_t^{(i)}\}_{i=1}^N$ from $\{x_t^{(i)}, \theta_t^{(i)}\}_{i=1}^N$ with weights $g_{t+1}^{(i)} \propto p(y_{t+1} \mid \mu_{t+1}^{(i)}, m_t^{(i)})$ ;
    Propagate $\{\widetilde{\theta}_t^{(i)}\}_{i=1}^N$ to $\{\widehat{\theta}_{t+1}^{(i)}\}_{i=1}^N$ by sampling from $N(\widetilde{m}_t^{(i)}, (1-a^2)V_t)$ ;
    Propagate $\{\widetilde{x}_t^{(i)}\}_{i=1}^N$ to $\{\widehat{x}_{t+1}^{(i)}\}_{i=1}^N$ by sampling from $p(x_{t+1} \mid \widetilde{x}_t^{(i)}, \widehat{\theta}_{t+1}^{(i)})$ ;
    Resample $\{x_{t+1}^{(i)}, \theta_{t+1}^{(i)}\}_{i=1}^N$ from $\{\widehat{x}_{t+1}^{(i)}, \widehat{\theta}_{t+1}^{(i)}\}_{i=1}^N$ with weights $w_{t+1}^{(i)} \propto \frac{p(y_{t+1}|\widehat{x}_{t+1}^{(i)}, \widehat{\theta}_{t+1}^{(i)})}{p(y_{t+1}|\widetilde{\mu}_{t+1}^{(i)}, \widetilde{m}_t^{(i)})}$ ;
end
```

where we use

$$\mu_{t+1}^{(i)} = \mathrm{E}(x_{t+1} \mid x_t^{(i)}, \theta_t^{(i)})$$

$$m_t^{(i)} = a\theta_t^{(i)} + (1-a)\overline{\theta}_t$$

$$\overline{\theta}_t = \sum_{i=1}^{N} \frac{1}{N}\theta_t^{(i)}$$

$$V_t = \sum_{i=1}^{N} \frac{1}{N}(\theta_t^{(i)} - \overline{\theta}_t)(\theta_t^{(i)} - \overline{\theta}_t)^T$$

The quantity $\mu_{t+1}^{(i)}$ follows from the state equation. The derivation of Algorithm 5 can be found in section 6.2. In each iteration we use N particles. We resample the particles at time t using weights based on observation y_{t+1}. Then we propagate θ_t and x_t to time $t+1$ and resample again, with weights based on the quotient of the target density and the estimated density. Note that all weights have to be normalized. How to choose the shrinkage coefficient a can be found in Liu and West (2001). We will take $a = 0.995$. After the resampling in the final step, the Liu and West filter provides us with an unweighted sample of state variables $x_t^{(i)}$ and parameter values $\theta_t^{(i)}$. The final estimates at time t are then the unweighted averages:

$$\widehat{x}_t = \sum_{i=1}^{N} \frac{1}{N} x_t^{(i)} \ , \ \widehat{\theta}_t = \sum_{i=1}^{N} \frac{1}{N}\theta_t^{(i)}$$

6.2 Derivation of Liu and West filter for parameter estimation

Suppose we are given a sample of state variables and parameters at time t, $\{x_t^{(i)}, \theta_t^{(i)}\}_{i=1}^N$, and associated weights $\{w_t^{(i)}\}_{i=1}^N$, which represent an importance sample approximation to the posterior $p(x_t, \theta_t \mid D_t)$, with $D_t = \{D_{t-1}, y_t\}$ and D_{t-1} the observations up to and including time $t-1$. Then for the posterior density of x_{t+1} and θ_{t+1} we have (see also Liu and West (2001)) :

$$p(x_{t+1}, \theta_{t+1} \mid D_{t+1}) = p(x_{t+1}, \theta_{t+1} \mid y_{t+1}, D_t) \propto p(y_{t+1} \mid x_{t+1}, \theta_{t+1}, D_t)p(x_{t+1}, \theta_{t+1} \mid D_t)$$

$$= p(y_{t+1} \mid x_{t+1}, \theta_{t+1})p(x_{t+1}, \theta_{t+1} \mid D_t) \propto p(y_{t+1} \mid x_{t+1}, \theta_{t+1})p(x_{t+1} \mid \theta_{t+1}, D_t)p(\theta_{t+1} \mid D_t)$$

where $p(y_{t+1} \mid x_{t+1}, \theta_{t+1})$ follows from the measurement model and $p(x_{t+1} \mid \theta_{t+1}, D_t)$ follows from the state equation. We approximate the posterior $p(\theta_{t+1} \mid D_t)$ by:

$$p(\theta_{t+1} \mid D_t) \approx \sum_{i=1}^{N} w_t^{(i)} N(\theta_{t+1} \mid m_t^{(i)}, h^2 V_t)$$

which is a mixture of multivariate normal densities which have mean $m_t^{(i)}$ and variance matrix $h^2 V_t$, and

$$m_t^{(i)} = a\theta_t^{(i)} + (1-a)\overline{\theta}_t$$

$$V_t = \sum_{i=1}^{N} w_t^{(i)}(\theta_t^{(i)} - \overline{\theta}_t)(\theta^{(i)} - \overline{\theta}_t)^T$$

$$\overline{\theta} = \sum_{i=1}^{N} w_t^{(i)}\theta_t^{(i)}$$

Liu and West (2001) state that this mixture of normal densities has variance $(1+h^2)V_t$ if we use $m_t^{(i)} = \theta_t^{(i)}$, and is thus overdispersed. As a solution they take $m_t^{(i)} = a\theta_t^{(i)} + (1-a)\overline{\theta}_t$, with $a = \sqrt{1-h^2}$ the shrinkage coefficient and h the smoothing parameter. Then the mixture has the correct mean $\overline{\theta}_t$ and variance V_t. The derivation of this can be found in Liu and West

(2001) and we derive it here as well. Assume that $\Theta_t \sim N(\overline{\theta}_t, V_t)$, where $\overline{\theta}_t = \frac{1}{N}\sum_{i=1}^N \theta_t^{(i)}$ and $V_t = \frac{1}{N}\sum_{i=1}^N (\theta_t^{(i)} - \overline{\theta}_t)(\theta_t^{(i)} - \overline{\theta})^T$. We consider $\overline{\theta}_t, V_t$ to be fixed. Let $a \in (0,1)$. Take

$$\Theta_{t+1} \mid (\Theta_t = \theta_t) \sim N(a\theta_t + (1-a)\overline{\theta}_t, (1-a^2)V_t) \tag{14}$$

Then Θ_{t+1} has again a Normal distribution with, using the law of total expectation,

$$\mathrm{E}(\Theta_{t+1}) = \mathrm{E}(\mathrm{E}(\Theta_{t+1} \mid \Theta_t)) = \mathrm{E}(a\Theta_t + (1-a)\overline{\theta}_t) = a\overline{\theta}_t + (1-a)\overline{\theta}_t = \overline{\theta}_t$$

and, using the law of total variance,

$$\begin{aligned}\mathrm{Cov}(\Theta_{t+1}) &= \mathrm{E}(\mathrm{Cov}(\Theta_{t+1} \mid \Theta_t)) + \mathrm{Cov}(\mathrm{E}(\Theta_{t+1} \mid \Theta_t)) \\ &= \mathrm{E}((1-a^2)V_t) + \mathrm{Cov}(a\Theta_t + (1-a)\overline{\theta}_t) = (1-a^2)V_t + a^2\mathrm{Cov}(\Theta_t) = V_t\end{aligned}$$

with $h^2 = 1 - a^2$ and $a = \sqrt{1-h^2}$. So we see that by taking $m_t^{(i)} = a\theta_t^{(i)} + (1-a)\overline{\theta}_t$ and variance $(1-a^2)V_t$ we get the correct mean $\overline{\theta}_t$ and variance V_t for Θ_{t+1}. Now for the posterior $p(\theta_{t+1} \mid D_t)$ we have:

$$p(\theta_{t+1} \mid D_t) = \int p(\theta_{t+1} \mid \theta_t, D_t)p(\theta_t \mid D_t)d\theta_t = \int p(\theta_{t+1} \mid \theta_t)p(\theta_t \mid D_t)d\theta_t$$

We estimate $p(\theta_t \mid D_t)$ by $\sum_{i=1}^N w_t^{(i)}\delta(\theta_t^{(i)})$. Then:

$$p(\theta_{t+1} \mid D_t) = \sum_{i=1}^N p(\theta_{t+1} \mid \theta_t^{(i)})w_t^{(i)}$$

Using (14) we get that:

$$p(\theta_{t+1} \mid D_t) \approx \sum_{i=1}^N w_t^{(i)} N(\theta_{t+1} \mid a\theta_t^{(i)} + (1-a)\overline{\theta}_t, (1-a^2)V_t)$$

with $N(x \mid \mu, \Sigma)$ the normal density with mean μ and covariance matrix Σ, evaluated in x. Next, we will show how to relate the smoothing parameter h and shrinkage coefficient a as follows. We can write:

$$\theta_{t+1} = a\theta_t + (1-a)\overline{\theta}_t + h\varepsilon_t$$

with $\varepsilon_t \sim N(0, V_t)$. We can rewrite this as:

$$\theta_{t+1} - \overline{\theta}_t = a\theta_t + (1-a)\overline{\theta}_t - \overline{\theta}_t + h\varepsilon_t = a(\theta_t - \overline{\theta}_t) + h\varepsilon_t$$

If we assume stationarity, then $\overline{\theta}_t = \mu$. Define $Z_t = \theta_t - \mu$, then:

$$Z_{t+1} = aZ_t + h\varepsilon_t$$

This is an AR(1) process. Under stationarity, we have $\mathrm{Var}(Z_t) = c^2$. Then:

$$\begin{aligned}\mathrm{Var}(Z_{t+1}) &= a^2\mathrm{Var}(Z_t) + h^2\mathrm{Var}(\varepsilon_t) \\ c^2 &= a^2c^2 + h^2V_t \\ (1-a^2)c^2 &= h^2V_t\end{aligned}$$

The variance of θ_t, and therefore the variance of Z_t, has to be equal to V_t. So $c^2 = V_t$. This means that $a = \sqrt{1-h^2}$. In the last step of the Liu and West filter we resample for a second time, using importance sampling. This works the same as in the derivation of the Liu and West filter for state estimation. However, now we have the following target density (see Nemeth et al. (2015)) :

$$p(x_{t+1}, \theta, i \mid D_{t+1}) = p(x_{t+1}, \theta, i \mid y_{t+1}, D_t) \propto p(y_{t+1} \mid x_{t+1}^{(i)}, \theta^{(i)})p(x_{t+1} \mid x_t^{(i)}, \theta^{(i)})w_t^{(i)}$$

and importance density q:

$$q(x_{t+1}, \theta, i \mid D_{t+1}) = p(x_{t+1}, \theta, i \mid y_{t+1}, D_t) \propto p(y_{t+1} \mid \mu_{t+1}^{(i)}, m_t^{(i)})p(x_{t+1} \mid x_t^{(i)}, \theta^{(i)})w_t^{(i)}$$

So the importance weights are:

$$w_{t+1}^{(i)} = \frac{p(x_{t+1}, \theta, i \mid D_{t+1})}{q(x_{t+1}, \theta, i \mid D_{t+1})} \propto \frac{p(y_{t+1} \mid x_{t+1}^{(i)}, \theta^{(i)})p(x_{t+1} \mid x_t^{(i)}, \theta^{(i)})w_t^{(i)}}{p(y_{t+1} \mid \mu_{t+1}^{(i)}, m_t^{(i)})p(x_{t+1} \mid x_t^{(i)}, \theta^{(i)})w_t^{(i)}} = \frac{p(y_{t+1} \mid x_{t+1}^{(i)}, \theta^{(i)})}{p(y_{t+1} \mid \mu_{t+1}^{(i)}, m_t^{(i)})}$$

6.3 Application to AR(1) model

Before we apply the Liu and West filter to our diffusion model, we will first apply it to the following AR(1) model:

$$x_{t+1} = \theta x_t + \varepsilon_t$$

$$y_{t+1} = x_{t+1} + \eta_t$$

with ε_t and η_t independent standard Gaussian random variables. We apply Algorithm 5 to the AR(1) model, with initial weights

$$g_{t+1}^{(i)} \propto \exp(-\frac{1}{2}(y_{t+1} - \mu_{t+1}^{(i)})^2) \ ,$$

transition distribution

$$N(\theta_{t+1}^{(i)} x_t^{(i)}, 1) \ ,$$

and importance weights

$$w_{t+1}^{(i)} = \frac{\exp(-\frac{1}{2}(y_{t+1} - x_{t+1}^{(i)})^2)}{\exp(-\frac{1}{2}(y_{t+1} - \widetilde{\mu}_{t+1}^{(i)})^2)} \ ,$$

where $\mu_{t+1}^{(i)} = \theta_t^{(i)} x_t^{(i)}$. Then the estimates for the state variable and parameter at time $t+1$ are the unweighted averages:

$$\widehat{x}_{t+1} = \sum_{i=1}^{N} x_{t+1}^{(i)} \ , \ \widehat{\theta}_{t+1} = \sum_{i=1}^{N} \theta_{t+1}^{(i)}$$

We simulate $n = 1000$ observations from the AR(1) model, using $\theta = 0.8$. In the Liu and West filter we set: $a = 0.995$ as the shrinkage coefficient, $N = 5000$ as the number of particles and $x_0 = 0$ as the initial state. Recall that an AR(1) model is stationary for $\theta \in (-1, 1)$. Since $\theta = 0.8 > 0$, we use the Uniform$[0, 1]$ distribution as the prior for θ. We simulate a sequence of length n from the AR(1) model, setting $rng(0)$ in MATLAB. Observations are then simulated according to the measurement equation. The MATLAB code can be found in Appendix F. Confidence bounds are calculated as the empirical quantiles of the particles (i.e. we sort the particles and look at the $0.025N$-th and $0.975N$-th values). For the weights we calculate at $t = n$ the effective sample size. Furthermore, we will compare the estimate from the Liu and West filter with the maximum likelihood estimate using the conditional maximum likelihood (see e.g. lecture notes by Zivot (2005)), which can be computed in closed form.

Lemma 1. *Given an AR(1) model, $x_{t+1} = \theta x_t + \varepsilon_t$, with $\varepsilon_t \sim N(0,1)$ and observations $y = \{y_1, \dots, y_n\} = \{x_1, \dots, x_T\}$. Then the Conditional Maximum Likelihood Estimate (CMLE) of θ is given by:*

$$\widehat{\theta}_{CMLE} = \frac{\sum_{t=2}^{T} y_t y_{t-1}}{\sum_{t=2}^{T} y_{t-1}^2}$$

Proof. We can compute the likelihood function as:

$$L(y_1, \dots, y_T; \theta) = f(y_1; \theta) \prod_{t=2}^{T} f(y_t \mid D_{t-1}; \theta)$$

with $D_{t-1} = \{y_1, \dots, y_{t-1}\}$ and $f(.)$ the pdf of the observations. The log-likelihood is then:

$$\mathcal{L}(y_1, \dots, y_T; \theta) = \ln(L(y_1, \dots, y_T; \theta)) = \ln(f(y_1, \theta)) + \sum_{t=2}^{T} \ln(f(y_t \mid D_{t-1}, \theta))$$

The conditional maximum likelihood estimate is then defined as:

$$\widehat{\theta}_{CMLE} = \arg\max_{\theta} \sum_{t=2}^{T} \ln(f(y_t \mid D_{t-1}, \theta))$$

In the AR(1) model we have:

$$y_t \mid D_{t-1} \sim N(\theta y_{t-1}, 1) \ , \ t = 2, 3, \ldots, T$$

So the conditional density is:

$$f(y_t \mid D_{t-1}, \theta) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(y_t - \theta y_{t-1})^2}$$

So the conditional maximum likelihood estimate is:

$$\widehat{\theta}_{CMLE} = \arg\max_{\theta} \sum_{t=2}^{T} -\frac{1}{2}\ln 2\pi - \frac{1}{2}(y_t - \theta y_{t-1})^2$$

$$= \arg\max_{\theta} \sum_{t=2}^{T} -\frac{1}{2}(y_t - \theta y_{t-1})^2 = \arg\max_{\theta} \sum_{t=2}^{T} \theta y_t y_{t-1} - \frac{1}{2}\theta^2 y_{t-1}^2$$

We can work this out analytically by calculating the derivative and setting it equal to 0:

$$0 = \sum_{t=2}^{T} (y_t y_{t-1} - y_{t-1}^2 \widehat{\theta}_{CMLE})$$

$$\widehat{\theta}_{CMLE} \sum_{t=2}^{T} y_{t-1}^2 = \sum_{t=2}^{T} y_t y_{t-1}$$

So we find that:

$$\widehat{\theta}_{CMLE} = \frac{\sum_{t=2}^{T} y_t y_{t-1}}{\sum_{t=2}^{T} y_{t-1}^2}$$

□

Next, we calculate 95% confidence bounds for the ML estimate. We know the asymptotic distribution of our ML estimate (see e.g. lecture notes by Blasques and Koopman (2019)):

$$\widehat{\theta} \sim N(\theta, (1 - \theta^2)/T)$$

So the approximate confidence lower and upper bound for the ML estimate are the $\alpha/2$ and $1 - \alpha/2$ quantiles of the $N(\theta, (1 - \theta^2)/T)$ distribution, where we take $\alpha = 0.95$. In the figures below we plot the estimate of the parameter θ over time, for both the Liu and West filter and the Maximum Likelihood, including their confidence bounds.

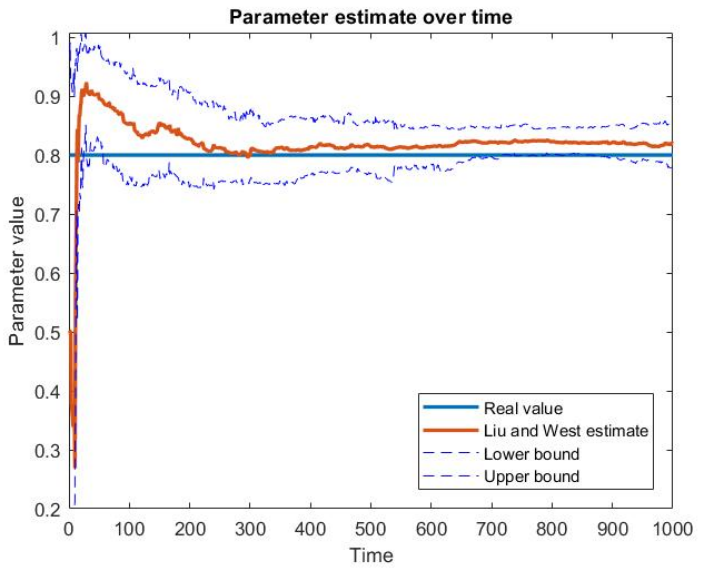


Figure 10: Estimate for the parameter θ over time by the Liu and West filter

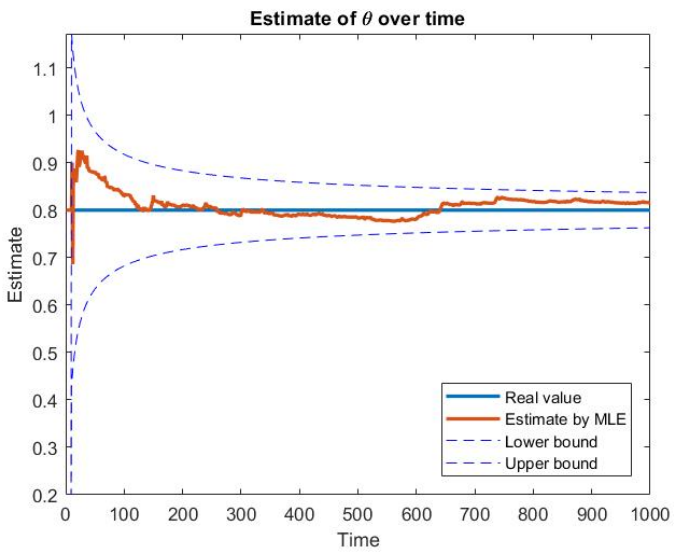
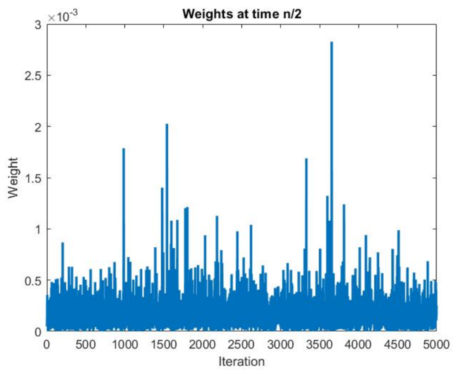
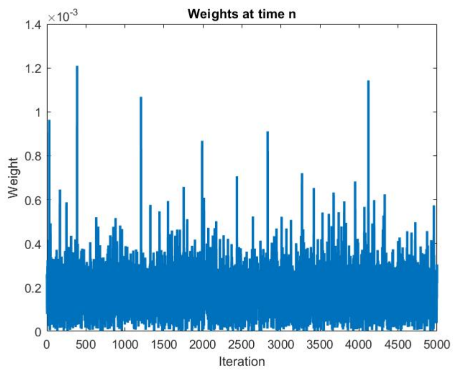


Figure 11: Estimate for the parameter θ over time by Maximum Likelihood

Furthermore, we find that the effective sample size is 3953, which is relatively high compared to $N = 5000$, so the weights are of good enough quality. As a second check we we plot the weights at time $n/2$ and time n, which are shown in Figure 12.



(a)



(b)

Figure 12: Weights at time $n/2$ and time n

We see that there are still many non-zero weights, so we do not have weight degeneracy. The estimate for θ at $t = 1000$ by Maximum Likelihood and Liu and West, together with the confidence bounds, can be found in the table below.

method	θ estimate	Lower bound	Upper bound
MLE	0.816	0.763	0.837
Liu and West	0.820	0.779	0.851

Table 4: Maximum likelihood and Liu and West estimate for the parameter θ, including corresponding confidence bounds

From the table we see that the real value, $\theta = 0.8$, is contained in the 95% confidence interval for both the ML estimate and the Liu and West estimate. The MLE is somewhat better than the Liu and West estimate, since it is closer to the true value and the confidence interval is somewhat more concentrated around the true value, but the performance of the Liu and West filter in case of an AR(1) model is very satisfactory.

6.4 Application to the diffusion model

In this section we apply Algorithm 5 to the diffusion model in equations (4) an (5), with initial weights

$$w_{t+1}^{(i)} \propto \frac{\exp(-\frac{1}{2}(y_{t+1} - H\mu_{t+1}^{(i)})^T R(m_t^{(i)})^{-1}(y_{t+1} - H\mu_{t+1}^{(i)}))}{\sqrt{\det R(m_t^{(i)})}} \ ,$$

transition distribution

$$N(F(\widehat{\theta}_{t+1}^{(i)})\widetilde{x}_t^{(i)} + C(\widehat{\theta}_{t+1}^{(i)}), Q(\widehat{\theta}_{t+1}^{(i)})) \ ,$$

and importance weights

$$w_{t+1}^{(i)} \propto \frac{\exp(-\frac{1}{2}(y_{t+1} - H\widehat{x}_{t+1}^{(i)})^T R(\widehat{\theta}_{t+1}^{(i)})^{-1}(y_{t+1} - H\widehat{x}_{t+1}^{(i)}))}{\exp(-\frac{1}{2}(y_{t+1} - H\mu_{t+1}^{(i)})^T R(\widehat{\theta}_{t+1}^{(i)})^{-1}(y_{t+1} - H\widehat{\mu}_{t+1}^{(i)}))} \ ,$$

where $\mu_{t+1}^{(i)} = F(\theta_t^{(i)})x_t^{(i)} + C(\theta_t^{(i)})$ and $\widehat{\mu}_{t+1}^{(i)} = F(\theta_t^{(i)})\widehat{x}_t^{(i)} + C(\theta_t^{(i)})$.

6.4.1 Estimating one parameter

Before estimating all parameters, we will start by estimating only one parameter, namely: $\theta = \alpha$. The true value of this parameter is 31.60. As a prior for θ we use a normal distribution with mean $\theta_0 = 31.60$ and standard deviation 10. This prior is taken just as an example, in section 6.4.2 we will look closer at how to choose the prior distributions when we estimate all parameters. We apply the Liu and West filter with $N = 5000$ particles and the estimate for α is the mean of the particle values at time $t = 40$. If we plot the estimated value for θ over time and use partially observed data, we get the following graph:

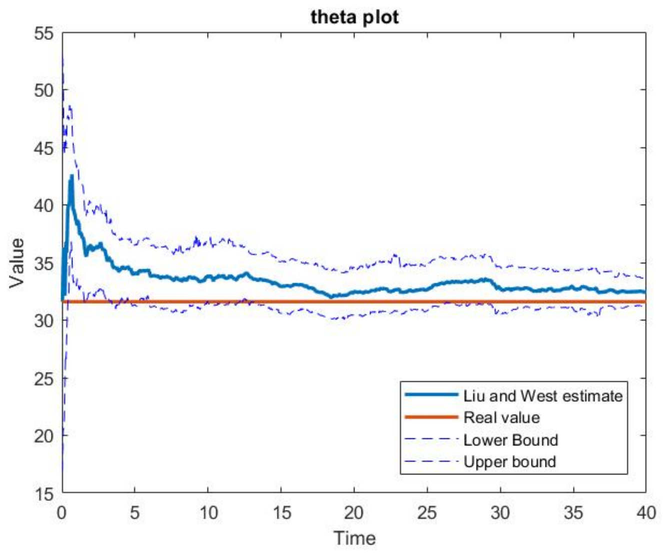


Figure 13: Estimated value for parameter α over time, by Liu and West filter

The estimated value at $t = 40$ with confidence bounds can be seen in the table below.

Parameter	True value	Estimate	Lower bound	Upper bound
α	31.60	32.43	31.28	33.54

Table 5: Parameter estimate for $\theta = \alpha$ and confidence bounds

We see that for estimating one parameter, the Liu and West filter gives good results, since the true value of the parameter is contained in the confidence interval and the estimate is close to the true value. Also, we find that $N_{ESS} = 4057$, which is relatively high, so the quality of the weights is good.

6.4.2 Choice of priors

For the priors we will take normal distributions around the real parameter values. We need to be careful in choosing the standard deviation of this distribution. We will use the condition that all parameters have to be positive and that $\lambda < \kappa$ to come up with a choice of the standard deviation. We know as a rule of thumb that for a normal distribution the chance that a random variable is less than 2σ away from the mean is approximately 95%. So, if we take σ such that $\mu - 2\sigma = 0$, then almost all values generated from $N(\mu, \sigma)$ will be positive. This gives rise to the following standard deviations for the prior distributions from Table 6. Note that choosing the priors with a mean equal to the true value has as a consequence that the initial parameter estimate, which is the unweighted average of the particles, will already be close to the true value. In section 6.4.4 we look at what happens when this is not the case.

Parameter	Standard deviation of prior
β	1.11
λ	0.05*
κ	0.08*
α	15.8
σ	0.5
σ_1	1.24
σ_2	0.38

Table 6: Choice of standard deviation of prior distributions

*For the parameters λ and κ we have the additional condition $\lambda < \kappa$. Using the same rule of thumb as before, we get that $\lambda + 2\sigma_\lambda \leq \kappa - 2\sigma_\kappa$ (i). The positivity constraint means that $\kappa - 2\sigma_\kappa \geq 0$ (ii) and $\lambda - 2\sigma_\lambda \geq 0$ (iii). One can easily verify that the choice $\sigma_\lambda = 0.05$ and $\sigma_\kappa = 0.08$ satisfies (i), (ii) and (iii).

6.4.3 Estimating all parameters

In this section we apply the Liu and West filter algorithm, see Algorithm 5, in order to estimate all 7 parameters from the state space model. We use $N = 10000$ particles and the prior distribution for the parameters from the table above. The MATLAB code for the estimation can be found in Appendix G. We will look at both partially observed and fully observed data. For a fixed time t, we have $N = 1000$ particles for each parameter. The estimate for a parameter at time t is then the average of the particle values. The results of the estimation can be seen in the tables below.

Parameter	True value	Estimate	Lower bound	Upper bound
β	2.23	2.31	2.03	2.56
λ	0.11	0.13	0.12	0.14
κ	0.37	0.39	0.36	0.42
α	31.60	33.21	29.11	36.93
σ	1	1.00	0.98	1.01
σ_1	2.49	2.61	2.35	2.87
σ_2	0.77	0.88	0.76	0.99

Table 7: Parameter estimates at time $t = 40$ for fully observed data, using $N = 10000$

Parameter	True value	Estimate	Lower bound	Upper bound
β	2.23	2.74	2.47	2.97
λ	0.11	0.12	0.10	0.13
κ	0.37	0.44	0.41	0.46
α	31.60	40.87	36.66	45.15
σ	1	1.00	0.97	1.02
σ_1	2.49	2.71	2.44	3.01
σ_2	0.77	1.16	0.84	1.52

Table 8: Parameter estimates at time $t = 40$ for partially observed data, using $N = 10000$

Furthermore, we find an effective sample size of 7423 for fully observed data, and 7544 for partially observed data. We see that the results are rather good. In the case of fully observed data, the parameter estimates are close to the true values and the true value is for all but one parameter contained in the 95% confidence interval. For partially observed data, we see that 4 of the 7 true parameter values are contained in the confidence interval for that parameter. So it is clear that for fully observed data the results are better than for partially observed data, which is what we expect. Note that these results depend on the data set which we simulated. For more accurate results we have to perform a Monte Carlo study, where in each iteration we simulate a new data set. However, due to the large computational time of the MATLAB code in Appendix G, we stick to one simulated data set. This also means that any bad estimation results may be due to this particular simulated data set. To gain some insight in how the parameter estimates evolve over time, we plot the estimates for α and β over time, using fully observed data.

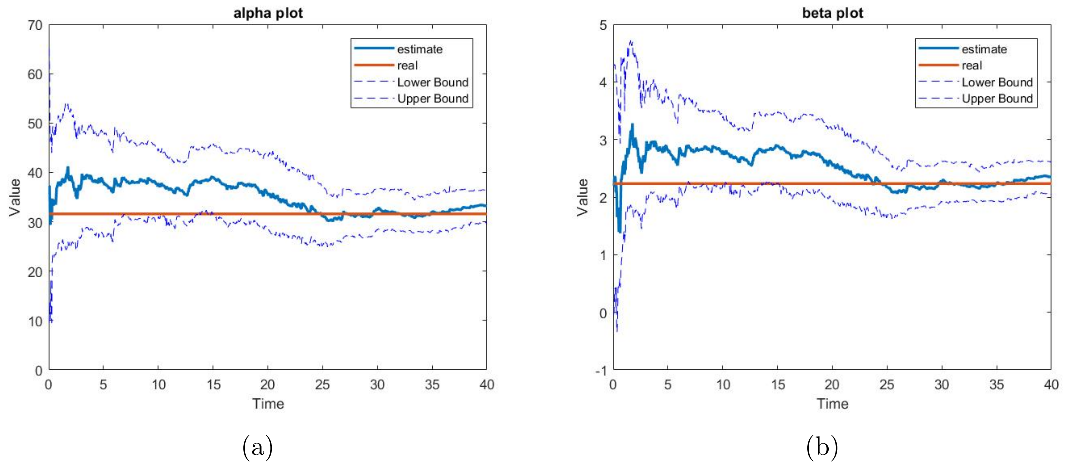


Figure 14: Parameter estimates over time for the parameter (a) α and (b) β

Due to the choice of priors, the initial estimate is already close to the true value. What we can see, however, is that the variance in the parameter estimate decreases over time. Also, for both α and β, the estimate for the parameter stays close around the true value after some time. To investigate the influence of the number of particles N, we will also look at the case where $N = 5000$. Those results can be seen in the tables below.

Parameter	True value	Estimate	Lower bound	Upper bound
β	2.23	2.35	2.17	2.53
λ	0.11	0.08	0.06	0.10
κ	0.37	0.34	0.31	0.38
α	31.60	32.83	30.01	35.39
σ	1	1.05	1.03	1.07
σ_1	2.49	2.85	2.68	3.05
σ_2	0.77	1.10	1.00	1.22

Table 9: Parameter estimates at time $t = 40$ for fully observed data, using $N = 5000$

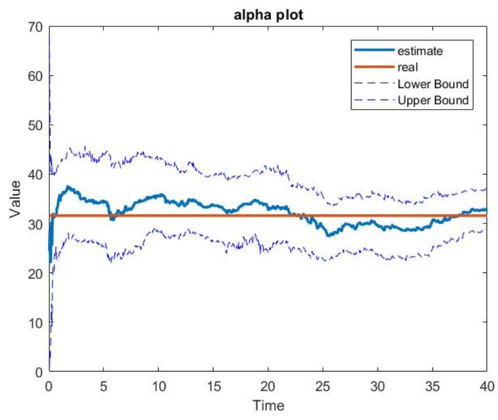
Parameter	True value	Estimate	Lower bound	Upper bound
β	2.23	3.11	2.72	3.46
λ	0.11	0.10	0.07	0.13
κ	0.37	0.37	0.32	0.42
α	31.60	45.18	38.51	51.00
σ	1	0.98	0.95	1.02
σ_1	2.49	3.08	2.91	3.24
σ_2	0.77	0.85	0.58	1.06

Table 10: Parameter estimates at time $t = 40$ for partially observed data, using $N = 5000$

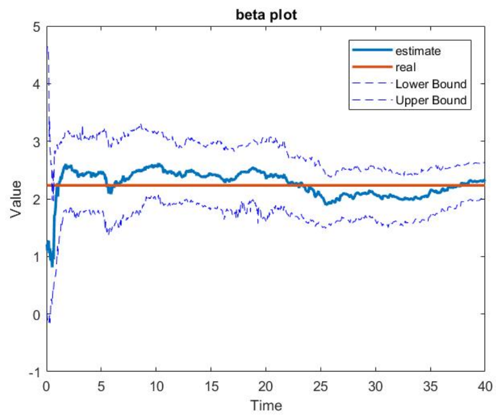
Here we find effective sample sizes of 3755 and 3724. For fully observed data we see that now in only 3 out of the 7 cases the true value is contained in the confidence interval, so the results are not as good as for $N = 10000$. For partially observed data, we see that still in 4 out of the 7 cases we have that the true value lies inside the confidence interval. However, for the other parameters, the estimate is farther away from the true value than for $N = 10000$. Furthermore, we see that there is a smaller difference between the performance of the estimation in case of fully and partially observed data. For partially observed data, for more parameters we have that the true value is contained in the confidence interval, but when this is not the case, the estimate is often farther away from the true value then for fully observed data. All in all, we have satisfactory estimation results and we see that results are better for a higher value of the number of particles.

6.4.4 Change of prior distributions

In the previous section we have estimated the parameters of the diffusion model, where we used priors concentrated around the true value of the parameter. This means that when we generate N parameter values according to this prior, the average of those values will be close to the mean of the prior distribution, which is the true value. So the initial estimate for the parameter is already close to the true value. In this section we will look at a case where some priors are not concentrated around the true value. We will change the priors for the parameters β and α. We take the exponential distribution for this. We try 2 different scenarios: one where the mean of the exponential distribution is below the true parameter value and one where it is above the true parameter value. In the first case, we take for β the mean 1 and for α we take the exponential distribution with mean 15. The other priors we keep the same as before. A graph of the estimates for α and β over time can be found in the figures below.



(a)

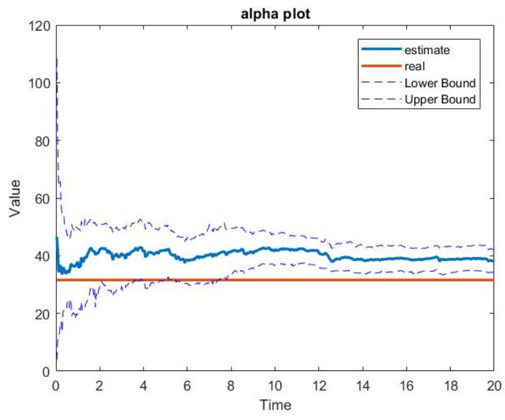


(b)

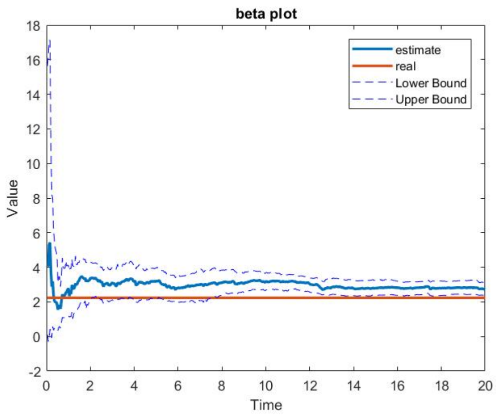
Figure 15: Parameter estimates over time for the parameter (a) α and (b) β, using exponential priors with a mean lower than the true value

We see that after a short period of time the estimate for the parameter is better than the initial estimate. Also, we see that the estimates for α and β behave similarly over time. If we go back to Figure 1, then we see that this is no surprise: the parameter α represents the inflow of contrast agent concentration and β the outflow.

The second scenario we will now look at, is the case where the exponential priors have a mean that is larger than the true parameter value. For α we take an exponential distribution with 60 as the prior mean and for β we take 4. We then find the following two graphs.



(a)



(b)

Figure 16: Parameter estimates over time for the parameter (a) α and (b) β, using exponential priors with a mean higher than the true value

We see that in this case the estimates are not as good as before. For both α and β we have an overestimate of the parameter value and the confidence interval does not include the true value. However, the estimate is still better than the initial estimate. It is again important to note that this is based only on one simulated data set. What we can see from this, is that the Liu and West filter for parameter estimation performs better for certain priors than for other priors.

7 State and parameter estimation in a non-linear model

In this section we will see if the Liu and West filter is able to estimate state variables and parameters in a non-linear AR(1) model. We will look at the following model:

$$x_{t+1} = \theta \arctan x_t + \varepsilon_t$$

$$y_{t+1} = x_{t+1} + \nu_t$$

with $\varepsilon_t, \nu_t \sim N(0,1)$. We will start by taking $\theta = 0.8$ constant, and try to estimate the state variable x_t. We use Algorithm 3 for this, with initial weights

$$g_{t+1}^{(i)} \propto \exp(-\frac{1}{2}(y_{t+1} - \mu_{t+1}^{(i)})^2) \ ,$$

transition distribution

$$N(\theta \arctan \widetilde{x}_t^{(i)}, 1) \ ,$$

importance weights

$$w_{t+1}^{(i)} \propto \frac{\exp(-\frac{1}{2}(y_{t+1} - \widehat{x}_{t+1}^{(i)})^2)}{\exp(-\frac{1}{2}(y_{t+1} - \widetilde{\mu}_{t+1}^{(i)})^2)}$$

and $\mu_{t+1}^{(i)} = \theta \arctan x_t$, $\widetilde{\mu}_{t+1}^{(i)} = \theta \arctan \widetilde{x}_t^{(i)}$, which follow from the state equation. We generate a sequence of length $n = 1000$ from the non-linear AR(1) model, using $x_0 = 0$. We generate observations from this sequence based on the observation equation. We use $N = 5000$ particles in the Liu and West filter. The state estimate at time t is calculated as the unweighted average of the particles at that time. The result for the state estimation can be seen in the graph below.

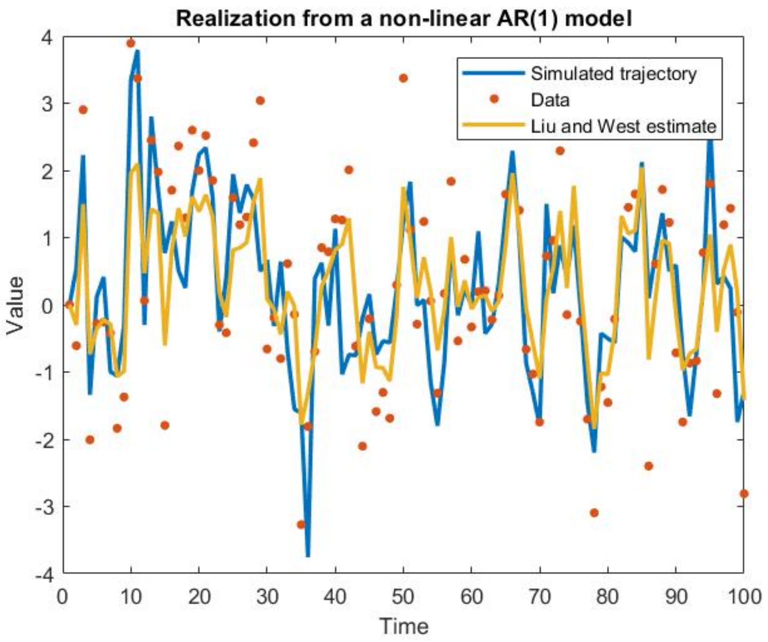


Figure 17: Simulated trajectory, observations and Liu and West estimate for the state variable

The RMSE is 0.76 and the effective sample size is 4098. We see that the Liu and West filter performs well, and is often closer to the simulated trajectory than the data points are. Next, we look at parameter estimation in the non-linear AR(1) model. We apply Algorithm 5, with initial weights

$$g_{t+1}^{(i)} \propto \exp(-\frac{1}{2}(y_{t+1} - \mu_{t+1}^{(i)})^2) \ ,$$

transition distribution

$$N(\widehat{\theta}_{t+1}^{(i)} \arctan \widetilde{x}_t^{(i)}, 1) ,$$

and importance weights

$$w_{t+1}^{(i)} \propto \frac{\exp(-\frac{1}{2}(y_{t+1} - \widehat{x}_{t+1}^{(i)})^2)}{\exp(-\frac{1}{2}(y_{t+1} - \widetilde{\mu}_{t+1}^{(i)})^2)} ,$$

where $\mu_{t+1}^{(i)} = \theta_t^{(i)} \arctan x_t^{(i)}$ and $\widetilde{\mu}_{t+1}^{(i)} = \widetilde{\theta}_t^{(i)} \arctan \widetilde{x}_t^{(i)}$. We now use $N = 10000$ particles. The MATLAB code for the non-linear AR(1) model is very similar to the one with the linear AR(1) model, which can be found in Appendix *F*. In the table below we show the Liu and West estimate, together with the confidence bounds.

Parameter	True value	Liu and West estimate	Lower bound	Upper bound
θ	0.8	0.81	0.73	0.90

Table 11: Liu and West estimate and corresponding confidence bounds

We also plot the Liu and West estimate over time, in order to see how the estimate behaves over time. This can be found in Figure 18.

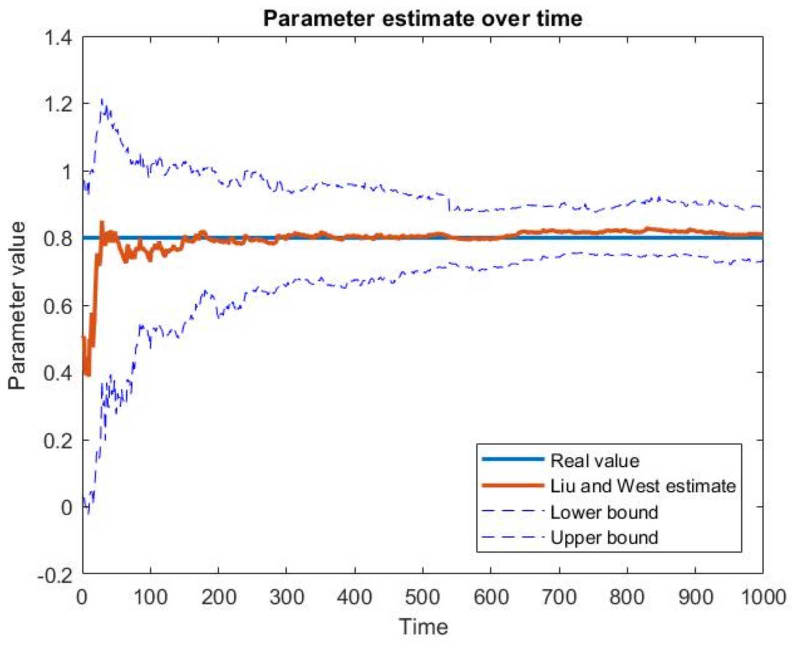


Figure 18: Liu and West estimate for the parameter θ over time

From Table 11 we can see that the estimate is very close to the true value and the true value is contained in the confidence interval. Also, from Figure 18 we can see that the estimate converges nicely to the true value and that the variance decreases over time. Thus it seems that the Liu and West filter also performs well in case of a simple non-linear model.

8 Application to real data

In this section we will test the Liu and West filter for parameter estimation on a real data set. This data is from Thomassin (2008) and contains $n = 115$ observations of the sum of the contrast agent concentrations. The observations are shown in the figure below.

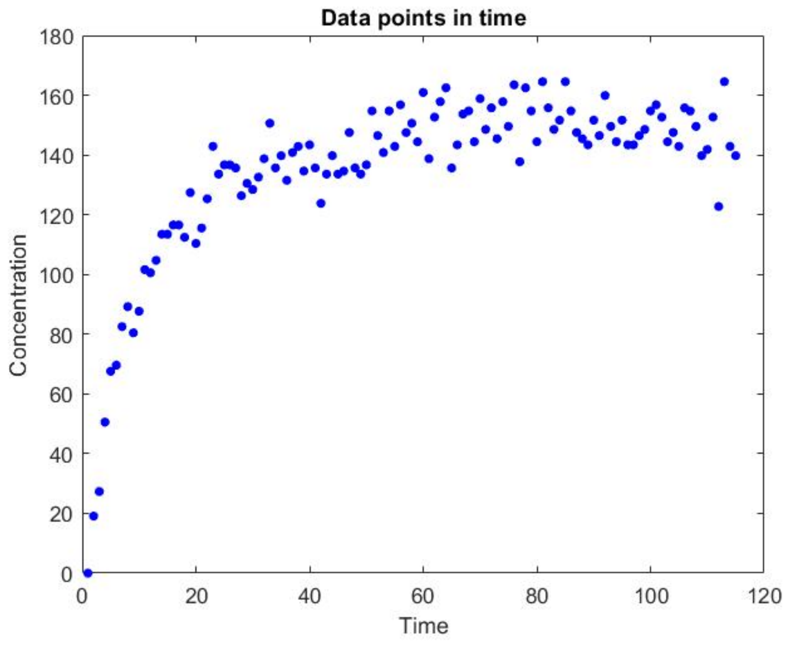


Figure 19: Real data points in time

The MATLAB code for the estimation is exactly the same as we used in section 6.4.3, only with a different data set and different priors. Since we do not know the true parameter values, we have to make sure that the prior is wide enough. We choose the exponential distribution as prior distribution. The choice of the mean of this prior distribution can be seen in the table below.

Parameter	Mean of prior distribution
β	5
λ	5
κ	5
α	300
σ	5
σ_1	5
σ_2	5

Table 12: Choice of prior distributions

For most parameters we take a mean of 5 for the exponential distribution, which we think is high enough considering previous parameter values were all well below 5. Since λ has to be smaller than κ, we first draw 2 values from the Exp(5) distribution and then assign the largest value to κ and the smallest one to λ. We repeat this until we have N values for both parameters. From Figure 19 it is clear that the real data points are much higher than the simulated data we used previously. The magnitude of those values is for a large part due to the parameter α, which is the injected contrast agent concentration. For the simulated data we had $\alpha = 31.6$, with $S(t)$ fluctuating around 20 for the most part. In Figure 19 we see that now the data points fluctuate around 150, so α is probably also a large value, which is why we take the Exp(300) as its prior. For the parameter estimation we use $N = 10000$ particles in the Liu and West filter. Estimates are calculated as the mean of the particle values. The result can be seen in the table below.

Parameter	Estimate	Lower bound	Upper bound
β	5.13	3.26	7.50
λ	2.83	0.47	5.59
κ	9.33	4.85	14.73
α	512	429	607
σ	7.15	6.31	7.99
σ_1	5.65	-0.94	12.53
σ_2	3.80	-2.15	9.83

Table 13: Parameter estimates at time $t = 115$, using $N = 10000$

Also, we find an effective sample size of 8612. In the table we see that most parameters have rather wide confidence intervals. However, in all cases the variance in the estimate is smaller than the prior variance. To see if the estimated values are accurate, we look at the state variable estimate for $S(t)$, which is also provided by the Liu and West filter. We plot this trajectory together with the data points, which can be seen in the figure below, together with the estimated trajectory for the interstitium concentration $I(t)$.

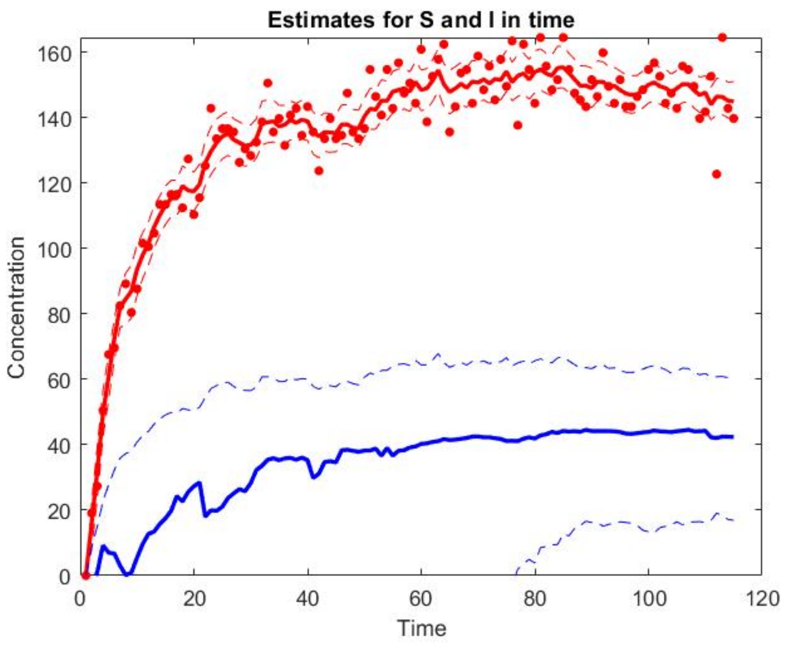


Figure 20: Liu and West estimate for the state variable S (red) and I (blue) in time, dotted lines are the confidence bounds, dots are the data points and solid lines are estimated trajectories.

From the figure we see that the estimated trajectory for $S(t)$ is close to the data points. We have estimated the standard deviation of the noise in the data, σ, as 7.15, so the result from the graph seems to be reasonable. For the interstitium concentration $I(t)$ we also get an estimated trajectory, but, as can be seen by the confidence bounds, there is much uncertainty here, with the lower bound even being negative for the most part. So it seems that for real data the Liu and West filter performs rather good when it comes to estimating $S(t)$, but for $I(t)$ there is much uncertainty in the estimate.

9 Conclusion

In this thesis we started with the state estimation in a bidimensional Ornstein-Uhlenbeck process. We have used 3 algorithms for this: the Kalman filter, the smoother algorithm and the Liu and West filter. We have seen that the smoother algorithm outperforms the other two. We expect the Kalman filter to be preferable for a linear Gaussian system, but from the Monte Carlo experiments we have seen that the Liu and West filter also performs well and is competitive with the Kalman filter.

Furthermore, we have used the Liu and West filter for parameter estimation. In case of the linear AR(1) model, results were very satisfactory and close to results of Maximum Likelihood. Next, we applied the Liu and West filter to the bidimensional Ornstein-Uhlenbeck process. We have simulated 1000 observations from this model. When we only estimated the parameter α, we obtained good results where the true value was contained in the confidence interval. For the estimation of all parameters, we used $N = 10000$ particles and we first looked at priors centered around the true value. For fully observed data, we have seen that the estimation results were rather good: for all but one parameter the true values was contained in the confidence interval. For partially observed data the results were slightly worse: for 4 out of the 7 parameters the true value was contained in the confidence interval. We have also seen that the choice of the number of particles, N, is important, since for $N = 5000$ the results became worse, especially for the fully observed data, because in that case only for 3 out of the 7 parameters the true value was contained in the confidence interval. Note that we only used one simulated data set, which may influence estimation results. We have also seen that the choice of prior distribution is important: certain priors for α and β led to better results than others.

Next, we have applied the Liu and West filter to a non-linear model. For both state and parameter estimation the performance was good, with a low RMSE and a confidence interval that contained the true parameter value. Finally, we applied the Liu and West filter to real data, which also seemed to give reasonable results: the estimated trajectory for $S(t)$ seemed reasonable considering the data, but for $I(t)$ there was much uncertainty in the estimate.

Throughout this thesis we have assumed that the parameter $\alpha(t)$, which describes the injected contrast agent concentration, is constant. This assumption is rather unnatural. For further research it might be interesting to look at the case where α is not constant. We can for example take a parametrization of a function with 2 unknown parameters. This function has to initially increase (injection of the contrast agent), then become constant (reaching equilibrium) and then decrease again (wash out of contrast agent). Another option is to use an Ornstein-Uhlenbeck process for $\alpha(t)$ itself: $d\alpha_t = (\lambda - \mu\alpha_t)dt + \sigma dW_t$, with $\lambda, \mu, \sigma > 0$. Since this is a mean-reverting model, $\alpha(t)$ will first increase due to the drift term, stay more or less constant for a while, and eventually decrease again until it reaches some mean value.

10 References

Bijma, F., Jonker, M. and Van der Vaart, A. (2017) *An introduction to Mathematical Statistics*, Amsterdam University Press.

Blasques, F. and Koopman, S. (2019), *ARMA models: parameter estimation, forecasting and impulse response functions* [lecture notes], Vrije Universiteit Amsterdam.

Carvalho, C., Johannes, M., Lopes, H. and Polson, N. (2010) Particle Learning and Smoothing *Statistical Science*, Vol. 25: 88-106, DOI: 10.1214/10-STS325.

Favetto, B. and Samson, A. (2010), Parameter estimation for a Bidimensional Partially Observed Ornstein-Uhlenbeck Process with Biological Application, *Scandinavian Journal of Statistics*, Vol. 37: 200-220.

Haugh, M. (2017), *Simulating Stochastic Differential Equations* [lecture notes]. Columbia University, New York City, NY. Retrieved from http://www.columbia.edu/ mh2078/MonteCarlo/MCS-SDEs.pdf.

Jászai, J. and Schmidt, M. (2019), Trends and Challenges in Tumor Anti-Angiogenic Therapies, *Cells*, DOI: 10.3390/cells8091102

Johansen, A. and Evers, L. (2007), *Monte Carlo methods* [lecture notes], University of Bristol, England.

Kim, Y. and Bang, H. (2018), Introduction to Kalman Filter and Its Applications, *Introduction and Implementations of the Kalman Filter*, DOI: 10.5772/intechopen.80600.

Klebaner, F. (2005), *Introduction to Stochastic Calculus With Applications*, Imperial College Press.

Liptser, R., Shiryaev, A. (2001), *Statistics of Random Processes*, Second Edition, Springer-Verlag Berlin Heidelberg.

Liu, J. , West, M. (2001) Combined parameter and state estimation in simulation-based filtering, *Sequential Monte Carlo Methods in Practice*, Springer, New York, NY.

Majumdar, R. and Majumdar, S. (2017) On the Conditional Distribution of a Multivariate Normal given a Transformation - the Linear Case, *Heliyon*, Vol.5, DOI: 10.1016/j.heliyon.2019.e01136

Miller, J. (2016), *Lecture Notes on Advanced Stochastic Modeling* [lecture notes], Duke University, Durham, NC.

Nemeth, C., Fearnhead, P. and Mihaylova, L. (2015), Sequential Monte Carlo Methods for State and Parameter Estimation in Abruptly Changing Environments, *IEEE Transactions on Signal Processing*, Vol. 62.

Oksendal, B. (2000), *Stochastic differential equations: an introduction with applications*, Fifth Edition, Springer-Verlag Berlin Heidelberg.

Sarkka, S. (2011), *Optimal Smoothing* [Powerpoint slides], Aalto University, Espoo, Finland. Retrieved from https://users.aalto.fi/ ssarkka/course-k2011/pdf/handout7.pdf.

Thomassin, I. (2008), Dynamic contrast-enhanced magnetic resonance imaging: A useful tool for characterizing ovarian epithelial tumors, *Journal of Magnetic Resonance Imaging*, Vol. 28: 111-120.

Wiersema, U. (2008), *Brownian motion calculus*, John Wiley & Sons Ltd.

Zivot, E. (2005), *Estimation of ARMA models* [lecture notes], University of Washington.

11 Appendix

Appendix A: Transformation of the parameters

```
function param = convparam(theta, h)
    mu_1 = log(theta(1))/h ;
    mu_2 = log(theta(2))/h ;
    theta3_hat = (theta(3)*2*mu_1*(mu_2-mu_1)^2)/(exp(2*mu_1*h)-1) ;
    theta4_hat = (theta(4)*2*mu_2*(mu_2-mu_1)^2)/(exp(2*mu_2*h)-1) ;
    theta5_hat = (theta(5)*(mu_1+mu_2)*(mu_2-mu_1)^2)/(exp( (mu_1+mu_2)*h)-1) ;

    fun = @root3d ;
    x0 = [0.5, 0.5, 0.5] ;
    x = fsolve(fun, x0) ;
    beta = x(1) ;
    sigma_1 = sqrt(x(2)) ;
    sigma_2 = sqrt(x(3)) ;

    kappa = -mu_1 - mu_2 - beta ;
    lambda = -mu_1*mu_2/beta + kappa ;

    d = (beta-kappa)^2+4*beta*lambda ;
    H = [1 1] ;
    D = [mu_1 0; 0 mu_2] ;
    P = [1 1 ;(beta-kappa-sqrt(d))/(2*beta) (beta-kappa+sqrt(d))/(2*beta) ] ;
    F = [1 ; 0] ;
    c = - ( (H/D) / P) *F ;
    alpha = theta(6)/c ;

    param(1) = beta ;
    param(2) = sigma_1 ;
    param(3) = sigma_2 ;
    param(4) = kappa ;
    param(5) = lambda ;
    param(6) = alpha ;

    function F = root3d(x)
        b = x(1) ;
        s1_sq = x(2) ;
        s2_sq = x(3) ;

        F(1) = (mu_2+b)^2*s1_sq+mu_2^2*s2_sq-theta3_hat ;
        F(2) = (mu_1+b)^2*s1_sq+mu_1^2*s2_sq-theta4_hat ;
        F(3) = s1_sq*b^2+s1_sq*(mu_1+mu_2)*b +(s1_sq+s2_sq)*mu_1*mu_2 + theta5_hat ;

    end
end
```

Appendix B: Discretization of the SDE

```
n = 1000 ; % number of grid points
h = 0.04 ; % step size

P = zeros(1,n) ;
I = zeros(1,n) ;

rng(0) ; % use default random number generator with seed 0
Z = normrnd(0,1,[2,n-1]) ;

for k = 2:n
    P(k) = P(k-1) + h*(alpha-(lambda+beta)*P(k-1)+(kappa-lambda)*I(k-1))...
    +sigma_1*sqrt(h)*Z(1,k-1) ;
    I(k) = I(k-1) + h*(lambda*P(k-1)-(kappa-lambda)*I(k-1))...
    +sigma_2*sqrt(h)*Z(2,k-1) ;
end
```

Appendix C: Kalman filter algorithm

```
%% loading the simulated data
data = load('data.mat') ;
S_sim = data.S_sim ;
I_sim = data.I_sim ;

%% loading physical trajectory
data2 = load('phys.mat') ;
I_phys = data2.I ;
S_phys = data2.S ;

%% Setting up Kalman filter
F0 = [alpha ; 0 ] ;
d = (beta-kappa)^2 + 4*beta*lambda ;
P0 = [1 1 ; (beta-kappa-sqrt(d))/(2*beta) (beta-kappa+sqrt(d))/(2*beta) ] ;
mu1 = .5*(-beta-kappa-sqrt(d)) ;
mu2 = .5*(-beta-kappa+sqrt(d)) ;
D = [mu1 0 ; 0 mu2] ;
S = [sigma1 sigma2 ; 0 sigma2 ] ;
C = (P0\S)*(P0\S)' ;
R_i = [(exp(2*mu1*h)-1)/(2*mu1)*C(1,1),(exp(mu1*h+mu2*h)-1)/(mu1+mu2)*C(1,2);...
    (exp(mu1*h+mu2*h)-1)/(mu1+mu2)*C(2,1),(exp(2*mu2*h)-1)/(2*mu2)*C(2,2)];

F = expm(D*h) ;
Q = R_i ;
B = D\( (expm(D*h) - eye(2))* (P0\F0) ) ;
H = P0 ; % or H = [1,1] for partially observed data
R = sigma^2*eye(2) ; % or R = sigma^2 for partially observed data
mu0 = [0;0] ;
V0 = [0 0; 0 0] ;

%% Kalman algorithm
z = zeros(2, n) ;
z(:,1) = mu0 ;
P = zeros(2, 2*n) ;
P(:, 1:2) = V0 ;
for j = 2:n
    % prediction step
    z(:,j) = F*z(:,j-1) + B ;
    P(:,2*j-1:2*j) = F*P(:,2*j-3:2*j-2)*F' + Q ;
    % update step
    y = [S_sim(j) ; I_sim(j)] ;
    res = y - H*z(:,j) ;
    S = R + H*P(:,2*j-1:2*j)*H' ;
    K = (P(:,2*j-1:2*j)*H') / S ;
    z(:,j) = z(:,j) + K*res ;
    P(:,2*j-1:2*j) = (eye(2)-K*H)*P(:,2*j-1:2*j) ;
end

% transforming back
for j = 1:n
   z(:,j) = P0*z(:,j) ;
end

S_kal = z(1,:) ;
I_kal = z(2,:) ;
```

Appendix D: RTS Smoother algorithm

```
z_smooth = zeros(2,n) ;
P_smooth = zeros(2,2*n) ;

% initialize
z_smooth(:,end) = z(:,end) ;
P_smooth(:, end-1:end) = P(:, end-1:end) ;
% z and P are from the Kalman filter

% iterate
for j = flip(1:n-1)
    z_min = F*z(:,j) + C ;
    P_min = F*P(:, 2*j-1:2*j)*F' + Q ;
    G = (P(:,2*j-1:2*j)*F')/P_min ;
    z_smooth(:,j) = z(:,j) + G*(z_smooth(:,j+1) - z_min) ;
    P_smooth(:, 2*j-1:2*j) = P(:,2*j-1:2*j) + G*(P_smooth(:, 2*j:2*j+1) - P_min)*G' ;
end

% transforming back
for j = 1:n
   z_smooth(:,j) = P0*z_smooth(:,j) ;
end

S_smooth = z_smooth(1,:) ;
I_smooth = z_smooth(2,:) ;
```

Appendix E: Liu and West filter with constant parameters

```
N = 100 ; % #sample points
w = zeros(n,N) ;
w(1,:) = repmat(1/N, [1,N]) ; % equal weights at time t=0
x = zeros(2*n,N) ;

for t=1:n-1
    % resample
    for i = 1:N
        mu = F*x(2*t-1:2*t, i)+B ;
        y = [S_sim(j,t+1) ; I_sim(j,t+1) ] ;
        w(t+1,i) = exp(-.5*(y-H*mu)'/(R)*(y-H*mu)) ;
    end
    w(t+1,:) = w(t+1,:)/sum(w(t+1,:)) ;
    x(2*(t)-1:2*(t), :) = datasample(x(2*(t)-1:2*(t), :), N, 2, 'Weights', w(t+1,:)) ;

    % propagate x
    for i = 1:N
        mu = F*x(2*(t)-1:2*(t), i)+B ;
        x(2*(t+1)-1:2*(t+1), i) = mvnrnd(mu, Q) ;
    end

    % resample again
    for i = 1:N
        y = [S_sim(j,t+1); I_sim(j,t+1) ] ;
        mu = F*x(2*(t)-1:2*(t), i)+B ;
        w(t+1,i) = exp(.5*(y-H*mu)'/(R)*(y-H*mu) -.5*(y-H*x(2*(t+1)-1:2*(t+1), i))'/(R)...
                *(y-H*x(2*(t+1)-1:2*(t+1), i)) ) ;
    end
    w(t+1,:) = w(t+1,:)/sum(w(t+1,:)) ;
    x(2*(t)-1:2*(t), :) = datasample(x(2*(t)-1:2*(t), :), N, 2, 'Weights', w(t+1,:)) ;
end
```

Appendix F: Liu and West filter applied to AR(1) model

```
theta_real = 0.8 ;
n = 1000 ; % t = 1,...,n
N = 5000 ; % #sample points
a = 0.995 ; % shrinkage coefficient
struct = load('AR1') ;
y = struct.x_data ;

% initialisation
x = zeros(n,N) ;
x(1,:) = 0 ;
w = zeros(n,N) ;
w(1,:) = 1/N ;
theta = zeros(n,N) ;
theta(1,:) = unifrnd(0,1,[1,N]) ;
x_est = zeros(1,n) ;
x_est(1) = 0 ;
theta_est = zeros(1,n) ;
theta_est(1) = mean(theta(1,:)) ;

for t = 1:n-1
   % calculate initial weights
   g = zeros(1,N) ;
   for i = 1:N
      % a priori estimates
      mu = theta(t,i)*x(t,i) ;
      g(i) = exp(-.5*(y(t+1)-mu)^2) ;
   end
   g = g/sum(g) ;

   % resample for first time
   resamp = datasample(cat(1, x(t,:), theta(t,:)) , N,2, 'Weights', g) ;
   x(t,:) = resamp(1,:) ;
   theta(t,:) = resamp(2,:) ;

   % propagate theta
   theta_bar = sum(1/N*theta(t,:)) ;
   V = sum(1/N*(theta(t,:) - theta_bar).^2) ;
   for i = 1:N
      m = a*theta(t,i)+(1-a)*theta_bar ;
      theta(t+1,i) = normrnd(m, sqrt( (1-a^2)*V)) ;
   end

   % propagate x
   for i = 1:N
      x(t+1,i) = normrnd(theta(t+1,i)*x(t,i), 1) ;
   end

   % calculate weights
   for i = 1:N
      mu = theta(t+1,i)*x(t,i) ;
      w(t+1,i) = exp(-.5*(y(t+1)-x(t+1,i))^2)/exp(-.5*(y(t+1) - mu)^2)  ;
   end
   w(t+1,:) = w(t+1,:)/sum(w(t+1,:)) ;

   % resample for second time
   resamp = datasample(cat(1, x(t+1,:), theta(t+1,:)) , N,2, 'Weights', w(t+1,:)) ;
   x(t+1,:) = resamp(1,:) ;
   theta(t+1,:) = resamp(2,:) ;

   % calculate characteristic of interest
   x_est(t+1) = mean(x(t+1,:))  ;
   theta_est(t+1) = mean(theta(t+1,:)) ;
end
```

Appendix G: Liu and West filter for estimating all parameters

```
%% initialisation
theta0 = [ beta, sigma1, sigma2, kappa, lambda, alpha sigma] ;
part = 0 ; % 0 for fully observed, 1 for partially observed
N = 10000 ; % # number of sample points
w = zeros(n,N) ;
w(1,:) = repmat(1/N, [1,N]) ; % equal weights at time t=0
x = zeros(2*n,N) ;
theta = zeros(7*n,N) ;
a = 0.995 ;

% initial estimates for theta
theta(1,:) = normrnd(theta0(1), 1.11, [1,N]) ; % beta
theta(2,:) = normrnd(theta0(2), 1.24, [1,N]) ; % sigma1
theta(3,:) = normrnd(theta0(3), 0.38, [1,N]) ; % sigma2
theta(4,:) = normrnd(theta0(4), 0.08, [1,N]) ; % kappa
theta(5,:) = normrnd(theta0(5), 0.05, [1,N]) ; % lambda
theta(6,:) = normrnd(theta0(6), 15.8, [1,N]) ; % alpha
theta(7,:) = normrnd(theta0(7), 0.5, [1,N]) ; % sigma

%% parameter estimation
for t=1:n-1
    % resample
    theta_bar = zeros(7,1) ;
    for i = 1:N
        theta_bar = theta_bar + 1/N*theta(7*t-6:7*t,i) ;
    end

    for i = 1:N
        m = a*theta(7*t-6:7*t,i)+(1-a)*theta_bar ;
        output = CalcMatrix(m, part) ;
        F = output(:, 1:2) ;
        Q = output(:, 3:4) ;
        B = output(:, 5) ;
        if part == 0
            H = output(:, 6:7) ;
            H = reshape(H, [2,2]) ;
            R = m(7)^2*eye(2) ;
            y = [S_sim(t+1); I_sim(t+1) ] ;
        else
            H = [1,1] ;
            R = m(7)^2 ;
            y = S_sim(t+1) ;
        end
        mu = F*x(2*t-1:2*t, i)+B ;
        w(t+1,i) = 1/sqrt(det(R))*exp(-.5*(y-H*mu)'/(R)*(y-H*mu) ) ;
    end
    w(t+1,:) = w(t+1,:)/sum(w(t+1,:)) ;
    list = cat(1,x(2*t-1:2*t, :),theta(7*t-6:7*t,:)) ;
    list = datasample(list, N, 2, 'Weights', w(t+1,:)) ;
    x(2*t-1:2*t, :) = list(1:2,:) ;
    theta(7*t-6:7*t,:) = list(3:9,:) ;

    % propagate theta
    theta_bar = zeros(7,1) ;
    for i = 1:N
        theta_bar = theta_bar + 1/N*theta(7*t-6:7*t,i) ;
    end

    V = zeros(7,7) ;
    for i = 1:N
        V = V + 1/N*(theta(7*t-6:7*t, i)-theta_bar)*(theta(7*t-6:7*t,i)-theta_bar)' ;
    end
    for i = 1:N
        m = a*theta(7*t-6:7*t,i)+(1-a)*theta_bar ;
        theta(7*(t+1)-6:7*(t+1),i) = mvnrnd(m, (1-a^2)*V ) ;
    end

    % propagate x
    for i = 1:N
        output = CalcMatrix(theta(7*(t+1)-6:7*(t+1),i),part) ;
        F = output(:, 1:2) ;
        Q = output(:, 3:4) ;
        B = output(:, 5) ;
        mu = F*x(2*t-1:2*t, i)+B ;
        x(2*(t+1)-1:2*(t+1), i) = mvnrnd(mu, Q ) ;
    end

    % resample again
    theta_bar = zeros(7,1) ;
    for i = 1:N
        theta_bar = theta_bar + 1/N*theta(7*t-6:7*t,i) ;
    end

    for i = 1:N
        m = a*theta(7*t-6:7*t,i)+(1-a)*theta_bar ;
        output = CalcMatrix(m,part) ;
        F = output(:, 1:2) ;
        Q = output(:, 3:4) ;
        B = output(:, 5) ;
        if part == 0
            H = output(:, 6:7) ;
            H = reshape(H, [2,2]) ;
            R = m(7)^2*eye(2) ;
            y = [S_sim(t+1); I_sim(t+1) ] ;
        else
            H = [1,1] ;
            R = m(7)^2 ;
            y = S_sim(t+1) ;
        end
        mu = F*x(2*t-1:2*t, i)+B ;
        est_dens = 1/(sqrt(det(R)))* exp(-.5*(y-H*mu)'/(R)*(y-H*mu) ) ;


        output = CalcMatrix(theta(7*t-6:7*t,i),part) ;
        vec = theta(7*t-6:7*t,i) ;
        if part == 0
            H = output(:, 6:7) ;
            H = reshape(H, [2,2]) ;
            R = vec(7)^2*eye(2) ;
        else
            H = [1,1] ;
            R = vec(7)^2 ;
        end

        target_dens = 1/( sqrt(det(R)) ) * ...
            exp(- .5*(y-H*x(2*(t+1)-1:2*(t+1), i))'/R*(y-H*x(2*(t+1)-1:2*(t+1), i))) ;

        w(t+1,i) = target_dens/est_dens ;

    end
    w(t+1,:) = w(t+1,:)/sum(w(t+1,:)) ;
    list = cat(1,x(2*(t+1)-1:2*(t+1), :),theta(7*(t+1)-6:7*(t+1),:)) ;
    list = datasample(list, N, 2, 'Weights', w(t+1,:)) ;
    x(2*(t+1)-1:2*(t+1), :) = list(1:2,:) ;
    theta(7*(t+1)-6:7*(t+1),:) = list(3:9,:) ;
end
```

```
% transforming back
d = (beta-kappa)^2 + 4*beta*lambda ;
P0 = [1, 1 ; (beta-kappa-sqrt(d))/(2*beta), (beta-kappa+sqrt(d))/(2*beta) ] ;
for j = 1:n
    for i = 1:N
        x(2*j-1:2*j, i) = P0*x(2*j-1:2*j,i) ;
    end
end

theta_est = zeros(7,n) ;
I_est = zeros(1,n) ;
S_est = zeros(1,n) ;
for i = 1:n
    theta_est(1:7,i) = zeros(7,1) ;
    for j = 1:N
        theta_est(1:7,i) = theta_est(1:7,i) +1/N*theta(7*i-6:7*i,j) ;
    end
    state = x(2*i-1:2*i, :) ;
    I_est(i) = sum(1/N*state(2,:)) ;
    S_est(i) = sum(1/N*state(1,:)) ;
end

function output = CalcMatrix(theta,part)
    for i = 1:length(theta)
        if theta(i) < 0 % theta's should be positive
            theta(i) = 0.001 ;
        end
    end

    if theta(4) < theta(5)
        theta(4) = theta(5)+0.001 ; % kappa should be larger than lambda
    end

    beta = theta(1) ;
    sigma1 = theta(2) ;
    sigma2 = theta(3) ;
    kappa = theta(4) ;
    lambda = theta(5) ;
    alpha = theta(6) ;
    h = 0.04 ; % step size

    F0 = [alpha ; 0 ] ;
    d = (beta-kappa)^2 + 4*beta*lambda ;
    P0 = [1, 1 ; (beta-kappa-sqrt(d))/(2*beta), (beta-kappa+sqrt(d))/(2*beta) ] ;
    mu1 = .5*(-beta-kappa-sqrt(d)) ;
    mu2 = .5*(-beta-kappa+sqrt(d)) ;
    D = [mu1 0 ; 0 mu2] ;
    S = [sigma1 sigma2 ; 0 sigma2 ] ;
    C = (P0\S)*(P0\S)' ;
    R_i = [(exp(2*mu1*h)-1)/(2*mu1)*C(1,1) (exp(mu1*h+mu2*h)-1)/(mu1+mu2)*C(1,2) ; ...
    (exp(mu1*h+mu2*h)-1)/(mu1+mu2)*C(2,1)  (exp(2*mu2*h)-1)/(2*mu2)*C(2,2)] ;
    F = expm(D*h) ;
    Q = R_i ;
    B = D\( (expm(D*h) - eye(2))* (P0\F0) ) ;

    output = zeros(2,7) ;
    output(:, 1:2) = F ;
    output(:, 3:4) = Q ;
    output(:, 5) = B ;

    if part == 0
        output(:,6:7) = P0 ;
    end
end
```