Delft University of Technology

Tidal Wave

Delft University of Technology, Delft, South Holland, 2628CD

Antoine Pomari (5369614)
Amadeo Villar (5377447)

June 9, 2021

Submission date:	June 10, 2021
Course:	WI4231
Group:	-
Supervisors:	Dr. Arnold Heemink [1] Dr. Martin Verlaan [2]

[1] Assistant Professor, Faculty of Electrical Engineering, Mathematics and Computer Science, TU Delft.
[2] Assistant Professor, Faculty of Electrical Engineering, Mathematics and Computer Science, TU Delft.

Contents

Introduction

The Western Scheldt estuary in the south-western part of the Netherlands contains an important shipping channel to Antwerp harbour. At the same time, it is surrounded by low polder areas, that potentially are at risk of flooding during severe storms in combination with spring tides. In this project we will model the hydrodynamics of these tides and storm surges with the linearized shallow water equations. Along the coast of the Western Scheldt the waterlevel is measured in a number of tide gages; we only consider a limited number of them. The data provided with the project files are the actual observed values as made available by Rijkswaterstaat[3]. The period studied in is during the storm Xaver of December 6 2013, also known as the Sinterklaasstorm in the Netherlands[4]. The peak waterlevel in Vlissingen was second highest one over the last century, only second to the storm of 1953. The description of this project is accompanied by a zip file with measurements and program files in Matlab, Python and Julia. To carry out this project we have selected the `Matlab` Software.

Question 1

The linearized, one-dimensional equations for the "shallow water problem" read as:

$$\begin{cases} \frac{\partial h}{\partial t} + D\frac{\partial u}{\partial x} = 0, \\ \frac{\partial u}{\partial t} + g\frac{\partial h}{\partial x} + fu = 0, \end{cases} \tag{1}$$

where:

- $u = u(x,t) :=$ velocity of the wave $[\mathrm{m\,s^{-1}}]$.
- $h = h(x,t) :=$ relative height of the waterfront with respect to some chosen reference depth of the bottom ground [m].
- $D :=$ is precisely the chosen reference depth [m], taken as constant.
- $g :=$ gravitational acceleration coefficient $[\mathrm{m\,s^{-2}}]$.
- $f :=$ friction coefficient.

To show that these equations can be written as a wave equation, we can first of all compute the derivative of the second equation in (1) with respect to x. What we obtain is:

$$\begin{aligned} \frac{\partial}{\partial x}\Big(\frac{\partial u}{\partial t} + g\frac{\partial h}{\partial x} + fu = 0\Big) = 0, \\ \frac{\partial^2 u}{\partial x \partial t} + g\frac{\partial^2 h}{\partial x^2} + f\frac{\partial u}{\partial x} = 0. \end{aligned}$$

If we neglect the friction term f, we are left with:

$$\frac{\partial^2 u}{\partial x \partial t} = -g\frac{\partial^2 h}{\partial x^2},$$

[3] https://waterinfo.rws.nl/#!/kaart/waterhoogte-t-o-v-nap/
[4] https://en.wikipedia.org/wiki/Cyclone_Xaver

Lastly, applying Fubini's theorem to interchange the derivatives and using the first of (1) we obtain the desired wave equation:

$$\frac{\partial^2 h}{\partial t^2} = gD\frac{\partial^2 h}{\partial x^2}. \tag{2}$$

It is also interesting to notice that differentiating the first equation in (1) and using the previous procedure, we can obtain a wave-type equation also for the velocity u:

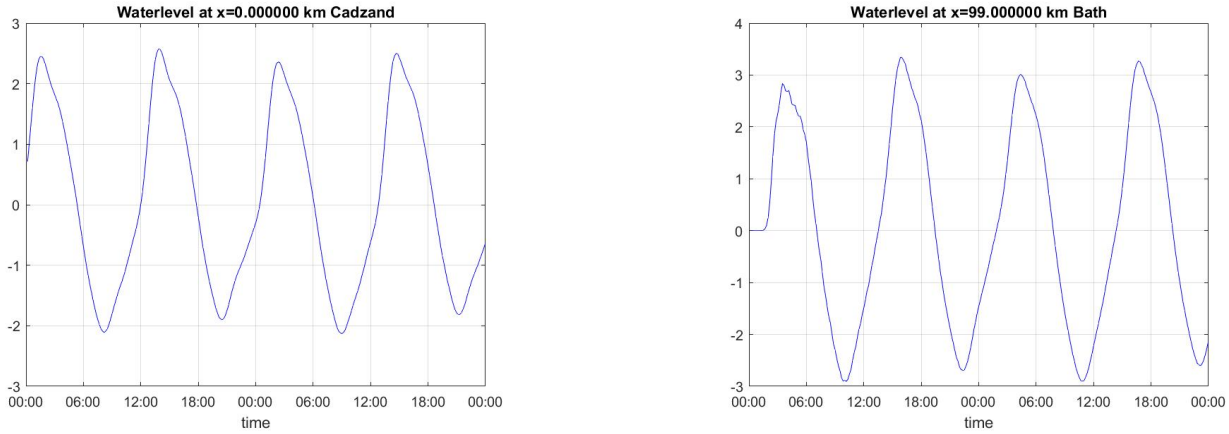
$$\frac{\partial^2 u}{\partial t^2} = gD\frac{\partial^2 u}{\partial x^2}. \tag{3}$$

Now we turn our attention back to (2). The general form of the wave equation is $\partial^2\phi/\partial t^2 = c^2\Delta\phi$. The solution to this general equation are the so-called waves, and they translate inside the domain with characteristic speed c. Hence we conclude that the characteristic speed of "our" waves is:

$$c := \sqrt{gD} \tag{4}$$

Substituting the values used in the model code of the problem yields $\boxed{\mathbf{c = 14}\ m.s^{-1}}$.

We can also calculate the propagation velocity in the numerical model. To do so, we run our model using the programme: `wave1d` and we plot the graphs of the predicted water level in Cadzand and Bath, the first and last cities respectively, as shown in Figure 1.



(a) Cadzand (b) Bath

Figure 1: Waterlevel computed by our model vs Time for the cities of Cadzand and Bath

To compute the propagation velocity, we first calculate the differences between the times at which the wave reaches consecutive maxima and consecutive minima in both cities. We then divide the distance between the 2 cities ($d = 99000$m) between these times, and we average to obtain the numerical propagation speed: $\boxed{\mathbf{c = 14.375}\ m.s^{-1}}$.

It turns out that the numerical velocity is slightly greater that the theoretical characteristic speed. One possible reason behind this is the fact that we are measuring maxima in 10 minute windows, so the accuracy of these measurements is not optimal. Reducing the time windows would most likely improve the numerical results. Furthermore, in the numerical model we are also considering the frictional term f, however this fact would diminish this value.

Question 2

We have seen in the previous section that our model is relatively simple and the numerical results obtained might be not completely accurate. A common approach to improve the output of a model in comparison to observations is to improve the model itself. In this section we overview some ideas which could help enhance the accuracy of our model. We divide these ideas into 3 different groups:

- Physics:

  First of all, we are considering only one dimension, when it is clear from the map that the width of the estuary is large enough not to be neglected, so a 2D model would be more appropriate [1]. Because we use a one-dimensional model we can not take into account the Coriolis effect[5]. Moreover, the estuary is not a straight and homogeneous line (it has curves and small portions of land along its length). This fact can generate wave reflection phenomena that our model is not considering. Thirdly, we are assuming that the depth D is constant throughout all the estuary, a simplification that could have consequences in our model. It is also worth mentioning that we are neglecting second order terms when considering the geostrophic balance setting.

- Numerical Analysis:

  To solve our system of partial differential equations (1) numerically, we have to discretise in both time and space. For time integration we use the 1^{st} order Euler scheme:

  $$\frac{dh(t,x)}{dt} = \frac{h(t+dt,x) - h(t,x)}{dt}, \tag{5}$$

  while for space discretization we implement the 2^{nd} order Trapezium scheme:

  $$\frac{dh(t,x)}{dx} = \frac{h(t,x+dx) - h(t,x-dx)}{2dx}. \tag{6}$$

  We can check using Taylor expansions, that the Euler scheme is $O(dt)$ accurate and the Trapezium scheme is $O(dx^2)$ accurate. In order to obtain more accurate approximations, we could have used higher order methods. For example the Crack-Nicolson method[6], which has order 2 accuracy in time, and order 4 in accuracy in space. Moreover Crank-Nicolson is unconditionally stable independently of the value of the Courant number. Therefore, we can choose the time and space discretization steps as small as we want, and the choice will not upset the convergence of the method. Finally, one note about the steps used in this model: they are $dt = 10$ minutes and $dx \approx 1$km; choosing smaller steps would also lead to more accurate results.

- Input data:

  On one hand, the introduction of more input parameters in our model would most likely improve the results. We are only studying the velocity and the waterlevel, and the correlations between these two entities. It would be interesting to take into account other variables, such as the wind in this estuary or the water temperature (so the seasonal dependence). Other examples of parameters that we could take into account for our model and that are available in [2] are: air temperature, astronomical tide, currents, water drainage and salinity of the water.

  On the other hand it would be interesting to tune with more precision some of the parameters already considered in the model. For example, the value of the depth D influences the characteristic speed of the solutions to our wave equation. Therefore it would be interesting to model D with more precision. As a first step, we could think of D as a piece-wise constant function, rather than as a constant number throughout all the estuary. Another way of better tuning the given parameters would be to have more precise measurements of the distances between cities (precision in the order of meters rather than kilometers).

[5] https://en.wikipedia.org/wiki/Coriolis_force
[6] https://en.wikipedia.org/wiki/Crank%E2%80%93Nicolson_method

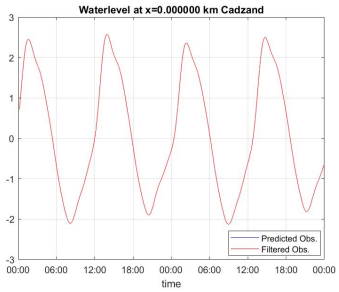
Question 3

In this section we will focus on the study of model uncertainty. That is, we do not consider the measurement update equations and we study the uncertainty in the time-update equation of our model:

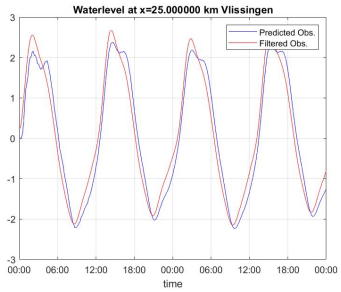
$$x_{k+1} = \underbrace{Mx_k}_{\text{deterministic}} + \underbrace{Bu_k}_{\text{control}} + \underbrace{Gw_k}_{\text{uncertain}} \tag{7}$$

Our main objective is to obtain information about w_k by comparing the values obtained by our model with the filtered observations. Identifying the uncertain parts of a model and describing these as system error is an important part of any realistic data-assimilation application. One can also think of the system noise as the control variables of the data-assimilation problem. Our main tool to achieve this is to check for the consistency of the error statistics.

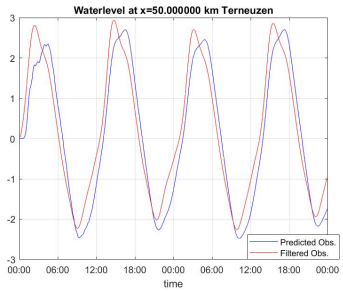
We will first look at calm weather conditions. The "tide"-files that we were given contain the filtered observations in the 5 cities that we are studying. To calculate the model errors, we run our model using `wave1d` and we subtract the results obtained from the filtered observations. Figure 2 shows a comparison between the results obtained with our model and the filtered observations.



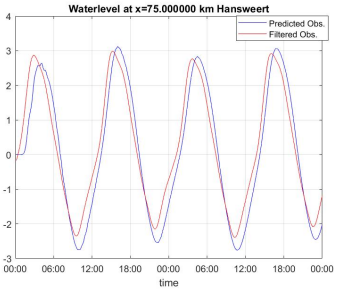
(a) Cadzand



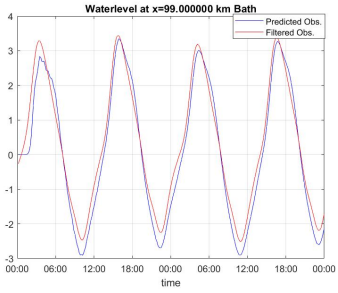
(b) Vlissingen.



(c) Terneuzen.



(d) Hansweert.



(e) Bath.

Figure 2: Waterlevel computed for our model vs filtered observations for the 5 different cities.

For the city of Cadzand we note that the results obtained are identical. This is mainly because we are using the filtered observations as left boundary conditions of our system in `wave1d_settings`. Therefore in this case, we only consider the last 4 cities when studying the error. Normally in the field of data assimilation, errors are modeled as Gaussian errors (see for example [3]):

$$w_k \sim N(0, \sigma^2) \quad i.i.d \tag{8}$$

This is due among other things to the computational facilities offered by working with conditional probabilities when using Bayes Theorem[7]. Moreover, Kalman made the assumption in one of his original studies that all probability distributions involved in the Kalman Filter are Gaussian [4].

[7] https://en.wikipedia.org/wiki/Bayes%27_theorem

Now, back to the error analysis. First of all we want to verify if the uncertainty in our state model can be implemented as in (8), or if we need to take special care of it, and model it in a different way. The statistics that we use to analyse the errors are:

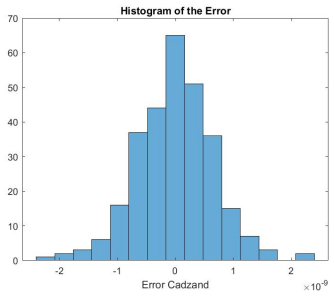
- Mean: to check if the results obtained by our model have mean 0 like in 8.
- Variance: to obtain an estimator for σ^2 in 8, which will serve us as a base value when we analyze the spread of the errors.
- Quantiles: we include the 0.25, the 0.5 (median), and the 0.75 quantiles.

In Table 1 we show the value of these statistics.

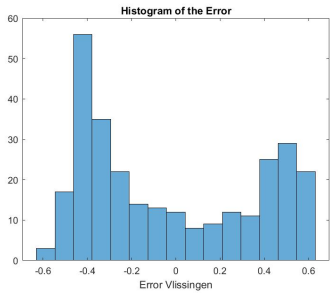
Table 1: Mean, Variance and quantiles for the model errors in the 4 cities.

	Error Vlissingen	Error Terneuzen	Error Hansweert	Error Bath
Mean	-0.0265	-0.1277	-0.1639	-0.2428
Variance	0.1464	0.3421	0.3314	0.1199
0.25 quantile	-0.3869	-0.6894	-0.6722	-0.4663
median	-0.1458	-0.1844	-0.1927	-0.3570
0.75 quantile	0.3979	0.4829	0.3775	-0.0263

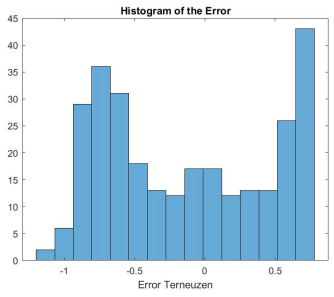
In order to give a better visual representation of the errors, and in order to check if the hypothesis of them being Gaussian is a well-thought hypothesis, we plot the histograms for the model error in each of the five cities in Figure 3.



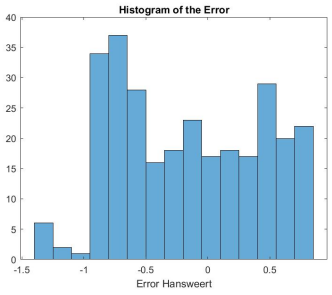
(a) Cadzand



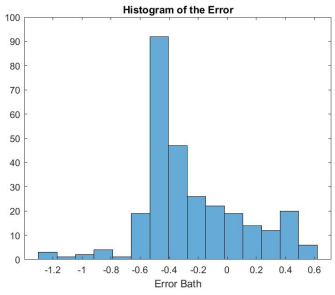
(b) Vlissingen.



(c) Terneuzen.



(d) Hansweert.



(e) Bath.

Figure 3: Histograms of the model errors for the 5 different cities.

From the shape of the histograms, it appears that some of the errors do not have a Gaussian shape. However, for the sake of simplicity, we maintain this hypothesis. As for the variance estimator, we

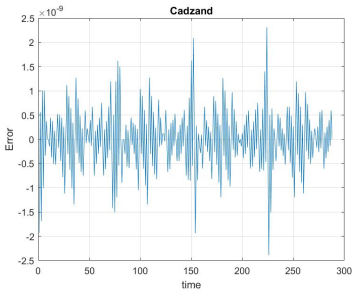
average the variance of the error in each of the five cities[8]:

$$\hat{\sigma}^2 = Var[\bar{x}] = \frac{1}{n^2}\sum_{i=1}^{5} Var[x_i] = 0.0303 \rightarrow \boxed{\sigma = 0.1741 \approx 0.2} \tag{9}$$

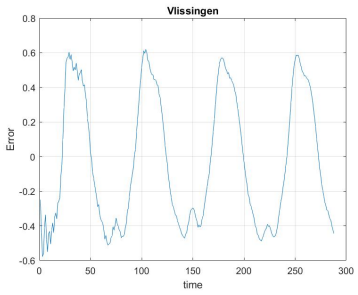
Finally, we want to check if these errors are independent, which is the situation we model in (8), or if there is any temporal relationship between them. To do so, we first compute the covariance matrix of the errors, that is given by:

$$COV = \begin{bmatrix} 0.1464 & 0.2181 & 0.2069 & 0.1087 \\ 0.2181 & 0.3421 & 0.3313 & 0.1703 \\ 0.2069 & 0.3313 & 0.3314 & 0.1771 \\ 0.1087 & 0.1703 & 0.1771 & 0.1199 \end{bmatrix}$$

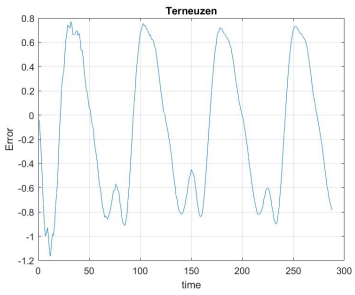
In addition, we plot in Figure 4 these model errors with respect to time for the different 5 cities:



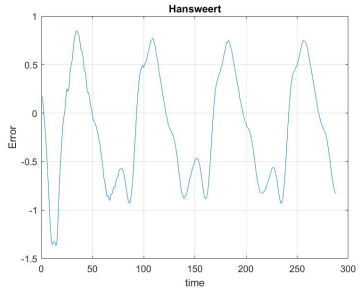
(a) Cadzand



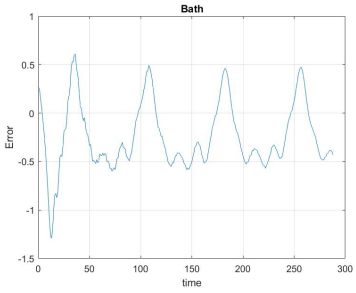
(b) Vlissingen.



(c) Terneuzen.



(d) Hansweert.



(e) Bath.

Figure 4: Model error vs time for the 5 different cities.

We can see that there is clearly a temporal relationship between the errors. This indicates that there most likely exist a time correlation which we will need to take into account when we introduce uncertainty in our problem. We will see in section 4 that a way to include this correlated noise is to model an external stochastic forcing term applied to the left boundary, a so-called auto-regressive type of noise. As a final note, we propose that it would be interesting to try to model the uncertainty in our model using SARIMA (Seasonal Auto Regressive Integrated Moving Average) Time Series models [5]. This option might be more appropriate to adequately describes the temporal correlations that we observed, and it might also take into account seasonal dependence of the problem.

[8]NB for Cadzand we have in fact $\hat{\sigma} = 0$

Question 4

As we saw in section 3, model errors appear to be time correlated, so we need some way to take this dependence into account. For this reason we add an AR(1) type stochastic forcing to the western boundary:

$$N(k+1) = \alpha N(k) + W(k), \tag{10}$$

where: $N(0) = 0$, $\alpha = exp(-dt/T)$ with $dt = 10min$ the time step of our model and $T = 6$hours. In addition, $W(k) \sim N(0, \sigma_W^2)$ is a series of i.i.d normal random variables and $\forall k \in \mathbb{N}$, $N(k)$ is independent of the terms in future time-steps: $\{W(k), W(k+1), \dots\}$. Regarding the expectation of the forcing terms $N(k)$, we have that:

$$E[N(k+1)] = E[\alpha N(k)] + E[W(k)] = \alpha E[N(k)],$$

and since $N(0) = 0$:

$$\boxed{E[N(k)] = 0 \ , \ \forall k \in \mathbb{N}}$$

Regarding the variance, we want this correlated process to verify that:

$$\sigma_N = \lim_{k\to\infty} \sqrt{E[N^2(k)]} = 0.2m,$$

to obtain a standard deviation very close to the one that we derived in (9). This can be achieved, since we still have a degree of freedom in determining the value of σ_W. Now, by recurrence of (10) we get:

$$N(k) = \alpha^m N(k-m) + \sum_{i=0}^{m-1} \alpha^i W(k-i). \tag{11}$$

So taking $m = k$ yields:

$$N(k) = \sum_{i=0}^{k-1} \alpha^i W(k-i). \tag{12}$$

After squaring this previous expression and taking expectations, we have:

$$E[N(k)^2] = E\left[\sum_{i=0}^{k-1}\sum_{j=0}^{k-1} \alpha^i \alpha^j W(k-i)W(k-j)\right] = \sum_{i=0}^{k-1}\sum_{j=0}^{k-1} \alpha^{i+j} E[W(k-i)W(k-j)],$$

And using the independence between the noisy random variables gives:

$$E[N(k)^2] = \sum_{i=0}^{k-1}\sum_{j=0}^{k-1} \alpha^{i+j}\sigma_w^2 \delta_{i,j} = \sum_{i=0}^{k-1} \alpha^{2i}\sigma_w^2$$

Lastly we take the limit $k \longrightarrow +\infty$. Using the Dominated Convergence Theorem[9] to interchange limit and expectation, we obtain:

$$\sigma_N = \lim_{k\to\infty} \sqrt{\sum_{i=0}^{k-1} \alpha^{2i}\sigma_w^2} = \sqrt{\sum_{i=0}^{\infty} \alpha^{2i}\sigma_w^2} = \sigma_w \sqrt{\frac{1}{1-\alpha^2}}$$

Therefore, in order to obtain our desired results, we have to impose:

$$\boxed{\sigma_W^2 = \sigma_N^2(1-\alpha^2) = 0.04(1-\alpha^2)}$$

[9] https://en.wikipedia.org/wiki/Dominated_convergence_theorem

This completes the theoretical side of the question. We now move to the numerical part. First we need a way implement the forcing (10). There are two possible ways of incorporating this term. The first one consist of introducing this contribution directly in the left boundary condition (which can be done for example by modifying the `wave1d_settings` file). Whereas the second one consists of creating an *extended state* vector to incorporate the forcing as an additional variable, and then modify the time-evolution equations. We will see in section 5 that this formulation is the desired one for our model, as it will verify the Markov property.

We then implement the model with the added forcing term. To do so, we created a new, modified routine based on `wave1d_timestep` to include the stochastic forcing, and we also modified the `wave1d_settings file` to take into account for the extended state vector. Finally, we ran the model for an ensemble size $N = 50$ The idea behind the ensemble is to approximate the time-update stochastic equation (20) (we will study its derivation later in section 5) with samples[10] ξ:

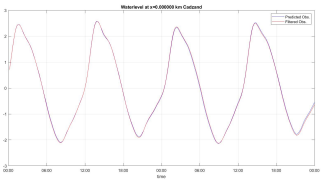
$$\xi^f_{k+1}(i) = M(\xi^a_k(i)) + Gw_k(i), \tag{13}$$

then we average over the ensemble to compute the vector state for the following time step, and we compute the covariance matrix:

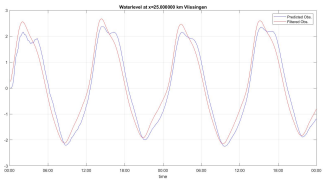
$$x^f_{k+1} = \frac{1}{N}\sum_{i=1}^{N} \xi^f_{k+1}(i) \tag{14}$$

$$P^f_{k+1} = \frac{1}{N-1}\sum_{i=1}^{N} (\xi^f_{k+1}(i) - x^f_{k+1})(\xi^f_{k+1}(i) - x^f_{k+1})' \tag{15}$$

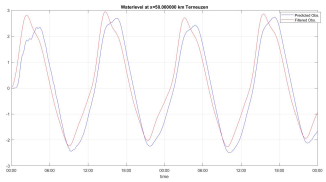
Remark that the run of a complete ensemble Kalman filter also requires a data-assimilation step. In this section we are only considering the time-update equations. We now plot both the new values obtained from our model and the filtered observations. The results that we have obtained can be seen in Figure 5.



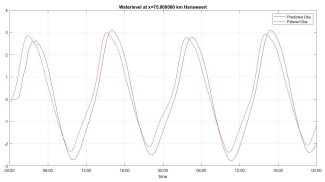
(a) Cadzand



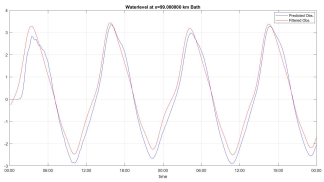
(b) Vlissingen.



(c) Terneuzen.



(d) Hansweert.



(e) Bath.

Figure 5: Waterlevel computed for our model vs filtered observations for the 5 different cities, including the left stochastic forcing term.

In this case we can appreciate small differences in Cadzand compared to Figure 2, due to the stochastic forcing term that we introduced. Subtracting the results obtained from the filtered observations, we get the new model errors. We then proceed to perform a quantitative analysis of them. The main objective is to check the differences with the model errors obtained in section 3. Remark that in this case, we also consider the errors in the first city. Our results are shown in Table 2:

[10] The initialization of these samples is an interesting point and we will return on it later in the report

Table 2: Mean, Variance and quantiles for the model errors in the 5 cities, including the left stochastic forcing term.

	Error Cadzand	Error Vlissingen	Error Terneuzen	Error Hansweert	Error Bath
Mean	0.0071	-0.0205	-0.1225	-0.1591	0.1180
Variance	0.0011	0.1449	0.3384	0.3273	0.1199
0.25 quantile	-0.0123	-0.3795	-0.6673	-0.6665	-0.4526
median	0.0084	-0.1271	-0.1799	-0.1758	-0.3533
0.75 quantile	0.0294	0.3866	0.4900	0.3796	-0.0300

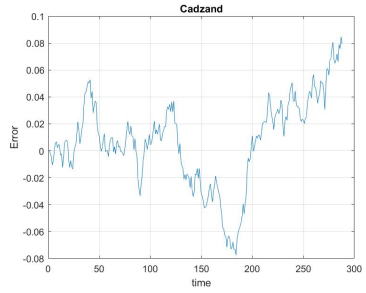
We can see that the mean of the errors is approximately 0. Regarding the variance, taking the variance of the mean of the errors we obtain:

$$\hat{\sigma}^2 = Var[\bar{x}] = \frac{1}{n^2}\sum_{i=1}^{5} Var[x_i] = 0.03718 \rightarrow \boxed{\sigma = 0.1928 \approx 0.2}$$

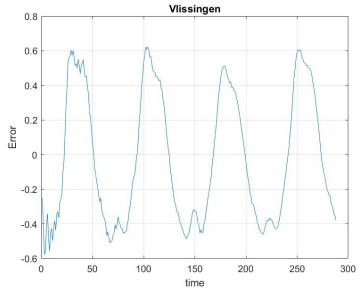
Again, we want to check if these errors are independent as in (8) or if is there any time relation between them. If the $AR(1)$ model could perfectly fit the correlations between the errors in section 3, then we would expect our errors to follow a white noise process, or in other words, time-uncorrelated. We first compute the covariance matrix, that is given by:

$$COV = \begin{bmatrix} 0.0011 & 0.0006 & 0.0006 & 0.0007 & 0.0006 \\ 0.0006 & 0.1449 & 0.2158 & 0.2043 & 0.1064 \\ 0.0006 & 0.2158 & 0.3384 & 0.3273 & 0.1670 \\ 0.0007 & 0.2043 & 0.3273 & 0.3273 & 0.1740 \\ 0.0006 & 0.1064 & 0.1670 & 0.1740 & 0.1180 \end{bmatrix}$$

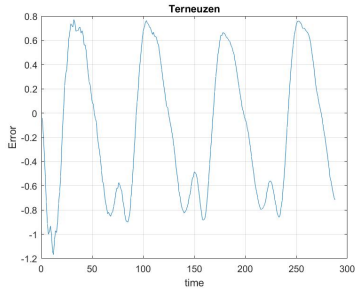
In addition, plotting the error vs time we can see our results in Figure 6.



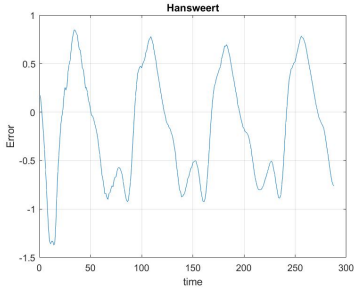
(a) Cadzand



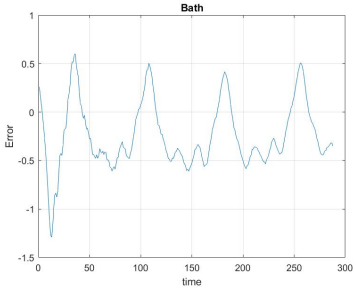
(b) Vlissingen.



(c) Terneuzen.



(d) Hansweert.

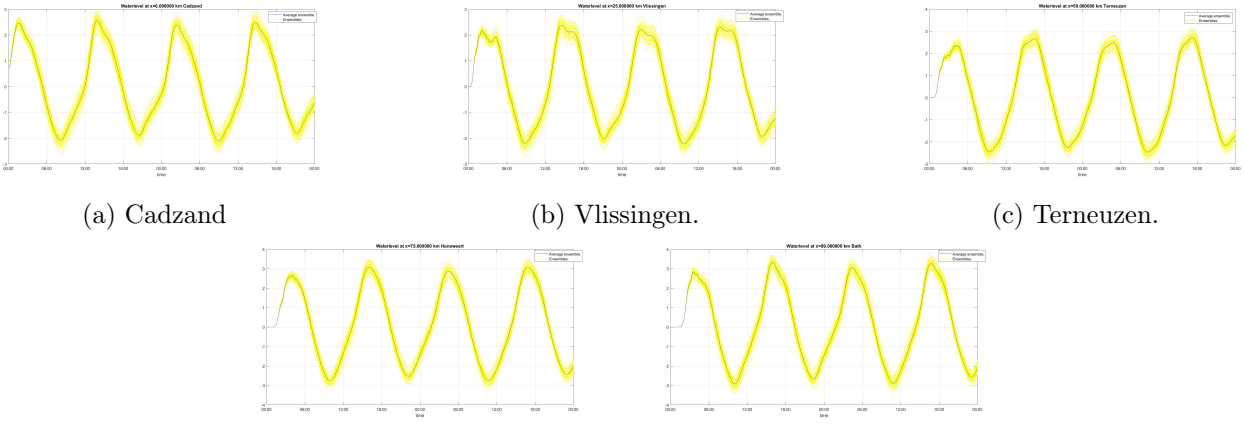


(e) Bath.

Figure 6: Waterlevel computed for our model vs filtered observations for the 5 different cities, including the left stochastic forcing term.

We observe that temporal correlations between the errors are still present. Moreover the covariance matrix is not diagonal as we were expecting if there were no correlations between the errors. We therefore conclude that although AR(1) allows us to introduce a time-correlated error in the model, it might not be the most appropriate model for our data. Again, as we mentioned in section 3, a SARIMA model might be advisable to better account for the phenomenon of seasonality.

Finally, we will analyze the ensemble spread. We first plot all the ensembles against the vector state obtained by averaging them. The results can be seen in Figure 7.



(a) Cadzand (b) Vlissingen. (c) Terneuzen.

(d) Hansweert. (e) Bath.

Figure 7: Spread of the ensembles $\xi(i)$ vs state vector x^f, for different times and cities.

Once the qualitative results have been obtained, we move on to a quantitative analysis of the results. We created a routine (`wave1d_ensemble`) that computes two matrices: the first one refers to the average of the errors between the ensembles and the state vector. Each row of the matrix represents a different city and each column represents each of the ensembles. Whereas the second matrix computes the variance of these errors. As these matrices are elements of $\mathcal{R}^{5\times 50}$, we only show the most remarkable results in Table 3:

Table 3: Statistics for the mean and the variance for the ensemble spread

	Error Cadzand	Error Vlissingen	Error Terneuzen	Error Hansweert	Error Bath
Max. Mean	0.2118	0.2162	0.2198	0.2219	0.2225
Min. Mean	-0.2007	-0.1958	-0.1929	-0.1920	-0.1924
Min. abs. mean	0.0047	0.0051	0.0053	0.0054	0.0054
Max. Variance	0.0708	0.0703	0.0761	0.0839	0.0893
Min. Variance	0.0113	0.0091	0.0085	0.0094	0.0124

From this table we can deduce that in our group of ensembles there are some that are very close to the state vector and others that are a little far from it. Moreover, the more ensembles we average, the closer we will be to represent the dynamics of a stochastic system governed by the equation (10).

Question 5

One key requirement for the use of the Kalman Filter is the Markov property. The Markov property means that evolution of the Markov process in the future depends only on the present state and does not depend on past history. A Markov process "does not remember the past if the present state is given"[11]. Mathematically, if our time evolution and measurement equations are respectively given by:

$$x_{k+1} = Mx_k + Bu_k + Gw_k, \tag{16}$$

$$z_k = Hx_k + v_k, \tag{17}$$

then the Markov property can be expressed as:

$$p(x_{k+1}|x_1, \ldots, x_k, u_1, \ldots, u_k, w_1, \ldots, w_k) = p(x_{k+1}|x_k, u_k, w_k), \tag{18}$$

$$p(z_k|x_1, \ldots, x_k, v_1, \ldots, v_k) = p(z_k|x_k, v_k). \tag{19}$$

For the AR(1) type stochastic uncertainty that we introduced in our model (see equation (10)) it holds that:

$$p(N_k|W(k-1), \ldots, W(1)) = \begin{cases} 1 \text{ if } N_k = \sum_{i=0}^{k-1} \alpha^{k-i} W(i) \\ 0 \text{ if} N_k \neq \sum_{i=0}^{k-1} \alpha^{k-i} W(i) \end{cases}$$

$$N_k|W(k-1) \sim \alpha W(k-1) + \underbrace{\sum_{i=0}^{k-2} \alpha^{k-i} W(i)}_{\text{Normal Random Variable}} .$$

From which we conclude that:

$$p(N_k|W(k-1), \ldots, W(1)) \neq p(N_k|W(k-1)).$$

So the sthocastic unceratinty itself is *not* Markov. One way of dealing with this problem is to include the forcing term into the state vector[12] :

$$\mathbf{x_k} = [h_1(k), u_1(k), \ldots, h_n(k), u_n(k)]^T,$$

which becomes the so-called *extended state* vector:

$$\tilde{\mathbf{x}}_\mathbf{k} = [h_1(k), u_1(k), \ldots, h_n(k), u_n(k), N(k)]^T,$$

One remark regarding numerical methods: we will need to change the matrices A and B that modeled the dynamics of the system in `wave1d_initialize` (this is done in a new function `wave1d_initialize_enKF`). We now have:

$$A_{new}\tilde{\mathbf{x}}_{\mathbf{k+1}} = \underbrace{\begin{bmatrix} A & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}}_{A_{new}} \begin{bmatrix} \mathbf{x_{k+1}} \\ N_k \end{bmatrix} = \underbrace{\begin{bmatrix} B & \mathbf{e_1} \\ \mathbf{0} & \alpha \end{bmatrix}}_{B_{new}} \begin{bmatrix} \mathbf{x_k} \\ N_k \end{bmatrix} + \sigma_w \begin{bmatrix} \mathbf{0} \\ N(0,1) \end{bmatrix} = B_{new}\tilde{\mathbf{x}}_\mathbf{k} + \sigma_w \begin{bmatrix} \mathbf{0} \\ N(0,1) \end{bmatrix}, \tag{20}$$

where $\mathbf{e_1} = [1, 0, \ldots, 0]^t$, allows us to introduce the forcing to the western boundary and the second term accounts for its stochastic nature.

[11] This property is often called *absence of memory*

[12] The subindex of $\tilde{\mathbf{x}}_\mathbf{k}$ represent the temporal component and the subindex i of the elements $h_i(k), u_i(k)$ represent the spatial component.

As we implement the extended state vector, the update for the first component of the right-hand-side needs to be modified: $rhs(1) = settings.h\_left(i) + N_{k-1}$. To do this we implemented a new routine (`wave1d_timestep_enKF`) very similar to `wave1d_timestep`. With the new routine we can then simply run the model forward in time (equivalent with solving the previous system), in order to obtain the value of $\tilde{\mathbf{x}}_{\mathbf{k+1}}$.

With these modifications we can verify that the extended system we obtain complies with the Markov property. Indeed, looking at (20), we find:

$$p(\mathbf{x_{k+1}}|\mathbf{x_1}, \ldots, \mathbf{x_k}, N_1, \ldots, N_k, W(1), \ldots, W(k)) = p(\mathbf{x_{k+1}}|\mathbf{x_k}, N_k, W(k)), \tag{21}$$

$$p(N_{k+1}|N_1, \ldots, N_k, W(1), \ldots, W(k)) = p(N_{k+1}|N_k, W(k)). \tag{22}$$

In conclusion: in this new setting we can apply the Kalman filter to our system.

Question 6

6.1 Implement an Ensemble Kalman Filter and set up an identical twin experiment

Once the model errors have been addressed in section 4 and we have modified our system so that the Markov property is met in section 5, we proceed with the actual data-assimilation. The main objective of this section is to implement the Ensemble Kalman Filter. This method is based on 2 steps. First, the time-update equations described by (13) (14) (15), using the dynamics in (20), take place. For computation efficiency instead of (15), we will use:

$$L^f_{k+1} = \frac{1}{\sqrt{N-1}}[(\xi^f_{k+1}(1) - x^f_{k+1}), \ldots, (\xi^f_{k+1}(N) - x^f_{k+1})] \tag{23}$$

Second and last, the measurement-update equations take place. These equations are described by:

$$\Psi_k = HL^f_k \tag{24}$$

$$K_k = L^f_k \Psi'_k \left(\Psi_k \Psi'_k + R\right)^{-1} \tag{25}$$

$$\xi^a_k(i) = \xi^f_k(i) + K_k(z_k - H\xi^f_k(i) - v_k(i)) \tag{26}$$

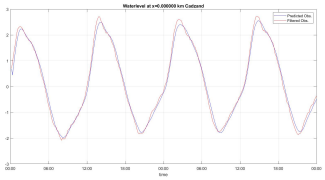
where: $H \in \mathcal{R}^{5\times 201}$ is the measurement matrix, $K_k \in \mathcal{R}^{201\times 5}$ is the Kalman gain, $R \in \mathcal{R}^{5\times 5}$ is the covariance matrix of the measurement error and $z_k \in \mathcal{R}^{5\times 1}$ are the observations. The programme that we have created to carry out this Ensemble Kalman Filter is `wave1d_enKF_WithMeasurementUpdate`. This same programme will be used later in section 9 when we move to the storm observations. Moreover in section 6, section 7 and section 8 we will test its operation and features.

We then set up an identical twin experiment to test our implementation of the ensemble Kalman Filter. We created a MATLAB routine (`wave1d_enKF_TwinSetup`) to run the model equations forward and obtain our "truth" and our pseudo-observations. To generate the following results we set up a twin experiment with perfect observations. That is, once our truth was generated we computed the pseudo-observations as:

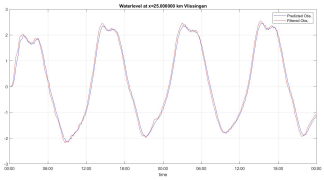
$$\tilde{z}_k = Hx_k,$$

where we are assyuming that there is no uncertainty in the measurements. Once the observations are obtained we simply "feed them" to the Ensemble Kalman filter to obtain the state estimators $\tilde{x}_k$.

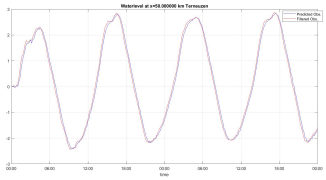
In the following we present the results with an ensemble size $N = 400$ and we plot the values computed for our model as well as the pseudo-observations. The results that we have obtained can be seen in Figure 8:



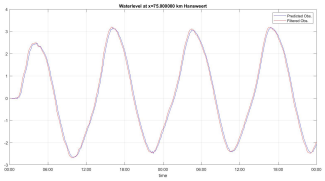
(a) Cadzand



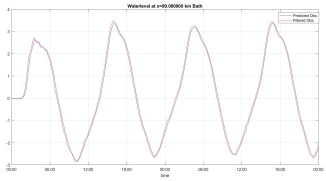
(b) Vlissingen.



(c) Terneuzen.



(d) Hansweert.



(e) Bath.

Figure 8: Waterlevel computed for our model vs pseudo-observations generated for our model: Identical twin experiment setup.

Comparing Figure 5 and Figure 8, we can see that clearly the assimilation of observations leads to more accurate results.

In real life application, when dealing with the twin experiment, we do not have access to the theoretical stochastic model. Hence we can no longer check the differences between the actual state vector x_k and our estimator $\tilde{x}_k$. However, there are 2 ways to tackle this problem for testing our implementation. The first one consists in studying the distribution of:

$$\tilde{z}_k - H\tilde{x}_k. \tag{27}$$

The second one consist on splitting the observations into two sets: the training set and the validation set. The former is used to feed the model and compute state estimators. The latter is used to test the accuracy of these estimators. In this section we use the first procedure, since the second one is more machine-learning oriented.

We then proceed to perform a quantitative analysis of the errors described by (27). The main objective is to check if the obtained results are more accurate that those obtained in section 4. In Table 4 we show the values of some relevant statistics.

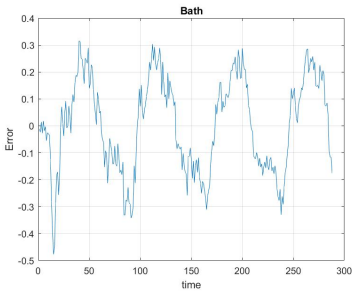
Table 4: Mean, Variance and quantiles for the model errors in the 5 cities, for the twin experiment.

	Error Cadzand	Error Vlissingen	Error Terneuzen	Error Hansweert	Error Bath
Mean	-0.01550	-0.00499	0.00004	0.00220	0.00260
Variance	0.02171	0.02051	0.02241	0.02652	0.03080
0.25 quantile	-0.11764	-0.10153	-0.12200	-0.14266	-0.14144
median	-0.00963	-0.00770	-0.01742	0.00137	0.01180
0.75 quantile	0.09686	0.10787	0.13952	0.14700	0.14864

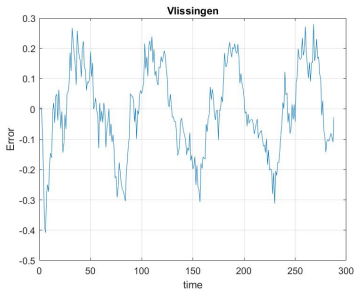
Moreover, the covariance matrix is:

$$COV = \begin{bmatrix} 0.02171 & 0.01681 & 0.01362 & 0.01113 & 0.01049 \\ 0.01681 & 0.02051 & 0.01798 & 0.01622 & 0.01526 \\ 0.01362 & 0.01798 & 0.02240 & 0.02130 & 0.021232 \\ 0.01113 & 0.01622 & 0.02130 & 0.02652 & 0.02647 \\ 0.01049 & 0.01526 & 0.02123 & 0.02647 & 0.03079 \end{bmatrix}$$

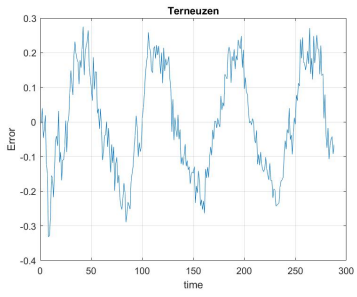
Comparing the values of Table 2 and Table 4, we can corroborate that when we introduce observations into our model, the accuracy increases considerably. This showcases the usefulness of Data Assimilation for forecasting. We also include in Figure 9 the plottings of these errors vs time.



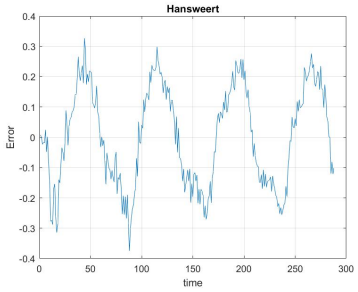
(a) Cadzand



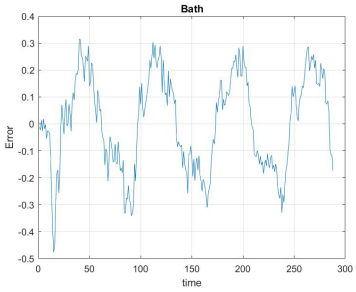
(b) Vlissingen.



(c) Terneuzen.



(d) Hansweert.



(e) Bath.

Figure 9: Model error vs time for the twin experiment.

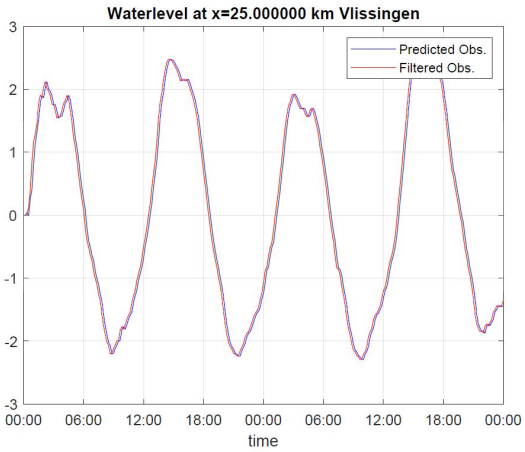
As we commented earlier in section 4, the correlation between these errors and time is most likely due to the fact that the $AR(1)$ model does not perfectly fit our model error. So these temporal correlations in the model errors propagate for our measurement errors.

6.2 Why should the EnKF work perfectly in this experiment?

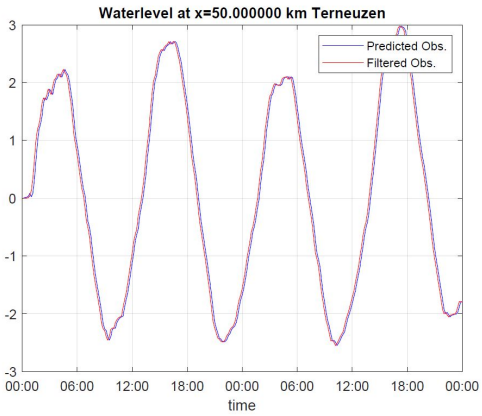
In a real-life situation we only have observations z at our disposal, and we do not know the true state x (if we did, filtering and forecasting would both become much easier). Not only we do not know the true state x_k at each moment k in time, but we also do not know the perfect description of the physical phenomenon that governs the evolution of x in time. We can only approximate this phenomenon through a mathematical model.

Now in fact, for our twin experiment, we *do know* the truth (because we generate it ourselves) and we do know the "phenomenon" that governs this truth (it is the model that we implement and run forward ourselves). For this reason, we expect that the approximations $\hat{x}_k$ generated by the Ensemble Kalman filter should agree with the truth x_k (in the limit of taking large ensemble sizes).

This fact can already be appreciated in Figure 8. Below we report the results for another set of perfect observations and for a larger ensemble size, $N = 600$. For the sake of clarity we only report the result in two cities:



(a) Vlissingen



(b) Terneuzen

Figure 10: Identical twin experiment setup for a large ensemble: precision of Ensemble Kalman filtering

As it turns out, the larger the ensemble size is, the more precise the filter is, almost to the point of perfection.

6.3 One remark: a discussion about the implementation

In the preceding subsection we presented the results we obtained for the ensemble Kalman filter applied to a twin experiment. We generated a new "truth" (that is, a new true state) using the same numerical model that describes the evolution of the system in time (including AR(1) forcing) and we generated new observations based on this "truth". Aside from the error analysis conducted above, we ran a few other tests to check our implementation. In particular we tried to investigate whether or not the assumption of perfect observations can be made in the real-life case (with the "tide-" or "waterlevel-" files given). To do so, we implemented the ensemble Kalman and we performed multiple runs for various scenarios:

1. *enKF* with assimilation of data generated by the twin experiment. In the model we recovered the observations z_k *without* addition of noise v_k.
2. *enKF* with assimilation of data generated by the twin experiment. This time we recovered the observations z_k *with* addition of noise v_k. We considered at each time-step Gaussian noise $v \sim N(\mu = 0, \sigma^2 = 0.01)$ and we assumed uncorrelated noise both in time and space. This is done to describe the fact that the error in a specific location depends only on the accuracy of the instrument that made the observation.
3. *enKF* with assimilation of data coming from the real-world observations (the "tide_....txt" files) under the assumption of perfect observations.
4. *enKF* with assimilation of data coming from the real-world observations (the "tide_....txt" files) assuming noisy observations. The description of the data made in [6], as well as the observation the "tide" files given, are the two factors that made us choose $\sigma^2 = 0.01\text{m}^2$ for the variance[13]: it appears that the values of the water levels (given in meters) are up to the second decimal place, so the uncertainty is of the order of centimeters.

The objective of this testing was to verify whether or not in the real-life case, we should model the uncertainty of the measurements or whether the observations as given should be considered perfect. Note

[13]NB the remarks about non-correlation both in time and space made in point 2 still hold.

that making a decision on how to consider the data impacts the implementation of the 'data assimilation' step of the *enKF* (presence or absence of matrix R). Our findings can be very simply resumed for each case[14]:

1. The implementation yielded satisfactory results (in fact this case is the one we considered in our answer in section 6).
2. The implementation yielded satisfactory results.
3. The implementation did *not* yield satisfactory results.
4. The implementation again yielded satisfactory results.

After having obtained these results, we wondered if the problem in case 3 was due to an incorrect implementation of the AR(1) noise[15]. To verify this, we ran more twin experiments, both with and without the assumption of Gaussian noise[16]. Only this time, we *did not* include the AR(1) term in the model equations (that is, we used the given file `wave1d_timestep` to run the model forward and obtain our truth). Subsequently, we ran the *enKF* for these new sets of "truth" and "pseudo-observations". Once again, the results after running *enKF* using these various observations were satisfying. To recap this case: no AR(1) noise to generate the truth; observations were generated without noise, and afterwards observations were generated with noise.

In conclusion, all of the above results suggested us the following:

- It appears that our implementation of the *enKF* is not systemically flawed.
- It also appears that the assumption of *noisy* observations for the real-life case is an assumption that makes sense. This is in line with the analysis conducted in [7], an article which we found while looking for an analytical explanation to our results.

In conclusion, we decided to implement the *enKF* with the additional hypothesis of Gaussian uncorrelated noise for the observations, even in the real-life case. We did our best to approximate the standard error of this noise based on the information we could find about the datasets, such as the information in [6].

Finally we add one analytical remark that again motivates the choice we made: the matrix L_K^f can have rank at most $N-1$, and if we choose $R=0$ then each ensemble member is updated with a very similar quantity, which can lead to a decrease in the rank of L_k^f. Moreover, after looking at (25) we can see that if matrix L_k^f behaves "badly" there is a chance for the Kalman gain matrix K_k to be very ill-conditioned. Now, because the Kalman gain is related to the covariance matrix of the ensemble by[17]:

$$K_k = P_k^f H' \Big(H P_k^f H' + R \Big)^{-1}, \tag{28}$$

a conditioning problem in K_k can lead to bad behavior of the covariance matrix of the ensemble spread P_k^f, which in turn means that our ensemble will not perform well.

In the end, this last subsection is only a discussion. In essence, all of the numerical experiments are done using the twin experiment setup and with the truth and data generated by us. But we thought it important to say something about the real-world data and on the different ways they can be treated.

[14] To keep the text light we do not include figures here, since many other plots are available in other parts of the report.

[15] NB it seems very strange that an incorrect implementation would only affect one out of the four settings, but we wanted to make sure nonetheless.

[16] NB we are of course not "mixing" perfect and noisy observations in the same set, but we generate distinct sets each time

[17] This is a general formula, so matrix R is also present

Question 7

During the course we studied that, assuming Gaussian noise the Kalman filter yield the best linear, unbiased estimator. The label best, refers to the fact that it is the estimator with the minimum variance. However, the high computational complexity of this filter is sometimes problematic when dealing with high-dimensional problems and restricts its applicability in the real world. To tackle this problem, people began to use Monte Carlo-based approximations of this filter. Among these approximations, the best known is the one we have used throughout this report, the Ensemble Kalman Filter. The main advantage of this algorithm is that the computational complexity is lower than that of the Kalman Filter. In our case, as our evolution time equation is linear, for a sufficiently large number of ensembles, the solution converges to that of the Kalman filter [8].

The numerical experiments were run again using the twin experiments. We generated a few sets of "truths" from which we extracted multiple sets of pseudo-observations. To generate our pseudo-observation, we once again considered the case of perfect measurements as well as the case of noisy measurements. Then, for each set of observation we:

- Ran the ensemble Kalman filter for an ensemble size $N = [50, 100, 200, 500, 1000, 1500, 3000]$
- Computed the Root Mean Square Error averaged both in time and space:

$$RMSE := \sqrt{\frac{1}{M \times K} \sum_{i=1}^{K} \sum_{j=1}^{M} (x_i(j) - \hat{x}_i(j)),,} \tag{29}$$

  where K is the total number of time steps and M of the spatial steps.

In Figure 11 we show our results:

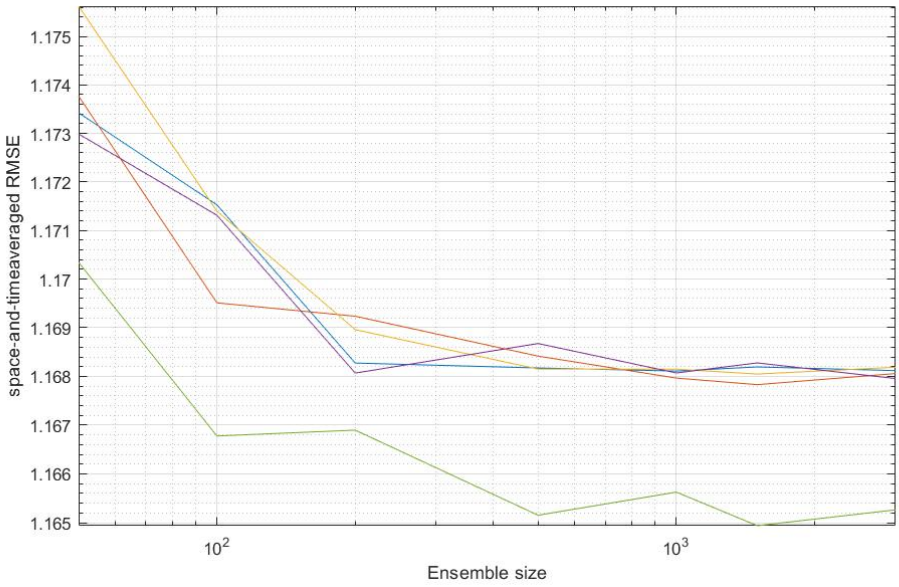


Figure 11: Convergence of the Ensemble Kalman filter

We can infer from the previous theoretical analysis that the solution should converge to the solution of the Kalman Filter. Figure 11 suggests that indeed the ensemble Kalman filter converges, although it seems like not every instance is converging to the same value.

Now, what about the theoretical rate of convergence? As we mentioned, Ensemble Kalman is essentially a Monte-Carlo type of method. Moreover our approximation $\hat{x}$ is the mean of the ensemble members. Therefore, it is reasonable to expect that the speed of convergence will be close to the theoretical speed of convergence of the mean of a Monte-Carlo method. For this reason we expect:

$$\text{Rate of Convergence} \sim \frac{1}{\sqrt{N}} \text{ for the mean of the ensemble members.}$$

This holds (or should hold) when we consider the mean of the ensemble[18], and the result is simply a consequence of the Central Limit Theorem[19]. However, if one is interested in the ensemble spread, then it might be the case that one will observe faster convergence of the spread, for instance with a rate $\mathbf{O(1/N)}$. In our practical experiment, most likely we would need to use very large ensembles ($O(10^4)$) to show definitively that for each pseudo-observation, ensemble Kalman converges to a certain value. Instead, we limited our analysis to $N = 3000$ for ease of computation. Finally, notice the $RMSE$ does not tend to 0. This is in line with the theoretical construction of the Kalman filter, where the estimators are obtained by averaging the observations with the model.

[18] https://en.wikipedia.org/wiki/Monte_Carlo_method
[19] https://en.wikipedia.org/wiki/Central_limit_theorem

Question 8

As we explained, the implementation of the ensemble Kalman filter has been carried out under the assumption of noisy observations. As was mentioned in section 6, our implementation we have set:

$$R = 0.01 I_5.$$

Finally, the initial conditions that we chose to carry out the analysis in section 6 were:

$$\xi_0(i) \sim N(\mathbf{0}, 0.2^2 I_{201}),$$

which led us to the results shown in Table 4 and Figure 9. This was done to avoid ill-conditioning of the matrix L in the first time-steps of the implementation. As we already mentioned, the Ensemble Kalman is based essentially on a Monte-Carlo train of thought. Therefore it is not unreasonable to initialize the ensemble members as white noise.

In this section we will analyze what happens if we change initial conditions. The new initial conditions we choose are precisely:

$$\xi_0(i) = \mathbf{0}.$$

The main objective is to check to what extent the choice of initial conditions affects our results. In Table 5 we show the value of some relevant statistics.

Table 5: Mean, Variance and quantiles for the model errors in the 5 cities, for $\xi_0(i) = \mathbf{0}$, ensemble size $N = 400$.

	Error Cadzand	Error Vlissingen	Error Terneuzen	Error Hansweert	Error Bath
Mean	-0.00055	0.00489	0.00683	0.00801	0.00834
Variance	0.01899	0.01797	0.02008	0.02349	0.02753
0.25 quantile	-0.10292	-0.09544	-0.10338	-0.13856	-0.14009
median	0.00077	-0.00017	-0.01411	0.01033	0.00836
0.75 quantile	0.10335	0.11906	0.13167	0.14467	0.14889

Moreover, the covariance matrix is:

$$COV = \begin{bmatrix} 0.01899 & 0.01451 & 0.01192 & 0.01031 & 0.00947 \\ 0.01451 & 0.01797 & 0.01643 & 0.01465 & 0.01390 \\ 0.01192 & 0.01643 & 0.02008 & 0.01938 & 0.01890 \\ 0.01031 & 0.01465 & 0.01938 & 0.02349 & 0.02397 \\ 0.00947 & 0.01390 & 0.01890 & 0.02397 & 0.02753 \end{bmatrix}$$

We can observe that there are no remarkable differences between the values in Table 4 and Table 5. The means of the first table were closer to 0, while the variances in the second table are smaller. It appears that the choice of the initial values is somewhat of little importance.

Out of curiosity, we also tested what would happen with the choice of initial conditions with mean nonzero, for example $\xi_0(i) = \mathbf{1}$ and $\xi_0(i) \sim N(\mathbf{1}, 0.2^2 I_{201})$. In these cases the errors were in general larger, especially for first time steps. But even so, we still observed convergence between the values of our model and the pseudo-observations. Regardless of the initial conditions, the ensemble Kalman filter method converges. Although it would be useful to study in more depth which initial values lead to the most accurate results.

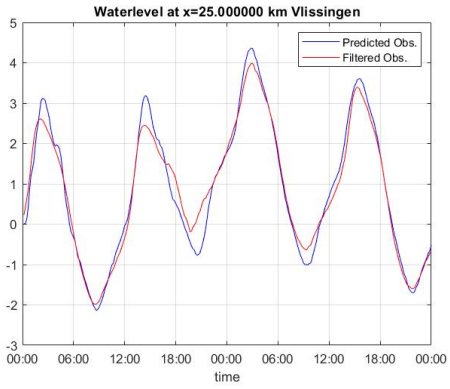
In conclusion, what we observe is that as the ensemble size increases, and after a few time-steps in the implementation, the effect of the initialization is washed away. This can be explained if we adopt a more Bayesian approach: the effect of the prior distribution that we assign to the ensemble members is washed away as we assimilate observations.

Question 9

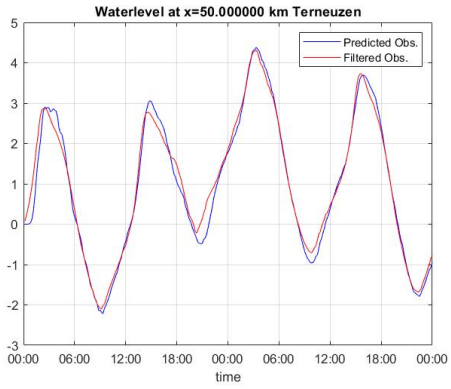
Once we have analyzed the properties of the Kalman filter using the twin experiment, we move on to the real observation with the storm effect included. This time we do not generate a new truth or twin data, but we simply run the Ensemble Kalman and in the assimilation step we assimilate the data of the "waterlevel_..." files.

Notice that for the first city, Cadzand, there are no observations available. In a first instance, we used the "tide" file for Cadzand, so that we could have assimilation at each location. However if the goal is to only assimilate the data from the "waterlevel_..." files, it makes more sense to exclude this city completely from the assimilation step. For this reason, we decided to modify our implementation to assimilate only data at four locations. In short, we modified the matrix H which is a (4×201) matrix this time, instead of a (5×201) as in the previous questions. We still consider the measurements as noisy[20]. Of course also the size of the matrix R and of the "vector of noises" that we introduce are modified according to the new settings of this question.

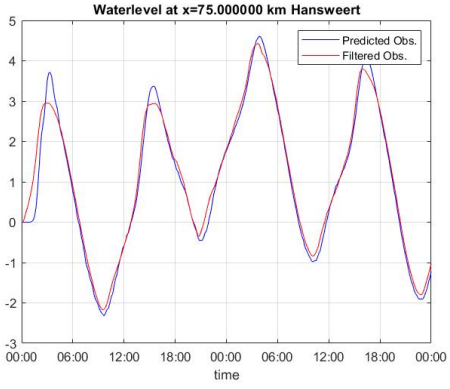
Now, $N = 400$, we plot the values computed for our model and the storm observations. The results that we have obtained can be seen in Figure 12.



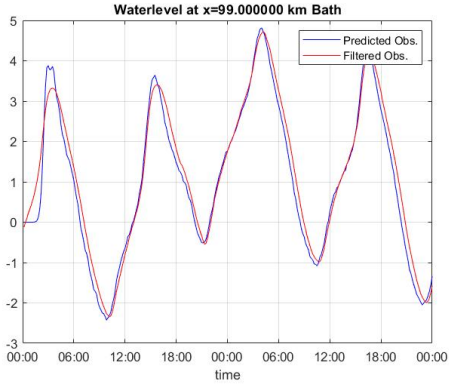
(a) Vlissingen.



(b) Terneuzen.



(c) Hansweert.



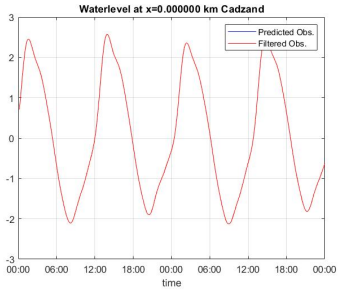
(d) Bath.

Figure 12: Waterlevel observations vs Ensemble Kalman filter data: Xavier Storm data.

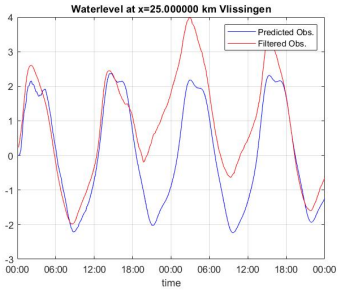
One interesting observation is that the closer the filtered values are to the observations, the smoother the curve is (the weight of the observation becomes more important) while the further we are from the observations, the more rough the curve of the *enKF* becomes (wherein the uncertainty has a heavier weight). As it turns out, the Ensemble Kalman filter does not perform as well as in the case of the twin

[20] For more information once more we refer to the discussion in section 6

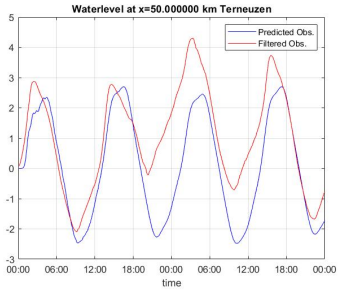
experiments, but it still outperforms the simple stochastic dynamical model. The results for the former are show in Figure 13:



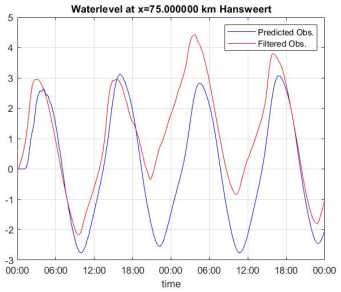
(a) Cadzand



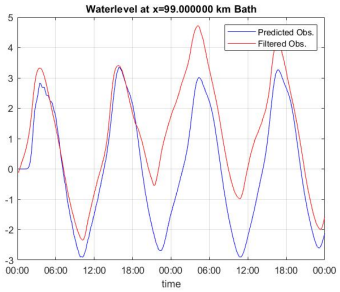
(b) Vlissingen.



(c) Terneuzen.



(d) Hansweert.



(e) Bath.

Figure 13: Waterlevel observations vs numerical model data: Xavier Storm data.

Of course the numerical model itself is independent of the data (except for the left-end points where we consider the value of the observations as input), which is why the blue lines in the above plots are the same as the blue lines in Figure 2.

Next we try to quantify these observations: we compute some statics for the error in both cases (ensemble Kalman and numerical model) then we compare these statistics[21]. The main objective is to check the performance of the ensemble Kalman Filter when we have unusual events.

In Table 6 we show the values of some relevant statistics of the measurement error using enKF.

Table 6: Mean, Variance and quantiles for the model errors in the 5 cities, for the twin experiment.

	Error Vlissingen	Error Terneuzen	Error Hansweert	Error Bath
Mean	0.0053	-0.063	-0.0971	-0.1505
Variance	0.096	0.0564	0.0603	0.0943
0.25 quantile	-0.1412	-0.1733	-0.1608	-0.3960
median	0.0317	-0.0434	-0.1100	-0.1475
0.75 quantile	0.2088	0.0721	-0.0265	-0.0938

Moreover, the covariance matrix is given by:

$$COV = \begin{bmatrix} 0.960 & 0.0448 & 0.0177 & 0.0352 \\ 0.0448 & 0.0564 & 0.0440 & 0.0116 \\ 0.0177 & 0.0440 & 0.0603 & 0.0333 \\ 0.0352 & 0.0116 & 0.0333 & 0.0943 \end{bmatrix}$$

[21] NB we will only consider four cities for the error of the numerical model

In Table 7 we show the values of some relevant statistics of the measurement error using enKF.

Table 7: Mean, Variance and quantiles for the model errors in the 5 cities, for the twin experiment.

	Error Vlissingen	Error Terneuzen	Error Hansweert	Error Bath
Mean	-0.7988	-0.9469	-1.0375	-1.1071
Variance	0.7780	1.0589	1.0904	0.8574
0.25 quantile	-1.4676	-1.5903	-1.6298	-1.6233
median	-0.5240	-0.7570	-0.8410	-0.8025
0.75 quantile	-0.1616	-0.1667	-0.2631	-0.4964

Moreover, the covariance matrix is given by:

$$COV = \begin{bmatrix} 0.7780 & 0.8782 & 0.8592 & 0.7505 \\ 0.8782 & 1.0589 & 1.0608 & 0.8964 \\ 0.8592 & 1.0608 & 1.0904 & 0.9312 \\ 0.7505 & 0.8964 & 0.9312 & 0.8574 \end{bmatrix}$$

Looking at the values in Table 6 and Table 7 we found that the results obtained with data assimilation are more accurate. We emphasize once again the importance of these data assimilation methods to make predictions.

Question 10

To answer this question, we first had to find the time at which the peak waterlevel was registered[22]. Call this time t_{PEAK}. Notice the moment in which the peak waterlevel is reached is slightly different for each city. For our purposes, we considered t_{PEAK} as the moment in which the overall maximum (over the four cities) waterlevel is recorded.

Then, the forecast model runs as follow:

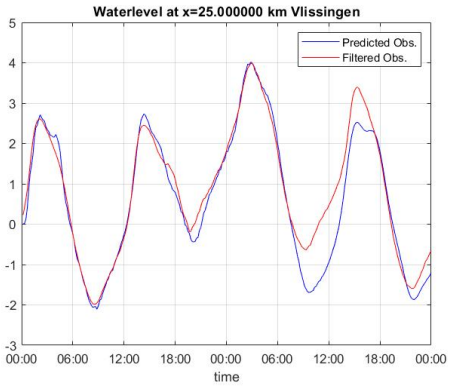
- *Filtering part:* run the Ensemble Kalman filter (ensemble size $N = 400$) with assimilation of the data in the "waterlevel" files until the time $t_{PEAK} - \Delta t$, for some chosen Δt, the so-called lead-time.
- *Forecasting part:* stop assimilating the data, but keep running the Ensemble Kalman filter forward in time (so only time update).

The time Δt that we chose are:

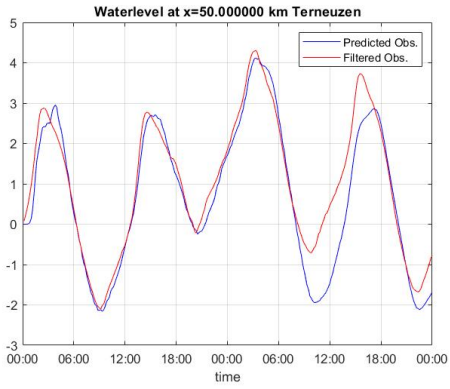
$$\Delta t = 0, 6, 9, 12, 24 \quad hrs.$$

The goal is then to check how the accuracy of forecasts for the peak waterlevel depend the on lead-time Δt.

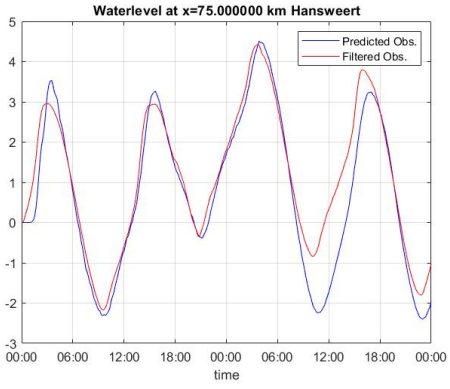
Firs, we give qualitative analysis with some plots. In Figure 14 we observe the forecast made when $\Delta t = 0$, so when we stop assimilating data at the moment in which the waterlevel peak is registered. The results of the forecast for the four cities are:



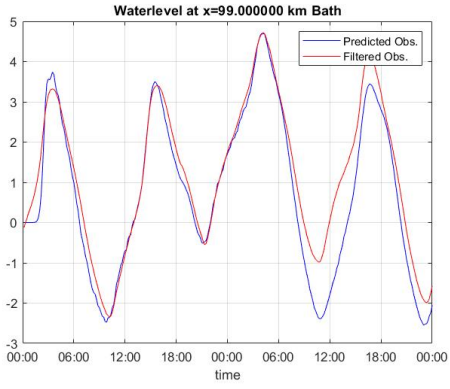
(a) Vlissingen.



(b) Terneuzen.



(c) Hansweert.

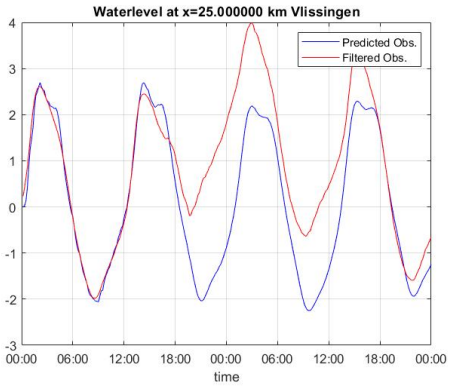


(d) Bath.

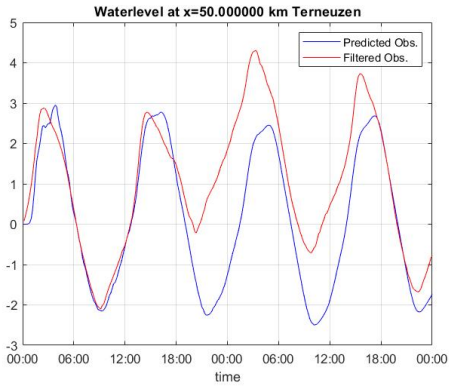
Figure 14: Forecast for lead-time $\Delta t = 0 hrs$.

[22]NB this is computationally trivial

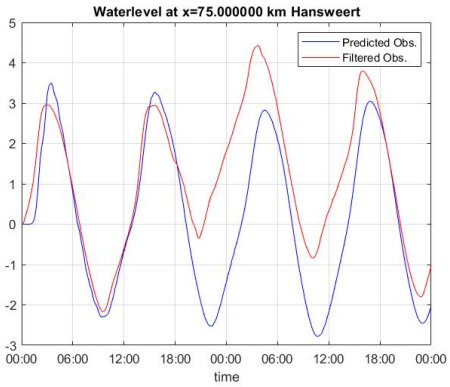
Instead in Figure 15 we show the results when we start our forecast $\Delta t = 12hrs$ before the moment t_{PEAK}:



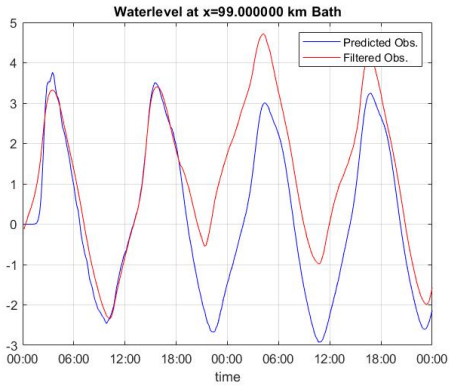
(a) Vlissingen.



(b) Terneuzen.



(c) Hansweert.



(d) Bath.

Figure 15: Forecast for lead-time $\Delta t = 12hrs$.

Qualitatively we observe that the overall behavior of the forecast is worse. In order to quantify this behavior, we compared the various the error committed.

In Table 8 we show the prediction errors made for the peak of the storm in the different cities[23]:

Table 8: Error of the forecasts for the peak waterlevel depending on lead-time

	Error Vlissingen	Error Terneuzen	Error Hansweert	Error Bath
$\Delta t = 0$	0.0576	0.0433	0.0351	0.1095
$\Delta t = 6$	-0.8769	-0.8876	-0.8670	-0.6406
$\Delta t = 9$	-1.6190	-1.5511	-1.4707	-1.1744
$\Delta t = 12$	-1.7636	-1.6860	-1.6002	-1.3020
$\Delta t = 24$	-1.7665	-1.6915	-1.6018	-1.3100

Finally, Figure 16 is the graph representation of the previous table, the error in the peak-level forecast as a function of the lead-time Δt:

[23]NB As we mentioned, t_{PEAK} is the moment of the overall waterlevel peak, and it is the moment that we used in our implementation of the forecast. For this table, we evaluated the error of the peak level in each city, at the time t_{PEAK}^{city} where the local peak waterlevel is recorded

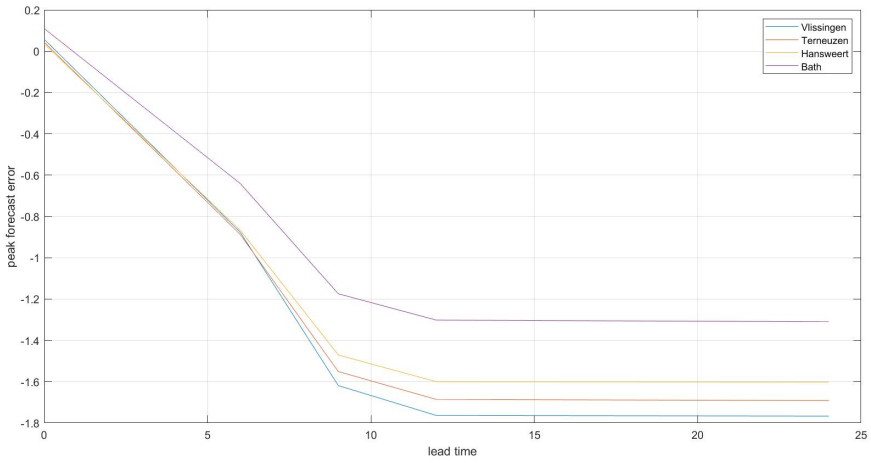


Figure 16: Waterlevel prediction vs Storm: dependence on the lead-time

10.1 How to further improve the results?

In essence, the forecast is a combination of running the Ensemble Kalman filter with assimilation of data (up to a certain time) and then running it without assimilation. Therefore, the two main entities that come into play are:

- The observations that we use in the data assimilation step (when they are used).
- The stochastic numerical model with which we model the phenomenon that we are studying.

For this reason, everything we mentioned in section 2 should be remembered: a more refined model will most likely lead to better forecasting. Moreover, as discussed in section 6, having more information on the uncertainty of the measurements should also lead to an improved performance in the forecast.

References

[1] D. Parrish and S. Cohn. "A Kalman filter for a two-dimensional shallow-water model, formulation and preliminary experiments". In: 1985.

[2] Rijkswaterstaat. *Rijkswaterstaat Waterinfo*. 2021. URL: %5Curl%7Bhttps://waterinfo.rws.nl/%5C#!/kaart/waterhoogte-t-o-v-nap/%7D.

[3] J.M et al. Lewis. *Data Assimilation: a Least-Square Approach*. Cambridge University Press, 2006.

[4] Rudolph Emil Kalman et al. "A new approach to linear filtering and prediction problems". In: *Journal of basic Engineering* 82.1 (1960), pp. 35–45.

[5] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications (Springer Texts in Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2005. ISBN: 0387989501.

[6] Deltares. *Matroos Standard Names*. 2021. URL: %5Curl%7Bhttps://publicwiki.deltares.nl/display/NETCDF/Matroos+Standard+names%7D.

[7] Burgers et al. "Analysis Scheme in the Ensemble Kalman Filter". In: (1998).

[8] J. Mandel, L. Cobb, and J. Beezley. "On the convergence of the ensemble Kalman filter". In: *Applications of Mathematics* 56 (2011), pp. 533–541.