Delft University of Technology

Performing data assimilation experiments with hydrodynamic models
A Java Sea case

G.C. Markensteijn MSc

TU Delft

Witteveen Bos

Performing data assimilation experiments with hydrodynamic models

A Java Sea case

by

G.C. Markensteijn MSc

A thesis submitted in partial fulfilment of the requirements for the degree of

Master of Science
in Applied Mathematics

at the Delft University of Technology,
to be defended publicly on Tuesday June 27, 2017 at 15:00h.

Student number:	4085906	
Project duration:	September 1, 2016 – June 27, 2017	
Thesis committee:	Prof. dr. ir. A. W. Heemink,	TU Delft
	Dr. ir. M. B. van Gijzen,	TU Delft
	Dr. ir. J. L. Korving,	Witteveen+Bos
	Dr. C. van Velzen,	VORtech

An electronic version of this thesis is available at http://repository.tudelft.nl/.
Cover image by Cynthia Boll [34]

TU Delft

Abstract

Floods are a big problem for Jakarta; the capital of Indonesia is subsiding below mean sea level and floods will become more frequent. Flood protection and measures against subsidence are, therefore, of high importance to Jakarta's government. One of the options they are studying is a great sea wall to regulate the water levels in Jakarta Bay. These projects have to be tested and should be build to withhold extreme water levels. In order to simulate these water levels one can use a hydrodynamic model or look at observations. In theory these could be combined by using data assimilation which should improve the hydrodynamic model in such a way that it predicts the results for both observed as unobserved locations quite good.

One of the most used data assimilation methods is the Ensemble Kalman Filter, which will be applied to an hydrodynamic model on the Java Sea. The model is simulated using Delft3D-FM and performs quite well as its results are in good correspondence with the observations. On this model data assimilation will than be applied, by making use of OpenDA, in order to further improve the model. With this set-up various twin-experiments will be performed to test the data assimilation method; some experiments with real observations will also be performed to test if it the set-up applicable in real projects.

The experiments with real observations perform on the same level as the twin-experiments; the observations thus seem to be suitable for using in real projects. Unfortunately, the twin-experiment does not perform well; the assimilation worsens the results of the hydrodynamic model in both observed, as unobserved locations. In the assimilation process only one component, the localization, seems to performing quite well. This performance can not be studied fully, because the twin-experiments do not give correct results. It will be shown that future work is needed to research the various problems encountered, and that at the time of writing the assimilation with OpenDA does not improve the Delft3D-FM model on the Java Sea.

Keywords: Data assimilation · Ensemble Kalman Filter · EnKF · Hydrodynamics · Java Sea · Jakarta · OpenDA · Delft3D-FM

Preface

During my Bachelors in Applied Mathematics, and during the courses of my Master of Science in Applied Mathematics, it became clear to me that I'm mostly interested in the modeling part of mathematics. In my Bachelors project I have applied this interest through a project on a hydrodynamic model on global tides; during this project I found that I am also interested in Hydrodynamics. I knew, therefore, I would like to this this in my Master thesis, which concludes my Master Applied Mathematics at Delft University of Technology, as well.

This goal of this thesis was formulated by Hans Korving, who was at that time an employer of Witteveen+Bos. My other supervisors were Arnold Heemink, full time professor at the TU Delft, and Nils van Velzen, employer of VORtech and the TU Delft.

I would like to thank Witteveen+Bos for the opportunity to experience working in their company. Furthermore, I would, especially, like to thank my supervisors for their help in completing this thesis. It was, in my opinion, a great cooperation and I really liked working with them.

G.C. Markensteijn MSc
Delft, June 2017

Contents

List of Figures

List of Tables

1

Introduction

Jakarta, located at the north-west of the island Java, is the capital and largest city of Indonesia. Jakarta's metropolitan area is the second largest in the world, after Tokyo's, and might soon pass Tokyo, as it is still one of the fastest growing metropolitan areas in the world [45]. This population growth is one of the reasons that Jakarta's economic growth has been above national average and stable in recent years [31]. However, its growth has also lead to a lot of problems including traffic congestion, sanitary problems and large problems with water quality. Furthermore, Jakarta is mostly situated below the mean sea level, this proportion is growing, as groundwater is extracted; leading to the city further subsiding below mean sea level. As large parts of the city are below mean sea level floods are very common; at least once a year parts of Jakarta are flooded by overflowing rivers and canals.

The impact of these floods will increase as the city grows, and, therefore, the Indonesian government has started various projects in order to reduce the flood probabilities. These projects are part of the 'masterplan Jakarta' [33] which introduces, not only solutions for the reduction of floods, i.e. a great sea wall [21], and soil subsidence, but also enhancements to the water supply system in order to improve the quality and quantity of water. In order to reduce the amount of floods the masterplan contains plans to build various islands and dams into the Jakarta Bay for protection of Jakarta itself. These plans have to be tested, and estimates on the hydrodynamics in Jakarta bay are, therefore, needed. These estimates can be derived in various ways. Firstly, you could start observing these conditions all over Jakarta Bay, however, in order to have a good overview of the dynamics, one would need various years of data. This is not feasible, because gathering this data would take to long and would delay the project, so existing observations should be used; the problem with these is ,however, that the available locations are often not on the desired locations. Also, there are only three of such locations around Jakarta bay. Finally, one could use models to estimate the hydrodynamics around Jakarta bay, although this is the most feasible solution, models still are a simplification of the real world and, therefore, models should be compared to observations.

One of the main parties working on such models for the masterplan is Witteveen+Bos. They created a model for the Java Sea (hereafter the Java model) and compared this with some observations [36]. They found that this method has its limitations, as you can only truly study the quality of the model on the small amount of observational locations

available. Next to this you can compare the model to known tidal components, as these are the main forcing of hydrodynamics in oceans. This will give an overview on how well the model performs; tidal dynamics do, however, not define the extreme water levels. These values are of high importance as these are the main predictor for giving probabilities to when, and how often dikes and other flood protection structures will be violated.

As observed locations are often far from the project locations, these are not sufficient to give a good prediction on water levels. Furthermore, a model, like the Java model, gives a general overview, but might not give reliable extreme values on project locations. That is why it would be best to combine the model and observational data into one improved model, which should generate the best results now available. Combining models with observations has been possible by Data Assimilation since research by [60]. However, only in recent years various software packages have become available that reduce production time of such methods. These packages provide a coupling between models and data assimilation methods, while in the past one had to code the data assimilation methods for each model by oneself. One of these packages is OpenDA, developed by various companies and institutions; i.e. Delares, VORtech, and the TU Delft. This package also includes the coupling with Delft3D-FM models. The research on the possibilities of combining these two software packages for the Java model is the topic of this thesis. OpenDA will be used to perform the Ensemble Kalman filter to the Delft3D-FM Java model. The main question that this thesis will try to answer is:

> **Research question:** Does data assimilation significantly improve extreme value analysis of the water levels in the Java Sea?

In the process of answering these questions various sub-questions will be answered first. These sub-questions are:

> **Subquestion 1:** Does the coupling between OpenDA and Delft3D-FM work for the Java model?
>
> **Subquestion 2:** Does data assimilation improve the Java model on observed locations?
>
> **Subquestion 3:** Does data assimilation improve the Java model on non-observed locations?
>
> **Subquestion 4:** Can data assimilation create a sufficiently long time series for extreme value analysis by making use of the Java model and available observational data?

The remainder of this thesis is structured as follows. In Chapter 2 an overview on previous studies will be given, which are closely related to the subject of this thesis. Next, various chapters will cover the various theoretical aspects of this thesis, starting with the theory on hydrodynamical modeling in Chapter 3. This chapter will also give an explanation of the software, and the Java model will be discussed. Finally, the chapter will

discuss various data sources needed for the Java model. In Chapter 4 a review of the data for observations used in the data assimilation will be given, and the final piece of theory, the data assimilation method, will be thoroughly explained in Chapter 5. The final chapters are reserved for the experiments, with, first, in Chapter 6 the methodology. Chapter 7 will than discuss the results of these experiments and, finally, in Chapter 8 various conclusions and recommendations will be given.

2

Previous studies

In the introduction, it has been stated that the goal of this thesis is to use data assimilation to improve the model results of the water levels in the Java Sea, and to study whether this could lead to a better extreme value analysis. In this chapter an overview will be given on past work in this field. First, an overview of data assimilation methods and the state of the art they are in will be given, this will, however not include any theory as that will be explained in Chapter 5. Next, a model of Singapore will be shown which uses a method that closely resembles the method that will be used in this thesis, and in the last section a model of the IJmond will be shown which had the same goal as this thesis.

2.1. Data Assimilation

The first research on data assimilation was published in 1922 by Richardson [60], after which many have studied data assimilation and many methods have been created. At the beginning, it was almost exclusively used for meteorology studies, but later it also became a very useful tool in other subjects like oceanography [4] and aerospace [23]. Currently, two main assimilation classes exist: the variational analysis methods and the sequential methods.

Within the variational analysis class, there are two methods that are widely used: the 3D Variational analysis method (3DVar) [10, 49, 50, 57] and the 4D Variational analysis (4DVar) [9, 44, 48, 51]. In this thesis, however, the focus will be on one set of the sequential methods; namely, the Kalman filters, specifically the Ensemble Kalman Filter. There are three main reasons for choosing the Ensemble Kalman Filter instead of the 4D Variational method for this thesis. The first two reasons are theoretical, while the last one is more practical. First of all, when using the Ensemble Kalman Filter it is possible to include model errors, which ensures better performance and more information when analysing results. Furthermore, the 4D Variational method needs an adjoint model, while this is not the case for the Ensemble Kalman Filter; this makes the Kalman filter less computational expensive while still keeping sufficient performance. The final reason is, that the Ensemble Kalman Filter is already included in OpenDA, while the 4D Variational method is not, which makes it more practical to use the Kalman filter. For these three reasons the Kalman filter has been chosen and, except for a small introduction, the 4D Variational method will not be thoroughly discussed in this thesis.

For the Kalman class of data assimilation, the first method is the classic Kalman filter, or Kalman-Bucy filter. This method was first introduced by Kalman [37], while the first implementation was by Schmidt [61]. The main idea is that it calculates the mean and covariance of the model values and that these values are updated by making use of observations. After its introduction, many have used, researched and updated the filter. Two additional implementations have been widely used in recent years: the Extended Kalman Filter [53] and the Ensemble Kalman Filter [22]. The first is an implementation that can also be used with non-linear models, while the second uses ensembles to calculate multiple different model possibilities. These ensembles can than be used to calculate the mean and covariance of the model state.

The Ensemble Kalman filter is currently the most widely used Kalman filter, and will also be the designated data assimilation method in this thesis. After Evensen et al. first introduced this method, others have followed to use, and upgrade it. One of the most important upgrades was incorporation of localization, where the work of Hamill and Whitaker [24] and Zhang and Oliver [70] are most notable. They incorporated two different methods that use a distance dependent filter to decrease the background error in large scale models. In Chapter 5 a thorough explanation of the various aspects of the Kalman filter will be given.

2.2. Singapore Model

It has been shown that the Ensemble Kalman Filter is one of the most used sequential data assimilation methods. This method is also the method used to improve the hydrodynamical model of waters surrounding Singapore, which will be reviewed in this section. The research on this subject is discussed in various papers by Karri et al. [39–41]. The hydrodynamic model they use is simulated by the Delft3D software package, while the Ensemble Kalman filter is performed by the OpenDA software package. In this research OpenDA will also be used to perform the Ensemble Kalman filter. The hydrodynamic model, however, will be simulated by Delft3D-FM, which is the successor of Delft3D.

The region that this model covers is the coastal waters surrounding Singapore, together with the Malacca Strait and small parts of both the Java Sea, as the South China Sea. The goal of their research is to improve the predictions for water levels in the region of interest. In order to research if this goal can be achieved three set-ups have been used. First, they start with a 1D model of the Malacca Strait with a connection to Singaporean waters [39]. On this model they perform a twin-experiment to validate if the goal of the thesis is feasible. They find that in such a simplified 1D test case data assimilation improves the results heavily, the optimal result has been found with an ensemble size of 32 that brings the water levels almost perfectly to the observed values; larger ensembles sizes only bring small improvements which are not worth the computational effort.

In the second set-up they perform a Steady State Ensemble Kalman Filter [40] on a coarse version of the Singapore Regional Model [43]. They again perform a twin experiment in order to test the models efficacy. They conclude that the twin experiment with 16 observation stations generates accurate estimates of water levels in both observed and unobserved locations, which is very promising. Finally, they also performed various forecast runs [41]; they conclude that their results are very promising, and also that the Ensemble Kalman Filter can greatly improve hydrodynamic models. Various limitations, however, are also mentioned; first of all, observed locations are mostly influencing

regional areas and not the entire model. Furthermore, the filter is limited by the amount of observed locations available; improvements are much worse when using less stations. These models show promising results for this thesis, however, it also shows that the reliability of the data assimilation is greatly depended on the amount of observed locations, and the quality of its data.

2.3. IJmuiden case

The last model that will be discussed in this overview, is the IJmond model considered in [26]. The method being used in this model is fairly different to the method used in this thesis. The model is created with a 3D version of WAQUA, TRIWAQ, which is a software package for the modelling of water levels and flows [32]. Where WAQUA simulates these components in mean values over one dimension, TRIWAQ can model the values in three dimensions. It uses a finite difference numerical scheme in solving the defining equations and needs boundary conditions for the forcing.

Data assimilation for this model is performed by making use of the Reduced Rank Square Root Kalman filter, which has been created for large scale models. This algorithm is based on the projection of the covariance matrix to a smaller subspace represented by leading eigenvectors [25]. This projection makes the algorithm more feasible for large scale models, as it significantly reduces the amount of computations. The filter is used to assimilate High Frequency Radar (HFR) measurements, and Acoustic Doppler Current Profiler (ADCP) measurements into the model. The goal of this model was to check the feasibility of data assimilation of large scale models. This makes it an interesting article for this research as this thesis' goal is closely related; the results of this paper can therefore be a forecast for the results of this thesis and might give some in sights in the various problems that might be encountered.

The conclusion of their study was that data assimilation works sufficiently. Furthermore, they found that high quality data is essential for the data assimilation to perform properly. In their case the ADCP data gave sufficient improvements in both locations close by, as locations further away in the model. The HFR data, however, did not show any improvements, because the HFR data was of lower quality. It is important to note that even before they performed the assimilation they already questioned the quality of the data.

It has been shown that data assimilation is feasible for large scale models; the quality of observations is, however, of great importance for the assimilation to work. This makes it very important to, first, make sure the method works by making use of a twin experiment, before the method is tested with real observational data.

3

Hydrodynamical modeling

3.1. Theory

Hydrodynamica is a subject in engineering that consists of any model that, in some way, represents the flow of water. This does not necessarily mean that such a model should be computational; physical models are included as well. In this thesis, however, the physical models are excluded and when a hydrodynamic model is mentioned, it will refer to a computational model.

In order to create a computational model, a set of equations to solve are needed. In the case of hydrodynamical models this set of equations consists of the Navier stokes equation for an incompressible fluid, the continuity equation, and some boundary conditions [46]. These equations represent the conservation of mass and momentum. There can, however, be exceptions; many hydrodynamic models have open boundaries, which means that water can flow out of the region of interest. Which can result in the non-conservation of mass and momentum in the region of interest.

For most cases this set of equations does not have an explicit solution; therefore, numerical models will be used to solve them. In order to solve such a model, either an existing software package can be used, or a new code can be created. The main differences between existing packages are on the kind of region of interest. Some focus on smaller water bodies like buckets, and tubes that are used at industrial sites. Others focus on larger bodies like rivers, lakes and oceans; these are the software packages that are interesting for this thesis.

Two packages were options for the hydrodynamic models in this thesis; namely, Delft3D [13] and its successor Delft3D Flexible Mesh (Delft3D-FM) [16]. The main difference between these packages is that the latter can make use of unstructured grids, while the first can only use structured grids. This offers the option to create a large coarse grid with only refinements at the area of interest; which is very useful when you have large model regions where large parts of the region are included for its forcing, while only smaller areas are important in the research. This is also the case in the Java model of Kaji et al., which will be explained in full in Section 3.3. This is, therefore, the main reason why Delft3D-FM has been chosen to use in this thesis.

3.2. Delft3D-FM

This section will explain various parts of the software package Delft3D-FM. First, how the Navier-Stokes equation is solved will be shown. Next, in Section 3.2.2, the incorporation of boundary conditions will be explained, while at the end, in Section 3.2.3, an example of dealing with an unstructured grid will be shown.

3.2.1. Shallow water equations

Delft3D-FM is specifically created for the modelling of geophysical water bodies. This specific purpose has lead to a more characteristic set of equations than the Navier-Stokes equation. As geophysical bodies, commonly, have a much lower depth than length, and width, the Navier-Stokes equation can be rewritten to the shallow water equations by making use of various assumptions, which can be found in the User Manual [15]. As mentioned, there are two important sets of equations: the momentum equations, which for the shallow water equations consist of three equations, and the continuity equation.

The two momentum equations in x- and y-directions are, in this case, defined as

$$\begin{aligned} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} - fv &= -\frac{1}{\rho_0}\frac{\partial P}{\partial x} + F_x + \frac{\partial}{\partial z}\left(\nu_V \frac{\partial u}{\partial z}\right) + M_x, \\ \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} + fu &= -\frac{1}{\rho_0}\frac{\partial P}{\partial y} + F_y + \frac{\partial}{\partial z}\left(\nu_V \frac{\partial v}{\partial z}\right) + M_y. \end{aligned} \tag{3.1}$$

Where M_x and M_y are the momentum contribution, in the specified direction, by hydraulic structures. In the case of the z-direction, only a pressure equation is left, defined as:

$$\frac{\partial P}{\partial z} = -\rho g h \tag{3.2}$$

Finally, for the conservation of mass, a depth-average approach is used. This results is the following continuity equation:

$$\frac{\partial h}{\partial t} + \frac{\partial Uh}{\partial x} + \frac{\partial Vh}{\partial y} = Q, \tag{3.3}$$

where U and V are the depth averaged water velocities, in x and y direction respectively, and Q is representing external contributions per unit area due to, for instance, precipitation and evaporation. The momentum and continuity equations are spatially discretized in a staggered manner, which means that velocities will be defined on cell faces and water levels on cell centers, to create a numerical model; a thorough explanation of the discretization can be found in the Technical reference manual [14].

These equations, however, do not define a complete model, because, as mentioned before, boundary conditions should also be concluded. These will be explained in the next section.

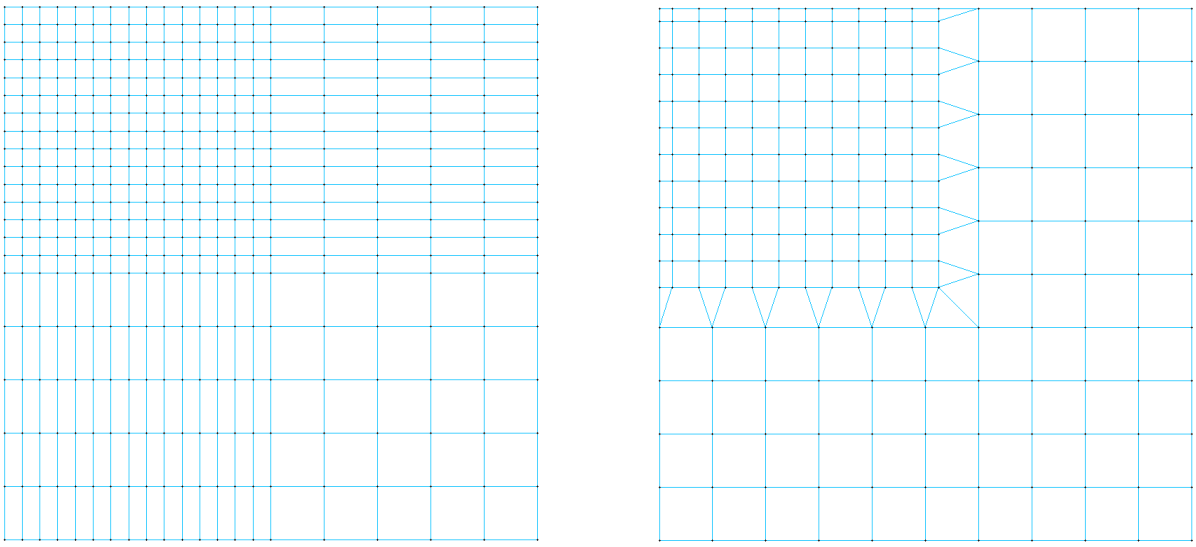
3.2.2. Boundary conditions

Various boundary conditions can be applied in Delft3D-FM; they can be summarized in three main classes:

- Open boundaries
- Vertical boundaries
- Closed boundaries

The open boundaries are quite important for this research. The main water dynamics in large scale ocean models are defined by ocean tides, which can be incorporated by using an external forcing, specific per location, on the open boundaries. The other two classes are, however, of no importance. The vertical boundaries will not be used, because the Java model is in 2D, with depth averaged values, and vertical directions are thus not used. While the quite large area of interest in the Java model has led to the choice of treating the closed boundaries as if no tangential stress is generated on them and the influence on water dynamics by closed boundaries can, therefore, be neglected.

3.2.3. Unstructured grid



(a) (b)

Figure 3.1: Examples of a structured grid and an unstructured grid, both with local refinement of the top left corner. These figures have been generated with the grid editing software RGFGrid [12], which is incorporated in the Delft3D-FM installation.

In Section 3.1, it was already mentioned that the main reason for choosing Delft3D-FM over Delft3D is the introduction of unstructured grids. Structured grids can only be refined in a very specific way, as in the case of structured grids all vertices need to have the same amount of links. Unstructured grids, however, can have different amount of links per vertices. This results in a grid that is less computationally efficient, but gives much more

possibilities for refinement without the extreme increase in the amount of grid notes and, consequently, the computation time. This introduction is, therefore, especially interesting when research makes use of large scale models, as a lot more grids are possible than when you are working with structured grids.

In Figures 3.1a and 3.1b two examples are shown; the first is of a structured grid, and the second of an unstructured grid. It can be seen that, in order to refine the top left corner, in a structured grid we have to refine all cells in the particular rows and columns, while in the case of the unstructured grid those cells can be refined, and connected to the complementary grid by making use of triangular cells. This results in a much lower amount of nodes when working with large grids. In the case of Figures 3.1a and 3.1b the difference is already quite large; the grid on the left has 400 nodes while the grid on the right only has 229 nodes. Thus choosing for a unstructured grid makes it much easier to refine specific areas of interest in the model. One should, however, take note that the boundaries between areas with different resolutions should not be too complicated; these boundaries will be less orthogonal, which will lead to larger modelling errors at those locations.

3.3. The Java model

In this section the computational grid and boundary conditions of the Java model by [36] will be shown, together with various characteristics, and time-series. Other components, which are based on various data sources, like the bathymetry and tidal components, will be explained in Section 3.4.

3.3.1. Computational grid

Figure 3.2: This shows the full grid of the Java model with grid resolutions between 0.1° and 0.01°.

In Figure 3.2 a plot of the Java model computational grid is displayed; This grid has been generated such that a low resolution can be found at deeper waters, while shallow water and complex geometries have a higher resolution. This refinement has been

performed by making use of the bathymetry that will be shown in Section 3.4.1. This method has been used, because small local disruptions have a larger effect on water dynamics in shallow waters than in deep waters. Therefore, a fine grid is not, necessarily, needed in deep water, while in shallow waters it can greatly improve the model. The resulting minimum resolution is 0.1° in deeper water, while the maximum resolution can be found in the Jakarta bay with a resolution of approximately 0.01°. This can be seen in Figure 3.3, where only the grid surrounding Jakarta bay has been displayed. In the left corner, which is a small part of the Indian Ocean, of the figure cells are located with lowest resolution (0.1°) while around Jakarta bay, in the middle, the highest resolution can be found (0.01°). This part of the grid shows very clearly how grid refinement, while taking into account water depths, works. This technique results in a grid with approximately 75000 nodes; where if a structured grid would have been used, either the resolution in areas of interest would not have been sufficient, or the number of grid points would have been a multiple of the current amount; both undesirable.

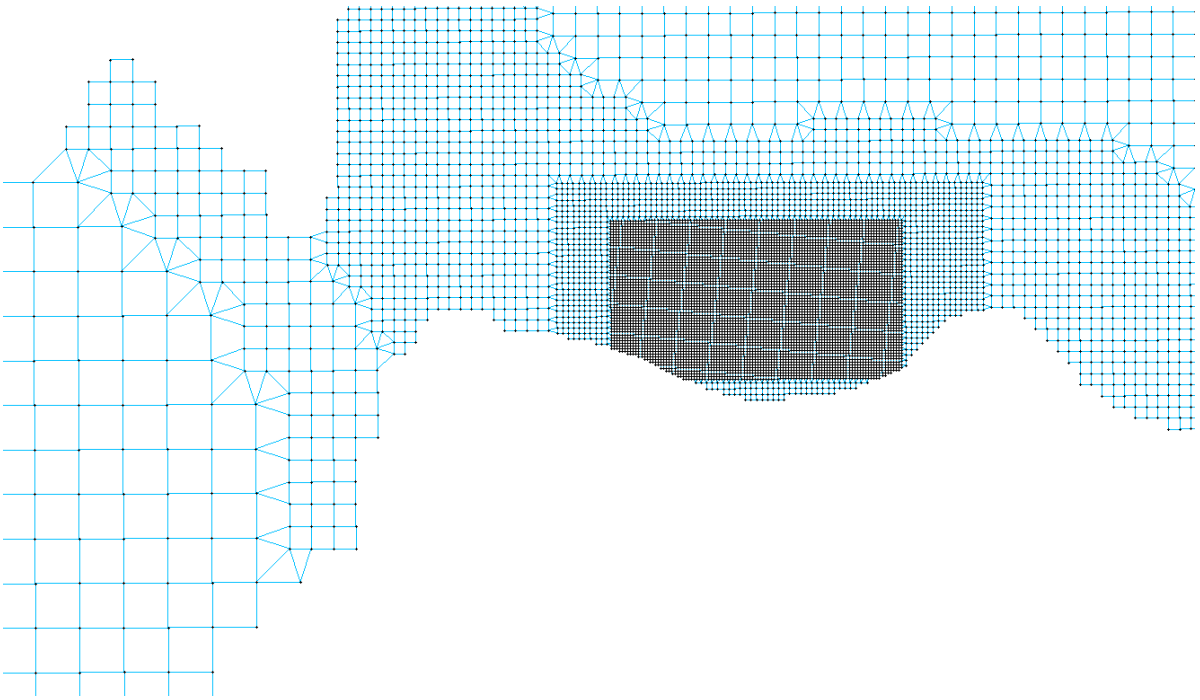


Figure 3.3: This figure displays the grid around Jakarta bay.

3.3.2. Boundary conditions

The boundary conditions for the model consist of two components. The first component is the forcing of the model; namely, the astronomic tides. These have been calculated with the TPXO model [19], which will be explained further in Section 3.4.2. It includes the eight primary constituents, as well as four higher and one long period constituent.

The second component deals with the inverted barometric response [17], which shows the dependency of the ocean to atmospheric pressure change. When the atmospheric pressure change is given by $p'_a(t)$, in millibars, the oceanic sea level change is calculated by the following equation

$$\eta(t) = -\frac{p'_a(t)}{\rho_0 g}. \tag{3.4}$$

In this model the atmospheric pressure change will be calculated by

$$p'_a(t) = P - \overline{P}, \tag{3.5}$$

where P is the atmospheric pressure and $\overline{P}$ is the average pressure over the global oceans, which, in correspondence with [36], has been set at 1,013.3hPa.

3.3.3. Characteristics

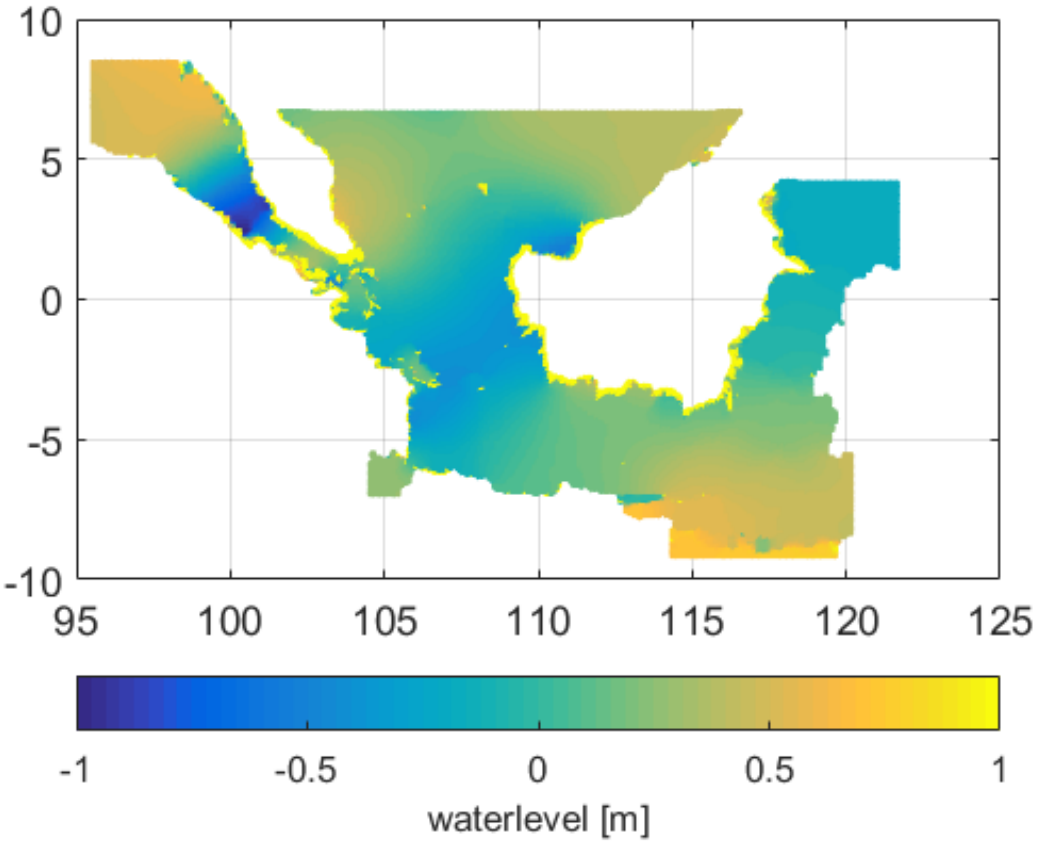


Figure 3.4: A plot of the water levels at May 30th 2015, in the Java model region, simulated with Delft3D-FM.

Finally, in order to get an insight on some of the model characteristics, some figures with model outputs will be shown. First, in Figure 3.4, the water levels of the model output for one time step can be found, this gives a better overview of the model area, and shows the differences in water level, with respect to mean sea level, through the model. The area consists of the Java Sea, various straits that connect the model to the Indian and Pacific Ocean, and finally part of the South China Sea. This region has been chosen in such a way that the boundary conditions, as specified in the previous section, are located in deeper waters, which is where the tides force the main hydrodynamics. Next, in Figure 3.5, the

water levels at three locations have been shown. For each location two months of data has been plotted to give an overview on the changes in water heights during such time period. The locations are Benoa at Bali and Jakarta at Java, both in Indonesia, and Penang, located at the west coast of Malaysia. It can clearly be seen that the tides are forcing the model, as the main trend in water heights show strong daily and monthly periodic components. Furthermore, at Benoa, which is close to the Indian Ocean, the tides show larger amplitudes, than at the other two locations, which are located at the Java Sea and Malacca Strait respectively. The final characteristic that stands out is, that in Penang there seems to be a cut-off level for the water levels. This can be explained by its location, as the measurement location is located in a small strait between a small island and the main land of Malaysia. The depth on this location is, therefore, quite low and that means that the water dynamics will mainly consists of surface waves.

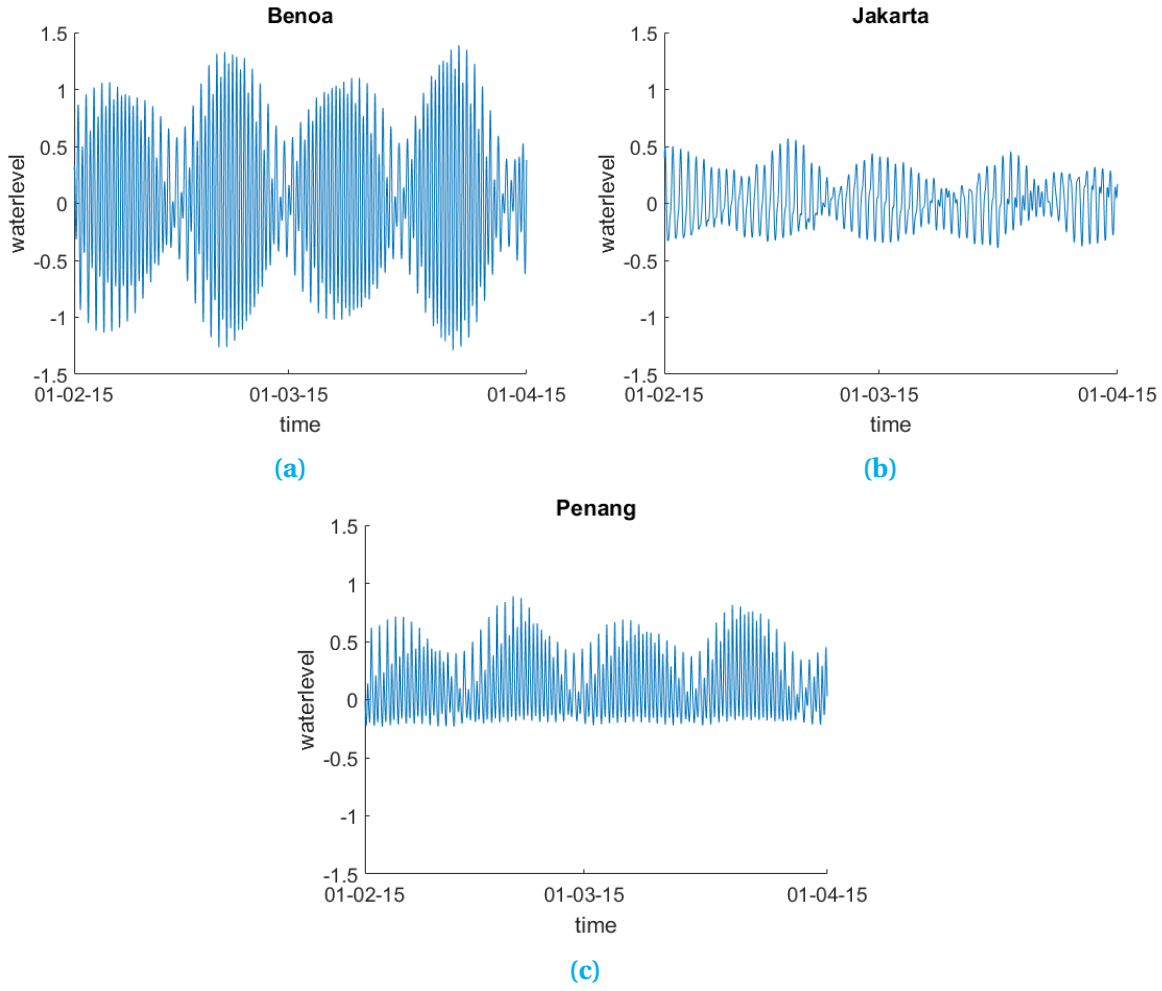


Figure 3.5: The water levels propogating through time at a) Benoa b) Jakarta and c) Penang.

As mentioned, these figures show strong monthly and daily periodic trends. These trends are generated by the tidal components. To give a better overview of the fluctuations in water levels, these components will be removed from the time series by making use of a harmonic analysis. This harmonic analysis is now performed on the time series for Benoa, by making use of the Matlab tool t_tide explained in [58]. The resulting residual water

levels can be found in Figure 3.6; it is clear that the tidal constituents indeed are the main component of the water level fluctuations, as the new plot shows much smaller differences. These differences have mainly been generated by the wind forcing in the model. The tidal amplitudes will not change much over time, while the residual water levels will change when the wind changes. These residuals are very important when looking for extreme values as extreme values will normally occur during storms and are generated by the extreme wind speeds. So large fluctuations in wind speeds will results in larger fluctuations in water levels; these fluctuations can be found within the residual water levels.

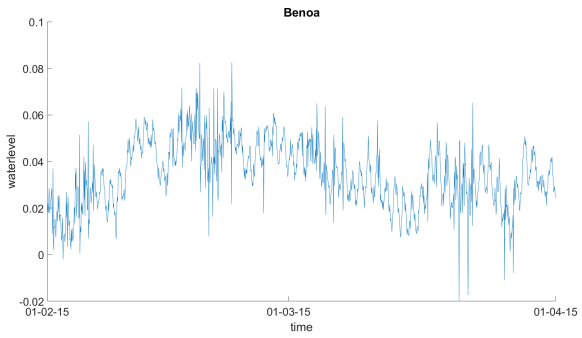


Figure 3.6: A plot of the residual water levels at Benoa after removing harmonic components.

When working with models it is important to validate the quality of its results. This validation will be performed by studying the two components mentioned before; the tidal components of the resulting time series will be compared with the TPXO model. Furthermore, observational data, see Chapter 4, will be used to validate both the residual fluctuations as the tidal components. Lastly, various model parameters should be set in Delft3D-FM. Some important parameters and its values have been shown in Table 3.1; these values will, however, not be discussed further.

Table 3.1: Model presets

Model parameter	Parameter value
Maximal Courant number	0.7
Friction type	Manning
Friction coefficient	0.026
Average water density	$1,025 kg/m^3$
Gravitational acceleration	$9.81 m/s^2$
Wind drag coefficient type	Smith and Banke [63]
Wind drag coefficients	0.00063 and 0.00723
Time step	$600 s$

Notes: In this table, various model parameters are shown with the values used in Delft3D-FM. These values are in correspondence with [36] and are often the suggested values by the Delft3D-FM manual [14].

3.4. Input data

This section will be used to explain the various data sources used in generating the bathymetry, wind and tidal components of the Java model. First the data needed for the geometry will be shown, next those essential to the boundary conditions, and finally those that are needed for the atmospheric forcing.

3.4.1. Geometry

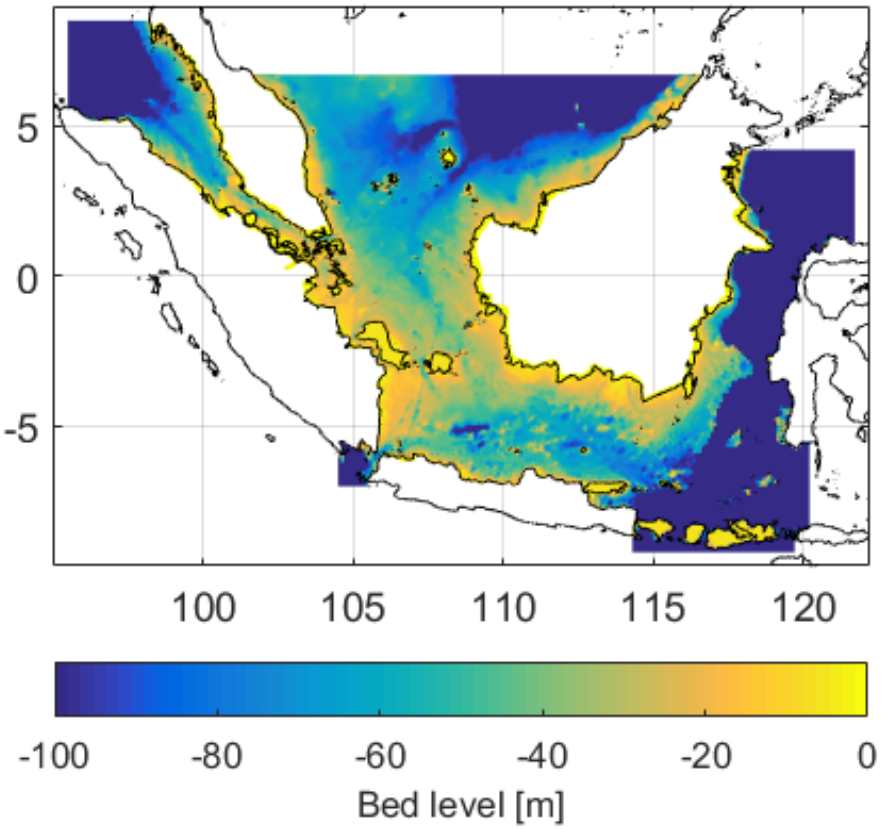


Figure 3.7: A plot of the bathymetry data of the Gebco database for the Java model.

One of the most important components of any hydrodynamic model is the geometry of the model; in the case of the Java model this consists of the bathymetry. The bathymetry is the seabed level of the grid with respect to the mean water level. The bathymetry data for the Java model is based on the General Bathymetric Chart of the Oceans (GEBCO) database [6]. This database consists of grids of various resolutions. In the case of the Java model, the bathymetry has been based on the 30 arc-second grid [5]. This grid has been generated by combining bathymetry data from various sources. The main components used are quality-controlled ship depth soundings and satellite-derived gravity data, while other, smaller, data sets have been included to improve the grid at various locations. In the region of interest of the Java model some of these smaller data sets are for instance the Electronic Navigation Charts [62] and a data set gathered by HMS Scott [28]; a full list of data sources can be found in the GEBCO documentation. A

schematic overview of the resulting bathymetry can be found in Figure 3.7. This is also the bathymetry used for refining the grid in Figure 3.3 and can be used to get a better understanding on the refinement when needed.

3.4.2. Boundary conditions

In Section 3.3 it was already mentioned that boundary conditions are important, and that the boundary conditions of the Java model consist of tidal components. The tidal components for this model have been generated from the TPXO model [20]. The model uses a system for inverse modeling, which uses a least squares method to find a best fit model to the Laplace tidal equations with TOPEX/POSEIDON data. A thorough explanation of the model used can be found in [19], this also includes an overview of the TOPEX/POSEIDON data. The model output gives insight in the constituents, and its amplitude and phase in the region of interest. All main constituents are included in the analysis, and some secondary constituents as well. These together give a good overview on the tidal components and are, as mentioned before, also useful in validating the model.

3.4.3. Atmospheric forcing

The last important component of the Java model, that will be discussed, are the components needed for the atmospheric forcing. Three components will be used; the wind in longitudinal and latitudinal direction, and the atmospheric pressure. The data for these components has been downloaded from the ERA-interim database of the European Centre for Medium-Range Weather Forecasts (ECMWF) [18]. ECMWF is an intergovernmental organization that produces weather forecasts for its members. Other than forecast they also use their data to perform reanalysis of previous years. A reanalysis uses historical data to perform data assimilation on a long period of time; the main purpose of such a analysis is to generate a long, consistent range of results that give the best possible results taking into account the errors on the model and data available. The reanalyses performed by ECMWF aim to provide the most accurate description of the atmosphere available; this can be used for historic meteorological studies. It, therefore, also provides an accurate description of the history of the atmosphere in terms of wind and atmospheric pressure, which are components that are often important in large scale hydrodynamic models.

One of the reanalyses, performed by ECMWF, is the ERA-Interim model, this is a global reanalysis of atmospheric data from 1979 until the present. It is currently maintained in real time and uses data assimilation to improve the model continuously. A thorough explanation of the ERA-Interim model is given in a paper by [11]. This paper gives descriptions of the model, the data assimilation method, and the observations database that has been used.

4

Observational data

The data sources needed for the Java model in Delft3D-FM have now been introduced, however, during the validation of the model, and, in order to perform the data assimilation, observational data is needed as well. Which data has been used in this thesis will be explained in this chapter.

The observations used in this thesis have been retrieved from the University of Hawaii Sea Level Center (UHSLC) [8]. The UHSLC supports oceanographic and climate research in various ways; one way is supporting hydrodynamical research by maintaining a global network of tide gauge stations. The observations these tide gauge stations generate, are analysed and the data sets are distributed through their website. The data they make available is split into two groups: fast delivery data and research quality data. The quality control of the first data set only consists of removing clear outliers, and large level shifts. The advantage of this set is that it becomes available quite fast; within two months after collection. The data is, however not of such quality that it can instantly be used for research. The second dataset has received a full quality control, and is thus ready to be used in research as soon as it is released. The problem with this quality control is, that it takes quite a long time. This results in the research quality dataset only being available upto two years after collection.

As mentioned, these data sets are retrieved from a global network, which means the tide gauge locations within the Java models region have to be extracted from a large data set. These locations have been found by comparing the coordinates of all locations in both the fast delivery, as the research quality data set, to the model region. This resulted in approximately 30 locations with research quality data and 9 locations with fast delivery that lay within the Java model region. As previous studies showed that high quality data is important when performing data assimilation, it was decided to try and find a sufficient set of observations from the research quality. The fast delivery data will only be used when the research quality data does not lead to a sufficient set of observations. This will most likely lead to a higher quality set of water levels and will not, necessarily, lead to less observed locations, as each location that contains fast delivery data has also got research quality data available.

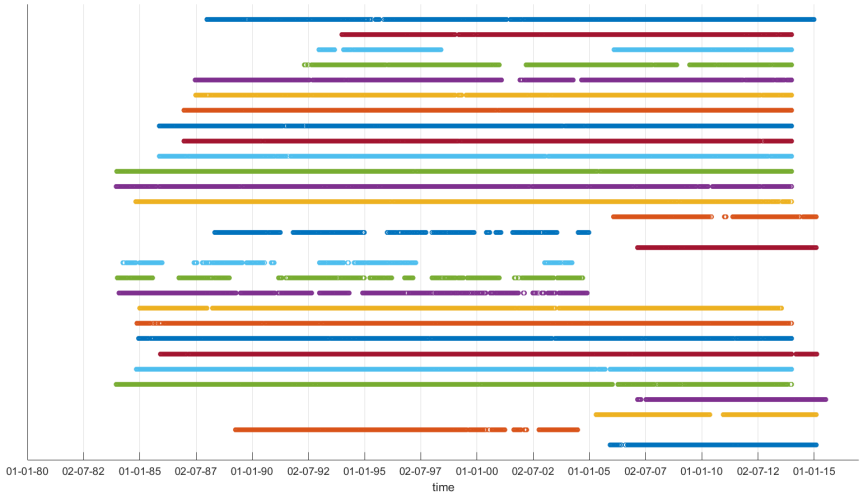


Figure 4.1: This graph shows on what time intervals data is available for each observed location; every row is a different location. So when a row is empty this means that it contained missing values.

The locations retrieved from the UHSLC database now have to be studied to see what data is available and what parts are useful in this research. The first thing that has been done is to make sure that the observed locations at least contained data of this century, if this is not the case the location will be removed from further analysis. This needed to be done, because more locations have been added over time, which lead to a larger set of observations locations for more recent years. Furthermore, some locations got more than one file in the database and this way the files that do not contain recent data will be removed. The next steps were focused on the amount of data available in recent years. As the files contained a lot of missing values it was needed to see how much of the observation times do actually contain numerical values. In order to get an overview of these numbers, a plot was created of the locations that contained observations in this century. Figure 4.1 displays the plot, where the empty spots resemble missing values. Looking at the figure it becomes clear that some locations contain quite large periods without observations, while others show an almost complete 30 year period of data. Furthermore, the time period between approximately 2007 and 2014 shows the most promising dataset; it provides the largest set of locations without large disruptions. This period will be analysed in the remainder of this chapter, and the locations without observations after 2007 will thus be removed, which means that the analysis will continue with 24 locations. The next step was to find the selections of data of these locations without any disruptions. It was found that various periods between 2007 and 2013 contains large sequences without any missing value. The specific period that will be used with the data assimilation experiment will depend on the length of the time-period modelled and will be explained in Chapter 6.

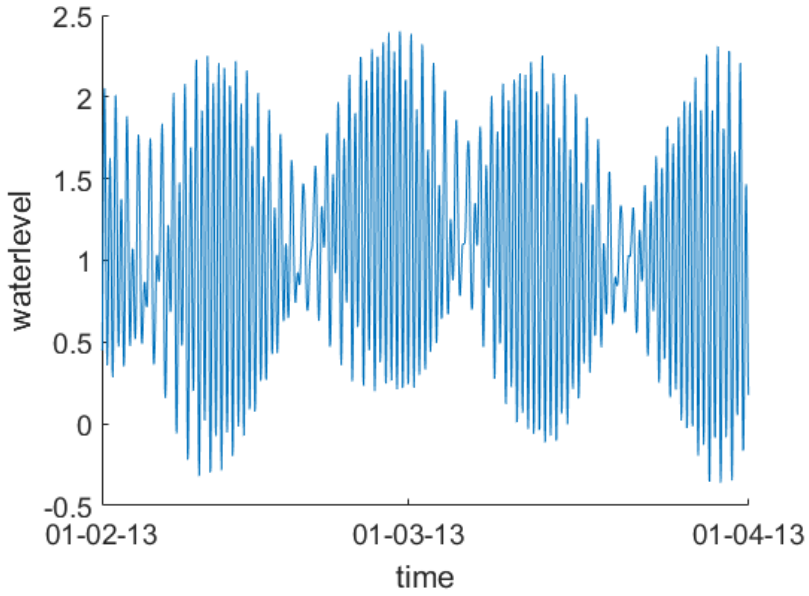


Figure 4.2: This figure shows the water levels at one observed station, Benoa, for 2 months in 2013.

The final aspect of the observational data that this chapter will discuss, is the data itself. In Figure 4.2 a plot has been shown of the observations in Benoa for two months. The observed data for Benoa does, unfortunately, not contain values for 2015 as was modelled and shown in Section 3.3.3. However, the figure still gives a good overview on what the data looks like. Furthermore, some of the characteristics in this figure can also be seen in Figure 3.5a; both show strong monthly and daily periodic components. One big difference is, however, the offset of the data; in the modelled values the water levels are oscillating around zero, while the observed values oscillate around approximately one. The offset can be found in every observed locations and differs between these locations. The reason for these different offsets is that water levels are,normally, oscillating around mean sea level. The mean sea level can, however, be calculated in many different manners; for the Java model the mean comes from the GEBCO database; mean sea level is the same as having zero depth. In the case of the observations it is not clear how the mean has been calculated. This does, however, not mean that these two can not be compared; the main components that should be compared are, as mentioned before, the tidal components.

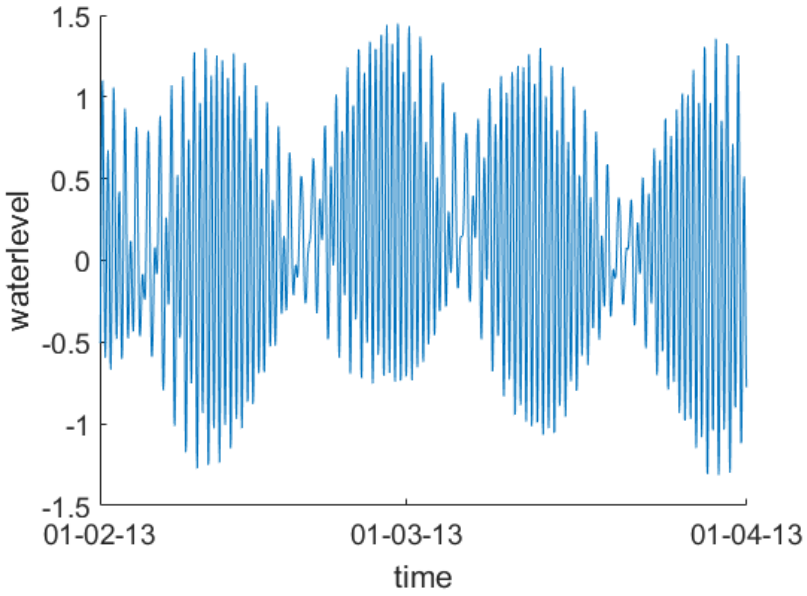


Figure 4.3: This figure shows the adjusted water levels at one observed station, Benoa, for 2 months in 2013.

These components can be retrieved from both models and, as it are the amplitude and period that will be compared, do not take into account the offset. Furthermore, in the residuals of the tidal analysis the offset will also be removed, which means that these can also be compared to each other. The offset is, unfortunately, a problem for the data assimilation, as in the assimilation the values of the model and the observations will be compared and than the offset can make a huge difference. However, as the offset does not define the time series characteristics it is possible to remove the offset of the data sets without compromising the results. The easiest and most logical way to remove such an offset, is to take the mean of the time series and subtract that from all values in the time series. For Figure 4.2 this results into Figure 4.3. It can be seen that the plots are completely the same except for the values on the y-axis, which shows that the offset can indeed be subtracted without losing any characteristics of the time series.

5

Data Assimilation

In this chapter a thorough explanation of the data assimilation method will be given. First, some basic concepts will be shown. Next, a thorough explanation of the Kalman filter, which has been used in this thesis, will be given. Finally, in the last section, an overview of the software package used will be given, this includes an explanation on the incorporation of the Kalman filter.

5.1. Theory

Data assimilation is an analysis technique that has, as stated in [55], the aim to:

> use measured observations in combination with a dynamical system model in order to derive accurate estimates of the current and future states of the system, together with estimates of the uncertainty in the estimated states.

Two classes of data assimilation methods are commonly used: the variational and the sequential methods [4]. Variational methods use optimization methods derived from control theory, to find the minimum of a cost function, in estimating the unknown parameters. This cost function measures the data misfit of a model and its minimum contains thus the best fitting parameters. In the case of variational methods that are specific for oceanography, adjoint models, although they expect a perfect model, are often used. Sequential methods, however, use a probabilistic approach to find various estimates of system states. The results of these methods consist of the mean of these states, which combined with its covariance gives a complete overview of the system state. These methods perform forward in time and only use data from the past and the present. Variational methods, however, can also move backwards in time and are thus also useful for a reanalysis of past model results. Due to this reason variational methods result in a continuous state of the model during the assimilation window as can be seen in Figure 5.1. The results of sequential methods are, however, part-wise continuous with respect to time.

The gain in using sequential methods, lies in the fact that there is no need for an inverse or adjoint method; this makes it more computational efficient in updating the model. Furthermore, in the case of an extreme value analysis of a data set there is also the need for a probabilistic distribution, which makes the sequential methods more suitable for this kind of research. This is, as mentioned before, one of the reasons that in this

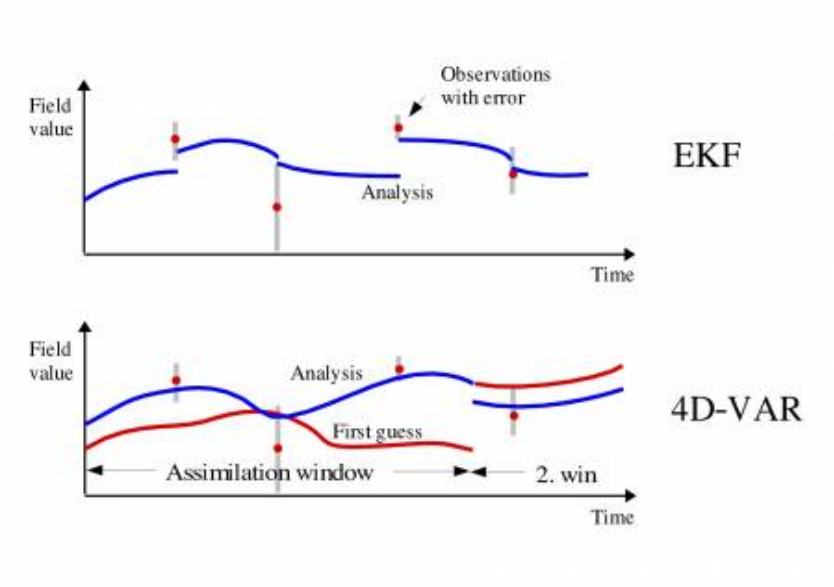


Figure 5.1: In this figure examples of both a sequential (The Extended Kalman Filter, above) and a variational (The 4D Variational method, below) data assimilation method are shown.[47]

research the focus is on sequential methods, specifically the Kalman filters. These filters will be explained thoroughly in Section 5.2.

5.1.1. Definitions

When working with data assimilation various concepts must be taken into account. Some of these will, therefore, be explained in this section.

One of the most important concepts in data assimilation is the state vector $\mathbf{x}$, which contains the state of the model at any given time [69]. This state can be represented by the values of various model variables, like the wind speed and the water level, and is related to the true state of the model. The exact relation between the true state and the state vector depends on the discretization method used.

Furthermore, data assimilation can be a computational expensive technique, especially when modeling larger areas. Therefore, control variables are often introduced; such a variable will either be kept constant or its resolution will be lowered. The first does often happen with variables like the gravitational acceleration g or the sea water density ρ_0. While the second happens, for instance, with the wind, as there is often not enough data available to consistently calculate wind fields for the whole domain. Various important variants of the state vector can, therefore, be defined[7]:

- The a priori state vector $\mathbf{x}_{k|k-1}$, with all background information
- The analysis state vector $\mathbf{x}_{k|k}$, with the analysis components
- The true state vector $\mathbf{x}^t$, with the true values of the analysis components

With these different variants of the state vector an alternative state space can be created. In this state space the optimal analysis increment $\partial\mathbf{x} = \mathbf{x}^a - \mathbf{x}^b$ should be found such that the analysed values will be as close to the true values as possible.

The final concept this section will introduce is the difference between model states and observations. Normally, the amount of observations is much lower than the size of the state vector, therefore, an operator, to map the state vector to the observed locations, is needed. This operator is the observational operator $H(\mathbf{x})$; it maps the state space onto the observation space under the assumption that no modelling errors exist. What the comparison of the state vector with the observation vector $\mathbf{y}$ resembles depends on the variant of the state vector that will be used. When the true state vector is used the difference, $\mathbf{y} - H(\mathbf{x}^t)$, will show the measurement error, however, as the true state of a model is not commonly known, these can not be calculated. It is more useful to use the a priori or the analysis state vector. These calculate, respectively, the innovation vector, $\mathbf{i}$, and residual error vector, $\boldsymbol{\varepsilon}^a$. The various symbols that have been defined in this section are for the time-independent problem; most problems, however, are, like the problem studied in this research, time-dependent. When this is the case, a subscript k will be added to each symbol for it to represent the corresponding definition at time k; i.e. $\mathbf{x}_k^t$ resembles the true state vector at time t_k.

5.1.2. Java model

In this section a short explanation on the state vector for the Java model will be given. This is needed for a better understanding how the equations for the Kalman filters, which will be explained in the next sections, are applied to the Java model. The state vector for the Java model consists of two parts: the water levels and the water velocities in longitudinal and latitudinal direction. Now the shallow water equations, Equation (3.1), needs to be rewritten, which will be done by making use of a time-discretization. This gives the following equations:

$$\begin{aligned}
\frac{u_{k+1}-u_k}{\Delta t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} - fv &= -\frac{1}{\rho_0}\frac{\partial P}{\partial x} + F_x + \frac{\partial}{\partial z}\left(\nu_V \frac{\partial u}{\partial z}\right) + M_x, \\
\frac{v_{k+1}-v_k}{\Delta t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} + fu &= -\frac{1}{\rho_0}\frac{\partial P}{\partial y} + F_y + \frac{\partial}{\partial z}\left(\nu_V \frac{\partial v}{\partial z}\right) + M_y, \\
\frac{h_{k+1}-h_k}{\Delta t} + \frac{\partial Uh}{\partial x} + \frac{\partial Vh}{\partial y} &= Q,
\end{aligned} \tag{5.1}$$

Now using the state vectors $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ this can be written into a non-linear system of equation:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k), \tag{5.2}$$

where, with $\mathbf{x}_k = [u_k, v_k, h_k]^T$,

$$f\begin{pmatrix} u_k \\ v_k \\ h_k \end{pmatrix} = \begin{cases} u_k & -\frac{1}{\Delta t}\left[u_k\frac{\partial u_k}{\partial x} + v_k\frac{\partial u_k}{\partial y} + w\frac{\partial u_k}{\partial z} - fv_k + \frac{1}{\rho_0}\frac{\partial P}{\partial x} - F_x - \frac{\partial}{\partial z}\left(\nu_V\frac{\partial u_k}{\partial z}\right) - M_x\right] \\ v_k & -\frac{1}{\Delta t}\left[u_k\frac{\partial v_k}{\partial x} + v_k\frac{\partial v_k}{\partial y} + w\frac{\partial v_k}{\partial z} - fu_k + \frac{1}{\rho_0}\frac{\partial P}{\partial x} - F_y - \frac{\partial}{\partial z}\left(\nu_V\frac{\partial v_k}{\partial z}\right) - M_y\right] \\ h_k & -\frac{1}{\Delta t}\left[\frac{\partial U_k h_k}{\partial x} + \frac{\partial V_k h_k}{\partial y} - Q\right] \end{cases} \tag{5.3}$$

5.2. Kalman filters

This section will explain the specific set of sequential data assimilation methods that are important for this research. This set consist of the Kalman filters of which the Ensemble Kalman Filter will be used. First, however, the Kalman-Bucy filter will be explained, next various successors and, finally, the Ensemble Kalman filter.

5.2.1. Kalman-Bucy filter

The main principle of the Kalman-Bucy filter [38], hereafter the Kalman filter, is that there is a system, for instance a model, and measurements, together with their specific uncertainties; a visualization of such a system can be found in Figure 5.2. Together these will be the input of the Kalman Filter, which than compares the system state with the measurements. The filter will than optimize the results of the system and update its parameters. The new system state should be a better representation of the, measured, reality.

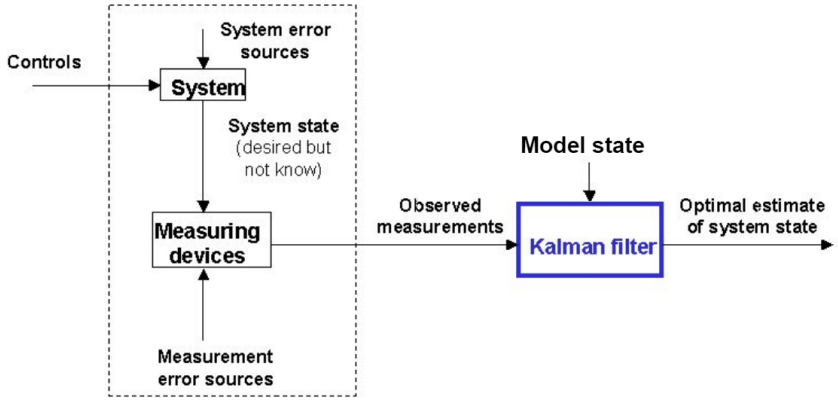


Figure 5.2: Schematic visualization of a system that makes use of a Kalman Filter. [59]

The Kalman filter is a linear filtering approach; the basis for this approach has been created in 1960 by R.E. Kalman [37]. Other researchers, such as Stanley Schmidt [61] and Richard Bucy [38], have used and updated his approach before reaching to the current filter. The filter was originally applied to applications within the aerospace sector, but later it became important in other sectors, like oceanography [4], as well.

The Kalman filter can, as suggested by Schmidt [61], be divided in two important parts: the time update and the measurement update. This separation makes it easier to implement, and will be used in the the following section while explaining the filter. The filter is used when working with linear systems and, therefore, in order to explain the filter, a general linear system will be used. This system will consist of two equations; the first equation is the model equation for the state vector, $\mathbf{x}$, and is given by

$$\mathbf{x}_{k+1} = A_k \mathbf{x}_k + B_k \boldsymbol{\alpha}_k + G_k \mathbf{w}_k \tag{5.4}$$

While the second equation will specify the measurements, part of the true state, $\mathbf{x}^t$, that will be included in the system. This equation reads

$$\mathbf{y}_k = M_k \mathbf{x}_k^t + \mathbf{z}_k, \tag{5.5}$$

In this system the matrices A_k, B_k and G_k, and vector α_k define the model dependencies between different times and locations. Furthermore, M_k is the observation matrix which maps the state vector onto the observed locations. Finally, $\mathbf{w}_k$ and $\mathbf{z}_k$ are white noise processes of the model and measurements respectively; they specify the uncertainty in the filter. These are generated from a Gaussian process with zero mean and, respectively, covariance matrices Q_k and R_k. To what components the noise models will be applied, and what kind of covariance matrix will be chosen depends on the model, the filter, and the software chosen; this concept will, therefore, be explained more thoroughly when introducing the software package in Section 5.4.

The time update of the Kalman filter is for such a linear system represented by two equations [59, 69], the first of these is given by :

$$\hat{\mathbf{x}}_{k|k-1} = A_{k-1}\hat{\mathbf{x}}_{k-1|k-1} + B_{k-1}\alpha_{k-1}. \tag{5.6}$$

This equation predicts the state vector $\hat{\mathbf{x}}$ at time t_k while taking the predictions and measurements of times $t_0 \dots t_{k-1}$ into account. These values will be calculated by making use of a Gaussian noise and will, therefore, not be deterministic. This means that the values computed with the first equation are just the mean of a probability distribution, and, in order to include all information a covariance matrix is also needed. This matrix will be calculated with the second equation of the time update, which is defined as:

$$P_{k|k-1} = A_{k-1}P_{k-1|k-1}A_{k-1}^T + G_{k-1}Q_{k-1}G_{k-1}, \tag{5.7}$$

As mentioned, this was the first part of the filter, the second part is the measurement update, which again consists of two equations; one for the mean state vector and one for the covariance matrix. The equations are

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k\left[\mathbf{y}_k - M_k\hat{\mathbf{x}}_{k|k-1}\right], \tag{5.8}$$

for the state vector, and

$$P_{k|k} = [I - K_k M_k]\, P_{k|k-1}, \tag{5.9}$$

for the covariance matrix. For the state vector first the difference between the measurements and the corresponding predicted values will be calculated. Next, it will be multiplied with K_k, which will be explained shortly, and, finally, that value will be added to the prediction of the state vector for the specified time t_k.

Now the K_k term will be explained, it is called the Kalman gain and is calculated by

$$K_k = P_{k|k-1}M_k^T\left[M_kP_{k|k-1}M_k^T + R_k\right]^{-1}. \tag{5.10}$$

The Kalman gain is one of the most important parts of the Kalman Filter as it makes the filter more robust. For an easy way to show this robustness the scalar case of the Kalman gain will be considered. In this case the Kalman gain can be rewritten to the following form:

$$K_k = \frac{P_{k|k-1}M_k^T}{M_kP_{k|k-1}M_k^T + R_k}. \tag{5.11}$$

As mentioned before R_k is the covariance matrix of the white measurement noise, which, in the case of scalar values, is just the variance. If there is large variance, it means that the uncertainty of the measurements is high. In that case it is undesirable for the measurements to have a large influence on the final predictions and, therefore, the gain from these measurements should be lower. This is what also happens in the non-scalar case of the Kalman gain; when R_k contains large values the denominator will be large and the Kalman gain will thus be small. However, when R_k equals zero, so the measurements are perfect, the Kalman gain will become one; this means that Equation (5.8) will read

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{y}_k - M_k \hat{\mathbf{x}}_{k|k-1}. \tag{5.12}$$

In that case the predictions on observed locations will become equal to the measurements. The Kalman gain, thus, specifies the impact of the measurements on the predictions. This means that when the measurements are very uncertain, the Kalman filter will not update the modelled values in accordance with the measurements and the output will be the same as the output of the time update. When the measurements are, however, fairly certain, the Kalman Filter will make sure the output of the time-update is updated correspondingly.

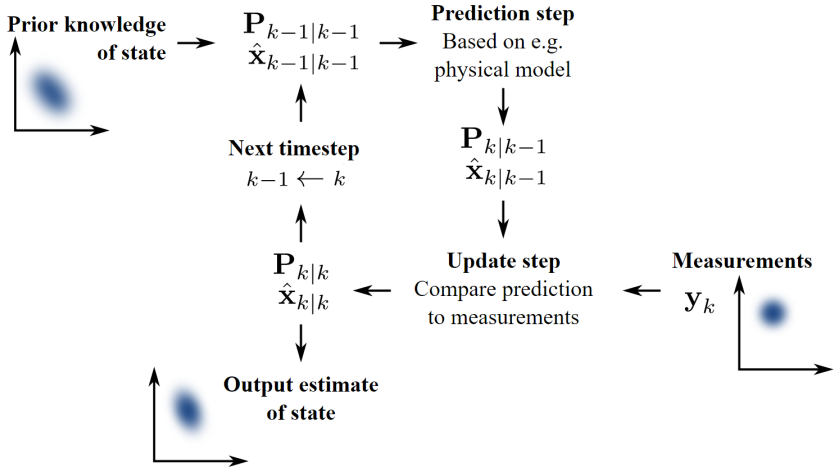


Figure 5.3: Schematic overview of the Kalman Filter. [2]

A schematic overview of the Kalman Filter has been shown in Figure 5.3. The prediction step and update step are, respectively, the time update and measurement update. Furthermore, it can be seen that the filter iterates, which is important as past measurements can, therefore, have an impact on the current prediction. The final part of the algorithm that has not yet been discussed is the initial state. This state can be chosen arbitrarily, however, in that case it could be that the filter will not produce a reliable results. In most cases it will be suitable to start with initializing the model, and start the Kalman Filter later when a reliable state vector has been produced. For the covariance matrix an educated guess should be made as initial state.

5.2.2. Extended Kalman Filter

In the previous section the Kalman Filter was shown for a linear system. Many systems and models are however non-linear. The most widely used non-linear Kalman Filter is the Extended Kalman Filter (EKF) which linearizes the model in such a way that the conventional Kalman Filter can be applied. Although other non-linear Kalman Filters exist as well [68], only the EKF will be explained in this section. First, a non-linear model will be specified in accordance with Equation (5.2); it is represented by the following equation:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k) + G_k \mathbf{w}_k. \tag{5.13}$$

The function $f(\cdot)$ can be non-linear; it is, however, supposed to be a C^1 function [65]. This equation will be linearized using a Taylor approximation with respect to a reference trajectory. For the EKF this reference trajectory is the mean state vector of the most recent prediction[69], so when looking at $\mathbf{x}_k$ this is $\mathbf{x}_{k-1}$. Therefore the first order Taylor approximation becomes

$$f(x_k) = f(x_{k-1}) + J_k(x_k - x_{k-1}) + h.o.t, \tag{5.14}$$

where J_k is the Jacobian with $(J_k)_{i,j} = \frac{\partial f_i}{\partial (x_k)_j}(x_{k-1})$. Substituting this approximation in Equation (5.13), and neglecting the higher order terms, leads to

$$\mathbf{x}_{k+1} = f(x_{k-1}) + J_k(x_k - x_{k-1}) + G_k \mathbf{w}_k. \tag{5.15}$$

Which can be rewritten to Equation (5.4) with $A_k = J_k$ and

$$\boldsymbol{\alpha}_k = -A_k \mathbf{x}_{k-1} + f(x_{k-1}). \tag{5.16}$$

As the model has now been rewritten to a standard linear form it is now possible to apply the conventional Kalman filter.

5.2.3. Ensemble Kalman Filter

The last Kalman filter that will be discussed is the Ensemble Kalman filter (EnKF). This filter can be used for non-linear equations and is the filter that will be used in this research. This method is an updated version of the EKF, but is computational more efficient, and uses less memory, during the computations [22]. The main reason that its more efficient is that it does not calculate and store the covariance matrix. Instead, it calculates an ensemble of state vectors, and with these state vectors it is in the end possible to calculate the covariance when needed, but it is not needed in the filter itself. In this section, first the filter will be shown and afterwards the method for calculation of the covariance matrix.

The EnKF was first introduced by Evensen et al. [22] and has later been updated in various papers [29, 30]. These papers do not show a clear overview of the equations used in the filter and mostly give a thorough explanation of the idea behind them. Kasanicky and Eben [42], however, focus on an overview of the filter with a clear formulation of the equations; these equations have been used in this section. The equations have been updated such that it works with non-linear systems and the notation has been rewritten to the notation used in this thesis.

Let the state vector depend on a non-linear system that reads

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k) + G_k \mathbf{w}_k. \tag{5.17}$$

The final goal of the EnKF is to find the mean state vector with it's covariance matrix. The mean will not be computed directly, instead N ensembles of the state vector will be computed. The filter will thus be applied to each ensemble separately. The time update for this filter, in the case of ensemble j, is given by:

$$\bar{\mathbf{x}}^j_{k|k-1} = f\left(\bar{\mathbf{x}}^j_{k-1|k-1}\right) + G_{k-1}\mathbf{w}_{k-1}, \tag{5.18}$$

with mean

$$\hat{\mathbf{x}}_{k|k-1} = \frac{1}{N}\sum_{j=1}^{N} \bar{\mathbf{x}}^j_{k|k-1}. \tag{5.19}$$

The measurement update will be applied next. The equation is given by

$$\hat{\mathbf{x}}^j_{k|k} = \hat{\mathbf{x}}^j_{k|k-1} + K_k\left(\mathbf{y}^j_k - M_k\hat{\mathbf{x}}^j_{k|k-1} + \mathbf{z}^j_k\right). \tag{5.20}$$

The most important difference with the conventional Kalman filter in these equations is that the noise is included within the calculation of the state vectors for both the time, and measurement update. This is not the case for the conventional filter, as the mean is computed directly, while it will now be computed afterwards. Another important difference can be found in the Kalman gain. The Kalman gain can still be calculated with the same equation:

$$K_k = P_{k|k-1}M_k^T\left[M_k P_{k|k-1} M_k^T + R_k\right]^{-1}. \tag{5.21}$$

This equation, however, involves the covariance matrix, which is given by

$$P_{k|k-1} = \frac{1}{N-1} L_{k|k-1} L^T_{k|k-1}, \tag{5.22}$$

where

$$L_{k|k-1} = \left[\bar{\mathbf{x}}^j_{k|k-1} - \hat{\mathbf{x}}_{k|k-1}, \ldots, \bar{\mathbf{x}}^N_{k|k-1} - \hat{\mathbf{x}}_{k|k-1}\right]. \tag{5.23}$$

Instead of computing and storing the covariance matrices for each time step, in EnKF the vectors L will be stored and used to compute the Kalman gain. This slightly reduces the computational time, but mostly reduces the memory needed. Finally, after the ensembles have all been computed, it is possible to calculate the mean state vector and it's covariance matrix. The mean state vector can be computed by

$$\hat{\mathbf{x}}_{k|k} = \frac{1}{N}\sum_{j=1}^{N} \bar{\mathbf{x}}^j_{k|k}, \tag{5.24}$$

while the covariance matrices can than be found by

$$P_{k|k} = [I - K_k M_k] P_{k|k-1}, \tag{5.25}$$

where $P_{k|k-1}$ can be calculated with Equation (5.22). However, for practical reasons this will, like $P_{k|k-1}$, not be computed unless one truly needs this matrix for its research.

5.3. Localization

In this section localization in data assimilation will be explained. Various researchers have concluded that the Ensemble Kalman Filter is one of the best data assimilation methods currently available [4, 7]. However, when a more thorough look will be taken into the model errors of the filter it can be found that these errors can only be represented in a low dimensional space. This means that using a small ensemble size will lead to a systematic underestimation of variances and an overestimation of large cross covariances. One of the reasons for these errors is the signal-to noise ratio of an observation; this ratio decreases as the distance to the observed location increases, which is expected as when you get further away from a location you expect the observation on that location to have a smaller influence. The problem, however, is that this ratio will even decrease below one, which means that assimilating the observation on model locations past a certain distance will have a negative effect on the model [24]. This effect can especially be found in spatially correlated variables, where a large distance means a small correlation and therefore spurious correlations will have a bigger influence.

Various techniques have been implemented to negate these effect; the most common used is the distance-dependent covariance localization by Hamill and Whitaker [24]. This method uses the physical distance between a model location and observed location to put a weighting to the Kalman gain for each model location. Furthermore, a threshold will be applied such that model locations that are beyond a specified distance will have a zero weight. The problems with this approach are that, first, each model will need a different kind of function to get sufficient performance from the distance based localization, and secondly it is hard to find an optimal distance for the threshold. Finally, this method can only be used with spatially correlated variables, which is not always the case. That is why other methods have been implemented.

One of the most used methods that does not need a distance correlated variable is the hierarchical ensemble filter method by Anderson [3]. This filter uses a theoretical basis to tackle the issues with localization weights. They want to split all ensembles into various groups of ensembles; from these groups, confidence factors can be calculated by minimizing

$$\sqrt{\sum_{j=1}^{m} \sum_{i=1, i\neq j}^{m} \left(\alpha\beta_i - \beta_j\right)^2}, \tag{5.26}$$

over α. Here β_i is the regression coefficient of the i-th group. These confidence factors give a good indication on the performance of the estimates, and can be used to reduce the impact of sampling errors. Furthermore, the method also includes mechanisms to discriminate between spurious and real correlations, which makes it easier to remove the assimilation effects of those spurious correlations. The problem with this approach is, however, that it needs a large ensemble size, which makes it much less computational

efficient than the distance based approach.

New research by Zhang and Oliver [70], however, suggests a new approach in achieving the results of Anderson. They want to use a bootstrap technique in discriminating between spurious and real correlations. Their method randomly draws N_B groups of the original ensemble set, these groups will contain various ensembles multiple times. The confidence factors will be calculated from these groups of ensembles, and as a result, as the groups do not need to be independent of each other, a much smaller ensemble size is needed. This makes this method much more computational efficient when working with a moderate amount of observations. As this is the case for the Java model, with approximately 20 observed locations, this is the localization method that has been selected to run with the Ensemble Kalman filter in this thesis.

5.4. OpenDA

As mentioned, the data assimilation method that will be used is the Ensemble Kalman filter. It has now been shown how the filter will be applied to a model. The software package that will be used for the appliance of the filter to the Delft3D-FM model is the OpenDA package. This package has been specifically created for data assimilation projects and has various data assimilation methods incorporated. Furthermore, scripts are included for OpenDA to be able to work with a diverse set of modelling software packages. Some software packages that has been included are Delft3D, Delft3D-FM, NEMO and Swan. Furthermore, it is also possible to create some model in Java and incorporate that into OpenDA.

As the EnKF is applied by OpenDa in the same manner as explained in Section 5.2.3, the only thing remaining to explain is the way OpenDA handles the noise model in Equation (5.17). This is, therefore, what will be explained in the remainder of this section.

5.4.1. Noise

Data assimilation can be performed on two kind of systems; namely, a deterministic or a non-deterministics system [69]. The first means that when the a priori state is perfect the end result will be as well. However, modelling errors, measurement errors or errors in calculating other input of the model normally exist. This is also the case in the Java model, where the atmoshperic components from ECMWF have been generated by making use of data assimilation, which thus contains errors. It is still possible to perform data assimilation without noise, but without noise the wind will most likely not contain many outliers, which are needed to generate an extreme value analysis. So a non-deterministic system is needed and thus also noise. The noise will be performed on two components of the assimilation. The first is the measurements noise, which is the noise generated because of errors in observations. These will be added by a normal distribution with a small standard deviation. The second noise component is the most important one. This it the background noise, which is generated because of errors in the background information.

In the case of ocean models the most used components to put background noise on are the boundary conditions and the wind field. In order to generate noise a probability distribution should be specified. The most used distribution is the so called Gaussian noise which is a white noise process with zero mean; this is also known as the normal distribution [7]. There are two main reasons why this distribution is often chosen; first, because it can

easily be adopted for a multivariate probability distribution and is, therefore, also useful when multiple variables are chosen to put noise on. Furthermore, it is a distribution that gives higher probability to smaller disturbances than to large disturbances. This is very useful for noise, as it is expected that an error is small in most cases, however, sometimes this error could be large; i.e. when extreme weather conditions occur. This is thus also needed for an extreme value analysis as extremes should occur sometimes.

Depending on the chosen component, however, some additional requirements can be given to the noise. Most common of these requirements are that the noise should be correlated in time and space. This means that two adjacent values, in either time or space, should be dependent on each other. This is required as large differences cannot occur over small differences in time or space. This correlation are thus also implemented in OpenDA and will be explained in the next two sections.

Spatial correlation

The spatial correlation in OpenDA will be achieved by making use of spatial covariance in space [27, 67]. The elements of the covariance matrix that will be used are a special case of the Matérn isotropic covariance function [52], which is given by:

$$F(d) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} d}{r} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu} d}{r} \right), \tag{5.27}$$

where ν is the smoothness parameter, d is the distance between two sample locations, Γ is the Gamma function, r is the distance parameter, K_{ν} is the modified Bessel function of second kind, and σ is the standard deviation of the noise model. In the case of OpenDA the spatial process is expected to be smooth and it is, therefore, possible to take the limit $\nu \to \infty$. At this limit the Matérn function converges to the squared exponential covariance function [54], also known as the Gaussian model, which is given by:

$$Cov_{ij} = \sigma^2 e^{-\frac{d^2}{2r^2}}. \tag{5.28}$$

In order to calculate these values two parameters are needed. The first one is the distance parameter, r, or, as it is called in OpenDA, the spatial correlation scale. This parameter is different per model and should be chosen in correspondence with the scale of the model. The second parameter is the distance between two sample locations, d. In the case of curvilinear ocean models it is needed to calculate the distance between two locations on a sphere. This will be done by making use of the great circle chord, which will be calculated by:

$$C = \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2}, \tag{5.29}$$

where Δx, Δy and Δz are given by the Cartesian subtractions between two points $\phi_1, \psi_1)$ and $\phi_2, \psi_2)$. These subtractions are calculated with the following equations:

$$\begin{aligned} \Delta x &= \cos(\phi_2)\cos(\psi_2) - \cos(\phi_1)\cos(\psi_1), \\ \Delta y &= \cos(\phi_2)\sin(\psi_2) - \cos(\phi_1)\sin(\psi_1), \\ \Delta z &= \sin(\phi_2) - \sin(\phi_1). \end{aligned} \tag{5.30}$$

The distance d, which is the great circle distance on Earth, can than be calculated, assuming that the Earth is a sphere, by

$$d = 2r_a \arcsin\left(\frac{C}{2}\right). \tag{5.31}$$

As there exist correlation in both x, and y direction this covariance matrix is calculated twice. Once with x-values and the mean value of y, and once the other way around. This results in two covariance matrices. The noise model with mean zero and both correlations incorporated can than be computed as follows:

$$N^s = Cov_x^{\frac{1}{2}} N(0,1) \left(Cov_y^{\frac{1}{2}}\right)^T, \tag{5.32}$$

where $N(0,1)$ is the Gaussian distribution with mean zero and variance one, the covariance matrices are for x and y direction, respectively, and the square root will be performed by taking a Cholesky decomposition of the matrix [54]. However, it is also needed to have correlation in time, which will be explained in the next section.

Time correlation

Time correlation can be achieved by making use of various processes. where the noise matrix at some time step depends on the noise matrix of the previous step. For this thesis the AR(1) process will be used, which is a simple linear process:

$$N_k = N^s + \alpha \cdot N_{k-1}, \tag{5.33}$$

where $\alpha = e^{\frac{t_k - t_{k-1}}{t_s}}$ is dependent on the time correlation scale and the time step, which will be constant in this thesis, chosen. Next to the correlation process, the time correlation scale will also adapt the standard deviation chosen for the noise. The adapted standard deviation will be calculated by

$$\sigma_a = \sigma\sqrt{1-\alpha^2}, \tag{5.34}$$

and will also be used as standard deviation in Equation (5.28).

6

Methodology

All theory has now been explained, and the software packages, Delft3D-FM and OpenDA, that will be used have been introduced. So in this chapter the focus will be on the experiments; first the two classes of experiments used will be explained. Than the original experimental strategy will be shown, which will conclude, due to various reasons, into a final experimental strategy.

6.1. Experimental methods

The experiments that have been performed, and will be shown in the following sections, all make use of just two different experimental methods. The first method uses artificially generated observations in a, so called, twin-experiment, while the second method makes use of real observations. These two methods are used, as they serve different purposes; a twin-experiment is used to test the data assimilation method, while the second method tests if the experiment works with real observations.

6.1.1. Twin-experiment

In a twin-experiment artificially generated observations are used; the set of locations can be found in Table B.1. These observations are generated by performing one model run, which will be seen as the 'true state' of the system, and will, therefore, not contain any unexplained values, as can happen with real data. The mean of the various ensembles, that are generated in the assimilation process, will, however, almost be the same as the model, because the noise used in the process has zero mean. This means that the assimilation does not have to analyse anything in generating the same results as the 'true state', which means that the process can not be tested. That is why in the model run that generates the 'true state' one of the components will also be disrupted by making use of noise with mean zero. Now it is known that the 'true state' can, under certain circumstances, be reproduced by the model, but is not directly produced by the mean of various ensembles.

In this experiment one does, therefore, not need to worry if the model can predict the observations correctly, and neither where these are located, as it is known that the model can adequately represent the observations. These experiments are thus useful for testing the performance of the data assimilation method and software itself. This is, however, not the only reason a twin-experiment is useful. In this thesis, as is often the case in data assimilation studies [1], an important question is if data assimilation will improve the

model values on all unobserved locations. This evidently cannot be achieved when using real observations, as these are always on a much smaller amount of points than a model can generate. This aspect, knowing the 'true' state on unobserved locations, can also be used to validate the performance of localization. Which will, therefore, also be tested.

6.1.2. Real experiments

The other method for experimenting with data assimilation is the assimilation of real observations. These experiments are evidently the most important ones, as they will show if assimilation will improve the model significantly when using real life data. This is not necessarily the case when the twin-experiment works, as has been seen in the IJmuiden model [26]. The observations in this thesis have been generated from the UHSLC database. In Chapter 5 it has been shown that the dataset of the observations around the Java Sea contains empty spot throughout the full period. One of the best periods of data is 2013, which is, therefore, the period that will be used in the experiments with real life data; the final set of locations used can be found in Table B.2. However, the data is not the only part that can lead to a failing assimilation; it could also be that the model doesn't resemble the observations correctly. In Section 3.3 and Chapter 4 it has, however, been shown that the time-series for the model and the observations looks fairly similar. In this research, it is expected that the model does resemble the observations correctly.

There are two ways to perform these experiments: all locations available can be used or some locations are not used for validation. In the first experiment the only locations that can be tested are, however, those with observations. This means that it is not possible to test the assimilation with real life data for unobserved locations. This is still of high interest as those locations are the most important; at observed locations one just uses observations and doesn't need data assimilation. This means that for the performance of the model on unobserved locations, the results of the twin-experiment have to be used, but only when the performance on observed locations is similar for both experiments.

In the second experiments both observed and unobserved locations have data available and can thus be tested. The problem with these experiments is, however, that the amount of observed locations is reduced. In Section 2.2 it has been shown that the amount of observations available is of high importance for the performance of data assimilation and removing locations from the assimilation can thus lead to a reduction in performance. This experiment will, therefore, only be performed when the twin-experiment performs well, as a reduction in observations might not be a problem in that case.

6.2. Experimental strategy

The original research question, as written in the introduction, was:

> **Research question:** Does data assimilation significantly improve extreme value analysis of the water levels in the Java Sea?

In order to perform such an extreme value analysis a large enough time series is needed. The preference would be to have a data set of multiple years, however, this was thought not to be feasible during the thesis. Therefore, it was decided to try and get a time series of one year to perform an extreme value analysis. Quite early in the process it was found that

it might not be possible to apply the extreme value analysis to the time series during this research. The focus of the thesis was thus refocussed on the data assimilation itself, with as goal to create a time series of a year with a better performance than the Java model. In order to achieve this, two experiments were planned. The first was a twin-experiment with artificial data to test if the assimilation worked. Than, if the outcome would be positive, an experiment with real observational data would be performed to see if the real life data is of sufficient quality for a data assimilation experiment on the Java Sea.

6.2.1. Experimental testing

At first various experiments were run with a very small, with an area of approximately $25 km^2$, test model. This test model has been used throughout the remainder of this thesis when new settings were to be tested, because at the first try of assimilating with OpenDA on the Java model it was found that the computational time was quite high. In Delft-3D FM, the Java model took approximately three hours to perform a simulation for a year, with the computations being more than 95% of the time used. When assimilating, however, the initialization was taking most of the Delft3D-FM time, as this initialization is needed for each time step of one hour. Only this part of the assimilation, already, made the assimilation to computationally expensive for a normal desktop. Therefore, it was decided, as the focus was already on the performance of assimilation on the Java model, to perform various runs of smaller periods to research the various aspect of the assimilation method. The plan was to have at least one run for a monthly period with assimilation per hour and a run of a half year period with assimilation every four hours; both with approximately 20 ensembles.

These two runs were, however, still too computational expensive for a normal desktop, as only 3 ensembles could be run at the same time. These ensembles were performed parallel to each other by using the threading option of OpenDA, while more ensembles would be performed sequential to these. As these runs still needed a quite large amount of time to run, it was decided that a more powerful computer system was needed. The first option was the cluster of Witteveen+Bos, however, problems with obtaining the correct Delft3D-FM linux license created a large delay. Furthermore, when the license was obtained, runs could not be started due to the cluster already being used almost full time for projects. So it became clear that the cluster could not be used for the experiments in this thesis. Various other options were, therefore, investigated and, finally, the 'Access to the National Computational Facilities' [56] was found. This is a funding scheme for researchers to gain access to Dutch supercomputer SURFsara [64]. It was decided to apply for a pilot project on the Cartesius supercomputer, which was granted quite fast.

6.2.2. Cartesius

The first thing that had to be done was to get everything up an running on Cartesius. At first only Delft3D and not Delft3D-FM was running on Cartesius, however, a Linux executable of Delft3D-FM was obtained which could be used for this project only. Furthermore, some problems with the Java version had to be countered, but eventually everything worked with the test model, so testing with the Java model was started. These runs were performed on one node with 24 cores, and thus also 24 ensembles. However, the runs took longer than expected; each analysis step took two hours, which means that the two runs previously mentioned would take approximately two months. This was

especially interesting as one time step on the desktop too about half an hour; on desktop, however, only 3 ensembles could be run parallel, while on Cartesius 24, so Cartesius was still an improvement.par After some research and various runs of only a few analysis steps it was found that the bottleneck was, most likely, the input and output of files. This was the case because at each run the time of a computational step, consisting of both the noise computation as well as the model computation, was only for approximately one percent CPU time; the rest is either disk reading or idle time. At this stage two decisions were made: first of all, as the end of this thesis came near, it was decided that various short runs of a week model time would be sufficient to decide if data assimilation improves the model. The other decision was to try and get OpenDA running with Remote Method Invocation (RMI) by Java; RMI lets you run the various threads of an application on multiple systems by opening connections with virtual machines on these systems [35]. This means that it is possible to run an assimilation with 25 ensembles on, for instance, five nodes such that the input and output of files is distributed along these nodes, which should, in theory, reduce the computational time.

While working on the RMI scripts for running OpenDA, various runs of one week were started already. These runs used 5,15 and 25 ensembles, respectively, and showed a few other problems to deal with. First of all, the 15 and 25 ensemble runs were stopped after five days, because Cartesius has a five day limit on using its nodes. These run could not be restarted, as they were stopped in the middle of an analysis step. This can, however, easily be dealt with by cutting a run into shorter pieces of, for instance, two model days such that a run finishes before the five day time limit. The 5 ensemble run finished on its own within the five days, which is a positive sign for the RMI to be able to work properly. An analysis step in this run took approximately half an hour, being at the same level as a desktop run. However, this run also showed two problems. The first problem is that the run was stopped after six model days, showing an error that 'no computed value could be found for the observed locations for 20-01-2015'. This is a weird error, as the computed values could be found for all times before, and also the Delft3D-FM runs seem to have finished as they should. After looking at the files that OpenDA uses for finding the observed values, it was found that for some reason those files showed values from the day before. What the precise reason for this error is, is still unknown, and in the final strategy a five ensemble run will therefore be included to see if it happens again.

6.2.3. Final strategy

The last problem found can be seen in the Figure 6.1; for most observed locations it has been found that the assimilation does not work very well. At Lembar (Figure 6.1c) and Benoa the assimilated values do follow a trend that is the same as the observations, although with smaller amplitude. At all other observed locations, however, the results are like Figures 6.1a and 6.1b, and do not show any resemblance. The reason for this seems to be the wind noise, as the wind noise applied in OpenDA shows similar values for all times before 15 january, so in the initializing phase, but after the first analysis the noise becomes of a order of magnitude higher. That is why for the final strategy it has been decided to also include experiments were the noise has been generated with Matlab before the runs; this means that the analysis does not change the noise, but only the water levels that will be fed back into the model.

As the RMI was working well too, this lead to the following final experimental strategy:

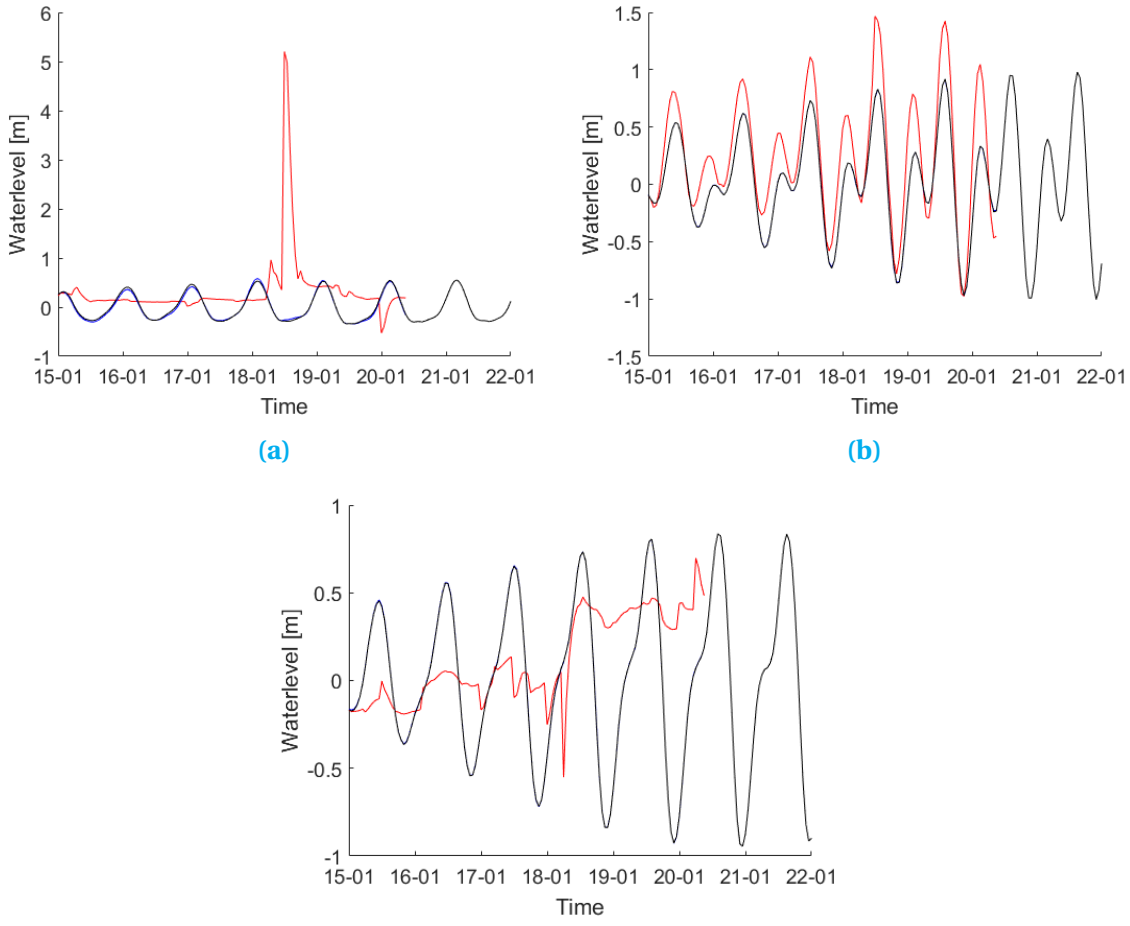


Figure 6.1: The water levels propogating through time at a) Kolinlamil, b) Lembar, and c) Miri for the five ensemble run ending at 20-01-2015 for unknown reasons. In each figure three timeseries are shown: the ensemble mean (red), the observations (blue) and the Java model without noise (black).

seven runs will be performed. These runs consist of two runs with five ensembles, and five runs with 25 ensembles. In the case of the runs with 5 ensembles one run will be performed with pre-generated noise, while the second will use noise generated in OpenDA. For the 25 ensemble case 3 runs will be a twin-experiment and 2 runs will use real observations. For the latter, again, one will be performed with pre-generated noise and one with noise generated in OpenDA. While on two of the three twin-experiments localization will be performed, one with and one without pre-generated noise, and the last experiment will be without localization and with pre-generated noise. The results of these experiments will be shown in the next chapter.

7

Results and Discussion

In this chapter the results of the various experiments will be shown and discussed. First the twin-experiments will be shown and next the experiments with real observations.

7.1. Twin-experiments

The five twin-experiments will, as mentioned, be discussed in this section. In the next section, the results of the experiments with five ensembles will be discussed, with first the experiment with localization and than the one without. Next, in Section 7.1.2, the results of the experiments with 25 ensembles will be shown; here we will first take a look at the two experiments with localization applied, and finally the experiment without localization.

7.1.1. 5 Ensembles

First, we will take a look at the two five-ensemble experiments; these have been performed making use of noise, added to the wind, that has been generated with Matlab. This generation has been performed by the scripts in Appendices C.1 and C.2, which make use of the same method as explained in Section 5.4.1. It is, therefore, interesting to find out if these runs do not have the same problems as the experiment shown in the previous chapter. In Figure 7.1 the time series for three observed locations, again showing Kolinlamil, Lembar, and Miri, can be found for the experiment with localization applied. The experiment was again run for a model period of seven days; from the 15th of January 2015 upto the 22nd of January. Unfortunately, the figures show that the experiment has again failed to complete; this time, however, after the 21st of January at 7h. This does not look to be an incident and should, therefore, be researched more thoroughly. In this thesis we will see if it happens in the other experiments as well, more research should be done in the future.

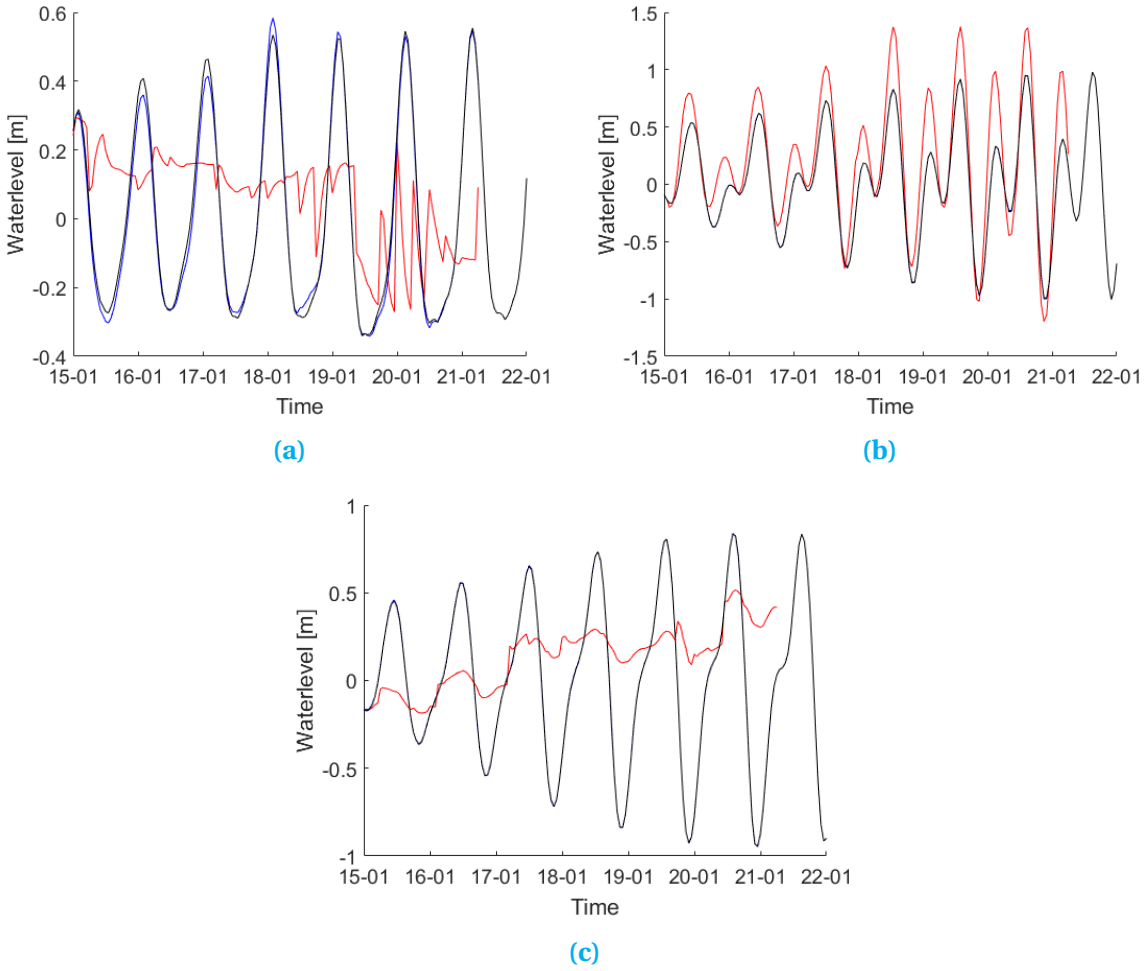


Figure 7.1: The water levels propogating through time at a) Kolinlamil, b) Lembar, and c) Miri for the five ensemble run with noise generated by Matlab, and localization applied. In each figure three timeseries are shown: the ensemble mean (red), the observations (blue) and the Java model without noise (black).

The other problem encountered in the test run was, that the analysis failed to perform well. This was the reason for choosing noise generated by Matlab in this experiment. Unfortunately, as can be seen from Figure 7.1, the results have not been improved by much. It can still be seen that only the values at Lembar give a close resemblance to the observations. In the other two cases the results are completely off, although the extreme value that was encountered at Kolinlamil can not be found in this time-series. Furthermore, the results at the locations that have not been plotted were similar to Kolinlamil and Miri, except for Benoa which is similar to Lembar. This was not what was expected, as the noise is now generated by Matlab in advance and not by OpenDA, and the experiment was thus performed without stochastic forcing. Unfortunately, it was found, after comparing the wind noise files, from before, and after the experiment, with each other, that OpenDa still updates the noise. This explains that the values again fail after the first analysis step. Next, we will take a look at the five-ensemble experiment without localizations.

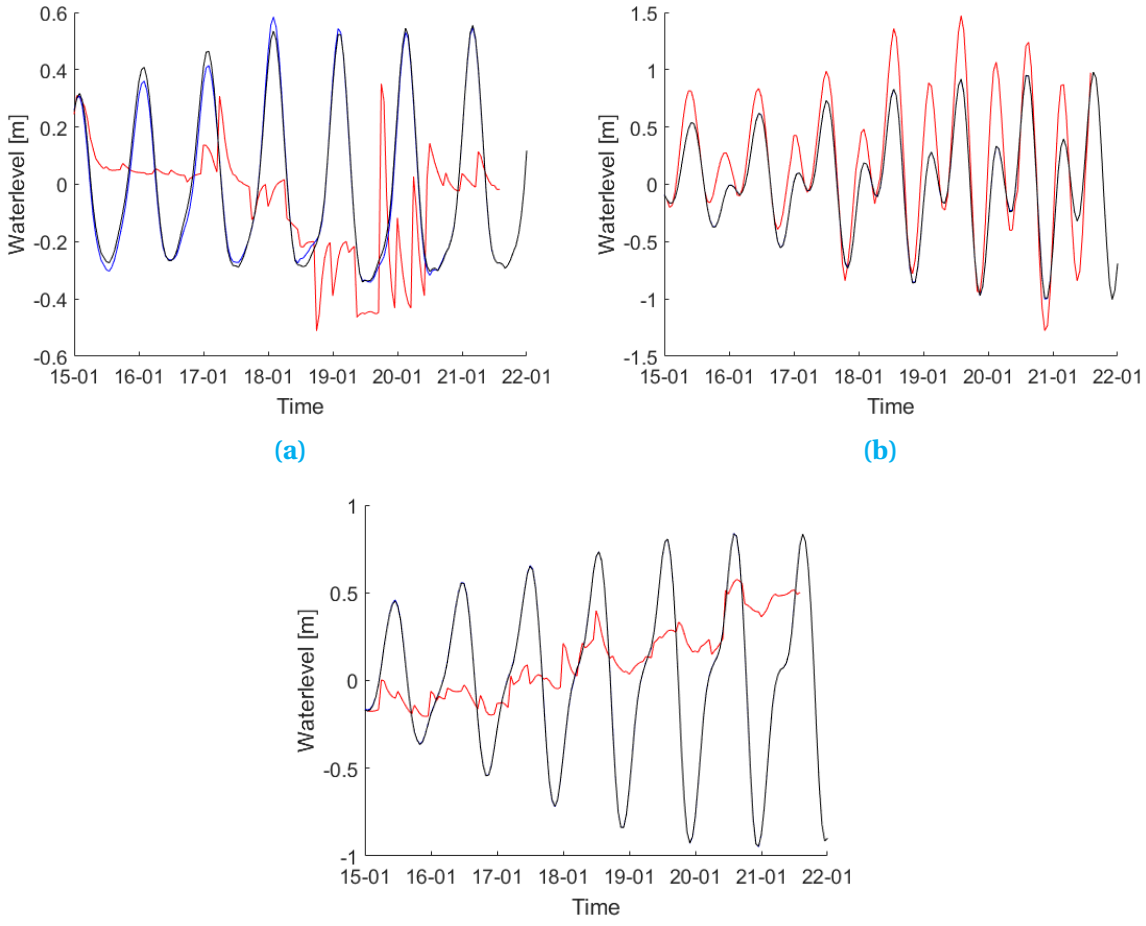


Figure 7.2: The water levels propogating through time at a) Kolinlamil, b) Lembar, and c) Miri for the five ensemble run with noise generated by Matlab, and no localization applied.In each figure three timeseries are shown: the ensemble mean (red), the observations (blue) and the Java model without noise (black).

In these two runs the same pre-generated noise files have been used such that the only difference between the two runs is the use of localization. In Figure 7.2 the same time series have been shown as in Figure 7.1. The experiment without localization is again stopped on a different analysis step than the previous two runs; this time at the 21st of January at 15h. Furthermore, there are some small differences in the results as well; the fluctuations are, mostly, a bit larger for the second run than for the first. However, in order to find out the influence of localization on the assimilation we also have to compare the unobserved locations.

In Figure 7.3 the Root Mean Square Errors (RMSE) between the assimilated water levels and the artificial observations have been shown for the whole Java model grid. It can be seen that the spots with large RMSE are quite different from each other. The results of the experiment with localization (Figure 7.3b) contain larger RMSE than the results of the experiment without localization. This is not what one would expect, as the idea of localization is that it improves the assimilation and, therefore, the results on unobserved

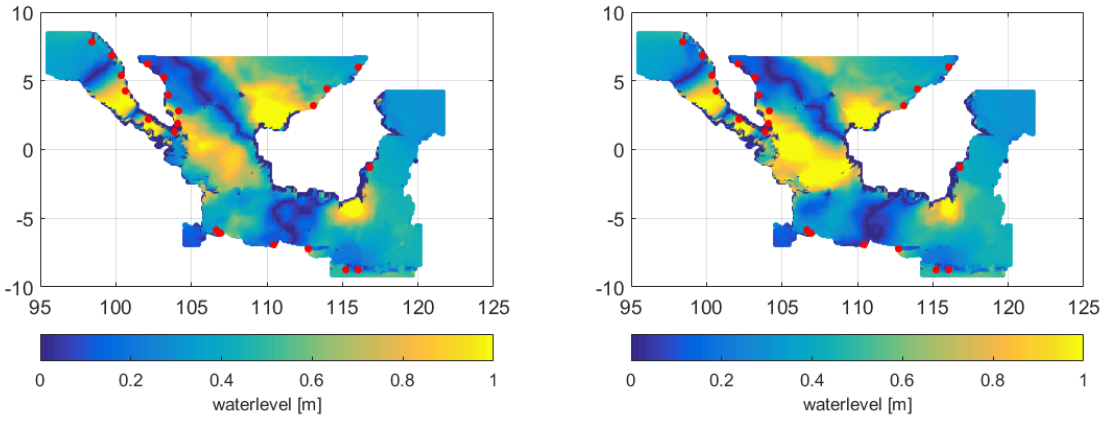


Figure 7.3: The Root Mean Square Errors between the assimilated water levels and artificially generated observations for, a) the five-ensemble experiment without localization, and, b) the five-ensemble experiment with localization. The red dots are the observed locations and the size of the errors can be read from the colorbar.

location. However, in the case of these two experiments there is a quite reasonable explanation. Figures 7.1 and 7.2 have shown us that the modelled values on observed locations are for most locations not very close to the observed values. This means that with localization bad values are mapped to the unobserved locations which could results in worsening the results when using localization; this seems to be happening in this case. In order to fully know if this is indeed what happens, we also have to compare the results of the 25 ensemble experiments, as an higher amount of ensembles should improve localization and assimilation.

7.1.2. 25 Ensembles

Now we will study the results of the 25-ensemble experiments; first, we'll compare the experiment with noise generated by OpenDA with the experiment with pre-generated noise; both with localization. We again plotted the time series of three observed locations, the results for the experiments with and without pre-generated noise can, respectively, be found in Figures 7.4 and 7.5. The first thing that stands out, is that these runs have not been stopped at a certain analysis step. Furthermore, there are less fluctuations in both experiments than in the experiments with five ensembles, which is as expected, as more ensembles should give a smoother results. Comparing these two experiments with each other it seems that the experiments with noise generated by Matlab is somewhat better, which especially can be seen in the time series of Miri, as the last analysis steps seems to get the modelled value into the range of the observed values. It, however, still does not show the trend we want. When looking at the RMSE over time for each observed location the differences seem not to be significant; showing even that at most locations the experiment with noise generated within OpenDA performs slightly better.

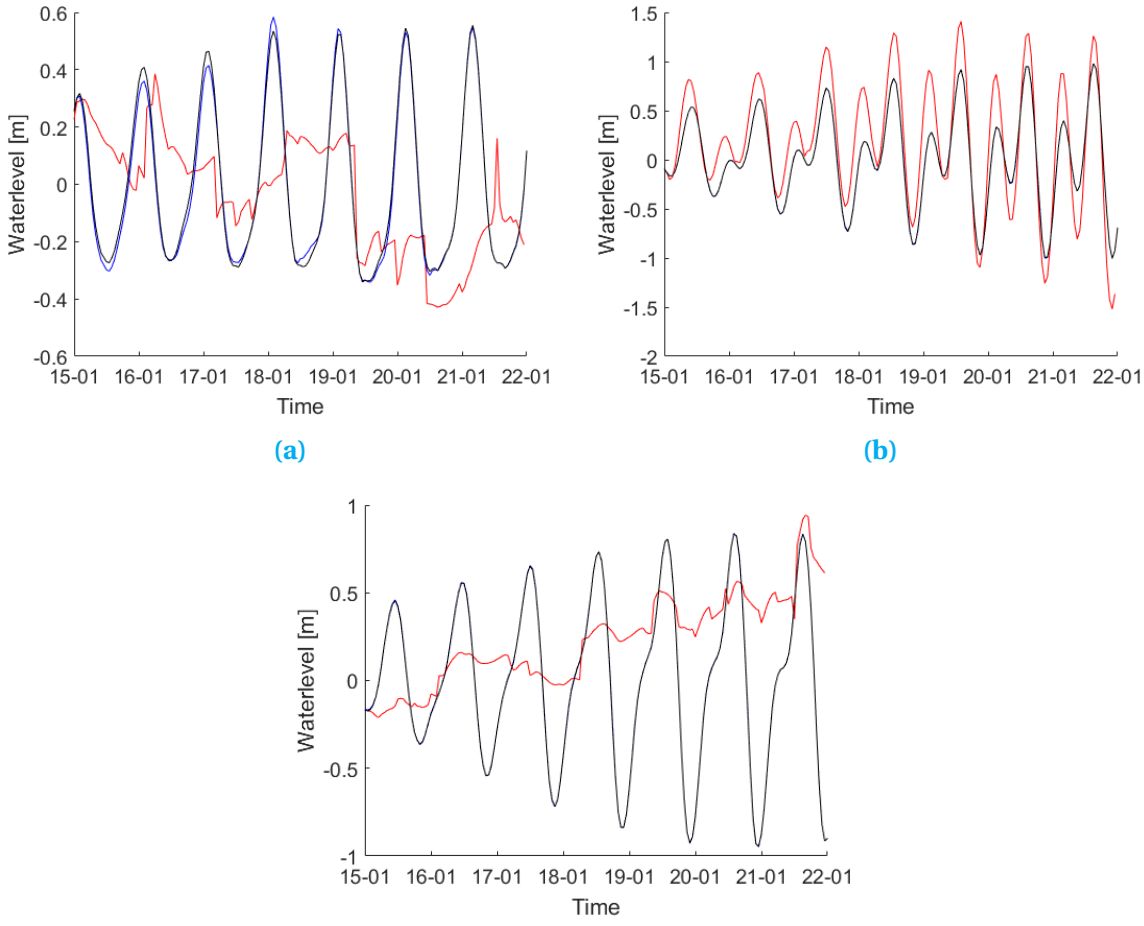


Figure 7.4: The water levels propogating through time at a) Kolinlamil, b) Lembar, and c) Miri for the 25 ensemble run with noise generated by Matlab, and localization applied. In each figure three timeseries are shown: the ensemble mean (red), the observations (blue) and the Java model without noise (black).

The final twin-experiment performed was the experiment with pre-generated noise and without localization. The time-series generated by this experiments are shown in Figure 7.6, and it is immediately clear that these assimilated time-series are of lower quality than the ones from the previous two experiments. This difference is especially interesting as the difference between the five-ensemble runs were not as large as with these time-series; although even with those time series we saw that the fluctuations became larger with the experiment without localization than with localization. Furthermore, it is quite easy to explain why the difference is larger; localization needs quite some ensembles to truly perform well, and thus it is expected that it works better when using more ensembles. The question now is, if this also means that the difference in RMSE around the Java model is larger between these experiments than with the five-ensembles experiments. Furthermore, we will see if the same trend is shown that the localization increases the impact of low quality observations, which leads to higher errors.

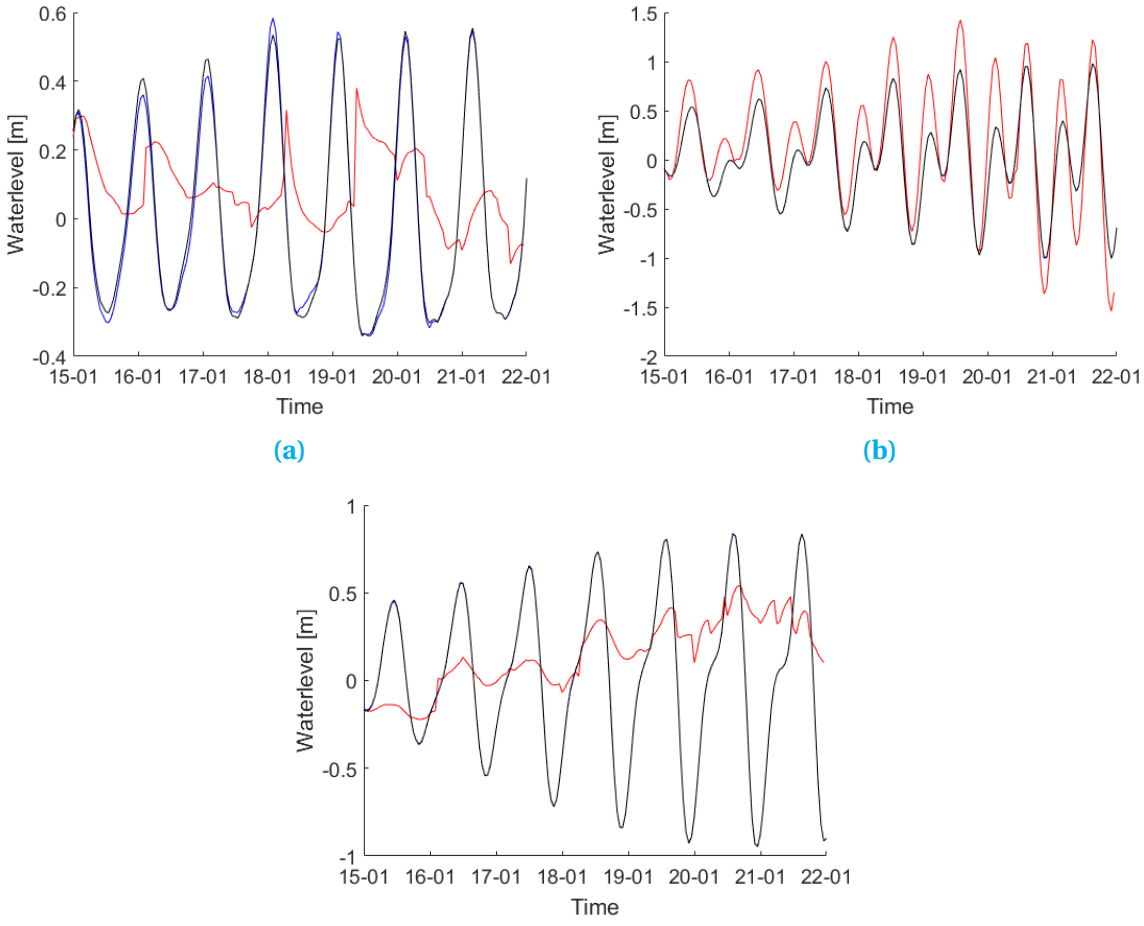


Figure 7.5: The water levels propogating through time at a) Kolinlamil, b) Lembar, and c) Miri for the 25 ensemble run with noise generated in OpenDA, and localization applied. In each figure three timeseries are shown: the ensemble mean (red), the observations (blue) and the Java model without noise (black).

The RMSE of the three 25-ensemble experiments can, together with the RMSE for the Java model without noise, for the full Java model grid at 18-01-17 0h, be found in Figure 7.7. It is immediately clear that the Java model shows a much better results than the assimilated model. Furthermore, from Figures 7.4 to 7.6 we see that the same holds for the observed locations with a much lower difference between the model and observations. This means that the assimilation has not worked as it should. We will, however, analyse the results of the other three experiments in order to find which experiment shows the most promising results, and to research if the reason for the failure of the assimilation can be found.

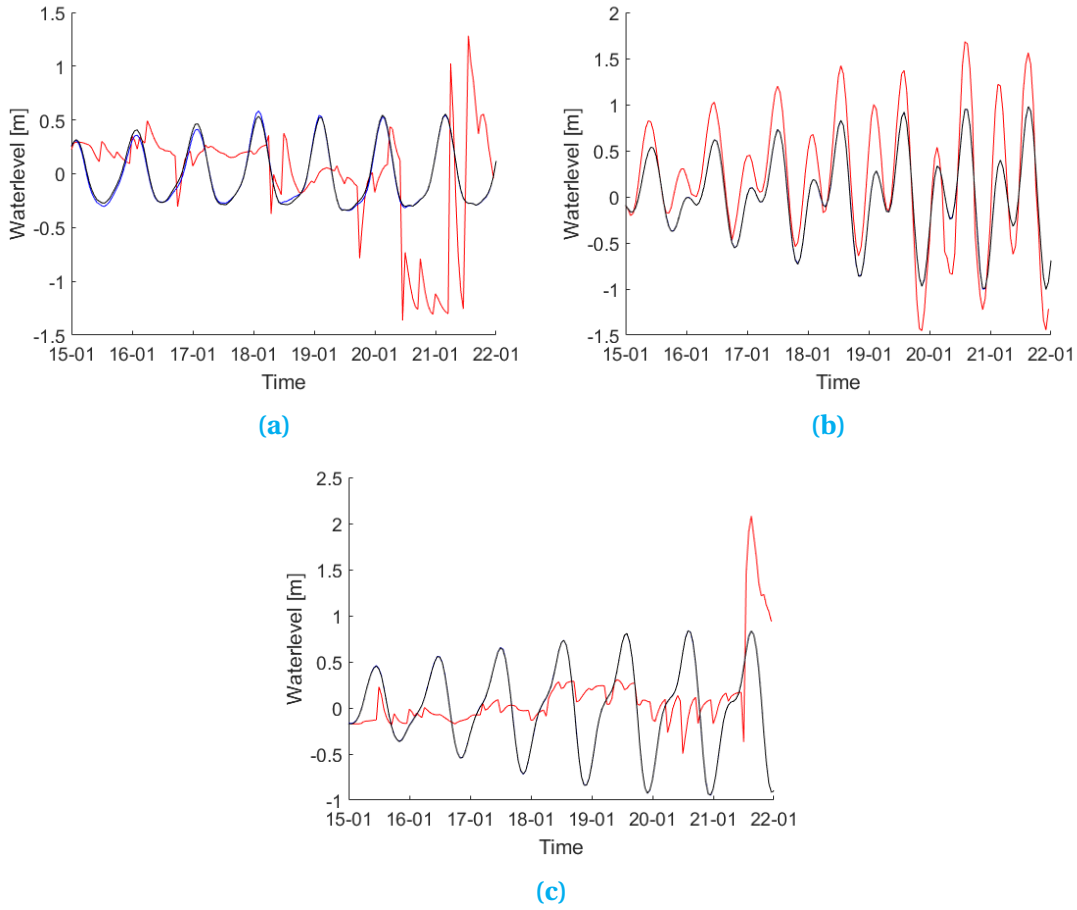


Figure 7.6: The water levels propogating through time at a) Kolinlamil, b) Lembar, and c) Miri for the 25 ensemble run with noise generated by Matlab, and no localization applied. In each figure three timeseries are shown: the ensemble mean (red), the observations (blue) and the Java model without noise (black).

Looking at the the full grids for the three experiments we see four spots of interest, because they have a much larger RMSE than the rest of the region. These spots are:$(100;4)°, (105;0)°, (115;-5)°$ and $(110;3)°$. Looking at these spots we find that the experiment with noise generated by Matlab and with localization applied gave the worst results. In the other two cases it is, however, not as clear which experiment performs better. The reason for this is that both experiments show the better results for two of the locations. The experiment with noise generated by OpenDA shows the most promising results for $(105;0)°$ and $(115;-5)°$. These are both locations that do not have an observed location close to them, or at least not without an island in between, which can be seen from Figure 3.7. The locations $(100;4)°$ and $(110;3)°$, however, both have a location quite close, and have the experiment without localization performing best. Furthermore, the only two locations, Benoa and Lembar, which perform well under assimilation are located around $(115,-9)°$. This is quite close to the boundary, and are thus largely influenced by the boundary conditions. Furthermore, these influences are not disrupted by the wind noise, as the distance is small, and thus the change to the observations is small.

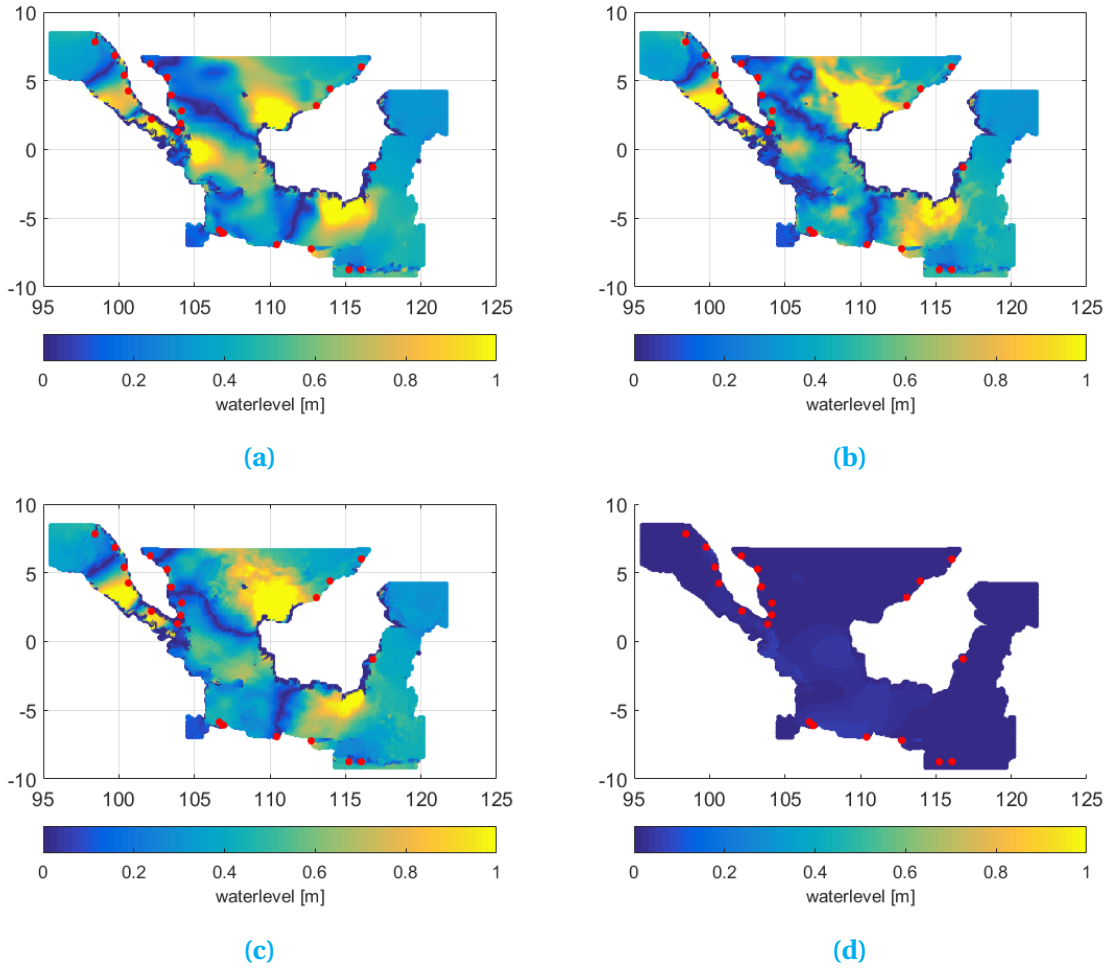


Figure 7.7: The Root Mean Square Errors between the assimilated water levels and artificially generated observations at 18-1-2015 0h for, a) the 25-ensemble experiment without localization, b) the 25-ensemble experiment with localization, c) the 25-ensemble experiment with noise generated in OpenDA, and, d) the Java model without noise. The red dots are the observed locations and the size of the errors can be read from the colorbar.

In Figures A.1 to A.3 the same four plots can be found, however, this time for three different times. These are shown because the above is the case for one time step, which does not mean it also the case for other time steps. The times that have been printed are: 18-01-2015 6h,18-01-2015 17h and 16-01-2015 6h. As expected, because of the large difference at the 18-01-2015 0h results, the model without noise produces the best results for each of these times. However, in the case of 18-01-2015 6h and 16-01-2015 6h the experiments perform much better than before, although still worse than the model. In order to explain this we will take a look at Figures 7.4 to 7.6, where it can be seen that for the observed location these time steps show a much better conformation between results and observations; although this can only be said for the values, not for the trend. This probably the reason why the results for the whole grid shows a better conformation. Furthermore, we can see that for those two times the results for the four interesting spots of before are, either quite similar between the three experiments, or the experiments with

localization show the best results. This again, as in the previous section, suggests that the localization works correctly, as it seems to increase the influence from observed locations. In the case of the results for 18-01-2015 17h we, however, see the results being worse than before. Furthermore, each plot has a spot where it shows the best results. Finally in Figure A.4 the results for a run with noise generation within OpenDA and without localization applied can be found for the same four time steps. Comparing with the rest we can see that it does not differ much from the previous results, and does not lead to new conclusions.

We have seen that the results of the twin-experiments are not very promising. Especially, as it was expected that the results would be quite good, because it has been shown that the assimilation with Delft3D and OpenDA for a similar region worked fine Section 2.2. There are three main differences between these two studies: the assimilation method, the hydrodynamic software and the grid structure. The assimilation method used in this thesis has, however, also been used in [66] where it was shown to perform quite well. So we arrive at the next difference, the hydrodynamic software, as in this study the successor of the software used by Karri et al. is used it could be that something is wrong in the coupling between OpenDA and Delft3D-FM. However, various test models exist within the OpenDa software package for Delft3D-FM and these seem to perform well with EnKF. These models, however, are on a much smaller region, so it could be that the problem lays within the size of the model; although this seems unlikely.

The most probable cause might be the difference in structure. In the Java model an unstructured grid is used, while in [39–41],they used a structured grid. As has been shown in Section 3.2.3, an unstructured grid becomes more complex to work with. It could, therefore, be that there is a problem in handling the unstructured grid in OpenDA, which results in the data assimilation method failing. However, with the results of this research none of these problems can be proven to be the cause of the failing assimilation. Therefore, future research that tackle these problems should be performed to find the definite answer.

7.2. Real experiments

Although the twin-experiments show that the assimilation reduces the performance of the Java model this chapter will still show the results for the experiments with real observations. It is not expected that this will suddenly show very promising results, but might give an insight if the real observations give comparable results to the twin-experiment, which would show that the real observations from UHSLC can be used in data assimilation. Furthermore, the observations can be compared to the model to find out if the model can resemble the observations quite well.

After running a first test with real observations various observed locations had to be removed as no water levels could be found for those locations. This left 15 locations to work with, as this already a quite small amount, and because the twin-experiment didn't perform well, it was decided that no locations would be left from the model for validation. The experiments performed will just be used to study if the results for observed locations of these experiments are comparable to those of the twin-experiments.

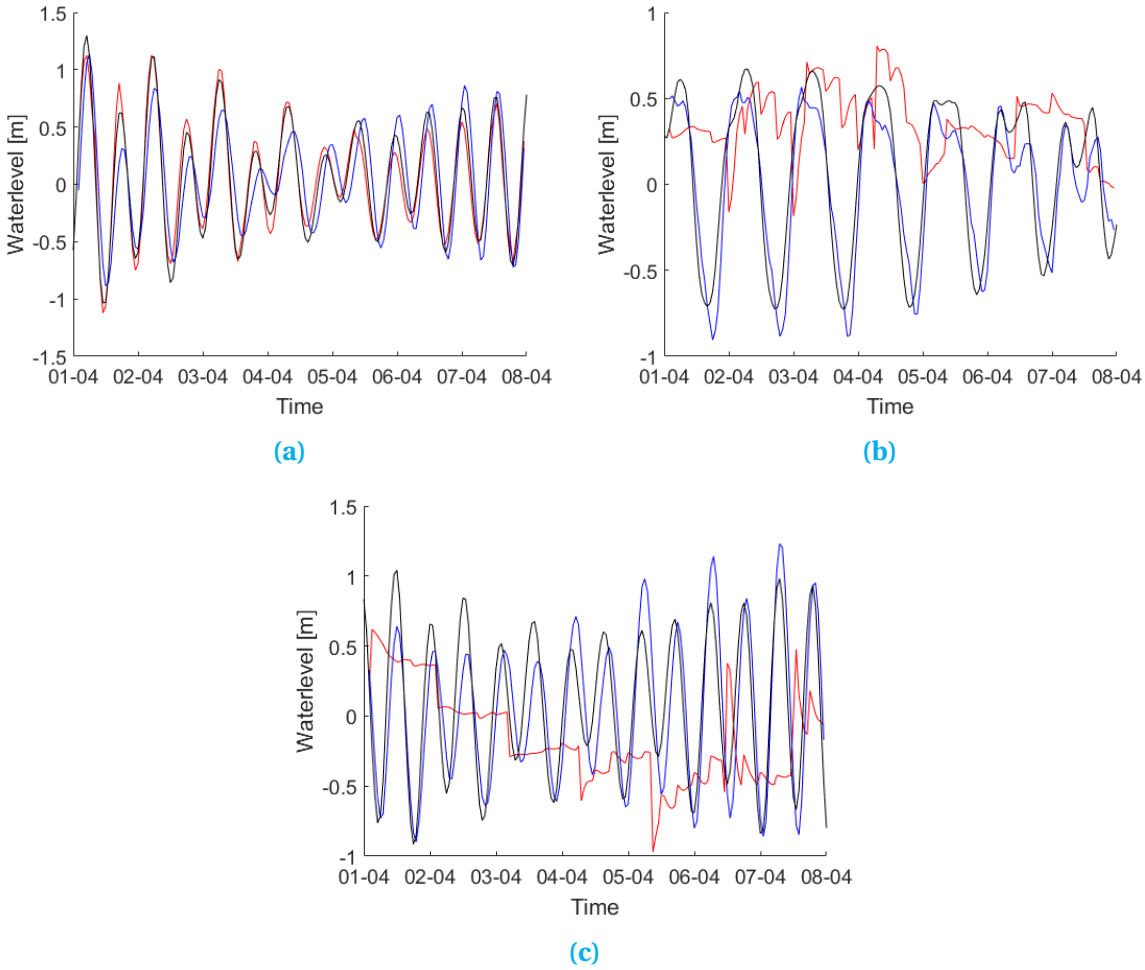


Figure 7.8: The water levels propogating through time at a) Benoa, b) Bintulu and c) Lumut for the 25 ensemble run with noise generated by Matlab, and localization applied. In each figure three timeseries are shown: the ensemble mean (red), the observations from UHSLC (blue) and the Java model without noise (black).

In Figure 7.8 the time series for three observed locations are shown in the case of the experiment with noise generated in Matlab and with localization applied. The time series for Benoa, Figure 7.8a, show quite good results; this is, however, the only one for the whole model. In the other cases we see that the model without noise shows the most promising results by following the correct trend. This is promising as it shows that it is possible for the Java model to approximate the real observations, but the data assimilation, like in the twin-experiment, worsens the results. The same is happening in the experiment with noise generated within OpenDA, as can be seen in Figure 7.9. So the real observations seems to perform as well as the twin-experiment performs, which is promising for future assimilation project. We have, however, seen that at this moment the assimilation is not working well; this should be researched further in order to find exactly what the problem is.

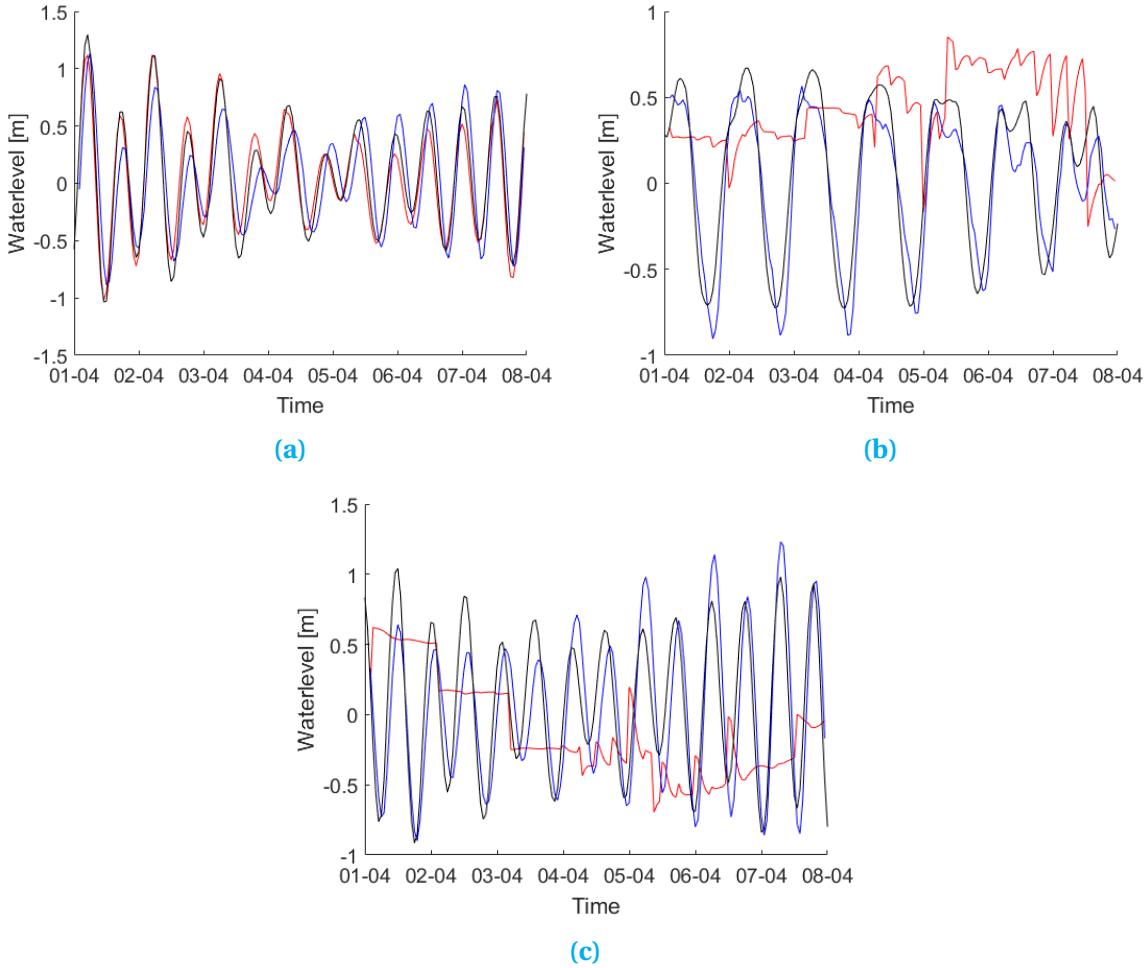


Figure 7.9: The water levels propogating through time at a) Benoa, b) Bintulu, and c) Lumut for the 25 ensemble run with noise generated by Matlab, and no localization applied. In each figure three timeseries are shown: the ensemble mean (red), the observations from UHSLC (blue) and the Java model without noise (black).

8

Conclusions and Recommendations

The goal of this research was to study the effects of data assimilation on an extreme value analysis of the water levels in the Java Sea. Unfortunately, this goals has not been reached as it became clear quite early that the extreme value analysis would not be performed. Therefore, it had been decided to focus on answering the four sub-questions as written down in the Introduction. These questions specified four sub-goals and in this chapter it will be shown if these goals have been achieved.

The first sub-goal was to study if the coupling between OpenDA and Delft3D-FM worked correctly in the case of the Java model. It has been shown that the coupling worked as multiple experiments have been performed. It has, however, also been shown that problems exists as the experiments with five ensembles were stopped after some runtime without a clear reason. Therefore, it is recommended to perform more runs with a small amount of ensembles in order to find what precisely causes the experiments to stop.

The next sub-goal was to research the improvements of data assimilation on the Java model for observed locations. Unfortunately, it has been shown that at this moment data assimilation does not only, not improve the Java model, it even reduces its performance as the Java model without noise shows better correspondence with the observations in both the twin-experiments, as the experiments with real observations. Various problems have been suggested that could be the cause. First of all, it has been shown that the problems are with the wind noise which becomes an order of magnitude larger after the first analysis step. Furthermore, this was the first project, that could be found, at the time of writing that uses an unstructured grid, which could therefore be another cause. This, however, could not easily be verified as there wasn't time to recreate the model with a structured grid. It is thus recommended to perform the same experiment with a structured grid in order to find if this does improve the assimilation over the experiment with the unstructured grid. If this is not the case a thorough research should be performed on the analysis of the wind noise within OpenDA in order to find why the noise becomes an order of magnitude larger during the analysis.

The third sub-goal was to research the improvements of data assimilation on the Java model for unobserved locations.The results on unobserved locations have been found to be quite bad; this was, however, expected as the observed locations already showed bad results. Therefore, the focus for this question was on researching the effects of localization on the assimilation. The localization seems to perform its job as it should. However, this

conclusion comes from the results being worse, when observed values are worse, while being better, when observed values are better. This could of course be the case because of different reasons, but could not be studied. It is thus recommended to study the effects of localization in a later stadium when the problems mentioned in the previous paragraph have been fixed.

The final sub-goal was to study if a sufficiently long time series could be created by making use of data assimilation, and the Java model, as this is needed for an extreme value analysis to perform correctly. Unfortunately, because of computational expensive experiments, only short runs, with a simulation period of a week, have been performed. These experiments have already had various problems, like being stopped after simulating only a few days, which could therefore also happen with longer runs. It has also be shown that finding a long enough time series of real observations is hard, which is not promising for long runs. It is, therefore, recommended to study this goal more thoroughly after other problems have been fixed. Furthermore, as it has been shown that the assimilation is quite computational expensive, it is advised to make sure high performance computational systems are available to compute on. This will, however, only be efficient if one uses something, like remote method invocation, to distribute the computational power available over the experiment.

To summarize, various recommendations have been given; first of all, experiments should be performed to study the various problems encountered in this research. The largest problem, namely the data assimilation making the Java model perform worse, can be studied by performing short runs with a few time steps. However, the latter problems, with localization and finding if a sufficiently long time series can be computed, will need longer runs to accurately study the improvements. Finally, if the latter experiments will be performed, one should make sure to have sufficient high performance computational systems available.

Bibliography

[1] Henry D. I. Abarbanel. Twin Experiments. pages 125–197. Springer New York, 2013. doi: 10.1007/978-1-4614-7218-6_5. URL http://link.springer.com/10.1007/978-1-4614-7218-6{_}5.

[2] Petteri Aimonen. File:Basic concept of Kalman filtering.svg, 2011. URL https://en.wikipedia.org/wiki/File:Basic{_}concept{_}of{_}Kalman{_}filtering.svg.

[3] Jeffrey L Anderson. A Hierarchical Ensemble Filter for Data Assimilation. pages 1–58, 2004.

[4] Laurent Bertino, Geir Evensen, and Hans Wackernagel. Sequential Data Assimilation Techniques in Oceanography. *International Statistical Review*, 71(2):223–241, 2003. ISSN 03067734. doi: 10.1111/j.1751-5823.2003.tb00194.x. URL http://dx.doi.org/10.1111/j.1751-5823.2003.tb00194.x.

[5] BODC. The GEBCO_2014 Grid. page 32, 2014.

[6] BODC. General Bathymetric Chart of the Oceans, 2016. URL http://www.gebco.net/.

[7] F Bouttier and P Courtier. Data assimilation concepts and methods. *Meteorological training course lecture series*, (January):1–58, 1999. URL http://www.msi.ttu.ee/{~}elken/Assim{_}concepts.pdf.

[8] P.C. Caldwell, M.A. Merrfield, and P.R. Thompson. Sea level measured by tide gauges from global oceans – the Joint Archive for Sea Level holdings (NODC Accession 0019568). *NOAA National Centers for Environmental Information*, 2015. doi: 10.7289/V5V40S7W. URL https://data.nodc.noaa.gov/cgi-bin/iso?id=gov.noaa.nodc:0019568.

[9] P. Courtier, J.-J. Thepaut, and A. Hollingsworth. A strategy for operational implementation of 4D-Var, using an incremental approach. *Quart. J. Roy. Meteor. Soc.*, 120, 1994. URL http://www.ecmwf.int/en/elibrary/8797-strategy-operational-implementation-4d-var-using-incremental-approach.

[10] P. Courtier, E. Andersson, W. Heckley, D. Vasiljevic, M. Hamrud, A. Hollingsworth, F. Rabier, M. Fisher, and J. Pailleux. The ECMWF implementation of three-dimensional variational assimilation (3D-Var). I: Formulation. *Quarterly Journal of the Royal Meteorological Society*, 124(550):1783–1807, jul 1998. ISSN 00359009. doi: 10.1002/qj.49712455002. URL http://doi.wiley.com/10.1002/qj.49712455002.

[11] D. P. Dee, S. M. Uppala, A. J. Simmons, P. Berrisford, P. Poli, S. Kobayashi, U. Andrae, M. A. Balmaseda, G. Balsamo, P. Bauer, P. Bechtold, A. C. M. Beljaars, L. van de Berg, J. Bidlot, N. Bormann, C. Delsol, R. Dragani, M. Fuentes, A. J. Geer, L. Haimberger, S. B. Healy, H. Hersbach, E. V. Hólm, L. Isaksen, P. Kållberg, M. Köhler, M. Matricardi, A. P. McNally, B. M. Monge-Sanz, J.-J. Morcrette, B.-K. Park, C. Peubey, P. de Rosnay, C. Tavolato, J.-N. Thépaut, and F. Vitart. The ERA-Interim reanalysis: configuration and performance of the data assimilation system. *Quarterly Journal of the Royal Meteorological Society*, 137(656):553–597, apr 2011. ISSN 00359009. doi: 10.1002/qj.828. URL http://doi.wiley.com/10.1002/qj.828.

[12] Deltares. RGFGRID - User Manual. 2015.

[13] Deltares. Delft3D, 2016. URL https://oss.deltares.nl/web/delft3d.

[14] Deltares. Delft 3D Flexible Mesh Suite. 2016.

[15] Deltares. D-FLOW FM User Manual. 2016.

[16] Deltares. Delft3D-FM, 2016. URL https://oss.deltares.nl/web/delft3dfm.

[17] A. T. Doodson. Meteorological Perturbations of Sea-Level and Tides. *Geophysical Journal International*, 1(s4):124–147, apr 1924. ISSN 0956540X. doi: 10.1111/j.1365-246X.1924.tb05363.x. URL http://gsmnras.oxfordjournals.org/cgi/doi/10.1111/j.1365-246X.1924.tb05363.x.

[18] ECMWF. European Centre for Medium-Range Weather Forecasts. URL https://www.ecmwf.int/.

[19] Gary D. Egbert and Svetlana Y. Erofeeva. Efficient Inverse Modeling of Barotropic Ocean Tides. *Journal of Atmospheric and Oceanic Technology*, 19(2):183–204, feb 2002. ISSN 0739-0572. doi: 10.1175/1520-0426(2002)019<0183:EIMOBO>2.0.CO;2. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0426{%}282002{%}29019{%}3C0183{%}3AEIMOBO{%}3E2.0.CO{%}3B2.

[20] Gary D. Egbert and Svetlana Y. Erofeeva. The OSU TOPEX/Poseidon Global Inverse Solution TPXO, 2010. URL http://volkov.oce.orst.edu/tides/global.html.

[21] Eva Oude Elferink. Wat Jakarta nodig heeft, is een eigen Afsluitdijk, jun 2017.

[22] Geir Evensen, Monte Carlo, and Monte Carlo. with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. 99, 1994.

[23] Mohinder S. Grewal and Angus P. Andrews. Applications of Kalman filtering in aerospace 1960 to the present. *IEEE Control Systems Magazine*, 30(3):69–78, 2010. ISSN 08880611. doi: 10.1109/MCS.2010.936465.

[24] Thomas Hamill and Jeffrey Whitaker. Distance-Dependent Filtering of Background Error Covariance Estimates in an Ensemble Kalman Filter. pages 2776–2790, 2001.

[25] R G Hanea, G J M Velders, A J Segers, M Verlaan, and A W Heemink. A Hybrid Kalman Filter Algorithm for Large-Scale Atmospheric Chemistry Data Assimilation. *Monthly Weather Review*, 135, 2007. doi: 10.1175/MWR3269.1. URL http://journals.ametsoc.org/doi/pdf/10.1175/MWR3269.1.

[26] A. W. Heemink, R. G. Hanea, J. Sumihar, M. Roest, N. Velzen, and M. Verlaan. Data Assimilation Algorithms for Numerical Models. pages 107–142. Springer, Berlin, Heidelberg, 2009. doi: 10.1007/978-3-642-03344-5_5. URL http://link.springer.com/10.1007/978-3-642-03344-5{_}5.

[27] A.W. Heemink. Storm Surge Prediction Using Kalman Filtering. 1986.

[28] Timothy J. Henstock, Lisa C. McNeill, and David R. Tappin. Seafloor morphology of the Sumatran subduction zone: Surface rupture during megathrust earthquakes? *Geology*, 34(6):485–488, 2006. ISSN 00917613. doi: 10.1130/22426.1.

[29] P L Houtekamer and H L Mitchell. A sequential ensemble {Kalman} filter for atmospheric data assimilation. *Monthly Weather Rev.*, 129(1):123–137, 2001.

[30] P. L. Houtekamer and Herschel L. Mitchell. Data assimilation using an ensemble Kalman filter technique. *Monthly Weather Review*, 126(3):796–811, 1998. ISSN 0027-0644. doi: 10.1175/1520-0493(1998)126<0796:DAUAEK>2.0.CO;2. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0493(1998)126{%}3C0796:DAUAEK{%}3E2.0.CO;2{%}5Cn{%}3CGotoISI{%}3E://000072409000016.

[31] INA. Jakarta's economic growth indicates improvement: Central Bank, 2016. URL http://www.ina.or.id/services/news/1892-jakartas-economic-growth-indicates-improvement-central-bank.

[32] Ingenieursbureau Witteveen+Bos. WAQUA en TRIWAQ. URL http://www.witteveenbos.nl/page/index/id/61190.

[33] Ingenieursbureau Witteveen+Bos. Masterplan for Jakarta, 2015. URL http://www.witteveenbos.com/en/masterplan-jakarta.

[34] Ingenieursbureau Witteveen+Bos. *BANJIR! BANJIR!* 2017.

[35] JavaTpoint. RMI (Remote Method Invocation). URL https://www.javatpoint.com/RMI.

[36] Aline Kaji, Hans Korving, and Maarten Jansen. Defining extreme water levels for design of coastal structures in Jakarta Bay , Indonesia. pages 1–12, 2016.

[37] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35, 1960. ISSN 00219223. doi: 10.1115/1.3662552. URL http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:A+New+Approach+to+Linear+Filtering+and+Prediction+Problems{#}0{%}5Cnhttp://fluidsengineering.asmedigitalcollection.asme.org/article.aspx?articleid=1430402.

[38] R. E. Kalman and R. S. Bucy. New results in linear filtering and prediction theory. *Journal of Basic Engineering*, 83(1):95–108, 1961. ISSN 00219223. doi: 10.1115/1.3658902. URL http://fluidsengineering.asmedigitalcollection.asme.org/article.aspx?articleid=1430804.

[39] Rama Rao Karri, Seng Keat Ooi, Vladan Babovic, and Herman Gerritsen. Improving predictions of water levels and currents for Singapore regional waters through Data Assimilation using OpenDA. (January), 2011.

[40] Rama Rao Karri, Stef Hummel, Ghada El Serafy, and Vladan Babovic. Data Assimilation for Water Levels and Currents in the Singapore Region : An Ensemble Steady State Kalman Filtering Approach. (July), 2012.

[41] Rama Rao Karri, Xuan Wang, and Herman Gerritsen. Ensemble based prediction of water levels and residual currents in Singapore regional waters for operational forecasting. *Environmental Modelling and Software*, 54(April):24–38, 2014. ISSN 13648152. doi: 10.1016/j.envsoft.2013.12.006.

[42] I Kasanicky and K Eben. Ensemble Kalman Filter. pages 25–30, 2011.

[43] H.W.J. Kernkamp and F. Zijl. Further hydraulic model studies for Pulau Ubin & Pulau Tekong reclamation scheme. *Delft Hydraulics Reports Z3437 for Housing and Development Board*, 2004.

[44] E. Klinker, F. Rabier, G. Kelly, and J.-F. Mahfouf. The ecmwf operational implementation of four-dimensional variational assimilation. III: Experimental results and diagnostics with operational configuration. *Quarterly Journal of the Royal Meteorological Society*, 126(564):1191–1215, jul 2000. ISSN 00359009. doi: 10.1002/qj.49712656417. URL http://doi.wiley.com/10.1002/qj.49712656417.

[45] Joel Kotkin and Wendell Cox. The world's fastest-growing megacities, 2013. URL https://www.forbes.com/sites/joelkotkin/2013/04/08/the-worlds-fastest-growing-megacities/{#}280500727519.

[46] L.D. Landau and E.M. Lifshitz. *Fluid Mechanics*. second edition, 1987. ISBN 978-0-08-033933-7.

[47] Lappeenrante University of Technology. Data Assimilation, 2011. URL http://personal.lut.fi/wiki/doku.php/en/technomathematics/inverse{_}problems/data{_}assimilation.

[48] John M. Lewis and John C. Derber. The use of adjoint equations to solve a variational adjustment problem with advective constraints. *Tellus A*, 37A(4):309–322, aug 1985. ISSN 02806495. doi: 10.1111/j.1600-0870.1985.tb00430.x. URL http://tellusa.net/index.php/tellusa/article/view/11675.

[49] A. C. Lorenc. Analysis methods for numerical weather prediction. *Quarterly Journal of the Royal Meteorological Society*, 112(474):1177–1194, oct 1986. ISSN 00359009. doi: 10.1002/qj.49711247414. URL http://doi.wiley.com/10.1002/qj.49711247414.

[50] Andrew C. Lorenc. A global three-dimensional multivariate statistical interpolation scheme. 1981.

[51] Andrew C. Lorenc. The potential of the ensemble Kalman filter for NWP—a comparison with 4D-Var. *Quarterly Journal of the Royal Meteorological Society*, 129 (595):3183–3203, 2003. ISSN 1477-870X. doi: 10.1256/qj.02.132. URL http://onlinelibrary.wiley.com/doi/10.1256/qj.02.132/abstract.

[52] Bertil Matérn. Spatial Variation. 37(5):84, 1960. ISSN 00390526. doi: 10.2307/2348387.

[53] L. A. Mcgee and S. F. Schmidt. Discovery of the Kalman filter as a practical tool for aerospace and industry. 1985. URL https://ntrs.nasa.gov/search.jsp?R=19860003843.

[54] Budiman Minasny and Alex B. McBratney. The Mat??rn function as a general model for soil variograms. *Geoderma*, 128(3-4 SPEC. ISS.):192–207, 2005. ISSN 00167061. doi: 10.1016/j.geoderma.2005.04.003.

[55] NK Nichols. Mathematical Concepts of Data Assimilation. *Data Assimilation: Making Sense of Observations*, pages 13–39, 2010. ISSN 1098-6596. doi: 10.1007/978-3-540-74703-1.

[56] NWO. Access to the National Computer Facilities for Pilot projects, 2017. URL https://www.nwo.nl/en/funding/our-funding-instruments/ew/access-to-the-national-computer-facilities/access-to-the-national-computer-facilities-for-pilot-projects/access-to-the-national-computer-facilities-for-pilot-projects.html.

[57] David F. Parrish, John C. Derber, David F. Parrish, and John C. Derber. The National Meteorological Center's Spectral Statistical-Interpolation Analysis System. *Monthly Weather Review*, 120(8):1747–1763, aug 1992. ISSN 0027-0644. doi: 10.1175/1520-0493(1992)120<1747:TNMCSS>2.0.CO;2. URL http://journals.ametsoc.org/doi/abs/10.1175/1520-0493{%}281992{%}29120{%}3C1747{%}3ATNMCSS{%}3E2.0.CO{%}3B2.

[58] Rich Pawlowicz, Bob Beardsley, and Steve Lentz. Classical tidal harmonic analysis including error estimates in MATLAB using T_TIDE. *Computers & Geosciences*, 28(8):929–937, oct 2002. ISSN 00983004. doi: 10.1016/S0098-3004(02)00013-4. URL http://linkinghub.elsevier.com/retrieve/pii/S0098300402000134.

[59] Maria Isabel Ribeiro. Kalman and Extended Kalman Filters : Concept , Derivation and Properties. *Institute for Systems and Robotics Lisboa Portugal*, (February):42, 2004. doi: 10.1.1.2.5088. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.5088{&}rep=rep1{&}type=pdf.

[60] Lewis Richardson. Weather prediction by numerical process. *Quarterly Journal of the Royal Meteorological Society*, 48(203):282–284, aug 1922. ISSN 00359009. doi: 10.1002/qj.49704820311. URL http://doi.wiley.com/10.1002/qj.49704820311.

[61] S. F. Schmidt. The Kalman filter - Its recognition and development for aerospace applications. *Journal of Guidance, Control, and Dynamics*, 4(1):4–7, 1981. ISSN 0731-5090. doi: 10.2514/3.19713.

[62] EAHC secretary. East Asia Hydrographic Commission, 2010. URL http://home.eahc.asia/activity.htm.

[63] S. D. Smith and E. G. Banke. Variation of the sea surface drag coefficient with wind speed. *Quarterly Journal of the Royal Meteorological Society*, 101(429):665–673, jul 1975. ISSN 00359009. doi: 10.1002/qj.49710142920. URL http://doi.wiley.com/10.1002/qj.49710142920.

[64] SURFsara. Cartesius: the Dutch supercomputer. URL https://userinfo.surfsara.nl/systems/cartesius/.

[65] Gabriel A Terejanu. Extended Kalman Filter Tutorial.

[66] Nils van Velzen, Muhammad Umer Altaf, and Martin Verlaan. OpenDA-NEMO framework for ocean data assimilation. *Ocean Dynamics*, (May 2015):1–12, 2016. ISSN 16167228. doi: 10.1007/s10236-016-0945-z. URL http://dx.doi.org/10.1007/s10236-016-0945-z.

[67] M Verlaan. *Efficient Kalman filtering algorithms for hydrodynamic models*. Number april. 1998. ISBN 9036934524.

[68] E.A. Wan and R. Van Der Merwe. The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pages 153–158. IEEE, 2000. ISBN 0-7803-5800-7. doi: 10.1109/ASSPCC.2000.882463. URL http://ieeexplore.ieee.org/document/882463/.

[69] P. Wilders. *Reader for Environmental simulation and data assimilation (WI4054)*. 2015.

[70] Yanfen Zhang and Dean S. Oliver. Improving the Ensemble Estimate of the Kalman Gain by Bootstrap Sampling. *Mathematical Geosciences*, 42(3):327–345, 2010. ISSN 18748961. doi: 10.1007/s11004-010-9267-8.

A

Additional figures

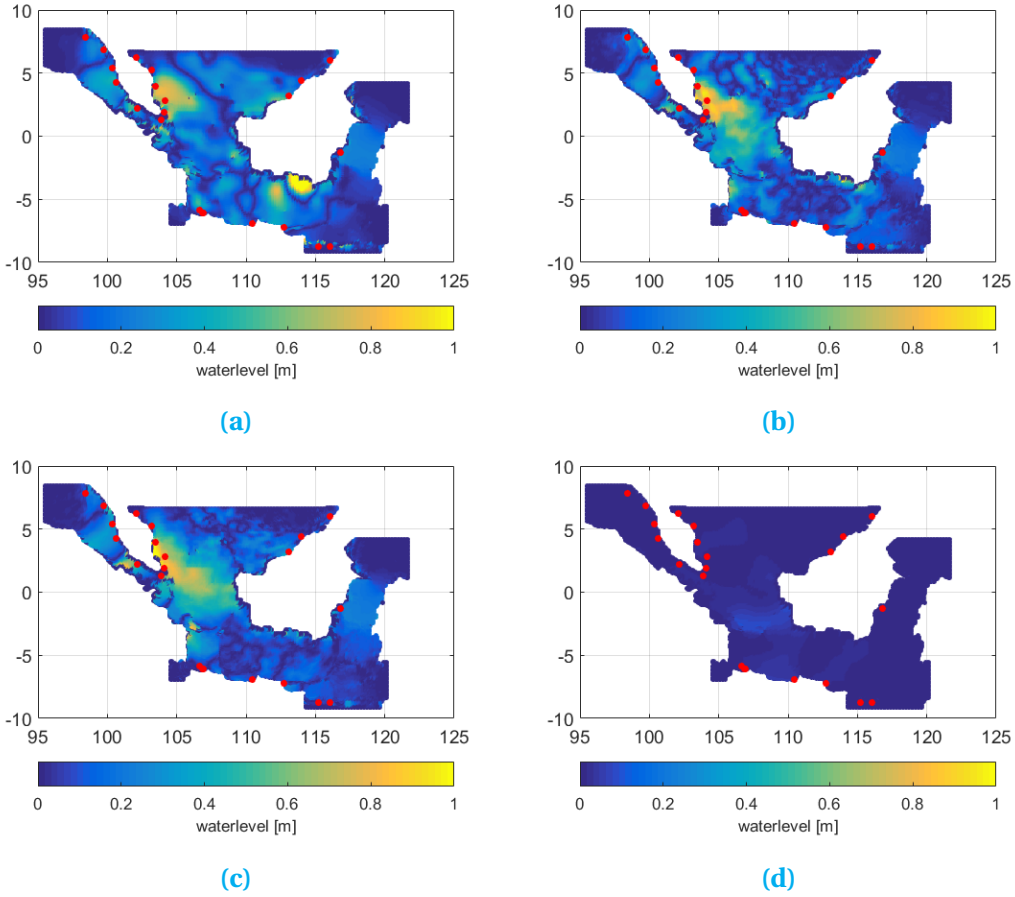


Figure A.1: The Root Mean Square Errors between the assimilated water levels and artificially generated observations at 18-1-2015 6h for, a) the 25-ensemble experiment without localization, b) the 25-ensemble experiment with localization, and, c) the 25-ensemble experiment with noise generated in OpenDA. The red dots are the observed locations and the size of the errors can be read from the colorbar.

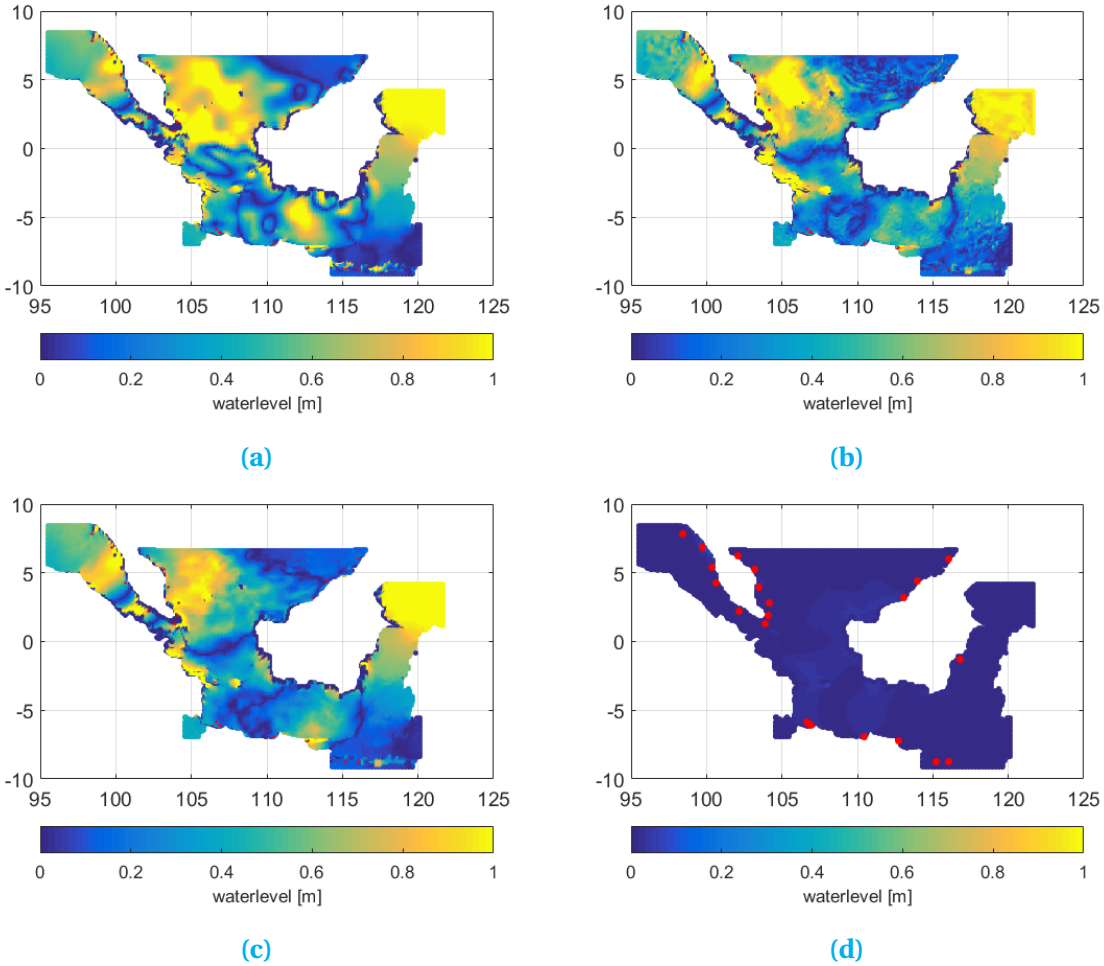


Figure A.2: The Root Mean Square Errors between the assimilated water levels and artificially generated observations at 18-1-2015 17h for, a) the 25-ensemble experiment without localization, b) the 25-ensemble experiment with localization, and, c) the 25-ensemble experiment with noise generated in OpenDA. The red dots are the observed locations and the size of the errors can be read from the colorbar.

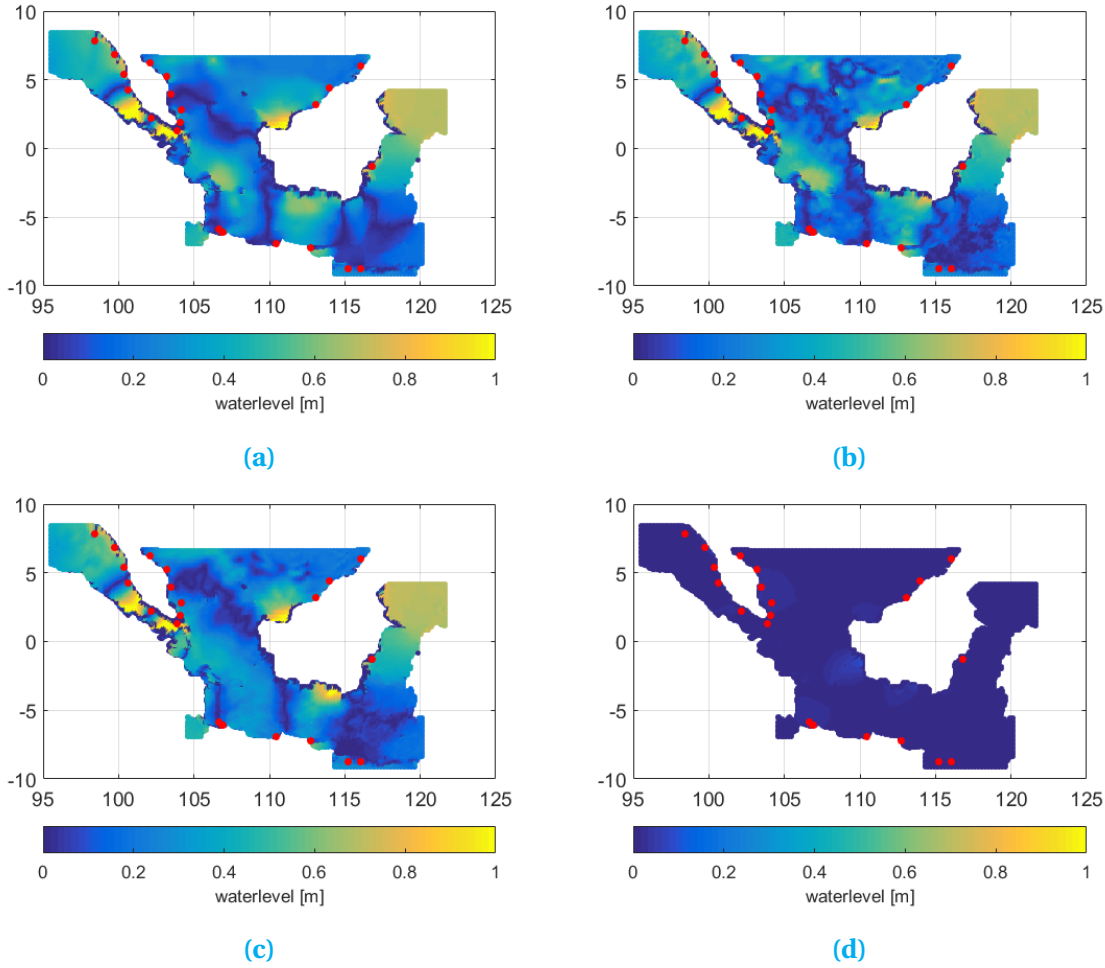


Figure A.3: The Root Mean Square Errors between the assimilated water levels and artificially generated observations at 16-1-2015 6h for, a) the 25-ensemble experiment without localization, b) the 25-ensemble experiment with localization, and, c) the 25-ensemble experiment with noise generated in OpenDA. The red dots are the observed locations and the size of the errors can be read from the colorbar.

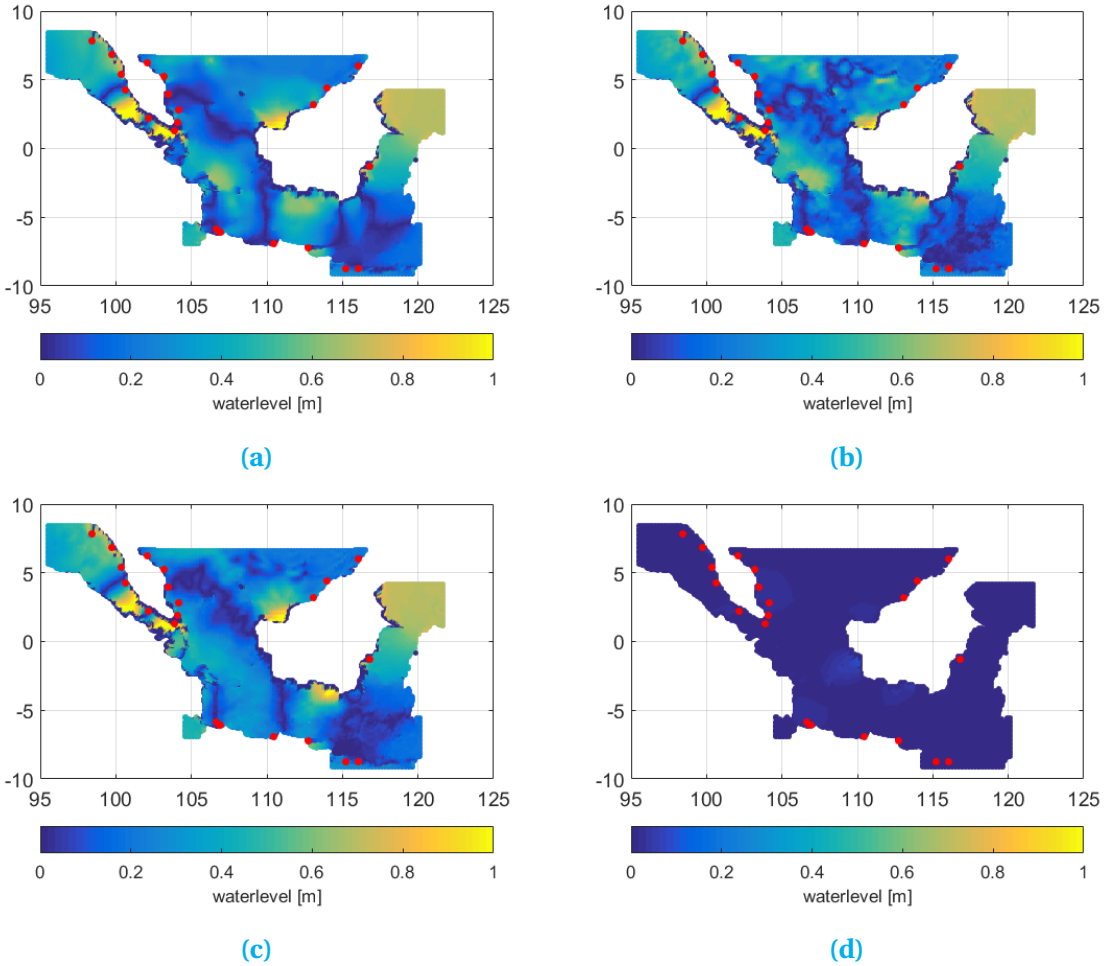


Figure A.4: The Root Mean Square Errors between the assimilated water levels and artificially generated observations at a) 18-01-2015 0h, b) 18-01-2015 6h, c) 18-01-2015 17h, and, d)16-1-2015 6h for the 25-ensemble experiment with noise generated in OpenDA and without localization applied. The red dots are the observed locations and the size of the errors can be read from the colorbar.

B

Additional tables

Table B.1: Observed locations twin-experiment

Location name	Longitude	Latitude
Balikpapan	116.80	−1.27
Benoa	115.22	−8.75
Bintulu	113.07	3.22
Cendering Chendering	103.18	5.27
Geting	102.10	6.23
Jakarta	106.85	−6.12
Keling	102.15	2.22
Kolinlamil	106.89	−6.11
Kota Kinabalu	116.07	5.98
Ko Taphao Noi	98.43	7.83
Kuantan	103.43	3.98
Langkawi	99.77	6.87
Lembar	116.07	−8.73
Lumut Pengkalan	100.62	4.23
Miri	113.97	4.40
Pari	106.62	−5.85
Penang	100.35	5.42
Sedili	104.12	1.93
Semarang	110.42	−6.95
Surabaya	112.74	−7.19
Tanjong Pagar	103.85	1.26
Tioman	104.13	2.80

Notes: In this table, the names and coordinates of the locations that have been observed within the twin-experiments are shown.

Table B.2: Observed locations UHSLC

Location name	Longitude	Latitude
Benoa	115.22	−8.77
Bintulu	113.06	3.26
Cendering Chendering	103.18	5.27
Geting	102.11	6.23
Johor Bahru	103.47	1.47
Keling	102.15	2.22
Keppel Harbour Tanjong Pagar	103.83	1.47
Ko Taphao Noi	98.42	7.82
Kuantan	103.43	3.98
Kukup	103.45	1.33
Langkawi	99.77	6.87
Lumut Pengkalan	100.62	4.23
Miri	113.97	4.40
Pinang Penang	100.35	5.42
Sedili	104.12	1.93

Notes: In this table, the names and coordinates of the locations, that have been used in experiments, from the UHSLC database are shown.

C

Matlab codes

C.1. Generation wind noise

```
clear variables;close all;clc;
%% Generate data needed
%file with winddata, needed for spatial and time components of the data
ncfile = 'D:\PIANC\ERA.nc';

t0 = ncread(ncfile,'time');
begintime=[2012,12,20];
endtime=[2014,1,1];

time = double(t0)/24 + datenum(1900,1,1); %make sure dates are in matlab datenumbers
clear t0
time=time(time >= datenum(begintime) & time < datenum(endtime));

filehandle{1}='windx_noise.amu'; %name_handles for noise files in Delft3D-FM
filehandle{2}='windy_noise.amv';

y=ncread(ncfile,'latitude');
x=ncread(ncfile,'longitude');
n_cols = size(x,1);
n_rows = size(y,1);

dx=abs(mean(diff(x))); % Regular grid expected, can also specify value directly
dy=dx;

stochModellocation='D:\Thesis\model_own\stochModel_noisematlab';
addpath(genpath(stochModellocation))


nr_ensembles=1; %Number of ensembles, each ensembles has need of it's own noise files
standarddeviation=2; %Standard deviation without correction for time correlation
lengthscale=100000; %Horizontal correlation scale in meters
time_step=time(2)-time(1); %time_steps of wind file in days
timescale=0.50; %Time correlation scale in days

alpha=exp(-time_step/timescale); %AR(1) alpha value
```

```
whitenoise_standarddeviation=standarddeviation*sqrt(1-alpha^2);

%% specify various values needed in header of the file.
OPT.header           = '';

OPT.filetype         = 'meteo_on_equidistant_grid';
OPT.nodata_value     = -999;
OPT.grid_file        = 'D:\Thesis\model own\wind.grd';
OPT.writegrd         = true;
OPT.quantity         = {{'x_wind'},{'y_wind'}};
OPT.unit             = 'm s-1';

OPT.refdatenum       = datenum(1970,1,1);
OPT.hr               = (time-OPT.refdatenum)*24;
OPT.timezone         = '+00:00';

OPT.OS               = 'unix';
OPT.CoordinateSystem = 'cartesian';
OPT.fmt              = '%.3g';
OPT.comment          = '';

%% Generate the noise

% Generate spatial correlation matrices
% This correlation is constant over all ensembles and timesteps
[spatial_cov_x_sqrt, spatial_cov_y_sqrt]=spatial_correlation(x,y,lengthscale);
% Generate a noise file for each ensemble

for i=1:nr_ensembles
    % Generate a noise matrix for each time_step of each wind component
    for j=1:size(filehandle,2) %generate both x and y noise
        noise=zeros(n_rows,n_cols,size(time,1));
        for k=1:size(time,1)

            noise_matrix=normrnd(0,1,n_rows,n_cols);

            correlated_noise_matrix=whitenoise_standarddeviation* ...
                spatial_cov_y_sqrt*(noise_matrix*spatial_cov_x_sqrt');

            if k==1
                noise(:,:,k)=correlated_noise_matrix;
            else
                noise(:,:,k)=correlated_noise_matrix+alpha*noise(:,:,k-1);
            end
        end

%% Open file
        %Open folder that should contain the noise file
        workfolder=strcat(stochModellocation,'\work',num2str(i));
        mkdir(workfolder)
        cd(workfolder)
        fid=fopen(filehandle{j},'w');

%% Write header to file
        fprintf  (fid,'### START OF HEADER');
        fprinteol(fid,OPT.OS)
```

```
        fprintf  (fid,['### Created with $Id:noise_generation.m ' ...
            '2017-01-31 13:00:00 marc4 $ $Headurl: $ on %s'],datestr(now));
        fprinteol(fid,OPT.OS)
        fprintf  (fid,'### creation of noise files with all zeros');
        fprinteol(fid,OPT.OS);

        fprintf  (fid,'FileVersion      = 1.03')                    ;
        fprinteol(fid,OPT.OS);
        fprintf  (fid,'filetype         = meteo_on_equidistant_grid');
        fprinteol(fid,OPT.OS);
        fprintf  (fid,['n_cols           =',num2str(n_cols)]);
        fprinteol(fid,OPT.OS);
        fprintf  (fid,['n_rows           =',num2str(n_rows)]);
        fprinteol(fid,OPT.OS);
        fprintf  (fid,'grid_unit        = m');
        fprinteol(fid,OPT.OS);
        fprintf  (fid,['x_llcorner       = ',num2str(min(min(x)))]);
        fprinteol(fid,OPT.OS);
        fprintf  (fid,['y_llcorner       = ',num2str(min(min(y)))]);
        fprinteol(fid,OPT.OS);
        fprintf  (fid,['dx               = ',num2str(dx)]);
        fprinteol(fid,OPT.OS);
        fprintf  (fid,['dy               = ',num2str(dy)]);
        fprinteol(fid,OPT.OS);
        fprintf  (fid,'n_quantity       = 1');
        fprinteol(fid,OPT.OS);
        quant=OPT.quantity{j};quant=quant{1};
        fprintf  (fid,['quantity1        = ',quant]);
        fprinteol(fid,OPT.OS);
        fprintf  (fid,['unit1            = ',OPT.unit]);
        fprinteol(fid,OPT.OS);
        fprintf  (fid,['NODATA_value     = ',num2str(OPT.nodata_value)]);
        fprinteol(fid,OPT.OS);

        fprintf(fid,'### END OF HEADER');
        fprinteol(fid,OPT.OS)
    %% Write data to file

        for l=1:size(time,1)
        fprintf  (fid,'TIME = %d hours since %s %s',...
            OPT.hr(l),... % write all decimals
            datestr(OPT.refdatenum,'yyyy-mm-dd HH:MM:SS'),...
            OPT.timezone);
        fprinteol(fid,OPT.OS)

            for m=size(noise,1):-1:1
                fprintf  (fid,[OPT.fmt,' '],noise(m,1:end-1,l));
                fprintf(fid, '%.6g',noise(m,end,l));
                fprinteol(fid,OPT.OS);
            end

        end
        fclose(fid);
    end
end
```

C.2. Spatial correlation

```
function [spatial_cov_x_sqrt, spatial_cov_y_sqrt]= ...
    spatial_correlation(x,y,lengthscale)

xmid=mean(x)*ones(size(y));
ymid=mean(y)*ones(size(x));

x_rad=degtorad(x);
y_rad=degtorad(y);
xmid_rad=degtorad(xmid);
ymid_rad=degtorad(ymid);

earthradiusopenda=6372800;

spatial_cov_x=zeros(size(x,1),size(x,1));
spatial_cov_y=zeros(size(y,1),size(y,1));

for i=1:size(x,1)
    for j=1:size(x,1)
        dx=cos(ymid_rad(j))*cos(x_rad(j))-cos(ymid_rad(i))*cos(x_rad(i));
        dy=cos(ymid_rad(j))*sin(x_rad(j))-cos(ymid_rad(i))*sin(x_rad(i));
        dz=sin(ymid_rad(j))-sin(ymid_rad(i));
        dist=earthradiusopenda*2.0*asin(0.5*sqrt(dx^2+dy^2+dz^2));
        spatial_cov_x(i,j)=exp(-0.5*dist^2/(lengthscale^2));
    end
end

spatial_cov_x_sqrt=sqrtm(spatial_cov_x);

for i=1:size(y,1)
    for j=1:size(y,1)
        dx=cos(y_rad(j))*cos(xmid_rad(j))-cos(y_rad(i))*cos(xmid_rad(i));
        dy=cos(y_rad(j))*sin(xmid_rad(j))-cos(y_rad(i))*sin(xmid_rad(i));
        dz=sin(y_rad(j))-sin(y_rad(i));
        dist=earthradiusopenda*2.0*asin(0.5*sqrt(dx^2+dy^2+dz^2));
        spatial_cov_y(i,j)=exp(-0.5*dist^2/(lengthscale^2));
    end
end

spatial_cov_y_sqrt=sqrtm(spatial_cov_y);

end
```

D

Scripts

```
#!/bin/bash
#SBATCH -t 5-00:00:00
#SBATCH -N 5

# Load correct Java module which compiled OpenDA
# copy inputfiles to temporary directory nodes.
module load java/oracle/8u73

cp -r $HOME/input /scratch-shared/goriscas1
cd /scratch-shared/goriscas1/input

# Compile runrmi.sh file

chmod +x runrmi.sh
export PATH=$PATH:~/runrmi.sh

# Specify variables; i.e. amount of threads, names of nodes

tot=25
k=0
hostnamesvar=""
facvar=""

nodes="$(nodeset -e $SLURM_NODELIST)"
echo "$nodes"

# Loop over all nodes used by batch run

for i in ${nodes}
do

# Start rmiregistry on the remote node with corresponding classpath
ssh $i killall -v rmiregistry

ssh $i
ipaddress=$(ssh $i "/sbin/ip -o -4 addr list eth0 | awk '{print $4}' ...
    | cut -d/ -f1" | awk '{print $4}')
```

```
echo $ipaddress

# append all jars in opendabindir to java classpath
for file in $OPENDADIR/*.jar ; do
   if [ -f "$file" ] ; then
       export CLASSPATH=$CLASSPATH:$file
   fi
done

ssh $i "module load java/oracle/8u73; export CLASSPATH=$CLASSPATH:$file; rmiregistry"

sleep 5

echo "$i"
echo "$k"
((START=$k * 5))
((END=$START+4))
((k+=1))
for ((j=START; j<=END; j++))
do
# start a Java remote server with on specified server with specified name
    echo "$j"
    cmd="/scratch-shared/goriscas1/input/runrmi.sh fac${j} $j $tot"
    echo ssh $i $cmd
       ssh $i "export CLASSPATH=$CLASSPATH:$file; $cmd" &

# Create string with node- and server names
if (( $j < 24 ))
then
    hostnamesvar+="$ipaddress,"
    facvar+="fac${j},"
else
    hostnamesvar+="$ipaddress"
    facvar+="fac${j}"
fi
done
unset ipaddress
echo $ipaddress
done
sleep 10

# Create and save rmi config file

echo '<rmiConfig>
<serverAddress>'"$hostnamesvar"'</serverAddress>
<factoryID>'"$facvar"'</factoryID>
<stochModelFactory className="org.openda.blackbox.wrapper.BBStochModelFactory">
<workingDirectory>/scratch-shared/goriscas1/input/stochModel</workingDirectory>
<configFile>dflowfmStochModel.xml</configFile>
</stochModelFactory>
</rmiConfig>' > /scratch-shared/goriscas1/input/stochModel/rmiconfig.xml

# Run OpenDA, is not parallel, so not needed to run with SLURM i.e.
cd /scratch-shared/goriscas1/input

$OPENDADIR/oda_run.sh Enkf.oda
```

```
# Copy output back to homefolder
cp -r /scratch-shared/goriscas1/input $HOME/output
```