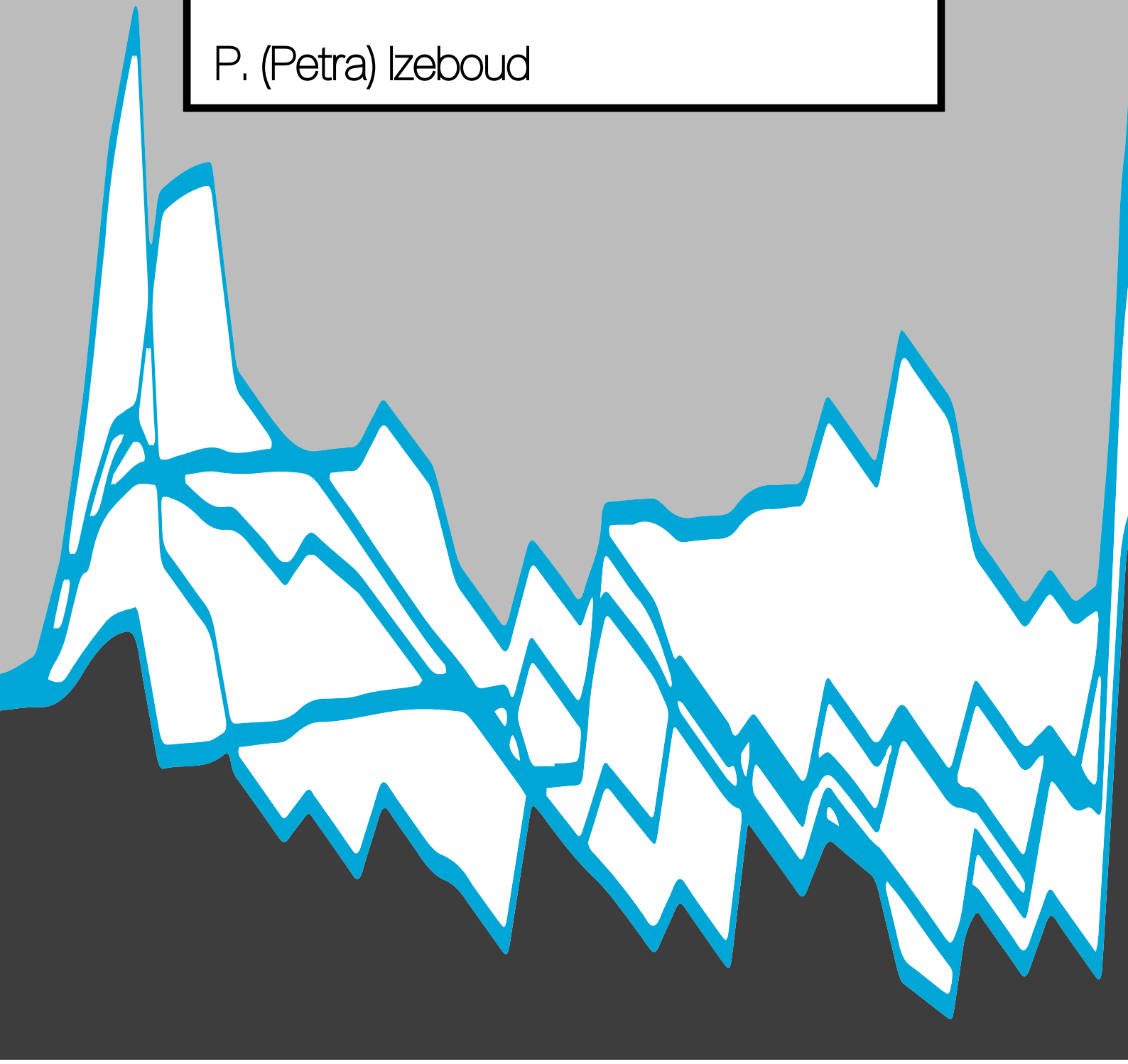
Reducing Water Level Prediction Uncertainty in a Delfland Polder using Data Assimilation

P. (Petra) Izeboud



TU Delft

Witteveen + Bos

Deltares
Enabling Delta Life

On the cover

The cover shows the assimilated water levels in Tedingerbroek polder, Delfland from 18:00 on June 22^{nd}, 2016 to 04:00 on June 23^{rd}, 2016. The blue lines represent different ensemble members, which show possible water levels. Sudden corrections that can be seen in the ensemble members correspond to the update step of the assimilation, where the modelled values of the water level are compared to observed water levels. The image was obtained using a data assimilation set-up with a rainfall multiplier. The cover is a detail of Figure 6.2b, which shows the influence of the non-linearity of the pumps to the application of data assimilation.

Reducing Water Level Prediction Uncertainty in a Delfland Polder using Data Assimilation

by

P. Izeboud

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday November 3rd, 2017 at 15:00 PM.

Student number:	4413601	
Project duration:	January 18, 2017 – November 3, 2017	
Thesis committee:	Dr. ir. G. H. W. Schoups,	Delft University of Technology
	Prof. dr. ir. N. C. van der Giesen,	Delft University of Technology
	Prof. dr. ir. M. Verlaan,	Delft University of Technology, Deltares
	Ir. H. Mondeel,	Witteveen + Bos
	Ir. S. Hummel,	Deltares

Preface

"I may not have gone where I intended to go, but I think I have ended up where I intended to be."
–Douglas Adams

This report is the result of ten months of research performed to finish my study in Water Management, Civil Engineering at the University of Technology in Delft. In my thesis, I enjoyed diving into a topic that dangled somewhere between applied mathematics and water management. After a bachelor in theoretical mathematics, this was the perfect topic: I could work on a new and challenging topic, which had a practical implementation and whose use could be explained to family and friends. During the time of my thesis, I learned to use new software, including Sobek RR and OpenDA, new programming languages, including Java and Python, and indulged in the mathematical fields of probability theory and data assimilation.
It is my belief that by definition af a thesis, some set-backs are required. My share was the long process of coupling the Sobek model to the software package OpenDA. Making this coupling started out optimistically, but ended up in debugging faulty settings for over three months. On the bright side, this is the part which taught me most about the process of research.
I would like to thank my committee; professor Nick van der Giesen, Gerrit Schoups, Herman Mondeel, Stef Hummel and professor Martin Verlaan for their support. Stef Hummel for always responding immediately to my mails, even when out of office, to tackle Software issues. The latter also holds for Martin Verlaan, my go-to supervisor for mathematical theories and OpenDA software implementation. Furthermore I would like to thank Herman Mondeel for his enthusiasm about my research and above all, my daily supervisor Gerrit Schoups for the many hours I could take up discussing the results which took so long before they made sense.
Besides my committee, I owe a big thank you to Bart Dekens, showing me the ropes at Witteveen + Bos, Erik Pelgrim my Java Yoda and everyone in the master room 4.84, of which especially Dorien Lugt my thesis-buddy and weekly ear for progress and discussion. Finally, I would like to thank Rens, Maaike, Phine, Larissa, Simone and my parents. It feels good when people are convinced you do great, without knowing what it is you did exactly.

Petra Izeboud

Delft, 22nd of October, 2017

Abstract

Polder areas as typical in the Netherlands require real time control on the water levels. Efficient pumping is a subject with increasing interest due to a required use of 20% green energy by 2020 set by the government. In 2010, the energy requirements already exceeded 175 GWh per year. For optimal control, water level predictions are important. However, they are uncertain due to errors in model structure, parameter estimation, initial states and observed forcing data. Especially the rainfall was found to contribute to water level prediction uncertainty. The uncorrected radar rainfall forcing data available in real time underestimate the corrected radar rainfall measurement over 100%, worsening with increasing rainfall intensities.

The aim of the research was to asses whether data assimilation can reduce the uncertainty on the water level predictions in a Delfland polder. The polder is modelled with a conceptual rainfall-runoff model in Sobek RR, while the data assimilation is implemented as an ensemble Kalman filter in OpenDA. In order to manipulate the states in Sobek RR for data assimilation, the model was extended by a black-box wrapper that was newly created in Java for this research.

To asses the theoretical applicability of data assimilation in a polder area, four twin experiments were carried out, which differed only in the model components to which noise was applied. The four set-ups included the use of a rainfall multiplier, additive noise on the model states, additive noise on the groundwater state and a single multiplier for all states. The results were compared visually and objectively through the probabilistic Nash Sutcliffe, a model efficiency coefficient.

All set ups for the twin experiment generated good results for the assimilated water level, mimicking the observed water levels closely with probabilistic Nash Sutcliffe values ranging between 0.92 to 0.97. However, only the set up with the rainfall multiplier exhibited good score measures for the hidden states, with an average of 0.6, showing a significant improvement over the case without assimilation. The other twin experiments failed to represent the hidden states, with average score measures below zero.

The data assimilation set-up with the rainfall multiplier was then applied using water level observations from Tedingerbroek polder. The pump pattern and magnitude of water levels of the modelled and observed water level differed significantly. The influence of the different model components on the modelled water level was assessed, from which was concluded that these could not account for the difference in modelled and observed values. Possible explanations are due to a non-representative measurement point in the polder or additional unknown water fluxes coming into the polder.

Recommended further research should focus on creating a larger observation network; implementing multiple water level observation locations, collecting pump discharges and taking measurements of the storage in greenhouse basins. The last could give a method to validate the assimilation of the hidden states, whereas the pump discharges should be used as additional observation in the data assimilation.

Contents

List of Figures

List of Tables

Nomenclature

List of Acronyms

CRR	Conceptual rainfall-runoff
BOS	Besturings ondersteunend systeem
DA	Data assimilation
EnKF	Ensemble Kalman filter
HHvD	Hoogheemraadschap van Delfland
KF	Kalman filter
GH	Greenhouse
RR	Rainfall-runoff
MHE	Moving horizon estimate
MjD	Modified Julian date
MPC	Model predictive control
NS	Nash Sutcliffe
NSp	Probabilistic Nash Sutcliffe
OW	Open water
PA	Paved
PF	Particle filter
RTC	Real time control
SSM	State-space model
UP	Unpaved
WWTP	Waste water treatment plant

List of Symbols and Operators

A_{GH}	[ha]	Total greenhouse area
$A_{GH,i}$	[ha]	Area of greenhouses of class i
A_{OW}	[ha]	Open Water Area
A_{PA}	[ha]	Paved area
A_{UP}	[ha]	Unpaved Area
c	[-]	Concentration coefficient
$d_G H$	[m]	Typical depth of greenhouse basins
ET	[mm/day]	Evapotranspiration
ET_a	[mm/day]	Actual evapotranspiration
F		Collective symbol for forcing data

F_{obs}		Observed forcing data
h_3	[-]	Soil water pressure head
H_k		Observation matrix
I_{max}	[mm/hour]	Maximum infiltration capacity
K		Kalman Gain
k_t		Error on parameters
m_t		Error in model structure
M_k		Evolution matrix
$\mathcal{M}$		Deterministic CRR model
P	[mm/5 min]	Precipitation
P_{unc}^{obs}	[mm/5 min]	Uncorrected radar rainfall
P_{1st}^{obs}	[mm/5 min]	First correction of radar rainfall
P_{2nd}^{obs}	[mm/5 min]	Second corrected radar rainfall
q_{CR}	[m/s]	Capillary rise
Q_D	[m^3/s]	Drainage to open water
Q_{GH}	[m^3/s]	Outtake of basins for use in GH
Q_I	[m^3/s]	Infiltration to unsaturated zone
Q_{OF}	[m^3/s]	Overflow from GH basins to open water
Q_P	[m^3/s]	Percolation to deep groundwater
Q_{pump}	[m^3/s]	Discharge of pumps
Q_{sew}	[m^3/s]	Discharge to sewage
Q_{SO}	[m^3/s]	Sewage overflow
Q_{SR}	[m^3/s]	Surface runoff
Q_{US}	[m^3/s]	Flux from unsaturated zone to groundwater
Q_{WWTP}	[m^3/s]	Discharge to WWTP
R	[day]	Resistance factor
r	[-]	Rainfall multiplier
s	[-]	Soil type
s_0	[m^3]	Error on initial states
$S_{basins,i}$	[m^3]	Storage in basin i
S_{land}	[mm]	Storage on land (UP)
S_{GW}	[m^3]	Storage in groundwater
S_{sew}	[mm]	Storage in sewage
S_{street}	[mm]	Storage on the street
S_{OW}	[m^3]	Open water storage
S_{US}	[m^3]	Storage in unsaturated zone
T	[mm/day]	Transpiration
V_{eq}	[-]	Equilibrium moisture storage
V_{pot}	[-]	Potential water content in the root zone
v_t	[m]	Observation error of the water level
w_t	[mm/5 min]	Error in forcing observations
x		State vector
y	[m]	Water level state
α_E	[-]	Limiting factor for evaporation
α	[-]	Factor for correlation in data assimilation
Δh	[m]	Difference in groundwater and open water level
Δh_{ref}	[m]	Water level difference from set water level
Θ		Collective symbol for parameters

1

Introduction

Technological process has merely provided us with more efficient ways of going backwards
– Aldous Huxley, Brave New World

Water resource management aims at controlling water systems such that they meet the quantity and quality needs that are set. This often includes the operation of structures such as pumps and dams (Grigg, 2005). Regulating a water system does not just require a certain capacity but also the proper operation of the structures present. This is done by controllers which can be either human or software driven and in practice are often a mix of both. The operation of a water system is often challenging, amongst other things because of uncertainties in forcing data and model predictions.
For flood control, management needs to be enforced in a relative small time scale in the range of minutes or hours, referred to as *Real Time Control (RTC)*. In the Netherlands, RTC of pumping stations and weirs is needed to keep the water in polders at a desired level. These systems can be operated in real time by processing new information (e.g. predicted weather conditions or measured water levels) as it comes in (Mayne et al., 2000). Over the last decade, much research has been devoted to improving and defining new methods that allow better real time water management. This thesis follows in that trend.
In this chapter an explanation of the current methods used to improve RTC systems can be found, including background information on model predictive control, data assimilation and model uncertainty analysis. This leads to the motivation and relevance for this thesis, including the research questions. The chapter concludes with a guide for reading the report.

1.1 Background

1.1.1 Model Predictive Control

The method most commonly used to calculate optimal operation of a RTC system is model predictive control (MPC) (Lee, 2011; Bemporad, 2011). In RTC systems using MPC, different control options (e.g. pump discharges) are optimized such that their corresponding system response (e.g. water level) fulfils set goals (e.g. a water level close to a set water level). The system response is calculated using a model over a finite horizon, meaning only a limited time horizon is taken into account to evaluate the system response. Instead of giving the optimal actions for the complete time horizon, the output of the MPC is the optimal control action of the current time-step only. This way, when new information about the states or forecasts come in, the optimization can be recalculated using the current state of the system as initial state (Mayne et al., 2000). Using real time data to immediately update the model makes it an on-line approach.
The biggest difficulty for MPC in real time operational systems is the small time scale in which optimal actions need to be calculated. Solving the optimization problem is

generally slow and requires recalculations of the coupled hydrological model (Wang and Boyd, 2010). Therefore, the coupled hydrological models usually need to be simplified drastically to ensure the requirement is met. This can reduce the reliability of the model outcome and hence of the proposed control actions based on this.

1.1.2 Data Assimilation

Another method to improve the operation of water resource systems is Data Assimilation (DA) (Liu et al., 2012), often used in combination with MPC. DA is the concept of updating model parameters or states using online measurements of the real system (e.g. discharge or precipitation measurements). These are used to reduce the difference between the modelled and the measured output (Pedersen et al., 2016). Different DA approaches have been developed to suit a wide variety of models including conceptual rainfall-runoff (CRR) models and, even though much less common, real-time control models (Pedersen et al., 2016). In combination with hydrological models, DA methods were found to be promising in reducing model uncertainty.

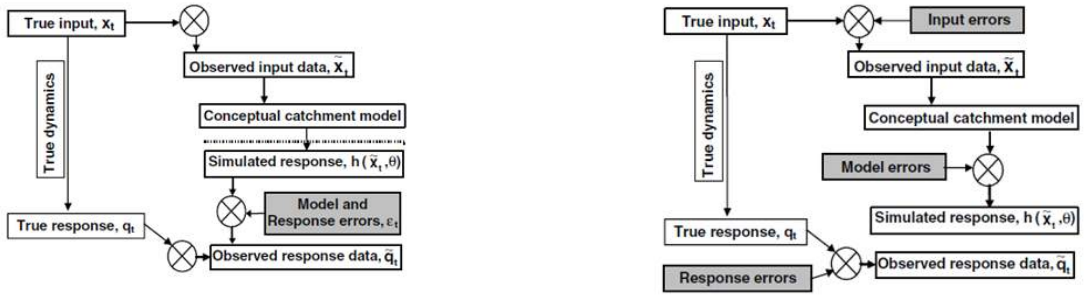
1.1.3 Model Uncertainty

For all models assumptions need to be made on how errors propagate through the model. This becomes more so if the model needs to give 'real time' outputs, requiring often oversimplification of the model to reduce calculation time. The uncertainty in predicted stream flow by CRR models has been a topic of interest among hydrologists for a long time and its importance for good decision making in control of structures is well known (e.g. Pappenberger and Beven, 2006; Schaake et al., 2006; Brown, 2010).
Figure 1.1a shows the error that is usually taken into account when validating, calibrating or applying models; one lumped term called the model error representing all uncertainties caused by errors in parameter estimation, model structure, initial states and observed forcing data. The forcing error is mostly neglected completely as source of uncertainty by assuming that the measured and forecast input data equal the true forcing data (Kuczera et al., 2006). This not only influences the calibration of models, but also reduces the understanding of where the errors originate from. In its turn, this reduces the option to target the most import sources of error. An improved conceptualisation of model errors is illustrated in Figure 1.1b, where three different sources for model errors can be distinguished (e.g. Kavetski et al., 2002; Kuczera et al., 2006):

1. **Errors in the observation of forcing inputs.** Especially precipitation measurements can be very different from the physical reality.
2. **Errors in the model structure and parameters**. These cause the deviation in model response and true response even if the forcing input and the output were measured without error.
3. **Errors in the measured response of the catchment.** For example, measured water level is subject to errors caused by the measurement device.

Another important error, not mentioned in Figure 1.1b is **the error in the initial conditions** (N. K. Ajami, Q. Duan, 2007); a rainfall event on a system that is already saturated will generate a different response than on an empty system.

Dealing with different sources of uncertainty In the last ten years, finding the different error sources has become of interest. It allows to find the main causes of uncertainty in the model, so that improvements can be pinpointed to the appropriate causes (Liu



(a) Current conceptualization

(b) Improved conceptualization

Figure 1.1: Schematic of error propagation in a conceptual RR model (taken from Kavetski et al. (2002))

et al., 2012). Furthermore, knowing the contribution of model uncertainties also allows to compare a model to others without obscuring the results through errors in the forcing and output data (Renard et al., 2010).
One method to address model uncertainties is by abandoning the notion of a deterministic model and making it a probabilistic model instead, for example by putting it in an Bayesian context (Jin et al., 2010). DA methods such as the Kalman Filter can also be interpreted in a Bayesian context (Hairer et al., 2005) and can be used not only to reduce the input uncertainty but also help showing the pain-points in the model.

Role of forcing data uncertainty Forcing errors can dominate the causes for model uncertainty. Rainfall is seen as the forcing input that is most uncertain due to short correlation length scales as well in space as in time (e.g. Huard and Mailhot, 2006; Renard et al., 2010; Xu et al., 2006). Furthermore, Kuczera et al. (2006) found the rainfall multiplier (used to simulate the error in precipitation) to be the second most sensitive parameter of the CRR model in their case study.
Input precipitation data is difficult to improve because new methods that account for better spatial and temporal variation, such as radar and satellites, have errors that exceed those of operational station networks (Volkmann et al., 2010). These (ground) radar measurements are useful and commonly used in RTC systems since they give near real time data with high spatial scale, at the price of higher uncertainty (Carpenter and Georgakakos, 2004).

1.2 Objective & Research Questions

The different backgrounds come together in the main research question of this thesis:

'How can data assimilation methods be used in real time hydrological models to reduce the uncertainty on predicted water levels?'

The scientific relevance of this question is that it brings together three topics in which a lot of research has been done in each topic on its own, but not in the combined field. It is obvious that larger water level prediction uncertainties can be expected when models are oversimplified to fulfil the short calculation requirements needed for RTC. Hence, the possible gain in applying DA methods to improve the modelled states is high. Furthermore, DA methods are very suitable for RTC systems in water management,

since often real time observations of pump discharges and water levels are collected and available to be used to update the modelled states.
The most important reason that these fields are not more commonly combined is that both DA and MPC require large calculation times, while RTC systems require outcomes in a very small time slot. Furthermore, mixing DA and MPC introduces difficulties because both methods influence the outcome of the optimization, which can make it impossible for the algorithm to find a solution close to the optimum.
The practical relevance is for to improve the optimization of control actions and was initiated by the Water Boards, responsible for the water levels in polders, and civil consultancy firm Witteveen and Bos. The water boards are required to use 20 % green energy in 2020. Energy-efficient use of the pumps to reduce the amount of energy used is therefore a great possibility, as the combined energy needed to keep the water at set level of the water boards in the Netherlands was already 176.8 GWh per year in 2010 (Dahm et al., 2010). This is the equivalent energy use for approximately 50.500 households. In order to reduce energy for pumping, e.g. by reducing pumping time, accurate forecasts of the water levels need to be possible. DA could potentially assist in improving this forecast.
The research will be applied to a hydrologic CRR model for a polder area as are common in The Netherlands. To obtain an answer for the main research question, five sub-questions will be answered.

1. *Which model components have the largest influence on the uncertainty in water level predictions; parameter values, initial states or forcing data?*

   Finding the components which are most uncertain and to which the water level predictions are most sensitive will help to pinpoint the uncertainties which the data assimilation should aim to reduce.

2. *Are there technical or practical issues for the application of DA?*

   Some models are more suitable for DA than other. The aim should be to find whether technical issues that reduce the applicability of DA exist and how these should be dealt with.

3. *Which model components can best be perturbed for data assimilation, such that optimal state estimates are found?*

   The DA should reduce uncertainty in the model components which can result in improved estimates of the initial states which in their turn can improve the forecast water levels. Following the first research question, the aim is to find *how* to target the biggest source of uncertainty for water level predictions in the technical set-up. Noise can be applied to different model components, either by adding a certain value or by multiplying the model component. One example is the use of a rainfall-multiplier to represent uncertainty in the forcing data. Different methods will be tested in their ability to estimate the initial states of all the modelled storages, such as the groundwater storage or storage in greenhouse basins.

4. *Can DA, with a perfect model, recover the right initial states while running with distorted forcing data?*

   Before testing with an actual case study, for which no validation is possible, it is interesting and necessary to know if DA can work in theory on a real-time polder model.

5. *Can prediction uncertainty be reduced using DA in a case study where many different components contribute to the uncertainty in water level predictions?*

When considering an actual case, much more uncertainty about sources of errors has to be dealt with. The distributions of the uncertainties are now unknown and validation is not easily possible. Data is clustered and different water levels even within the polders apply. Is it possible to apply the data assimilation in this situation?

1.3 Thesis Outline and Guidelines for Reading

In Chapter 2 the model structure is explained and uncertainties used for this research are defined, giving body to the probabilistic model used in this research. Chapter 3 describes the polder area under control of Hoogheemraadschap Delfland and the observed forcing data, used for the case study. How the model is implemented will be discussed in Chapter 4. This includes an overview of possible DA techniques and the theory behind it for readers less familiar with the topic. It also explains the concept of a twin experiment, which is used to quantify which data assimilation method works best. Chapter 5 includes a discussion of the software used to perform the hydrological and data assimilation calculations. To let these programs work together, a coupling was programmed for this thesis. Some flaws and useful points were found and discussed. In Chapter 6 the results are presented, which are discussed in Chapter 7. This chapter also includes the recommendations. Conclusions to the main research question and sub-questions can be found in Chapter 8.

2

Probabilistic rainfall-runoff model of a polder

"We balance probabilities and choose the most likely. It is the scientific use of the imagination. "
–Sherlock Holmes, The Hound of the Baskervilles. AC Doyle, 1901.

A **polder** is part of a canal-drainage system, of which a representation is given in Figure 2.1. The lower areas in the canal-drainage system are called the polder areas, where polder pumps (poldergemalen) discharge the excess water to the **boezem**. The boezem is a connected system of larger canals. In dry periods, water can be channeled from the boezem back into the polder to maintain the desired water level. During rain events, water can be pumped to the boezem to prevent flooding in the polder. The water levels in the polders are maintained at different set water levels (peil) depending on the function of the polder such as housing, nature area and farm land.
As the water levels in the polder determine a big part of the inflow into the boezem, it is important that they can be modelled well. To do this, proper modelling of the rainfall-runoff (RR) processes in the polder are of importance. This chapter will describe a probabilistic framework focusing on the deterministic model set-up and its probabilistic extension.

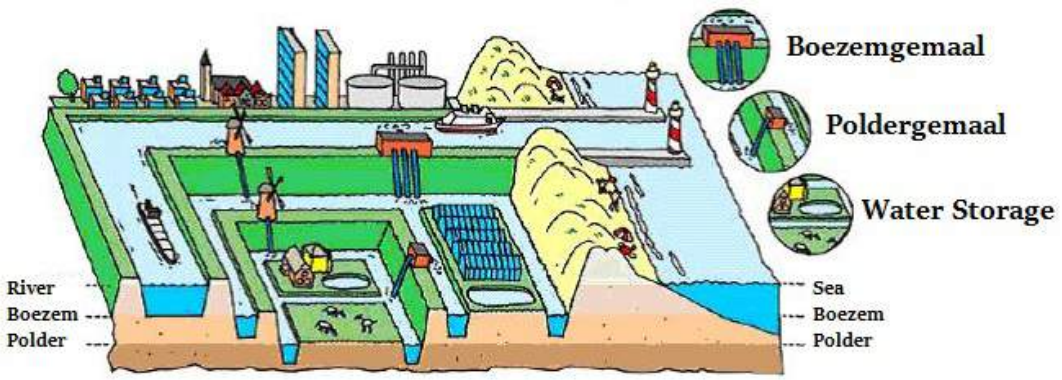


Figure 2.1: The concept of a canal drainage system (taken from Gemeente Rotterdam, 2017)

2.1 Deterministic rainfall-runoff model

The conceptual rainfall-runoff (CRR) set-up for a polder as used in Sobek RR is taken for the deterministic model in this research. Sobek RR is a software calculating rainfall-runoff using a distinction of four different land-use types: paved, unpaved, greenhouse and open water. Within these areas, different hydrological processes are dominant such

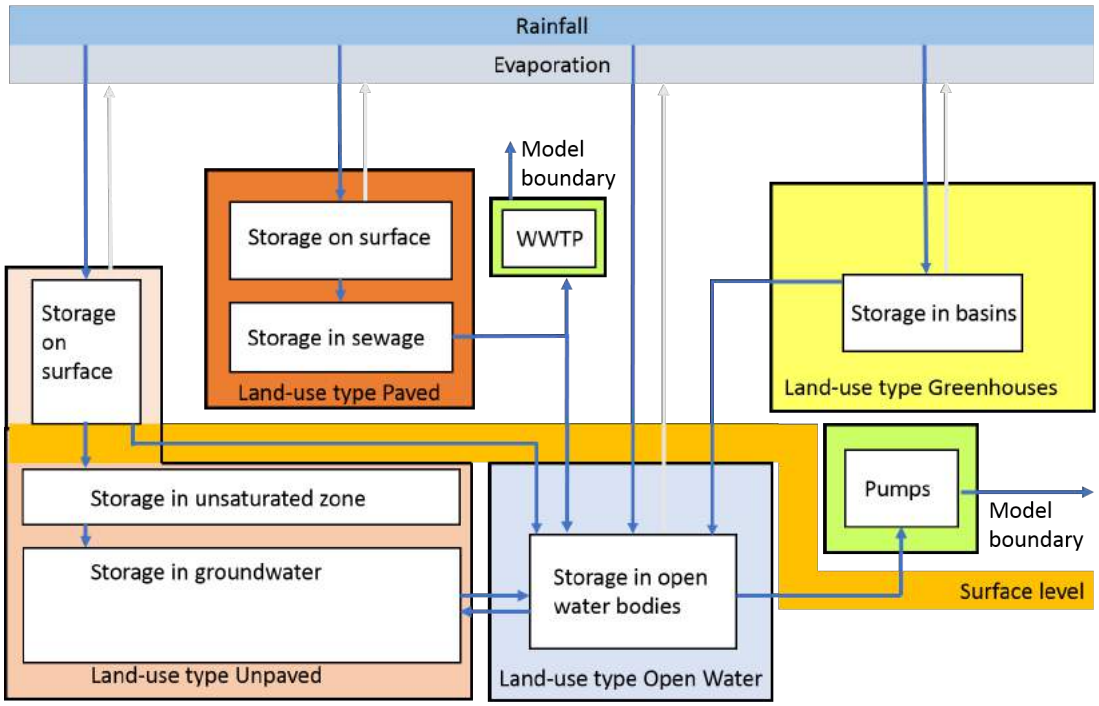


Figure 2.2: Overview of the deterministic model structure; different land-use classes contain different storage reservoirs. The paved, unpaved and greenhouse land-uses are disjointly modelled and all discharge to the open water land-use. The pumps regulate the open water level by removing water from, or adding water to, the water balance of the model. The blue arrows show which components interact with each other. WWTP stands for waste water treatment plant.

that their model structure is different. A complete overview of the model set-up can be found in Appendix A and is included because the nature of the system influences the results of the data assimilation significantly. A short explanation about the model is given in the next paragraph and is visualized in Figure 2.2. An overview of the parameters, states, forcing data and model names and symbols is given in Table 2.1.
The model implements forcing data (evaporation and rainfall in L/T) by multiplying it with the area of each land-use type. For the different land-use types, the rainfall is channelled to various reservoirs, including the groundwater storage, storage in greenhouse basins and storage in the unsaturated zone. These reservoirs allow a change in stored water volume and delay in run-off to the open water. The different land-use types, with exception of the open water land-use, are disjointly modelled. The allocation between the different storages per land-use type is given by equations mimicking rainfall-runoff processes such as percolation, infiltration and sewage overflow. All land-use types contain a series of the reservoirs, of which some are connected to the open water reservoir. The accumulated water in the open water bodies is maintained around a set water level by pumps, transporting the water outside the model boundaries. The pumps have a set turn on and off point; 0 cm above set water level and -5 cm below respectively.
In the bigger picture, these pumps discharge to the boezem and form the forcing input for the model predictive control system determining the optimal control of the boezem pumps. It is the pump discharge from the polder to the boezem the model output that this research aims to improve. Since the polder pump discharge depends on the water level, improving water level predictions has the same effect.

	Parameters Θ			States S			Forcing data F			Model $\mathcal{M}$	
	Name	Symbol	unit	Name	Symbol	unit	Name	Symbol	unit	Name	Symbol
Paved	Paved Area	A_{PA}	ha	Storage on Street	S_{street}	mm	Precipitation	P	mm/5 min	Deterministic CRR Model	$\mathcal{M}$
	Maximum Storage on the street	$S_{street,max}$	mm	Storage in sewage	S_{sew}	mm	Potential Evaporation	E_{pot}	mm/day		
	Maximum storage in the sewage	$S_{sew,max}$	mm								
	Maximal sewer pump capacity	$Q_{sew,max}$	m^3/s								
Greenhouse	Total greenhouse area	A_{GH}	ha	Storage in basins for $i = 1 \dots 10$	$S_{gh,i}$	m^3	Precipitation	P	mm/5 min		
	Area of greenhouse classes $i = 1 \dots 10$	$A_{GH,i}$	ha				Potential Evaporation	E_{pot}	mm/day		
	typical depth of storage basins	d_{GH}	m				Irrigation greenhouses	Q_{GH}	m^3/s		
Unpaved	Unpaved area	A_{UP}	ha	Storage on land	S_{land}	m^3	Precipitation	P	mm/5 min		
	Resistance coefficients $i = 1 \dots 4$	R_i	-	Storage in unsaturated zone	S_{US}	m^3	Potential Evaporation	E_{pot}	mm/day		
	Maximal storage on land	$S_{land,max}$	mm	Shallow groundwater storage	S_{SG}	m^3	Percolation to deep groundwater	Q_P	m^3/s		
	crop type	c	-								
	Max infiltration capacity	I_{max}	mm/hour								
	soil type	s	-								
Open Water	Open water area	A_{OW}	ha	Open Water	S_{OW}	m	Precipitation	P	mm/5min		
	Maximal pump capacity	$Q_{pump,max}$	m^3/s				Potential Evaporation	E_{pot}	mm/day		
	Maximal inlet capacity	$Q_{inl,max}$	m^3/s								

Table 2.1: Parameters, states, model and forcing data in the probabilistic model: names and symbols. Note that at the beginning of a model run, all states are represented by an initial state $S_{x,ini}$.

2.2 Uncertainties

Using the represented deterministic model to describe the rainfall-runoff introduces multiple types of uncertainty. Besides an overview of names and symbols, Table 2.1 shows the different contributors to uncertainty for this CRR model per land-use type taken into account in this study, grouped into four classes.
The first column gives the parameters, a first source of uncertainty because they are implemented as fixed values. In reality the value of these parameters are spatially distributed, or even a lumped representation of different properties themselves. The second column shows the model states, which at the start of each model run have initial filling degrees. The initial values of these states form a second source of uncertainty. Empty storages or a filled-up system will generate a completely different response in the next rainfall event. The third uncertainty comes from observation errors in the forcing data used to run the model and can account for a huge part of the uncertainty (Kuczera et al., 2006). The fourth column shows the uncertainty introduced by the simplified model structure f.
The fifth uncertainty, not shown in Table 2.1, stems from the uncertainty in the observed water level. It is not given in Table 2.1 because it does not influence the deterministic model. However, it will influence the data assimilation and is therefore taken into account as well in the probabilistic model set-up.
The uncertainty of the different components is represented by a probability distribution for each component. Since the distributions are not generic for different models, they are discussed in the case study in Chapter 3.

2.3 Probabilistic model

2.3.1 Why is a probabilistic model needed?

To account for the uncertainties introduced by parameters, initial states, forcing data, observation data and model structure, the model can be placed into a probabilistic context. That way, the stochastic errors that arise from the uncertainties, and get lost in a deterministic model, can be addressed (Renard et al., 2010). It has been argued that the notion of a deterministic model is not defensible at all due to the inevitable uncertainties in modelling (e.g. Kuczera et al., 2006). Besides this, DA works in a probabilistic framework, requiring the model to be defined as a probabilistic model as well.
The overall idea is to use the new uncertain observed water levels to reason backwards to the most likely corresponding forcing data and unmeasured states. In this approach, the uncertainties are represented by a probability distribution, which was propagated through a deterministic structure. One of the methods of 'backwards' reasoning is using Bayes' rule, which can be applied if the probabilistic model fulfils some characteristics (namely a state-space model). Bayes' rule can then be applied several times, calculating an improved (posterior) distribution of the states. The following sections will explain the concepts of the probabilistic framework and the state space model. The next chapter will then explain Bayes' rule and how this is used in the data assimilation.

2.3.2 Probabilistic model formulation

The deterministic model of section 2.1 (and more elaborate in Appendix A) can be seen as a function $\mathcal{M}$, which depends on a vector of states (x_{t-1}) at the previous time step, the forcing observations (F) and the set of parameters (Θ). The outcome of the model is

given by x_t, the state vector, giving the storages in the different reservoirs. The model outcome is mapped, by a function which shall be named H, to the state that is observed; for this research the water level (y_t) at time t. Furthermore, as initial condition, an initial state vector $x_{t=0}$ needs to be given. In formulas the deterministic system can be represented by:

$$x_t = \mathcal{M}(x_{t-1}, F_{t,obs}, \Theta) \tag{2.1}$$
$$y_t = H(x_t) \tag{2.2}$$
$$x_{t=0} = x_0 \tag{2.3}$$
$$y_{t=0} = H(x_0) \tag{2.4}$$

where

$F_{t,obs}$ = The precipitation and evaporation forcing from time $t-1$ to t.
Θ = The 28 dimensional parameter vector.
$\mathcal{M}$ = The non-linear deterministic CRR-model described in section 2.1, depended on the forcing data, the parameters and the previous state vector.
H = The function mapping state vector x to the modelled observable state y.

and where

x_t = The state vector at time t, including also the water level state.
y_t = The water level at time t, the state that can be observed

As discussed, each of the components is modelled with a single value, while in reality it carries uncertainties. To create a probabilistic model, a probability distribution of the errors for each item, such as the initial storage of the greenhouse basins, in each component, such as the initial storages, is estimated. Furthermore, the modelled water level including model uncertainty can be represented by the observation of the water level including the observation uncertainty v_t. The new schematic representation of the probabilistic model then becomes:

$$y_t = y_{t,obs} + v_t \tag{2.5}$$
$$x_t = \mathcal{M}(x_{t-1}, F_{t,obs} \cdot w_t, \Theta + k_t) + m_t \tag{2.6}$$
$$x_0 = x_0 + s_0 \tag{2.7}$$

where the uncertainties are represented by:

v_t = the observation error of the water level
w_t = the forcing error (evaporation and rainfall)
s_0 = the error on the initial states
k_t = the error on the parameters
m_t = the model structural error

The errors are chosen to be additive. An exception is the error on the rainfall which is taken to be a multiplier, following the approach of Kuczera et al. (2006). The reason for this is that often no-rain/rain can be predicted well whereas errors in observations are known to increase as rainfall-intensity increases (see the uncertainty analysis in Chapter 3).

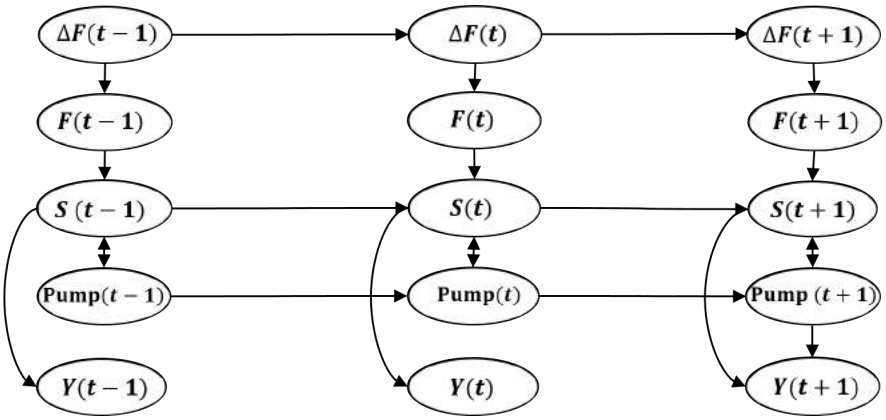


Figure 2.3: Representation of the model as a state-space model. The forcing F influences the (mostly hidden) states S, which can be mapped to the observed water level Y. As the model progresses in time, only the previous time step influences the next, such that the model fulfills the first order Markov property when also the pump state is included. ΔF represents the time correlation in the noise on the rainfall.

2.3.3 A State-Space Model

As discussed in Section 2.3.1, the aim is to apply data assimilation on the probabilistic model sketched in 2.3.2 to update the initial states using online data. Basically this means the aim is to apply state-space modelling: *'a form of modelling with the objective to estimate hidden states using a recursive application of Bayes' rule given the observations'* (Chen, 2003).
These so-called **state-space models (SSM)** satisfy two properties:

1. Firstly, the model has **hidden variables**. These include states and parameters that are not or cannot be measured by the observer (e.g. the soil moisture content) (Gharamani, 2001). The observations (in this case, the observed water levels) have to be a result of these hidden variables through the rainfall-runoff processes mimicked by the model.
2. Secondly, the model has to fulfill the **first order Markov property**, which states that if one knows the previous state S_{t-1}, one does not need to know states S_j with $j < t - 1$ to compute the current state S_t (Gharamani, 2001).

The hidden variables are clearly present in the model presented section 2.1. Of the 16 states given in Table 2.1, only one is measured; the storage in the open water bodies. The storage in the greenhouse basins could easily be measured but this is not the case. The first order Markov property only holds if besides the storage reservoirs, also the pump discharge and the correlation between the rainfall are viewed as states. The first because given a water level at time t, one can not deduce the pump discharge at time t without knowing the previous discharge. The following equation illustrates this:

$$Q_{pump}(t) = \begin{cases} Q_{pump,max} & \text{if } \Delta h_{ref}(t) > 0.2 \\ & \text{or } Q_{pump}(t-1) = Q_{pump,max} \ \& \ \Delta h_{ref}(t) > 0 \\ 0.5 \cdot Q_{pump,max} & \text{if } 0.1 < \Delta h_{ref}(t) < 0.2 \\ & \text{or } Q_{pump}(t-1) = 0.5 \cdot Q_{pump,max} \ \& \ \Delta h_{ref}(t) > -0.1 \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

where

$Q_{pump}(t) =$ The pump discharge in m^3/s from the polder to the boezem at time t.
$Q_{pump,max} =$ The maximal pump discharge in m^3/s from the polder to the boezem.
$\Delta h_{ref}(t) =$ The water level difference (m) from the set water level (reference point) at time t.

This shows that the pump can be turned on at time t even though the water level is already within the operational bounds of the water level. Furthermore, note that the pump is a discrete variable, only able to take three values. This is likely to cause problems for the data assimilation, which works with continuous states only.
The second additional state that needs to be introduced to fulfill the first order Markov property is the rainfall correlation ΔF. This component is not explained in the set-up of the deterministic model in Appendix A because it is not part of the Sobek model. However, it is implemented in the DA framework to account for correlation in the errors on rainfall and so should be part of the probabilistic framework. It is given by:

$$\Delta F(t+1) = \alpha \Delta F(t) + \eta(t), \tag{2.9}$$
$$\eta(t) \sim \mathcal{N}(0, \sigma_\eta) \tag{2.10}$$
$$N_0 = 0 \tag{2.11}$$

where

$\Delta F(t) =$ noise on rainfall at time t
$\alpha =$ dimensionless factor determining the correlation between two timesteps
$\eta(t) =$ independently drawn noise for time step t

The rainfall implemented in the model ($\widetilde{F}(k)$) is then:

$$\widetilde{F}(k) = F(k)(1 + \Delta F(k)) \tag{2.12}$$

Figure 2.3 gives a representation of the State Space model that is obtained.

3

Case: Polder Area Hoogheemraadschap Delfland

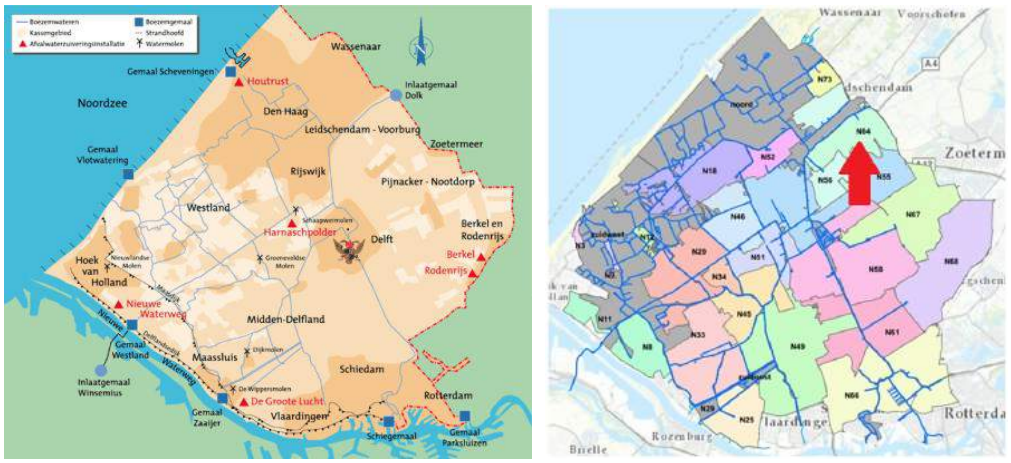
What the use of P implies, therefore, is that a hypothesis that may be true may be rejected because it has not predicted observable results that have not occurred.

– Sir Harold Jeffreys, Theory of probability, 1961

As much as theoretical cases can support hypothesis and theories, civil engineering is an applied science and it is good to know how the approach works with actual cases and data. That way, not only can the hypothesis be rejected because it cannot predict observable results that have not occurred, but also because it might not predict the observable result that *have* occurred.

3.1 Area of Hoogheemraadschap van Delfland

The area under consideration for the case study is the polder area regulated by the Hoogheemraadschap of Delfland (HHvD) and is shown in Figure 3.1a. The HHvD is an organization in the Netherlands which is among other things responsible for regulating the water levels in the polders and boezem to prevent flooding and drought.
The area of HHvD consists of a hundred polders (Beukema, 2017), bounded by the North Sea, the Nieuwe Waterweg and the line from Rotterdam to Zoetermeer and Wassenaar (Hoogheemraadschap van Delfland. www.hhdelfland.nl, 2017). The control for the polders is supported by a model system, BOS2.0, which consists of a Sobek RR model, a hydraulic model in Sobek and implementation of MPC in RTC-Tools. In the RR model, the polders are grouped into 25 polder areas as depicted in Figure 3.1b. The clustering involves an aggregation of storage and an averaging of the rainfall and potential evaporation over these areas. The total area is around 41,000 hectares and is inhabited by approximately 1,4 million people. Around 40,000 businesses are situated in the area and it is one of the most intensified glasshouse horticulture in the world (Hoogheemraadschap van Delfland. www.hhdelfland.nl, 2017).
The maximal allowed deviation from the set water levels in the polders is 0.2 m for a rainfall event of 100 mm in 48 hours with a return period of 190 years (Delfland, 2017). When a lot of rain is expected, the poldergemalen will lower the water level in anticipation (Hoogheemraadschap van Delfland. www.hhdelfland.nl, 2017). The set water level in Tedingerbroek polder is -1 meter below sea level. In anticipation of big rainfall events this can be lowered to -1.2 meter below sea level.
The boezem system that receives the excess water from the polders is controlled by the HHvD and is called 'Boezem van het Westland'. It consists of bigger canals like the Schie in Delft and the Vliet in Voorburg. The water level of the boezem is maintained at a constant level called the 'boezempeil', which equals -0.43 m NAP for Delfland (Hoogheemraadschap van Delfland. www.hhdelfland.nl, 2017) . The six boezemgemalen (placed in Rotterdam, Schiedam, Maassluis, Hoek van Holland, Ter Heijde and



(a) The modelled area of Delfland. The red triangles show the boezemgemalen, the blue lines give the main boezem water ways

(b) Clustering of the polders in BOS2.0, the red arrow indicating the Tedingerbroek polder (N64), the test case of this thesis.

Figure 3.1: Area of Case study: Delfland

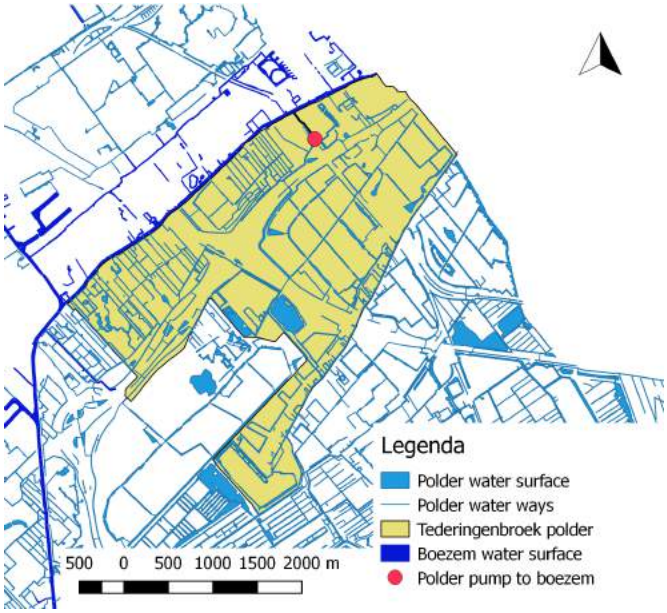


Figure 3.2: Tedingerbroek polder in Delfland; the area modelled in this research

Scheveningen) pump out the excess water present in the polder, whereas two inlets can raise the water level in case of droughts.
The Tedingerbroek polder (Figure 3.2) is chosen to test the hypothesis whether data assimilation can improve the predicted water levels in the polders. This polder was chosen since it has only one outlet to the boezem, consists of all land-uses described in Section A and has well documented water levels.

3.2 Data Sets

Different data sets are input for the probabilistic model discussed in Chapter 2. These are precipitation, potential evaporation, and water level measurements. An overview of the data products, their sources, temporal resolution and time horizon is given in Table 3.1.

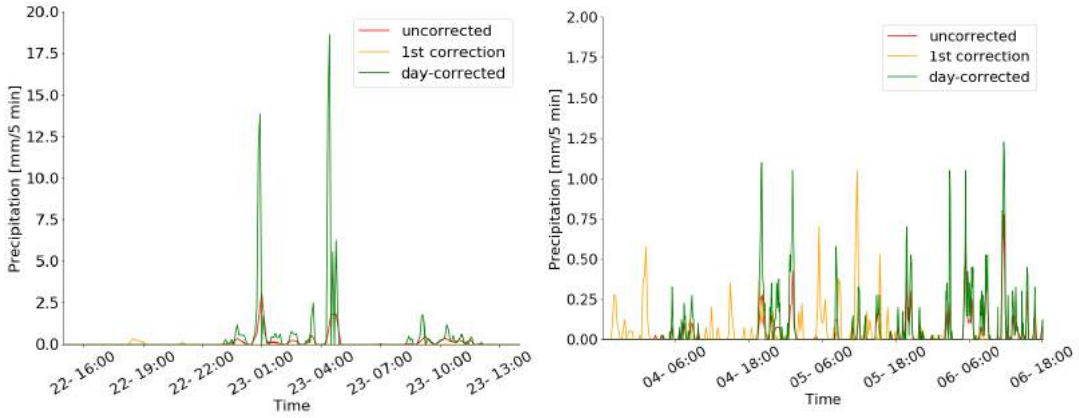
product	source	location	frequency available	measurement frequency	start series	end series
uncorrected radar-rainfall	Hydronet	AECL_N64 (Tedingerbroek polder)	15 minutes	5 minutes	02-03-2016 10:35	06-01-2017 19:00
1st corrected radar-rainfall	Hydronet	AECL_N64 (Tedingerbroek polder)	1 hour	5 minutes	09-06-2016 19:10	06-01-2017 19:00
day-corrected radar-rainfall	Hydronet	AECL_N64 (Tedingerbroek polder)	1 day	5 minutes	08-06-2016 19:10	06-01-2017 19:00
KNMI rainfall	KNMI	station 330 (Hoek van Holland)	1 day	1 hour	01-01-2015	31-05-2017
KNMI rainfall	KNMI	station 334 (Rotterdam)	1 day	1 hour	01-01-2015	31-05-2017
KNMI potential evaporation	KNMI	station 449 (Delft)	1 day	1 day	01-01-1951	31-05-2017
Water levels	HHvD	408101 (Tedingerbroek polder)	5-15 minutes	1-15 minutes	09-07-2015 09:00	17-03-2017 09:00

Table 3.1: Overview of used data series with corresponding source, frequency, location and start-end of time series. The frequency available gives the time-slot in which the data becomes accessible after the moment of measurement. The measured frequency gives the interval between two data points.

Precipitation The past, current and predicted precipitation used by Delfland and therefore also in the probabilistic model is given by HydroNET, a product provided by Hydrologic B.V.. The rainfall product has a spatial resolution of 1 x 1 km^2 and a temporal resolution of nearly real time: each 5 minutes. The real time data are the uncorrected radar rainfall product which is used for the calculation of the next 'model' time step. This product shall from here on be called the **uncorrected radar precipitation: $\mathbf{P^{obs}_{unc}}$**.
The uncorrected product is calibrated with precipitation measurements of more than 300 KNMI ground measurement stations in $24-32$ hours. This product shall be called the **2nd-corrected radar precipitation: $\mathbf{P^{obs}_{2nd}}$** and its reliability is supposedly equal to the precipitation measurements of a properly installed rain-gauge (Lobbrecht et al., 2012). This product is used by Delfland to restart the model as soon as it becomes available. Furthermore there is an hourly product, which gives a first correction on the uncorrected precipitation. This product does not differ much from the uncorrected rainfall product and is not used from here onwards. To give an idea of the huge differences between these products, they are shown for two events in Figure 3.3 and the cumulative amounts over a timespan of seven months in Figure 3.5. These differences are further discusses in Section 3.3.
To compare the radar rainfall product to a different rainfall product, the KNMI rainfall data at location Hoek van Holland and Rotterdam are also included in Table 4.1.

Potential evaporation Daily, gridded potential evaporation is given by a product of KNMI and is calculated using the method of Makkink. It depends on the temperature, wind and solar radiation. The product is determined daily at the KNMI station in Rotterdam.

Water level The water level is measured every 5 minutes at many places within the boezem and polder of Hoogheemraadschap van Delfland. Water levels near the polder pump of Tedingerbroek, as can be seen in 3.2, are supplied by the HHvD. The measurements are taken using a sensor in a tube with holes over which a filter is applied to



(a) Extreme rainfall event june 23 2016

(b) steady rainfall event 4 - 6 november 2016

Figure 3.3: Illustration of the difference between uncorrected, 1st correction and day (2nd) corrected radar rainfall products.

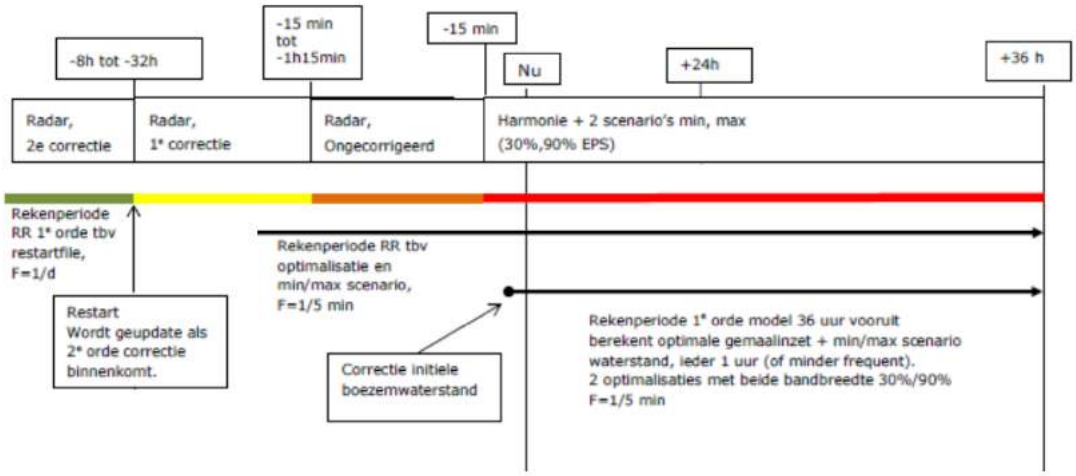


Figure 3.4: Timescale of the used rainfall products and restarts in BOS2.0, as used by Delfland for the operation of the pumps.

reduce the noise caused by disturbances (such as waves).

Practical implementation of the data for RTC The uncorrected and 2nd-corrected radar rainfall data and the water level measurements are used by Delfland to determine the operation of the pumps and the state of the system. The uncorrected radar is used to calculate the current water levels in the boezem and upto 36 hours in the future, so that the pumps in the boezem can be optimized using MPC. As soon as the 2nd-corrected rainfall becomes available, the model is recalculated and a restart is made with the updated states. The water levels are real time available so that the operators can check whether the operation is going well and they overrule the modelled pump settings whenever need be (and this happens daily). The time line is shown in Figure 3.4. For forecast calculations, a rainfall prediction product is used called Harmonie. Since data assimilation uses real time measurements to update the modelled system up to the current time, this product is not of interest for this research.

3.3 Probability distribution of the variables

The probabilistic CRR model for the polder as discussed in Chapter 2 is applied to the polder area of Delfland for this research. The probability distributions of the parameters, states, forcing and model structure as presented in Section 2.3 can now be specified for this particular case study. These are given for the parameters and initial states in Table 3.2 and were found from literature and interviews with water-operators. For the rainfall uncertainty, an analysis of uncorrected and corrected radar rainfall data is given below.

Uncertainty on the Radar Rainfall

The difference between the radar rainfall products is emphasized in Figure 3.5. This figure shows the sum of the 5-minute rainfall products over time. The difference in the total fallen rainfall in the period from the 9th of June 18.00 2016 until 6th of January 5.00 2017 between the uncorrected and 2nd corrected product is approximately 190 mm, an underestimation of 40%. The first period resulting in a significant difference in products corresponds to the extreme event on June $23^r d$. Two locations with KNMI rainfall data are also shown in Figure 3.5, showing a similar pattern as the 2nd corrected radar rainfall, strengthening the belief that the main error is in the uncorrected rainfall instead of the 2nd corrected rainfall. It is expected that a difference this large will influence the modelled water levels significantly. The 1st correction on the radar rainfall does not show a real improvement to the uncorrected product.
Figure 3.6 shows that the underestimation of the uncorrected radar rainfall increases when intensities increase. The error between these products was analysed from 58180 measurements. The found relation between the 2nd-corrected and the difference in

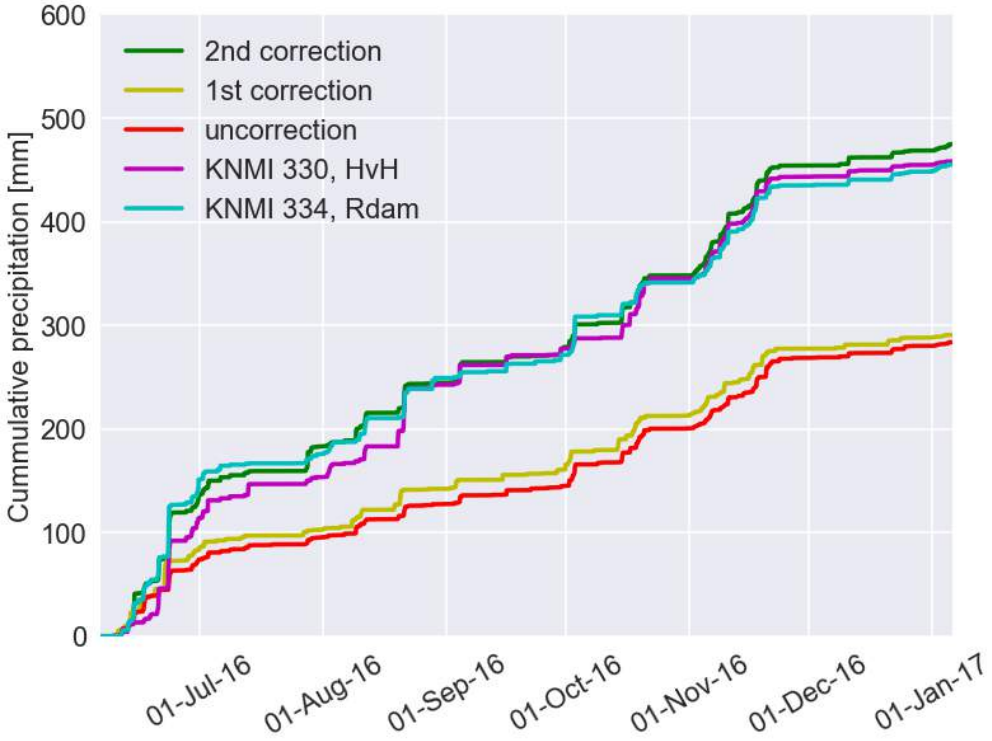
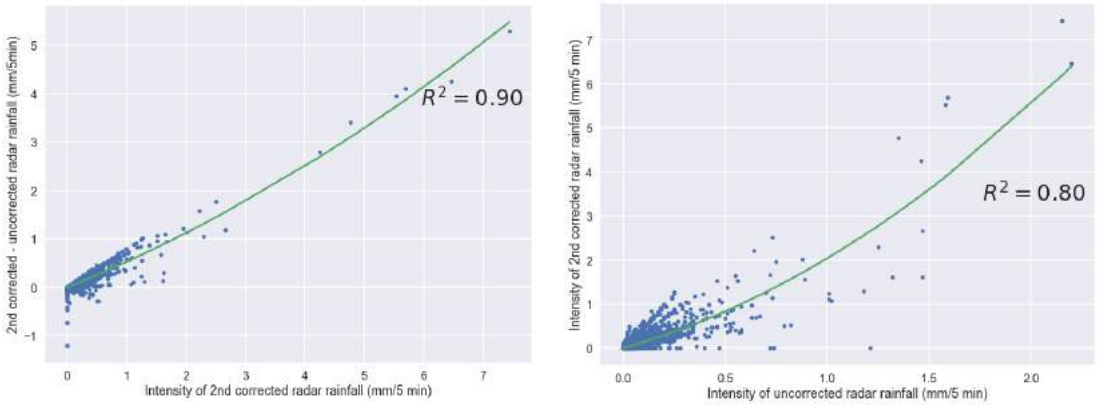


Figure 3.5: Cumulative rainfall over the period of half a year; difference between radar rainfall products at location Tedingerbroek polder. Two KNMI products for the locations of Hoek van Holland (HvH) and Rotterdam are included to have a reference value.

	Name	Value	Based on Source	year	Estimated Uncertainty	Comments
	Areas A_i	Paved: 255.2 ha Greenhouse: 3.3 ha Unpaved 687.3 ha Open water: 75 ha	LGN5	2004	$\mathcal{N}(A_i, A_i \cdot 0.05)$	
	Initial States State_{ini}	Paved: 21744 m^3 Greenhouse: 903 m^3 Unpaved 95090832 m^3 Open water: 0.32 m from peil			$\mathcal{N}(\text{State}_{ini}, \text{State}_{ini} \cdot 0.3)$	The initital states result from a model run.They take lumped errors of forcing, structure and parameters
Paved						
	Maximum stroage on the street	2 mm	RIOKEN (INTWIS)		$\mathcal{N}(2, 0.5)$	Based on commonly used values
	Maximum storage in the sewage	3 mm	RIOKEN (INTWIS)		$\mathcal{N}(3, 0.2)$	Based on commonly used values
	Maximal sewer pump capacity	211 m^3/h	RIOKEN (INTWIS)		$\mathcal{N}(211, 5)$	
Greenhouse						
	Area of greenhouse classes $i = 1 \ldots 10$	See Table A.3	Studie Watervraag Kasgebied Nelen Schuurman	2005	$\mathcal{N}(A_{gh,i}, A_{gh,i} \cdot 0.1)$	outdated information, new legislation and only lowest bound of the classes is taken
	typical depth of storage basins	1 m	Studie Watervraag Kasgebied, Nelen Schuurman	2005	–	Only volume matters
Unpaved						
	Resistance coefficients $i = 1, \ldots 4$	Surface Run-off: 0.3 day Inflow 1300 day Drainage high 70 day Drainage low 200 day	BOEZ09	2011	$\mathcal{N}(R_i, 0.1 \cdot R_i)$	Based on soil type.
	Maximal storage on land	5 mm	BOEZ09	2011	$\mathcal{N}(5, 1)$	
	crop type	urban: miscellaneous (6) rural: Grass nature: Nature(13)	LGN5	2004	–	Not preferable to put noise on the crop type
	soil type	Podzol (loamy, light sand)	Stiboka soil map	1980	–	Relative certain
	Max infiltration	20 mm/h			$\mathcal{N}(20, 3)$	Based on commonly used values of soil type
Open water						
	Max pump capacity	365 m^3/h	Delfland		$\mathcal{N}(340, 20)$	Pumps usually do not live up to their maximum capacity
	Max inlet capacity	365 m^3/h	Delfland		$\mathcal{N}(340, 20)$	" "

RIOKEN (INTWIS): RIOKEN is a data model for sewage information in Holland, and is a module of the integrated waterboard information system (INTWIS) *https://www.riool.net/-/andere-standaardisatie-van-invloed-op-stedelijk-waterbeheer*
LGN5: Landelijk grondgebruik kaar Nederland versie 5; A detailed 25 x 25 m landuse map produced by Alterra. The land is devided into 39 landuse classes, devided over main classes: argraic, forest, water, urban and nature. It is based on satellite images from 2003-2004 (Hazeu, 2005)
Study 'Watervraag Kasgebied t.b.v. de droogte'. In this study by Nelen en Schuurmans, an inventarisation was made on the distribution and sizes of the basins in Delfland.
BOEZ09 'Uitgangspuntennotitie modelinstrument Boezemmodel Delfland (BOEZ09)', a study where the groundwork for the BOS2.0 model was defined(Nelen & Schuurmans and Delfland, 2011)
Stiboka Soil map The Stiboka (stichting voor bodem kartering) map is made by Alterra. It consists of information about the top soil layer in whole Netherlands, differentiating between 3168 soil types. This is simplified using a conversion list of Alterra to 21 soil types, of which Podzol is one (Nelen & Schuurmans and Delfland, 2011).

Table 3.2: Values and estimated distributions for the deterministic parameters presented in Table 2.1



(a) The relation between the difference in radar rainfall products and the intensity of the 2nd radar rainfall.

(b) Relation between uncorrected and 2nd-corrected radar rainfall products.

Figure 3.6: Relationships between radar rainfall products.

uncorrected radar and 2nd-corrected, shown in Figure 3.6a was

$$(P_{unc} - P_{day}) = -0.04P_{day}^2 - 0.48P_{day}, \tag{3.1}$$

with an R^2 of 0.9. This shows that given the 2nd correction, a relative good estimate of the uncorrected radar rainfall product can be given. The other way around, illustrated in Figure 3.6b, the relation between the uncorrected radar rainfall and the corresponding error, is less clear. However, the relation still illustrates the underestimation and is given by

$$P_{day} = 0.45P_{unc}^2 + 1.5P_{unc}, \tag{3.2}$$

with an R^2 of 0.8.

4

How to Solve the Model

An approximate solution to the right problem is worth more than a precise solution to the wrong problem

– Tukey, 1962

Many different methods for data assimilation exist. These are discussed in Section 4.1 and an appropriate method for this system is chosen, which is explained in further detail in Section 4.2. Sections 4.3, 4.4 and 4.5 describe the set-up of the tests that were carried out. The methodology to compare the results objectively is discussed in Section 4.6.

4.1 Data Assimilation Methods

Three main approaches that divide the great number of data assimilation algorithms are (Pedersen et al., 2016):

1. Updating model states. This can be done directly from measurements (e.g. Hansen et al. (2014) and Ho and Lee (2015)), or using more sophisticated methods such as Kalman filtering approaches (e.g. applied by Vrugt et al. (2006) and Borup et al. (2015)).
2. Error correction. Using a more statistical approach, e.g. using time series models or ARMA filters (Liu et al., 2012).
3. Joint state and parameter estimation approaches. Examples also include particle filtering approaches (e.g. Moradkhani et al. (2005)) or evolutionary algorithms (e.g. Dumedah and Coulibaly (2013)).

The main interest in this thesis is updating model states, i.e. the storages in the different reservoirs of the deterministic RR model of Chapter 2. Three possible data assimilation methods include the Ensemble Kalman Filter (EnKF), Particle Filters (PF) and the Moving Horizon Estimate (MHE). These should transform the output observations (e.g. water levels or pump discharges) into an estimate of all states. The main ideas, advantages and disadvantages are discussed below and summarized in Table 4.1.

Moving Horizon Estimate MHE is a state estimation method that was developed to deal with non-linear and constrained dynamic systems (Copp and Hespanha, 2014). It can be applied even if prior distributions are estimated poorly and works well with MPC. However calculation speed is slow, which makes it unattractive for real time updating.

Particle Filters PF is the collective name for sequential Monte Carlo methods, which allows the distribution of the parameters to vary over time (Rawlings and Bakshi, 2006). This is done by sampling, where the samples are called particles. When the PF were first introduced, they ignited a spark in the believe that they could be a robust and accurate solution to solve non-linear dynamic systems. However, this relative good performance is only true if the initial distribution on the uncertainties can be estimated well. For simple CRR-models, this can be difficult making it not the ideal method for this research.

	Ensemble Kalman Filter	Particle Filters	Moving Horizon Estimate
Idea	Ensemble representation of the KF	Sampling	Optimization
Information propagated by the algorithm	Ensemble representation of mean vector and covariance matrix	Complete probability density conditioned on the measurements	Estimated state vector
Prediction of statistics from one measurement time to the next	approximate the posterior distribution of the states using Bayes' rule	MC integration using importance sampling & resampling	Minimizes an estimator cost function based on a finite number of time stages (**?**). The cost function consists of a prediction error and an arrival cost that summarizes past data.
Accuracy of estimation state vector	Performance depends on knowledge of priors	Optimal performance for low dimensional problems. Can be poor for high dimensions because it is limited by real time computer speed	High quality solution for constrained, non-linear systems (Rawlings, 2009).
Computational complexity	relative low, depends on ensemble size	Beats the curse of dimensionality for 'nice' problems. Not necessarily otherwise.	Relative high. Computational cost does not grow as more data becomes available due to fixed time-frame (Copp and Hespanha, 2014)
Advantages	Much applied. Also works for non-linear, non-Gaussian systems. (Copp and Hespanha, 2014)	Can handle constraints & non linearity in state estimation (Rawlings, 2009). Simple to program, fast to execute.	Attractive to use with MPC (Copp and Hespanha, 2014). Can handle constraints & non linearity in state estimation. Recovers robustly from poor prior distributions and un-modelled disturbances (Rawlings, 2009).
Disadvantages	cannot deal well with non-continuous state options	Does not recover from poor guess of initial distribution (Rawlings, 2009).	Requires on-line computation

Table 4.1: Overview of the properties of different DA techniques. Unless stated otherwise, information taken from Daum (2005).

The Ensemble Kalman filter (EnKF) is a filtering method approximating the standard Kalman filter (KF) (Kalman, 1960) and was introduced by Evensen (1994). The EnKF works with an ensemble of vectors that approximates the distribution of the state or input (Katzfuss et al., 2016) every time new data comes available. It works by calculating the distribution using a linear 'shift', where the error of the state space model is assumed

to be linear Gaussian. Since hardly any real-life models are linear Gaussian state-space models, additional approximations are always made.
The EnkF is often applied to cases where the theoretical conditions do not hold. Many examples can be found in geophysical literature where the EnKF is successful and robust for systems to which the assumptions of linearity and Gaussian do not apply (e.g. Evensen, 2003; Katzfuss et al., 2016,) . Using this method on systems where these assumptions do not hold is justified by the principle that *"an approximate solution to the right problem is worth more than a precise solution to the wrong problem"* (Tukey, 1962). This research will embrace the same principle.
The main difference between the Kalman filter and other sequential Monte Carlo methods (e.g. Particle Filters), is that it uses a shift instead of re-weighting when new data becomes available. This shift allows that the method stays very stable, even in high dimensional problems. It is said that for realistic, complex systems, the EnKF is still the only way to do approximate inference, while other techniques can give exact solution but only of simplified versions of the model (Katzfuss et al., 2016).
Weighing the methods against each other gives the EnKF as preferable approach for its robustness and calculation speed. It is an advantage that it is the most common applied DA method, so that if results are applicable, implementation will be accepted more easily. This method will in the next Section be explained in detail and depends on the recursive application of Bayes' rule which shall be explained first.

4.1.1 Bayes' theorem

With the probabilistic model, the likely resulting water levels given a certain causes, such as errors in rainfall or initial states, can be determined. However, the result is observed and the aim is to determine from this the underlying causes. This can be done using Bayes' rule in order to get a better understanding of the system, but also to update the states so the future initial states can be improved.
This means Bayes' theory gives a method of describing the conditional probability of an event A *after* the observed data B becomes available. This is the probability $P(A|B)$, visualized in Figure 4.1b. This probability can be calculated based on two other probabilities which are known or can be estimated beforehand: the conditional probability of the data B given the event A ($P(B|A)$), and the probability of event A occurring in the first place ($P(A)$). This is visualized graphically in Figure 4.1a. Additional reading on probability theory can be found in the MIT open course as taught in 2014 (Orloff and Bloom, 2014). Bayes' theorem is as follows:

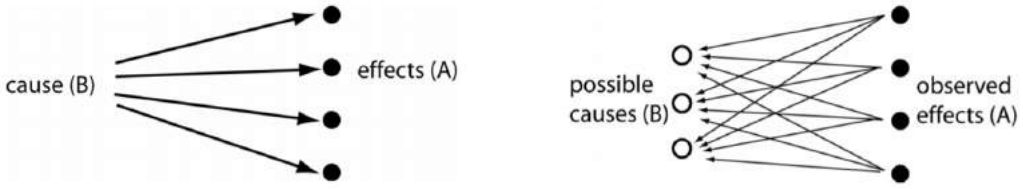
Bayes' theorem Consider multiple events $A_1, \ldots, A_k$, which give a finite, disjunct partition of the sample space $\mathcal{F}$. If B is an event in $\mathcal{F}$ (note that $B \notin \{A_1 \ldots A_k\}$ necessarily), with $P(B) > 0$, then Bayes' theorem gives the following rule:

$$P(A_j|B) = \frac{P(B|A_j)P(A_j)}{\sum_{i=1}^{k} P(B|A_i)P(A_i)} \tag{4.1}$$

For the continuous case:

$$f_{X|Y}(x|y) = \frac{f_{Y|X}(y|x)f_X(x)}{f_Y(y)} = \frac{f_{Y|X}(y|x)f_X(x)}{\int f_{Y|X}(y|x)f_X(x)dx} \tag{4.2}$$

As the research at hand will deal with the continuous case, this is most important and an example of this can be found in Example 2.1 of which the results are shown in Figure 4.2.



(a) Information of the different effects A from a cause B, representing the probability $P(A|B)$ can, together with the probability $P(A)$,...

(b) ... by Bayes theorem be turned into information about the probable cause given observed results A ($P(B|A)$).

Figure 4.1: Graphical representation of Bayes' theorem, giving information about the cause after the effect is observed. Illustration adjusted from (Sivia, 1996).

Figure 4.2: Prior and posterior pdf's of Example 2.1, with prior distribution $\theta \sim \mathcal{N}(2,1)$, $x \sim \mathcal{N}(\theta,1)$ and observation $x = 4$. The posterior is again normally distributed; $\theta \sim \mathcal{N}(3,0.5)$. Note how the mean of the posterior lies in between the prior and the observation and that the variance of the posterior is smaller than that of the prior and the observation distributions.

Example 2.1: **Bayesian Example with continuous data and priors**
For the continuous case, let the notations be given by:

- Observation x
- Prior $f(\theta)d\theta$
- Likelihood $f(x|\theta)dx$
- Posterior $f(\theta|x)d\theta$.

This example looks at the case of water level observations versus modelled water levels. Both contain uncertainties. The goal is to find an improved estimation of the water level from the modelled water level and the observation combined. Let θ be the modelled water level, assumed to have a normal prior distribution $\theta \sim \mathcal{N}(2,1)$. Let x denote the uncertain observed water level, assumed to be related to θ and drawn from the normal distribution $x \sim \mathcal{N}(\theta,1)$.
First the ingredients for the calculation are obtained and then Bayes' rule will be applied to find an improved posterior distribution of θ, given observations x. The prior pdf of θ is (because it is normally distributed) given by

$$f(\theta) = \frac{1}{\sqrt{2\pi}} e^{-(\theta-2)^2/2} \tag{4.3}$$

and the likelihood function of the error in water level given an error θ on the rainfall is:

$$f(x|\theta) = \frac{1}{\sqrt{2\pi}} e^{-(x-\theta)^2/2}. \tag{4.4}$$

The numerator of Bayes' rule is composed of the product of the likelihood and the prior:

$$\begin{aligned} f(\theta) \cdot f(x|\theta) =& \frac{1}{\sqrt{2\pi}} e^{-(\theta-2)^2/2} \cdot \frac{1}{\sqrt{2\pi}} e^{(-x-\theta)^2/2} \\ =& \frac{1}{2\pi} e^{-(2\theta^2-(4+2x)\theta+4+x^2)/2} \\ =& \frac{1}{2\pi} e^{-(\theta^2-(2+1x)\theta)+(4+x^2)/2} \\ =& \frac{1}{2\pi} e^{-(\theta-(1+x/2))^2-(1+x/2)^2+(4+x^2)/2} \\ =& c_1 e^{-(\theta-(1+x/2))^2} \end{aligned}$$

where c_1 is constant for given x and is given by

$$c_1 = \frac{1}{2\pi} e^{-(1+x/2)^2)+(4+x^2)/2}.$$

To calculate the denominator of Bayes' rule, the product of the likelihood and the prior need to be integrated over θ;

$$\begin{aligned} f(x) =& \int_a^b f(x|\theta) f(\theta) d\theta \\ =& c_1 \int_a^b e^{-(\theta-(1+x/2))^2} d\theta \end{aligned}$$

Since the integral for given x will be a constant (say c_2) the posterior can be calculated as

$$\begin{aligned} f(\theta|x)d\theta =& \frac{f(x|\theta) f(\theta) d\theta}{f(x)} \\ =& \frac{c_1 e^{-(\theta-(1+x/2))^2}}{c_2} \end{aligned}$$

From the form of the posterior density function it can be seen that it is again a normal distribution, but with mean $1 + x/2$ and variance $1/2$. The normalization constant can therefore also be calculated as $\frac{1}{\sigma\sqrt{2\pi}}$, hence the final posterior distribution is:

$$f(\theta|x)d\theta = \frac{1}{\sqrt{\pi}} e^{-(\theta-(1+x/2))^2}$$

Now what does this mean? The posterior shows an updated version of the way θ is assumed to distributed. This new distribution has a smaller variance than the prior distribution and will result in a posterior distribution for θ in between the likelihood and the posterior. For the case where $x = 4$ was drawn as observation, the results are shown in Figure 4.2.

4.2 Mathematical Application of the EnKF to the Probabilistic Model

The EnKF is the approximation of the exact method of the Kalman Filter, which calculates the full posterior distributions of the states. This is computationally not be feasible in the small time-slot in real-time systems and is not possible because the linear condition would need to hold, making the EnKF approximations necessary. First the KF is explained in Section 4.2.1, after which the ensemble representation by the EnKF is explained in Section 4.2.2. This is followed by an explanation of the computational steps that are to implement the EnKF.

4.2.1 The Kalman Filter

The main idea of the Kalman Filter is to estimate the states based on the observations collected so far. This is done by calculating the posterior (also called filtering) distribution of the states for each discrete time step k. As the KF theoretically only gives an exact solution for linear-Gaussian state-space models, assume the system can thus be represented by:

$$y_k = H_k x_k + v_k \tag{4.5}$$
$$x_k = M_k x_{k-1} + w_k, \tag{4.6}$$

where the errors are Gaussian distributed, i.e.

$$v_t \sim \mathcal{N}_{m_k}(0, R_k), \text{ the Gaussian distributed observation error with zero-mean}$$
$$w_t \sim \mathcal{N}_{n_k}(0, Q_k), \text{ the Gaussian distributed innovation error with zero-mean}$$

and where

$y_k =$ the vector with observation with m_k entries at time k
$x_k =$ the n-dimensional state vector at time k, including the state of the openwater level
$H_k =$ the observation matrix, linking the state to the observation
$M_k =$ the evolution matrix, linking the previous state to the next

The errors are assumed to be mutually and serially independent. Furthermore, assume the observation- and innovation matrix and the variances $Q_k = Q$ and $R_k = R$ are known and time independent. Let the posterior distribution at time $k-1$ (or equivalently the prior distribution for time k) be

$$x_{k-1}|y_{1:k-1} \sim \mathcal{N}(\widehat{\mu}_{k-1}, \widehat{\Sigma}_{k-1}). \tag{4.7}$$

Notation: The accent $\wedge$ on the mean or variance implies these belong to the posterior distribution of that time-step.
The posterior distribution can now be calculated in two main steps, the *forecast step* (giving the likelihood) and the *update step*.

Forecast step

The aim of the forecast step is to calculate the parameters of the forecast distribution

$$x_k|y_{1:k-1} \sim \mathcal{N}_{n_k}(\widetilde{\mu}_k, \widetilde{\Sigma}_k) \tag{4.8}$$

at time k, denoted by the accent $\sim$. Given the posterior distribution of time $k-1$, this is calculated by propagating the uncertainties through the observation- and evolution model (equation 4.6 and 4.5 respectively), i.e.

$$\widetilde{\mu}_k = M\widehat{\mu}_{k-1}, \tag{4.9}$$
$$\widetilde{\Sigma}_k = M\widehat{\Sigma}_{k-1}M^T + Q. \tag{4.10}$$

Note here that 4.9 and 4.10 only hold so nicely since the errors v_k and w_k are drawn from a Gaussian distribution with zero-mean.

Update step

The update step aims to include the newly observed data y_k and combine this with the forecast step to find the posterior distribution, i.e.

$$x_k|y_{1:k} \sim \mathcal{N}_n(\widehat{\mu}_k, \widehat{\Sigma}_k), \tag{4.11}$$

In order to find the values for $\widehat{\mu}_k$ and $\widehat{\Sigma}_k$, one looks first at the joint distribution of x_k and y_k, given observations $y_{1:k-1}$ and is explained in Text-box 4.1. This gives the following joint distribution:

$$\begin{pmatrix} x_k \\ y_k \end{pmatrix}|y_{1:k-1} \sim \mathcal{N}_{nxm}\left(\begin{pmatrix} \widetilde{\mu}_k \\ H\widetilde{\mu}_k \end{pmatrix}, \begin{pmatrix} \widetilde{\Sigma}_k & \widetilde{\Sigma}_k H^T \\ H\widetilde{\Sigma}_k & H\widetilde{\Sigma}H^T + R \end{pmatrix}\right) \tag{4.12}$$

Note that in order to derive 4.12, v_k and w_k have to be mutually and serially independent, which was conveniently (and for water systems often reasonably) assumed. The mean of the distribution of y_k is found by propagating the mean of the state in the observation equation, where this results in $H\widetilde{\mu}_k$ only because the noise on v is zero-mean Gaussian. From the properties of the multivariate normal distribution and Bayes' rule, one can derive that the filtering distribution at time k is defined by

$$\widehat{\mu}_k = \widetilde{\mu}_k + K_k(y_k - H\widetilde{\mu}_k) \tag{4.13}$$
$$\widehat{\Sigma}_k = (I_n - K_k H)\widetilde{\Sigma}_k, \tag{4.14}$$

where K is the $n \times m$ Kalman-gain matrix defined by

$$K_k = \widetilde{\Sigma}_k H^T (H\widetilde{\Sigma}_k H^T + R)^{-1}. \tag{4.15}$$

In words equation 4.13 means that the posterior distribution is a weighted average of the forecast mean $\widetilde{\mu}$ and the observations at time k, where the weighing occurs based on the forecast variance $\widetilde{\Sigma}_k$, the observation matrix H and the observation variance matrix R. The filtered covariance matrix $\widehat{\Sigma}_k$ depends on the forecast variance from the previous time and the data distribution.

***Text-box 4.1:* Determining the joint distribution of x_k and y_k given observations $y_{1:k-1}$.**

The joint distribution of x and y given the observations at times $k = 1 \dots k-1$ is given by:

$$\begin{pmatrix} x_k \\ y_k \end{pmatrix}|y_{1:k-1} \sim \mathcal{N}_{nxm}\left(\begin{pmatrix} \mathbb{E}x_k|_{y_{1:k-1}} \\ \mathbb{E}y_k|_{y_{1:k-1}} \end{pmatrix}, \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}\right) \tag{4.16}$$

The expectation

The expected (mean) values can be calculated using the linear properties of the expectation, the known forecast distribution 4.8 and the fact that the observation and innovation error are zero-mean Gaussian distributed (hence there expectation is zero). The derivation of $\mathbb{E}x_k|_{y_{1:k-1}}$ follows:

$$\begin{aligned}\mathbb{E}x_k|_{y_{1:k-1}} &= \mathbb{E}[M(x_{k-1}|_{y_{1:k-1}}) + w_k] && (4.17)\\ &= M\mathbb{E}[x_{k-1}|_{y_{1:k-1}}] + \mathbb{E}w_k && (4.18)\\ &= M\widehat{\mu}_{k-1} + 0 && (4.19)\\ &= \widetilde{\mu}_k && (4.20)\end{aligned}$$

The expectancy $\mathbb{E}y_k|_{y_{1:k-1}}$ uses similar properties, but is now propagated trough 4.5.

The covariance matrix

The covariance matrix consists of four sub-matrices, where Σ_{11} and Σ_{22} denote the variance of $x_t|_{y_{1:k-1}}$ and $y_k|_{y_{1:k-1}}$ respectively and Σ_{12} and Σ_{21} denote the covariance between the two. For the calculation, remember that if A is a matrix, $\text{Var}(AX) = A(\text{Var}X)A^T = A\mathbb{E}[X - \mathbb{E}(X)]^2A^T$. Now:

$$\begin{aligned}\Sigma_{11} &= \text{Var}(x_k|_{y_{1:k-1}}) && (4.21)\\ &= \text{Var}(Mx_{k-1}|_{y_{1:k-1}}) + \text{Var}(w_k) && (4.22)\\ &= M\widehat{\Sigma}_{k-1}M^T + Q_t && (4.23)\\ &= \widetilde{\Sigma}_k && (4.24)\end{aligned}$$

The variance matrix Σ_{22} follows in a similar fashion. For the covariance matrix Σ_{12} (and Σ_{21}):

$$\begin{aligned}\Sigma_{12} &= \text{Cov}(x_k|_{y_{1:k-1}}, y_k|_{y_{1:k-1}}) && (4.25)\\ &= \mathbb{E}[x_k|_{y_{1:k-1}} \cdot (y_k|_{y_{1:k-1}})^T] - \mathbb{E}[x_k|_{y_{1:k-1}}](\mathbb{E}[y_k|_{y_{1:k-1}}])^T && (4.26)\\ &= \mathbb{E}[(x_k \cdot (Hx_k + v)^T)|_{y_{1:k-1}}] - \widetilde{\mu}_k(H\widetilde{\mu}_k)^T && (4.27)\\ &= \mathbb{E}[x_k x_k^T H^T|_{y_{1:k-1}}] + \mathbb{E}[x_k v^T|_{y_{1:k-1}}] - \widetilde{\mu}_k\widetilde{\mu}_k^T H^T && (4.28)\\ &= \mathbb{E}[x_k x_k^T|_{y_{1:k-1}}]H^T + 0 - \widetilde{\mu}_k\widetilde{\mu}_k^T H^T && (4.29)\end{aligned}$$

where $\mathbb{E}[x_k v^T|_{y_{1:k-1}}] = 0$ because v is Gaussian distributed with $\mathbb{E}[v] = 0$. The trick to continue is to make a substitution for $\mathbb{E}[x_k x_k^T|_{y_{1:k-1}}]$, as for this expectancy nothing can be said. Remember:

$$\text{Var}(x_k|_{y_{1:k-1}}) = \mathbb{E}[x_k x_k^T|_{y_{1:k-1}}] - \mathbb{E}[x_k|_{y_{1:k-1}}]\mathbb{E}[x_k|_{y_{1:k-1}}]^T. \quad (4.30)$$

Substituting gives:

$$\begin{aligned}\Sigma_{12} &= (\text{Var}(x_k|_{y_{1:k-1}}) + \mathbb{E}[x_k|_{y_{1:k-1}}]\mathbb{E}[x_k|_{y_{1:k-1}}]^T)H^T - \widetilde{\mu}_k\widetilde{\mu}_k^T H^T && (4.31)\\ &= (\widetilde{\Sigma} + \widetilde{\mu}\widetilde{\mu}^T)H^T - \widetilde{\mu}_k\widetilde{\mu}_k^T H^T && (4.32)\\ &= \widetilde{\Sigma}_k H^T && (4.33)\end{aligned}$$

The case of Σ_{21} follows similarly, where the matrix H^T ends up on the left of $\widetilde{\Sigma}_k$. Hence the joint distribution of 4.12 is obtained. □

4.2.2 The Ensemble Kalman Filter

The EnKF uses a range of possibilities (the ensemble) to represent the prior, forecast and posterior distributions. The ensemble members are propagated forward in time as soon as new data becomes available, giving again a spread of discrete values instead of a complete distribution. Since the parameters of the posterior distribution no longer need to be calculated nor stored, calculation time is reduced. This can be viewed as a form of dimension reduction (Katzfuss et al., 2016), that ensures the algorithm stays computationally feasible.
The same steps as in the KF prevail; a forecast and an update step. Two different EnKF branches exist for the update step; stochastically or deterministic. Since the focus in this research is on uncertainties, the choice for stochastic updates comes naturally. The steps can then be carried out as follows:

Start with an initial ensemble $\widehat{x}_0^{(1)}, \ldots, \widehat{x}_0^{(N)}$, representing the prior distribution. Here N denotes the number of ensemble members. Iteratively from time step $k-1$ to k, the ensemble is updated using the KF approach as follows:

Forecast Step

The ensemble $\widehat{x}_{k-1}^{(1)}, \ldots, \widehat{x}_{k-1}^{(N)}$ represents a sample from the theoretical posterior distribution of the KF given in 4.7 at time $k-1$. In Bayesian terms, this is the sampled prior distribution for time k. Now for each ensemble member i, a noise component $w_k^{(i)}$ is randomly drawn from $w_k^{(i)} \sim \mathcal{N}_n(0, Q)$. The sample of the forecast distribution is then calculated as:

$$\widetilde{x}_k^{(i)} = M\widehat{x}_{k-1}^{(i)} + w_k^{(i)}, \tag{4.34}$$

for $i = 1 \ldots N$. In its turn the forecast ensemble samples the forecast distribution given by the KF in equation 4.8, i.e.

$$\widetilde{x}_k^{(i)} \sim \mathcal{N}(\widetilde{\mu}_k, \widetilde{\Sigma}_k). \tag{4.35}$$

This is a representation for the likelihood. The forecast step is now reduced to simply Monte Carlo sampling, such that the EnKF can also be applied to non-linear models. However, when the model is non-linear this would imply that the ensemble will not represent the normal distribution of the forecast by the KF any more.

(Stochastic) Update Step

When a new observation y_k comes available, the update step provides a method to incorporate this knowledge. The aim of this step is again to find a sample of $\widehat{x}_k^{(1)}, \ldots, \widehat{x}_k^{(N)}$ of the posterior distribution such that the distribution corresponds to the posterior distribution of the original KF, i.e.

$$\widehat{x}_k^{(i)} \sim \mathcal{N}_n(\widehat{\mu}_k, \widehat{\Sigma}_k). \tag{4.36}$$

In order to find this, the forecast states are propagated in the observation matrix H with observation error $v \sim \mathcal{N}(0, R)$ such that the set of simulated observations becomes:

$$\widetilde{y}_k^{(i)} = H\widetilde{x}_k^{(i)} - v_k^{(i)}, \tag{4.37}$$

for $i = 1 \dots N$. Using the Kalman-gain factor K (equation 4.15) as a weighing factor between uncertainties in the observations and the simulated observations, the filtered state becomes

$$\widehat{x}_k^{(i)} = \widetilde{x}_k^{(i)} + K_k(y_k - \widetilde{y}_k^{(i)}). \tag{4.38}$$

This leaves two things. The first is to show that filtered states of 4.38 sample the distribution from 4.36. This is shown in Text-box 4.2 . The second is to find a method to estimate the Kalman Gain K, since the solution offered by the KF would require full computation of the forecast covariance matrix, while the EnKF avoids this to reduce calculation time. The determination of K is explained in Section 4.2.4 where the computational steps are shown.

Text-box 4.2: The updataed EnKF ensemble represents the posterior distribution of the KF To check this, the expectancy and the variance need to be calculate. The first follows easily:

$$\begin{aligned}
\mathbb{E}\widehat{x}_k^{(i)} &= \mathbb{E}\widetilde{x}_k^{(i)} + K(y_k - \mathbb{E}[\widetilde{y}_k^{(i)})]) \\
&= \widetilde{\mu}_k + K_k(y_k - H\widetilde{\mu}_k) \\
&= \widehat{\mu}_k
\end{aligned}$$

The variance is calculated as

$$\begin{aligned}
\mathrm{Var}(\widehat{x}_k^{(i)}) =& \mathrm{Var}(\widetilde{x}_k^{(i)}) + \mathrm{Var}(K_t\widetilde{y}_k^{(i)}) \\
& - 2\mathrm{Cov}(\widetilde{x}_k^{(i)}, K_k\widetilde{y}_k^{(i)}) \\
=& \widetilde{\Sigma}_k + K_k H\widetilde{\Sigma}_k \quad \text{(See calculation } \Sigma_{21} \text{ in textbox 1)} \\
=& \widehat{\Sigma}_k
\end{aligned}$$

Hence the ensemble samples the posterior distribution 4.11 of the KF. □

4.2.3 Assumptions of the EnKF that are not met

In the case of the probabilistic polder model described in Chapter 2, some of the assumptions of the theoretical explanation above do not apply. Even though the EnKF has proven to be very robust concerning deviations from these requirements, it seems appropriate to mention the 'failed' properties:

1. **Linearity of the system** The probabilistic model of Chapter 2 is non-linear due to for example threshold behaviour in the reservoir system of the CRR-model, delay coefficients for runoff over paved areas, pump settings and due to the interaction between the shallow groundwater and the open water.
2. **Additive noise** From the analysis of the precipitation error it is clear that a multiplier for the error on the rainfall would make more sense than the mandated additive noise by the EnKF. Whether it rains or not can be determined well and the higher intensity of precipitation, the bigger the correction needed. The observation error v_t remains additive, the noise on the intitial states will be tested as both multiplicative and additive. This deviation from the requirements is not expected to cause too large problems.
3. **Gaussian distributions on the noise** The noise in this research will be put on the state and observation equation, i.e. the error associated with the modelled

transition from the previous state to the next and from the state to the observation. These error distributions were defined in Chapter 3. As the priors were difficult to estimate for the parameters, these were conveniently assumed to be normal. For the precipitation error, more information was available. The estimated relation between uncorrected and 2nd correction of the radar rainfall product was the relation $y = 0.45x^2 + 1.5x$ given in equation 3.2, hence not normally distributed. This is adjusted to suit the requisite of normality. The normal distribution of the rainfall multiplier r_k representing the error in rainfall is determined by solving the multiplier needed to let $r_k P_{unc,k} = P_{2nd_k}$. For these values of r_k for all time steps k where the uncorrected precipitation measurement was larger than zero (4870 points), the mean and variance were deduced . The normal distribution for the multiplicative error on the rainfall (r_k) then becomes:

$$w_k = \mathcal{N}(1.32, 1.4). \tag{4.39}$$

4. **Zero-mean noise** When working with a multiplier, the mean has to be 1 instead of 0 as required by the EnKF. Since the error becomes multiplicative, this means that the 'nice' properties that $\widetilde{\mu}_t = M\widehat{\mu}_{t-1}$ and $\widetilde{\Sigma}_t = M\widehat{\Sigma}_{t-1}M^T + Q$ in the forecast step still holds if the mean is indeed 1. Therefore the normal distribution of the rainfall is adjusted to

$$w_k = \mathcal{N}(1, 1.4). \tag{4.40}$$

If it becomes clear from the test runs that this prior cannot live up to the day-corrected rainfall, the standard deviation can be enlarged to compensate for the reduced mean.

4.2.4 Computational steps for applying the EnKF to the probabilistic model

Let $i = 1, \dots N$ the ensemble members, k the discrete time-step and $j = 1 \dots m$ the different states (where $m = 16$ in this case), so $x_j^i(k)$ is the jth state of ensemble member i at time-step k. Since the only available observation is the water level, vector H of the observation equation 4.5 needs to map the state vector x to the modelled water level state (for this explanation let state $j = m = 16$ represent the observable state). Thus, the observation equation becomes:

$$y(k) = \begin{bmatrix} 0 & \dots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_m(k) \end{bmatrix} + v(k) \tag{4.41}$$

where $x(k) \in \mathbb{R}^{mx1}$, $H \in \mathbb{R}^{1xm}$ and $v(k)$ and $y(k) \in \mathbb{R}$.
Computationally, the forecast and update step are calculated as follows:

Forecast-step For each ensemble member j at time step k, noise $\eta_j(k)$ is drawn and added or multiplied to the respective component (e.g. the rainfall and water level observation). The forecast state vector $\widetilde{x}^i(k)$ for each ensemble member i is then calculated by running the eterministic CRR model with the perturbed components (e.g. rainfall or states) for that ensemble member:

$$\widetilde{x}^i(k) = \mathcal{M}(\widehat{x}^i(k-1), \eta^i(k)). \tag{4.42}$$

Let $\widetilde{\xi}$ denote the average of the ensemble members, then the forecast expectancy $\mathbb{E}^f$ is given by

$$\mathbb{E}^f(k) = \begin{bmatrix} \widetilde{x}^1 - \widetilde{\xi} & \widetilde{x}^2 - \widetilde{\xi} & \dots & \widetilde{x}^N - \widetilde{\xi} \end{bmatrix} \in \mathbb{R}^{mxN} \tag{4.43}$$

Update step The forecast variance $\widetilde{\Sigma}$, the numerical version of equation 4.10, is calculated as

$$\widetilde{\Sigma} = \frac{1}{N}\mathbb{E}^f(k)\mathbb{E}^f(k)^T \in \mathbb{R}^{mxm} \tag{4.44}$$

$$= \frac{1}{N}\begin{bmatrix} \sum_{i=1}^N(\widetilde{x}_1^i - \widetilde{\xi}_1)^2 & \sum_{i=1}^N(\widetilde{x}_1^i - \xi_1)(\widetilde{x}_2^i - \widetilde{\xi}_2) & \dots & \sum_{i=1}^N(\widetilde{x}_1^i - \xi_1)(\widetilde{x}_m^i - \widetilde{\xi}_m) \\ \sum_{i=1}^N(\widetilde{x}_2^i - \xi_2)(\widetilde{x}_1^i - \widetilde{\xi}_1) & \sum_{i=1}^N(\widetilde{x}_2^i - \widetilde{\xi}_2)^2 & \dots & \sum_{i=1}^N(\widetilde{x}_2^i - \widetilde{\xi}_2)(\widetilde{x}_m^i - \widetilde{\xi}_m) \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^N(\widetilde{x}_m^i - \widetilde{\xi}_m)(\widetilde{x}_1^i - \widetilde{\xi}_1) & \sum_{i=1}^N(\widetilde{x}_m^i - \widetilde{\xi}_m)(\widetilde{x}_2^i - \widetilde{\xi}_2) & \dots & \sum_{i=1}^N(\widetilde{x}_m^i - \widetilde{\xi}_m)^2 \end{bmatrix} \tag{4.45}$$

The Kalman Gain K as in equation 4.38 can now be calculated as

$$K = \widetilde{\Sigma}H^T[H\widetilde{\Sigma}H^T + \sigma_{obs}^2]^{-1} \tag{4.46}$$

$$= \widetilde{\Sigma}H^T\left(\begin{bmatrix}0 & \dots 0 & 1\end{bmatrix}\widetilde{\Sigma}\begin{bmatrix}0\\ \vdots \\ 0\\ 1\end{bmatrix} + \sigma_{obs}^2\right)^{-1} \tag{4.47}$$

$$= \widetilde{\Sigma}H^T\left(\frac{\sum_{i=1}^N(\widetilde{x}_m^i - \widetilde{\xi}_m)^2}{N} + \sigma_{obs}^2\right)^{-1} \tag{4.48}$$

$$= \frac{1}{N}\begin{bmatrix}\sum_{i=1}^N(\widetilde{x}_1^i - \widetilde{\xi}_1)(\widetilde{x}_m^i - \widetilde{\xi}_m) \\ \sum_{i=1}^N(\widetilde{x}_2^i - \widetilde{\xi}_2)(\widetilde{x}_m^i - \widetilde{\xi}_m) \\ \vdots \\ \sum_{i=1}^N(\widetilde{x}_m^i - \widetilde{\xi}_m)^2.\end{bmatrix}\left(\sigma_m^2 + \sigma_{obs}^2\right)^{-1} \tag{4.49}$$

$$= \frac{\sigma_m^2}{\left(\sigma_m^2 + \sigma_{obs}^2\right)}\begin{bmatrix}\sum_{i=1}^N(\widetilde{x}_1^i - \widetilde{\xi}_1)(\widetilde{x}_m^i - \widetilde{\xi}_m)/\sigma_m^2 \\ \vdots \\ \sum_{i=1}^N(\widetilde{x}_{m-1}^i - \widetilde{\xi}_{m-1})(\widetilde{x}_m^i - \widetilde{\xi}_m)/\sigma_m^2 \\ 1\end{bmatrix} \tag{4.50}$$

Equation 4.49 shows that the 'gain' for each state depends on the sampled covariance with state m, representing the open water watel. The variance in the observations determines the magnitude of the 'pull' provided by K. Finally, the posterior state of the ensemble members is calculated as

$$\widehat{x}_i = \widetilde{x}_i + K(y_{obs} - x_m + v) \tag{4.51}$$

4.3 Sensitivity analysis

Before deciding which part of the model components can best be improved using data assimilation, it is good to know which factors influence the model outcome most. The states are conditionally depended on the forcing, the initial states and the parameters. As there are many parameters, coupled to their own land-uses, it was chosen to first focus on which land-uses the model is most sensitive to. To test the influence of the initial states and the forcing, Sobek is run for a very short time period multiple times, with each time a small deviation from the 'normal' run in one of the components.

The settings Sobek is run for 15 minutes from 04:15 to 04:30 on the 23nd of June 2016. The pumps were removed from Sobek to prevent confusing results due to turning on and off of pumps. The short timespan is chosen because this corresponds to the time step which will be used as assimilation step. This means the uncertainties in OpenDA and the comparison with the observed data will be applied every 15 minutes as well. Therefore this time step gives the most important method of determining the sensitivity relative for this case. If a longer run period would be taken, the influence of initial states would reduce.
The specific period is chosen because it covers the start of the rainfall event already discussed in Chapter 3 and the initial states as specified in Table 3.2. This means non of the system states are empty , allowing the different initial states to be calculated with some form of multiplicative factor. The factor is taken as $\sigma = 0.3 \cdot S_{i,ini}$ for the states and $\sigma = 1.4$ for the precipitation as was found to be the rainfall distribution in Section 3.3. Sobek is then run for the cases where the initial states are perturbed per land-use class with $S_{i,ini} + \sigma$, $S_{i,ini} + 2\sigma$, $S_{i,ini} - \sigma$ and $S_{i,ini} - 2\sigma$. The same is done for the rainfall. Note: The only forcing data that is chosen for this analysis is the precipitation, since the evaporation is known not to make a huge impact on such a short time-scale.

Evaluation of the results The results are compared with the 'zero' run where no deviations in settings or forcing data were made. Since the areas are taken from the case study, these might not give an honest representation for the sensitivity of different Sobek parts for general polder models. Therefore also the relative influence is looked into by calculating the produced water level difference per hectare of area.

4.4 Experiment 1: Twin Experiment

In order to test the effectiveness of the EnKF updates, a twin experiment is executed. The test consists of two parts, a control run and a data assimilation run, which are depicted in Figure 4.3.

1. The model-run generating the synthetic data is called the **control run** (Zhang et al., 2014). This creates the *'true'* states and outputs. To implement the control run, a fixed set of parameters, initial states and forcing data is used as input in the model and the resulting water level is called the control (or 'true') water level. The assimilation is said to work well if the updated water level predictions approximate the control water level.
2. The **data assimilation run** that follows the control run uses a set of *a priori* input (precipitation) and states which are perturbed with noise drawn from a mean-zero Gaussian distribution. The *'actual'* observation (the water level) is assimilated after adding random mean-zero Gaussian errors to the 'true' water level (Nie et al., 2011). The DA run uses the *same* model as in the control run.

The advantage of a twin experiment is that it gives a method to test and compare the effectiveness of a new data assimilation approach, because it limits other sources of error such as model structure. These real-life uncertainties can make it impossible to see the potential of these methods. This immediately leads to the 'but' of the twin experiment: it will give only an idea of how well the DA method can work in ideal situations, where one has the perfect parameters and model structure. Hence, the twin experiment will give a very optimistic view on the applicability of the DA method. To test the effectiveness on a real-life system, the best working DA approach from the twin experiment is applied to the case study set out in Chapter 3.

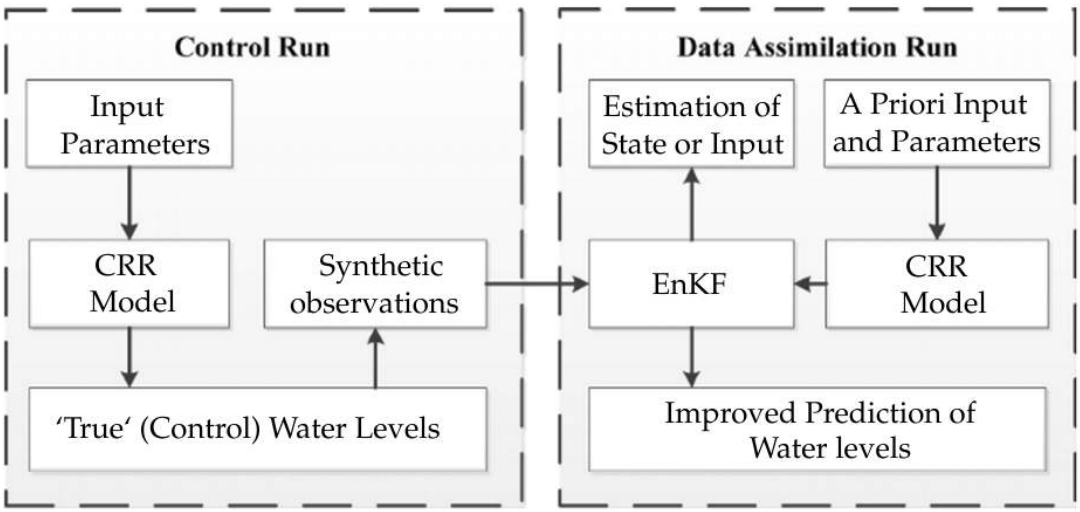


Figure 4.3: Schematic diagram of a twin experiment, comprised of a control run and data assimilation run, based on the paper by Zhang et al. (2014).

4.4.1 The Set-Up

Four DA approaches are compared in different twin experiments. These twin experiments have the same control run, but different DA runs. The DA implementations differ by implementing perturbations on different model components:

1. **Multiplicative noise on the Rainfall** In this approach, the distribution of the ensemble is generated by applying a multiplication factor on rainfall as only noise before generating the resulting states with the CRR model. The first advantage of this approach is that the root of the uncertainty is well defined such that one keeps a grip on what the assimilation is correcting for. Second, the rainfall is known to be a very uncertain product (see Chapter 3) and the prior distribution can be estimated well due to many measurements available. Disadvantages are that the assimilation can only perturb the rainfall when it is greater or equal to zero. Furthermore, errors in other model components are ascribed to the rainfall as well.
2. **Additive noise on all states** This approach adds an independently drawn error of the prior distribution for each state to the initial state of that assimilation time step. This has as advantage that the states can be updated separately, such that all sorts of model and forcing errors can be taken into account. The disadvantage of it is that is will be unclear which root causes of the uncertainty are being corrected for. The second disadvantage is that without rain, the reservoirs with threshold processes (all states except for the states connected to the unpaved-land use) are not directly connected the water level. For example, the storages in the greenhouse basin only overflow if the maximum storage is reached; any adjustment of the storage below this level does not influence the observation of that time. The distributions on the initial states are taken as defined in 3.2: a zero-Gaussian distribution with a variance of 5 % of the initial states.
3. **Additive noise on the groundwater state** The groundwater state is the only state that is also connected to the water level when no rain occurs, making it the unpaved area the only land-use class that can be assimilated separately. The advantage is that the correction is again more pinpointed and can also be applied during dry spells. The disadvantage is that it is a slow responding land-use, such that it could

be difficult to assimilate for peak flows, which are most important to the controller. Furthermore, all other model errors are ascribed to the errors in groundwater modelling. The additive noise v_k is drawn from $v_k \sim \mathcal{N}(0, 2000\ \mathrm{m}^3)$ (which is approximately one percent of the average volume in the groundwater reservoir).

4. **One multiplication factor on all states** This is a method that aims at ascribing the main prediction error to the rainfall, but in an indirect method. The reasoning behind this is that one multiplication factor for all states can represent the dry or wetness of the system, but also represent a part of the model error. The advantage over a rainfall multiplier could be that now corrections can also be applied without rainfall, for example if their is a time lag between the modelled and observed response. The factor has a chosen prior of $r_{states} = \mathcal{N}(1, 0.1)$.

The time span of the twin experiment covers 1.5 day, from 22-06-2016 06.00 to 23-06-2016 22.00. This period was chosen because it contains an extreme storm, a period of interest for the controller. The uncorrected rainfall data is used as a priori input for the data assimilation run. The forcing input for the control run is a synthetic rainfall event created by perturbing the uncorrected rainfall randomly with noise drawn from the normal distribution $\mathcal{N}(1.4, 1.32)$. The time-step of the CRR model is five minutes, corresponding to the frequency of the forcing data.
Before comparing the different DA approaches, the DA method with the rainfall multiplier is used to determine the best settings for ensemble size and time increment. The different settings are compared by the ability of the DA run to approach the control run within short computational times. Furthermore the influence of the pumps on the assimilation is tested.

4.5 Experiment 2: Case study

For the case study, the same deterministic model structure is used as in the Twin Experiment. The uncorrected rainfall is used as the a priori input for the model. The water level observations are now changed from synthetic values to actual observations supplied by Delfland.
The main difference between the twin experiment and the experiment for the case study is the lack of validation data to compare the update of the states other than the water level to. The interpretation of the results is therefore less objective and will need to be done by qualitatively looking at the patterns of the hidden states.

4.6 Comparing results: The probabilistic Nash Sutcliffe

The evaluation of results is mainly visual, but to objectively rank the success of the forecasts, a score measure is applied. The score measures are the Nash Sutcliffe (NS) and the probabilistic Nash Sutcliffe (NS_p), following the approach of Bulygina et al. (2011). The NS is a commonly applied model efficiency coefficient in hydrological models and is for state x_j defined as

$$NS = 1 - \frac{\sum_{t=1}^{T}(x_m^j(t) - x_{obs}^j(t))^2}{\sum_{t=1}^{T}(x_{obs}^j(t) - \overline{x}_{obs}^j)^2} \tag{4.52}$$

where

$$x_m^j(t) = \text{ the modelled state } x_j \text{ at time t} \tag{4.53}$$

$$x_{obs}^j(t) = \text{ the value of the observation or control value of state } x_j \text{ at time t} \tag{4.54}$$

$$\overline{x}_{obs}^j = \text{ The average of the control values of the state } x_j \text{ over } t = 1 \dots T. \tag{4.55}$$

The value ranges between $-\infty$ and 1, where values close to one show modelled and observed values close to each other, while low values show little relation between both. The coefficient is sensitive to extremes.
The probabilistic Nash Sutcliffe gives an extension of the regular Nash Sutcliffe value of the average modelled values $\overline{x}_m^j$ by taking into account the variance of the ensemble. The NS_p is defined as:

$$NS_p = NS - \frac{\sum_{t=1}^{T} \text{Var}(x_m^j(t))}{\sum_{t=1}^{T} (x_{obs}^j(t) - \overline{x}_{obs}^j(t)^2} \tag{4.56}$$

where $\text{Var}(x_m^j(t))$ is the variance of the ensemble for the modelled state x^j.

5

Software

Every program has at least one bug and can be shortened by at least one instruction - from which, by induction, one can deduce that every program can be reduced to one instruction which doesn't work.
– Anonymous

5.1 SobekRR

The program used to implement the conceptual rainfall-runoff polder model is Sobek RR. Sobek is a software package that integrates the different aspects of the water management systems; river, urban and rural areas, by linking rivers, canals and sewage systems (Prinsen, 2013a). The schematic representation of Tedingerbroekpolder, the case study for this research, in Sobek RR can be seen in Figure 5.1. Given forcing data and initial state vector x_0, the program was used to calculate the water levels and the corresponding pump discharge to the boezem deterministically. The different processes and reservoirs used for this research were explained in Appendix A.

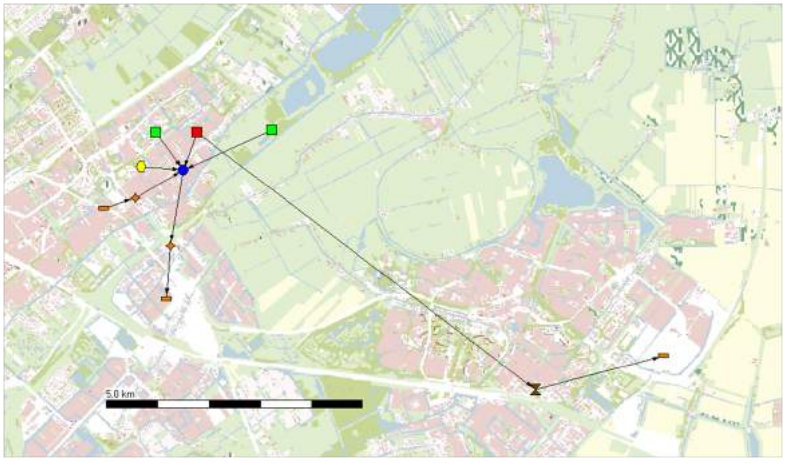


Figure 5.1: Visualization of Tedingerbroekpolder as seen in the Sobek RR interface . The green squares are the unpaved nodes. The yellow, red and blue the greenhouse, urban and open water areas respectively. The orange rectangles are the model boundary, the stars the inlet and outlet pumps and the hourglass the WWTP.

5.2 OpenDA

OpenDA is a software package developed by Deltares, VORtech and TU Delft, and is used for data assimilation and calibration for numerical models (OpenDA-Association, 2016). It is open source and was first launched in 2010. Its toolbox comprises different DA methods, including the EnKF, the ensemble square root Kalman filter (EnSR), steady

state Kalman filter, the particle filter and 3DVar. These methods can be applied to different numerical models, but the coupling needs to be prepared for each model separately and is available for certain models.

5.2.1 OpenDA interfaces

Different components of OpenDA work together as indicated by their 'interfaces'. For the model, these interfaces consist of a *model interface* and a *stochastic model interface*. The objective of the *model interface* is to define the functionalities that a coupled deterministic model (in this case Sobek RR) should implement, e.g. the calculation horizon and the list of items that can be provided as input to the model or that can be retreived as output from the model. The *stochastic model interface* is an extension of the *model interface* that makes the model a probabilistic model. The OpenDA algorithm use the *stochastic model interface* to access and manipulate the model.
The exterior model (Sobek RR) needs to be extended with these interfaces, which is called *'wrapping the model'*. For quite some models the wrappers already exist, but not for Sobek RR. This means a wrapper needed to be created, which was done as a *black-box wrapper*. The advantage of this is that it requires in principle no alterations in the model code. As Sobek is not a model in which code can easily be changed this was beneficial. For implementing a black black box wrapper, OpenDA offers utilities that reduce the actual implemention to:

- Coding to the implementation of the so called *IDataObject* interface for those SobekRR input files and output files that needs to be adjusted by OpenDA. The *data object* exposes the various items in these files as *input exchange items* and *output exchange items*.
- Setting up a *black box configuration* to specify on which input exchange items noise has to be added, which exchanges items are part of the model state that is adjusted by OpenDA, and which output exchange items should be used to compare the model results with the observations.

From the *black box configuration* the OpenDA *black box utilities* instantiate a stochastic model, that can be accessed by the algorithm.
A visualization of the elements in a black-box wrapper are given in Figure 5.2.

5.2.2 OpenDA configuration with Sobek RR

The configuration for the black-box set up of Figure 5.2 is specified in three different categories, the model configuration, the algorithm configuration and the stochastic observer configuration. Each configuration branch is specified in the main file, called 'RRDelfland.oda'. The different properties of the main configuration files are explained below.

Model configuration The model configuration consists of three main files, from inner to outer layer:

1. **The WrapperConfig file** Is the file 'closest' to the model executable, or the most inner layer of the configuration. It consists of different parts. The *run* section starts the initialActions that are carried out to start the OpenDA run. This includes the cloning of a template directory containing the information on the Sobek model for each ensemble member. The clones are called *instance* 1, *instance* 2,... and *instance* 0 which is approximately the average of the other instances. The next section

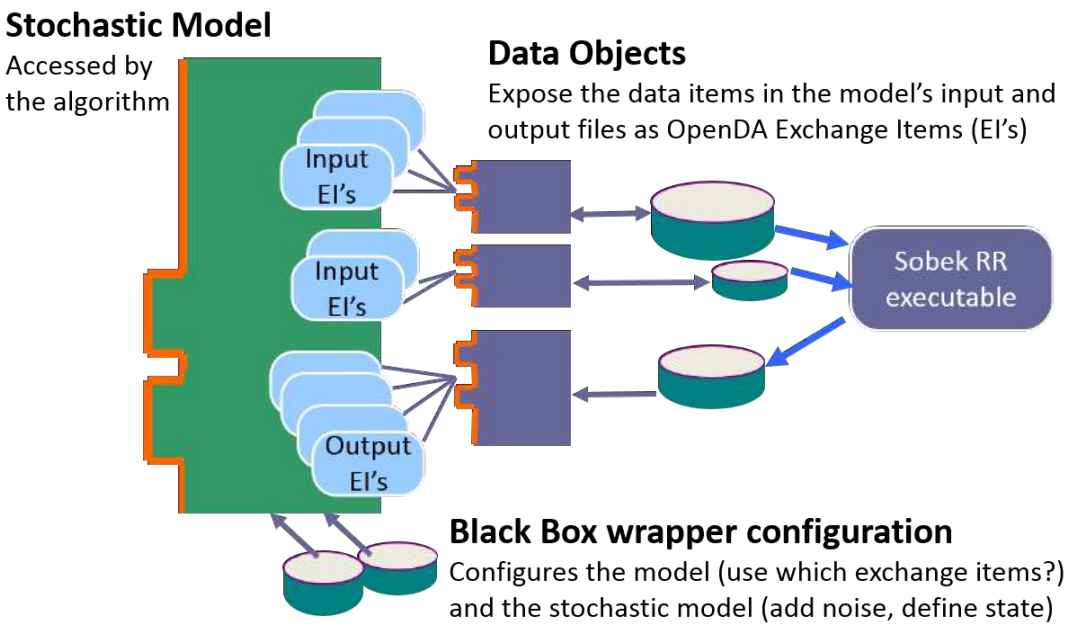


Figure 5.2: Schematization of the black-box model wrapper. A configuration specifies how the input and output items in the numerical model (Sobek RR) were extended to a stochastic model which can be used for DA. .

is the *computeActions* section which defines the model executables (the run.bat) and the stacking of the output.his files from Sobek for the consecutive restarts. The last section, *inputOutput* is the part where the input and output data objects are defined. This is the part which was most time-consuming to set up in the blackbox wrapper because it requires the coding of java classes that implement the OpenDA *data object interface*. These input and output data objects are called **ioObjects** and include an ascii- restart file for Sobek, a rainfall input file (.BUI) for Sobek, the setting for Sobek (.INI) and the output that needs to be compared to the observation (in this case the waterlevel of the open water nodes in Sobek). The ioObjects are transferred back and forth each assimilation step for each ensemble member and adjusted. The java code that specifies the different actions were written for this research and can be found in Appendix B. The code for reading out the precipitation file of Sobek and reading in the perturbed rainfall data created by OpenDA is given in Appendix B.1. The code storing and transferring the restart information of the states is given in Appendix B.2 and the code to read out and change the time specifications of the runtime in Sobek each assimilation run is given in Appendix B.3

2. **The ModelConfig file** The ModelConfig file is in the middle of the model configuration. It gets called by the StochModelConfig file and starts with referring to the WrapperConfig file. The important part of this file is the *timeInfoExchangeItems*, which calls an exchange item which takes the value specified by one of the ioObjects from the Java code given by SobekRRTimeStepExchangeItem.java, specifying the start and end time of the simulation run. The exchange items here determine which information from Sobek RR is used, and the id they contain determines their name within the OpenDA configuration and refers to the ioObjects defined in the wrapper. Other important exchange items that are called here are the AsciiRestart file, the rainfall file, the Sobek settings file and modelled water level states.
3. **The StochModelConfig file** We are now getting to the stochastic part of the black-

box configuration, the left side of Figure 5.2. The first section of this file links to the ModelConfig file. The second section called *vectorSpecification*, states all the stochastic parts of the model and includes a state and predictor part. As the name suggests, the first part gives the state exchange items which are considered by the EnKF algorithm; all the states defined here will be updated accordingly. Another part of this section is the reference to Noisemodels, giving normal distributions on the uncertainty of the variables in a separate file. The outcome of the noisy vector is checked for feasibility in *RangeValidationConfigReader.java*. In this case that means for example that if the multiplicative noise on the rainfall is negative it is changed to 0 instead. The predictor part specifies the id of the state that should be compared to the observation as defined in StochObservation.

Stochastic Observer configuration This consists of two configuration files, the stochObsConfig and the ObservationUncertainties file. The first defines the uncertainty on the found observations. This file is referred to in the stochObsConfig, which first refers to the second file and then specifies which Java class and filename are used to retrieve the data used for observation.

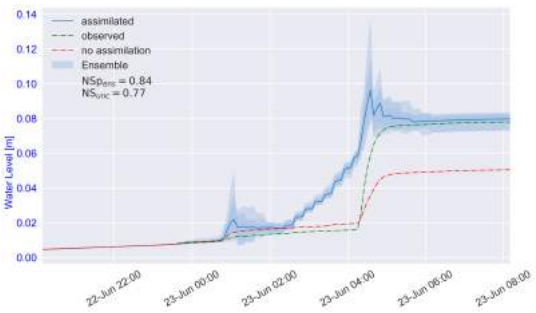
Algorithm configuration The last part of the main configuration is the file specifying the algorithm used for the DA. In this case, it includes also the settings of the EnKF such as the timeincrement between the assimilation steps, the size of the ensemble and the timeformat (modified julian date). The mathematical implementation of the EnKF was explained in section 4.2.4.

5.3 Problems with coupling both programs

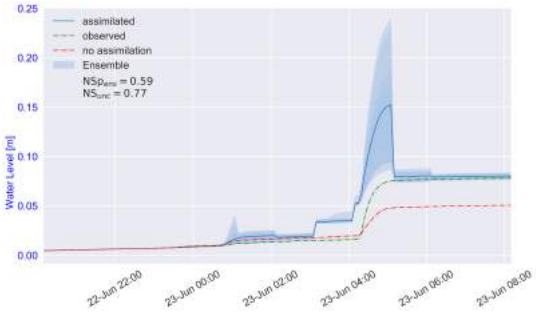
Both Sobek RR and OpenDA have been tested extensively, so it is likely that any errors that occur when using the wrapper are due to incorrect coupling of the software. However, some flaws can also appear once the black-box wrapper is in place. Especially since Sobek RR was originally not designed to be restarted this often, which could introduce inconsistencies and model errors when changing the model states. The problems with the coupling or with settings that occurred during this research are discussed in this Section.

Noise resets to average zero The first big problem encountered was a strange phenomenon that the EnKF algorithm seemed to loose 'strength' over time. The distribution $\mathcal{N}(1, 1.4)$ of which the rainfall multiplier is drawn would be corrected to $\mathcal{N}(0, 1.4)$ in a few assimilation steps. The resulting figures were confusing, starting out well but losing grip as the figures moved along in time. This was an OpenDA flaw, fixed in the noise time series model files in OpenDA by Deltares.

Different time zones in Sobek and OpenDA If the time zone was not explicitly specified in the observations, these observed values were interpreted differently by Sobek and OpenDA, causing presumptuous corrections and strange over-corrections as result (see Figure 5.3). This is easily overlooked and it is not so self-explanatory from the results what goes wrong. However, the solution is simple and can be implemented using the Noose module of OpenDA in which the time-zone GMT can be specified to the data. A similar specification can be done in the StochObsConfig file.



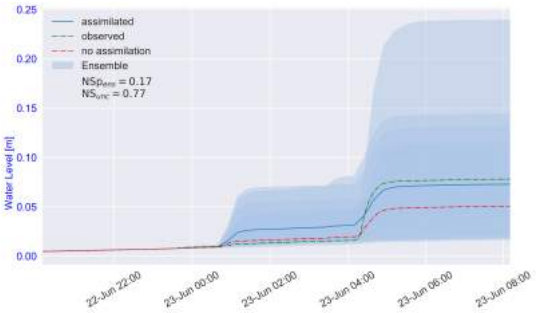
(a) Assimilation steps of 15 minutes



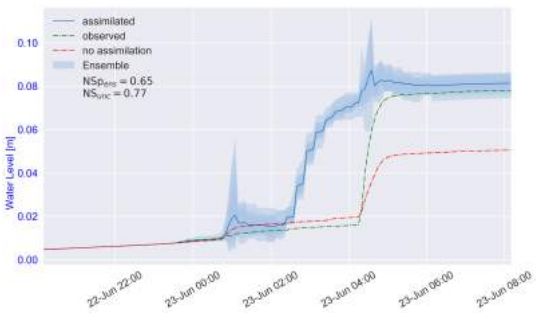
(b) Assimilation steps of 1 hour

Figure 5.3: The results on assimilated water levels when not explicitely specifying the observations are in GMT

Average time step in Sobek OpenDA requires the current modelled water level for the analysis. However, the settings in Sobek were set such that the output gave the average of the time step. This caused very different behaviour for different output times of Sobek as can be seen in Figure 5.4. The ensemble band became very wide when applying a large model time step of 15 minutes. However, when the model output time step was only 5 minutes, the assimilation over corrected the water level. This was fixed by changing the Sobek settings to give the current state as output.



(a) Sobek output every 15 minutes



(b) Sobek output every 5 minutes

Figure 5.4: The results on assimilated water levels when Sobek RR gives the average water levels of the time period as output.

6

Results

"Louise: "How did you get here?" Johnny: "Well, basically, there was this little dot, right? And the dot went bang and the bang expanded. Energy formed into matter, matter cooled, matter lived, the amoeba to fish, to fish to fowl, to fowl to frog, to frog to mammal, the mammal to monkey, to monkey to man, amo amas amat, quid pro quo, memento mori, ad infinitum, sprinkle on a little bit of grated cheese and leave under the grill till Doomsday."

– from the movie Naked, by Mike Leigh

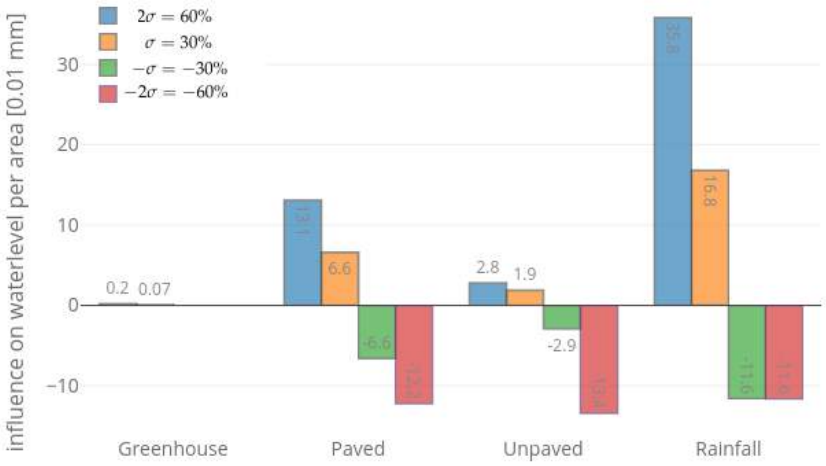
In this chapter, the main results which are relevant to the research questions are shown. The model settings and uncertainties were used as set out in Chapter 4. Some recurring terminology for this Chapter:

- **time increment** - the time span between two analysis steps of the EnKF
- **assimilated** - the resulting values of a data assimilation run
- **no assimilation** - the resulting values when Sobek RR is run without any adjustments
- **observed** - the values used as observations by the EnKF. These can be physical observations (in the case study), or made by a control run (in the twin experiment)
- $\mathbf{NS_{unc}}$ **and** $\mathbf{NS_p}$ - NS_{unc} denotes the Nash Sutcliffe value of the uncorrected (not assimilated) run compared to the observations. NS_p is the probabilistic Nash Sutcliffe comparing the ensemble with the observations.

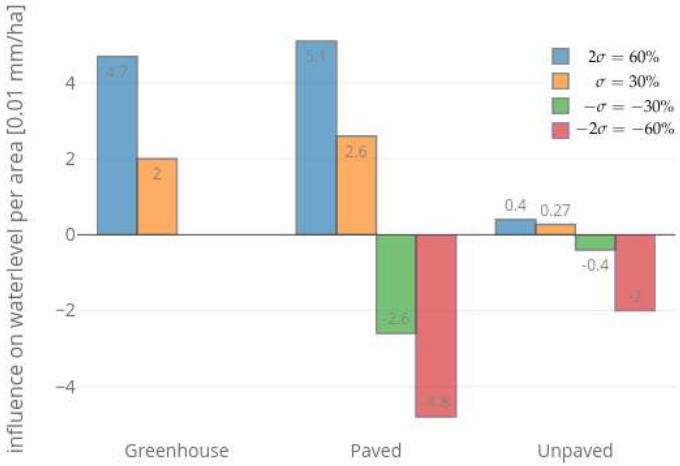
6.1 Contributions of the different model components to predicted water levels

The results on the sensitivity of the modelled water level to deviations in model components are shown in the bar-plots of Figure 6.1a and 6.1b. These were obtained following the set up described in Section 4.3, implementing relative increases per component of $\sigma = 30\%$,$2\sigma = 60\%$, $-\sigma = -30\%$ and $-2\sigma = -60\%$ The distinguished components were the precipitation forcing data and the initial states grouped per Sobek RR node. The resulting difference in water level predictions from the reference run for the particular case study of Tedinger broek are shown in Figure 6.1a. Since this is highly depended on the chosen areas of the land-use classes, Figure 6.1b normalizes the influence over the the area, obtaining results in millimetre per hectare. The results for the different components are summarized individually:

1. **Sensitivity on errors in the precipitation:** Figure 6.1a shows that for Tedingerbroek polder, the largest influence on the water level comes from variations in the precipitation. The effect of the 'range validation' algorithm as implemented in OpenDA can also be seen in Figure 6.1a: The algorithm prevented the rainfall



(a) Sensitivity of modelled water level on uncertainty in initial states (per Node) and rainfall for Tedingerbroek polder



(b) Relative sensitivity of modelled water level on uncertainty in initial states (per Node), where the influence to the water level was normalized over the land-use area.

Figure 6.1: Sensitivity of modelled water level on uncertainty in the initial states. The figure shows the corresponding deviation in water level for different test-cases. Sobek was run for 15 minutes, where the restart file was perturbed by adding $x \cdot \sigma$, where σ was defined as 0.3 times the initial volume in the states, for $x = 1, -1, 2, -2$. For the rainfall it was multiplied with $x \cdot (1 + 1.4)$ as 1.4 was the variance found in the rainfall analysis in Section 3.3.

from becoming negative, explaining the identical results for the cases where the initial storage decreased with -30% and -60%.

2. **Sensitivity on errors in the states of the greenhouse node:** The Greenhouse node contributes only 0.2 and 0.07 mm to the open water level for 2σ and σ respectively. There is no negative influence on the water level when the initial storage is de-

creased. The latter is logical since the only influence on the open water node is when the basin overflows. The reason for the small contribution compared to the states of the other nodes is the size of 3.3 hectares of the greenhouse area, only 3% of the total area of Tedingerbroek polder. Figure 6.1b shows that the influence on the water level per hectare is second highest of the areas with an increase of 4.7 mm on the water level when the storage increases with 60%.

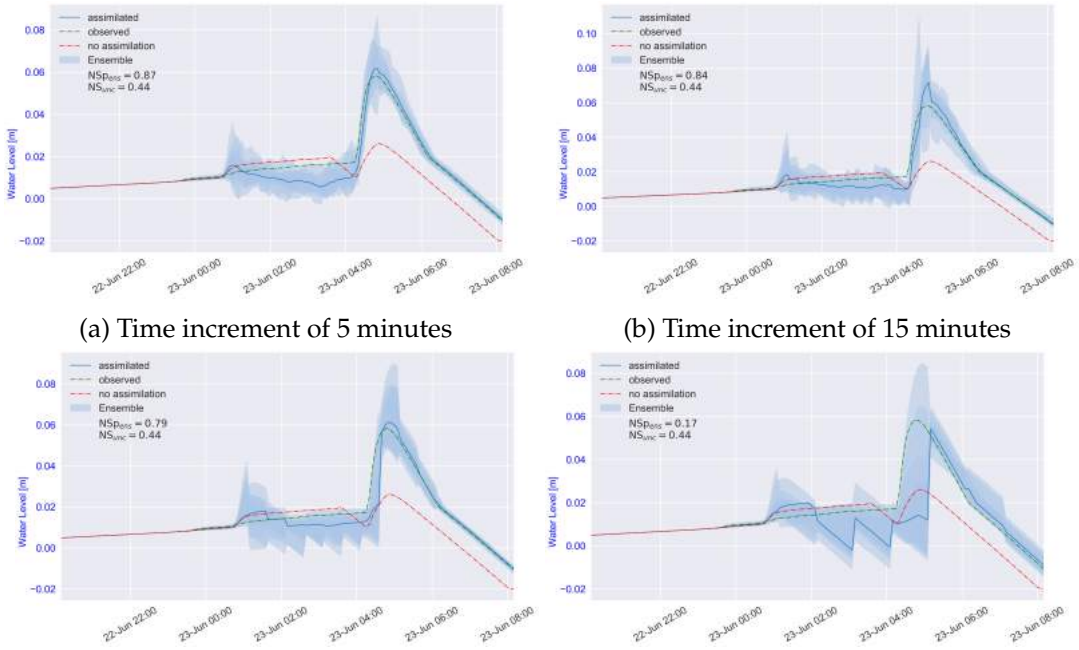
3. **Sensitivity on errors in the states of the paved node:** Errors in the initial filling of the states of the paved area (storage on land and storage in the sewage) contribute both absolute and relatively speaking significantly with a 13.1 mm total increase and $5.1 \cdot 10^{-2}$ mm per hectare in the water level for a 60% increase.
4. **Sensitivity on errors in the states of the unpaved node:** The states of the unpaved node (groundwater volume, volume in the unsaturated zone and storage on land) show low influence on the water level with a maximum of 2.8 mm when the state is perturbed with $+60\%$. This is in contrast to the size of the area; 690 hectares, a 68% of the total area, illustrated in Figure 6.1b. The negative influence on the water level when reducing the initial storage is largest of the components, a total reduction of 13.4 mm. This visualizes the non-linear behaviour of the storages of the unsaturated zone and the groundwater volume. The resistance factors R given in equation A.8 determine this behaviour. The higher the difference between the groundwater level and the open water level, the bigger the flux towards the open water. As soon as the groundwater level is lower than the water level ($\Delta h < 0$), water flows to the groundwater instead turning from the flux $q = \frac{\Delta h}{200}$ the flux changes direction and flows towards the groundwater, reducing the open water level.

6.2 Twin Experiment

The conceptual feasibility of data assimilation to reduce input uncertainty was tested with four twin experiments as set out in Section 4.4. Before testing the different DA approaches, the influence of the algorithm settings was evaluated. The evaluated settings are the influence of the pumps, the ensemble size and the time increment.

6.2.1 Influence of pumps

The influence of the pumps discharging water from the open water state to the boezem can be seen in Figure 6.2 for different time increments. It can be seen that the assimilated water level is lower than both the uncorrected as the observed water levels for the first half of the time run, showing that the data assimilation has a negative effect on the predicted water levels in this period. This becomes more so when the time increment increases.
The negative effect DA has on predicted water levels is the result of the restart pump settings for each individual ensemble member. Due to the perturbations on each ensemble member, every ensemble member will reach a point where the water level reaches above the turn-on point of the pump (in the twin-experiment 2 cm above the peil). From that moment on, the pump will stay turned on until the water level of the ensemble member reaches the turn-off point (set at -2 cm). As every ensemble member reaches this turn-on point at a different time, there is a big spread in water level results of the ensemble members. The water levels in the control run have not necessarily reached the turn-on level. If they have not, the water level of the ensemble members will drop under that of the observations. This results in upward corrections on the forecast water level of



(a) Time increment of 5 minutes

(b) Time increment of 15 minutes

(c) ime increment of 30 minutes

(d) ime increment of 60 minutes

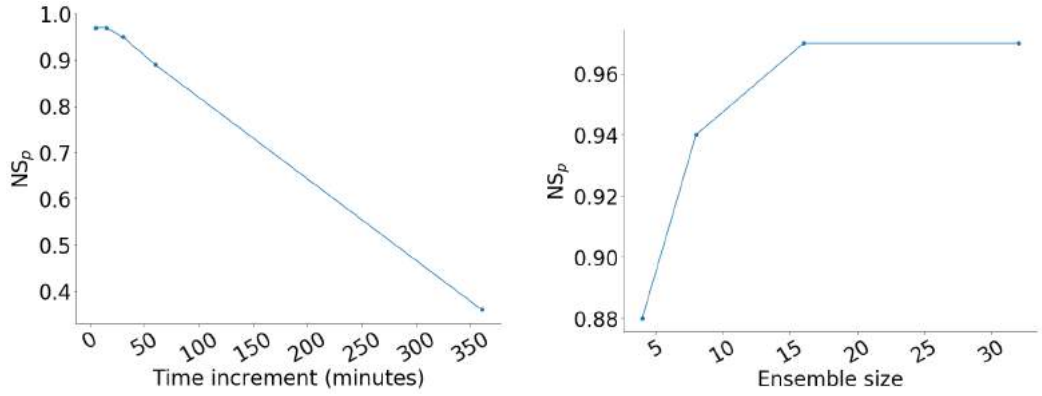
Figure 6.2: The influence of the pump settings on the assimilated ensemble. The figures correspond to a run with an ensemble of 16 members with the perturbation on the rainfall.

the ensemble members. Therefore, the ensemble member can take a long time to reach its turn-off point, resulting in the zigzag movement shown in Figure 6.2. As soon as the water levels of the control run become higher than the turn-on point of the pump, the control and assimilation run approach each other again.
The time increment influences the magnitude of the zigzag movement. The bigger the time increment, the longer the pump gets to discharge water to the boezem and reduce the water level before it is corrected upwards again. However, even if the time increment is only 5 minutes, the pumps reduce the water level below the uncorrected and observed values, as can be seen in Figure 6.2a. This results in a decrease of predictive power of the water level. This holds not only for the water level but for the complete assimilated state vector, as can be found in Figure C.1.
To prohibit the pump settings from obscuring the comparison of the different DA approaches, the pumps were removed for the remaining of the twin experiments.

6.2.2 Testing algorithm settings

The algorithm settings that were fine-tuned are the time increment and the ensemble size. To compare different settings, the set-up of Section 4.4 with the rainfall as only perturbed uncertainty was used. The NS_p results for the assimilated water level for time increments between 5 minutes and 5 hours are given in Figure 6.3a. The NS_p results for the water level with ensemble sizes between 4 and 64 are given in Figure 6.3b. The figures showing the assimilation of the water level can be found in Appendix C.3 and C.2 respectively. The results are discusses per setting:

Time increment: From Figure 6.3a it can be seen that the time increment influences the ensemble approximation of the observed water level significantly. A time increment of an hour still improves the prediction over the run without assimilation, while a time



(a) (b)

Figure 6.3: Influence of algorithm settings; ensemble size and time increment

increment of five hours reduces the NS_p from 0.89 to 0.36 which is far lower than the original $NS_{unc} = 0.77$. With reducing time increments, the computational cost increases. The optimal settings are therefore where both the NS_p results and the absolute value of the derivative of Figure 6.3a are high. A **time increment of** 15 **minutes** is therefore optimal for this model and will be used throughout the following experiments.

Ensemble size: From Figure 6.3b it can be seen that the highest rate of change, together with high NS_p values occurs at an **ensemble size of 16**. Note that only 4 ensemble members give already an improvement to the $NS_{unc} = 0.77$ corresponding to the setting without assimilation. Note that the influence of the ensemble size is relatively low compared to the influence of an increasing time increment.

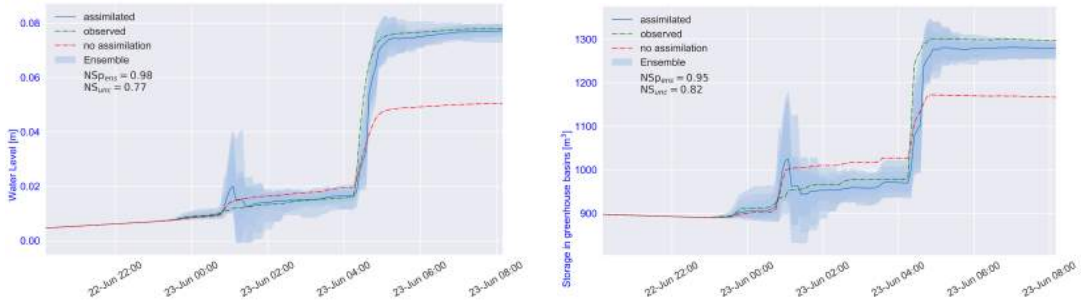
The computational time needed with a time increment of 15 minutes and an ensemble size of 16 is approximately half an hour to calculate a time-period of 1 day, which equals **20 seconds per assimilation step of 15 minutes**. This is a feasible computational cost for real time implementation because the optimal solution can be calculated in a time smaller than the implementation time, which can be taken as the time increment.

6.2.3 Perturbation on different components of uncertainty

The different DA cases implemented noise on the following model components, following the set-up discussed in Section 4.4.1. The four approaches were:

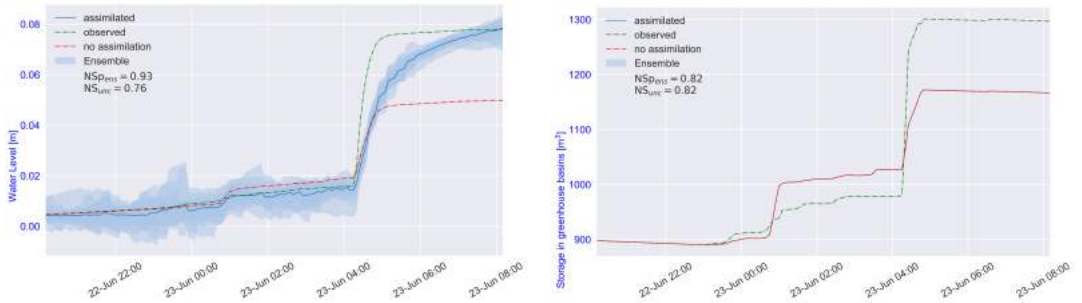
1. **Multiplicative noise on rainfall:** The rainfall multiplier was drawn from the Gaussian distribution $\mathcal{N}(1, 1.4)$.
2. **Additive noise on the groundwater state:** The noise was drawn from $\mathcal{N}(0, 470000)$ m^3, where 470000 is one percent of the initial state of a half filled system.
3. **Additive noise on all states:** The noise was drawn from multiple normal distributions, where the variation is taken as one percent of the initial state of a half filled system.
4. **Multiplicative noise on all states:** One multiplier drawn from $\mathcal{N}(1, 0.05)$, where the variance represents relative change of 5 percent of each storage component.

The assimilated water levels for these different approaches are shown in the left column of Figure 6.4. The resulting NS_p values for the assimilated water level range between



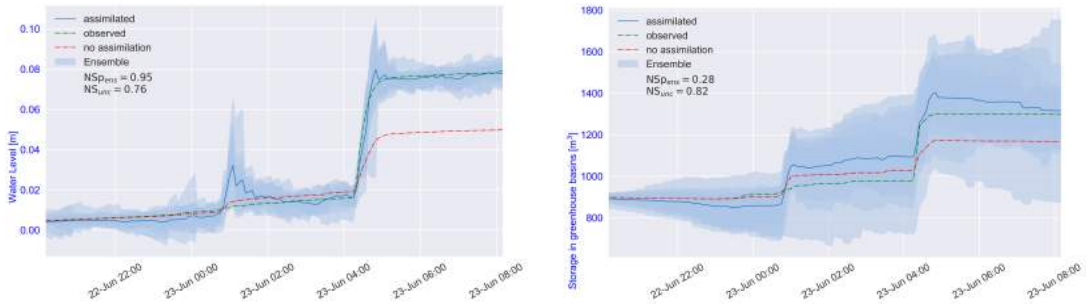
Water level Added storage in GH basins

(a) Multiplicative noise on Rainfall



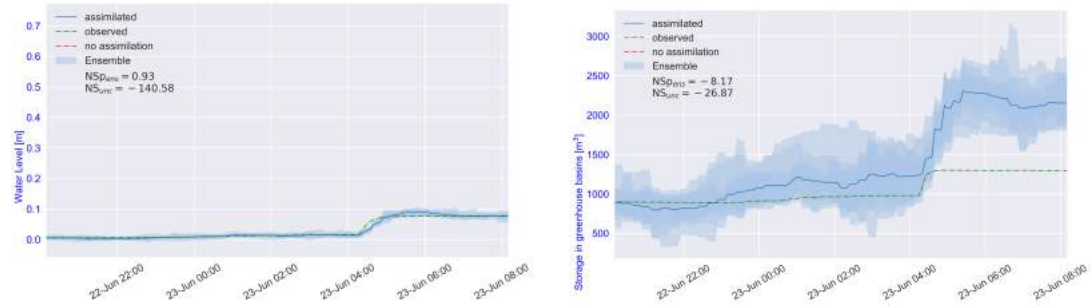
Water level Added storage in GH basins

(b) Additive noise on one state, the groundwater volume



Water level Added storage in GH basins

(c) Additive noise on all states



Water level Added storage in GH basins

(d) One multiplication factor on all states

Figure 6.4: Assimilated results for data assimilation with the noise on different model components. For each approach, the resulted water level and the combined storage in the greenhouse (GH) basins are shown.

0.92 and 0.97. This does not need to imply that the other assimilated states also mimic the control values well. This is illustrated in the right column of Figure 6.4, where the summed storages of the greenhouse basins are shown. The NS_p values for the assimilated greenhouse storage range between -8.31 for the single multiplier for all states and 0.8 for the case with a rainfall multiplier. The DA set-up with the rainfall multiplier generates the best results for the water level and the greenhouse storage, but also for the other states as can be seen in the additional figures in Appendix C.3.
The DA set-up with additive noise on all states shows a NS_p of only 0.13 for the assimilated greenhouse state. However, visual inspection shows that the ensemble average follows the observations reasonably, such that the low NS_p value is due to the high variance shown by the wide blue band. Even though the the NS_p is low, the assimilated result could still be seen as an improvement to the system without assimilation.
The result of the greenhouse basins in Figure 6.4.b shows that for model components without noise, there is no ensemble spread and therefore not updating either. The assimilated results overlay the modelled values without assimilation.

6.3 Case study

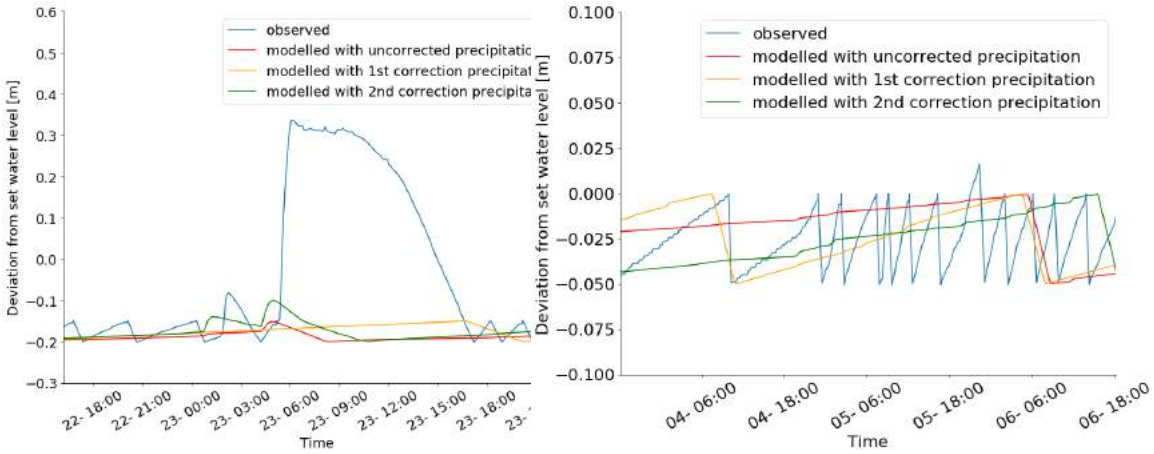
6.3.1 Difference between modelled and measured water levels

The physical observations of the water level near the pump in Tedingerbroek polder are shown in blue in Figure 6.5. The same figure shows the water levels as modelled in Sobek with respectively the uncorrected radar rainfall product, the 1st corrected radar product and the 2nd-corrected radar rainfall. The rainfall corresponding to these periods could be found in Figure 3.3. Observations when comparing modelled and observed water levels were:

1. The rainfall generated an immediate effect on the modelled water level, while the observed water levels showed a time-lag.
2. The modelled pump does not behave in the same way as the observations suggested. In a period of two days in November, as shown in Figure 6.5b, the observations suggested that the pumps turned on 12 times. In contrast, the modelled pumps turned on only once, independent of the chosen radar rainfall product.
3. The amplitude of the modelled water level during the extreme rainfall event on the 23^{rd} of June was much smaller for the modelled water levels than compared to the observed values. The observed water levels reached a maximum of 0.33 m above set water level, while the modelled values in Sobek did not exceed the set water level.

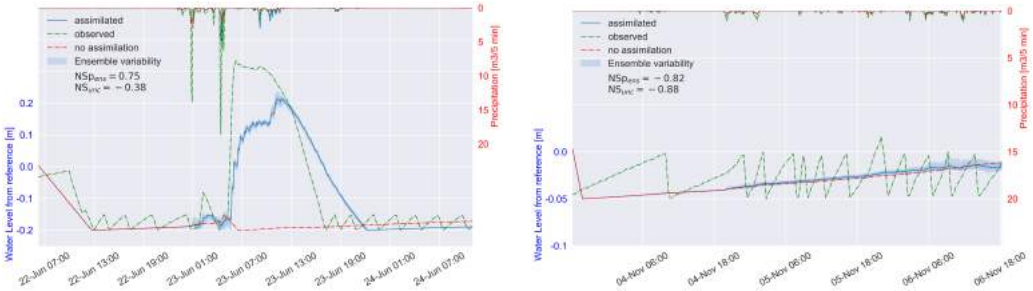
6.3.2 Data assimilation applied to case study

From the previous results, the set up with the rainfall multiplier generated the highest NS_p for the assimilated states compared to the control values of the Twin experiment. This method was chosen to be applied to the case study of Tedingerbroek. Both the events of Figure 6.5a and Figure 6.5b were assimilated and the resulting water levels are shown in Figure 6.6. The results for the hidden states can be found in Figures D.1 and D.2.
The assimilation for the event in June showed that the water levels were corrected towards the measurements, but that the magnitude of the observed water level was never reached. After the observations drop again, the assimilated ensemble overestimated the water level for at least 9 more hours with a high average of 20 cm. For the assimilation



(a) Event on the 23rd of June, 2016 (b) Event from 4 to 6 November, 2016

Figure 6.5: Observed water levels compared to modelled water levels. The respective input for the modelled water levels in Sobek are the uncorrected (15 minutes) radar rainfall, the first corrected (hourly) radar rainfall and the second corrected (daily) rainfall radar rainfall.

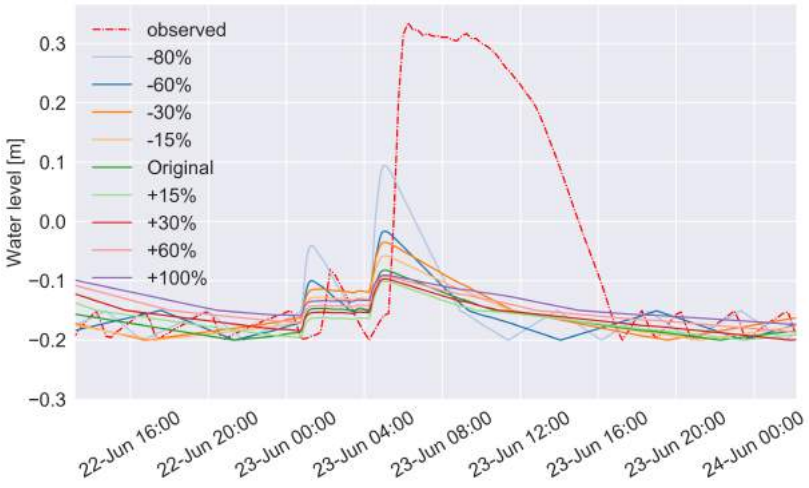


(a) Assimilation for the 23rd of June, 2016 (b) Assimilation from 4 to 6 November 2016

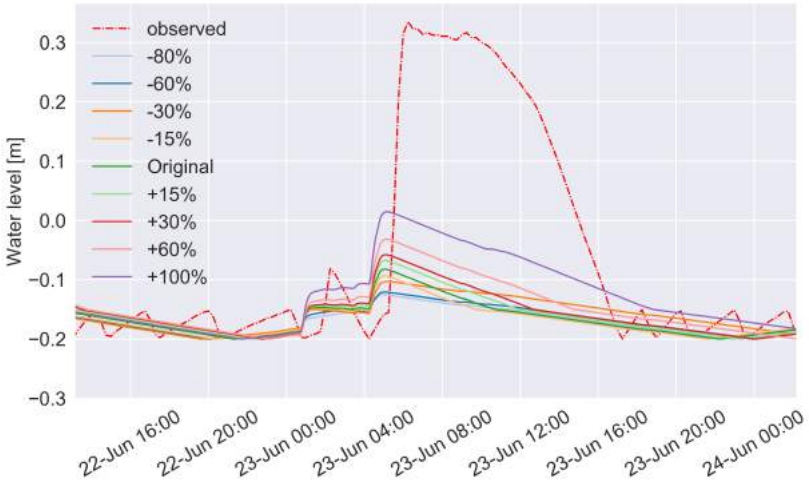
Figure 6.6: Assimilated water levels for case study, with multiplicative noise on the rainfall.

in November, the results showed only a small ensemble spread in the water level and the ensemble mimicked the observations no more than the results without DA did.
Even though the parameter values used in the case study were based on literature and existing models of the polder, the results represented in Figure 6.5 suggested that the assimilation could benefit from a model calibration. A manual calibration was performed, based on an analysis of relative increases of -80%, -60% ,-30%, -15%, $+15\%$, $+30\%$, $+60\%$ and $+100\%$ on different parameters and their effect on modelled water levels. The analysis was applied to the parameters that were expected to influence the magnitude, pump pattern and time lag, which included the different land-use areas, pump capacity and concentration time. The results are illustrated in Figure 6.7 for the open water area and the paved area. Relevant points of this analysis for the calibration were:

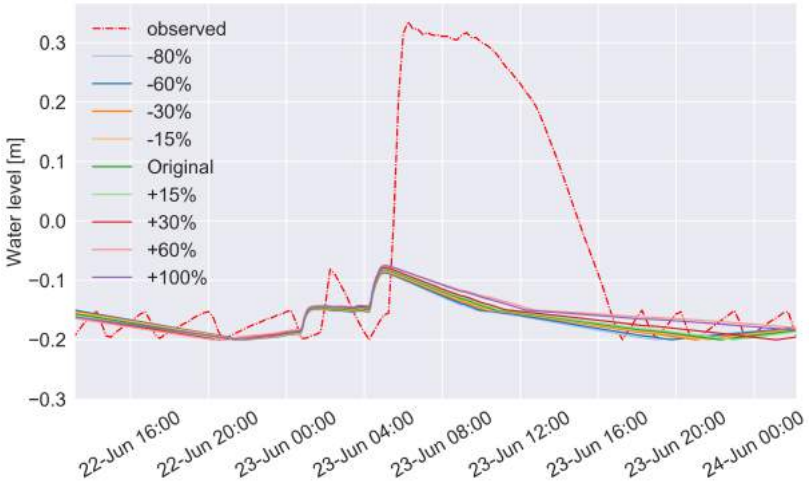
1. Small water bodies fill and empty faster, creating higher, but short, peaks and more frequent turning on and off of the pumps. In the calibrated model, the open water area was changed from 40 hectares to 10.
2. The paved areas did not only influence the magnitude of the water level peak, but also showed a wider peak compared to reducing the water bodies. These were increased from 280 to 400 hectares in the calibrated model.



(a) OW area



(b) PA area



(c) UP area

Figure 6.7: Influence of variations in land-use areas on the modelled water level

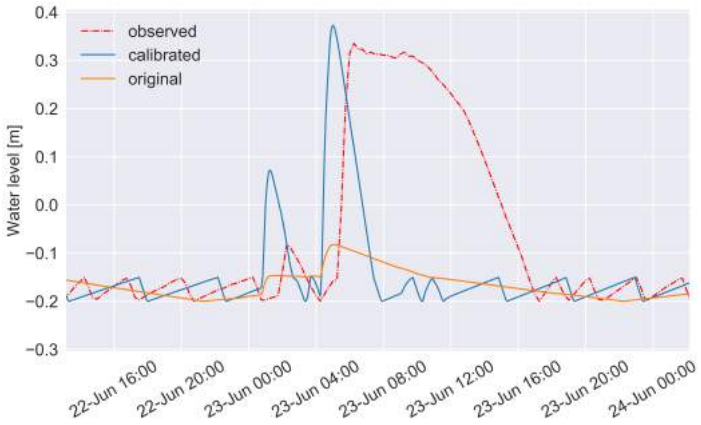
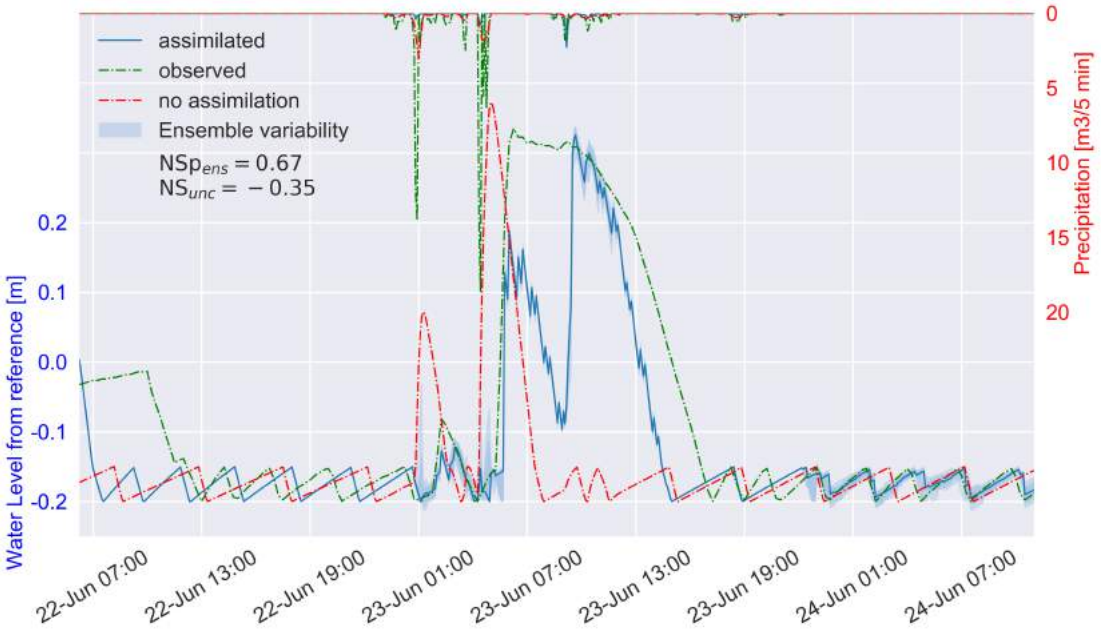


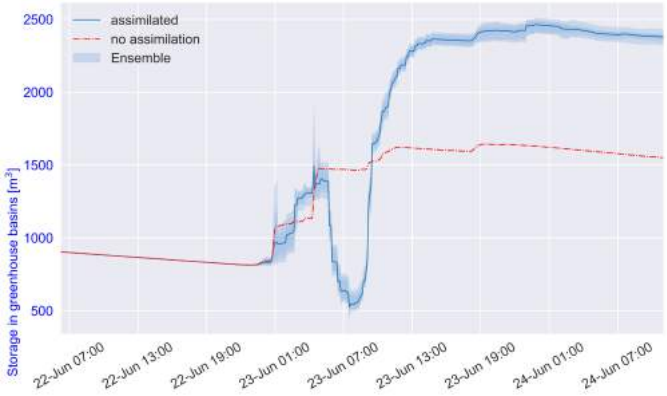
Figure 6.8: Resulted water levels from calibrated model, original model and observations

3. The groundwater contributed little to the peak flows, but implemented a baseflow, increasing with the size of the unpaved area. The surface area of the groundwater was left unchanged but the area for groundwater computations was increased with the paved area to 600 hectares.
4. The relative increases in greenhouses did not contribute to a change in water level because their representation in Tedingerbroek polder is small, as it is not realistic to implement this feature as a dominant part of Tedingerbroek polder, the value was left unchanged.
5. Higher pumping capacities reduced the peak flow, but implemented a pump pattern better comparable to the observation. In the calibrated model, the pumping capacity for Tedingerbroek, which equalled 56 m^3/min, was changed to the pumping capacity as was implemented in the original Sobek polder cluster, equalling 202 m^3/min.

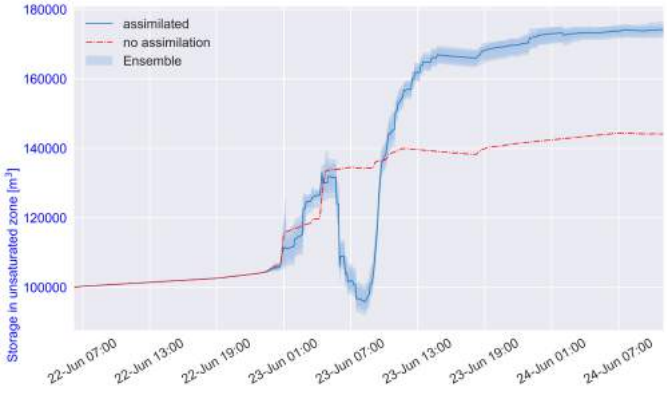
The modelled water levels corresponding to the calibrated areas are displayed in Figure 6.8, and showed improvement in the sense that the magnitude of the peak could match the observations and the pattern of the pumps was similar to the observations. However, the time lag remained, as did the difference in duration of the peak water level. For the implementation of DA on this case, the same assimilation set-up was implemented of which the results, given in Figure 6.9, showed an increase compared to the DA applied to the original model. Visually, the DA of the calibrated model captured the water level behaviour much better because the ensemble reached the high water levels and had a similar amount of turn on- and off points. The assimilated hidden values go down when the rain cedes, following the trend of the model, but correct upwards as soon as there is rainfall again.



(a) Water levels and rainfall



(b) Storage in greenhouse basins



(c) Storage in unsaturated zone

Figure 6.9: Results of DA with calibrated RR model

7

Discussion

Scientists should always state the opinions upon which their facts are based.

–Author Unknown

The discussion will review the results for their quality and shortcomings with respect to the research questions. Furthermore, the resulting applicability of the data assimilation approaches in practical situations is assessed. Possible improvements or further research are given where necessary.

7.1 Sensitivity to errors in model components

Sensitivity on errors in the rainfall forcing data The rainfall multiplier was the model component that influenced the response water level most when relatively increased. This underlines the research of (Kuczera et al., 2006) and justifies the further approach in the results of using data assimilation techniques which mainly tackle the uncertainty in the rainfall by use of a rainfall multiplier.

Subjectiveness of the analysis The results on the sensitivity of errors in initial states and precipitation depend on the following factors:

1. **The value of the initial states and rainfall:** The effects of the paved- and rainfall components will increase during wet periods or extreme rainfall. More so, without rain the only model component influencing the water level is the unpaved area, whose groundwater state is directly linked to the water level. The justification for the chosen initial states is that the high water levels and hence the corresponding wet system state are of interest most for the controllers because their main task is to prevent water nuisance. Thesame argumentation justifies for an analysis period of a rainfall event.
2. **The areas corresponding to the typical area-nodes:** The area sizes per node influence the volume of water the nodes receive during rainfall events influencing their runoff. For the states corresponding to an area-node, their response is normalized over this area to reduce this bias, shown in Figure 6.1b. This could not be done for the rainfall, as its contribution per area is again depended on the distribution of the polder area of the area classes (paved, unpaved, greenhouse and open water). Therefore, the sensitivity of errors in the rainfall on predicted water levels will always be a weighed number of the normalized contribution of the other areas.
3. **Time increment chosen for the analysis.** Initial states would yield smaller changes in the water level if the analysis is applied over a longer run-period. Furthermore, Changes in initial storage in the reservoirs with fast runoff processes will generate a relative larger effect on the water level in a short time increment than slow processes. Hence, the short run favours fast runoff processes over slow processes. The

small time increment is chosen to be 15 minutes, the same as the time between data assimilation updates, such that the influence of the uncertainty on the storages in the states can be assessed for the application of the DA.

7.2 DA and perturbed model components

From the sensitivity analysis and the assimilated results it can be seen that perturbing different model components gives a large deviations in assimilated results. These results for the twin experiment can be biased towards the approach with the multiplicative noise on the rainfall, because the control values were also simulated by adding noise on the rainfall forcing data. To be able to draw better conclusions, different twin experiments should be compared and this is advised to do for further research. The effect of the different approaches is discussed per set-up:

Perturbations on the precipitation

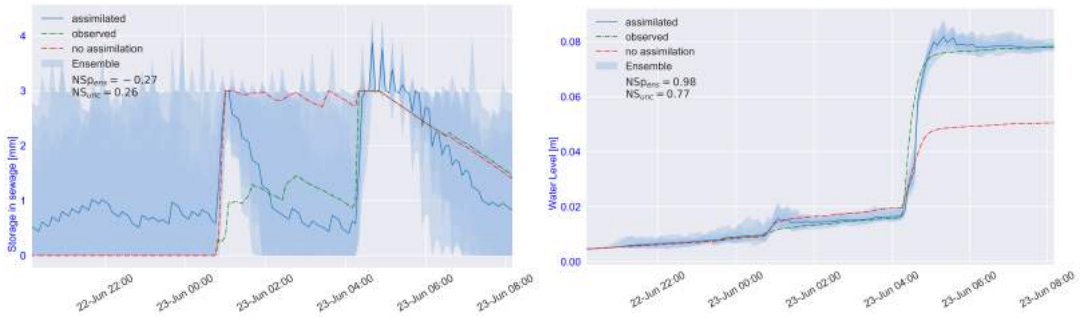
Applying noise on the rainfall using a rainfall multiplier is the best option as noise component. The assimilated NS_p values do not only score high for the assimilated water level but also highest for the different states. These results are better compared to the other noise approaches, and yield improved results over no assimilation both visually and in terms of NS_p values. Furthermore, the concept of applying the noise directly on the rainfall has conceptual benefits:

1. The precipitation has in a short time span the largest influence on the water level, making it a sound parameter to scale.
2. The uncertainty on the rainfall is significant, can be estimated well and influences the water level most, making it an ideal noise component for DA. Different rainfall products are available to determine the prior including the uncorrected radar rainfall, 1st corrected radar rainfall, 2nd corrected radar rainfall and KNMI data. The prior in form of a multiplier $r \sim \mathcal{N}(1, 1.4)$ shows the uncertainty is significant and Figure 6.1a showed that the system response is also sensitive to this error.
3. By putting noise on the forcing data, the whole system is updated 'equally', i.e. the incoming volume of water changes equally for the different land-uses relative to their areas. This ensures that the data assimilation cannot correct for the error in one parameter in one land-use by choosing a different multiplier, since then also the other land-uses will respond differently.
4. The sources of error stay disentangled, following the conceptual hopes that DA can be used to target specific elements of model uncertainty set out in the introduction.

There are two downsides of using only the the rainfall as source of error in the DA. The first, as no other sources of error are specified, the distribution of the ensemble predicted water level might have a small variance allowing for only small corrections. Furthermore, the assimilation is only effective during rainfall events.

Perturbations on the states

The advantage of applying perturbations on the states is that the assimilation is not dependent on whether it rains or not. The states were perturbed in different ways, the results of which are discussed below:



(a) Storage in sewage

(b) Water level

Figure 7.1: Assimilation with only the storages of the paved area (storage on land and storage in sewage) perturbed. The water level observation are still approached by the assimilatation run (7.1b), but this is reached by over-adjusting the storage on paved land and in the sewage (7.1a). The method to generate the images is DA on the Twin-experiment set-up with additive noise on the storage on land and storage in the sewage. The chosen noise distribution (in mm) was $\mathcal{N}(0, 0.3)$ for the storage on land and $\mathcal{N}(0, 0.5)$ for the storage in the sewage.

1. **Additive noise on the groundwater state:** When only the groundwater was perturbed, the states of the other land-uses were unchanged for each ensemble member because the Sobek land-use classes are not linked. Remember that the update is calculated using

$$\widehat{x}_i = \widetilde{x}_i + K(y_{obs} - x_m + v), \tag{7.1}$$

with Kalman gain K given by

$$K = \frac{\sigma_m^2}{\left(\sigma_m^2 + \sigma_{obs}^2\right)} \begin{bmatrix} \sum_{i=1}^{N} (\widetilde{x}_1^i - \widetilde{\xi}_1)(\widetilde{x}_m^i - \widetilde{\xi}_m)/\sigma_m^2 \\ \vdots \\ \sum_{i=1}^{N} (\widetilde{x}_{m-1}^i - \widetilde{\xi}_{m-1})(\widetilde{x}_m^i - \widetilde{\xi}_m)/\sigma_m^2 \\ 1 \end{bmatrix}. \tag{7.2}$$

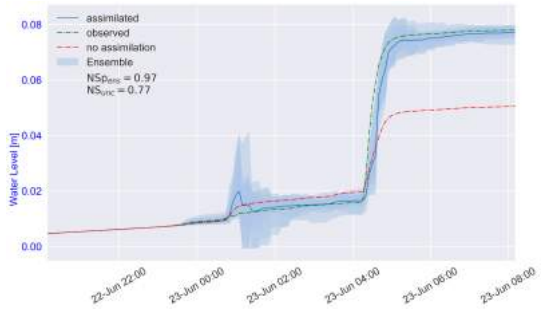
As the average ξ is the same as x_j^i for all states j disconnected form the perturbed states, this gives $K[j] = 0$, implying that also the updated states of the ensemble members remain unchanged for the unperturbed areas. This can be seen in Figure 6.4b. Perturbing only one state is therefore only a viable option if the prediction uncertainty is believed to root dominantly in one of these area-classes. A downside is that the perturbed states will try to correct for all model deficiencies, not just an underestimated rainfall event or discharge processes. This is illustrated by 7.1 showing an increase in NS_p value over NS_{unc} from 0.77 to 0.98, but a decrease from 0.26 to -0.27 for the storage in the sewage when only applying additive noise on the states of the paved area.

The groundwater volume state is the only state which always communicates with the open water. It is also the state with the biggest water volumes. According to the controllers (peilbeheerders) in Delfland, the unsaturated zone and groundwater are not always modelled well. It is said that the unsaturated zone does not hold enough water such that the groundwater discharges to the open water bodies too quickly. This would opt that the groundwater volume would be a possibility to perturb. However the results only approximate the water level better, not the states of the unpaved area.

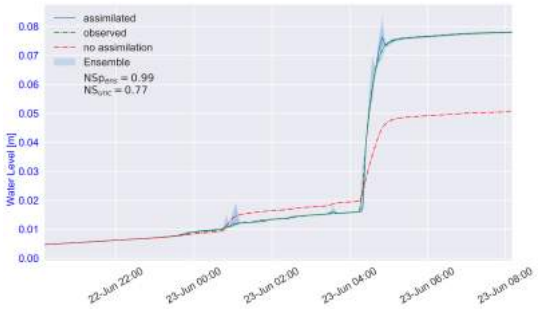
2. **Additive Noise on all states:** This method allows each state to be perturbed with the noise drawn from its own prior uncertainty distribution. The average of the ensemble showed to approximate the control values of most states well, but failed to capture the behaviour of some of the states. The first possible explanation for this is that many states, such as the greenhouse storage basins, work with threshold behaviour, such that different storages generate a similar behaviour until that limit is reached. This can create a very wide ensemble representation for the states, which all generate the same water level. The second explanation for the ensemble to be wide and not capturing the behaviour of some states, is because the prior was chosen quite broad, because not much was known about the uncertainties of the different parameters and states. Perhaps the results could be improved with better fine-tuning of the respective distributions, which would require additional information of the modelled system. Thirdly, it is possible that due to random drawing of errors, the states corrections nullify each other, which could disturb the algorithm. This method might therefore benefit from an increased ensemble size. Before this method can be implemented in practice, it will need additional testing.
3. **Multiplicative noise on all states:** This approach assumes that the error in the states is correlated due to wrong forcing data and has one multiplicative factor drawn from an uncertainty distribution which is used to perturb all the states. The results for the single multiplicative noise component on all states do not show good results. This can be explained by a scaling problem; the different states work with very different orders of magnitude. For the systems generating fast runoff, such as the storage in the sewage, the volumes are often low and would need to be multiplied by six before creating runoff, while a multiplication of the groundwater by six would not be realistic at all. The problem of scaling is also within the storage buckets. For example, if the storage in the sewage is relatively filled (e.g. 2.5 mm), it is unrealistic to create ensemble members that contain a sixfold of this, which would generate $2.5 \cdot 6 - S_{sew,max} = 12$ mm runoff instead of zero. Hence, it is difficult to choose one prior distribution for the multiplication factor.

7.3 Influence of pumps

For the twin experiment, pumps were eliminated from the technical set-up such that the theoretical applicability of different DA methods could be assessed. For the case study, the pumps could not be removed from the set-up because the observed water levels were also influenced by the pumps. Therefore, the introduced errors by the pumps cloud the results on applicability of DA in the case study. Since the results from the twin experiment showed significant improvements to the modelled response without assimilation, it is worth fixing the problem introduced by the non-linearity of the pumps. There are two reasons the pump settings could not be implemented as a state themselves that could be updated by the assimilation. The first is that these settings cannot be changed per restart by the user or by the DA because the state of the pumps is not given in the ascii-restart file that was created for this thesis. The second reason is that even if the pumps could technically be implemented as state, the three pump settings, off, half capacity and full capacity, are not continuous. This introduces an integer problem: The data assimilation can only deal with continuous states and will update the pump discharge as a real number, which cannot be implemented in Sobek RR. One solution would be to translate this number again into one of the three settings, e.g. by rounding off. This would be worth looking into, but is not the preferred approach.
The recommended approach to fix the problems introduced by the pumps is first to allow additional information on the pump settings in the ascii-restart files. This software



(a) Sobek output when the uncertainty on the observation is 1.5 cm.



(b) Sobek output when the uncertainty on the observation is 1 mm.

Figure 7.2: The influence of the observation uncertainty on assimilated water levels

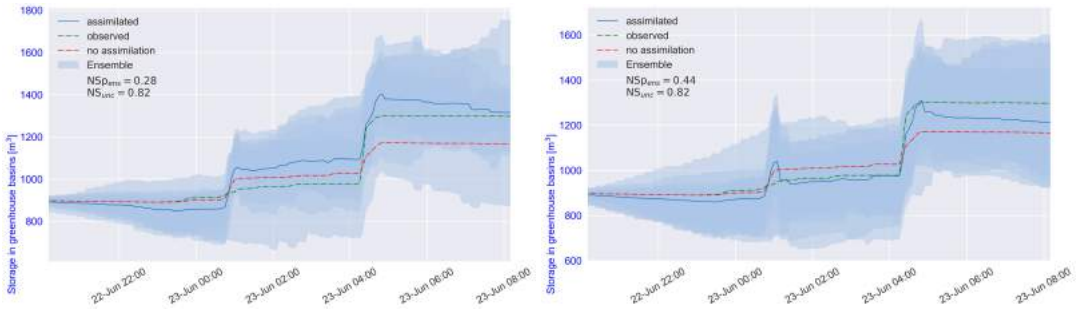
extension is possible and should be implemented by Deltares. The recommended approach to change the pump settings to match the reality better is by implementing the pump settings as an additional observation in OpenDA. The observation should be limited to whether the pumps are turned on or off and implemented as an observation with zero uncertainty. That way, the ensemble members can recover for falsely turning on of the pumps. In this approach, the human actions, which do not always follow the modelled optimal control, can also be taken into account.
In practice in Delfland, the observations and restarts are handled by a data handling platform called FEWS. This means that in practise, the pump settings can be communicated via this platform to Sobek RR or the DA wrapper such that the ensemble members can be tuned accordingly during each restart.

7.4 Conceptual feasibility of DA

Considering the DA study with only perturbations on the rainfall, the concept worked really well. The ensemble spread with a small variance around the control values. As was seen already in Figure 7.1b and in the right column of Figure 6.4, the EnKF algorithm can 'pull' the posterior to have a distribution with its mean approximating the observed value and a reasonable small sigma. The approximation of the observed value was only limited by the observation uncertainty. This is illustrated in Figure 7.2, where the realistic water level uncertainty of 1.5 cm was changed to 0.1 mm.
Caution needs to be applied to the effects the assimilation has on the other modelled states. Even when the assimilated water levels approach the observed water levels well, the approach does not necessarily generate realistic results for the other states. Considering this, the only DA method that demonstrated plausible results for all states in the twin experiment was the approach with the rainfall multiplier, giving an average value for the NS_p of 0.6, and through visual inspection an improvement over the predicted water levels without assimilation for all states, hidden and observed. This showed that by only implementing the forcing uncertainty, the water level predictions could be improved.

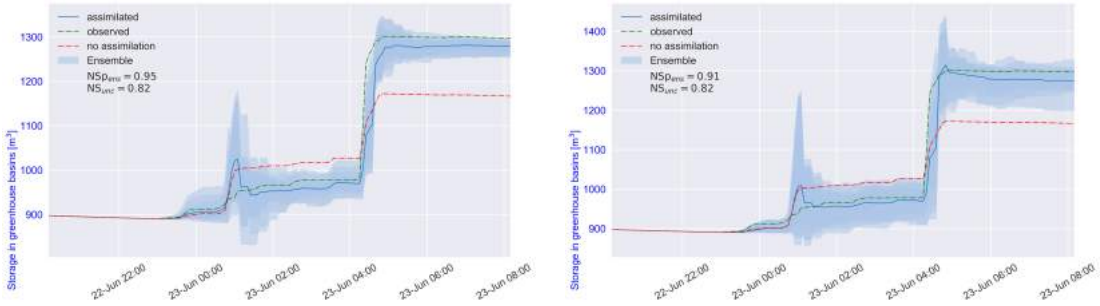
7.5 Assesment of algorithm settings

The algorithm settings were only tested with the DA approach with the rainfall multiplier, and only for the assimilated water level. It could be that the settings have a different effect on other states. Furthermore, it could mean that the results are biased, favouring the DA method with the rainfall multiplier because the algorithm is optimized for this approach. One could hypothesize, that as the number of noise elements drawn



16 ensemble members 32 ensemble members

(a) Assimilated storage in greenhouse basins, for DA approach with additive noise on all states



16 ensemble members 32 ensemble members

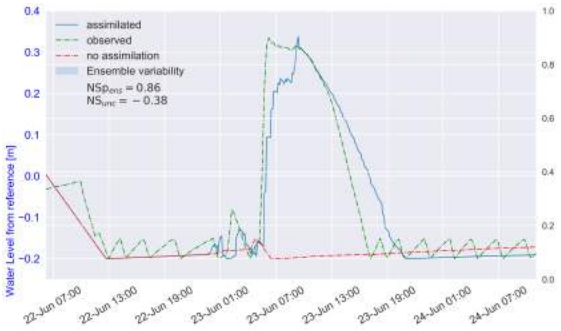
(b) Assimilated storage in greenhouse basins, for DA approach with multiplicative noise on rainfall

Figure 7.3: Influence of ensemble size on the assimilated storage in the greenhouse basins for two DA approaches; rainfall multiplier and additive noise on all states.

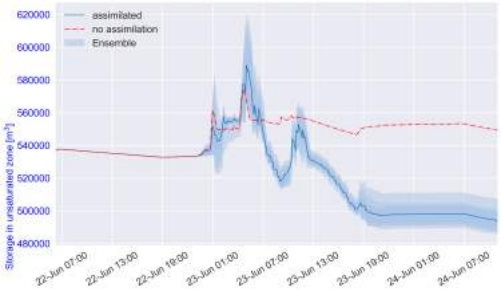
from different distributions increase, so should the number of ensemble members. This could increase the optimal ensemble size for the approach where additive noise was implemented on all states. Nothing on this was found in literature and a simple test of comparing to 16 to 32 ensemble members for the set-up with additive noise on all states, gave no large improvement on the assimilated water levels. However, comparing the assimilation of different states in the control run does show an improvement with increasing ensemble size for both the noise-approaches. The NS_p values for the storage in the basin for the additive-noise component increases from 0.13 to 0.44. The results for both noise-approaches is shown in Figure 7.3. For practical application it is advised to use a maximal ensemble number allowed by the time span available for computation. For the twin experiment, the small time increment of 15 minutes is ideal for all cases.

7.6 Application of DA in the case study

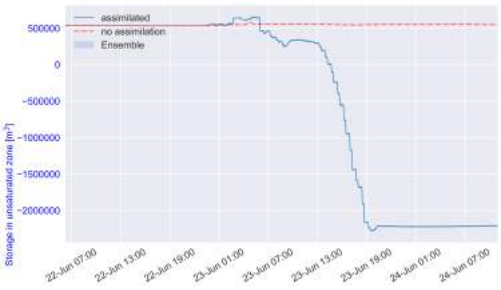
However, as shown in Figure 6.5, the modelled values showed a complete different pattern in the turning on- and off of the pumps, a large difference in the magnitude of the water level rise and a significant time lag in the observed response. These are factors that were not tested in the twin experiment because the control values were created with the same Sobek RR model. These differences between model and observed water levels resulted in concerns about the applicability of the data assimilation to the case study because if the model could not represent the observations, how would the data assimilation result in updates of hidden states with a proper physical meaning? Furthermore, as the case study had no state observations besides the water level, it was not impossible to validate whether the assimilated states made physically sense. However, the extreme drop in the values of hidden states half way throughout the



(a) Water level assimilated with observation uncertainty of 1.5 cm



(b) Unsaturated zone assimilated with observation uncertainty of 1.5 cm



(c) Unsaturated zone assimilated with observation uncertainty of 1 mm

Figure 7.4: Influence of different observation uncertainties on the assimilated unsaturated storage for the case study with original RR model.

rainfall event, as shown in Appendix D.2, confirmed the suspicion that the updated hidden states would not result in probable values.
As in the case study, the assimilated water level could be improved when the uncertainty on the observation was low. However, even though the assimilated water levels improved visually, this only increased the errors in the hidden states, which dropped to unrealistic levels, as shown in Figure 7.4.
In order to get a better insight in the applicability of DA, the model was calibrated. A visual inspection showed improved values for the assimilated water level compared to the DA set up with the original model and compared to the set up without assimilation. More so, the hidden states now showed a pattern as would be expected. The temporary drop in storage in the states of the greenhouse and unsaturated zone could be linked to the corresponding drop in water level, and remained within a realistic range of the modelled values without assimilation. Furthermore, the hidden states increased in storage again when the water level was adjusted upwards by the EnKF. This resulted at the end of the run in values higher than the results produced without assimilation, which would be expected because of the underestimation of the real time rainfall during extreme events.
Figure 6.9b shows that the calibrated model still contained the problem of the time-lag. The difference in time-lag is assumed to be a model-based problem instead of an observation problem because the comparison of rain or no rain of the uncorrected and day-corrected radar rainfall products was rather good. The influence of the time lag in the case study on the calibrated model can be seen in Figure 6.9b and can be explained as followsS: First the occurrence of a rainfall peak caused the modelled water level to exceed the observations. The Kalman gain became negative and the states were all lowered instead of filled. After the time lag, the observations exceeded the modelled water levels. However, since the rainfall had already stopped by that time, the variance

in the ensemble spread (σ_m^2) remained small, meaning the factor

$$\frac{\sigma_m^2}{\sigma_m^2 + \sigma_{obs}^2} = \frac{1}{1 + \frac{\sigma_{obs}^2}{\sigma_m^2}} \tag{7.3}$$

in the Kalman gain remained small too, such that K would become small, reducing the effectiveness of the updates.
Two ideas are proposed here to reduce the inhibiting effect the time lag is expected to have on the assimilation. The feasibility of the options and a small example are discussed here. The first is most straightforward and should always be favoured; improve the knowledge of the system and incorporate this into a better RR model giving the appropriate time-lag in the system. A parameter that is expected to create unrealistic fast modelled runoff is the greenhouse storage of the lowest class. This class implements greenhouse areas with a storage capacity of less than 500 m^3/ha. The reservoir is modelled without storage creating direct runoff. This greenhouse class takes up a significant 35 % of the total greenhouse area, while it is even obligated by law for greenhouse farmers to have a storage of at least 500 m^3/ha (Keizer, 2017). The model could already be improved by setting the are of the lowest class to zero and implementing this in the greenhouse class of basins with a storage between 500 and 1000 m^3/ha. Note that implementing this measure would reduce the magnitude of the generated response. Another component determining the fast runoff during large rainfall events is the concentration time of the water on the paved surface to the sewage. This is now modelled as 0.1 [1/min], but this could be increased. Last, the unsaturated zone is said not to hold water long enough (Beukema, 2017), which would result in too fast runoff.
The second proposed approach to reduce the problem of the time-lag for data assimilation is by implementing a larger time increment in the DA. If the timespan between assimilation steps is larger, the effect of the rainfall event on the observation can already be measured and the responses could be compared better.
The effectiveness of the case studies were difficult to determine because there was no possibility of validation for the other hidden states. A recommended approach to improve the case study is by collecting water levels at different locations within the same polder to create a more representative water level observation for the whole polder. Furthermore, very detailed information on some of the Greenhouse basins were collected in 2017 for the master thesis of Keizer (2017). The measurement devices to measure the water level in the storage basins are cheap or even already in place. Collecting data of the greenhouse storages would allow validation of the assimilated state of the storage in the greenhouse area. Furthermore, collecting and processing data of greenhouse storages will give a better understanding of the discharges of the basins to the open water level.

8

Conclusions and Recommendations

The chances of finding out what is really going on in the universe are so remote, the only thing to do is hang the sense of it and keep yourself occupied.

– Douglas Adams, The Hitchhikers Guide to the Galaxy

The individual research questions are first discussed in this chapter, after which an answer will be formulated to the main research question. The chapter will be concluded with recommendations for the direction of future research and practicalities to allow the implementation of the data assimilation. The aim of the research was to answer the main research question:

'How can data assimilation methods be used in a real time hydrological models to reduce water level prediction uncertainty?'

by answering the following sub-questions:

1. *Which model components have the largest influence on the uncertainty in water level predictions; parameter values, initial states or forcing data?*
2. *Are there technical or practical issues for the application of DA?*
3. *Which model components can best be perturbed for DA, such that optimal state estimates are found?*
4. *Can DA, with a perfect model, recover the right initial states while running with distorted forcing data?*
5. *Can prediction uncertainty be reduced using DA in a case study where many different components contribute to the uncertainty in water level predictions?*

As an answer to the first sub-question, this thesis demonstrated that the largest source of uncertainty for the predicted water levels is the rainfall component. Firstly, because the water level was most sensitive to errors in the observed rainfall, which was represented by a rainfall multiplier. Secondly, because for real time control, the rainfall product is limited to the uncorrected product, which contained large errors. The resulting prior distribution on the uncorrected radar rainfall product, found through an analysis between the uncorrected and 2nd corrected radar rainfall products, was $\mathcal{N}(1.4, 1.3)$. This error causes a modelled water level of only a third of the water level generated by the 2nd corrected radar product. Errors in other model components, of which mostly the groundwater and paved states, influenced water level prediction uncertainties as well.
To answer the second sub-question, one technical issue remained, caused by the influence of the pumps on the assimilation. The pumps disturb the assimilation when the pumps state is opposite of that of the observation. The pumps were not implemented as state in the DA set up, because of the technical limitation that they were not included in the ascii

file created to alter initial states in Sobek RR for this thesis. Furthermore, assimilating the pump state would impose an integer problem because DA can only deal with continuous states. To prevent this behaviour from obscuring the results of the theoretical case, the pumps were removed for the twin experiment.
As an answer tot the third sub-question, the results of the twin experiments showed that the technical set up where the rainfall is perturbed, produced better results than for the set up with additive noise to all states, additive noise to the groundwater or a single multiplication factor for the states. The assimilation with rainfall multiplier showed not only for the water level, but also for the hidden states a significant improvement over the case without assimilation. The approach to assume the main uncertainty to be caused by errors in rainfall observation is justified by the answer to the first sub-question. Furthermore, this method has as conceptual advantage that it is exactly known which error is being targeted. The NS_p values of the assimilated results of the water level were not bounded by the algorithm set up or the prior, but the uncertainty of the observations, set at 1.5 cm. For all set ups for the twin experiment, the NS_p values for the assimilated water level scored between 0.92 and 0.97, suggesting all methods to work. This leads to a note of caution when applying DA techniques; it is important to check the physical feasibility of the updated state of the model. To illustrate this, the set up with rainfall multiplier showed visual and objective improvements for the assimilated hidden states, with an average of the NS_p values 0.6. However, The other DA set-ups showed for some states unrealistic model behaviour and the average NS_p values were negative for the three approaches showing the assimilation results increased model prediction uncertainty for the hidden states compared to a situation without assimilation.
To answer the fourth sub-question, data assimilation *can* recover the right initial states when implementing the set up of the twin experiment with the rainfall multiplier and a perfect model. When implemented with a time increment of 15 minutes and an ensemble size of 16, good results for all states were generated within an acceptable computation time of twenty seconds per assimilation step, well within the bounds of the fifteen minutes of the time increment.
The answer of the fifth and final sub-question is that DA works in a case study, but only when the model showed a similar pattern to the observations. The results of the original case study were clouded by large differences in response time, the number of times the pump had to turn on and the amplitude of the modelled and observed water levels in the Tedingerbroek polder. The maximal observed water level was 0.3 meter, while the maximum modelled water level after during calibration was 0.06 m. To improve the model, it was recalibrated for the landuse areas, concentration time and pump capacity, the parameters expected to reduce the difference between model and observation most. Within a reasonable bound of uncertainty for the parameters, the modelled water levels could not mimic the pattern of the water level observations, resulting in the current belief that the Tedingerbroek polder receives additional water fluxes, or that the water level observation taken near a pump are not representative enough for the polder. However, when allowing larger deviations from the original parameters, the calibrated model could mimic the maximal modelled water level and the pump pattern. The remaining time lag disturbed the DA application, but the assimilated results of water level, with a NS_p of 0.67, showed an improvement and the assimilated hidden states gave results that corresponded with the expected pattern.
The main question can be answered using the answers on the previous question. It is indeed possible to use data assimilation methods to reduce the influence of the error in precipitation forcing data on the predicted water levels. The corresponding initial states were deduced from the observed water levels in the theoretical twin experiment. However, the assimilation was not directly applicable to the test case considered in

this thesis because the model did not approximate the observations enough. After recalibration of the model, the assimilated results improved, showing that with a model that can predict the observation relatively well, DA can be used to update the model states and reduce water level prediction uncertainty.
Future research should be focussed on methods to validate the updated states for a case study. The suggested approach is to collect time series of the storages in the greenhouse basins. Keizer (2017) started a project with Deltares to collect this data, which would complement this study well. Furthermore, the representativeness of the observed water levels should be assessed by using multiple locations for measurements within one polder. Whether pumps are turned on or off should be implemented as an observation with zero uncertainty in OpenDA. To make this technically possible, the request is to add pump information in the Ascii restart file for Sobek. This should then be implemented in FEWS such that the ensemble members receive additional observations of the pump settings which can be updated directly. A practical recommendation for Delfland would be to multiply the uncorrected radar rainfall product $P = 0.45P_{unc}^2 + 1.5P_{unc}$ in advance, to account for the consistent underestimation of the rainfall, especially for high rainfall intensities.
To conclude, considering the trend of water authorities in the Netherlands to improve the operation of the pumping, this research showed that data assimilation has the potential to be a method applicable to real time water systems. It is therefore relevant to further research methods for validation and implementation in real-case studies.

Bibliography

Bemporad, A. (2011). Lecture notes: Model Predictive Control, Stanford University. pages 1–78.

Beukema, P. (2017). Interview besturingings systeem Delfland met BOS1.0.

Borup, M., Grum, M., Madsen, H., and Mikkelsen, P. S. (2015). A partial ensemble Kalman filtering approach to enable use of range limited observations. *Stochastic Environmental Research and Risk Assessment*, 29(1):119–129.

Brown, J. D. (2010). Prospects for the open treatment of uncertainty in environmental research. *Progress in Physical Geography*, 34(1):75–100.

Bulygina, N., Mcintyre, N., and Wheater, H. (2011). Bayesian conditioning of a rainfall-runoff model for predicting flows in ungauged catchments and under land use changes. 47(November 2010):1–13.

Carpenter, T. M. and Georgakakos, K. P. (2004). Impacts of parametric and radar rainfall uncertainty on the ensemble streamflow simulations of a distributed hydrologic model. *Journal of Hydrology*, 298(1-4):202–221.

Chen, Z. H. E. (2003). Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond. *Statistics*, 182(1):1–69.

Copp, D. and Hespanha, J. P. (2014). Nonlinear Output-Feedback Model Predictive Control with Moving Horizon Estimation. *Proc. of the 53nd Conf. on Decision and Contr.*, pages 1–17.

Dahm, R. D., Bruggers, M. D., Clevering-Loeffen, P. G., and Kamermans, J. G. (2010). Energieverbruik van het Nederlandse waterbeheer. *H2O*, (17):20–21.

Daum, F. (2005). Nonlinear filters: beyond the Kalman filter. *Aerospace and Electronic Systems Magazine, IEEE*, 20(8):57–69.

Delfland (2017). Website Midden Delfland, http://delfland.middendelfland.net/.

Dumedah, G. and Coulibaly, P. (2013). Evaluating forecasting performance for data assimilation methods: The ensemble Kalman filter, the particle filter, and the evolutionary-based assimilation. *Advances in Water Resources*, 60:47–63.

Evensen, G. (1994). Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research*, 99(C5):10143–10162.

Evensen, G. (2003). The Ensemble Kalman Filter: Theoretical formulation and practical implementation. *Ocean Dynamics*, 53(4):343–367.

Gemeente Rotterdam (2017). Waterveiligheid gemeente Rotterdam. http://www.rotterdam.nl/waterveiligheidinrotterdam.

Gharamani, Z. (2001). An introduction to Hidden Markov Models and Bayesian Networks. *International journal of Pattern Recognition and Artificial Inteligence*, 15(1):9–42.

Grigg, N. (2005). Water Resources Management. In *Water Encyclopedia*, number 2, pages 586–587.

Hairer, M., Stuart, A. M., and Voss, J. (2005). A Bayesian Approach to Data Assimilation. pages 1–20.

Hansen, L. S., Borup, M., Møller, A., and Mikkelsen, P. S. (2014). Flow forecasting using deterministic updating of water levels in distributed hydrodynamic urban drainage models. *Water (Switzerland)*, 6(8):2195–2211.

Hazeu, G. (2005). Landelijk Grondgebruiks-bestand Nederland (LGN5). *Geo-Info*, 2(10):456–462.

Ho, J. Y. and Lee, K. T. (2015). Grey forecast rainfall with flow updating algorithm for real-time flood forecasting. *Water (Switzerland)*, 7(5):1840–1865.

Hoogheemraadschap van Delfland. www.hhdelfland.nl (2017). Website Hoogheemraadschap van Delfland.

Huard, D. and Mailhot, A. (2006). A Bayesian perspective on input uncertainty in model calibration: Application to hydrological model 'abc'. *Water Resources Research*, 42(7):1–14.

Jin, X., Xu, C. Y., Zhang, Q., and Singh, V. P. (2010). Parameter and modeling uncertainty simulated by GLUE and a formal Bayesian method for a conceptual hydrological model. *Journal of Hydrology*, 383(3-4):147–155.

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35.

Katzfuss, M., Stroud, J. R., and Wikle, C. K. (2016). Understanding the Ensemble Kalman Filter. *THE AMERICAN STATISTICIAN*, 70(4):350–357.

Kavetski, D., Franks, S. W., and Kuczera, G. (2002). Confronting input uncertainty in environmental modelling. *Science*, 6:49–68.

Keizer, K. (2017). *Exploring the hydrological response of greenhouse reservoirs*. Master thesis, Technical University of Delft.

Kuczera, G., Kavetski, D., Franks, S., and Thyer, M. (2006). Towards a Bayesian total error analysis of conceptual rainfall-runoff models: Characterising model error using storm-dependent parameters. *Journal of Hydrology*, 331(1-2):161–177.

Lee, J. H. (2011). Model predictive control: Review of the three decades of development. *International Journal of Control, Automation and Systems*, 9(3):415–424.

Liu, Y., Weerts, A. H., Clark, M., Hendricks Franssen, H. J., Kumar, S., Moradkhani, H., Seo, D. J., Schwanenberg, D., Smith, P., Van Dijk, A. I. J. M., Van Velzen, N., He, M., Lee, H., Noh, S. J., Rakovec, O., and Restrepo, P. (2012). Advancing data assimilation in operational hydrologic forecasting: Progresses, challenges, and emerging opportunities. *Hydrology and Earth System Sciences*, 16(10):3863–3887.

Lobbrecht, A. H.-I., Clemens, F. T. D. W., and Einfalt, T. h. (2012). Internationaal neerslagradar- composiet gereed. *H2O*, (9):24–25.

Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814.

Moradkhani, H., Sorooshian, S., Gupta, H. V., and Houser, P. R. (2005). Dual state-parameter estimation of hydrological models using ensemble Kalman filter. *Advances in Water Resources*, 28(2):135–147.

N. K. Ajami, Q. Duan, S. S. (2007). An integrated hydrologic Bayesian multimodel combination framework: Confronting input, parameter, and model structural uncertainty in hydrologic prediction. *Water Resources*, 43:1–19.

Nelen & Schuurmans and Delfland (2011). Uitgangspuntennotitie modelinstrument, Boezemmodel Delfland (BOEZ09). Technical report, Nelen & Schuurmans, Delft.

Nie, S., Zhu, J., and Luo, Y. (2011). Simultaneous estimation of land surface scheme states and parameters using the ensemble Kalman filter: Identical twin experiments. *Hydrology and Earth System Sciences*, 15(8):2437–2457.

OpenDA-Association (2016). OpenDA User Documentation. Technical report, Deltares, Delft.

Orloff, J. and Bloom, J. (2014). Bayesian Updating with Continuous Priors. In *Introdcution to Probability and Statistics*, chapter 13a, pages 1–9. MIT open courseware.

Pappenberger, F. and Beven, K. J. (2006). Ignorance is bliss: Or seven reasons not to use uncertainty analysis. *Water Resources Research*, 42(5):1–8.

Pedersen, J. W., Lund, N. S. V., Borup, M., Löwe, R., Poulsen, T. S., Mikkelsen, P. S., and Grum, M. (2016). Evaluation of Maximum a Posteriori Estimation as Data Assimilation Method for Forecasting Infiltration-Inflow Affected Urban Runoff with Radar Rainfall Input. *Water*, 8(381).

Prinsen, G. (2013a). Sobek user manual. Technical report, Deltares, Delft.

Prinsen, G. (2013b). Technical Reference Manual SOBEK Rainfall Runoff. Technical report, Deltares, Delft.

Rawlings, J. B. (2009). State Estimation using Moving Horizon Estimation and Particle Filtering. *UW Math Probability Seminar*, pages 1–45.

Rawlings, J. B. and Bakshi, B. R. (2006). Particle filtering and moving horizon estimation. *Computers and Chemical Engineering*, 30(10-12):1529–1541.

Renard, B., Kavetski, D., Kuczera, G., Thyer, M., and Franks, S. W. (2010). Understanding predictive uncertainty in hydrologic modeling: The challenge of identifying input and structural errors. *Water Resources Research*, 46(5):1–22.

Schaake, J., Franz, K., Bradley, A., and Buizza, R. (2006). The Hydrologic Ensemble Prediction EXperiment (HEPEX). *Hydrology and Earth System Sciences Discussions*, 3(5):3321–3332.

Sivia, D. (1996). *Data Analysis: A Bayesian Tutorial*. Clarendon Press, Oxford.

Tukey, J. W. (1962). The Future of Data Analysis. *The Annals of Mathemetical Statistics*, 33(1):1–67.

Volkmann, T. H. M., Lyon, S. W., Gupta, H. V., and Troch, P. A. (2010). Multicriteria design of rain gauge networks for flash flood prediction in semiarid catchments with complex terrain. *Water Resources Research*, 46(11):1–16.

Vrugt, J. A., Gupta, H. V., Nualláin, B., and Bouten, W. (2006). Real-Time Data Assimilation for Operational Ensemble Streamflow Forecasting. *Journal of Hydrometeorology*, 7(3):548–565.

Wang, Y. and Boyd, S. (2010). Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, 18(2):267–278.

Wesseling, J. G. (1991). CAPSEV: Steady state moisture flow theory Program description - User manual. Technical report, The WINAND STARING CENTRE, Wageningen (The Netherlands).

Xu, C., Tunemar, L., Chen, Y. D., and Singh, V. (2006). Evaluation of seasonal and spatial variations of lumped water balance model sensitivity to precipitation data errors. *Journal of Hydrology*, 324(1-4):80–93.

Zhang, X. L., Su, G. F., Yuan, H. Y., Chen, J. G., and Huang, Q. Y. (2014). Modified ensemble Kalman filter for nuclear accident atmospheric dispersion: Prediction improved and source estimated. *Journal of Hazardous Materials*, 280(September):143–155.

Appendices

A

Deterministic Rainfall Run-off model

The CRR set-up for a polder as used in Sobek RR is taken for the deterministic model in this research. Sobek uses the distinction of four different land-use types: paved, unpaved, greenhouse and open water. Within these areas, different hydrological processes are dominant such that their model structure is different. The next sections explains the model set up and settings per land-use type.

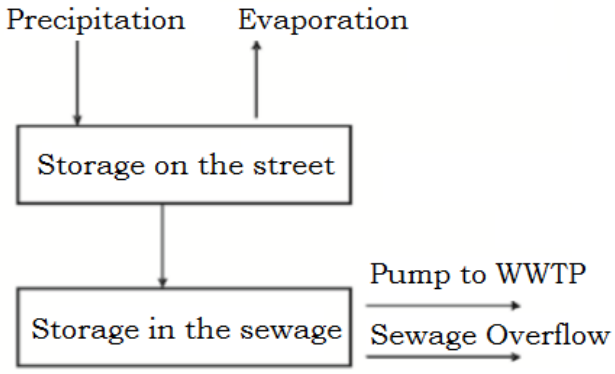


Figure A.1: Conceptual representation of the paved areas (adjusted from Prinsen, 2013b).

A.1 Paved Area

A schematic representation of the model
set up for paved area is given in Figure A.1. It consists of two storage buckets; one for storage on the street one for storage in the sewage pipes. An overview of the processes, water balance and fluxes are given in Table A.1.

Storage on the street

The storage on the street has as input **precipitation (P)**, calculated as the *area* multiplied by the rainfall intensity and time-step. The **evaporation (E)** takes place between 7.00 and 19.00 and equals the potential evaporation when storage on the street is present. As soon as the *maximal storage on the street* $S_{street,max}$ is reached, the surplus water **drains to the sewage** ($\mathbf{Q_{sew}}$). This is modelled with a delay using the rational method given by

$$Q_{sew} = c \cdot h \cdot A_{PA}, \tag{A.1}$$

where

Q in L^3/T is the flow to the sewage

c in T^{-1} is the runoff factor

h in L is the hight of the water on the surface exceeding the maximum storage on the surface

A_{PA} in L^2 is the total paved area

For illustration purposes: a runoff factor of $c = 0.1$ min^{-1} would mean that 10% of the excess surface volume reaches the sewage in one minute. Note that this implies that the storage on the street can be higher than the $S_{street,\,\max}$ would indicate.

Process	Water Balance	Constitutive Relationships
Street Interception	$\frac{dS_{\text{street}}}{dt} = P - E_a - Q_{sew}$	$E_a = min\left(E_{pot} \cdot, S_{street}/dt\right)$ $Q_{wwtp} = min\left(\frac{S_{\text{sew}}}{dt}, Q_{wwtp,\text{max}}\right)$
Sewage	$\frac{dS_{\text{sew}}}{dt} = Q_{sew} - Q_{wwtp} - Q_{so}$	$Q_{so} = max\left(\frac{S_{\text{sew}} - S_{\text{sew,max}}}{\Delta t} + Q_{sew} - Q_{wwtp}, 0\right)$ $Q_{sew} = max\left(((S_{\text{street}} - S_{\text{street,max}} + P - E) \cdot A_{PA}) \cdot c, 0\right)$

Table A.1: Processes, water balance and fluxes for the model structure of the paved areas

Storage in the sewage

The storage in the sewage has as only input (Q_{sew}), assuming a dry weather flow of zero. Outgoing fluxes are the **discharge to the waste water treatment plant** ($\mathbf{Q_{wwtp}}$) and the **sewage overflow** $\mathbf{Q_{SO}}$. As long as there is storage in the sewage, $\mathbf{Q_{wwtp}}$ equals *the maximum sewage capacity* $Q_{sew,max}$. When the *maximal storage in the sewage* $S_{\text{sew,max}}$ is exceeded , additional inflow discharges directly to open water via **sewage overflow** $\mathbf{Q_{SO}}$. The sewage is assumed to have no leakages.

A.2 Greenhouse Area

The conceptualization used for greenhouse areas presented here was developed by Alterra and Deltares in the 90s (Prinsen, 2013b). As shown in Figure A.2, only one kind of storage bucket is in place, the storage basin. These storage basins are divided into ten classes, which are defined by the volume of water that can be stored per hectare of greenhouse area. An overview of the class intervals (between < 500 and $5000 - 6000$ m^3/ha) is given in Figure A.3. An overview of the processes, water balance and fluxes are given in Table A.2.

Storage in Basins

The input flux is **precipitation**. The outward fluxes are **evaporation, outtake for watering the greenhouses** ($\mathbf{Q_{GH}}$) and **overflows to open water** ($\mathbf{Q_{OF}}$).
The precipitation input to the basin is modelled as the *total greenhouse area* A_{GH} multiplied with the rainfall intensity and the time step. The evaporation is taken as the potential evaporation between 7.00 and 19.00, but limited to the surface of the storage basins. *The storage per basin* $S_{basin,i}$ is taken as the lowest bound of the greenhouse class. Note that this implies that the lowest class (<500 m^3/ha) is modelled without storage or time lag.

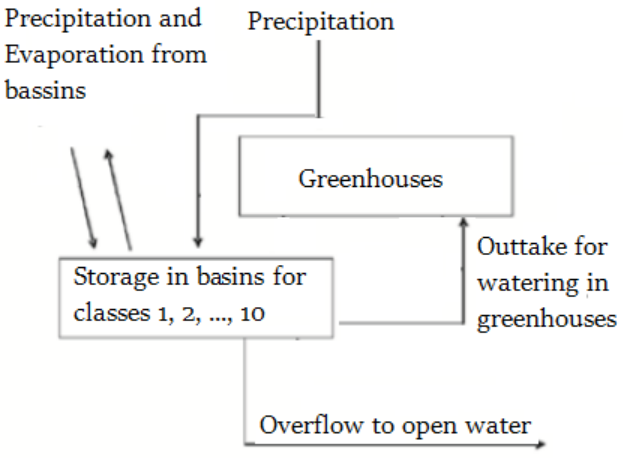


Figure A.2: Conceptual representation of the greenhouse areas (adjusted from Prinsen, 2013b).

When the depth in the basin exceeds the typical depth after extraction of the evaporation and the outtake for watering of the plants, the surplus overflows to open water. The water used in the greenhouses is taken as a constant timeseries, which are compared to

Process	Water Balance	Constitutive Relationships
Storage Basins	$\frac{dS_{\text{gh,i}}}{dt} = P - E - Q_{GH} - Q_{OF}$	$Q_{OF} = \max\left(\frac{S_{\text{gh,i,max}} - S_{\text{gh,i}}}{\Delta t}, 0\right)$
		$Q_{GH} = \begin{cases} \min\left(Q_{GH,\text{input}}, \frac{S_{\text{gh,i}} - 0.1 \cdot S_{\text{gh,i,max}}}{\Delta t}\right) & \text{for } S_{\text{gh,i}} > 0.1 \cdot S_{\text{gh,i,max}} \\ 0 & \text{for } S_{\text{gh,i}} < 0.1 \cdot S_{\text{gh,i,max}} \end{cases}$

Table A.2: Processes, water balance and fluxes for the model structure of greenhouse area

Greenhouse class	Storage per area (m^3/ha)	Percentage of paved area (%)
1	<500	34
2	500 - 1000	19
3	1000 - 1500	12
4	1500 - 2000	7
5	2000 - 25000	5
6	2500 - 3000	5
7	3000 - 4000	5
8	4000 - 5000	7
9	5000 - 6000	6
10	> 6000	0

Table A.3: Distribution of greenhouse classes

the basin storage which needs to retain a minimum filling percentage of 10 percent. No underground storage or storage in silo's is taken into account.

A.3 Unpaved Area

To model the unpaved areas (in rural and urban areas), three storage basins are taken into account (Figure A.3), the surface storage, storage in the unsaturated zone and the shallow groundwater storage. The water balance is given by Table A.4, but the constitutive relationships for Q_{US}, ET_a and Q_D are explained below since they entail some further elaboration. For the unpaved area; the groundwater calculations are applied over the total area of paved and unpaved areas, while the rainfall and evaporation only depend on the size of the unpaved surfaces.

Storage on the land surface

The incoming flux on the land surface is the precipitation. From the surface the water either **infiltrates** ($\mathbf{Q_I}$) or evaporates. Infiltration is limited to the *maximum infiltration capacity* (I_{max}) and the available water on the surface. When the precipitation intensity

Process	Water Balance	Constitutive Relationships
Storage on Land	$Q_{SR} = max\left(S_{\text{land}} - S_{\text{land,max}}, 0\right)/\Delta t$	
Unsaturated Zone	$\frac{dS_{US}}{dt} = Q_I - T - Q_{US}$	$Q_I = max\left(I_{max}, \frac{S_{\text{land}}}{\Delta t}\right)$
Shallow Groundwater	$\frac{dS_{SG}}{dt} = Q_{US} - Q_D - Q_P$	

Table A.4: Processes, water balance and fluxes for the model structure of the unsaturated zone

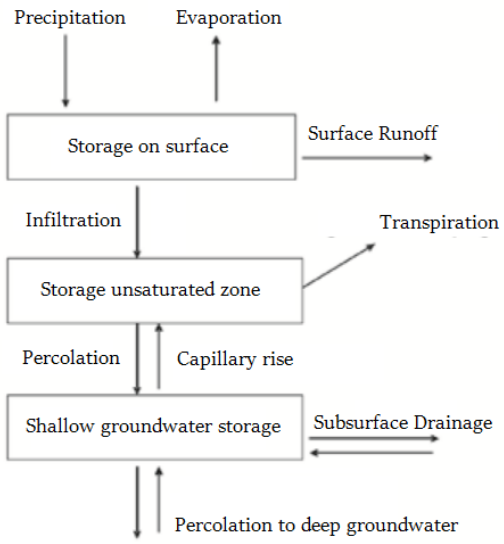


Figure A.3: Conceptual representation of the unpaved area (adjusted from Prinsen, 2013b).

exceeds the infiltration capacity or when the groundwater level exceeds the surface level, the storage on the surface increases. Once the *maximum storage on land* ($S_{land,max}$) is exceeded, any additional water is directly routed to the open water bodies.

Storage in unsaturated zone

The unsaturated zone (or root zone) can receive water either from infiltration or capillary rise from the shallow groundwater. Water leaves the storage unit through transpiration and percolation.

Capillary rise ($\mathbf{q_{CR}}$) takes place when the water content in the root zone is less than the *equilibrium moisture storage* (V_{eq}), defined as a water deficit. When the storage in the unsaturated zone exceeds the equilibrium, **percolation** ($\mathbf{q_P}$) occurs . The water is assumed to reach the groundwater (via percolation) or root zone (via capillary rise) within the same time step. The following scheme is used to calculate the flux q between the unsaturated zone and the shallow groundwater:

$$q = \begin{cases} q_p = \frac{V_{eq} - (V_{t-1} + \Delta V)}{\Delta t}, & \text{for } V_{pot} > V_{eq} \quad \text{Rootzone too wet, percolation} \\ q_{CR,act} = q_{CR,pot} & \text{for } V_{pot} < V_{eq} \text{ and } V_{act} < V_{pF3} \quad \text{Rootzone too dry, capillary rise} \\ q_{CR,act} = q_{CR,pot} \frac{V_{eq} - V_{act}}{V_{eq} - V_{pf3}} & \text{for } V_{pot} < V_{eq} \text{ and } V_{act} > V_{pF3} \quad \text{Rootzone much too dry, CR} \end{cases} \tag{A.2}$$

To solve this equation, some elements need to be defined. First, *the potential water content in the root zone* (V_{pot}) at time t:

$$V_{pot} = V_{t-1} + (P - E_a) \cdot \Delta t \tag{A.3}$$

Second and third, *the equilibrium moisture storage* (V_{eq}) and *the potential capillary rise* ($Q_{CR,pot}$). These change over time and are depended on the following factors:

1. $V_{eq} = f($ soil type, groundwater table (t), thickness root zone$)$
2. $q_{CR,pot} = f($ drainage resistance, groundwater table $(t))$

These values are predetermined for different inputs using CAPSEV, a program developed by Wesseling (1991). For different conditions, V_{eq} and $q_{CR,pot}$ are available from CAPSEV in a table, which is used as input in Sobek.

Last but not least, the amount of **Transpiration (T)** needs to be calculated. T depends on the amount of moisture in the root zone, *the potential evapotranspiration* ET_{pot} and *the limiting factor* α_E, which in its turn is a function of *the soil water pressure head* h_i . The T is calculated as

$$ET_a = ET_{pot} \cdot \alpha_E, \tag{A.4}$$

where α_E is determined by

$$\alpha_E = \begin{cases} 0 & \text{if } 0 < V/V_{eq} < h_4 \quad \text{Root zone much too dry, no } ET_a \\ 1 - (x - h_{3i})/(h_4 - h_{3i}) & \text{if } h_4 < V/V_{eq} < h_{3i} \quad \text{Root zone dry, reduced } ET_a \\ 1 & \text{if } V/V_eq > h_{3i}. \quad \text{Root zone saturated, } ET_a = E_{pot} \end{cases} \tag{A.5}$$

The factors h_i depend on the root zone thickness, the crop and soil type and are calculated using CAPSEV. The *'reduction point'* h_3 is also dependent on the *potential evaporation* ET_{pot} and can for each situation obtain different values:

$$h_{3i} = \begin{cases} h_{3High} & \text{for } ET_{pot} > 5 \text{ mm/d} \\ h_3 Low & \text{for } ET_{pot} < 1 \text{ mm/d} \end{cases} \tag{A.6}$$

Between the potential evaporation values, a linear interpolation between the two values for h is made.

Shallow groundwater storage

The shallow groundwater drains to the open water using Ernst equation (Prinsen, 2013b) given by:

$$q_D = \frac{\Delta h}{R}, \tag{A.7}$$

where Δh [m] is the difference between the groundwater level and the water level and R is *the resistance factor* [day]. R depends on *the drainage level* d [m below surface]:

$$R = \begin{cases} 0.3 & \text{if } d < 0 \text{ and } \Delta h > 0 \\ 70 & \text{if } 0 \leq d < 0.3 \text{ and } \Delta h > 0 \\ 200 & \text{if } d > 0.3 \text{ and } \Delta h > 0 \\ 1300 & \text{if } \Delta h < 0 \end{cases} \tag{A.8}$$

The interaction with deeper groundwater is implemented as a fixed time series.

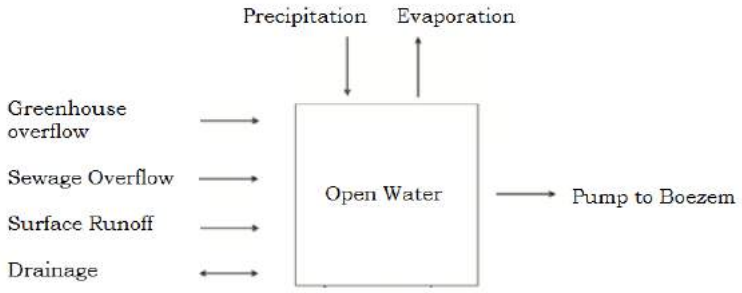


Figure A.4: Conceptual representation of the open water

A.4 Open Water

Open Water is modelled as a very simple storage box with water balance:

$$\frac{S_{OW}}{dt} = (P - E_{pot}) \cdot A_{OW} + Q_{SO} + Q_{SR} + Q_D + Q_{OF} - Q_{pump}, \tag{A.9}$$

as represented in Figure A.4, with A_{OW} *the area of the open water bodies* and Q_{pump} the discharge to the boezem.
The pumps responsible for Q_{pump} are regulated on zero/half/full capacity and switch on or off at certain reference levels in order to keep the water to a set level (peil). The mechanisms are provided in equation A.10 and depend on the water level difference from the set water level (Δh_{ref}) and the previous pump settings:

$$Q_{pump}(t) = \begin{cases} Q_{pump,max} & \text{if } \Delta h_{ref}(t) > 0.2 \\ & \text{or } Q_{pump}(t-1) = Q_{pump,max} \ \& \ \Delta h_{ref}(t) > 0 \\ 0.5 \cdot Q_{pump,max} & \text{if } 0.1 < \Delta h_{ref}(t) < 0.2 \\ & \text{or } Q_{pump}(t-1) = 0.5 \cdot Q_{pump,max} \ \& \ \Delta h_{ref}(t) > -0.1 \\ 0 & \text{otherwise} \end{cases} \tag{A.10}$$

B

Code for coupling Sobek RR and OpenDA

B.1 Exchange of forcing data

The code that was written for the exchange of forcing data can be found in **SobekRRFile.java, SobekRRExchangeitem.java and SobekRRTimeStepExchangeItem.java** SobekRRFile.java initializes by reading out the precipitation file from Sobek and creates exchange items of the class SobekRRExchangeItems (defined in the corresponding SobekRRExchangeitem.java). These contain the corresponding ID of the weather station, the times and the values of the precipitation. Furthermore the exchange item containing the timestep of the data is created. After the OpenDA calculations, the values in the exchange item of the precipitation are perturbed. The module SobekRRFile.java then reloads them in the finalize part and writes an identical text file (.BUI) as the precipitation file that it read in, but with the perturbed rainfall values at the times corresponding to the DA step. This file is loaded into Sobek as the new precipitation event.

```
package nl.deltares.openda.models.sobek;

import nl.wldelft.util.DateUtils;
import nl.wldelft.util.io.LineReader;
import nl.wldelft.util.io.LineWriter;
import org.openda.exchange.timeseries.TimeUtils;
import org.openda.interfaces.IDataObject;
import org.openda.interfaces.IExchangeItem;
import org.openda.interfaces.IPrevExchangeItem;
import org.openda.utils.Time;

import java.io.File;
import java.io.IOException;

import java.text.NumberFormat;
import java.text.ParseException;


import java.util.*;

/**
 * Created by petraizeboud on 07/02/2017.
 */
public class SobekRRFile implements IDataObject {

   LinkedHashMap<String, IExchangeItem> exchangeItems = new LinkedHashMap<>();

   //The variable that has the restart file that is read & rewritten
   private File buistations;
```

```
private static final String FIRST_6_LINES = "*Name of this file:
↪  \\SBKRural\\FIXED\\DT458_1A.BUI\r\n" +
      "*Date and time of construction: 25-02-2016   11:54:05\r\n" +
      "*Enige algemene wenken:\r\n" +
      "*Gebruik de default dataset (1) of de volledige reeks (0) voor overige
      ↪  invoer\r\n" +
      " 1\r\n" +
      "*Aantal stations";
private static final String SECOND_1_LINES = "*Namen van stations";
private static final String THIRD_2_LINES = "*Aantal gebeurtenissen (omdat het
↪  1 bui betreft is dit altijd 1)\r\n" +
      "*en het aantal seconden per waarnemingstijdstap";
private static final String FOURTH_4_LINES = "*Elke commentaarregel wordt
↪  begonnen met een * (asteriks).\r\n" +
      "*Eerste record bevat startdatum en -tijd, lengte van de gebeurtenis in
      ↪  dd hh mm ss\r\n" +
      "*Het format is: yyyymmdd:hhmmss:ddhhmmss\r\n" +
      "*Daarna voor elk station de neerslag in mm per tijdstap.";
//System.lineSeparator() can be used instead of \r\n, \r is for Unix systems
↪  \r\n for windows.


@Override
public String[] getExchangeItemIDs() {
   return exchangeItems.keySet().toArray(new
    ↪  String[exchangeItems.keySet().size()]);
}

@Override
public String[] getExchangeItemIDs(IPrevExchangeItem.Role role) {
   return new String[0];
}

@Override
public IExchangeItem getDataObjectExchangeItem(String exchangeItemID) {
   return exchangeItems.get(exchangeItemID);
}

@Override
public void finish() {

   String splitline;
   //Loop over exchange items to save id's, time and values
   List<String> idList = new ArrayList<>();
   Collection<IExchangeItem> values = exchangeItems.values();
   double[][] valuesDoubles = new double[values.size()][];
   double[] times = null;
   int i = 0;
   for (IExchangeItem exchangeItem : values) {
      if (exchangeItem instanceof SobekRRTimeStepExchangeItem) continue;
      valuesDoubles[i] = exchangeItem.getValuesAsDoubles();
      times = exchangeItem.getTimes();
      String id = exchangeItem.getId();
      idList.add(id);
      i++;
   }
```

```
//Calculate timestep, length of timeseries & dateline
int timestep = (int) (TimeUtils.mjdInMinutes(times[1] - times[0])) * 60;
 ↪  //# seconds in timestep
int nmbtimestepts = valuesDoubles[0].length;

//Convert mjd to  yyyy mm dd hh mm ss  dddd hh mm ss
String time0 = TimeUtils.mjdToString(times[0]); //in mjd
StringBuilder splitdate = new StringBuilder(35);
splitdate.append(" " + time0.substring(0, 4));
// Split the string 201001010000 to 2010 1 1 0 0 0
for (int k = 4; k < time0.length() - 1; k = k + 2) {
   String substring = time0.substring(k, k + 2);
   String sub = substring.substring(0, 0);
   String sub2 = substring.substring(0, 1);
   // Translate the 01 to " 1"
   if (substring.substring(0, 1).equals("0")) {
      substring = substring.substring(1);
      splitdate.append(" " + substring);
   } else {
      splitdate.append(" " + substring);
   }
}
// to find # days, hours, minutes, seconds to put in string splitetimelength
double totalseconds = (nmbtimestepts) * (timestep);
int days = (int) Math.floor(totalseconds / (60 * 60 * 24));
int hours = (int) Math.floor((totalseconds / (60 * 60 * 24) - days) * 24);
int min = (int) Math.floor((((totalseconds / (60 * 60 * 24) - days) * 24)-
 ↪  hours)*60);
int sec = (int) Math.floor((((((totalseconds / (60 * 60 * 24) - days) * 24)-
 ↪  hours)*60)-min))*60;
splitdate.append(" 0 " + Integer.toString(days) + " " +
 ↪  Integer.toString(hours) + " "
     + Integer.toString(min) + " " + Integer.toString(sec));
// The first zero in splitdate is to account for zero seconds.

//Rewrite the new buistations file for Sobek
try (LineWriter lineWriter = new LineWriter(buistations)) {
   //lineWriter.writeLine(String.valueOf(doubles));
   lineWriter.writeLine(FIRST_6_LINES);
   int nmbstations = idList.size();
   lineWriter.writeLine(" " + nmbstations);
   lineWriter.writeLine(SECOND_1_LINES);
   for (int k = 0; k < nmbstations; k++)
    ↪  lineWriter.writeLine(idList.get(k));
   lineWriter.writeLine(THIRD_2_LINES);
   lineWriter.writeLine(" 1  " + timestep);
   lineWriter.writeLine(FOURTH_4_LINES);
   lineWriter.writeLine(splitdate); // Dateline
   for (int k = 0; k < valuesDoubles[0].length; k++) {
      splitline = Double.toString(valuesDoubles[0][k]);
      for (int j = 1; j < nmbstations; j++) {
         splitline = splitline + " " + valuesDoubles[j][k];
      }
      lineWriter.writeLine(splitline);
   }

} catch (IOException e1) {
   throw new RuntimeException(e1.getMessage(), e1);
```

```
        //System.out.println("Something went wrong writing the file with new
        ↪  values " + e1.getMessage());
    }
}


@Override
public void initialize(File workingDir, String[] arguments) {
    //code for reading bui file, get values, times, ids


    // File that is read and later rewritten
    buistations = new File(workingDir, arguments[0]);

    // Variables defined for outside the loop
    String dateline;
    String[] stationnames = null;
    int nmbstations = 0;
    int timestep = 0; // in seconds
    int nmbtimesteps = 0;
    double[][] doubles = null;
    double[] timedouble = null;
    NumberFormat decimalnumber = NumberFormat.getNumberInstance(Locale.US);


    // First loop determines the timestep, # stations and, number of timesteps
    // With this the number of exchange items can be determined and the right
    // double string can be made.
    try (LineReader lineReader = new LineReader(buistations)) {
        String line = lineReader.readLine();
        while (line != null) {
            // Find number of weather stations
            if (line.equals("*Aantal stations")) {
                String values = lineReader.readLine().trim();
                nmbstations = Integer.parseInt(values);
                lineReader.readLine();
                stationnames = new String[nmbstations];
                for (int j = 0; j < nmbstations; j++) {
                    stationnames[j] = lineReader.readLine();
                }
            }
            // Find length of timesteps (in seconds)
            if (line.equals("*en het aantal seconden per waarnemingstijdstap")) {
                String values = lineReader.readLine();
                String seconds = values.substring(2);
                timestep = Integer.parseInt(seconds.trim());

                // Timestep is given to ExchangeItem SoebkRRTimeStepFile in
                ↪  Modified Julian date
                double[] timestepMjd = new double[1];
                timestepMjd[0] = Double.valueOf(timestep) / Double.valueOf(24 * 60
                ↪  * 60); // Timestep(seconds)/1 day (in seconds)
                SobekRRTimeStepExchangeItem exchangeItemTimestep = new
                ↪  SobekRRTimeStepExchangeItem("TimeStep", timestepMjd);
                exchangeItems.put(exchangeItemTimestep.getId(),
                ↪  exchangeItemTimestep);
            }
```

```
// For the different stations, doubles are made. They are named after
// the corresponding stations and contain all the rainfall or
↪  evaporation data
// for that station.
if (line.equals("*Daarna voor elk station de neerslag in mm per
↪  tijdstap.")) {
   dateline = lineReader.readLine();
   String[] splitline = dateline.split(" ");
   //Find start date model run
   List<String> goodsplit = new ArrayList<String>();
   //List[] goodsplit = null;
   //StringBuilder goodsplit = new StringBuilder(35); //Without empty
   ↪  values due to irregular spacing
   for (int j = 0; j < splitline.length; j++) {
      if (splitline[j].isEmpty()) continue;
      goodsplit.add(splitline[j]);
   }
   // Recover time length of data
   int year = Integer.parseInt(goodsplit.get(0));
   int month = Integer.parseInt(goodsplit.get(1));
   int day = Integer.parseInt(goodsplit.get(2));
   int hour = Integer.parseInt(goodsplit.get(3));
   int min = Integer.parseInt(goodsplit.get(4));

   // Determine total seconds of time during measurements
   int Days = Integer.parseInt(goodsplit.get(6)) * 24 * 60 * 60;
   int Hours = Integer.parseInt(goodsplit.get(7)) * 60 * 60;
   int Minutes = Integer.parseInt(goodsplit.get(8)) * 60;
   int Seconds = Integer.parseInt(goodsplit.get(9));
   int timelength = Days + Hours + Minutes + Seconds;
   nmbtimesteps = timelength / timestep;

   timedouble = new double[nmbtimesteps];
   doubles = new double[nmbstations][nmbtimesteps];

   for (int i = 0; i < nmbtimesteps; i++) {
      //Set time of timestep to Mjd.
      long millies = DateUtils.getTime(year, month, day, hour, min +
      ↪  i * timestep / 60);
      timedouble[i] = Time.milliesToMjd(millies);
      String read = lineReader.readLine();
      int j = 0;
      String[] splitRain = read.split(" ");
      // The length of split array is dependent on whether or not
      // values of rain & Ep are zero. This is why the following loop
      // is constructed.
      for (int k = 0; k < splitRain.length; k++) {
         String rain = splitRain[k];
         if (rain.trim().length() != 0) {
            // To ensure scientif (E) notation can be dealt with
            Number number = decimalnumber.parse(rain);
            doubles[j][i] = number.doubleValue();
            j++; // To ensure the #rows in doubles object are equal
            ↪  to #stations
         }
      }
   }
}
```

```
            line = lineReader.readLine();
         }
      } catch (ParseException e1) {
         e1.printStackTrace();
      } catch (Exception e) {
         throw new RuntimeException(e.getMessage(), e);
      }

      //Creation of Exchange Items

      // The exchange items have as ID the station name, e.g. "'Scheveningen'",
      ↪  the times (in modified Julian Date)
      // over which the raindata are avaialble and the corresponding values of
      ↪  those times as doubles.
      for (int k = 0; k < nmbstations; k++) {
         SobekRRExchangeItem exchangeItem = new
         ↪  SobekRRExchangeItem(stationnames[k], timedouble, doubles[k]);
         exchangeItems.put(exchangeItem.getId(), exchangeItem);
      }
   }
}
```

```
package nl.deltares.openda.models.sobek;

import org.openda.interfaces.*;

/**
 * Created by petraizeboud on 07/02/2017.
 */
public class SobekRRExchangeItem implements IExchangeItem {

   private final String id;
   private double[] values;
   private final double[] times;

   public SobekRRExchangeItem(String id, double[] times, double[] values) {
      this.id = id;
      this.times = times;
      this.values = values;
   }

   @Override
   public Role getRole() {
      return null;
   }

   @Override
   public double[] getValuesAsDoubles() {
      return values;
   }

   @Override
   public void axpyOnValues(double alpha, double[] axpyValues) {
      if (this.values != null) {
         for (int i = 0; i < values.length; i++) {
            values[i] += alpha * axpyValues[i];
         }
      }
```

```
    }

    @Override
    public void multiplyValues(double[] multiplicationFactors) {
       if (this.values != null) {
          for (int i = 0; i < values.length; i++) {
             values[i] *= multiplicationFactors[i];
          }
       }
    }

    @Override
    public void setValues(Object values) {
       this.values = (double[]) values;
    }

    @Override
    public void setValuesAsDoubles(double[] values) {
       this.values = values;
    }

    @Override
    public double[] getTimes() {
       return times;
    }

    @Override
    public void setTimes(double[] times) {

    }

    @Override
    public String getId() {
       return id;
    }

    @Override
    public String getDescription() {
       return null;
    }

    @Override
    public void copyValuesFromItem(IExchangeItem sourceItem) {

    }

    @Override
    public ITimeInfo getTimeInfo() {
       return null;
    }

    @Override
    public IQuantityInfo getQuantityInfo() {
       return null;
    }

    @Override
    public IGeometryInfo getGeometryInfo() {
```

```
        return null;
    }

    @Override
    public ValueType getValuesType() {
        return null;
    }

    @Override
    public Object getValues() {
        return null;
    }

    @Override
    public Class getValueType() {
        return null;
    }
}
```

B.2 Exchange of restart file

SobekRRRestartFile.java and SobekRRRestartExchangeItem As the name suggest, the information of the states are stored in an Exchange item of the form SobekRRRestartExchangeItem, with as IDs the names of the storages (e.g. Unsaturated zone, storage on land, storage in sewage, etc.), and the corresponding values. This corresponds to 16 sobekRRRestartExchangeItems, since there are 16 different storage states. Once the values of the states in the exchange items are replaced by the posterior value calculated by the DA, SobekRRRestartFile finalizes by writing an ascii- restart file with the updated states. This was previously not possible since the restart file of Sobek was a binary file, but this was adjusted especially for this thesis such that Sobek RR can now be restarted with a text file specifying the initial storage in the states.

```
package nl.deltares.openda.models.sobek;

/**
 * Created by Izep on 15-02-2017.
 * code for reading out states from Sobek and saving them as exchange items
 * can be found under public void inititialize. The arguments refer to the
 * document with the states from SobekRR.
 * The exchange items are recognized in OpenDA and the values of the states
 * are changed as desired. This is then written to a restart file fed to
 * Sobek in public void finish.
 */

import nl.wldelft.util.DoubleArrayUtils;
import nl.wldelft.util.io.LineReader;
import nl.wldelft.util.io.LineWriter;
import org.openda.interfaces.IDataObject;
import org.openda.interfaces.IExchangeItem;
import org.openda.interfaces.IPrevExchangeItem;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
```

```java
import java.util.*;
import java.util.List;

public class SobekRRRestartFile implements IDataObject {

   //Map for exchange items
   LinkedHashMap<String, IExchangeItem> exchangeItems = new LinkedHashMap<>();

   // These save the node-ID names of the different node types
   List<String> nodesUnpv = new ArrayList<String>();
   List<String> nodesPave = new ArrayList<String>();
   List<String> nodesGrhs = new ArrayList<String>();
   List<String> nodesOpwa = new ArrayList<String>();

   //The variable that has the restart file that is read & rewritten
   private File sobekRRRestartFile;


   // These are Fixed
   public static final String[] STATENAMES_UNPV = new String[]{" gwvolume ", "
    ↪  onvzonevolume ", "  bergingland "};
   // Note: the additional space before bergingland similar to the given restart
    ↪  file by Geert Prinsen
   public static final String[] STATENAMES_PAVE = new String[]{" streetvolume ",
    ↪  " sewervolumeRWA ", " sewervolumeDWA ", " dynamicvolume "};
   public static final String[] STATENAMES_GRHS = new String[]{" roofvolume ",
         " basin1 "," basin2 "," basin3 "," basin4 "," basin5 "," basin6 ","
          ↪  basin7 "," basin8 "," basin9 ", " basin10 ",
         " silovolume "};
   public static final String[] STATENAMES_OPWA = new String[]{" openwaterlevel
    ↪  "};
   public static final String[] STATENAMES = new String[]{   " streetvolume ", "
    ↪  sewervolumeRWA ", " sewervolumeDWA ", " dynamicvolume ",
         " gwvolume ", " onvzonevolume ", "  bergingland ",
         " roofvolume ", " basin1 "," basin2 "," basin3 "," basin4 "," basin5 ","
          ↪  basin6 "," basin7 "," basin8 "," basin9 ", " basin10 ", " silovolume
          ↪  ",
         " openwaterlevel "};


   @Override
   public String[] getExchangeItemIDs() {
      return exchangeItems.keySet().toArray(new
       ↪  String[exchangeItems.keySet().size()]);
   }

   @Override
   public String[] getExchangeItemIDs(IPrevExchangeItem.Role role) {
      return new String[0];
   }

   @Override
   public IExchangeItem getDataObjectExchangeItem(String exchangeItemID) {
      return exchangeItems.get(exchangeItemID);
   }

   @Override
   public void finish() {
```

```
// Make new File
File sobekRRRestart_reload = new
 ↪  File("D:\\Users\\izep\\Documents\\Afstuderen\\Java\\Test",
 ↪  "sobekRRRestart_all_reload.txt");
try {
   sobekRRRestart_reload.createNewFile();
} catch (IOException exception) {
   System.out.println("Something went wrong: " + exception.getMessage());
}

//Collect data from Exchange Items

Collection<IExchangeItem> values = exchangeItems.values();

//Variables defined outside loop

double[][] valuesDoubles = new double[values.size()][];
List<String> idList = new ArrayList<>();

//Information from Exchange items

int i = 0;
for (IExchangeItem exchangeItem : values) {
   valuesDoubles[i] = exchangeItem.getValuesAsDoubles();
   String id = exchangeItem.getId();
   idList.add(id);
   i++;
}



try (LineWriter lineWriter = new LineWriter(sobekRRRestartFile)) {
   // Write information of the unpaved nodes
   //lineWriter.writeLine("test");
   for (i = 0; i < nodesPave.size(); i++) {
      StringBuilder string = new StringBuilder(35);
      string.append("PAVE id " + nodesPave.get(i) );
      for (int j = 0; j < STATENAMES_PAVE.length; j++) {
         string = string.append(STATENAMES_PAVE[j] +
               valuesDoubles[j][i]);
      }
      string.append(" pave");
      lineWriter.writeLine(string);
   }
   for (i = 0; i < nodesUnpv.size(); i++) {
      StringBuilder string = new StringBuilder(35);
      string.append("UNPV id " + nodesUnpv.get(i) );
      for (int j = 0; j < STATENAMES_UNPV.length; j++) {
         string = string.append(STATENAMES_UNPV[j] + valuesDoubles[j+
          ↪  STATENAMES_PAVE.length][i]);
      }
      string.append(" unpv");
      lineWriter.writeLine(string);
   }
   for (i = 0; i < nodesGrhs.size(); i++) {
      StringBuilder string = new StringBuilder(400);
      string.append("GRHS id " + nodesGrhs.get(i));
```

```
            string.append(STATENAMES_GRHS[0] +
             ↪  valuesDoubles[STATENAMES_UNPV.length +
                  STATENAMES_PAVE.length][i] + " basinvolumes ");
            for (int j = 1; j < 11; j++) {
               string = string.append(" " + valuesDoubles[j +
                ↪  STATENAMES_UNPV.length +
                     STATENAMES_PAVE.length][i]);
            }
            string.append(STATENAMES_GRHS[11] +
             ↪  valuesDoubles[STATENAMES_UNPV.length +
                  STATENAMES_PAVE.length][i]);
            string.append(" grhs");
            lineWriter.writeLine(string);
         }
         for (i = 0; i < nodesOpwa.size(); i++) {
            StringBuilder string = new StringBuilder(35);
            string.append("OPWA id " + nodesOpwa.get(i) );
            for (int j = 0; j < STATENAMES_OPWA.length; j++) {
               string = string.append(STATENAMES_OPWA[j] + valuesDoubles[j +
                ↪  STATENAMES_UNPV.length +
                     STATENAMES_PAVE.length + STATENAMES_GRHS.length][j]);
            }
            string.append(" opwa");
            lineWriter.writeLine(string);
         }

      } catch (IOException e1) {
         e1.printStackTrace();
         System.out.println("Something went wrong writing the file with new
          ↪  values " + e1.getMessage());
      }
   }


   @Override
   public void initialize(File workingDir, String[] arguments) {




      //***********************************Load restart file
       ↪  ********************//
      sobekRRRestartFile = new File(workingDir, arguments[0]);
      //sobekRRRestartFile2 = new
       ↪  File("D:\\Users\\izep\\Documents\\Afstuderen\\Java\\", arguments[1]);
      //test = arguments[0];
      //***********************************Data verzamelen**********************//

      // Save all the storage of the different Nodes per 'bucket' in a List for
       ↪  each bucket
      List<ArrayList<Double>> states = new ArrayList<ArrayList<Double>>();
      // Add empty lists to the states
      for (int i = 0; i < STATENAMES_UNPV.length + STATENAMES_PAVE.length +
       ↪  STATENAMES_GRHS.length
            + STATENAMES_OPWA.length; i++)
         states.add(new ArrayList<Double>());
```

```
// Collect states to be put in List<ArrayList<Double>> states
// Collect nodeIDs to be put in nodesXXX.
try (LineReader lineReader = new LineReader(sobekRRRestartFile)) {
   String line = lineReader.readLine();
   while (line != null) {
      String[] splitline = line.split(" ");
      List<String> goodsplit = new ArrayList<String>();
      for (int j = 0; j < splitline.length; j++) {
         if (splitline[j].isEmpty()) continue;
         goodsplit.add(splitline[j]);
      }
      //List[] goodsplit = null;
      //StringBuilder goodsplit = new StringBuilder(35); //Without empty
       ↪  values due to irregular spacing
      if (splitline[0].equals("PAVE")) {
         nodesPave.add(goodsplit.get(2));
         for (int i = 0; i < STATENAMES_PAVE.length; i++) {
            states.get(i).add(Double.valueOf(goodsplit.get(2 * i + 4)));//0
             ↪  tm 3
         }
         line = lineReader.readLine();
         continue;
      }
      if (splitline[0].equals("UNPV")) {
         for (int i = 0; i < STATENAMES_UNPV.length; i++) {
            states.get(i+4).add(Double.valueOf(goodsplit.get(2 * i +
             ↪  4)));//4 t/m 6
         }
         nodesUnpv.add(goodsplit.get(2));
         line = lineReader.readLine();
         continue;
      }
      if (splitline[0].equals("GRHS")) {
         nodesGrhs.add(goodsplit.get(2));
         states.get(7).add(Double.valueOf(goodsplit.get(4))); // roofvolume
         for (int i = 0; i < STATENAMES_GRHS.length-2; i++) {
            states.get(i + 8).add(Double.valueOf(goodsplit.get(i + 6)));
             ↪  //8 t/m (+10) 17
         }
         states.get(18).add(Double.valueOf(goodsplit.get(17))); //
          ↪  rilovvolume
         line = lineReader.readLine();
         continue;
      }
      if (splitline[0].equals("OPWA")) {
         nodesOpwa.add(goodsplit.get(2));
         states.get(19).add(Double.valueOf(goodsplit.get(4)));
         line = lineReader.readLine();
         continue;
      }
   }
} catch (FileNotFoundException e) {
   e.printStackTrace();
   System.out.println("ascii Restartfile openda not found " +
    ↪  e.getMessage());
} catch (IOException e) {
   e.printStackTrace();
```

```
            System.out.println("Something went wrong in Linereader " +
             ↪  e.getMessage());
        }

        //
        int size = STATENAMES_UNPV.length + STATENAMES_PAVE.length
              + STATENAMES_GRHS.length + STATENAMES_OPWA.length;
        for (int i = 0; i < size; i++) {
            //Steps for changing Arraylist to double[] that can be recognized by
             ↪  openDA
            ArrayList<Double> stateList = states.get(i); //Gives storages for
             ↪  state(i), e.g. [0.0 3.0 310] for 'unvzone'
            Double[] stateArray = new Double[stateList.size()]; //Creates from the
             ↪  Arraylist an Array of Doubles
            stateList.toArray(stateArray); //puts the values of the doubles in the
             ↪  new Array of Doubles.
            double[] unboxedStateDoubles = DoubleArrayUtils.unbox(stateArray);
             ↪  //Creates from the Array of objects a double[]
            // Create Exchange items
            SobekRRRestartExchangeItem exchangeItem = new
             ↪  SobekRRRestartExchangeItem(STATENAMES[i].trim() + "_state",
             ↪  unboxedStateDoubles);
            exchangeItems.put(exchangeItem.getId(), exchangeItem);
        }
    }
}
```

B.3 Exchange of time functionalities

SobekRRModelRunTimeFile and SobekRRModelRunTimeExchangeItem. This part only reads out the information on the duration of the sobek run, i.e. the data assimilation should be done the multiple of times that the DA-timestep fits in the time-horizon specified in the CRR-model.

```
package nl.deltares.openda.models.sobek;
/**
 * Created by Izep on 20-02-2017.
 */

import nl.wldelft.util.DateUtils;
import nl.wldelft.util.FileUtils;
import nl.wldelft.util.io.LineReader;

import org.openda.exchange.timeseries.TimeUtils;
import org.openda.interfaces.IDataObject;
import org.openda.interfaces.IExchangeItem;
import org.openda.interfaces.IPrevExchangeItem;
import org.openda.utils.Time;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;

import java.util.*;

public class SobekRRModelRunTimeFile implements IDataObject {
```

```
LinkedHashMap<String, IExchangeItem> exchangeItems = new LinkedHashMap<>();

private File delft3Bini;


@Override
public String[] getExchangeItemIDs() {
   return exchangeItems.keySet().toArray(new
    ↪  String[exchangeItems.keySet().size()]);
}

@Override
public String[] getExchangeItemIDs(IPrevExchangeItem.Role role) {
   return new String[0];
}

@Override
public IExchangeItem getDataObjectExchangeItem(String exchangeItemID) {
   return exchangeItems.get(exchangeItemID);
}

@Override

public void finish() {

   String text = null;
   List<String> datelines = new ArrayList<String>();


   // Get data from Exchange Items

   Collection<IExchangeItem> values = exchangeItems.values();
   double[][] valuesDoubles = new double[values.size()][];
   //List<String> idList = new ArrayList<>();

   //Loop over Exchange Items
   int i = 0;
   for (IExchangeItem exchangeItem : values) {
      valuesDoubles[i] = exchangeItem.getValuesAsDoubles();
      String id = exchangeItem.getId();
      //idList.add(id);
      i++;
   }

   //Create new date line to be put in delft3Bini

   for (int j = 0 ; j < 2; j++) {
      String time = TimeUtils.mjdToString(valuesDoubles[j][0]); //in mjd
      String year = time.substring(0,4);
      String month = time.substring(4,6);
      String day = time.substring(6,8);
      String hour = time.substring(8,10);
      String min = time.substring(10,12);
      datelines.add( year + "/" + month + "/" + day + ";" + hour + ":" + min +
       ↪  ":00");
   }
```

```
        //Read file delft_3B.ini and save text in String
        try {
            text = FileUtils.readText(delft3Bini);
        } catch (IOException e) {
            e.printStackTrace();
        }


        // Replace old date string in text by new date string
        if (text.contains("PeriodFromEvent=0")){
            text = text.replaceFirst("(.*)StartTime=(.*)", "StartTime='" +
            ↪  datelines.get(0) + "'");
            text = text.replaceFirst("(.*)EndTime=(.*)", "EndTime='"+
            ↪  datelines.get(1) + "'");
        }

        //Rewrite the delft3Bini file with new text
        try {
            FileUtils.writeText(delft3Bini, text);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void initialize(File workingDir, String[] arguments) {
        //code for reading actual time span that is used in Sobek
        //A different start and end time are used if periodfromevent = 0 in file
        ↪  DELFT_3B.INI
        //If indeed PeriodFromEvent = 0, the actual start and enddate are saved as
        ↪  ExchangeItems.
        //If PeriodFromevent =! 0, it will print "Period from event not 0", and no
        ↪  exchange items are made


        //Variables

        String time = null;
        String[] timeID = {"start_time", "end_time"};
        double[][] timedouble = new double[2][1];


        //Load File

        delft3Bini = new File(workingDir, arguments[0]);

        //Read out Start and End time

        try (LineReader lineReader = new LineReader(delft3Bini)) {
            String line = lineReader.readLine();
            while (line != null) {
                if (line.contains("PeriodFromEvent=0")) {
                    for (int i = 0; i < timeID.length; i++) {
                        line = lineReader.readLine();
                        if (i == 0) time = line.substring(11);
                        if (i == 1) time = line.substring(9);
                        int yearInt = Integer.parseInt(time.substring(0, 4));
                        int monthInt = Integer.parseInt(time.substring(5, 7));
```

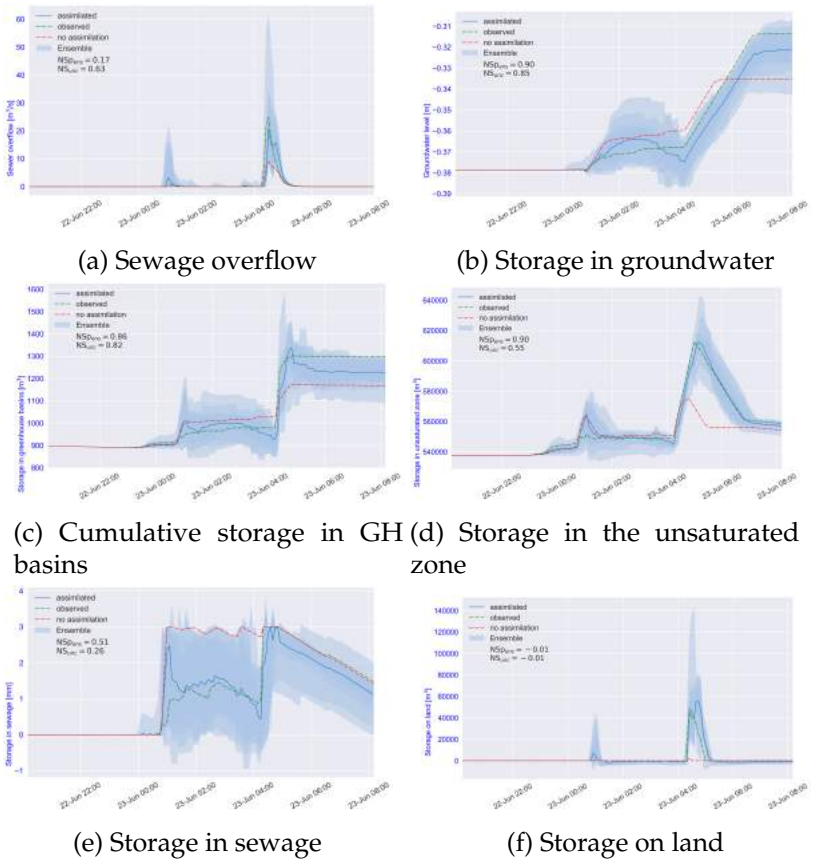
```
                int dayInt = Integer.parseInt(time.substring(8, 10));
                int hourInt = Integer.parseInt(time.substring(11, 13));
                int minInt = Integer.parseInt(time.substring(14, 16));
                long millies = DateUtils.getTime(yearInt, monthInt, dayInt,
                ↪  hourInt, minInt);
                timedouble[i][0] = Time.milliesToMjd(millies);
             }
             break;
          }
          line = lineReader.readLine();
       }
    } catch (FileNotFoundException e) {
       e.printStackTrace();
       System.out.println("File DELFT_3B.INI not found " + e.getMessage());
    } catch (IOException e) {
       e.printStackTrace();
       System.out.println("Something went wrong reading DELFT_3B.INI file, for
       ↪  Period of Event " + e.getMessage());
    }


    //Exchange Items

    if (time == null) System.out.println("PeriodFromEvent is not 0");
    else {
       for (int i = 0; i < 2; i++) {
          // Note: Because time is asked to be a double[] timedouble[i] is a
          ↪  double[] with only 1 entry. This seems a bit weird.
          SobekRRModelRunTimeExchangeItem exchangeItem = new
          ↪  SobekRRModelRunTimeExchangeItem(timeID[i], timedouble[i]);
          exchangeItems.put(exchangeItem.getId(), exchangeItem);
       }
    }

 }
}
```

C

Additional results of the twin experiment

C.1 Additional results on the influence on the states due to pump settings

The figures show the influence of the pumps on the assimilated states. The results are generated with a data assimilation run for the twin experiment described in 4.4, where pumps are in place. The ensemble size is 16 and the time increment is 15 minutes.



(a) Sewage overflow

(b) Storage in groundwater

(c) Cumulative storage in GH basins

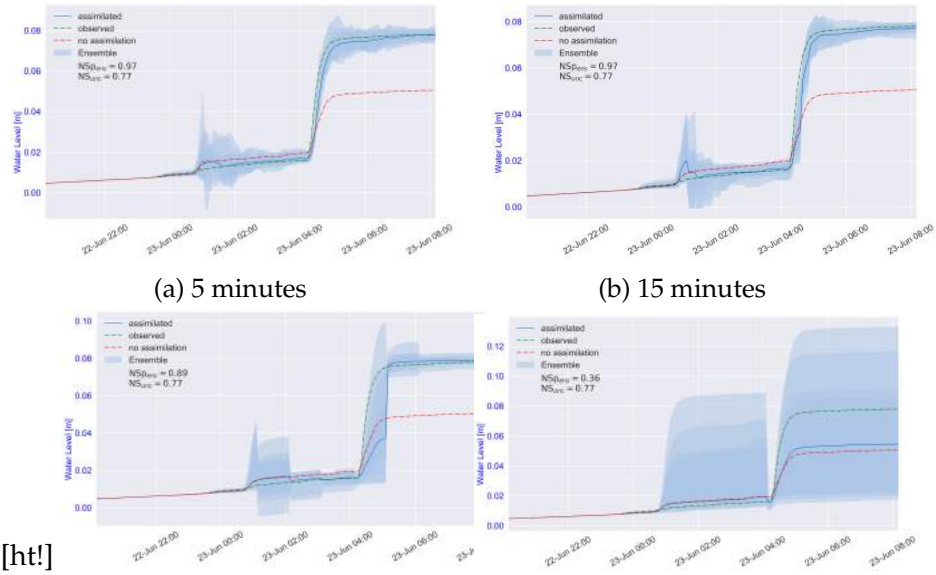
(d) Storage in the unsaturated zone

(e) Storage in sewage

(f) Storage on land

Figure C.1: The influence of the pump settings on the assimilated states.

C.2 Additional results for different algorithm settings

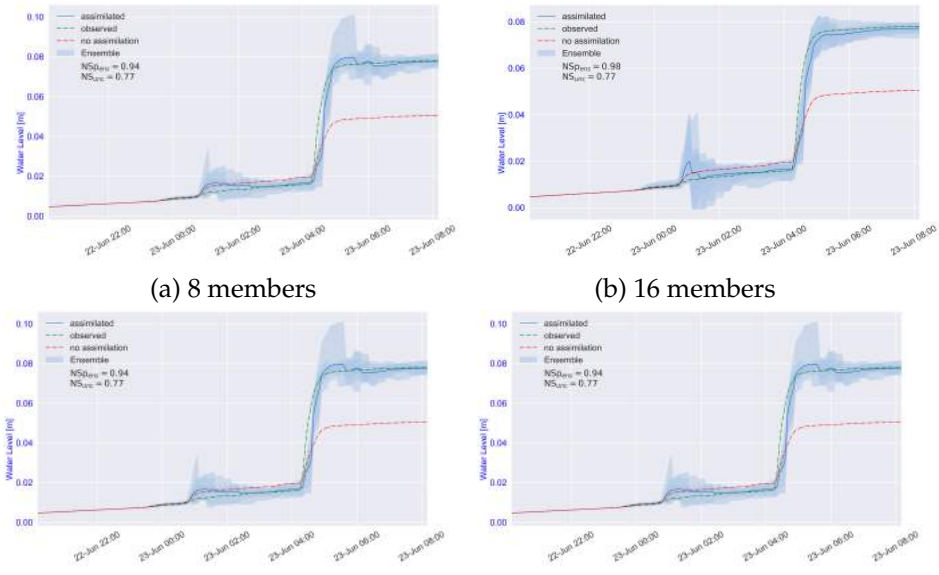


(a) 5 minutes (b) 15 minutes

(c) 1 hour (d) 5 hours

[ht!]

Figure C.2: Assimilated results for different time increments with an ensemble size of 16.

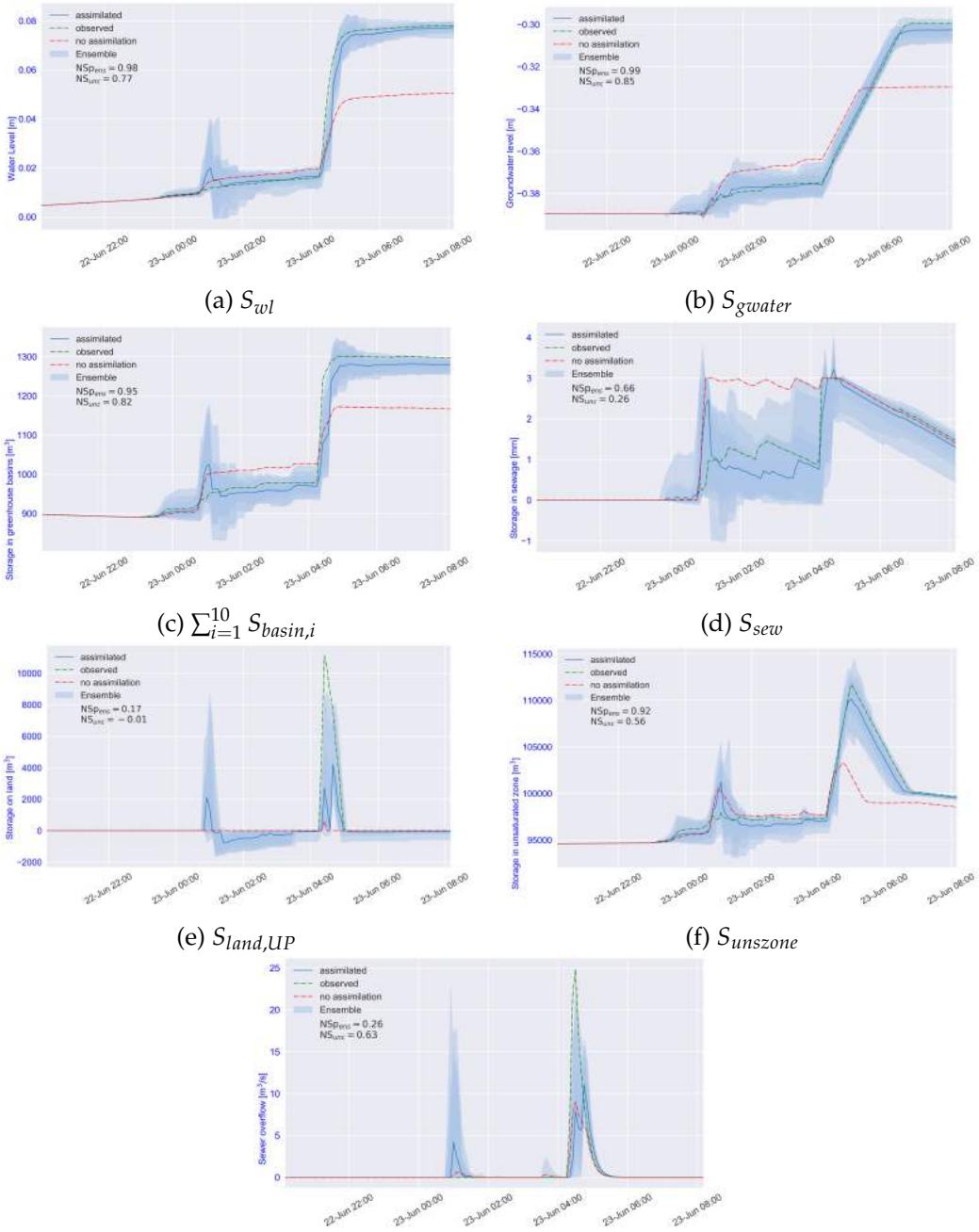


(a) 8 members (b) 16 members

(c) 32 members (d) 64 members

Figure C.3: Assimilated results for different ensemble sizes with time increment of 15 minutes.

C.3 Additional results for DA with different noise-components



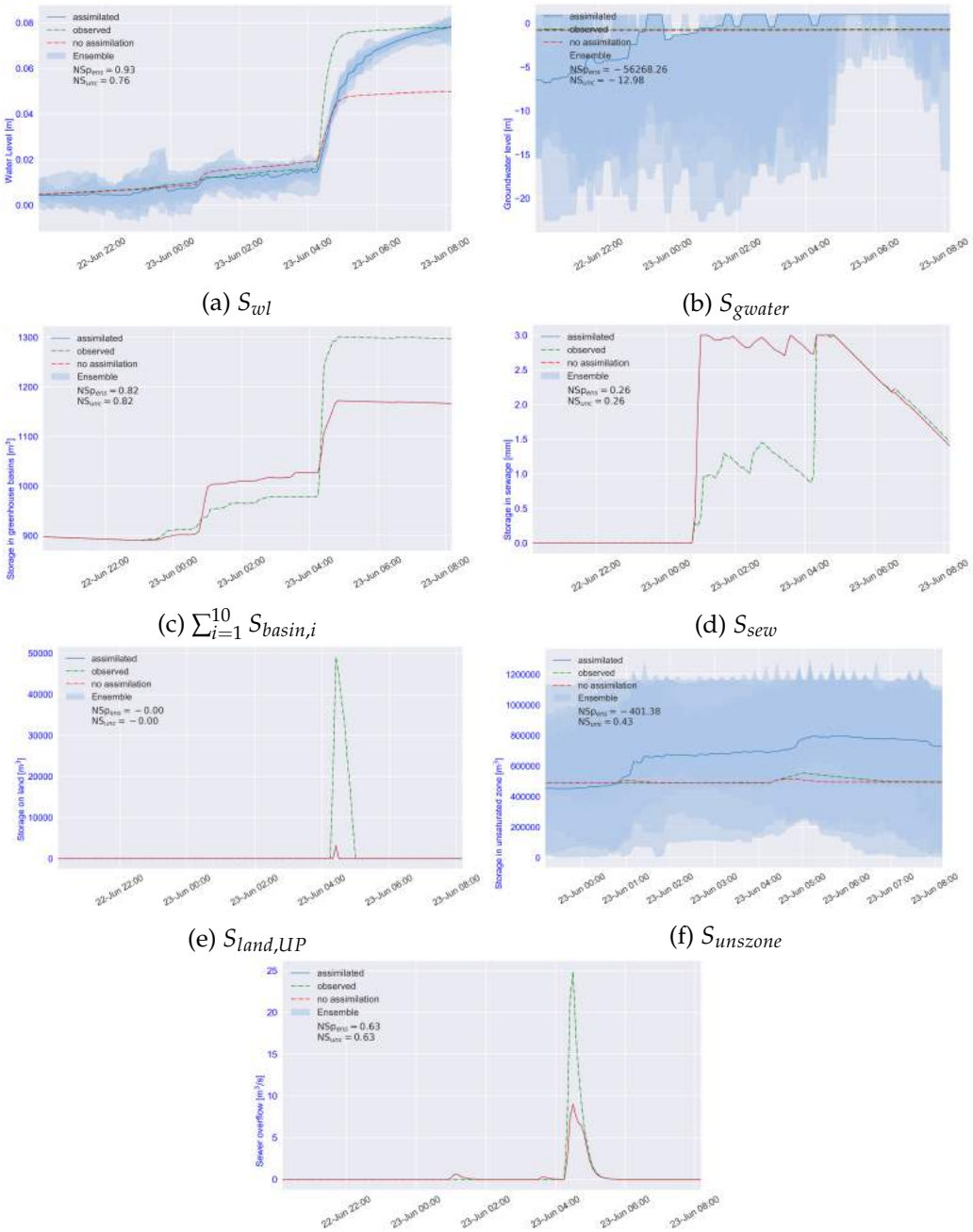
(a) S_{wl} (b) S_{gwater}

(c) $\sum_{i=1}^{10} S_{basin,i}$ (d) S_{sew}

(e) $S_{land,UP}$ (f) $S_{unszone}$

(g) Q_{SO}

Figure C.4: Multiplicative noise on Rainfall



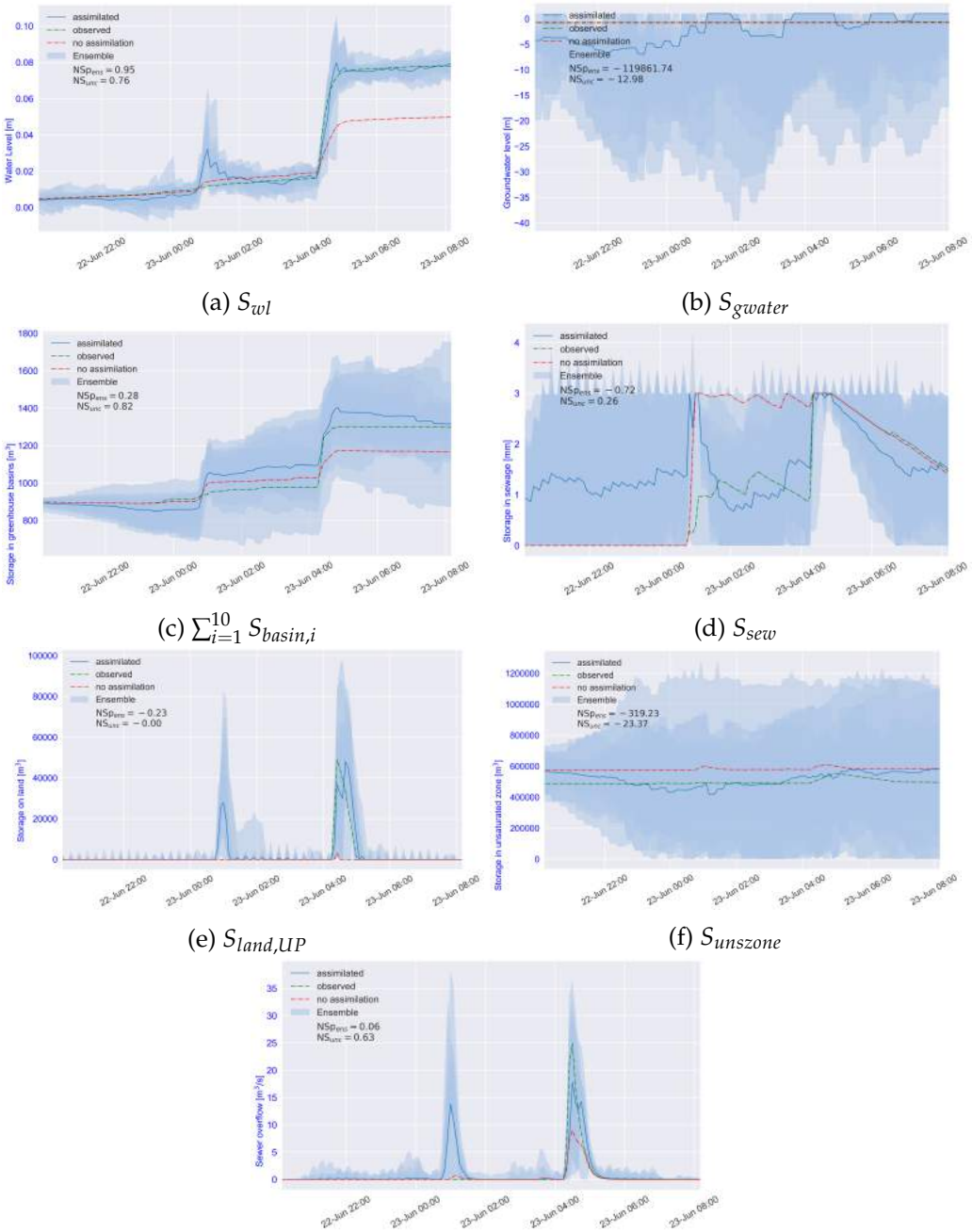
(a) S_{wl} (b) S_{gwater}

(c) $\sum_{i=1}^{10} S_{basin,i}$ (d) S_{sew}

(e) $S_{land,UP}$ (f) $S_{unszone}$

(g) Q_{sew}

Figure C.5: Additive noise on one state: the groundwater volume



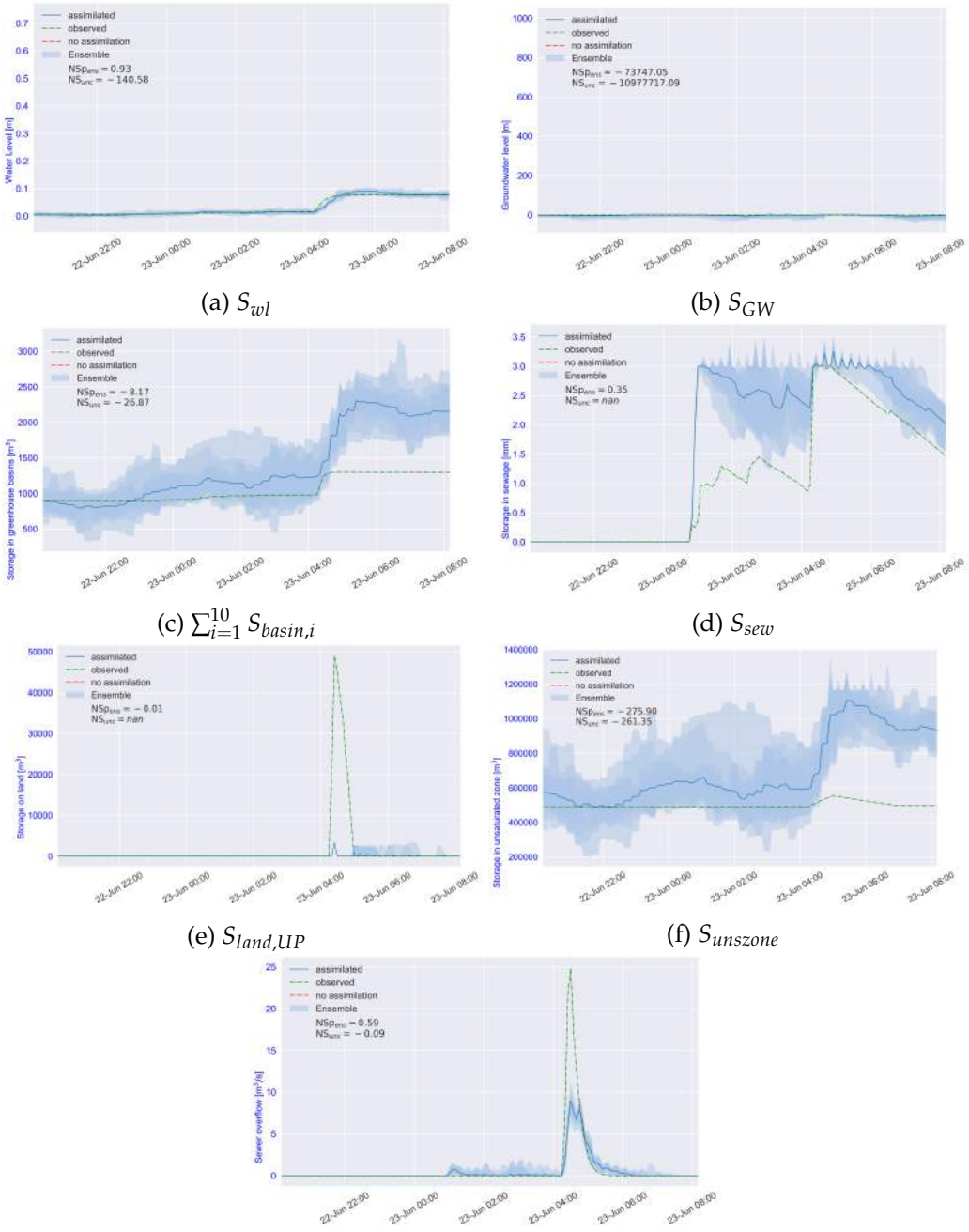
(a) S_{wl} (b) S_{gwater}

(c) $\sum_{i=1}^{10} S_{basin,i}$ (d) S_{sew}

(e) $S_{land,UP}$ (f) $S_{unszone}$

(g) $S_{unszone}$

Figure C.6: Additive noise on all states



(a) S_{wl}

(b) S_{GW}

(c) $\sum_{i=1}^{10} S_{basin,i}$

(d) S_{sew}

(e) $S_{land,UP}$

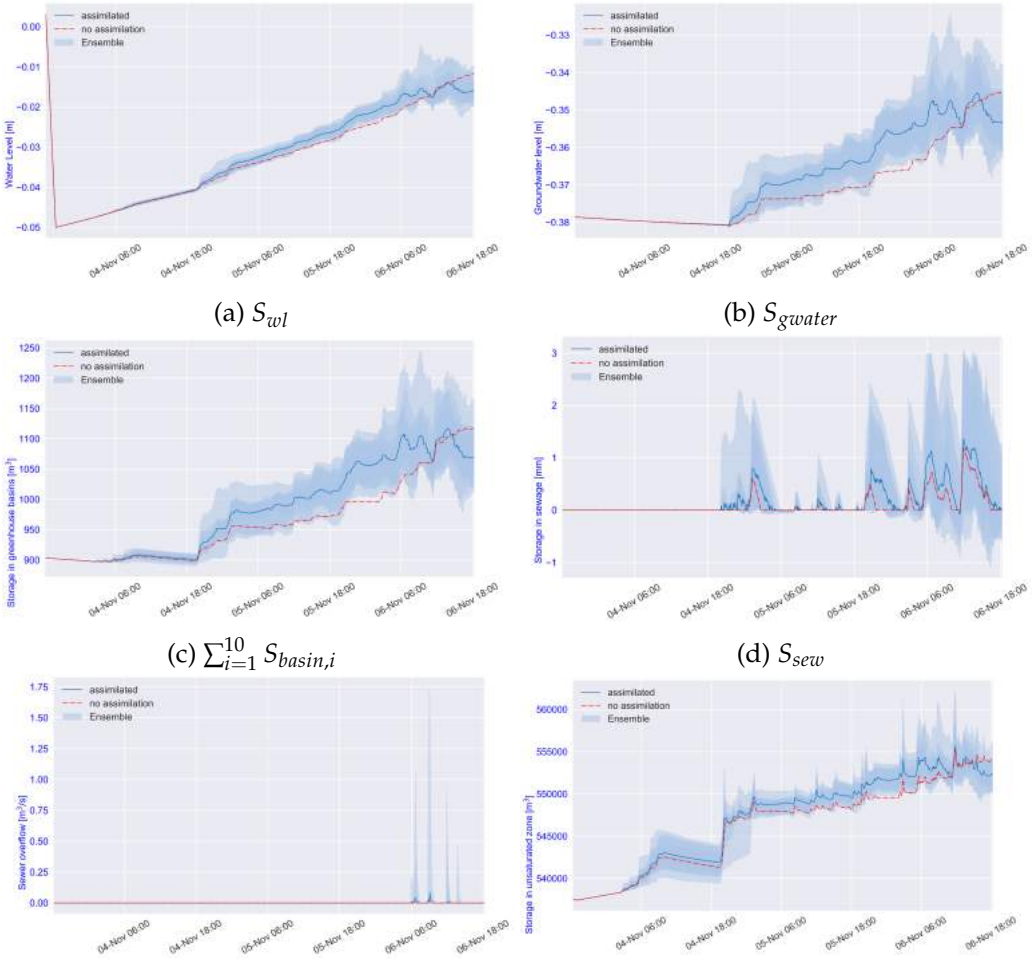
(f) $S_{unszone}$

(g) $S_{unszone}$

Figure C.7: Multiplicative noise on all states

D

Additional results of the case study

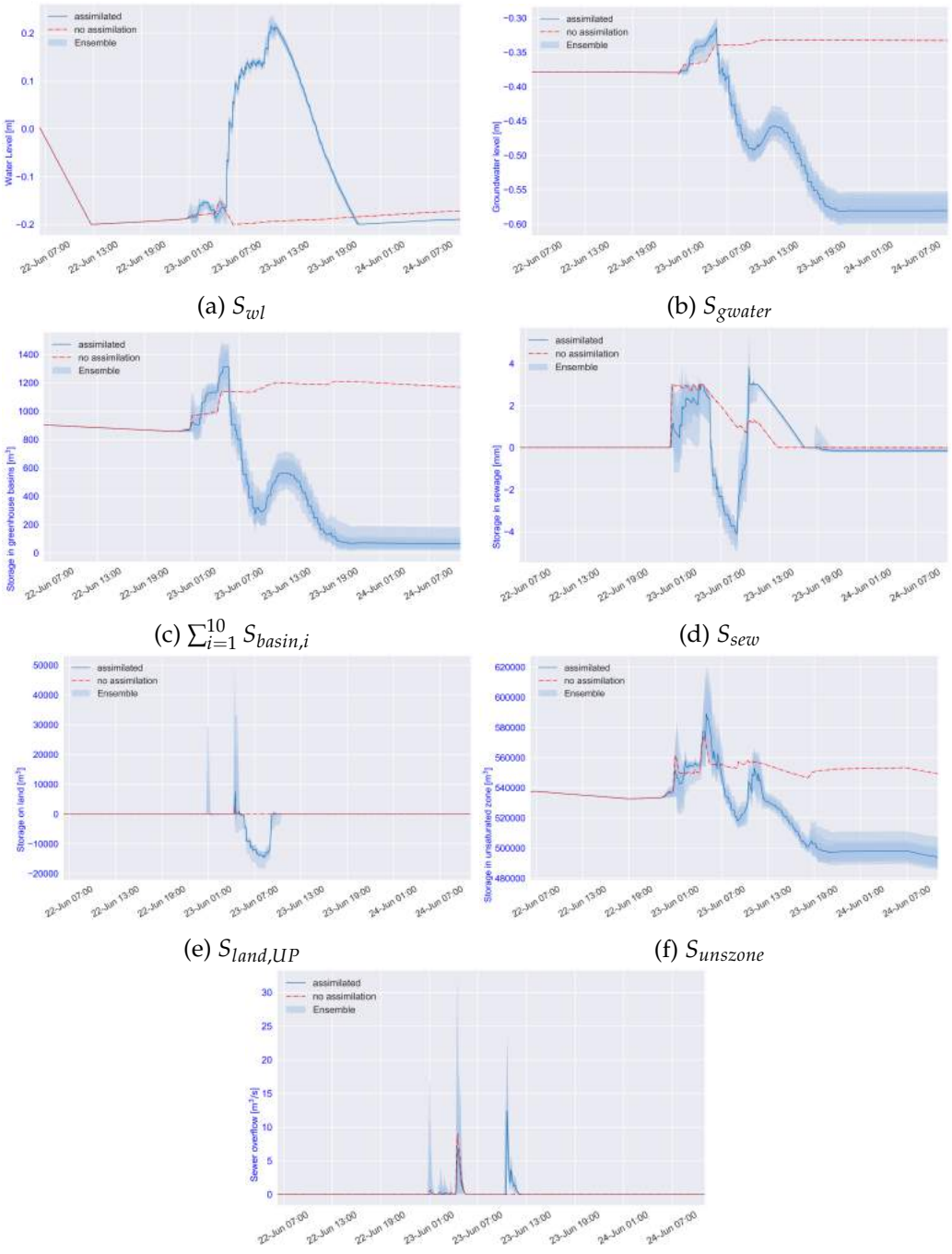


(a) S_{wl} (b) S_{gwater}

(c) $\sum_{i=1}^{10} S_{basin,i}$ (d) S_{sew}

(e) $S_{unszone}$ (f) $S_{unszone}$

Figure D.1: Additional results for case study on 23rd of November



(a) S_{wl}

(b) S_{gwater}

(c) $\sum_{i=1}^{10} S_{basin,i}$

(d) S_{sew}

(e) $S_{land,UP}$

(f) $S_{unszone}$

(g) $S_{unszone}$

Figure D.2: Additional results for case study on 23rd of June