Delft University of Technology

Internship report

Improving OpenDA's blackbox wrapper for Sobek

Author:
Daniel Mulder

Supervisors:
Dr. Nils van Velzen
Dr. Hans Korving

December 10, 2012

Contents

1 Introduction

Studying applied mathematics without actually applying it isn't really applied mathematics, is it...? An internship seemed to be a nice way to actually work, using mathematical tools, on some real world problems or projects. Of course this had nothing to do with the fact that doing an internship, either in the general master program or as part of the master thesis, has been made mandatory.

Let me quote from the studyguide of the Master Applied Mathematics, about the internship: *The project is carried out at an institute where the mathematics itself is generally not of primary interest but merely a tool to solve practical problems.* With this in mind and after consulting the responsible instructor, Arnold Heemink, I decided to contact V$\mathcal{O}$Rtech, a company in Delft. Specifically Mark Roest, managing director.

Before long, I got invited to come talk with Mark about the possibility of doing an internship and as it turns out, also the possibility of doing my master thesis (if all involved parties agree). I was asked to make a decision between two possible projects: modelling a sewer system or working on an interactive Navier-Stokes solver. I chose the former. This project is done at V$\mathcal{O}$Rtech in cooperation with Witteveen+Bos. The people involved (my supervisors) are Nils van Velzen (V$\mathcal{O}$Rtech) and Hans Korving (Witteveen+Bos).

I had a second meeting with Mark, Nils and Hans ragarding this specific project. A lot of information was given and eventually I got the assignment to write a short project description, to determine if I knew what was to be done in the internship. My project description got approved and I started my internship October 1st.

2 The project

The project I worked on is about modelling a sewer system using a program called SOBEK. In SOBEK it is possible to edit and/or make a sewer network using a simple network editor. Adjusting different settings is also possible and eventually the program will calculate how the sewer system behaves in time. As for details on how to create a simple sewer system project in SOBEK, section 3.2 is particularly suited as a guide through the process. Another part of the project was to use data assimilation in order to improve the model even further. The challenge was to establish a way to effectively combine SOBEK with OpenDA. I did not need to start from scratch thought, a part of the work of combining SOBEK and OpenDA had already been done by my predecessor, Johan Post, as part of his master thesis. After about 1.5 weeks in my internship, I had familiarized myself a bit with OpenDA and SOBEK and it was time to discuss exactly what to do in my internship. Which came down to the following:

1. Get Johan's version of the SOBEK plus OpenDA things working correctly.
2. Upgrading from OpenDA 2.0 to the trunk version of OpenDA.
3. Find a way to access the NEFIS files which are part of the SOBEK output. These files are used to store the state of the model.
4. There seems to be a problem while restarting in OpenDA (needed in order to assimilate inbetween). It looks like the run-off model omitts the last time step, which causes differences in results compared to a full model run without restarts. Look into this problem and see if it can be fixed.
5. Get the sewer system model of Delft running using observations that are created using the model itself (twin experiment).

Other points, though less important for the internship itself:

6. Improve the noise modelling in OpenDA.
7. Assimilate multiple observations simultaneously.

Section 3 covers points 1 and 2, section 4 covers point 3, points 4 and 5 will be discussed in section 5. The noise modelling in point 6 is being worked on at the moment and point 7, multiple assimilations simultaneously, is to be done.

3 Modelling software

3.1 OpenDA

OpenDA is an open source data-assimilation toolbox used to quickly implement data-assimilation and calibration for numerical models. OpenDA has quite some assimilation and calibration methods, we will however restrict ourselves to using the Ensemble Kalman Filter for the assimilation and the DUD method for calibration (parameter estimation).

An OpenDA project consists of several xml files. Here is an overview of the general setup of a project.

main configuration file
: This is the main file for the project. In this file the names and locations for other files needed are specified. Resultwriters are also specified in this file. These are used to generate output in e.g. matlab format to visualize results.

stochastic observer
: In this file (or files, can be more than one), observation data is specified as well as some information about the uncertainty in the data.

algorithm
: The algorithm that is to be used is actually specified in the main configuration file by the algorithm classname. This algorithm configuration file specifies the necessary input parameters.

stochastic model
: In this file, information about the model is specified. In the case where a blackbox model is used, this part consists of three files. Since we will be making use of SOBEK for the modelling part, this indeed is a blackbox model. The specific files involved are the following.

  StochModelConfig
  : In this file, different state and/or parameter vectors are defined to be calibrated or adjusted according to some noise model.

  ModelConfig
  : Here certain aliasvalues are defined, linking names or labels to be used in OpenDA to their name or label outside of OpenDA. This file also contains specifications on exactly what exchange items to load from the ioObjects (which are specified in the wrapper).

  WrapperConfig
  : This is a file consisting of alias definitions, a run sequence and input-output definitions (the ioObjects). The alias definitions are a link to the aliases in the ModelConfig (which link to the actual values). The run part of the file contains initialize, compute and finalize actions, all of which do more or less what their name suggests. Last part of the wrapper are the ioObjects, in- and output of data.

More details about the OpenDA blackbox configuration for SOBEK later in the report. For that we need to set up a SOBEK (test) model first.

3.2 Sobek (testcase)

In order to model rainfall runoff into a sewer system and the flow in the sewer system itself, we use SOBEK. SOBEK is a software package for river, urban or rural management. It has several program modules that work together to give a comprehensive overview of waterway systems. As far as our sewer system goes, we make use of two of these program modules. These are the RR (Rainfall Runoff) module and the 1DFLOW (Urban) module. In this section, setting up a toy sewer model is shortly discussed.

Starting SOBEK and creating a new project will lead to the SOBEK case screen (figure 1), in which you can open a saved case or simply open a default case. Yellow boxes are the ones that can be set up at that moment. Red boxes are not accessible at the time and green boxes have been set up and allow the user

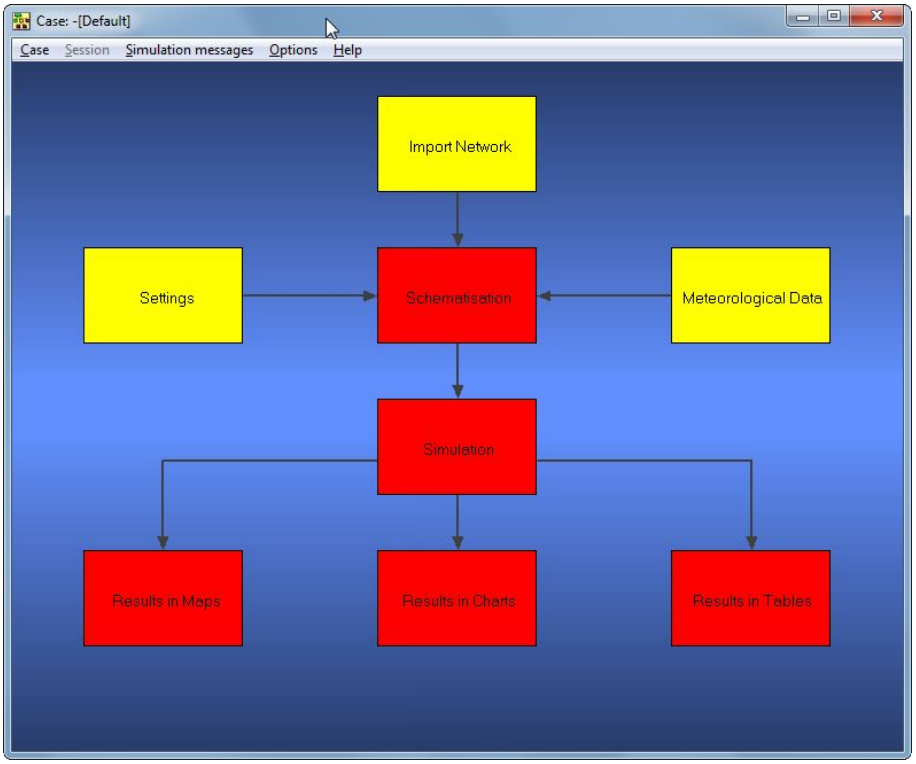


Figure 1: SOBEK Case screen

to move on to the next box (note that a green box does not mean it is set up correctly i.e. the simulation might still fail).

Settings, Import Network, Meteorological Data
In the import network part we will either "start from scratch" or do nothing in the case that we have loaded a previous saved case which has already been set up. The meteorological data contains information about the precipitation to be used, as well as the wind velocity and direction and the solar radiation. The

settings contain general information to set up the different modules to be used, most of which are pretty straight forward.

Schematisation
In the schematisation, the actual sewer network is created. The toy sewer model that is used consists of a few different parts: six manholes with runoff, two boundary nodes, five pipes connecting the manholes to eachother, three internal weirs to regulate waterflow between boundary nodes and manholes. More detail about the specific test model can be found in section 5.2 as well as a visual representation of the sewer network (figure 3)
After finishing the network, it is a good idea to check the flow model, see if there are no complications with the network.

Simulation
Double clicking this box will start the actual simulation. It will either end without a message, meaning the simulation succeeded or it will give an error with an indication as to went wrong.

Using sobek restarts
For the use with OpenDA later, we need to be able to restart a SOBEK model at certain times, these being the assimilation times. In order to do this, there is an option in both the RR module and the 1Dflow module to use and write restart files.

First thing to do is to write some restart files that can be used at the start of the main model. This is done by choosing a precipitation that ends at the start time of the main model and write restart files on this time. For the restart file names we use RESTAR_U for the 1DFLOW/Urban module and RESTAR_R for the RR module. After having created restart files to be used at the beginning of the main model, save these in a different location since they will be overwritten in the main model simulation later. Create a folder where the restart files can be saved, e.g. `c:\sobek_restartfiles\`, and save the following files:

`c:\sobek212\test.lit\WORK\`	`RESTAR_U.RDA`
	`RESTAR_U.RDF`
	`RESTAR_U.RDS`
`c:\sobek212\test.lit\FIXED\`	`RESTAR_R.RDS`
	`RESTAR_R.RRR`
`c:\sobek212\test.lit\CMTWORK\`	`sobek.nda`
	`sobek.ndf`
	`sobek.rtn`

With these restart files saved, set up the main model with using and writing restart files (with the same names in the write and use box).

3.3 Combining Sobek and OpenDA

Now it is time to finally combine SOBEK with OpenDA. This is a tricky process with its own little manual. So we will go through it step by step. The first

requirement is to finish the entire SOBEK model set up, including making the basic restart files and setting up the SOBEK model to use and write restart files at the end of section 3.2. Also it is necessary to keep the SOBEK case screen open, since closing it will clear the `WORK` and `CMTWORK` folders. Let the following be our SOBEK and OpenDA project paths
SOBEK: `c:\sobek212\test.lit\WORK` and `\CMTWORK`
OpenDA: `c:\SobekDA\EnKF.oda` (main config file) and folders `stochModel, algorithm, stochobserver`

step 1. First thing to do, is to create a SOBEK model as described in section 3.2. Go through all the steps in setting up and running the model, including the simulation. When done, do not exit the case screen, since doing so will clear the `WORK` and `CMTWORK` folders but we actually need those files.

step 2. For the next step we assume there's an OpenDA project available, consisting of the main config file and 3 folders: algorithm, stochModel and stochObserver. Now we need to copy both the `WORK` and `CMTWORK` folders to `\stochModel\sobekModel\`

step 3. Copy the basic restart files that have been made at the end of section 3.2 in `c:\sobek_restartfiles\` to `\stochModel\sobekModel\`. One extra thing, the contents of `FIXED` goes into the `\stochModel\sobekModel\WORK` folder.

step 4. From here, we will work in the OpenDA project folder. Select all files in the `CMTWORK` folder and open them all (or at least all non-binary files) using a text editor and

replace `\sobek212\text.lit\WORK`
by `..\WORK\`
replace `\sobek212\text.lit\CMTWORK`
by `..\CMTWORK\`
replace `\sobek212\text.lit\FIXED\RESTAR`
by `..\WORK\RESTAR`

Now open all files in the `WORK` folder and

replace `c:\sobek212\text.lit\WORK`
by `c:\SobekDA\stochModel\sobekModel\WORK`
replace `c:\sobek212\text.lit\CMTWORK`
by `c:\SobekDA\stochModel\sobekModel\CMTWORK`

Changing the paths in these files ensures that the working paths for the SOBEK model have shifted from the original SOBEK project path to our OpenDA project path. And thus any output when running the sequence of SOBEK executables will also appear in the OpenDA project folders.

With all the model files in place it is possible to run a simulation from OpenDA. Let's elaborate on the blackbox setup mentioned in section 3.1, specifically the

blackbox setup for SOBEK. As mentioned, the blackbox wrapper configuration usually consists of three files. For the SOBEK model, these files are:

`bbSobekWrapperConfig.xml`
: This file contains the actions to be performed in order to run the model (by SOBEK), as well as the files related to the input and output that can be used by OpenDA to interact with the model.
The three parts of this file are:

  `aliasDefinitions:` this is a list of definitions, without its actual values assigned, that can be used in the wrapper to call to e.g. a generic input or output file, without actually knowing the name of this file. This is a way to make the wrapper file itself more generic.

  `run:` in this part the sequence of actions to be performed is listed. Starting with making a copy from our template directory to an instance directory (plus instance number). Then there are several SOBEK executables called that work in the instance directory.

  `inputOutput:` the inputOutput part of the file is the reading and writing of several SOBEK files. Linking the SOBEK data files to something that can be used by OpenDA is done by means of several java routines. The classname is specified as well as the file(s) to be read. Depending on the specific data to be read, there are more arguments that have to be given. Finally, an id has to be given to the ioObject so that it can be refered to by the SobekModelConfig file.

`bbSobekModelConfig.xml`
: This file contains the specifications for the model that is to be used. It contains:

  `wrapperConfig:` here the wrapper config file is specified, in this case that would be `bbSobekWrapperConfig.xml`.

  `aliasValues:` here specific values are given for the aliases e.g. the crossSectionFile that will be used is `work\pluvius.alg`.

  `timeInfoExchangeItems:` the exchange items that are used to modify the starttime and endtime of the model. Needed to control the model times in order to assimilate.

  `exchangeItems:` specific model variables that can be accessed by OpenDA. Each exchangeItem will have an id which can either be the name that can be used elsewhere or it can be "allElementsFromIoObject", which will simply take all the exchange items from that specific ioObject. For example, from the timeFile ioObject, the startTime and endTime are taken, to be used in the timeInfoExchangeItems.

`bbSobekStochModelConfig.xml`
: This is the specification of the stochastic model, which consists of two main parts:

  `modelConfig:` set to `bbSobekModelConfig.xml`, which is the model config file that is used.

  `vectorSpecification:` this part is, again, divided into three seperate parts itself:

- the state can be manipulated by an OpenDA filtering algorithm. In the SOBEK case, this would be the Ensemble Kalman Filter.
- the parameters can be adjusted in a calibration algorithm, DUD is used in order to calibrate the SOBEK model.
- predictions are the values on the observation locations that are computed by the model. These are used in both the filtering and the calibration.

Turning data from the SOBEK files into something that OpenDA can use, is done by means of exchange items. These exchangeitems are created using some java routines to read and write the SOBEK files. Most files SOBEK generates are text files, which are easy to read. There are however some binary files, the so called NEFIS files. These files contain some important and useful information, like the state of the system (e.g. waterlevels, flowrate) at a certain time, and are used as input for the SOBEK model. The ability to read and modify these values is a desireble one, which is discussed in section 4.

4 NEFIS

There were already some files that could be accessed by OpenDA as exchange items. The files that contain the information about the state of the system and boundary conditions were not among those though. These NEFIS cannot be accessed by simply reading them as if they were text files. Instead we make use of a library of functions to do so.
The boundary data and the state data each consist of 2 seperate files. A data file and a definition file. The difference between the boundary and state data is that the boundary data is stored in NEFIS (version) 4 files and the state data in NEFIS 5. In order to read these files, there are several tools we need.

nefis.dll
: This dll file is the library containing functions to access the data stored in the NEFIS files. There are two versions of this dll, one for NEFIS 5 files, called nefis.dll, and one for NEFIS 4 and previous versions, called nefis4.dll.

nefislib.java (nefislib5.java)
: A java routine used to call functions from the dll and linking them to methods that can be used in other java routines.

mdafile.java (rdafile.java)
: This java routine uses methods from nefislib and nefislib5 to actually access the NEFIS files. rdafile is used to access NEFIS 5 files (thus using nefislib5.java) and mdafile is used to access version 4 and before NEFIS files (using nefislib.java).

boundaryDataObject.java (stateDataObject.java)
: This java routine makes the exchange items for use in OpenDA.

The boundaryDataObject file had already been created before. However, there was one thing that needed to be changed. For the boundary exchange items, a time stamp is required. In the boundaryDataObject the method setTimes had been used in order to put a timestamp on this exchange item, but this method did not work and actually threw an exception. Instead setTimes, setTimeInfo is used to put timestamps on exchange items. This method has been added to the DoublesExchangeItem class.

A stateObjectFile was still missing, so that has been created. In order to get the state of the system as an exchange item. The specific data that will be retrieved from the rda file are the waterlevels in the nodes and the flowrates in the reaches. The RdaFile was implemented initially in order to retrieve only the waterlevels from the rda file. It has been adjusted in order to also get the flowrate of the reaches.

From OpenDA, it is also possible to dump all the exchange items created by a certain class in the output. For this the ioDumperMain and DataDumperMain are particularly useful. By using these classes as a computeAction in the blackbox wrapper and giving the correct arguments (file, ioObject classname, optional extra arguments if necessary), this action will list all the exchange items created by the particular ioObject classnmae in the output.

5 results

5.1 The restart error

Initially, while already having a working blackbox setup for SOBEK, there were some weird results regarding amount of precipitation when comparing an interrupted or restarted run (to apply data assimilation) with a full run. The amount of total precipitation or a full run did not match the amount of accumulated precipitation from the restarted runs.
At first glance this seemed to be the result of the runoff module missing one timestep, as could be seen in the output of the SOBEK executable. This would only happen with the executable called from OpenDA. After some extensive testing, however, it looks like SOBEK itself induces this error. At certain (re)start times, SOBEK seems to use the wrong precipitation. We are working with constant precipitation at five minute intervals (in mm precipitation per m^2 surface area, see table 1), the testing has been done at start times 0, 1, ..., 20 minutes. Total surface area per node is 12000 m^2, total number of nodes (in the SOBEK test model) is 6, and so the surface of the entire area is 72000 m^2. The precipitation used is `STNBUI01`, a standard precipitation event that is used by SOBEK. The event period of this precipitation is 1-1-1995, 00:00 (start time 0) until 02:00.

Time	Amount	Time	Amount
0	0.3	60	0.15
5	0.6	65	0.15
10	0.9	70	0.15
15	1.2	75	0
20	1.5	80	0
25	1.5	85	0
30	1.05	90	0
35	0.9	95	0
40	0.75	100	0
45	0.6	105	0
50	0.45	110	0
55	0.3	115	0

Table 1: Precipitation used to examine the error. Amounts are mm precipitation per m^2 surface.

In order to view rainfall results, we can check the `rrbalans.his` file (see appendix 6). Starting the simulation at time 0 up to time 4 yields the expected results. Starting at time 5 however gives a strange result. After recalculating the precipitation from the total amount of rainfall, it turns out the precipitation SOBEK actually uses is listed in table 2 and is slightly different from the one SOBEK should use, the one from table 1. In fact, the event has shifted five minutes.
Starting at times 6, 7, ..., 14 yields good results. But starting at time 15 gives the same error as the one at start time 5. Here again, the precipitation event is shifted 5 minutes.
After refining the precipitation interval to one minute instead of five, some more

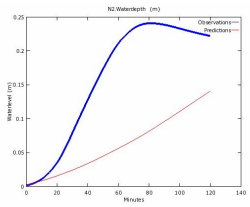
Time	Amount	Time	Amount
		30	1.5
5	0.3	35	1.05
10	0.6	40	0.9
15	0.9	45	0.75
20	1.2	50	0.6
25	1.5	55	0.45

Table 2: Precipitation calculated from SOBEK output.

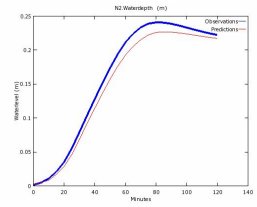
testing is done. As it turns out, the error occurs again at the same times and the precipitation is shifted over the length of one interval. Some further testing shows that this error does not occur on any of the start times 20, 25, . . . , 115.

Investigation of the error with OpenDA

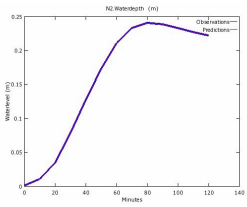
The error has also been investigated by running simulations in OpenDA (in fact the error was first discovered in an OpenDA run, leading to believe there was some OpenDA - SOBEK control error). In figure 2 the model predictions and observations are displayed. This has been done for observation (restart) times with 1, 5, 10, 15, 20 minute intervals. As is obvious from the figures, only restarts done in 10 and 20 minute intervals seem to give results that do not contain this error. The results have been created by use of the OpenDA EnKF project for SOBEK. Though it was necessary to also provide a state parameter with noise, this noise model standard deviation was kept so small that it had no noticeable influence on the results, as can actually be seen from figures 2c and 2e.



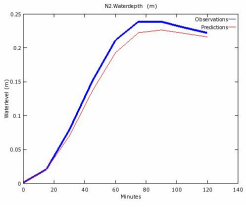
(a) 1 minute interval



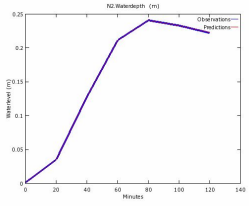
(b) 5 minute interval



(c) 10 minute interval



(d) 15 minute interval



(e) 20 minute interval

Figure 2: Results of simulations using different restart time intervals.

The result of the 1 minute restart simulation, seen in figure 2a, is a very odd one. The reason for this behaviour is not known, but we could make an educated

guess as to what causes this. It would seem that there is a more or less constant precipitation since the speed at which the waterlevel rises only slightly increases as time progresses. this slight acceleration in waterlevel rising speed could then be due to inflow from a neighbouring node.

To wrap up (for now) things about this error, the simulations should be run from OpenDA with 10 minute assimilation (observer) intervals. Other interval sizes are likely to induce this error (though 20 minutes seems to go ok).

5.2 The test model

Testing of the blackbox wrapper has been done using the test model that was very briefly introduced in section 3.2. We will describe the test model in more detail here (see figure 3 for a representation of the test network). First of all

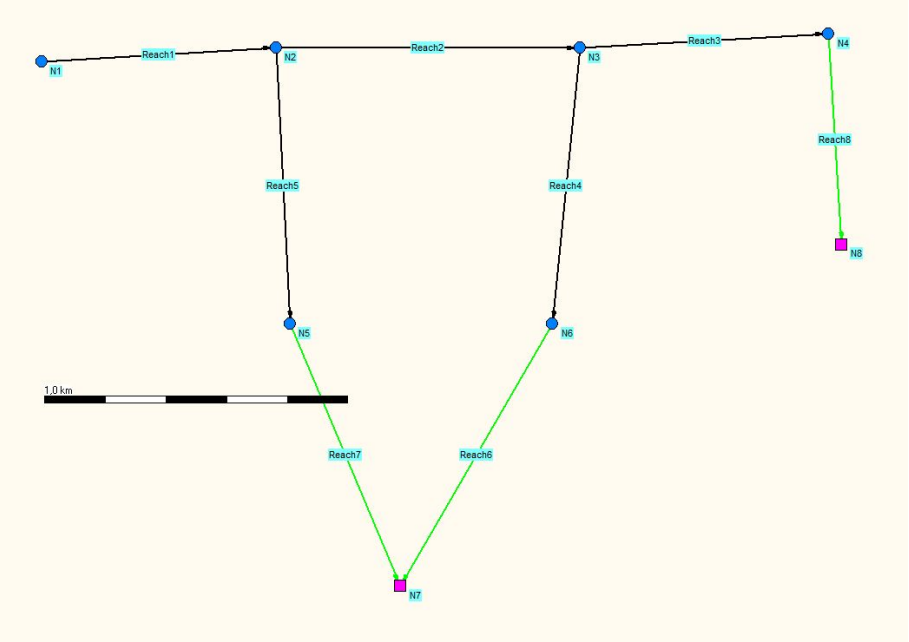


Figure 3: The SOBEK test sewer network

there are six manholes with runoff, denoted by N1 to N6. This means that in each of these nodes, surface areas are defined which catch rainfall that will eventually lead to the sewer. Each node has a total surface area of 6300 m^2 which is divided into several different types of surfaces (i.e. area type: closed paved, open paved, roof and unpaved. All of which can have a certain runoff type: with a slope, flat or stretched flat. Leading to a total of 12 different surface types). Every surface type is in the order 400-600 m^2. The storage area in each manhole is 4 m^2 and the depth is 4 m, so flooding occurs if the waterlevel in a manhole exceeds 4 m.
To connect the nodes, there are pipes (called reaches). There is a total of five reaches to connect the manholes, Reach1 to Reach5. In the test model, the

Reach	Length (m)	Radius (m)	Volume (m^3)
1	776	0.15	17.46
2	1003	0.15	22.57
3	822	0.15	18.50
4	916	0.15	20.61
5	912	0.15	16.02

Table 3: Length, radius and volume of the reaches in the test network.

Boundary node	Value	Connected to	Internal weir	Crest level
N7	0.1	N5	Reach7	0.17
		N6	Reach6	0.19
N8	0.2	N4	Reach8	0.22

Table 4: Information about the boundary nodes and internal weirs.

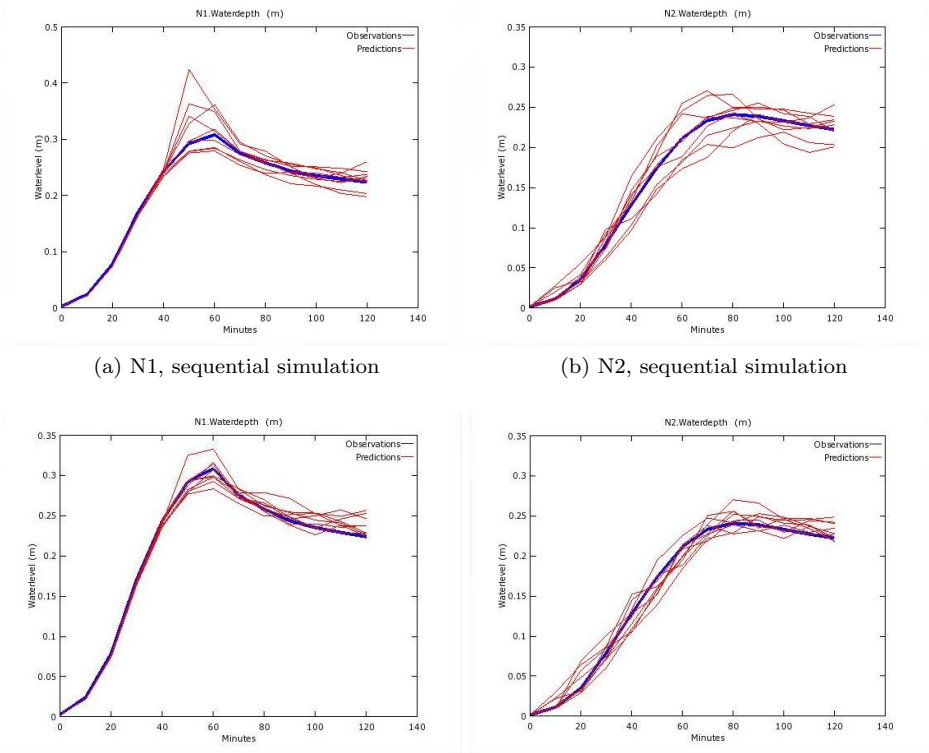
manholes are quite distant from eachother, as can be seen in table 3.
To get rid of excess water, there are boundary nodes connected to manholes by means of internal weirs. The boundary nodes have a constant value (waterlevel height), and water can flow either in or out of the sewer system depending on the crest levels in the internal weirs, the waterlevels in the sewer system and the waterlevel on the boundary (though constant, if we choose to put some noise on the boundary, this level might rise above the crest level). Specifics for the boundary nodes and internal weirs are listed in table 4.
The precipitation used in most (if not all) testing is `STNBUI01`, the same as the one used in section 5.1 (see table 1).

Calibration

The newly set up SOBEK and OpenDA project can be used to do some parameter calibrations. This is done by a sort of twin experiment. Running the model in SOBEK itself creates the observer file `calcpnt.his` that will be used in OpenDA as well as files with other parameter values of course. When the OpenDA project is set up for this model, the `pluvius.alg` file is edited. The 12 numbers that come after "rf" are the runoff parameters. In order to test the calibration, one (or more) of these numbers is changed before running the simulation with OpenDA. The point of this simulation would be to calibrate the parameter that was changed, and see if it is calibrated in such a way that it converges to (or at least approaches) the original parameter value

Simulation

Testing of the newly created exchange items has been done by imposing noise on one or more nodes. Results of one of those simulations are plotted in figure 4. In this particular case, there is only noise on node N2. N1 is plotted as well to illustrate the behaviour of the waterlevel in this node as the waterlevel in N2 varies. It also becomes obvious that by using the Kalman filter, the predictions become better as they are corrected at each analysis step.



(a) N1, sequential simulation

(b) N2, sequential simulation

(c) N1, kalman filter

(d) N2, kalman filter

Figure 4: Results in two nodes (N1 and N2), figures (a) and (b) display water-levels without filtering and figures (c) and (d) with kalman filtering.

The lines plotted are the observations (thick blue line) and the predictions from the different ensemble members (several red lines). It should be noted that the predictions are the ones that come from the blackbox model after a forcast is made, and not the analysed state. This because after the analysis, the waterlevels are adjusted in the `sobek.rda` file and not in the `calcpnt.his` file. The former is the input file for the SOBEK executables, the latter is the file produced by these same executables for visualization only (and in this case, predictions that are compared to observed states).

5.3 Sewer model of Delft

The sewer system model of Delft has been implemented in SOBEK. And can also be used in OpenDA in the same way the test model was used. The sewer network used for Delft is modeled after the real sewer network of Delft and thus a more realistic one than the test model. A part of the sewer network of Delft can be seen in figure 5. Some of the important differences compared to the test model are that in the manholes with runoff, the 12 different surface areas are more realistic. In the test model every surface type was represented almost equally, where in the network of Delft surface types like the unpaved ones are non existent. The bulk of the runoff surface area for the network of

Delft consists of open flat area and roof sloped area. The smaller surface areas are closed flat area and roof flat area.

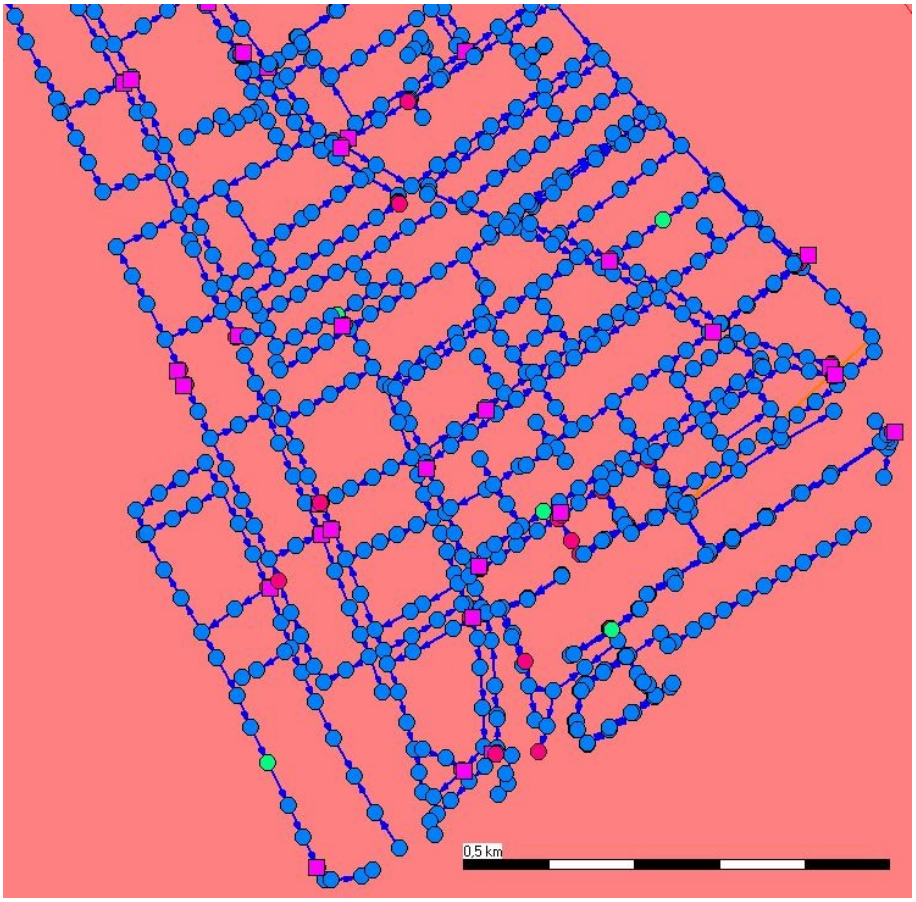


Figure 5: Part of the SOBEK sewer network for Delft

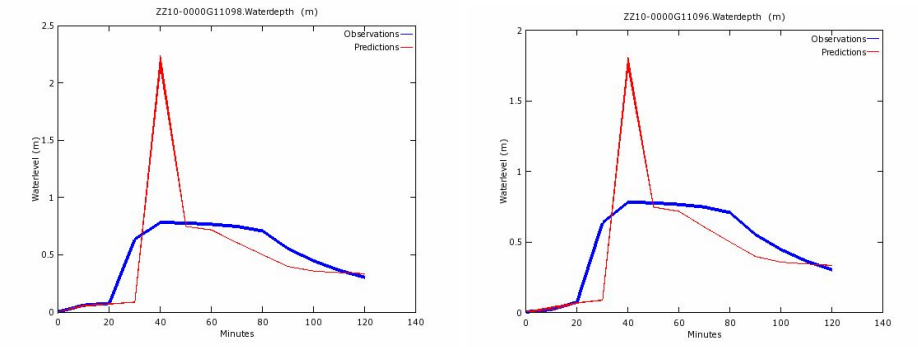
Running a simulation with SOBEK seems to finish alright, though there are some remarks in the `sobek.log` file:

```
Minimum Length set for reachsegment "id" "original number" 1.0
Lateral inflow on closed pipe, storage on inflow node <
                              2.0*pipediam*pipediam
Crest level adjusted to bed level in structure Weir
Extreme differences in bottomlevels of connected reaches in node
```

Despite these messages, the logfile still ends with `Normal end of Sobeksim`. Running this model from OpenDA will give these same remarks in the logfile. However, in some cases for some ensemble members it will not give the normal end of Sobeksim. Instead the logfile ends with

```
** FATAL:
Abnormal end of Sobeksim
```

Despite this Abnormal end of Sobeksim, it is still possible to run simulations using OpenDA successfully. Wether or not the results are actually reliable remains to be seen. The results of an OpenDA EnKF simulation with an ensemble size of 10 can be seen in figure 6. In this case, noise has been put only on node ZZ10-0000G11098. The figure shows observations and predictions in this node (fig. 6a) and in its neighbouring node (fig. 6b). The high waterlevel at time 40 could



(a) ZZ10-0000G11098 (b) ZZ10-0000G11096

Figure 6: Results in two nodes for the Delft sewer system using EnKF.

possibly be due to the noise model not doing what is expected, eventhough the standard deviation of this noise model is rather low. Or it might have something to do with some abnormal end of the Sobeksim. Or... it might be something that has yet to be discovered. Obviously this requires some looking into.

6 Conclusions and further research

The main conclusion is that the basic blackbox set up for in OpenDA for SOBEK works. The NEFIS files can be read and information from them can be used to create exchange items. There are, however, several more things that can or need to be done.

A better understanding of the noise models that are used in OpenDA is needed in order to put the right amounts of noise on the state vectors. These noise models should indicate the uncertainty in the model results instead of being more or less completely random.

Another thing that could be improved is the fact that the analysed state could not be plotted yet but only the forcasted state that come from the model simulation. This requires some editing in the current matlab file that plots the results to also plot the analysed state after the ensemble forcasts.

Parameter calibration works, but needs to be investigated further. Runoff parameters have been used to test the calibration, but waterlevels in nodes seem to be not very sensitive to changes in just one or some of these 12 runoff parameters. Adjusting the observer standard deviation to a very small number (increasing reliability of the observed values) does decrease the difference between the optimal parameter value according to OpenDA and the exact value used in the SOBEK model. Yet there are more parameters that can be calibrated, such as weir crest levels, weir discharge constants, friction in pipes and likely several more. Thouhg parameter calibration would be more interesting in sewer system that contain obstructions.

Boundary value exchange items, taken from `sobek.mda`, can be used in OpenDA in combination with noise models to actually simulate open water levels in a canal so that the water not only flows from the sewer into the canal (when the sewer waterlevel exceeds the crest level), but it is also possible for the canal waterlevel to exceed the crest level and actually have water flow from the canal into the sewer system. Running some simulations while putting noise on these boundary exchange items gives no changes in the model, not even when the noise is so large that the boundary values exceed crest levels of the weirs. Doing this would let water flow from the canal into the sewer system, so it should be quite obvious in the results if something like this would occur. The conclusion is that these boundary value exchange items do not work as intended. It is likely that the values for the boundaries used by SOBEK are taken from another file alltogether. At first glance the used boundary values seem to be taken from `boundary.dat`, but this requires some further research.

Appendix A: Sobek filelist

Since SOBEK will create in the order of 250 files in the `WORK` and `CMTWORK` folders, here is a list of some important files.

In the WORK folder:

calcpnt.his
: This file will contain, among other things, the waterlevels in all the nodes. It will be used in OpenDA as an observer data file, reading waterlevels from it as if they were observed rather than produced by a model. This file will also be used, again in OpenDA, to get predictions (to compare to observations). Obviously the calcpnt.his used as observer is different from the one used for predictions.

rrbalans.his
: This file contains information about the rainfall, amount of run-off and waterbalans in the sewer system.

The .his files are binary files that can be read using ODS_View, which is part of the programs that come with SOBEK. There are more .his files in the CMTWORK and WORK folder, but these 2 are likely the most important and useful.

pluvius.alg
: This is a simple text file which contains several parameters used by SOBEK in its model computations. The 12 numbers that come after rf are the run-off parameters, ms is the initial storage and the other denote the infiltration.

boundary.dat
: This file contains names of boundary elements and their values.

struct.dat and .def
: These files contain data for the weir coefficients and their ID.

In the CMTWORK folder:

grid.tbl
: Grid.tbl contains IDs for the different nodes and reaches (connections) in the network. The reaches are listed in a column in order of first to last. There is an other column that lists node IDs, these are ordered in such a way that they are in the same row as the reach they are connected to. This means that several node IDs may occur more than once in a column, and that every 2 rows have the same reach.

sobek.mda, .mdf
: These are NEFIS files. The .mda file is the data file and the .mdf file the definition file. Among other things, these contain information about the boundary values (which is the data we are interested in for the time being).

sobek.rda, .rdf
: These are also NEFIS files. These contain information about the state of the system. It has a time stamp and waterlevels in each node at that time as well as the flowrate in the reaches.