Averaging Level Control for Urban Drainage System

Yongjie Wang[1] Finn Aakre Haugen[2]

[1]Faculty of Technology, Natural Sciences and Maritime Sciences. Department of Electrical engineering, Information Technology and Cybernetics. University of South-Eastern Norway (USN), Porsgrunn, Norway.
{Yongjie.Wang, Finn.Haugen}@usn.no

Abstract

In the work, averaging level control using model-based control and estimation algorithm on a buffer tank system is studied. Implementation of Model Predictive Control (MPC) and Proportional-Integral (PI) control together with Kalman filter for state and disturbance estimation show decent benefits and potentials. Results show that acceptable setpoint tracking of water level in the basin under varying inflow can be achieved. MPC precedes PI for smoother pump actions. Python as a popular programming language is adopted and showed potential for real-time control (RTC).

Keywords: Averaging level control, MPC, PI, Extended Kalman filter, Urban Drainage System

1 Introduction

Real-time control (RTC) of Urban Drainage System (UDS) is an important part for different goals in the drainage network.

Literature review (Lund *et al.*, 2018) shows that MPC is an efficient tool for UDS control, and there have been a few projects provided promising results, even though the total number of operational implementations is limited. MPC has been used for controlling different components in UDS, including basin, pipe, junction, reservoir, etc. with linear and/or nonlinear models available internally or externally. More than 60 percent of 113 references addressed using MPC for UDS control from 1983 to 2018 in a few cities globally. Reported applications are mostly found in North America and central European countries. In particular, more active research projects can be found in Span, Canada and Denmark.

Kalman filter is an important data assimilation algorithm in weather forecast to combine numerical methods and observations (Sun *et al.*, 2016).

We present results from a research project, which is about potential use of automatic control on an existing UDS in Norway. Figure 1 illustrates a 42 km long tunnel, which is a main component of the drainage system, transports total volume up to 110 million m^3/year combined sewage overflow (CSO) to one of the largest Water Resource Recovery Facility (WRRF) in Norway named VEAS. An equalization magazine downstream the tunnel works as a buffer tank of the wastewater before it enters the VEAS plant, being processed and discharged into the Oslo Fjord.

Due to the process requirement at VEAS and flow control along the tunnel, the combined drainage must be controlled for different purposes:

- Smoothed inflow to the plant.
- Relatively short retaining time of water inside the tunnel.
- The water flow has certain constraints/ limits, i.e., the tunnel should not be total empty, meanwhile, as less overflows into the Oslo Fjord as possible.
- Dealing with precipitation according to weather data/ forecast.

A laboratory buffer tank in Figure 2 is to be used to emulate the actual basin part in the end of the tunnel. Details of the system is to be presented in Section 2.1.

This work aims at:

1. Mathematical modelling of the buffer tank.
2. Averaging level control using model-based control.
3. Inflow estimation using Extended Kalman filter (EKF).

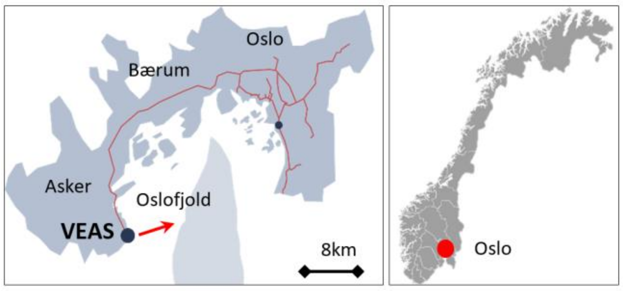

Figure 1. VEAS tunnel for transporting wastewater from urban areas to the treatment plant. (VEAS, 2018)

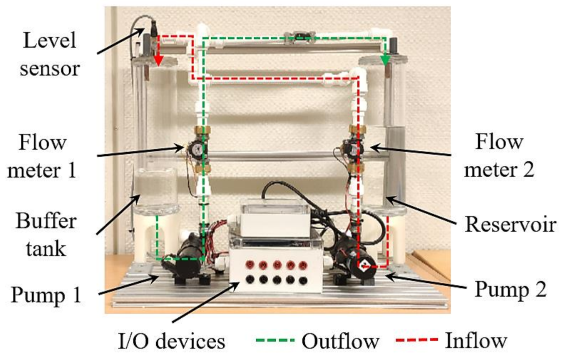

Figure 2. Buffer tank system for averaging level control.

2 Materials and methods

The simulated buffer tank as the testing bench and algorithms for control and estimation are introduced.

2.1 Buffer tank system

Main components of the laboratory buffer tank in Figure 2 consist of:

- Left-hand side: A buffer tank equipped with an ultrasonic level transmitter (Level sensor); a pump (Pump 1) followed by a flowmeter measuring the flowrate of outflow from the buffer tank;
- Right-hand side: A tank as a reservoir of outflow; a pump (Pump 2) to transport simulated varying "precipitation and wastewater" into the buffer tank.
- Bottom center: I/O devices for communication between computer and the buffer tank system. Power supply unit for powering the electronic devices and electrical components. Table 1 lists the main components used in the system.
- Local PI controllers: The flow from each pump is controlled based on readings of inline flow meters.

Table 1. Main components in the buffer tank system for averaging level control.

Component	*Brand/ Model*	*Measurement range*
Pump (x2)	Johnson Pump/ SPXFlow CM30P7-1	<26 L/min
Level sensor	Pepperl Fuchs/ UB300-18GM40-I-V1	35~300 mm
Flow meter (x2)	Sea/ YF-S201	1~30 L/min
I/O device	National Instruments/ USB-6008	AO: 0~5V AI: 0~5V

2.2 Simulation and testing environment

Python (Python.org, 2021) with open libraries for computation and interfaces is used as the simulation and testing environment in this project. The open-access libraries are listed in Table 2.

Table 2. Open-access libraries for averaging level control.

Module	*Function*	*Reference*
Numpy	Matrix and random noise related calculation	(numpy.org, 2021)
Scipy	Optimization	(SciPy.org, 2021b)
nidaqmx	Interface for NI-DAQmx driver	(National Instruments, 2017)
Matplotlib	Data visualization	(Matplotlib.org, 2021)

2.3 Mathematical modelling

The mathematical model of the system can be derived from mass balance of the water tank, given in continuous state-space form by (1):

$$\begin{cases} \dot{h} = \frac{1}{a}(F_{in} - F_{out}) + w \\ y = h + v \end{cases} \qquad (1)$$

where,

- h [cm], the process state variable, the water level inside the tank.
- y [cm], the process output, the water level measurement.
- F_{out} [cm^3/s], the control variable of the pump to manipulate the outflow from the buffer tank.
- a [cm^2], the tank cross-sectional area. In this system, a cylindrical tank is installed vertically so $a = \pi D^2/4$ is a constant, with tank inner diameter $D = 8.5$ cm. In reality, the basin cross-sectional area varies with the water level, i.e., $a = H(h)$, where H is a nonlinear function.
- F_{in} [cm^3/s], inflow into the tank, which in the real VEAS case is unknown.
- w and v are process noise and measurement noise with covariances $Q = E[w^2]$ and $R = E[v^2]$, respectively.

2.4 Control and estimation algorithms

MPC and PI as the control algorithms and EKF as the estimation method are introduced in this section.

2.4.1 Averaging level control

The overall purpose of averaging level control is to smooth the inflow in real-time through the buffer tank so that the variation of outflow from the buffer tank is smoothed (Haugen, 2010). Block diagram of such principle is presented in Figure 3.

The level of water inside the buffer tank is to be maintained close to a user-specified value by manipulating Pump 1 based on the level measurement from Level sensor. Situations like full tank and empty tank are restricted.

The pump control signals are limited to be:

$$u \in [u_{min}, u_{max}] \qquad (2)$$

where, u_{min} and u_{max} are the allowed minimum and maximum flow, respectively. In addition, to have smooth pump actions as the process required, constraint for Δu is introduced as:

$$\left|\frac{\Delta u_k}{T_s}\right| = \left|\frac{u_k - u_{k-1}}{T_s}\right| \leq L$$
$$\Rightarrow u_k \in [u_{k-1} - L \times T_s, u_{k-1} + L \times T_s] \qquad (3)$$

where, L=10 cm^3/s^2 is the user-specified limit, T_s=0.2 s is the time step, the pump action u_k is limited to be:

$$u_k \in [u_{k-1} - 2\text{ cm}^3/\text{s}\,, u_{k-1} + 2\text{ cm}^3/\text{s}]$$

2.4.2 State and inflow estimation

The water level measurement is the only measurement for feedback control in the buffer tank.

As mentioned, the inflow F_{in} in (1) is a random variable representing the unknown precipitation and wastewater flowing into the buffer tank/ basin. Extended Kalman filter (EKF) is used for the inflow estimation in this work (Simon, 2006). To estimate the inflow, the state vector is augmented with inflow disturbance as is given by (4):

$$\dot{x} = \begin{bmatrix} \dot{h} \\ \dot{F_{in}} \end{bmatrix} = \begin{bmatrix} \frac{1}{a}(F_{in} - F_{out}) \\ 0 \end{bmatrix} + \boldsymbol{w} \tag{4}$$

where, $\dot{F}_{in}$ [cm^3/(s^2)] is the first order derivative of the inflow, $\dot{F}_{in} = \frac{dF_{in}}{dt}$. The only measurement $y = h$ is used to update the EKF in this work. $\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \sim \mathcal{N}(0, \delta_{\boldsymbol{w}}^2)$ is assumed Gaussian noise vector for the augmented state vector with zero-means and variance matrix $\delta_{\boldsymbol{w}}^2 = \begin{bmatrix} \delta_{w_1}^2 \\ \delta_{w_2}^2 \end{bmatrix}$ to be determined by experiments.

Process distribution covariance:

$$Q = \begin{bmatrix} Q_h & 0 \\ 0 & Q_{F_{in}} \end{bmatrix} = \begin{bmatrix} \delta_{w_1}^2 & 0 \\ 0 & \delta_{w_2}^2 \end{bmatrix}$$

Tuning of the EKF with covariance matrices is done by using:

$$Q_{KF} = Q, R_{KF} = 10 \times R$$

2.4.3 Discretized state-space model

Based on the model (1), discretization of (4) gives the process model in discretized state-space form as in (5):

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w_k \\ y_k &= Cx_k + v_k \end{aligned} \tag{5}$$

where, the system matrices are

$$A = \begin{bmatrix} 1 & \frac{T_s}{a} \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} -\frac{T_s}{a} \\ 0 \end{bmatrix}, C = [1 \quad 0]$$

with state vector $x_k = \begin{bmatrix} h_k \\ F_{in,k} \end{bmatrix}$, control variable $u_k = F_{out,k}$, time step $T_s = 0.2\text{s}$, process noise vector $w_k \sim \mathcal{N}(0, Q_k)$, measurement noise $v_k \sim \mathcal{N}(0, R_k)$.

The controllability matrix $\mathcal{C}$ and observability matrix $\mathcal{O}$ can be calculated as:

$$\mathcal{C} = [B \; AB] = \begin{bmatrix} -\frac{T_s}{a} & -\frac{T_s}{a} \\ 0 & 0 \end{bmatrix}$$

$$\mathcal{O} = \begin{bmatrix} C \\ CA \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & \frac{T_s}{a} \end{bmatrix}$$

Checking the rank of these matrices, giving:
$Rank(\mathcal{C}) = 1$, h is controllable.
$Rank(\mathcal{O}) = 2$, the system is observable.

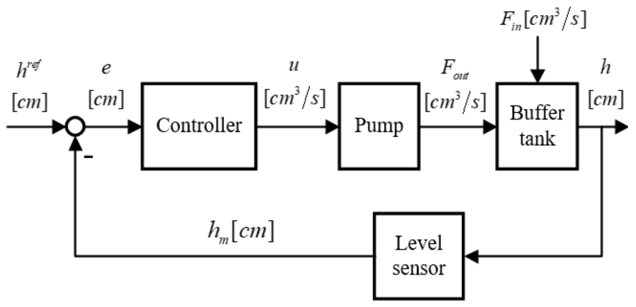

Figure 3. Block diagram of averaging level control.

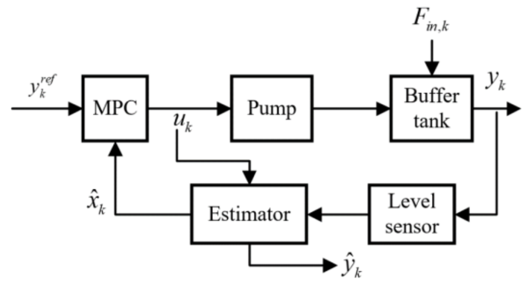

Figure 4. Block diagram of output feedback MPC.

2.4.4 Output feedback MPC

The main idea of implementing MPC is to solve an open-loop optimal control problem over a moving horizon with finite length at each sampling time, starting at the current state. At the next time step, the computation is repeated starting from the new state and over a shifted horizon.

In an output feedback MPC, feedback of states is obtained through an estimator, i.e., EKF, to recursively estimate the states based on the measurement at every time step. The estimated states $\hat{x}_k = \begin{bmatrix} \hat{h}_k \\ \hat{F}_{in,k} \end{bmatrix}$ is sent to MPC, instead of the states measured directly from the process x_k.

Figure 4 shows the block diagram of the principle of output feedback MPC.

For a Single-Input-Single-Output (SISO) MPC, the cost function in this case is defined by (6):

$$\min_{u_k} J = \frac{1}{2}\left\{\sum_{k=1}^{N_p} M \times e_k^2 + \sum_{k=1}^{N_c} N \times \Delta u_k^2\right\} \tag{6}$$

where,

- u_k is the control signal to be optimized. Control error $\Delta u_k = u_k - u_{k-1}$.
- Constraints on u_k and Δu_k are as in (2) and (3).
- $e_k = y_k^{ref} - y_k$, is the prediction error defined as the difference between the setpoint y_k^{ref} and process output y_k.
- N_p and N_c are the length of prediction horizon ($T_{predict} = N_p \times T_s$) and control horizon ($T_{control} = N_c \times T_s$), respectively.
- $M \in \mathbb{R}^{1\times1}$ and $N \in \mathbb{R}^{1\times1}$ are positive definite weighting matrices for prediction error and control error, respectively.

2.4.5 PI control

PI is a control algorithm widely used in various industries. The PI controller in time domain to be used is of the discretized form (Haugen, 2010) in (7):

$$u_k = u_{k-1} + [u_{0,k} - u_{0,k-1}] + K_p[e_k - e_{k-1}] + \frac{K_p T_s}{T_i} e_k \qquad (7)$$

where,

- u_k, controller output at time step t_k with constraints introduced as in (2) and (3).
- u_0, manual control input.
- $e_k = y_k^{ref} - y_k$, where y_k and y_k^{ref} are the measurement and the reference at t_k, respectively.
- K_p, T_i, proportional gain and integral time of the PI controller. Skogestad's method as a model-based tuning method (Haugen, 2010) is used for tuning of these parameters.

3 Results and discussion

In this section, results of averaging level control are presented and discussed.

3.1 Averaging level control using PI

Experiment results are shown in Figure 5 with controller settings: K_p^h= -5.67 (direct actions), T_i^h=20 s.

In general, the water level h showed smooth changes in the meantime of tracking the setpoint h_{sp}, as can be seen from the top plot. The real-time measurement h_m and the estimated signal $\hat{h}$ match each other well, except for slight deviation in the beginning phase (< 10 s) due to a guessed initial value used for the EKF estimator.

PI controller calcucates $F_{out,k}$ based on the estimated inflow $\hat{F}_{in,k}$ for updating the pump actions at each time step. The estimated inflow $\hat{F}_{in,k}$ started to reflect the real data after about 10 s as the middle plot shows. The large deviation of the estimated inflow from the real value in the beginning phase caused delayed controller actions, in addition to the $\Delta u/\Delta t$ constraint for smooth pump actions shown in the bottom plot. Even though the estimation of inflow is noisy, the Pump 1 control actions $u = F_{out}$ slowly resembled the varying inflow, including conditions of two stepwise changes at t=0 s and t=50 s. The Pump 1 control actions fluctuated at low flow rate, i.e., from t=50 s to t=100 s when the inflow is ~20 cm^3/s shown in the middle plot, caused by the nonlinear pump characteristics.

3.2 Averaging level control using MPC

Results of MPC control algorithm are presented in Figure 6, with the following controller settings:

- Optimization method: SLSQP (SciPy.org, 2021a), tolerance = 0.001.
- Constraint/ boundaries: as in (2) and (3).

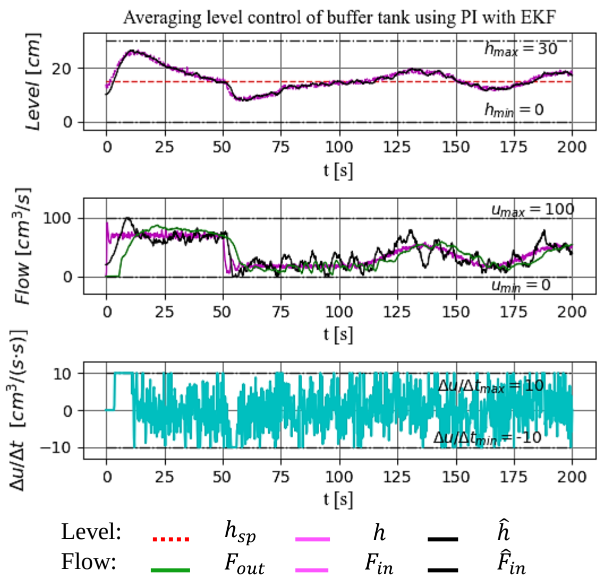

Figure 5. Results of implementing PI for buffer tank averaging level control.

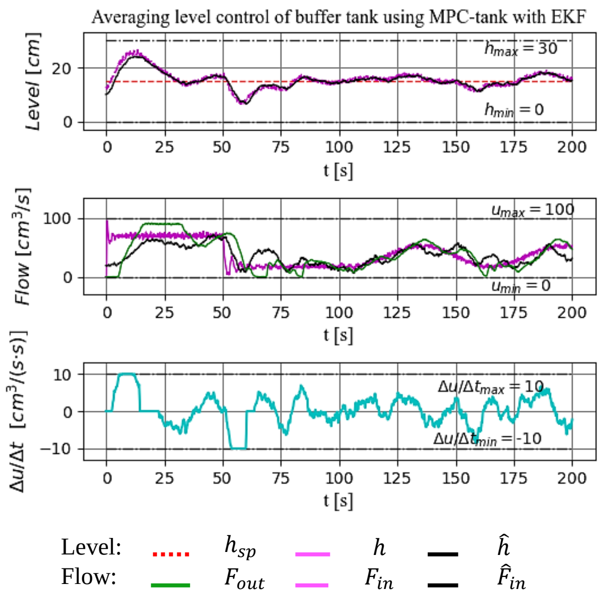

Figure 6. Results of implementing MPC for buffer tank averaging level control.

- $T_{predict} = T_{predict}$ =10 s, and $N_c = N_p$=20.
- Weighting matrices: M = 5, N = 1.
- F_{in}: Estimated inflow at each time step $F_{in,k}$ is used for prediction and cost function calculation, meaning that the inflow is a "fixed" value for the entire prediction horizon N_p.

Comparing the results from MPC to from PI control, one can firstly notice that the MPC outperforms PI with "smoother" F_{out} (middle plot) and much less varied outflow $\Delta u/\Delta t$ (bottom plot). This is because of the moving horizon and optimization algorithm used for optimal control in the prediction horizon, given the same constraints of on F_{out} and $\Delta u/\Delta t$ as for PI controller.

It is worth noting that, with smoother outflow, more overall setpoint deviation of the water level should have been observed since the volume would have varied more to buffer against inflow. However, from the results, the differences are noticeable but not obvious, comparing the two "level" plots in Figure 5 and Figure 6.

Estimation of both water level $\hat{h}$ and the inflow $\hat{F}_{in}$ are about the same overall performance as in PI controlled case. However, the less noisy control actions led to less noisy $\hat{F}_{in}$.

Both PI controller and MPC can be tuned for less tracking error suffering more abrupt control action changes or more varied water level with less fluctuated pump actions.

3.3 "Time" issue in real-time control

One problem of using Python for real-time control (RTC) is how to ensure the time step T_s during the computation, since Python is not designed for real-time purpose and the computation speed is dependent on many factors.

Figure 7 presents the differences between computational load for the averaging level control using MPC and PI, given computer configurations in Table 3. The two upper plots are the cycle time for each cycle (totally 200 s / 0.2 s = 1000 cycles). The lower histogram plots show the distribution of the cycle time for both control algorithms, in which the 1000 cycles are binned into 40 groups with the largest group marked red.

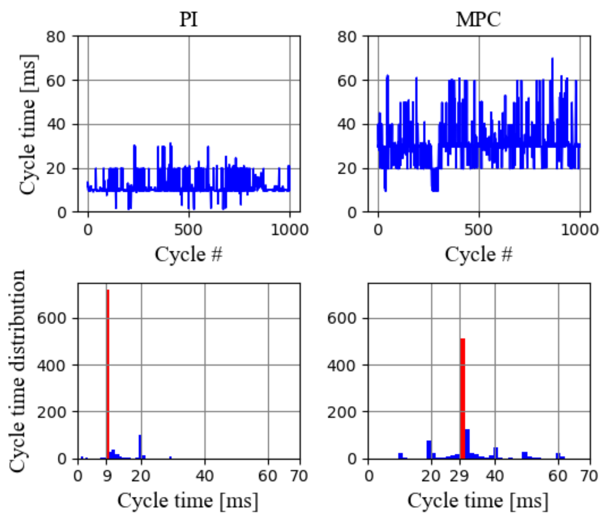

Figure 7. Comparison of computation load of PI and MPC for averaging level control.

Table 3. Computer configurations.

Hardware	Processor: Intel(R) Core(TM) i7-8750H @ 2.20GHz Installed RAM: 32.0 GB
Operating system	Windows 10 Enterprise 64-bit Version 20H2 OS build 19042.985

The computational load is relatively less heavy using PI control than using MPC. In the experiments based on PI control, more than 70% cycles have about 9~10 ms computation time (not the "average" computation time) and maximum cycle time is ~31 ms. For MPC case, more than 50% cycles have 29~31 ms running time and maximum is ~70 ms, both are much higher than using PI control. As can be seen from the histogram, the cycle time distribution for MPC is much "wider", meaning the cycle time varies a lot from loop to loop. This is because MPC is an optimization-based algorithm. The computation depends highly on the problem construction, optimization method, and solver setup, etc. A few other factors can also influence the cycle time including file I/O, hardware communication and setup, computer configurations, etc.

In this work, Python built-in "time" module is used with the following method to handle the issue:

1. Define a time step/ cycle time, i.e., T_s = 0.2 s. The value has to be far greater than the largest actual cycle time, which can be determined from experiments.
2. For each loop:
   a. Count the actual cycle time $T_{cycle,k}$ [s] of each loop k.
   b. "Wait" for $T_{wait,k} = T_s - T_{cycle,k}$ [s] at the end of each loop. In Python, the built-in "time" module can be used as:
   ```
   time.sleep(T_wait_k)
   ```
   c. If for some reason $T_{wait,k} \geq T_s$, let $T_{wait,k} = 0$.
   d. Pump control signal u_k is maintained until next time step.

To speed up the simulation/ shorten the cycle time, some options are recommended: Cython (Cython.org, 2020), multi-threading and multi-processing (Python.org, 2021), use well-accepted open libraries and proper setup of modules, separate file I/O and computational tasks, etc.

4 Conclusions and future development

The work presents a demonstration of using model-based control algorithms for averaging level control of urban drainage system, to be specific, a wastewater equalization magazine using a small-scale buffer tank. The conclusions are:

- Averaging level control using model-based control algorithms is successful, with unknown inflow.
- MPC is based on the system model and optimization solver. PI controller does not require the system model to compute the control action. With proper settings, MPC can achieve smoother control actions than PI control.
- The process can benefit from both PI and MPC algorithms for averaging level control, given constrained control signal with upper/ lower limits

($[u_{max}, u_{min}]$) and allowed maximum change of flow rate ($|\Delta u/\Delta t|$).

- For the buffer tank system, the inflow is observable so that it can be estimated as an augmented state with water level as the only measurement. EKF is easy to implement for the estimation but some effort is required for tuning.
- Python is a promising option for programming for real-time control. As one can see, a number of open libraries are available for the purpose.

To improve the MPC performance in the future, instead of using a “fixed” F_{in} for the entire prediction horizon, a “forecast horizon” N_f can be used to obtain a sequence of “future inflow” for the MPC optimization, i.e., $\left[F_{in,t_k}, F_{in,t_{k+1}}, \dots, F_{in,t_{k+N_f}}\right]$. Choosing of N_f is based on the information available and computation resource. $N_f \geq N_p$ is suggested so that the inflow information is available throughout the prediction horizon N_p for the optimizer. Different algorithms for estimating/ forecasting inflow can be tested, i.e., Particle Filter (PF), Ensemble Kalman filter (EnKF), Moving Horizon Estimation (MHE), neural network, etc. Trade-off between extra computational load and performance should be taken into consideration.

References

Finn A. Haugen. *Basic DYNAMICS and CONTROL*. TechTeach. 2010.

N. S. V. Lund, A. K. V. Falk, M. Borup, H. Madsen, and Peter S. Mikkelsen. Model Predictive Control of Urban Drainage Systems: A Review and Perspective towards Smart Real-Time Water Management. *Critical Reviews in Environmental Science and Technology* 48 (3):279–339. 2018. doi: 10.1080/10643389.2018.1455484.

Dan Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. John Wiley & Sons. 2006.

L. Sun, O. Seidou, I. Nistor, and K. Liu. Review of the Kalman-Type Hydrological Data Assimilation. *Hydrological Sciences Journal* 61 (13):2348–66. 2016. doi: 10.1080/02626667.2015.1127376.

National Instruments. *NI-DAQmx Python API Documentation*. Release 0.5.0. 2017.

VEAS.nu. From Sewage Treatment Plant to Biorefinery - Brochure in English with Key Numbers for 2018. 2018. https://www.veas.nu/global/upload/rBPPQ/files/5155-VEAS-profileringsbrosjyre-A5-ENG-v1-oppslag.pdf.

Cython.org. Cython 3.0a0 Documentation. 2020. https://cython.readthedocs.io/en/latest/. numpy.org. NumPy User Guide. 2021. https://numpy.org/doc/stable/user/index.html.

Matplotlib.org. Matplotlib: Python Plotting - Matplotlib 3.4.2 Documentation. 2021. https://matplotlib.org/.

Python.org. Python 3.9.5 Documentation. 2021. https://docs.python.org/3.9/.

SciPy.org. Minimize(Method=’SLSQP’) - SciPy v1.7.0 Manual. 2021a. https://docs.scipy.org/doc/scipy/reference/optimize.minimize-slsqp.html#optimize-minimize-slsqp.

SciPy.org. SciPy v1.4.1 Reference Guide. 2021b. https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html#rdd2e1855725e-12.