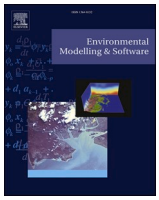
Environmental Modelling and Software 145 (2021) 105175

Contents lists available at ScienceDirect

Environmental Modelling and Software

journal homepage: www.elsevier.com/locate/envsoft



StormReactor: An open-source Python package for the integrated modeling of urban water quality and water balance

Brooke E. Mason *, Abhiram Mullapudi, Branko Kerkez

Department of Civil and Environmental Engineering, University of Michigan, 2350, Hayward St, Ann Arbor, MI, United States

ARTICLE INFO

Keywords:
Water quality
Water balance
Model integration
Real-time control
Stormwater

ABSTRACT

Retrofitting watersheds with sensing and control technologies promises to enable autonomous water systems, which control themselves in real-time to improve water quality. To realize this vision, there is a need to improve the degree of fidelity in the underlying representation of pollutant processes. This paper presents an open-source Python package, *StormReactor*, which integrates the Stormwater Management Model's water balance engine with a new water quality module. *StormReactor* includes a variety of predefined pollutant generation and treatment processes, while allowing users to implement additional processes on their own. To demonstrate the range of possible water quality methodologies that can be modeled, we simulated suspended solids and nitrates in a real and anonymized stormwater network. To illustrate *StormReactor*'s real-time control capabilities, a control strategy was implemented to maximize denitrification. Case study results indicate a controlled asset can achieve the same pollutant improvements as an uncontrolled asset in a quarter of the spatial footprint.

1. Introduction

A reliable and cost-effective method for treating stormwater pollutants is real-time control (Sun et al., 2020; Garofalo et al., 2017; Shishegar et al., 2019). Retrofitting stormwater assets with sensing and control technologies enables watersheds to adapt in real-time to individual storms or pollutant loads (Persaud et al., 2019; Zhang et al., 2018). These smart stormwater assets can be coordinated at the watershed-scale to maximize pollutant treatment (Kerkez et al., 2016; Eggimann et al., 2017; Berglund et al., 2020). In essence, this supports the analogy of transforming our natural or urbanized watersheds into distributed treatment plants by combining knowledge from stormwater systems and process control (Mullapudi et al., 2017). To realize this vision, we must first be able to model both pollutant transformations and the impact of real-time control actions on water quality at the watershed scale (Wong et al., 2006; García et al., 2015; Berglund et al., 2020).This can be achieved with integrated environmental modeling.

Integrated environmental modeling dynamically links distinctly separate models during run-time to better understand the environmental system's response to human and natural stressors (Laniak et al., 2013; Sutherland et al., 2017). Recently, integrated environmental modeling has been used to combine climate and streamflow data with a water budget model and a dynamic groundwater model (Shuler and Mariner, 2020), simulate the hydrological effects of land use changes on karst systems (Bittner et al., 2020), link precipitation forecasts with real-time hydrological and hydraulic modeling for urban flood forecasting (Brendel et al., 2020), couple hydrodynamic and closed nutrient cycle ecological models to predict dissolved oxygen (DO) in surface waters (Suarez et al., 2019), and create a catchment-scale water quality modeling and monitoring framework (Wang et al., 2019).

Integrated environmental modeling of stormwater requires the coupling of water quantity and quality models. This necessitates simulating a number of underlying processes, including precipitation, runoff, climatic variables, land use, flow and pollutant routing, and pollutant transformations (Deletic and Maksimovic, 1998; Egodawatta et al., 2007; McCarthy et al., 2007). While a number of existing models are able to represent these individual components effectively at a granular scale, an all-in-one modeling package is still lacking. Given the complexity of stormwater, specifically its nonlinear dynamics (Overton and Meadows, 2013; García et al., 2015), most existing models understandably seem to draw a line between flow and quality (Obropta and Kardos, 2007; Bach et al., 2014). There has been a stated need to integrate these two types of environmental models (Mullapudi et al., 2017; Wang et al., 2019; Tuomela et al., 2018). To that end, the specific contributions of this paper are:

* Corresponding author.
E-mail address: bemason@umich.edu (B.E. Mason).

https://doi.org/10.1016/j.envsoft.2021.105175
Accepted 17 August 2021
Available online 24 August 2021
1364-8152/Published by Elsevier Ltd.

1. *StormReactor*, a new water quality package implemented as an extension of the popular US Environmental Protection Agency's (EPA) Stormwater Management Model (SWMM), which provides an open-source Python programming interface for simulating complex pollutant generation, treatment, and real-time control processes.
2. An evaluation of the package's ability to model complex pollutant transformations and real-time control actions using two case studies.

These contributions provide researchers and practitioners more flexibility in simulating water quality processes and pollutant-based real-time control at site and watershed scales.

2. State of stormwater quality modeling

Existing stormwater models can be broadly grouped into two categories: water quantity models and water quality models. Most stormwater models primarily focus on coupled hydrologic-hydraulic processes with limited capabilities for modeling water quality (e.g., MIKE URBAN+, DR3M, STORM, MUSIC, SWMM) (Obropta and Kardos, 2007; Bach et al., 2014). However, some stormwater models do focus on high resolution water quality processes. These finite element models (e.g., HYDRUS-CWMI, FITOVERT) simulate complex pollutant transformations within individual sites (Rizzo et al., 2014; Pálfy and Langergraber, 2014; Giraldi et al., 2010). Unfortunately, scaling from site to watershed scale becomes very difficult due to the input data requirements and the difficulty of parameterization. The chasm between these two types of stormwater models forces a trade-off between either comprehensively modeling water quality at the site scale, or less comprehensively modeling watershed-scale processes.

To avoid this tradeoff, researchers have modified existing stormwater models, like SWMM, to expand their pollutant modeling capabilities. SWMM, widely used in the US stormwater community, is an open-source urban stormwater model (Rossman, 2015). SWMM's water quality model provides users the ability to introduce pollutants and pollutant treatment, while also routing and calculating mass balance for each pollutant (Rossman and Huber, 2016). *SWMM-TSS* modified SWMM to simulate total suspended solids (TSS) transport, accumulation, and erosion in sewers and retention tanks (Sun et al., 2017). As its name implies, this modification is only for TSS. Baek et al. (2020) modified SWMM's water quality module for low impact development (LID) to include straining, decay, and decomposition of pollutants. However, this modification does not work for stormwater storage assets or links. Talbot et al. modified PCSWMM, a licensed version of SWMM, to simulate sediment loading due to soil erosion (Talbot et al., 2016). This modification is not open source and thus not open for exploration or expansion by the community. All of these packages are very useful for specific modeling tasks; however, they do not offer general water quality modeling solutions.

Although these packages provide additional functionality, SWMM has many remaining pollutant modeling limitations that must be addressed. The water quality module is limited by the range of treatment measures that can be modeled (Wong et al., 2006), specifically, limited nutrient treatment capabilities inside storage nodes (e.g., basins, wetlands) (Troitsky et al., 2019; Niazi et al., 2017). SWMM cannot simulate pollutant treatment inside links (e.g., conduits, channels) or pollutant generation processes (e.g., resuspension, erosion) inside any stormwater asset. Pollutant treatment cannot be turned on or off based on site conditions or other parameters, requiring treatment to run for the entire simulation. All of these constraints limit a user's ability to model complex pollutant transformations, necessitating a more generalizable and scalable approach.

Aside from water quality limitations, many stormwater quality models have limited or no ability to simulate real-time control. Real-time control is made possible through the installation of sensors (which can monitor the flow and quality parameters) and actuators (which can control the flow of water) (Kerkez et al., 2016; Schu et al., 2004). To realize the goal of autonomous watersheds, we must be able to model real-time control strategies (García et al., 2015; Vanrolleghem et al., 2005). One open-source and popular real-time control package is PySWMM, a Python wrapper for the SWMM computational engine. PySWMM queries stormwater states directly from SWMM, which is used to apply control actions by setting the control parameters for valves, gates, and pumps in real-time (Mcdonnell et al., 2020). However, PySWMM presently only enables real-time control decisions to be made based on water quantity parameters (e.g., flow, head, depth, volume). Therefore, there is a need for a comprehensive package that can both simulate water quality processes and real-time control.

3. New package for modeling stormwater quality

A watershed-scale pollutant transformation model is comprised of the water quantity and water quality representations of the stormwater network (Fig. 1). These representations provide insight into which sub-components are already well addressed by existing models, and which others should be expanded or developed. The water quantity representation focuses on the conveyance of water through the network of links (e.g., channels, conduits) and nodes (e.g., detention basins, retention basins, wetlands). The hydrologic and hydraulic processes, which underpin the water quantity sub-component, are well established in stormwater models (Obropta and Kardos, 2007; Bach et al., 2014). The water quality representation includes the pollutant generation and treatment processes that occur in stormwater assets (e.g., wetland as a continuously tank reactor (CSTR), retention basin as a settling tank). Often, this sub-component is significantly simplified (e.g., first order decay models) instead of drawing from water treatment process literature (Mullapudi et al., 2017), leaving room for expansion.

Guided by the state of these sub-components in current stormwater models, we developed *StormReactor*, a new water quality Python package, coupled with SWMM. The choice to build a module for SWMM was based on a number of factors. First, SWMM has a verified hydraulic solver, which is critically important for accurately modeling flow and pollutant routing (Rossman, 2015). In addition, building upon SWMM's popularity engages a large user base ensuring it is accessible to more people. Finally, SWMM is open source, which enables modification of its code and the use of popular Python wrappers, such as PySWMM.

Section 3.1 and Section 3.2 detail the development and structure of *StormReactor*. *StormReactor* was created by (i) modifying the SWMM and PySWMM source code to allow water quality states to be modified and (ii) building an additional Python library to interface water quality modeling with these popular tools.

3.1. SWMM and PySWMM

To address the limitations of SWMM's water quality module, we modified SWMM's C source code[1] by introducing *getters* and *setters* to allow for real-time access of the model states during simulation (Table 1). A *getter* enables a user to access a variable while a *setter* enables a user to change the value of a variable. We then modified PySWMM's Python source code[2] to gain access to SWMM water quality states and to provide the convenience of modeling in a popular scripting language. While PySWMM already allowed for the interaction with SWMM's quantity states (e.g., flows, depths), it needed to be expanded to support interaction with water quality states (Table 1). Now a user can interact with a pollutant's concentration in any node or link during any routing time step. In this way, SWMM is used to transport pollutants using its reliable hydraulic and routing engine, PySWMM is used to support Python interaction with SWMM's C engine, and *StormReactor* adds supplementary support for water quality modeling (Fig. 2).

[1] https://github.com/OpenWaterAnalytics/Stormwater-Management-Model.
[2] https://github.com/OpenWaterAnalytics/pyswmm.

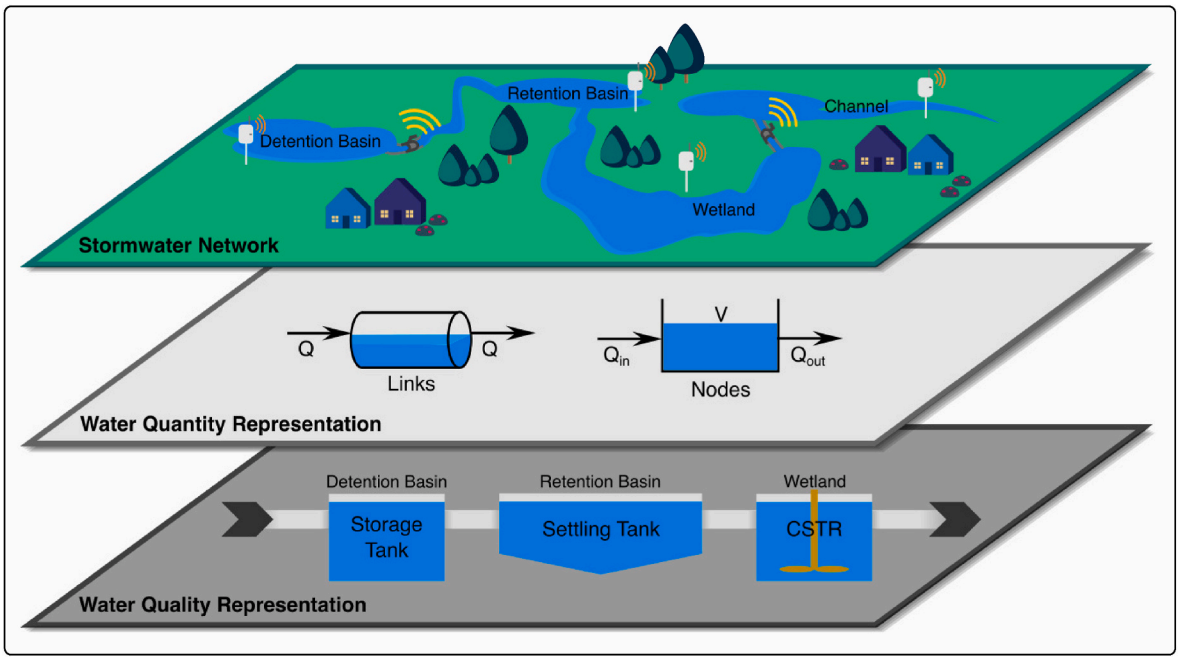


Fig. 1. A watershed-scale pollutant transformation model is comprised of the water quantity and quality representations of the stormwater network. The water quantity representation, often modeled by SWMM, focuses on the conveyance of water through the network of links (e.g., channels, conduits) and nodes (e.g., detention basins, retention basins, wetlands). The water quality representation, often modeled using water treatment plant process literature, focuses on the water treatment processes that occur in stormwater assets.

Table 1
The *getters* and *setters* added to both SWMM and PySWMM.

Variable	Type	Description
NODEQUAL	*Getter*	current pollutant concentration in a node
NODECIN	*Getter*	inflow concentration in a node
NODEREACTORC	*Getter*	updated concentration after the mass balance of flows and pollutants in a node
NODEHRT	*Getter*	hydraulic residence time (hours) in a node
LINKQUAL	*Getter*	current pollutant concentration in a link
TOALLOAD	*Getter*	total quality mass loading in a link
LINKREACTORC	*Getter*	updated concentration after the mass balance of flows and pollutants in a link
Node.extQual	*Setter*	current pollutant concentration in a link
Link.extQual	*Setter*	current pollutant concentration in a node

3.2. StormReactor

StormReactor enables users to model water quality, while fully leveraging the well validated SWMM functionality for flow and routing. *StormReactor* provides a high-level programming interface that removes the user from the complex interactions between SWMM, PySWMM, and *StormReactor*, and only requires a few Python command statements to model pollutant transformations. Users have the ability to select a water treatment method in any stormwater asset and specify the routing time steps across which to carry out simulations. To promote uptake by an existing community of modelers, a user can select any of the already existing SWMM treatment functions outlined in the *SWMM Reference Manual Volume III: Water Quality* (Table 2) (Rossman and Huber, 2016). Users can also select from a library of our new water quality methods, including reactor models and stream processes, such as erosion (Table 2). More importantly, users can implement their own custom pollutant models using a Python interface (Section 3.2.3). These custom pollutant models can be built upon states of the various water quantity and quality parameters in SWMM (e.g., flow, depth, volume, concentration) as well as interact with other Python packages (e.g., SciPy). Readers are directed to Zenodo for *StormReactor*'s source code and documentation (Mason and Mullapudi, 2021).

3.2.1. User experience

StormReactor can be installed using pip.[3] To use *StormReactor*, first import both *StormReactor* and PySWMM (Fig. 3). Next, define a configuration dictionary stating at which nodes and links water quality will be modeled, as well as the desired pollutants, water quality methods, and the parameters required for each method. Then, create an instance of the water quality class by calling WaterQuality() which takes two arguments: config, the configuration dictionary; and sim, a PySWMM simulation object, which encapsulates all the SWMM simulation functionality (e.g. start/stop simulation, get/set attributes). Finally, call the class instance method updateWQState() to run the desired water quality method.

Once initialized, *StormReactor* executes the simulation loop. First, *StormReactor* queries the necessary water quantity and quality parameters (e.g., water depth, pollutant concentration) for specific stormwater assets at the current routing time step. Next, it uses the queried parameters to compute and set the new pollutant concentration using a predefined or custom water quality method. If a water quality computation requires a time parameter, the length of the routing time step is used. If real-time control is being modeled, selected water quality and/or quantity data are used to calculate the control decisions. SWMM then enacts the real-time control decisions and routes the pollutant(s) and flows through the network. This process can be repeated at any or every routing time step. The simulation loop terminates after the number of desired routing time steps or the SWMM model is complete.

3.2.2. Architecture

StormReactor's architecture follows an object-oriented programming paradigm. This matches already popular Python conventions and maximizes potential for user customization. *StormReactor* begins by defining a class: WaterQuality(). The class has an _init_ method which takes three parameters: self, an instance of the class; sim, the PySWMM simulation object; and config, the configuration dictionary. When an

[3] https://pypi.org/project/stormreactor.

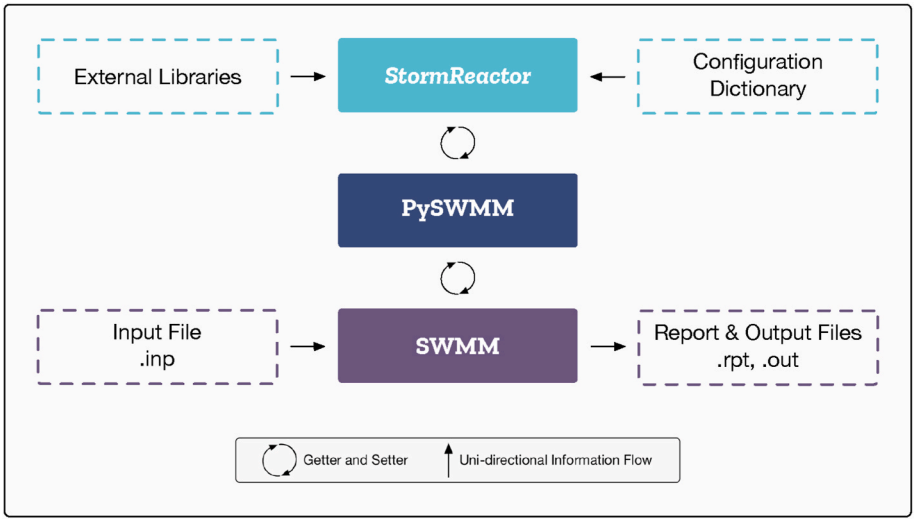


Fig. 2. *StormReactor* follows an object-oriented programming paradigm. This modular approach allows for modifications and reuse by users. *StormReactor* uses a configuration dictionary and can work with external Python libraries. *StormReactor* interacts with PySWMM which interacts with SWMM all via *getters* and *setters*. SWMM requires an input file and then when a simulation is complete, it creates the report and output files.

Table 2
Overview of the current water quality methods that can be selected from *StormReactor* including a method explanation and the asset type (node, link, or both) it can be used for.

Water quality method	Asset type	Method explanation
Event Mean Concentration	Both	Treatment results in a constant concentration
Constant Removal	Both	Treatment results in a constant percent removal
Co-Removal	Both	Removal of some pollutant is proportional to the removal of some other pollutant
Concentration-Dependent Removal	Both	When higher pollutant removal efficiencies occur with higher influent concentrations
Nth Order Reaction Kinetics	Both	When treatment of pollutant X exhibits nth order reaction kinetics where the instantaneous reaction rate is kC^n
k-C* Model	Node	The first-order model with background concentration made popular by Kadlec and Knight (1996) for long-term treatment performance of wetlands.
Gravity Settling	Both	During a quiescent period of time within a storage volume, a fraction of suspended particles will settle out
CSTR	Node	CSTR is a common model for a chemical reactor. The behavior of this CSTR is modeled assuming it is not in steady state because outflow, inflow, volume, and concentration are constantly changing.
Erosion	Link	Engelund and Hansen (1967) developed a procedure for sediment transport in streams.

instance of the class is created, it automatically calls the _init_ method, which does the following: (1) initializes the asset flag; (2) calls the PySWMM method sim.start_time to get the start time of the simulation; (3) initializes the variable last_timestep to aid in calculating the length of the routing time step; (4) initializes the ordinary differential equation (ODE) solver for the CSTR water quality method; and (5) defines the callable names of the water quality instance methods. The WaterQuality () class also defines two important methods: updateWQState() and updateWQState_CSTR(), which update the pollutant concentrations during a SWMM simulation for non-CSTR and CSTR methods, respectively. The class also has a collection of Python instance methods which specify the various treatment and generation processes that can be performed on a pollutant (Table 2).

Time steps are handled by *StormReactor* by relying on SWMM. Many of the treatment methods do not require a time parameter (e.g., event mean concentration, constant removal, k-C* method). *StormReactor* handles these methods just as they would be handled in native SWMM. These methods grab the current pollutant concentration and then calculates and sets the new concentration at the end of the current routing time step. For the methods that do require a time parameter (e.g., N-th order reaction kinetics, erosion, gravity settling), *StormReactor* computes the routing time step length (dt) using the same method as SWMM. To calculate dt, *StormReactor* calls the PySWMM function sim.current_time to get the current simulation time, subtracts the previous routing time step saved in the variable

last_timestep, and then converts it to seconds. In this way, *StormReactor* is dependent on SWMM to get dt. Once dt is calculated and the current concentration is queried, the new concentration is computed and set at the end of the current routing time step. This new concentration then becomes the concentration at the beginning of the next routing time step. Routing time steps are usually on the order of seconds, whereas water quality processes may take much longer. Therefore, users must also parameterize water quality coefficients on the order of seconds.

3.2.3. *Implementing custom pollutant models*

To implement a new custom pollutant model, users can either (1) add their new class instance method to *StormReactor*'s code base or (2) build their model directly in their Python script using the appropriate *getters* and *setters* (Table 1). We recommend the first option if code is to be more seamlessly shared with others. To add a new method to the code base a user must:

1. Define the new method using the following convention: _NewMethod (self, ID, pollutantID, parameters, flag). Non-public Python instance methods should always start with an underscore. The new method requires five parameters: self, an instance of the class; ID, the node or link name in SWMM; pollutantID, the pollutant index in SWMM; parameters, the water quality method parameters; and flag, used to determine if the method is for a link or node.
2. Provide a text description of the method including the water quality method parameters and their required units. Be sure to note if the method is for links, nodes, or both.
3. Write the pollutant transformation code for the new method.

```python
# import packages
from StormReactor import WaterQuality
from pyswmm import Simulation

# build water quality configuration dictionary
config = { 'detention_basin': { 'pollutant': 0, 'method': 'GravitySettling', 'parameters': {'k': 0.0005,'C_s': 21.0}, \
    'retention_basin': { 'pollutant': 0, 'method': 'GravitySettling', 'parameters': {'k': 0.0005,'C_s': 21.0} }, \
    'wetland': { 'pollutant': 1, 'method': 'CSTR', 'parameters': {'k': -0.000089,'n': 1.0,'Co': 0.0} }, \
    'channel': { 'pollutant': 0, 'method': 'Erosion', 'parameters': {'w': 10.0,'So': 0.037,'Ss': 1.6, 'd50': 0.04} }, \
               { 'pollutant': 0, 'method': 'GravitySettling', 'parameters': {'k': 0.0005, 'C_s': 21.0} } }

# initialize water quality
with Simulation ('example.inp') as sim:
    WQ = WaterQuality(sim,config)

    for step in sim:
        # update each routing time step
        WQ.updateWQState()
```

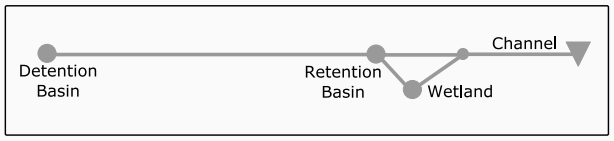


Fig. 3. A Python code snippet that illustrates how some of the TSS and nitrate methods from the two case studies were implemented using *StormReactor*. The package is imported and the configuration dictionary is defined. The configuration dictionary includes the node/link IDs from the SWMM input file, the pollutant indices based on the order in which they are defined in the SWMM input file, the pollutant transformation methods selected, and the required pollutant transformation parameters. The methods are initialized by calling waterQuality(sim,config) and the pollutant transformations are computed by calling updateWQState() each routing time step.

(a) Define any variables that may be needed for the pollutant transformation calculations.
(b) Query SWMM variables that are necessary for the computation (e.g., pollutant concentration, water depth, current simulation time) using PySWMM *getters*.
(c) Compute the pollutant transformation concentration.
(d) Set the new pollutant concentration using PySWMM *setters*.

4. Define the callable name in the _init_ method.
5. Write unit tests for the new method and add them to test_links.py and/or test_nodes.py in the tests folder.

Once the new method is added to *StormReactor*'s code base, the user can then use it following the steps outlined in Section 3.2.1.

4. Water quality case studies

The study area is a 7.8 km^2 urban, separated stormwater network (Fig. 1) located in Michigan, which suffers from erosion problems due to high flashy flows. In this network, stormwater first flows through a detention basin into a long channel. A detention basin has its outlet at the bottom of the basin so between storms it is usually dry. The long channel then flows into a retention basin. A retention basin has its outlet at a higher point so it tends to retain a permanent pool of water. If the height of the water in the retention basin is less than a specified threshold, water flows directly into a constructed treatment wetland. Otherwise, water bypasses the wetland and overflows into another channel. Water leaving the wetland flows into the same channel as the overflow from the retention basin. The end of this channel is considered the outfall of the stormwater network.

For the two case studies, we isolated the network described above from a calibrated SWMM model of the larger, regional stormwater network. Since we removed the upstream assets from the model, we added inflows to simulate the real system response. The network was forced with a 5-year, 12-h storm, which corresponds with design guidelines in the study region (Wong and Kerkez, 2018). Readers are directed to Zenodo for the SWMM input files and simulation code (Mason, 2021a,b).

We provide these case studies to illustrate the following capabilities of *StormReactor*: (1) *StormReactor* can model SWMM's pollutant treatment equations as if we used SWMM's water quality module directly; (2) *StormReactor* can model new water quality processes (e.g., channel erosion, CSTRs in series); and (3) *StormReactor* enables water quality-based real-time control actions. The first case study uses TSS to illustrate the first two capabilities (Section 4.1) and the second case study uses nitrate to demonstrate the third capability (Section 4.2).

4.1. TSS case study

TSS (often measured as concentration in mg/L) is a commonly monitored pollutant because it negatively impacts water quality. These impacts include increasing turbidity, inhibiting plant growth, reducing species diversity, as well as providing transportation for nutrients and heavy metals (Shammaa and Zhu, 2001; Schilling et al., 2017; Dong et al., 1984). To mitigate these negative impacts, researchers and practitioners must be able to model deposition, erosion, and transport processes. Section 4.1.1 details how *StormReactor* was used to model these TSS processes and Section 4.1.2 provides the simulation results and discussion.

4.1.1. TSS methods

Gravity settling was assumed to occur in the wetland, basins, and channels. We selected the gravity settling equation from the *SWMM Reference Manual Volume III: Water Quality* to illustrate how *StormReactor* allows users to model and match existing SWMM treatment equations (Rossman and Huber, 2016). The gravity settling equation is defined as:

$$C = C^* + (C - C^*)exp(-k\,\Delta t/d) \quad (1)$$

The values for the steady state concentration ($C^* = 21$ mg/L) and the settling velocity ($k = 0.0005$ m/s) were selected based on prior monitoring campaigns in the region. At each routing time step (Δt), depth (d) was queried from SWMM and the current concentration (C) was computed.

Along with gravity settling, erosion was also assumed to occur in both channels. Many equations exist for modeling erosion and sediment transport, many of which can be implemented in our library. For illustration purposes, we selected the Engelund-Hansen sediment transport formula (Engelund and Hansen, 1967).

The formula of Engelund and Hansen formula (1967) can be expressed as:

$$f \cdot \varphi = 0.10^{5/2} \quad (2)$$

where

$$f = (2 \cdot g \cdot d \cdot S_o) / v^2 \quad (3)$$

$$\theta = (d \cdot S_o) / [(S_s - 1) d_{50}] \quad (4)$$

$$q_t = \varphi \left[(S_s - 1) g \cdot d_{50}^3 \right]^{1/2} \quad (5)$$

where f is a friction factor, φ is a dimensionless sediment transport function, θ is a dimensionless shear parameter, g is gravitational acceleration, d is hydraulic depth, S_o is channel slope, v is mean channel velocity, S_s is specific gravity of sediment, d_{50} is mean particle diameter, and q_t is total bed-material sediment discharge by weight per unit width (USDA, 1983; Wu et al., 2004). The values for mean particle diameter ($d_{50} = 0.04$ mm), sediment specific gravity ($S_s = 1.6$), and channel slope (0.037–1.8 m/m) were selected based on site data. At each routing time step, the required parameter values were queried from SWMM, the sediment discharge concentration was computed, and the new TSS concentration was set in SWMM.

Root mean square error was used to validate both settling and erosion in the nodes and links. For gravity settling in the nodes, root mean square error was calculated for the cumulative TSS load from the *StormReactor* simulation and a native SWMM simulation (Fig. 4). The root mean squared error was zero for all three nodes. Since treatment in SWMM links is a new feature of *StormReactor*, gravity settling and erosion in the channels had to be validated differently. The load leaving the channel was compared to the load entering the outfall. The root mean squared error was 6.19E-13.

TSS concentrations measured directly downstream of our outfall average 21 mg/L during steady state conditions and 175 mg/L during storm conditions. For our simulation, TSS was assumed to follow an event mean concentration (EMC) wash-off model (Rossman, 2015). Since this network is dominated by channel erosion and not subcatchment wash-off, the steady state EMC was used in the wash-off model. The additional TSS needed to match storm event concentrations was provided by the erosion model.

4.1.2. TSS results and discussion

Results show that this system is dominated by erosion processes with only small reductions due to gravity settling (Fig. 5). The detention basin's TSS concentration averaged 13 mg/L due to the small EMC used in the wash-off model. The retention basin saw higher concentrations throughout the simulation, with an average TSS concentration of 121 mg/L. This was a result of significant erosion occurring in the channel that connects the two basins. The wetland's TSS concentration was lower than in the retention basin, but still averaged 100 mg/L during the simulation. The reduction was due to settling in the wetland. The outfall's average TSS concentration was 107 mg/L. The increase in concentration at the outfall was again due to channel erosion occurring between the wetland and the outfall.

StormReactor improved TSS process representation by including channel erosion. Prior to *StormReactor*, users could not model pollutant generation processes unless they modified the parameters in the SWMM build-up and wash-off equations. In our case study, this would have not reflected reality because it would have resulted in high TSS

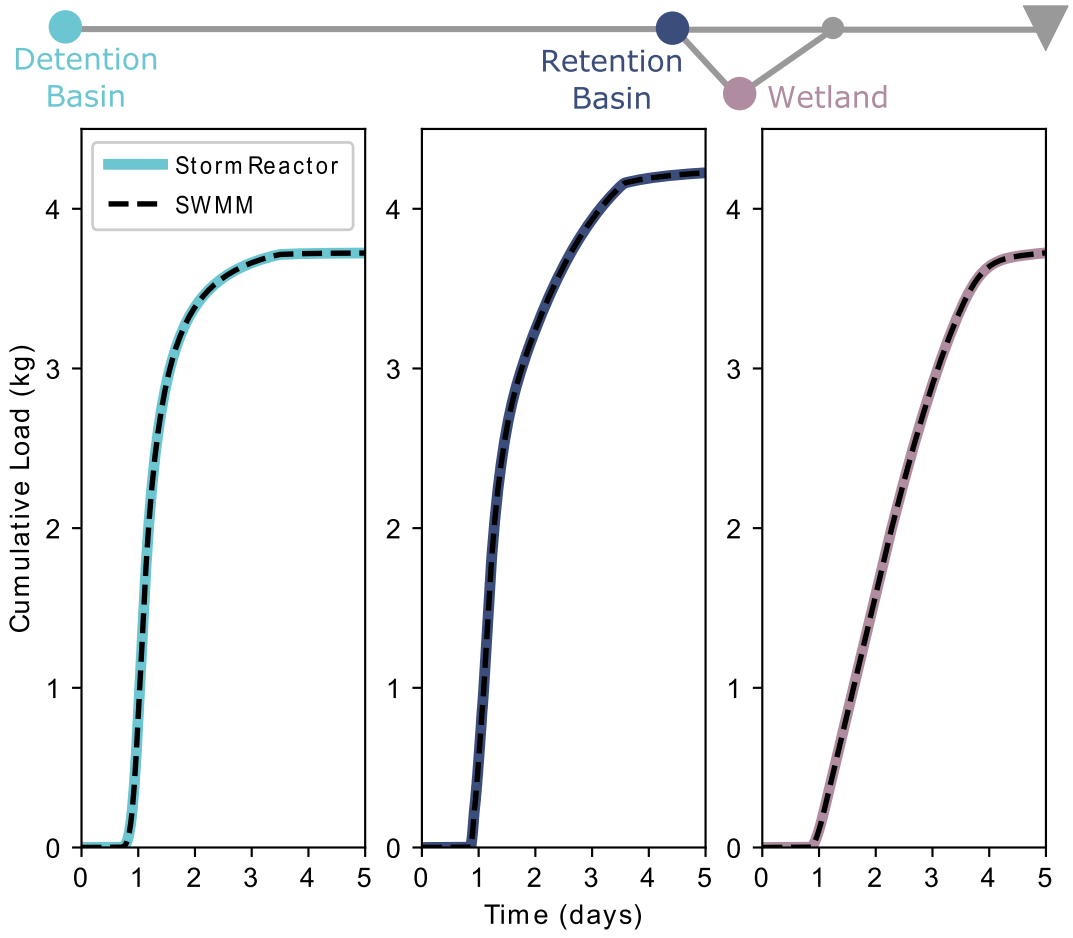


Fig. 4. Cumulative TSS load comparing gravity settling using SWMM's traditional water quality module and *StormReactor*'s water quality module for the detention basin, retention basin, and wetland. Root mean squared error was zero for each asset.

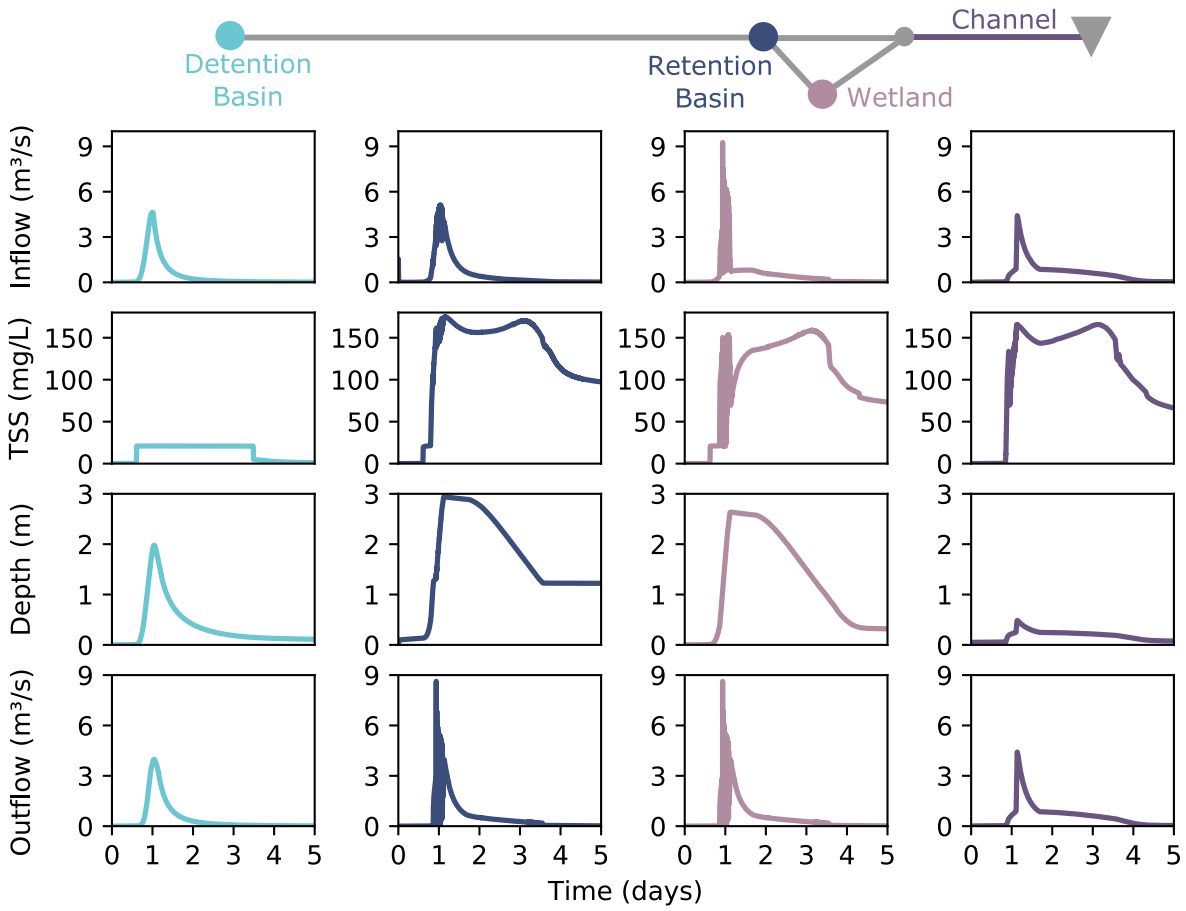


Fig. 5. Simulation results for the various assets in the stormwater network including inflow rate (top panel), TSS concentration (second panel), storage depth (third panel), and outflow rate (bottom panel).

concentrations in the detention basin. Since most of the TSS added to this system comes from downstream channel erosion, high TSS concentrations should only be found in the downstream assets. *StormReactor* now provides the ability to model pollutant generation processes in the assets in which they occur.

The TSS simulation took 42.35 s on a 2018 MacBook Pro (Processor: 2.2 Ghz 6-Core Intel Core i7; Memory: 16 GB 2400 MHz DDR4) as compared to 6.75 s without the TSS model. As we scale to larger networks, future work must evaluate the computational efficiency of *StormReactor*.

4.2. Nitrate case study

Excess nitrogen can cause water quality impairments, such as eutrophication, harmful algal blooms, and fish kills (Conley et al., 2009; Howarth and Paerl, 2008). In order to mitigate these negative impacts, researchers and practitioners must be able to model the nitrogen cycle. This is presently not possible in models like SWMM, because the multiphase, multicomponent reactions which are affected by the aerobic/anoxic conditions in the network cannot be simulated (Troitsky et al., 2019; Niazi et al., 2017).

Section 4.2.1 details how *StormReactor* was used to model nitrate. Section 4.2.2 explains the addition of real-time control, which will control the stormwater network in response to water quality states. To our knowledge, this case study is the first to model nitrate treatment through real-time control at the scale of an entire stormwater network.

4.2.1. Nitrate methods

Modeling nitrogen interactions in stormwater is difficult because nitrogen exists in various forms (e.g., nitrate, nitrite, particulate nitrogen, ammonia, ammonium, dissolved organic nitrogen, nitrogen gas) and undergoes numerous transformations (e.g., denitrification, nitrification, ammonification, fixation, and dissimilatory reduction) (Troitsky et al., 2019). In stormwater basins and wetlands, nitrogen is typically removed through three main mechanisms: assimilation, sedimentation, and denitrification. However, the primary mechanism is denitrification (Yang and Lusk, 2018). High denitrification rates are a result of high nitrate concentrations, low DO concentrations, and readily available sources of carbon (e.g., decaying plants and grass) (Kadlec and Wallace, 2009; Perryman et al., 2011).

For this case study, we focused only on nitrogen in the form of nitrate and therefore, denitrification as the primary removal mechanism. We selected nitrate because site data and other studies indicate runoff is dominated by this form of nitrogen (Kadlec and Wallace, 2009). Denitrification was assumed to occur only in the wetland because wetlands tend to have large quantities of biomass and thus higher denitrification capacity than other storage nodes (White and Reddy, 2009; Scholes et al., 2008). Since this case study assumed high nitrate concentrations and readily available sources of carbon, DO became the limiting factor for denitrification, necessitating us to model DO concentrations as well.

The wetland DO model was implemented using the CSTR method in *StormReactor*. Based on findings by Kadlec (2010), we assumed the wetland functioned as three CSTRs in series. We selected CSTRs to illustrate how *StormReactor* enables wastewater treatment process models. Often CSTRs are modeled assuming steady state conditions, where the influent concentration, inflow rate, and outflow rate are constant, and therefore, the concentration in the control volume is also constant. Steady state condition allows for a closed form solution to the

CSTR equation. However, in a wetland, influent concentration and flows are dynamic and therefore, the CSTR should be assumed to be unsteady. We solved the unsteady CSTR with an ODE solver to show how *StormReactor* integrates with other computational Python packages. We selected the SciPy ODE numerical solver using the explicit runge-kutta method[4] (Virtanen et al., 2020). The CSTR equation is defined as:

$$\frac{dC}{dt}\ V = Q_{in}C_{in} - Q_{out}C - kCV \tag{6}$$

Based on data collected in this network, the influent DO concentration (C_{in}) to the wetland was assumed to be 9.6 mg/L. The reaction rate constant (k_{DO}) was assumed to be 0.2/hr (Reddy and Patrick, 1984). At each routing time step, the dynamic parameters were queried from SWMM (Q_{in}, Q_{out}, V) and the ODE solver computed the current concentration (C). Since the DO concentration was only relevant to triggering denitrification in the wetland, DO was tracked only in Python and therefore, the new DO concentration did not need to be set in SWMM (i.e., DO was not added as a pollutant in the SWMM input file).

Nitrate treatment was triggered when the DO concentration dropped below 1 mg/L, signaling anoxic conditions. Nitrate treatment in the wetland was also modeled in *StormReactor* using three CSTRs in series (Kadlec and Wallace, 2009). The nitrate concentration in the real stormwater network averages less than 1 mg/L during steady state and storm conditions. Although this low level may exceed recommended water quality criteria (EPA, 2002), assuming a larger concentration will result in higher rates of denitrification for simulation purposes. Therefore, for our simulation, nitrate was added to the system using SWMM's wash-off model assuming an EMC of 10 mg/L, which aligns with 13% of stream sites monitored by Mueller and Spahr (2005). The nitrate reaction rate constant (k_{NO}) was assumed to be 1.5/day (Reddy and Patrick, 1984). At each routing time step, the dynamic parameters were queried from SWMM (Q_{in}, Q_{out}, C_{in}, V), the ODE solver computed the current concentration (C), and that concentration was then set in SWMM. To validate the CSTRs in series model, *StormReactor*'s steady state concentration at the end of the simulation was compared with the steady state analytical solution. The wetland's nitrate concentration from *StormReactor* converged to the computed steady state analytical solution (5.7% error).

4.2.2. Nitrate real-time control strategy

A water quality-based controller was constructed to maximize denitrification without flooding the wetland (Algorithm 1). The controller held water in the wetland until the nitrate was treated or flooding was imminent. It also held water in the upstream detention basin until the downstream wetland had sufficient storage capacity to handle more inflow. When the controller opened a valve, it regulated the size of the opening (0–100%) to release water at a rate proportional to the asset's water level by solving the submerged orifice equation (Rossman, 2015). It was assumed that the network had the necessary water quantity and quality sensors and the outlets of the detention basin and the wetland had controllable valves. To reflect real world implementation, control decisions were constrained to every 15 min. The controlled scenario was compared against a baseline, uncontrolled scenario to determine the effectiveness of the controller.

Algorithm 1. The controller's objective was to maximize denitrification without flooding the wetland. The controller computed the valve's percent opening for the detention basin ($valve_{DB}$) and wetland ($valve_W$). Water was released proportionally by solving the submerged orifice equation ($Q_{max} = CA/\sqrt{2gd}$) for C, the discharge coefficient, where Q_{max} was the maximum flow rate desired ($Q_{max} = 2\ m^3/s$), A was the completely open orifice area, g was acceleration due to gravity, and d was water depth. Q_{max} was the flow rate threshold at which downstream sediments were assumed to re-suspend (Mullapudi et al., 2017). The computed value for C was multiplied by a scaling factor f ($f = 1.75$ in this study).

Compute wetland's DO and nitrate concentrations: $\frac{dC}{dt} = \frac{Qin_t \cdot Cin_t - Qout_t \cdot Cout_t}{V_t} - k_t \cdot C_t$
for i in controllable valves **do**
  if $DO^t > 1mg/L$ **then**
    if $d_W \leq 3m$ **then**
      $valve_W = 0$
      $valve_{DB} = f \cdot Q_{max}/A_{DB}\sqrt{2 \cdot g \cdot d_{DB}}$
    else
      $valve_W = f \cdot Q_{max}/A_W\sqrt{2 \cdot g \cdot d_W}$
      $valve_{DB} = 0$
  else if $DO \leq 1mg/L$ **then**
    if $C_t \leq 5mg/L$ **then**
      $valve_i = f \cdot Q_{max}/A_i\sqrt{2 \cdot g \cdot d_i})$
    else
      if $d_W \leq 3m$ **then**
        $valve_i = 0$
      else
        $valve_W = f \cdot Q_{max}/A_W\sqrt{2 \cdot g \cdot d_W})$
        $valve_{DB} = 0$
end

4.2.3. Nitrate results and discussion

The controller met the control objective of maximizing denitrification (Fig. 6). The controlled scenario saw a 95% nitrate load reduction at the outfall as compared to the uncontrolled scenario. The load reduction was a result of keeping the valves closed when either the wetland was oxic or the wetland's nitrate concentration was too high. The controller used both the wetland and the upstream basin for storage until the conditions were appropriate to release flows. To put this load reduction into context, SWMM was used to determine how large the studied wetland would need to be to obtain the same load reduction without real-time control. After incrementally increasing the area of the wetland and rerunning the SWMM simulation several times, it was determined that the wetland would need to be four times as large to obtain the same load reduction.

The controller also ensured that flooding did not occur in any of the assets (Fig. 6). The water depths in the detention basin and wetland were kept below their flooding thresholds. These two assets did not flood because the detention basin had significant storage capacity, and the controller opened the wetland valve whenever it was close to its maximum capacity. In both scenarios, the retention basin depth resulted in some flows bypassing the wetland. Unfortunately, this is because of how the retention basin/wetland system was designed. If a control valve was installed or the bypass height was increased on the retention basin, these bypass flows could have been reduced.

StormReactor provided the ability to implement a water quality-based controller in SWMM. Prior to this package, users trying to meet water quality goals with controllers could only access water quantity states. Now, users can access water quality states and build a pollutant concentration-based controller with only a few lines of Python code.

The real-time controlled nitrate simulation took 86.84 s on a 2018 MacBook Pro (Processor: 2.2 Ghz 6-Core Intel Core i7, Memory: 16 GB

[4] https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.ode.html.

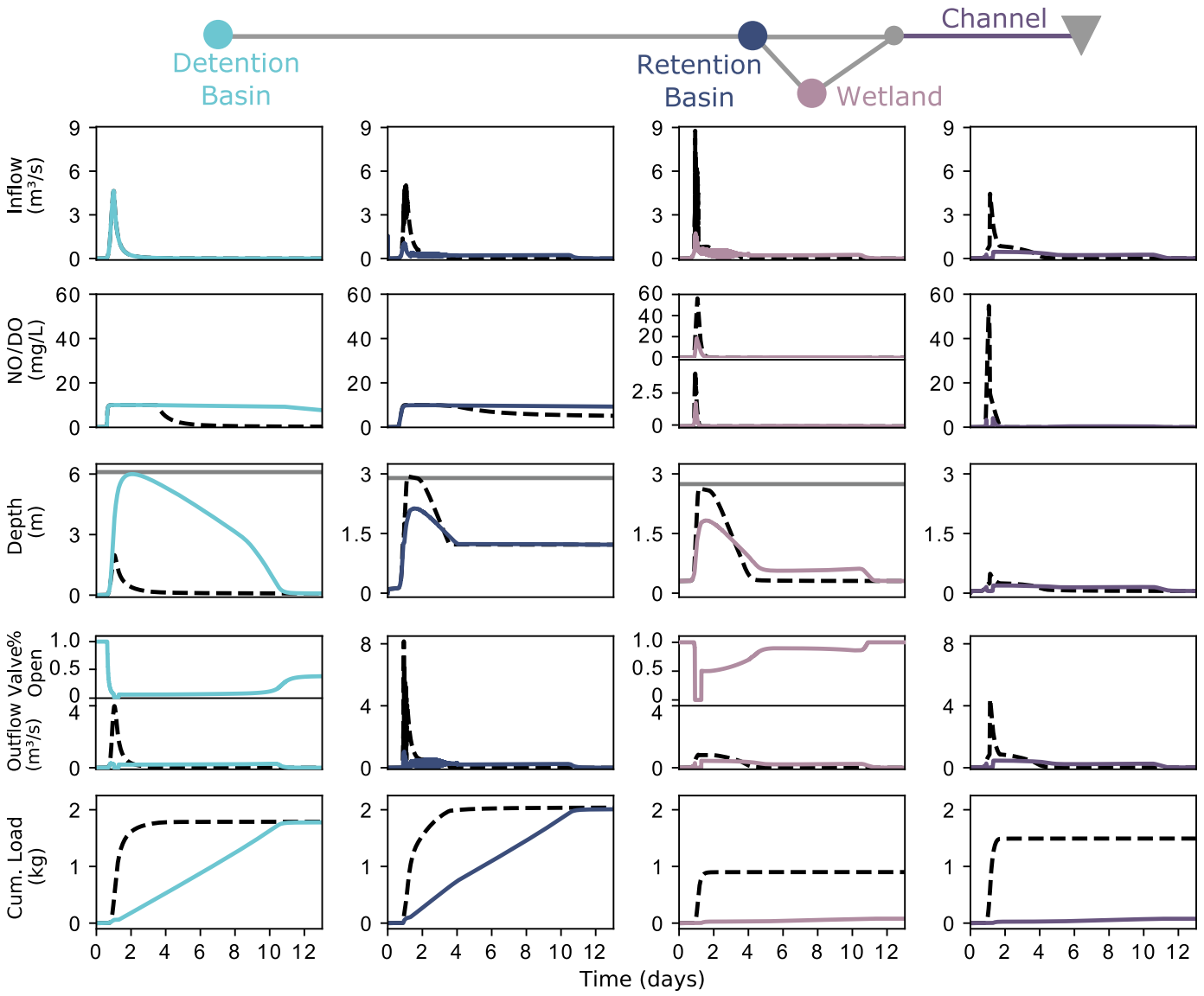


Fig. 6. Comparison of the uncontrolled (dotted lines) and controlled (solid lines) scenarios for the various assets in the stormwater network including inflow rate (top panel), nitrate and DO concentration (second panel), storage depth (third panel), valve position and outflow rate (fourth panel), and cumulative nitrate load (bottom panel). In the depth panel, the gray solid lines depict the flooding thresholds for the detention basin and the wetland and the bypass threshold for the retention basin. No flooding occurred but some flows did bypass the wetland in both the uncontrolled and controlled scenarios.

2400 MHz DDR4). The nitrate simulation without real-time control took 86.22 s, as compared to the simulation without water quality or real-time control which took 12.30 s. The increased computational time was a result of the longer simulation (twelve days instead of five) and the ODE solver. Therefore, to increase computational efficiency in the future, a discrete form update could be used instead of an ODE solver.

5. Discussion

As shown in the case studies, *StormReactor* improved water quality process representation at both the site and watershed scale. Rather than implementing an all-in-one quality-quantity model, we coupled the popular water quantity features of SWMM with *StormReactor*'s water quality model. To illustrate the fidelity of *StormReactor*, we showed how a variety of pollutant transformations (e.g., erosion, settling, CSTR) matched expectations from established models and methods. Therefore, *StormReactor* was shown to be an effective tool for modeling water quality.

To the best of our knowledge, our modular framework supports many of the features seen in advanced hydraulic and water quality packages. For advanced users, *StormReactor*'s integration with Python will support numerical solvers and packages, higher order reaction kinetics, wastewater process models (e.g., ASM-1), and combined sewer networks. In its present implementation, *StormReactor* poses a few constraints which users need to be aware of before choosing to use it in their stormwater studies. It does not presently support LID (i.e., green infrastructure) water quality processes because SWMM handles LID water quality outside of its link and node data structures. In addition, *StormReactor* does not support high spatial resolution water quality processes (e.g., advection, diffusion, dispersion). Both LID access and high spatial resolution models can be added and are proposed as future work. Aside from these limitations, *StormReactor* provides a general water quality modeling solution that is flexible and expandable.

The nitrate case study points to the potential of using real-time control or "smart" stormwater systems for ecological benefits. Watershed water quality goals can be acheived by tuning real-time control. The ability to model complex water quality interactions enables the development and testing of real-time control algorithms that use

pollutant concentration, load, and sensor data. We can now utilize formal control theory (e.g., PID, MPC, genetic algorithms) to explore emergent behavior, stability, and optimal control strategies at both the site and watershed scale. We can then use this information to optimize asset treatment performance, pushing our watersheds to behave like distributed water treatment plants, and ultimately improve watershed water quality.

6. Conclusions

StormReactor improves the fidelity of modeling pollutant transformations and pollutant-based real-time control; moving us a step closer to realizing the goal of controlling entire watersheds as real-time distributed treatment plants. Additional fidelity could be gained by adding LID access and high spatial resolution models to *StormReactor*. The flexibility of *StormReactor* gives researchers and practitioners immense freedom in modeling water quality. We hope that this package will become a community-driven resource. We see opportunities for the research community to collaborate on the development of *StormReactor* by contributing their own pollutant generation and treatment methods. As we scale to larger networks, future work must evaluate the computational efficiency of *StormReactor*. In addition, significant future research stands to be enabled through the use of holistic frameworks, such as those posed in this paper. In particular, future studies have the potential to evaluate how to control entire watersheds in response to ecological objectives.

Software availability

Name of software: *StormReactor* Developers: Brooke Mason, Abhiram Mullapudi Year first available: 2020 Operating system: OSX, Windows, or Linux Software required: Python 3.6.0+, pyswmm 1.0.1+, numpy 1.21.0+, scipy 1.7.0+ Availability and online documentation: https://github.com/kLabUM/StormReactor. A snapshot of the GitHub repository consistent with the description in this paper is available in Zenodo (Mason and Mullapudi, 2021).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was funded by the U.S. National Science Foundation (Award Numbers: 1750744 and 1737432).

References

Bach, P.M., Rauch, W., Mikkelsen, P.S., McCarthy, D.T., Deletic, A., 2014. A critical review of integrated urban water modelling - urban drainage and beyond. Environ. Model. Software 54, 88–107. https://doi.org/10.1016/j.envsoft.2013.12.018.

Baek, S.S., Ligaray, M., Pyo, J., Park, J.P., Kang, J.H., Pachepsky, Y., Ahn Chun, J., Hwa Cho, K., 2020. A novel water quality module of the SWMM model for assessing Low Impact Development (LID) in urban watersheds. J. Hydrol. 586 https://doi.org/10.1016/J.JHYDROL.2020.124886.

Berglund, E.Z., Monroe, J.G., Ahmed, I., Noghabaei, M., Do, J., Pesantez, J.E., Khaksar Fasaee, M.A., Bardaka, E., Han, K., Proestos, G.T., Levis, J., 2020. Smart infrastructure: a vision for the role of the civil engineering profession in smart cities. J. Infrastruct. Syst. 26 https://doi.org/10.1061/(ASCE)IS.1943-555X.0000549.

Bittner, D., Rychlik, A., Klöffel, T., Leuteritz, A., Disse, M., Chiogna, G., 2020. A gis-based model for simulating the hydrological effects of land use changes on karst systems – the integration of the lukars model into freewat. Environ. Model. Software 127. https://doi.org/10.1016/j.envsoft.2020.104682.

Brendel, C.E., Dymond, R.L., Aguilar, M.F., 2020. Integration of quantitative precipitation forecasts with real-time hydrology and hydraulics modeling towards probabilistic forecasting of urban flooding. Environ. Model. Software 134. https://doi.org/10.1016/j.envsoft.2020.104864.

Conley, D.J., Paerl, H.W., Howarth, R.W., Boesch, D.F., Seitzinger, S.P., Havens, K.E., Lancelot, C., Likens, G.E., 2009. Controlling eutrophication: nitrogen and phosphorus. Science 323. https://doi.org/10.1126/science.1167755.

Deletic, A.B., Maksimovic, C.T., 1998. Evaluation of water quality factors in storm runoff from paved areas. J. Environ. Eng. 124, 869–879. https://doi.org/10.1061/(ASCE)0733-9372(1998)124:9(869).

Dong, A., Chesters, G., Simsiman, G.V., 1984. Metal composition of soil, sediments, and urban dust and dirt samples from the Menomonee River Watershed, Wisconsin, U.S.A. Water, Air, and Soil Pollution 22, 257–275. https://doi.org/10.1007/BF00159348.

Eggimann, S., Mutzner, L., Wani, O., Schneider, M.Y., Spuhler, D., Moy De Vitry, M., Beutler, P., Maurer, M., 2017. The Potential of Knowing More: A Review of Data-Driven Urban Water Management. Environ. Sci. Technol. 51, 2538–2553. https://doi.org/10.1021/acs.est.6b04267.

Egodawatta, P., Thomas, E., Goonetilleke, A., 2007. Mathematical interpretation of pollutant wash-off from urban road surfaces using simulated rainfall. Water Res. 41, 3025–3031. https://doi.org/10.1016/j.watres.2007.03.037.

Engelund, F., Hansen, E., 1967. A Monograph on Sediment Transport in Alluvial Streams. 2. Teknisk Forlag, Copenhagen.

EPA, 2002. Summary Table for the Nutrient Criteria Documents. Technical Report. United States Environmental Protection Agency.

García, L., Barreiro-Gomez, J., Escobar, E., Téllez, D., Quijano, N., Ocampo-Martinez, C., 2015. Modeling and real-time control of urban drainage systems: A review. Adv. Water Resour. 85, 120–132. https://doi.org/10.1016/J.ADVWATRES.2015.08.007.

Garofalo, G., Giordano, A., Piro, P., Spezzano, G., Vinci, A., 2017. A distributed real-time approach for mitigating CSO and flooding in urban drainage systems. J. Netw. Comput. Appl. 78, 30–42. https://doi.org/10.1016/j.jnca.2016.11.004.

Giraldi, D., de Michieli Vitturi, M., Iannelli, R., 2010. FITOVERT: A dynamic numerical model of subsurface vertical flow constructed wetlands. Environ. Model. Software 25, 633–640. https://doi.org/10.1016/j.envsoft.2009.05.007.

Howarth, R., Paerl, H.W., 2008. Coastal marine eutrophication: Control of both nitrogen and phosphorus is necessary. In: Proceedings of the National Academy of Sciences of the United States of America, vol. 105. https://doi.org/10.1073/pnas.0807266106.

Kadlec, R.H., 2010. Nitrate dynamics in event-driven wetlands. Ecol. Eng. 36, 503–516. https://doi.org/10.1016/j.ecoleng.2009.11.020.

Kadlec, R.H., Knight, R.L., 1996. Treatment Wetlands. Lewis Publishers, Boca Raton, FL.

Kadlec, R.H., Wallace, S.D., 2009. Treatment Wetlands, 2 ed. CRC Press. https://doi.org/10.2166/9781780408774.

Kerkez, B., Gruden, C., Lewis, M., Montestruque, L., Quigley, M., Wong, B.P., Bedig, A., Kertesz, R., Braun, T., Cadwalader, O., Poresky, A., Pak, C., 2016. Smarter Stormwater Systems. Environ. Sci. Technol. 50, 7267–7273. https://doi.org/10.1021/acs.est.5b05870.

Laniak, G.F., Olchin, G., Goodall, J., Voinov, A., Hill, M., Glynn, P., Whelan, G., Geller, G., Quinn, N., Blind, M., Peckham, S., Reaney, S., Gaber, N., Kennedy, R., Hughes, A., 2013. Integrated environmental modeling: A vision and roadmap for the future. Environ. Model. Software 39, 3–23. https://doi.org/10.1016/j.envsoft.2012.09.006.

Mason, B., 2021a. bemason/StormReactor-CaseStudy-Nitrate: Release for StormReactor Research Article (V1.0). Zenodo. https://doi.org/10.5281/zenodo.4913515.

Mason, B., 2021b. bemason/StormReactor-CaseStudy-TSS: Release for StormReactor Research Article (v1.0). Zenodo. https://doi.org/10.5281/zenodo.4913501.

Mason, B., Mullapudi, A., 2021. kLabUM/StormReactor: Version for StormReactor Research Article (v1.0). Zenodo. https://doi.org/10.5281/zenodo.4913493.

McCarthy, D.T., Mitchell, V.G., Deletic, A., Diaper, C., 2007. Escherichia coli in urban stormwater: Explaining their variability. Water Sci. Technol. 56, 27–34. https://doi.org/10.2166/wst.2007.752.

Mcdonnell, B.E., Ratliff, K., Tryby, M.E., Jia, J., Wu, X., Mullapudi, A., 2020. PySWMM: The Python Interface to Stormwater Management Model (SWMM). J. Open Source Software 5, 2292. https://doi.org/10.21105/joss.02292.

Mueller, D.K., Spahr, N.E., 2005. Water-quality, streamflow, and ancillary data for nutrients in streams and rivers across the nation, 1992-2001. Technical Report. United States Geological Survey. https://doi.org/10.3133/ds152.

Mullapudi, A., Wong, B.P., Kerkez, B., 2017. Emerging investigators series: building a theory for smart stormwater systems. Water Research & Technology 3, 66–77. https://doi.org/10.1039/C6EW00211K.

Niazi, M., Nietch, C., Maghrebi, M., Asce, A.M., Jackson, N., Bennett, B.R., Tryby, M., Massoudieh, A., Asce, M., 2017. Storm water management model: Performance review and gap analysis. J. Sustainable Water Built Environ. 3 https://doi.org/10.1061/JSWBAY.0000817.

Obropta, C.C., Kardos, J.S., 2007. Review of Urban Stormwater Quality Models: Deterministic, Stochastic, and Hybrid Approaches. JAWRA.J. Am. Water Resour. Assoc. 43, 1508–1523. https://doi.org/10.1111/j.1752-1688.2007.00124.x.

Overton, D.E., Meadows, M.E., 2013. Stormwater Modeling. Elsevier Science.

Pálfy, T.G., Langergraber, G., 2014. The verification of the Constructed Wetland Model No. 1 implementation in HYDRUS using column experiment data. Ecol. Eng. 68, 105–115. https://doi.org/10.1016/j.ecoleng.2014.03.016.

Perryman, S.E., Rees, G.N., Walsh, C.J., Grace, M.R., 2011. Urban Stormwater Runoff Drives Denitrifying Community Composition Through Changes in Sediment Texture and Carbon Content. Microb. Ecol. 61, 932–940. https://doi.org/10.1007/s00248-011-9833-8.

Persaud, P.P., Akin, A.A., Kerkez, B., Mccarthy, D.T., Hathaway, J.M., 2019. Real time control schemes for improving water quality from bioretention cells. Blue Green Systems 1, 55–71. https://doi.org/10.2166/bgs.2019.924.

Reddy, K.R., Patrick, W.H., 1984. Nitrogen Transformations and Loss in Flooded Soils and Sediments. Crit. Rev. Environ. Contr. 13, 273–309. https://doi.org/10.1080/10643388409381709.

Rizzo, A., Langergraber, G., Galvão, A., Boano, F., Revelli, R., Ridolfi, L., 2014. Modelling the response of laboratory horizontal flow constructed wetlands to unsteady organic loads with HYDRUS-CWM1. Ecol. Eng. 68, 209–213. https://doi.org/10.1016/j.ecoleng.2014.03.073.

Rossman, L.A., 2015. Storm Water Management Model User's Manual Version 5 . 1. U.S. EPA, Cincinnati.

Rossman, L.A., Huber, W.C., 2016. Storm Water Management Model Reference Manual Volume III-Water Quality. United States Environmental Protection Agency, Cincinnati.

Schilling, K.E., Kim, S.W., Jones, C.S., 2017. Use of water quality surrogates to estimate total phosphorus concentrations in Iowa rivers. J. Hydrol.: Reg. Stud. 12, 111–121. https://doi.org/10.1016/j.ejrh.2017.04.006.

Scholes, L., Revitt, D.M., Ellis, J.B., 2008. A systematic approach for the comparative assessment of stormwater pollutant removal potentials. J. Environ. Manag. 88, 467–478. https://doi.org/10.1016/j.jenvman.2007.03.003.

Schu, Tze, M., Campisano, A., Colas, H., Schilling, W., Vanrolleghem, P.A., 2004. Real time control of urban wastewater systems - Where do we stand today? J. Hydrol. 299, 335–348. https://doi.org/10.1016/j.jhydrol.2004.08.010.

Shammaa, Y., Zhu, D.Z., 2001. Techniques for Controlling Total Suspended Solids in Stormwater Runoff. Can. Water Resour. J. 26, 359–375. https://doi.org/10.4296/cwrj2603359.

Shishegar, S., Duchesne, S., Pelletier, G., 2019. An Integrated Optimization and Rule-based Approach for Predictive Real Time Control of Urban Stormwater Management Systems. J. Hydrol. 577, 124000. https://doi.org/10.1016/j.jhydrol.2019.124000.

Shuler, C.K., Mariner, K.E., 2020. Collaborative groundwater modeling: Open-source, cloud-based, applied science at a small-island water utility scale. Environ. Model. Software 127. https://doi.org/10.1016/j.envsoft.2020.104693.

Suarez, V.V.C., Brederveld, R.J., Fennema, M., Moreno-Rodenas, A., Langeveld, J., Korving, H., Schellart, A.N., Shucksmith, J.D., 2019. Evaluation of a coupled hydrodynamic-closed ecological cycle approach for modelling dissolved oxygen in surface waters. Environ. Model. Software 119, 242–257. https://doi.org/10.1016/j.envsoft.2019.06.003.

Sun, C., Joseph-Duran, B., Maruejouls, T., Cembrano, G., Meseguer, J., Puig, V., Litrico, X., 2017. Real-Time Control-Oriented Quality Modelling in Combined Urban Drainage Networks. In: IFAC-PapersOnLine, pp. 3941–3946. https://doi.org/10.1016/j.ifacol.2017.08.142.

Sun, C., Svensen, J.L., Borup, M., Puig, V., Cembrano, G., Vezzaro, L., 2020. An MPC-Enabled SWMM Implementation of the Astlingen RTC Benchmarking Network. Water 12, 1–13. https://doi.org/10.3390/w12041034.

Sutherland, J., Townend, I.H., Harpham, Q.K., Pearce, G.R., 2017. From integration to fusion: The challenges ahead. Geol. Soc. Spec. Publ. 408, 35–54. https://doi.org/10.1144/SP408.6.

Talbot, M.T., Mcguire, O., Olivier, C., Fleming, R., 2016. Parameterization and Application of Agricultural Best Management Practices in a Rural Ontario Watershed Using PCSWMM. Journal of Water Management Modeling C400. https://doi.org/10.14796/JWMM.C400.

Troitsky, B., Zhu, D.Z., Loewen, M., van Duin, B., Mahmood, K., 2019. Nutrient processes and modeling in urban stormwater ponds and constructed wetlands. Can. Water Resour. J. 44, 230–247. https://doi.org/10.1080/07011784.2019.1594390.

Tuomela, C., Sillanpää, N., Koivusalo, H., 2018. Assessment of stormwater pollutant loads and source area contributions with storm water management model (SWMM). J. Environ. Manag. 233, 719–727. https://doi.org/10.1016/j.jenvman.2018.12.061.

USDA, 1983. Transmission of Sediment by Water (chapter 4). In: National Engineering Handbook, pp. 1–39.

Vanrolleghem, P.A., Benedetti, L., Meirlaen, J., 2005. Modelling and real-time control of the integrated urban wastewater system. Environ. Model. Software 20, 425–442. https://doi.org/10.1016/j.envsoft.2004.02.004.

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, d., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., Contributors, S., 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat. Methods 1–12. https://doi.org/10.1038/s41592-019-0686-2.

Wang, Q., Zou, R., Khalid, A., Yang, T., 2019. Uncertainty-based parameter estimation for urban pollutant buildup and washoff simulation using a multiple pattern inverse modeling approach. Math. Comput. Simulat. https://doi.org/10.1016/J.MATCOM.2019.07.009.

White, J.R., Reddy, K., 2009. Biogeochemical Dynamics I: Nitrogen Cycling in Wetlands. In: Maltby, E., Barker, T. (Eds.), The Wetlands Handbook. Blackwell Publishing Ltd, pp. 213–227 (chapter 9).

Wong, B.P., Kerkez, B., 2018. Real-Time Control of Urban Headwater Catchments Through Linear Feedback: Performance, Analysis, and Site Selection. Water Resour. Res. 54, 7309–7330. https://doi.org/10.1029/2018WR022657.

Wong, T.H.F., Fletcher, T.D., Duncan, H.P., Jenkins, G.A., 2006. Modelling urban stormwater treatment-A unified approach. Ecol. Eng. 27, 58–70. https://doi.org/10.1016/j.ecoleng.2005.10.014.

Wu, B., Asce, M., Molinas, A., Julien, P.Y., 2004. Bed-Material Load Computations for Nonuniform Sediments. J. Hydrol. Eng. 130 https://doi.org/10.1061/(ASCE)0733-9429(2004)130:10(1002).

Yang, Y.Y., Lusk, M.G., 2018. Nutrients in Urban Stormwater Runoff: Current State of the Science and Potential Mitigation Options. Current Pollution Reports 4, 112–127. https://doi.org/10.1007/s40726-018-0087-7 (publisher: Springer International Publishing).

Zhang, P., Cai, Y., Wang, J., 2018. A simulation-based real-time control system for reducing urban runoff pollution through a stormwater storage tank. J. Clean. Prod. 183, 641–652. https://doi.org/10.1016/j.jclepro.2018.02.130.