ABHIRAM MULLAPUDI

STATISTICAL LEARNING APPROACHES FOR THE CONTROL OF STORMWATER SYSTEMS

STATISTICAL LEARNING APPROACHES FOR THE CONTROL OF STORMWATER SYSTEMS

ABHIRAM MULLAPUDI

Doctor of Philosophy
Department of Civil and Environmental Engineering
University of Michigan

June 2020 – 0.1

ABSTRACT

Our existing stormwater systems are unable to keep pace with the evolving weather events and rapid urbanization. Smart Stormwater systems are an effective solution for operating such systems so they can tackle the storms events. In this dissertation, scientific knowledge and tools for developing control algorithms for stormwater systems. Also tools being developed for making this accisible to the wider research community.

PUBLICATIONS

Publications from this thesis.

[1] **Mullapudi, Abhiram**, Brandon P. Wong, and Branko Kerkez. "Emerging investigators series: building a theory for smart stormwater systems." In: *Environmental Science: Water Research & Technology* 3.1 (2017), pp. 66–77. ISSN: 2053-1419. DOI: 10.1039/c6ew00211k. URL: http://dx.doi.org/10.1039/C6EW00211K.

[2] **Mullapudi, Abhiram**, Matthew D. Bartos, Brandon P. Wong, and Branko Kerkez. "Shaping Streamflow Using a Real-Time Stormwater Control Network." In: *Sensors* 18.7 (2018), p. 2259. ISSN: 1424-8220. DOI: 10.3390/s18072259. URL: http://dx.doi.org/10.3390/s18072259.

[3] **Mullapudi, Abhiram**, Matthew J. Lewis, Cyndee L. Gruden, and Branko Kerkez. "Deep reinforcement learning for the real time control of stormwater systems." In: *Advances in Water Resources* (Apr. 2020), p. 103600. ISSN: 03091708. DOI: 10.1016/j.advwatres.2020.103600.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

1 INTRODUCTION

As the cities grow larger and they alter the landscape of the watersheds. As the watershed shape changes, it alters the how the water flows on the watershed. This altered water flow can be dangerous and can be harmful to the people in the system. Hence, we need to infrastructure in place to move this water away from the urban environment. This is where, stormwater infrastructure comes in. This infrastructure moves water accumulated in cities away from them into downstream water bodies.

Stormwater systems though designed to handle the runoff, are not able to handle the rising demands in the urban environments. Redesigning and retrofitting these systems to handle the rising demands, such an approach might not work always. Given the dynamic nature of the systems, such a static solution might not always be the best way to tackle these challenges. Furthermore, given the these systems have to achieve multiple objectives, having them as static systems might not be the best solutions. Also these systems are designed as localized systems, with the intent that localized solutions will eventually scale up and improve the performance of the system as a whole. But Maryland et al have demonstrated that might not always be the case. Hence, we need a more system scale approach that takes into account all the moving parts and treats the entire network as a single entity.

These rising demands can be addressed by rebuilding the infrastructure with larger capacities to handle the increasing inflows. Though adding capacity to the existing system, can help us handle the incoming flows, it is an expensive process and it is not sure that it will work. Alternatively, by retrofitting the existing system with sensors and controllers, we can monitor the state of the stormwater network in real-time and control its response to achieve system scale objectives. This enables us to dynamically control the stormwater network to fully utilize the existing potential in the stormwater network to handle the incoming stormevents. Dynamics of stormwater networks are inherently complex and given the scale and the impact they can have on general public, we need good algorithms for controlling them. Over the past decade, there has been a lot of interest on developing algorithms for the control of stormwater systems.

1.1 CONTROL OF STORMWATER SYSTEMS

The state-of-the-art in stormwater control can be broadly classified under two categories, based on how they identify control actions:

- Control algorithms reliant on parametrized models (e.g. Model Predictive Control) for identifying control actions.
- Search based algorithms (e.g. evolutionary approaches like Genetic Algorithms) that exhaustively simulate models for identifying control actions.
- Heuristic based approaches that identify the control actions solely based on the state of the system. (e.g. Fuzzy logic controllers).

Though these control algorithms have been applied for localized control in stormwater systems[1], their investigation in the context of coordinated control has been limited. To fully realize the potential of the stormwater infrastructure and to safeguard our water bodies, we need to synthesize control algorithms that are able to coordinate the response of many distributed control assets in the network, while simultaneously achieving a diverse set of water quality and flow objectives. Technologically, we are at a point where we can monitor and control these assets in real-time, but the development of control algorithms is hampered by a number of fundamental knowledge gaps.

MPC in stormwater control literature is broadly. In this dissertation, MPC is the explicit use of process based dynamical models for control.

Our existing stormwater infrastructure systems are unable to keep pace with rapidly evolving storm events and changing landscapes. These infrastructure systems — designed for an "average" event — are still proving to be inefficient in tackling dynamic weather conditions and achieving diverse urban sustainability objectives[2]. While existing stormwater systems could be rebuilt to reduce flooding and improve water quality, such an undertaking is often not financially viable, nor guaranteed to work. In lieu of new construction, one alternative would be to retrofit existing stormwater systems with sensors and controllers, so that these systems can be dynamically controlled in real-time to achieve the desired objectives. The goal of my dissertation is to make fundamental discoveries that will inform the control of smart stormwater systems, specifically focusing on statistical learning approaches that can be used to generate safe and reliable control algorithms.

1 e.g. maintaining constant water levels and flows in individual basins.
2 e.g. improving water quality and minimizing erosion

1.2 STATISTICAL METHODS

Knowledge Gaps

1. We do not know how to design control algorithms that can target pollutants in stormwater runoff, nor do we have the simulation tools necessary for such studies.
2. We do not know to how to characterize the controllability of an urban watershed, especially in the context of water quality.
3. We do yet know how to synthesize control algorithms for distributed storm-water assets without making explicit dynamical assumptions (e.g. linearity).
4. We do not know how to quantify the uncertainty of algorithms used in the real-time control of stormwater systems.
5. We do not know how to explicitly incorporate and account for hydraulic travel time within a real-time controlled system.
6. We do not have open platforms for the systematic evaluation and comparison of different control algorithms.

1.3 DISSERTATION OUTLINE

My dissertation addresses these knowledge gaps, leveraging statistical approaches, to develop tools and algorithms for enabling control of stormwater systems. The first chapter of this dissertation focuses on the development of a theoretical framework and the necessary tools for simulating control in stormwater systems. The second chapter demonstrates how a real-world wireless sensor network can be used for shaping the flow response of an entire urban watershed. In the next three chapters, various control algorithms are proposed, and their performance is evaluated across diverse scenarios to quantify the strengths and limitations. In the final chapter, I introduce a python-based simulation sandbox, which is being developed specifically for the systematic evaluation and comparison of stormwater control algorithms.

2 BUILDING A THEORY FOR SMART STORMWATER SYSTEMS

Rapid advances in sensing, computation, and wireless communications are promising to merge the physical with the virtual. Calls to build the "smart" city of the future are being embraced by decision makers. While the onset of self-driving cars provides a good example that this vision is becoming a reality, the role of information technology in the water sector has yet to be fleshed out. These technologies stand to enable a leap in innovation in the distributed treatment of urban runoff, one of ourlargest environmental challenges.

Retrofitting stormwater systems with sensors and controllers will allow the city to be controlled in real-time as a distributed treatment plant. Unlike static infrastructure, which cannot adapt its operation to individual storms or changing land uses, "smart" stormwater systems will use system-level coordination to reduce flooding and maximize watershed pollutant removal. Given the sheer number of storm water control measures in United States, even a small improvement to their performance could lead to a substantial reduction in pollutant loads. Intriguingly, such a vision is not limited by technology, which has matured to the point at which it can be ubiquitously deployed. Rather, the challenge is much more fundamental and rooted in a system-level understanding of environmental science. Once stormwater systems become highly instrumented and controlled, how should they actually be operated to achieve desired watershed outcomes? The answer to this question demands the development of a theoretical framework for smart stormwater systems. In this paper we lay out the requirements for such a theory. Acknowledging that the broad adoption these systems may still be years away, we also present and evaluate a modeling framework to allow for the simulation of smart stormwater systems before they become common place. Recent urban floods [38], many of which are driven by flashy events and inadequately sized infrastructure, are all too common example that aging stormwater infrastructure is struggling to keep pace with a dynamic and changing climate. While flood control often emerges as one of the most promising application areas, to illustrate the flexibility of smart stormwater systems this paper will focus on the impacts to urban water quality.

2.1 DO BEST LOCAL PRACTICES ACHIEVE THE BEST GLOBAL OUTCOMES?

Pollutants in runoff are threatening the health of downstream ecosystems, as evinced by harmful algal blooms, such as those on Lake Erie [80] and the Chesapeake Bay [12, 105]. Simultaneously, "dry" regions of the country are struggling to find new and clean sources of water. By some estimates, the capture of stormwater in Los Angeles [45] and San Francisco [44] could offset the water used by these cities. This, however, requires at least some level of treatment to ensure that captured stormwater is safe for direct use or aquifer injection. In the face of these challenges, novel solutions for stormwater management are needed.

Reductions in hydraulic or pollutant loads are commonly achieved via a set of distributed stormwater solutions [17, 51], such as ponds or treatment wetlands. Our body of knowledge on the treatment potential of these systems is extensive, showing that significant water quality and hydraulic benefits can be achieved at the level of individual sites [20, 132]. Most recently, an exciting and growing research area has formed around smaller-scale and more distributed Green Infrastructure (GI) solutions, such as green roofs or bioswales[9]. Most of these solutions are grouped under the broader umbrella of Best Management Practices[141] (BMPs) or Storm Water Control Measures (SCMs)[27].

Given the aggressive adoption of these stormwater practices, rarely is the question asked: Does doing the "best" at a local scale translate to doing the best at the watershed scale? Research on this question is limited[102, 103, 117], but paints a cautionary picture. Unless designed as part of a coordinated, city-scale solution, a system of SCMs may actually worsen watershed-scale outcomes. For example, unless coordinated at design-time, hydrographs from individual SCMs may add up to cause larger downstream flows compared to the same watershed without these SCMs[34]. This, in turn, can lead to increased stream erosion and re-suspension of sediment-bound pollutants. More examples can be given, but there is an urgent need to investigate the scalability of SCMs and to ensure that their functionality is tuned in the context of the broader stormwater system.

Even if system-level optimization is used to determine the placement of SCMs[26, 156], it is difficult to guarantee that the overall system will perform as designed. The sheer variability in rainfall[22], seasonal pollutant loadings[98], and broader land use changes[49] will always push stormwater systems beyond their intended design or the "average" storm[35]. As such, it becomes imperative to find a way to adapt to these uncertain disturbances. One solution relies on real-time sensing and control. By equipping stormwater elements with control valves,

which can be operated in real-time based on sensor readings, the overall performance of the entire system can be adapted to achieve watershed-scale benefits (Figure.2.1a).

2.1.1 Existing studies on real-time control

The bulk of existing literature on real-time control of stormwater SCMs focuses on water quality and hydraulic impacts at individual sites, particularly ponds and basins. These studies assume that the outlet of a BMP has been retrofitted with a remotely controllable gate or valve. By strategically controlling outflows before or during storm events, internal volumes can be modified and hydraulic retention time (HRT) can be increased. Jacopin et al.[57] demonstrated that detention basins, designed for flood control, can reduce sediment-based pollutant loading (57% decrease) in downstream water bodies by simply opening and closing a valve. Middleton et al.[81] analyzed the water quality response of a controlled detention basin, observing up to a 90% improvement in TSS and ammonia-nitrate removal. Recent studies[40, 41, 90] in Quebec, Canada, proposed a rule-based control logic for a pond, based on rainfall forecasts, to maximize retention time and reduce hydraulic shocks to the downstream water bodies. The authors observed a 90% improvement in TSS retention. A comprehensive review of these and other studies is summarized in Kerkez et al.[60], along with additional information on how these solutions are deployed in the field. While these studies demonstrate significant potential to improve water quality at the scale of individual sites, the mechanisms behind the removal of pollutants in controlled SCMs remain a research challenge. This is particularly true in the removal of dissolved pollutants, such as ammonia and nitrate. Furthermore, the scalability of real-time control must be evaluated to ensure that local benefits do not overshadow watershed-scale benefits.

Since the 2000 European Union's Water Framework Directive [100] there has been an increasing emphasis on integrated, system-level control of sewer water distribution systems. The resulting control strategies vary in complexity[8, 37, 126] and have since been implemented in a number of urban water networks[85]. Applying these methods to distributed stormwater solutions introduces a new set of challenges, however. Unlike in well-maintained sewer networks, the exposed and distributed nature of stormwater systems introduces complexities associated with the urban hydrologic cycle, such as infiltration, evaporation, soil moisture and groundwater dynamics. Furthermore, one major function of stormwater systems relates to the distributed control of a large variety of solid, dissolved and emerging pollutants. Control of sewer networks is often targeted at volume control to mitigate sewer overflows or overloading treatment plants. As such, much work remains to be conducted on investigating how these methods can be applied to the distributed control of SCMs.

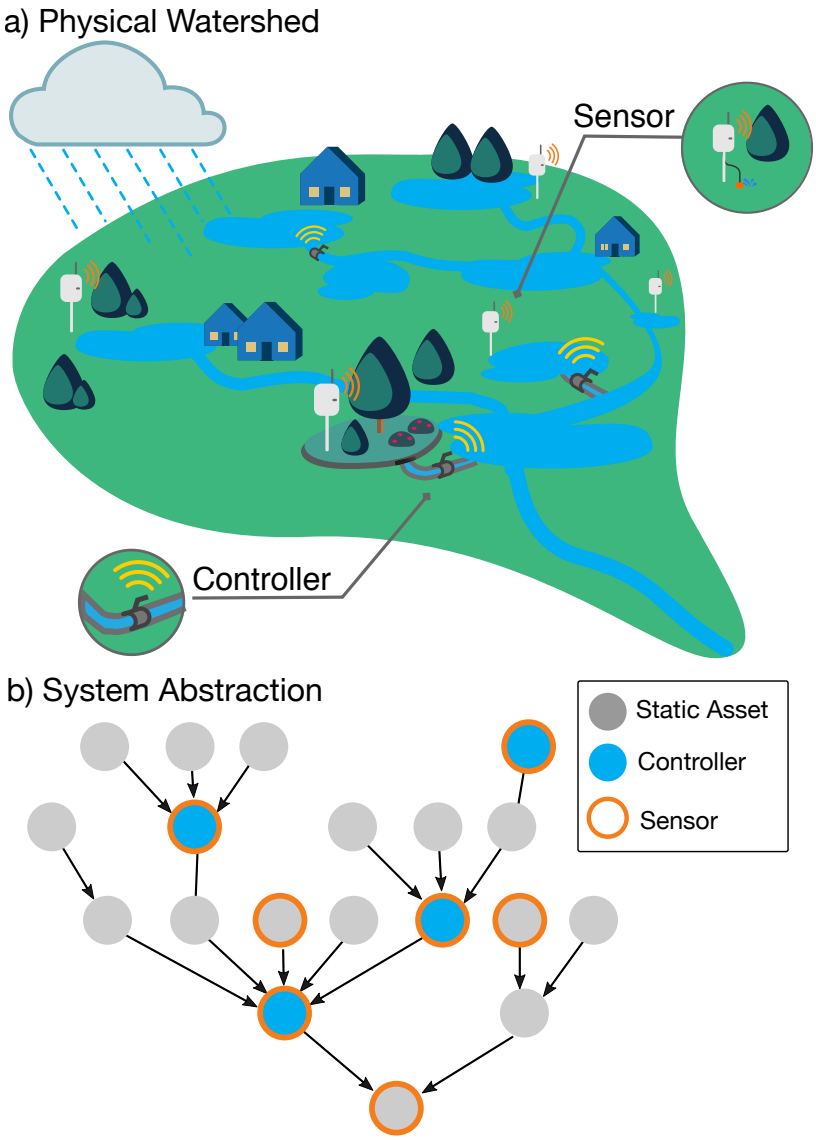

Figure 2.1: Application of control and optimization methods to the real-time operation of stormwater systems will be made possible by abstracting physical models to system-theoretic representations.

2.2 TOWARD A FRAMEWORK FOR SMART STORMWATER SYSTEMS

Many methods have been developed by the operational research and control theory communities to optimize the operation of networked systems[4, 127]. Given their inherent non-linearity and complexity, existing stormwater models are not compatible with these tools. To that end, our knowledge of treatment processes and the physical nature of stormwater systems must first be embedded in a system-theoretic framework (Figure.2.1b). Such a formal and mathematical approach will be crucial toward developing a system-level understanding of stormwater. Not only will this framework help to control future stormwater systems, but it will also create a foundation upon which to answer critical questions, such as: How many controllers are needed and where should they be placed to achieve best system-level benefits? Consequently, how many sensors are needed and where should they be placed to help the control system achieve these objectives?

Until sensors and controllers become ubiquitously deployed across stormwater systems, which may take years to accomplish, there is enough domain knowledge embedded in existing models to begin answering these questions through simulation.

2.2.1 Limitations of existing simulations approaches

Existing stormwater models can be broadly grouped into two categories: those that focus on hydrology (including hydraulics) and those that focus on water quality. The former range across simple routines, such as Muskingum routing[15] and the Rational Method[23], to more complex hydrodynamic models that solve the St.Venant's equation, such as popular packages like SWMM[115] and HEC-RAS[16]. The latter, which include models such as HYDRUS[99, 109] and FITOVERT[46], are used to simulate treatment processes within individual sites, such as wetlands and green infrastructure. The coupling of the two approaches often yields While some packages support extended features that model both hydrology and storm water quality, much work needs to be conducted to improve their accuracy. This often forces a trade-off between comprehensively modeling system-level hydrology or local-level treatment.

Pollutant removal in stormwater is a highly complex and dynamic process. The rate at which pollutants undergo transformation is dependent upon the pollutant-type and its interaction with a given stormwater element (oxygen concentrations, soil types, biomass, settling times, water temperature, etc). Given the complexity of these interactions, popular

hydraulic models, such as SWMM, MUSIC[152] and SUSTAIN[66] often approximate pollutant treatment using first order decay models[59]:

$$\frac{dC}{dt} = -kC \tag{2.1}$$

where the concentration C of a pollutant is assumed to decrease exponentially following a decay coefficient k. While this may be sufficient for approximating the settling dynamics of sediment-bound pollutants, it does not capture the nuanced and complex transformation of dissolved compounds. This often leads to treating the hydraulic retention time (HRT) as the main proxy for water quality.

To that end, a number of approaches have been developed to extend first order decay models to account for variations in background concentration[128], temperature[59], loading rates[83] or mixing conditions[101, 153]. A number of process models have also been developed, applying knowledge from treatment plant operations to stormwater[68]. Langergraber et al.[67, 93] used finite element analysis to model pollutant transformations in subsurface flow wetlands. While these more comprehensive water quality models are highly promising, their ability to simulate system-level treatment remains to be explored.

Given the need to develop a better understanding of the system-level transport and treatment of stormwater, there is a need to couple existing hydraulic and water quality models.

2.3 SIMULATING CONTROLLED SYSTEMS

The real-time operation of gates and valves introduces dynamics that impact hydraulics and water quality. To that end, the biggest limitation of existing models is their ability to simulate the system-wide impacts of real-time control. This includes the ability to dynamically route flows based on a variety of desired control actions, as well as the capacity to simulate a variety of pollutant buildup, washoff, and non-steady state treatment dynamics. While models such as SWMM do have some rudimentary control capabilities, the built-in control logic is limited to site-level control (e.g. maintaining levels or flow in a pond)[115]. Advanced features, such as system-level control, optimization, or the ability to control around external factors (such as weather forecasts) are not yet implemented[107].

While it would be possible to extend an existing model to capture all this functionality, the effort would be significant. To that end, we contend that a coupled modeling approach[47] will be the most flexible way to accomplish this. By coupling models, rather than translating their features in into one large model, it becomes possible to construct a modeling chain whose complexity varies based on the scientific or management question that needs to be answered. More importantly, if

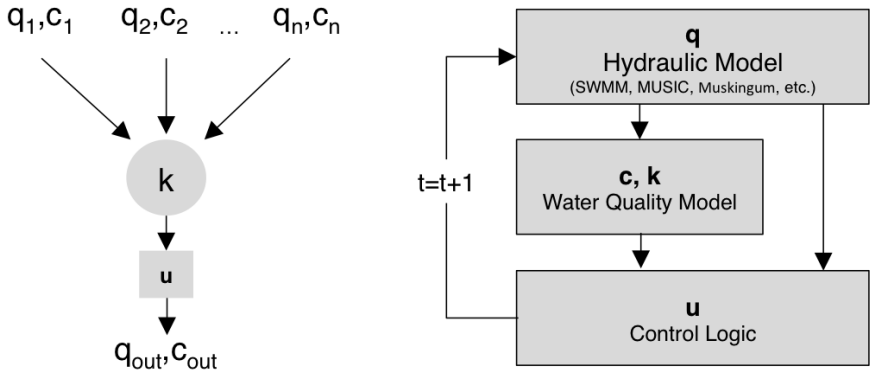

Figure 2.2: Each element in the broader stormwater system can be modeled in a step-wise fashion that simulates hydraulic, water quality and control dynamics.

individual models undergo updates by their respective domain experts, these new features would become available to the coupled model as well without much implementation overhead.

In our coupled modeling approach (Figure2.2), each element in the broader stormwater system can be represented as a storage node, which receives inflows $q_1, q_2, \dots, q_n$ from upstream nodes, each of which has a corresponding concentration $c_1, c_2, \dots, c_n$ for a pollutant of interest. The node has an outflow q_{out} which, unlike in static hydraulic infrastructure, is governed by a real-time control action u. A treatment potential k governs the removal or transformation of the pollutant based on a number of hydraulic and water quality states.

Given that control actions change the hydraulic behavior, which in turn affects the treatment of the pollutants, it becomes necessary to implement a modeling cycle that couples these processes in an interconnected, step-wise fashion. In our implementation, the hydraulic simulation can be carried out by any number of hydraulic models, ranging from simple hydraulic routing schemes, to more complex models such as SWMM or MUSIC. Outputs from the hydraulic model are fed to the water quality model, which, depending on the pollutant of interest, can range from simple first-order process-based methods to more complex finite-element models. Finally, the control module processes the outputs from the hydraulic and water quality models. Based on the objective, which can depend on the states of multiples elements in the overall systems, it sets the discharge rate q_{out} by closing or opening the outlet. The benefits of the coupled approach relate to its flexibility since individual elements can be connected together to represent highly complex stormwater networks.

2.4 SIMULATED STUDIES

To illustrate the potential benefits that can be achieved through real-time stormwater control, we applied the proposed simulation framework to two simulated case studies, which were inspired by our current research efforts in the Midwestern United States. Multiple sites are currently being retrofitted for control and will be compared to these simulations in the coming years. The analysis was targeted on nitrate removal since most of the existing literature focuses on hydraulic control or sediment-bound pollutants.

1. Local scale: The first study investigated the impacts of real-time control to nitrate removal in a single stormwater pond.
2. System scale: The second study evaluated how nitrate removal can be coordinated between a system of controlled stormwater elements.

2.4.1 Model Implementation

Given the scope of the use cases, a simple flow balance module was sufficient to simulate the hydraulic behavior of each element. The change in water volume was modeled as the difference between inflows and outflows, which could be used to calculate the water height h in each element based on its area A. Outflows from each element were proportional to the instantaneous pressure head, unless the element was controlled, in which case it was assumed that outflow can be set such that:

$$0 \leq q_{out} \leq \sqrt{2gh} \tag{2.2}$$

Inflow into upstream elements was based on a hydrograph that was directly measured at one of our study sites in Ann Arbor, Michigan (Figure.2.4). Overflows were simulated in the case that the storage volume was exceeded. For simplicity, infiltration was assumed to be negligible in the study sites.

A water quality model was developed to simulate nitrogen removal in each stormwater element. While nitrogen removal processes are complex, we can simplify their function for this example by assuming that the removal of nitrogen in stormwater ponds and wetlands occurs through two primary pathways: nitrification (conversion of ammonia to nitrate) and de-nitrification (conversion of nitrate to nitrogen gas)[59, 106]. Nitrification is an aerobic process (oxygen acts an electron acceptor), while denitrification is anoxic (nitrate as electron acceptor). While denitrification requires sufficient biomass, it is often not limited by this requirement since plants, grass and other sources of carbon are readily

present in stormwater ponds and wetlands[147]. As such, oxygen availability becomes a critical factor in nitrogen removal. This can readily be tuned through hydraulic control since retention can be used create anaerobic conditions.

We constrained our case studies by focusing only on denitrification, assuming that the majority of nitrogen entering our system was in the form of nitrate. While ammonia is present in some stormwater systems, prior measurement of our study sites, as well other literature[59], indicated a nitrate-dominated runoff. Future studies will investigate the more complex dual-pathway conversion. A synthetic time series for Nitrate inflow concentrations was generated to simulate loads to upstream elements. This was achieved by assuming a rough correlation between flow and nitrate (2 mgL^{-1}/m^3s^{-1}), which was based on prior measurements[60].

The water treatment for each element was simulated using a continuously stirred tank reactor (CSTR) representation, which is commonly used to simulate similar processes in wastewater treatment plants[54]. Given the dynamic flow conditions that result from real-time control, a closed-form solution that is based on hydraulic residence time does not adequately capture the change in concentration of the pollutant. As such, it becomes necessary to expand into a complete CSTR mass-balance relation[3, 58, 89] to model the concentration C of the dissolved pollutant:

$$\frac{dc}{dt}V + \frac{dv}{dt}C = q_{in}C_{in} - q_{out}C - kCV \tag{2.3}$$

At each time step, the CSTR module communicates with the hydraulic module to update the hydraulic states ($\frac{dV}{dt}, V, q_{out}$ *and* q_{out}). The transformation rate k is computed at each time step based on the hydraulic conditions of the stormwater element. Specifically, denitrification can begin once the oxygen concentration at the soil-water interface drops below a minimum threshold (following a first-order decay assumption). Once this occurs, a constant removal rate k is activated. After the element drains, soil is exposed to the air and must be submerged before denitrification can begin again. As such, the model assumes that cumulative denitrification is maximized when the water is in contact with the most anaerobic soil area.

All simulations were implemented in MATLAB Simulink[76] using a fixed time step solver (ode8 Dormand-Prince[32]) at 5 minute intervals. The step-wise coupled modeling approach was implemented by representing each each module (hydraulic, water quality, and control) as an individual Simulink object (Figure.2.3). All of the source code, inputs and implementation details are attached to this paper as supplementary material.

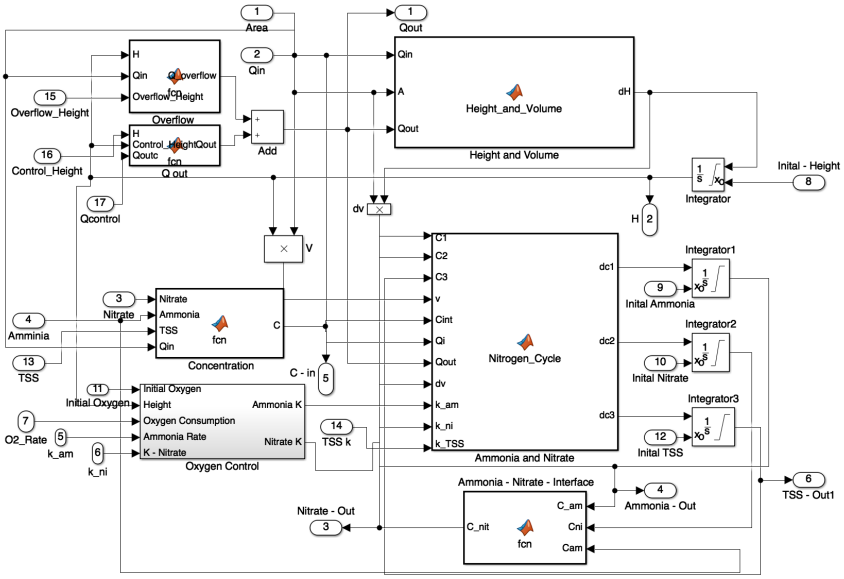

Figure 2.3: MATLAB Simulink implementation of the first case study. The overall model executed in a step-wise fashion and couples stand-alone hydraulic, water quality, and control models.

2.4.2 Case Study 1: Local Control

The first case study is motivated by the objective of controlling a single stormwater basin, which was originally designed for flood remediation as a detention pond (flow-through). The model parameters and physical attributes are provided in the appendix of this paper. In its original configuration the pond merely serves to attenuate peak flows, with little emphasis on water quality. By equipping this pond with a control valve, its original functionality can remain unaffected during large storms by simply keeping the valve open. Major water quality benefits can arise, however, by controlling this pond during smaller and more frequent events.

When enabled, the control algorithm keeps the valve closed and only opens it if the water height exceeds 1.0 m to prevent the pond from overflowing. As a further constraint, when the height exceeds 1.0 m the valve is modulated to ensure that outflows do not exceed $2m^3s^{-1}$, which is the threshold at which downstream sediments are assumed to be re-suspended. Two variations of the control algorithm are also evaluated. The first strategy completely drains the pond before a rain event, thus maximizing captured volumes. Based on the magnitude of the rain event (assumed to be known through a weather forecast), the second strategy only partially drains the pond, maximizing the anaerobic conditions at the soil-water interface and thus speeding up denitrification of the inflows. In this case study, the height of the partially drained configuration was set to 0.15 m, assuming that this height

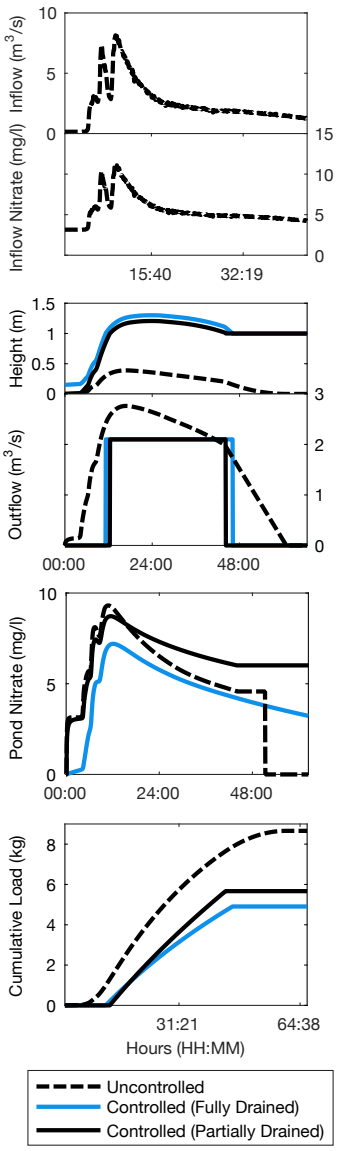

Figure 2.4: Impact of real-time control to hydraulic behavior and nitrate treatment, showing inflow concentrations (top panel), pond water height and outflows (second panel), nitrate concentrations inside the pond (third panel), and cumulative nitrate loads exiting the pond (bottom panel).

would be sufficient to maintain the saturated conditions and prevent the diffusion of oxygen into soil [106].

Compared to the uncontrolled scenario, which only attenuated the peak flow, both controlled scenarios retained a water height of 1.0 m after the storm (Figure.2.4). Since the pond can be drained at a later time, this volume of water was effectively removed from downstream infrastructure during the storm event. In static stormwater systems, volume reductions strategies are typically only assumed to be possible through upstream infiltration and capture. As such, control may effectively serve as a volume reduction strategy by shifting flows outside of the storm window. Furthermore, outflows for the controlled scenarios resembled a "step", which kept flows below a predetermined erosion threshold. This reduces downstream sediment loads, compared to the uncontrolled scenario, whose outflows spent over 50% of the time exceeding the 2 m^3/s erosion threshold.

Nitrate inside the pond and the effluent revealed distinct dynamics between each control configuration. In the uncontrolled scenario, very limited treatment was present due to short hydraulic retention time. The effluent concentrations peaked before dropping to zero since the pond drained completely following the storm. The controlled scenarios did not see this drop-off in internal nitrate because the flows were retained for treatment. The partially-drained scenario showed lower nitrate concentrations at the beginning of the storm due higher anaerobic soil area and denitrification potential.

While internal concentrations are an indicator of treatment dynamics inside the pond, perhaps the best measure of treatment capacity is given by the cumulative nitrate load exiting the pond (bottom panel, figure.2.4). The uncontrolled scenario exhibited the largest cumulative nitrate loads since the runoff effectively just flowed through pond with limited treatment. The controlled pond showed a nearly 43% mass reduction (from 8.6 kg to 4.9 kg) in nitrate due to increased volume capture, HRT and denitrification. The partially-drained control strategy did indicate an improved load reduction compared to the fully-drained controlled (14% improvement). This suggests that, rather than simply draining the pond before storm even, improved load reductions may be achieved through more complex control approaches. More complex control comes at the cost uncertainty however. The partially-drained controller assumed prior knowledge about inflows to decide how much water to drain before a storm. If these decisions are made around weather forecasts, the uncertainty embedded in the inputs may cause adverse impacts, such as overflows. The anticipated benefits of any control strategy should thus always be weighted against the uncertainty of any inputs.

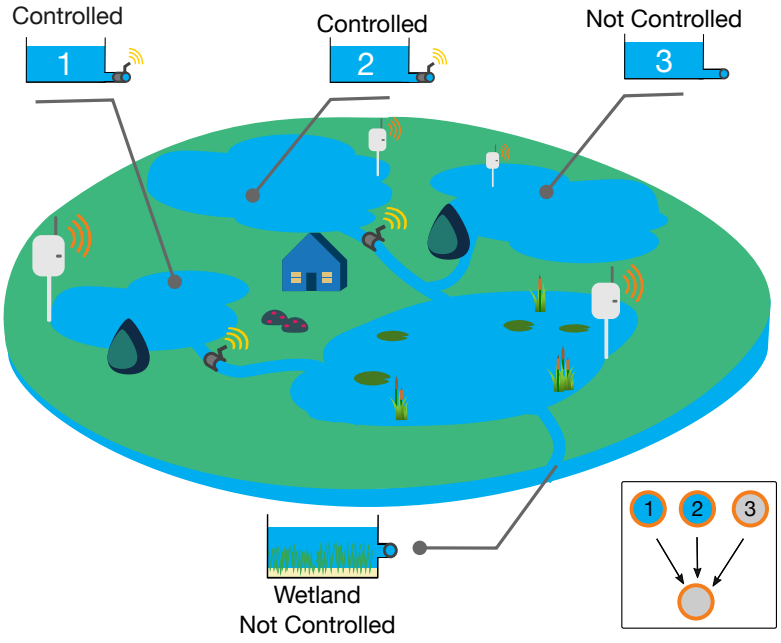

Figure 2.5: System-level control case study: three ponds, two of which are controlled, draining into a treatment wetland.

2.4.3 Case Study 2: System-level Control

The second case study evaluated how control strategies may change when a system of multiple stormwater assets is controlled. A system of four elements was considered, consisting of three parallel ponds draining into a constructed wetland. (Figure.2.5). Two of the upstream ponds were controlled while the treatment wetland and the other pond remained uncontrolled. The objective was to control the upstream ponds to boost the nitrate treatment and reduce the effluent concentrations at the outlet of the wetland. The configuration was based on a real-world site currently being retrofitted for control in southeastern Michigan.

Due to their large biomass area, wetlands have a higher nitrate treatment capacity than ponds[120]. As such, the control objective was to keep the downstream wetland "active", by maximizing its water height and thus the biomass treatment area. While a prolonged inundation may damage the emergent vegetation in the wetland, the proposed control algorithm maximizes the treatment area of wetland only during the duration of the storm event, which should improve the treatment while only briefly inundating the wetland. In the uncontrolled scenario the flows from the upstream ponds actually added up to cause the wetland to overflow (Figure. 2.6, fourth column), which also impaired treatment.

The controlled scenario (see appendix for implementation details) balanced the outflows from the two ponds to ensure that the wetland remained filled (2 m — just below its overflow height) as long as the uncontrolled third pond was discharging. Once the third pond was

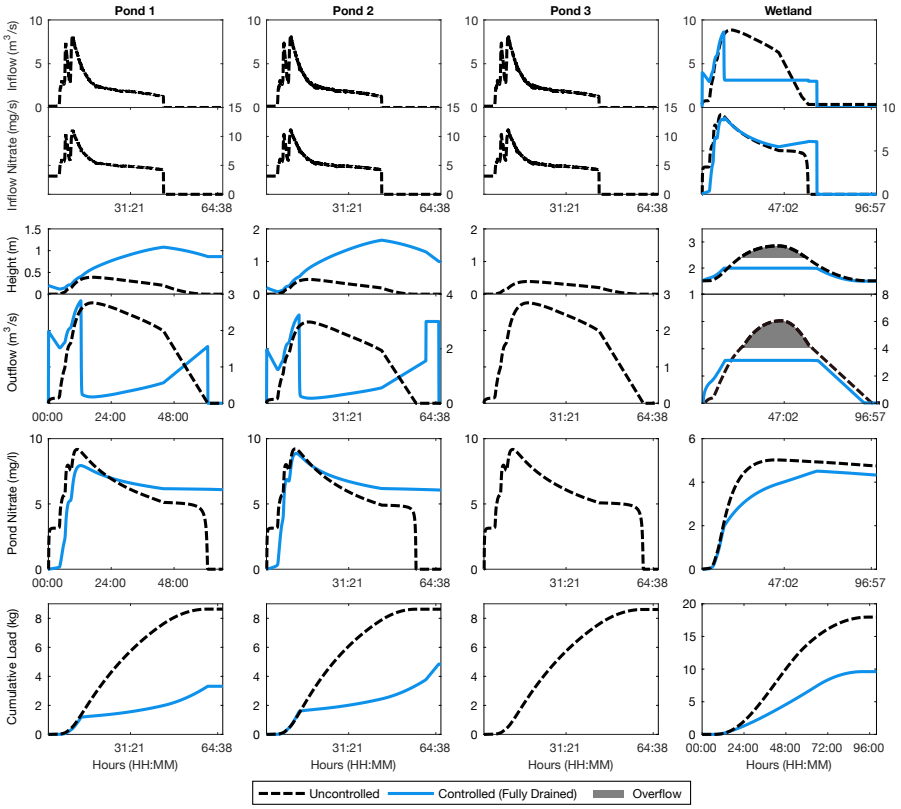

Figure 2.6: Impact of real-time control to hydraulics and nitrate treatment across a system of stromwater elements: inflow concentrations (top row), pond water height and outflows (second row), nitrate concentrations inside each element (third row), and cumulative nitrate loads exiting the pond (bottom row).

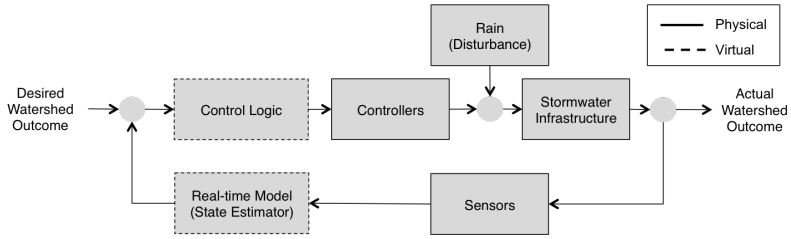

Figure 2.7: The stormwater *feedback control loop*. A desired watershed outcome is compared, in real-time, to actual watershed state based on sensor measurements. The control logic then adjusts the states of valves, gates and pumps to drive the system toward the desired state. Disturbances, such as precipitation, may drive the system away from the desired outcome and must be controlled against when the feedback loop repeats.

entirely drained, the upstream ponds retained any additional inflows, as long as it would not cause them to overflow. This strategy eliminated downstream overflows while simultaneously increasing the wetland's anaerobic treatment area. As such, flows from the third pond were exposed to a larger denitrification than in the uncontrolled case. Overall, the controlled system achieved a 46.48% (from 17.9 kg to 9.6 kg) reduction in cumulative nitrate loads. While some of this overall reduction was driven by the fact that the two controlled ponds remained filled after the storm, thus retaining some nitrate mass upstream, two major benefits arose compared to the uncontrolled scenario. Firstly, the wetland effluent concentrations were reduced over time, showing a 15.25% reduction in concentration. Secondly, the case study showed that a subset upstream elements may be controlled to reduce downstream hydraulic loads, which, similar to the first case study, has the potential to reduce erosion.

A natural extension of this control strategy would be the direct control of the wetland. In many real-world situations, however, not all elements of the system will be controllable. In these instances, system-level benefits may still be achieved via control of other elements. The purpose of this case study was to illustrate one possible example focused on system-level nutrient control. While simple, this control strategy was nonetheless effective at improving the hydraulic and water quality behavior of the overall system. More complex control strategies will be evaluated in the future, especially in the context of larger and more heterogeneous stormwater systems.

2.5 DISCUSSION

Sensor-driven, real-time control of stormwater presents an exciting new paradigm and research area. It is presently unclear, however, how results generated by existing research, as well as the case studies presented

in this paper, can be scaled to large watersheds. Many existing studies focus solely on the control of individual elements and, specifically, on sediment reduction or flood remediation. While the case studies in this paper took a step toward simulating the removal of more complex dissolved pollutants in a multi-element system, it is important to note that the control logic was uniquely tailored to one specific storm and study area. The efficacy of the controls in our case studies was reliant on the ability to hold water after a storm to allow for extended treatment. This strategy may be impacted by limits on hydraulic retention time. Modifying the water levels and residence times, may introduce issues related to aesthetics, plant survival and mosquito breeding[63]. Thus, the potential benefits to water flow and quality must studied as part of a mutli-objective optimization problem.

Much of the real word is underpinned by significant uncertainty, especially related to weather forecasts. Since these forecasts determine when and how much water needs to be released, the stochastic nature of weather must be taken into consideration when controlling such systems.

Control strategies may also change entirely if the removal of different pollutants is required. A simple example can be given by watersheds in which runoff is dominated by ammonia rather than nitrate, thus requiring stages of both nitrification and denitrification. The intricacy of control strategies will likely increase with the number of objectives[138] and the complexity of runoff dynamics. This introduces the exciting paradigm of controlling the overall system to create treatment chains in which individual elements are tuned to achieve specific objectives. By tuning the hydraulic behavior of each element, there will be an unprecedented opportunity to begin applying process-based knowledge from wastewater treatment to distributed stormwater modeling. The modeling of such complete control approaches will be made easier by the simulation approach proposed in Figure.2.2, which will permit for coupling of knowledge spanning hydrology, hydraulics, and water quality.

2.6 KNOWLEDGE GAPS

While research is needed to improve our fundamental understanding and modeling of system-level stormwater, two major knowledge gaps become evident when we view stormwater control in a system-theoretic framework. This can be accomplished by visualizing it as a *feedback loop* (Figure 2.7), a technique common in the control communities and dynamical systems theory[95]. This loop estimates the difference between a desired watershed outcome (downstream nitrate concentrations, for example) and the actual watershed outcome and *feeds* it into control logic to drive the system toward the desired outcome. The physical re-

quirements of this feedback loop, which include sensors, controllers and the physical infrastructure already exist or have matured to the point at which they do not present a major research challenges. Rather, our biggest knowledge gaps span the *virtual* components of the feedback loop and include the (1) assimilation of noisy, sparse, and heterogeneous sensor data into real-time models (state estimation), and (2) the automated synthesis of control logic in response to these estimates.

2.6.1 Toward a new generation of real-time models

Unlike in static infrastructure systems, where adaptation strategies take place on monthly or yearly times scales, real-time control reduces adaptation to minutes or seconds. Existing stormwater models have not been designed to interface with real-time data. Rather, sensor data is often used merely as a convenience to parameterize the model. It is not uncommon for these predictions to drift away from real-world conditions over the modeling horizon. Given the need to base control actions on the best sources of information, a new generation of data-driven and real-time models must be developed. Rather than executing unchecked into the future, they will "learn" from the data and update their states to reflect changing field conditions. Such models will need to be self-calibrating, robust to uncertainty, and computationally efficient to execute in the amount of time required to make control decisions. This raises the question: how complex does a system model need to be to enable an effective and robust control loop? While the answer to this question remains to be investigated, many other control applications (aircraft autopilots, for example) suggest that it is very likely that stormwater control models will not need to be as complex as the models currently used for simulation. This does not mean that existing physically-driven models or our proposed simulation framework (figure 2.2) will not be needed. In fact, existing simulations approaches will be critical in the planning and design of control systems, while real-time models will be used for the actual control.

In our case studies an assumption was made that control actions were informed by known in-situ conditions, such as water flows, pond levels, and nitrate concentrations. This will be far from true in many real-world control systems, where sensors will be sparsely placed and noisy. New models will thus have to be developed to make predictions at locations that are uninstrumented and for parameters that are unmeasured. By quantifying the uncertainty inherent in such models, it will also be possible to develop sensor placement algorithms to determine how many sensors are required and where they should be placed to improve real-time model performance. Many of the methods required for these tasks already exist in other communities (system identification, data assimilation, machine learning, etc), but their application to stormwater systems remains to be investigated.

2.6.2 Control Algorithms

Presently, it is unclear which real-time control and optimization techniques will be the most robust and suitable for distributed stormwater systems. Most current studies, as well as the case studies presented in this paper, have been built around simple rule-based control (e.g. drain a pond before a storm). While such control approaches preserve intuition and incorporate operator expertise, the approach does not scale for systems of arbitrary sizes. This impedes the ability to transfer lessons from one watershed to another. The complexity of operational rules will increase drastically with the size of a watersheds or extended control objectives. The logic associated with operating a network of distributed stormwater assets, comprised of hundreds or thousands of controllers, will become overwhelming unless formal mathematical methods are developed to abstract the physical stormwater dynamics into a system-theoretic framework. These mathematical underpinnings will finally allow for performance or safety guarantees to be provided. This, in turn, will enable new methods to determine how many controllers are needed and where they should be placed to ensure that desired watershed outcomes are met.

2.7 CONCLUSIONS

The goal of this paper was to illustrate the need for a "smart" stormwater systems theoretical framework. Before such systems become adopted, much work remains to be conducted on simulating their performance, which can be accomplished by coupling existing hydrologic, hydraulic and water quality models. As demonstrated by our case studies, real-time control of stormwater has the potential to significantly improve the performance of existing infrastructure, introducing new alternatives to tightly manage nutrients, metals and other pollutants in urban watersheds. Considering current funding mechanisms for stormwater, especially in the United States, the cost of retrofitting will provide a more budget-conscious alternative to new construction while achieving similar or better water quality outcomes. Aside from technical or research gaps, which must be addressed before these systems become reality, it will be imperative to encourage a broad community of researchers, engineers, and cities to adopt these technologies as part of their existing toolboxes. To that end, our team has been spearheading the open-storm.org portal, a collaborative and open-source initiative aimed at sharing end-to-end blueprints and tutorials on software, hardware and sensors required to instrument and control urban watersheds. As the community grows around this exciting new area of research, open-storm.org will track and disseminate its future work.

3 SHAPING THE RESPONSE IN WATERSHEDS USING A SENSOR NETWORK

Burdened by aging infrastructure, growing populations and changing hydrologic conditions, many municipalities struggle to adequately manage stormwater [60]. Flash flooding can occur when stormwater infrastructure is unable to convey runoff away from developed areas [154]. At the same time, pollutants from urban runoff—such as nutrients, heavy metals and microbes—can contaminate downstream waterbodies, damaging aquatic habitats and resulting in toxic algal blooms [60]. Traditionally, civil engineers have addressed these challenges by building larger storage and conveyance infrastructure (e.g. basins and pipes). However, this approach suffers from a number of important disadvantages. First, new construction is expensive, and is often unfeasible for chronically underfunded stormwater departments [87]. Second, static designs are inflexible to future changes in weather, population growth, and regulatory requirements [154]. Third, overdesigned conveyance systems can cause flooding, erosion and damage to downstream property and ecosystems, which ultimately necessitates further remediation and construction [60]. In the face of increasing urbanization and more frequent extreme weather events [14, 133], new strategies are needed to ensure effective management of stormwater.

In contrast to traditional *steel-and-concrete* solutions, real-time control has emerged as a novel means to improve the performance of stormwater systems at minimal expense. Drawing on wireless communications, low-power microcontrollers, and modern advances in control theory, these systems achieve performance benefits by reconfiguring water infrastructure in real time [7, 60]. Real-time control of stormwater basins, for instance, can improve water quality following a storm event by enhancing removal of contaminants [60]. Similarly, active regulation of discharges through constructed wetlands can improve water quality and rehabilitate aquatic habitats [7, 88]. More broadly, by controlling flows over a large network, operators can harness the latent treatment capacity of many distributed stormwater assets, effectively turning urban watersheds into distributed wastewater treatment plants [7, 60].

A small number of studies have evaluated the benefits of real-time stormwater control. Most of these studies describe retrofits of isolated sites for rainwater capture and on-site pollutant treatment. Middleton and Barrett (2008) show that equipping existing retention basins with real-time controllers can reduce stormwater pollutant loads downstream by increasing the retention time of captured stormwater [82].

Roman et al. (2017) describe an adaptively-controlled rainwater harvesting system in New York City that captures 35–60% more rainwater than conventional systems [110]. Similarly, Klenzendorf et al. (2015) describe a rainwater harvesting pilot project and a retention basin retrofitted for real-time control in Austin, Texas [62]. The authors show that the controlled retention basin reduces deposition of nitrogen and total suspended solids (TSS) into the downstream system. These studies demonstrate that active control can significantly improve the performance of existing sites at a lower cost than new construction. However, benefits are only examined at a local scale. This distinction is important, given that localized practices do not necessarily achieve the best system-scale outcomes. Indeed, some research indicates that when local best management practices are implemented without accounting for global outcomes, they can produce adverse flow conditions at the watershed scale [34].

Currently, the benefits of coordinated stormwater control are poorly understood. Inspiration for the benefits of system-level control can be taken from sewer operations. While most sewer systems still only rely on local control logic, such as water level setpoints [123], recent work has demonstrated how wider benefits can be achieved through the cooperative action of multiple controllers working in tandem. The cities of Copenhagen and Barcelona, for instance, implement a combination of local rule-based control, and some higher-level optimization that jointly coordinates actions between groups of actuators [85]. Montestruque and Lemmon (2015) describe CSOnet, a sewer control network consisting of 120 sensors and 12 actuators in the city of South Bend, Indiana [87]. This network uses dynamic control algorithms to adaptively balance hydraulic loads throughout the sewer's interceptor lines, ultimately reducing combined sewer overflows (CSOs) by as much as 25%. While these systems achieve impressive system-scale control of a large sewer networks, it is still unclear how lessons learned from these proprietary sewer control approaches may translate to the broader control of urban watersheds and separated stormwater systems.

In this study, we describe an approach for managing stormwater discharges across an urban watershed using internet-connected valves and sensors. We show that by actively coordinating releases from two parallel retention basins, we can produce desirable flow regimes at a target location downstream, which would not be possible with passive infrastructure alone. This study takes place in four phases. In the first phase, we describe the development of a real-time stormwater control system in the city of Ann Arbor, Michigan. Building on an existing wireless sensing and control network described in Bartos et al. (2018) [7], we demonstrate how static retention basins can be retrofitted with internet-controlled valves, and present a new method for controlling these basins using a controller scheduling application. In the second phase, we characterize the ability of the control network to

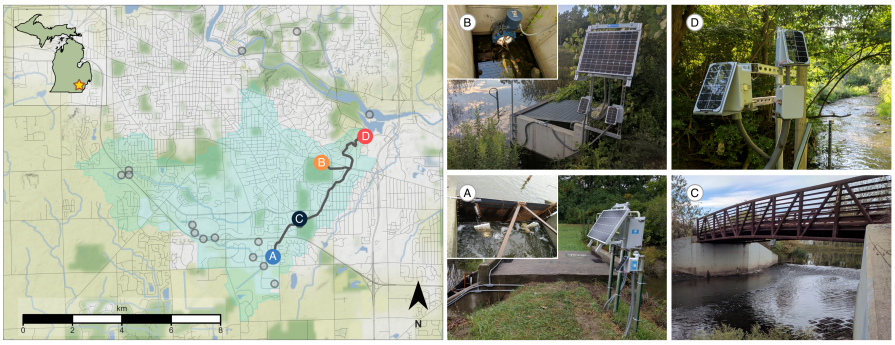

Figure 3.1: Overview of the study area. The map (left) shows the location of relevant control and sensor sites, additional sensor sites (light grey), flow paths between each site (dark grey), and the contributing area of the watershed (light blue). Site images (right) show the two control sites (A & B) along with two downstream sensor locations (C & D).

shape the downstream hydrograph by releasing impulses of different sizes from two retention basins and determining the magnitude, travel time, and decay envelope of the resulting waves. In the third phase, we use the data gathered from this exploratory analysis to determine the control input needed to produce a flat hydrograph at the outlet of the watershed. We discuss how this control strategy can be used to prevent erosion and reduce phosphorus loads into downstream waterbodies. Finally, in the fourth phase, we show how control inputs can be timed to produce synchronized and de-synchronized pulses at a downstream target location. In addition to demonstrating the precision of the control system, this experiment shows how interleaving pulses can be used to free up capacity in upstream retention basins without inducing synchronized flashy flows downstream. We discuss how these simple control "building blocks" can be used by system operators to achieve more sophisticated stormwater management targets. Unlike most existing systems, our control network uses an open-source hardware and software stack, making it freely available to municipalities that are interested in implementing their own smart stormwater control systems. Thus, when combined with supplementary *how-to* documentation on `open-storm.org`, this study provides the foundation for an "operator's manual" for real-time control of urban watersheds.

3.1 STUDY AREA AND TECHNOLOGIES

3.1.1 Study area

This study focuses on a wireless control network in the Mallets creek watershed—an urbanized creekshed located in the city of Ann Arbor, Michigan. This creekshed has been the focus of ongoing efforts to re-

duce peak flows and improve water quality [71]. The creekshed has an area of about 26.7 km^2 and contains streams that altogether exceed 16 km in length. These streams drain into the Huron River and ultimately the Great Lakes. With high areas of development and over 33% imperviousness, little natural land is available for infiltration and uptake, resulting in flashy flows that erode stream banks and result in unstable habitats. These rapid flows drive stream erosion and increased transport of sediments and nutrients out of the watershed [71]. While there are no lakes in the creekshed, there are several natural and manmade stormwater basins that that have been constructed to help stabilize flows throughout the creekshed and mitigate the impacts of non-point source runoff.

To investigate the effects of real-time control on the creekshed, we deploy a control network that measures and regulates flows from two large stormwater basins. The control network consists of four sites centered around the main stem of the creek. Figure 3.1 shows the locations of each of these four sites in the control network. Water first flows into a large retention basin with a storage capacity of 19M liters (site A), located at the most upstream point in the control network. From this retention basin, water travels 1.4 km downstream to a constructed wetland (site C), designed to slow the flow of water and remove contaminants. After passing through the wetland, water travels another 3 km until it is joined by flows arriving from a smaller retention basin with a storage capacity of 7.5M liters (site B). The combined flows exit the creek at the outlet of Mallet's creek (site D), after which they enter the Huron River. Internet-controlled valves are deployed at the two stormwater basins at sites A and B. These valves are used in subsequent experiments to regulate flows at the outlet of the creek.

3.1.2 Technologies and Architecture

Flows throughout the creekshed are measured and controlled using a custom wireless sensing and control network. This network is built using the `open storm` hardware and software stack, which has been described and documented in Bartos et al. (2018) [7]. The hardware layer uses an ultra-low power ARM Cortex-M3 microcontroller (Cypress PSoC), which implements the sensing and control logic in its firmware. Internet connectivity is achieved using a CDMA cellular modem (Telit DE910), which facilitates wireless bi-directional communication between the field device and a remote server. The full unit is powered using a solar-rechargeable 3.7V lithium-ion battery. To measure the hydrologic response of the system, wireless sensor nodes are deployed along the main stem of the creek. Each sensor node is equipped with an ultrasonic depth sensor (Maxbotix MB7384) to measure water levels (shown in Figure 3.1, Site C). At the time of writing, sensor nodes can be constructed using less than $500 USD of parts.

To control discharges throughout the creekshed, stormwater basins are retrofitted with one of two valves: (i) a 0.3 m diameter butterfly valve (Dynaquip MA44) (Figure 3.1, Site B) or (ii) a 0.3 m gate valve (Valterra 6912) mated to a linear actuator (AEI 6112CH) (Figure 3.1, Site A). Each control valve is connected to a sensor node. The valves are actuated by the microcontroller and powered by rechargeable 12V sealed lead-acid batteries. Solar panels allow the control sites to operate without line power. Assuming that the valve can be attached to a basin's outlet without structural modification, each control site can be constructed using less than $3500 USD of parts at the time of writing.

Remote control of valves and sensors is implemented using a *polling* scheme, in which field-deployed nodes request commands from a remote server (Figure 3.2). To conserve power, nodes spend most of their time in a deep sleep state, consuming only 1–10 μA of current. Upon waking up, each node takes sensor readings and transmits the readings to a cloud-hosted time series database (InfluxDB) via authenticated (and optionally encrypted) *HTTP* requests. Before going back to sleep, the node polls a set of commands from a dedicated feed in the same database. The commands may include, but are not limited to, changing the sampling frequency, triggering additional sensor readings, or opening a valve. Operations can be cancelled and rescheduled either by the application or by an operator. This is useful if, for example, the application detects that a control action was not successfully executed and that pending operations need to be rescheduled. Most importantly, the database supports modern web service standards and application programming interfaces (APIs), which allow the control logic to be quickly implemented via simple web applications. These applications can be written in any number of popular programming languages (Python, Matlab, etc). This feature improves flexibility, reduces reliance on low-level firmware updates, and allows for the seamless integration of external data sources, such as public weather forecasts [7, 151].

For the experiments described in this study, field devices in the creekshed are controlled using a simple Python web application. This application can be executed in either automatic or manual mode. In automatic mode, the application queries water level sensor feeds, rainfall forecasts, and external flow measurements from a publicly-listed measurement station at the outlet of the creekshed (USGS 04174518). Based on these sensor readings, new commands are then written to the database to open and close valves. In manual operation, a predefined set of commands is written to the database, then subsequently executed by the field device. For this study, the manual operation mode is used. The application toolchain is implemented on an Amazon Web Services (AWS) medium-sized linux Elastic Compute Cloud (EC2) instance.

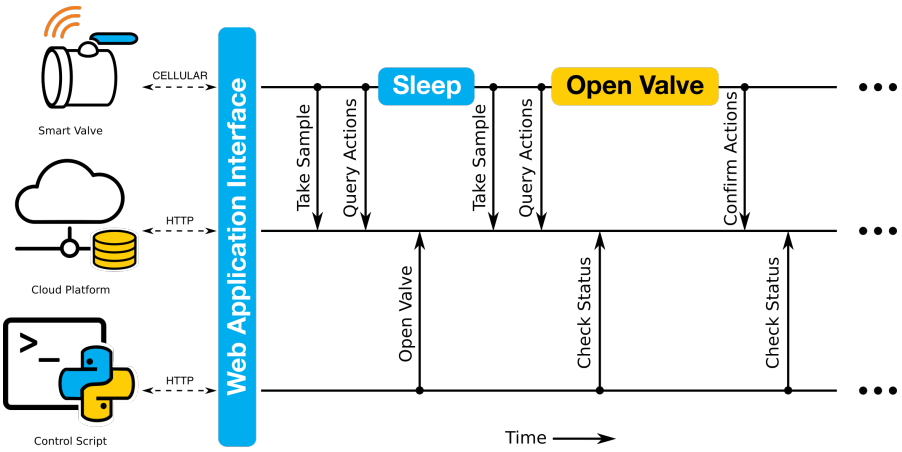

Figure 3.2: Control system architecture. Field-deployed nodes use a polling system to download and execute commands issued from a remote server. Control actions can be specified manually, or through automated web applications and scripts.

3.2 CHARACTERIZING CONTROL ACTIONS

Before evaluating potential control strategies, we first characterize the ability of each control site to shape downstream flows. Specifically, we quantify the travel time P and decay time D, of various waves as they move between the originating control site and the outlet of the watershed. The characterization is accomplished by releasing pulses of different durations from each stormwater basin and then observing the resulting waves that these pulses generate downstream. To limit confounding effects caused by rainfall, these experiments are carried out during dry conditions (at least 4 days following a storm). Figure 3.3 shows a 1-hour release, 4-hour release, and 48-hour release from retention basin A (shown left to right, respectively). The 48-hour release empties the retention basin, meaning that this release characterizes the maximum possible output from site A. The travel times for each wave from site A to site C are approximately 3.5 hours (time to start of rise) and 6–8 hours (time to peak), with faster rise times for the larger releases due to nonlinearities in the speed of wave propagation. The decay times for each release are 6 hours, 18 hours and 44 hours, respectively. From this experiment, it can be seen that the maximum change in flow that site A can generate at the outlet is roughly 0.17 m^3/s. Similar experiments are used to characterize site B. From these experiments, we estimate average travel times from site B to the outlet of 1.5 hours (time to start of rise) and 1.8 hours (time to peak), with an average decay time of 3 hours, and a maximum change in flow of approximately 0.2 m^3/s.

In addition to release duration, sites are also characterized with respect to the hydraulic head (water level) of the originating retention basin. Figure 3.4 shows the result of releasing three 1-hour pulses from

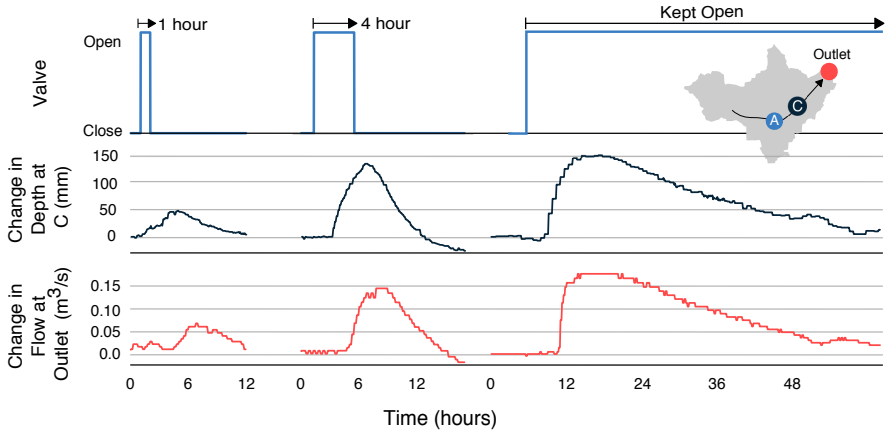

Figure 3.3: Characterization of control actions from site A. In the first two experiments, the valve at site A is opened for 1-hour and 4-hour durations. For the third experiment, the valve is held open indefinitely. The resulting waves travel through a constructed wetland (site C) before arriving at the outlet of the watershed. Wave depth (black line) is measured at the wetland, while flow rate (red line) is measured at the outlet.

site B, without allowing the basin to refill between releases. While the same duration is used for each release, the hydraulic head (stored volume) of the retention basin decreases with each pulse. Thus, the resulting wave becomes smaller with each successive opening of the valve, even though the same input signal is used. In spite of this difference, the travel time and decay time of the wave remain consistent between each release. The magnitude of the resulting wave varies from roughly 0.2 m^3/s to 0.13 m^3/s, depending on the water level in the basin.

Although retention basin B is significantly smaller than retention basin A, it can produce a comparable change in flow at the watershed outlet (approximately 0.2 m^3/s). This effect can be attributed to two main factors. First, site B is located closer to the outlet (3.0 km as opposed to 5.9 km for site A), meaning that the wave is subject to less hydraulic dispersion. Second, the retention basin at site B is elevated higher above the receiving stream, meaning that flows exit the control structure more rapidly than flows released from site A. Thus, compared to site A, site B produces short pulses with a rapid onset and large peak. Despite its relatively smaller volume, control actions from site B must thus be tailored to avoid generating flashy flows at the outlet.

One crucial result of these experiments is that for the purposes of control, nonlinearities in wave propagation can be safely ignored. Shallow-water waves exhibit a nonlinear relationship between wave height and wave speed, meaning that larger waves propagate faster [61]. If these nonlinearities were significant, then control strategies would need to account for changes in travel time due to (i) variations in release durations, (ii) variations in basin head, and (iii) superposition of waves originating

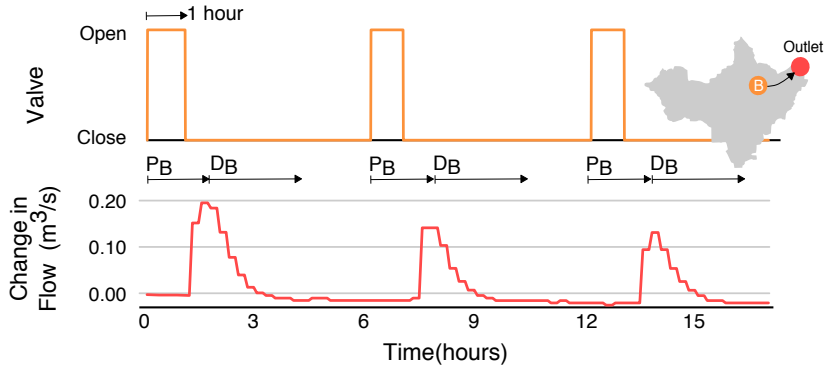

Figure 3.4: Characterization of control actions originating from site B. Three subsequent pulses are released. While the duration of each control pulse is the same (1 hour), the magnitude of the flow at the outlet decreases because the hydraulic head (pressure) in the basin is reduced with each release.

from different locations. For the system examined in this study, the effect of these nonlinearities is small. Namely, while nonlinearities in wave propagation affect the shape of the resulting hydrograph (skewing the peak toward the left), they do not significantly affect the bulk travel time of an isolated wave. Specifically, the travel times for Site A and Site B remain consistent (3.5 hours and 1.5 hours, respectively) despite scheduling releases of different durations and magnitudes. This result is consistent with findings from previous studies that use linear dynamics for stormwater system control [42, 74, 78]. Thus, for the scale of our creekshed the travel time of a wave originating at an upstream stormwater basin can be considered independent of both the amount of water released and the water level of the originating basin. Moreover, superposition of two waves from two parallel sources does not effect a noticeable change in bulk wave speed. This result suggests that for the purposes of control, the channel network may be approximated as a linear system in which waves originating from each retention basin can be superimposed in order to produce a desired output hydrograph downstream.

By characterizing the downstream response to various impulsive inputs, these initial experiments yield a set of "building blocks" that are subsequently used to achieve more complex control objectives at the watershed outlet. While the propagation of waves within a channel network is described by nonlinear equations, we find that a linear system approximation adequately describes the dynamics needed to generate control strategies. Thus, the characterization experiments described in this section are conceptually analogous to quantifying the unit impulse response of a linear system. This framework suggests that desired waveforms can be generated via simple linear combinations of known input signals. With this conceptual model in hand, we carry out a number of control experiments to showcase the utility of the stormwater control

network. First, we show how pulse-width modulation of a valve can be used to produce a flat hydrograph that meets but does not exceed a given flow threshold. Next, we show how valve releases can be timed to generate synchronized and desynchronized waves at the outlet. These experiments provide recipes for managing releases from upstream retention basins while simultaneously fostering desirable flow conditions downstream.

3.3 SET-POINT HYDROGRAPHS

Real-time control can be used to flatten downstream hydrographs, helping to reduce erosion and maintain healthy aquatic ecosystems. In passive stormwater systems, hydrographs often exhibit a distinct peak, preceded by a rapid rise and followed by a slower decay. While typically associated with rain events, this phenomenon can also be observed when water is released from a retention basin (see Figures 3.3 and 3.4). Peak flows that exceed downstream capacity will often lead to flooding. Furthermore, urban streams can become unstable if a critical flow velocity or flow rate is reached [11]. Exceedance of these thresholds may lead to ecological damage and stream erosion, as well the mobilization of sediments. These sediments in turn may carry nutrients, metals and other pollutants downstream, impairing water quality and promoting the growth of algal blooms [80]. This particular impairment underpins the major challenge of "urban stream syndrome", forcing many cities to spend millions of dollars to reduce downstream flow rates [118, 148]. While active control has been proposed as a means to condition stormwater flows, the specific control strategies needed to achieve stable flow conditions within an urban watershed are currently not well understood.

To address this challenge, a sequence of control actions is designed to yield a constant set-point condition at the outlet of the watershed. Specifically, we aim to create a flat hydrograph, for which the flow rate remains close to (but does not exceed) a specified value. While the set-point used in this experiment is chosen arbitrarily, this threshold may be chosen to control for objectives related to downstream flooding and water quality—for instance, ensuring that the critical flow threshold for sediment transport is not surpassed. To achieve a constant set-point flow rate, we derive inspiration from *pulse-width modulation*—a method used in electrical systems to generate analog signals from discrete digital pulses. Isolated pulses of water are emitted from the control site, spaced apart such that the arrival time of each wave overlaps with the receding limb of the prior wave. As the pulses travel through the channel network, they disperse, causing the individual waves to overlap and combine. The resulting superposition of partly-dispersed waves results in an approximately constant flow rate.

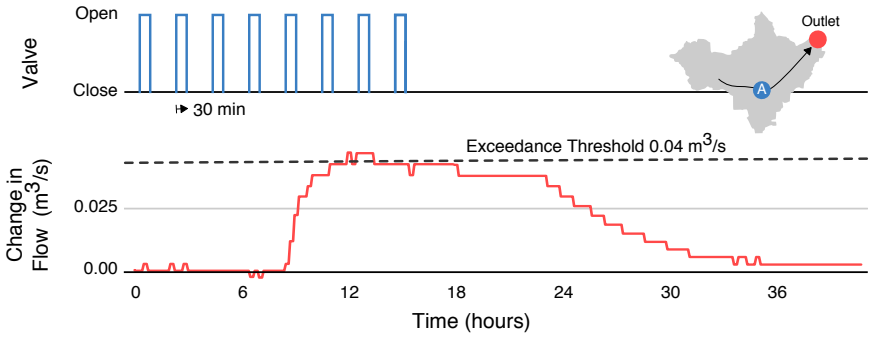

Figure 3.5: Generating a set-point hydrograph. Small, evenly spaced pulses (30-minute duration) are released from the controlled basin. The pulses disperse as they travel through the 6 km-long stream, leading to a relatively flat response at the outlet of the watershed.

As seen in the hydrograph response (Figure 3.5), the "flat hydrograph" objective is achieved by modulating the valve position in successive 30-minute pulses. The flows at the outlet remain approximately flat, without significantly exceeding a setpoint of 0.04 m^3/s. Of course, the shape is not perfectly flat, given the large distance between the two sites and nonlinearities inherent in wave propagation. However, these experimental results show that active modulation of a valve can produce highly stable flow conditions downstream that would not be possible using passive infrastructure alone. In a real-world scenario, this control strategy could be used to drain a watershed as fast as possible without exceeding a critical flood conditions downstream. Minimizing the change in flows downstream also reduces the likelihood of stream erosion. From our prior studies in this creekshed that were not affected by real-time control [150], it can be estimated that pollutant concentrations during this flat stage were no greater than 127 mg/L for sediment and 0.209 mg/L for total phosphorus. For comparison, keeping the valve open would have resulted in concentrations of at least 390 mg/L for sediment and 0.618 mg/L for total phosphorus. By modulating the valve position to achieve a relatively flat and steady outflow, the control actions likely reduced the total mass of solids and phosphorus that would otherwise contribute to ecological damage and harmful algal blooms. Future studies will confirm and refine these estimates by measuring real-time water quality changes that result from control.

3.4 COORDINATED RELEASES BETWEEN MULTIPLE CONTROL SITES

Motivated by the larger goal of watershed-scale control, a final experiment is devised to evaluate the level of precision that can be achieved when coordinating releases from multiple sites. Namely, we schedule re-

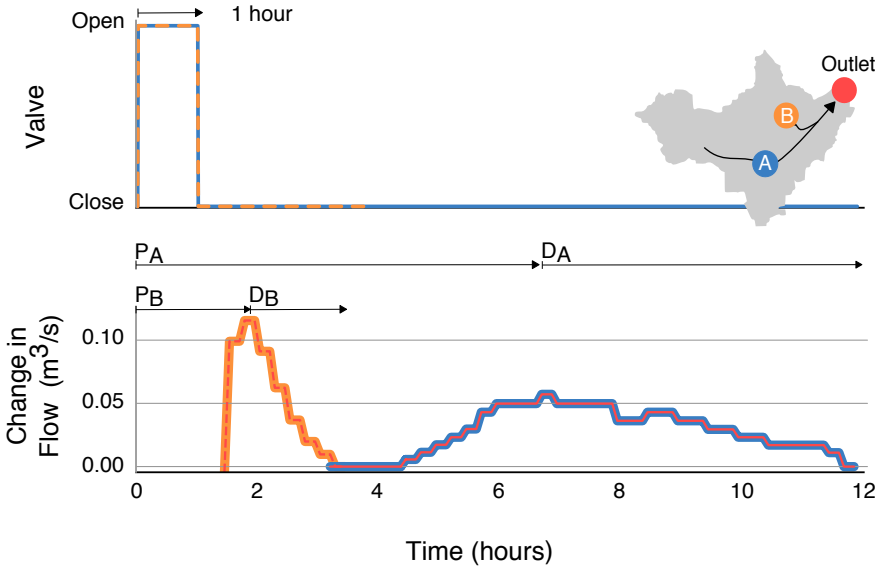

Figure 3.6: Flows at outlet of watershed resulting from 1 hour releases from each control site. Time to peak P, magnitude, and decay time D for each release are labeled.

leases from the two controlled basins in order to produce synchronized and interleaved pulses at the outlet. Before running the experiment, we first determine the control signals needed to generate the combined and interleaved waves, respectively, by assessing the travel time and decay time of waves released from each retention basin. Figure 3.6 shows the hydrographs resulting from 1-hour pulses released simultaneously from site A and site B. Based on the travel times of each wave, it can be seen that in order to achieve a synchronized wave at the outlet, a 1-hour release from site B must be scheduled approximately six hours after a 1-hour release from site A. Conversely, to achieve an interleaved pattern at the outlet, the following pulse train can be used: (i) release a 1-hour pulse from site A, (ii) release a pulse from site B approximately 12 hours later, (iii) release a pulse from site A after waiting an additional four hours, and (iv) repeat the pattern starting at step (ii).

Once the input signals required to produce each desired shape are known, we schedule a series of commands to be executed by each valve. The experiment is divided into two stages. During the first stage, flows from the control sites are released such that the peaks of the hydrographs overlap. In the second stage of the experiment, the flows are released off-phase, such that the flows arriving from one site begin exactly when the flows from the other site recede. Figure 3.7 shows the result of this experiment, with the overlapping waves occurring from hours 6 to 15, and the interleaved waves occurring from hours 15 to 44. As hypothesized earlier, the superposition of waves is approximately linear. In other words, the maximum change in flow is approximately equal to the sum of the maximum flow of each component wave. More-

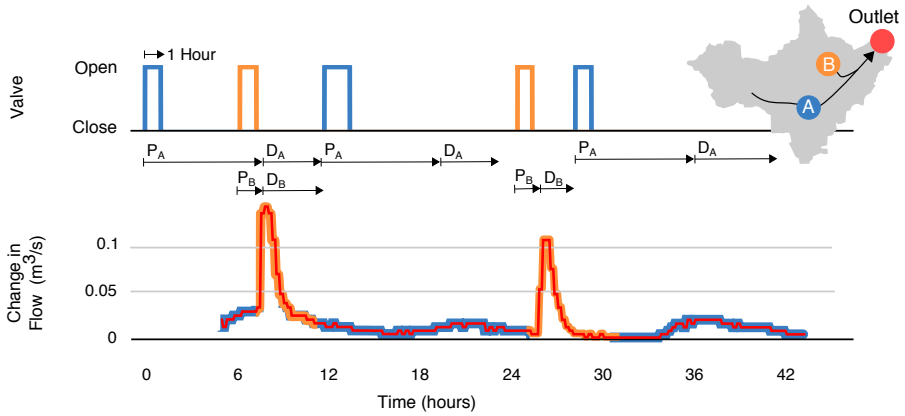

Figure 3.7: Superposition and interleaving of waves from retention basins A and B. Overlapping waves (coincident peaks) are generated from hours 6 to 12. Interleaved waves (off-phase peaks) are generated from hours 18 to 44.

over, the superposition of the two waves does not appear to appreciably change the bulk travel time.

This experiment shows that real-time control of stormwater systems can achieve precise control over downstream flow conditions, and also suggests a strategy for coordinating releases in order to remove stormwater from retention basins while simultaneously achieving target flow conditions downstream. Like the set-point experiment, an interleaving control pattern can be used to de-water upstream retention basins without exceeding a particular flow threshold downstream. When waves generated by several upstream retention basins combine, they can generate large, flashy flows at a downstream location. This in turn can contribute to erosion of the surrounding channel. For this reason, it is desirable to avoid the collision of waves from two different upstream sources. By interleaving flows from upstream retention basins, one can free up capacity in the system without generating adverse flow conditions downstream. More broadly, the results of this experiment demonstrate the fine level of flow control that can be achieved across urban watersheds using a low-cost sensor and control network. While the underlying control logic only uses rudimentary time-of-travel metrics it nonetheless produces desirable flow regimes that would be difficult to achieve with passive infrastructure alone. As such, this experiment builds a foundation for more complex control strategies by verifying that the watershed responds consistently and predictably to individual control actions. This result suggests that future studies may one day demonstrate more complex, possibly near-arbitrary, hydrograph shapes. Time of travel may not be sufficient for such approaches, however, and more complex and analytical control techniques should be considered.

3.5 CONCLUSIONS

This study shows how internet-connected stormwater control valves can be used to shape streamflows within a large urban watershed. To our knowledge, this study is the first to document how coordinated releases between multiple stormwater control sites can satisfy system-scale watershed performance goals—such as maintaining downstream flow at a constant rate or preventing sediment transport. Building on an existing wireless sensor network, we demonstrate how static stormwater retention basins can be retrofitted with internet-controlled valves to enable active control at a low cost. Characterizing the system in a series of exploratory experiments, we find that a linear approximation is sufficient to describe the downstream response associated with a given input. Next, we use the system to generate two flow conditions downstream: (i) a set-point hydrograph in which flow is maintained at a roughly constant rate, and (ii) a series of overlapping and interleaved waves. We find that pulse-width modulation of upstream valves generates a flat downstream response. Similarly, interleaving of discharges provides an effective tool for emptying upstream retention basins without inducing flashy flows downstream. In addition to demonstrating the precision of the control system, these experiments suggest strategies for managing stormwater transfers across a watershed while maintaining desired flow conditions. To make the smart stormwater system described in this paper accessible to water managers worldwide, all hardware, software and documentation for this project are made available at `open-storm.org`.

4 DEEP REINFORCEMENT LEARNING FOR THE CONTROL OF STORMWATER NETWORKS

Urban stormwater and sewer systems are being stressed beyond their intended design. The resulting symptoms manifest themselves in frequent flash floods [69] and poor receiving water quality [146]. Presently, the primary solution to these challenges is the construction of new infrastructure, such as bigger pipes, basins, wetlands, and other distributed storage assets. Redesigning and rebuilding the existing stormwater infrastructure to keep in pace with the evolving inputs is cost prohibitive for most communities [60]. Furthermore, infrastructure is often upgraded on a site-by-site basis and rarely optimized for system-scale performance. Present approaches rely heavily on the assumption that these individual upgrades will add up to cumulative benefits, while the contrary has actually been illustrated by studies evaluating system-level outcomes [34]. The changing and highly variable nature of weather and urban environments demands stormwater solutions that can more rapidly adapt to changing community needs.

Instead of relying on new construction, a new generation of smart stormwater systems promises to dynamically re-purpose existing stormwater systems. These systems will use streaming sensor data to infer real-time state of a watershed and respond via real-time control of distributed control assets, such as valves, gates, and pumps [60]. By achieving system-level coordination between many distributed control points, the size of infrastructure needed to reduce flooding and improve water quality will become smaller. This presents a non-trivial control challenge, however, as any automated decisions must be carried with regard to public safety and must account for the physical complexity inherent to urban watersheds [88, 123].

In this paper, we investigate *Deep Reinforcement Learning* for the real-time control of stormwater systems. This approach builds on very recent advances in the artificial intelligence community, which have primarily focused on the control of complex autonomous systems, such as robots and autonomous vehicles [73, 84]. In this novel formulation, our algorithm will *learn* the best real-time control strategy for a distributed stormwater system by efficiently quantifying the space of all possible control actions. In simple terms, the algorithm attempts various control actions until discovering those that have the desired outcomes. While such an approach has shown promise across many other domains, it is presently unclear how it will perform and scale when used for the real-time control of water systems, specifically urban drainage networks.

The fundamental contribution of this paper is a formulation of a control algorithm for urban drainage systems based on Reinforcement Learning. Given the risk to property and public safety, it is imprudent to hand over the control of a real-world watershed to a computer that learns by mistake. As such, a secondary contribution is the evaluation of the Reinforcement Learning algorithm across a series of simulations, which span various drainage system complexities and storms. The results will illustrate the benefits, limitations, and requirement of Reinforcement Learning when applied to urban stormwater systems. To our knowledge, this is the first formulation of Deep Reinforcement Learning for the control of stormwater systems. The results of this study stand to support a foundation for future studies on the role of Artificial Intelligence in the control of urban water systems.

4.1 REAL TIME CONTROL OF URBAN DRAINAGE SYSTEMS

Since the European Union's Directive on water policy [100], there has been a significant push towards the adoption of real-time control for improving wastewater and sewer systems [85, 123]. Many of these control approaches fall broadly under the categories of real-time control (RTC, control decisions made solely on the real-time state of the system), and Model Predictive Control (MPC, decisions that account for predicted future conditions). During the past decade, MPC has emerged as a state-of-the-art methodology for developing control strategies and analyzing their potential for controlling urban drainage and sewer networks in simulated setting. MPC has been used to regulate dissolved oxygen in the flows to aquatic bodies [77], control inflows to wastewater treatment plants [104], and enhance the system-level performance and coordination of sewer network assets [79, 85]. These and many other simulation based studies [149] have illustrated the benefits of control, the biggest of which is the ability to cost-effectively re-purpose existing assets in real-time without the need to build more passive infrastructure.

The performance of MPC depends on the extent to which the underlying process can be approximated using a linear model [142]. A benefit of this linearity assumption is the ability to analytically evaluate the stability, robustness and convergence properties of the controller [95], which is valuable when providing safety and performance guarantees. Network dynamics of storm and sewer systems and transformations of the pollutants in runoff are known to be heavily non-linear. This demands a number of approximations and a high level of expertise when applying Model Predictive Control. Furthermore, real-world urban watersheds are prone to experiencing pipes blockages, sensor breakdowns, valve failures, or other adverse conditions. Adapting and re-formulating linear control models to such non-linear conditions is difficult, but is

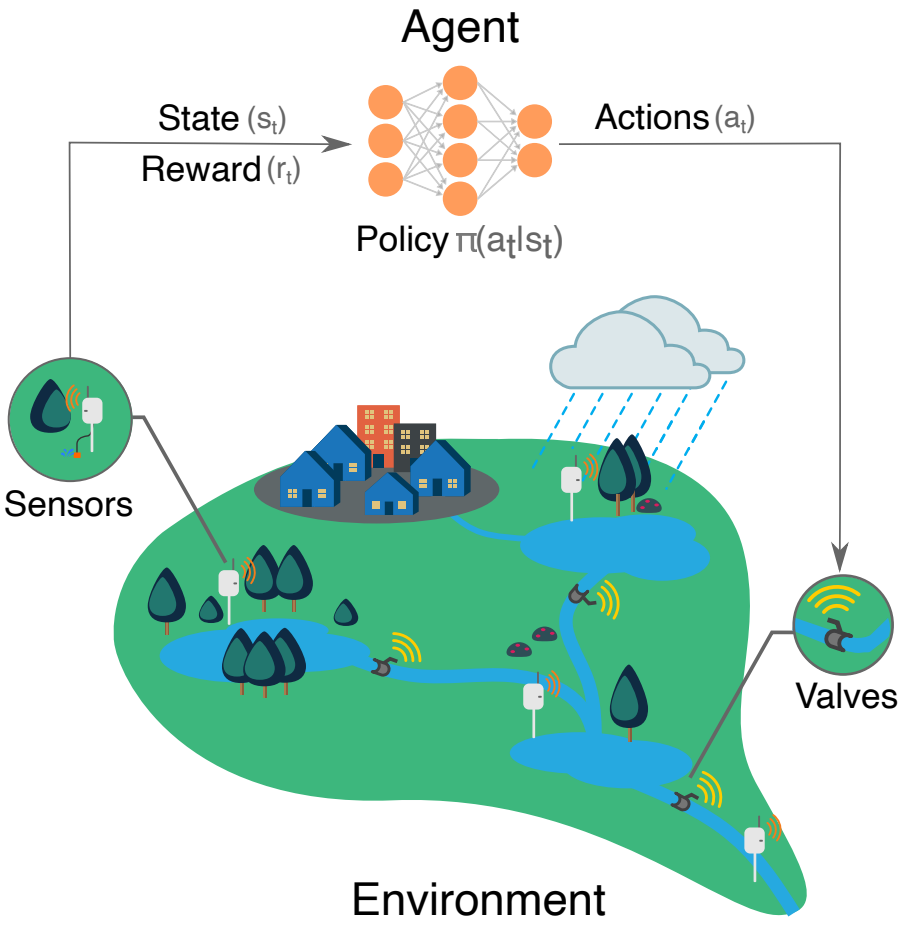

Figure 4.1: During a storm event, a reinforcement learning controller observes the state (e.g. water levels, flows) of the stormwater network and coordinates the actions of the distributed control assets in real-time to achieve watershed-scale benefits.

being addressed by promising research [149]. The constraints of linear approximations and the need for adaptive control algorithms open the door to exploring other control methodologies, such as the one presented in this paper.

4.2 REINFORCEMENT LEARNING

Across the Artificial Intelligence and Behavioral research communities, Reinforcement Learning (RL) has emerged as a state-of-the-art methodology for autonomous control and planning systems. Unlike in classical feedback control, where the controller carries out a pre-tuned and analytical control action, an RL controller (i.e. an RL agent) learns a control strategy by interacting with the system — effectively trying various control strategies until learning those that work well. Rather than just learning one particular control strategy, an RL agent continuously at-

tempts to improve its control strategy by assimilating new information and evaluating new control strategies [135]. RL can be used in a model free context since the system's dynamics are implicitly learned by evaluating various control actions. Leveraging the recent advancements in Deep Neural Networks and the computational power afforded by the high performance clusters (HPCs), RL agents have been able to plan complex tasks, such as observing pixels to play video games at a human level [84], defeating world champions in the game of GO [131], achieving "superhuman" performance in chess [130], controlling high speed robots [64], and navigating autonomous vehicles [92]. Despite the wide adoption of Deep Neural Network based Reinforcement Learning (Deep RL) in various disciplines of engineering, its adoption in civil engineering disciplines has been limited [2, 10, 21]. Deep RL control has yet to be applied to the real-time control of urban drainage systems.

Deep RL agents approximate underlying system dynamics implicitly, hence not requiring a simplified or linearized control model [135]. A Deep RL agent instantaneously identifies a control action by observing the network dynamic, thus reducing delay in the decision process [84, 130]. The explorative nature of the Deep RL agents also enables the methodology to adapt its control strategy to changing conditions of the system [135]. Hence, Reinforcement Learning shows promise as a potential alternative or supplement to existing control methods for water systems. To that end, the goal of this paper is to formulate and evaluate of Reinforcement Learning for the real-time control of urban drainage systems. The specific contributions of the paper are:

1. The formulation and implementation of a reinforcement learning algorithm for the real-time (non-predictive) control of urban stormwater systems.

2. An evaluation of the control algorithm under a range of storm inputs and network complexities (single stormwater basins and an entire network), as well as an equivalence analysis that compares the approach to passive infrastructure solutions.

3. A fully open-sourced implementation of the control algorithm to promote transparency and permit for the direct application of the methods to other systems, shared on open-storm.org.

4.3 METHODS

4.3.1 Reinforcement learning for stormwater systems

When formulated as a Reinforcement Learning (RL) problem, the control of stormwater systems can be fully described by an agent and environment (figure 4.1). The environment represents an urban stormwater system and the agent represents the entity controlling the system. At

any given time t, the agent takes a control action a_t (e.g. opening a valve or turning on a pump) by observing any number of states s_t (e.g. water levels or flows) in the environment. Based on the outcomes of its action, the agent receives a reward r_t from the environment. The reward is formulated to reflect the specific control objectives. For example, an agent could receive positive reward for preventing flooding or a negative reward for causing flooding. By quantifying these rewards in response to various actions over time, the agent learns the control strategy that will achieve its desired objective [135]. The agent's control actions in any given state are governed by its policy π. Formally, the policy is a mapping from a given state to the agent's actions:

$$\pi : s_t(\mathbb{R}^n) \rightarrow a_t(\mathbb{R}) \tag{4.1}$$

The primary objective of the RL control problem is to learn a policy that maximizes the total reward earned by the agent.

While the reward r_t at the end of each control action teaches the agent the immediate desirability of taking a particular action for a given state, it does not necessarily covey any information about the long-term desirability of that action. For many water systems, maximizing short-term rewards will not necessarily lead to the best long-term outcomes. An agent controlling a watershed or stormwater system should have the ability to take individual actions in the context of the entire storm duration. For example, holding water in a detention basin may initially provide high rewards since it reduces downstream flooding, but may lead to upstream flooding if a storm becomes too large. Instead of choosing an action that maximizes the reward r_t at time t, the agent seeks to maximize the expected long-term reward described by state-value v or action-value q.

$$v(s_t) = \mathbb{E}\left[\sum_{k=0}^{\infty}\left[\gamma^k r_{t+k+1}\middle|s_t\right]\right] \tag{4.2}$$

$$q(s_t, a_t) = \mathbb{E}\left[\sum_{k=0}^{\infty}\left[\gamma^k r_{t+k+1}\middle|s_t, a_t\right]\right] \tag{4.3}$$

The state-value provides an estimate for an instantaneous action, as well as potential future rewards that may arise after state s_t, discounted with a factor γ ($0 \leq \gamma \leq 1$). The action-value provides a similar estimate conditioned on taking an action a_t in state s_t. The discount factor γ governs the temporal context of the reward. For example, a γ of 0 forces the agent to maximize the instantaneous reward, while a γ of 1 forces it to equally weigh all the rewards it might receive for present and future outcomes. γ is specific to the system being controlled and can vary based on the control objective [135].

An RL agent can learn to control a system by learning the policy directly [137]. Alternatively, the agent can learn the state-value or action-value estimates and follow a policy that guides it towards the states with

high estimates [135]. Several methods based on dynamic programming [136, 145] and Monte Carlo sampling [135] have been developed to learn the functions that estimate the policy and value functions. While these algorithms were computationally efficient and provided guarantees on the convergence, their application was limited to simple systems whose state action space can be approximated using lookup tables and linear functions [**Mnih2013PlayingLearning**, 135].

Given the scale and the complexity of urban watersheds and stormwater networks, a simple lookup table or a linear function cannot effectively approximate the policy or value functions for each state the agent may encounter while controlling the system. As a simple example, considering just ten valves in a stormwater system and assuming that each valve has ten possible control actions (closed, 10% open, 20% open, …) this gives 10^{10} (10 billion) possible actions that can be taken at any given state, making it computationally impossible to build an explicit lookup table for all possible states. This, however, is where very recent advances in Deep Learning, become important. It has been shown that, for systems with large state-action spaces, such as stormwater systems, these functions can be approximated by a Deep Neural Network [84, 135].

Deep Neural Networks are a class of feed-forward artificial neural networks with large layers of interconnected neurons. This Deeply layered structure permits the network to approximate highly complex functions [55], such as those needed for RL-based control. Each layer in the network generates its output by processing the weighted outputs from the previous layer. This means that each layer's output is more complex and abstract than its previous layer. Given the emergence of cheap and powerful computational hardware over the past decade — in particular graphical processing units (GPUs) and high performance clusters (HPCs) — Deep Neural Networks and their variants have emerged as the state of the art in the approximation of complex functions in large state spaces [72]. This makes them a good candidate for approximating the complex dynamics across stormwater systems. For purposes of this paper, a brief mathematical summary of Deep Neural Networks is provided in SI-B.1.

4.3.2 Deep Q Learning

Deep reinforcement learning agents (Deep RL) use Deep Neural Networks as approximators for value or policy functions to control complex environments. In their relatively recent and seminal Deep Q Network (DQN) paper Mnih et al. (2015) [84] demonstrated the first such algorithm, which used Deep Neural Networks to train an Deep RL agent to play *Atari* video games at a human level. This algorithm identifies the optimal control strategy for achieving an objective by learning a function that estimates the action values or q-values. This function (i.e.

q-function) maps a given state-action pair (s_t, a_t) to the action value estimate.

At the beginning of the control problem, the agent does not know its environment. This is reflected by assigning random q-values for all state-action pairs. Over time, as the agent takes actions, new information obtained from the environment is used to update these initial random estimates. After each action, the reward obtained from the environment is used to incorporate the new knowledge:

$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a q(s_{t+1}, a) - q(s_t, a_t) \right] \tag{4.4}$$

The more actions an agent takes at any given state, the closer it gets to converging to the true action value function [135]. The α (step-size) parameter governs how much weight is placed on the new knowledge [135].

An agent will choose an action that maximizes its long-term reward. This process is known as exploitation since it greedily seeks to maximize a known long-term reward. This may not always be the best choice, however, since taking another action may lead the agent to discover a potentially better action, which it has not yet tried. As such, the agent also needs to explore its environment. This is accomplished by taking a random action periodically, just in case this action leads to better outcomes. In such a formulation, the *exploration vs. exploitation* is addressed via a ϵ-greedy policy, where the agent explores for ϵ percent of time and chooses an action associated with the highest action value for the rest. This gives the final policy for the RL agent:

$$\pi(s_t) = \begin{cases} \text{random } a, & \epsilon \\ \arg\max_a q(s_t, a), & else \end{cases} \tag{4.5}$$

ϵ is often set at a high value (e.g. 50%) at the start of the learning process and gradually reduced to a lower value (e.g. 1%) as the agent identifies a viable control strategy.

While there have been prior attempts to approximate the action value function using Deep Neural Networks, they were met with minimal success since the learning is highly unstable [84]. Mnih et al. (2015) [84] addressed this by introducing a replay buffer and an additional target Neural Network. The replay buffer acts as the RL agent's memory, which records only its most recent experience (e.g. the past 10^3 states transitions and rewards). During the training the RL agent randomly samples data from the replay buffer, computes the neural network's loss and updates its weights using stochastic gradient descent:

$$Loss = ||(r_t + \gamma \max_{a'} q^*(s_{t+1}, a')) - q(s_t, a_t)||^2 \tag{4.6}$$

This random sampling enables the training data to be uncorrelated and has been found to improve the training process. The target neural network q^* has the same network architecture as the main network q, but acts as a moving target to help stabilize the training process by reducing the variance [84]. Unlike the neural network approximating q, whose weights are constantly updated using gradient decent, q^* weights are updated sporadically (e.g. every 10^4 timesteps). For more background information, Mnih et al. (2015) [84] and Lillicrap et al. (2016)[73] provide an in-depth discussion on the importance of replay memory and target neural networks in training Deep RL agents.

4.3.3 Evaluation

Here, we investigate the real-time control of urban stormwater infrastructure using Deep Reinforcement Learning. To begin, we formulate and evaluate reward functions for the control of an individual stormwater basin. We then extend these lessons to the control of a larger, interconnected stormwater network. Given the relatively nascent nature of Deep RL, the need to account for public safety, and the desire to evaluate multiple control scenarios, a real-world evaluation is outside of the scope of this paper. As such, our analysis will be carried out in simulation as a stepping-stone toward real-world deployment in the future. To promote transparency and broader adoption, the entire source code, examples, and implementation details of our implementation are shared freely as an open source package[1].

4.3.4 Study Area

Motivated by a real-world system, we apply RL control to a stormwater system inspired by an urban watershed in Ann Arbor, Michigan, USA (4.2). Our choice to use this watershed is motivated by the fact that it has been retrofitted by our group with wireless sensors and control valves already [6] and will, in the future, serve as a real-world testbed for the ideas proposed in this paper. This headwater catchment features 11 interconnected stormwater basins that handle the runoff generated across $4km^2$ of predominantly urbanized and impervious sub-catchment areas. A Stormwater Management Model (SWMM) of the watershed has been developed and calibrated in prior, peer-reviewed studies [149]. It is assumed that each controlled basin in the system is equipped with a $1m^2$ square gate valve. The valves can be partially opened or closed during the simulation, which represents the action taken by an RL agent. The states of the control problem are given by the water levels and outflows at each controlled location. Given the small size of the study area, as well as the need to constrain this initial study, uniform rainfall across

1 https://github.com/kLabUM/rl-storm-control

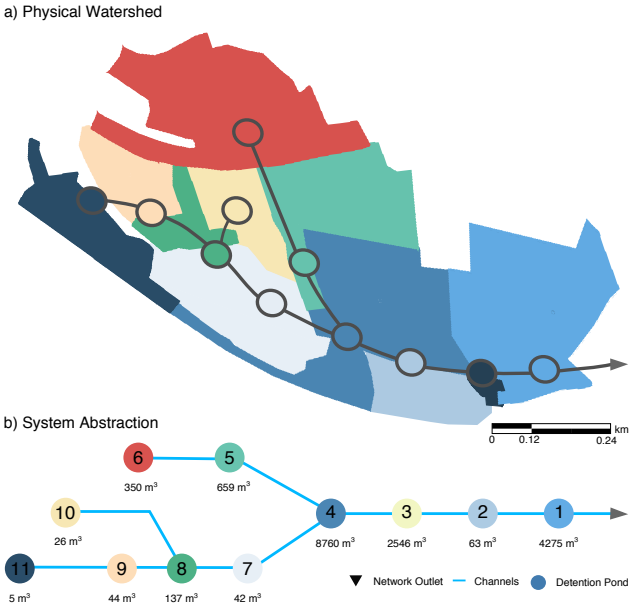

Figure 4.2: Stormwater system being controlled in this paper. The urban watershed includes a number of sub-catchments which drain to 11 stormwater basins of varying storage volumes. The first control scenario applies RL to the control of a single basins, while the second scenario evaluates control of multiple basins. The colors correspond with the catchment that contributes local runoff into each basin. Average volumes experienced by the ponds during a 25 year 6 hour storm event are presented.

the study area is assumed. Groundwater base flow is assumed to be negligible, which has also been confirmed in prior studies [149].

4.3.5 Analysis

Prior Deep RL studies have revealed that performance is dependent on the formulation of reward function, quality of neural networks approximating action value function, as well as the size of state space [53, 135]. This creates a number of "knobs", whose sensitivity must be evaluated before any conclusion can be reached regarding the ability to apply Deep RL to control real stormwater systems. As such, in this paper, we formulate a series of experiments across two scenarios to characterize Deep RL's ability to control stormwater systems. In the first scenario, we control a single valve at the outlet of the watershed, comparing its particular performance under various reward function formulations. Given that Deep RL has not been used to control water systems, this will constrain the size of the state space to establish a baseline assessment of the methodology. In the second scenario, we scale these findings to simultaneously control multiple valves across the broader watershed and to analyze sensitivity to function approximation (neural networks). Finally, the system-scale scenario is subjected to storm inputs of varying intensities and durations to provide broader comparison of the benefits of the controlled system in relation to the uncontrolled system.

4.3.6 Scenario 1: Control of a single basin

In this scenario, we train a Deep RL agent to control the most downstream detention basin in the network (basin 1 in figure 4.2). This basin was chosen because it experiences the total runoff generated in the watershed, and because its actions have direct impact on downstream water bodies. At any given point in time, the RL agent is permitted to set the basin's valve to a position between fully closed or open, in 1% increments (i.e. 0%, 1%, 2%, ..., 100% open) based on the water height in the basin. All other upstream basins remain uncontrolled.

The overall control objective is to keep the water height (state:$\{h_t\}$) in the basin below a flooding threshold $H_{\max}$ and the outflows from the basin (state:$\{f_t\}$) below a desired downstream flooding or stream erosion threshold F:

$$h_t \leq H_{\max} \tag{4.7}$$

$$f_t \leq F \tag{4.8}$$

Three reward functions are formulated to reach this objective, each incorporating more explicit guidance (in the form of constraints) to guide the RL agent.

In the first reward function the RL agent receives a positive reward for maintaining the basin's outflow below the specified threshold, a negative reward for exceeding the threshold, as well as a larger but less likely negative reward if the basin overflows:

$$r_1(s_t) = \begin{cases} +1, & f_t \leq F \\ -1, & f_t > F \\ -10, & h_t > H_{\max} \end{cases} \tag{4.9}$$

The reward function is represented visually in the first row of figure 4.3. This reward function formulation is inspired from the classic inverted pendulum problem [145] where the agent receives +1 for success and -1 for failure.

The second reward function is formulated to exhibit a more complex and gradual reward structure. In lieu of a jagged or discontinuous "plus/minus" reward structure, the agent is rewarded for reaching flows that are close to the desired flow threshold. It has been shown that more smooth and continuous rewards such as this, may help the agent converge onto a solution faster [5, 135]. Visually, the reward function looks like a parabola (figure 4.3), where the maximum reward is achieved when the flow threshold is met exactly:

$$r_2(s_t) = c_1(f_t - c_2)(f_t - c_3) \tag{4.10}$$

c_1, c_2, and c_3 are constants representing the scaling and inflection points of the parabola. Here we choose c_1=-400 e, c_2=0.05, and c_3=0.15 to maintain the general scale of the first reward function. Note that this formulation does not explicitly include the local constraint on the basin's water level since the agent gets implicitly penalized by receiving a negative reward for low outflows.

The third reward function seeks to provide the most explicit guidance to the RL agent by embedding the most relative amount of information (third column, figure 4.3). In this heuristic formulation, the agent receives the highest reward for keeping the basin empty (water levels and flows equal to zero). Intuitively, this reward formulation seeks to drain all of the water from the basin as fast as possible without exceeding the flow and height thresholds. If water level in the pond rises, the agent gets penalized, thus forcing it to release water. If flows remain below the flow threshold F, the agent is penalized linearly proportional to the water level in the basin, with a more severe factor applied if the height

of the basin exceeds the height threshold H. If the outflow exceeds the flow threshold F an even more severe penalty is incurred:

$$r_3(s_t) = \begin{cases} c_1 - c_2 h_t, & h_t < H f_t \leq F \\ c_1 - c_3 h_t, & h_t \geq H f_t \leq F \\ -c_4 f_t - c_2 h_t + c_5, & h_t < H f_t > F \\ -c_4 f_t - c_3 h_t + c_5, & h_t \geq H f_t > F \end{cases} \tag{4.11}$$

The penalty rates are governed by a set of five parameters c=$\{c_1, c_2, c_3, c_4, c_5\}$, which were parametrized $\{2.0, 0.25, 1.5, 10, 3\}$ to match the scales of the other two reward functions.

To illustrate the transferability of the control approach to variable inflows, storage volumes, and the location of a basin in the network, control by an agent trained on the third reward function is evaluated on four basins (basins 1, 4, 6, and 9 in figure 4.2). These basins are chosen to represent distinct components in the network. Basin 1 is located at the outlet of the watershed. Basin 4 is the largest in the network and receives flows from the two major branches in the system. Basin 6 is the largest of the upstream basins, while basin 9 is a smaller basin in series with larger basins.

Additionally, to analyze the performance and sensitivity of the agent to the reward function formulation, two variants of the third reward are evaluated in the supplementary information (please see SI- B.4) section of this paper. The goal of this analysis is to determine the sensitivity of the agent's performance to the choice of mathematical equations in the reward function.

4.3.7 Scenario 2: Controlling multiple basins

This scenario evaluates the ability of an agent to control multiple distributed stormwater basins. Specifically, basins 1, 3, and 4 (figure 4.2) are selected for control because they experience the largest average volume during a storm event, which often corresponds with the larger control potential [122]. It is assumed that at any time step the agent has knowledge of the water levels and valve positions for each of these basins, as well as the basin between them (basin 2 in figure 4.2), thus quadrupling the number of observed states compared to the control of a single basin. The action space must also be reduced to make the problem computationally tractable. For the control of the single basin, there are 101 possible actions at any given time step (valve opening with 1% granularity). For three controlled basins, this increases to 101^3 possible control actions at any given time step. This is not only intractable given our own computational resources, but is well beyond the size of any action space covered in other RL literature. Here, to reduce the action

space the agent is allowed to only throttle the valves. Specifically, at any time step, agent can only open or close the valve in 5% increments or leave its position unchanged. This results in only three possible actions for each site and thus 27 (or 3^3) possible actions for the entire network.

The agent receives an individual reward for controlling each basin. These rewards are weighted equally and added together to provide a total reward for controlling the larger system. The reward for controlling each basin is given by:

$$r_4(s_t) = \begin{cases} -c_1 h_t + c_4, & h_t \leq H, f_t \leq F \\ -c_2 h_t^2 + c_3 + c_4, & h_t > H, f_t \leq F \\ -c_1 h_t + (F - f_t) c_5, & h_t \leq H, f_t > F \\ -c_2 h_t^2 + c_3 + (F - f_t) c_5, & h_t > H, f_t > F \end{cases} \tag{4.12}$$

where reward parameters c=$\{c_1, c_2, c_3, c_4, c_5\}$ are chosen as $\{0.5, 1, 3, 1, 10\}$ to retain the relative scale of the single-basin reward formulations. This reward seeks to accomplish practically identical objectives as the third reward function used in the single-basin control scenario. The difference is the quadratic penalty term that is applied to the height constraint. This modification is made to provide the agent with a more explicit guidance in response to the relatively larger state space compared to the single-basin control scenario. In the rare instance that flooding should occur at one of the basins, agent also receives an additional penalty.

4.3.8 Simulation, Implementation, and Evaluation

Beyond the formulation of the reward function, the use of RL for the control of stormwater systems faces a number of non-trivial implementational challenges. The first relates to the hydrologic and hydraulic simulation framework, which needs to support the integration of a simulation engine that is compatible with modern RL toolchains. The second challenge relates to the implementation of the actual RL toolchain, which must include the Deep Neural Network training algorithms.

Most popular stormwater modeling packages, such as the Stormwater Management Model (SWMM) [115] and MIKE Urban [33] are designed for event based or long-term simulation. Namely, the model is initialized, inputs are selected, and the model run continues until the rainfall terminates or simulation times out. While these packages support some rudimentary controls, the control logic is pre-configured and limited to simple site-scale action, such as opening a valve when level exceed a certain value. The ability to support system-level control logic is limited, let alone the ability to interface with external control algorithms, such as the one proposed in this paper. To that end, we implement a stepwise co-simulation approach that was described in one of our prior studies [88].

Our co-simulation framework separates the hydraulic solver from the control logic by halting the hydraulic model at every time step. The states from the model (water levels, flows, etc.) are then transferred to the external control algorithm, which makes recommendation on which actions to take (valves settings, pump speeds, etc.). Here, we adopt a python based SWMM package for simulating the stormwater network [108]. This allows the entire toolchain to be implemented using a high-level programming environment, without requiring any major modifications to hydraulic solvers that are often implemented in low-level programming languages and difficult to fuse with modern libraries and open source packages. While other or more complex stormwater or hydrologic models could be substituted, model choice is not necessarily the main contribution of this paper. Rather, we content that SWMM adequately captures runoff and flow dynamics for the purposes of this paper. SWMM models the flow of water in the network using an implicit dynamic wave equation solver[115]. This allows it to effectively model the nuanced conditions (e.g. back channel flows, flooding) that might develop in the network though real-time control. Furthermore, the authors have access to a calibrated version of the model for this particular study area, which has been documented in a prior study [18, 149].

One major task is the implementation of the Deep Neural Network that is used to approximate the RL agent's action value function. Deep Neural Networks are computationally expensive to train [72]. Efficient implementation address this by leveraging a computer's graphical processing unit (GPU) to carry out this training, which is a non-trivial task. To that end, a number of open source and community libraries have emerged, the most popular of which is *TensorFlow* [1]. This state-of-the-art library has been used in some of the most well-cited RL papers and benchmark problems, which is the reason we choose to adopt it for this study. *TensorFlow* is a python library and can be seamlessly interfaced with our python-based stormwater model implementation.

Multiple agents are trained and evaluated across the two scenarios: eight for the control of individual basins (across multiple reward function variants and basins), and two agents for the multi-basin control. A Deep Neural Network is designed and implemented to learn the action-value function of each agent. The network contains 2 layers with 50 neurons per layer. This network is set up with a *ReLu* activation function [48] in the internal layers and a linear activation function in the final layer. The full parameters used in the study, including those for gradient descent and the DQN, are provided in the SI-B.2 of this paper. A Root Mean Square Propagation (RMSprop) [48] form of stochastic gradient descent is used for updating the neural network as this variant of gradient descent has been observed to improve convergence.

One storm event is used to train these agents. The SWMM model is forced with a 25-year storm event of 6 *hour* duration and 63.5 *mm*

intensity (figure 4.3). This event generates a total runoff of 3670.639 m^3 with a peak flow of 0.35 m^3/s at the outlet of watershed. The agents are provided with an operational water level goal H of 2 m, flooding level $H_{\max}$ of 3.5 m and outflow exceedance threshold of F of 0.10 $m^3 s^{-1}$. It is important to note that the outflow threshold, in particular, serves more as an approximate guide rather than exact requirement, since the discrete valve settings used by the RL agents may not allow the exact setpoint to be physically realizable (e.g. throttling a valve by 5% will limit outflow precision correspondingly). These setpoints are chosen to reflect realistic flooding and stream erosion goals in the study watershed. Agents are trained on a Tesla K20 GPUs on University of Michigan's advanced research computing's high performance cluster.

The second multi-basin control agent uses the same neural network architecture of the other multi-basin RL control agent, trained this time, however, using *batch normalization* [56]. Batch normalization is the process of normalizing the signals between the internal layers of the neural network to minimize the internal covariance shift and has been observed to improve the performance of the Deep RL agents [73]. Ioffe et al. (2015) [56] provides a detailed discussion on batch normalization.

The performance of each agent is evaluated by comparing the RL controlled hydrographs and water levels to those that are specified in the reward functions. For the agents controlling the individual basins, this is used to determine the importance of the reward formulation on performance, reward convergence, and training period duration. For the multi-basin control scenario, the same approach is used to quantify overall performance, comparing this time the agent that uses the batch normalized neural network to the agent that uses the non-normalized network.

To evaluate the ability of an RL-agent to control storms that it is not trained on, a final analysis is carried out. Since the agent controlling multiple basins presents the most complex of the scenarios, it is first trained on one of storms and evaluated on a spectrum of storm events with varying return periods (1 to 100 years) and durations (5 min to 24 hours). These storm events are generated based on the SCS type II curve and historical rainfall intensities for the study region [125]. The performance of the agent across these 70 storms is compared to the uncontrolled system to evaluate the boarder benefits of real-time control. For comparison with an other control algorithm, we also implement and compare the performance of RL to an *Equal Filling Degree* controller, which seeks to control the volume in each basin to achieve equal relative filling [124]. Implementation details of the equal filling algorithm can be found in the SI-B.3. We also evaluate the performance of the RL controller on a back-to-back storm event (3 hour 5 year event, followed by a 2 hour 2 year event). To allow for a comparison between the controlled and uncontrolled system, a non-negative performance metric is introduced to capture the magnitude and time that the system

deviates from desired water level and flows thresholds. Specifically, across a duration T the final performance P adds together the deviation of all N controlled sites from their desired water level (P_h) and flow thresholds (P_f), where:

$$P_h(h) = \begin{cases} h - H, & h > H \\ h - H + 100h, & h > H_{\max} \\ 0, & otherwise \end{cases} \tag{4.13}$$

$$P_f(f) = \begin{cases} 10(f - F), & f > F \\ 0, & otherwise \end{cases} \tag{4.14}$$

$$P = \sum_{n=1}^{N} \sum_{i=0}^{T} P_h(h_i^n) + P_f(f_i^n) \tag{4.15}$$

A relatively lower performance value is more desirable, since it implies that the system is not flooding, nor exceeding desired flow thresholds.

4.4 RESULTS

4.4.1 Scenario 1: Control of single basin

The ability of an RL agent to control a stormwater basin is highly sensitive to the reward function formulation. Generally, a more complex reward function — one that embeds more information and explicit guidance — performs better, as illustrated in figure 4.3. Each column of the figure corresponds with an individual RL agent, each of which is trained using a different reward function (r_1, r_2, r_3). The reward functions are plotted in the first row, while the reward received during training is plotted in the second row. The training period is quantified in terms of episodes, each of which corresponds to one full SWMM simulation across an entire storm. The third and fourth rows in the figure compare the uncontrolled flows and water levels, respectively, for the episode that resulted in the highest reward.

The RL agent that uses the simplest reward function has the relatively worst performance (figure 4.3, first column). Even after 5000 training episodes (a week of real-world simulation time), the mean reward does not converge to a stable value. Playing back the episode that resulted in the highest reward (figure 4.3, rows 3–4, column 1), reveals that the RL agent does retain more water than would have been held in the uncontrolled basin. While this lowers the peak flows relative to the uncontrolled basin, the RL agent is generally not able to keep flows below the desired threshold. More importantly, the RL agent's control actions begin oscillating and become unstable toward the middle of the

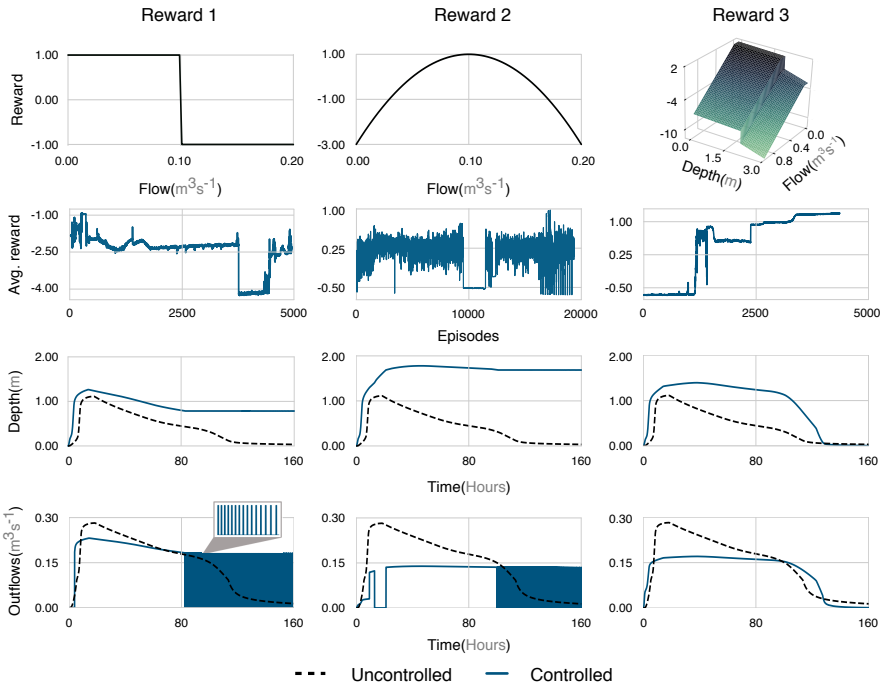

Figure 4.3: RL control of a single basin, trained using three reward formations (grouped by column). The first row plots each reward function used during training. The second row plots the average reward received during training (please note that the scale of Y-axis differers for each reward function). The third and fourth rows compare the controlled flows and water levels with the uncontrolled, for the episode that resulted in the highest reward. Generally, reward function formulations with more explicit guidance lead to relatively better control performance and improved convergence during raining. Agents trained using relatively simple reward also exhibit a rapidly-changing and unstable control behavior, shown as a close up in the bottom left plot.

simulation. In this episode, the agent keeps the water level in the basin relatively constant by opening the valve very briefly to release just a small amount of water. This "chattering" behavior (shown as a close up in the figure) results in an unstable outflow pattern that oscillates in a step-wise fashion between 0 m^3/s and 0.18 m^3/s. For various practical reasons, such rapid control actions are not desirable. Since the RL agent never once receives a positive reward, it may have converged onto an undesirable local minimum during the training. Providing more time for training does not appear to resolve this issue, which may also suggest that a stable solution cannot be derived using this particular reward formulation.

Embedding more explicit guidance (harder constraints) into the reward formulation improves the control performance of the RL agent (figure 4.3, second column). When the second and more continuous reward function is used by the agent, the highest reward episode reveals that the RL agent is relatively more effective at maintaining flows at a constant value. Unlike the RL agent using the simple step-wise reward function, the RL agent using the parabolic reward function has more opportunities to receive smaller, more gradual rewards. During most of the episode, this increased flexibility allows the second RL agent to receive positive rewards and keep the outflow below a flow threshold of 0.14 m^3/s. While relatively improved, the RL agent using the parabolic reward also does not converge to a stable reward value. However, toward the end of the episode, this RL agent also carries out irregular and sudden control actions by opening and closing the valve in short bursts. In this case, the RL agent is oscillating between a maximum (valve open) and minimum (valve closed) reward rather than taking advantage of variable rewards in other configurations. This suggests that the agent has either not yet learned a better strategy or, again, that a stable solution cannot be converged upon using this particular reward formulation. This speaks to the need for more explicit constraints as well, since a real-world stormwater system could not be throttled in this fashion. Simply put, the reward formulations used in this case was too simple to achieve realistically desirable outcomes.

The RL agent using the third and most constrained formulation exhibits the relatively best control performance. This agent regulates flow and water levels in a relatively gradual and smooth manner. Unlike in the case of the other two RL agents, after 3500 training episodes, the third agent does converge to a steady reward. Evaluating the episode resulting in the highest reward (figure 4.3, rows 3–4, column 3), the desired "flat" outflow hydrograph is achieved. No unstable or oscillatory control actions are evident, as in the case of the other two reward functions. The agent is able to maintain flows below a constant threshold of 0.15 m^3/s. While this is not the exact threshold that was specified (0.1 m^3/s), it is close considering that the achievable threshold is dependent on water levels and the ability to only throttle the valve in

1% increments. As stated in the methods section, matching the exact threshold may not be physically realizable in any given situation due to constraints enforced by discretized throttling. Furthermore, the RL agent must balance the desired outflow against the possibility of flooding and is thus more likely to release a greater amount of water than is specified by the threshold. Interestingly, this agent does not change its valve configuration at all. Rather, it keeps its valve 54% open the entire time of the simulation, which allows it to meet a mostly constant outflow given the specific inflows. Overall, the general shape of the outflows is improved compared to the uncontrolled scenario. Furthermore, an added benefit of real-time control is that the overall volume of water leaving the basin is also reduced by 50% due to infiltration.

Similar to the third reward function, agents trained on the 3a and 3b reward functions are successfully able to maintain the outflows close to the threshold during the stormevent (figure B.3 in SI B.4). While these reward functions may appear similar, the solution identified by their respective agents differs. This is a result of the difference between the decay rates in the exponential and squared terms. Performance of the agent trained on the 3a and 3b reward functions (SI B.4) indicates that the ability of the agent to identify a viable control strategy is not dependent on the choice of equations used for the creating the reward functions, but rather on the general shape of the reward function in the domain.

The agent using the third reward function (trained on basin 1), is able to control basins 4,6 and 9 without any further modifications (SI B.5, figure B.4). The agent in this formulation makes its control decisions only based on the depth at the current time step and does not incorporate any predictions. Hence, the ability of the controller to shape of outflows should not dependent on the location of the basin in the network, magnitude of inflows or the storage curves. Our simulation results indicate the same. Though the degree to which the agent is able to achieve the objective is governed by these physical constraints, its ability to discover a solution is not influenced by them.

This scenario, which focuses on the control of a single site, emphasizes the importance of the reward function formulation in RL-based control of stormwater systems. The complexity of the reward formulation plays an important role in allowing the RL agent to learn a control policy to meet the desired hydrologic outcomes. The importance of reward formulations has been acknowledged in prior studies[91, 135]. Generally, reward functions with more explicit guidance lead to a more rapid convergence of a control policy, while avoiding unintended control actions, such as the chattering behavior seen in figure 4.3. In fact, prior studies have attributed such erratic control actions to the use of oversimplified reward functions [91], but have offered little specificity or concrete design recommendations that could be used to avoid such actions. As such, our approach heuristically evaluates reward formula-

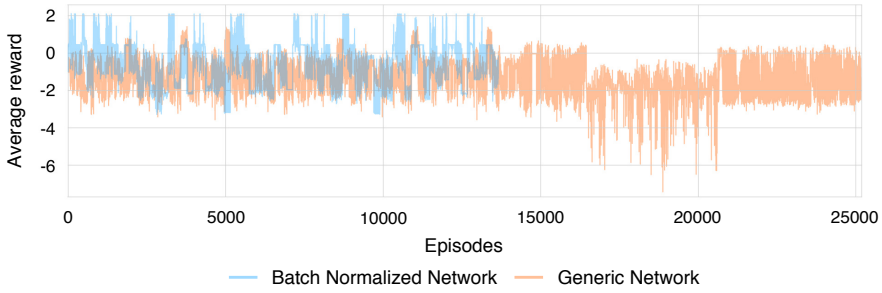

Figure 4.4: Average reward earned by the RL agent when learning to control multiple basins. The use of neural network batch normalization (blue) leads to consistently higher rewards when compared to the use of a generic neural network (orange). The batch normalized network also leads to higher rewards earlier in the training process.

tions of increasing complexity until arriving at one that mostly meets desired outcomes. This introduces an element of design into the use of RL for the real-time control of stormwater, as one cannot simply rely on the implicit black box nature of neural networks to solve a control problem under complex system dynamics. The reward function needs to embed enough information to help guide the RL agent to a stable solution. This introduces only a limited amount of overhead, as reward functions can be intuitively formulated by someone with knowledge of basic hydrology.

For control of individual basins, the reward function presented here should be directly transferable. If more complex outcomes are desired, modifications to the reward function may need to be carried out. Objectively, the convergence of the reward will serve as one quality measure of control performance. The ultimate performance of RL for the control of individual sites will, however, need to be assessed on a case-by-case basis by a designer familiar with the application. Taking the baseline lessons learned during the control of a single basin, the second scenario can now evaluate the simultaneous control of multiple basins.

4.4.2 Scenario 2: Control of multiple basins

When trained using the generic feed forward neural network configuration that was used for the control of a single basin, the RL agent controlling multiple assets was unable to converge to a stable reward, even after 25000 episodes of training (figure 4.4). This totaled to ≈52 days of computation time on our GPU cluster, after which the training procedure was halted due to lack of improved reward. Overall, learning performance was low in this configuration. Not only did the learning procedure not converge to a stable reward, but the vast majority of rewards were negative. Given this observation, this ineffective neural network was then replaced with one that was batch normalized. The agent using the batch normalized neural network achieved a higher

average reward than the agent with a generic feed forward neural network (figure 4.4). Furthermore, the agent using the batch normalized neural network achieved a relatively high rewards early on in the training process, thus making it more computationally favorable. While beyond the scope of this study, this suggests that the choice of neural network architecture is likely a major design factor in the successful implementation of RL-based stormwater control.

Even with batch normalization, the RL agent did not consistently return to the same reward or improve its performance when perturbed. The exploration in its policy caused the RL agent to oscillate between local reward maxima. Similar outcomes have been observed in a number of RL benchmark problems [53, 84], which exhibited a high degree of sensitivity to their exploration policy. Prior studies have noted that the exploration-exploitation balance is difficult to parameterize because neural networks tend to latch onto a local optimum [70]. As such, it is likely that the lack of convergence observed in this scenario was caused by the use of a neural network as a function approximator. Forcing neural networks to escape local minima is still an ongoing problem of research [97]. Nonetheless, even without a consistent optimum, the maximum reward obtained during this scenario can still be used as part of an effective control approach.

Selecting the episode with the highest reward revealed the actions taken by the RL agent during the training storm (figure 4.5). The figure compares the controlled and uncontrolled states of the four basins during a 25-year 6-hour storm event, showing the depth in each basin, inflows, outflows, and control actions taken by the RL agent. Though basin 2 is not explicitly controlled by the controller, given that the water level and outflows in this basin are impacted by the actions taken in the upstream basin, we have chosen to present its response. No flooding occurred during this simulation, which means that the reward received by the RL agent was entirely obtained by meeting outflow objectives. The valves on basins 1 and 3 throttled between 100% and 95% open, which for all practical considerations could be considered uncontrolled. As such, the RL agent in this scenario earned its reward by only controlling the most upstream basin in this network.

While the outcome of control was somewhat favorable compared to the uncontrolled systems, the playback of the highest reward in figure 4.5 does not show drastically different outcomes. Control of the 4^{th} basin shifted the timing of the outflows from the basin but did not reduce its outflows. This resulted in improvements at the 1^{st}, 2^{nd} and 3^{rd} basins. By delaying flows from the 4^{th} basin, the RL agent allowed the downstream basins to drain first and to spend less time exceeding the flow threshold. Interestingly, the RL agent did not control basin 1, even while the single-basin control scenario makes it is clear that a more favorable outcome can be achieved with control (figure 4.3). As such, a better control solution may exist, but converging to such a

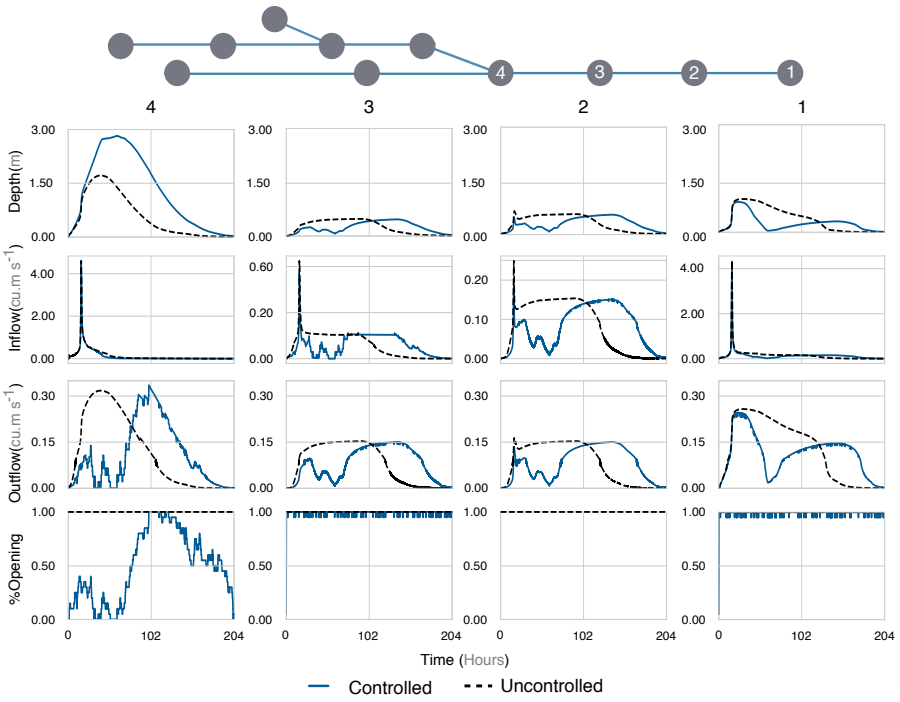

Figure 4.5: RL agent controlling multiple stormwater basins during a 6-hour, 25-year storm event. Control actions at each of the controlled basins are shown as valve settings in the fourth row of the plot. In this scenario, the agent achieves a high reward by just controlling the most upstream control asset (4) and shifting the peak of the hydrograph. Difference in the scale of Y-axis in second row demonstrates the wide range of inflows in the network.

solution using a neural network approximator is difficult. This likely has to do with the larger state action space. While the site-scale RL agent was only observing water level at one basin, the system level RL agent had to track levels and flows across more basins, which increases the complexity of the learning problem. The rewards received by the RL agent in the scenario are cumulative, which means that improvement at just a few sites can lead to better rewards, without the need to control all of them. Increasing the opportunity to obtain rewards thus increases the occurrence of local minima during the learning phase.

In the single basin control scenario, the RL agent can immediately observe the impact of its control actions. In the system scale scenario more time is needed to observe water flows through the broader system, which means that the impact of a control action may not be observed until later timesteps. This introduces a challenge, as the RL agent has to learn the temporal dynamics of the system. This challenge has been observed in other RL studies, which have shown better performance for reactive RL problems, as opposed to those that are based on the need to plan for future outcomes [5]. The need to include planning is still an active area of RL research. Potential emerging solutions include adversarial play [130, 131], model-based RL [28], and policy-based learning [121]. The benefits of these approaches have recently been demonstrated for other application domains and should be considered in the future for the control of water systems.

It is important to note that figure 4.5 represents an evaluation of the RL agent for one storm only – namely, the training storm. Realistically, the control system will need to respond to storms of varying durations and magnitudes. As an example, the RL agent's response to a 24-hour, 10-year storm is shown in figure 4.6. Performance of the controller in controlling a back-to-back event is presented in SI-B.6. Here, the RL agent outperformed the uncontrolled system much more notably compared to the training storm. The controlled outflows were much closer to the desired threshold, even when only one basin was controlled. This broader performance is captured in figure 4.7, which quantifies performance (eq. 4.15) across a spectrum of storm inputs. Figure 4.7 compares the uncontrolled system to the RL-controlled system. Both the controlled and uncontrolled systems perform equally well during small-magnitude and short events (e.g. the training storm in figure 4.5). The benefits of control become more pronounced for larger events, starting at 10-year storms and those that last over 2 hours. This visualization holistically captures the benefits of real-time control by highlighting new regions of performance and showing how control can push existing infrastructure to perform beyond its original design.

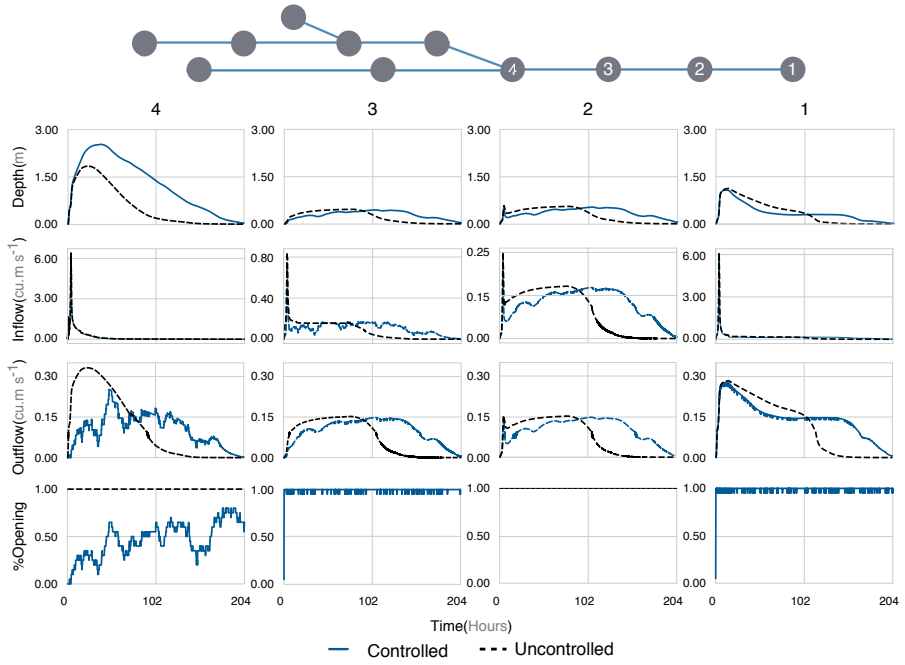

Figure 4.6: RL agent controlling multiple stormwater basins during a 24-hour, 10-year storm event. Control actions at each of the controlled basins are shown as valve settings in the fourth row of the plot. In this scenario, the agent achieves a high reward, by maximizing the storage utilization in the most upstream control asset (4) and regulating the outflow from it to meet the downstream objectives. Difference in the scale of Y-axis in second row demonstrates the wide range of inflows in the network.

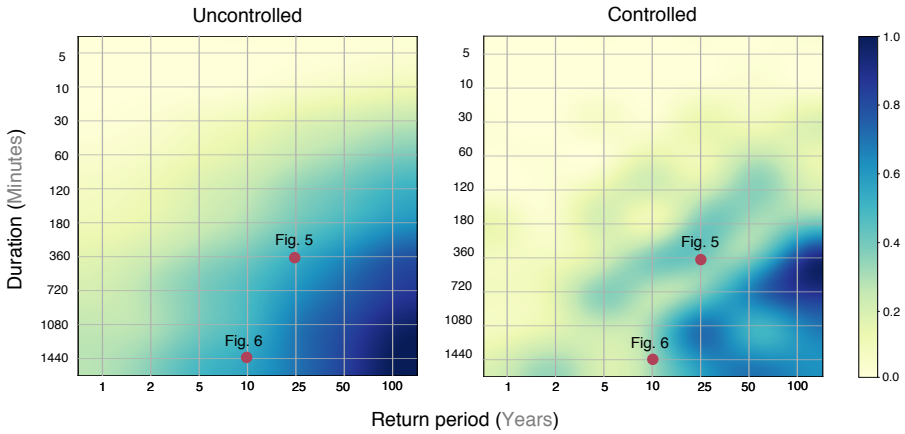

Figure 4.7: Normalized performance of stormwater system (eq. 4.15) for the uncontrolled system (left) and RL-controlled system (right). The use of control enhances the performance stormwater network by allowing the system to achieve desired outcomes across larger and longer storms. The lighter color (closer to zero) corresponds with a relatively better performance.

4.5 DISCUSSION

Given the recent emergence and popularity of Reinforcement Learning, much research still remains to be conducted to evaluate its potential to serve as a viable methodology for the RTC of water systems. Our study brings to light a number of benefits and challenges associated with this task. Arguably, it seems that the major benefit of using RL to control water systems is the ability to simply hand the learning problem to a computer without needing to worry about the many complexities, non-linearities and formulations that often complicate other control approaches. However, as this study showed, this comes with a number of considerable caveats. These include the challenges associated with formulating rewards, choosing function approximators, deciding on the complexity of the control problem, as well as contending with practical implementation details.

Our study confirms that the performance of RL-based stormwater control is sensitive to the formulation of the reward function, which has also been observed in other application domains [91]. The formulation of the reward function requires domain expertise and an element of subjectivity, since the RL agent has to be given guidance on what constitutes appropriate actions. In the first scenario, it was shown that a reward function that is too simple may lead to adverse behavior, such as the chattering or sudden actions. The reward may also not converge to a stable solution since the neural network can take advantage of the simple objective to maximize rewards using sudden or unintuitive actions. The formulation of the problem, which depends heavily on neural networks, also makes it difficult to determine why one specific reward function may work better than another. Increasing the complexity of the reward function, by incorporating more explicit guidance, was shown to help guide the RL agent to a more desirable outcome. In other control approaches, such as genetic algorithms or model predictive control, the design of reward is an iterative process, and sometimes involves anticipating fringe cases to improve the robustness of the controller. Similar to these approaches, we can however begin using this early study to formulate a number of practical considerations when formulating reward functions:

- Define the reward function for entire domain of the state-action space, ensuring that it distinguishes the desirable actions from the undesirable ones.
- Ensure that the reward function represents a specific hydrologic response that the controller is to achieve, while anticipating, as much as possible, alternate and adverse hydrologic responses that the controller may discover to maximize the reward function.

- Relax the mathematical formulation of the reward function and focus rather on the two above points (e.g. the shape of a reward surface rather than its specific mathematical form).

Reward formulations are an ongoing research area in the RL community and some formal methods have recently been proposed to provide a more rigorous framework for reward synthesis [39]. These formulations should be investigated in the future.

Even when the choice of reward function is appropriate or justifiable, the control performance can become sensitive to the approximation function, which in our case took the form of a Deep Neural Network. Choosing the architecture and structure of the underlying network becomes an application dependent task and can often only be derived through trial and error [53, 135]. Secondly, for challenging control problems, such as the one studied here, learning the mapping between rewards and all possible control decisions becomes a complex task. The neural network must be exposed to as many inputs and outputs as possible, which is computationally demanding. In our study we ran simulations for many real-world months on a high performance cluster, but it appears that the learning phase could have continued even longer. This, in fact, has been the approach of many successful studies in the RL community, where the number of computers and graphical processing units can be in the hundreds [36, 96]. This was not feasible given our own resources, but could be evaluated in the future.

Aside from the formulation of the learning functions and framework, the actual complexity and objectives of the control problem may pose a barrier to implementation. We showed that an RL agent can learn how to control a single stormwater basin effectively, but that controlling many sites at the same time is difficult. A major reason is the increase in the number of states and actions that must be represented using the neural network. While computational time may remedy this concern, the structure of the neural network may also need to be altered. In a system-scale stormwater scenario, actions at one location may influence another location at a later time. As such, the agent would benefit from a planning-based approach which considered not only current states, but future forecasts as well. Such planning-based approaches have been proposed in the RL literature and should be investigated to determine if they lead to an improvement in performance [28, 30]. Furthermore, model-based approaches have also recently been introduced and could allow some elements of the neural network to be replaced with an actual physical or numerical stormwater model [50]. Such approaches should be evaluated in the future since they may permit more domain knowledge from water resources to be embedded into training the controller.

It is important to note that the Equal-filling algorithm outperforms the RL agent in this study (SI-B.3). It achieves the objective of maintaining the outflow below the desired threshold without causing flooding. Since Equal-filling outperforms RL, it could very well be considered a superior choice in this study. That said, developing and deploying Equal-filling often requires an intuitive understanding of the system and require a highly manual tuning of parameters. While it may be relatively straightforward to design control approaches in smaller systems and simple outcomes —such as the one in this study — developing coordinated control strategies for large scale systems with multiple-objective might not be as easy. As such, we see RL-based control as a long-term goal, which should be investigated in future studies across bigger scales and complex outcomes. Our study presents an initial goal toward the broader study of RL-based stormwater control, after which an comprehensive apples-to-apples comparison may be possible with current state-of-the-art approaches.

Finally, the use of RL for the control of stormwater systems is underpinned by a number of practical challenges. Computational demands are very high, especially compared to competing approaches, such as dynamical systems control, model predictive control, or market-based controls. While computational resources are becoming cheaper, the resources require to carry out this study were quite significant and time demanding. Since actions taken by neural networks cannot easily be explained and explicit guarantees cannot be provided, this may limit adoption by decision makers who may consider the approach a "black box". It is also unlikely that the control of real-world stormwater systems will simply be handed over to a computer that learns through mistakes. Rather, simulation-based scenarios will be required first. It has recently been shown as long as a realistic simulator is used — in our case SWMM — then the agent can be effectively trained in a virtual environment before refining its strategy in the real world [96].

4.6 CONCLUSION

This paper introduced an algorithm for the real-time control of urban drainage systems based on Reinforcement Learning (RL). While RL has been used successfully in the computer science communities, to our knowledge this is the first instance for which it has been explicitly adopted for the real-time control of urban water systems. The methodology and our implementation show promise for using RL as an automated tool-chain to learn control rules for simple storage assets, such as individual storage basin. However, the use of RL for more complex system topologies faces a number of challenges, as laid out in the discussion. Simultaneously controlling multiple distributed stormwater

assets across large urban areas is a non-trivial problem, regardless of the control methodology. To that end, the concepts, initial results and formulations provided by this paper should help build a foundation to support RL as a viable option for stormwater control. The source code accompanying this paper should also allow others to evaluate many other possible architectures and parameterizations that could be used to improve the results presented in the paper.

5 BAYESIAN OPTIMIZATION FOR SHAPING THE RESPONSE OF THE STORMWATER SYSTEMS

6 PYSTORMS: A SIMULATION SANDBOX FOR THE DEVELOPMENT AND EVALUATION OF STORMWATER CONTROL ALGORITHMS

This chapter is written with collaboration with Sara. P. Rimer and Sara. C. Troutman.

The advent of smart cities is poised to transform the management of our built environment [25, 52]. Specific to stormwater, a new generation of smart and connected stormwater systems promises to reduce flooding and improve water quality management by autonomously sensing watershed parameters and subsequently controlling corresponding hydraulic components across complete watersheds, both *adaptively* and *in real-time*. These smart systems will provide an alternative to costly concrete-and-steel construction by squeezing even more performance out of existing stormwater and sewer infrastructure, and reimagining the design and operation of new infrastructure. While the idea of controlling distributed stormwater systems in real-time dates back to the 1970s [140], the concept has only recently gained widespread traction in large part due to the affordability of internet-connected sensors, the increased capacity of data services, and the broader acceptance and popularity of other autonomous systems (e.g. self-driving cars and robots). Relative to other fields of autonomy, however, smart water systems are still early in their stage of adoption. Thus, developing and implementing smart water systems presents an exciting opportunity for researchers and practitioners alike to propose new visions, standards, and technologies.

The intelligence of smart stormwater systems broadly refers to the acquisition (i.e. "sensing") and processing of data into decisions and actions (i.e. "control strategies") that are then used to guide the operation of gates, valves, pumps, and other actuators within a water system. Ultimately, the logic embedded via these control rules determines how water is moved around the collection system to meet specific performance objectives or reduce adverse outcomes (e.g. flooding, overflows, and/or water quality impairments). As such, the emerging field of smart stormwater systems stands to benefit greatly from researchers and stakeholders who can bring to bear new control strategies and techniques.

However, due to the complex, bureaucratic nature of watershed management, it can be impenetrable for new groups working in this field to obtain the necessary details of how real-world stormwater systems operate, as those details are unlikely to be opened up to just anyone who wants to try out new ideas of controlling them. To that end, computational toolchains exist for simulating stormwater systems and then evaluating various control rules implemented by them. Yet, developing these simulations and adapting them to specific control strategies often requires a significant amount of effort and expertise. Furthermore, while a number of promising control algorithms have been proposed, they have all been evaluated on highly specific examples and simulators, making it difficult to establish cross-comparisons of their performance. In an effort to address these limitations, the contribution of this paper is `pystorms`, an open-source Python package comprised of:

1. A collection of real world-inspired smart stormwater control scenarios that facilitate the quantitative evaluation of control strategies, coupled with
2. A programming interface and a stormwater simulator to provide a stand alone package for developing stormwater control strategies.

Our aspiration is for `pystorms` to emerge as a community-driven resource that fosters accessibility and collaboration amongst smart stormwater control's field of researchers and practitioners, both novices and experts alike.

6.1 BACKGROUND

6.1.1 Control of Stormwater Systems

A stormwater control problem can be defined as the development of an infrastructural strategy to manipulate the behavior of stormwater in order to achieve a desired response. Traditionally, stormwater control has relied on *passive* solutions, in which control strategies are large-scale, construction-heavy, and often financially-burdensome. However, the emergence of microcontrollers, wireless communication technologies, and low-cost sensors has allowed for small-scale, modular, and automated control components (e.g. hydraulic valve operated by cellularly-connected actuator) to be installed at strategic locations throughout a stormwater network for *active* control, with decisions that can be automated, adjusted remotely, and made in real-time. Consequently, stormwater infrastructure can now be instantly redesigned to respond to its dynamic environment.

Although implemented smart stormwater control engineering solutions were documented at least a decade earlier, research-oriented

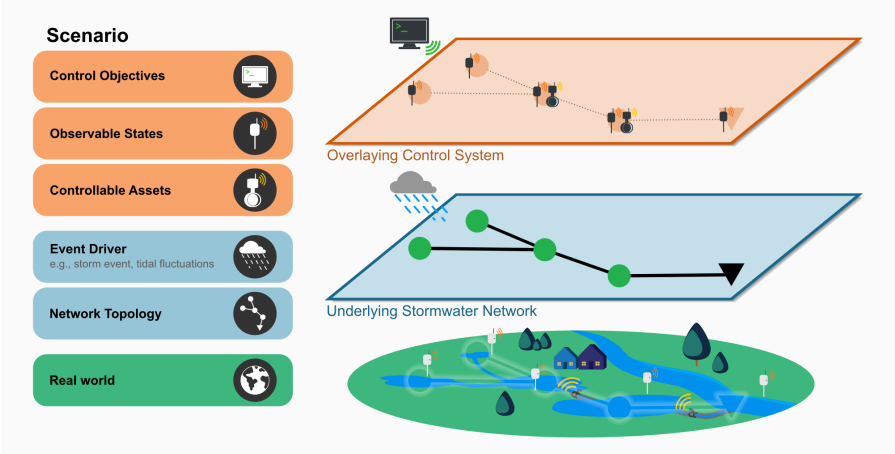

Figure 6.1: `pystorms` abstracts the control of stormwater systems as scenarios, which are characterized by a computational representation of a stormwater network, a corresponding event driver, set of observable states, and controllable assets that can be leveraged to manipulate the behavior of a stormwater network in real-time to achieve control objectives. This is coupled with a streamlined programming interface and a stormwater simulator to provide the users with a standalone package for the development and evaluation of control algorithms.

discussion of these implementations did not occur until 1989 [119]. Furthermore, while implementation of smart stormwater control began at the end of the 20th-century, the 21st-century has seen far more extensive systematic successes, as seen beginning with the foundational reviews of **Schutze2004** and Vanrolleghem, Benedetti, and Meirlaen [143]. Some notable adaptive real-time stormwater control implementations that have been installed hand-in-hand with extensive research dissemination include **Gaborit2013**, **Gaborit2016**, García et al. [43], Montestruque [86], Mullapudi, Wong, and Kerkez [88], Ocampo-Martinez [94], Sadler et al. [116], and Vezzaro and Grum [144]. These references we specifically emphasize as to us they represent diverse and well-documented implementations of smart stormwater control from single control assets to watershed-scale implementations. For more comprehensive reviews of stormwater control implementations, we direct the reader to some recently published survey articles on the topic [29, 43, 75, 129, 155].

6.1.1.1 Simulating Stormwater Systems

Due to the variability of storm events and the safety concerns of experimental uncertainty, it is infeasible to test various control strategies on *actual* stormwater networks. Thus, a more practical method to test the outcomes of different control decisions is to use a computational simulation of a stormwater network that is able to give us a "good enough" estimate of what the actual in-situ physical results might be. Often, these simulations can be carried out using computational stormwater

models that have already been developed to inform the design and operation of the stormwater system being studied.

As a stormwater system is designed to route rainfall and other runoff to a wastewater treatment plant and/or discharge to a receiving body of water, the computational components of stormwater models primarily include a (i) runoff module and (ii) a routing module, and are driven by (iii) precipitation events (e.g. rain, snow). The runoff module converts precipitation into overland runoff; the overland runoff then undergoes hydrological processes (e.g. infiltration, evaporation) and is hydraulically transported to the stormwater collection system, which is carried out computationally via the routing module.

Over the years, several different software applications have been developed for modeling and simulating stormwater networks. The different software applications all function in a similar manner in which they computationally estimate the dynamics of stormwater as it moves through predefined temporal and spatial bounds to varying degrees of mathematical accuracy and fidelity to the underlying hydraulic and hydrological governing processes. The US-EPA's Stormwater Management Model (SWMM) [111], MIKE URBAN+ from the MIKE Powered by DHI software suite[1], and the Model for Urban Stormwater Improvement Conceptualisation (MUSIC) by eWater[2] are a few examples of widely-used stormwater software applications. Furthermore, in addition to modeling runoff and its routing, some of these software applications have also been developed with the capabilities of modeling urban flooding (e.g. MIKE FLOOD) as well as the generation and transport of pollutants (e.g. SWMM). The computational details underlying the models produced by these software applications are not the purpose of this paper, and instead for clarity and further detail, the reader is directed to Rossman and Huber [113], Rossman [112], and Rossman and Huber [114]. Additionally, we provide further detail about the hydraulic simulator we utilize in Section 6.2.3.

6.1.1.2 *Implementing Control*

For those unfamiliar with control theory and control systems engineering, the field can be evasive. Here, we aim to maintain the idea of control in its broadest but most straightforward meaning: after receiving some sort of *cue* (for our systems, this cue may be readings from sensors), an *action* is taken on the system to alter it for a desired outcome (an action for stormwater systems may be as simple as the opening and closing of a valve). When we implement *control*, we are making decisions and taking actions that try to optimize our system in order to meet some specified

1 mikepoweredbydhi.com
2 ewater.org.au

objective. Thus, a stormwater control strategy can be simplified as the method of developing rules that determine actions to be implemented by the stormwater system. The computational process of implementing this method to find these actions is what we define as the *stormwater control algorithm*.

It is easy to imagine how finding the "best" actions to implement is a complex undertaking. Suppose a stormwater system with only one valve was installed, and that valve could be either completely opened or closed every hour. Deciding on a pattern for the complete opening and closing of the valve — even over the period of a few days — in order to meet some sort of objective is actually quite difficult, with no guarantee of a singularly "correct" solution. Now, imagine if the action could be to open the valve as a percentage between 0–100% — the combination of actions that can be implemented becomes even more endless. From an initial perspective, this process of finding the "best" and "correct" solution from what is an inexhaustible set of possibilities might seem futile. However, for stormwater applications, finding such an absolute "best" solution is usually not necessary, and most likely does not even exist. Instead, a solution that provides a "better" outcome than the current one is often sufficient and can still drastically benefit the system. Additionally, such better solutions are actually dependent on how a system's underlying needs are even defined. That is, *how we define the objective determines what is considered optimal*. Thus, research on stormwater control focuses on both (i) the formulation of control objectives and (ii) analyzing and differentiating the myriad potential solutions based on said formulations. While the former research is essential in this field, it is not the focus of this paper. Rather, our initiative here concentrates on the latter: via the presented simulation sandbox `pystorms`, we aim to develop a systematic means for analyzing and differentiating control solutions for stormwater systems with pre-defined objectives.

Thus far, we have discussed "smart stormwater control" in its broadest sense, which encompasses all layers that a smart stormwater control system would entail — from the sensors chosen, the communication protocol implemented for those sensors, the data management of what is sensed, the wireless actuators controlling a control asset, even to the human operators who may interact or intervene physically with the system. However, from here on out, we focus our discussion of smart stormwater control strategies strictly on *the computational algorithm that is used to determine the discrete actions to be taken by the system's control assets based on information known regarding the system's past, current, and/or future state*. This algorithm can allow for the control strategy to be implemented and adjusted over any number of given time periods, and can be coordinated amongst any number of control assets within the system. By focusing strictly on the computational algorithm, we are able

to isolate one component of smart stormwater systems that can allow for *quantitative* cross-comparisons of strategies used across a multitude of stormwater systems. Additionally, the focus on the computational algorithm also centers the component of a smart stormwater system that can specifically benefit from experts outside the discipline of water resources engineering.

6.1.2 The Need for a Simulation Sandbox

Even though smart stormwater control has been successfully implemented for decades, there still does not exist a standard to systematically evaluate the performance of different control strategies across diverse stormwater networks and contexts. Consequently, this inability to systematically evaluate smart stormwater control directly impedes our field's ability to bring new and necessary expertise to solve some of our most essential and complex problems.

As demonstrated by the smart stormwater control survey papers referenced in Section 6.1.1, this desire for direct and systematized comparisons of control strategies is not an isolated realization. While there have been previous efforts to introduce benchmarking stormwater networks for evaluating control strategies [**Schutze2017**, 13], we identify the need to make available a more extensive assortment of example stormwater networks to the broader research community. This assortment of networks must be both nonexclusive and nonrestrictive, and capture the complexity and diversity of features unique to stormwater. Furthermore, we recognize that there is a need for an unambiguous programming interface that explicates the computational backend and aids researchers to easily utilize the example networks for prototyping stormwater control solutions.

We developed `pystorms` as a Python-based *simulation sandbox* to accelerate a researcher's ability to computationally simulate and evaluate stormwater control strategies. `pystorms` provides a collection of diverse stormwater control scenarios, which are drawn from real-world urban watersheds to encompass diverse features appertaining to stormwater systems. These scenarios are coupled with a stormwater simulator and streamlined programming interface, which together provide researchers with a standalone package that focuses its usage on stormwater control algorithm development and testing. Our intention is that `pystorms` reduces the programming learning curve that can be a barrier to those aspiring to learn stormwater control, and also curates an open repository of smart stormwater control examples which foster the development and evaluation of any number of new control strategies applied to them. In the following section, we present a detailed overview

of the design and architecture of `pystorms`, and how it facilitates the systematic evaluation of stormwater control strategies.

Table 6.1: Terminology defined for the `pystorms` package and to delineate stormwater scenarios.

Network Topology	We distinguish a *network* to be the physical system of conduits (e.g. pipes, culverts), storage elements (e.g. retention and detention basins), and any other subcatchment infrastructure (e.g. green infrastructure, wetlands) that collect, convey, and/or treat stormwater runoff.
Event Driver	Any inputs or "disturbances" to the network that govern the generation and flow of runoff are defined as event drivers. Most often, an event driver is the precipitation generating runoff in the watershed. It can also include wastewater flows, tidal fluctuations of connected water bodies, or any other such phenomenon that influence the flow of runoff in the network.
Controllable Assets	Any elements (e.g. basins, wetlands, CSO pump stations) that are equipped with valves, pumps, or any other flow control infrastructure that can be actuated to manipulate stormwater flow.
Observable States	The collection of states in the network (e.g. water levels, flows, pollutants) that can be accessed by the users during a simulation.
Control Objectives	The overall goal or set of goals (e.g. preventing flooding, improving water quality, reducing erosion) of manipulating the behavior of a stormwater network using controllable assets during a simulation. The ability of a controller to achieve a particular objective is quantified using a performance metric.

6.2 `pystorms`

Developed in Python, `pystorms` is supported on all major operating systems (OSX, Windows, and Linux) and can be installed using `pip`[3]. `pystorms` is distributed under the GNU General Public GPLv3 license[4], which ensures that this package and its derivatives remain open-source and can be used free of cost. Additionally, source code for the package is available on Github[5], alongside comprehensive documentation and tutorials to utilize and contribute to its broader development[6].

3 pypi.org/project/pystorms
4 gnu.org/licenses/gpl-3.0.html
5 github.com/kLabUM/pystorms
6 open-storm.org/pystorms

6.2.1 Scenarios

pystorms abstracts smart stormwater systems as *scenarios*. Each scenario is described by (i) *an underlying stormwater network*–which includes the network's topology (e.g. a sewer system draining into a water body) and its event driver (e.g. storm event)–and its *overlaying control system*, which includes a set of observable states (e.g. water levels), controllable assets (e.g. basins with controllable valves at outlet), and a specific control objective (e.g. preventing flooding). The specific terminology for a scenario in pystorms is described in further detail in Table 6.1, and the corresponding delineation between (i) and (ii) is illustrated in Fig. 6.1.

By abstracting our stormwater systems as scenarios, we are able to create new scenarios with relative ease by interchanging different scenario components. For example, let us assume we have evaluated a control algorithm when applied to an individual scenario. We can now broaden our inquiry and test the algorithm's *scalability* by interchanging components of our scenario's overlaying control system (e.g. sequentially increase its number of controllable assets), and evaluating the algorithm on this newly derived set of scenarios. Similarly, we can quantify the *generalizability* of a control algorithm as it is applied to a specific control objective (e.g. maintaining water level set points) by systematically altering the underlying stormwater network (e.g. cycle through a set of design storms as the event driver) while retaining the overall control system. We can then calculate a performance metric of the algorithm when applied across this new subsequent set of scenarios. Thus, not only can can now evaluate a control algorithm applied to an individual stormwater scenario, but we can also evaluate it more universally when applied across a spectrum of these interchanged scenarios.

pystorms provides an collection of seven scenarios, drawn from real-world smart stormwater systems across North America and Europe and named as a letter from the Greek alphabet. The collection of scenarios span a multitude of stormwater systems that address a diverse set of urban watershed needs with various smart control objectives. The subcatchment areas range from $0.12 - 67\,\text{km}^2$ in size, and include both combined and separated stormwater arrangements. A brief summary of the collection's scenarios are presented in Table 6.2, with their more detailed descriptions provided throughout this paper's appendices.

While our aim is for this collection of scenarios to be representative of a myriad of smart stormwater control, we recognize that it is certainly not exhaustive. As such, we aspire to grow the pystorms repository of stormwater scenarios through community-driven contributions of new

scenarios. Accordingly, we provide extensive documentation [7] for users to contribute their own scenarios, or modify the existing ones.

To demonstrate what is a scenario in the context of `pystorms`, we present here Scenario `theta`, an idealized stormwater network that can be used for rapid prototyping of control strategies. Scenario `theta`'s *network topology* can be described as two 1000 m^3 storage basins connected in parallel and draining through a shared outlet into a downstream water body. The *event driver* is a synthetic rain event lasting 9 hr with a peak intensity of 3.2 in. We stipulate the *observable states* to be the water levels at the two basins at each 15 min time-step of the simulation, and the *controllable assets* are outlet valves of both storage basins adjustable at each time-step between 0 – 100 % open. The *control objective* is to maintain the outflow into the downstream water body below a specified threshold of 0.5 $m^3 s^{-1}$, while simultaneously preventing flooding at the basins. The ability of a control strategy to meet `theta`'s control objective is quantified using a pre-defined performance metric that computes a penalty for violating the control objective at each time-step, and sums these penalties across the whole simulation. We provide the specific details on this performance metric (eq. 6.1a) in Section 6.3 where we evaluate the performance of two different example control strategies applied to `theta`.

6.2.2 Programming Interface

The `pystorms` programming interface is inspired by the principles of control theory, where the control of a system is abstracted as an iterative process (also known as a control loop) in which a controller monitors the underlying state(s) of the system of interest, and makes calculated adjustments — via control actions — to the system for it to achieve a desired behavior. In the context of smart stormwater control, our system of interest is our stormwater network, and the states and control actions are represented by the set of observable states and the specific control asset configurations (e.g. valve positions and pump settings). Thus, we have discretized the simulation of stormwater control in `pystorms` as the following series of steps:

1. **Query the set of observable states** for specified locations in the stormwater network at the current time-step; then potentially use these queried states to
2. **Compute control actions** to manipulate the system to achieve a desired behavior; and finally,
3. **Implement the control actions** by adjusting the settings of the controllable assets that serve as inputs into the underlying system.

7 open-storm.org/pystorms/docs/build-scenarios

```python
import pystorms


def controller(state):
    # your control algorithm goes here
    return actions


# initialize scenario
env = pystorms.scenarios.theta()
done = False

while not done:
    # query current state
    state = env.state()
    # compute the control action
    actions = controller(state)
    # implement the action
    done = env.step(actions)

# check your controller's performance
env.performance()
```

Figure 6.2: `pystorms` provides a high-level abstraction for simulating control in stormwater networks for users to quantitatively evaluate the performance of control strategies with minimal overhead.

We initialize a `pystorms` scenario by creating an instance of it using the statement: `pystorms.scenarios.<scenario name>()`. As seen in Fig. 6.2, `theta` is initialized with `pystorms.scenarios.theta()`. The initialization then configures the stormwater simulator with the computational representations necessary to simulate the respective scenario, and returns it as a Python object. This Python object (`env` in Fig. 6.2) can be used to progress and/or pause the stormwater simulator, read and/or write parameters to the network, and utilize any additional `pystorms` functionality. The current state of the underlying stormwater network in the scenario can be queried using the `<scenario object>.state()` call (`env.state()` in Fig. 6.2). `<scenario object>.step(<actions>)` implements the control actions in the stormwater network, progresses the simulation forward a time-step, and returns the current status of simulation (`True` when the simulation has terminated and `False` other-

wise). In Fig. 6.2, `done = env.step(actions)` implements `actions` in the stormwater network and progresses the simulation being handled by the `env` Python object, which in this case is the Scenario `theta`. `done` is assigned `True` when the simulation has terminated, and `False` otherwise.

During the each time-step of the simulation, the ability for the implemented control actions to achieve the scenario's control objective are evaluated by computing the time-step's corresponding performance metric. This computed value is then stored for each time-step, and can be accessed at any time during the simulation using `<scenario object>.performance()` (`env.performance()` in Fig. 6.2). Additional parameters are logged throughout the simulation. While an initial set of these logged parameters is predefined, the user is able to customize this set for any additional parameters of interest.

The series of steps for implementing a control loop into our stormwater simulation is seamlessly integrated throughout the `pystorms` programming interface. Users carry out Step 1 using `<scenario object>.state()`, and Step 3 using `<scenario object>.step(<actions>)`. Separated out to be defined by the user is the controller (Step 2), which maps the observed states to control actions. While implementing the controller into `pystorms` is ultimately left to the user, for our example presented here, we implement it as a Python function block (see Fig. 6.2).

6.2.3 Architecture

The `pystorms` architecture follows the object oriented programming paradigm which relies on classes as its core building blocks. This style of software architecture was chosen to allow `pystorms` to be modular such that users can customize it to meet their own specific requirements and/or workflows. While the `pystorms` programming interface is designed with the intent to be intuitive for all potential users, it particularly caters to those who may only have a rudimentary understanding of stormwater dynamics and/or basic familiarity with programming in Python. However, it can also be easily customized to meet the requirements of researchers who want to incorporate advanced functionality, such as custom water quality or rainfall-runoff modules (for details on how to utilize `pystorms` modularity and customization, we again direct the reader to its online documentation).

The `pystorms` architecture is organized to accomplish two tasks: (1) the configuration of the scenario metadata, and (2) the simulation of the stormwater network. These two tasks are carried out using three core interacting modules: `environment`, `scenario`, and `config`. These

three modules interface with each other to build and execute the various scenarios. Fig. 6.3 provides a schematic of this architecture. The first two modules handle the stormwater simulation, while the latter handles the computational representation of the stormwater networks and the metadata pertaining to the control problem (i.e. states, actions, and objectives).

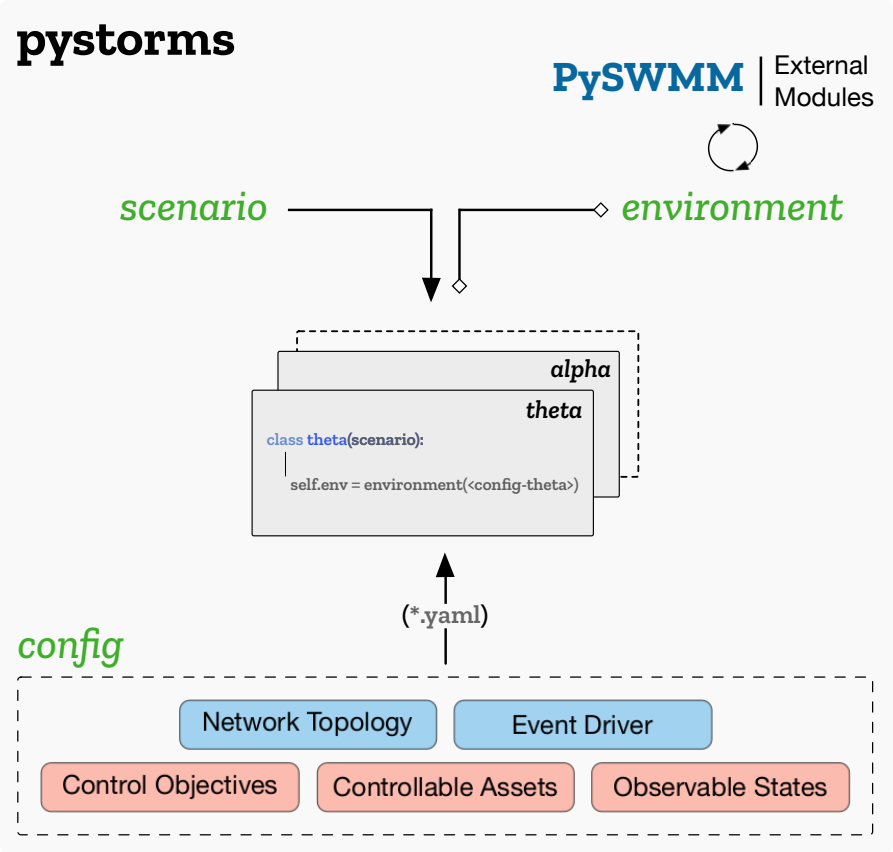

Figure 6.3: `pystorms` is built with three interacting core modules: (i) `config` represents the metadata and computational representations of the stormwater network and event driver; (ii) `environment` acts as an interface for scenarios to interact with the stormwater simulators; and (iii) `scenario` provides a consistent structure for the scenarios in the package. A scenario object in `pystorms` inherits (represented by arrows) from the base `scenario` class, and interfaces (represented by line) with the stormwater simulator though the `environment`.

6.2.3.1 Configuration

The `config` module is used to manage the configuration in the `pystorms` architecture. `config` contains a configuration file for each scenario, which delineates the stormwater network, its set of observable states and controllable assets, and the set of parameters that are used to compute its control objective's corresponding performance metric. The config-

uration files are written using YAML, a mark-up language commonly used for developing configuration files in software applications. With YAML, the parameters of interest defined in the configuration file are formatted as vertical lists rather than data structures. As a result, the configuration file becomes more human-readable, and creates a scalable and easy workflow for developing scenarios.

6.2.3.2 Simulation

Scenarios in `pystorms` are implemented as Python classes. To ensure consistent functionality across scenarios, each scenario is instantiated as its own independent class with an inherited structure from a base `scenario` module. The scenario classes interface their corresponding configuration files with the stormwater simulator and implement any of the functions specific to that scenario (e.g. functions used for computing performance metrics of corresponding control objectives).

The `environment` module is the interface between the stormwater simulator (e.g. EPA-SWMM) and the scenarios. This module is specifically included to ensure `pystorms` is able to remain agnostic to whatever stormwater simulator is used. For instance, if a user wants to utilize a customized hydrologic solver for simulating stormwater, they can do so by modifying the `environment` module to call their solver when the scenarios query it, thus ensuring compatibility to a wide array of simulators with minimal overhead.

`pystorms` uses SWMM as its default stormwater simulator. SWMM, developed by the US-EPA, is an open-source stormwater simulation model that is extensively used for the design and analysis of stormwater systems across the world. SWMM is built with the C programming language, a low-level language that results in significant computational efficiency. However, the trade off for using C is SWMM's difficulty to be interfaced with the latest scientific libraries. As a result, there have been several efforts over the years to build wrappers for SWMM such that its functionality can be exploited via high-level programming languages, such as Python.

PySWMM is a Python implemented package that not only provides a wrapper to communicate with SWMM, but also yields a high-level user interface for querying the various stormwater parameters. `pystorms` — by means of the `environment` module — interfaces with SWMM using PySWMM, and as a result, all functionality included in PySWMM can also be accessed using `pystorms`. Readers are directed to the documentation for additional details and examples to customize `pystorms` to meet their requirements.

Algorithm 1: Equal-Filling Control Algorithm: Let i be a tank in the network of $\mathcal{N}$ tanks. In scenario theta, $\mathcal{N} = 2$ and Max depth in each tank is 2.0m

Let λ be the target flow to be achieved
for *all* $\mathcal{N}$ *tanks* **do**
  Compute the *filling degree*; $f_i = {}^{\text{depth}_i}/_{\text{Max depth}_i}$
Estimate the *average filling degree*; $\bar{f} = \sum_i^N f_i/N$
for *all* $\mathcal{N}$ *tanks* **do**
  Let $\psi_i = f_i - \bar{f}$
  if $\psi_i < 0.0$ **then**
    $\psi_i = 0.0$
  else if $\psi_i = 0.0$ **then**
    $\psi_i = \bar{f}$
for *all* $\mathcal{N}$ *tanks* **do**
  Assign valve positions; $v_i \propto \lambda \times \{\psi_i / \sum_i^N \psi_i\}$

```python
def controller(depths,
               N=2,
               LAMBDA=0.3,
               MAX_DEPTH=2.0):
    # Compute the filling degree
    f = depths/MAX_DEPTH

    # Estimate the average filling degree
    f_mean = np.mean(f)

    # Compute psi
    psi = np.zeros(N)
    for i in range(0, N):
        psi[i] = f[i] - f_mean
        if psi[i] < 0.0:
            psi[i] = 0.0
        elif psi[i] == 0.0:
            psi[i] = f_mean

    # Assign valve positions
    actions = np.zeros(N)
    for i in range(0, N):
        if depths[i] > 0.0:
            k = 1.0 / np.sqrt(2 * 9.81 * depths[i])
            action = k * LAMBDA * psi[i]/np.sum(psi)
            actions[i] = min(1.0, action)

    return actions
```

Figure 6.4: Equal-filling controller maintains the flows at the outlet below a desired threshold by coordinating its actions such that it equally utilizes the storage in the controllable assets of the network. Algorithm 1 and the corresponding code snippet illustrate the algorithm and its Python implementation. An interactive example of algorithm implementation and its evaluation on Scenario `theta` can be accessed at open-storm.org/pystorms/demo

6.3 DEMO: EVALUATING CONTROL STRATEGIES

Throughout this section, we demonstrate how `pystorms` facilitates developing smart stormwater control strategies by evaluating the performance of two control algorithms applied to Scenario `theta`.

While there exist many control strategies that can be adopted to achieve `theta`'s control objective, to simplify our illustration of `pystorms`, we implement two basic control strategies here. The algorithms used to implement the control strategies are described below and in Fig. 6.4. Both control strategies are simple reactive control strategies, in which the valve settings of `theta`'s two basin outlets are adjusted to either retain or release storage depending on the observed states compared some corresponding water level limits.

RULE-BASED CONTROL Our basic rule-based control strategy adjusts our basin outlets based on their respective water levels. Specifically, each basin's outlet setting is equal to its relative water level (i.e., the current water level of the basin divided by its maximum depth). Therefore, our control algorithm will set a full basin's outlet to 100% open, and a basin that is half full will have its outlet set to 50% open, etc. While this strategy provides a means to mitigate local flooding at each basin, it notably does not consider the other control objective for the network's outflow into the downstream water body to stay below a given threshold.

EQUAL-FILLING DEGREE CONTROL The equal-filling degree is a control strategy often applied to stormwater networks with distributed stormwater storage assets, and has commonly been used as a starting

point when comparing more than one control strategies [13, 19, 31, 65, 144]. For this strategy, we begin by defining a storage asset's "filling degree" — which is typically the ratio a storage asset is full based on its volume or depth — and compute it for each asset in the collection system. The algorithm seeks to "balance" these filling degrees across the system based on its average. The exact manner in which this balancing is carried out is not necessarily consistent in literature. Our method for this balancing is delineated in the algorithm in Fig. 6.4. If all assets have a filling degree equal to the average (i.e., all assets are equally filled), then each should release an equal fraction of the target outflow. Otherwise, the released flows across the assets should be differentiated such that, when an asset has a filling degree less than the average, it does not release any flow; but if an asset is greater than the average, it releases flows based on its deviation from the average.

The implementation of the equal-filling degree algorithm using `pystorms` can be seen in Figure 6.4. We carry out the simulation for each of the two algorithms, as well as for the *uncontrolled case*, in which control actions are never implemented and the basin outlets are always open. The resulting hydraulic behavior at the two basins and the network's outflow for each of these simulation runs can be seen in Figure 6.5.

Our aim to find a control strategy that can meet `theta`'s *control objective* to maintain the outflow into the downstream water body below a specified threshold of $0.5\,\mathrm{m^3 s^{-1}}$ and also minimize flooding at the basins. As discussed in Section 6.2.1, we pre-define a performance metric to quantify our control algorithm's ability to meet the corresponding control objective. For Scenario `theta`, this performance metric, P, is defined as:

$$P = \sum_{t=0}^{T} \left(\mathcal{H}_t + \sum_{i=1}^{2} \mathcal{G}_{i,t} \right) \tag{6.1a}$$

$$\mathcal{H}_t = \begin{cases} Q_t - 0.5\,, & \text{if } Q_t > 0.5 \\ 0.0\,, & \text{otherwise} \end{cases} \tag{6.1b}$$

$$\mathcal{G}_{i,t} = \begin{cases} 10^3\,, & \text{if any flooding at basin } i \\ 0.0\,, & \text{otherwise} \end{cases} \tag{6.1c}$$

where $\mathcal{H}_t$ is a flow exceedance penalty of the stormwater network's outflow, Q_t, over the $0.5\,\mathrm{m^3 s^{-1}}$ threshold; and $\mathcal{G}_{i,t}$ is an arbitrary flooding penalty of 10^3 added whenever there exists flooding at either of our two basins, both calculated and summed across every time-step t in the simulation.

The performance metric calculated across the simulations for both implemented control algorithms and the uncontrolled case can be seen

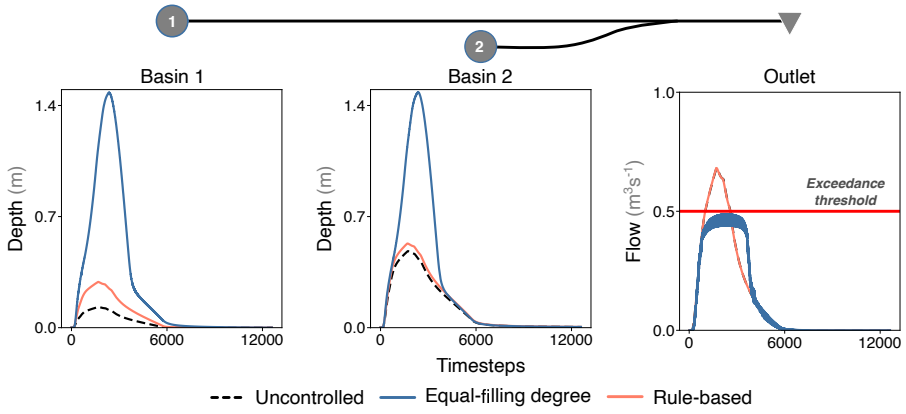

Figure 6.5: In Scenario `theta`, the equal-filling degree control strategy is successfully able to maintain the flows at the outlet of the watershed below the desired threshold of 0.5 m^3s^{-1} by uniformly using the storage in the networks. Static rule-based control and uncontrolled responses of the networks are also presented for comparison. The maximum depth in each of the two basins is 2 m.

in Table 6.3). Additionally, the hydraulic behavior of our two basins and the network outlet when these algorithms are applied versus the uncontrolled case can be seen in Fig. 6.5.

As can be seen, the equal-filling degree strategy is able to achieve the control objective of the outflow threshold, as well as avoidance of flooding. Alternatively, the rule-based control strategy only is able to avoid flooding at the basins. The stormwater network behavior for both strategies follow their corresponding implemented algorithm. For example, as the rule-based control strategy does not directly consider the outflow threshold when determining the implemented control actions, it follows that the outflow in the network's outlet exceeds this threshold (see the outlet plot in Fig. 6.5).

The results for each implemented control strategy versus the uncontrolled case are also captured using `theta`'s performance metric seen in Equation 6.1. As the performance metric is ultimately a sum of penalties for violating the control objective, a smaller calculated performance metric value indicates a better performing control algorithm. The respective performance metric values for each control strategy presented here can be seen in Table 6.3. With a calculated performance metric of 0, the equal-filling degree strategy perfectly meets `theta`'s control objective; comparatively, the rule-based and uncontrolled cases have higher performance metric values, and thus, we can conclude perform worse than the equal-filling degree.

6.4 DISCUSSION

This ability for stormwater systems to be instantly modified is critical as communities prepare for more frequent, uncertain, and destructive weather events due to climate change. More so, *often the most basic control strategies* can have large-scale impacts on the complex, dynamic systems they operate, potentially leading to millions of dollars in savings for the communities they serve. Even though sensor-actuator components may be successfully deployed at individual sites throughout a stormwater network, determining strategies for their coordination across the entire watershed may only add further complexity. As a result, there is a great need — along with endless opportunities — to develop and implement novel control strategies for transforming stormwater systems. While the sandboxing efforts of `pystorms` serves as an initial effort to foster the development of these strategies, we see specific opportunities to, first, methodically facilitate the development of new simulation frameworks and control algorithms, and subsequently, to then validate and extend their efficacy. We expound on these points here, and discuss next steps to put them into practice.

As discussed in Section 6.1.2, a critical limitation to progressing smart stormwater control research forward is the inability to systematically develop and analyze smart stormwater simulation workflows and control algorithms. `pystorms` can be customized and adapted for a multitude of other uses beyond its initially provided collection of scenarios and stormwater simulator provided. For example, alternative stormwater simulation software be easily integrated into `pystorms`. Furthermore, new scenarios can be assembled from the assortment of components in the scenario collection such that additional research questions can be studied. For instance, for each of the scenarios we provide at the outset, `pystorms` specifies only a subset of a scenario's total observable states that are able to be queried throughout the simulation. However, this initial subset of observable states is never claimed as the optimal; in fact, to the best of our knowledge, there does not yet exist a methodology for identifying an optimal set of observable states. Thus, new scenarios can be made with different subsets of observable states (e.g. flows, pollutant concentrations), and new research questions can now be asked about which states may be most critical for informing control actions to be taken.

Beyond the coordination and integration of smart stormwater control methods, we view a more expansive opportunity for `pystorms` to impel the research community to extend its analysis of "control." Specifically, we see a need for improved validation of control methods, and an opportunity for new approaches in defining a control method's success. In the current iteration of `pystorms` discussed here, we present a means to

assess the performance of a control algorithm via its ability to achieve the pre-defined control objective (e.g. maintain flow below a threshold, avoid flooding). However, there are many other assessment metrics that can define a control algorithm's "success," including computational efficiency, applicability across real-world contexts, and the incorporation of social considerations for actual implementation. `pystorms` can serve as a mechanism for assessing the performance of control algorithms across these definitions. For example, by increasing the number of controllable assets available out of the eleven pond outlets presented in Scenario `gamma`, one can assess the *scalability* of a control algorithm as the state-action space increases. Additionally, control algorithm *generalizability* across storm characteristics can be assessed with the multiple rain events provided in Scenario `epsilon`. These are just a few illustrations of how `pystorms` provides a way to broaden and assess the definition of control efficacy to include factors that are critical for the implementation of smart stormwater approaches in real-world systems.

6.5 CONCLUSIONS AND NEXT STEPS

`pystorms` provides a curated collection of scenarios, coupled with an accessible programming interface, to enable the development and quantitative evaluation of stormwater control algorithms. We have developed `pystorms` with the intent to make research into smart stormwater control more accessible to the broader research community. It is our hope that this package will emerge as a community-driven resource that is able to address key knowledge gaps and enable the advancement of smart stormwater systems. To this extent, we see proximate opportunities for the broader research community to collaborate on `pystorms` by contributing their own stormwater scenarios and/or control algorithms to the package initiated here. Likewise, we encourage the broader research community to further build upon `pystorms` by imparting their own smart stormwater control instances using the `pystorms` architecture and integrating their own stormwater control simulation workflows into it.

Table 6.2: `pystorms` includes a curated collection of real world-inspired stormwater scenarios for developing and quantitatively evaluating the performance of stormwater control algorithms.

Scenario	Network	Control Objectives
`theta`	2 km^2 idealized separated stormwater network	Maintain the flows at the outlet below a threshold and avoid flooding (2 storage basin outlets)
`alpha`	0.12 km^2 residential combined sewer network	Minimize total combined sewer overflow volume (5 weirs at interceptor connections)
`beta`	1.3 km^2 separated stormwater network with a tidally-influenced receiving river	Minimize flooding (1 detention pond outlet, 1 storage basin outlet, 1 pump)
`gamma`	4 km^2 highly urban separated stormwater network	Maintain channel flows below threshold and avoid flooding (11 detention pond outlets)
`delta`	1.7 km^2 combined sewer network in which the stormwater ponds also serve as waterfront	Maintain water levels within upper and lower thresholds for water quality and aesthetic objectives (5 storage basin outlets)
`epsilon`	67 km^2 highly urban combined sewer network	Maintain sewer network outlet TSS load below threshold and avoid flooding (11 in-line storage dams)
`zeta`	1.8 km^2 combined and separated sewer network (based on the Astlingen benchmarking network [**Schutze2017**, 134])	Maximize flow to downstream wastewater treatment plant and minimize total combined sewer overflow volume (4 storage basin outlets)

Table 6.3: Calculated performance metric values from Equation 6.1 for simulations corresponding to the two implemented control algorithms and the uncontrolled simulation. As can be seen, the equal-filling degree control strategy performs better than the rule-based control strategy, which then outperforms the uncontrolled case.

Control Strategy	**Performance Metric**
Uncontrolled	1630
Rule-based	1624
Equal-filling Degree	0

7 CONCLUSION

Part I

APPENDIX

A APPENDIX-1

A.1 CASE STUDY 1: LOCAL CONTROL

- Area : 500 m^2
- Max height : 1.5 m
- $K_{Nitrate}$ = 42.048 $year^{-1}$
- K_{Oxygen} = 31.536 $year^{-1}$

A.2 CASE STUDY 2: SYSTEM-LEVEL CONTROL

- Pond 1
  - Area : 1000 m^2
  - Max height : 2.5 m
  - $K_{Nitrate}$ = 21.024 $year^{-1}$
  - K_{Oxygen} = 525.60 $year^{-1}$
- Pond 2
  - Area : 600 m^2
  - Max height : 2.5 m
  - $K_{Nitrate}$ = 21.024 $year^{-1}$
  - K_{Oxygen} = 1051.2 $year^{-1}$
- Pond 3
  - Area : 1000 m^2
  - Max height : 2.5 m
  - $K_{Nitrate}$ = 15.768 $year^{-1}$
  - K_{Oxygen} = 1051.2 $year^{-1}$
- Wetland
  - Area : 1000 m^2
  - Max height : 2.4 m
  - Weir-height : 1.5 m
  - $K_{Nitrate}$ = 25.228 $year^{-1}$
  - K_{Oxygen} = 1051.2 $year^{-1}$

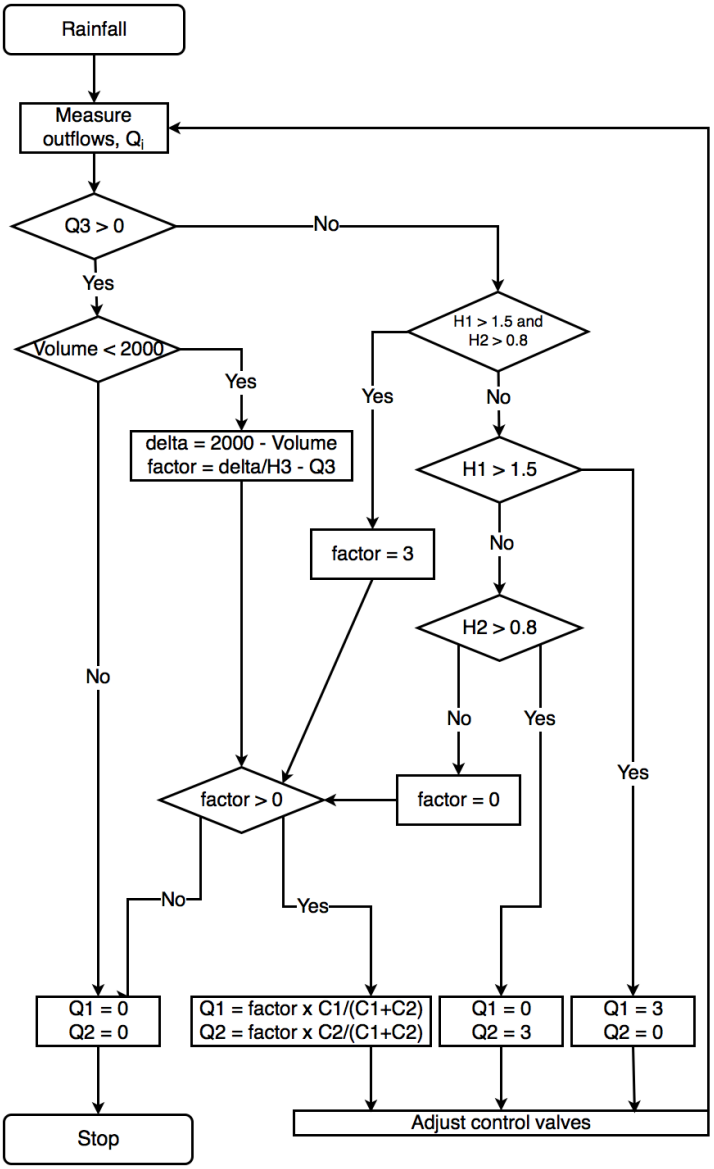

Figure A.1: Rule-based algorithm used for controlling the system in the second case study.

B APPENDIX-2

B.1 DEEP NEURAL NETWORKS

Broadly, neurons are the fundamental processing elements of a neural network. They receive their inputs (x_{ij}) as the weighted (w_{ij}) outputs from the neurons in the previous layers and produce a single output signal (y_j) dependent upon a bias (b_j) and activation function $f(*)$

$$z_j = \sum_{i}^{n} w_{ij} x_{ij} + b_j \tag{B.1}$$

$$y_j = f(z_j) \tag{B.2}$$

More specifically, "Deep" neural networks are a collection of such neurons organized as distinct layers. A neural network approximates a function by fine tuning its weights and biases so that its output signal closely resembles the output of the function it is approximating for the same inputs. The degree of resemblance between the signals is computed based on a loss function $L(*)$

$$Loss = L(Q_p, Q_o) \tag{B.3}$$

$$w_{ij} = w_{ij} - \alpha \times \frac{dL(Q_p, Q_o)}{(dw_{ij})} \tag{B.4}$$

$$b_j = b_j - \alpha \times \frac{dL(Q_p, Q_o)}{(db_j)} \tag{B.5}$$

The choice of the loss function is dictated by the nature of the function being approximated. For example, a neural network approximating rainfall runoff may use mean squared error[139]

$$Loss = |Q_p - Q_o|^2 \tag{B.6}$$

The closer the correspondence between the signals, lower the loss.

Neural networks minimize the loss though stochastic gradient descent, starting with a set of weights and biases (either random or values sampled from a distribution). Based on the value of loss function, the values of weights and biases are adjusted, and the neural network attempts to approximate the function with these updated values. This process of tuning the weights and biases is repeated until the neural network can approximate the function to satisfaction or loss is minimized. While Deep Neural Networks show significant promise in approximating functions, their ability to do so is contingent upon several factors: Stability of the learning process, the size and depth of the network, the underlying data distributions [24, 48]. Fundamental description of Deep Neural Networks can be found in established textbooks [48].

B.2 HYPER PARAMETERS AND ARCHITECTURE

Neural Network

Layers	2
Number of Neurons per layer	50

Gradient Descent

Learning Rate	10^{-3}
Rho	0.9
Epsilon	10^{-8}
Decay	0.0

Batch Normalization

Momentum	0.99
Epsilon	0.001

Deep Q Network

Target Network Update	10000
Gamma	0.99
Replay Buffer	100000

B.3 EQUAL-FILLING DEGREE ALGORITHM

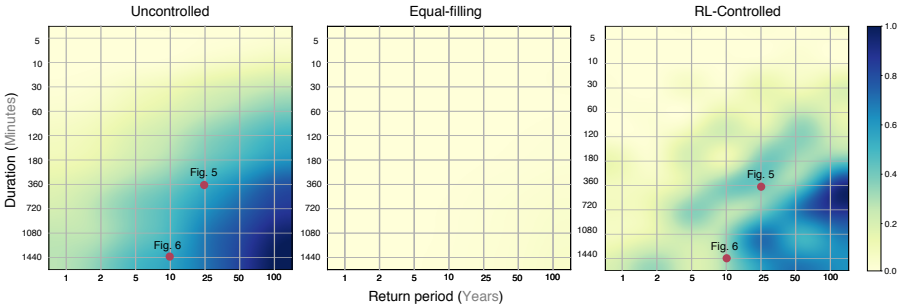

Figure B.1: Performance comparison between uncontrolled and controlled systems (RL and equal-filling) across a spectrum of stormevents. Equal-filling approach is able to successfully control the system to achieve the objective of minimizing flows in all the stormevents.

Equal filling degree algorithm is a rule based real-time control approach for controlling stormwater networks. In this approach, control decisions of either releasing or holding back water, are made based on the filling degree (f_i) in the controlled asset (i). In this context, filling degree is defined as the normalized depth in an asset. By releasing water based on the difference between the average filling degree of the network ($\bar{f}$) and the filling degree (f_i) in each asset, equal-filling approach ensures that the storage potential in the control assets is utilized uniformly.

Algorithm 1: Equal-Filling Control Algorithm: Let i be a basin in the network of $\mathcal{N}$ tanks. In this scenario, $\mathcal{N} = 4$.

for *all* $\mathcal{N}$ *tanks* **do**
  Compute the *filling degree*; $f^i = \frac{\text{depth}^i}{\text{Max depth}^i}$
Estimate the *average filling degree*; $\bar{f} = \frac{\sum_i^N f^i}{N}$
for *all* $\mathcal{N}$ *tanks* **do**
  if $f^i > \bar{f}$ **then**
    $valve^i = c \times (f^i - \bar{f})$
  else if $\bar{f} - f^i \leq \theta$ **then**
    $valve^i = \bar{f}$
  else
    $valve^i = 0.0$

Hyper-parameter c can be used to regulate the valve opening and θ can be used to control the flashiness of the outflow hydrograph. In this study, these parameters were chosen to be 3.0 and 0.18 respectively. Performance of the equal-filling algorithm across the 70 rainevents is

presented in figure B.1. Performance of the equal-filling control approach in controlling a 6 hour 25 year is presented in figure B.2.

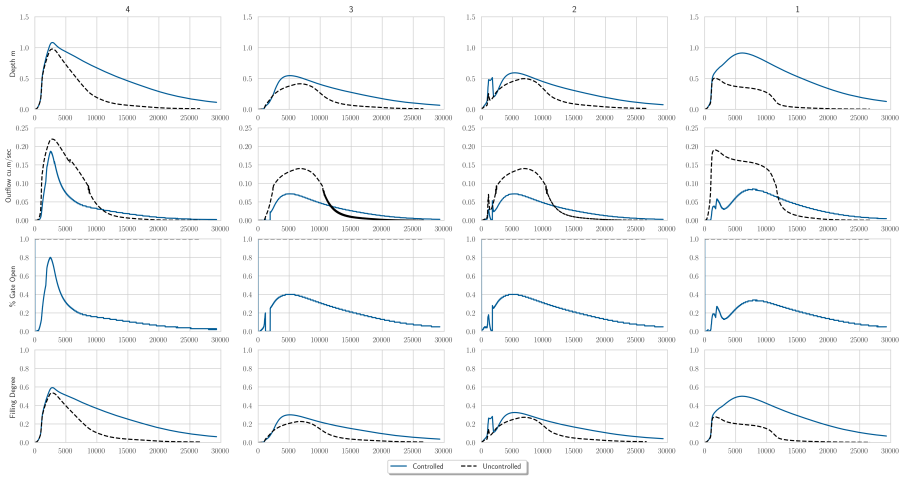

Figure B.2: Equal-filling approach successfully maintains the outflows from the four basins below the desired threshold.

B.4 SCENARIO 1 — REWARD FUNCTION FORMULATION SENSITIVITY

Reward functions 3a and 3b are formulated as a variant of the third reward function (eq.11) to analyse the sensitivity of the agent's performance to the choice of equation used in the formulation reward functions. Reward function 3a (eq. B.7) is formulated using exponential and the 3b reward function is built using squared terms (eq. B.8).

$$r_{3a}(s_t) = \begin{cases} c_1 - c_2 e^{h_t}, & h_t < Hf_t \leq F \\ c_1 - c_3 e^{h_t}, & h_t \geq Hf_t \leq F \\ -e^{f_t} - c_2 e^{h_t} + c_4, & h_t < Hf_t > F \\ -e^{f_t} - c_3 e^{h_t} + c_4, & h_t \geq Hf_t > F \end{cases} \tag{B.7}$$

$$r_{3b}(s_t) = \begin{cases} c_1 - c_2 h_t^2, & h_t < Hf_t \leq F \\ c_3 - c_4 h_t^2, & h_t \geq Hf_t \leq F \\ -c_5 (c_6 f_t)^2 - c_2 h_t^2 + c_7, & h_t < Hf_t > F \\ -c_5 (c_6 f_t)^2 - c_4 h_t^2 + c_7, & h_t \geq Hf_t > F \end{cases} \tag{B.8}$$

3a and 3b reward function are parametrized by $\{c_1 = 2.5, c_2 = 0.5, c_3 = 0.75, c_4 = 3\}$ and $\{c_1 = 2.0, c_2 = 0.5, c_3 = 2.5, c_4 = 0.9, c_5 = 0.1, c_6 = 10, c_7 = 1.5\}$ respectively. These parameters are chosen to constrain the scale of these reward functions to the third reward function.

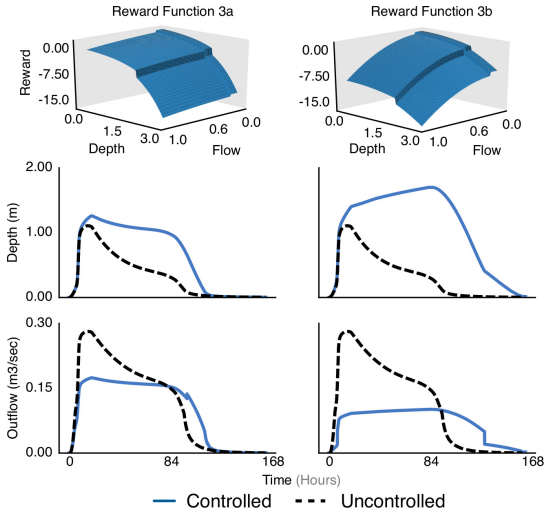

Figure B.3: Performance of the RL agent is independent of the choice of equations used for creating the reward functions.

B.5 SCENARIO 1 — LOCATION SENSITIVITY

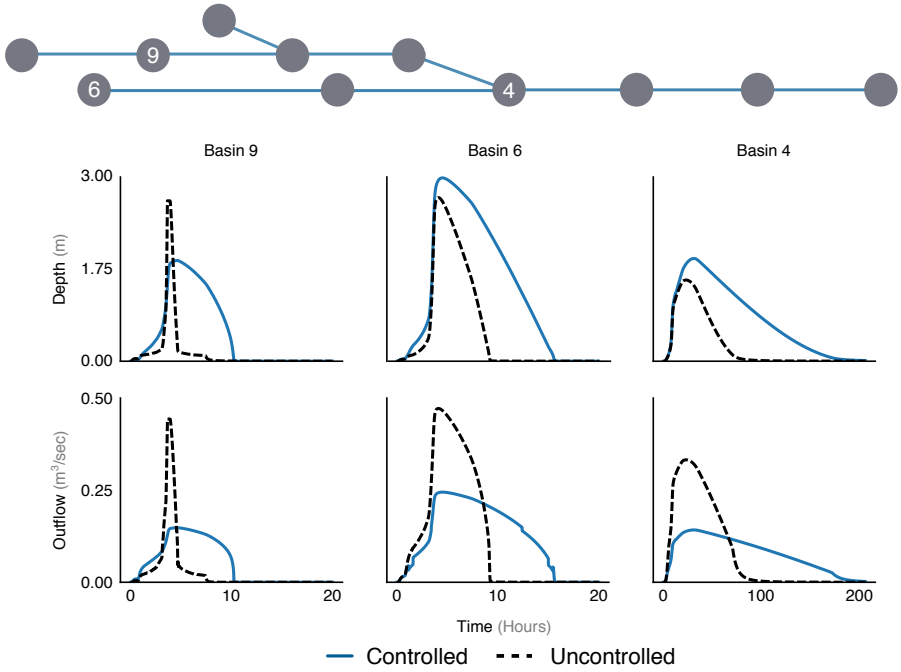

Figure B.4: Performance of the RL agent controlling the basins across the network. Controller is able to control the response of the basins irrespective of the location, indicating the independence of the control algorithm's performance on the location of the basin. Results presented in this figure are independent simulations; each column represents a simulation where only that particular basin is controlled.

B.6 SCENARIO 2 — BACK-TO-BACK EVENT

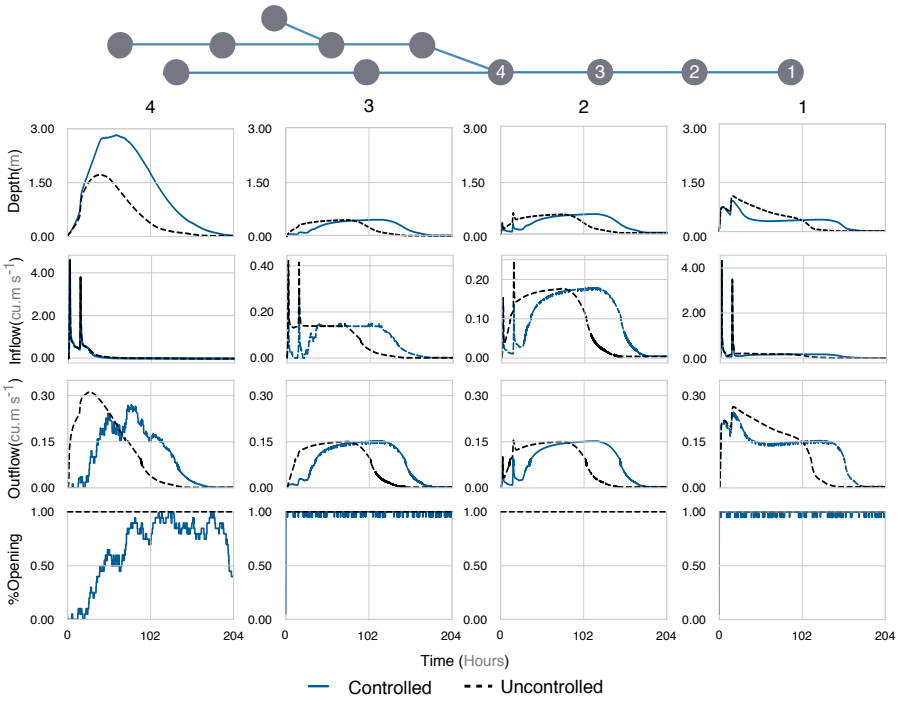

Figure B.5: Performance of the RL controller in a back-to-back event. Given that the controller is trained to react based on the depth in the basins and not the rainfall experienced in the watershed, controller treats this back-to-back event as if it were a single event.

BIBLIOGRAPHY

[1] Martín Abadi et al. "Tensorflow: a system for large-scale machine learning." In: *OSDI*. Vol. 16. 2016, pp. 265–283.

[2] Baher Abdulhai and Lina Kattan. "Reinforcement learning: Introduction to theory and potential for transport applications." In: *Canadian Journal of Civil Engineering* 30.6 (2003), pp. 981–991.

[3] H.H. Alvord and R.H. Kadlec. "Atrazine fate and transport in the Des Plaines Wetlands." In: *Ecological Modelling* 90.1 (1996), pp. 97–107. ISSN: 03043800.

[4] Karl Joha Aström, Richard M Murray, and Richard Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2006.

[5] Yusuf Aytar et al. "Playing hard exploration games by watching YouTube." In: *arXiv preprint arXiv:1805.11592* (2018).

[6] Matthew Bartos, Brandon Wong, and Branko Kerkez. "Open storm: a complete framework for sensing and control of urban watersheds." In: *arXiv preprint arXiv:1708.05172* (2017).

[7] Matthew Bartos, Brandon Wong, and Branko Kerkez. "Open storm: a complete framework for sensing and control of urban watersheds." In: *Environmental Science: Water Research & Technology* 4.3 (2018), pp. 346–358. DOI: 10.1039/c7ew00374a.

[8] Lorenzo Benedetti et al. "A new rule generation method to develop a decision support system for integrated management at river basin scale." In: *Water Science & Technology* 60.8 (2009), p. 2035.

[9] Mark A Benedict, T Edward, and J D Mcmahon. "Green Infrastructure: Smart Conservation for the 21st Century." In: *Green Infrastructure: A Strategic Approach to Land Conservation*. The Conservation Fund, 2006. Chap. 1, p. 5.

[10] B Bhattacharya, AH Lobbrecht, and DP Solomatine. "Neural networks and reinforcement learning in control of water systems." In: *Journal of Water Resources Planning and Management* 129.6 (2003), pp. 458–465.

[11] Brian P Bledsoe. "Stream erosion potential and stormwater management strategies." In: *Journal of Water Resources Planning and Management* 128.6 (2002), pp. 451–455.

[12] Donald F Boesch, Russell B Brinsfield, and Robert E Magnien. "Chesapeake Bay Eutrophication: Scientific Understanding, Ecosystem Restoration, and Challenges for Agriculture." In: *Journal of Environmental Quality* 30.2 (2001), pp. 303–320.

[13] Péter Borsányi et al. "Modelling real-time control options on virtual sewer systems." In: *Journal of Environmental Engineering and Science* 7.4 (2008), pp. 395–410. DOI: 10.1139/S08-004.

[14] Axel Bronstert, Daniel Niehoff, and Gerd Bürger. "Effects of climate and land-use change on storm runoff generation: present knowledge and modelling capabilities." In: *Hydrological processes* 16.2 (2002), pp. 509–529.

[15] Gary W Brunner and Jurgen Gorbrecht. "A Muskingum-Cunge Channel Flow Routing Method for Drainage Networks." In: *ASCE Journal of Hydraulics* 117.5 (1991), pp. 629–642.

[16] Gary Brunner and CEIWR-HEC. *HEC -RAS River Analysis System User Manual*. 5.0. US Army Corps of Engineers, Institute for Water Resources, 2016.

[17] Matthew J. Burns, Tim D. Fletcher, Christopher J. Walsh, Anthony R. Ladson, and Belinda E. Hatt. "Hydrologic shortcomings of conventional urban stormwater management and opportunities for reform." In: *Landscape and Urban Planning* 105.3 (2012), pp. 230–240. ISSN: 01692046.

[18] CDMSmith. *City of Ann Arbor Stormwater Model Calibration and Analysis Project*. Accessed: 2018-09-25. 2015.

[19] Alberto Campisano, Wolfgang Schilling, and Carlo Modica. "Regulators' setup with application to the Roma–Cecchignola combined sewer system." In: *Urban Water* 2.3 (2000), pp. 235–242. DOI: 10.1016/S1462-0758(00)00061-3.

[20] James N Carleton et al. "Performance of a Constructed Wetlands in Treating Urban Stormwater Runoff." In: *Water Environment Research* 72.3 (2000), pp. 295–304.

[21] A Castelletti, Stefano Galelli, Marcello Restelli, and Rodolfo Soncini-Sessa. "Tree-based reinforcement learning for optimal water reservoir operation." In: *Water Resources Research* 46.9 (2010).

[22] I. Chaubey, C.T. Haan, S. Grunwald, and J.M. Salisbury. "Uncertainty in the model parameters due to spatial variability of rainfall." In: *Journal of Hydrology* 220.1 (1999), pp. 48–61. ISSN: 00221694.

[23] David A Chin, Mazumdar, Asis, Roy, and Kumar. *Water-resources engineering*. 12th ed. Prentice Hall Englewood Cliffs, 2000.

[24] Francois Chollet. *Deep learning with python*. Manning Publications Co., 2017.

[25] Hafedh Chourabi et al. "Understanding smart cities: An integrative framework." In: *Proceedings of the Annual Hawaii International Conference on System Sciences*. IEEE, 2012, pp. 2289–2297. DOI: 10.1109/HICSS.2012.615.

[26] Shih-Kai Ciou, Jan-Tai Kuo, Pin-Hui Hsieh, and Gwo-Hsing Yu. "Optimization Model for BMP Placement in a Reservoir Watershed." In: *Journal of Irrigation and Drainage Engineering* 138.8 (2012), pp. 736–747.

[27] Adrienne R. Cizek and William F. Hunt. "Defining predevelopment hydrology to mimic predevelopment water quality in stormwater control measures (SCMs)." In: *Ecological Engineering* 57 (2013), pp. 40–45. ISSN: 0925-8574.

[28] Ignasi Clavera et al. "Model-Based Reinforcement Learning via Meta-Policy Optimization." In: *arXiv preprint arXiv:1809.05214* (2018).

[29] Petra van Daal, Günter Gruber, Jeroen Langeveld, Dirk Muschalla, and François Clemens. "Performance evaluation of real time control in urban wastewater systems in practice: Review and perspective." In: *Environmental Modelling and Software* 95 (2017), pp. 90–101.

[30] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. "Learning and policy search in stochastic dynamical systems with bayesian neural networks." In: *arXiv preprint arXiv:1605.07127* (2016).

[31] G. Dirckx et al. "Cost-efficiency of RTC for CSO impact mitigation." In: *Urban Water Journal* 8.6 (2011), pp. 367–377. DOI: 10.1080/1573062X.2011.630092.

[32] J.R. Dormand and P.J. Prince. "A family of embedded Runge-Kutta formulae." In: *Journal of Computational and Applied Mathematics* 6.1 (1980), pp. 19–26. ISSN: 03770427.

[33] A.H. Elliott and S.A. Trowsdale. "A review of models for low impact urban stormwater drainage." In: *Environmental Modelling & Software* 22.3 (Mar. 2007), pp. 394–405. ISSN: 1364-8152. DOI: 10.1016/J.ENVSOFT.2005.12.005.

[34] Clay H. Emerson, Claire Welty, and Robert G. Traver. "Watershed-Scale Evaluation of a System of Storm Water Detention Basins." In: *Journal of Hydrologic Engineering* 10.3 (May 2005), pp. 237–242. ISSN: 1084-0699. DOI: 10.1061/(ASCE)1084-0699(2005)10:3(237).

[35] Department of Environmental Protection. *Pennsylvania Stormwater Best Management Practices Manual*. Bureau of Watershed Management, 2006.

[36] Lasse Espeholt et al. "IMPALA: Scalable distributed Deep-RL with importance weighted actor-learner architectures." In: *arXiv preprint arXiv:1802.01561* (2018).

[37] David Fiorelli, Georges Schutz, Kai Klepiszewski, Mario Regneri, and Stefanie Seiffert. "Optimised real time operation of a sewer network using a multi-goal objective function." In: *http://dx.doi.org/10.1080/1573062X.2* (2013).

[38] Dan Frosch and Cameron McWhirter. *Houstons Rapid Growth, Heavy Rains, Heightend Flood Risk*. 2016.

[39] Justin Fu, Katie Luo, and Sergey Levine. "Learning Robust Rewards with Adversarial Inverse Reinforcement Learning." In: *arXiv preprint arXiv:1710.11248* (2017).

[40] E Gaborit, D Muschalla, B Vallet, P A Vanrolleghem, and F Anctil. "Improving the performance of stormwater detention basins by real-time control using rainfall forecasts." In: *Urban Water Journal* 10.4 (2013), pp. 230–246. DOI: 10.1080/1573062X.2012.726229.

[41] E Gaborit, F Anctil, G Pelletier, and P A Vanrolleghem. "Exploring forecast-based management strategies for stormwater detention ponds." In: *Urban Water Journal* 13.8 (2015), pp. 1–11. DOI: 10.1080/1573062X.2015.1057172.

[42] L. Garcia et al. "Modeling and real-time control of urban drainage systems: A review." In: *Advances in Water Resources* 85 (2015), pp. 120–132. ISSN: 0309-1708. DOI: https://doi.org/10.1016/j.advwatres.2015.08.007.

[43] L. García et al. "Modeling and real-time control of urban drainage systems: A review." In: *Advances in Water Resources* 85 (2015), pp. 120–132. DOI: 10.1016/j.advwatres.2015.08.007.

[44] Noah Garrison. *Stormwater Capture Potential in Urban and Suburban California*. Tech. rep. Natural Resources Defense Council, 2014.

[45] Geosyntec. *Stormwater Capture Master Plan Interim Report*. Tech. rep. Geosyntec, 2014.

[46] D. Giraldi, M. de Michieli Vitturi, and R. Iannelli. "FITOVERT: A dynamic numerical model of subsurface vertical flow constructed wetlands." In: *Environmental Modelling & Software* 25.5 (2010), pp. 633–640. ISSN: 13648152.

[47] Jonathan L. Goodall, Bella F. Robinson, and Anthony M. Castronova. "Modeling water resource systems using a service-oriented computing paradigm." In: *Environmental Modelling & Software* 26.5 (2011), pp. 573–582. ISSN: 13648152.

[48] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. Vol. 1. MIT press Cambridge, 2016.

[49] Ashantha Goonetilleke, Evan Thomas, Simon Ginn, and Dale Gilbert. "Understanding the role of land use in urban stormwater quality management." In: *Journal of Environmental Management* 74.1 (2005), pp. 31–42. ISSN: 03014797.

[50] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. "Continuous deep q-learning with model-based acceleration." In: *International Conference on Machine Learning*. 2016, pp. 2829–2838.

[51] Perrine Hamel, Edoardo Daly, and Tim D. Fletcher. "Source-control stormwater management for mitigating the impacts of urbanisation on baseflow: A review." In: *Journal of Hydrology* 485 (2013), pp. 201–211. ISSN: 00221694.

[52] Colin Harrison and Ian Abbott Donnelly. "A theory of smart cities." In: *Proceedings of the 55th Annual Meeting of the International Society for the Systems Sciences*. Vol. 55. 1. 2011.

[53] Peter Henderson et al. "Deep reinforcement learning that matters." In: *arXiv preprint arXiv:1709.06560* (2017).

[54] M Henze. *Activated sludge models ASM1, ASM2, ASM2d and ASM3*. IWA publishing, 2000.

[55] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators." In: *Neural networks* 2.5 (1989), pp. 359–366.

[56] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In: *arXiv preprint arXiv:1502.03167* (2015).

[57] C Jacopin, E Lucas, M Desbordes, and P Bourgogne. "Optimisation of operational management practices for the detention basins." In: *Water science and technology* 44.2 (2001), pp. 277–285.

[58] Kadlec and Robert H. "Phosphorus dynamics in event driven wetlands." In: *Transformations of Nutrients in Natural and Constructed Wetlands* (2001), pp. 365–391.

[59] Robert H Kadlec and Scott Wallace. *Treatment wetlands*. CRC press, 2008.

[60] Branko Kerkez et al. "Smarter Stormwater Systems." In: *Environmental Science & Technology* 50.14 (2016), pp. 7267–7273. ISSN: 0013-936X. DOI: 10.1021/acs.est.5b05870.

[61] Ingemar Kinnmark. *The shallow water wave equations: formulation, analysis and application*. Vol. 15. Springer Science & Business Media, 2012.

[62] Brandon Klenzendorf, Michael Barrett, Marty Christman, and Marcus Quigley. *Water Quality and Conservation Benefits Achieved via Real Time Control Retrofit of Stormwater Management Facilities near Austin, Texas*. Tech. rep. OptiRTC, 2015.

[63] Robert L Knight, William E Walton, George F O'Meara, William K Reisen, and Roland Wass. "Strategies for effective mosquito control in constructed treatment wetlands." In: *Ecological Engineering* 21.4–5 (2003), pp. 211–232. ISSN: 0925-8574.

[64] Jens Kober, J. Andrew Bagnell, and Jan Peters. "Reinforcement learning in robotics: A survey." In: *The International Journal of Robotics Research* 32.11 (Sept. 2013), pp. 1238–1274. ISSN: 0278-3649. DOI: 10.1177/0278364913495721.

[65] Stefan Kroll et al. "Modelling real-time control of WWTP influent flow under data scarcity." In: *Water Science and Technology* 73.7 (2016), pp. 1637–1643. DOI: 10.2166/wst.2015.641.

[66] Fu-Hsiung Lai et al. "SUSTAIN -AN EPA BMP PROCESS AND PLACEMENT TOOL FOR URBAN WATERSHEDS." In: *Proceedings of the Water Environment Federation* 5 (2007), pp. 946–968.

[67] Guenter Langergraber, Diederik P L Rousseau, Joan García, and Javier Mena. "CWM1: A general model to describe biokinetic processes in subsurface flow constructed wetlands." In: *Water Science and Technology* 59.9 (2009), pp. 1687–1697.

[68] Günter Langergraber. "Modeling of Processes in Subsurface Flow Constructed Wetlands: A Review." In: *Vadose Zone Journal* · 7.2 (2008), pp. 830–842.

[69] Kevin Uhrmacher Laris Karklis and John Muyskens. *Before-and-after photos of Harvey flooding in Texas - Washington Post*. 2017.

[70] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. "Exploring strategies for training deep neural networks." In: *Journal of machine learning research* 10.Jan (2009), pp. 1–40.

[71] Ric Lawson, Elizabeth Riggs, Debi Weiker, and Jonathan Doubek. *Total Suspended Solids Reduction and Implementation Plan for Malletts Creek: October 2011 - September 2016*. Tech. rep. Huron River Watershed Council, 2016.

[72] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 0028-0836. DOI: 10.1038/nature14539. URL: http://www.nature.com/articles/nature14539.

[73] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning." In: *arXiv preprint arXiv:1509.02971* (2015).

[74] X. Litrico and V. Fromion. "Simplified Modeling of Irrigation Canals for Controller Design." In: *Journal of Irrigation and Drainage Engineering* 130.5 (Oct. 2004), pp. 373–383. DOI: 10.1061/(asce)0733-9437(2004)130:5(373).

[75] Nadia Schou Vorndran Lund, Anne Katrine Vinther Falk, Morten Borup, Henrik Madsen, and Peter Steen Mikkelsen. "Model predictive control of urban drainage systems: A review and perspective towards smart real-time water management." In: *Critical reviews in environmental science and technology* 48.3 (2018), pp. 279–339.

[76] MATLAB. *R2016a)*. Natick, Massachusetts: The MathWorks Inc., 2016.

[77] Mahmood Mahmoodian, Orianne Delmont, and Georges Schutz. "Pollution-based model predictive control of combined sewer networks, considering uncertainty propagation." In: *International Journal of Sustainable Development and Planning* 12.01 (Jan. 2017), pp. 98–111. ISSN: 1743-7601. DOI: 10.2495/SDP-V12-N1-98-111.

[78] Magdalene Marinaki and Markos Papageorgiou. "Linear-quadratic regulators applied to sewer network flow control." In: *2003 European Control Conference (ECC)*. IEEE, Sept. 2003. DOI: 10.23919/ecc.2003.7085327.

[79] Elbys Meneses et al. "Coordinating Rule-Based and System-Wide Model Predictive Control Strategies to Reduce Storage Expansion of Combined Urban Drainage Systems: The Case Study of Lundtofte, Denmark." In: *Water* 10.1 (Jan. 2018), p. 76. ISSN: 2073-4441. DOI: 10.3390/w10010076.

[80] A. M. Michalak et al. "Record-setting algal bloom in Lake Erie caused by agricultural and meteorological trends consistent with expected future conditions." In: *Proceedings of the National Academy of Sciences* 110.16 (Apr. 2013), pp. 6448–6452. ISSN: 0027-8424. DOI: 10.1073/pnas.1216006110.

[81] John R Middleton and Michael E Barrett. "Water Quality Performance of a Batch-Type Stormwater Detention Basin." In: *Water Environment Research* 80.2 (2008), pp. 172–178.

[82] John R Middleton and Michael E Barrett. "Water Quality Performance of a Batch-Type Stormwater Detention Basin." In: *Water Environment Research* 80.2 (Feb. 2008), pp. 172–178. DOI: 10.2175/106143007x220842.

[83] Cynthia Mitchell and Dennis McNevin. "Alternative analysis of BOD removal in subsurface flow constructed wetlands employing Monod kinetics." In: *Water Research* 35.5 (2001), pp. 1295–1303. ISSN: 00431354.

[84] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning." In: *Nature* 518.7540 (2015), pp. 529–533. ISSN: 0028-0836. DOI: 10.1038/nature14236. URL: http://arxiv.org/abs/1312.5602.

[85] A. L. Mollerup, P. S. Mikkelsen, D. Thornberg, and G. Sin. "Controlling sewer systems – a critical review based on systems in three EU cities." In: *http://dx.doi.org/10.1080/1573062X.2016.1148183* (2016).

[86] Luis A. Montestruque. "An agent-based storm water management system." In: *Smart Water Grids: A Cyber-Physical Systems Approach*. Ed. by Panagiotis Tsakalides, Athanasia Panousopoulou, Grigorios Tsagkatakis, and Luis Montestruque. CRC Press, 2018. Chap. 6, pp. 151–168.

[87] Luis Montestruque and M. D. Lemmon. "Globally Coordinated Distributed Storm Water Management System." In: *Proceedings of the 1st ACM International Workshop on Cyber-Physical Systems for Smart Water Networks - CySWater'15*. ACM Press, 2015. DOI: 10.1145/2738935.2738948.

[88] Abhiram Mullapudi, Brandon P. Wong, and Branko Kerkez. "Emerging investigators series: building a theory for smart stormwater systems." In: *Environ. Sci.: Water Res. Technol.* 3.1 (2017). ISSN: 2053-1400. DOI: 10.1039/C6EW00211K.

[89] Ronald K. Munson et al. "Model Prediction of the Effects of Changing Phosphorus Loads on the Everglades Protection Area." In: *Water, Air, and Soil Pollution* 134.1/4 (2002), pp. 255–272. ISSN: 00496979.

[90] Dirk Muschalla et al. "Ecohydraulic-driven real-time control of stormwater basins." In: *Journal of Hydrology* 511 (2014), pp. 82–91. ISSN: 00221694.

[91] Andrew Y Ng, Daishi Harada, and Stuart Russell. "Policy invariance under reward transformations: Theory and application to reward shaping." In: *ICML*. Vol. 99. 1999, pp. 278–287.

[92] Andrew Y Ng et al. "Autonomous inverted helicopter flight via reinforcement learning." In: *Experimental Robotics IX*. Springer, 2006, pp. 363–372.

[93] Gü Nter Langergraber and Jirka Š Imů Nek. "Modeling Variably Saturated Water Flow and Multicomponent Reactive Transport in Constructed Wetlands." In: *Vadose Zone Journal* 4.4 (2005), pp. 924–938.

[94] Carlos Ocampo-Martinez. *Model predictive control of wastewater systems*. Springer, 2010, pp. 1–236. DOI: 10.1007/978-1-84996-353-4.

[95] Katsuhiko Ogata. *Modern Control Engineering*. 5th ed. Prentice Hall, 2011. ISBN: 9780136156734.

[96] OpenAI. *OpenAI Five*. 2018.

[97] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. "Deep exploration via bootstrapped DQN." In: *Advances in neural information processing systems*. 2016, pp. 4026–4034.

[98] Y. Ouyang, P. Nkedi-Kizza, Q.T. Wu, D. Shinde, and C.H. Huang. "Assessment of seasonal variations in surface water quality." In: *Water Research* 40.20 (2006), pp. 3800–3810. ISSN: 00431354.

[99] T.G. Pálfy and G. Langergraber. "The verification of the Constructed Wetland Model No. 1 implementation in HYDRUS using column experiment data." In: *Ecological Engineering* 68 (2014), pp. 105–115. ISSN: 09258574.

[100] The European Parliament and the council of European Union. "Directive 2000/60/EC of the European Parliament and of the Council of 23 October 2000 establishing a framework for Community action in the field of water policy." In: *Official Journal of the European Communities* (2000), pp. 01–73.

[101] Jesper Persson and Hans B Wittgren. "How hydrological and hydraulic conditions affect performance of ponds." In: *Ecological Engineering* 21.4 (2003), pp. 259–269. ISSN: 09258574.

[102] Guido Petrucci, Jose-Frederic Deroubaix, and Bruno Tassin. "Urban stormwater source control policies: why and how?" In: *CWRS 2014: Evolving Water Resources Systems: Understanding, Predicting and Managing Water-Society Interactions* 364 (2014), pp. 1–7.

[103] Guido Petrucci, Emilie Rioust, José-Frédéric Deroubaix, and Bruno Tassin. "Do stormwater source control policies deliver the right hydrologic outcomes?" In: *Journal of Hydrology* 485 (2013), pp. 188–200.

[104] Martin Pleau, Hubert Colas, Pierre Lavallee, Geneviève Pelletier, and Richard Bonin. "Global optimal real-time control of the Quebec urban drainage system." In: *Environmental Modelling & Software* 20.4 (2005), pp. 401–413.

[105] Fred Powledge. "Chesapeake Bay restoration: a model of what?" In: *BioScience* 55.12 (2005), pp. 1032–1038.

[106] K R Reddy, W H Patrick, and C W Lindau. "Nitrification-Denitrification at the Plant Root-Sediment Interface in Wetlands." In: *Source: Limnology and Oceanography Limnol. Oceanogr* 34.346 (1989), pp. 1004–1013.

[107] G Ria, J Barreiro-Gomez, A Ramirez-Jaime, N Quijano, and C Ocampo-Martinez. "MatSWMM e An open-source toolbox for designing real-time control of urban drainage systems." In: *Environmental Modelling & Software* 83 (2016), pp. 143–154.

[108] G. Riaño-Briceño, J. Barreiro-Gomez, A. Ramirez-Jaime, N. Quijano, and C. Ocampo-Martinez. "MatSWMM – An open-source toolbox for designing real-time control of urban drainage systems." In: *Environmental Modelling & Software* 83 (Sept. 2016), pp. 143–154. ISSN: 1364-8152. DOI: 10.1016/J.ENVSOFT.2016.05.009.

[109] A. Rizzo et al. "Modelling the response of laboratory horizontal flow constructed wetlands to unsteady organic loads with HYDRUS-CWM1." In: *Ecological Engineering* 68 (2014), pp. 209–213. ISSN: 09258574.

[110] David Roman, Andrea Braga, Nandan Shetty, and Patricia Culligan. "Design and Modeling of an Adaptively Controlled Rainwater Harvesting System." In: *Water* 9.12 (Dec. 2017), p. 974. DOI: 10.3390/w9120974.

[111] L. Rossman. *Storm Water Management Model User's Manual Version 5.1 - Manual*. EPA/600/R-14/413 (NTIS EPA/600/R-14/413b). US EPA Office of Research and Development. Washington, DC, 2015.

[112] L. Rossman. *Storm Water Management Model Reference Manual Volume II – Hydraulics*. EPA/600/R-17/111. US EPA Office of Research and Development. Washington, DC, 2017.

[113] L. Rossman and W. Huber. *Storm Water Management Model Reference Manual Volume I – Hydrology*. EPA/600/R-15/162A. US EPA Office of Research and Development. Washington, DC, 2015.

[114] L. Rossman and W. Huber. *Storm Water Management Model Reference Manual Volume III – Water Quality*. EPA/600/R-16/093. US EPA Office of Research and Development. Washington, DC, 2016.

[115] Lewis A Rossman. *Storm Water Management Model User's Manual Version 5.1*. US Environmental Protection Agency, 2010.

[116] Jeffrey M. Sadler et al. "Leveraging open source software and parallel computing for model predictive control of urban drainage systems using EPA-SWMM5." In: *Environmental Modelling and Software* (2019). DOI: 10.1016/j.envsoft.2019.07.009.

[117] Jérémie Sage, Emmanuel Berthier, and Marie Christine Gromaire. "Stormwater Management Criteria for On-Site Pollution Control: A Comparative Assessment of International Practices." In: *Environmental Management* 56.1 (2015), pp. 66–80.

[118] Joseph Schilling and Jonathan Logan. "Greening the rust belt: A green infrastructure model for right sizing America's shrinking cities." In: *Journal of the American Planning Association* 74.4 (2008), pp. 451–466.

[119] Wolfgang Schilling. "Real-time control of urban drainage systems: The state-of-the-art." In: *IAWPRC Task Group on Real-Time Control of Urban Drainage Systems, London* (1989).

[120] Lian Scholes, D. Michael Revitt, and J. Bryan Ellis. "A systematic approach for the comparative assessment of stormwater pollutant removal potentials." In: *Journal of Environmental Management* 88.3 (2008), pp. 467–478. ISSN: 03014797.

[121] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms." In: *arXiv preprint arXiv:1707.06347* (2017).

[122] M. Schütze, V. Erbe, U. Haas, M. Scheer, and M. Weyand. "Sewer system real-time control supported by the M180 guideline document." en. In: *Urban Water Journal* 5.1 (Mar. 2008), pp. 69–78. ISSN: 1573-062X, 1744-9006. DOI: 10.1080/15730620701754376. (Visited on 01/29/2020).

[123] Manfred Schütze, Alberto Campisano, Hubert Colas, Wolfgang Schilling, and Peter A. Vanrolleghem. "Real time control of urban wastewater systems—where do we stand today?" In: *Journal of Hydrology* 299.3 (2004), pp. 335–348. ISSN: 00221694. DOI: 10.1016/j.jhydrol.2004.08.010.

[124] Manfred Schütze, Maja Lange, Michael Pabst, and Ulrich Haas. "Astlingen – a benchmark for real time control (RTC)." en. In: *Water Science and Technology* 2017.2 (May 2018), pp. 552–560. ISSN: 0273-1223, 1996-9732. DOI: 10.2166/wst.2018.172. (Visited on 05/20/2019).

[125] US Scs. "Urban Hydrology for Small Watersheds, Technical Release No. 55 (TR-55)." In: *US Department of Agriculture, US Government Printing Office, Washington, DC* (1986).

[126] Katja Seggelke, Roland Löwe, Thomas Beeneken, and Lothar Fuchs. "Implementation of an integrated real-time control system of sewer system and waste water treatment plant in the city of Wilhelmshaven." In: *http://dx.doi.org/10.1080/1573062X.2013.820331* 10.5 (2013), pp. 330–341.

[127] Yosef Sheffi. *Urban transportation networks*. Prentice Hall, 1984. ISBN: 0139397299.

[128] Author H L Shepherd, George Tchobanoglous, and M E Grismer. "Time-Dependent Retardation Model for Chemical Oxygen Demand Removal in a Subsurface- Flow Constructed Wetland for Winery Wastewater Treatment All use subject to JSTOR Terms and Conditions Time-Dependent Chemical Oxygen for Winery Retardation Demand Constr." In: *Water Environment Research* 73.5 (2001), pp. 597–606. ISSN: 10614303.

[129] Shadab Shishegar, Sophie Duchesne, and Geneviève Pelletier. "Optimization methods applied to stormwater management problems: A review." In: *Urban Water Journal* 15.3 (2018), pp. 276–286.

[130] David Silver et al. "Mastering chess and shogi by self-play with a general reinforcement learning algorithm." In: *arXiv preprint arXiv:1712.01815* (2017).

[131] David Silver et al. "Mastering the game of Go without human knowledge." In: *Nature* 550.7676 (2017), p. 354.

[132] Donald W Stanley. "Pollutant Removal by a Stormwater Dry Detention Pond." In: *Source: Water Environment Research* 68.6 (1996), pp. 1076–1083.

[133] Thomas Stocker. *Climate change 2013: the physical science basis: Working Group I contribution to the Fifth assessment report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, 2014.

[134] Congcong Sun et al. "An MPC-enabled SWMM implementation of the Astlingen RTC benchmarking network." In: *Water* 12.1034 (2020), pp. 1–13. DOI: 10.3390/w12041034.

[135] R Sutton and A Barto. *Reinforcement learning*. MIT Press, Cambridge, 1998.

[136] Richard S Sutton. "Planning by incremental dynamic programming." In: *Machine Learning Proceedings 1991*. Elsevier, 1991, pp. 353–357.

[137] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. "Policy gradient methods for reinforcement learning with function approximation." In: *Advances in neural information processing systems*. 2000, pp. 1057–1063.

[138] E. D. Tillinghast, W. F. Hunt, G. D. Jennings, and Patricia D'Arconte. "Increasing Stream Geomorphic Stability Using Storm Water Control Measures in a Densely Urbanized Watershed." In: *Journal of Hydrologic Engineering* 17.12 (2012), pp. 1381–1388. eprint: http://dx.doi.org/10.1061/(ASCE)HE.1943-5584.0000577.

[139] A Sezin Tokar and Peggy A Johnson. "Rainfall-runoff modeling using artificial neural networks." In: *Journal of Hydrologic Engineering* 4.3 (1999), pp. 232–239.

[140] Paul D. Trotta, J. W. Labadie, and N. S. Grigg. "Automatic control strategies for urban stormwater." In: *Journal of the Hydraulics Division* (1977), pp. 1443–1459.

[141] Ben Urbonas. "ASSESSMENT OF STORMWATER BMPs AND THEIR TECHNOLOGY." In: *Sci. Tech* 29.12 (1994), pp. 347–353.

[142] Peter-Jules Van Overloop. *Model predictive control on open water systems*. IOS Press, 2006.

[143] Peter A. Vanrolleghem, Lorenzo Benedetti, and J. Meirlaen. "Modelling and real-time control of the integrated urban wastewater system." In: *Environmental Modelling and Software* 20.4 (2005), pp. 427–442. DOI: 10.1016/j.envsoft.2004.02.004.

[144] Luca Vezzaro and Morten Grum. "A generalised Dynamic Overflow Risk Assessment (DORA) for real time control of urban drainage systems." In: *Journal of Hydrology* 515 (2014), pp. 292–303. DOI: 10.1016/j.jhydrol.2014.05.019.

[145] Christopher J. C. H. Watkins and Peter Dayan. "Q-learning." In: *Machine Learning* 8.3-4 (May 1992), pp. 279–292. ISSN: 0885-6125. DOI: 10.1007/BF00992698.

[146] Susan B. Watson et al. "The re-eutrophication of Lake Erie: Harmful algal blooms and hypoxia." In: *Harmful Algae* 56 (June 2016), pp. 44–66. ISSN: 1568-9883. DOI: 10.1016/J.HAL.2016.04.010.

[147] John R. White and K. R. Reddy. "Biogeochemical Dynamics I: Nitrogen Cycling in Wetlands." In: *The Wetlands Handbook* (2009), pp. 213–227.

[148] Steve Wise. "Green infrastructure rising." In: *Planning* 74.8 (2008), pp. 14–19.

[149] BP Wong and B Kerkez. "Real-time control of urban headwater catchments through linear feedback: performance, analysis and site selection." In: *Water Resources Research* 54 (2018), pp. 7309–7330. DOI: 10.1029/2018WR022657.

[150] Brandon P. Wong and Branko Kerkez. "Adaptive measurements of urban runoff quality." In: *Water Resources Research* 52.11 (2016), pp. 8986–9000. ISSN: 1944-7973. DOI: 10.1002/2015WR018013.

[151] Brandon P. Wong and Branko Kerkez. "Real-time environmental sensor data: An application to water quality using web services." In: *Environmental Modelling & Software* 84 (Oct. 2016), pp. 505–517. DOI: 10.1016/j.envsoft.2016.07.020.

[152] Tony H F Wong, Tim D Fletcher, Hugh P Duncan, John R Coleman, and Graham A Jenkins. "A Model for Urban Stormwater Improvement Conceptualisation." In: *Global Solutions for Urban Drainage* (2002), pp. 1–14.

[153] Tony H F Wong, Tim D. Fletcher, Hugh P. Duncan, and Graham A. Jenkins. "Modelling urban stormwater treatment-A unified approach." In: *Ecological Engineering* 27.1 (2006), pp. 58–70. ISSN: 09258574.

[154] Jeffrey Wright and Dayton Marchese. "Briefing: Continuous monitoring and adaptive control: the 'smart' storm water management solution." In: *Proceedings of the Institution of Civil Engineers - Smart Infrastructure and Construction* 170.4 (Dec. 2017), pp. 86–89. DOI: 10.1680/jsmic.17.00017.

[155] Zhiguo Yuan et al. "Sweating the assets - The role of instrumentation, control and automation in urban water systems." In: *Water Research* 155 (2019), pp. 381–402. DOI: 10.1016/j.watres.2019.02.034.

[156] Xiao-Yue "Jenny" Zhen, Shaw L. Yu, and Jen-Yang Lin. "Optimal Location and Sizing of Stormwater Basins at Watershed Scale." In: *Journal of Water Resources Planning and Management* 130.4 (2004), pp. 339–347. ISSN: 0733-9496.

DECLARATION

Put your declaration here.

Ann Arbor, June 2020

Abhiram Mullapudi

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

Thank you very much for your feedback and contribution.