Flood mitigation in coastal urban catchments using real-time stormwater infrastructure control and reinforcement learning

Benjamin D. Bowes, Arash Tavakoli, Cheng Wang, Arsalan Heydarian, Madhur Behl, Peter A. Beling and Jonathan L. Goodall

ABSTRACT

Flooding in coastal cities is increasing due to climate change and sea-level rise, stressing the traditional stormwater systems these communities rely on. Automated real-time control (RTC) of these systems can improve performance, and creating control policies for smart stormwater systems is an active area of study. This research explores reinforcement learning (RL) to create control policies to mitigate flood risk. RL is trained using a model of hypothetical urban catchments with a tidal boundary and two retention ponds with controllable valves. RL's performance is compared to the passive system, a model predictive control (MPC) strategy, and a rule-based control strategy (RBC). RL learns to proactively manage pond levels using current and forecast conditions and reduced flooding by 32% over the passive system. Compared to the MPC approach using a physics-based model and genetic algorithm, RL achieved nearly the same flood reduction, just 3% less than MPC, with a significant 88× speedup in runtime. Compared to RBC, RL was able to quickly learn similar control strategies and reduced flooding by an additional 19%. This research demonstrates that RL can effectively control a simple system and offers a computationally efficient method that could scale to RTC of more complex stormwater systems.

Key words | real-time control, reinforcement learning, smart stormwater systems, urban flooding

Benjamin D. Bowes
Arash Tavakoli
Cheng Wang
Arsalan Heydarian
Madhur Behl
Peter A. Beling
Jonathan L. Goodall (corresponding author)
Department of Engineering Systems and Environment,
University of Virginia, P.O. Box 400747,
Charlottesville, VA 22904,
USA
E-mail: *goodall@virginia.edu*

HIGHLIGHTS

- Reinforcement learning (RL) creates policies for real-time coastal stormwater system control.
- RL's ability to mitigate flooding and manage ponds is compared to a passive system, model predictive control, and rule-based control.
- RL was more efficient than model predictive control using a physics-based model and genetic algorithm.
- RL's ability to mitigate flooding exceeded the passive system and rule-based control.

doi: 10.2166/hydro.2020.080

INTRODUCTION

As the frequency and intensity of storms increase due to changes in climate, the ability of existing stormwater infrastructure to mitigate urban flooding is being increasingly stressed (Mynett and Vojinovic 2009; Berggren *et al.* 2012; Neumann *et al.* 2015; Wuebbles *et al.* 2017; Mounce *et al.* 2020). In coastal cities, gravity-driven stormwater systems are critical for flood management, but their functionality is also being reduced by sea-level rise (Sadler *et al.* 2020). These stressors, combined with the flat, low elevation topography of many coastal cities, mean that these communities are already experiencing increased flooding during high tide events (Sweet and Park 2014; Moftakhari *et al.* 2015, 2017).

Advances in urban hydroinformatics (Mynett and Vojinovic 2009), including smart stormwater systems (Kerkez *et al.* 2016), provide promising tools and approaches to improve stormwater system performance in coastal communities. In the smart stormwater system approach, existing stormwater systems are retro-fitted with real-time sensors and actuators (e.g., remotely controlled valves and pumps) to allow active monitoring and control. Active control is a cost-effective way to more efficiently use the existing capacity of stormwater infrastructure (Jose Meneses *et al.* 2018). In addition, active control can allow a system to function as a whole, which can be much more effective than a piece-wise capital improvement of passive infrastructure systems (Wong and Kerkez 2018).

Key to the effectiveness of active systems is the use of real-time control (RTC) (Schwanenberg *et al.* 2015; Kerkez *et al.* 2016; Mounce *et al.* 2020). RTC uses streaming sensor data (i.e., current rainfall and retention pond depths) to approximate the current system states. The system state can then be used to inform changes to control assets (e.g., valves, pumps) that adapt the behavior of the system to current or forecast conditions. The decisions on when and what structures to control, and how to change them in order for a system to meet certain objectives (e.g., minimize flooding, maintain certain flow conditions), are based on a control policy. In a smart stormwater system, control policies map system states, such as water levels in a pipe network, to actions that need to be taken in order to meet management objectives (Sadler *et al.* 2019). In current practice, control policies are often predefined simple heuristics, such as opening a valve when a storage pond reaches a certain depth (level control or feedback control) and may be based on the experience of a human operator (García *et al.* 2015; Abou Rjeily *et al.* 2018). This heuristic approach may be effective in simple systems (i.e., a system with only a few controlled assets); however, it requires that control actions are predefined for all scenarios and becomes increasingly difficult as the number of assets grows and/or more external factors start influencing the system.

Heuristic control can be improved by incorporating some aspects of feedback control (i.e., system observations) and feed-forward control (i.e., forecasts and predicted system states) (Schwanenberg *et al.* 2015), termed rule-based control (RBC) in this research. RBC can be implemented in stormwater systems based on watershed characteristics and forecast rainfall data in order to meet flooding, water quality, and/or channel protection objectives (Marchese *et al.* 2018). By continuously monitoring retention pond depths and rainfall forecasts, inflow from storm events can be estimated using simple rainfall–runoff models and used to inform control decisions. For example, if a storm event has been forecast, the available volume in a pond can then be adjusted based on the estimated inflow from the storm event. This adjustment is made by actuating a valve at the pond's outlet that can be opened to drain water until the necessary storage volume in the pond is reached. RBC is intuitive to end-users and can be effective for controlling individual stormwater system components (OptiRTC and Geosyntec Consultants Inc. 2017). However, as system complexity increases (e.g., releases of water need to be coordinated if multiple ponds drain to the same downstream location in order to prevent flooding), RBC becomes more difficult to implement.

Two approaches for finding control policies for RTC beyond simple heuristics and RBC include model predictive control (MPC) and reinforcement learning (RL). MPC uses a process model to evaluate various control strategies for a specified control horizon (Camacho and Bordons 2007; García *et al.* 2015). It has been successfully applied to large-scale water systems for multi-objective optimization

(Schwanenberg *et al.* 2015) and for stormwater system control (Sadler *et al.* 2019, 2020). Sadler *et al.* (2019) examined the effectiveness of MPC for a tidally influenced stormwater system and successfully found control policies that reduced flooding compared to the passive system. This particular formulation of MPC used a physics-based stormwater system model optimized with a genetic algorithm. Using high-performance and cloud-based computing, the authors were able to speed up this computationally expensive MPC formulation. However, as a limitation of this approach, the authors indicate that for large real-world stormwater systems, such MPC techniques could be prohibitively slow for RTC. When scaled to part of a real-world stormwater system, Sadler *et al.* (2020) were able to run MPC for three simulated actuators and determined that RTC could help reduce flooding for future sea-level rise scenarios.

RL is a category of machine learning that aims to learn from trial-and-error experience by interacting with an environment (Sutton and Barto 2018). An RL agent does not have known answers to learn from but instead tries to maximize the amount of reward it can receive from its environment by taking certain actions. One of the differences between RL and MPC is that RL can learn control policies offline, which moves the computational burden to prior to taking any actions. The use of RL in water resources engineering has been compared with Dynamic Programming (DP) for multi-objective reservoir management (Lee and Labadie 2007; Castelletti *et al.* 2013, 2014; Pianosi *et al.* 2013; Delipetrev *et al.* 2017). These studies used tabular RL methods, which can be more computationally efficient than DP but are constrained to simple systems (i.e., a small number of potential system states and actions) and demonstrated that a physics-based model could act as the environment for RL agents to train on. Recent advances in deep learning have allowed RL to use neural networks as function approximators to overcome the limitations of tabular RL. For instance, Mullapudi and Kerkez (2018) demonstrated control of a stormwater system with a Deep Q-Network (DQN) (Mnih *et al.* 2015). By throttling valves, they were able to make control decisions for a discrete action space (a limitation of DQN). Their work also highlighted how rewards can be shaped for real-time stormwater control through deep RL and illustrated how RTC with RL can increase a stormwater system's effective capacity. However, their DQN agent was only reactive to the current conditions of the stormwater system; effective RTC strategies should be based on both the current conditions and forecasts. Additionally, discretizing control actions may not always provide the most efficient policy. Further research is needed, therefore, to determine if RL control can be further refined using a continuous action space that allows any valve position to be used and create forecast-based predictive control policies.

In this paper, RL is used to create control policies for RTC of a tidally influenced stormwater system. In addition to presenting the first work exploring RL for RTC of tidally influenced stormwater systems, we advance on prior work in smart stormwater by exploring the Deep Deterministic Policy Gradient (DDPG) RL algorithm to control valves over a continuous action space, allowing the agent to learn more refined control policies. Additionally, forecasts of rainfall and tide are included as part of the state information received by the RL agent, allowing the agent to learn proactive control strategies. The RL agent's performance is compared to (i) a passive, gravity-driven system (ii) MPC (as implemented by Sadler *et al.* (2019)), and (iii) RBC. Through this comparison, we illustrate the applicability of RL for RTC of a simulated coastal urban stormwater system.

METHODOLOGY

In this section, the simulated stormwater system is introduced, a hypothetical scenario for categorizing the impact of flood events is detailed, and each RTC method is described. The stormwater system is similar to that used by Sadler *et al.* (2019) and is the same for all scenarios (RL, Passive, MPC, RBC), except that retention ponds in the passive system have weirs at a fixed elevation to maintain a depth of water in the ponds while the ponds for the RTC scenarios have a controllable valve at the bottom of the pond side. An overview of the methodology used to compare the performance of an RL-controlled stormwater system with the other scenarios is provided in Table 1. Open-source code for these scenarios is available on github (Bowes 2020b). SWMM simulation files and data are available as a resource on HydroShare (Bowes 2020a).

Table 1 | Overview of methodology for stormwater system control scenarios

Method	Valve	Training/optimization	Control policy	Testing
RBC system	Controllable valve at 0 m	N/A	Fixed for simulation duration	Test on 2010–2019 data
MPC system	Controllable valve at 0 m	Online optimization with genetic algorithm	Adjusted valve positions for specified time horizon based on modeled scenarios and objective functions	Test on first week of August 2019 data (due to computational cost)
RL system	Controllable valve at 0 m	Offline training on August 2019 data for 197,000 control steps	Learned policy relating states to actions	Test on 2010–2019 data
Passive system	Fixed weir at 0.61 m	N/A	N/A	Test on 2010–2019 data

Stormwater simulation

Stormwater system simulations are carried out using the U.S. Environmental Protection Agency's Stormwater Management Model (SWMM), version 5. The hypothetical stormwater system used in this study is a simple model inspired by conditions within an urban catchment located in Norfolk, Virginia, USA. It consists of two subcatchments, two storage units (retention ponds), and pipes going to the system outfall that discharges to a tidally influenced waterbody (Figure 1). In the passive scenario, each pond has a weir at a fixed elevation and cannot be completely emptied. Infiltration and evaporation are excluded in this simple system as the focus is on flood control. Ponds in the RTC scenarios have been retro-fitted by removing the weirs and adding controllable valves (orifices) at the bottom of the pond side. This allows the full pond volume to act as active storage that can be adapted for different storm events. Input to this system is rainfall, which falls onto a subcatchment and is transformed into runoff that flows into a storage unit. Flow out of the storage units can be regulated by the valves; both ponds drain directly into a single node before flowing through two pipe segments to the outfall. The tail water condition at the outfall is influenced by the tide level. At high tide, the tail water condition can cause sea level to partially block the outfall, backing up stormwater drainage. The physical parameters of the SWMM model can be found in Table 2.

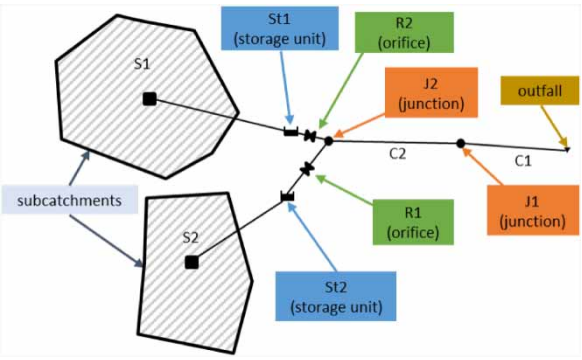

Figure 1 | SWMM simulation schema.

In this SWMM model, flooding can be caused by (i) rainfall, (ii) high tide, or (iii) a combination of rainfall and tide. Flooding caused by these factors can be in the form of the ponds over-topping or the downstream node J1, which can be thought of as a roadway storm drain, surcharging. Ponds can over-top if the subcatchment runoff volume for a storm event exceeds current pond capacity and inflow is greater than outflow. Flooding at the downstream node can occur if the flow from the ponds is not regulated and coordinated by adjusting the two valves. Node J1 can also flood if tidal conditions at the outfall are preventing the normal flow of water from the system or causing backflow if the tide is especially high.

The pyswmm (McDonnell *et al.* 2020) Python wrapper for SWMM is used for step-by-step running of simulations as needed for the RTC methods. Each control scenario can be updated once every 15 min (an adjustable control time step).

SWMM input data

Input data for the SWMM simulation come from observed data for stations in Norfolk, Virginia (Figure 2). Rainfall data are collected at a 15-min timestep by the Hampton

Table 2 | Properties of SWMM model

Subcatchment properties				
Name	Area (hectares)	Width (km)	Slope (%)	Impervious (%)
S1	32	0.46	0.5	30
S2	20	0.61	0.5	45
Storage unit properties				
Name	Surface area (m^2) (constant)	Initial depth (m)	Max depth (m)	Bottom Elev (m)
St1	6,039	0.61	1.41	0.91
St2	4,645	0.61	1.41	0.91
Pipe properties				
Name	Length (m)	Diameter (m)	Roughness	
C1, C2	122	0.3	0.01	
Node properties				
Name	Max depth (m)	Bottom Elev (m)		
J2	1.5	0.91		
J1	0.6	0.30		
Outfall	NA	0		
Orifice properties				
Name	Height (m)	Discharge coefficient		
R1 and R2	0.61	0.65		

Roads Sanitation District; rainfall data for subcatchment S1 is from gauge Rain1 and data for subcatchment S2 is from gauge Rain2. Rainfall data are cleaned through a number of checks. First, any values over the 1000-year, 15-min rainfall for Norfolk (59.18 mm) are assumed to be erroneous and removed. Next, missing data in each rainfall time series are replaced; if both rain gauges are missing values at the same time stamp, both get zero. Otherwise, if one station is missing a value but the other is not, the missing value is replaced with the known value from the other station. Observed tide level data come from the National Oceanic and Atmospheric Administration Sewells Point gauge and are measured at 6-min intervals. For use as an SWMM boundary condition, tidal data are resampled to an hourly interval and referenced to the North American Vertical Datum of 1988 (NAVD88). All of the observed data are for the period between 1 January 2010 and 6 November 2019 and are divided into individual months to make simulation run times tractable.

Forecasts were created from the observed data for use in the various control methods, therefore assuming perfect knowledge of future events. A single forecast, in this case, is an array of values representing the rainfall or tide measurement over the next n time steps. For example, a 24-h forecast of 15 min rainfall would contain 96 values. In future work, noise could be added to these forecasts to explore how RL (or any other RTC method) handles uncertainty, but this is beyond the scope of this research (for future directions, see Hartono and Hashimoto (2007) and van Andel *et al.* (2008, 2014)).

Flood event classification

In order to quantify the flooding impact, a hypothetical scenario is developed from physical data for Norfolk. In this scenario, the subcatchments considered are residential neighborhoods where any flooding of the ponds will impact roadway intersections. The downstream node J1 is considered a storm drain at a roadway intersection; flooding at this node will make the intersection impassable if the depth of the flood water is above a certain threshold. For this hypothetical scenario, 0.2 m of roadway flooding slows traffic considerably, the

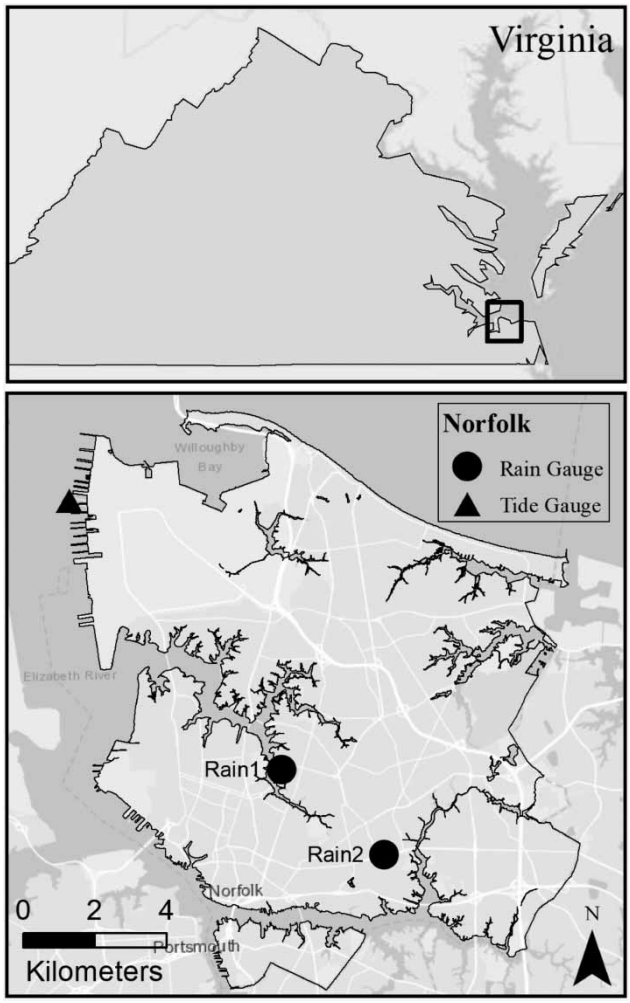

Figure 2 | Gauge locations in Norfolk, VA, USA.

threshold for safe passenger vehicle passage is 0.3 m (Pregnolato *et al.* 2017), and 0.4 m is the limit for safe emergency vehicle passage. The relationship between flood volume and depth was developed from digital elevation data from Norfolk as described in Appendix B. The number, volume, and duration of these flood events are used as an additional metric for quantifying flooding along with the total volume.

Implementing RL in stormwater systems

In RL, an agent learns to optimize its behavior by interacting with its environment (Sutton and Barto 2018). The environment is usually modeled as a Markov Decision Process: $\langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s'|s, a)$ is the stochastic probability of transitioning to a new state s' after taking action a at the current state s, $r(s, a, s')$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor that weighs the importance of short-term and long-term reward. The RL agent's goal is to find an optimal policy that maximizes the expected discounted return

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{\infty}^{k=0} \gamma^k r_{t+k} \tag{1}$$

where $r_t = r(s_t, a_t, s_{t+1})$.

In this paper, the states $\mathcal{S}$ are defined as the current depths and rate of flooding (if any) at the ponds and downstream nodes (St1, St2, J1), the current valve positions (R1, R2), the sum of the 24 h rainfall forecast (F) for each subcatchment, and the mean value of the 24 h tide forecast. These values are gathered from the SWMM simulation at each control time step. The actions $\mathcal{A}$ that the RL agent can take at any step are to close or open any valve to any degree. Finally, the reward r the RL agent receives in this system is based on how well the agent prevents flooding and maintains certain target pond water depths. It is defined as

$$r = \begin{cases} -\Sigma(\text{flooding}) & F > \delta \\ -\text{J1}_{\text{flooding}} - (|\text{St1}_{\text{depth}} - \text{target}| + |\text{St2}_{\text{depth}} - \text{target}|) & F = 0 \end{cases} \tag{2}$$

where flooding is the flooding rate at each particular node (St1, St2, J1) and δ represents a forecast rainfall threshold (>0 in this case); target is the target water depth for the storage ponds (St1 and St2). In this relatively simple stormwater system, the target depth is 0.61 m for both ponds. In a real system, different ponds would most likely have different target depths; this can be taken into account in the RL implementation by having different target depths for each pond in the reward function.

As an example, consider a case where the agent is in a specific state s in $\mathcal{S}$ (e.g., the water depth in a specific pond is 1.0 m) and takes an action a in $\mathcal{A}$ (e.g., completely opens the valve) with a probability given by the policy $\pi(a|s)$. The agent will then transition to a new state s' with a probability of $P^a_{s,s'} = P(s'|s, a)$ (e.g., the water depth in a specific pond is 0.75 m) and receives a reward $r(s, a, s')$.

The value of this action depends on the reward that the agent receives and the discounted value of all the future rewards if the agent follows the policy afterwards. Using a discount factor γ, the value of a future reward of x after n steps is $x\gamma^{n-1}$. The expected discounted return when starting in state s, then taking action a, and following π is called the Q-value function:

$$\begin{aligned} Q^{\pi}(s,a) &= E[G_t|s,a] \\ &= r(s,a,s') + \gamma \sum_{s' \in \mathcal{S}} P^{a}_{s,s'} \sum_{a' \in \mathcal{A}} \pi(a'|s') Q^{\pi}(s',a'), \end{aligned} \tag{3}$$

where the second equation is known as the Bellman equation (Sutton and Barto 2018).

By having the optimal Q-values, one can find the optimal policy by finding the specific actions in each state that give the maximum Q-value. However, due to the curse of dimensionality, this tabular type of Q-learning is limited to problems with relatively small state and action spaces. Recent advances in deep learning have been applied to RL to overcome this problem by using deep neural networks to approximate value functions instead of storing them in tables (Mnih *et al.* 2015).

In order to have an RL agent that can set the valves to any position over a continuous action space, the Deep Deterministic Policy Gradients (DDPG) (Lillicrap *et al.* 2015) actor-critic algorithm is used. DDPG uses deep neural networks to approximate a policy and the difference between policies, the gradient, is used to update the agent. In this case, the agent consists of two parts: an actor which represents the policy, and a critic which estimates the q-value of actions taken by the actor. During the training process, the actor is fed information on the state of the stormwater system and outputs the actions to be taken. These actions, along with the state information, are used as input to the critic. The actions and q-value estimates output from the critic are used to update the agent.

The keras-rl (Plappert 2016), openai gym (Brockman *et al.* 2016), and Tensorflow (Abadi *et al.* 2016) python packages are used to implement the DDPG algorithm for this research. Each part of the DDPG agent, the actor and the critic, is composed of a deep feed-forward neural network (Table 3). The hyperparameters of each neural network are determined by trial and error (Maier *et al.* 2010). Through experimentation, it can be found that training the RL agent on the August 2019 dataset and looping through the SWMM simulation approximately 100 times provided enough experience of a wide range of rainfall and tidal events for the agent to learn from. This month has a total of 256.54 mm of rainfall over seven events. The average tide level is 0.16 m with a maximum value of 1.01 m from Tropical Storm Erin late in the month. A visualization of these data is given in Figure 4. RL training and testing are carried out on a standard PC with 8 cores, 16GB RAM, and an NVIDIA Quadro P2000 Graphical Processing Unit.

Table 3 | DDPG RL agent architecture and hyperparameter settings

	Actor		Critic	
NN layer	**Neurons**	**Activation**	**Neurons**	**Activation**
Input	Current state s	N/A	Current state s and action a	N/A
Hidden 1	16	RELU	32	RELU
Hidden 2	16	RELU	32	RELU
Hidden 3	8	RELU	32	RELU
Output	1 [R1, R2]	Sigmoid	1 [q-value]	Linear

MPC settings

The swmm_mpc software developed by Sadler *et al.* (2019) is used to implement MPC for comparison with RL; readers are referred to this paper for full details on the MPC implementation. Briefly, swmm_mpc uses SWMM as a process model and an evolutionary algorithm to search for a control policy. At each time step in an SWMM simulation, swmm_mpc runs many variations of the SWMM simulation in order to determine which control actions minimize an objective function for a specified time horizon. In this case, the objective function is based on the amount of flooding and deviations from target water level depths as

$$\text{MPC objective function} = \alpha(\boldsymbol{a} \cdot \boldsymbol{v}(\boldsymbol{u}, \boldsymbol{x})) + \beta(\boldsymbol{b} \cdot \boldsymbol{d}(\boldsymbol{u}, \boldsymbol{x})) \tag{4}$$

where $\boldsymbol{v}$ and $\boldsymbol{d}$ are one-dimensional vectors of flood volumes at each node and deviations from target depths, respectively. The two-dimensional vectors $\boldsymbol{u}$ and $\boldsymbol{x}$ represent the control

policies for all controls and the system states, respectively. The user-defined parameters and their definitions are given in Table 4. The scalar multipliers α and β are overall weights for the cost of flooding and the cost of water-level deviations. These will be adjusted in order to optimize the MPC control.

Because of the computational expense of running swmm_mpc, where many variants of the SWMM model have to be executed at each simulated time step to find the best control actions, a high-performance computer (HPC) was used to run the software. The HPC computational resources consisted of 28 cores with a CPU speed of 2.4 GHz, an Intel Xeon processor, and 128 GB RAM.

Rule-based control

RBC was implemented based on documented industry-standard methods (OptiRTC and Geosyntec Consultants Inc. 2017; Wright and Marchese 2017; Marchese *et al.* 2018). In practice, this type of control uses forecasts of rainfall and watershed characteristics to inform the control of valves on stormwater assets (wet/dry ponds, bioswales) in order to meet flood control, water quantity, and/or quality objectives.

Because the current research is done on a simulated system, the expected flood volume from a forecast of rainfall, if any, is used to control the level of water in an individual pond. For example, if a forecast storm event is expected to cause 1,000 m^3 of flooding, the pond's outlet valve would open before the storm in order to drain out a corresponding volume of water plus a 20% safety factor. After the pond's depth is drawn down by the appropriate level, the valve can be closed to retain the incoming stormwater, which helps improve water quality. In this way, storm runoff should not flood the pond and will be retained to prevent flooding downstream. After a storm event, water can be held in the pond for a specified settling period (24 h in this case) and then slowly released (over 24 h) to bring the pond back to its standard operating range. Outside of storm events, rules can also be in place to maintain the pond level within the operating range or maintain certain flow conditions. The exact control rules and their hierarchy, as implemented in this research, are detailed in Figure 3.

Table 4 | MPC cost function parameters

Parameter (description)	Value
α (overall flood weight)	Scalar
a (individual node flood weight [St1, St2, J1, J2])	[1, 1, 1, 1]
β (overall deviation weight)	Scalar
b (individual deviation weight [St1, St2, J1, J2])	[1, 1, 0, 0]
Target depths (m)	[0.61, 0.61, NA, NA]

RESULTS

Comparison of RL and passive system

A comparison of the RL agent's policy against the passive system shows that the agent can learn to effectively control valves to maximize its reward. As indicated in Figure 4, the training data show valves are opened when rainfall is in the forecast, allowing additional storage space in the retention ponds. After a storm is over, valve positions are adjusted again in order to maintain a pond depth close to the target of 0.61 m. Following this policy allows the RL agent to reduce total flood volume for this month by approximately 70% (5,936 vs. 19,957 m^3) compared to the passive system. For example, in the first storm of August 2019, both valves (R1 and R2) are opened to drain water in response to the rainfall forecast (a detailed figure with this comparison is available in Appendix A, Figure 1). However, due to the difference in rainfall on the subcatchments, R2 closes earlier than R1 in order to maintain the target depth. Directly after storm events, the RL agent tries to balance returning the ponds to the target depth and preventing flooding downstream at J1. Because the RL agent also needs to maintain the target pond depths compared to the passive system, there is an increase in the number of events at the downstream node, despite these being minor events.

Applying the policy learned on the August 2019 training data to the test sets shows that this RL agent has learned a policy that works well in many other conditions (Appendix A, Figure 2). Compared to the passive system, total flooding was reduced by RL in 85 of the 120 months of data (71%). In particular, this policy works well on test sets with similar (e.g., August 2018) or larger (e.g., September 2016) amounts of flooding than the August 2019 training set (the mean total

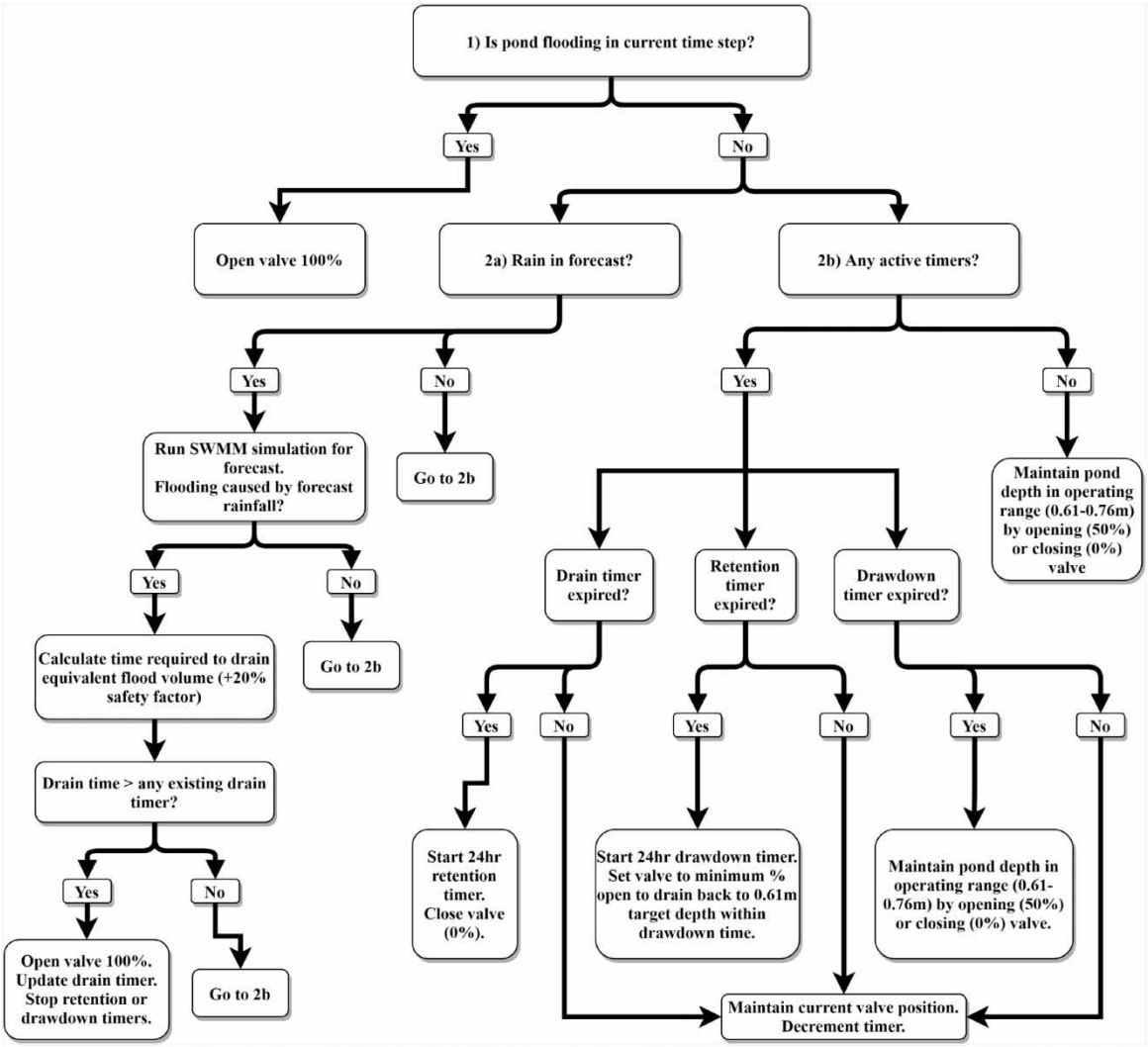

Figure 3 | Rule-based control hierarchy and settings.

flood volume for these months was 4,278 m^3). In a few cases (such as April 2019 or September 2017), the RL policy increases the amount of flooding. These are months with little or no flooding (mean of 606 m^3) and the agent has learned to respond to rainfall events in a manner that is not ideal for these months with less flood risk. The agent's performance on these months can be improved by increasing the threshold value for rainfall forecasts used in the conditional reward. For example, if the conditional reward threshold is increased from 0 to 1.3 mm of rain, the agent's performance on months with low flooding is improved (Appendix A, Figure 2, 'RL: 1.3'). However, this is at the expense of performance on the larger storms. Overall, the agent trained with the 1.3 mm threshold had lower flooding than the agent trained with the 0.0 mm threshold in 28 months (23% of the data), but increased flooding in the remaining 92 months (77%).

Comparison of RL and MPC

In order to make computational expense tractable, the MPC setup from Sadler *et al.* (2019) was only run using data from the first week in August 2018. However, due to the nature of this MPC formulation (online optimization using a genetic algorithm), computational times are still high. Finding an MPC policy for this week of data took almost 50 h (2 days,

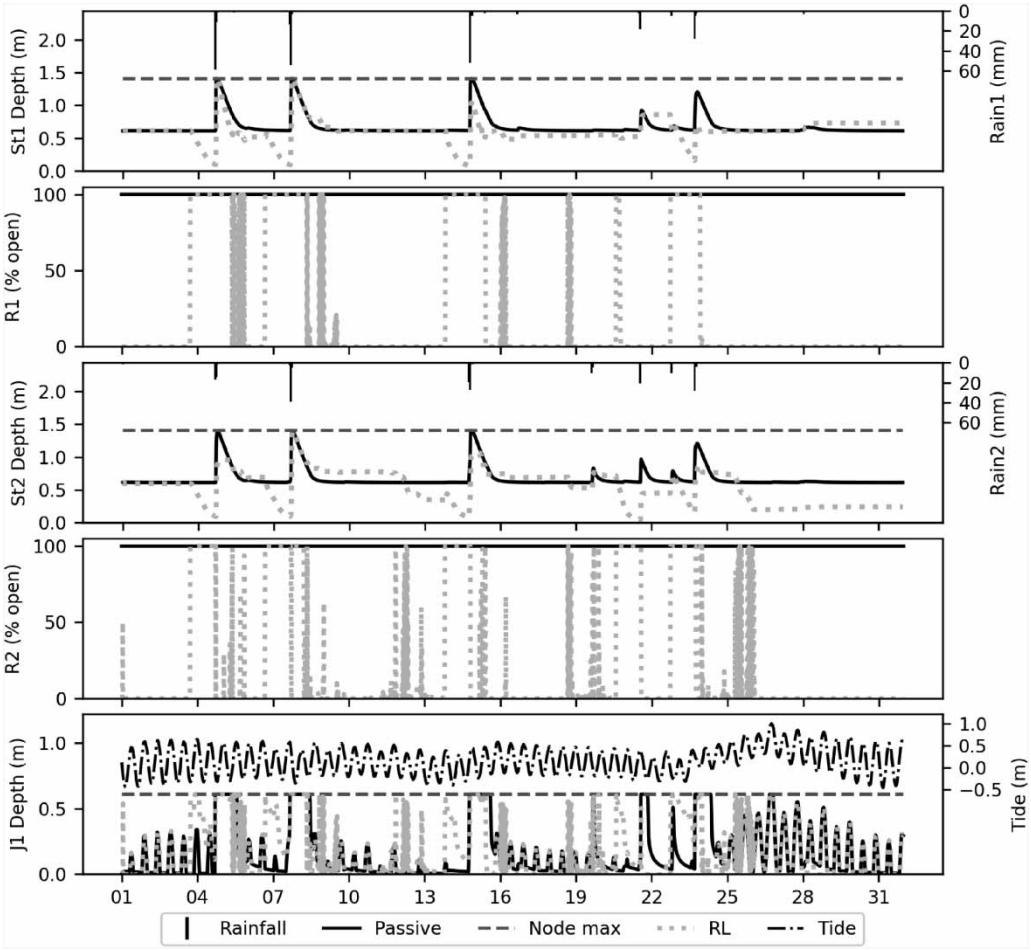

Figure 4 | Comparison of RL controlled and passive system performance on August 2019 training data.

1 h, and 48 min); computational time for each 15 min simulated control step is 3.9 min. This is tractable for a simple system but is partly a function of the simulation length and would increase with system complexity (Sadler *et al.* 2019).

To investigate MPC's performance for this specific dataset, several combinations of objective function weights were tried in order to prevent flooding and maintain target storage pond depths (Table 5). The best performing of these combinations is having the flood weight set to 1 and the deviation weight set to 10 (MPC4). This result was unexpected given that an even weighting seems like it would provide the best balance of flood mitigation and pond depth maintenance. Further, this MPC formulation was the only one in which both ponds were not kept empty for the dry periods in the simulation. A visualization of the policies carried out by MPC and RL shows that while MPC modulated orifice R2 and kept pond St2 close to the target depth, orifice R1 was slightly open and static for much of the time (Figure 5). This allowed pond St1 to essentially

Table 5 | MPC trials and performance comparison with the passive and RL systems for the first week in August 2019

Model	Alpha (overall flood weight)	Beta (overall deviation weight)	Control horizon (h)	Total flood volume (m^3)	Accumulated deviation (m)
Passive	N/A	N/A	N/A	13,586	126.3
MPC1	1,000.0	0.5	1.0	2,767	620.2
MPC2	0.75	0.25	1.0	2,582	614.5
MPC3	1.0	1.0	1.0	2,714	591.5
MPC4	1.0	10.0	0.5	3,929	439.2
RL	N/A	N/A	N/A	4,058	234.5

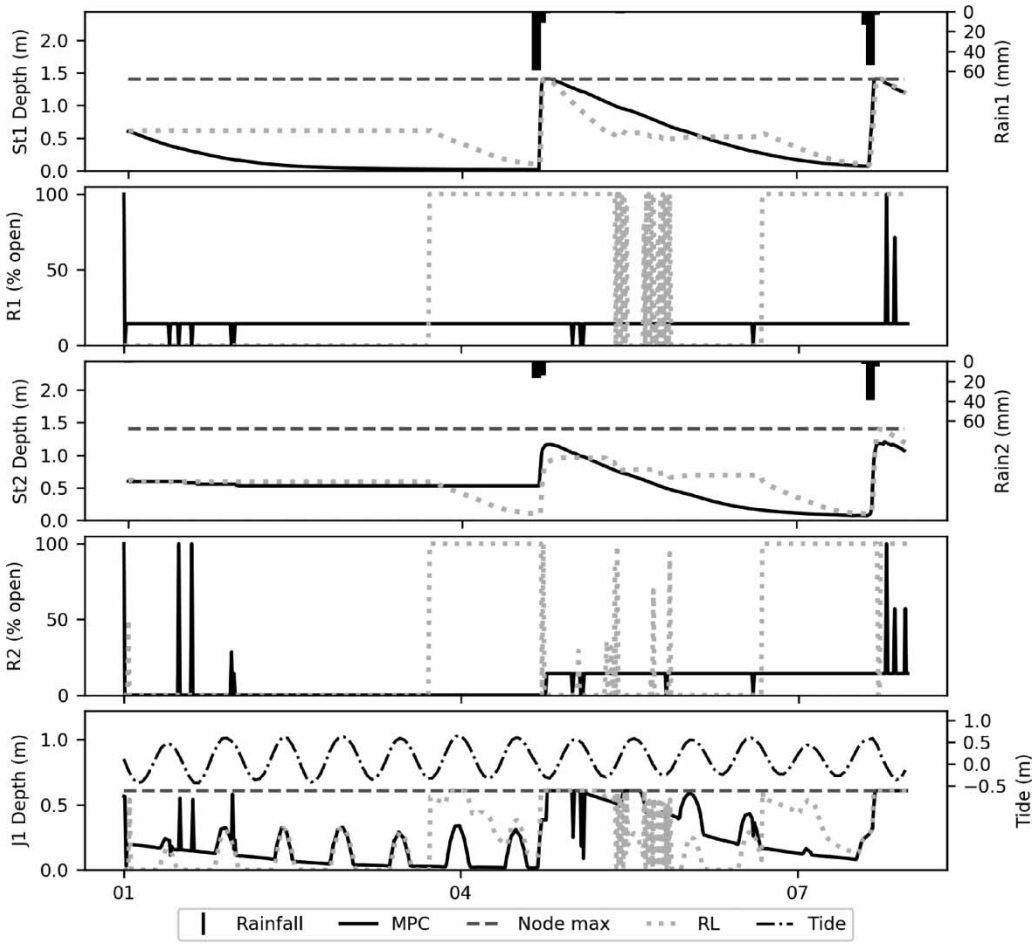

Figure 5 | RL and MPC control policies and states for the first week of August 2019.

empty during dry periods, which is an undesirable behavior. In contrast, RL was better at maintaining the target depth before the first storm and between the storm events.

Comparison of RL and RBC

RBC results were generated for the same monthly datasets used in the RL training and testing. RBC was able to reduce flooding by 57% for the month of August 2019 compared to the passive system (8,540 vs. 19,957 m^3). This method of control is also able to extend the retention time of stormwater in the ponds and maintain target pond depths during dry periods or small rainfall events. Because this RBC is based on rainfall forecasts (with perfect knowledge of future events), it is able to drawdown ponds prior to a storm event based on the expected flood volume (Figure 6).

The RL agent's performance on the August 2019 training data is similar to RBC but has an advantage in that the entire system state is used to inform control decisions, as opposed to using only the depth in the individual ponds (Figure 7). Because of this increased system knowledge and flexibility in its valve control settings, the RL agent was able to reduce flooding by 30% over RBC (5,936 vs. 8,540 m^3). The RBC logic is based on individual pond depths; conditions at other parts of the system (e.g., flooding downstream or tidal influence on the outfall) are only considered indirectly if they impact pond depth.

Over all the months of data, RBC reduced flooding over the passive system in 45 months (38%) (Figure 8). However, these were months with large total flood volumes (mean

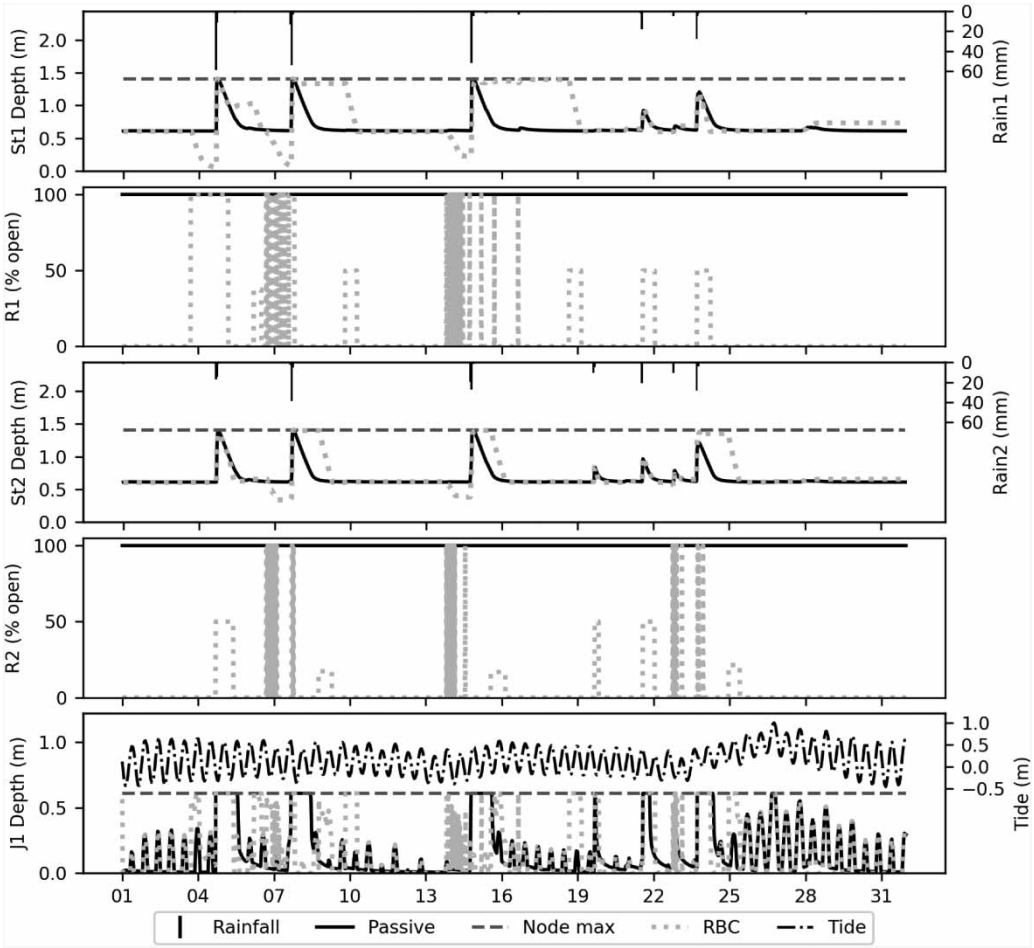

Figure 6 | Comparison of RBC and passive system performance on August 2019.

total flood volume of 8,559 m^3). Similar to RL, RBC performs less well on the months with little or no flooding of the passive system (e.g., February 2019, March 2014, March 2019) and a few months with more flooding (e.g., September 2018, October 2012). In comparison with RL, RBC had more or equal flooding in 101 months (84% of data, with a mean increase in flooding of 60%) and reduced flooding in the remaining 19 months (16% of data, with a mean decrease in flooding of 25%).

Examining the month of July 2010 shows a situation where RBC outperforms RL (see Appendix A, Figure 3). For this month, the difference in flooding between RL and RBC is relatively small (17,844 vs. 16,311 m^3, respectively) and both reduced flooding compared to the passive system (21,997 m^3), but RBC better maintained the target pond depths. This example illustrates the difficulty in shaping rewards for RL; because the conditional reward function is based on the rainfall forecast, very small amounts of rainfall will cause the agent to only be rewarded for preventing flooding. As mentioned in the RL/Passive system results, the rainfall threshold used in the reward function can influence this behavior. However, using a threshold value of >0 mm lets the agent keep pond St1 higher than the 0.61 m target depth. Because RBC maintained the target pond depths better than RL in this case, RBC did not have as much water to drain out of the ponds before the large storm event at the end of the month.

Examining the months where the passive system had lower total flooding than RBC helps illustrate its limitations. For these 75 months, the mean total flooding was 80 m^3. For

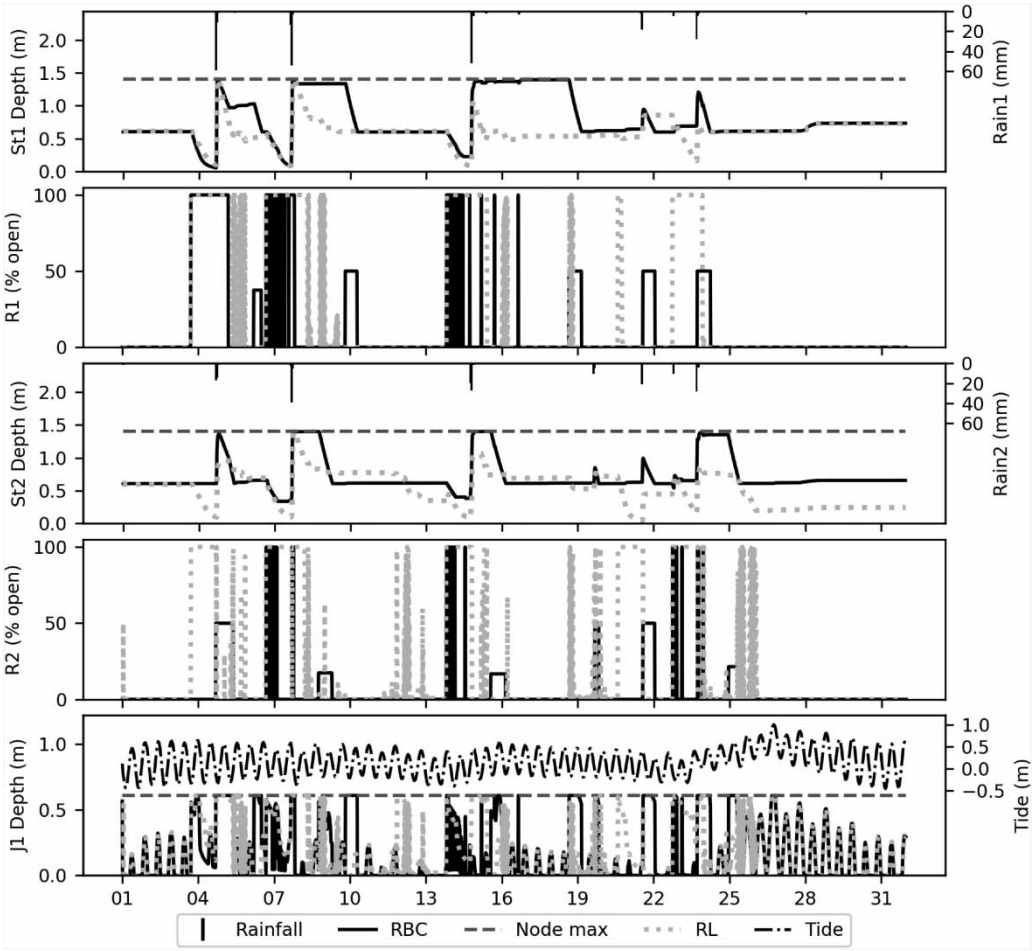

Figure 7 | Comparison of RL and RBC system performance on August 2019.

example, in March 2014, RBC increased total flooding of the system by nearly 5.5 times over the passive system (2,396 vs. 439 m^3MG) (see Appendix A, Figure 4).

Flood event classification results

Based on the flood event analysis at the two storage ponds, RL had the fewest flood events (St1: 22, St2: 26), followed by the passive system (St1: 41, St2: 42). RBC had the greatest number of pond flooding events (St1: 56, St2: 59). In terms of maximum flood volume for a single event, RL had the lowest, followed by RBC, and the passive system at pond St1 (27,558, 28,883, 30,094 m^3, respectively) and pond St2 (26,119, 27,331, 27,596 m^3). The mean single event flood volume showed a similar pattern at pond St1 (719, 719, 1,060 m^3) and pond St2 (871, 871, 1,136 m^3) for the RL, RBC, and passive systems, respectively. Flood event duration at the two ponds was similar for the three scenarios, with RL having the lowest mean duration (St1: 0.45 h, St2: 0.47 h) followed by RBC (St1: 0.51 h, St2: 0.53 h) and the passive system (St1: 0.73 h, St2: 0.66 h).

Results of flood event classification for downstream node J1 (a hypothetical roadway storm drain inlet) are shown in Figure 9. RL had the lowest number of flood events classified at the 0.2 and 0.3 m thresholds but the highest for the 0.4 m threshold (Figure 9(a)). RBC had more flood events for the 0.2 and 0.3 m thresholds than RL or the passive system, but the lowest for the 0.4 m threshold. This was expected, as these control rules manage the two ponds individually and not as a unified system considering downstream conditions. Flood volume for the 0.2 and 0.3 m thresholds was similar across the three systems

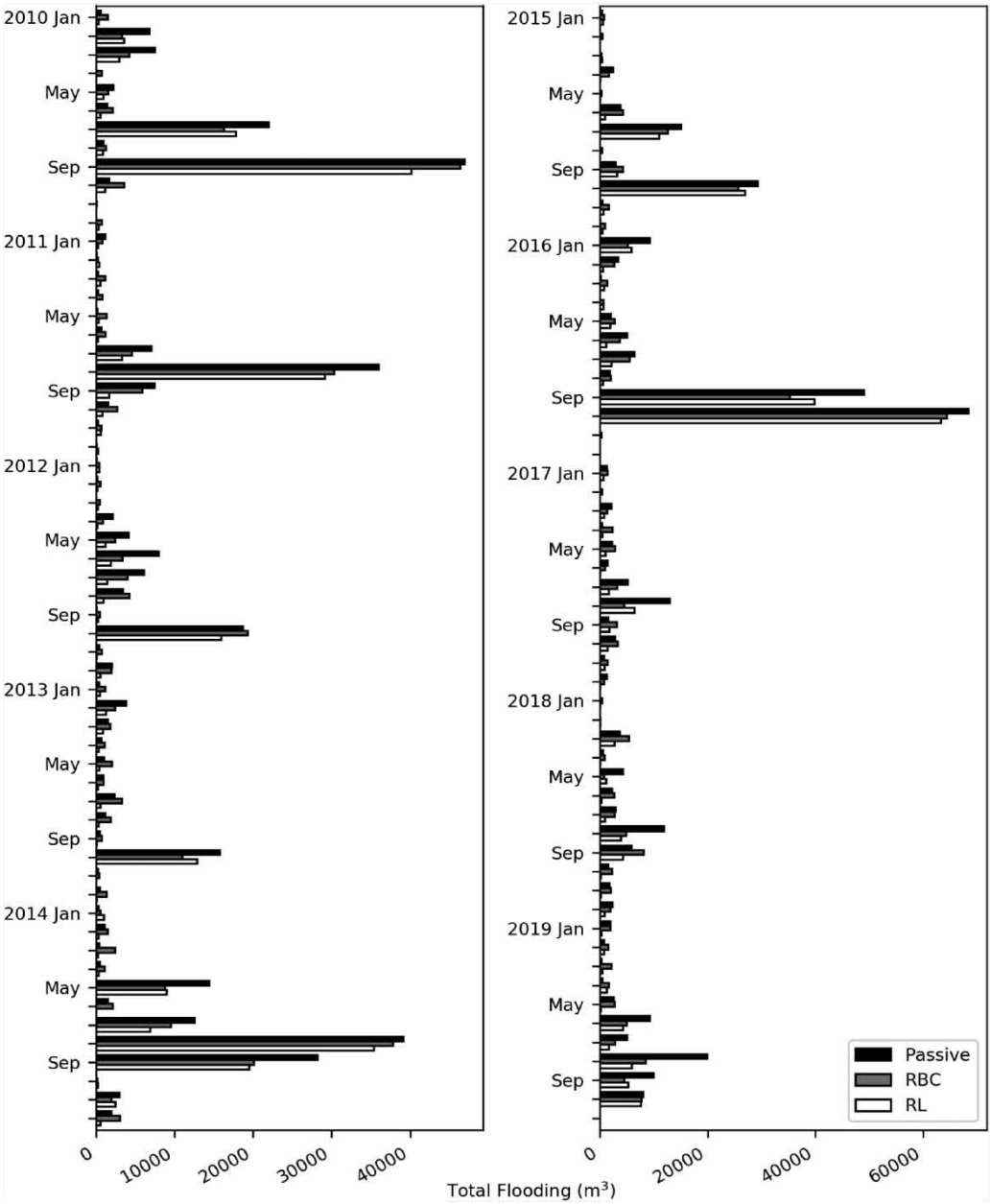

Figure 8 | Comparison of RL, RBC, and passive system performance on all months of data.

(Figure 9(c)). At the 0.4 m threshold, however, the two RTC methods (RL and RBC) had lower minimum, maximum, and mean flood volumes than the passive system.

Flood event duration for node J1 was lowest for the passive system, followed by RBC and RL (Figure 9(b)). This result makes sense when viewed in context with the flood volumes coming from the upstream ponds. RL prevented flooding at the ponds by routing more water downstream to node J1. RL had a similar mean flood duration to the passive systems, but a 60% higher maximum, indicating that RL allowed more low volume, but long duration flood events at J1 in order to reduce flooding at the two storage ponds. This behavior is influenced by the reward function; if the reward function was based on whether or not a node was flooding instead of the rate of flooding, the agent may have learned a different trade-off for managing flooding between the

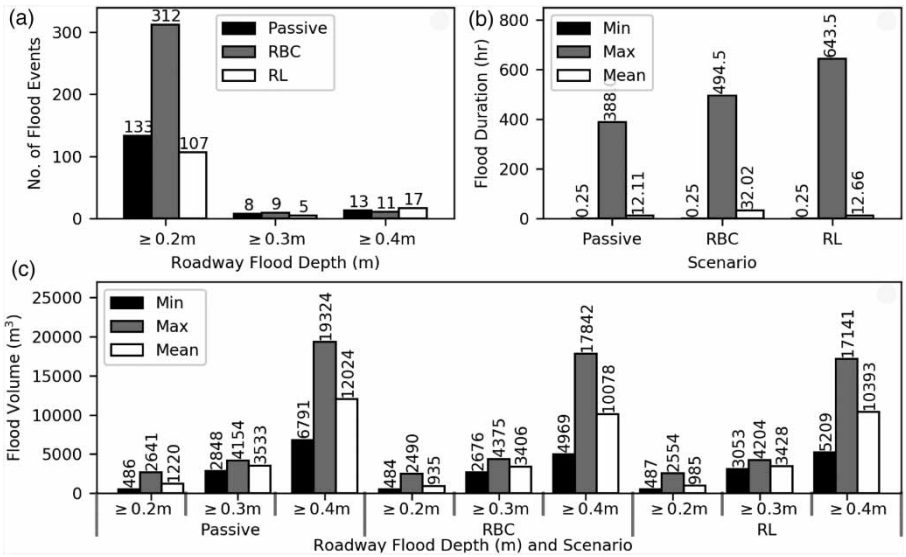

Figure 9 | Number (a), duration (b), and volume (c) of flood events at downstream node J1. Flood volumes at node J1 were categorized as causing ≥0.2, 0.3, or 0.4 m of water depth on the roadway.

three nodes. Due to the lack of system coordination, RBC had longer duration flood events at J1 than the passive system, but at lower volumes. In the passive system, the short duration, but a high volume of flood events at node J1 shows that without control, this system is flashy, which is a challenge in many urban systems.

DISCUSSION

One important aspect of any RTC application is the computational cost, both in terms of when the computation needs to happen and the time needed to compute a policy. With the RL algorithm used here, the agent learns offline on a training dataset. Once learned, the RL policy can be quickly applied. This formulation of MPC, in contrast, uses a genetic algorithm to perform on-line optimization (i.e., the best control actions are not known until the time that they need to be implemented). In this research, RL was able to learn a policy in approximately 34 min using 1 month of data on a standard desktop computer. Once developed, testing the RL policy simply requires running an SWMM simulation, passing the system state at each control time step through the agent, and implementing the resulting control actions (this takes 9 s for the August 2019 training data). Running MPC with the genetic algorithm and physics-based model as implemented in Sadler *et al.* (2019) required access to a HPC and took almost 50 h for 1 week of simulation. Because optimization is on-line, additional testing of MPC on other datasets would take a similar amount of time and would increase as the complexity of the system increases (Sadler *et al.* 2019). In practice, MPC may only need to run using the available forecast data (e.g., 18, 24, 36 h), not an entire week, reducing the computational burden (Sadler *et al.* 2020). Additionally, other formulations of MPC could use a different process model than SWMM (for instance using a state-space model learned from observed data) which could dramatically reduce MPC's computational cost (Balchen *et al.* 1992; Cigler *et al.* 2013; Corbin *et al.* 2013; Li *et al.* 2013; Behl *et al.* 2014). The training time for RL is also dependent on system complexity but needs further research to determine the feasibility and limitations for larger, more complex stormwater systems.

The RBC logic used in this research, like RL, can be considered an offline policy. Instead of an RL agent learning the control policy by interacting with the system, a human operator must understand the system well enough to formulate the rules. The growing adoption of Internet of Things (IoT) sensors for monitoring water levels provides the data needed to create control rules. In practice, the amount of time required to create these rules, and their quality, is

dependent on factors like the availability of data on the physical watershed characteristics and the complexity of the system to be controlled. For the relatively simple system used in this research, and real single ponds (OptiRTC and Geosyntec Consultants Inc. 2017; Marchese *et al.* 2018), developing control rules is feasible. Ensuring coordinated and effective system-level control, however, will become increasingly difficult as complexity increases. As an example, the rule for valve position when trying to maintain the target depth was originally to completely open the valve. Through simulation, it was found that this often caused increased flooding at the downstream node. Adjusting that rule to only open the valve 50% when maintaining the target depth helped eliminate downstream flooding but is system specific and most likely not optimal. Adding a depth sensor in the downstream pipe as an additional factor in the control rules would be possible in this case; with enough time and IoT sensors, it may be possible to create control rules considering system-wide performance. However, there could be many such factors in a real urban stormwater system and accounting for each one and their interactions under different flow conditions will quickly become unmanageable. RL has an advantage here because the relationships between components of the system do not have to be known or stated explicitly, but can be learned. The disadvantage of RL, however, is that it is much less transparent than RBC in terms of how and why certain control decisions are made.

While this paper has explored RTC with RL, MPC, and RBC, there are other methods from the field of control theory that could be applied to stormwater systems. Wong and Kerkez (2018) provide an elegant example by using a linear quadratic regulator to manage storage pond depths in urban headwater catchments. This uses a state-space model as a linear representation of a watershed and performs control with a feedback controller. Another key contribution of this work is the ability to optimize the location of control structures and show that the entire system does not have to be controlled to achieve system-wide benefits. The state-space representation used by Wong and Kerkez (2018) or the discrete time dynamic system shown in Schwanenberg *et al.* (2015) could be used in RL or MPC as a replacement to the more computationally expensive SWMM model to speed up control of larger systems, but the full dynamics of the system represented in physics-based models may be lost.

Groundwater could contribute a significant amount to retention ponds that are being actively controlled, especially in coastal cities with high groundwater tables like Norfolk, Virginia, that respond quickly to storm events (Bowes *et al.* 2019). For a retention pond in Norfolk, we have estimated that groundwater would contribute approximately 0.16 m or 11% of the pond's volume per hour if the pond is completely emptied (see Appendix C for details on these approximations). Considering the storm event of 4–5 August 2019, the RL agent lowers the depth of water in the simulated ponds by almost 0.61 m over a 24-h period (Figure 1). Over that time, groundwater would have contributed an additional 0.71 m or 50% of the pond's total volume. This is not currently reflected in the SWMM simulations but has important implications in practice. While the additional inflow would most likely not change the general policy learned by the RL agent (i.e., lowering depths before a storm and maintaining depths otherwise), a larger valve may be needed to drain the ponds more quickly or the agent may need a longer forecast in order to drain the ponds prior to a storm event. Additionally, evaporation should be included in these simulations before being applied to real-world systems.

When implementing any of the RTC methods presented in this research, the method's interpretability will influence its adoption and use by decision makers. While RBC is easy to understand and highly transparent, MPC is less so, and RL is the least transparent. The control policies created by RL, while effective, can cause the system to make decisions that are non-intuitive to a human operator. Therefore, fully automating smart stormwater systems with RL may not be advisable at this time until more testing and safety controls can be put in place. However, RL could assist human operators in determining control policies and support decision making, for example, as part of a recommendation system (Solomatine and Ostfeld 2008). RL-based policies should continue to be trained with new data as it becomes available to increase confidence that the RL policies will produce desirable outcomes. This study shows, however, that even with a single month's worth of training data, RL shows great potential for determining effective control policies.

CONCLUSIONS AND FUTURE WORK

This research has explored the application of an RL agent for real-time stormwater system control where both rainfall and tidal level can impact flooding and retention pond depths in the system. In contrast to previous work, this paper used a continuous action space to create more refined control policies, by implementing the DDPG RL algorithm. A conditional reward structure based on the rainfall forecast and inclusion of forecasts in the system state allowed the RL agent to learn proactive control strategies. The performance of RL was compared to a passive system as well as two other RTC methods: MPC and RBC.

Results of this research show that both RL and RBC can improve stormwater system performance compared to the passive system. Using a control policy developed from a single month of rainfall and tide data, RL reduced total flood volume by 32% over the passive system for the 2010–2019 data. RBC, while only controlling ponds individually, still reduced total flood volume by 13% compared to the passive system. Additionally, this research showed that RL was able to learn to balance flooding throughout the system to maximize the conditional reward and meet the control objectives of mitigating flooding and maintaining target pond water levels. When implemented using the SWMM physics-based model, as described in Sadler *et al.* (2019, 2020), MPC was too computationally expensive to run for more than a small portion of the datasets. In this research, RL provided an 88× speedup in the creation of control policies compared to MPC.

Although the simple stormwater system, which is inspired by conditions in the coastal city of Norfolk, Virginia, demonstrates that RL can outperform other methods, more complex systems will face different computational burdens that could be a barrier to using such methods in real-time. This needs to be explored through future research testing RL on real-world systems. In addition, an alternative implementation of MPC using a state-space model, instead of the SWMM model used here, could dramatically reduce the computational cost for this control method. Lastly, the feasibility of using, and potentially combining, any of the RTC methods for decision support to enhance stormwater system performance should be investigated.

In order to move this work toward implementation within real-world systems, it may be valuable to explore more complex reward functions than the one used in this study. For example, it may be better to base the reward on different variables beyond flood volume and pond water depth. It may be the case that costs due to valve operation and drainage of ponds in specific cases are higher than a small amount of flooding that does not have a societal impact. Additionally, the flow rates and velocities in the system may have additional restrictions to consider in the reward function (e.g., maintaining certain flow conditions for water quality or stream biota health or preventing flow velocities that can cause soil erosion). More complex reward functions can be explored in future work to account for more complex situations, move towards the control of real systems, and integrate specific characteristics of valves and ponds in real-world systems. Finally, this paper trained an RL agent on a single month of data. In future research, the sensitivity of the algorithm to the amount and diversity of the data during training should be investigated. This will help construct a trade-off analysis between the amount of data needed, the training time required, and the accuracy of the predictive models needed for the training procedure.

ACKNOWLEDGMENTS

This work was funded as part of two National Science Foundation grants: Award #1735587 (CRISP-Critical, Resilient Interdependent Infrastructure Systems and Processes) and Award #1737432 (SCC-IRG Track 1: Overcoming Social and Technical Barriers for the Broad Adoption of Smart Stormwater Systems). We acknowledge HRSD for continued access to their high-quality data and the City of Norfolk for information regarding their stormwater retention ponds.

DATA AVAILABILITY STATEMENT

All relevant data are available from an online repository or repositories. The code repository link is https://github.com/UVAdMIST/swmm_rl. The data repository link is http://www.hydroshare.org/resource/e2d21c9224ab4aefaf1a5b6394b270b1.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X. & Research, G. 2016 *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. arXiv preprint arXiv:160304467.

Abou Rjeily, Y., Abbas, O., Sadek, M., Shahrour, I. & Hage Chehade, F. 2018 Model predictive control for optimising the operation of urban drainage systems. *Journal of Hydrology* **566**, 558–565.

Balchen, J. G., Ljungquist, D. & Strand, S. 1992 State space predictive control. *Chemical Engineering Science* **47** (4), 787–807.

Behl, M., Nghiem, T. X. & Mangharam, R. 2014 Model-iq: uncertainty propagation from sensing to modeling and control in buildings. In: *2014 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*. pp. 13–24. IEEE.

Berggren, K., Olofsson, M., Viklander, M., Svensson, G. & Gustafsson, A.-M. 2012 Hydraulic impacts on urban drainage systems due to changes in rainfall caused by climatic change. *Journal of Hydrologic Engineering* **17** (1), 92–98.

Bowes, B. D. 2020a *Monthly SWMM Input Files for Real-time Control Simulation*. http://www.hydroshare.org/resource/e2d21c9224ab4aefaf1a5b6394b270b1

Bowes, B. D. 2020b *UVAdMIST/swmm_rl: Repo for SWMM RL code*. https://github.com/UVAdMIST/swmm_rl

Bowes, B. D., Sadler, J. M., Morsy, M. M., Behl, M. & Goodall, J. L. 2019 Forecasting groundwater table in a flood prone coastal city with long short-term memory and recurrent neural networks. *Water* **11** (5), 1098.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. & Zaremba, W. 2016 *OpenAI Gym*.

Camacho, E. F. & Bordons, C. C. 2007 *Model Predictive Control*. Springer, London, New York.

Castelletti, A., Pianosi, F. & Restelli, M. 2013 A multiobjective reinforcement learning approach to water resources systems operation: Pareto frontier approximation in a single run. *Water Resources Research* **49**, 3476–3486.

Castelletti, A., Yajima, H., Giuliani, M., Soncini-Sessa, R. & Weber, E. 2014 Planning the optimal operation of a multioutlet water reservoir with water quality and quantity targets. *Journal of Water Resources Planning and Management* **140** (4), 496–510.

Cigler, J., Gyalistras, D., Široky, J., Tiet, V. & Ferkl, L. 2013 Beyond theory: the challenge of implementing model predictive control in buildings. In *Proceedings of 11th Rehva world congress*, Clima, Vol. 250.

Corbin, C. D., Henze, G. P. & May-Ostendorp, P. 2013 A model predictive control optimization environment for real-time commercial building application. *Journal of Building Performance Simulation* **6** (3), 159–174.

Delipetrev, B., Jonoski, A. & Solomatine, D. P. 2017 A novel nested stochastic dynamic programming (nSDP) and nested reinforcement learning (nRL) algorithm for multipurpose reservoir optimization. *Journal of Hydroinformatics* **19** (1), 47–61.

García, L., Barreiro-Gomez, J., Escobar, E., Téllez, D., Quijano, N. & Ocampo-Martinez, C. 2015 Modeling and real-time control of urban drainage systems: a review. *Advances in Water Resources* **85**, 120–132.

Hartono, P. & Hashimoto, S. 2007 Learning from imperfect data. *Applied Soft Computing Journal* **7** (1), 353–363.

Jose Meneses, E., Gaussens, M., Jakobsen, C., Steen Mikkelsen, P., Grum, M. & Vezzaro, L. 2018 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. *Water* **10** (76), 1–15.

Kerkez, B., Gruden, C., Lewis, M., Montestruque, L., Quigley, M., Wong, B., Bedig, A., Kertesz, R., Braun, T., Cadwalader, O., Poresky, A. & Pak, C. 2016 Smarter stormwater systems. *Environmental Science and Technology* **50**, 72677273.

Lee, J.-H. & Labadie, J. W. 2007 Stochastic optimization of multireservoir systems via reinforcement learning. *Water Resources Research* **43** (11), 1–16.

Li, P., O'Neill, Z. & Braun, J. 2013 Development of control-oriented models for model predictive control in buildings. In *ASHRAE Trans*, Intelligent Buildings Operations Workshop, Boulder, CO.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. & Wierstra, D. 2015 Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*. p. 14.

Maier, H. R., Jain, A., Dandy, G. C. & Sudheer, K. P. 2010 Methods used for the development of neural networks for the prediction of water resource variables in river systems: current status and future directions. *Environmental Modelling and Software* **25**, 891–909.

Marchese, D., Johnson, J., Akers, N., Huffman, M. & Hlas, V. 2018 Quantitative comparison of active and passive stormwater infrastructure: case study in Beckley, West Virginia. *Proceedings of the Water Environment Federation* **2018** (9), 4298–4311.

McDonnell, B. E., Ratliff, K. M., Tryby, M. E., Wu, J. J. X. & Mullapudi, A. 2020 *PySWMM: The Python Interface to Stormwater Management Model (SWMM)*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. & Hassabis, D. 2015 Human-level control through deep reinforcement learning. *Nature* **518**, 529–543.

Moftakhari, H. R., AghaKouchak, A., Sanders, B. F., Feldman, D. L., Sweet, W., Matthew, R. A. & Luke, A. 2015 Increased nuisance flooding along the coasts of the United States due to

sea level rise: past and future. *Geophysical Research Letters* **42** (22), 9846–9852.

Moftakhari, H. R., AghaKouchak, A., Sanders, B. F. & Matthew, R. A. 2017 Cumulative hazard: the case of nuisance flooding. *Earth's Future* **5** (2), 214–223.

Mounce, S. R., Shepherd, W., Ostojin, S., Abdel-Aal, M., Schellart, A. N. A., Shucksmith, J. D. & Tait, S. J. 2020 Optimisation of a fuzzy logic-based local real-time control system for mitigation of sewer flooding using genetic algorithms. *Journal of Hydroinformatics* **22** (2), 281–295.

Mullapudi, A. & Kerkez, B. 2018 Autonomous control of urban storm water networks using reinforcement learning. *EPiC Series in Engineering* **3**, 1465–1469.

Mynett, A. E. & Vojinovic, Z. 2009 Hydroinformatics in multi-colours-part red: urban flood and disaster management. *Journal of Hydroinformatics* **11.3** (4), 166–180.

Neumann, J. E., Price, J., Chinowsky, P., Wright, L., Ludwig, L., Streeter, R., Jones, R., Smith, J. B., Perkins, W., Jantarasami, L. & Martinich, J. 2015 Climate change risks to US infrastructure: impacts on roads, bridges, coastal development, and urban drainage. *Climatic Change* **131** (1), 97–109.

OptiRTC and Geosyntec Consultants Inc 2017 *Water Quality Summary Report National Fish and Wildlife Foundation Smart, Integrated Stormwater Management Systems Anacostia River Watershed Water Quality Study*. Technical report.

Pianosi, F., Castelletti, A. & Restelli, M. 2013 Tree-based fitted Q-iteration for multi-objective Markov decision processes in water resource management. *Journal of Hydroinformatics* **15** (2), 258–270.

Plappert, M. 2016 *keras-rl*.

Pregnolato, M., Ford, A., Wilkinson, S. M. & Dawson, R. J. 2017 The impact of flooding on road transport: a depth-disruption function. *Transportation Research Part D: Transport and Environment* **55**, 67–81.

Sadler, J. M., Goodall, J. L., Behl, M., Morsy, M. M., Culver, T. B. & Bowes, B. D. 2019 Leveraging open source software and parallel computing for model predictive control of urban drainage systems using EPA-SWMM5. *Environmental Modelling & Software* **120**, 104484.

Sadler, J. M., Goodall, J. L., Behl, M., Bowes, B. D. & Morsy, M. M. 2020 Exploring real-time control of stormwater systems for mitigating flood risk due to sea level rise. *Journal of Hydrology* **583**, 1–10.

Schwanenberg, D., Becker, B. P. J. & Xu, M. 2015 The open real-time control (RTC)-tools software framework for modeling RTC in water resources systems. *Journal of Hydroinformatics* **17** (1), 130–148.

Solomatine, D. P. & Ostfeld, A. 2008 Data-driven modelling: some past experiences and new approaches. *Journal of Hydroinformatics* **10** (1), 3–22.

Sutton, R. S. & Barto, A. G. 2018 *Reinforcement Learning: An Introduction*, 2nd edn. The MIT Press, Cambridge, Massachusetts.

Sweet, W. V. & Park, J. 2014 From the extreme to the mean: Acceleration and tipping points of coastal inundation from sea level rise. *Earth's Future* **2** (12), 579–600.

van Andel, S. J., Price, R. K., Lobbrecht, A. H., van Kruiningen, F. & Mureau, R. 2008 Ensemble precipitation and water-level forecasts for anticipatory water-system control. *Journal of Hydrometeorology* **9** (4), 776–788.

van Andel, S. J., Price, R., Lobbrecht, A., van Kruiningen, F., Mureau, R. & Cordero, W. B. 2014 Framework for anticipatory water management: testing for flood control in the Rijnland storage basin. *Journal of Water Resources Planning and Management* **140** (4), 533–542.

Wong, B. P. & Kerkez, B. 2018 Real-time control of urban headwater catchments through linear feedback: performance, analysis, and site selection. *Water Resources Research* **54** (10), 7309–7330.

Wright, J. & Marchese, D. 2017 Briefing: continuous monitoring and adaptive control: the 'smart' storm water management solution. *Proceedings of the Institution of Civil Engineers – Smart Infrastructure and Construction* **170** (4), 86–89.

Wuebbles, D., Fahey, D., Hibbard, K., Dokken, D., Stewart, B. & Maycock, T. 2017 *Climate Science Special Report: Fourth National Climate Assessment, Volume I. Technical Report, U.S.* Global Change Research Program (USGCRP), Washington, DC.

First received 4 May 2020; accepted in revised form 16 September 2020. Available online 20 October 2020