Flooding and Overflow Mitigation through a Model-free Deep Reinforcement Learning based on Koopman Emulators of Urban Drainage System

Wenchong Tian[1], Zhenliang Liao[1†], Zhiyu Zhang[1], Hao Wu[2], Kunlun Xin[1].

[1] College of Environmental Science and Engineering, Tongji University, 200092 Shanghai, China.

[2] School of Mathematical Sciences, Tongji University, 200092 Shanghai, China.

†Corresponding author: Zhenliang Liao

Email address: 04150@tongji.edu.cn

Tel: +86 18701974804

Abstract

Deep reinforcement learning has been used to establish real-time control of urban drainage system (UDS) for flooding mitigation in recent studies. However, only model-based reinforcement learning was under consideration, which means that a mathematical model of UDS is necessarily needed during RL's training process. Although this is a natural way to establish RL system, it causes several problems, including (i) too much training time, (ii) too "rich" cache data, and (iii) too "perfect" training environment. To address these problems, a model-free RL training framework based on two Koopman emulators is provided and validated through simulation with respect to an UDS in a city located in eastern China. This framework achieves shorter training time and higher efficiency of data usage through the fast nonlinear emulation capability of Koopman emulators and the equalization between the dimension of emulator's observable and RL's state. Also, certain randomness is provided in RL training process through emulation. According to the results, compared with model-based RLs, this framework achieves a similar control effect with a 20 to 23 times faster training process and 79.67 times higher efficiency of data usage. The uncertainty analysis shows that slight perturbation which does not statistically change the control system in the training and testing process will not leverage the control effect of both model-based and model-free RLs. Meanwhile, the performances of the Koopman emulators of UDS are strongly related to their hyperparameters and the similarity between training data and test data.

Key words: Overflow and Flooding, Deep Reinforcement learning, Model-free, Koopman emulator, Fast model training.

1. Introduction

Flooding and overflow are inevitable problems of urban drainage system (UDS) (Xie et al., 2017; Liao et al., 2019; Ochoa et al., 2019; Zhi et al., 2020; Qi et al., 2021). A natural solution to reduce overflow and flooding is to enhance drainage infrastructure through pipeline expanding (Yazdi, 2018) and low-impact development (Batalini et al., 2021; Chan et al., 2018). But these methods might be financially and practically unfeasible in many places (Lund et al., 2020).

Recently, solutions in the context of smart city and urban water management received more attention. One of them is real time control, which uses system observations, numerical modelling, and control strategies in coordination to enhance the use of the existing systems for a given objective (Schütze et al., 2002; (García et al., 2015; Kerkez et al., 2016), and can be classified as heuristic control (Schütze et al., 2002; Lund et al, 2018) and optimization-based control (Joseph-Duran et al., 2015; Sun et al., 2020).

Meanwhile, deep reinforcement learning (RL) was also used to establish RTC systems for flooding mitigation (Mullapudi et al., 2020; Saliba et al., 2020). It trains an AI agent to control an UDS, called environment in RL, in real time. Currently, they are all model-based RL, which means that a model, such as SWMM, is employed as the environment during training process. This is a natural way to establish an agent-environment framework for the RLs of UDS, but along with some problems. The very first one is training time. Model-based RLs use a model as environment to simulate many times during training process, causing a huge demand of computing resource (Mullapudi et al., 2020). The second problem is the "richness" of cache data. Current RLs for UDS only need some selected features for controlling (Mullapudi et al., 2020; Saliba et al., 2020), while many other data provided by model are excluded, leading to low efficiency of data usage and wasting of computer's memory. The third problem is that its training environment has zero noise, causing differences compared with real-world conditions. If the RL is trained under ideal simulation conditions, the input, or the state, can be obtained directly without any noise. However, this doesn't hold in real-world conditions as inputs usually contain randomness (Saliba et al., 2020).

To address these problems, a basic solution in the field of RL is to use model-free method. Usually, model-free RLs have two types, using real-world system as environment (Mnih et al., 2015), or using a surrogate model as environment (Chua et al., 2018; Kalweit & Boedecker, 2017), which means using an untrained RL to control the real-world system or its surrogate model at the initial stage to learn by trial and error. For the first type, the risk of this process is acceptable for gaming (Shao et al., 2018) and robot (Song et al., 2012), as it only loses a game or breaks a robot at most, but is not for UDS control, as it might cause damage to property and public safety (Mullapudi et al., 2020). Thus, a safe option is to use an accuracy surrogate model, or an emulator, as the environment, rather than interact with real-world system directly.

Many promising emulators for UDS have been developed in the recent study. For instance, mechanistic emulator (MEM) (Machac et al., 2018), data-driven emulator (DDE) (Carbajal et al., 2017), and polynomial expansions emulator (PEE) (Nagel et al., 2020), were established for sensitivity analysis and calibration of UDS. Linear surrogate model (LSM) (Lund et al., 2020) is also used in model predictive control of stormwater system. However, these emulators still suffer from the nonlinearity and diversity of water environmental dynamics. MEM and PEE might fail to express some nonlinear dynamics, as they lack ob-

jectivity in model selection. DDE, LSM, and RSM are sensitive to the sparseness of training data, causing a poor generalization when data is unevenly sampled. Restricted by these, it is unknown how far the emulators can be helpful to other applications, including the model-free framework of UDS.

The Koopman emulator is originated from Koopman operator, which is a concept from statistical physics and dynamic system (Otto & Rowley 2019; Klus et al., 2016; Klus et al., 2018; Wu & Noé, 2019; Mardt et al., 2018). Every dynamical system, linear or nonlinear, has an associated Koopman operator, which encodes properties of the system and characterizes the temporal evolution of observables (Li et al., 2017). Its main idea is to approximate a non-linear dynamic system through a finite dimensional linear system that has an efficiency trade-off between error and dimension (Williams et al. 2015) and can be used as an emulator (Peitz & Klus, 2019). Importantly, this nearly optimal linear emulator of original model can be automatically found through data and approximation algorithms without using a user-defined model, thereby achieving better objectivity and adaptability in nonlinear emulation than others. Benefit from this, Koopman operator theory and related algorithms have been widely used for model simplify (Rowley et al., 2009), prediction (Budišić & Mezić, 2012), data fusion (Williams et al., 2015b), and system control (Brunton et al., 2016; Korda & Mezić, 2018).

In this study, Koopman emulator is employed to set up a model-free RL training framework for (i) reducing the training time of RLs, (ii) improving the efficiency of RLs' data usage, and (iii) providing a training environment with some randomness. Specifically, two Koopman emulators based on two approximation algorithms are provided and plugged into model-free framework to train different RLs for flooding and overflow mitigation, and then compared with model-based RLs. The remainder of this paper is organized as follows: In Section 2, we briefly introduce some related works, including RL and its application on UDS and Koopman operator theory. In Section 3, we describe the details of our method. The case study is introduced in Section 4. The results and corresponding discussions are given in Section 5 and 6. Conclusions are given in Section 7.

1. Preliminaries and related works

   (a) Brief review of RL and its application on UDS

RL is a kind of methods using experimental trials and relatively simple feedback to train an AI for controlling and planning under different situations and maximizing the expectation of the weighted sum of reward signal (Sutton & Barto, 2018). Usually, An RL model has an agent and an environment. The environment is the system controlled by agent. The control loop of RL can be described as Fig.1.

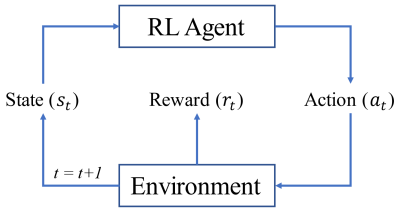

Fig.1 Loop of RL control

All the RLs learn from job, regarding different training algorithms, means that they collect information during controlling and use this to upgrade themselves. After several loop of control, the states, actions, rewards will be collected and used to train agent through a given RL algorithm to improve the its performance, which mathematically means that it maximizes expected total rewards (value function Eq.1 or q value Eq.2).

@ >p(- 4) * >p(- 4) * >p(- 4) * @

$$\& \; V(s_t) = \mathbb{E}_t \left[\sum_{k=0}^{\infty} \left[\gamma^k r_{t+k+1} | s_t \right] \right] \; \& \tag{1}$$

$$\& \; q(a_t, s_t, r_t) = \mathbb{E}_t \left[\sum_{k=0}^{\infty} \left[\gamma^k r_{t+k+1} | a_t, s_t, r_t \right] \right] \; \& \tag{2}$$

The a_t, s_t, r_t are action, state, reward, the discount factor γ is a hyperparameter between 0 and 1 to guarantee the convergence of the functions and govern the temporal context of the reward. The k is the forward step, which means how many steps should the agent consider (Sutton & Barto, 2018). Usually, the RLs at initial status will not be used for real-world controlling, thus some researches separate the training from testing (Chen et al., 2015).

RLs can be classified as value-based and policy-based by its training algorithm (Mnih et al., 2016). The former uses deep neural network to search a higher value functions, and uses this to guide the selection of action (Mnih et al., 2015; Hasselt, 2010). The latter uses deep neural network to find the high-value action directly (Schulman et al., 2015; Schulman et al., 2017). Usually, the policy-based methods have better performance in the continuous policy space, while

the value-based methods are better in terms of the optimality of the control process (Mnih et al., 2016; Sutton & Barto, 2018).

Model-based RLs use a specifical model, such as SWMM (Mullapudi et al., 2020; Saliba et al., 2020), as an environment in training process. While model-free RLs use a surrogate model (Chua et al., 2018; Kalweit & Boedecker, 2017) or interact with real-world environment directly (Mnih et al., 2015) in training process. They can be described as Fig.2.

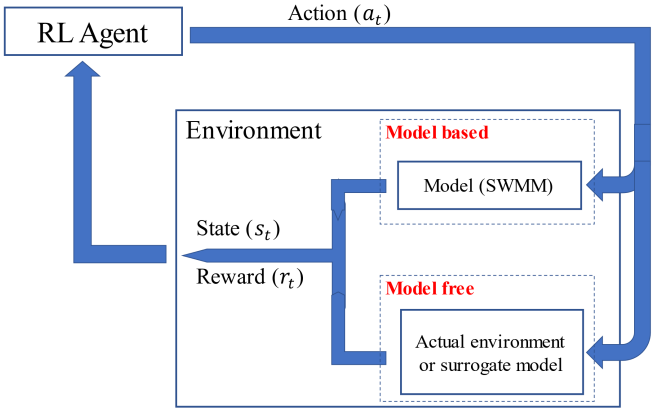

Fig.2 Model-based and model-free

Currently, RL becomes a state-of-art method in many different fields of study. Such as autonomous driving (Pan et al., 2017), robotic control (Kober and Peters, 2012), and game AI (Wu et al., 2018; Shao et al., 2018; Silver et al., 2017). In recent years, RL has also been used in flow controlling (Ochoa et al., 2019), multi-objective reservoir scheduling systems (Madani & Hooshyar, 2014; Castelletti et al., 2013), in-line storage control (Labadie, 2014), and flooding mitigation (Mullapudi et al., 2020; Saliba et al., 2020). However, RLs for UDS are model-based RL (Mullapudi et al., 2020; Saliba et al., 2020), means that a model of UDS, SWMM for example, is needed during RL's training process. This may lead to longer training time, lower efficiency of data usage, and the zero noisy training environment.

1. Koopman operator and its approximation algorithm

   (a) Dynamic system and Koopman operator

The definition of Koopman operator and its approximation theory can be found in previous researches (Williams et al., 2015a; Li et al., 2017). We only provide a step-by-step introduction of it. Given a nonlinear dynamic system on measurable space M

$$x(n+1) = \mathbf{f}(x(n)),\ x(n) \in M,\ n \geq 0 \quad (3)$$

The Koopman formalism focuses on the evolution of observables represented by functions on M in a suitable Hilbert function space with measure p

$$L^2(M,p) = \left\{\phi : M \to C : \|\phi\|_{L^2(M,p)} < \infty\right\} \quad (4)$$

$$\|\phi\|_{L^2(M,p)} = \int |\phi(x)|^2 \,\mathrm{pdx}$$

Then, for any given observable function $\phi \in L^2(M,p)$, its Koopman operator $\mathbf{K}$ can be given through Riesz representation theorem.

$$\mathbf{K}\phi = \phi \circ \mathbf{f} \quad (5)$$

Suppose the Koopman operator is a bounded linear operator, it is amenable to spectral analysis. The so called Koopman mode decomposition of Eq.3 can be described by Eq.6, where $\{\varphi_1, \varphi_2, \ldots, \varphi_k\}$ and $\{\mu_1, \mu_2, \ldots, \mu_k\}$ are the leading eigenfunctions and eigenvalues of Koopman operator, $\{\xi_1, \xi_2, \ldots, \xi_k\}$ are corresponded Koopman modes.

$$\mathbb{E}\left[g\left(x(n+1)\right)\right] = \mathbb{E}\left[g\left(\mathbf{f}\left(x(n)\right)\right)\right] = \mathbf{K}^T\mathbb{E}\left[g\left(x(n)\right)\right] = \sum_i \xi_i \mathbf{K}^T\varphi_i(x(n)) = \sum_i \xi_i\mu_i\varphi_i(x(n)) \approx \sum_{i=1}^{k} \xi_i\mu_i\varphi_i(x(n$$

Thus, a linear approximation can be obtained through Koopman operator and its leading eigenfunctions and eigenvalues (Li et al., 2017; Otto & Rowley, 2019).

1. Approximation algorithm

   The key idea of Koopman operator theory is to use a linear system with finite dimension to approximate a nonlinear dynamic. How to find a specific linear system is achieved by approximation algorithm. Currently, data driven methods are mainly studied in this field, and they can be classified as two types: Regression based and Rayleigh Variational Principle based. The former projects the original nonlinear system onto some Hilbert space, and use regression to find linear dynamic system. The latter uses the singular functions of Koopman operator as the basis functions of linear system and find the singular function directly through Rayleigh Variational Principle. We provide reference of these two types of approximation algorithms in Table1.

Table 1. Approximation algorithm

Name	Regression based or variational principle based	Reference
DMD	Regression based	Page & Kerswell, 2018
EDMD	Regression based	Williams et al., 2015a
KEDMD	Regression based	Klus et al., 2018
DLEDMD	Regression based	Li et al., 2017
VAC	Variational principle based	Noé & Nüske, 2012
VAMP	Variational principle based	Wu & Noé, 2019
KVAD	Variational principle based	Tian & Wu, 2021

Recent study shows that Rayleigh Variational Principle based methods suffer from two drawbacks (Tian & Wu, 2021): First, it is necessary to assume that the Koopman operator of given dynamic system is compact so that the maximum values of variational scores exist. But it can be proved that this assumption does not hold for most deterministic systems, including SWMM model. Second, the common variational scores are possibly sensitive to small modeling variations, which could affect the effectiveness.

1. Methodology

A model-free framework for RL training is provided based on Koopman emulator. Specifically, two approximation algorithms, Kernel Extended Dynamic Mode Decomposition (KEDMD) and Dictionary Learning Extended Dynamic Mode Decomposition (DLEDMD), are used to construct two emulators of UDS model (SWMM in this case) and plug them into a model-free RL system as training environment. We provided the details of the method in this section, including KEDMD emulator and DLEDMD emulator, model-free framework based on these two emulators, and uncertainty analysis of control system. The route map is given as Fig.3.

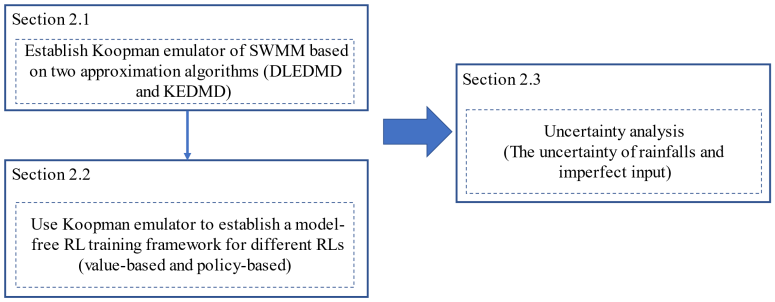

Fig.3 Route map of methodology

1. Koopman emulator for UDS

We use two Koopman emulators to approximate UDS model. We take SWMM (ver 5.1) as an example in this study. The basic mathematical formalism of Koopman emulators, the approximation methods, and the index of their performance are provided in this section.

1. Problem formulation

   We first provided the mathematical formulation of SWMM in the view of nonlinear dynamic system, which can be described as Eq.7. The observable, or dynamic variable, x_t is a vector variable representing information of SWMM at time point t, including current overflow and flooding volume, current water level of some selected junctions and subcatchments. The action a_t is the vector variable which represents the control signal of control assets, such as pumps, from time point t to $t+1$. The rain_t is the rainfall intensity from time point t to $t+1$.

$$x_{t+1} = SWMM\left(x_t, a_t, \text{rain}_t\right) \tag{7}$$

$$x_t = \left[\text{overflow}_t + \text{flooding}_t, \text{junctions}_t, \text{subcatchments}_t\right]$$
$$a_t = \left[\text{pumps}_t\right]$$

   According to Koopman operator theory, the dynamic of SWMM can be considered as a nonlinear dynamic system and has an associated Koopman operator with respect to action and rainfall (Eq.8) encoding the properties of the system characteristic and evolution (Li et al., 2017).

$$\mathbf{f}(x_{t+1}) = \mathbf{K}_{a_t,\text{rain}_t}\mathbf{f}(x_t) \tag{8}$$

The $\mathbf{K}_{a_t,\text{rain}_t}$ and f are Koopman operator and dictionary functions. Because the Eq.8 is linear and only focuses on the dynamic of some observables, its computing time is faster than SWMM.

1. Approximation algorithms and training process

An explicit formulation of Eq.8 can be found through different approximation algorithms and used as emulator of SWMM. We use two data-driven approximation algorithms to achieve this target. They are KEDMD (Klus et al., 2019) and DLEDMD (Li et al., 2017), which are both regression-based methods. We first introduce these two algorithms then provide our main steps of establishing two emulators.

1. KEDMD

KEDMD is original from Extend Dynamic Mode Decomposition (EDMD) which uses user-defined basis functions to find linear approximation. A very obvious drawback of EDMD is the selection of basis functions. A user-defined one may lead more subjective and less optimization. To avoid this, KEDMD, a variant of EDMD, is developed. It uses kernel functions as the basis functions, means that appropriate basis functions are given by data rather than hand-made. More detail can be found in previous research (Junge & Koltai, 2009; Klus et al., 2018), we only provided basic steps of KEDMD:

1. Prepare M training data $\left(x_t, a_t, \mathrm{rain}_t, x_{t+1}\right)_i, \quad i = 1, \ldots, M$ through the simulation of SWMM with respect to different rainfalls and actions.

2. Define a kernel function k (Gaussian kernel is used in this study), and compute Gram matrix G_{YX} and G_{XX} by Eq.9. The σ is the kernel bandwidth and is chosen as 1 in this study.

$$k(x, y) = exp\left(-\frac{\|x-y\|_2{}^2}{\sigma}\right) \tag{9}$$

$$[G_{\mathrm{YX}}]_{i,j} = k\left(Y_j, X_i\right), \ [G_{\mathrm{XX}}]_{i,j} = k\left(X_i, X_j\right)$$
$$X_i = [x_t, a_t, \mathrm{rain}_t]_i, \ Y_i = [x_{t+1}]_i$$

1. Solve the eigenvalue problem given by Eq.10, and obtain eigenfunctions $\{\widetilde{\varphi}_1, \widetilde{\varphi}_2, ...\}$ and eigenvalues $\{\lambda_1, \lambda_2, ...\}$

$$G_{\mathrm{XX}}{}^{-1} G_{\mathrm{YX}} \widetilde{\varphi} = \lambda \widetilde{\varphi} \tag{10}$$

1. Use the leading eigenfunctions and eigenvalues to reconstruct a linear system through Eq.11, and use it as emulator. The number of leading eigenfunctions in this study is L.

$$\mathbb{E}[Y] \approx \sum_{j=1}^{L} \lambda_j \sum_{i=1}^{M} k\left(X, x_i\right) \widetilde{\varphi}_j(x_i) \tag{11}$$

$$X = [x_t, a_t, \mathrm{rain}_t], \ Y = [x_{t+1}]$$

1. DLEDMD

Similar to KEDMD, DLEDMD is also original from Extend Dynamic Mode Decomposition (EDMD) but takes the advantage of deep learning to find better basis functions through training and data. More detail can be found in previous research (Li et al., 2017), we only provided basic steps of DLEDMD here:

1. Prepare M training data $\left(x_t, a_t, \mathrm{rain}_t, x_{t+1}\right)_i, \quad i = 1, \ldots, M$ through the simulation of SWMM with respect to different rainfalls.

2. Define a neural network $\psi(x; w)$ with a finite dimensional output layer, and train it through the extended minimization problem given by Eq.12.

The $\lambda\left(\widetilde{\mathbf{K}},\widetilde{w}\right)$ is a suitable regularizer. The architecture of $\psi(x;w)$ is N-10-10-N in this study where N is the dimension of observable or dynamic variable.

$$(\mathbf{K},w)=\arg\min\sum_{i=1}^{M}\left\|\widetilde{\mathbf{K}}\psi\left(X_i;\widetilde{w}\right)-Y_i\right\|^2+\lambda\left(\widetilde{\mathbf{K}},\widetilde{w}\right) \quad (12)$$

$$X_i=[x_t,a_t,\mathrm{rain}_t]_i,\ Y_i=[x_{t+1}]_i$$

1. Use the $\mathbf{K}$ and $\psi(x;w)$ as emulator.

1. Training process

In this study, we use these two algorithms to obtain two emulators of SWMM. We bring the training steps based on the above algorithms. For easy understanding, we call these two emulators KEDMD and DLEDMD in this paper. First, different rainfall data and control actions are given and send into SWMM to simulate, and obtaining the simulation results. Then, these results are used as training data to train two emulators through KEDMD and DLEDMD algorithms separately, and we have two emulators. These steps are given in Fig.4.

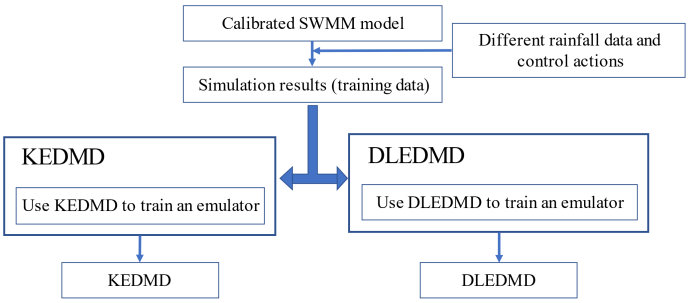

Fig.4 Training process of KEDMD and DLEDMD

1. The index of emulator performance

   The mean square error (MSE, Eq.13) and Nash-Sutcliffe efficiency coefficient (NSE, Eq.14) are used as the indices of emulators' performance in this study. The $S_{i,t}$ is the i variables given by SWMM model at time point t, the $E_{i,t}$ is the i variable given by emulators at time point t. A lower MSE and a higher NSE indicate better performance of emulators.

$$MSE = \frac{1}{\mathrm{TN}} \sum_{t=0}^{T} \sum_{i=0}^{N} \left(S_{i,t} - E_{i,t}\right)^2 \quad (13)$$

$$NSE = 1 - \frac{\sum_{t=0}^{T} \sum_{i=0}^{N} \left(S_{i,t} - E_{i,t}\right)^2}{\sum_{t=0}^{T} \sum_{i=0}^{N} \left(S_{i,t} - \mathbb{E}[E_{i,t}]\right)^2} \quad (14)$$

1. Model-free RL based on emulators

Two emulators, KEDMD and DLEDMD, are used to establish a model-free training framework for two RLs: deep q learning (DQN) and proximal policy optimization (PPO), which are value-based RL and policy-based RL. Then, the trained RLs are used to control the UDS in real-time for overflow and flooding mitigation. Considering the risk of real-world implement, a well-calibrated SWMM model (ver5.1) is used as the environment for the testing in this study. The system is established through python 3.7, pyswmm 0.6.0, and tensorflow 1.14.0.

1. Training process of RLs

Although the training algorithm of RLs are different, they all share a similar framework including sampling and upgrading. First, the untrained RL agents are used to control the environment in a given time length (called training sampling interval) with respect to some given rainfalls (the number of rainfall events is called batch size). The state, reward, action data of these controlling process are collected. Then, the collected data are used to upgrade RLs through different RL algorithms, such as DQN and PPO in this study. This sampling-upgrading process will be iterated for several times until the RL achieves a high enough q value or the training step is equal to a user-defined maximum step. This process can be described as Fig.5.

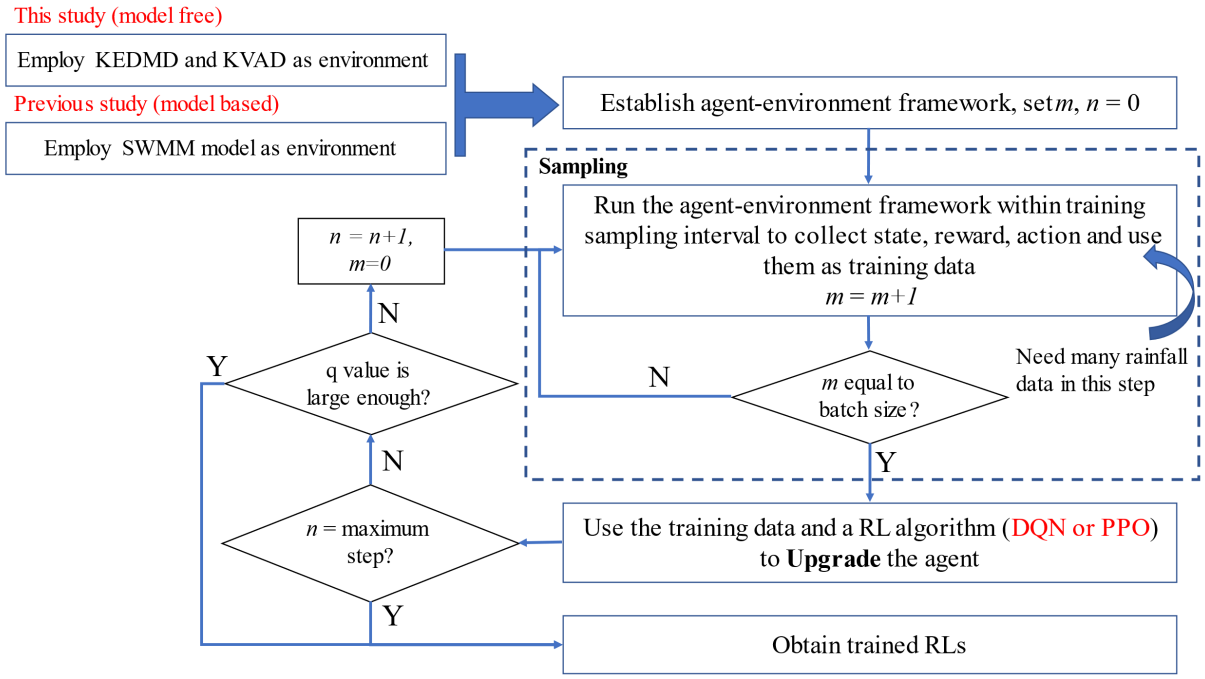

Fig.5 Training process of RLs (with SWMM or emulators). In this study, the training sampling interval is the time steps of rainfall, the batch size is 200, the maximum step is 100. The threshold of q value is 3 times of it in the initial step.

The main different between model-based and model-free is that a model-based RL use a mathematic model (such as SWMM) as environment during sampling, while a model-free RL use real-world system or emulator as environment during sampling. Because the sampling-upgrading process needs to run multi times, means that the environment will simulate many times, leading to the fact that model-based methods have more computing time and cache data than model-free methods.

Two RL algorithms, deep q learning (DQN) and proximal policy optimization (PPO), are used in this study. Their configurations are given as follow.

1. The selection of state, reward, and action

There are few studies focuses on how to select state, reward, action in the RTC of UDS. Therefore, the following only provides a benchmark selection. For easy comparison, all the RLs use the same state, reward, and action in this study. Their states s_t are current flooding and overflow volume, current inflow volume, current outflow volume, and current stored water volume in the pipeline. These can be provided by the simulation results of SWMM and two emulators. Their action are the variables representing the control signal of control assets, such

as pumps in this study, from time point t to time point $t+1$. Their reward r_t is given by Eq.15 to evaluate their performance of overflow and flooding mitigation.

@ >p(- 4) * >p(- 4) * >p(- 4) * @ & $r_t = \begin{cases} 1 & \\ & -1 \quad if \text{ overflow and } flooding \ \ in \ [t-1,t] \ is \ larger \ than \end{cases}$ &

(15)

1. DQN

The DQN is developed based on the theory of q learning. It uses deep neural network to approximate q value function and maximizes it through training (Mnih et al., 2015; Sutton & Barto, 2018). The deep neural network of DQN in this case has an architecture with N-20-20-1, N is the dimension of state vector, which is 4 in this study. The ReLU function (Eq.16) and the linear function (Eq.17, W^T and b are trainable parameters) are used as the activation function in the hidden layers and the output layer. The discount factor is 0.9. The training algorithm is Adam algorithm (LeCun et al. 2015). The iteration step is 200, learning rate is 0.001.

x x > 0 (16)

$$\text{linear}(x) = W^T x + b \quad (17)$$

1. PPO

Proximal policy optimization model uses a deep neural network to approximate the policy function (input state, and output action) by computing an estimator of the policy gradient (Eq.18) through sampled data (state, reward, action), and plugging it into any kind of gradient ascent algorithm (Schulman et al., 2015; Schulman et al., 2017).

@ >p(- 4) * >p(- 4) * >p(- 4) * @ & $g = \mathbb{E}_t \left[\nabla_\theta \log \pi_\theta \left(a_t | s_t \right) q \right]$ &

(18)

The $\pi_\theta \left(a_t | s_t \right)$ is policy function given by neural network with parameter θ, q is the q value. The deep neural network has an architecture with N-100-100-M, N and M are the dimension of state and action vector, which are 4 and 8 in this study. The ReLU (Eq.16) and the hard-sigmoid function (Eq.19) are used as the activation function in the hidden layers and the output layer. Their discount

factor is 0.9, training algorithm is Adam algorithm (LeCun et al. 2015). The iteration step is 200, learning rate is 0.001.

1 x_i > 0.5 (19)

1. Rainfall data used for training

The sampling process of RL is triggered by many rainfall data, therefore, the Chicago rainfall (Eq.20) and historical research of the rainstorm intensity formula parameters in the nearby area (Wang & Xu, 2016) are used to provide a method to sample enough rainfall data for RL training. This method can also be used in other area to provide random rainfall events. The range of rainfall intensities are given in Table 2.

$$i\left(t_b\right)=\frac{[A(1+Clog(P))]\left[\frac{(1-n)t_b}{K}+b\right]}{\left(\frac{t_b}{K}+b\right)^{1+n}} \quad (20)$$

$$i\left(t_a\right)=\frac{[A(1+Clog(P))]\left[\frac{(1-n)t_a}{1-K}+b\right]}{\left(\frac{t_a}{1-K}+b\right)^{1+n}}$$

Table 2. The range of rainfall parameters

A (mm)	C	P (year)	n	b	K
21~35	0.939~1.20	1~5	0.86~0.96	16~22	0.3~0.8

The t_a and t_b are the time, $i\left(t_b\right)$ and $i\left(t_a\right)$ are the rainfall intensity, A is the rainfall intensity with the recurrence period of one year, C is an experience parameter, P is the rainstorm return period, K is the peak intensity position coefficient, n and b are parameters related to region. A total of 20 rainfall events were generated and used for RLs' training. Each rainfall event has a two-hour duration.

1. Index of data usage efficiency

   We provide an index called data usage rate (DUR, Eq.21) to measure the efficiency of data usage.

$$DUR=\frac{\mathrm{NSD}}{\mathrm{NRD}} \quad (21)$$

$NSD = The\ number\ of\ state\ data\ used\ by\ RL$
$NRD = The\ number\ of\ results\ data\ provided\ by\ environment$

It is the ratio of the number of state data to the number of result data given by environment. For instance, if an environment, such as SWMM model, provides total of 100 results in one loop of sampling, but only 4 of them are selected as the state of RL, then its DUR is 4%. A higher DUR means better efficiency of data usage.

1. Uncertainty analysis

According to the modelling process given in Section 3.1 and 3.2, the uncertainty of this system is coming from two aspects: i) the diversity of rainfall events, and ii) the imperfect input of RLs. We analyze both of them in this study.

1. The index for uncertainty analysis

Usually, the flooding and overflow volume of a given UDS is strongly influenced by rainfall conditions (van Daal et al., 2017; Lund et al., 2018; Lund et al., 2020). Therefore, flooding and overflow volume may be unobjective when used as the measure. Considering this, we use the ratio of the total flooding and overflow to the total inflow (RCI) as an index to measure the performance of the RLs in the uncertainty analysis.

@ >p(- 4) * >p(- 4) * >p(- 4) * @ & RCI $= \frac{\text{total overflow and flooding}}{\text{total inflow}}$ &

(22)

For instance, a system with good control effect should have an RCI close to zero, no matter what kind of rainfall. Therefore, it can reduce the fluctuations caused by rainfall data to a certain degree. Compared with overflow and flooding volume, it can reflect the drainage capacity of RLs more objectively.

1. The uncertainty of rainfalls

The trained RLs will face various rainfall events in real-world conditions. Therefore, the randomness of rainfall events is one of the uncertainty resources. We use the forward uncertainty analysis with Monte Carlo simulations to test this uncertainty (van Daal et al., 2017). Because the rainfall pattern formula (Eq.20) is adopted, the affected parameters are the 4 parameters including A, C, P, b. For each RL, we randomly select 50 sets of 4 parameters within the range given in Table 2 to generate 50 designed rainfalls (parameter K is 0.5), and use them to simulate. Then the median value (50%) and upper (95%) and lower (5%) boundaries of the RCI are extracted.

1. The uncertainty of imperfect input

The state of RLs is subjected to noise in real-world implementation (Saliba et al., 2020), causing imperfect input and uncertainty to RLs. We us the method of previous research (Saliba et al., 2020) to analyze this uncertainty. Given a test rainfall and RLs, a uniform distribution $U(0.95, 1.05)$ is sampled to generate a value X and added to the state through Eq.23.

@ >p(- 4) * >p(- 4) * >p(- 4) * @ & $\overline{s_t} = s_t \times X,\ X \sim U(0.95, 1.05)$ &

(23)

Then, the new state $\overline{s_t}$ is used as the input of all the RLs at time point t. We also use this method in this study.

1. Case study

   (a) Urban drainage system

The UDS in a city located in eastern China is used as the case study. It is a combined sewer system with 139 nodes, 140 pipelines, and three pump stations. The maximum capacity of the pumping stations may exceed the max capacity of the pipe during storm weather. Therefore, the control system needs to smartly balance the water volume in both upstream and downstream. Considering the risk to property and public safety, a SWMM model of this combined sewer system is used as the testing environment. The schematic diagram of the model is shown in Fig.6. More details of this model can be found in our previous researches (Liao et al., 2019; Zhi et al., 2019).

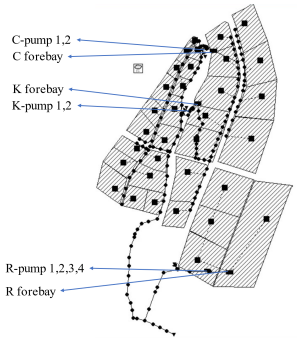

Fig.6 The SWMM model of case study

A rule-based RTC system called water level system is currently used in this place. It sets a sequence of water-level threshold values, or set-points, to operate the pumps. The pump starts working if the water level of the forebay reaches its onset threshold and shuts down when the water level falls down to the shutoff threshold. The detailed onset/shutoff threshold values are given in Table 3. As the pumps drain water when the water level is high, this RTC system has a certain level of capability of reducing combined sewer overflow (CSO) and flooding.

Table 3. The onset/shutoff threshold values of the water-level system.

	Onset threshold (m)	Shutoff threshold (m)

	Onset threshold (m)	Shutoff threshold (m)
C-pump-1	4.56	3.26
C-pump-2	4.87	4.56
K-pump-1	4.56	3.26
K-pump-2	4.87	4.56
R-pump-1	5.00	4.71
R-pump-2	6.31	5.00
R-pump-3	7.00	6.31
R-pump-4	7.78	7.00

1. Test rainfall data

We use both designed rainfall and real rainfall (given in Fig.7) to test the performance of the RLs in CSO and flooding reduction. The four designed rainfalls have 10 minutes interval and 2 hours duration. Since there is no rainfall data with 10 minutes interval in the study area, these 4 real rainfall data come from online monitoring of surrounding city (a data API provide by http://caiyunapp.com/index.html#api). To further testing the system's behavior under extreme rainfall, we enlarged the intensity of 4 real rainfalls by 10 times.

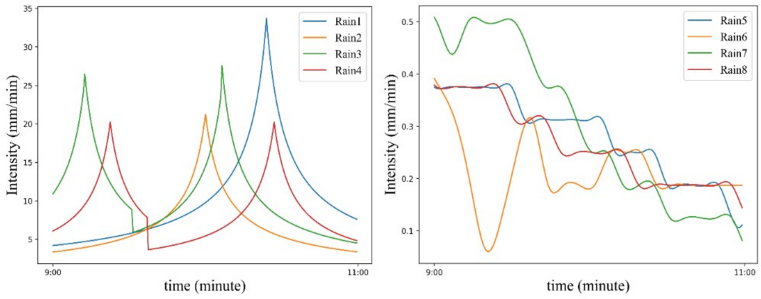

Fig.7 Test rainfalls. Designed rainfall and real rainfall

1. Emulators and RLs in the case study

First, 10 rainfall data with 2 hours duration are given through Eq.20 combined with 10 random actions sets are sent into SWMM model to obtain 100 sets of simulation results. Then, these data are used to train KEDMD and DLEDMD through the methods in Section 3.1. The observable x_t is vector including total CSO and flooding, stored water volume, inflow volume, and outflow volume.

The action a_t is the control strategy of 8 pumps in the system, which is an 8-dimensional 0-1 vector. The rain_t is the rain intensity of current time point.

Then these two emulators are used to train two types of RLs through the model-free framework given in Section 3.2. Considering the combination of model-based (SWMM) or model-free (KEDMD and DLEDMD), and DQN or PPO, we provide 6 RLs in this study, they are KEDMD-DQN, KEDMD-PPO, DLEDMD-DQN, DLEDMD-PPO, SWMM-DQN, and SWMM-PPO.

1. Results

5.1. Emulation of KEDMD and DLEDMD

The trained emulators are used to compare with SWMM in the simulation under 4 rainfalls given by Eq.20 and different actions of pumps. A linear model based on LassoLars linear regression (Efron et al., 2004), called Linear in this paper, is also provided for comparison. The MSE and NSE of these simulations are given in Table 4. The trajectories of total CSO and flooding, stored water, inflow, outflow of these two emulators as well as SWMM are given in Fig.8, from which shows that DLEDMD and KEDMD achieve better performance than Linear.

Table 4. MSE and NSE of KEDMD, DLEDMD, and Linear

		MSE	NSE
Test rain1	KEDMD	0.935	0.994
	DLEDMD	1.095	0.991
	Linear	1.516	0.983
Test rain2	KEDMD	1.139	0.989
	DLEDMD	1.143	0.989
	Linear	1.445	0.983
Test rain3	KEDMD	1.076	0.991
	DLEDMD	1.113	0.991
	Linear	1.477	0.984
Test rain4	KEDMD	1.200	0.972
	DLEDMD	1.169	0.973
	Linear	1.310	0.966

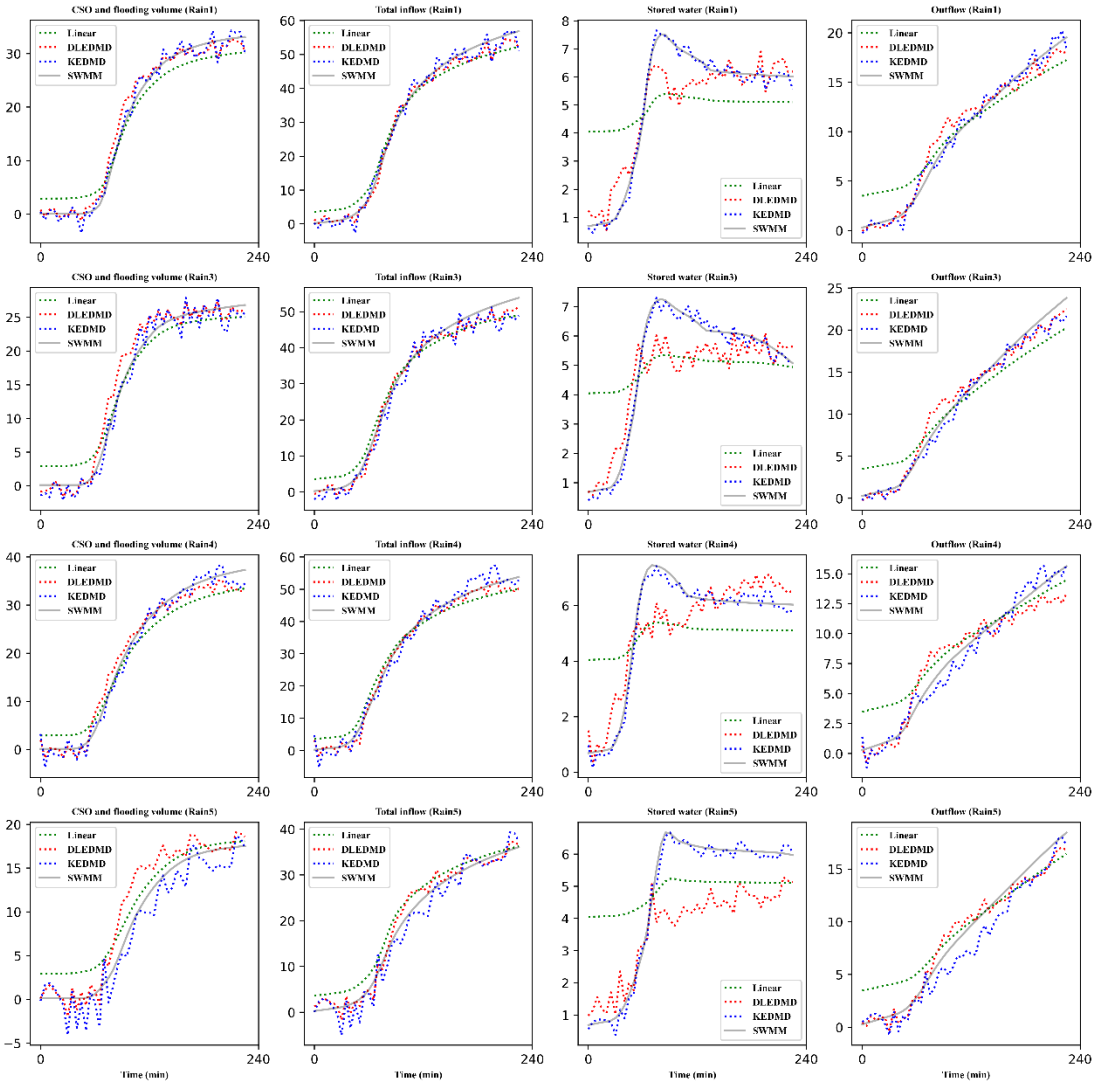

Fig.8 Simulations of KEDMD, DLEDMD, and Linear

5.2. Control effect, training time, and data usage of RLs

1. Control effect

We use the trained KEDMD and DLEDMD to replace the SWMM in the training of two RLs, DQN and PPO. And then use these trained RLs to control the combined sewer system (SWMM in this study) under Rain1-8. A water-level based control system and model-based RLs (SWMM-DQN and SWMM-PPO) are also given for comparison. The trajectories of the CSO and flooding of each RLs are shown in Fig.9 and Fig.10. The total CSO and flooding volume are

listed in Table 5. According to the results, the RLs achieve better control effect than the water-level system, proving that the model-free framework is capable of training RLs to achieve a similar control effect compared with the model-based method. Also, the DQNs are better in terms of the optimality of the control process than PPOs, which is consistent with the conclusion of previous researches (Mnih et al., 2016; Sutton & Barto, 2018).

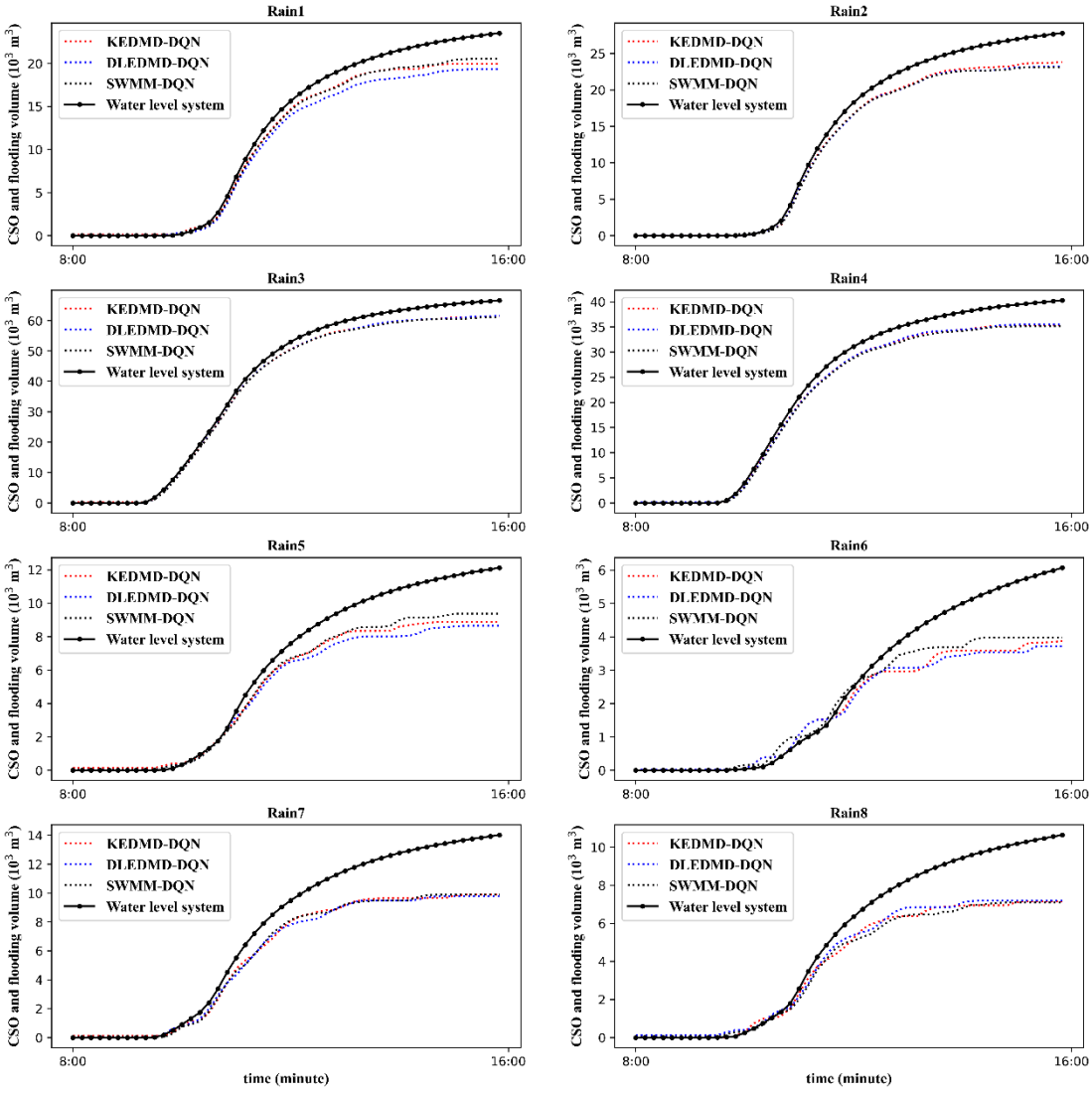

Fig.9 Trajectories of CSO and flooding volume of SWMM-DQN, KEDMD-DQN, and DLEDMD-DQN

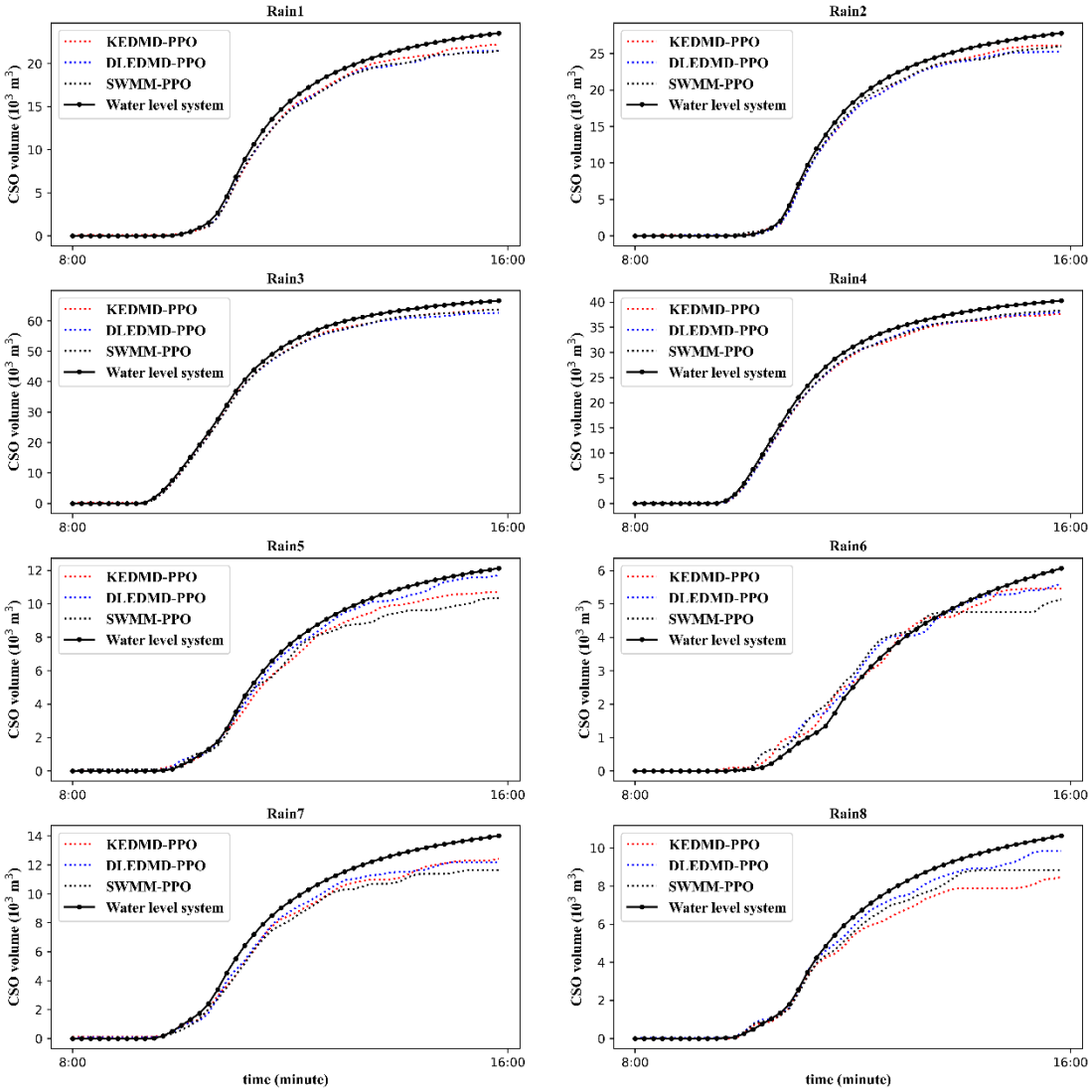

Fig.10 Trajectories of CSO and flooding volume of SWMM-PPO, KEDMD-PPO, and DLEDMD-PPO

Table 5. CSO and flooding volume (10^3) of each RLs

	KEDMD-DQN	KEDMD-PPO	DLEDMD-DQN	DLEDMD-PPO	SWMM-DQN	SWMM-PP
Rain1	19.975	22.132	19.310	21.482	20.519	21.341
Rain2	23.754	26.127	23.201	25.210	23.101	25.875
Rain3	61.297	63.686	61.495	62.604	61.047	63.670
Rain4	35.437	37.531	35.524	37.852	35.118	38.227

	KEDMD-DQN	KEDMD-PPO	DLEDMD-DQN	DLEDMD-PPO	SWMM-DQN	SWMM-PP
Rain5	8.884	10.696	8.654	11.729	9.377	10.358
Rain6	3.871	5.454	3.720	5.611	3.980	5.139
Rain7	9.882	12.417	9.802	12.189	9.888	11.654
Rain8	7.162	8.475	7.185	9.845	7.088	8.823

1. Training time and the efficiency of data usage

The training time and *DUR* of RLs are given in Table 6 and 7. It is obvious that model-free RLs achieve about 20 to 23 times faster than model-based RLs in training process under different sampling number. Also, their *DUR* is about 79.67 times higher than model-based RLs, indicating a better efficiency in data usage.

Table 6. Training time of different RLs with different batch size of sampling

	batch size = 20	batch size = 200	batch size = 2000
KEDMD-DQN	31 s	5 min 10 s	55 min 34 s
KEDMD-PPO	32 s	5 min 7 s	54 min 55 s
DLEDMD-DQN	31 s	5 min 12 s	55 min 20 s
DLEDMD-PPO	30 s	5 min 9 s	55 min 15 s
SWMM-DQN	10 min 58 s	1 h 48 min	18 h 32 min
SWMM-PPO	11 min 34 s	1 h 47 min	18 h 30 min

Table 7. *DUR* of RLs (batch size = 20)

	NSD	*NRD*	*DUR=NSD/NRD*
KEDMD-DQN/PPO	3920	4000	0.98
DLEDMD-DQN/PPO	3920	4000	0.98
SWMM-DQN/PPO	3920	317520	0.0123

5.3. Uncertainty analysis

5.3.1. The uncertainty of rainfalls

We use the method in the section 3.2.2 to generate 50 different rainfalls and use them to run the uncertainty analysis of different rainfalls. The control effect, represented by *RCI*, are given in Table 8. From the results, it can be seen that different RLs maintain a certain range of control effects when facing different rainfalls, achieving a certain degree of reliability. And also, DQNs achieve better control effect than PPOs, which is consistent with results of section 5.2.1 and the previous researches (Mnih et al., 2016; Sutton & Barto, 2018)

Table 8. The *RCI* of all the RLs in 50 different designed rainfalls

@ >p(- 6) * >p(- 6) * >p(- 6) * >p(- 6) * @ & 5% & 50% & 95%
DLEDMD-DQN

KEDMD-DQN

SWMM-DQN

DLEDMD-PPO

KEDMD-PPO

SWMM-PPO & 0.006 & 0.035 & 0.142
& 0.005 & 0.035 & 0.139
& 0.006 & 0.036 & 0.143
& 0.011 & 0.043 & 0.183
& 0.011 & 0.045 & 0.174
& 0.007 & 0.038 & 0.162

5.3.2. The uncertainty of imperfect input

For each of the Rain1-4, 50 trajectories based on the random states in section 3.3.3 are simulated and used as the reference of imperfect input uncertainty analysis. The range of control trajectories are given in Fig.11. It shows that all the trajectories of RLs with imperfect inputs achieve better performance than water level system. And their trajectories stay in a range during control process, indicates a certain robustness.

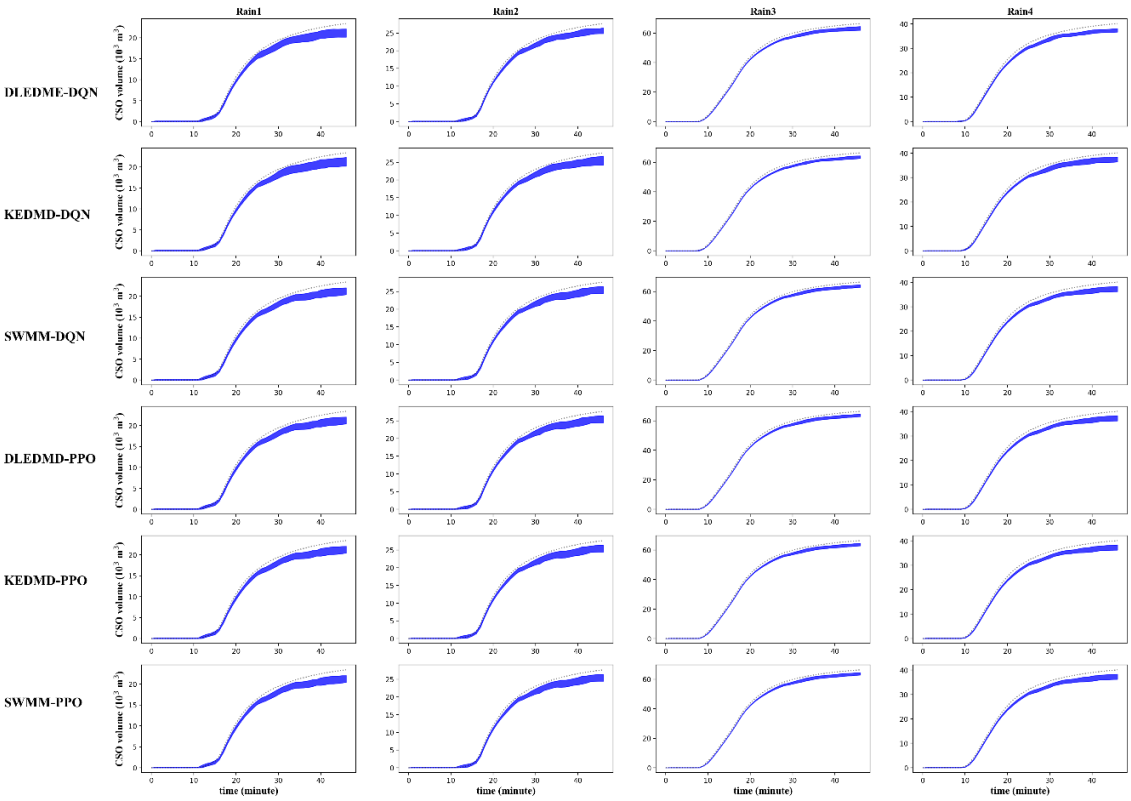

Fig.11 the range of RLs' control trajectories with respect to imperfect input

1. Discussions

6.1. The designing of RLs' state and the observable of emulators

According to the results in section 5.2.2, model-free RLs achieve short training time and better efficiency in data usage. The reason is that the Koopman emulators have an excellent nonlinear emulation capability and its observable dimension and the RL's state can be flexibly equalized. It means that the Koopman emulators are simplified models with linear structure and they only provide the data needed for controlling, while model-based methods use nonlinear model SWMM and compute a "full rank" data set but only use some of them.

In this view point, the training time and the *DUR* of model-free RLs are influenced by their designing. If the dimension of emulator variable is high and the size of model-free RL's state is low, then its computing time will be high, and its *DUR* will close to zero, indicating high demand of computing resource and low efficiency of data usage. Considering this, we believe that a feasible model-free framework should (i) equalize the dimension of emulator's variable and RL's state, and (ii) achieve an acceptable control effect. The cross validation from previous research (Wu & Noé, 2019) may help us in balancing these two aspects,

and will be present in our future works.

6.2. Explanation to the influence of uncertainty

The results in section 5.3 and previous research (Saliba et al., 2020) show that slight perturbation in training and testing process will not leverage the control effect of RLs. An intuitive explanation can be given from a statistical view. The main target of RLs is to use deep neural network and sampling data to find a conditional distribution of action given state $\mathrm{P}\left(a_t|s_t\right)$ and use it to achieve higher value function $V\left(s_{t+1}\right)$. The statistic model of this can be described through RL theory and Bayesian formula:

$$@ >\mathrm{p}(\text{- } 4) * >\mathrm{p}(\text{- } 4) * >\mathrm{p}(\text{- } 4) * @ \,\&\, V\left(s_{t+1}\right) = \max\left[r_{t+1} + \int \mathrm{P}\left(a_t|s_t\right) V\left(s_t\right) ds_t\right] \,\&\,$$

(24)

$$\&\ \mathrm{P}\left(a_t|s_t\right) = \frac{\mathrm{P}(s_t|a_t)\mathrm{P}(a_t)}{\mathrm{P}(s_t)}\ \&$$

The $\mathrm{P}\left(a_t|s_t\right)$ is the conditional probability of a_t given s_t. The $\mathrm{P}\left(s_t|a_t\right)$ is conditional probability of s_t given a_t, $\mathrm{P}\left(s_t\right)$ and $\mathrm{P}\left(a_t\right)$ are the probability distribution of system state and action. All the probabilities are influenced by both UDS and rainfall condition.

Theoretically, we use some data to training a neural network model to represent the conditional probability $\widehat{\mathrm{P}}\left(a_t|s_t,\ \theta\right)$, where θ is the trainable parameters of neural network. If the training process of RL is good, then we can say that $\widehat{\mathrm{P}}\left(a_t|s_t,\ \theta\right)$ and $\mathrm{P}\left(a_t|s_t\right)$ are close to each other, and we have

$$@ >\mathrm{p}(\text{- } 4) * >\mathrm{p}(\text{- } 4) * >\mathrm{p}(\text{- } 4) * @ \,\&\, \widehat{\mathrm{P}}\left(a_t|s_t,\ \theta\right) \approx \mathrm{P}\left(a_t|s_t\right) = \frac{\mathrm{P}(s_t|a_t)\mathrm{P}(a_t)}{\mathrm{P}(s_t)}\ \&$$

(25)

However, if we import randomness into this system, the right side will change and Eq.25 may not hold.

For instance, different rainfalls will lead to different state, causing various $\mathrm{P}\left(s_t\right)$ and $\mathrm{P}\left(s_t|a_t\right)$. Thus, the right side of Eq.25 will be changed and the trained $\widehat{\mathrm{P}}\left(a_t|s_t,\ \theta\right)$ may no longer an approximated solution of Eq.24, leading to different control effect. This is consistent with the results in Table 8, which shows that the control effects of one RL are different under various rainfalls.

For imperfect input, the uniform distribution is used as a randomness of input. Thus, the distribution of state and conditional probability of s_t given a_t will be changed as:

@ >p(- 4) * >p(- 4) * >p(- 4) * @ & $\overline{\mathrm{P}\left(s_t|a\right)} = \mathrm{P}(X)\mathrm{P}\left(s_t|a\right),\ \overline{\mathrm{P}\left(s_t\right)} = \mathrm{P}(X)\mathrm{P}\left(s_t\right),\ X \sim U(0.95, 1.05)$ &

(26)

And we have

@ >p(- 4) * >p(- 4) * >p(- 4) * @ & $V\left(s_{t+1}\right) = \max\left[r\left(s_{t+1}\right) + \int \widetilde{\mathrm{P}}\left(a_t|s_t\right) V\left(s_t\right) ds_t\right]$ &

(27)

& $\widetilde{\mathrm{P}}\left(a_t|s_t\right) = \frac{\mathrm{P}(s_t|a_t)\mathrm{P}(X)\mathrm{P}(a_t)}{\mathrm{P}(s_t)\mathrm{P}(X)}$ &

If we suppose the $\mathrm{P}\left(s_t\right)$ and $\mathrm{P}\left(s_t|a_t\right)$ are normal distributions, then the $\overline{\mathrm{P}\left(s_t\right)}$ and $\overline{\mathrm{P}\left(s_t|a\right)}$ are also normal distributions with same expectation value and slight changed variance. Therefore, the X does not statistically change the control system and the solution given by Eq.24 can still be used as an approximated solution of Eq.27, thus the influence of X to the control trajectories is not obvious. This is consistent with the Fig.11 that the RLs have a certain robustness to imperfect input.

6.3. Two factors influence the performance of emulators

From the results in section 5.1, we can see that Koopman emulators achieve better performance than linear model. The reason can be abstracted into two aspects: (i) The Koopman emulators are more suitable to nonlinear dynamic system, and (ii) the target of Koopman emulators is to find a linear structure behind the nonlinear system, rather than pure reduce regression error. However, these emulators are not perfect. Although they handle the nonlinear characteristic well, some approximation errors still exist. Intuitively, approximation error is hard to avoid when we use approximation algorithms to obtain Koopman emulators. And we find that their errors are related to the selection of hyperparameters (Wu & Noé, 2019) and training data.

1. Hyperparameters

For the first one, we test KEDMD and DLEDMD with different hyperparameters, and their performance are given Table 9 and 10. These results show that different selection of hyperparameters will lead to different emulation performance, thus, how to select the best hyperparameter set with respect to a given training data set is still worth exploring. Usually, the "trial and error" is used to achieve this target (Tian et al., 2019), but we believe that the posterior distribution of hyperparameters will be able to provide a better method for selecting hyperparameters (Liao et al., 2020). More researches will be given in our future studies.

Table 9. The performance of KEDMD with different kernel functions

		MSE	NSE
Test rain1	Polynomial kernel	2.739	0.945
	Lapacian kernel	0.808	0.995
	Chi-squared kernel	0.935	0.994
Test rain2	Polynomial kernel	17.752	-1.585
	Lapacian kernel	1.448	0.983
	Chi-squared kernel	1.139	0.989
Test rain3	Polynomial kernel	10.787	0.136
	Lapacian kernel	0.992	0.993
	Chi-squared kernel	1.076	0.991
Test rain4	Polynomial kernel	3.866	0.707
	Lapacian kernel	1.726	0.942
	Chi-squared kernel	1.200	0.972

Table 10. The performance of DLEDMD with different architectures, where N is the dimension of observable or dynamic variable.

		MSE	NSE
Test rain1	N-10-N	1.356	0.987
	N-10-10-N	0.790	0.995
	N-10-10-10-N	0.781	0.996
Test rain2	N-10-N	1.431	0.983
	N-10-10-N	0.903	0.993
	N-10-10-10-N	0.919	0.993
Test rain3	N-10-N	1.677	0.979
	N-10-10-N	1.275	0.988
	N-10-10-10-N	0.949	0.993
Test rain4	N-10-N	1.492	0.956
	N-10-10-N	1.011	0.980
	N-10-10-10-N	1.033	0.979

1. Training data

We test KEDMD and DLEDMD under different training and testing data. First, we use Eq.20 with different parameter P to provide 4 rainfall data and sent them into the SWMM to obtain 4 set of simulation results. In this case, other parameters of Eq.20 are fixed, and all the 8 pumps are replaced as pipeline. Then, we use the simulation results given by the rainfall with P=1 as training data to train the DLEDMD and KEDMD. After that, these trained emulators are used to emulate other three rainfalls which have P=3, 5, 10. The MSE and NSE of all emulations are provided in Table 11, from which we can see that their MSE increase with the parameter P and their NSE decrease with the parameter

P, indicating that the performance of Koopman emulators in UDS is related to the similarity between training data and test data. That is, a trained Koopman emulator will achieve better emulation if the test data is similar to its training data. From this viewpoint, we believe that using different types of rainfall data to provide "rich" training data set is a reasonable solution and is already used in the above test. More details and theoretical analysis will be provided in our future research.

Table 11. MSE and NSE of RLs under different parameter P.

P								
KEDMD	MSE	NSE	MSE	NSE	MSE	NSE	MSE	NSE
DLEDMD								

1. Conclusion

Although model-based RL achieved a milestone of urban water management in the direction of the smart city, some problems still exit, including large demand of computing time, low efficiency of data usage, and zero noise training environment. To address these problems, a model-free framework is provided through Koopman emulators to (i) reduce the training time, (ii) improve the efficiency of data usage, and (iii) provide a training environment with certain randomness. Specifically, we take the advantage of the nonlinear emulation capability of Koopman emulator and the equalization between observable's dimension and RL's state to achieve faster RL training process and efficient data usage. Meanwhile, since the emulator is simplified model, its error and noise are able to provide randomness for RL training process. According to the results, several conclusions can be given:

1. Compared with model-based RLs, the model-free framework achieves a similar control effect with a 20 to 23 times faster training process and 79.67 times higher efficiency of data usage. Also, it provides a training environment with some randomness.
2. Slight perturbation which does not statistically change the control system in training and testing process will not leverage the control effect of both model-based and model-free RLs.
3. The Koopman emulators is able to be used as the surrogate model of UDS. Their performances are related to the hyperparameters and the similarity between training data and test data.

Acknowledgement

This study was financially supported by the Major Science and Technology Program for Water Pollution Control and Treatment (grant no. 2018ZX07208006)

and the National Natural Science Foundation of China (grant no. 51778451). We also thank the 111 Project (B13017) of Tongji University.

The calibrated SWMM model in the case study is available in these references: Liao et al., (2019) and Zhi et al., (2019). The real rainfall data is provided by http://caiyunapp.com/index.html#api.

Reference

Batalini de Macedo, M., Nóbrega Gomes Júnior, M., Pereira de Oliveira, T. R., H. Giacomoni, M., Imani, M., Zhang, K., Ambrogi Ferreira do Lago, C., & Mendiondo, E. M. (2021). Low Impact Development practices in the context of United Nations Sustainable Development Goals: A new concept, lessons learned and challenges. Critical Reviews in Environmental Science and Technology, 1–44. https://doi.org/10.1080/10643389.2021.1886889

Brunton, S. L., Brunton, B. W., Proctor, J. L., & Kutz, J. N. (2016). Koopman Invariant Subspaces and Finite Linear Representations of Nonlinear Dynamical Systems for Control. PLOS ONE, 11(2), e0150171. https://doi.org/10.1371/journal.pone.0150171

Budišić, M., Mohr, R. M., & Mezić, I. (2012). Applied Koopmanism. Chaos: An Interdisciplinary Journal of Nonlinear Science, 22(4), 047510. https://doi.org/10.1063/1.4772195

Castelletti, A., Pianosi, F., & Restelli, M. (2013). A multiobjective reinforcement learning approach to water resources systems operation: Pareto frontier approximation in a single run: MOFQI for Large-Scale Water Resources Systems Operation. Water Resources Research, 49(6), 3476–3486. https://doi.org/10.1002/wrcr.20295

Carbajal, J. P., Leitão, J. P., Albert, C., & Rieckermann, J. (2017). Appraisal of data-driven and mechanistic emulators of nonlinear simulators: The case of hydrodynamic urban drainage models. Environmental Modelling & Software, 92, 17–27. https://doi.org/10.1016/j.envsoft.2017.02.006

Chan, F. K. S., Griffiths, J. A., Higgitt, D., Xu, S., Zhu, F., Tang, Y.-T., Xu, Y., & Thorne, C. R. (2018). "Sponge City" in China—A breakthrough of planning and flood risk management in the urban context. Land Use Policy, 76, 772–778. https://doi.org/10.1016/j.landusepol.2018.03.005

Chen, C., Seff, A., Kornhauser, A., & Xiao, J. (2015). DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving. 2015 IEEE International Conference on Computer Vision (ICCV), 2722–2730. https://doi.org/10.1109/ICCV.2015.312

Chua, K., Calandra, R., McAllister, R., & Levine, S. (2018). Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. ArXiv:1805.12114 [Cs, Stat]. http://arxiv.org/abs/1805.12114

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., Ishwaran, H., Knight, K.,

Ridgeway, G. (2004). Least angle regression. Annals of Statistics, 32(2), 407–499.

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. https://doi.org/10.1016/j.advwatres.2015.08.007

Hasselt, H. (2010). Double Q-learning. Advances in Neural Information Processing Systems, 23, 2613–2621.

Joseph-Duran, B., Ocampo-Martinez, C., & Cembrano, G. (2015). Output-feedback control of combined sewer networks through receding horizon control with moving horizon estimation. Water Resources Research, 51(10), 8129–8145. https://doi.org/10.1002/2014WR016696

Junge, O., & Koltai, P. (2009). Discretization of the Frobenius–Perron Operator Using a Sparse Haar Tensor Basis: The Sparse Ulam Method. SIAM Journal on Numerical Analysis, 47(5), 3464–3485. https://doi.org/10.1137/080716864

Kalweit, G. & Boedecker, J. (2017). Uncertainty-driven Imagination for Continuous Deep Reinforcement Learning. Proceedings of the 1st Annual Conference on Robot Learning, in PMLR 78:195-206

Kerkez, B., Gruden, C., Lewis, M., Montestruque, L., Quigley, M., Wong, B., et al. (2016). Smarter stormwater systems. ACS Publications.

Klus, S., Nüske, F., Koltai, P., Wu, H., Kevrekidis, I., Schütte, C., & Noé, F. (2018). Data-driven model reduction and transfer operator approximation. Journal of Nonlinear Science, 28(3), 985–1010. https://doi.org/10.1007/s00332-017-9437-7

Klus, S., Koltai, P., & Schütte, C. (2016). On the numerical approximation of the Perron-Frobenius and Koopman operator. Journal of Computational Dynamics, 3(1), 1–12. https://doi.org/10.3934/jcd.2016003

Klus, S., Husic, B. E., Mollenhauer, M., & Noé, F. (2019). Kernel methods for detecting coherent structures in dynamical data. Chaos: An Interdisciplinary Journal of Nonlinear Science, 29(12), 123112. https://doi.org/10.1063/1.5100267

Klus, S., Bittracher, A., Schuster, I., & Schütte, C. (2018). A kernel-based approach to molecular conformation analysis. The Journal of Chemical Physics, 149(24), 244109. https://doi.org/10.1063/1.5063533

Korda M. & Mezić I., (2018). Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. Automatica, 93: 149-160, 0005-1098. https://doi.org/10.1016/j.automatica.2018.03.046

Labadie J W., 2014. Advances in Water Resources Systems Engineering: Applications of Machine Learning[M]//Modern Water Resources Engineering. Humana Press, Totowa, NJ: 467–523.

Li, Q., Dietrich, F., Bollt, E. M., & Kevrekidis, I. G. (2017). Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the Koopman operator. Chaos: An Interdisciplinary Journal of Nonlinear Science, 27(10), 103111.

Liao Z, Gu X, Xie J, Wang X, Chen J. An integrated assessment of drainage system reconstruction based on a drainage network model. Environ Sci Pollut Res Int. 2019 Sep;26(26):26563-26576. https://doi.org/10.1007/s11356-019-05280-1

Liao, Z., Tian, W. & Wang, X. (2020). Responses to the letter on "Transfer learning for neural network model in Chlorophyll-a dynamics prediction". Environ Sci Pollut Res 27, 39667–39668. https://doi.org/10.1007/s11356-020-09394-9

Lund, N. S. V., Borup, M., Madsen, H., Mark, O., & Mikkelsen, P. S. (2020). CSO Reduction by Integrated Model Predictive Control of Stormwater Inflows: A Simulated Proof of Concept Using Linear Surrogate Models. Water Resources Research, 56(8). https://doi.org/10.1029/2019WR026272

Lund, N. S. V., Falk, A. K. V., Borup, M., Madsen, H., & Steen Mikkelsen, P. (2018). Model predictive control of urban drainage systems: A review and perspective towards smart real-time water management. Critical Reviews in Environmental Science and Technology, 48(3), 279–339. https://doi.org/10.1080/10643389.2018.1455484

Machac, D., Reichert, P., Rieckermann, J., Del Giudice, D., & Albert, C. (2018). Accelerating Bayesian inference in hydrological modeling with a mechanistic emulator. Environmental Modelling & Software, 109, 66–79. https://doi.org/10.1016/j.envsoft.2018.07.016

Madani, K., & Hooshyar, M. (2014). A game theory–reinforcement learning (GT–RL) method to develop optimal operation policies for multi-operator reservoir systems. Journal of Hydrology, 519, 732–742. https://doi.org/10.1016/j.jhydrol.2014.07.061

Mardt, A., Pasquali, L., Wu, H., & Noé, F. (2018). VAMPnets for deep learning of molecular kinetics. Nature Communications, 9(1), 5. https://doi.org/10.1038/s41467-017-02388-1

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., et al. (2016). Asynchronous methods for deep reinforcement learning. In International conference on machine learning (pp. 1928–1937).

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(7540), 529–533.

Mullapudi, A., Lewis, M. J., Gruden, C. L., & Kerkez, B. (2020). Deep reinforcement learning for the real time control of stormwater systems. Advances in Water Resources, 140, 103600. https://doi.org/10.1016/j.advwatres.2020.103600

Nagel, J. B., Rieckermann, J., & Sudret, B. (2020). Principal component analysis and sparse polynomial chaos expansions for global sensitivity analysis and model calibration: Application to urban drainage simulation. Reliability Engineering & System Safety, 195, 106737. https://doi.org/10.1016/j.ress.2019.106737

Noé, F., & Nüske, F. (2012). A variational approach to modeling slow processes in stochastic dynamical systems. ArXiv:1211.7103 [Math-Ph, Physics:Physics, Stat]. http://arxiv.org/abs/1211.7103

Ochoa, D., Riano-Briceno, G., Quijano, N., & Ocampo-Martinez, C. (2019). Control of Urban Drainage Systems: Optimal Flow Control and Deep Learning in Action. 2019 American Control Conference (ACC), 4826–4831. https://doi.org/10.23919/ACC.2019.8814958

Otto, S. E., & Rowley, C. W. (2019). Linearly-Recurrent Autoencoder Networks for Learning Dynamics. ArXiv:1712.01378 [Cs, Math, Stat]. http://arxiv.org/abs/1712.01378

Page, J., & Kerswell, R. R. (2018). Koopman analysis of Burgers equation. Physical Review Fluids, 3(7), 071901. https://doi.org/10.1103/PhysRevFluids.3.071901

Peitz, S., & Klus, S. (2019). Koopman operator-based model reduction for switched-system control of PDEs. Automatica, 106, 184–191. https://doi.org/10.1016/j.automatica.2019.05.016

Qi, W., Ma, C., Xu, H., Chen, Z., Zhao, K., & Han, H. (2021). A review on applications of urban flood models in flood mitigation strategies. Natural Hazards. https://doi.org/10.1007/s11069-021-04715-8

Rowley, C. W., Mezić, I., Bagheri, S., Schlatter, P., & Henningson, D. S. (2009). Spectral analysis of nonlinear flows. Journal of Fluid Mechanics, 641, 115–127. https://doi.org/10.1017/S0022112009992059

Saliba, S. M., Bowes, B. D., Adams, S., Beling, P. A., & Goodall, J. L. (2020). Deep Reinforcement Learning with Uncertain Data for Real-Time Stormwater System Control and Flood Mitigation. Water, 12(11), 3222. https://doi.org/10.3390/w12113222

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In International conference on machine learning (pp. 1889–1897).

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. ArXiv Preprint ArXiv:1707.06347.

Schütze, M., Campisano, A., Colas, H., Schilling, W., & Vanrolleghem, P. A. (2004). Real time control of urban wastewater systems—Where do we stand today? Journal of Hydrology, 299(3–4), 335–348. https://doi.org/10.1016/j.jhydrol.2004.08.010

Shao, K., Zhu, Y., & Zhao, D. (2018). Starcraft micromanagement with reinforcement learning and curriculum transfer learning. IEEE Transactions on Emerging Topics in Computational Intelligence, 3(1), 73–84.

Song, Y., Li, Y., Li, C., & Zhang, G. (2012). An efficient initialization approach of Q-learning for mobile robots. International Journal of Control, Automation and Systems, 10(1), 166–172.

Sun, C., Romero, L., Joseph-Duran, B., Meseguer, J., Muñoz, E., Guasch, R., Martinez, M., Puig, V., & Cembrano, G. (2020). Integrated pollution-based real-time control of sanitation systems. Journal of Environmental Management, 269, 110798. https://doi.org/10.1016/j.jenvman.2020.110798

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Tian, W., & Wu, H. (2021). Kernel Embedding Based Variational Approach for Low-Dimensional Approximation of Dynamical Systems. Computational Methods in Applied Mathematics, 21(3), 635–659. https://doi.org/10.1515/cmam-2020-0130

Tian, W., Liao, Z., & Wang, X. (2019). Transfer learning for neural network model in chlorophyll-a dynamics prediction. Environmental science and pollution research international, 26(29), 29857–29871. https://doi.org/10.1007/s11356-019-06156-0

van Daal, P., Gruber, G., Langeveld, J., Muschalla, D., & Clemens, F. (2017). Performance evaluation of real time control in urban wastewater systems in practice: Review and perspective. Environmental Modelling & Software, 95, 90–101. https://doi.org/10.1016/j.envsoft.2017.06.015

Williams, M. O., Rowley, C. W., Mezić, I., & Kevrekidis, I. G. (2015b). Data fusion via intrinsic dynamic variables: An application of data-driven Koopman spectral analysis. EPL (Europhysics Letters), 109(4), 40007. https://doi.org/10.1209/0295-5075/109/40007

Williams, M. O., Kevrekidis, I. G., & Rowley, C. W. (2015a). A Data-Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition. Journal of Nonlinear Science, 25(6), 1307–1346. https://doi.org/10.1007/s00332-015-9258-5

Wu, H., & Noé, F. (2019). Variational approach for learning Markov processes from time series data. ArXiv:1707.04659 [Math, Stat]. http://arxiv.org/abs/1707.04659

Xie, J., Chen, H., Liao, Z., Gu, X., Zhu, D., & Zhang, J. (2017). An integrated

assessment of urban flooding mitigation strategies for robust decision making. Environmental Modelling & Software, 95, 143–155.

Yazdi, J. (2018). Rehabilitation of Urban Drainage Systems Using a Resilience-Based Approach. Water Resources Management, 32(2), 721–734. https://doi.org/10.1007/s11269-017-1835-y

Zhi, G., Liao, Z., Tian, W., Wang, X., & Chen, J. (2019). A 3D dynamic visualization method coupled with an urban drainage model. Journal of Hydrology, 577, 123988. https://doi.org/10.1016/j.jhydrol.2019.123988

Zhi, G., Liao, Z., Tian, W., & Wu, J. (2020). Urban flood risk assessment and analysis with a 3D visualization method coupling the PP-PSO algorithm and building data. Journal of Environmental Management, 268, 110521. https://doi.org/10.1016/j.jenvman.2020.110521