UDM 2018
11th International Conference on Urban Drainage Modelling
23-26 Sep | Palermo - Italy

Leveraging Open Source Software and Parallel Computing for Model Predictive Control Simulation of Urban Drainage Systems using EPA-SWMM5 and Python

Jeffrey M. Sadler[1], Jonathan L. Goodall[1,2], Madhur Behl[2,3], and Mohamed M. Morsy[1,4]
[1]University of Virginia, Civil and Environmental Engineering, Charlottesville, Virginia, USA
[2]University of Virginia, Computer Science, Charlottesville, Virginia, USA
[3]University of Virginia, Systems and Information Engineering, Charlottesville, Virginia, USA
[4]Cairo University, Irrigation and Hydraulics Department, Giza, Egypt

Abstract: The active control of stormwater systems is a potential solution to increased street flooding in low-lying, low-relief coastal cities due to climate change and accompanying sea level rise. Model predictive control (MPC) has been shown to be a successful control strategy generally and as well as for managing urban drainage specifically. This research describes and demonstrates the implementation of MPC for urban drainage systems using open source software (Python and The United States Environmental Protection Agency (EPA) Storm Water Management Model (SWMM5). The system was demonstrated using a simplified use case in which an actively-controlled outlet of a detention pond is simulated. The control of the pond's outlet influences the flood risk of a downstream node. For each step in the SWMM5 model, a series of policies for controlling the outlet are evaluated. The best policy is then selected using an evolutionary algorithm. The policies are evaluated against an objective function that penalizes primarily flooding and secondarily deviation of the detention pond level from a target level. Freely available Python libraries provide the key functionality for the MPC workflow: step-by-step running of the SWMM5 simulation, evolutionary algorithm implementation, and leveraging parallel computing. For perspective, the MPC results were compared to results from a rule-based approach and a scenario with no active control. The MPC approach produced a control policy that largely eliminated flooding (unlike the scenario with no active control) and maintained the detention pond's water level closer to a target level (unlike the rule-based approach).

Keywords: Active stormwater control**;** Model predictive control; Flood prevention; Low-relief coastal cities

1. INTRODUCTION

Stress on stormwater systems will likely increase as warmer global temperatures are predicted to cause more intense storm events on average (Berggren et al., 2012). At the same time, the effectiveness of coastal cities' stormwater systems will likely decrease due to sea level rise which reduces the already limited elevation head needed to drain stormwater from streets to receiving bodies. Since significant changes to existing stormwater infrastructure in coastal cities is often cost prohibitive, other options are needed to increase its effectiveness. An option for increasing the effectiveness of stormwater infrastructure is to actively manage the existing stormwater infrastructure, making it a "smart" system (Kerkez et al., 2016). This approach does not increase the actual capacity of stormwater infrastructure, but rather more efficiently uses the existing infrastructure, increasing its effective capacity. An example of active management of stormwater infrastructure would be the use of an automated valve at the outlet of a detention basin which can be opened or closed based on conditions and forecasts.

The effective control the actuators in a stormwater system can have a large impact on the system's ability to achieve its objective (e.g., minimize flooding). One approach to determining the optimum control policy for a system (i.e., which actuators should change, when to change them, and to what setting) is model predictive control (MPC). MPC has been used effectively in urban drainage scenarios by Gelormino and Ricker (1994). The non-linear behaviour of the urban drainage system, especially in coastal environments where backwater effects have to be considered, makes optimization difficult. These authors therefore converted the system into a linear approximation greatly simplifying the selection of the optimum control policy.

Another implementation of MPC was done by Heusch and Ostrowski (2011). Their approach used the United States Environmental Protection Agency's (EPA) Storm water Management Model version 5 (SWMM5) a public domain and widely used distributed dynamic rainfall-runoff model. Heusch and Ostrowski used SWMM5 to simulate the non-linear dynamics of their urban drainage system as an "opaque" model meaning that the control policies were evaluated by the model without considering the mathematical form of the governing equations. This approach precludes the possibility of guaranteed optimality (a "practical optimum" can be found using a metaheuristic such as an evolutionary algorithm), but maintains the non-linear dynamics of the system.

Although Heusch and Ostrowski (2011) developed software that implements MPC with SWMM5, there were some drawbacks to their approach including sustainability and availability of the software which was closed-source and is no longer available. The main objective of this study was to create an open-source implementation of MPC for SWMM5. An additional objective was to leverage parallel computing since a computationally-expensive metaheuristic is needed. To accomplish these objectives, the open-source Python programming language was used in conjunction with SWMM5. To evaluate the MPC implementation, it was applied to a simplified use case. The MPC results were compared to the results from two other scenarios applied to the same use case: a rules-based approach and a scenario with no active control. The remainder of this abstract describes the methods used to implement the open-source MPC and the results of the evaluation. Finally a brief conclusion is given.

2. Methods

2.1 MPC overview

MPC consists of the following components: 1) information from the system, 2) a process model which accepts input from the system and is used to simulate the effect of a given control policy, and 3) an optimization routine to determine the optimum control policy. In MPC the optimization routine is performed at each control time step. For this routine: 1) system states are read from the system, 2) a series of control policies is evaluated, 3) the best control policy is selected, and 4) the best control policy is implemented. Although the best control policy is obtained for the entire control horizon, only the first step in the control policy is used since the procedure occurs at every control time step.

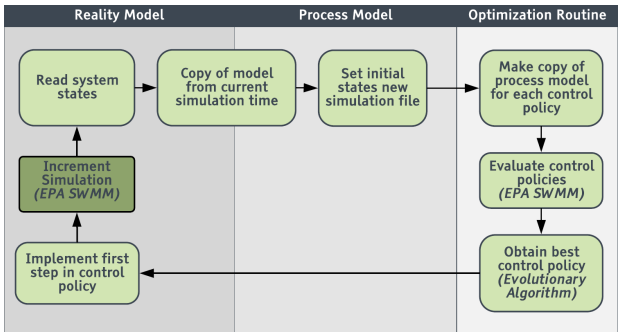

Figure 1. MPC workflow

2.2 MPC Implementation for SWMM5 using Python

The design of our MPC implementation for SWMM5 using the Python programming language used three main Python libraries: pyswmm (https://github.com/OpenWaterAnalytics/pyswmm), Distributed Evolutionary Algorithms for Python (DEAP) (https://github.com/DEAP/deap), and Scalable COncurrent Operations in Python (SCOOP) (https://github.com/soravux/scoop). The pyswmm library provides a Python interface to the SWMM5 model which is written in the C programming language. Through pyswmm, a SWMM5 model can be run step-by-step. This is a critical functionality for MPC since the best control policy needs to be found at each control time step. The DEAP library is used to select the best control policy using an evolutionary algorithm. The SCOOP library provides functionality for parallelizing the evolutionary algorithm execution.

The Python MPC workflow is shown in Figure 1. For each control time step, the system states are read from "reality". In our case we are simulating "reality" with a SWMM5 model, termed "reality model" in the figure. The system states that are read from the "reality model" are the heads at each node in the system and the flows at each link. Next these states are written to another SWMM5 model the "process model" in the figure. The DEAP library uses an evolutionary algorithm to select the practically optimum policy. To do this, many simulation runs of the process model are executed (one for each control policy) which is computationally expensive. Since the model runs are independent, this process can be parallelized using the functionality provided by the SCOOP library. The best policy selected by the evolutionary algorithm is returned to the "reality model" and implemented. The next time step is then executed and the process repeats.

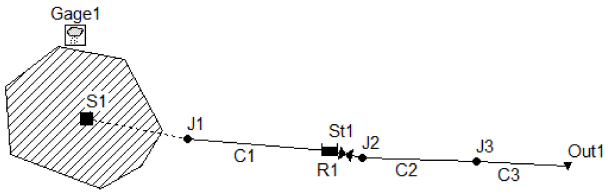

Figure 2. Schematic of simplified use case model

2.3 MPC Evaluation

A simplified use case motivated by an actual flood-prone area in Norfolk, Virginia USA was used to evaluate our MPC implementation. A simple schematic of this is shown in Figure 2. In the use case a detention pond (St1) is upstream from a node (J3) at which we would like to minimize flooding. The active control of the outlet of the pond is simulated using an "orifice" structure in SWMM5 which can have a setting between 0 (completely closed) and 1 (completely open). Therefore the control policies for the use case were an array of settings between 0 and 1, one setting for each control time step in the control horizon. To reduce the number of possible control settings to be evaluated by the computationally expensive evolutionary algorithm, the settings were constrained to be even tenths (e.g., 0.1, 0.2). The simulated rainfall event was an arbitrary synthetic event shown in Table 1. A control horizon of 6 hours was used with a control time step of 15 minutes. For the evolutionary algorithm, 8 generations were evaluated and the initial generation population was 80 individual policies.

Table 1. Rainfall data for evaluation use case

Time	Rainfall depth [mm]
04:00	6.35
05:00	12.7
06:00	10.16
07:00	6.35
08:00	3.175

In the use case, the control policies were evaluated using SWMM5 based on the following objective function

$$Cost = \alpha F_{st} + \beta F_{ds} + \phi D_{st} \quad (1)$$

Where F_{st} is the total volume of flooding from the storage node in millions of gallons (1 gallon = 3.785 liter), F_{ds} is the total volume from the downstream node J3 in millions of gallons, and D_{st} is the average deviation from a target level for the detention pond in feet (1 foot in this case) (1 foot = 0.3048 meter). The α, β and ϕ values are weight coefficients. In our case these values were 100, 100, and 0.05, respectively. These values were chosen to give more weight to flooding than to deviation from the target level at the storage unit.

3. Results

The MPC implementation was successful at running as described above including the use of multiple processing cores through the SCOOP library. The control policy resulting from the MPC in the evaluation use case significantly reduced flooding at the downstream node compared to the passive control (0.01 million gallons of flooding compared to 0.05 million gallons, respectively). Additionally, the MPC policy was able maintain the depth at the storage node closer to the target value, something the rules-based approach was not able to do (see Figure 3).

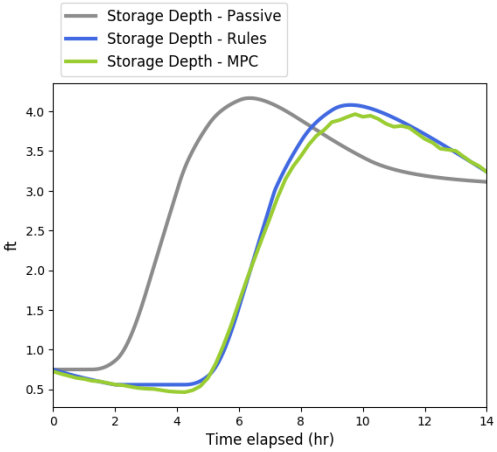

Figure 3. Depth of water in detention pond for three control scenarios

4. Conclusions

The objective of this research was to develop, implement, and evaluate an open-source solution for model predictive control (MPC) for the United States Environmental Protection Agency's (EPA) Stormwater Management Model (SWMM). The implementation was accomplished using the Python programming language and key Python libraries for step-by-step running of the model, use of evolutionary algorithms, and parallel computing. The system worked as designed and the resulting control policy significantly reduced flooding compared to a situation with no control in a simple, simulation use case. The MPC implementation described here can be used to perform MPC for any control in a SWMM5 model and could be useful for understanding the potential utility of smarter stormwater systems. The code is accessible at https://github.com/uva-hydroinformatics/swmm_mpc. Future improvements may include adjusting weights in the objective function and taking advantage of SWMM5's hotstart file capabilities to ensure consistency between the reality model and the process model.

References

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. J. Hydrol. Eng. 17, 92–98. https://doi.org/10.1061/(ASCE)HE.1943-5584.0000406

Gelormino, M.S., Ricker, N.L., 1994. Model-predictive control of a combined sewer system. Int. J. Control 59, 793–816. https://doi.org/10.1080/00207179408923105

Heusch, S., Ostrowski, M., 2011. Model Predictive Control with SWMM. J. Water Manag. Model. https://doi.org/10.14796/JWMM.R241-14

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. Environ. Sci. Technol. 50, 7267–7273. https://doi.org/10.1021/acs.est.5b05870