# MatSWMM - An Open-Source Toolbox for Designing Real-Time Control of Urban Drainage Systems

G. Riaño-Briceño[a,b,*], J. Barreiro-Gomez[a,c], A. Ramirez-Jaime[a], N. Quijano[a], C. Ocampo-Martinez[c]

[a] *Departamento de Ingeniería Eléctrica y Electrónica, Universidad de los Andes, Carrera 1A No 18A-10, Bogotá, Colombia*
[b] *Departamento de Ingeniería Civil, Universidad de los Andes, Carrera 1A No 18A-10, Bogotá, Colombia*
[c] *Automatic Control Department, Universitat Politècnica de Catalunya, Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Llorens i Artigas, 4-6, 08028 Barcelona, Spain*

**Abstract**

This manuscript describes the MatSWMM toolbox, an open-source Matlab, Python, and LabVIEW-based software package for the analysis and design of real-time control (RTC) strategies in urban drainage systems (UDS). MatSWMM includes control-oriented models of UDS, and the storm water management model (SWMM) of the US Environmental Protection Agency (EPA), as well as systematic-system edition functionalities. Furthermore, MatSWMM is also provided with a population-dynamics-based controller for UDS with three of the fundamental dynamics, i.e., the Smith, projection, and replicator dynamics. The simulation algorithm, and a detailed description of the features of MatSWMM are presented in this manuscript in order to illustrate the capabilities that the tool has for educational and research purposes.

*Keywords:* Open-Source Toolbox, Urban Drainage System Integrated Model, Real-Time Control, Matlab, Python, LabVIEW

## Software availability

Name of software: MatSWMM.
Contact address: Gerardo Riaño-Briceño, School of Engineering, Universidad de los Andes, Carrera 1E 19A - 40; ga.riano949@uniandes.edu.co.
Year first available: 2015.

*Corresponding author: Tel.: +57 3016687553
*Email addresses:* ga.riano949@uniandes.edu.co (G. Riaño-Briceño), j.barreiro135@uniandes.edu.co - jbarreiro@iri.upc.edu (J. Barreiro-Gomez), af.ramirez236@uniandes.edu.co (A. Ramirez-Jaime), nquijano@uniandes.edu.co (N. Quijano), cocampo@iri.upc.edu (C. Ocampo-Martinez)

Software requirements: Windows x86 and x64 environments with Matlab 2008 (or higher), Python (2.5), and/or LabVIEW 2012 (or higher).
Program size: $\tilde{1}$5 Mb.
Availability: Open-source repository, licensed under the GNU General Public License v3.0 (`https://github.com/water-systems/MatSWMM`).

## 1. Introduction

Software packages for urban drainage systems (UDS) can be categorized into two classes of tools: commercial and open source. The former ones are popular among industry and utilities because of their all-in-one capabilities and their efficiency. Some of the commercial packages available in the market are CivilStorm, InfoWorks CS, MOUSE, MIKE, SewerCAD, SOBEK-Urban, MicroDrainage, and SIMBA [1]. Most of these are holistic [2], i.e., these packages allow to model not only flow-routing through a drainage network, but also real-time control (RTC), rainfall, runoff, water quality, street flooding (1D and 2D), and many other features. Despite of the completeness of commercial packages, these tools can be inappropriate for educational and research purposes, since they are closed, i.e., they do not allow modifications over the source code or the addition of new algorithms, a feature that is crucial when testing new theories, specially new RTC strategies. Additionally, commercial tools can be expensive despite allowing integration with coding tools.

On the other hand, there are open-source tools aimed to give flexibility to the simulation algorithm, allowing the integration of several features related to UDS, like the ones stated before [3]. Some tools of this type are CITY DRAIN © [4] and SWMM [5]. CITY DRAIN © has been developed to operate with Simulink and Matlab in order to enhance its functionality with already developed control systems blocks and toolboxes for Matlab. CITY DRAIN © includes simplified flow-routing models like the Muskingum model [6]. SWMM is the storm water management model of the US Environmental Protection Agency (EPA). It is oriented to model with high level of precision, using the one-dimensional Saint Venant equations (SVE), runoff processes that take place in UDS. Additionally, SWMM can be useful for rule-based RTC modeling. However, open-source tools can be also limited, since extending and/or modifying SWMM for RTC purposes requires programming skills with a low-level programming language such as C, and adding a more complex model to the CITY DRAIN © framework (like the SVE-based model) can be difficult in some cases. Additionally, Pathirana [7] developed a Python-based toolbox to extract easily results from SWMM, attempting to satisfy this need. However, it is not possible to design RTC with it, since that tool only simplifies the extraction of simulation results from SWMM.

For this reason, in order to facilitate the design and testing processes of real-time controllers for UDS, MatSWMM, an open source tool has been created. MatSWMM is a flexible tool, i.e., a software package that gives the user the possibility to manipulate the simulation results easily for data analysis and/or system edition functionalities, since it has been structured for three high-level

programming languages (i.e., Matlab, Python, and LabVIEW), guaranteeing the possibility of implementing easily interfaces, and physical applications, taking advantage of matrix-oriented programming, plotting capabilities, optimization and control toolboxes. The toolbox works as a co-simulation engine, which is based on SWMM and it has been developed as a collection of functions in order to facilitate the expansion of the framework.

An important but often missed issue is that both Matlab and LabVIEW environments are commercial and closed products, thus their kernel and libraries cannot be neither modified nor freely distributed. To allow exchanging ideas effectively improving scientific research, both the toolbox and the platform on which MatSWMM runs should be free [8]. At this aim, MatSWMM can run on Python, which is open source and has a great variety of optimization, data analysis, control and numerical analysis libraries, e.g., NumPy, SciPy and python-control. It is also possible to work with ArcPy, which is a GIS library that can be used to enhance the functionality of MatSWMM in the future, in order to offer the same geographical positioning capabilities of commercial packages such as InfoWorks CS or MIKE.

In this manuscript, a detailed description of the toolbox functionalities is presented and some applications for it are suggested, emphasizing on the RTC design and modeling applications. The remainder of the manuscript is organized as follows. Section 2 introduces some preliminaries related to the simulation algorithm and the MatSWMM environment. Section 3 gives a detailed description of all the functionalities of MatSWMM, which is complemented in Section 4 with the description of the population-dynamics-based controller for UDS that is included into the toolbox. Section 5 gives a brief description of the models that are adapted to MatSWMM in order to implement model-based controllers. Section 6 presents an application example and some realizable applications with the toolbox. Finally, in Section 7, some conclusions are drawn and a discussion is made about further applications of the co-simulation by using this novel tool.

## 2. Preliminaries

In this section, important aspects for hydraulics computation and RTC with MatSWMM are presented. First, a brief explanation of the SWMM model is given, highlighting the modifications that are done to the SWMM flow-routing algorithm, and how several enhanced functionalities have been adapted to it. Then, it is emphasized how RTC can be designed with MatSWMM, and how it can be scalable for large-scale systems. It is important to consider that MatSWMM can be only executed through command-line instructions when using Matlab and Python. In contrast, if LabVIEW is preferred, functions are called like typical LabVIEW blocks with their corresponding inputs and outputs.

### *2.1. The SWMM Model*

SWMM is a dynamic rainfall-runoff simulation model that is used for planning, analysis, and design of infrastructure related to stormwater runoff, com-

bined and sanitary sewers, and other drainage systems in urban areas [5]. The platform consists of an interface where a drainage network can be created by using elements such as pipes, canals, storage units, sub-catchments, among others. Additionally, it has a calculation module that uses the one-dimensional SVE to simulate runoff throughout the network [6].

SWMM has been also equipped with dynamic and static control elements. Static elements such as weirs and outlets, whose purpose is to limit water levels along the system, work as barriers that reduce the energy of the flow. Dynamic control elements, such as orifices and gates, can be programmed to operate with either logical or rule-based control. Additionally, local PID controllers can be implemented to set operation points for valves and gates settings in order to manage water levels, cumulative volumes, and flow directions.

The RTC default functionalities of SWMM have become limited for large-scale control of UDS, as the ones developed in [9] [10], and [11]. However, it has a big potential as an open-source software since it is possible to enhance its functionality in order to include new methods that allow users to simulate complex RTC strategies, and to edit systematically any UDS implemented on it.

### *2.2. Structure of the MatSWMM Toolbox*

A special structure of folders and work-files, presented in Figure 1, has been designed for MatSWMM in order to maintain an organized environment while working with the toolbox. The structure of MatSWMM can be categorized in three main parts: the SWMM files, the MatSWMM files, and the simulation results. The SWMM files (i.e., the input, report, and output files), are stored in a single folder called "`swmm_files`". In order to run a simulation, it is only required to store the input file created with SWMM, and the path to the input file must be described through code.

The simulation results are stored in four different folders that are related to the simulation time and the three main types of elements in UDS (i.e., links, nodes, and subcatchments). The results of the simulation are stored in "`.csv`" files that contain information of different attributes depending on the type of object as exposed in Figure 1.

Moreover, the MatSWMM files that are required to compute a simulation vary depending on the programming language. Since Matlab and Python can be used for object-oriented programming (OOP), the MatSWMM environment has been conceived as a class file with several methods. In contrast, MatSWMM for LabVIEW is composed by a set of "`.vi`" files related to the functionalities of the toolbox. There are basic functions and enhanced functions that can be used to reproduce the simulation algorithm (these are highlighted in Figure 1), extract information, and modify the state of actuators from the simulation. It is important to notice that a dynamic link library (DLL) with a modified version of SWMM has to be saved at the workspace regardless the programming language used.

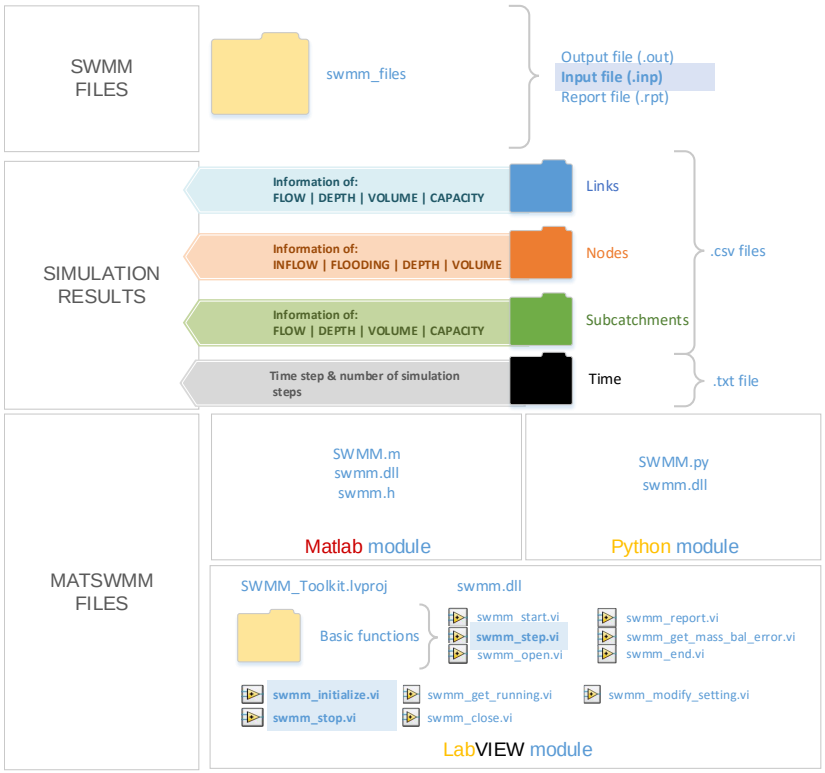


Figure 1: Corresponding folders and files of the MatSWMM toolbox. Basic functionalities inherited from SWMM are highlighted in blue.

### *2.3. Computational Aspects for Hydraulics*

It is emphasized that SWMM is a physically-based discrete-time simulation model that uses principles of conservation of mass, energy and momentum [5]. As a discrete-time tool, its algorithm finds iteratively solutions for the flow-routing problem described by the SVE, using a fifth-order accurate Runge-Kutta method (RK) with adaptative step size control [12]. This RK algorithm is based on the embedded Runge-Kutta formulas, originally formulated by Fehlberg [13].

Hydraulics in SWMM are computed through several models of subsystems related to physical processes taking place in the UDS, such as, surface runoff, groundwater, flow routing, water quality routing, infiltration, snowmelt, and surface ponding [5]. Since the purpose of the toolbox is to facilitate the design of control strategies for UDS composed of actuators (e.g., weirs, valves, gates) within conduits and storage structures, emphasis is placed on the flow routing calculation module.

MatSWMM offers the possibility to manipulate the SWMM flow routing calculation module, whose algorithm can be decomposed into three parts as

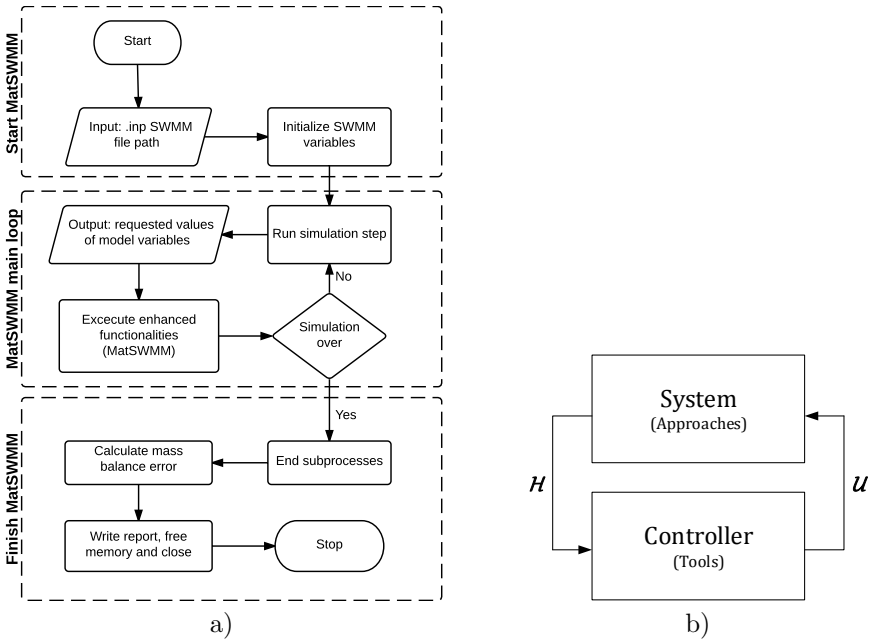


Figure 2: a) The MatSWMM algorithm composed of three main parts; and b) closed-loop scheme with co-simulation.

presented in Figure 2a). First, an initialization process is carried out, and memory is allocated to store the computation results of the simulation as well as IDs[1], and other object attributes. Secondly, the RK algorithm is invoked and a loop is created in order to calculate runoff results for a time interval; this is denominated as the main loop of the simulation. Finally, the algorithm finds solutions for the flow routing problem and two files are generated, i.e., a report file and an output file whose extensions are "`.rpt`" and "`.out`" respectively. The first file gives a summary of the main simulation results in plain text, while the second one is a binary file that stores the values of the results, in order to make the data storage and retrieving processes efficiently when large data-sets are generated.

In order to solve the flow routing problem, it is necessary to compute the lateral inflow at each node (manhole). This estimation is provided by a rainfall-runoff model that is embedded into the subcathment object of SWMM. Even though, it is also possible to provide the direct runoff information as a parameter. On the one hand, if a subcatchment model is used to compute the lateral inflow at each manhole, then SWMM uses an non-linear reservoir model based on one out of the three posible infiltration models provided by SWMM, i.e., the

[1]Identification names for the UDS components.

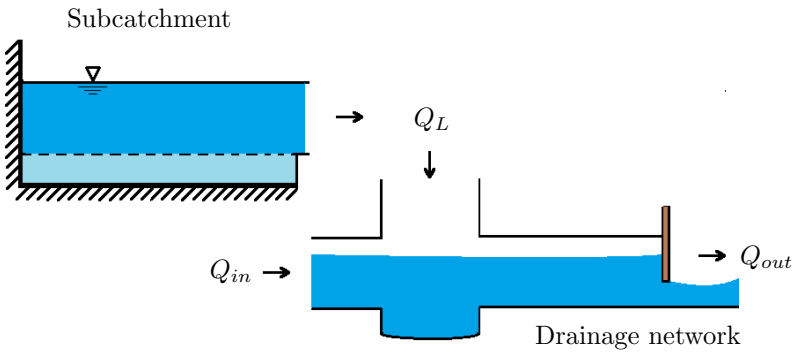


Figure 3: Flow routing scheme adopted by SWMM. The lateral inflow $Q_L$ is computed for each manhole of the network, and then, it is summed up with the inflow $Q_{in}$, in order to compute $Q_{out}$ using the Saint Venant equations (SVE).

Horton equation, the Green-Ampt method, and the curve number method. Each infiltration model has its own parameters that need to be defined by the user, as explained in [5]. Likewise, if the user decides to specify the lateral inflows at each node, it is necessary to create a time series representing the pattern of direct runoff that goes into the node [5].

Once the lateral inflow is computed, it is possible to use the well-known SVE [6] to estimate the flows throughout the drainage network (see Figure 3). The SWMM user has a choice of the level of sophistication used to solve these equations, i.e., the steady–flow routing, the kinematic–wave routing, and the dynamic–wave routing [5].

MatSWMM has been thought to be portable, therefore, the modified version of the SWMM model has been compiled as a C-language-based DLL, which can be used from any C-based programming language such as Matlab, Python, or LabVIEW. Because of this, MatSWMM can be easily extended for other programming languages using the base code[2].

MatSWMM can be considered as a co-simulation engine that retrieves information of hydraulic variables from SWMM, and allows to modify attributes in order to perform data analysis efficiently with the libraries provided for its base languages (Matlab, Python, and LabVIEW). With that fact in mind, the main loop of the algorithm presented in Figure 2 can be adapted for for RTC modeling, and the initialization process for systematic system edition, i.e., using the functionalities of the toolbox to modify attributes of the UDS model programmatically and/or automatically.

[2]The source code can be downloaded from: `https://github.com/water-systems/MatSWMM` and the documentation of the developed functionalities is available at `https://github.com/water-systems/MatSWMM/wiki`.

*2.4. Computational Aspects for Real-Time Control*

Taking into consideration the MatSWMM algorithm shown in Figure 2a), it is possible to explain how the toolbox can be used for RTC design. After the computation of each flow-routing step, the enhanced functionality is executed, i.e., it is possible to retrieve a set of state variables $\mathcal{H}$ and modify parameters during the simulation, through the set of control actions $\mathcal{U}$, as shown in Figure 2b).

It is important to notice that for RTC design, a control strategy can be implemented and tested for one of the base languages of MatSMWM under the assumption of instantaneous execution. Since the loop is stopped until the setting points are changed as desired, the setting of actuators is changed as it would be an instantaneous action, which is not what happens in reality. However, due to the flexibility of the coding scheme, if a hydraulic structure requires a large amount of time (i.e., larger than the sampling time) to move from one state to another, the delay can be modeled taking advantage of the time resolution of the simulation, since the change of setting of gates and valves in UDS tends to occur in minutes.

Another relevant issue is the time complexity of the MatSWMM algorithm, and its computational burden when large-scale systems are required to be managed. This issue is directly related to the time complexity of the proposed functionality for controlling UDSs with a population-dynamics-based approach. If the system has $m$ partitions [9], the time complexity of the control algorithm is $O(m)$, i.e., it is executed in linear time [14].

Additionally, if information of the model is requested at each time step, the computational burden of the simulation is going to be increased proportionally to the number of variables $k$ that are requested at each time step.

Thereby, the time complexity for the enhanced functionality is going to be $O(nm + nk)$, where $n$ is the number of time steps required for the simulation. With that fact in mind, the efficiency of a simulation with MatSWMM is going to be mostly dominated by the simulation time (i.e., the time window for the rain scenario and runoff modeling), and the size of the time step. This can be proved if a performance test is carried out. For this test, several simulations of a two hours rain-scenario are executed leaving the value of $n$ fixed, and modifying the values of $m$ and $k$. The performance test is developed using a computer with an Intel® Core i3 2.53GHz x64 processor, and four gigabytes of RAM memory.

It is shown in Figure 4 that both the population-dynamics-based control algorithm, and the use of *getters* at each iteration, can be solved in linear time independently, since a plane surface is formed. Additionally, the simulations with the enhanced functionality takes between three and fifty seconds, depending on the magnitudes of $m$ and $k$.

However, if either the cardinality of $\mathcal{H}$ or the number of actuators are larger than the number of simulation steps, the time efficiency is going to be dominated by the large-scale nature of the UDS network.

In order to improve the efficiency of the co-simulation algorithm, the use of *getters* (see Section 3.1) for the main loop must be limited to RTC purposes.

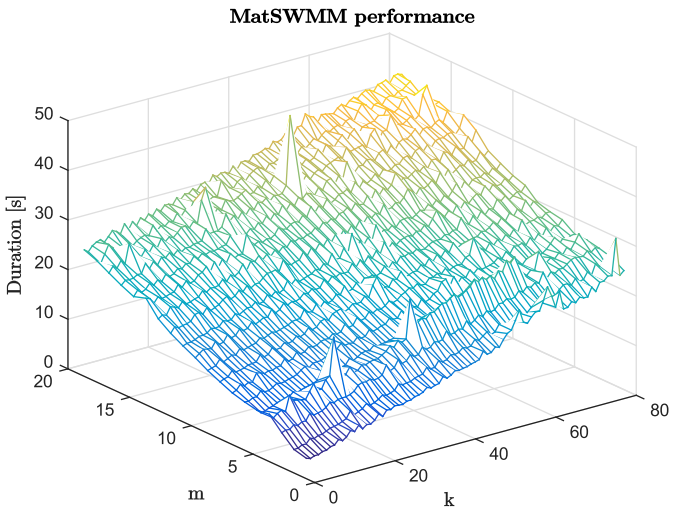


Figure 4: Performance of the proposed enhanced functionality. The duration of several simulations with a fixed number of steps is computed. At each iteration $k$ variables are extracted from the SWMM model, and $m$ partitions of the system are controlled.

If information is required for data analysis, the use of special functions after the simulation is encouraged. These functions are efficient, since they extract information from the output file without using significant memory resources, and have the capability to save the values of the main variables of all the objects in the system (i.e., nodes, links, and catchments) in CSV files. The use of special functions for data analysis is explained in Section 3.2.

## 3. MatSWMM Functionalities

In this section, a detailed explanation of the MatSWMM functionalities is given. Emphasis is placed on basic functionalities, most of them inherited from SWMM, e.g., systematic system edition, data management, and system modeling.

### *3.1. Basic functionalities*

The default functionalities of MatSWMM are related to the computation module of SWMM, which is in charge of executing the runoff model using the SVE. The basic functionalities are also related to the systematic edition capabilities of MatSWMM. The inherited functions from SWMM are: initializing SWMM, running a step, ending sub-processes, calculating the mass balance error, writing a report file, and closing the application. However, as shown in Figure 2, these functions can be grouped into three main processes that can

Table 1: MatSWMM basic functionalities.

| Function | Description |
|---|---|
| `initialize` | It opens the files required to run a SWMM simulation, and starts it. |
| `run_step` | It advances the simulation by one routing time step. |
| `is_over` | It determines whether or not the simulation is over. |
| `finish` | It saves all the results of the simulation and closes the program to run a new simulation. |
| `run_simulation` | It runs a SWMM simulation. |

be executed through the functions presented in Table 1. Using these functions the SWMM algorithm can be replicated with the advantage of having greater control of the whole process.

With that in mind, *getters* and *setters* were developed to take advantage of the partition of the SWMM algorithm (see Table 2). On the one hand, *getters* are executed at each iteration to obtain properties of the model as if they were measured by a sensor device, e.g., flow, velocity, volume, flooding, Froude number, etc. On the other hand, *setters* are used to modify properties of the model, either at each iteration or at each simulation scenario. One of the *setters* that can be useful for RTC design is the one that allows to modify the setting of actuators such as valves, gates, and orifices. It can be done using the `modify_settings` function. In addition, before the simulation is initialized, properties of the model can be modified using the `modify_input` function.

Table 2: Getters and setters included in MatSWMM.

| Function | Description |
|---|---|
| `step_get` | It retrieves the values of an specific property of multiple objects while running the simulation. |
| `get_from_input` | It returns the value of the attribute in the "`.inp`" file. |
| `get_all` | It returns all the objects IDs of a certain type (e.g., NODES, LINK, SUBCATCH, STORAGE, OUTFALL, JUNCTION) and the value of one of their properties. |
| `modify_input` | It modifies a specific attribute from the "`.inp`" file. |
| `modify_settings` | It modifies the setting of several orifices during the simulation. |

### *3.2. Data Analysis*

If the considered model is of large-scale nature, i.e., the number of nodes and pipes is such that it is not possible to retrieve state variables or modify

Table 3: Data management functionalities.

| Function | Description |
|---|---|
| `get_incidence_matrix` | It returns the graph representation of the network in matrix form. |
| `save_results` | It saves all the results of the simulation in "`.csv`" files, organized in four folders in the workspace directory. The folders are related to the type of objects (Link, Node, Subcatch). A folder called *Time* with information of the step size is also saved. |
| `create_graph` | It creates a graph data structure in Python that represents the network as a connection of nodes (manholes) and conduits (canals and pipes). |

the setting of actuators easily. Hence, *getters* and *setters* become inefficient to manage data, since they can only be executed during the main loop. Therefore, there are special methods to handle large data sets and large groups of objects (see Table 3).

For instance, it is required to extract all the results of the simulation, which can be saved in a "`.csv`" file that is easily readable by using any of the base programming languages. Additionally, special data structures were adapted to the SWMM model in order to characterize any network with canals, tanks, and actuators. Because of the structure of SWMM, graphs (modeled by the SWMMGraph class) are used to fit the model properly. Additionally, when convergence topologies prevail, the structure can be simplified as a tree (modeled by the SWMMTree class). The enhanced functionality related to these data structures is oriented to split the model into manageable pieces. Thus, the process of extracting information and modifying parameters can be done efficiently, through breath-first and deep-first search algorithms [14] (also included in MatSWMM for Python). This can be useful when implementing decentralized control, testing a fault-detection strategy [15], or simplifying the network topology, using an equivalent one, as done in [11] and [16].

### *3.3. System Modeling*

MatSWMM is useful to characterize UDS through a simplified control-oriented model (COM) [3]. Nowadays, two COMs have been included in MatSWMM, i.e., the virtual-tank (VT) based model [11], and the Muskingum model [6][17], both explained in Section 5. In order to obtain a COM with MatSWMM, it is necessary to have the real data of input and output flows of the structure that is going to be characterized (e.g., canals or pipes). It is possible to use the SWMM model included in MatSWMM to obtain accurate data that represents the behavior of the system, as it was a virtual reality (i.e., the physical behavior of the system). Furthermore, a simulation can be run and data from the model can be easily extracted with MatSWMM. Those data sets can be used to calibrate a COM and then, it can be used for the design of model-based controllers.

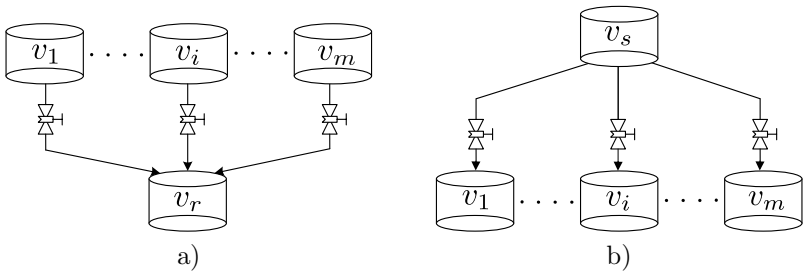


Figure 5: a) Convergence topology, i.e., $m$ source reservoirs whose outflows converge to a receptor reservoir; and b) divergence topology, i.e., a source reservoir ($v_s$) whose outflow diverges to $m$ receptor reservoirs ($v_r$).

## 4. Population-dynamics-based controllers

A detailed description of the distributed controllers based on population dynamics for MatSWMM is presented in this section. In order to introduce this concept, first a mathematical framework of the population dynamics is presented, and then special emphasis on the distributed replicator dynamics, the Smith dynamics, and the projection dynamics is made. Furthermore, a proof of concept example by using the mentioned control module is shown. Finally, the results exhibit the performance of the distributed real-time population-dynamics-based controllers by using the developed toolbox funtionalities.

*Notation*

Column vectors are denoted by lower-case letters and bold style, e.g., $\mathbf{p}$. Scalars are denoted by non-bold style, e.g., $n$. Sets are denoted by calligraphic style, e.g., $\mathcal{S}$. The column vector of $m$ unitary entries is denoted by $\mathbb{1}_m \in \mathbb{R}^m$, i.e., $\mathbb{1}_m = [1 \quad \dots \quad 1]^\top$. The cardinality of a set $\mathcal{X}$ is denoted by $|\mathcal{X}|$. The set of real numbers is denoted by $\mathbb{R}$, the set of non-negative real numbers is denoted by $\mathbb{R}_+$, and the set of strictly positive real numbers is denoted by $\mathbb{R}_{++}$. Similarly, the set of positive integer numbers is denoted by $\mathbb{Z}_+$.

### *4.1. Population Dynamics*

The population dynamics approach is presented by making an analogy with either an UDS or with a drinking water network (DWN). In order to make the analogy, it is also taken into account that the UDS is mainly associated to the convergence topology shown in Figure 5a) as in [9], while the DWN is mainly associated to the divergence topology shown in Figure 5b) as in [18].

First of all, the analogy between elements in the population dynamics approach with elements in either USD or DWN is presented in Table 4.

For instance, consider the system with the convergence topology shown in Figure 5a) with $m \in \mathbb{Z}_+$ source reservoirs (strategies). The total flow through the system (population mass) is denoted by $Q(t) \in \mathbb{R}_+$, which corresponds to

Table 4: Equivalence between population dynamics and UDS/DWN.

| **Population dynamics** | **UDS** | **DWN** |
|---|---|---|
| Population | System | System |
| Strategy | Source reservoirs | Receptor reservoirs |
| Population mass | Total inflow to receptor reservoir | Total outflow source reservoir |
| Agent | Flow unit | Flow unit |
| Proportion of agents | Proportion of flow | Proportion of flow |
| Strategic distribution | Flow distribution in source reservoirs | Flow distribution in receptor reservoirs |
| Fitness of a strategy | Current volume | Available volume capacity |

the inflow of the receptor reservoir in the convergence topology. Hence, $Q(t)$ is composed of a large and finite amount of flow units (agents). Each flow-unit is assigned to an outflow of the source reservoirs, where the set of source reservoirs is denoted by $\mathcal{S} = \{1, ..., m\}$.

On the other hand, regarding the divergence topology shown in Figure 5b), there are $m \in \mathbb{Z}_+$ receptor reservoirs (strategies). Similarly as in the convergence topology case, $Q(t) \in \mathbb{R}_+$ corresponds to the outflow of the source reservoir in the divergence topology. Each flow-unit is assigned to an inflow of the receptor reservoirs, and then the set of receptor reservoirs is denoted by $\mathcal{S} = \{1, ..., m\}$.

In both the convergence and divergence topologies, the scalar $p_i(t)$ is the proportion of flow units assigned to each flow associated to the reservoir $i \in \mathcal{S}$ as a percentage, i.e., the outflow/inflow for the $i^{\text{th}}$ reservoir in the convergence/divergence case is given by $p_i(t)Q(t)$. The vector $\mathbf{p}(t) \in \mathbb{R}_+^m$ is the flow proportion distribution involving the $m$ reservoirs according to the topology. The set of the possible distributions of flow is given by a simplex

$$\Delta = \left\{ \mathbf{p}(t) \in \mathbb{R}_+^m : \mathbf{p}(t)^\top \mathbb{1}_m = 1 \right\},$$

and the interior of the set of the possible distributions of flow is given by the set

$$\text{int}\Delta = \left\{ \mathbf{p}(t) \in \mathbb{R}_{++}^m : \mathbf{p}(t)^\top \mathbb{1}_m = 1 \right\}.$$

Finally, the tangent space of the set of possible distributions of flow is defined as

$$\text{T}\Delta = \left\{ \mathbf{z}(t) \in \mathbb{R}^m : \mathbf{z}(t)^\top \mathbb{1}_m = 0 \right\}.$$

For the convergence topology, each flow unit is assigned to each reservoir $i \in \mathcal{S}$ depending on the current volume, which is described by a function denoted by $F_i(\mathbf{p}(t))$. Then, more outflow is assigned to those reservoirs close to be filled up impeding overflows in them. In contrast, for the divergence topology, each flow unit is assigned to each reservoir $i \in \mathcal{S}$ depending on the current volume capacity, which is described by a function $F_i(\mathbf{p}(t))$. Therefore, less inflow is assigned to those reservoirs close to be filled up.

The design of the population-dynamics-based controllers are given by the proper selection of the fitness functions that define the incentives for the proportion of agents to choose a particular strategy. The proper selection of the fitness functions is further discussed bellow and it depends on the type of topology, i.e., an appropriate selection of the fitness function depends on if the system has a convergence or divergence topology. Furthermore, it is necessary that the fitness functions satisfy conditions to obtain a class of population game known as stable game [19].

**Definition 1.** *The game* $\mathbf{F}(\mathbf{p})$ *is stable if the Jacobian matrix* $\mathbf{J} = D\mathbf{F}(\mathbf{p})$ *is negative semi-definite with respect to the tangent space* $T\Delta$ *[19], i.e.,*

$$\mathbf{z}^\top \mathbf{J}\, \mathbf{z} \leq 0, \;\; \textit{for all } \mathbf{z} \in T\Delta,\ \mathbf{p} \in \Delta.$$

*Then, it implies that a game is stable if the fitness functions are decreasing with respect to the proportion of agents.* ♢

Notice that for the convergence topology (see Figure 5a)), the fitness functions can be selected increasing with respect to the current volumes as in [9] (see Figure 6a)). For the convergence topology, when a proportion of agents is increased, it is expected that the corresponding volume decreases (see Figure 6b)). Consequently, due to the fact that the fitness function is increasing with respect to the volume, the fitness function decreases with respect to the proportion of agents (necessary condition for a stable game). Contrary, for the divergence topology counterpart (see Figure 5b)), the fitness functions can be selected decreasing with respect to the current volume, e.g., the error with respect to the maximum capacity volume as in [18] (see Figure 6c)). When a proportion of agents is increased it is expected that the corresponding volume increases (see Figure 6d)). Consequently, due to the fact that fitness functions are increasing with respect to the volume, the fitness function decreases with respect to the proportion of agents (necessary condition for a stable game).

### *4.2. Particular distributed population dynamics*

The traditional replicator dynamics, Smith dynamics, and projection dynamics equations are part of the six fundamental population dynamics [20], which require full information (i.e., all strategies associated to reservoirs need information about all other reservoir states in order to evolve). However, the distributed population dynamics are deduced in [21] from a local revision protocol that only needs partial information. Due to the fact that only local information is needed, then there is an undirected non-complete connected graph describing possible interaction denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes, which represents the reservoirs, and $\mathcal{E} \subset \{(i,j) : i,j \in \mathcal{V}\}$ is the set of links representing the information sharing within the system. Furthermore, the set of neighbors of the node $i \in \mathcal{V}$ is given by $\mathcal{N}_i = \{j : (i,j) \in \mathcal{E}\}$. Notice that $i \notin \mathcal{N}_i$, and that $\mathcal{N}_i \neq \emptyset$, for all $i \in \mathcal{V}$ since $\mathcal{G}$ is connected.

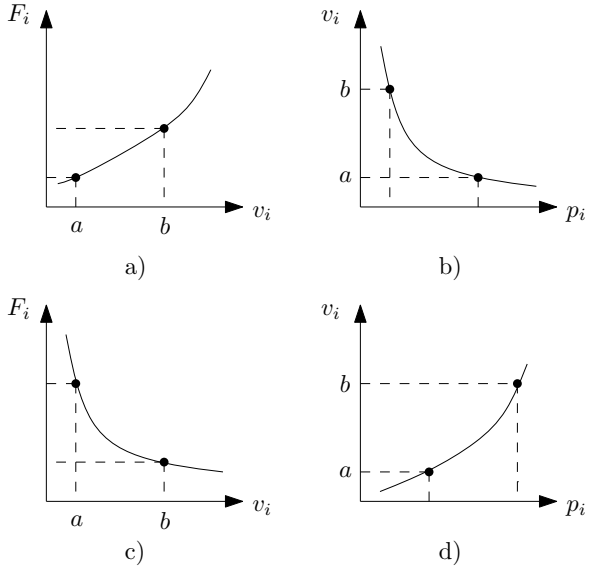


Figure 6: Proper selection of fitness functions for convergence topology a) and b), and divergence topology c) and d). Correspondance is as follows: a) increasing fitness function with respect to volume; b) decreasing relation existing between proportion of agents and volume for convergence topology; c) decreasing fitness function with respect to volume; and d) increasing relation existing between proportion of agents and volume for divergence topology.

The distributed replicator dynamics are given by

$$\frac{dp_i(t)}{dt} = p_i(t)\left(F_i(\mathbf{p}(t))\sum_{j\in\mathcal{N}_i} p_j(t) - \sum_{j\in\mathcal{N}_i} p_j(t)F_j(\mathbf{p}(t))\right), \quad \text{for all } i \in \mathcal{S}.$$

The distributed Smith dynamics are given by

$$\frac{dp_i(t)}{dt} = \sum_{j\in\mathcal{N}_i} p_j(t)\left[F_i(\mathbf{p}(t)) - F_j(\mathbf{p}(t))\right]_+ - p_i(t)\sum_{j\in\mathcal{N}_i}\left[F_j(\mathbf{p}(t)) - F_i(\mathbf{p}(t))\right]_+,$$

for all $i \in \mathcal{S}$, and where $[\cdot]_+ = \max(\cdot, 0)$. And finally, the distributed projection dynamics are given by

$$\frac{dp_i(t)}{dt} = |\mathcal{N}_i|F_i(\mathbf{p}(t)) - \sum_{j\in\mathcal{N}_i} F_j(\mathbf{p}(t)), \quad \text{for all } i \in \mathcal{S}.$$

MatSWMM includes a function that can be used to implement a population-dynamics-based controller for UDS with divergence and convergence topologies (i.e., the `pdyncontrol` function). It includes the Smith, the projection, and the replicator dynamics.

## 5. Models of UDS

It is possible to characterize any UDS as a composition of two topologies, i.e., convergence and divergence. As shown in Figure 5a), the convergence topology is related to the case where the flow of several storage units or pipes is merged in a single receptor structure. Furthermore, the divergence topology, shown in Figure 5b), is formed when the flow of a single structure is distributed along more than one receptor. The former one prevails in stormwater UDS while the latter is commonly related to drinking water networks (DWN). However, both topologies are used to build combined sewer networks where not only stormwater is transported but also sanitary and wastewater flows.

MatSWMM has been designed to characterize both topologies easily, guaranteeing a proper modeling of the run-off phenomenon throughout large networks and allowing the user to handle the problem of flow assignation with an optimization-based controller for each case, as the one described in [9].

### *5.1. Control-Oriented Models*

The SVE used by SWMM to simulate the run-off throughout the network describe in a quite high level of detail the behavior of the system. Usually, this level of detail is not required in RTC applications and COMs are used instead. As stated before, MatSWMM incorporates the so-called VT-based model, which is a widely used COM for modeling UDS, and the Muskingum model, also used for predictive control [22][23].

#### *5.1.1. Virtual Tanks Model*

In the virtual reservoir approach, the UDS is divided into a set of interconnected real and VTs. According to [11], a VT is a storage element that represents the total volume of sewage inside the sewer mains associated with a determined portion of a given network. The volume is computed through the mass balance of the stored volume, the inflows (from both sewage mains and stormwater), and the outflows (to both sewage mains and street) of the reservoir. For the model developed in MatSWMM, the outflow of a given tank (virtual or real) is assumed to be proportional to the volume of the tank. Therefore, the model of a tank is given by $\frac{dV_i(t)}{dt} = q_i^{in}(t) - K_i V_i(t)$, where $V_i$ is the volume stored in the $i$-th tank, $q_i^{in}$ is the total inflow to the $i$-th tank, and $K_i$ is the *volume/flow conversion* (VFC) coefficient. As SWMM provides measures for both, the volume and the outflow of the reservoirs, the VFC coefficient for each reservoir can be computed via a least-squares algorithm given by

$$K_i = \arg\min_{K_i} \int_0^{t_f} \left( q_i^{out}(t) - K_i V_i(t) \right)^2 dt, \tag{1}$$

where $t_f$ is the total simulation time, and $q_i^{out}$ is the total outflow of the $i$-th reservoir. Finally, the maximum capacity volume of the $i$-th tank is denoted by $V_i^{max}$.

In order to solve the least–squares algorithm to calibrate the linear reservoir model using MatSWMM, a discretized variant of (1) can be proposed as follows:

$$K_i = \arg\min_{K_i} \|\mathbf{q}_i^{out} - K_i \mathbf{v}_i\|^2. \tag{2}$$

In this way, the parameter $K_i$ for the $i^{\text{th}}$ reservoir can be computed regardless of either the form of the outflow or the volume.

*5.1.2. Muskingum Model*

The Muskingum model describes the water flow through UDSs based on the conservation mass principle. The mass balance for the $i^{th}$ reservoir in the UDS is given by

$$\frac{dV_i(t)}{dt} = q_i^{in}(t) - q_i^{out}(t),$$

and the relation between its inflows and outflows [6] is given by

$$V_i(t) = Aq_i^{in}(t) + Bq_i^{out}(t),$$

where $A$ and $B$ are parameters for the model calibration. Similarly to the case of the linear reservoir, the estimation of parameters $A$ and $B$ for the discretized Muskingum model is done by solving (2). This problem has been solved analytically in [24], then, both $A$ and $B$ can be computed in terms of the inflow, outflow, and stored volume by using the following expressions:

$$A_i = \frac{(\mathbf{q}_i^{out})^{\mathrm{T}}\mathbf{q}_i^{out}\mathbf{v}_i^{\mathrm{T}}\mathbf{q}_i^{in} - (\mathbf{q}_i^{out})^{\mathrm{T}}\mathbf{q}_i^{in}\mathbf{v}_i^{\mathrm{T}}\mathbf{q}_i^{out}}{(\mathbf{q}_i^{\mathrm{in}})^{\mathrm{T}}\mathbf{q}_i^{in}(\mathbf{q}_i^{out})^{\mathrm{T}}\mathbf{q}_i^{out} - ((\mathbf{q}_i^{out})^{\mathrm{T}}\mathbf{q}_i^{in})^2}, \tag{3}$$

$$B_i = \frac{(\mathbf{q}_i^{in})^{\mathrm{T}}\mathbf{q}_i^{in}\mathbf{v}_i^{\mathrm{T}}\mathbf{q}_i^{out} - (\mathbf{q}_i^{out})^{\mathrm{T}}\mathbf{q}_i^{in}\mathbf{v}_i^{\mathrm{T}}\mathbf{q}_i^{in}}{(\mathbf{q}_i^{\mathrm{in}})^{\mathrm{T}}\mathbf{q}_i^{in}(\mathbf{q}_i^{out})^{\mathrm{T}}\mathbf{q}_i^{out} - ((\mathbf{q}_i^{out})^{\mathrm{T}}\mathbf{q}_i^{in})^2}. \tag{4}$$

By expressing the outflows as function of $V_i(t)$, $q_i^{in}(t)$, $A$, and $B$, it is obtained that $dV_i(t)/dt = q_i^{in}(t)\,(1 + A/B) - V_i(t)/B$, then

$$\frac{dV_i(t)}{dt} = \gamma q_i^{in}(t) - K_i v_i(t),$$

where $\gamma = (1 + A/B)$, and $K_i = 1/B$. Furthermore, $K_i > 0$ scales the outflow, and it can be seen as the discharge coefficient of the reservoir. However, for some particular cases $K_i < 0$, and the considered system output can show a non-minimum phase behavior [17].

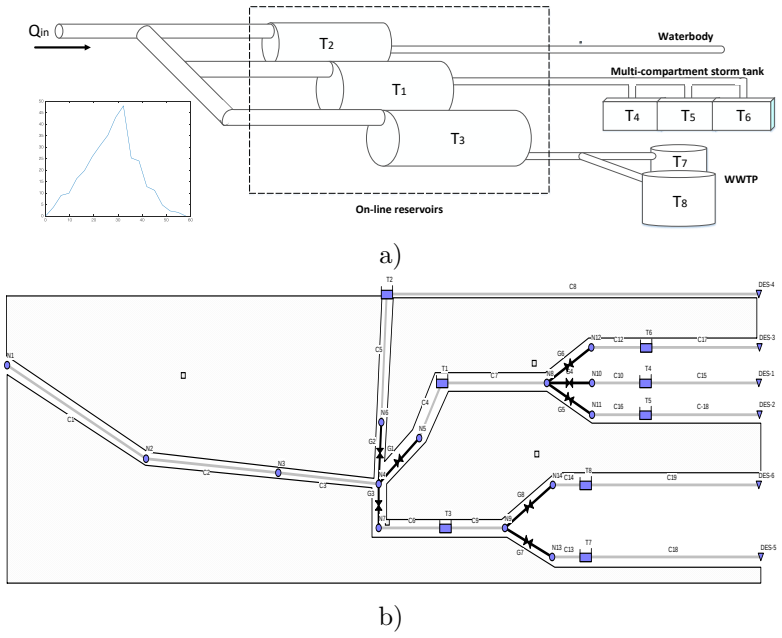


Figure 7: Divergence case study a) conceptual scheme and input direct runoff hydrograph; and b) system implementation in SWMM.

### *5.2. Simulation-Oriented Models*

MatSWMM includes the SWMM model as its simulation-oriented model [3]. SWMM is known as an accurate physically-based model that describes the propagation of a wave in an open channel through the SVE. It is based on the conservation of mass and momentum principles [6]. One of the best features of SWMM consists on representing the UDS as a composition of nodes and links, which facilitates the modeling in discrete time of the physical prototype and the mathematical solution of the non-uniform flow SVE.

The main variable in the links is the flow rate, and the solution to the flow-routing problem is for the average flow in each link, which is assumed to be constant over a time step. For each step of the simulation, the velocity of the flow, and the depth of each link is calculated. In the case of the nodes, the main variable is the pressure head, which is assumed to be changing in time but constant over a time step [25].

The continuity and the momentum SVE are reported in [6]. The numerical integration of the two SVE is accomplished with a RK algorithm [12]. It has been shown in [25] and [5] that the SWMM model and its algorithm is stable and accurate through the calculation of three types of mass-balance errors, i.e., runoff, flow-routing, and water quality errors. This fact guarantees that the SWMM model is appropriate to model UDS with a high level of precision, i.e., better than the one obtained with the COMs.

## 6. Examples of application

This section illustrates some features of MatSWMM for RTC by means of a divergent UDS case study (see Figure 7). The system is composed by four subcatchments that are affected by a precipitation event, the precipitation becomes runoff, and then, it is transported through a system of pipes that is divided into three branches. After the bifurcation, typical on-line storage units, like the ones described [26], are used to handle and retain the rise of flow. The storage units have different dimensions and are located at different elevations. These differences are common in UDS because of the unavailability of land for infrastructure [26]. Finally, as it is shown in Figure 7, the first division carries water to a tank with three different bodies, the second division meets an outfall and the third division is splitted in two branches, carrying water to a waste water treatment plant (WWTP) with two storage units.

The control objective in this model is to spread efficiently flows at each bifurcation, handling the differences between reservoirs, i.e., differences between their location, elevation, and dimensions. To this end, a partition of the system is done as proposed in [9] and the distributed control based on population dynamics included into MatSWMM is used.

Because of the nature of the system, the fitness function that will be used by the controller to solve the optimal assignment problem is related to the occupancy rate or the capacity of each tank, which is equivalent to the normalized volume, as presented in Table 4.

An example of the code that can be used in Matlab with MatSWMM to implement the control strategy in the first divergence zone is outlined below:

```
>> input_file = 'example.inp'; settings = [1/3 1/3 1/3];
>> tanks = {'T1', 'T2', 'T3'}; max_depths = [5 4 4.5];
>> gates = {'G1', 'G2', 'G3'};
>> SWMM.initialize(inp); % MatSWMM initialization
>> while (~SWMM.is_over) % MatSWMM main loop
     SWMM.run_step;
     normalized_volumes = SWMM.step_get(tanks, SWMM.VOLUME, SWMM.SI) ...
     ./ max_depths;
     settings = SWMM.pdyncontrol('smith', 'divergence', ...
     normalized_volumes, 0.02);
     SWMM.modify_settings(gates, settings/max(settings));
   end
>> [errors, duration] = SWMM.finish; % MatSWMM closing
>> [time, volumes] = SWMM.read_results(tanks, SWMM.NODE, SWMM.VOLUME);
```

First of all, some variables are declared, such as the SWMM input file path, initial conditions for the setting of gates, and the IDs of tanks and gates. Then, the three parts of the MatSWMM algorithm are invoked, i.e., the initialization, the main-loop, and the closing of the program. Within the main-loop the population dynamics-based controller is used to manage the flows of one of the partitions of the system, which is composed of three on-line reservoirs, i.e., $T_1$, $T_2$, and $T_3$. Finally, information of the volumes at each reservoir is extracted for data analysis.

The same procedure is applicable to implement RTC on the remaining divisions. However, the controller should be tuned depending on the speed of the dynamics, and the type of dynamics (i.e., Smith, projection, or replicator). The results of a scenario where no-control strategy is implemented are shown in Figure 8. The distribution of water volumes is not efficient, since reservoirs $T_2$, $T_3$, and $T_4$ are overused, i.e., they are operating at its maximum capacity, while the others are underused, i.e., their used capacity is less than 100%. Additionally, $6.99 \times 10^3 \mathrm{m}^3$ of flooding appear, due to the excess of water that flows out from the overused tanks. This problem may occur when designing hydraulic infrastructure based on a static design storm, i.e., the climate variations due to the heat island effect were not considered [27].

When the distributed population-dynamics-based controller is implemented at each of the nodes where the flow diverges, flooding is prevented and the water is better distributed. The results when RTC is implemented are obtained for the projection, the replicator, and the Smith dynamics (see Figure 9).

The simulation with MatSWMM allows to indentify the main advantage of implementing RTC, which is to distribute efficiently flows in divergent and convergent sections of the UDS to minimize the total flooding throughout the network. Throughout this approach, it is possible to handle dynamic-rain scenarios, different to the static ones used for design, with the already built infrastructure.

Additionally, tunning the controller can be easily done for a physical implementation (even empirically), because of the advantage of programming the SWMM model, which is accurate [25], through MatSWMM, in order to obtain the results of several simulations systematically.

*Other applications*

Because of the capability that MatSWMM brings to its users for editing systematically any UDS implemented with the SWMM framework, it is possible to extend the scope of MatSWMM to optimal design of UDS, since most of the optimization problems proposed for UDS can be solved with heuristics, but require values of variables from an efficient simulation-oriented model (e.g., total flooding, volumes, etc.) [28][29][30].

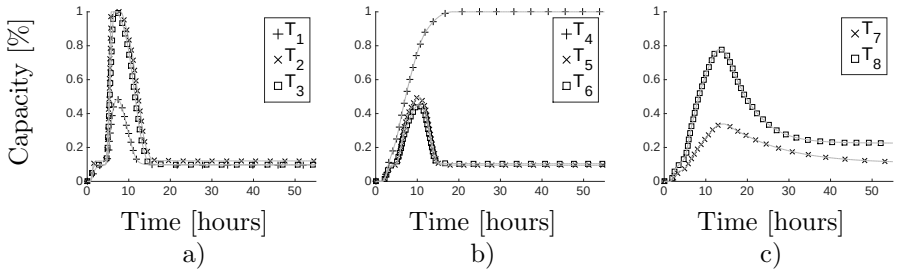


Figure 8: Occupancy rate, or capacity of storage tanks when no control strategy is applied, a) Capacity of reservoirs $T_1$, $T_2$, and $T_3$; b) capacity of reservoirs $T_3$, $T_4$, and $T_5$; c) capacity of reservoirs $T_7$, and $T_8$.

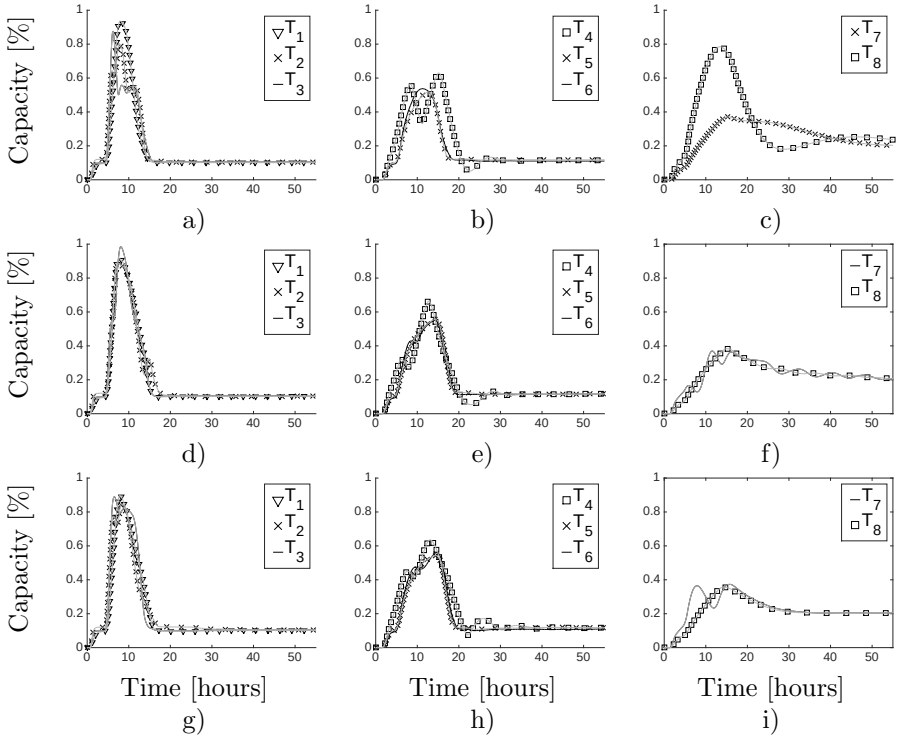


Figure 9: (a, b, c) capacity of tanks when the RTC strategy with the projection dynamics is applied; (d, e, f) capacity of tanks when the RTC strategy with the replicator dynamics is applied; and (g, h, i) capacity of tanks when the RTC strategy with the Smith dynamics is applied.

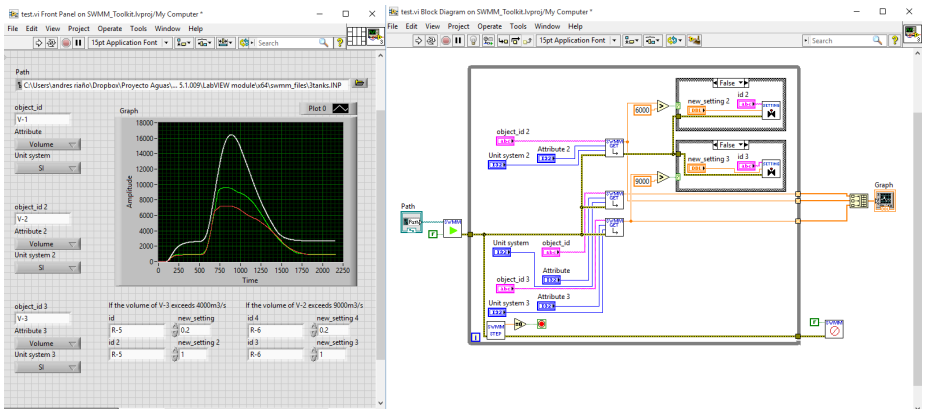

Figure 10: LabVIEW interface based on MatSWMM.

Also, interfaces such as the one presented in Figure 10, can be implemented easily with the MatSWMM basic functionalities for LabVIEW, to characterize a physical UDS. Additionally, the ArcPy library for Python can be used in parallel with the MatSWMM library in order to extract GIS data and solve the problem of optimal positioning of UDS structures, specifically, on-line and off-line storm tanks [26].

With that in mind, the capabilities of MatSWMM can be useful for optimal design of infrastructures, considering not only the typical rain-scenario and a critical service operation condition, but also taking into consideration the possibility of including sensors and actuators before the commissioning of the system, and to analyze how these elements an their proper management can significantly improve the performance of UDS.

## 7. Conclusions and Further Work

This manuscript has presented a new open-source toolbox for the design of RTC in UDS, i.e., MatSWMM, which runs on Matlab, LabVIEW, and Python. MatSWMM includes a variety of functionalities to edit systematically any UDS implemented on the SWMM framework, during and after a rainfall simulation. These features make MatSWMM suited for both educational and research purposes, specially for the design of RTC applied to UDS. xAdditionally, it is possible to test heuristic algorithms for optimal design and positioning of retention structures in UDS, since the only requirements for these tests are the results from the simulation, and a function to edit the properties of the system elements. As a matter of fact, MatSWMM is currently used by several masters, and Ph.D. students, and has an active Github repository[3]. Among future projects, there are supporting Linux-based OSs, adding parallel computing techniques for optimal design of UDS, and extending the toolbox including more control strategies, such as optimization-based controllers (e.g., MPC, LQR, evolutionary-based controllers), and heuristic algorithms (e.g., rule-based, fuzzy control). The toolbox is going to be enhanced for realizable applications such as optimal positioning of UDS elements (e.g., sensors, actuators, retention structures), based geographical information provided by new GIS capabilities. Also, as more features are developed, more case studies are going to be included into the repository. Any suggestion and/or bug report are very welcome.

## Acknowledgments

Authors would like to thank Mexichem (Colombia) for supporting this research through the project *Drenaje Urbano y Cambio Climático: hacia los sistemas de alcantarillado del futuro.* Fase II. COLCIENCIAS 633/2013. Also,

[3]The link to the repository is: `https://github.com/water-systems/MatSWMM.git`.

this work has been partially supported by the project ECOCIS (Ref. DPI2013-48243-C2-1-R). J. Barreiro-Gomez is partially supported by COLCIENCIAS-COLFUTURO and Agencia de Gestio d'Ajust Universitaris i de Recerca AGAUR.

## References

[1] R. K. Price, Z. Vojinović, Urban hydroinformatics: data, models, and decision support for integrated urban water management, IWA publishing, 2011.

[2] D. Butler, M. Schütze, Integrating simulation models with a view to optimal control of urban wastewater systems, Environmental Modelling & Software 20 (4) (2005) 415–426.

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

[4] S. Achleitner, M. Möderl, W. Rauch, City drain©–an open source approach for simulation of integrated urban drainage systems, Environmental Modelling & Software 22 (8) (2007) 1184–1195.

[5] W. C. Huber, L. A. Rossman, R. Dickinson, EPA storm water management model SWMM 5.0, National Risk Management Research Laboratory, Office of Research and Development, US Environmental Protection Agency, 2010.

[6] V. Te Chow, Open channel hydraulics, McGraw-Hill Book Company, Inc; New York, 1959.

[7] A. Pathirana, SWMM5 1.1.0.3 : Python package (2015).
URL `https://pypi.python.org/pypi/SWMM5`

[8] R. Stallman, Free software, free society: Selected essays of Richard M. Stallman, Lulu, 2002.

[9] J. Barreiro-Gomez, G. Obando, G. Riaño Briceño, N. Quijano, C. Ocampo Martinez, Decentralized control for urban drainage systems via population dynamics: Bogotá case study, in: Proceedings of the 14th European Control Conference, Linz, Austria, 2015, pp. 2431–2436.

[10] A. Martin, Mathematical optimization of water networks, Springer Basel AG, 2012.

[11] C. Ocampo-Martinez, Model Predictive Control of Wastewater Systems, Springer Verlag, 2010.

[12] S. Eslamian, Handbook of Engineering Hydrology: Fundamentals and Applications, CRC Press, 2014.

[13] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical recipes in C, Vol. 2, Citeseer, 1996.

[14] T. H. Cormen, L. E. Leiserson, R. Rivest, C. Stein, Introduction to algorithms, MIT press, 2009.

[15] S. Tornil-Sin, C. Ocampo-Martinez, V. Puig, T. Escobet, Robust fault detection of non-linear systems using set-membership state estimation based on constraint satisfaction, Engineering Applications of Artificial Intelligence 25 (1) (2012) 1–10.

[16] G. Riaño Briceño, A. Ramirez-Jaime, J. Barreiro-Gomez, N. Quijano, C. Ocampo Martinez, Co-simulation for the design of controllers in urban drainage systems, in: Proceedings of the 2nd Colombian Conference on Automatic Control, Manizales, Colombia, 2015, pp. 1–6.

[17] Y. Bolea, V. Puig, J. Blesa, On the use of the Muskingum model in real-time sewer network control, in: Proceedings of the 6th European Control Conference, Kos, Greece, 2007, pp. 3717–3723.

[18] E. Ramirez-Llanos, N. Quijano, A population dynamics approach for the water distribution problem, International Journal of Control 83 (9) (2010) 1947–1964.

[19] J. Hofbauer, W. H. Sandholm, Stable games and their dynamics, Journal of Economic Theory 144 (4) (2009) 1665–1693.

[20] W. H. Sandholm, Population games and evolutionary dynamics, Economic learning and social evolution, Cambridge, Mass. MIT Press, 2010.

[21] J. Barreiro-Gomez, G. Obando, N. Quijano, Distributed population dynamics: Optimization and control applications, IEEE Transactions on Systems, Man, and Cybernetics: Systems PP (99) (2016) 1–11.

[22] M. Gómez, J. Rodellar, J. A. Mantecón, Predictive control method for decentralized operation of irrigation canals, Applied Mathematical Modelling 26 (11) (2002) 1039–1056.

[23] J. A. Mantecón, M. Gómez, J. Rodellar, A Simulink-based scheme for simulation of irrigation canal control systems, Simulation 78 (8) (2002) 485–493.

[24] A. A. Aldama, Least-squares parameter estimation for muskingum flood routing, Journal of Hydraulic Engineering 116 (4) (1990) 580–586.

[25] G. Freni, G. B. Ferreri, P. Tomaselli, Ability of software SWMM to simulate transient sewer smooth pressurization, NOVATECH Report.

[26] J. Barro, P. Comas, P. Malgrat, D. Sunyer, Manual nacional de recomendaciones para el diseño de tanques de tormentas (2014) 32–33.

[27] P. Moonen, T. Defraeye, V. Dorer, B. Blocken, J. Carmeliet, Urban physics: Effect of the micro-climate on comfort, health and energy demand, Frontiers of Architectural Research 1 (3) (2012) 197–228.

[28] H. R. Maier, Z. Kapelan, J. Kasprzyk, J. Kollat, L. S. Matott, M. C. Cunha, G. C. Dandy, M. S. Gibbs, E. Keedwell, A. Marchi, Evolutionary algorithms and other metaheuristics in water resources: current status, research challenges and future directions, Environmental Modelling & Software 62 (2014) 271–299.

[29] J. Marques, M. Cunha, D. A. Savić, Multi-objective optimization of water distribution systems based on a real options approach, Environmental Modelling & Software 63 (2015) 1–13.

[30] A. Palumbo, L. Cimorelli, C. Covelli, L. Cozzolino, C. Mucherino, D. Pianese, Optimal design of urban drainage networks, Civil Engineering and Environmental Systems 31 (1) (2014) 79–96.