SWMM Model with 3-D Hydrodynamic Features
A Simple User's Manual

By Jiang Long, Ting Fong May Chui

Department of Civil Engineering,

The University of Hong Kong,

Oct 2021

ACKNOWLEDGEMENTS AND DISCLAIMER

This manual was prepared by Long Jiang and Ting Fong May Chui of the Department of Civil Engineering at The University of Hong Kong. The model codes have been uploaded in GitHub and are open for academic and research purposes. Note that approval does not signify that the contents necessarily reflect the views of the authors. Mention of trade names or commercial products does not constitute endorsement or recommendation for use.

Contents

1 INTRODUCTION

1.1 Overview of the model framework

This model is developed for simulating the hydrodynamic conditions for some drainage facilities with complex hydraulic structures within SWMM. The model couples OpenFOAM and SWMM, with the overall framework as shown in Fig. 1.

The first part is SWMM, where the pyswmm toolkit is utilized to import SWMM calculation libraries and read the input files. The pyswmm is a software package for the creation, manipulation, and study of the structure, dynamics, and function of drainage networks (McDonnell et al., 2020). The control algorithms can be developed exclusively in Python which allows the use of functions and objects as well as storing and tracking hydraulic changes for control actions. Since it is compatible with the GUI of EPA SWMM 5.1, users can first edit their input files by that software. In this model, some built-in functions from pyswmm are used for coupling with OpenFOAM (e.g. node.depth(), node.generate_inflow() and node.total_outflow()). The SWMM part extracts the simulation results (e.g., flow rate or water level) of specific time steps at certain nodes or receives the simulation results from the OpenFOAM part.

The second part is the OpenFOAM part, which mainly consists of existing OpenFOAM solvers for different areas (i.e., for surface water, hyporheic zone, and soil). They can couple with each other through a customized tool (described in detail below). Solvers for each of these areas could be interFoam, Darcy-Forchheimer, and groudwaterFoam, respectively. Through exchanging the conditions of boundary fields (including the velocity and pressure), these solvers are utilized to cooperatively simulate the hydrodynamics in different drainage facilities. This part also has the function of extracting simulated results to the SWMM part or reading the outputs from the SWMM part.

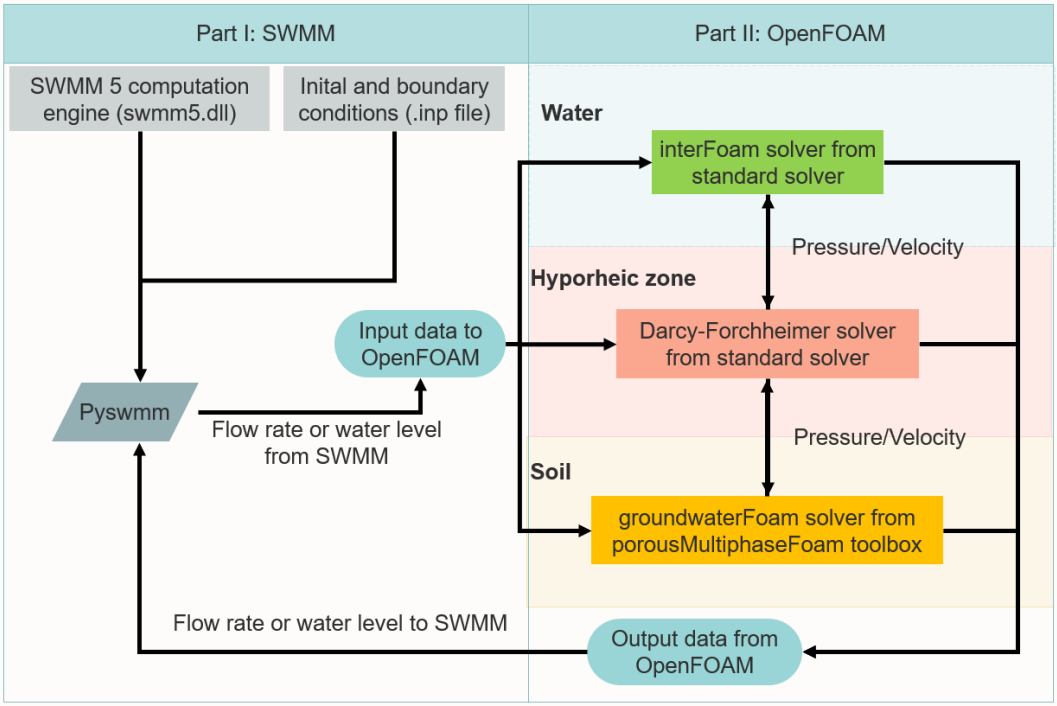


Fig. 1 Framework of the coupled model.

1.2 Installation

The model is designed to run under Linux operating system, where Ubuntu 18.04 and 20.04 are tested and recommended. It is distributed as a package published in GitHub that contains a self-installing setup script. To install it:

1. Select a place and use the below commands to download and install it
2. Download the package by command:

```
git clone https://github.com/jlonghku/SWMM_CFD
```

3. Change the permission of package:

```
sudo chmod 755 -R SWMM_CFD
```

4. Run the “install.sh” script file:

```
./SWMM_CFD/install.sh
```

Note: For Linux operating systems that are not ubuntu distributions, the installation of dependency packages needs to be modified accordingly based on the “install.sh” script and operating system. In addition, users might need to manually install OpenFOAM by the source code, details from the website for installation: https://cfd.direct/OpenFOAM/download/. Then, the python package “pyswmm” should also be installed in the same way: https://Pyswmm.readthedocs.io/en/stable/install.html.

1.3 General steps of operation

The following are the most common steps for operating the model:

1. Prepare the SWMM case input file by the GUI of EPA SWMM.
2. Prepare the OpenFOAM case by modifying the parameters on the example provided in the directory “Case” or customizing new cases accordingly.
3. Organize all SWMM and OpenFOAM cases into a folder according to the template provided.
4. Modify the parameters in the control file.
5. Run the model.
6. View or post-processing the results of the simulation.

For a quick start of a sample case, please refer to Section 2. For more settings of different control files, please refer to Section 3.

2 QUICK START TUTORIAL

There are four example cases for three types of drainage facilities (i.e., pipes, manholes, and a water channel), and overland and underground flow are provided. To run the existing cases, you just need to rename the file of input parameters to “para.py” and run the command:

```
python3 main.py
```

A simple case (incorporating pipes in the drainage system) is set up following the steps below as a quick start tutorial of case preparation.

2.1 Preparing the SWMM case

For each part of the coupled model, the case can be prepared in the same way as for the original case creation. Therefore, the setup of the SWMM part of the model can be done using the GUI of EPA SWMM 5.1. For more details of setting SWMM case procedures, please refer to the Storm Water Management Model User's Manual Version 5.1.

In the provided example cases, we select the drainage system in the existing SWMM model case provided in EPA’s manual (Gironás et al., 2009). The map of the SWMM case developed is shown in Fig. 2. The rainfall data comes from the weather station located in Hong Kong Wetland Park from 2019 to 2020. Details of other parameter settings are provided in the EPA’s manual. Then the SWMM input file should be copied to the subdirectory “SWMM” of the case.

```
cp -rf swmm.inp your_case/SWMM/
```

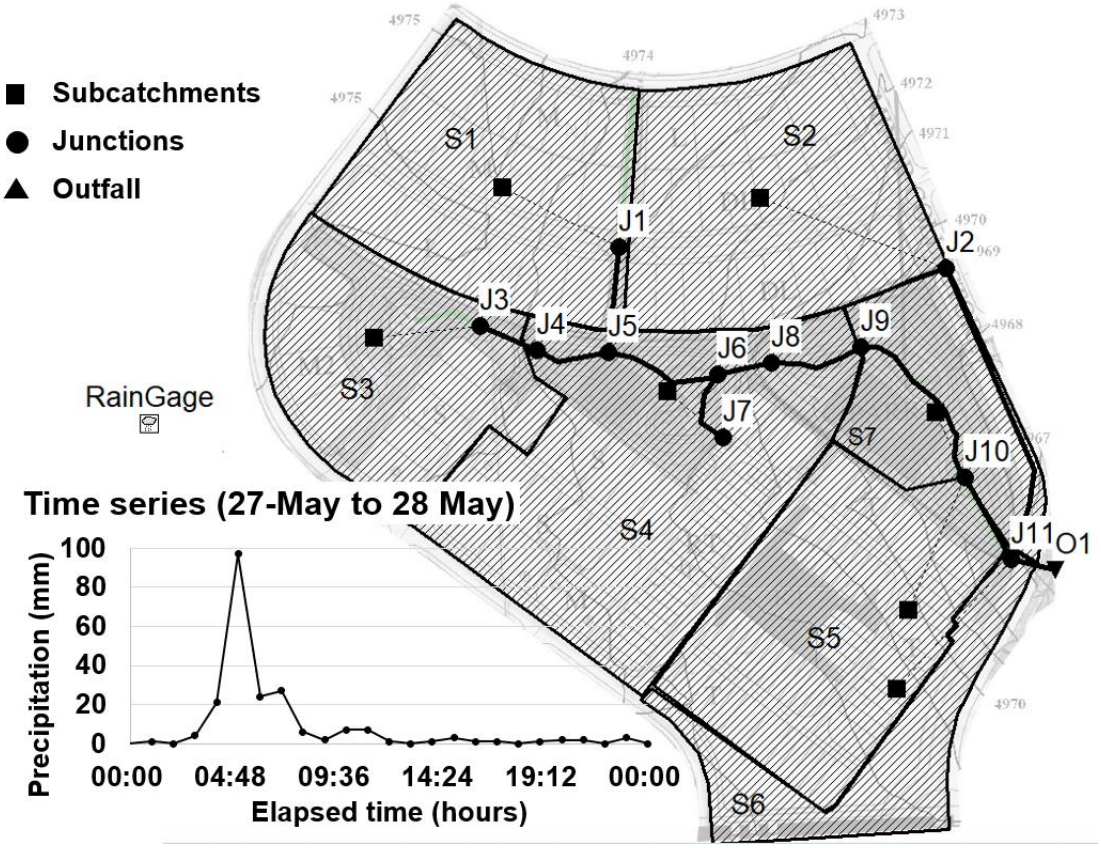


Fig. 2 Study area of SWMM case. The junction, outfall points, and sub-catchments are marked on the map and numbered J1 to J 11, O1, and from S1 to S6 respectively.

2.2 Preparing the OpenFOAM cases

(1) Copy example case

To set up an OpenFOAM case, an organized case directory is needed. For most users, copying and modifying from existing cases is the norm. In this case, the original pipe case is copied from OpenFOAM and modified manually.

```
cp -rf origin_case_dir your_case_dir
```

(2) Create mesh

To simulate the three-dimensional hydrodynamics of different drainage facilities a mesh should be first created for them. Fig. 3 shows the information about creating them. Several methods can generally be used for it including the hexahedral structural mesh created by “blockMesh”, “extrudeMesh” and “snappyHexMesh” in OpenFOAM. Some meshes provided by external tools can also be imported into the case if a suitable mesh conversion tool is used such as the “fluentMeshToFoam” for

Fluent, “gambitToFoam” for Gambit, etc. In this example case, this creation information is stored in the system/blockMeshDict file.

```
blockMesh
```

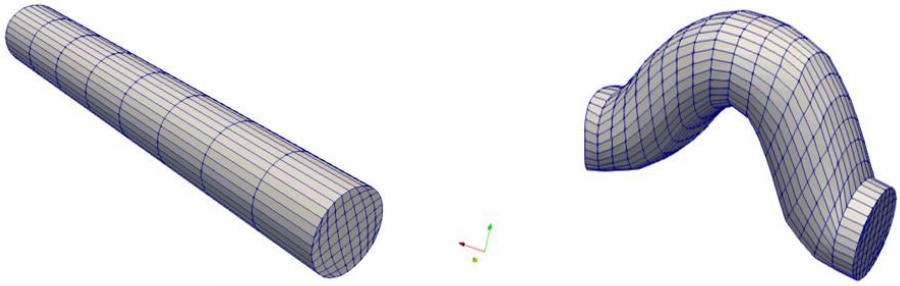
(3) Setup the initial and boundary condition

The initial conditional information for all simulated variables (e.g., velocity and pressure) is stored in the file 0/fieldname. The settings of p and U are vital in this case. This can refer to the user’s guide of OpenFOAM:

https://cfd.direct/openfoam/user-guide-v8/

(4) Setup the system file and preprocess the

Other parameters that control the simulation process, such as time step, write method, etc., are set in system/controlDict. This information may be automatically rewritten when we set the parameters of the coupling model.



Parameters	Description
Meshing Approach	Built-in function for simple meshes within OpenFOAM (“blockMesh”)
Geometry of meshes	Hexahedral structural mesh
Number of cells	500-2,000 for pipes
Boundary mapping	Junctions for pipe segment J6, J8 in SWMM to “inlet” and “outlet” the pipe in OpenFOAM meshes

Fig. 3 Meshes constructed and boundary mapping in OpenFOAM

2.3 Setting the initial and boundary conditions for each model

After the SWMM and OpenFOAM cases are set, their boundary conditions and initial conditions need to be assigned as well. In this step, the operation is the same as it is in the original model of each, e.g., in SWMM, the outflow is set to free outflow, and it is assumed that there is no water in the pipe at the beginning. Whereas in OpenFOAM the pipe walls were set to be impervious boundaries with wall functions

to set the roughness and the initial condition of inflow/outflow boundaries were also set to 0, etc. Among these settings, the most important ones are to specify the coupling boundaries between SWMM and OpenFOAM as shown in Fig. 1.

2.4 Set-up of the controlling parameters

The options file that controls the running of the coupled model should be edited after step (3). Using the format given in the manual, we need to set which model to be started and the boundary mapping mode between OpenFOAM and SWMM. We also need to set the start and end times of the coupled models (the whole year of 2019 in four example cases), the time step, and so on. For more details of setting the files of model input parameters, please refer to section 3.2.

2.5 Run the coupled model

Finally, the coupled model can be launched by running the command line.

```
python3 main.py
```

Note that if you need to run in parallel, you also need to add parallel in your parameters and set the parallel case in OpenFOAM.

3 SETTING OF FILES USED

3.1 File structures of model and cases

The file structures of the model and cases are shown in Fig. 4. Users can modify the model input files and case files for their research. Advanced users can also modify the corresponding modules to achieve their desired functionality.

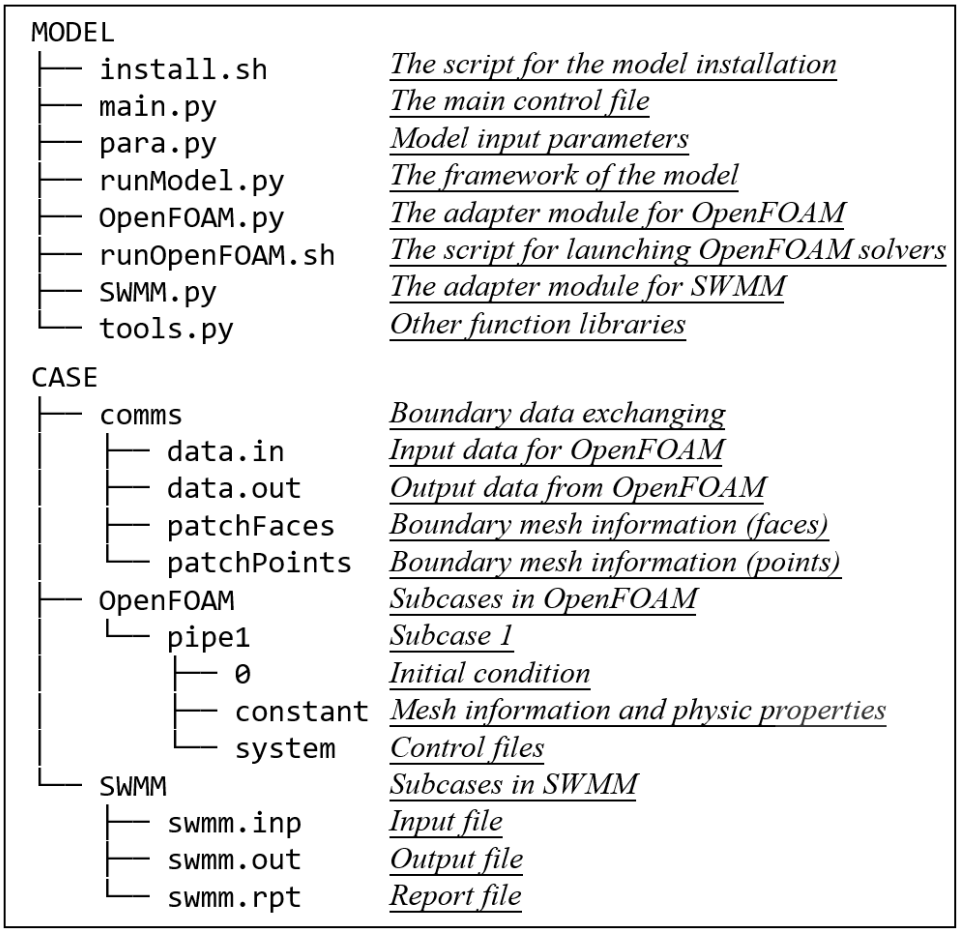


Fig. 4 File structures of model and cases

3.2 Setup of model input parameters

Among all model files, “para.py” stores the input parameter for controlling the whole model. Below is an example case of this file and main parameters (Table 1). The input parameters are divided into three categories: (1) I/O control parameters (2) time control parameters (3) boundary control parameters.

Table 1 Model input parameters

Parameters	**Description**	**Parameters**
dir	Specify case directory	
outSWMM	Output data for SWMM	*A list including parameters for outputting SWMM results:* ***name:*** *name of the node* ***type:*** *type of boundary value*
case	Input data for each subcase	***model:*** *model's name* ***name:*** *subcase's name* ***timeControl:*** *dictionary of time control* ***boundaryControl:*** *list of coupled boundaries*
cases	The sequences of subcases	*A list includes all subcases to run in order*
timeControl	The time data for each case running	***startTime:*** *start time for the subcase* ***endTime:*** *end time for the subcase* ***timeStep:*** *time step for each subcase* ***offSet:*** *time offset for each subcase*
boundaryControl	The boundary data for each case running	*A list including parameters in order:* ***Boundary control name*** ***Field type*** ***Boundary name*** ***Dictionary****: contains more specific information for each model*

The main I/O control parameter is the case directory and setup in parameter: dir. The "outSwmm" also stores the parameters to control the output data of the SWMM model. The time control parameters include start, end time, and time step. Users can input them in the format of string or float. Also, a parameter "offSet" is added for converting relative time in each subcase to a consistent time for the whole model. The boundary parameters are set in a list for each subcase. Their name, field type, corresponding boundary name, and more specific info in models are recorded in the list for each coupled boundary.

```
import os

# Dir
dir = os.path.dirname(os.path.realpath(__file__)) + "/Case"
outSWMM = [["J6", "depth"], ["J8", "total inflow"]]
# timeControl
```

```
timeControl01 = {"startTime": "11/01/2017", "endTime":
"11/10/2017", "timeStep": 300, "offSet": 0}
timeControl02 = {"startTime": 0, "endTime": 300, "timeStep":
30, "offSet": "11/02/2017"}
timeControl03 = {"startTime": 0, "endTime": 300, "timeStep":
30, "offSet": "11/03/2017"}

# boundaryControl
boundary01 = [
    [
        "U_Coupled01",
        "U",
        "J6",
        {"model": "SWMM"},
    ]
]
boundary02 = [
    [
        "U_Coupled01",
        "U",
        "inlet",
        {"model": "OpenFOAM"},
    ],
    [
        "U_Coupled02",
        "U",
        "outlet",
        {"model": "OpenFOAM"},
    ],
]
boundary03 = [
    [
        "U_Coupled02",
        "U",
        "inlet",
        {"model": "OpenFOAM"},
    ]
```

```
]

# Cases
## SWMM
case01 = {
    "model": "SWMM",
    "name": "swmm.inp",
    "timeControl": timeControl101,
    "boundaryControl": boundary01,
}

## OpenFOAM
case02 = {
    "model": "OpenFOAM",
    "name": "pipe1",
    "timeControl": timeControl02,
    "boundaryControl": boundary02,
}
case03 = {
    "model": "OpenFOAM",
    "name": "pipe2",
    "timeControl": timeControl03,
    "boundaryControl": boundary03,
}

# Sequence of cases running
cases = [case01, case02, case03]
```

4 COUPLING METHODS

4.1 Coupling between OpenFOAM and SWMM

To integrate OpenFOAM solvers into SWMM, the flow rate or water level are exchanged at the coupled boundary between OpenFOAM and SWMM. These boundary data of selected nodes from SWMM is first generated through the built-in function in pyswmm. The model then transforms the data into a recognizable format for OpenFOAM. This process requires the user to set the correspondence between each SWMM boundary and the OpenFOAM boundary in the control file. Then, the OpenFOAM solvers start running for each iteration. Finally, the updated boundary data from OpenFOAM is returned to the SWMM part and starts the cycle for the next iteration.

The newly developed tool “externalCoupledNew” and “mapPatch” are also employed to export and read the OpenFOAM boundary fields. This considers the change in the dimensionality of the boundary data (from OpenFOAM to SWMM), and the details of the implementation refer to the codes and manual. Meanwhile, it is important to note that pyswmm can only set the water level of outfalls (the terminal nodes of a drain line). This requires attention when setting the boundary types (i.e., water level and flow rate) for different nodes.

4.2 Coupling within OpenFOAM part

To simulate different areas (i.e., surface water zone, hyporheic zone, and soil zone) with multiple physics, the interFoam, Darcy-Forchheimer, and groudwaterFoam solver were chosen are presented and recommended in the manual.

The interFoam solver is for two incompressible, isothermal immiscible fluids using a volume of fluid (VOF) method. This solver can be used for multiphase simulation and in this study, the first and second phases are set as water and air, respectively. The Darcy-Forchheimer solver has been coupled into interFoam

through the “fvOption” tool in OpenFOAM. The groudwaterFoam solver in the toolbox for porous media developed by Horgue (P. Horgue et al., 2015) is used for the simulation of groundwater percolation for some drainage facilities involving soil. This solver has been verified and applied in previous studies (Pierre Horgue et al., 2015).

However, these solvers can only be applied to their respective regions. It is necessary to customize a tool for coupling the solvers together. Aiming at the boundary exchange between different solvers, this study establishes a new boundary condition “externalCoupledNew” and a new utility “mapPatch” to export and read the OpenFOAM boundary fields, respectively (details in section 5). These two tools consider the variation of density and shape of the boundary meshes, coupling is achieved by interpolation and mapping between each subcase. However, the error of this process increases when the variation becomes large, so it is recommended to set the boundary mesh of approximate shape and density for coupling. The workflow of the module is shown in Fig. 5. The iterative coupling method is applied, and users can set convergence criteria (e.g., CourantNo) or the number of iterations to control the processes. Pressure and velocity are the parameters to be exchanged between different regions with the mapping method (i.e., projecting the data from one to another). Meanwhile, the overall time steps are determined by the minimum one among three solvers when using the adjustable time step or are set to be an integer multiple of the minimum one when fixed time steps are used.

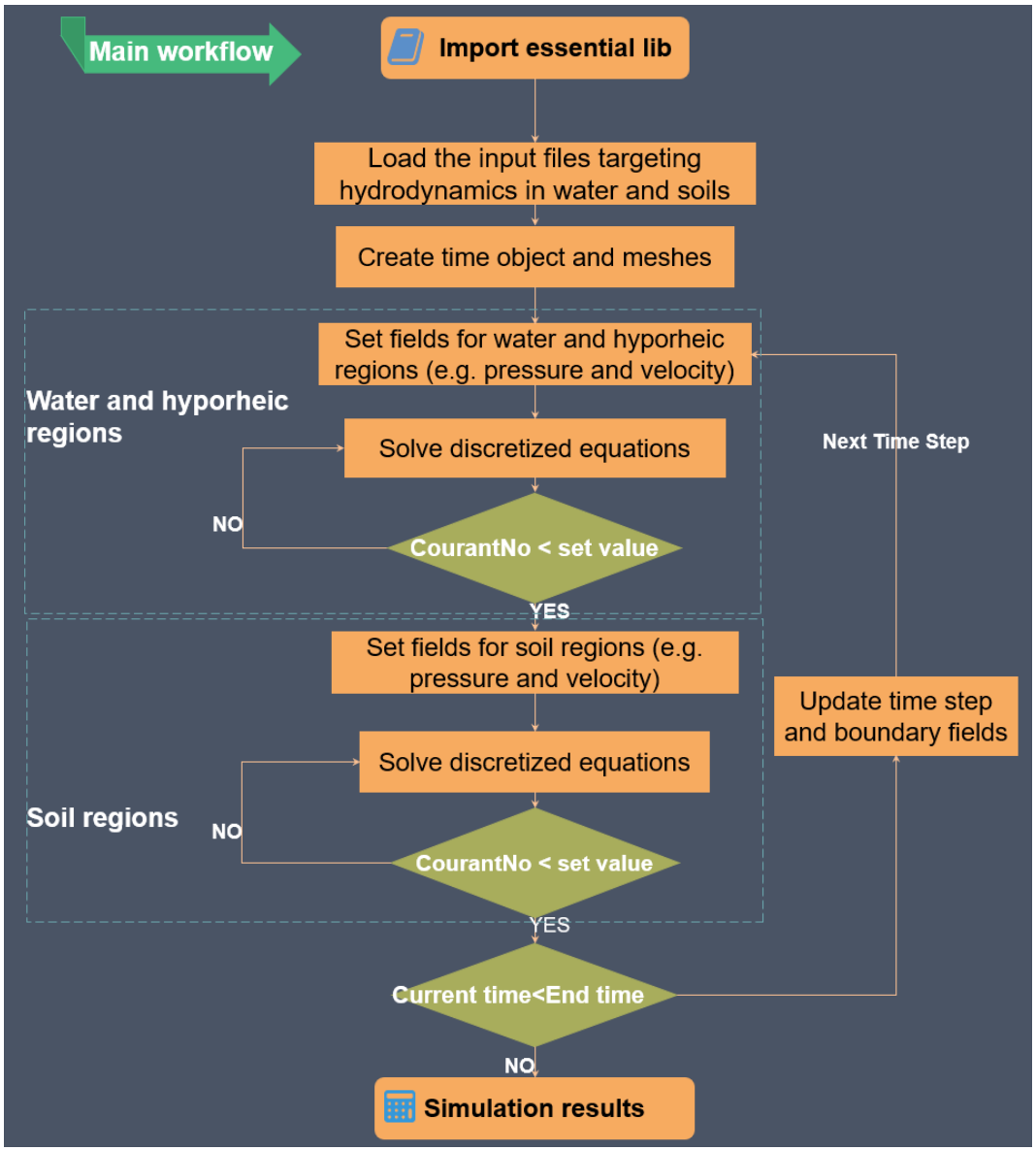


Fig. 5 The workflow of module for coupling solvers in OpenFOAM

4.3 Coupling other models

This established framework of the model can also integrate other models into SWMM to account for more situations in drainage systems. It can be implemented by following two steps:

1. Develop an adapter module to start the customed models

To standardize the code of the model, the new module is better named with the coupled customed model's name. The class "case" that inherits from multi-threaded or multi-process modules is consistently used for startup all subcases from the

customed models. Meanwhile, the following essential functions need to be implemented in the module.

Table 2 Essential functions for customed models in the adapter module class

Name	**Function**
__init__(self)	*Create a new case for running*
run(self)	*Start model simulation*
getSem(self)	*Setup semaphore for threads or processes to sync data*
TIME(self)	*Get current time with offSet*
end(self)	*Check if the case is finished*
BOUNDARY(self)	*Get boundary data*
setBOUNDARY(self)	*Get boundary data from other models*

Since these functions are implemented in various ways for different models, users can refer to SWMM.py and OpenFOAM.py provided and modify based on the example files.

2. Import the adapter module:

To import the new module, users should add the adapter file's name by the following command in main.py:

```
runModel.clsImportModels(['OpenFOAM','SWMM','Customed_model'])
```

5 TOOLS FOR DATA EXCHANGES ON BOUNDARIES

Aiming at the boundary exchange between different OpenFOAM solvers and SWMM's output, this coupled model includes two new tools: (1) new boundary type "externalCoupledNew" and (2) new utility "mapPatch" to export and read the OpenFOAM boundary fields, respectively.

The "externalCoupledNew" is a rewritten boundary type based on externalCoupled. They are roughly similar in terms of their workflow (https://cpp.openfoam.org/v8/classFoam_1_1externalCoupledMixedFvPatchField.html) and the following shows commonly used parameters set for the boundary (Table 3). However, the new boundary allows to output data using a specific output interval instead of integer multiples of the time step, while changing the output format for easy coupling into other models.

Table 3 Usage of "externalCoupledNew" boundary

Property	Description	Required	Default value
commsDir	communications directory	yes	
file	transfer file name	yes	
waitInterval	interval [s] between file checks	no	1
calcInterval	calculation interval [s]	yes	
timeOut	time after which error invoked [s]	no	100*waitInterval
calcFrequency	calculation frequency	no	1
initByExternal	external app to initialize values	yes	
log	log program control	no	no

Example of the boundary condition specification:

```
    <patchName>
    {
        type            externalCoupledNew;
        commsDir        "$FOAM_CASE/comms";
        file            data;
        calcInterval    30;
        initByExternal  yes;
    }
```

The “mapPatch” is a tool for mapping boundary fields between OpenFOAM solvers or transferring boundary data from SWMM to SWMM. It can be run by command line, here is the specific meaning of the main parameters (Table 3).

Table 4 Usage of “mapPatch” utility

Parameters	Description	Required	Default value
case <dir>	Specify case directory	no	Current directory
from <filename>	Source .out file (for SWMM)	no	Data.out
to <filename>	Target .in file (for SWMM)	no	Data.in
fromFiles <filenames>	Source .out file (for OpenFOAM)	yes	
toFiles <filenames>	Target .in file (for OpenFOAM)	yes	
v	Vector boundary fields	yes	
s	Scalar boundary fields	yes	

The “mapPatch” considers the variation of density and shape of the boundary meshes, coupling is achieved by interpolation and mapping between each subcase. Meanwhile, there are two mapping methods in “mapPatch” (Fig): point to point and face to face. The default method is face to face in this tool and can be changed by the interpolation method of “PatchToPatchInterpolation” in the source code. The figure shows how the method map fields when the boundary mesh is not consistent in different subcases.

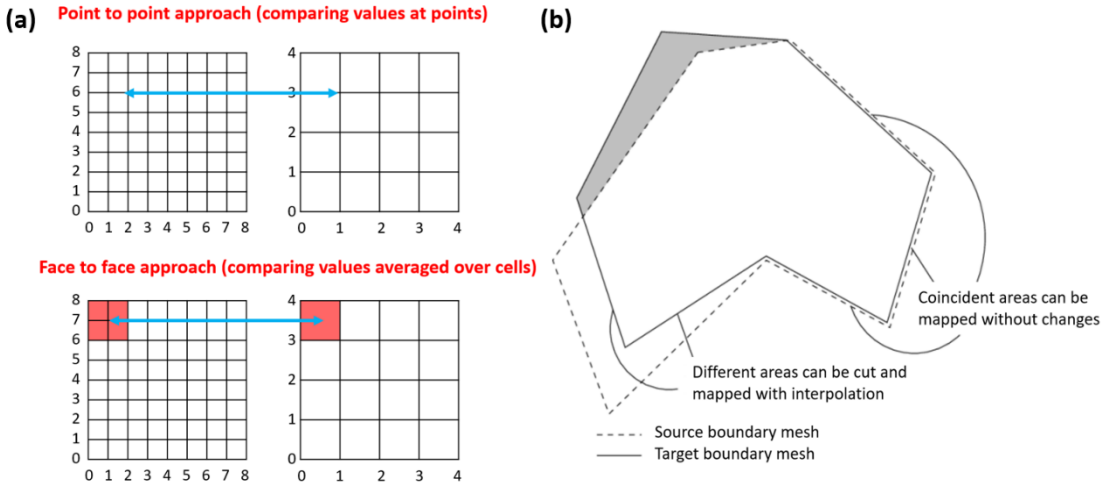


Fig. 6 Mapping methods for coupled boundaries, (a) is for different boundary mesh densities and (b) is for different boundary geometries.

However, the error of this process increases when the variation between source

and target boundaries becomes large, so it is recommended to set the boundary mesh of approximate shape and density for coupling. In addition, the tool changes the dimensionality of the boundary data (from OpenFOAM to SWMM), (e.g., from the velocity fields to total flow rate). It may suffer from loss of boundary data accuracy, i.e., the distribution of the boundary fields on the boundary of the OpenFOAM case. Our solution is to use the boundary field distribution of the open foam case in the previous step as the reference distribution and run iteratively to reduce the loss of coupling boundary accuracy within an acceptable range (Fig. 7).

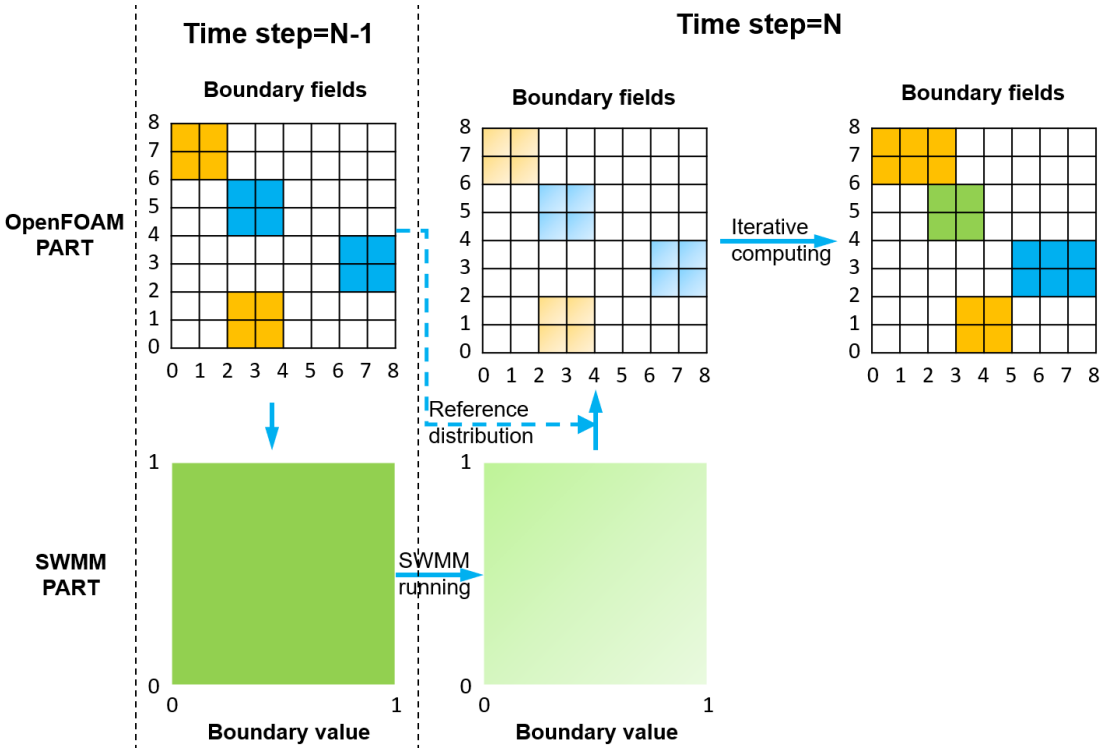


Fig. 7 Mapping boundary between OpenFOAM and SWMM, color blocks indicate different fields/values

6 POST-PROCESSING AND ATTENTIONS

6.1 Viewing results

The output of the model can be divided into two parts, one from the OpenFOAM cases and the other from the SWMM cases.

Paraview is usually utilized to post-process and display the results from OpenFOAM. Following are common steps for viewing the results:

1. Create a new file named *.foam in the root folder of an OpenFOAM case with the command:

   ```
   touch case.foam
   ```

2. Open the case results in paraview with the command:

   ```
   paraview case.foam&
   ```

After the file is opened in Paraview, the simulation results can be viewed by changing the variables and time. It can also be used to output variable changes on a boundary or at a point (Fig. 8). For more advanced operations, please refer to the Paraview user guide: https://docs.paraview.org/en/latest/.

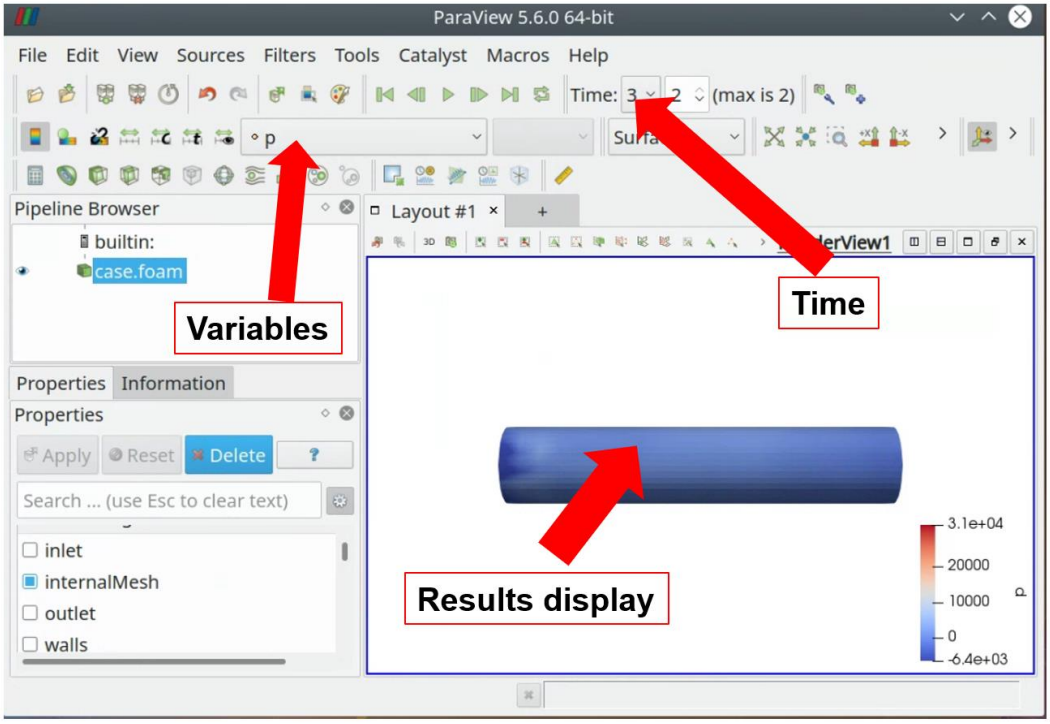


Fig. 8 Adjusting time and variables in paraview 5.6.0

For SWMM subcases, the results are stored in *.out file and can be read as tables and post-process then. If the user needs to export certain parameters for a specific facility, set "outSWMM" in the model input file and use the output file storing the source data for post-processing.

6.2 Attentions

- “Flow Routing” method in SWMM. For dealing with situations like channel storage, backwater, entrance/exit losses, flow reversal, and pressurized flow, please use the Dynamic Wave Routing method and set timestep of SWMM less than 30s.
- Avoid using variable timestep in SWMM
- The format of time (including timestep and time offset) should be str or float. other formats are not supported.

7 REFERENCES

Gironás, J., Roesner, L. A., Davis, J., Rossman, L. A., & Supply, W. (2009). *Storm water management model applications manual*. National Risk Management Research Laboratory, Office of Research and ….

Horgue, P., Soulaine, C., Franc, J., Guibert, R., & Debenest, G. (2015). An open-source toolbox for multiphase flow in porous media. *Computer Physics Communications*, *187*, 217–226. https://doi.org/10.1016/j.cpc.2014.10.005

Horgue, Pierre, Franc, J., Guibert, R., & Debenest, G. (2015). *An extension of the open-source porousMultiphaseFoam toolbox dedicated to groundwater flows solving the Richards' equation*. http://arxiv.org/abs/1510.01364

McDonnell, B., Ratliff, K., Tryby, M., Wu, J., & Mullapudi, A. (2020). PySWMM: The Python Interface to Stormwater Management Model (SWMM). *Journal of Open Source Software*, *5*(52), 2292. https://doi.org/10.21105/joss.02292