Abstract

This guide provides installation and basic operational instructions for v1.0 of the SWMM5+ hydraulics engine. This engine is a companion code to the US Environmental Protection Agency’s Storm Water Management Model (EPA SWMM). The SWMM5+ code is linked with EPA SWMM code during compilation to simulate a coupled hydrology/hydraulic system. The necessary source code and application libraries for installing and using SWMM5+ are provided in a Docker Container.
Software availability: The SWMM5+ source code is public-domain software produced under a cooperative agreement with the US Environmental Protection Agency. Formally, the code is released under the “unlicense,” whose terms and conditions can be found at https://unlicense.org. The latest version of the SWMM5+ source code can be found at https://github.com/CIMM-ORG/SWMM5plus.

Acknowledgements: This publication was developed under Cooperative Agreement No. 83595001 awarded by the U.S. Environmental Protection Agency to The University of Texas at Austin. It has not been formally reviewed by EPA. The views expressed in this document are solely those of the authors and do not necessarily reflect those of the Agency. EPA does not endorse any products or commercial services mentioned in this publication.
1 Introduction

What is SWMM5+? The SWMM5+ code is a companion to the EPA Storm Water Management Model (SWMM) that provides new capabilities for hydraulic routing using a mass-conservative finite-volume (FV) method. The new code is designed to provide efficient parallel computing for multi-core computers using Intel-based central processing units (CPU).

Public Domain: Similar to EPA SWMM, the SWMM5+ code is released as public domain (uncopyrighted) software that can be used and modified by anyone without restriction. Formally, we use the “Unlicense” license. The SWMM5+ code can be downloaded or cloned from a public repository hosted on github.

Expected performance: The SWMM5+ code is expected to provide three major improvements:

1. improved control over mass conservation
2. faster solution on large systems when using multi-thread computers
3. direct simulation of gradients within links

An important caveat is that SWMM5+ is likely to be slower than EPA SWMM on smaller systems or when only a single processor is used. Because SWMM5+ uses sub-discretization of links it has more elements to compute, so the increased speed of parallelization is somewhat offset by the larger number of computations. Furthermore, the explicit time-marching algorithm might require a smaller time step than used in EPA SWMM.

The important performance advantages of SWMM5+ will be seen in large systems run on multi-core computers. A “large” system can be thought of as one that has more 1000 elements per computational core. The reason for this can be understood by considering what would happen with only 1 element per core—in such a case every computation on a core requires 2 communications to neighbor cores so the communication slow-down will dominate the speed-up associated with multi-core computation. With effective partitioning a large number of computations per core can be coupled with a small number of communications, which provides significant parallel speed-up.

Users who will benefit most from SWMM5+ are those who have EPA SWMM models with intractable problems in loss of mass conservation and/or oscillatory solutions.

Coupled compiling: SWMM5+ is designed to link with EPA SWMM at compile time. As such, SWMM5+ cannot be run without linking to EPA SWMM. Furthermore, SWMM5+ cannot be compiled separately and cannot connect to an EPA SWMM executable—the two source codes must be compiled together to create the necessary library and executable files.

Linking SWMM5+ and EPA SWMM: Conceptually, the operation of SWMM5+ is quite simple — we are replacing the inner-loop (hydraulics) solution for the dynamic wave within EPA SWMM with a new algorithm. However, because of code structures required for parallelization in CoArray Fortran, the overall command of all operations in a coupled SWMM5+/EPA SWMM solution resides within the SWMM5+ Fortran code; that is, SWMM5+ is the main calling

1Porting to next-generation GPU machines should be possible with the SWMM5+ code structure, but likely will require significant re-coding.

2https://unlicense.org/
3https://github.com/CIMM-ORG/SWMM5plus
program for the coupled code and the EPA SWMM procedures are used as library routines. Thus, the entry into the coupled program is `main.f90`. SWMM5+ calls procedures within EPA SWMM that

- read EPA SWMM input files,
- access initialization variables,
- conduct hydrology time steps,
- apply EPA SWMM controls/monitoring

**Compiling using Intel oneAPI:** SWMM5+ has been developed on Linux machines and ported to Windows machines using a Docker container with the Intel oneAPI compiler. The Docker container will also work with Mac OSX, with the unfortunate limitation that only a single processor can be used. There appear to be hardware/firmware problems with Apple computers that prevent CoArray Fortran from providing multi-core computations.

We have also done some preliminary work with the open-source gfortran compiler coupled with the OpenCoarrays library, but there remains further porting work to make the SWMM5+ gfortran compatible.

**A note on executables:** We have developed SWMM5+ using the Intel oneAPI compiler as a freeware download. Users can also download Intel oneAPI for free and use it to compile and run their own SWMM5+ models. However, distribution of executables from the freeware version of Intel oneAPI is prohibited by their copyright restrictions. Users interested in developing executables of SWMM5+ for transfer between machines and/or users will need to purchase a oneAPI license from Intel.

**EPA SWMM compatibility:** The present SWMM5+ release has been compiled and tested with EPA SWMM version 5.1.013. It is likely that SWMM5+ will also work with later releases that are bugfixes, but we have not yet conducted comprehensive testing.

**Changes to dynamic routing:** The most important change between EPA SWMM and SWMM5+ codes is the introduction of the finite-volume (FV) method for solving the dynamic wave equation in the hydraulic solution. This has two major impacts: (i) the “links” of EPA SWMM are subdivided into multiple FV “elements” connected by “faces”, and (ii) the “nodes” of EPA SWMM become a special multi-branch FV elements. Whereas EPA SWMM solves for flowrate in the links and pressure (head) at the nodes, in SWMM5+ we solve for flow and head in every FV element, and interpolate these values to the faces between elements.

**Input files:** The standard *.inp input files from EPA SWMM are used with SWMM5+. The SWMM5+ code uses the EPA SWMM file-reading procedures to access the input file. To handle the custom features of SWMM5+ we use a *.json file that provides additional input.

**Output files:** SWMM5+ produces link-node output and finite-volume (FV) element output in two ways: (i) using text *.csv files, and (ii) using HDF5 binary files. Users have the option to get the full FV output (multiple elements per link) or have the FV data compressed back into the equivalent link-node system. For example, to compress the FV data to link-node data, the volume of a link will be the sum of all the volumes of the FV elements and the flow rate will be the average over the elements.

Note that SWMM5+ does not provide hydraulics output in the EPA SWMM *.out format. This limitation is due to the excess computational time required for translating the parallel-partitioned FV system back into an EPA SWMM link-node system.
Model time step: The EPA SWMM hydraulics algorithm poses the governing equation in an implicit discrete form and obtains a solution by iteratively converging the heads and flowrates in the link-node system. For SWMM5+ this approach is replaced with explicit time-marching using a Runge-Kutta 2nd order (RK2) method. The key advantage of the new approach is that it is easy and efficient to parallelize over a large number of processors. However, the explicit RK2 has the disadvantage that the allowable time step is strictly controlled by the Courant condition, which can be taken as a Courant-Friedrichs-Lewy (CFL) number less than $\sqrt{2}/2$. The SWMM5+ code is designed to compute the allowable time step during the simulation and automatically adjust the time step for optimal solution.

Changes for surcharge pipe: Like EPA SWMM, the SWMM5+ engine is designed to solve conservation of mass and momentum equations throughout a system—i.e., the Saint-Venant equations in open-channel flow and the incompressible equations for surcharged pipe. EPA SWMM has two options for surcharged pipe: a head-flow balance approach (EXTRAN) and, more recently, the Preissmann Slot. For SWMM5+ the only surcharge algorithm is the Preissmann Slot. We have developed a new non-dimensional formulation of the Preissmann Slot that solves many of the prior oscillation problems.

Limitations on EPA SWMM options: The present SWMM5+ includes about 90% of EPA SWMM functionality. We intend to eventually support all EPA SWMM functions, but at this time the present features are not included

1. culverts
2. multi-barrel conduits
3. pollutant transport
4. inflow patterns
5. dry time steps

The following geometry types are not yet supported

1. triangular cross-section
2. power function cross-sections
3. custom closed conduits

A number of features have not been fully tested. Users are encouraged to report problems that they encounter

1. pumps
2. controls
3. LID drainage
2 Quick Start Guide

2.1 Overview

To install, compile and run SWMM5+, users will need to be familiar with working with a command-line terminal for their operating system. Herein we use Command Terminal to refer to an application on any operating system that allows the user to navigate between directories and type commands to the operating system.

This installation procedure may require you to have Windows administrator or Linux/-MacOS superuser privileges.

This installation may take up to 25 GB of disk space on your computer.

Using SWMM5+ requires:

1. installing the Docker environment
2. installing the SWMM5+ code and libraries
3. compiling with the Intel OneAPI compiler (provided in the SWMM5+ Docker image).
4. running from the command line.
5. visualization of results using Python or custom HDF5 file access.

These steps are described in detail in the sections below

The SWMM5+ v1.0 code is run within a “Docker” container. The setup procedure below requires that the user install the Docker Desktop application, download the SWMM5+ tar file, and then “mount” the volume on your machine. The Docker installation process creates a Linux virtual machine on your local computer within a “container” that includes all the source code, libraries, and dependencies to compile and run SWMM5+. We use this approach to limit conflicts between SWMM5+ and existing libraries on your computer. Users who haven’t previously used Docker containers should read the Docker documentation to understand the capabilities and limitations of the system. The Docker app and its documentation can be found at [https://docs.docker.com/](https://docs.docker.com/)

Note that a Docker volume must be created to use the Docker Image (see Step 4, below). Docker uses volumes to allow for persistent memory — i.e., so that work done in the container does not disappear when you close the container. **WARNING: Docker will function without a mounted volume but you will lose all your data when you close the container and stop Docker.** So correctly mounting the volume is crucial.

2.2 Jargon

Some useful definitions and jargon:

**Command Terminal:** An application on your host computer that allows you to type commands to the operating system.

**Docker App:** The application on your host computer that creates and manages virtual machines.

**Docker Container:** A virtual machine and its associated files that may be running (open) or stopped (closed). Note that if a Docker Volume has not been “mounted” then closing a Container will destroy all its data.
Docker Image: A large compressed file that contains all the data for Docker to create and run a virtual machine.

Docker Volume: A relationship between files created/stored in the Docker container and a location in the file/folder structure of the host operating system.

2.3 File locations

Given below are recommended folders that should be created before installing SWMM5+ using Docker

1. Install the Docker application. We recommend users use the default installation location for your system.
2. Install Docker Image. We recommend, for Windows a user named “Fred” would have C:\Users\Fred\Desktop\Working\Docker_Image. In the following guide we will call this folder path <dimagepath>
3. Install SWMM5+ code. We recommend, for Windows a user named “Fred” would have C:\Users\fred\Desktop\Working\SWMM5plus_Default In the following guide we will call this folder path <swmmpath>

2.4 Installing SWMM5+ using Docker

Step 1. Docker installation: Install the Docker Desktop application for your host computer’s operating system from:

https://www.docker.com/get-started

Step 2. Downloading the Docker Image  The Docker Image is a custom file containing all the data for the Docker app to create the “container,” or virtual machine on your computer that will work with SWMM5+. Our Docker Image is 15 GB and can be found at:

https://www.cimm.org/swmm5plus

The downloaded file name is SWMM5_plus_docker.tar. Move this file to <dimagepath>, see §2.3. Note, that you should not untar or uncompress this file.

Step 3. Loading the Docker Image  In a Command Terminal, navigate to <dimagepath>, see §2.3. At the command line type...

docker load --input SWMM5_plus_docker.tar

...then hit return. the Docker App will install the Docker Image, which might take several minutes. To check for success, open the Docker App and go to the Image Menu and you should see the new image listed as SWMM5plus.

Step 4. Creating a Docker Volume  In the Command Terminal, create a Docker Volume named SWMM5plus by typing...

docker volume create SWMM5plus

...then hit return. In the above, the SWMM5plus will show up in the Docker App as a “volume” that will have persistent data. The Docker Volume only needs to be created once. After the installation has been completed this volume will always be connected to the Docker Container(s) you create using this Docker Image. The location of the Docker Volume on your machine is set by the Docker App (see Docker documentation).

Step 5. Delete the Docker Image file  You can delete the SWMM5_plus_docker.tar as it is no longer needed.
Step 6. Get the SWMM5+ v1.0 Release Code from Github

Download the latest SWMM5+ release from

https://github.com/CIMM-ORG/SWMM5plus/releases/tag/v1.0

You can also get to this location from our main github page by going to https://github.com/CIMM-ORG/SWMM5plus and looking for the Releases section on the right side of the page.

Move this compressed file to your chosen location <swmpath>, see §2.3. Use a decompression application (e.g., winzip, winrar, 7zip) to uncompress the file. Note that the compressed file has a top-level folder named SWMM5plus-1.0, which will be the subdirectory of your <swmpath> for the SWMM5+ installation.

Step 7. Creating a Docker Container

To create the Docker Container named swmm_container, open a Command Terminal (or use one already open). Your current directory is irrelevant for the next commands as Docker can be run from anywhere at the command line. If you are on a Windows machine, at the command line, type...

docker run -it --name swmm_container -v <swmpath>:/home swmm5plus

...and hit return. For example, our user named “Fred” might type...

docker run -it --name swmm_container -v 
C:\Users\Fred\Desktop\Working\SWMM5plus_Default:/home swmm5plus

All the above should be on a single command line.

With this setup, any files or folders that you drag into your <swmpath> folder will be available for use within the Docker Container.

Step 8. Check that Container is running

You can check that the Docker Container is running within the Docker App GUI by clicking the Container Menu. You should now see a container named swmm_container

Step 9. Create a compilation environment

From a Command Terminal, type in...

docker exec -it swmm_container bash

...and hit return. This creates a Linux bash shell for use with Docker. For this Docker command it doesn’t matter which directory you are in when you type the command.

Within the Command Terminal, navigate to the SWMM5plus-1.0 directory. Type in...

cd SWMM5plus-1.0

...and hit return. If you have moved away from the home directory, you would use

cd /home/SWMM5plus-1.0

Create a build folder for compilation as a subfolder of SWMM5plus-1.0, type in ...

mkdir build

...and hit return.

Navigate to the build subdirectory by

cd build

...and hit return.
Step 10. Compile the code  Compilation is a two-step process. First, at the command line within the build folder type...

  cmake ..

... and hit return. The above invokes the Cmake utility to install and check on dependencies and libraries. Note that the blank space after cmake and the two dots .. are needed to make it work correctly.

The second step is the actual make (compilation) of the code. At the command line, type ....

  make

... and hit return. Note that the first compilation of the code and libraries may take more than an hour because of the need to compile the HDF5 library. After the first compilation the libraries do not need to be recompiled so the system will compile faster.

Step 11. Setting up to run SWMM5+  Navigate back to the the SWMM5+ folder, e.g. from the build folder type...

  cd ..

...and hit return. Alternatively, type...

  cd /home/SWMM5plus-1.0

...and hit return.

Before running SWMM5+ you need to set the number of processors for a simulation. As a starting point, you should use only a single processor. At the command line, type

  export FOR_COARRAY_NUM_IMAGES=1

...and hit return. When you want to use more processors, replace the 1 with an integer that is consistent with the number of processor on your machine.

Project folder — It is convenient to have a project folder that holds the two input files required in SWMM5+, the standard *.inp of EPA SWMM and a *.json file that controls commands for SWMM5+. We recommend these project folders should be created as subfolders of the /home directory. For example, a project folder might be named /home/project01 and it could have input files of /home/project01/test.inp and /home/project01/test.json. Matching names of the *.inp and *.json files is a convenient way of ensuring that you are always using the configuration that you intend.

Output folder — It is usually best to have a separate folder for output. SWMM5+ creates a large number of output files, and having them in a separate folder from the input makes it easier for the user to delete unneeded files.

Json file — For users getting started with SWMM5+, the minimum SWMM5+ Json file can be very simple, as shown below

```json
{
  "JSON_FoundFileYN" : true,
  "JSON_CheckAllInputYN" : false,
  "CaseName" : {
    "Long" : "long␣case␣name",
    "Short": "ShortName"
  },
  "Discretization" : {
    "NominalElemLength" : 20.0
  }
}
```
In the above, the long case name can be up to 256 characters and the ShortName up to 16 characters. Note that the ShortName is used for naming output files from SWMM5+ so it is best if it has simple alphanumeric characters without blank spaces. The critical piece of information in the Json file is the target nominal element length (in meters) for subdividing links. Above it is set to 20 m. There must be at least 3 sub-elements per link in SWMM5+ and the SWMM5+ simulation will stop with an error message if you try to set too long of a length. Note that the nominal element length is in meters even if the EPA SWMM *.inp file is set to US units. More detail on the Json file will be provided with the SWMM5+ user guide (in development).

Step 12. Running SWMM5+  At this time, SWMM5+ must be run from the command line with the current working directory as the /home/SWMM5plus-1.0 directory. To get there, type...

```
    cd /home/SWMM5plus-1.0
```

... and hit return. This restriction will be lifted in future versions.

SWMM5+ is run from the command line with the general form of

```
<executable> -p <project_folder> -i <file>.inp -s <file>.json -o <out_folder>
```

which must be all on one command line. Note the placeholders within the <...> are paths and filenames of your setup, as described below

The executable is SWMM and the required storage location is the build folder, so the executable name is ./build/SWMM, where the initial ./ tells the command to start from the current working directory, which must be /home/SWMM5plus-1.0.

Using the above, a set of commands to run SWMM5+ might be

```
./build/SWMM -p ../project01 -i test.inp -s test.json -o ../outdata
```

In the above command, the -p ../test01 tells SWMM5+ to look in the directory above the current directory (../) and use the folder project01 as the project folder. SWMM5+ will look for the files test.inp and test.json within the project folder to read into the simulation. The output from SWMM5+ will be written in subfolders of the folder outdata that is also in the directory above the current directory.

The above command could also be written with absolute directories for the project and/or output, e.g.,

```
./build/SWMM -p /home/project01 -i test.inp -s test.json -o /home/outdata
```

Users may also skip the project folder and write all absolute directories for both input files and the output, e.g.,

```
./build/SWMM -i /home/project01/test.inp -s /home/project01/test.json -o /home/outdata
```

The -o flag is optional, but if it is not provided an output folder will be created as a subfolder of the /home/SWMM5plus_v1.0/build/ directory.

SWMM5+ will also run without a Json file if the -s option is missing. In this case, the code uses the default configuration in the module define_settings.f90. However, users are cautioned that the default NominalElemLength is 10 m, which might not be suitable for the user’s system.

Step 13. Shutting down the Docker Container  If you are finished with running SWMM5+ and want to free up resources for your machine, you can go to the Container menu on the Docker App and press the square stop button next the container named swmm_container.

The Docker App can be closed after the Docker Container has stopped.
**Step 14. Restarting down the Docker Container**  
To restart after Docker has been closed down, open the Docker App and go to the *Container* menu. Click the arrow *start* button next to *swmm_container* to restart the Docker Container.

Open a command terminal and type...

```
docker exec -it swmm_container bash
```

...and hit return. This step is necessary *every time the Docker Container is restarted* so that you can use the bash commands for running SWMM5+ as described above.

You may want to reset the number of processors; type...

```
export FOR_COARRAY_NUM_IMAGES=1
```

...and hit return.

You are now ready to setup and run additional SWMM5+ simulations

**Step 15. Accessing test cases**  
There are several test cases provided in the SWMM5+ v1.0 release. These can be found in subfolders of

```
/home/SWMM5plus_v1.0/test_cases
```

For example, to run the *Example1.inp* test case, navigate to the `/home/SWMM5plus_v1.0` directory and type...

```
./build/SWMM -p ./test_cases -i Example1.inp -s settings_dx10.json -o /home/outdata
```

**Step 16. Animated Profiles with Python**  
After running simulations a *output.h5* file will be created along with the ..*csv* files we have chosen to have an option to use HDF5 to store link-node and FV output as a replacement or used in conjunction with the *.* files. HDF5 is a library and file format created for high performance and large complex datasets, this makes it useful for SWMM5+. However, unlike *csv* files, HDF5 files aren’t readable with ASCII text and require scripts or external programs to access the data.

Current release includes a python script called *profile_animate.py*, which utilizes the HDF5 output. This script animates the piezometric head for a given profile of a network. A profile of a network can be created through the EPA SWMM5.1 GUI. Alternatively, profiles can also be added at the end of the SWMM input file as,

```
[PROFILES]
; Name Links
;--------------- ----------
"example_profile" link_1 link_2 link_3 link_4 link_5
```

here, *link_1, link_2, ...* should be the name of the actual links in the network that the user wants to include in the profile animation. After running the simulation, this script can be run from the SWMM5plus_v1.0 foldes as,

```
python profile_animate.py -o <path to the HDF5 file>
```

For example, running the *Example1.inp* test case using the command from Step 15, will create the HDF5 file at a timestamped folder location as,

```
/home/outdata/Example1_output/test_20230608_1516
```

Now, any saved profile can be animated as,

```
python profile_animate.py -o /home/outdata/Example1_output/test_20230608_1516
```
Besides the profile_animate.py, we do not provide users with tools to directly access and extract the data stored in the hdf5 files created during output, however if users wish to access the .h5 output they can use the HDF Group’s program HDFview [https://www.hdfgroup.org/downloads/hdfview/](https://www.hdfgroup.org/downloads/hdfview/). HDFview allows you to open and access the data stored in an .h5 file, which can then be exported to other programs like Excel or Google Sheets.

In the weeks following the release of the code, we plan on creating python functions for extracting and plotting time series output data from SWMM5+, these will be released in subsequent versions of the code.