SWMM5+

Info, Access, and FAQs

Table of Contents

Introducing SWMM5+

SWMM5+ is new a hydraulic engine for USEPA SWMM that replaces the traditional dynamic wave solver. The new engine:

1. fixes long-standing mass conservation problems with EPA SWMM;
2. provides smoother transitions between free-surface and surcharged flows; and
3. uses sub-discretization for heads and flows within links to capture detailed hydraulic features (e.g., hydraulic jumps, bores, drawdown to supercritical flow).

Why should I use SWMM5+?

SWMM5+ was developed to provide the following benefits over the current USEPA SWMM version 5:

1. better confidence in the hydraulic solution for systems where mass conservation and oscillations are problem for EPA SWMM:
2. detailed head/flow hydraulics with a pipe link, and
3. allow large systems on supercomputers with efficient parallelization over 100s or 1000s of processors.

However, if you are satisfied with your present USEPA SWMM model then you probably don't need SWMM5+.

What isn't in SWMM5+

The development team is working towards a SWMM5+ code that will contain all the hydraulics available in EPA SWMM. However, in v1.0 there are still a number of features that are missing or are not fully tested. These include (i) pollutant transport, (ii) culverts, (iii) multi-barrel conduits, (iv) power-function cross-section geometry, (v) custom closed conduit geometry (vi) force mains.

What's the future for SWMM5+?

The code structure is SWMM5+ was designed for expansion, Future versions will include improved mass-conservative pollutant transport that will allow users to go beyond the first-order kinetics of EPA SWMM. The finite-volume code also provides a flexible foundation for developing new multiphase solutions and/or air entrapment and pressurization. Linking with other 2D surface flooding codes should be straightforward because the "nodes" of a traditional EPA SWMM network are represented as finite-volume elements in SWMM5+. This approach should allow coupling of inlet flows and open-channel overbanking flows with similar coupling algorithms.

Licensing

SWMM5+ v1.0 is public domain software that was developed by the CIMM team of the University of Texas at Austin and the Urban Watersheds Research Institute under a Cooperative Agreement with US EPA (see acknowledgements).

Getting SWMM5+ for users

The installation package and documentation for the latest release of SWMM5+ is available at GitHub at https://github.com/CIMM-ORG/SWMM5plus/releases

Installing SWMM5+ for users

Users who do not intend to modify the SWMM5+ code should download the executable installation package (see GitHub link, above).

Installing SWMM5+ for developers

Those interested in contributing to SWMM5+ code development can set up their own development environment and link with the SWMM5+ github at the CIMM SWMM5+ GitHub site. To date, all development for SWMM5+ has used the Intel OneAPI compiler running on Linux. The hydraulic solver of SWMM5+ is written in Fortran 2008 with an API connecting it to the EPA SWMM C++ code. The SWMM5+ development team is interested in collaborating with anyone who would like to port the code to another compiler or modify to run natively on another operating system.

Does SWMM5+ use the EPA SWMM GUI?

The SWMM5+ code does not link directly with the EPA SWMM GUI. However, users are free to use the EPA GUI to create their *.inp file and define profiles for animation. SWMM5+ uses the *.inp file provided by the GUI. Output plots and animations similar to the GUI are accessed through Python scripts that the user can customize.

Will SWMM5+ run on my computer?

Yes --- if your computer is Linux or Windows. **Maybe ---** if your computer is a Mac. The SWMM5+ executable download/install has been tested under Ubuntu Linux and works under the Windows Subsystem for Linux under Windows 10. The SWMM5+ source code installation package for SWMM5+ uses the Intel OneAPI compiler and a "Docker" instance to setup a Linux virtual machine on your local computer. We have tested the Docker installation on both Windows and Linux computers. Although Docker is also compatible with OSX systems, Mac users cannot take advantage of SWMM5+ parallelization as there is an incompatibility between the Intel OneAPI compiler and code using the coarray standard of Fortran 2008.

The SWMM5+ solver

The core of SWMM5+ is a finite-volume solver for the dynamic (Saint-Venant) equations. The solution method is a Runge-Kutta explicit time-march with a new, smoothed, Preissmann Slot algorithm to handle surcharged flows. Unlike EPA SWMM, the new hydraulic solver does *not* require damping of nonlinear terms for supercritical flow, so flows in steep conduits are better represented. SWMM5+ directly calls the existing hydrology/LID solver from EPA SWMM, which is compiled as a linked library for coupling with SWMM5+.

Parallel operation with SWMM5+

The SWMM5+ code is inherently parallel, with an automatic algorithm for partitioning a network for any number of processors. The code is designed for good parallel speed-up as long as there are 1000+ computational elements per processor. The users do not need to know anything about parallelization other than the number of processors that they plan to use.

Using SWMM5+ input files

A key goal for the SWMM5+ development team is to maintain compatibility with the EPA SWMM input file and make the coupled code easy to use. SWMM5+ uses the standard EPA SWMM input file (*.inp) along with an optional *.json text file for the additional controls needed for the engine. For most purposes, a *.json file of less than 10 lines is sufficient. Details are provided in the SWMM5+ documentation at the GitHub site.

Using SWMM5+ output files

Unfortunately, SWMM5+ cannot use the standard EPA SWMM output format as it proved inefficient for writing from parallel processors. Instead, SWMM5+ uses the HDF5 standard that was designed for efficient parallel execution. The SWMM5+ installation package includes the HDF5 library and Python scripts that users can adapt for accessing data in the HDF5 output files. The installation package also includes a Python script for animating profiles similar to the EPA SWMM GUI. This script allows the user to customize flow movies in Python.

Sub-elements in SWMM5+

To get the best hydraulic solution, the SWMM5+ hydraulic engine subdivides the traditional EPA SWMM links into uniform sub-elements. The subdividing is done automatically by the code so the user does not have to change the SWMM *.inp file. To setup the finite-volume system, the user sets a desired sub-element length scale as the in the *.json input file. Note that SWMM5+ requires at least 3 sub-elements in every link of the network, so small links can require the user to set a very small nominal element link. If the user tries to set a length that is too large, SWMM5+ will exit on a USER CONFIGURATION ERROR. Because the computational time is a function of the number of elements, the SWMM5+ modeler should consider approaches to combining short elements with adjacent elements or representing as a hydraulic feature in the EPA SWMM *.inp file. For example, a short section of smaller-diameter pipe could be represented as an orifice as its head loss is caused by flow constriction rather than pipe roughness. Details are provided in the SWMM5+ User Guide available on the GitHub site.

What are the drawbacks of SWMM5+?

There are four drawbacks: (i) learning to install and use the new system, (ii) learning to access the new HDF5 output files in Python -- here we provide example Python scripts, (iii) longer computational times than EPA SWMM when parallel processors are not used, and (iv) users are likely to encounter bugs that the development team will need to address because SWMM5+ is a complicated code that is still under development.

Why is SWMM5+ slower than EPA SWMM (on a serial computer)?

The overarching goal of the SWMM5+ development was to create a code that can be efficiently parallelized on any size of computer. It was shown in Burger et al (2014) that EPA SWMM is inherently limited for computing large systems on large computers. The conventional SWMM hydraulics solver is the bottleneck that cannot get more than about 15x speedup no matter how many processors are available. We sought to "future proof" the next generation code for massively parallel simulation. Today's engineers typically have access of up to 64 computer cores, but is likely that in the near future this will be 245 or 512 cores. Furthermore, city-wide systems used for continuous simulation might require cloud supercomputers with 1000+ cores. In designing SWMM5+, we made algorithm choices with a focus on parallel computation efficiency, but some of these choices force the model to use a small time step. In particular, SWMM5+ v1.0 uses a junction algorithm that typically requires a time step near 0.1 s for standard manholes (i.e., 4 ft diameter). The time step is driven by the small volume of a typical manhole relative to the volumes connected pipes. This small time step means that SWMM5+ might be slower than EPA SWMM. A new junction algorithm that addresses this issue is under development.

What about bugs?

Version 1.0 of SWMM5+ is presently a research code -- it has been written by a professor and graduate students at the University of Texas at Austin. Our approach has been to include hard "stop" statements in the code when any condition appears to be leading to an instability. When a "CODE ERROR..." result is displayed at the command line the code has stopped because something has gone wrong. Users are encouraged to communicate with the SWMM5+ developers and provide us their input files so that we can find and fix the bug. Occasionally, the code might stop on a "USER CONFIGURATION ERROR..." which typically occurs when the user has requested a feature or capability that is not available in SWMM5+.

SWMM5+ mass conservation

In the SWMM5+ finite-volume solver, the flows in and out of each element must balance the change in volume to machine precision. Thus, each volume computation has the potential to introduce a mass conservation error at the 15th decimal place. Over the course of a simulation, machine precision errors can accumulate, but should always be smaller than the 8th decimal place. If SWMM5+ detects significant mass conservation the code will stop and issue a crash output. Generally, failure to conserve mass is an indication that the user has encountered an unknown bug in the SWMM5+ code.

What about oscillations?

Regular users of EPA SWMM will be familiar with high-frequency head and flow oscillations that sometimes show up in the output. In EPA SWMM, such oscillations are often accompanied by a mass conservation error and indicate that the dynamic flow algorithm is having difficulty converging the solution. SWMM5+ results might also show oscillations, but these are usually physically based. For example, a sudden increase in the flow rate can cause a bore, i.e., a frontal feature that propagates up or down a link. The sub-element discretization of SWMM5+ will simulate the bore propagation and, if the bore encounters a restriction at the next node, there may be a wave that back-propagates up the link. As another example: anyone who has tried to set up a hydraulic jump in a laboratory flume knows that it is difficult to stabilize a jump in one location: a jump tends to move up or down the flume. SWMM5+ represents hydraulic jumps on the face between computational sub-elements so a jump may adjust back and forth across the upstream and downstream faces of an element in seeking its balance point. Such features can show up as oscillations in SWMM5+. To date, our analysis shows that the SWMM5+ oscillations generally have a physical cause and detailed investigation provides insight into the hydraulic phenomena.

Compatibility with EPA SWMM

The SWMM5+ v1.0 code is coupled with EPA SWMM v5.1.0.13. Functionality of SWMM 5.2 is not presently available.

Fortran? I thought that language was dead...

The SWMM5+ hydraulic engine is written in Fortran 2008 and links through an application program interface (API) to the EPA SWMM code written in C++. The reasons we chose Fortran for the new solver are (i) the ease of writing parallel code with Fortran Coarrays, and (ii) the simplicity of writing array statements rather than "do-loops". The Fortran Coarray standard is part of Fortran 2008 that provides inherent parallelization using a Single-Program, Multiple Data (SPMD) approach. Each processor receives its own copy of the program and a section of the total network to operate upon. The communication between processors is limited to "shared" faces, which simplifies parallel coding. Developers familiar with writing code in Matlab or Python (where a = b + c operates over an array without a do-loop) will appreciate the similar array statements that have been part of Fortran since Fortran 90 but are still not available in C.

Can I put a Python wrapper around SWMM5+?

Yes. We have structure the code with global defaults to make it easy to access any variable and any setting parameter. Developers are encouraged to talk to the SWMM5+ team about data structures and wrapper functions.

Acknowledgements

The SWMM5+ code 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 code and documentation 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 the code or documentation.

References

Burger, G., R. Sitzenfrei, M. Kleidorfer, W. Rauch, "Parallel flow routing in SWMM 5," Environmental Modelling & Software, 53:27-34, 2014. http://dx.doi.org/10.1016/j.envsoft.2013.11.002