Water Quality: Installing and Using StormReactor

Project Links:

- https://www.pyswmm.org
- https://github.com/kLabUM/StormReactor

Date: December 27, 2022

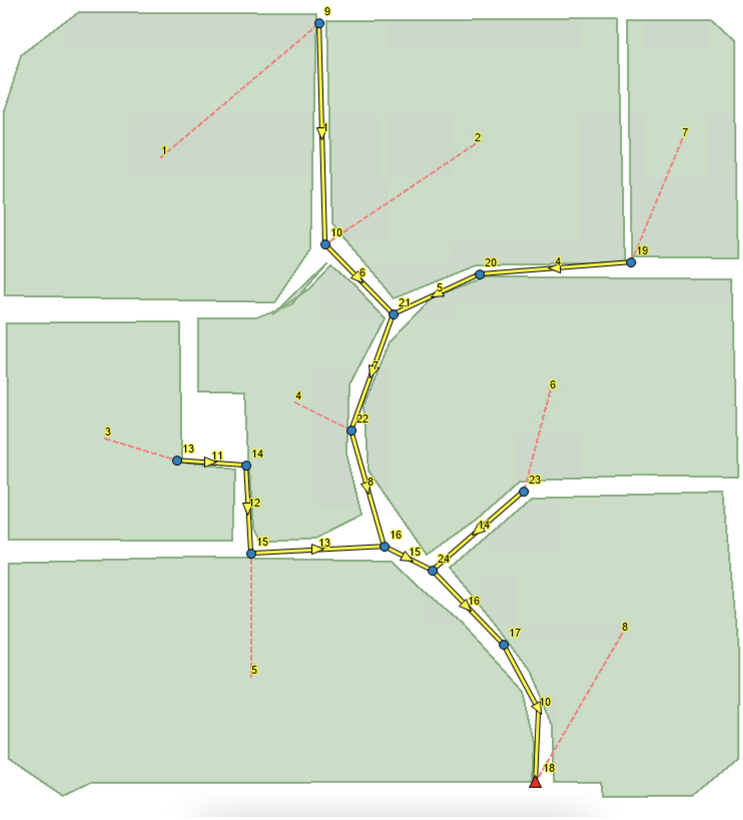
Author: Brooke Mason

Version: 1.0

Files: `WaterQuality.py`, `Example1_WQ.inp`

Background

This example walks the user through running a simulation with the additional library *StormReactor*. *StormReactor* was developed to expand the ability to model stormwater quality and water quality based real-time control (outside the scope of today's tutorial) in the EPA's Stormwater Management Model (SWMM) and pyswmm. We add treatment to a node and extract water quality-based data pertaining to the model/simulation using the `Example1_WQ` data set. The PNG file is simply to provide visual context of where we are reading information out of the model.



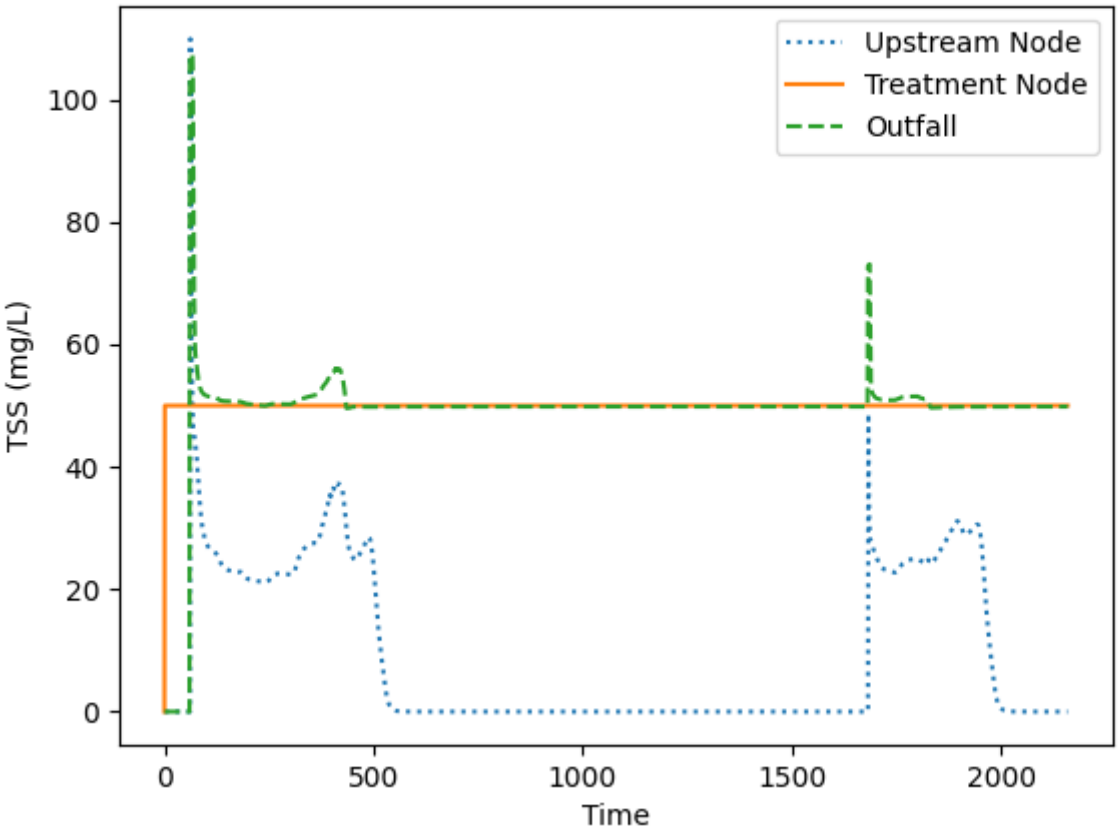
Code Example

This code builds off of the `Espresso` example:

```
'''
PySWMM and StormReactor Code Example A
Author: Brooke Mason
Version: 1
Date: December 27, 2022
'''
'''
PySWMM and StormReactor Code Example A
Author: Brooke Mason
Version: 1
Date: December 21, 2022
'''
# Import libraries
from pyswmm import Simulation, Nodes
from StormReactor import waterQuality
import matplotlib.pyplot as plt

# Define water quality configuration dictionary
config = {'17': {'type': 'node', 'pollutant': 'TSS', 'method':
'EventMeanConc', 'parameters': {'C': 50.0}}}

# Create lists to save TSS results
UpstreamNode_TSS = []
WQNode_TSS = []
OutfallNode_TSS = []

# Initalize SWMM simulation
with Simulation(r'Example1_WQ.inp') as sim:
    # Node information
    UpstreamNode = Nodes(sim)['24']
    WQNode = Nodes(sim)['17']
    OutfallNode = Nodes(sim)['18']

    # Initialize StormReactor
    WQ = waterQuality(sim, config)

    # Launch a simulation
    for step in sim:
        # Update water quality each simulation step
        WQ.updateWQState()
        # Get and save TSS concentrations
        UpstreamNode_TSS.append(UpstreamNode.pollut_quality['TSS'])
        WQNode_TSS.append(WQNode.pollut_quality['TSS'])
        OutfallNode_TSS.append(OutfallNode.pollut_quality['TSS'])

# Plot TSS concentrations
plt.plot(UpstreamNode_TSS, ':', label="Upstream Node")
plt.plot(WQNode_TSS, '-', label="WQ Node")
plt.plot(OutfallNode_TSS, '--',label="Outfall")
plt.xlabel("Time")
plt.ylabel("TSS (mg/L)")
plt.legend()
plt.show()
```

Let's decompose some of these things now. First, We have to import `StormReactor` just like we import `pyswmm`. Next we build our water quality configuration dictionary (e.g., `config`). This dictionary identifies in which nodes or links (e.g. node `17`) to enact water quality methods. For each one, we must state if it is a `node` or `link`, the name of the pollutant (e.g., `TSS`) from the input file, and the water quality method we want to use (e.g., `EventMeanConc`). Each water quality method has a required set of parameters. For `EventMeanConc`, we only need one parameter, `C`, which we set equal to `5.0`. More details on the other water quality methods and their required parameters are listed at the end of this document.

Output



Looking at the plot, we can see the upstream node's TSS concentration (blue dotted line) peaks at ~100 mg/L but is ~20-40 mg/L for most of the simulation. It drops to 0 mg/L during dry periods. The WQ node, or node 17, holds steady at 50 mg/L (orange solid line) throughout the simulation. We can see that this is actually an increase in concentration from the upstream node. *StormReactor* provides the ability to increase a pollutant's concentration. In traditional SWMM, you can only increase pollutants by adding them in the buildup/washoff functions or as additional inflows, both of which have limitations on how you can add them. But now, with *StormReactor*, you can increase or decrease pollutants at any timestep, providing more flexibility! For example, we could think of this example in a new light. The WQ node experiences constant erosion throughout the simulation, resulting in an increased TSS concentration throughout the simulation. To make this example more accurate, we could set this to add TSS only when flows exceed a certain threshold. This is beyond today's tutorial, but we just want to give you some ideas of what's possible with *StormReactor*! Now let's look at the outfall's TSS concentration (green dashed line). We can see its

maximum peak is ~100 mg/L as well, and then remains ~50 mg/L for the rest of the simulation. The two peaks come from subcatchment 8, which flows directly into the outfall, increasing the TSS concentration during the most intense rainfall periods.

Follow up

If you have run into problems, try posting your questions on Stack Overflow and tag it with `pyswmm`. The development team is very active on there and will for sure follow up!

Water Quality Methods

Here we provide a list of all of the SWMM water quality treatment methods we provide in *StormReactor* and their required parameters.

1. `EventMeanConc`:
    - Event Mean Concentration -- treatment results in a constant concentration
    - Treatment method parameters required:
        - `C` = constant treatment concentration for each pollutant (SI/US: mg/L)
2. `ConstantRemoval`
    - Constant Removal -- treatment results in a constant percent removal
    - Treatment method parameters required:
        - `R` = pollutant removal fraction (unitless)
3. `CoRemoval`:
    - Co-Removal Treatment -- removal of some pollutant is proportional to the removal of some other pollutant
    - Treatment method parameters required:
        - `R1` = pollutant removal fraction (unitless)
        - `R2` = pollutant removal fraction for other pollutant (unitless)
4. `ConcDependRemoval`:
    - Concentration-Dependent Removal -- when higher pollutant removal efficiencies occur with higher influent concentrations
    - Treatment method parameters required:
        - `R_l` = lower removal rate (unitless)
        - `BC` = boundary concentration that determines removal rate (SI/US: mg/L)
        - `R_u` = upper removal rate (unitless)
5. `NthOrderReaction`:
    - Nth Order Reaction Kinetics -- when treatment of a pollutant exhibits n-th order reaction kinetics where the instantaneous reaction rate is kC^n
    - Treatment method parameters required:
        - `k` = reaction rate constant (SI: m/hr, US: ft/hr)
        - `n` = reaction order (first order, second order, etc.) (unitless)

6. `kCModel`:
    - K-C Star Model -- the first-order model with background concentration made popular by Kadlec and Knight (1996) for long-term treatment performance of wetlands
    - Treatment method parameters required:
        - `k` = reaction rate constant (SI: m/hr, US: ft/hr)
        - `C_s` = constant residual concentration that always remains (SI/US: mg/L)
7. `GravitySettling`:
    - Gravity Settling -- during a quiescent period of time within a storage volume, a fraction of suspended particles will settle out
    - Treatment method parameters required:
        - `k` = reaction rate constant (SI: m/hr, US: ft/hr)
        - `C_s` = constant residual concentration that always remains (SI/US: mg/L)

StormReactor also includes a few additional water quality methods, like a continuously stirred tank reactor (CSTR) and a phosphorus model for bioretention systems. Users can also create their own water quality methods. Please see the StormReactor repository for more details.

References

Rossman, L. A. (2016). Storm Water Management Model Reference Manual: Volume III -- Water Quality (No. EPA-600/R-16/093). U.S. Environmental Protection Agency.

Mason, B. E., Mullapudi, A., & Kerkez, B. (2021). Stormreactor: An open-source python package for the Integrated Modeling of urban water quality and water balance. Environmental Modelling & Software, 145, 105175. https://doi.org/10.1016/j.envsoft.2021.105175