Espresso Machine: Controls Example

Project Link: https://www.pyswmm.org

Date: May 4, 2023

Author and Copyright © (2023): Bryant E. McDonnell

Version: 1.0

Files: `espresso_machine.py`, `DemoModel.inp`

Background

This example builds on top of the "Latte Art" example and shows you how to use `link.target_setting=x` feature **(and to develop ALL of your control logic in Pure Python!)** This example requires you to `pip install matplotlib` to make use of the plotter. If you followed through the Latte Art example, this step will be complete. Also, as a bonus, if you `pip install swmmio`, there are some nice profile plotting examples.

This feature is exciting to discuss since it pushes well beyond what you can do in standard USEPA-SWMM. It enables you to interact with the running simulation and control pumps, orifices, and weirs. You can now build controls from simple to incredibly sophisticated in PURE PYTHON! SWMM allows you to change the settings for weirs, orifices, and pumps. See the following table for setting ranges and how a setting is applied.

Link Type	Setting Range	Description
Weir	[0, 1]	A target setting of 0 means the asset is fully closed whereas the target setting of 1 means the asset is fully open.
Orifice	[0, 1]	Similar to Weir type, 0 means fully closed and 1 means fully open. If the user has enabled the "time to actuate" feature in SWMM, the target setting will be reached as a function of the travel time.
Pump	[0, inf)	The various pump types enable many options for moving flow. The best way to think about the pumping rate and target setting is separately. The pumping rate will convey flow as a function of pump type and (perhaps) head differential. The target setting is simply a scaler applied to the resolved flow. For Example, if you pump curve estimates 50LPS and you applied a target setting of 0.5, it will result in 25LPS. The target setting also allows you to go above 1. If a Type 4 pump is used with a pump curve of x[0, 1000] y[1, 1], for any determined pumping rate along the curve the model will always produce a value of 1LPS. In this case, using a target setting of 100 will produce 100LPS. It is important to note that regardless of target setting, if 100LPS is not coming into the upstream node, the pumping rate will be limited to the incoming flow rate.

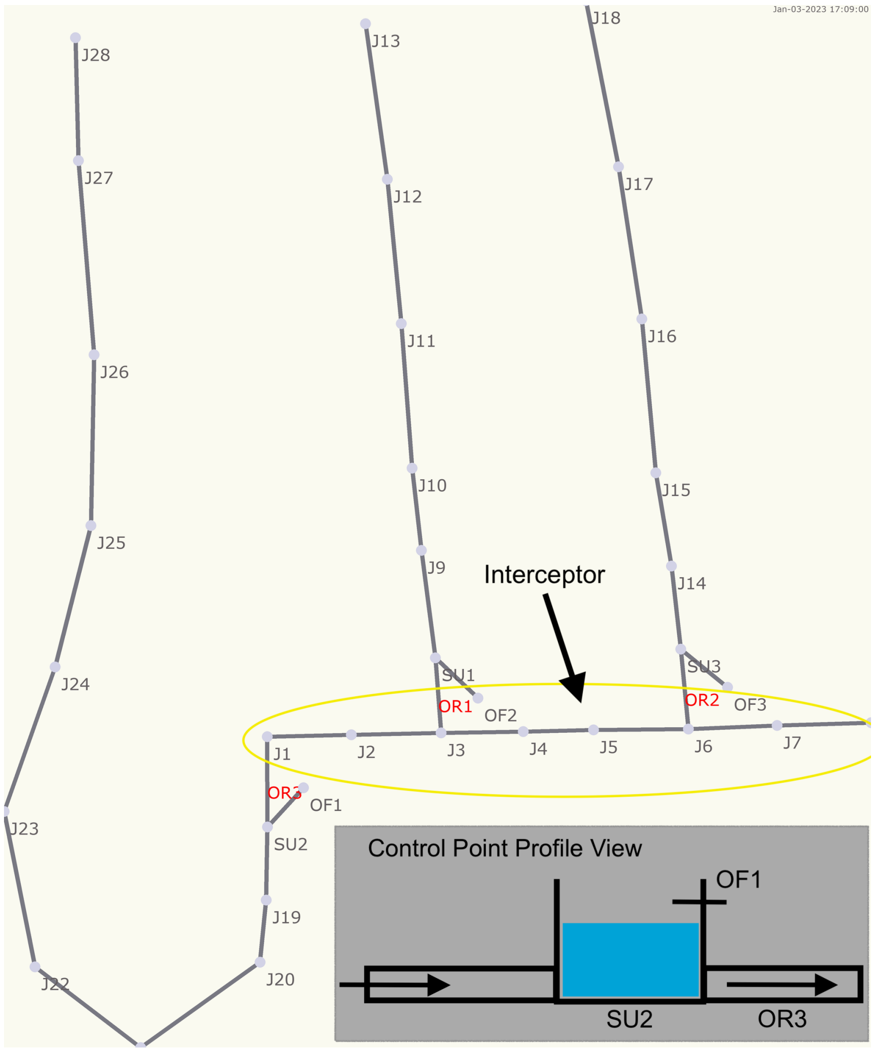
PITFALL #1: If you are planning to control a link using PySWMM you need to remember to remove any control rules inside the `[RULES]` section of your model. Some very interesting things can happen when SWMM and PySWMM compete for control - especially when using the `step_advance` feature. SWMM controls alway take higher priority over pyswmm. Furthermore, when using 1step_advance1, PySWMM will not be intervening on every routing step (as SWMM will).

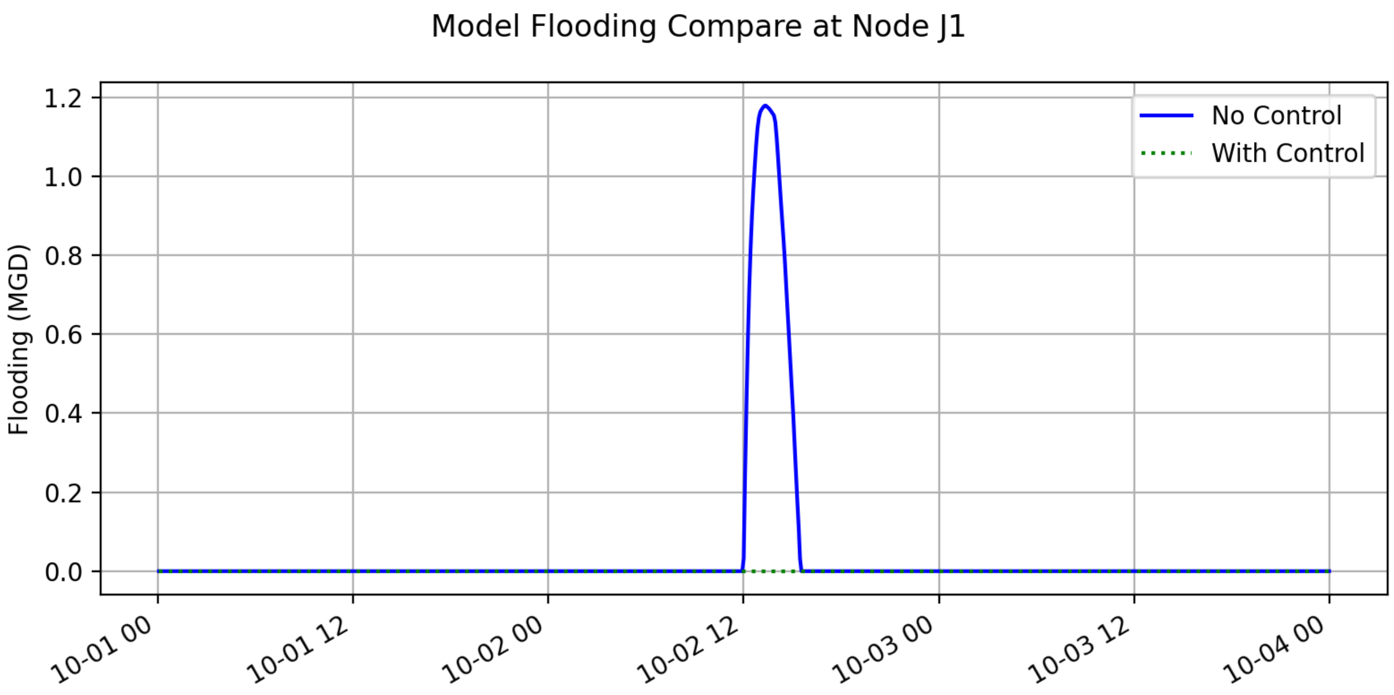
PITFALL #2: It you are migrating control logic for pumping behavior to PySWMM it is best to set the pump status to always ON and remove the hysteresis for the pump (On/Off depths) and set them to 0. You can rebuild all the control in PySWMM while making it easy on yourself to have once place to look.

Code Example

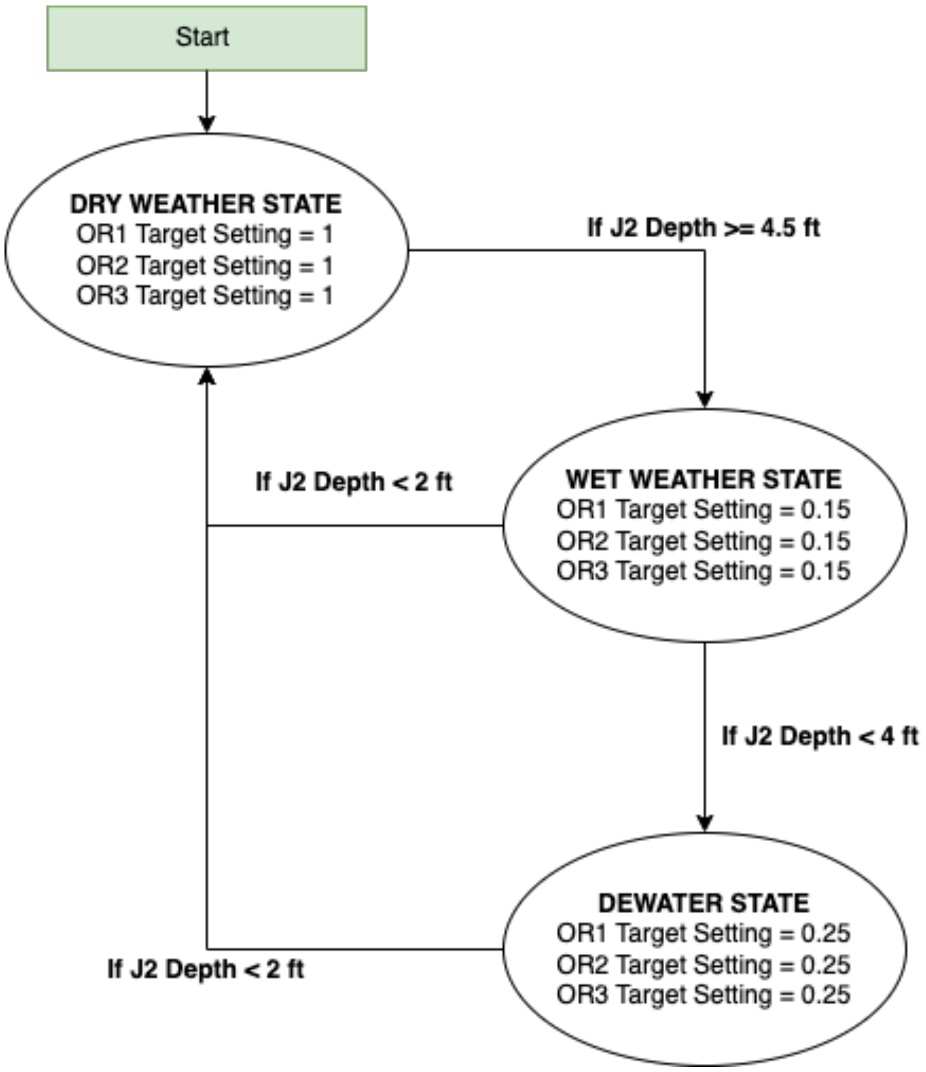
The following figure is used to show the node ID names in the network for quick reference. This example demonstrates how to write basic control in pure Python to control Orifice `OR1`, `OR2`, and `OR3` in the model. The mode itself represents a combined sewer overflow (CSO) system where nodes `J1`-`J8` make up and interceptor system. We also have prescribed CSO points at `OF1`, `OF2`, and `OF3`. Also, to make the problem more interesting, we also have large storage tanks upstream of each CSO points (`SU1`, `SU2`, and `SU3`). For each tank, if you surcharge it enough it will overflow the CSO.

The problem begins with a series of large storm events (to simplify the model setup, we are using time series inflows around the model to emulate inflow to our network.) In our no-control simulation we see that all CSO tributary areas have unfettered flow to the interceptor system. The excess inflow is exceeds the conveyance capacity of the interceptor has (See Following Figure).


Jan-03-2023 17:09:00
J28
J27
J26
J25
J24
J23
J22
J13
J12
J11
J10
J9
J18
J17
J16
J15
J14
Interceptor
SU1
OR1
OF2
SU3
OR2
OF3
J1
J2
J3
J4
J5
J6
J7
OR3
OF1
SU2
J19
J20
Control Point Profile View
OF1
SU2
OR3


Model Flooding Compare at Node J1
Flooding (MGD)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
10-01 00
10-01 12
10-02 00
10-02 12
10-03 00
10-03 12
10-04 00
No Control
With Control

Through the use of some simple (or sophisticated control) we can start using the 3 storage tanks and throttling the underflow lines. The following control logic was applied to the model. This is a very trivial control example that uses a State Machine.



In code, this looks as follows (the plotting code has been removed to keep it simple). We are taking advantage of variables very easily such as `in_wet_weather` and using `if` and `elif` conditionals to move around the different states.

```
'''
PySWMM Espresso Machine (Abridged)
Author: Bryant McDonnell
Version: 1
Date: May 4, 2023
'''
from pyswmm import Simulation, Nodes, Links

with Simulation(r'DemoModel.inp',
                'DemoModel_wControl.rpt',
                'DemoModel_wControl.out') as sim:
```

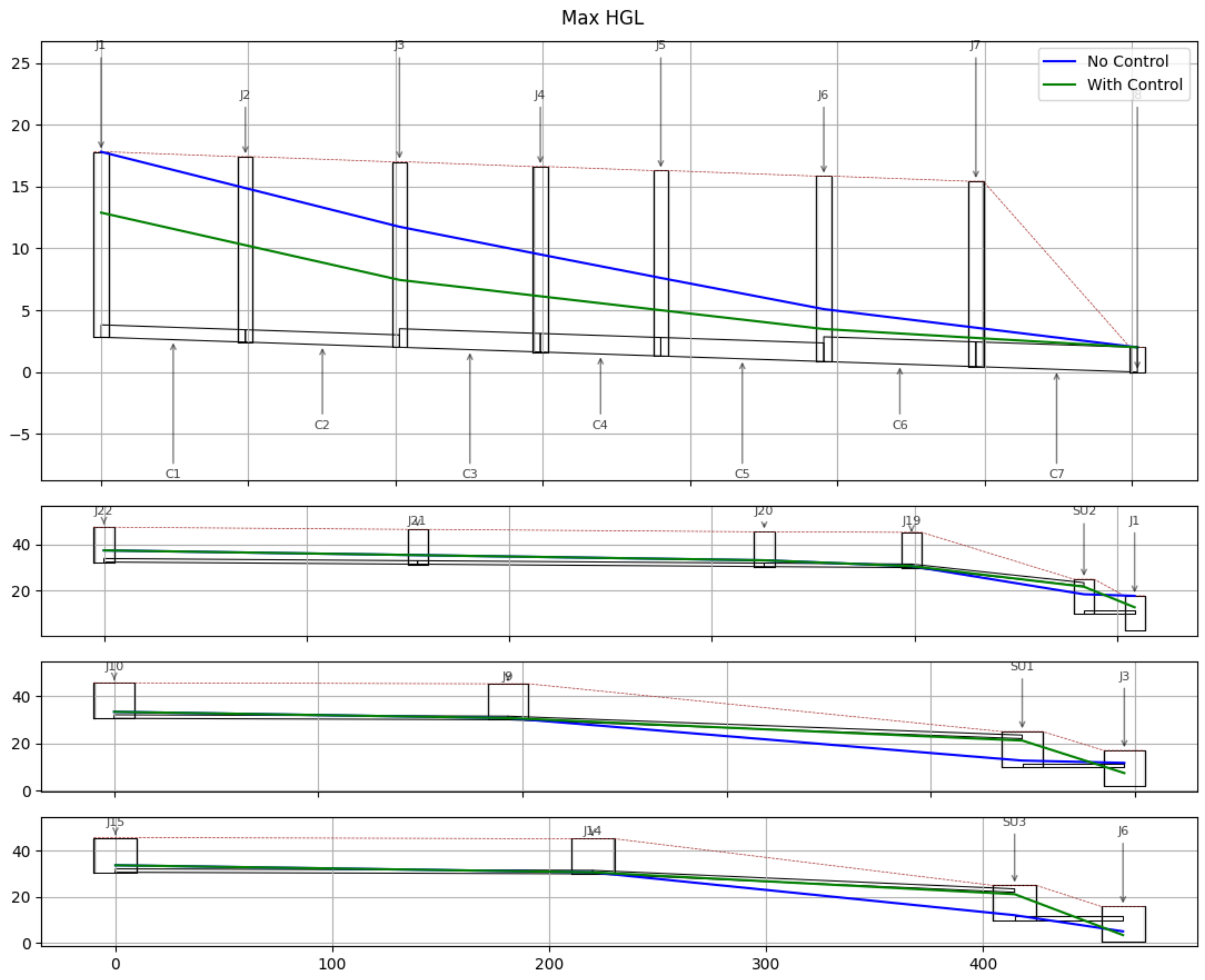
```
    # Instantiating the Orifices to Control
    OR1 = Links(sim)["OR1"]
    OR2 = Links(sim)["OR2"]
    OR3 = Links(sim)["OR3"]
    # Interceptor Nodes to Observe
    J2 = Nodes(sim)["J2"]

    sim.step_advance(300)
    # Launch a simulation!
    in_wet_weather = False
    for ind, step in enumerate(sim):
        if J2.depth > 4.5 and in_wet_weather == False:
            OR1.target_setting = 0.15
            OR2.target_setting = 0.15
            OR3.target_setting = 0.15
            in_wet_weather = True
        elif J2.depth <= 4 and in_wet_weather == True:
            OR1.target_setting = 0.25
            OR2.target_setting = 0.25
            OR3.target_setting = 0.25
        elif J2.depth < 2:
            OR1.target_setting = 1
            OR2.target_setting = 1
            OR3.target_setting = 1
            in_wet_weather = False
```

Output

Running the full **Espresso Machine** example will give you the following table. This shows the before and after condition of the total flooding.

Volume Type	No Control (MG)	W Control (MG)
external_inflow	19.413	19.413
final_storage	0.009	0.055
flooding	**0.119**	0.000
initial_storage	0.000	0.000
outflow	19.295	19.361
CSO Volume (OF1+OF2+OF3)	0.000	0.000
Downstream Volume (J8)	19.295	19.361

Finally, if you have `swmmio` installed, you should see the following profile plot as well. This plot has been configured to fetch the maximum node depths from throughout the model.



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!