Café con Leche: node.generated_inflow()

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

Date: January 3, 2023

Author: Bryant McDonnell

Version: 1.0

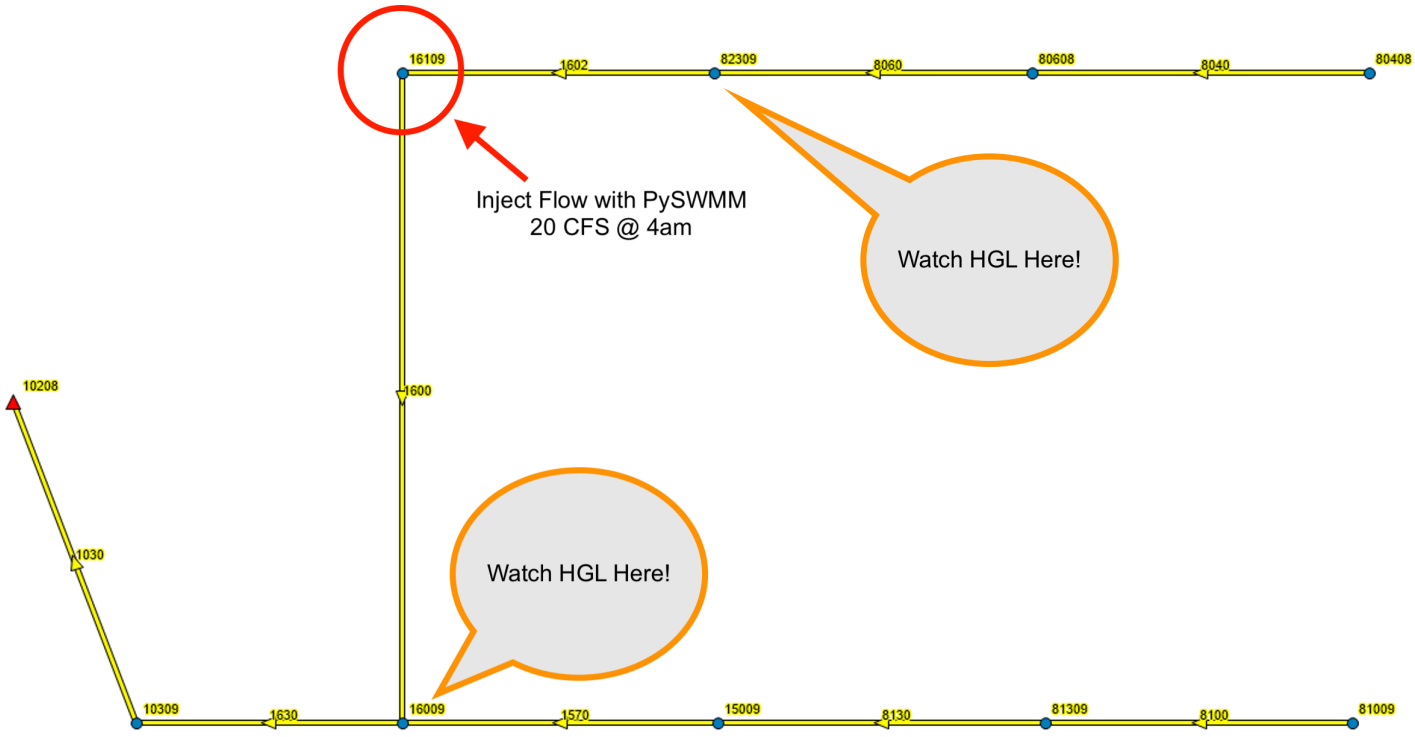
Files: `cafe_con_leche.py`, `Example2.inp`

Background

This example builds on top of the "Latte Art" example and shows the user how to use `Node.generated_inflow()` feature! 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.

> Some background on the use of `node.generated_inflow()`. This feature is very powerful. It conserves mass during your simulation as well (see Discussion). You can inject a time-varying flow rate of your choice into a node during a simulation. When you choose a flow rate, generated_inflow will hold that flow constant until you change it. In the following example, we call it continuously, however, that is not technically required.

The following figure is used to show the node ID names in the network for quick reference.



In this example, let's pretend that we have diverted some stormwater during a wet weather peak flow event to reduce our downstream flow rates. After the storm subsides, and our flows are coming back down, we want to dewater our stormwater storage systems back into the hydraulic network. This example is going to take advantage of the PySWMM feature `node.generated_inflow()` and show you how to inject flow. Furthermore, we will plot the inflow behaviors to make the demonstration clear.

Let's start by looking at Node "16109". In this node we can look at two type of flow coming in "Lateral Inflow" and "Total Inflow." For those of you who have less familiarity with SWMM's routing, it's important to know the difference. Lateral inflow represents all flow being loaded directly to a node (such as from subcatchments, dwf, average flow, groundwater, unit hydrographs, and etc.) Total Inflow, on the other hand, is all of the lateral inflow PLUS the net flow coming in from the connected links.

To make it simple, let's look at the simulation start and end times. The simulation starts on Jan 1, 2002 at 12am and ends on Jan 1, 2002 at 8am. Let's plan to inject 20 CFS into the "16109" on Jan 1, 2002 at 4am until the end of the simulation (so for 4 hours). What we should see is the lateral inflow increase by 20 CFS at the next routing time step. In addition, it would be wise for us to pay attention to the HGL both upstream and downstream just so we do not cause any undue harm. So let's also add those to or time series plots.

Code Example

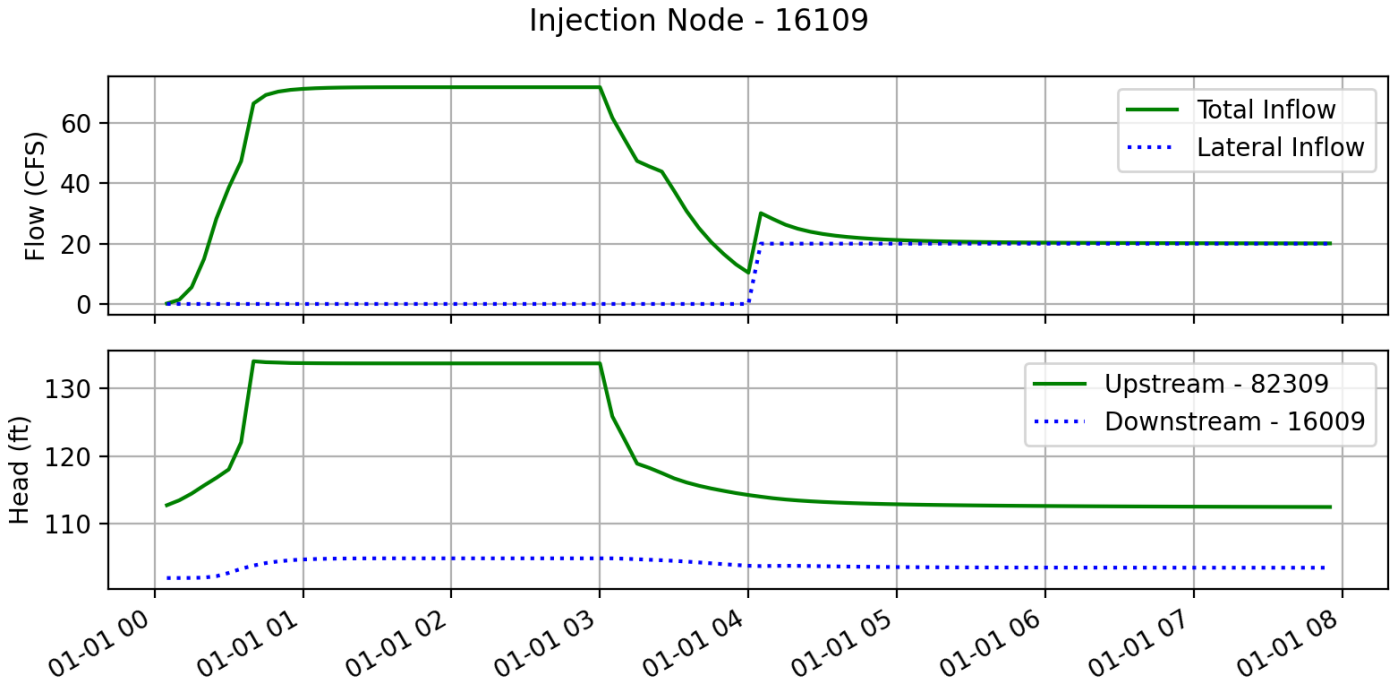
```
'''
PySWMM Cafe Con Leche
Author: Bryant McDonnell
Version: 1
Date: Jan 3, 2023
'''
import datetime
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from pyswmm import Simulation, Nodes, Links, Output

with Simulation(r'Example2.inp') as sim:
    node_a = Nodes(sim)["16109"] # Injection Node
    node_b = Nodes(sim)["82309"] # Upstream Node
    node_c = Nodes(sim)["16009"] # Downstream Node

    # Initialize Lists for storing data
    time_stamps = []
    node_total_flow = []
    node_lateral_inflow = []
    us_head = []
    ds_head = []

    sim.step_advance(300)
    # Launch a simulation!
    for ind, step in enumerate(sim):
```

```
        if sim.current_time >= datetime.datetime(2002, 1, 1, 4, 0, 0):
            node_a.generated_inflow(20) # CFS into node
        time_stamps.append(sim.current_time)
        node_total_flow.append(node_a.total_inflow)
        node_lateral_inflow.append(node_a.lateral_inflow)
        us_head.append(node_b.head)
        ds_head.append(node_c.head)

fig = plt.figure(figsize=(8,4), dpi=200) #Inches Width, Height
fig.suptitle("Injection Node - 16109")
# Plot from the results compiled during simulation time
axis_1 = fig.add_subplot(2,1,1)
axis_1.plot(time_stamps, node_total_flow, '-g', label="Total Inflow")
axis_1.plot(time_stamps, node_lateral_inflow, ':b', label="Lateral Inflow")
axis_1.set_ylabel("Flow (CFS)")
#axis_1.get_xticklabels().set_visible(False) # turns off the labels
axis_1.grid("xy")
axis_1.legend()
# second axis
axis_2 = fig.add_subplot(2,1,2, sharex = axis_1)
axis_2.plot(time_stamps, us_head, '-g', label="Upstream - 82309")
axis_2.plot(time_stamps, ds_head, ':b', label="Downstream - 16009")
axis_2.set_ylabel("Head (ft)")
#axis_1.get_xticklabels().set_visible(False) # turns off the labels
axis_2.grid("xy")
axis_2.legend()

fig.autofmt_xdate()
plt.tight_layout()
plt.savefig("TEST.PNG")
plt.show()
```

To run this the you just have to hit `F5` if using `Idle`.

Output

If you open the `TEST.PNG` example, it can be seen very clearly that the lateral inflow at node "16109" jumps up by 20 CFS. In turn, the "Total Inflow" to that node also jumps by the 20 CFS. Both the upstream and downstream HGLs remain almost completely unchanged.



Discussion on Mass Balance Checks

For more details on the accounting for mass, let's dive into the Report (RPT) file.

First, let's check to see if the amount of water we injected is showing up correctly in the RPT. We should look at the Node Inflow Summary. (For viewing ease, one column has been removed.)

```
  ********************
  Node Inflow Summary
  ********************
  -------------------------------------------------------------------------------------
--
                                Maximum  Maximum     Lateral     Total        Flow
                                Lateral    Total      Inflow     Inflow     Balance
                                 Inflow   Inflow      Volume     Volume       Error
  Node                  Type         CFS      CFS    10^6 gal   10^6 gal     Percent
  -------------------------------------------------------------------------------------
--
  80408                 JUNCTION   45.00    45.00      3.64       3.64      -0.357
  80608                 JUNCTION    0.00    46.16      0          3.64       1.553
  81009                 JUNCTION   50.00    50.00      4.04       4.04      -0.510
  81309                 JUNCTION    0.00    53.67      0          4.06       0.307
  82309                 JUNCTION   40.00    83.80      3.23       5.95      -1.057
  10309                 JUNCTION    0.00   122.23      0          12         0.914
  15009                 JUNCTION    0.00    50.91      0          4.05       0.058
  16009                 JUNCTION    0.00   122.26      0          12.1       1.009
  16109                 JUNCTION   20.00    71.90      2.15       8.17       1.485
```

```
10208              OUTFALL       0.00    122.20      0           11.9       0.000
```

Looking at the node that we chose to inject flow to "16109", we see that it has a total of 2.15MG classified as "Lateral Inflow". We can confirm that that is the mass we added by integrating 20CFS*4hours

20 ft^3/sec * 7.481 gal/ft^3 * 3600 sec/hr * 4hr = **2.152MG**

Going a step further, we can look at the total flow routing continuity.

```
*************************        Volume        Volume
Flow Routing Continuity        acre-feet      10^6 gal     "normal classification"
*************************     ---------     ---------      -----
Dry Weather Inflow .......         0.000         0.000       V_in
Wet Weather Inflow .......         0.000         0.000       V_in
Groundwater Inflow .......         0.000         0.000       V_in
RDII Inflow ..............         0.000         0.000       V_in
External Inflow ..........        40.077        13.060       V_in
-------------------------------------------------------------------------
External Outflow .........        36.409        11.864       V_out
Flooding Loss ............         2.680         0.873       V_out
Evaporation Loss .........         0.000         0.000       V_out
Exfiltration Loss ........         0.000         0.000       V_out
-------------------------------------------------------------------------
Initial Stored Volume ....         0.000         0.000       V_in
Final Stored Volume ......         1.291         0.421       V_out
-------------------------------------------------------------------------
Continuity Error (%) .....        -0.756
```

Our newly injected mass is accumulated in the "External Inflow" section of the Flow Routing Continuity Table. The continuity error is a manifestation of the SWMM solver not converging (nor resolving HGLs) among the set of a node and its upstream and downstream links. Continuity error less than <1% is generally "accepted" in the industry. Even higher values can be accepted but it's up to the modeler to report their certainty.

In its simplest form, assuming all positive flows, mass balancing error is calculated as follows (this does not consider negative value corrections:

$$V_{Total\_inflow} = V_{initial\_stored} + V_{ww\_Inflow} + V_{ii\_inflow} + V_{dwf} + V_{gw\_inflow} + V_{external\_inflow}$$

$$V_{Total\_outflow} = V_{final\_storage} + V_{flooding} + V_{evap\_loss} + V_{seepage\_loss} + V_{reacted} + V_{outflow}$$

Again, there are some rules to dictate how to handle negative values. For more information, see SWMM's codebase `massbal.c:massbal_getFlowError()`. SWMM calculates the percent difference between the total inflow and total outflow.

From the C code:

```
// --- find percent difference between total inflow and outflow
```

```
FlowTotals.pctError = 0.0;
if ( fabs(totalInflow - totalOutflow) < 1.0 )
{
  FlowTotals.pctError = TINY;
}
else if ( fabs(totalInflow) > 0.0 )
{
  FlowTotals.pctError = 100.0 * (1.0 - totalOutflow / totalInflow);
}
else if ( fabs(totalOutflow) > 0.0 )
{
  FlowTotals.pctError = 100.0 * (totalInflow / totalOutflow - 1.0);
}
FlowError = FlowTotals.pctError;
```

Finally, in our case, we calculate the following:

$$Continuity\_Error = 100 * (V_{Total\_inflow}/V_{Total\_outflow} - 1.0) \approx -0.75$$

Looks good to me!!!!

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!