Integrating Transient Simulation in water Networks (TSNet) model into PCSWMM for transient analysis in water distribution systems

2021-02-25

Hailiang Shen, Ph.D., P.Eng.
TSNet

• Open-source Python site-package (>=Python 3.5)
• Initial conditions – EPANET model output at a reporting step
• Governing equations – mass and momentum conservation

\[
\frac{\partial H}{\partial t} + \frac{c^2}{g} \frac{\partial V}{\partial x} - V \sin \alpha = 0 \\
\frac{1}{g} \frac{\partial V}{\partial t} + \frac{\partial H}{\partial x} + h_f = 0
\]

H = head
V = flow velocity
t = time
c = wave speed
\( \alpha \) = pipe slope
\( h_f \) = head loss per unit length due to friction
TSNet

• Governing equations solved using Method of Characteristics (MOC)

\[ C_+ : \frac{dV}{dt} + \frac{g}{c} \frac{dH}{dt} + gh_f - \frac{g}{c} V \sin \alpha = 0 \] along \[ \frac{dx}{dt} = +c \]

\[ C_- : \frac{dV}{dt} - \frac{g}{c} \frac{dH}{dt} + gh_f - \frac{g}{c} V \sin \alpha = 0 \] along \[ \frac{dx}{dt} = -c \]

TSNet Usage

Python script to use TSNet

```python
# Open an example network and create a transient model
tm = tsnet.network.TransientModel('networks/Tnet1.inp')

# Set wave speed
tm.set_wavspeed(1200.)  # m/s

# Set time options
tf = 20  # simulation period [s]
tm.set_time(tf)

# Set valve closure
vc = 5  # valve closure start time [s]
tc = 1  # valve closure period [s]
se = 0  # end open percentage [s]
mi = 2  # closure constant [dimensionless]
tm.valve_closure('VALVE',[tc,vc,se,mi])

# Initialize steady state simulation
T0=0
tm = tsnet.simulation.Initializer(tm,T0)

# Transient simulation
tm = tsnet.simulation.TranSim(tm)

# report results
node = ['N2', 'N3']
tm.plot_node_head(node)
```

Plot created by Python matplotlib

TSNet - Limitations

• TSNet only works with metric unit input (e.g., wave speed in m/s) – however, EPANET data is interacted using the WNTR Python library that can work with both the metric and imperial unit systems

• TSNet inputs (e.g., transient source, time control) are saved in the script file and has no standard format for entity specific inputs (e.g., pipe specific wave speed)

• Plot result time series with Python plotting libraries (e.g., matplotlib)

• Lack of spatial visualization (hotspots of impact)
PCSWMM Support for Python Scripting

• Embed IronPython
  • .NET implementation of the Python programming language
  • Support IronPython standard libraries (e.g., ctypes, json, csv, os, sys)
  • Support .NET framework (e.g., Thread, DateTime, or 3rd-party libs Accord.NET)

• 100+ PCSWMM classes to interact with the PCSWMM data model for EPANET and SWMM modeling

• 60+ ready-to-use script examples and tools

• Built-in script editor – debugging, file management, syntax highlight, autocomplete/intellisense
<table>
<thead>
<tr>
<th>TSNet</th>
<th>Challenge/Limitation</th>
<th>PCSWMM Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python 3 environment</td>
<td>PCSWMM embeds IronPython 2.7.8 that cannot directly call into Python 3 functions</td>
<td>Use subprocess.Popen to start Python 3 and listens to Python 3 print messages</td>
</tr>
<tr>
<td>Application programming interface</td>
<td>Requires editing the script when working with different model inputs</td>
<td>Create a script including the settings for all transient sources (e.g., pump shut down, valve closure) in loops. Only the applicable ones (e.g., only pump shut down) will be set</td>
</tr>
<tr>
<td>Transient model input units</td>
<td>Metric only</td>
<td>Accept transient model inputs in EPANET’s unit system (e.g., wave speed either in ft/s or m/s) and convert them to metric for TSNet run</td>
</tr>
<tr>
<td>Output data is plotted or saved to pickle file</td>
<td>Not easy to visualize multiple time series at multiple junctions/pipes</td>
<td>Save all TSNet output time series to PCSWMM TSB binary format files (pipe results to Pipes.tsb, junction to Junctions.tsb) Post-process the max, min, and range of surge pressures and save them to the Junctions layer for spatial view</td>
</tr>
</tbody>
</table>
TSNet Model Specific Inputs

1. Global wave speed (overwritten by pipe entity wave speed)
2. Global time step (if zero TSNet uses the maximum allowable time step)
3. Global initial time (the EPANET timestamp results as transient initial conditions)
4. Global simulation duration
5. Global demand type for initial conditions (demand driven, pressure driven)
6. Global friction model (steady or unsteady)

Saved into a TSNet input text file (.tra)
### TSNet Entity Specific Inputs

<table>
<thead>
<tr>
<th>Layer</th>
<th>Category</th>
<th>User-Defined Attribute</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valves</td>
<td>Transient – Valve Operation</td>
<td>Surge Source Type</td>
<td>None/Close/Open</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Start Time (s)</td>
<td>Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Periods (s)</td>
<td>Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>End Open Percent</td>
<td>Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Constant</td>
<td>Number</td>
</tr>
<tr>
<td>Pipes</td>
<td>Transient</td>
<td>Wave Speed (ft/s or m/s)</td>
<td>Number</td>
</tr>
<tr>
<td></td>
<td>Transient – Demand Pulse</td>
<td>Demand Pulse</td>
<td>YES/NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Total Period (s)</td>
<td>Number</td>
</tr>
<tr>
<td></td>
<td>Transient – Junction Burst</td>
<td>Junction Burst</td>
<td>YES/NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Start Time (s)</td>
<td>Number</td>
</tr>
<tr>
<td></td>
<td>Transient – Junction Leak</td>
<td>Junction Leak</td>
<td>YES/NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Emitter Coeff</td>
<td>Number</td>
</tr>
<tr>
<td></td>
<td>Transient – Surge Tank</td>
<td>Surge Tank Type</td>
<td>open/closed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section Area (ft² or m²)</td>
<td>Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Height (ft or m)</td>
<td>Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Initial Height (ft or m)</td>
<td>Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Surge Tank Type</td>
<td>open/closed</td>
</tr>
</tbody>
</table>
TSNet Model Run

• Create a user-interface using a scripting class PyForm
• Display run progress and estimated remaining time
• Stop model run prematurely (in the case of wrong model parameter or too long wait time)
Case Study

- TSNet example 2 – updated EPANET example 3
- Transient source
  - Source location: PUMP2 shutdown
  - Shutdown period: 0.1 seconds
  - End open percent: 0
  - Constant: linear
- TSNet model inputs
  - Wave speed: 3000 ft/s
  - Simulation duration: 60 seconds
  - Time step: 0.01 seconds
  - Demand type: DD
  - Initial time: 0
  - Friction model: Steady
New Project

Welcome to PC_SWMM 2021 Professional 2D. Please choose the type of project you would like to create:

- **SWMM Project**: Create an EPA SWMM project for stormwater, wastewater or watershed modeling.
- **EPANET2 Project**: Create an EPANET2 project for water distribution modeling.
- **RAP Project**: Create a Rainfall Acquisition and Processing (RAP) project for downloading, calibrating, analyzing and preparing rainfall for use in SWMM5 and other purposes.
- **Time Series Project**: Create a time series project for managing, editing, and/or analyzing any type of time series.
Potential Applications

• Classrooms – teach pipeline hydraulic transient principles
• Researchers – validate algorithm updates
• Practicing engineers – analyze small to medium size pressurized systems (a couple of hundreds of pipes)
Source Code

- TSNet - https://github.com/glorialulu/TSNet
- PCSWMM integration scripts – support articles (draft to be published) or contact me (hailiang@chiwater.com) to get a copy
Computational Hydraulics International