Advances in Engineering Software 149 (2020) 102884

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Transient simulations in water distribution networks: TSNet python package

Lu Xing, Lina Sela*

Department of Civil, Architectural and Environmental Engineering, University of Texas at Austin, Texas 78712, US

ARTICLE INFO

Keywords:
Transient simulation
Python
Open source software
Water distribution networks

ABSTRACT

Modeling transient flow conditions in water distribution networks (WDNs) has shown increasing usability for various applications, including burst and leak detection, sensor placement, model calibration, and risk assessment. To facilitate the integration of transient modeling in these simulation-based applications, this work contributes a new open source Python package for Transient Simulations in water Networks (TSNet). TSNet adopts the Method of Characteristics (MOC) for solving the system of partial differential equations governing the unsteady hydraulics. It allows users to simulate various conditions including operational changes in valves and pumps, as well as background leaks and pipe bursts. In this paper, the TSNet modeling framework is presented and a case study is used to showcase its capabilities of simulating WDN responses to valve closure, pump shut-off, leaks, and bursts with and without a surge protection tank. Results show that valve closure, pump shut-off, and pipe burst can generate significant transients in the WDN, while background leaks can help damp the transients to some extent.

Software Availability

TSNet source codes are available from GitHub repository at https://github.com/glorialulu/TSNet.

TSNet package documentation is available from Read the Docs at https://tsnet.readthedocs.io.

1. Introduction

Hydraulic transients in water distribution networks (WDNs), induced by rapid changes such as pipe bursts, valve and pump operations, can disturb the steady-state flow conditions by introducing fast flow changes, imposing abrupt internal pressure force onto the pipeline systems, and generating pressure waves propagating rapidly ($> 1000 m/s$) through the piped network [29,60]. The propagation of the pressure waves is mediated by the complex network topology and the interactions of the pressure waves propagating through the fluid with the conduit are reflected in the changes (e.g. attenuation and phase shift) of the pressure wave. These disturbances have been identified as one of the major contributing factors in the many pipe deterioration and catastrophic failures in WDNs [48], thereby disrupting water supply, wasting a significant amount of treated water, and creating unexpected opportunities for contamination intrusion [19,28]. Conventionally, transient simulation, as a prominent approach for modeling and predicting the propagation of transient waves, has been an essential requirement in the design process for ensuring the hydraulic integrity of WDNs.

In addition to the applications in WDN design, the transient-based approach has also gained its popularity in fault detection [1,5,23,63], condition assessment [53,55,65], model calibration [31,50], pressure management [13–15,27,30,47], and uncertainty quantification [1,18,34,44]. For these purposes, transient-based models are commonly believed to be complementary to other techniques because a significant amount of information about the WDN can be revealed during a very short period time as the transient wave travels quickly through the pipe [63]. This information can then improve the detection accuracy of pipe defects, reduce the ill-posedness in calibration problems, and maximize the information gain in assessing pipeline conditions.

Various transient-based methods have been developed using different techniques, which can be categorized into (a) model-driven methods [4,8,11,23,31,40,58,59], and (b) data-driven methods [5,20,39,43,51,62]. Although data-driven techniques gained increasing popularity over the past decade due to the rapid development of data logging and data mining technologies, it is practically impossible to collect data from every location in the WDNs. Thus, reliable transient models are still integral to simulate and extrapolate the flow conditions in the entire system using the data collected from the limited monitored locations. However, the previous model-driven applications are largely restricted to pipe segments, such as reservoir-pipeline-valve (RPV) systems [4,23,59], and simple networks [8,11,40]. The extension of

* corresponding author.
E-mail addresses: xinglu@utexas.edu (L. Xing), linasela@utexas.edu (L. Sela).

https://doi.org/10.1016/j.advengsoft.2020.102884
Received 28 September 2019; Received in revised form 19 June 2020; Accepted 27 July 2020
Available online 01 September 2020

these techniques to complex pipe networks has been substantially impeded by the lack of open-access software incorporating the capabilities of easy interaction [63]. Ultimately, the extensive transient-based applications require efficient and accurate hydraulic transient simulation tools as indispensable prerequisites.

Acknowledgedly, a number of commercial software for transient simulation in WDNs is available, such as Hammer [25], Pipe 2018 [37], InfoSurge [26], and TransAM [42]; however, the use of these software for research purposes is restricted. Two major restrictions hindering the usability of commercial software packages for research are : (1) the software is packaged as a black-box, and the source codes are not accessible, thus prohibiting any changes, including modification of existing and implementation of new elements, in the source codes; and (2) in addition to the high cost, the commercial software is designed to perform single transient simulations and do not offer the capabilities to perform multiple transient simulations automatically. Thus, users are required to modify the transient conditions using the graphical user interface (GUI), perform the simulation, and manually record the hydraulic responses in the various conditions, which significantly complicates the research process. Although [35] developed an open source software, the MATLAB codes were only applicable to RPV systems, which substantially limits its practicality. Hence, a clear gap exists between currently available transient simulation capabilities and the ever-growing research requirements. To bridge this gap, the authors considered it imperative to develop an open source package rendering easiness for interaction, modification, and extension of transient modeling and simulation.

This paper contributes a comprehensive software framework and a Python package developed under the MIT license for Transient Simulation in water Networks (TSNet). The motivation of this work is two-fold: (1) provide users with an open source and freely available Python code and package for simulating transients in WDNs that can be integrated with other case specific applications, e.g., sensor placement and event detection, and (2) encourage users and developers to further develop and extend the transient model. With these motivations in mind, TSNet was specifically designed such that users familiar with EPANET [49] and/or the Water Network Tool for Resilience (WNTR) Python package [36] can use TSNet with minimum efforts. The main capabilities of TSNet include: (1) simulating transient system responses to operational changes in valves and pumps as well as background leakage and pipe bursts, (2) simulating open and closed surge tanks for controlling transient response, (3) simulating steady, quasi-steady, and unsteady friction models, (4) simulating instantaneous demand at nodes using demand-pulse model, and (5) allowing the user to select the computational time step and control numerical accuracy and computational complexity. Section 2 describes the main components of the TSnet framework, and Section 3 uses an example application to demonstrate the modeling capabilities and user interactions with TSNet.

2. Modeling framework

TSNet is an open source Python package designed to perform transient simulations in WDNs. The primary components and capacities in TSNet include: (1) create transient models based on EPANET INP files [49]; (2) set up transient models, define wave speeds, time step, operational changes in valves and pumps, background leaks, pipe bursts, location of surge tanks, nodes experiencing instantaneous demand, as well as choose the friction model; (3) compute the initial conditions for the transient simulation using WNTR Python package [36]; (4) perform transient simulations; and (5) obtain flow and pressure results. Fig. 1 illustrates the main components of the modeling framework of TSNet.

2.1. Software overview

TSNet, tested for Python versions 3.5, 3.6, and 3.7, can be installed on Windows, Linux, and Mac OS X operating systems. Python distributions, such as Anaconda, are recommended to manage the Python environment as they already contain (or easily support installation of) many Python packages (e.g. SciPy, NumPy, Pandas, and Matplotlib) that are used in the TSNet package. TSNet is available in Python Package Index (PyPI), and the stable release version of TSNet can be installed through `Pip`. All source codes can be downloaded from the GitHub repository at https://github.com/glorialulu/TSNet, which also includes links to software documentation, examples, and contact information for reporting bugs and questions. The software documentation (https://tsnet.readthedocs.io) includes detailed descriptions of the modeling framework, modeling conventions and limitations, installation instructions, setting-up and performing transient simulation, and getting simulation results. Additionally, details including the numerical scheme and comparisons to a commercial software are provided. Three example applications are included to demonstrate the application program interface (API), code structure, and the modeling capabilities of TSNet.

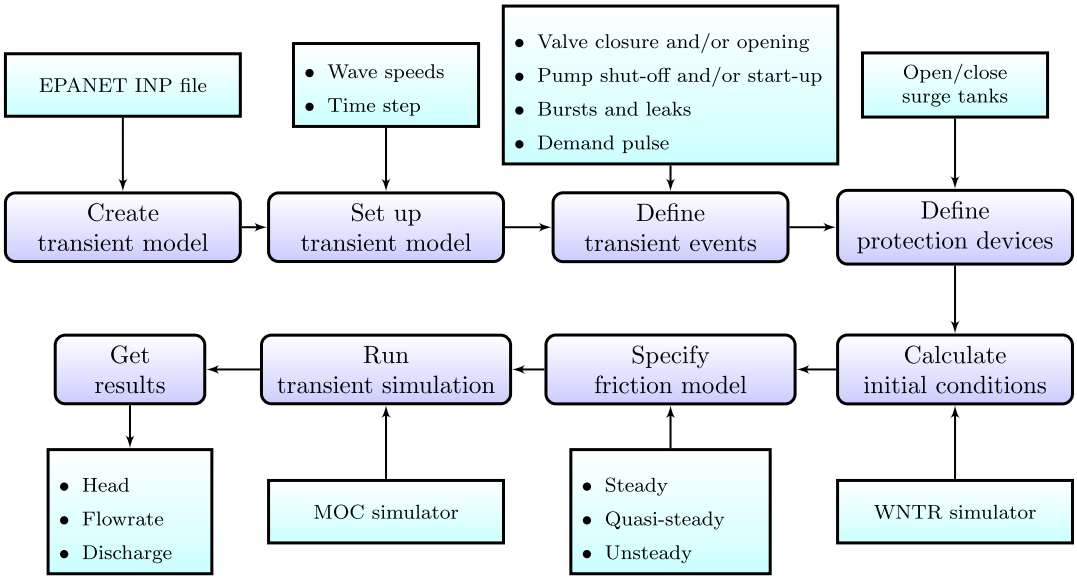

Fig. 1. TSNet modeling framework.

2.2. Modeling unsteady hydraulics

Hydraulic transients are governed by a system of partial differential equations, i.e., water hammer equations [61]. Due to the lack of analytical solutions to these systems of equations [33], many previous works have proposed to solve the equations with various numerical techniques, including but not limited to the method of characteristics (MOC) [10,16], wave characteristics method (WCM) [3,60], finite difference method (FDM) [9], finite volume method (FVM) [66], and generalized characteristic method (GCM) [45]. Among the myriad of techniques employed for transient simulation, MOC is generally considered the most popular numerical solution because of its relative accuracy and easiness in programing [22]. Thus, TSNet adopts MOC as the solution technique. The following sections present a brief overview of the main equations, boundary conditions, and the solution approach. Additional detailed information can be found in the following literature [38,61].

Transient flow in a pipe is governed by the mass and momentum conservation equations [61]:

$$\frac{\partial H}{\partial t} + \frac{a^2}{g}\frac{\partial V}{\partial x} - gV\sin\alpha = 0 \tag{1}$$

$$\frac{1}{g}\frac{\partial V}{\partial t} + \frac{\partial H}{\partial x} + h_f = 0 \tag{2}$$

where H is the head, V is the flow velocity in the pipe, t is time, a is the wave speed, g is the gravity acceleration, α is the pipe slope, and h_f represents the head loss per unit length due to friction.

The essence of MOC is to transform the set of partial differential equations (Eq. (1) and (2)) to a set of ordinary differential equations that apply along specific lines, i.e., characteristics lines. The characteristic lines represent the directions in which the disturbance in a pipe propagates, where $C+$ is associated with a positive propagation velocity and $C-$ with the negative. Then, the compatibility equations can be formulated as:

$$C+: \quad \frac{dV}{dt} + \frac{g}{a}\frac{dH}{dt} + gh_f - \frac{g}{a}V\sin\alpha = 0 \quad \text{along} \quad \frac{dx}{dt} = a \tag{3}$$

$$C-: \quad \frac{dV}{dt} - \frac{g}{a}\frac{dH}{dt} + gh_f - \frac{g}{a}V\sin\alpha = 0 \quad \text{along} \quad \frac{dx}{dt} = -a \tag{4}$$

The explicit MOC technique is then adopted to solve the compatibility equations by firstly discretizing in space and time along the characteristics lines [61]. Subsequently, given the initial conditions, the head and flow conditions can be matched to the next time step along the positive and negative characteristic lines by solving the compatibility equations simultaneously. The time-marching scheme continues until the end of the defined simulation period. In networked systems, the compatibility equations are augmented with element-specific head and flow conditions that characterize the flow behavior at the boundary nodes that connect neighboring elements (represented by computational units), such as other pipes, valves, pumps, reservoirs, as well as leaks and bursts. For example, the conservation of mass and work-energy principles are accounted for at junctions that connected pipes [38]; in-line valves, i.e., valves that are connected by pipes on both sides, and pumps are modeled in a similar manner with additional specific characteristic functions; head at the boundary is explicitly defined in case of reservoirs and tanks; the velocity boundary condition is combined when treating end-valves, i.e., valves located at the boundary of the network.

2.3. Friction method

The head loss per unit length (h_f) can be expressed as a sum of the quasi-steady (h_{fs}) and unsteady (h_{fu}) friction [6]. TSNet adopts the Darcy-Weisbach equation to compute quasi-steady head loss per unit length along a pipe [38]:

$$h_{f_s} = f\frac{V^2}{2gD} \tag{5}$$

where f is the quasi-steady friction factor, and D is the pipe diameter. The friction factor (f) is updated based on the Reynolds number at each time step using [24]:

$$\frac{1}{\sqrt{f}} = -1.8\log\left[\frac{6.9}{Re} + \left(\frac{K}{3.7D}\right)^{1.11}\right] \tag{6}$$

where Re is the Reynolds number at the current time step, and K is the equivalent roughness height. The Darcy-Weisbach method is chosen, instead of the Hazen-Williams method to model friction, as the Hazen-Williams is empirically based using experimental data [41]. Additionally, Darcy-Weisbach is quadratic with respect to the flow velocity, thus allowing more efficient and accurate numerical calculation. If the friction method specified by the user in the initial INP file is not defined as Darcy-Weisbach, TSNet computes the Darcy-Weisbach coefficients (f) based on the head loss (h_{f0}) and flow velocity (V_0) in initial conditions, using the following equation:

$$f = \frac{h_{f_0}}{V_0^2/2gD} \tag{7}$$

Unsteady friction models have also been proposed to improve the modeling accuracy of transient conditions [7,64,67]. TSNet incorporates the instantaneous acceleration-based model [6,57]:

$$h_{f_u} = \frac{k_u}{2g}\left(\frac{\partial V}{\partial t} + a\cdot\text{sign}(V)\left|\frac{\partial V}{\partial x}\right|\right) \tag{8}$$

where h_{fu} is the head loss per unit length due to unsteady friction, $\frac{\partial V}{\partial t}$ is the local instantaneous acceleration, $\frac{\partial V}{\partial x}$ is the convective instantaneous acceleration, and k_u is Brunone's friction coefficient, which can be analytically determined using Vardy's sheer decay coefficient (C^*) [56]:

$$k_u = \frac{C^*}{2} \tag{9}$$

$$C^* = \begin{cases} 0.00476 & \text{laminar flow}(Re \le 2000) \\ \dfrac{7.41}{Re^{\log(14.3/Re^{0.05})}} & \text{turbulent flow}(Re > 2000) \end{cases} \tag{10}$$

The acceleration terms, i.e., $\frac{\partial V}{\partial t}$ and $\frac{\partial V}{\partial x}$ in Eq. 8, are evaluated explicitly using first-order finite difference scheme.

2.4. Pressure-dependent demand

During the transient simulation in TSNet, the demands are treated as pressure-dependent discharge; thus, the actual demands will vary from the demands defined by the user. The actual demands (d_{actual}) are modeled based on the instantaneous pressure head at the node and the demand discharge coefficients, using the following equation:

$$d_{actual} = k\sqrt{H_p} \tag{11}$$

where H_p is the pressure head and k is the demand discharge coefficient, which is calculated as the ratio between the nominal demand (d_0) and the initial pressure head (H_{p0}) [36]:

$$k = \frac{d_0}{\sqrt{H_{p_0}}} \tag{12}$$

The pressure-dependent demand method allows the actual demands to fluctuate with the instantaneous local pressure, representing more realistic conditions [30]. It should be noted that if the pressure head is negative, the demand flow will be treated as zero, assuming that a backflow preventer is installed at each node. Moreover, TSNet also incorporates the simulation of hydraulic transients triggered by instantaneous demand pulses [13], which is detailed in the online documentation.

2.5. Choice of time step

In MOC, the time and space domain is discretized along the characteristic lines; thus, the temporal and spatial discretization can be uniquely determined by specifying the time increment, i.e., time step (Δt). Once the time step is specified, for a single pipe segment, the spatial increment (Δx) can be computed by $\Delta x = \Delta t \times a$. However, determining the time step for the entire WDN is not a trivial task. Several prior works have focused on exploring the choice of the time step and its effect on the numerical stability of the solutions, the resolution of the results, and the computational complexity [21,32]. To begin with, a tradeoff exists between computational complexity and numerical accuracy: a small time step can yield relatively accurate results, while requiring more computational resources due to the increase in the density of the computational grid. Additionally, the numerical scheme poses the following two constraints that have to be satisfied simultaneously: (1) the Courant's criterion has to be satisfied for each pipe [61], indicating the maximum time step allowed in the network transient analysis should satisfy: $\Delta t \leqslant \min\left(\frac{L_i}{N_i a_i}\right),\ i = 1, 2, \cdots, n_p$ where N_i is the number of computational units in pipe i and n_p is the number of pipes in the WDN; (2) the time step has to be the same for all the pipes in the network, therefore restricting the wave travel time $\frac{L_i}{N_i a_i}$ to be the same for any computational unit in the network. However, this is not realistic in a real network, because different pipe lengths and wave speeds usually result in different wave travel times. Moreover, the number of computational units in the i^{th} pipe (N_i) has to be an integer due to the grid configuration in MOC; nevertheless, the combination of time step and pipe length is likely to produce non-integer value of N_i, which then requires further adjustment.

This package adopted the wave speed adjustment scheme [61] to ensure that the two criterion stated above are satisfied. To begin with, the maximum allowed time step $\Delta t_{\max}$ is calculated, assuming there are two computational segments on the critical pipe:

$$\Delta t_{max} = \min\left(\frac{L_i}{2a_i}\right),\ i = 1, 2, \cdots, n_p \tag{13}$$

If the user defines a time step greater than $\Delta t_{\max}$, a fatal error will be raised and the program will be stopped. Otherwise, the user-defined value will be used as the initial guess for the upcoming two-step adjustment. The determination of time step is not straightforward, especially in large networks. Thus, we allow the user to ignore the time step setting, in which case $\Delta t_{\max}$ will be used as the initial guess for the upcoming two-step adjustment.

After setting the initial guess for the time step, the following adjustments are performed. Firstly, the i^{th} pipe with length (L_i) and wave speed (a_i) will be discretized into (N_i) segments:

$$N_i = \text{round}\left(\frac{L_i}{a_i \Delta t_{max}}\right),\ i = 1, 2, \cdots, n_p \tag{14}$$

Secondly, the discrepancies in $\Delta t_{\max}$ introduced by the rounding of N_i will be compensated by correcting the wave speeds as $a_i(1 \pm \phi_i)$, where ϕ_i is the wave speed adjustment for pipe i. Least squares approximation is then used to determine Δt such that the sum of squares of the wave speed adjustments ($\Sigma {\phi_i}^2$) is minimized [2], as follows:

$$\Delta t = \text{argmin}_{\phi, \Delta t}\left\{\sum_{i=1}^{n_p} {\phi_i}^2 \ \middle|\ \Delta t = \frac{L_i}{a_i(1 \pm \phi_i)N_i}\ \ i = 1, 2, \cdots, n_p\right\} \tag{15}$$

Ultimately, an adjusted Δt is determined and used in the transient simulation. The total number of time steps (tn) can then be calculated by dividing the simulation duration (tf) by the time step (Δt). The selection of the time step is further explained in the online documentation.

2.6. Initial conditions

Prior to performing a transient simulation, initial steady-state conditions, i.e. pipe flows and nodal heads, need to be established. TSNet employs WNTR [36] for simulating the steady state in the network to establish the initial conditions for the upcoming transient simulations. WNTR is chosen for simulating the initial conditions due to its capabilities of simulating demand-driven or pressure-dependent hydraulics simulations as well as background leaks.

2.7. Leaks and bursts

During the transient simulation, a leaking node is modeled using the two compatibility equations (Eq. (3) and (4)), a continuity equation, and an orifice equation, which quantifies the pressure-dependent leak discharge (Q_l) [38]:

$$Q_l = k_l \sqrt{H_{pl}} \tag{16}$$

where H_{pl} is the pressure head at the location of the leak, and k_l is the lumped leak coefficient, which aggregates the size of the leak, units, and leak coefficients. Moreover, if the pressure head is negative, the leak discharge will be set to zero, assuming a backflow preventer is installed at the leaking node.

The simulation of bursts and leaks is very similar, as they share a similar set of governing equations. The only difference is that in the burst model the lumped burst coefficient (k_l) changes with time. In other words, using a burst, the user can model new and evolving conditions, while the leak model simulates an existing leak in the system. In TSNet, the burst is assumed to be developed linearly in time, indicating that the burst area increases linearly from zero to a size specified by the user during the specified time period. Thus, a burst event can be modeled by defining the start time (ts), the time for the burst to fully develop (tc), and the final burst coefficient when the burst is fully developed.

In TSNet, leaks and bursts are assigned to the network nodes by specifying the location of the leak/burst node and the corresponding lumped leak/burst coefficient (k_l). Existing leaks should be included in the initial conditions calculated using WNTR simulator; thus, it is necessary to define the leaks before calculating the initial conditions. More information about the inclusion of leaks in the steady state calculation can be found in WNTR documentation [36].

2.8. Valve operations

Valve operations, including closure and opening, are supported in TSNet. A valve is modeled using the two compatibility equations (Eq. (3) and (4)), the continuity equation, and the valve characteristic curve equation. The default valve type is gate valve with a characteristic curve defined according to ([38], Figure 10.12). Other valve types can be defined by supplementing the valve characteristic curve, which defines how valve loss coefficient changes with open percentage. In TSNet, valve closure is simulated by defining the valve closure start time (ts), the operating duration (tc), the valve opening percentage when the closure is completed (se), and the operating constant (m), which characterizes the shape of the closure curve. These parameters define the valve closure curve, as shown in Fig. 2 (a). The solid black and dashed red curves correspond to the valve operating curves with $m = 1$ and $m = 2$, respectively.

Valve opening can be simulated by defining a similar set of parameters related to the valve opening curve. The valve opening curves with $m = 1$ and $m = 2$ are illustrated in Fig. 2 (b).

2.9. Pump operations

TSNet also includes the capability to perform controlled pump

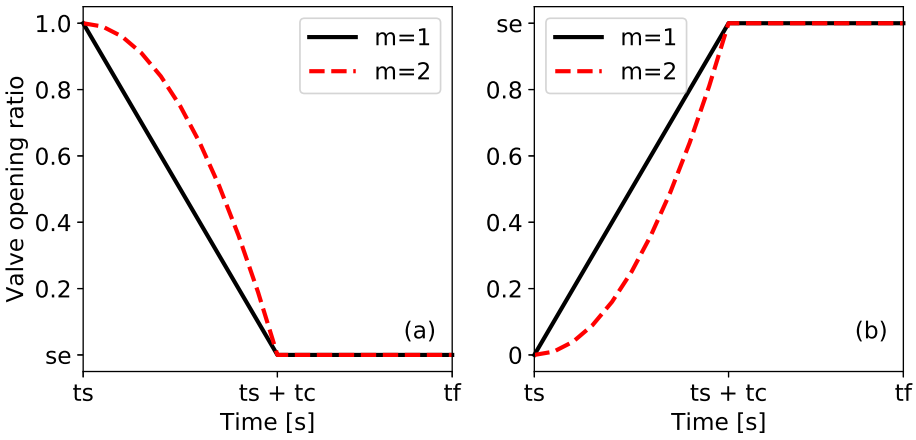

Fig. 2. Valve operating curve: (a) valve closure; (b) valve opening.

operations by specifying how pump rotational speed changes over time with the same set of parameters as in valve operations, i.e., *ts, tc, se, m*. Explicitly, during pump start-up, the rotational speed of the pump is increased based on the user defined operating rule. The pump is then modeled using the two compatibility equations (Eq. (3) and (4)), a continuity equation, the pump characteristic curve at the given rotational speed, and the affinity laws [38], thus resulting in the rise of pump flowrate and the addition of mechanical energy. Conversely, during pump shut-off, as the rotational speed of the pump decreased according to the user defined operating rule, the pump flowrate and the addition of mechanical energy decrease. Pump shut-off due to power failure, when the reduction of pump rotational speed depends on the characteristics of the pump, e.g., the rotational moment of inertia, has not been included in the current version of TSNet.

2.10. Surge tanks

The modeling of water hammer protection devices, including the open and closed surge tanks, are also incorporated in TSNet. An open surge tank is modeled as an open chamber connected directly to a pipeline and is open to the atmosphere [61]. In the initial conditions, the head in the tank is equal to the head in the connected pipeline. During a transient simulation, an open surge tank moderates pressure transients by storing the excess water when a pressure jump occurs in the surge tank connection, and supplies water in the event of a pressure drop. In TSNet, open surge tanks are assumed to have an infinite height such that water never overflows and can be added to the network by specifying the location and the cross-sectional area. Due to the modeling simplicity, open surge tanks can serve as a good initial approach to investigate the placement of surge protection devices. However, the infinite height assumption is not realistic and the major disadvantage of open surge tanks is that it typically cannot accommodate large pressure transients unless the tank is excessively tall and large, which limits its usefulness.

TSNet also incorporates a closed surge tank (i.e., air chamber) to simulate a more realistic surge protection device. An air chamber is a relatively small sealed vessel with compressed air at its top and water in the bottom [61]. During a transient simulation, the closed surge tank moderates pressure transients by slowing down the deceleration or the acceleration of water flow. For example, when pressure in the upstream connection increases, water flows into the tank and water level in the tank increases, then air volume compresses and air pressure increases, thus slowing down the acceleration of the water in flow into the tank and the increase in pressure. Similarly, when pressure in the upstream

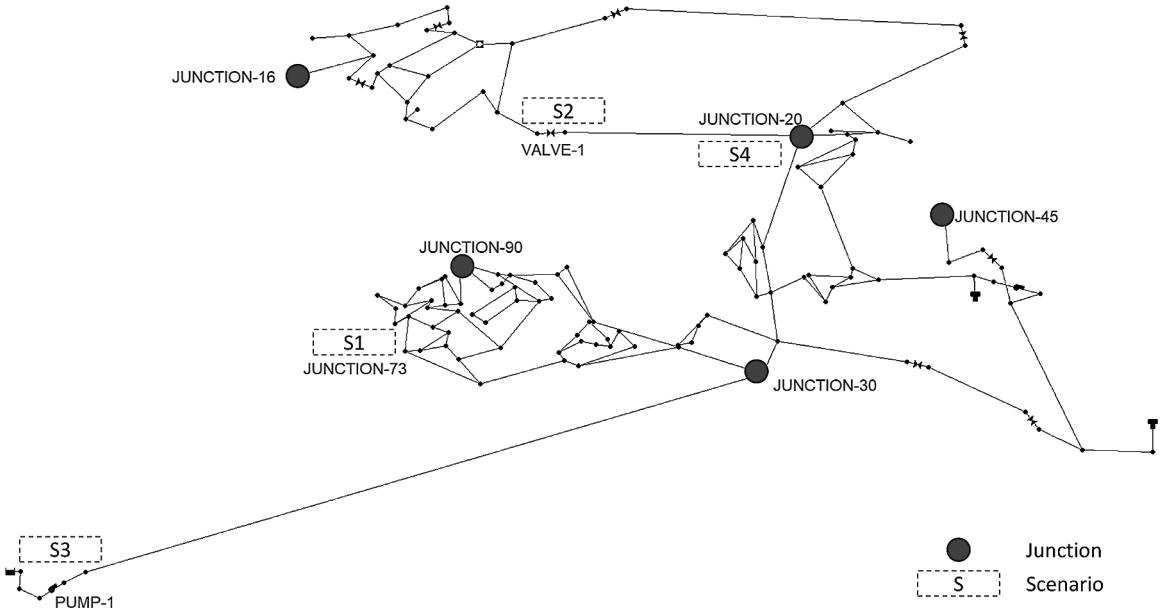

Fig. 3. Example network for demonstrating TSNet for scenarios: 1) pipe burst, 2) valve closure, 3) pump shut-off, and 4) leak. Results are reported at junctions: 16, 20, 30, 45, 90.

```
import tsnet
# open an example network and create a transient model
inp_file = './networks/Tnet3.inp'
tm = tsnet.network.TransientModel(inp_file)
```

Fig. 4. Creating transient model.

```
import numpy as np
wavespeed = np.random.normal(1200., 100., size=tm.num_pipes)
tm.set_wavespeed(wavespeed) # set wavespeed
tf = 20 # simulation period [s]
tm.set_time(tf) # set time step
```

Fig. 5. Defining time step and wave speed.

```
ts = 1 # burst start time
tc = 1 # time for burst to fully develop
final_burst_coeff = 0.01 # final burst coeff [ m^3/s/(m H20)^(1/2)]
tm.add_burst('JUNCTION-73', ts, tc, final_burst_coeff) # add burst
```

Fig. 6. Defining a pipe burst.

connection drops, water flows from the tank and water level in the tank decreases, then air volume increases and air pressure decreases, thus slowing the deceleration of the water flow and the decrease of pressure head. In TSNet, the user can add a closed surge tank by specifying the location, cross-sectional area, total height of the surge tank, and initial water height in the tank.

3. Example applications

The following example applications demonstrate the multiple capabilities of TSNet Python package, including pipe burst (Scenario 1), valve closure (Scenario 2), pump shut-off (Scenario 3), and background leak (Scenario 4). Additionally, system response with unsteady friction model, surge tank as well as comparison with Hammer [25] simulation results are presented. Additional examples including the numerical scheme, selection of the time step, demand-pulse for simulating instantaneous nodal demand, and the complete codes are provided in the online documentation (https://tsnet.readthedocs.io).

3.1. Creating transient model

Fig. 3 illustrates the example network adopted from [46], which will be used to demonstrate how to interact with TSNet and its simulation results for the different scenarios. The network comprises 126 nodes, 1 reservoir, 2 tanks, 168 pipes, 2 pumps, and 8 valves.

The first two steps to create and set up a transient model in TSNet include:

1. Create transient model: import TSNet package, read the EPANET INP file, and create transient model object (see Fig. 4);
2. Define the time step and wave speed in the transient model: in addition to the information included in the INP file, the user needs to specify the wave speeds for each pipe and the time step for the transient simulation, as shown in Fig. 5. For illustration purposes, we assume that the wave speed for the pipes is normally distributed with a mean of $1200m/s$ and standard deviation of $100m/s$. Then, assign the randomly generated wave speed to each pipe in the network according to the order of the pipes defined in the INP file. Moreover, we set the simulation period $t_f = 20s$. Here, we do not specify the time step, hence the suggested time step will be used as explained in Section 2.5.

Next, we will show how to define different scenarios for transient simulations, get and visualize results, as well as compare results between the different scenarios.

3.2. Scenario 1: Burst

In Scenario 1, a burst event at JUNCTION-73 is simulated by defining the burst location, burst start time ($ts = 1s$), time for burst to fully develop ($tc = 1s$), and the final burst coefficient (final_burst_coeff = 0.01 $m^3/s/(m\ H_2O)^{0.5}$), as shown in Fig. 6.

Once the transient model and the event are established, the transient simulation can then be initialized and executed with the specified friction model as presented in Fig. 7:

At the beginning of a transient simulation, TSNet will report the approximated simulation time based on the calculation time of the first few time steps and the total number of time steps. Additionally, the computation progress will be printed on the screen as the simulation proceeds, as shown in Fig. 8.

Once the simulation is completed, the results can be retrieved as shown in Fig. 9. The burst discharge at JUNCTION-73 is shown in Fig. 10. Noticeably, burst discharge increases as the burst develops from 1 to 2 seconds. After the burst is fully developed, the burst discharge fluctuates around $0.07m^3/s$. If we assume this burst can be fixed within half an hour, it will waste around $130m^3$ of treated water. Thus, TSNet

```
t0 = 0. # initialize the simulation at 0s
engine = 'DD' # using demand driven steady state engine
tm = tsnet.simulation.Initializer(tm, t0, engine) # initialize
result_obj = 's1' # name of the object for saving simulation results
friction = 'steady'  # or 'quasi-steady' or 'unsteady'
tm = tsnet.simulation.MOCSimulator(tm,result_obj,friction) # run
transient simulation
```

Fig. 7. Initializing and running transient simulation.

```
Simulation time step 0.01154 s
Total Time Step in this simulation 1732
Estimated simulation time 0:01:20.981392
Transient simulation completed 9 %...
Transient simulation completed 19 %...
Transient simulation completed 29 %...
Transient simulation completed 39 %...
Transient simulation completed 49 %...
Transient simulation completed 59 %...
Transient simulation completed 69 %...
Transient simulation completed 79 %...
Transient simulation completed 89 %...
Transient simulation completed 99 %...
Auctual computational time: 84.5517 seconds
```

Fig. 8. Runtime output: calculation time and progress.

can be used to assess water loss during different pipe failure scenarios.

Additionally, network information, operating rules, and simulation results are saved in the specified `tm` object, e.g., `s1.obj` as in this example (Fig. 7, line 4). Node results include head, discharge through leaks, bursts, and demand nodes. Link results include head, flow rate, and velocity at start- and end-node of each link. The result for each attribute is a Numpy array, representing the time history of the simulation results, the length of which equals the total number of simulation time steps. To retrieve the results from a previously completed simulation, one can read the `tm` object from the `s1.obj` file and access results using the Pickle module in Python (see example code in the online documentation).

Fig. 12 (a) reports the change in the hydraulic head with respect to the nominal head using different friction models at multiple junctions across the network, thus clearly showing the time of arrival and the amplitude of the pressure transient observed at each of the junctions. The dashed lines represent the simulation results with the steady and quasi-steady friction models, which predict identical pressure transients at all reporting junctions. The solid lines show the simulation results with the unsteady friction model. It can be noticed that the pressure transient arrives first to JUNCTION-90, and then followed by pressure transients depicted at JUNCTION-30, JUNCTION-20, JUNCTION-45, and finally JUNCTION-16. In this scenario, the order of arrival corresponds to the distance from the reporting location to the location of burst. Additionally, the highest amplitude of the pressure wave over $40m$ is observed at JUNCTION-90, closest to the location of the burst. The amplitude of the waves at the remaining locations decreased as the wave travels through the pipelines. It should be noted that pressure wave propagation depends on the wave speeds in the pipes and the paths taken; hence, the correlation between time of arrival and distance is case specific. These results can be used to predict the magnitude and shape of transients induced by burst events and inform sensor placement to maximize information gain for pipe failure detection. Moreover, further analysis can be performed to assess the internal stress conditions on the pipelines imposed by pressure transients, thereby informing proactive local inspection and maintenance.

Furthermore, the computational results using different friction models agree well for the first pressure drop, while the discrepancies between the pressure transients are magnified for the latter simulation period as the results from the unsteady friction model exhibit additional damping and positive phase shift. The discrepancies in damping and phase stem from the additional unsteady friction terms with the temporal acceleration term ($\frac{\partial V}{\partial t}$) contributing to the phase shift, and the convective acceleration term ($\frac{\partial V}{\partial s}$) producing additional damping [57]. However, the contributions of unsteady friction are relatively small in this example even for the latter period of the 20s simulation time. The observed changes in damping and phase shift in the pressure agree with results reported in the literature that unsteady friction model changes the transient response, but in large water networks these changes are often marginal [12,17,52,54].

To mediate the effects of pressure transients, a closed surge tank, i.e., air chamber, is added to the network at JUNCTION-89 by specifying the desired location, cross-sectional area (ta = $10m^2$), total height of the surge tank (th = $10m$), and initial water height in the tank (wh = $5m$), as illustrated in Fig. 11.

The simulation results with a closed surge tank are shown in Fig. 12(b). The comparison between Fig. 12(a) and (b) demonstrates that the surge tank can considerably moderate pressure transients. For example, in (a) without the surge tank, the pressure at JUNCTION-90 (purple line) drops more than 40m at the first cycle, while the amplitude of pressure drops with the closed surge tank is reduced significantly to well below 10m, as shown in (b). In fact, the amplitude of pressure transients at all reported junctions is below 10m with the surge tank added at JUNCTION-89.

```
node = 'JUNCTION-73' # name of the burst node
node = tm.get_node(node)
burst_discharge = node.emitter_discharge
```

Fig. 9. Retrieving results.

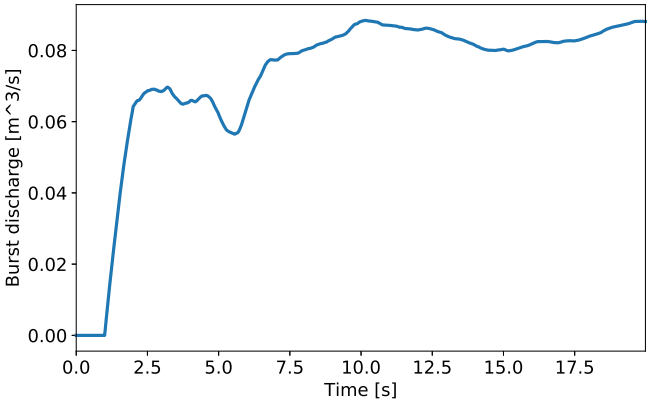

Fig. 10. Burst discharge at JUNCTION-73.

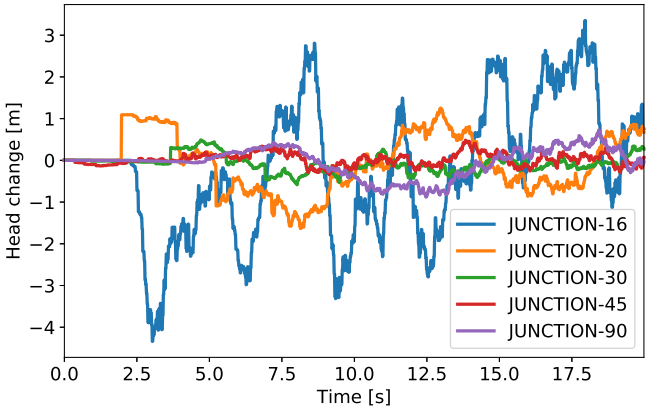

Fig. 14. Pressure transients at multiple junctions generated by closing VALVE-1.

3.3. Scenario 2: Valve closure

Scenario 2 simulates the closure of VALVE-1, which starts at the beginning of the transient simulation, $ts = 0s$, and takes $tc = 1s$ to complete. After creating the transient model, the valve closure operating rule is defined as in Fig. 13. Once the transient conditions are defined, the transient model is initialized by running a steady state simulation using the `tsnet.simulation.Initializer` method and the transient simulation is performed using the `tsnet.simulation.MOCSimulator` method, as shown in Fig. 7. Results can then be extracted using the `get_link` or `get_node` methods, as illustrated in

```
ta = 10   # tank cross sectional area [m^2]
th = 10   # tank height [m]
wh = 5    # initial water level [m]
tm.add_surge_tank('JUNCTION-90', [ta,th,wh], 'closed')
```

Fig. 11. Adding a surge tank.

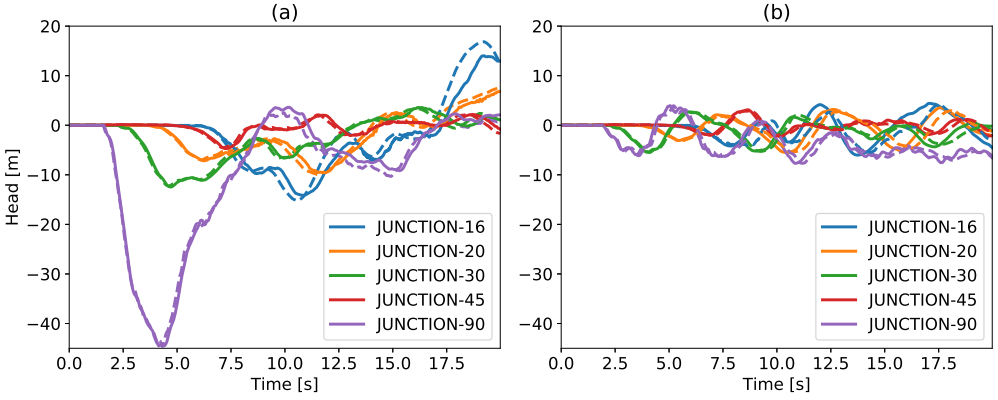

Fig. 12. Pressure transients at multiple junctions generated by the burst at JUNCTION-73: (a) without a surge tank; (b) with a surge tank. Solid lines represent the unsteady friction model and dashed lines represent the steady/quasi-steady friction models.

```
tc = 1 # valve closure period [s]
ts = 0 # valve closure start time [s]
se = 0 # end open percentage [s]
m = 1 # closure constant [dimensionless]
valve_op = [tc,ts,se,m]
tm.valve_closure('VALVE-1',valve_op) # set valve closure rule
```

Fig. 13. Defining valve closure.

```
tc = 2 # pump shut-off period
ts = 0 # pump shut-off start time
se = 0 # end open percentage
m = 1 # closure constant
pump_op = [tc,ts,se,m]
tm.pump_shut_off('PUMP-1', pump_op) # set pump shut-off rule
```

Fig. 15. Defining pump shut-off.

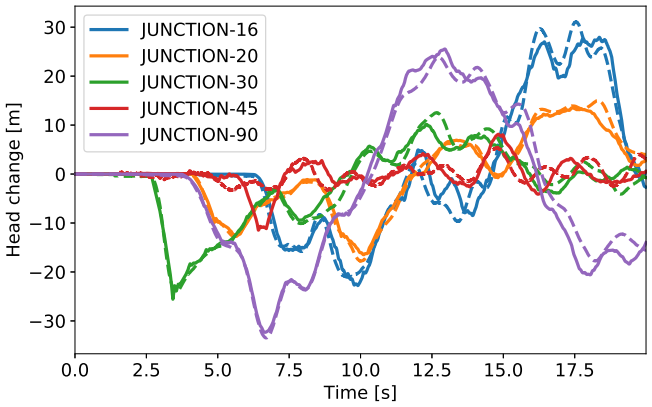

Fig. 16. Pressure transients at multiple junctions generated by shuting off PUMP-1: solid lines represent TSNet results and dashed lines represent Hammer results.

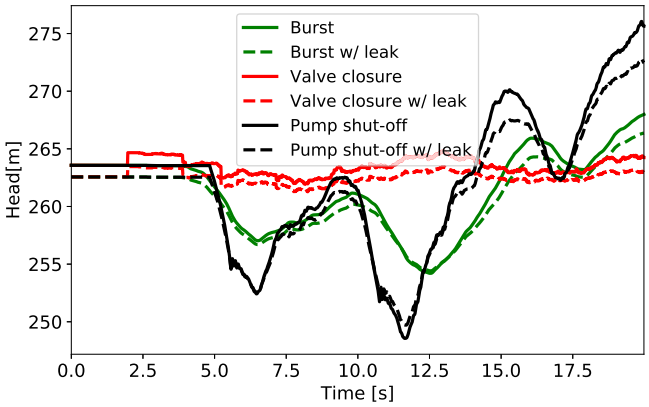

Fig. 18. Pressure at JUNCTION-20 during various events.

Fig. 9.

The results of head change at multiple junctions are illustrated in Fig. 14. The valve closure induces pressure drop at JUNCTION-16, located downstream of the valve, and pressure jump at JUNCTION-20, positioned upstream of the operating valve. The pressure changes at other three stations are well below 1m due to the fact that they are located further away from the valve. Moreover, the amplitude of the pressure transients generated by this valve closure is generally smaller than that in Scenario 1 because the initial flow in VALVE-1 is relatively small.

3.4. Scenario 3: Pump shut-off

Scenario 3 illustrates how TSNet models a transient event resulting from a controlled pump shut-off at PUMP-1, i.e., the pump speed is ramped down. The pump operating curve is defined by specifying how pump rotational speed is reduced over time as illustrated in Fig. 15:

Fig. 16 shows the pressure transients at multiple junctions. The pressure wave generated by pump shut-off reaches the junctions at different times depending on the distance from the pump: JUNCTION-30 senses the transient first, while JUNCTION-45 experiences it last. Additionally, pressure drops with amplitude greater than 10m can be discerned at all junctions, indicating the pump shut-off, especially when operated quickly, can generate significant transients in the WDN. Therefore, it is essential to evaluate the impacts of pump operations on the pipelines and design an appropriate procedure to guide pump operation.

Moreover, TSNet simulation results were compared with Hammer [25] for all scenarios. Fig. 16 demonstrates the simulation results at the reporting junctions, where the solid lines represent TSNet results, and the dashed lines represent Hammer results. Despite the slight discrepancies, which can be explained by the different wave speed adjustment schemes and boundary condition configurations adopted by the two software, the results from TSNet and Hammer closely resemble each other both in terms of attenuation and phase shift throughout the simulation period.

3.5. Scenario 4: Background leak

Scenario 4 introduces a background leak at JUNCTION-20 to the three scenarios described above. The leak coefficient is set as the final burst coefficient in Scenario 1. Fig. 17 shows a code snippet for defining the location and leak coefficient.

The head results at JUNCTION-20 during the three transient cases, i.e., burst, valve closure, and pump shut-off, with and without the leak are presented in Fig. 18. The solid lines represent the results without background leak, while the dashed lines illustrate results with the background leak. It can be observed that the leak effects the initial conditions by reducing the head at the leaking node by around 1m. Furthermore, the amplitude of the pressure changes with the presence of the leak is slightly reduced compared to the no background leak case. In other words, the leak acts as the damper for transients and relaxes the sensitivity of the system in regard to flow disturbances.

```
leak_coeff = 0.01 # [ m^3/s/(m H20)^(1/2)]
tm.add_leak('JUNCTION-20', leak_coeff) # add leak
```

Fig. 17. Defining background leak.

Table 1
TSNet computation time.

Time step (Δt)	steady	quasi-steady	unsteady
0.0115s	84s	131s	206s
0.0055s	238s	409s	679s

3.6. Computation time

The computation time of TSNet using different time step and friction models for this example network is reported in Table 1. The simulations are performed on a Window machine with Intel(R) Core(TM) i7-7700 CPU@3.60GHz. As expected, computational time increases significantly with smaller time steps and more complicated friction model.

4. Conclusions

In this paper, we present the TSNet, an open source Python package, for transient simulation in water networks. All source codes, software documentation as well as three complete examples including INP network files and codes are provided with the package and can be downloaded from the GitHub repository. The capability and user interaction with TSNet is demonstrated through the detailed simulation example of bursts, leaks, valve closure, surge tank, and pump shut-off. TSNet package provides users with open source and freely available Python codes and package for simulating transients in WDNs that can be integrated with other case specific applications, e.g., sensor placement, event detection, model calibration, and sensitivity analysis. Additionally, this package contributes a platform to encourage users and developers to further develop, improve, and extend the transient model.

TSNet does not include all the modeling capabilities of the commercial software; instead, it is designed to provide simulation capabilities for transient modeling in WDS for the research community that are currently not available in open source software including EPANET [49] and WNTR [36]. TSNet is under continuous maintenance, improvement, and development.

CRediT authorship contribution statement

Lu Xing: Methodology, Software, Validation, Writing - original draft, Visualization. **Lina Sela:** Conceptualization, Methodology, Writing - review & editing, Supervision, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the University of Texas at Austin Startup Grant and by the National Science Foundation under Grant 1953206.

References

[1] Alawadhi A, Tartakovsky DM. Bayesian update and method of distributions: Application to leak detection in transmission mains. Water Resources Research 2020. e2019WR025879
[2] Åström KJ, Wittenmark B. Computer-controlled systems: theory and design. Courier Corporation; 2013.
[3] Boulos PF, Karney BW, Wood DJ, Lingireddy S. Hydraulic transient guidelines for protecting water distribution systems. J-Am Water Work Assoc 2005;97(5):111–24.
[4] Brunone B. Transient test-based technique for leak detection in outfall pipes. J Water Resour Plann Manage 1999;125(5):302–6.
[5] Brunone B, Ferrante M. Detecting leaks in pressurised pipes by means of transients. J Hydraulic Res 2001;39(5):539–47.
[6] Brunone B, Golia U, Greco M. Effects of two-dimensionality on pipe transients modeling. J Hydraul Eng 1995;121(12):906–12.
[7] Brunone B, Karney BW, Mecarelli M, Ferrante M. Velocity profiles and unsteady pipe friction in transient flow. J Water Resour Plann Manage 2000;126(4):236–44.
[8] Capponi C, Ferrante M, Zecchin AC, Gong J. Leak detection in a branched system by inverse transient analysis with the admittance matrix method. Water Resour Manage 2017;31(13):4075–89.
[9] Chaudhry M, Hussaini M. Second-order accurate explicit finite-difference schemes for waterhammer analysis. J Fluids Eng 1985;107(4):523–9.
[10] Chaudhry MH. Transient-flow equations. Applied hydraulic transients. New York, NY: Springer New York978-1-4614-8538-4; 2014. p. 35–64.
[11] Covas D, Ramos H. Case studies of leak detection and location in water pipe systems by inverse transient analysis. J Water Resour Plann Manage 2010;136(2):248–57.
[12] Covas D, Stoianov I, Ramos H, Graham N, Maksimovic C. The dynamic effect of pipe-wall viscoelasticity in hydraulic transients. part i - experimental analysis and creep characterization. J Hydraulic Res 2004;42(5):517–32.
[13] Creaco E, Campisano A, Franchini M, Modica C. Unsteady flow modeling of pressure real-time control in water distribution networks. J Water Resour Plann Manage 2017;143(9):04017056.
[14] Creaco E, Campisano A, Modica C. Testing behavior and effects of PRVs and RTC valves during hydrant activation scenarios. Urban Water J 2018;15(3):218–26.
[15] Creaco E, Pezzinga G, Savic D. On the choice of the demand and hydraulic modeling approach to WDN real-time simulation. Water Resour Res 2017;53(7):6159–77.
[16] De Almeida AB, Koelle E. Fluid transients in pipe networks. Windsor, United Kingdom: Elsevier Applied Science; 1992.
[17] Duan H-F, Ghidaoui M, Lee P, Tung Y. Relevance of unsteady friction to pipe size and length in pipe fluid transients. J Hydraul Eng 2012;138(2):154–66.
[18] Duan H-F, Tung Y-K, Ghidaoui MS. Probabilistic analysis of transient design for water supply systems. J Water Resour Plann Manage 2010;136(6):678–87.
[19] Ebacher G, Besner M, Clément B, Prévost M. Sensitivity analysis of some critical factors affecting simulated intrusion volumes during a low pressure transient event in a full-scale water distribution system. Water Res 2012;46(13):4017–30.
[20] Ferrante M, Brunone B, Meniconi S. Wavelets for the analysis of transient pressure signals for leak detection. J Hydraul Eng 2007;133(11):1274–82.
[21] Ghidaoui MS, Karney BW, McInnis DA. Energy estimates for discretization errors in water hammer problems. J Hydraul Eng 1998;124(4):384–93.
[22] Ghidaoui MS, Zhao M, McInnis DA, Axworthy DH. A review of water hammer theory and practice. Appl Mech Rev 2005;58(1):49–76.
[23] Gong J, Simpson AR, Lambert MF, Zecchin AC, Kim Y-i, Tijsseling AS. Detection of distributed deterioration in single pipes using transient reflections. J Pipeline Syst Eng Pract 2012;4(1):32–40.
[24] Haaland SE. Simple and explicit formulas for the friction factor in turbulent pipe flow. J Fluids Eng 1983;105(1):89–90.
[25] HAMMER, Bentley. Water hammer and transient analysos software. 2019. URL https://www.bentley.com/en/products/product-line/hydraulics-and-hydrology-software/hammer.
[26] Innovyze. Infosurge - user giude. 2019. URL https://www.innovyze.com/en-us/products/infowater/infosurge.
[27] Izquierdo J, Montalvo I, Pérez-García R, Ayala-Cabrera D. Multi-agent simulation of hydraulic transient equations in pressurized systems. J Comput Civil Eng 2016;30(4):04015071.
[28] Jones S, Shepherd W, Collins R, Boxall J. Experimental quantification of intrusion volumes due to transients in drinking water distribution systems. J Pipeline Syst Eng Pract 2019;10(1):04018026.
[29] Kanakoudis V. A troubleshooting manual for handling operational problems in water pipe networks. J Water Supply 2004;53(2):109–24.
[30] Kanakoudis V, Gonelas K. Assessing the results of a virtual pressure management project applied in kos town water distribution network. Desalination Water Treat 2016;57(25):11472–83.
[31] Kapelan Z. Calibration of water distribution system hydraulic models.. University of Exeter; 2002.
[32] Karney BW, Ghidaoui MS. Flexible discretization algorithm for fixed-grid MOC in pipelines. J Hydraul Eng 1997;123(11):1004–11.
[33] Karney BW, McInnis D. Efficient calculation of transient flow in simple pipe networks. J Hydraul Eng 1992;118(7):1014–30.
[34] Khilqa S, Elkholy M, Al-Tofan M, Caicedo JM, Chaudhry MH. Uncertainty quantification for damping in transient pressure oscillations. J Water Resour Plann Manage 2019;145(9):04019039.
[35] Kjerrumgaard Jensen R, Kær Larsen J, Lindgren Lassen K, Mandø M, Andreasen A. Implementation and validation of a free open source 1D water hammer code. Fluids 2018;3(3):64.
[36] Klise KA, Hart D, Moriarty D, Bynum ML, Murray R, Burkhardt J, et al. Water

Network Tool for Resilience (WNTR) User Manual. Tech. Rep.. Washington, D.C.: U.S. Environmental Protection Agency; 2017.
[37] KyPipe, LLC. Pipe 2018 users guide. 2019. URL http://kypipe.com/.
[38] Larock BE, Jeppson RW, Watters GZ. Hydraulics of pipeline systems. CRC press; 1999.
[39] Lee P, Lambert M, Simpson A, Vitkovsky J, Misiunas D. Leak location in single pipelines using transient reflections. Austral J Water Resources 2007;11(1):53–65.
[40] Liggett JA, Chen L-C. Inverse transient analysis in pipe networks. J Hydraul Eng 1994;120(8):934–55.
[41] Liou CP. Limitations and proper use of the hazen-williams equation. J Hydraul Eng 1998;124(9):951–4.
[42] McInnis D., Karney B., Axworthy D.. Transam reference manual. 1998. URL http://hydratek.com/expertise/transient-analysis-model.
[43] Misiunas D, Vítkovskỳ J, Olsson G, Simpson A, Lambert M. Pipeline break detection using pressure transient monitoring. J Water Resour Plann Manage 2005;131(4):316–25.
[44] Moser G, Paal SG, Smith IF. Leak detection of water supply networks using error-domain model falsification. J Comput Civil Eng 2017;32(2):04017077.
[45] Nault J, Karney B, Jung B-S. Generalized flexible method for simulating transient pipe network hydraulics. J Hydraul Eng 2018;144(7):04018031.
[46] Ostfeld A, Uber JG, Salomons E, Berry JW, Hart WE, Phillips CA, et al. The battle of the water sensor networks (BWSN): adesign challenge for engineers and algorithms. J Water Resour Plann Manage 2008;134(6):556–68.
[47] Prescott SL, Ulanicki B. Improved control of pressure reducing valves in water distribution networks. J Hydraul Eng 2008;134(1):56–65.
[48] Rezaei H, Ryan B, Stoianov I. Pipe failure analysis and impact of dynamic hydraulic conditions in water supply networks. Procedia Eng 2015;119:253–62.
[49] Rossman L. EPANET 2 users manual. Tech. Rep.. Washington, D.C.: U.S. Environmental Protection Agency; 2000.
[50] Simpson A, Vitkovsky J, Lambert M. Transients for calibration of pipe roughnesses using genetic algorithms. BHR group conference series publication. 39. Bury St. Edmunds; Professional Engineering Publishing; 1998; 2000. p. 587–98.
[51] Srirangarajan S, Allen M, Preis A, Iqbal M, Lim HB, Whittle AJ. Wavelet-based burst event detection and localization in water distribution systems. J Signal Process Syst 2013;72(1):1–16.
[52] Stephens M, Simpson AR, Lambert MF, Vítkovskỳ JP. Field measurements of unsteady friction effects in a trunk transmission pipeline. Impacts of global climate change. 2005. p. 1–12.
[53] Stephens ML, Lambert MF, Simpson AR. Determining the internal wall condition of a water pipeline in the field using an inverse transient. J Hydraul Eng 2012;139(3):310–24.
[54] Stephens ML, Lambert MF, Simpson AR, Vitkovsky J. Calibrating the water-hammer response of a field pipe network by using a mechanical damping model. J Hydraul Eng 2011;137(10):1225–37.
[55] Tuck J, Lee P. Inverse transient analysis for classification of wall thickness variations in pipelines. Sensors 2013;13(12):17057–66.
[56] Vardy AE, Brown JM. Transient, turbulent, smooth pipe friction. J Hydraulic Res 1995;33(4):435–56.
[57] Vítkovskỳ JP, Bergant A, Simpson AR, Lambert MF. Systematic evaluation of one-dimensional unsteady friction models in simple pipelines. J Hydraul Eng 2006;132(7):696–708.
[58] Vítkovskỳ JP, Simpson AR, Lambert MF. Leak detection and calibration using transients and genetic algorithms. J Water Resour Plann Manage 2000;126(4):262–5.
[59] Wang X-J, Lambert MF, Simpson AR, Liggett JA, Vítkovskỳ JP. Leak detection in pipelines using the damping of fluid transients. J Hydraul Eng 2002;128(7):697–711.
[60] Wood DJ, Lingireddy S, Boulos PF, Karney BW, McPherson DL. Numerical methods for modeling transient flow in distribution systems. J Am Water Work Assoc 2005;97(7):104–15.
[61] Wylie EB, Streeter VL, Suo L. Fluid transients in systems. 1 Prentice Hall Englewood Cliffs, NJ; 1993.
[62] Xing L, Sela L. Unsteady pressure patterns discovery from high-frequency sensing in water distribution systems. Water Res 2019;158:291–300.
[63] Xu X, Karney B. An overview of transient fault detection techniques. Modeling and monitoring of pipelines and networks. Springer; 2017. p. 13–37.
[64] Zarzycki Z, Kudźma S, Urbanowicz K. Improved method for simulating transients of turbulent pipe flow. J Theoretic Appl Mech 2011;49(1):135–58.
[65] Zhang C, Gong J, Simpson AR, Zecchin AC, Lambert MF. Impedance estimation along pipelines by generalized reconstructive method of characteristics for pipeline condition assessment. J Hydraul Eng 2019;145(4):04019010.
[66] Zhao M, Ghidaoui MS. Godunov-type solutions for water hammer flows. J Hydraul Eng 2004;130(4):341–8.
[67] Zielke W. Frequency-dependent friction in transient pipe flow. J Basic Eng 1968;90(1):109–15.