IWA PUBLISHING

doi: 10.2166/9781789062380_0215

Chapter 9

Hydraulic transients in pipe systems

Juneseok Lee[1], Lu Xing[2] and Lina Sela[3]*

[1]*Department of Civil and Environmental Engineering, Manhattan College, Riverdale, NY*
[2]*Data Scientist, Xylem, Rye Brook, NY*
[3]*Department of Civil and Environmental Engineering, University of Texas, Austin*
**Corresponding author: juneseok.lee@manhattan.edu*

LEARNING OBJECTIVES

At the end of this chapter, you will be able to:

(1) Explain hydraulic transients in pressurized pipe systems.
(2) Use the hydraulic transients governing equations, codes and run for very simple cases.
(3) Run TSNet for simple water distribution systems.
(4) Assess and interpret modeling results.

9.1 INTRODUCTION

Many water utilities have in-house hydraulic modeling capacities to analyze their systems in terms of planning, design, operations, and management. However, many of the modeling efforts are geared toward or limited to steady state or extended period simulations, which assume that the water column is completely incompressible, and that pipe materials are not elastic. Clearly, the mass continuity and energy equations neglect to explain rapid changes that should be described by momentum equations (i.e., transient pressure waves generated due to sudden changes in flow). As is well known, the resulting pressure can result in pipe bursts and structural damage to other critical appurtenances. In addition, low flow due to transients can induce contamination intrusion in the systems (Lee, 2008; Lee *et al.*, 2012).

For transient flow analysis, Joukowski's equation is the most fundamental theory that is still used as a rough check for a head (H) change calculation:

$$\Delta H = -\frac{a}{g} \cdot \Delta V \tag{9.1}$$

where a is pressure wave speed, g is gravitational acceleration and V is the velocity. The model's assumptions are: (i) no friction in the pipe and (ii) no wave reflections in the system. In other words, there is no interaction among boundary conditions in the system. Here, we are presenting 1D classic water hammer equations (Wylie & Streeter, 1993) as follows:

Editor(s): Juneseok Lee and Jonathan Keck

Continuity equation:

$$\frac{\partial p}{\partial t} + V\frac{\partial p}{\partial x} + c^2\rho\frac{\partial V}{\partial x} = 0 \tag{9.2}$$

Momentum equation:

$$\frac{\partial V}{\partial t} + V\frac{\partial V}{\partial x} + \frac{1}{\rho}\frac{\partial p}{\partial x} + g\sin\alpha + \frac{f}{2D}V|V| = 0 \tag{9.3}$$

where p is pressure, V is velocity, c is wave speed, ρ is density, g is acceleration due to gravity, α is angle of inclination of pipe, f is friction factor, D is diameter, x is spatial dimension, and t is time.

These two equations are solved for a pipe network that incorporates suitable interior boundary conditions for appurtenances such as valves and junctions along the pipelines and external boundary conditions such as reservoirs and tanks. The solution of these equations yields the pressure/head, p *or* H (x, t) and velocity/flow rate, V *or* Q (x, t) as functions of spatial dimension x (taken along the length of the pipelines) and time t. The pressure can be highly positive and negative for hydraulic transients, and the velocity can be negative, indicating flow reversal. In the following, we provide a general overview of the mumerical schemes followed by the hydraulic transients computation tool, TSNet.

9.2 NUMERICAL METHOD CONSIDERING INITIAL AND BOUNDARY CONDITIONS

The governing equations for hydraulic transients are nonlinear hyperbolic Partial Differential Equations (PDE), so a closed-form solution is not available. Numerical methods must be used to solve these governing equations. There are several numerical methods such as, but not limited to, Methods of Characteristics (MOC), implicit Finite Difference (FD) method, and explicit FD schemes. In this article, we introduce an introductory explicit FD scheme, McCormack's method, which should be helpful for understanding why certain types of data are needed to run a specific hydraulic transients modeling package. The results obtained from McCormack's method are known to be satisfactory for many flow applications (Anderson, 1995). Please refer to Chaudhry (1987), Karney and McInnis (1990) and Wylie and Streeter (1993) for details on backgrounds as well as other numerical methods.

In McCormack's FD, the PDE is transformed into FDM (Finite Difference Method), such that the unknown conditions at a point at the end of a time step are expressed in terms of the known conditions at the beginning of the time step. Figure 9.1 shows the general schematic of the 1D explicit FD scheme. We solve unknown time level variables (H, Q) based on known time levels (H, Q). McCormack's explicit FD scheme is composed of a predictor and corrector step. One-sided FD is used for the spatial derivatives in each of these steps. First, forward FD is used in the predictor, and backward FD is used in the corrector part. Alternatively, backward FD is adopted in the predictor, and forward FD is used in the corrector part. Each alternative takes turns as the time step increases. Note that we are using simplified governing equations using (H, Q) here as follows:

$$\frac{\partial H}{\partial t} + \frac{a^2}{gA}\frac{\partial Q}{\partial x} = 0 \tag{9.4}$$

$$\frac{\partial Q}{\partial t} + gA\frac{\partial H}{\partial x} + RQ|Q| = 0 \tag{9.5}$$

where H is head, Q is flow rate, a is wave speed, g is acceleration due to gravity, f is friction factor, $R = f\Delta t/2D$, D is diameter, x is 1D spatial dimension, and t is time. See the formulations below. *: predicted values, i: space node and j: corresponding time step. H and Q's values are assumed to be known at all nodes at the time j level. We are solving for the time ($j + 1$) level (Figure 9.1).

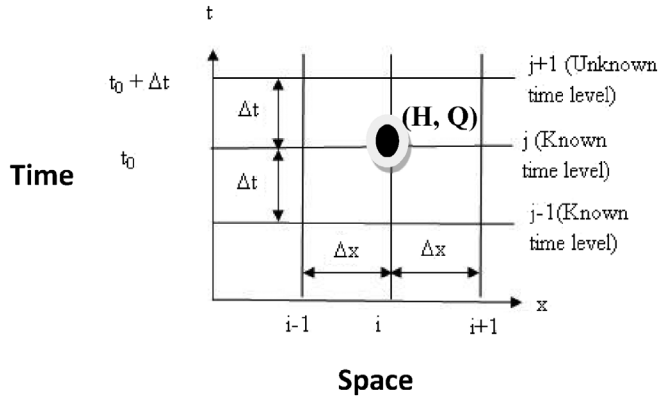

Figure 9.1 1D explicit FD scheme.

In alternative 1, the predictor part is:

$$\begin{aligned} H_i^* &= H_i^j - \frac{\Delta t}{\Delta x}\frac{a^2}{gA}\left(Q_{i+1}^j - Q_i^j\right), \\ Q_i^* &= Q_i^j - \frac{\Delta t}{\Delta x} gA\left(H_{i+1}^j - H_i^j\right) - RQ_i^j\left|Q_i^j\right|\cdot\Delta t, \quad (i = 1,\ 2,\ldots,\ n) \end{aligned} \tag{9.6}$$

The corrector part is:

$$\begin{aligned} H_i^{j+1} &= \frac{1}{2}\left(H_i^j + H_i^* - \frac{\Delta t}{\Delta x}\frac{a^2}{gA}\left(Q_i^* - Q_{i-1}^*\right)\right), \\ Q_i^{j+1} &= \frac{1}{2}\left(Q_i^j + Q_i^* - \frac{\Delta t}{\Delta x} gA\left(H_i^* - H_{i-1}^*\right) - RQ_i^*\left|Q_i^*\right|\cdot\Delta t\right), \quad (i = 2,\ldots,n+1) \end{aligned} \tag{9.7}$$

In alternative 2, the predictor and corrector parts are:
Predictor part:

$$\begin{aligned} H_i^* &= H_i^j - \frac{\Delta t}{\Delta x}\frac{a^2}{gA}\left(Q_i^j - Q_{i-1}^j\right), \\ Q_i^* &= Q_i^j - \frac{\Delta t}{\Delta x} gA\left(H_i^j - H_{i-1}^j\right) - RQ_i^j\left|Q_i^j\right|\cdot\Delta t, \quad (i = 2,\ldots,n+1) \end{aligned} \tag{9.8}$$

Corrector part:

$$\begin{aligned} H_i^{j+1} &= \frac{1}{2}\left(H_i^j + H_i^* - \frac{\Delta t}{\Delta x}\frac{a^2}{gA}\left(Q_{i+1}^* - Q_i^*\right)\right), \\ Q_i^{j+1} &= \frac{1}{2}\left(Q_i^j + Q_i^* - \frac{\Delta t}{\Delta x} gA\left(H_{i+1}^* - H_i^*\right) - RQ_i^*\left|Q_i^*\right|\cdot\Delta t\right), \quad (i = 1,2,\ldots,n) \end{aligned} \tag{9.9}$$

As mentioned, each alternative takes turns as the time step (j) increases. These formulations work for internal nodes, but we will have to consider separately for the boundary conditions. From the governing Equation (9.4), we will consider characteristic boundaries.

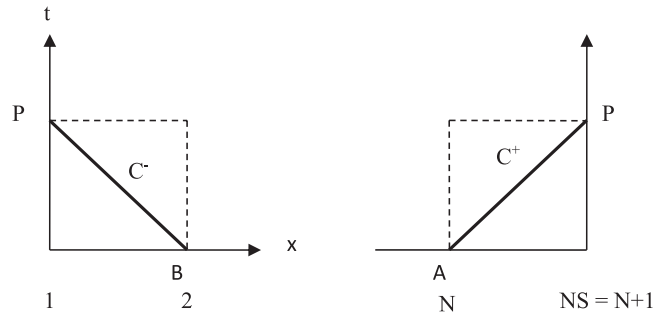

Figure 9.2 Boundary characteristic (upstream and downstream).

Multiplying Equation (9.4) by η and adding it to Equation (9.5), then we have:

$$\left(\frac{\partial H}{\partial t}+\eta gA\frac{\partial H}{\partial x}\right)+\eta\left(\frac{\partial Q}{\partial t}+\frac{a^2}{\eta gA}\frac{\partial Q}{\partial x}\right)+\eta RQ|Q|=0$$
$$\text{Let,}\quad \eta gA=\frac{dx}{dt}=\frac{a^2}{\eta gA},\quad \text{so}\,\eta=\pm\frac{a}{gA} \tag{9.10}$$

when

$$\lambda^+=\frac{dx}{dt}=a,\quad C+:\left(\frac{\partial H}{\partial t}+\lambda^+\frac{\partial H^+}{\partial x}\right)+\frac{a}{gA}\left(\frac{\partial Q}{\partial t}+\lambda^+\frac{\partial Q^+}{\partial x}\right)+\frac{aR}{gA}Q|Q|=0 \tag{9.11}$$

when

$$\lambda^-=\frac{dx}{dt}=-a,\quad C-:\left(\frac{\partial H}{\partial t}+\lambda^-\frac{\partial H^-}{\partial x}\right)-\frac{a}{gA}\left(\frac{\partial Q}{\partial t}+\lambda^-\frac{\partial Q^-}{\partial x}\right)-\frac{aR}{gA}Q|Q|=0, \tag{9.12}$$

At the boundary conditions, equations are solved with the condition imposed by the boundary. The characteristic of the boundary condition is shown in Figure 9.2. For the upstream boundary, the C– characteristic line is valid, and C+ is for downstream. These are used to depict the complete water phenomena. Each boundary condition is solved independently of the interior points' calculation and another end of the boundary. In this chapter, we are presenting several representative boundary conditions. Please refer to Chaudhry (1987) and Wylie and Streeter (1993) for more details.

9.2.1 Reservoir

During a short-period transient event, the upstream reservoir's hydraulic grade line elevation is assumed to be constant:

$$H_1=H_R \tag{9.13}$$

where H_R is hydraulic grade line above the reference datum, and H_1 is head value at the upstream section at point P. With $C-$ equation and $H_1=H_R$, Q can be obtained at the boundary. So, H and Q are obtained for the next time level $(j+1)$.

9.2.2 Junctions

A general junction with pipeline, non-pipeline elements (valve), and nodal inflow are shown in Figure 9.3. A continuity equation can be written at the junction. At any instant, the sum of the inflow is zero:

$$\sum Q_{in}=\sum Q_p+\sum Q_e+Q_n=0 \tag{9.14}$$

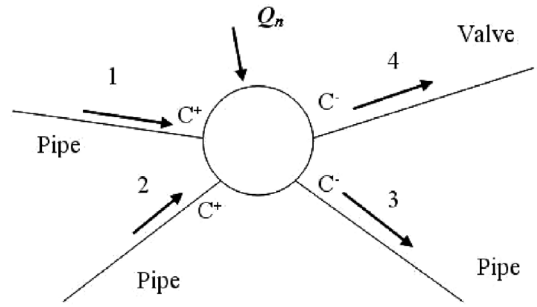

Figure 9.3 General junction.

where $\sum Q_{in}$ is the sum of the inflow (added to zero at any instant), $\sum Q_e$ is the sum of all instantaneous non-pipe flows, $\sum Q_p$ is the sum of all instantaneous pipe flows, and Q_n is a nodal flow. When minor losses are neglected at the junction, the energy equation can be written for each junction element:

$$H = H_{1,NS} = H_{2,NS} = H_{3,1} = H_{4,1} \tag{9.15}$$

Using the compatibility equations, $Q_{1,NS,}$ $Q_{2,NS,}$ $Q_{3,1}$, and $Q_{4,1}$ are obtained.

Figure 9.4 shows the valve located between two pipelines, where A and B show the interconnecting junctions on both sides of the valve. It is assumed that the inertia effects are neglected in the steady-state orifice equation, and the volume of fluid stored inside the valve is constant. For positive flow, $H_{1,NS} = H_A$ and $H_{2,1} = H_B$. The orifice equation is:

$$Q_{1,NS} = Q_{2,1} = Q_v = \frac{Q_0 \tau}{\sqrt{H_0}} \sqrt{H_A - H_B} \tag{9.16}$$

where H_0 is steady-state HGL drop across the valve with the flow of Q_0 ($t = \frac{C_d A_G}{(C_d A_G)_0} = \sqrt{\frac{K_0}{K}} = 1$: dimensionless valve opening). When combined with the compatibility equation, Q_v can be obtained. Also, for reversal flows:

$$Q_{1,NS} = Q_{2,1} = Q_v = -\frac{Q_0 \tau}{\sqrt{H_0}} \sqrt{H_A - H_B} \tag{9.17}$$

Combined with the compatibility equation, Q_v can be obtained.

While working on the numerical methods or solvers, you should be aware of a few critical issues, such as convergence and stability. We are introducing fundamental concepts as follows:

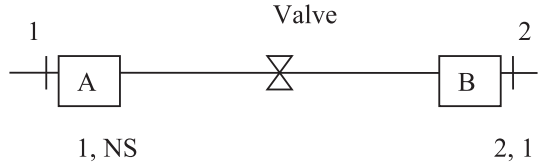

Figure 9.4 Valve located in-line.

9.2.3 Discretization error

Say, U (x, t) is the exact solution of the PDE, and u (x, t) is the finite difference equation's exact solution. Then, (U–u) is called *discretization error*. This is introduced when replacing the PDE with finite difference approximation.

9.2.4 Truncation error

$F_i^j(u) = 0$ is the finite difference equation at grid point $i\Delta x$ and $j\Delta t$, where i and j are the number of grid points in the x and t directions. Substituting the exact solution of PDE U (x, t) into the FD approximation equation, then $F_i^j(U)$ is the local truncation error at ($i\Delta x$, $j\Delta t$).

9.2.5 Consistency

FDM is consistent when the truncation error tends to zero as Δx and Δt approach zero.

9.2.6 Convergence

FDM is said to be convergent as the exact solution of FDM u approaches the exact solution of PDE U, as both Δx and Δt approach zero. It is not easy to directly prove convergence. However, FDM is convergent if the scheme is proved to be *consistent and stable*.

9.2.7 Stability

When the computations are performed to an infinite number of significant figures (decimal digits), the solution u (x, t) of the FDM will be exact. However, even in computers nowadays, round-off errors are introduced at each time step. So, the numerical solution we obtain is different from the exact solution. In some cases, round off errors are amplified, decrease, or stay the same. The scheme is s*table* when the amplification of the round off errors is bounded for all sections as time goes infinity. Unstable schemes result in very rapidly growing error in a few time steps. So, stability conditions must be satisfied.

9.2.8 CFL (Courant Friedrich Lewy) stability condition

$\Delta x \geq a\Delta t$, the courant number is defined as $C_N = (a/\Delta x/\Delta t) = (a\Delta t/\Delta x)$ and $C_N \leq 1$. This stability criterion applies only to linear equations (when the friction term is small). Even if the CFL condition is met, the scheme may become unstable: when the friction term is large (e.g., large friction factor, large time step, a large change in discharge, or small conduit diameter: according to the friction loss equation). The stability of the FDM may be done using von Neumann theory. In this approach, errors are expressed in a Fourier series simultaneously (for linear equations). The scheme is said to be stable if the errors decay as time increases. In the following, the pseudocode is shown for those who are interested in coding (Figure 9.5).

9.2.9 Example

For the given schematics, compare the pressure transient behavior when we shut the valve instantaneously (closing time = 0 at $t = 0$): Explicit scheme (McCormack's scheme) (Figure 9.6).

9.2.9.1 Given

Darcy Weisbach friction coefficient = 0.02; Time step = 0.02 sec; Distance step = 100'; Flow rate = 19.5 cfs.

9.2.9.2 Find

Pressure variation at $x = 500$' (from the valve; at the center of the pipe).

```
# Supply initial data
old =0;
new=1;
time=0;
Time entered by the user;

Loop on m from 0 to M      % set initial data
V(old,m)=u0(x(m));
End of loop on m

Loop for time <TIME_MAX
        Time=time+k                % time being computed
        n_time=n_time+1
        v(new,0)=beta(time)                % set boundary conditions
        loop on m from 1 to M-1
                v(new,m)=.... % McCormack scheme
        End of loop on m
        v(new, M)=v(new,M-1)      % apply boundary condition

        old=new                    % reset for the next time step
        new=mod(n_time,2)
end of loop on time
```

Figure 9.5 Pseudocode for running hydraulic transients.

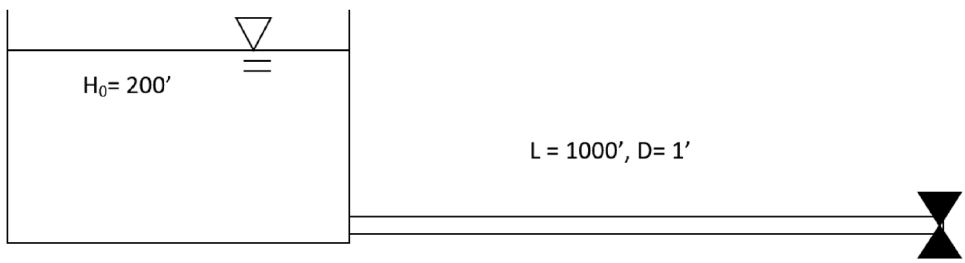

Figure 9.6 Reservoir problem.

9.2.9.3 Solution

MATLAB was used for running explicit scheme and is plotted in Excel (see Figure 9.7). It shows clear fluctuations in head values. The code is included below.

9.3 OTHER PHENOMENON OF INTERESTS | CAVITATION AND COLUMN SEPARATION

As mentioned previously, 'water hammer' is a transient flow phenomenon introduced in pipe flow systems by suddenly obstructing the flow. Consequently, there is a pressure rise and fall, and the

```
RESERVOIR PROBLEM
clear all;
delT = 0.02; %time step
h = 100;
M = 1000/h;
t_final = input('Enter end of time in seconds:');
time_step_N = t_final/delT;
a = 4000; %FT/SEC celerity
A = 0.785398; %FT2 area of pipe: PI/4*D⊥2, D = 1'
g = 32.2; %FT/sec2 gravitational acceleration
f = 0.02; %darcy-weisbach friction coeff.
D = 1;% FT; Diameter of the conduit
AA = delT*(a⊥2)/(h*g*A);%constant AA
BB = delT*g*A/h;% constant BB
CN = a*delT/h; % Courant Number should be less than 1
R = f/(2*D*A);
Hold = zeros(1,M + 1);
Qold = zeros(1,M + 1);
for m = 1:M + 1%set initial data
Hold(m) = 200-(20)*(m−1);
Qold(m) = 19.5;
end
Hstar = zeros(1,M + 1);
Qstar = zeros(1,M + 1);
Hnew = zeros(1,M + 1);
Qnew = zeros(1,M + 1);
% CP = Qold(M) + g*A/a*Hold(M)-R*delT*Qold(M)*abs(Qold(M));
% CN = Qold(2)-g*A/a*Hold(2)-R*delT*Qold(2)*abs(Qold(2));
% CA = g*A/a;
%----------------------------- McCommas Scheme----------
for j = 1:time_step_N %time step
CP = Qold(M) + g*A/a*Hold(M)-R*delT*Qold(M)*abs(Qold(M));
CN = Qold(2)-g*A/a*Hold(2)-R*delT*Qold(2)*abs(Qold(2));
CA = g*A/a;
alterna = mod(j,2);
if alterna = =1
for i = 1:M
Hstar(i) = Hold(i)-AA*(Qold(i + 1)-Qold(i));
Qstar(i) = Qold(i)-BB*(Hold(i + 1)-Hold(i))-R*abs(Qold(i))*Qold(i)*delT;
end
```

Continued

```
for i = 2:M
Hnew(i) = 0.5*(Hold(i) + Hstar(i)-AA*(Qstar(i)-Qstar(i−1)));
Qnew(i) = 0.5*(Qold(i) + Qstar(i)-BB*(Hstar(i)-Hstar(i−1))-R*abs(Qstar(i))*Qstar(i)*delT);
end
Hnew(1) = 200; %set the left B.C.
Qnew(M + 1) = 0; %set the right B.C.
Qnew(1) = CN + CA*Hnew(1); %set the left B.C.
Hnew(M + 1) = CP/CA; %set the right B.C.
else
for i = 2:M + 1
Hstar(i) = Hold(i)-AA*(Qold(i)-Qold(i−1));
Qstar(i) = Qold(i)-BB*(Hold(i)-Hold(i−1))-R*abs(Qold(i))*Qold(i)*delT;
end
for i = 2:M
Hnew(i) = 0.5*(Hold(i) + Hstar(i)-AA*(Qstar(i + 1)-Qstar(i)));
Qnew(i) = 0.5*(Qold(i) + Qstar(i)-BB*(Hstar(i + 1)-Hstar(i))-R*abs(Qstar(i))*Qstar(i)*delT);
end
Hnew(1) = 200; %set the left B.C.
Qnew(M + 1) = 0; %set the right B.C.
Qnew(1) = CN + CA*Hnew(1); %set the left B.C.
Hnew(M + 1) = CP/CA; %set the right B.C.
end
% for mm = 1:(time_step_N) + 1
% time(mm) = mm*delT;
% end
%
% figure (1)
% plot(time,Hnew(1),'-g',time,Hnew(10),'-r', time, Hnew(21),'-r')
H1(j) = Hnew(4);
H5(j) = Hnew(5);
H10(j) = Hnew(6);
AAA = [H1;H5;H10];
for kk = 1:M + 1
Hold(kk) = Hnew(kk);
Qold(kk) = Qnew(kk);
end
end
```

pattern is repeated until the transient energy decays. Typical vapor pressures of water (10–40°C) range from 0.012 to 0.073 atmosphere; the total dissolved gas pressure of natural water is typically in the range of 0.8–1.2. When the fluid pressure drops below the constituent gases' saturation pressure, bubbles comprised of dissolved gases are formed, which is known as *gaseous cavitation* (Lee, 2008).

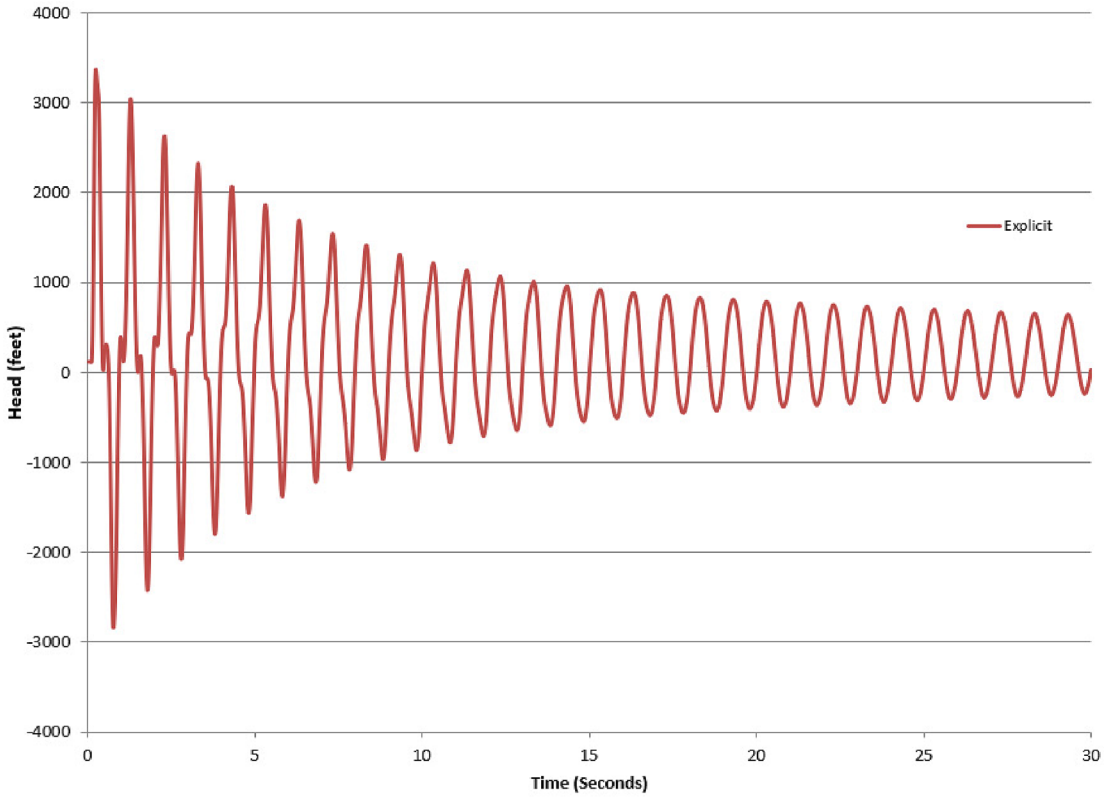

Figure 9.7 Reservoir results comparison.

When the pressure drops below the vapor pressure, vapor cavities are created in liquid by phase transformation, called vaporous cavitation.

In a liquid, gas may have two forms: dissolved and free gas. The dissolved gas is invisible in liquid and does not increase its volume and compressibility noticeably. However, free gas (or '*entrained gas*') is dispersed in the liquid as bubbles that may make the liquid look turbid. The liquid which is not entirely degassed will usually contain some entrained air in the form of microscopic or submicroscopic bubbles, either in the bulk of the liquid or near solid contaminants or near the container wall. Fluid mixtures can be categorized into five phenomena (Shu, 2003): (1) fully degassed fluid, (2) thoroughly degassed liquid with vapor, (3) liquid with dissolved gas, (4) liquid with dissolved and free gas, (5) liquid with dissolved gas, free gas, and vapor.

Vaporous cavitation can be present in cases (2) and (5). The cavity may become so large as to fill the entire or partial section of the pipe (known as *air pocket*) and divide the liquid into two columns in vertical pipes or pipes with steep slopes known as *column separation*. In horizontal pipes or mild slopes, however, a thin cavity is aggregated to the top of the pipe and extends over a long distance in the pipe (i.e. *cavitating flow*). So, the vapor cavities may be physically dispersed homogeneously or collected into a single or multiple void space, or a combination of the two phenomena. For gaseous cavitation, both liquid with dissolved and free gases can be observed. Free gas is distributed throughout the liquid in a homogeneous mix or lumped as pockets of free gas, trapped along the pipe wall, in pipe

joints, in surface roughness, and crevices. In this article, we introduce modeling concepts for vaporous cavitation.

9.3.1 Discrete vapor cavity model (DVCM)

DVCM is assumed to have a vapor cavity quantity concentrated at each computational section (see Figure 9.6). This model is the most commonly used model for column separation (Bergant *et al.*, 2006). Cavities are allowed to form at any computational grid point when the computed pressure is below the vapor pressure. The pressure wave speed is assumed to be constant between the vapor. Also, the absolute pressure in the vapor is set equal to the vapor pressure:

$$p^* = p_v^* \tag{9.18}$$

The upstream and downstream discharges at a cavity are computed from *classic water hammer equations* Equations (9.2), (9.3), or (9.4), and the vapor cavity volume (V) can be obtained as below:

$$\text{Continuity equation for vapor cavity}: \frac{\partial \forall}{\partial t} = Q_2 - Q_1 \tag{9.19}$$

where $\forall$ is vapor volume, Q_1 is downstream flow rate, and Q_2 is upstream flow rate (Figure 9.8).

It is assumed that mass transfer during cavitation is ignored. Also, flow rate discontinuity is assumed at each computational node. So, there will be two predicted values of flow rates (Figure 9.6). This continuity equation for the vapor volume V is applied at each computing section. The liquid flow in the pipe is instantaneously and entirely separated by its vapor phase when the cavity is formed. However, in reality, when a cavity is formed in a section of the pipe, it usually expands and propagates in the direction of flow as an elongated bubble. So, this formulation's phenomenon does not necessarily occur in a horizontal or near horizontal pipe (Shu, 2003). It is known that DVCM may generate unrealistic pressure spikes with a multi-cavity collapse. However, the oscillations may be suppressed by assuming small gas volumes in each grid (Wylie & Streeter, 1993).

Column separation or cavitating flow is caused by the negative or rarefaction waves passing through the pipelines. When these waves meet a boundary such as a reservoir, they are reflected as positive waves. This raises the local pressure higher (than total dissolved gas pressure or vapor pressure). They can reduce the cavity's size during column separation and compress the bubbles in the cavitating flow region. When the cavities collapse or when the separated column rejoins, very high pressure that may burst the pipes are generated. According to Kranenburg (1974), the inclusion of gas release had no effect when only cavitating flow occurs, but the gas release effect is large when column separation occurred with the cavitating flow. The implosion of gas or air bubbles in pressurized conduits introduces extra shock waves, namely, the intense pressure wave in water produced by explosions that create violent

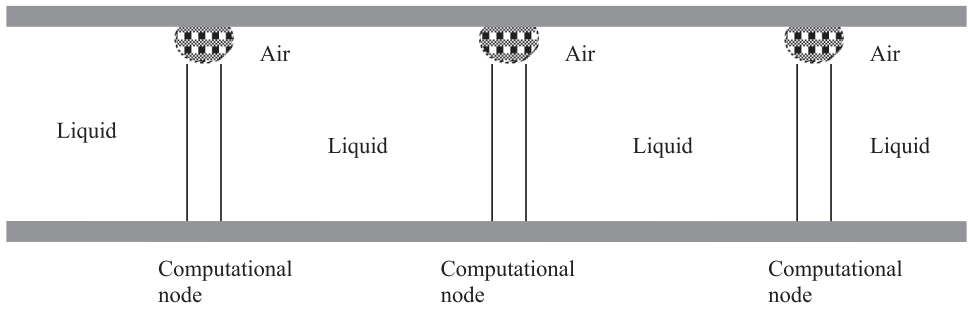

Figure 9.8 Discrete vapor cavity model sketch (modified from Bergant *et al.*, 2006).

pressure changes. Shock waves travel faster than the sound wave, and their speed increases as the amplitude is raised; however, the intensity of a shock wave also decreases faster than that of a sound wave. This is because some of the shock wave energy is dissipated due to the heat transfer in the water in which it travels. In this vein, steep waves or shocks may be generated at different boundaries due to abrupt changes in the discharge. However, damages from vaporous cavitation decrease with higher gas content as dissolved gas has a cushioning effect on implosion.

9.3.2 Short term pressure peaks following cavity collapse

According to Walsh (1964), the maximum possible pressure rise following the collapse of the first cavity at an upstream valve can be expressed as:

$$\Delta H_{\max} = \frac{a}{g}|V_f| + 2H_{RV} \tag{9.20}$$

where V_f is the velocity of the liquid column at the valve just before cavity collapse, and H_{RV} is the difference of reservoir head and vapor head at the valve. Wylie and Streeter (1993) also showed the pressure after collapsing of the first cavity for the case of instantaneous closure of upstream and downstream valves:

$$\Delta H_{\max} = \Delta H + 2\Delta H_{in} \tag{9.21}$$

Thus, the maximum pressure can be more than two times the Joukowski value.

9.4 TRANSIENT SIMULATIONS IN WATER DISTRIBUTION NETWORKS: TSNet

9.4.1 TSNet

Transient simulation in water networks (TSNet) is a Python package designed to perform hydraulic transients simulation in water distribution networks (Xing & Sela, 2020). TSNet adopts the Method of Characteristics (MOC) for solving the system of partial differential equations governing the unsteady hydraulics. The main capabilities of TSNet are: (1) allowing the user to select the computational time step and control numerical accuracy and computational complexity, (2) simulating transient system responses to the operation of valves and pumps, (3) simulating transient system response to background leakage and pipe bursts, (4) simulating open and closed surge tanks for controlling transient response, (5) simulating steady, quasi-steady, and unsteady friction models, (6) simulating instantaneous nodal demand changes using demand-pulse model, and (7) visualizing and postprocessing simulation results. In this section, we will see examples of running TSNet to simulate the transient events under different scenarios. For additional examples, see TSNet documentation (Xing & Sela, 2021a).

9.4.2 Use of Python, Spyder and Anaconda

Before looking into TSNet, it is beneficial to learn some Python basics. There are many useful resources out there, such as Python Programming and Numerical Methods: A Guide for Engineers and Scientists (Kong *et al.*, 2020).

In this section, we will be using Spyder as our Python environment. Spyder is a free and open-source integrated development environment (IDE) for scientific programming in the Python language (Spyder, 2021). Spyder is included by default in the Anaconda Python distribution (Anaconda, 2021), which comes with everything you need to get started in an all-in-one package. To download Anaconda, please visit the Anaconda website (Anaconda, 2021) and download the installer for your platform, that is Windows, Mac, or Linux. After installation, you can open Anaconda and should see something like Figure 9.9.

To run Spyder after installing it with Anaconda, you can open Anaconda Navigator, scroll to Spyder under Home, and click Launch. Then, you should see the Spyder interface as in Figure 9.10. The default layout of Sypder has three windows. *Editor* (left) is where you will be doing most of your

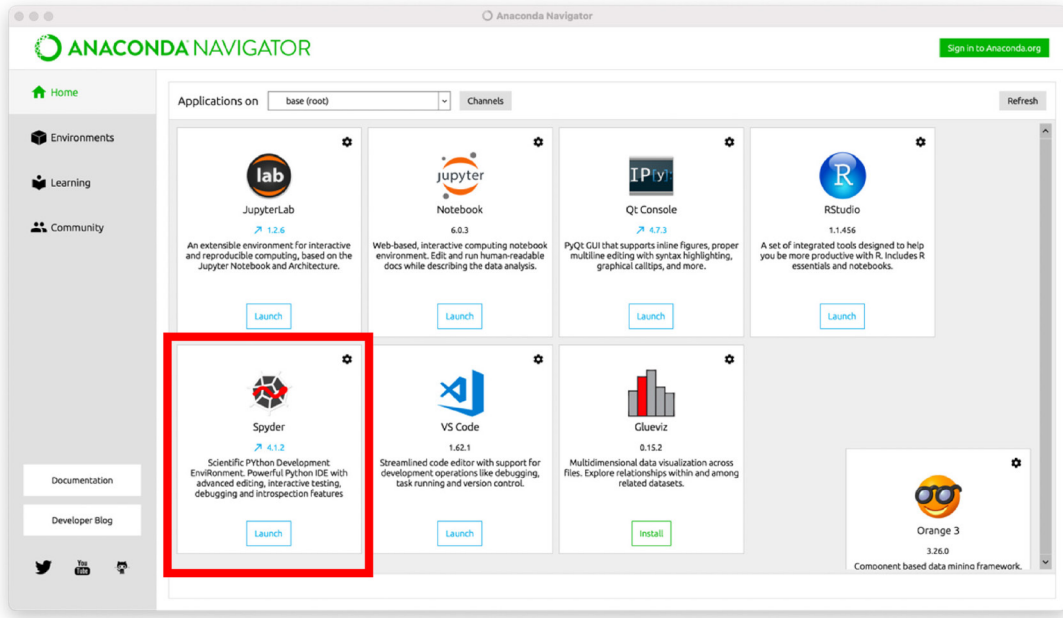

Figure 9.9 Anaconda navigator interface.

coding. In this window, you can create, edit, and save scripts and functions. You will run these scripts by selecting the green arrow at the top of the screen in the tool bar or by pressing F5. *Help* (upper right) is where you search for information on various functions within Python. For the upper right window, you can also choose to display *Variable Explorer*, where you can find all variables you create or import into Python, *Plots*, where you can view all figure outputs, and *Files*, where you can find and open various files under the current directory. On the lower right, *Console* is the main window for executing commands and viewing results. Press enter key to execute a command. Any code output and errors will be displayed in the Console. Then we can download the TSNet Python package by typing '! Pip install tsnet' in the Console as shown in Figure 9.10. This command will automatically download and install TSNet and other packages that TSNet depend on (e.g., Numpy, Matplotlib, WNTR, etc.).

9.4.3 Example application

Now that we have installed TSNet and have a basic understanding of how Python works, we can now move on to see how to use TSNet to simulate transient events. We will demonstrate how to use TSNet using an example network shown in Figure 9.11a, which is comprised of 113 pipes, 91 junctions two pumps, two reservoirs, three tanks, and one valve. The information of this network is stored in an EPANET INP file, Tnet2.inp, as shown in Figure 9.11b. The INP file and the codes that we will demonstrate here can be downloaded from Xing and Sela (2021b).

After installing TSNet, the main steps in setting up and running the transient model are: (1) read water network model input (EPANET INP) file and create the corresponding transient model; (2) set up a transient model by defining additional transient-related features, such as wave speed and time step; (3) define a transient scenario, such as operational changes in valves and pumps, and pipe bursts; (4) define if the system includes background leak conditions and obtain the initial conditions by conducting a steady-state simulation; (5) perform transient simulation using method of characteristics

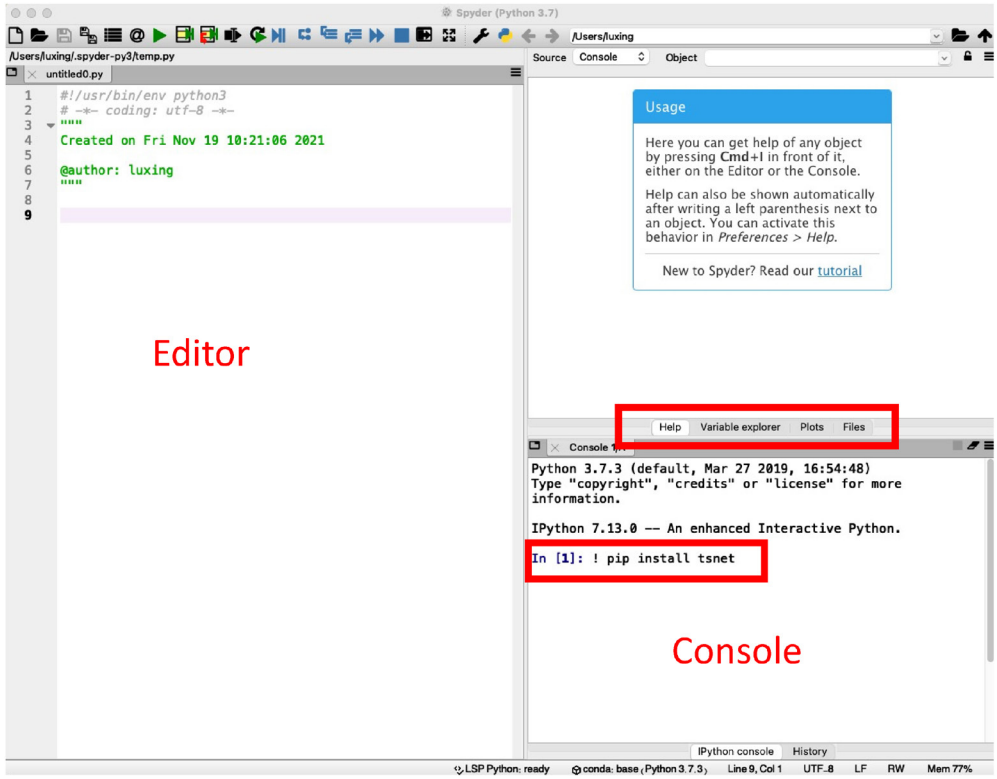

Figure 9.10 Spyder IDE interface.

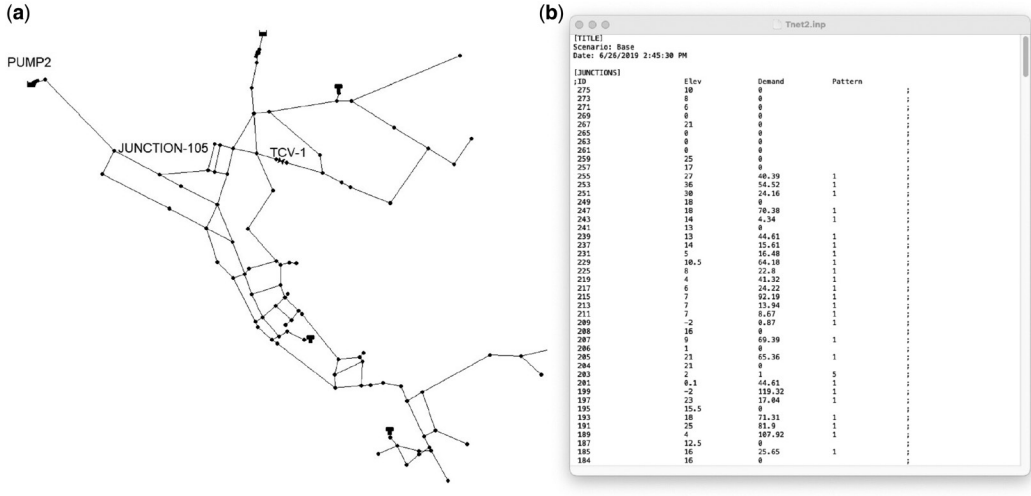

Figure 9.11 Example network: (a) network topology and (b) screenshot of Tnet2.inp.

```
1  import tsnet
2  inp_file = 'networks/Tnet2.inp'
3  tm = tsnet.network.TransientModel(inp_file)
4  # Set wavespeed and time
5  tm.set_wavespeed(1200.) # [m/s]
6  tf = 60 # simulation duration[s]
7  dt = 0.01 # time step [s]
8  tm.set_time(tf, dt)
```

Figure 9.12 Create transient model (lines 1–3) and define wave speed and time step (lines 5–8).

(MOC) (Wylie & Streeter, 1993); and (6) obtain and visualize flow and pressure results. The following sections will detail how to use TSNet to set up the transient model and simulate various transient events, including valve closure, pump shutdown, and pipe bursts.

9.4.4 Create and set up a transient model

To use TSNet for transient simulations, we first need to import the TSNet package to enable TSNet APIs (line 1), and read the EPANET INP file to import the network information and create the transient model (lines 2–3), as shown in Figure 9.12. In this example, the EPANET INP file is Tnet2.inp, which locates in the networks folder. The INP file contains all the information about network elements, for example junctions, pipes, reservoirs, tanks, pumps, and valves, as well as their characteristics, such as elevation and demands at junctions and pipe diameter, length, and roughness coefficient. More information about INP files can be found in Rossman (2000). In addition to the information included in the INP file, we also need to specify the wave speeds for each pipe (line 5), simulation duration and the time step (lines 6–8) as shown in Figure 9.12. These are required for TSNet to define the numerical grid that will be used to solve the equations that model transient hydraulics. More information about defining wave speeds and time steps in transient modeling can be found in Wylie and Streeter (1993). In this example, we assume the wave speeds for all pipes are 1200 m/s, simulation duration is 60 s, and time step is 0.01 s. It should be noted that the codes behind the # sign are comments for explanation purposes and are not executed when running the code.

Type these commands in the Editor window, save and run your script. In the Console window you should see 'Simulation time step 0.01043 s', and in the variable explorer you should see four variables (dt, inp_file, tf, tm). To test that the model was created properly, type the command tm in the Console, and you should see something like <tsnet.network.model.TransientModel at $0 \times 7fb9201db4e0$>. Now that we have created and set up a transient model in TSNet, we can move forward to define different scenarios for transient simulations.

9.4.5 Valve closure

Let us start with a valve closure scenario. Rapidly closing a valve in the system may cause a sudden change of flow rate, and the force resulting from the change in velocity will cause a pressure increase or decrease that may be significantly greater than the normal pressure in the pipeline. This pressure disturbance then propagates in the water network causing further pressure and velocity changes in the distribution system. We can simulate a valve closure event in TSNet by defining the valve closure start time (ts) from the beginning of the simulation, closure duration (tc), valve opening percentage when the operation 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 Figure 9.13, where the solid and dashed lines represent $m = 1$ and $m = 2$, respectively.

In this example, we simulate the closure of TCV-1 (shown in Figure 9.11a), which starts at $ts = 1$ s, and takes $tc = 1$ s to completely close the valve, using the code shown in Figure 9.14.

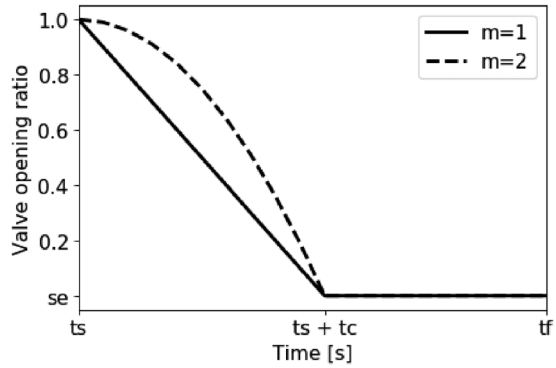

Figure 9.13 Valve closure operating curve.

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

Figure 9.14 Define valve closure.

Once the transient conditions are defined, the transient model is initialized by running a steady state simulation. TSNet uses Water Network Tool for Resilient (WNTR) (Klise *et al.*, 2018) to simulate the steady-state hydraulics, either demand or pressure driven in the event of background leaks. We initialize the hydraulic transients at t0 = 0 and use the demand driven simulator for the steady-state calculation in lines 17–19 in Figure 9.15. Moving forward, we specify the object name for saving results and run the actual transient simulation using lines 21–22 in Figure 9.15.

At the beginning of a transient simulation, TSNet will report the approximate simulation time. The computation progress will also be printed in the Console as the simulation proceeds, as shown in Figure 9.16.

Once the simulation is completed, we can then plot the pressure head at any junction, for example JUNCTION-105 (shown in Figure 9.11), using the command shown in Figure 9.17.

The pressure head at JUNCTION-105 versus time is shown in Figure 9.18. It can be observed that the transient wave induced by the valve closure arrives to JUNCTION-105 in around 7 s and causes a pressure jump of 5 m amplitude. The pressure then fluctuates and returns to the original level after approximately 35 s. The results indicate that the fast valve closure can introduce pressure transients; however, the amplitude of transient is not very significant. We will see much larger transients in the following examples.

```
# Initialize steady state simulation
t0 = 0. # initialize the simulation at 0s
engine = 'DD' # or PPD
tm = tsnet.simulation.Initializer(tm, t0, engine)
# Transient Simulation
results_obj = 'Tnet2' # name of the object for saving results
tm = tsnet.simulation.MOCSimulator(tm,results_obj)
```

Figure 9.15 Initialize (lines 17–19) and running transient simulation (lines 21–22).

```
Simulation time step 0.01043 s
Total Time Step in this simulation 5752
Estimated simulation time 0:06:24.526952
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 %...
```

Figure 9.16 Runtime outputs – computation time and progress.

```
35    tm.plot_node_head(['JUNCTION-105'])
```

Figure 9.17 Plot head results.

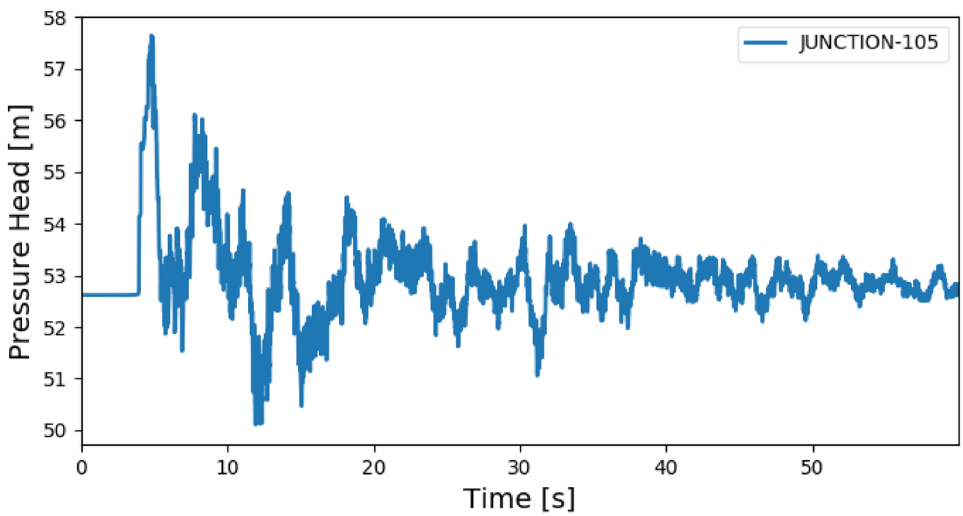

Figure 9.18 Pressure transients at JUNCTION-105 when closing valve TCV-1.

9.4.6 Pump shutdown

Now, let us move on to the pump shutdown scenario. When PUMP2 (see Figure 9.11) is being shut down, the rotating pump impeller begins to decelerate with the pressure dropping on the discharge side of the pump and rising on the suction side. The resultant transient may quickly lead to column separation with ensuing hard-to-predict consequences (Larock *et al.*, 1999). Hence, it is very important to be able to perform transient simulation to determine whether dangerous negative pressures may develop. With TSNet, we can define pump shutdown by specifying how pump rotational speed changes over time using pump shutdown start time (ts), operation duration (tc), the ratio of final pump rotational speed to the original speed (se), and the operating constant (m), which characterizes the

```
import tsnet
inp_file = 'networks/Tnet2.inp'
tm = tsnet.network.TransientModel(inp_file)
# Set wavespeed and time
tm.set_wavespeed(1200.) # [m/s]
tf = 20 # simulation duration[s]
dt = 0.01 # time step [s]
tm.set_time(tf, dt)
# Defien pump operation
tc = 1 # pump closure period
ts = 1 # pump closure start time
se = 0 # end open percentage
m = 1 # closure constant
pump_op = [tc,ts,se,m]
tm.pump_shut_off('PUMP2', pump_op)
# Initialize steady state simulation
t0 = 0. # initialize the simulation at 0s
engine = 'DD' # or PPD
tm = tsnet.simulation.Initializer(tm, t0, engine)
# Transient Simulation
results_obj = 'Tnet2' # name of the object for saving results
tm = tsnet.simulation.MOCSimulator(tm,results_obj)
# Visulize results
tm.plot_node_head(['JUNCTION-105'])
```

Figure 9.19 Transient simulation under the pump shutdown scenario.

shape of the operation curve in the same way as defined for the valve closure. Following the same simulation process as the valve scenario, we can perform the transient simulation and plot the pressure at JUNCTION-105. Figure 9.19 shows the entire code, starting with importing the TSNet package, network INP file, setting up the transient model, initializing and running the transient simulation, and plotting results. Figure 9.20 shows that the pressure wave generated by the pump shut-off reaches JUNCTION-105 after approximately 3 s and introduces a pressure drop with amplitude greater than 12 m. The pressure then fluctuates until reaching a new steady state after approximately 30 s.The

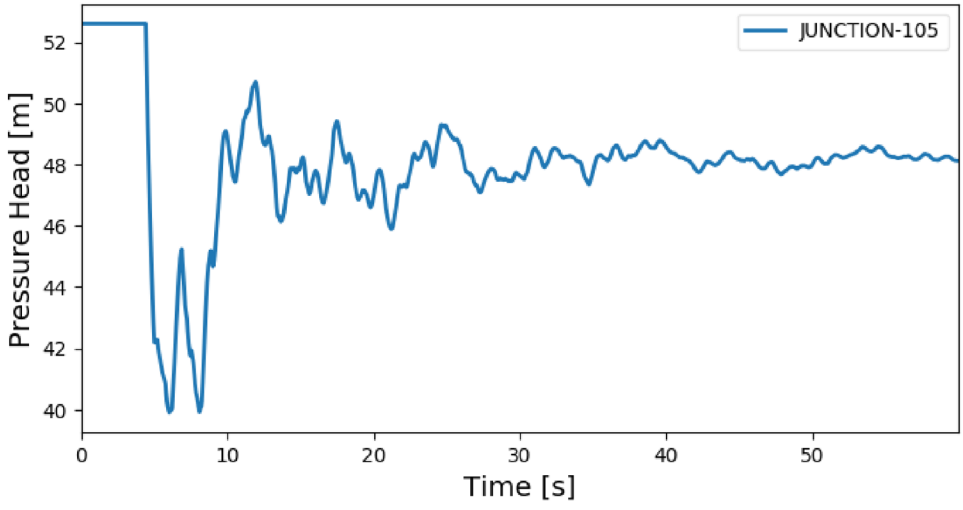

Figure 9.20 Pressure transients at JUNCTION-105 when shutting down PUMP2.

results indicate that pump shutdown, especially when operated quickly, can generate significant transients in the system. Therefore, it is essential to evaluate the impacts of pump operations on the pipelines and design an appropriate procedure to guide pump operations (Boulos *et al.*, 2005).

9.4.7 Pipe burst

Now we move to the simulation of pipe bursts. Pipe bursts, defined as sudden pipe rupture and break events, can introduce sudden and rapid hydraulic transients, which then propagate in the pipe system. TSNet simulates pipe bursts using the orifice equation, which quantifies the pressure-dependent burst discharge where $Q_b(t) = k_b(t)H_b(t)$, where t is time, H_b is the pressure head at the location of the burst, and k_b is the lumped burst coefficient, which changes with time, and aggregates the size of the leak, units, and burst coefficients (Larock *et al.*, 1999). In TSNet, pipe bursts can be specified at junctions. To model the burst occurring along a pipe, the user should introduce a new junction at the location of the burst in the INP file.

An example of simulating a pipe burst using TSNet is shown in Figure 9.21. In this example, a burst event at JUNCTION-105 is simulated by defining the burst location (line 13) and how the lumped burst coefficient (k_b) changes with time. The change in (k_b) is defined by specifying the burst start time (line 10), time for the burst to fully develop (line 11), and final burst coefficient when the burst is fully developed (line 12). In this example the burst starts at ts = 1 s, takes tc = 1 s to fully develop and reach a final burst coefficient of 0.01. The pressure head at JUNCTION-105 is shown in Figure 9.22. It can be seen that the pressure head at JUNCTION-105 decreases by more than 25 m as the burst is developing between 1 and 2 s. The pressure then recovers gradually to a pressure slightly lower than the original level. The results suggest that pipe bursts can introduce significant transient pressure changes in the system, and it is possible to detect pipe bursts by monitoring pressure signals.

9.4.8 Other applications

In addition to valve closure, pump shutdown, and pipe bursts, TSNet can also be used to simulate other transient events, such as valve opening, pump startup, and demand pulses with and without background leaks. Users can also test the effects of including open and closed surge tanks on damping pressure transients. Different friction models, that is steady, quasi-steady, and unsteady friction

```
1  import tsnet
2  inp_file = 'networks/Tnet2.inp'
3  tm = tsnet.network.TransientModel(inp_file)
4  # Set wavespeed and time
5  tm.set_wavespeed(1200.) # [m/s]
6  tf = 20 # simulation duration[s]
7  dt = 0.01 # time step [s]
8  tm.set_time(tf, dt)
9  # Add burst
10 ts = 1 # burst start time
11 tc = 1 # time for burst to fully develop
12 final_burst_coeff = 0.01 # final burst coeff [ m^3/s/(m H20)^(1/2)]
13 tm.add_burst('JUNCTION-105', ts, tc, final_burst_coeff)
14 # Initialize steady state simulation
15 t0 = 0. # initialize the simulation at 0s
16 engine = 'DD' # or PPD
17 tm = tsnet.simulation.Initializer(tm, t0, engine)
18 # Transient Simulation
19 results_obj = 'Tnet2' # name of the object for saving results
20 tm = tsnet.simulation.MOCSimulator(tm,results_obj)
21 # Visulize results
22 tm.plot_node_head(['JUNCTION-105'])
23
```

Figure 9.21 Transient simulation under the pipe burst scenario.

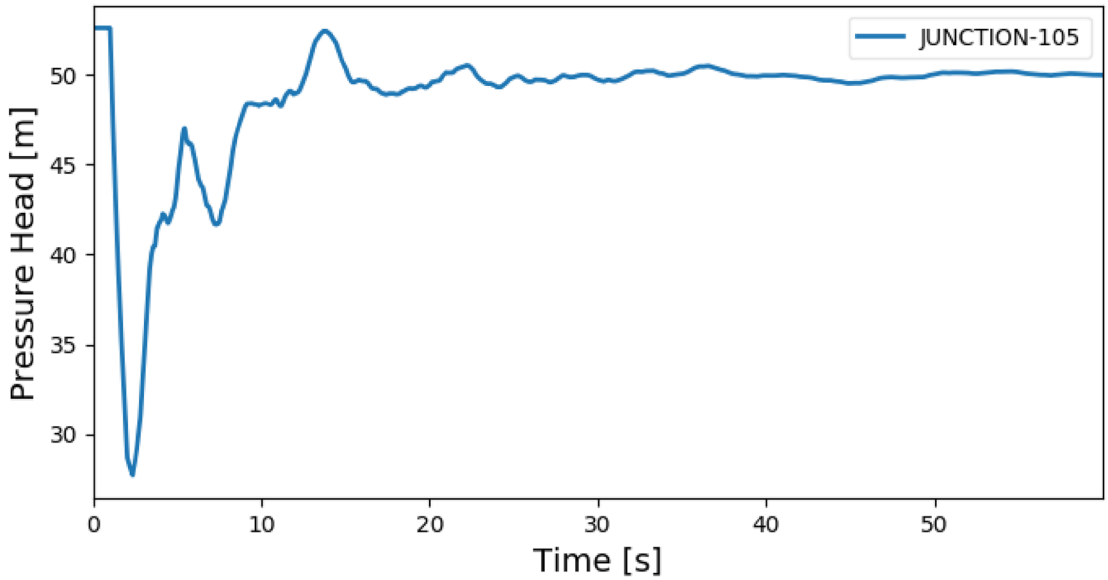

Figure 9.22 Pressure transients at JUNCTION-105 when a burst occurs at JUNCTION-105.

models, are implemented in TSNet. Additionally, more simulation results can be accessed, such as burst discharge, pipe flow rate, and pipe flow velocity. For more information about TSNet, please refer to the online documentation (Xing & Sela, 2021a).

REFERENCES

Anaconda. (2021). Anaconda: Your Data Science Toolkit. Available at: https://www.anaconda.com/products/individual (last accessed 3 March 2022)

Anderson J. D. (1995). Computational Fluid Dynamics. McGraw-Hill International Editions, Mechanical Engineering Series, New York, NY. Available at: https://www.amazon.com/Computational-Fluid-Dynamics-John-Anderson/dp/0070016852 (last accessed 10 May 2022)

Bergant A., Simpson A. R. and Tijsseling A. S. (2006). Water hammer with column separation: A historical review. *Journal of Fluids and Structures*, **22**, 135–171, https://doi.org/10.1016/j.jfluidstructs.2005.08.008

Boulos P. F., Karney B. W., Wood D. J. and Lingireddy S. (2005). Hydraulic transient guidelines for protecting water distribution systems. *Journal-American Water Works Association*, **97**(5), 111–124, https://doi.org/10.1002/j.1551-8833.2005.tb10892.x

Chaudhry M. H. (1987). Applied Hydraulic Transients. Van Nostrand Reinhold, New York.

Karney B. and McInnis D. (1990). Transient analysis of water distribution system. *Journal of American Water Resources Association*, **82**(7), 62–70.

Klise K. A., Murray R. and Haxton T. (2018). An overview of the Water Network Tool for Resilience (WNTR). WDSA/CCWI Joint Conference Proceedings. Available at: https://ojs.library.queensu.ca/index.php/wdsa-ccw/article/view/12150

Kong Q., Siauw T. and Bayen A. (2020). Python Programming and Numerical Methods: A Guide for Engineers and Scientists, pp. 480. Academic Press, Paperback ISBN: 9780128195499 eBook ISBN: 9780128195505.

Kranenburg C. (1974). Gas release during transient cavitation in pipes. *ASCE Journal of the Hydraulics Division*, **100**(HY10), 1383–1398, https://doi.org/10.1061/JYCEAJ.0004077

Larock B. E., Jeppson R. W. and Watters G. Z. (1999). Hydraulics of Pipeline Systems, pp. 552. CRC Press, Boca Raton. Available at: https://www.taylorfrancis.com/books/mono/10.1201/9780367802431/hydraulics-pipeline-systems-gary-watters-roland-jeppson-bruce-larock

Lee J. (2008). Two Issues in Premise Plumbing: Contamination Intrusion at Service Line and Choosing Alternative Plumbing Material. Doctoral dissertation, Virginia Tech., VA, USA.

Lee J., Lohani V. K., Dietrich A. M. and Loganathan G. V. (2012). Hydraulic transients in plumbing systems. *Water Science and Technology: Water Supply*, **12**(5), 619–629, https://doi.org/10.2166/ws.2012.036

Rossman L. A. (2000). *EPANET 2 User Manual*. *Social Studies of Science*. Available at: https://epanet22.readthedocs.io/en/latest/ (last accessed 3 March 2022)

Shu J. J. (2003). Modeling vaporous cavitation on fluid transients. *International Journal of Pressure Vessels and Piping*, **80**, 187–195, https://doi.org/10.1016/S0308-0161(03)00025-5

Spyder. (2021). Spyder: the Scientific Python Development Environment. Available at: https://docs.spyder-ide.org/current/index.html (last accessed 3 March 2022)

Walsh S. P. (1964). Pressure generated by cavitation in a pipe (Doctoral dissertation, Syracuse University).

Wylie E. B. and Streeter V. L. (1993). Fluid Transients in Systems. Prentice Hall, Upper Saddle River, NJ.

Xing L. and Sela L. (2020). Transient simulations in water distribution networks: TSNet python package. *Advances in Engineering Software*, **149**, 102884, https://doi.org/10.1016/j.advengsoft.2020.102884

Xing L. and Sela L. (2021a). TSNet Documentation. Available at: https://tsnet.readthedocs.io/en/latest/ (last accessed 3 March 2022)

Xing L. and Sela L. (2021b). Github Repositories for Transient Simulaiton in Water Networks (TSNet). Available at: https://github.com/glorialulu/TSNet/tree/master/examples (last accessed 3 March 2022)