1/ Introduction

The purpose of this lab is to be able to do simulation for new designing surge tanks for a hydraulic system. This system will be used for the Sacramento area. The surge tank has a hydraulic spring where it can absorb the pressure increase that is associated with the momentum of the fluid in the pipe. The outcome of this simulation is to have a plot of the resulting input, state and output trajectories and the explanations of all plots. It also uses Matlab's ode45 to numerically integrate ordinary differential equations. Moreover, figure 1 is the schematic of the hydroelectric dam.

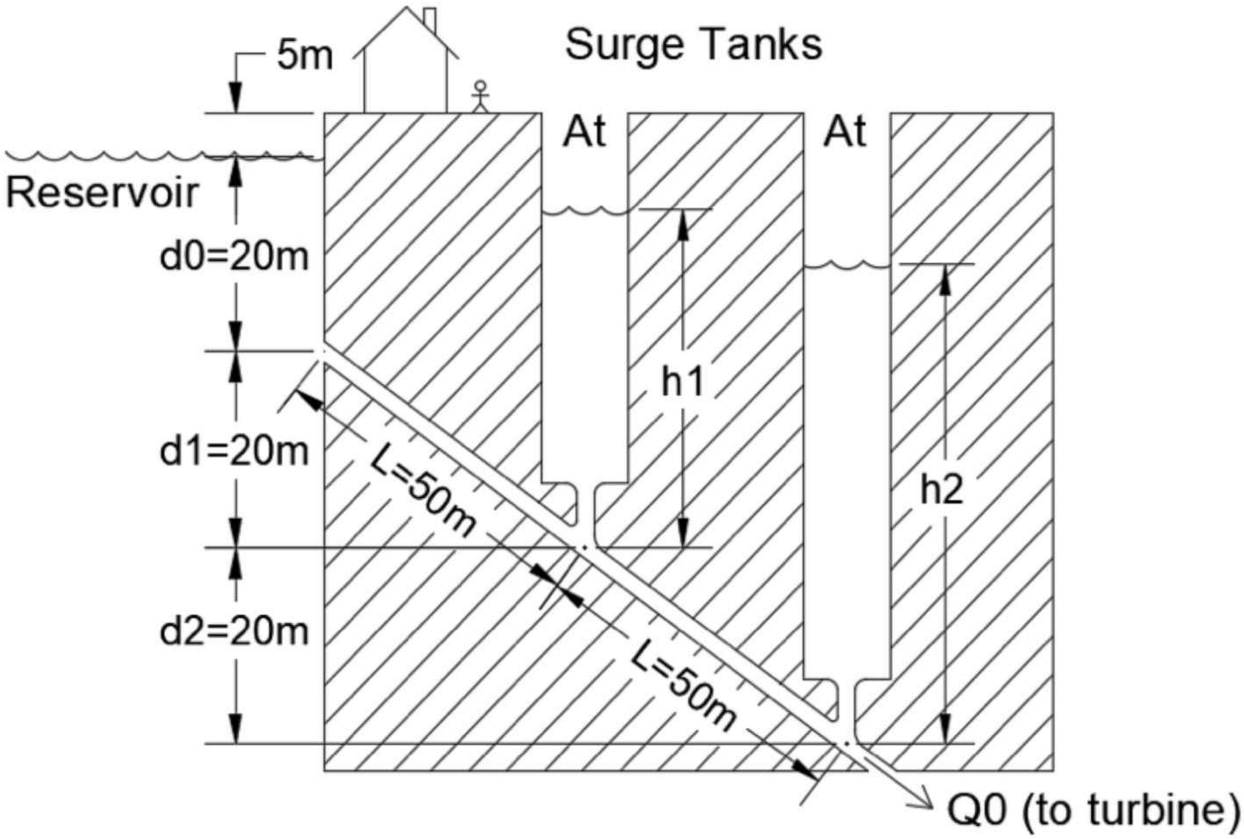


Figure 1: Schematic of the hydroelectric dam with reservoir and surge tanks

For this system, the tank area A_t was assumed to be significantly larger than the pipe area. Therefore, the tank inertia was neglected in comparison to the pipe inertia. Besides, the tank and reservoirs inlets had negligible resistance compared to the pipe resistances.

2/ Problem and Calculation

The bond graph of the system was given from the lab handout, we assigned causality and numbered bond as shown in figure 2.

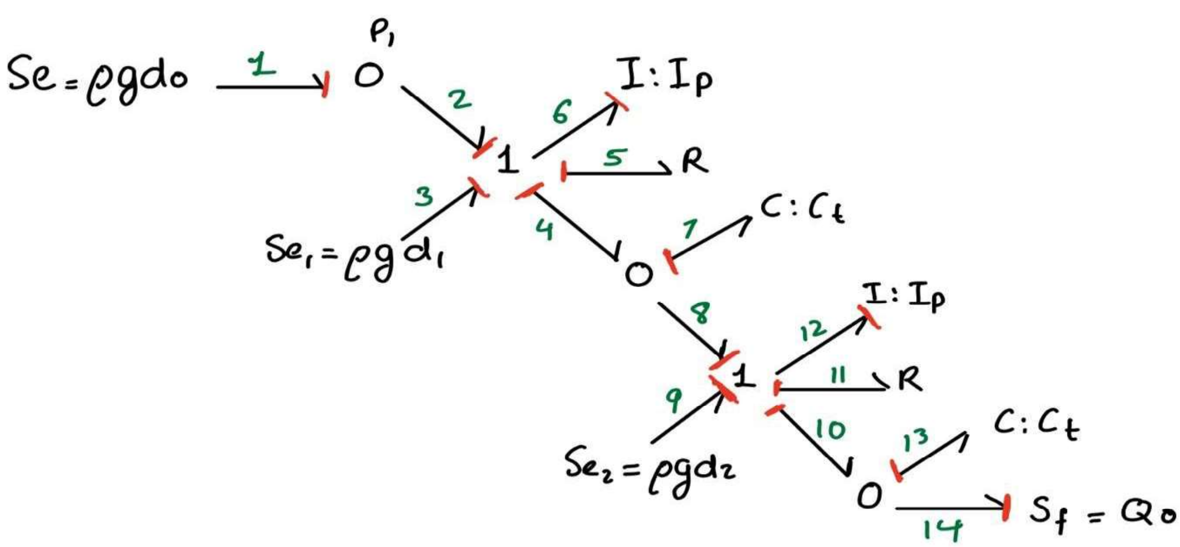


Figure 2: Bond Graph of the hydroelectric dam

Nevertheless, the pipe resistance in this system is not linear and it acts according to the constitutive law. The equation of this pipe resistance is shown below.

$$\Delta P = C_f\, Q|Q|$$

2.1 Constant Parameters

- Density of water
  - $\rho = 1000$ kg/m^3
- Acceleration due to gravity
  - $g = 9.81$ m/s^2
- Cross section area of the pipe from the reservoir to the turbine
  - $A_p = 0.1$ m^2
- Hydrostatic pressure
  - $P = \rho g \Delta h$ (N/ m^2)
- Fluid pipe resistance constant
  - $C_f = 49000$ kg/m^7
- Pipe section lengths
  - $L = 50$ m
- Fluid inertia of pipe sections
  - $I_p = \rho L/A_p = (1000*50)/0.1 = 500000$ kg/m^4
- Tank areas (to be determined)
  - A_t (m^2)
- Fluid capacitance of each tank
  - $C_t = A_t/\rho L$ (m^5/N)
- Distance between the ground surface and the reservoir
  - $d = 5$ m
- Distance between the reservoir and the entrance of the pipe
  - $\mathrm{d_o} = 20$ m
- Distance between the entrance of the pipe and the first 50m of the pipe
  - $\mathrm{d_1} = 20$ m
- Distance between the first 50 m the pipe and the second 50m of the pipe
  - $\mathrm{d_2} = 20$ m

2.2 State Equations

According to the bond graph, it has 4 state derivative equations. Among those equations, there are two of the fluid momentum state variables and other two are the volumetric displacement state variables. All equations are shown below.

- $\dot{p}_6 = e_6 = e_2 + e_3 - e_5 - e_4$
  - $\rightarrow e_1 + e_3 - e_5 - e_7$
  - $\rightarrow \rho g d_o + \rho g d_1 - C_f \frac{p_6}{I_p}\left|\frac{p_6}{I_6}\right| - \frac{q_7}{C_t}$
- $\dot{p}_{12} = e_{12} = e_8 + e_9 - e_{11} - e_{10}$
  - $\rightarrow e_7 + pgd_2 - Rf_{11} - e_{13}$
  - $\rightarrow \frac{q_7}{C_t} + pgd_2 - Rf_{12} - \frac{q_{13}}{C_t}$
  - $\rightarrow \frac{q_7}{C_t} + pgd_2 - R\frac{p_{12}}{I_p} + \frac{q_{13}}{C_t}$
- $\dot{q}_7 = f_7 = f_4 - f_8$
  - $\rightarrow \frac{p_6}{I_p} - \frac{p_{12}}{I_p}$
- $\dot{q}_{13} = f_{13} = f_{10} - f_{14}$
  - $\frac{p_{12}}{I_p} - Q_o$

2.3 Initial Condition

To find the initial condition, time (t) must be equal to zero and the initial turbine flow (Q_o) equals 1.5 $\frac{m^3}{s}$. Additionally, the steady state condition is setting to all the state derivative to zero. After substitute all of these condition and value, it produces the equation that is shown below.

- $\dot{p}_6 = \rho g d_o + \rho g d_1 - C_f \frac{p_6}{I_p}\left|\frac{p_6}{I_6}\right| - \frac{q_7}{C_t} = 0$
- $\dot{p}_{12} = \frac{q_7}{C_t} + pgd_2 - R\frac{p_{12}}{I_p} + \frac{q_{13}}{C_t} = 0$
- $\dot{q}_7 = \frac{p_6}{I_p} - \frac{p_{12}}{I_p} = 0$
- $\dot{q}_{13} = \frac{p_{12}}{I_p} - Q_o = 0$

By using the MATLAB, we wrote a code that can find the initial condition after includes all constant parameter at initial condition.

- $p_6 = 7.5 \times 10^5 \frac{kg}{ms}$
- $p_{12} = 7.5 \times 10^5 \frac{kg}{ms}$
- $q_7 = 20.99\, m^3$
- $q_{13} = 27.39\, m^3$

2.4 Time Parameter

$$w_n = \sqrt{\frac{1}{C_t \times I_p}} = 0.16\, \frac{rad}{s}$$

$$f_n = \frac{w_n}{2\pi} = 0.0263\, Hz$$

$$T_{period} = \frac{1}{f_n} = 38.04\, s$$

The time period (T_{period}) is the inverse of the frequency (f_n).

For this lab, it assumes the finish time to be around 5 times the vibration period and the time final is shown below

$$T_{period} = 5 \times T_{period} = 190.18\, s$$

The total number of the time steps can be determined based on both the final time and the time period by having at least 100 steps per oscillation.

$$dt = \frac{T_{period}}{100} = 0.38\, s$$

$$NumberStep = \frac{T_{final}}{dt + 1} = 501\, steps$$

2.5 Input Parameters

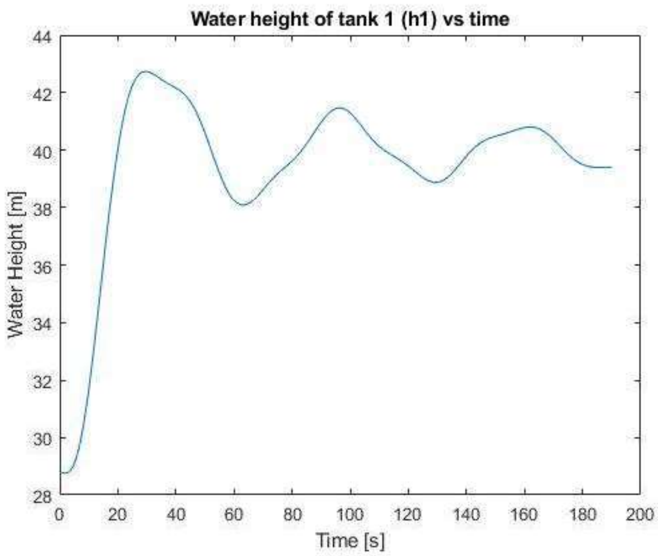
When the turbine was delivering power, the steady state flow was $Q_0 = 1.5\ m^3/s$. Initially, when t = 0s , the flow was $Q_0 = 1.5\ m^3/s$. We would then choose a random time when the turbine is turned off and within the next 0.15s the flow Q_0 decreased to zero. Therefore, we define our as shown below for before, during and after the shutdown period.

When $t < 1s \rightarrow Q_0 = 1.5\ (m^3/s)$
When $t <= 1s$ and $t < 1.15s \rightarrow Q_0 = 1.5 - 10*(t-1)\ (m^3/s)$
When $t > 1.15s \rightarrow Q_0 = 0\ (m^3/s)$

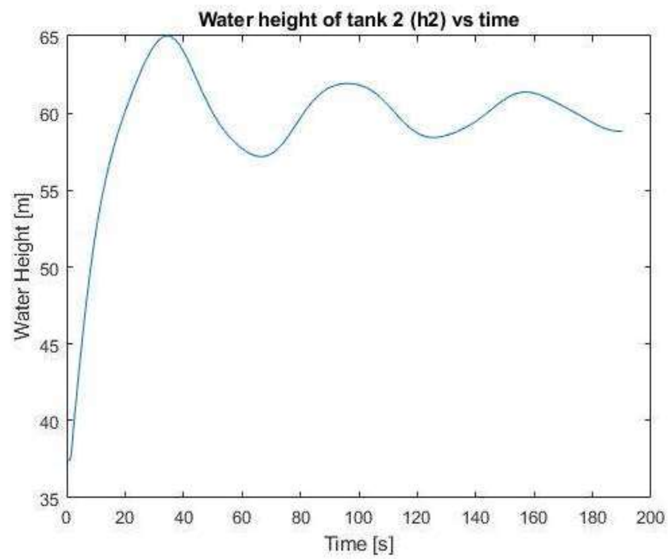
2.6 Output Parameters

For the output, we were tracking the height of water in surge tank 1 and 2.

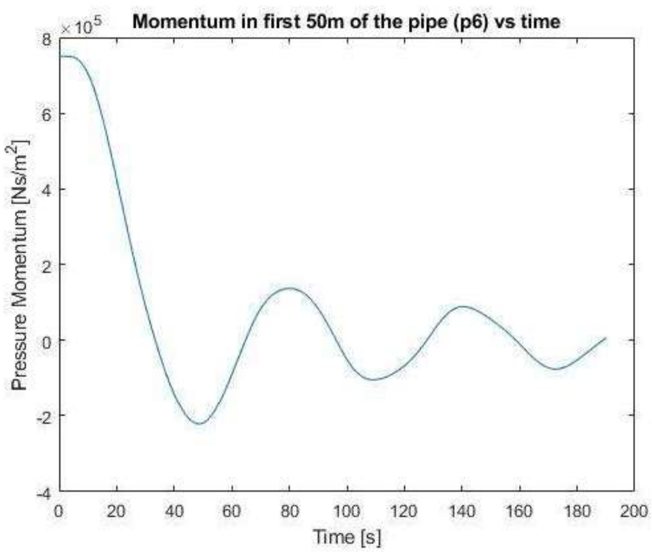
3/ Simulation



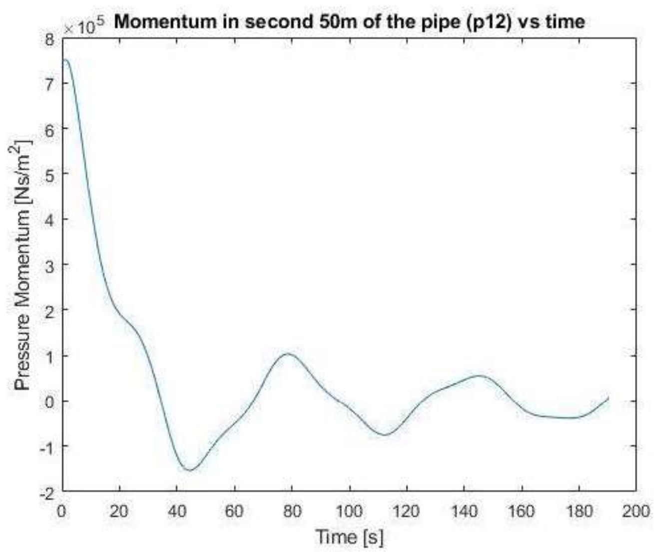
3a/ Water height of tank 1 vs time



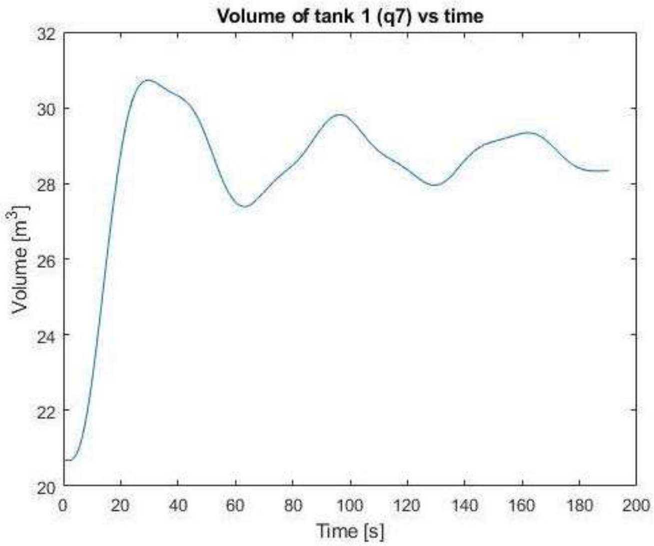
3b/ Water height of tank 2 vs time



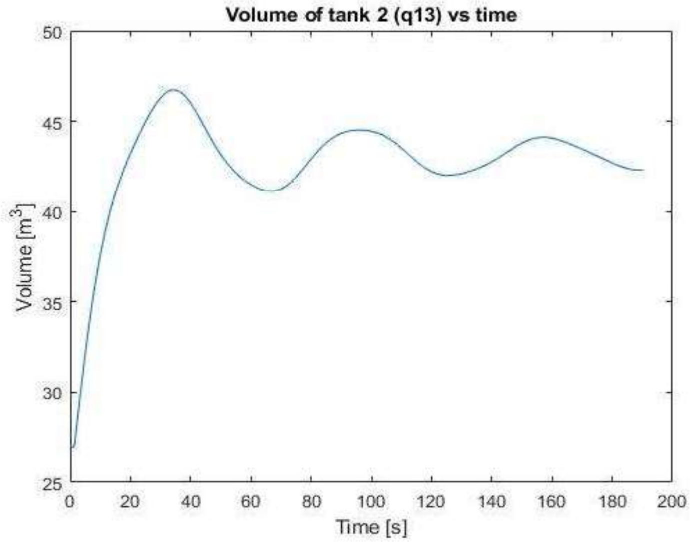
3c/ Pressure momentum in first 50 m of pipe vs time



3d/ Pressure momentum in second 50 m of pipe vs time



3e/ Volume of tank 1vs time



3f/ Volume of tank 2 vs time

Fig 3 a-f: First and second surge water height, pressure momentum and volume

1/ Looking at the water height in surge tank 1 and 2 in figure 3a and 3b, we recognized that when the water height in tank 1 was approaching 42.7 (m), the water height in tank 2 was already approaching 65 (m). The water height in tank 2 was 22.3 (m) higher than the water height in tank 1 within the same amount of time. Therefore, surge tank 2 was coming close to overflowing. Figure 3a and 3b also showed that the height of water in surge tank 1 and surge tank 2 were oscillating between 40 m and 60 m respectively. This satisfied the requirements that the water height in these tanks do not exceed 45 m and 60 m respectively. In comparison of inertia and pressure between tank 1 and 2, tank 2 had higher water inertia (20 m deeper) and higher pressure. These characteristics made tank 2 fill up faster hence coming closer to overflowing faster than tank 1.

2/ The vibration periods for the water height in tank 1 and 2 in figure 3a and 3b were found by taking the time difference between the first peak and third peak then divide that time difference by 2. This method applied to both surge tank to calculate the vibration period. The result showed that the vibration period for tank 1 and tank 2 was 66.2 s and 61.4 s respectively. While the period from the previously calculated frequency was 38.0 s for both tank 1 and tank 2. This result showed that the vibration period of tank 1 and 2 was approximately 54.1% and 47.1% higher than the period from the calculated natural frequency. This difference happened since the effect of resistors and combination of other compliance and inertia was ignored when calculating the natural frequency. Basically, the natural frequency was only calculated based on one compliance and one inertia. Therefore, the vibration period was higher than the period from the calculated natural frequency.

4/ Contribution

Team members were Vinh Nguyen and Borum Long. Both members worked together to ensure the lab report was completed effectively and on time. The report was done by both members. Borum Long developed the simulations code including the input, output, right hand side, and main function. Vinh Nguyen debugged the code and wrote the analysis part of the report including explanation of the results and the plots from Matlab. Borum also worked on the other parts of the lab including the introduction, bond graph, state variable, system parameters, initial conditions and time parameters. Vinh worked on the constant parameter and the graphs. Overall, the workload was evenly distributed for both persons.

5/ Code

eval_rhs.m – Evaluate the right hand side

```
function xdot = eval_rhs(t, x, w, c)

% eval_rhs - Returns the time derivative of the states
% Syntax: xdot = eval_rhs(t, x, w, c)
% Inputs:
```

```
% t - Scalar value of time, size 1x1.
% x - State vector at time t.
% w - Anonymous function, w(t, x, p), that returns the input vector at time t.
% c - Constant parameter vector.

% Outputs:
% xdot - Time derivative of the states at time t.

% unpack the states into useful variable names
p6 = x(1); % pressure momentum in the first 50m of pipe (Ns/m^2)
q7 = x(2); % tank 1 volume (m^3)
p12 = x(3);% pressure momentum in the second 50m of pipe (Ns/m^2)
q13 = x(4);% tank 2 volume (m^3)

% evaluate the input function
r = w(t, x, c);
% unpack the inputs into useful variable names
P1 = r(1); % pressure at the entrance of pipe (pa)
P2 = r(2); % pressure at bottom of tank 1 (pa)
P3 = r(3); % pressure at bottom of tank 2 (pa)
Q0 = r(4); % turbine flow rate (m^3/s)

% unpack the parameters into useful variable names
rho = c.rho; % Water density (kg/m^3)
g = c.g; % Gravitational Acceleration (m/s^2)
Ap = c.Ap; % Cross sectional area of the pipe from the reservoir to the turbine (m^2)
Cf = c.Cf; % Fluid pipe resistance constant (kg/m^7)
L = c.L; % Pipe section lengths (m)
Ip = c.Ip; % Fluid inertia of pipe sections (kg/m^4)
At = c.At; % Tank areas (m^2)
Ct = c.Ct; % Fluid capacitance of each tank (m^5/N)
d = c.d; % distance between the ground surface and the reservoir (m)
d0 = c.d0; % distance between the reservoir and the entrance of pipe (m)
d1 = c.d1; % distance between the entrance of pipe and the first 50m of pipe (m)
d2 = c.d2; % distance between the first 50m of pipe and the second 50m of pipe (m)

% calculate the state derivatives
p6dot= rho*g*d0+rho*g*d1-Cf*p6/Ip*abs(p6/Ip)- q7/Ct;
q7dot= (p6/Ip)-(p12/Ip);
p12dot= q7/Ct +rho*g*d2-Cf*p12/Ip*abs(p12/Ip)-q13/Ct;
q13dot= p12/Ip - Q0;

% pack the state derivatives into an mx1 vector
xdot = [p6dot;q7dot;p12dot;q13dot];
end
```

eval_input.m – Evaluate the input

```
function r = eval_input(t, x, c)

% eval_pipe - Returns the input vector at any given time.
% Syntax: r = eval_input(t, x, c)
% Inputs:
% t - A scalar value of time, size 1x1.
% x - State vector at time t, size mx1 where m is the number of states.
% c - Constant parameter vector, size px1 where c is the number of parameters.
%Outputs:
% r - Input vector at time t.

% unpack the parameters into useful variable names
```

```
rho = c.rho; % Water density in (kg/m^3)
g = c.g; % Gravitational constant in (m/s^2)
d0 = c.d0; % distance between the reservoir and the entrance of pipe in (m)
d1 = c.d1; % distance between the entrance of pipe and the first 50m of pipe in (m)
d2 = c.d2; % distance between the first 50m of pipe and the second 50m of pipe in (m)

%The pressure at different position of the pipe
P1 = rho*g*d0; % pressure at entrance of pipe (pa)
P2 = rho*g*d1; % pressure at bottom of tank 1 (pa)
P3 = rho*g*d2; % pressure at bottom of tank 2 (pa)

% At time T, the turbine is off
T = 1;
if t < T
    Q0 = 1.5;
elseif T <= t && t < (T +0.15)
    Q0 = 1.5 -10*(t-T);
else
    Q0 = 0; % turbine is off
end

r = [P1;P2;P3;Q0];
end
```

eval_output.m – Evaluate the output

```
function Y = eval_output(t, x, w, c)

% eval_output - Returns the output vector at the specified time.
% Syntax: Y = eval_output(t, x, r, c)
% Inputs:
% t - Scalar value of time, size 1x1.
% x - State vector at time t.
% r - Input vector at time t.
% c - Constant parameter vector.

%Outputs:
% Y - Output vector at time t.

p6 = x(1); % pressure momentum in the first 50m of pipe (Ns/m^2)
q7 = x(2); % tank 1 volume in (m^3)
p12 = x(3); % pressure momentum in the second 50m of pipe (Ns/m^2)
q13 = x(4); % tank 2 volume (m^3)

% evaluate the input function
r = w(t, x, c);

% unpack the inputs into useful variable names
P1 = r(1); % pressure at the entrance of pipe (pa)
P2 = r(2);% pressure at bottom of tank 1 (pa)
P3 = r(3);% pressure at bottom of tank 2 (pa)
Q0 = r(4); % turbine flow rate (m^3/s)


% unpack the parameters into useful variable names
At = c.At; % Tank areas (m^2)


% find the heights of tank 1 and 2
h1 = q7/At;
```

```
h2 = q13/At;

% pack the outputs into a 1x1 vector
Y = [h1;h2];
end
```

main_function.m – Main function

```
% Constant Parameter
c.rho = 1000; % Density of water in kg/m^3
c.g = 9.81; % Acceleration due to gravity in m/s^2
c.Ap = 0.1; % Cross sectional area of the pipe from the reservoir to the turbine in m^2
c.Cf = 49000; % Fluid pipe resistance constant in kg/m^7
c.L = 50; % Pipe section lengths in m
c.Ip = c.rho*c.L/c.Ap; % Fluid inertia of pipe sections in kg/m^4
c.At = 0.719; % Tank areas in m^2
c.Ct = c.At/c.rho/c.g; % Fluid capacitance of each tank in m^5/N
c.d = 5; % distance between the ground surface and the reservoir in m
c.d0 = 20; % distance between the reservoir and the entrance of the pipe in m
c.d1 = 20; % distance between the entrance of the pipe and the first 50m of the pipe in m
c.d2 = 20; % distance between the first 50m of the pipe and the second 50m of the pipe in m


% Initial conditions for the states
Q0_initial = 1.5;
syms p6_eq q7_eq p12_eq q13_eq
eqn1 = c.rho*c.g*c.d0+c.rho*c.g*c.d1-c.Cf*p6_eq/c.Ip*abs(p6_eq/c.Ip)- q7_eq/c.Ct == 0;
eqn2 = (p6_eq/c.Ip)-(p12_eq/c.Ip) == 0;
eqn3 = q7_eq/c.Ct + c.rho*c.g*c.d2-c.Cf*p12_eq/c.Ip*abs(p12_eq/c.Ip) -q13_eq/c.Ct== 0;
eqn4 = p12_eq/c.Ip - Q0_initial == 0;
sol = solve([eqn1, eqn2, eqn3, eqn4], [p6_eq, q7_eq, p12_eq, q13_eq]);
p6_eq = double(sol.p6_eq);
q7_eq = double(sol.q7_eq);
p12_eq = double(sol.p12_eq);
q13_eq = double(sol.q13_eq);
x0 =[p6_eq; q7_eq; p12_eq; q13_eq]

% Time steps: determining the time span.
Wn = sqrt(1/(c.Ct*c.Ip)) % natural frequency in Hz
Fn= Wn / (2* pi);
Tperiod = 1/Fn % period of 1 oscillation
dt = Tperiod/100;
Tfinal = 5 * Tperiod % 5 oscillations
num_steps = round(Tfinal / dt) +1;   % number of step
ts = linspace (0, Tfinal, num_steps); % time span

% Anonymous function
f_anon = @(t, x) eval_rhs(t, x, @eval_input, c);
[ts, xs] = ode45(f_anon, ts, x0);

for i=1:length(ts)
xdot = eval_rhs(ts(i), xs(i, :), @eval_input, c);
end
rs = zeros(length(ts), 4);
for i=1:length(ts)
rs(i, :) = eval_input(ts(i),xs(i, :), c);
end
Ys = zeros(length(ts), 2);
for i=1:length(ts)
Ys(i, :) = eval_output(ts(i), xs(i, :), @eval_input, c);
```

```
end

% plot output
figure(1)
plot(ts, xs(:, 1))
title('Momentum in first 50m of the pipe (p6) vs time')
ylabel('Pressure Momentum [Ns/m^2]')
xlabel('Time [s]')

figure(2)
plot(ts, xs(:, 3))
title('Momentum in second 50m of the pipe (p12) vs time')
ylabel('Pressure Momentum [Ns/m^2]')
xlabel('Time [s]')

figure(3)
plot(ts, xs(:, 2))
title('Volume of tank 1 (q7) vs time')
ylabel('Volume [m^3]')
xlabel('Time [s]')

figure(4)
plot(ts, xs(:, 4))
title('Volume of tank 2 (q13) vs time')
ylabel('Volume [m^3]')
xlabel('Time [s]')

figure(5)
plot(ts, Ys(:, 1))
title('Water height of tank 1 (h1) vs time')
ylabel('Water Height [m]')
xlabel('Time [s]')

figure(6)
plot(ts, Ys(:, 2))
title('Water height of tank 2 (h2) vs time')
ylabel('Water Height [m]')
xlabel('Time [s]')
```