# MAE 598: Project #2 Challenge #4

Shane Dombrowski

November 10, 2016

**Problem Collaboration Note.** I collaborated with William "Davis" Burk on this challenge problem. We simply helped each other with bugs and setting up the simulation. All work and code contained within this document is my own and created from my own workstation.

**Problem Challenge #4.** This challenge required us to create a 2D domain that consisted of two tanks connected by a pipe below them, allowing fluid to travel between the two. One tank had a higher level of water, causing an oscillation between the two. Below is a diagram of the geometry.

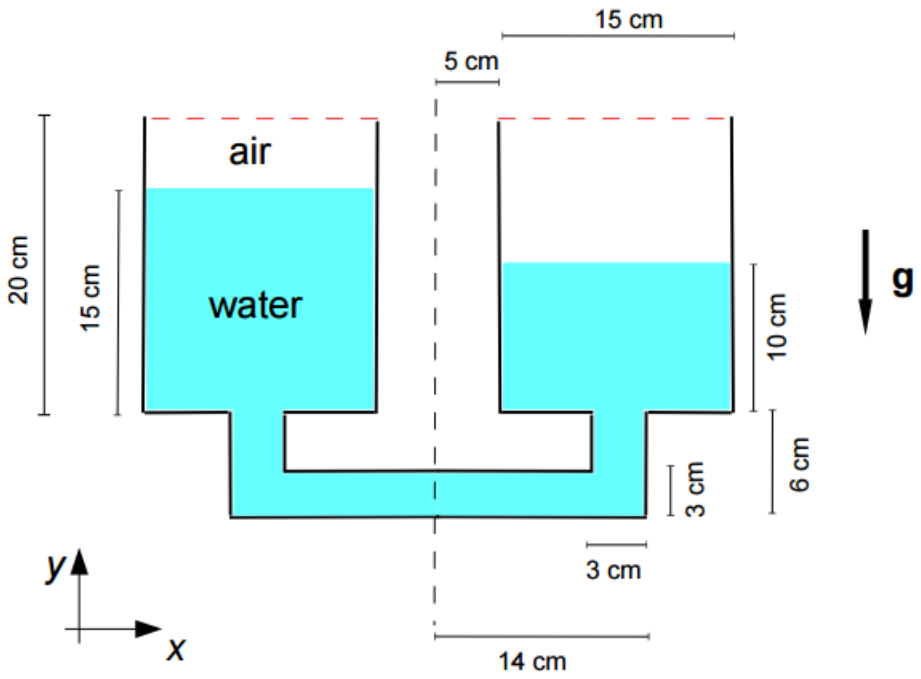


Figure 1: Initial geometry and problem setup.

From here, the meshing was done using a slightly finer mesh than the default "Fine" setting. The top two tank outlets were simply labeled "outlet-1" and "outlet-2" since we'd

be changing them later. In FLUENT, the boundary conditions were both set to pressure outlets with no outflow conditions. Each tank was then separated from the original surface body mesh so that we could perform a volume monitor on both tanks separately without data from the pipe. Water was designated as Phase 2 and was patched in two separate chunks to simulate the two different initial water levels. The simulation was run until the volume of each tank had done three full oscillations. Since the volume of water in each tank is proportional to the depth (2D), it was a clear indicator of depth and exported as a text file into MATLAB to create the following plot.

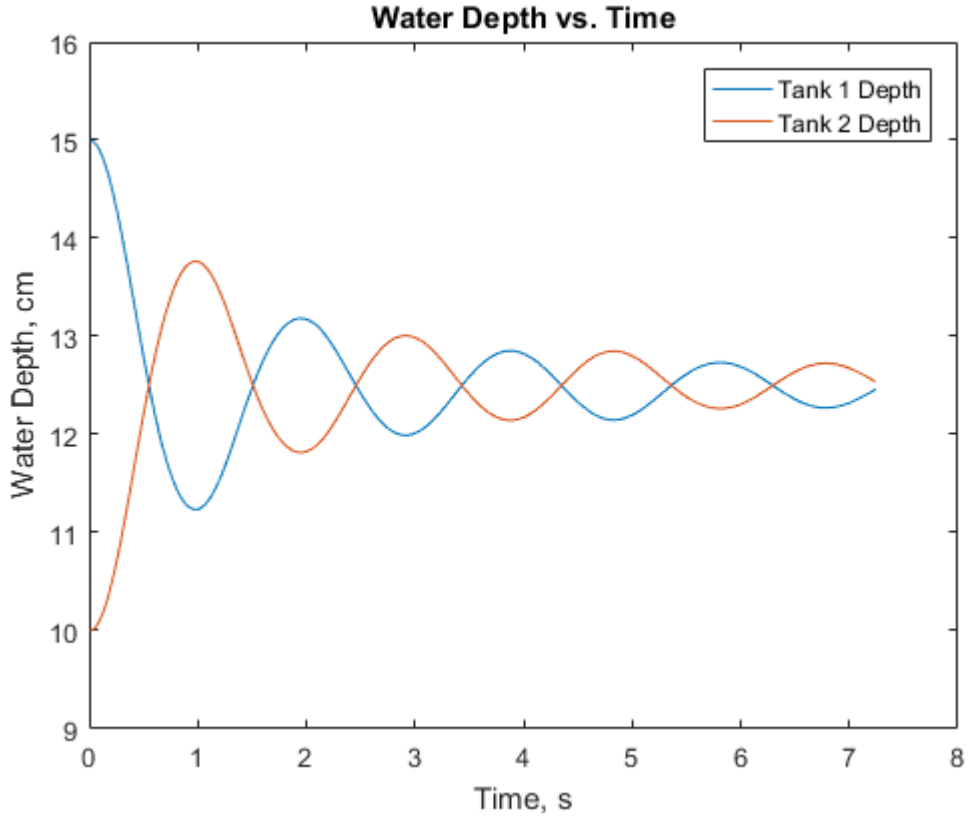


Figure 2: Plot showing the depth of water in each tank as a function of time. This was calculated by dividing volume of water in each tank by width to find average depth.

Using this data in MATLAB, we were able to determine the average period of oscillation and the times when the water levels were equal for the first two times. The average period was calculated to be approximately **1.92 seconds**. The code used to calculate this is attached. The first two times that the water levels were equal were found to be approximately **0.55 seconds** and **1.52 seconds**. Using these two times, the simulation was run again to these two points to generate velocity vector plots. These are shown below.

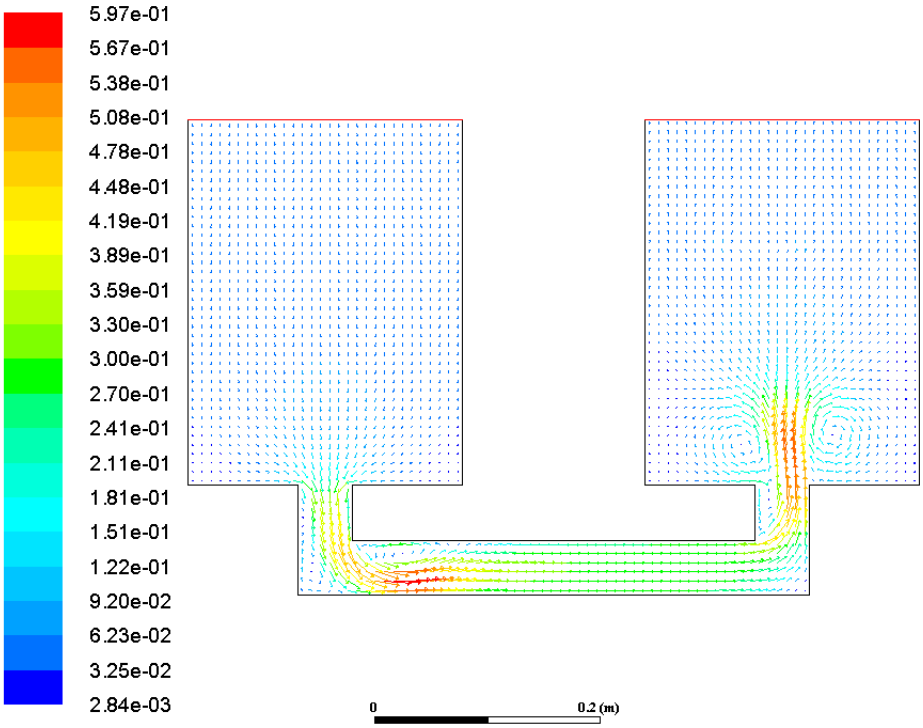


Figure 3: Velocity vector plots for 0.55 seconds.

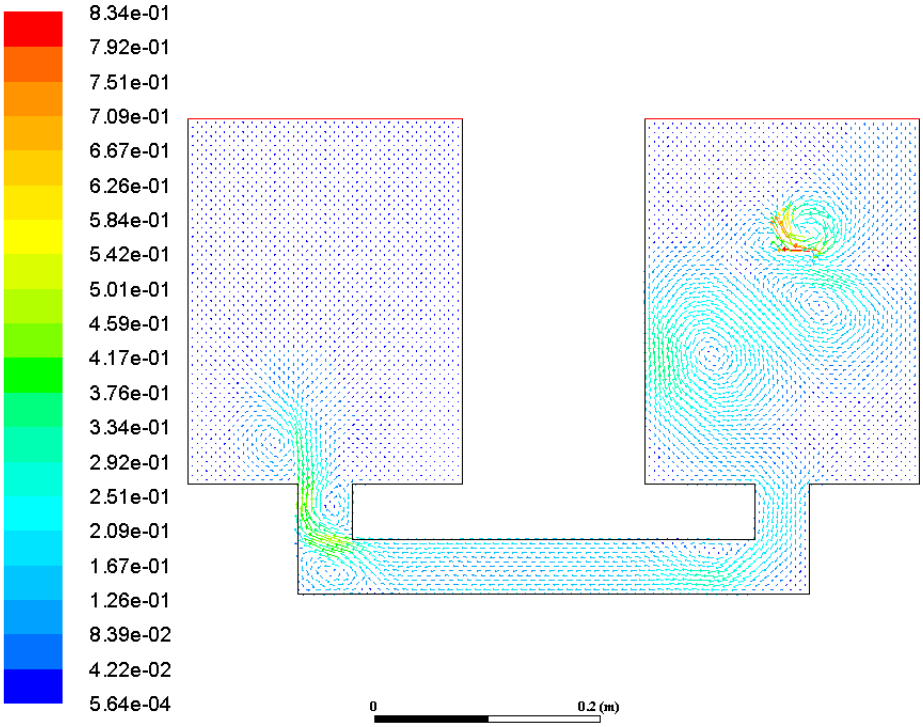


Figure 4: Velocity vector plots for 1.52 seconds. The circulation on the water surface.

Finally, a volume fraction contour was made for the time point when the right tank hit its maximum depth for the first time. This plot is shown below.

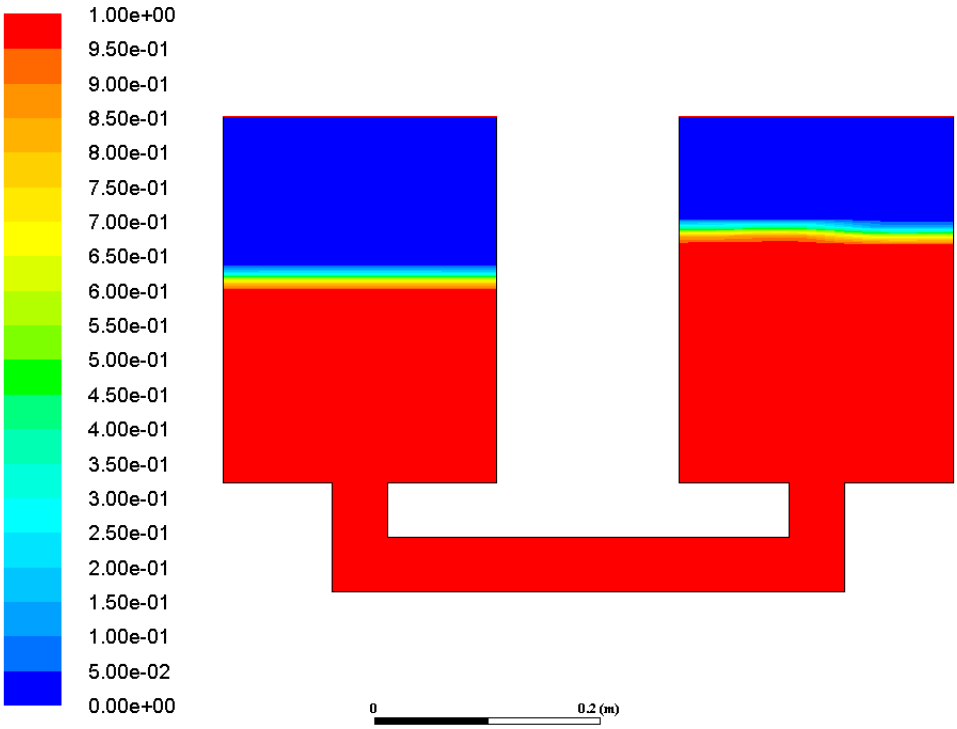


Figure 5: Volume fraction when the right tank peaks for the first time. Water is defined as red, phase 2. This occurred at approximately 0.98 seconds.

The final part of this challenge problem required us to use different combinations of boundary conditions for the tops of the tanks and observe FLUENT's response to them. The first combination I tried was giving the left tank a **wall** for a top boundary condition and leaving the right tank as a **pressure outlet**. I would assume the fluid wouldn't move due to the lack of a path for the air to take. Below is the result after 5 seconds.

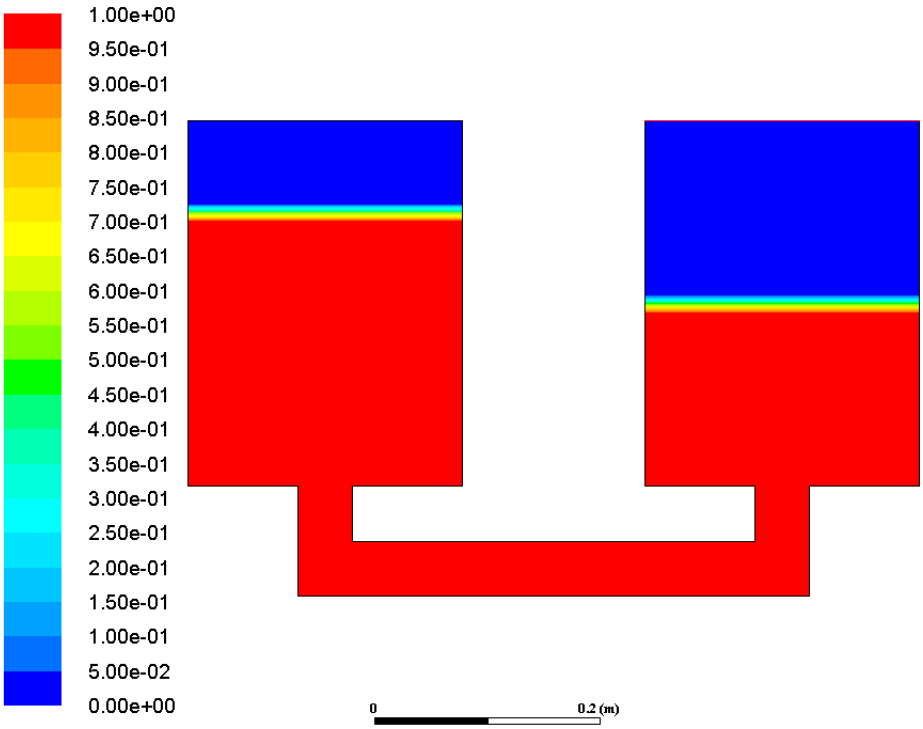


Figure 6: Left boundary condition is a wall while right is kept as pressure outlet at 5 seconds.

As predicted, the fluid does not move after 5 seconds because of the pressure in the cavity above it. This combination of boundary conditions does not work.

The next combination I tried was to make the left boundary an **intake fan** and the right boundary a **outlet vent**. Unfortunately, FLUENT did not like this combination and threw a floating point exception error relatively early in the calculation process.

```
Error at Node 5: floating point exception

Error at Node 6: floating point exception

Error at Node 7: floating point exception

Error: floating point exception
Error Object: #f
Registering VolumeMonitors, ("tank1.txt" "tank2.txt")

Calculation complete.
```

Figure 7: FLUENT is not as adventurous as myself.

The third combination I tried was a bit less wild. I set the left boundary to **pressure inlet** and the right boundary to **outflow.** Again, after running a bit longer than the previous simulation, my Courant number grew too large and the simulation had to stop.

```
Error at Node 6: Global courant number is greater than 250.00   The
velocity field is probably diverging. Please check the solution
and reduce the time-step if necessary.

Error at Node 7: Global courant number is greater than 250.00   The
velocity field is probably diverging. Please check the solution
and reduce the time-step if necessary.

Global Courant Number [Explicit VOF Criteria] : 1388.03
```

Figure 8: 1,388 is much larger than an ideal Courant number.

Determined to get a somewhat interesting result, I tried a fourth and final combination of boundary conditions. I set the left boundary to a **pressure inlet** and set the right boundary to a **pressure outlet** and attempted to run the simulation for 5 seconds. The result is below and looks very similar to when both boundary conditions are pressure outlets.

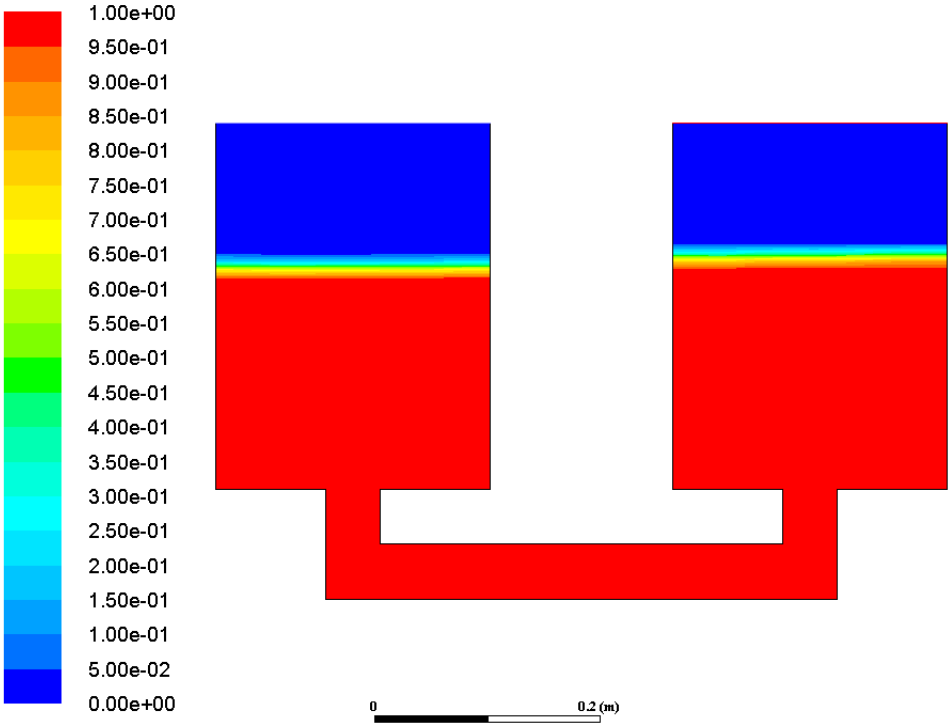


Figure 9: Water level is extremely similar. Pressure inlet must behave similarly to pressure outlet.

This shows that many boundary conditions, without knowledge or their operation and correct setup, will often throw errors within FLUENT that causes the simulation to stop. Some boundary conditions, such as pressure inlet, seem to behave similarly to pressure outlet. I personally do not know if there is a difference other than a name, but I

assume there is. Other conditions, such as outflow, seemingly work in previous simulation in Project #2 but do not work here. This may be due to flow going both ways throughout the oscillation and not a steady direction similar to the methane problem. Pressure outlet seems to be a good choice when flow varies in both velocity and direction through a boundary.

Below begins the MATLAB code used to calculate values in this challenge problem.

```
%Shane Dombrowski
%Applied CFD Challenge #4
clear,clc

tank1=importdata('tank1.txt')
tank2=importdata('tank2.txt')
tank1=tank1.data;
tank2=tank2.data;

width=0.15;      %width of tank in meter

time=tank1(:,1);
tank1vol=tank1(:,2);
tank2vol=tank2(:,2);

%This divides the volume of water in each tank by the width of the
    tank to
%calculate average height of the water.

height1=tank1vol./width;
height2=tank2vol./width;

height1=height1*100;    %converts meters to centimeters
height2=height2*100;

figure(1)
plot(time*0.01,height1,time*0.01,height2)
ylim([9 16])
xlabel('Time, s')
ylabel('Water Depth, cm')
title('Water Depth vs. Time')
legend('Tank 1 Depth','Tank 2 Depth')

%Find first two points in time where water levels are equal.

eq1=height1(1:100)-height2(1:100);
eq2=height1(100:200)-height2(100:200);
eq3=height1(200:300)-height2(200:300);
```

```
eq4=height1(300:400)-height2(300:400);
eq5=height1(400:500)-height2(400:500);
eq6=height1(500:600)-height2(500:600);
eq7=height1(600:700)-height2(600:700);

k1=find(abs(eq1)<0.05);
k2=find(abs(eq2)<0.01);
k3=find(abs(eq3)<0.02);
k4=find(abs(eq4)<0.01);
k5=find(abs(eq5)<0.01);
k6=find(abs(eq6)<0.01);
k7=find(abs(eq7)<0.006);
k2=k2+100;
k3=k3+200;
k4=k4+300;
k5=k5+400;
k6=k6+500;
k7=k7+600;

equal_time1=time(k1)*0.01
equal_time2=time(k2)*0.01
equal_time3=time(k3)*0.01;
equal_time4=time(k4)*0.01;
equal_time5=time(k5)*0.01;
equal_time6=time(k6)*0.01;
equal_time7=time(k7)*0.01;

%Calculate average period

p1=equal_time3-equal_time1;
p2=equal_time5-equal_time3;
p3=equal_time7-equal_time5;

avg_period=mean([p1 p2 p3]);
```