Team Meeting

Date: November 25 2020 **Location**: Discord **Time**: 4:00 PM EST

Meeting Facilitator:

Electronic Signatures of Attendees:

- Natalie Harvey
- Alyssa Devincenzi
- Matthew Stuber
- Agathiya Tharun

Agenda			
Item	**Item Description**	**Presenter**	**Minutes**
1	Read Project Description		10
2	Discuss Project Description		15
3	Set Up Live Sharing for Code		60
4			
5			

Notes	Action Items
• Looked over project description • Downloaded VS code and tried editing and running shared files	• Meet on Friday

Team Meeting

Date: November 27 2020 **Location**: Discord **Time**: 3:00 PM EST

Meeting Facilitator:

Electronic Signatures of Attendees:
- Matthew Stuber
- Natalie Harvey
- Alyssa Devincenzi
- Agathiya Tharun

Agenda			
Item	**Item Description**	**Presenter**	**Minutes**
1	Factors to include in the model		100
2			
3			
4			
5			

Notes	Action Items

• Check at least two of the extra conditions (One that can be ignored and one that cannot)
• Maybe check all four extra conditions mentioned in the slides (one condition each)
• Diameter of pipe - constant, or change at some point on the hill?
    ○ Keep constant once one diameter is chosen
• What do we allow users to determine?
    ○ Number of pipes?
• Find set values first
	• Meeting on Saturday
	    ○ Flowcharting
	• Meeting on Sunday
	    ○ Discuss research
	• Research: everyone researches one factor
	• Aggy: heating/cooling
	• Natalie: stuff falling in reservoir
	• Alyssa: leaks in pipes
	• Matt: max time to fill reservoir

Model Factors and Assumptions

Factors:
- Evaporation
- Pipe Friction
- Bends in Pipes
- Leaks in pipes
- Weather cools/heats water/pipes
- Energy lost to damage of equipment
- People/animals/litter falling into reservoir
  - Clogging pipes

Assumptions:
- Pipe diameter is consistent per simulation once chosen
- Energy out is exactly 120 (no more, no less)
- Pipes are above ground - resting on top of the hill, not buried
- How often will we be required to produce energy (affects reservoir fill time)
- Turbines and reservoir can connect to multiple pipes (not limited to 1 pipe opening)
- Base of each zone acts as the floor of the reservoir. Earth ground is a sufficient floor.

Things to consider (set values):
- Zone
- Time required to fill reservoir (pump speed)
  - Determine the average time the sun can provide solar energy
  - The reservoir must be able to fill in this amount of time, otherwise, the model is not valid.
- Base area of reservoir

Inputs:
- Number of pipes
- Diameter
- Bends
- Depth of reservoir

Questions:
- What constitutes a social factor?
- Can we just use the equations in the PowerPoints, or do we need to derive them on our own? Can we cite the PowerPoint?

Variable relationships
- Decrease in length: lower Ein
- Increase in depth: lower Ein

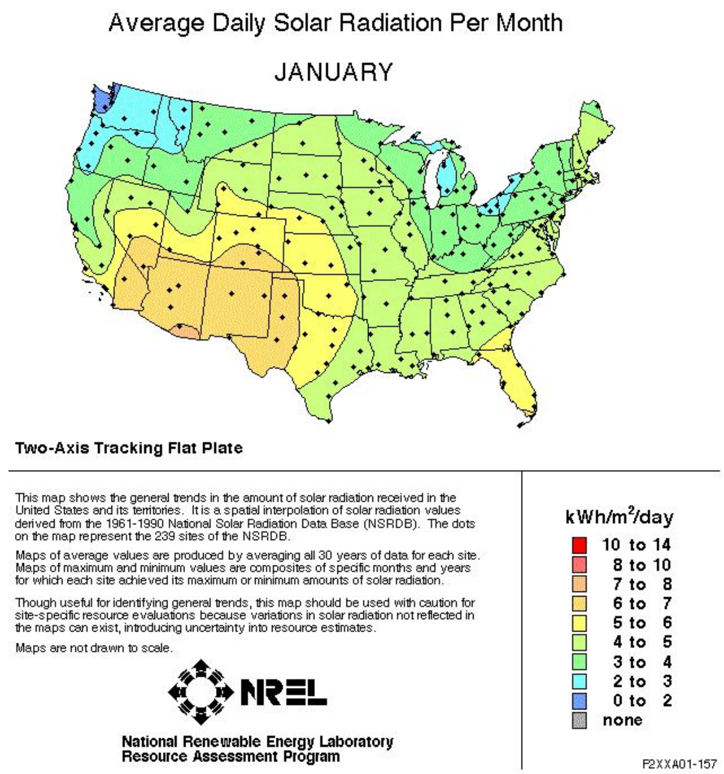
Set Specifications

- Refill time <5hrs
- Drain time <12hrs
- Depth <20m
- Eout = 120
- Surface area has to fit within zone constraints

Fill Time Research

If it is known how many hours the solar farm can produce energy to pump water up the reservoir, the maximum fill time is known. This is because the reservoir must be able to replenish its water supply while the solar panels can supply sufficient power, otherwise, the reservoir will not be able to produce the required energy output because there will not be enough water to turn the turbines. How many hours in a day can solar panels operate? Once this is known, the time needed to fill the reservoir can be determined.

According to the National Renewable Energy Laboratory (NREL), The average location in the United States sees 5 hours of usable sunlight that can be converted into energy. **Therefore, the pumped hydro storage system must be able to replenish its water supply within 5 hours if it is to replenish its water supply before it needs to generate again.**



Citations (Data and Graphic)

Sengupta, M., Y. Xie, A. Lopez, A. Habte, G. Maclaurin, and J. Shelby. 2018. "The National Solar Radiation Data Base (NSRDB)." Renewable and Sustainable Energy Reviews 89 (June): 51-60.

Image from:
https://www.solar-electric.com/learning-center/solar-insolation-maps.html/#:~:text=If%20you%20look%20at%20the,of%20full%20sun%20per%20day.

Leaky Pipes Research

Equation:
An equation for determining the amount of water lost in a leaking pipe is created from the Bernoulli equation, which describes flow of a fluid from one location to another, and a discharge coefficient, C. (2)

$$Q = CA\sqrt{\frac{2\,\Delta P}{S\,\rho_{w,std}}}$$

Variables:
C- Because the Bernoulli equation describes flow of an inviscid liquid, and the water leaks through the hole turbulently, a coefficient C is used to calculate the amount of water lost, C has been experimentally determined to be 0.61.

Q is the leak rate in m^3/s

A is the area of the hole/crack

P is the change in pressure from inside the pipe to outside

S is the specific gravity of the liquid

Pw,std is the mass density of water at standard conditions

Size of leak:
In the situation that a pipe breaks in half, due to something like a tree falling on it, and all of the water is leaking out, emergency repairs would need to be made. The effect of a smaller leak due to pipes wearing down over time will be examined here.

The most common material used to transport large amounts of water above ground is reinforced concrete (1).

In the world of reinforced concrete pipes, cracks exceeding a width of 0.100 inches are repaired. (3). This would make the worst case leak a width of 0.100 in.

Concrete pipes erode over time, so leakage would likely not be an issue with brand new pipes. Assuming the worst case, salvage pipes will be used, meaning it is likely that they are already corroded and leaking.

Plugging in worst case values:
A: largest area = 0.00254^2 m = 0.0000064516 m^2

S: at colder temps water is less dense, making S smaller ultimately making Q larger. The worst case is the coldest possible temperature of 0 degrees celsius, causing S = 0.99987

C: constant 0.61

Pw,std: constant, 1.00

P: The lowest barometric pressure recorded in the US (excluding hurricanes) is 92500 N/m^2 (4), the higher end of the usual pressure in large water pipes is 600000 N/m^2 (5). This makes the change in P = 507500 N/m^2

Highest possible Q for each possible crack is = 0.003965 m^3/s

Examining Significance:
Considering a reservoir with a depth of 100m, and therefore a volume of 100,000 m^3
Loss for one crack is 14.27 m^3 per hour, 342.6 m^3/day
342.6/100,000 is 0.34% per day, which is not significant

Sources:

1. https://sswm.info/sswm-university-course/module-2-centralised-and-decentralised-systems-water-and-sanitation-1/water-distribution-pipes
2. https://www.lmnoeng.com/Flow/LeakRate.php
3. file:///C:/Users/alyss/Downloads/dot_22407_DS1.pdf
4. https://www.wunderground.com/blog/weatherhistorian/world-and-us-lowest-barometric-pressure-records.html
5. https://www.plasticpipeshop.co.uk/Pressure_ep_62-1.html#:~:text=Mains%20water%20pressure%20is%20typically,between%2010%20and%2016%20bar.

Things Falling into the Reservoir Research

Humans falling into reservoir:

- Some water sticks to humans - approximately equal to surface area of a human
    - Surface area of a human: 1.9 m^2 - https://www.medicinenet.com/body_surface_area/definition.htm
    - Enough drops of water to coat a human
    - 1 drop of water = $5.0*10^{-8}$ m^3
    - Assume one layer of water on human body
        - $1.9\ m^2 * (5.0*10^{-8}\ m^3)^{⅓} = 0.0069997\ m^3$
    - 1 m^3 = 1000 kg → approximately 6.9997 kg of water lost per human
    - Approximately 7kg / 100000kg = 0.007% of water lost per human - insignificant
- Logically - when a human enters a swimming pool, water level does not visibly raise
    - Mass or volume added when a human enters a large body of water is insignificant

Animals falling into reservoir:

- Similar to humans, if not less
    - Many animals are smaller than humans
- Humans - insignificant → animals - insignificant

Debris falling into reservoir:

- Clogged pipes
    - Decreases diameter of pipes that water can travel through
    - Energy lost to friction of pipes - diameter is inversely proportional
        - Diameter decreases - energy loss increases
    - Compare to equation in slide show - diameter is halved
        - Smaller diameter - 4077.5m >>2040m
        - Significant!
- Consider a filtration system or cover to avoid this potential excess energy loss

Weather Affecting Pipes

https://www.rotorooter.com/blog/pipes/why-do-pipes-burst/

- Freezing temperatures can cause expansion and damage pipes
  - What material are the pipes made of? Can we assume no damage to the pipes?

http://ddbonline.ddbst.de/VogelCalculation/VogelCalculationCGI.exe
https://www.engineeringtoolbox.com/water-dynamic-kinematic-viscosity-d_596.html

- Viscosity of water depends on temperature as denoted by the following equations given the following constants:

Vogel Equation Parameters (Viscosity in mPa*s, T in K)

No.	A	B	C	T_{min}	T_{max}
(1)	-3.7188	578.919	-137.546	273	373

$$\eta = e^{A + \frac{B}{C+T}}$$

- The secondary source seems to use a similar model as values were cross checked:

Temperature	Pressure	Dynamic viscosity			Kinematic viscosity
[°F]	[psi]	[lb_f s/ft^2 *10^{-5}]	[lb_m/(ft h)]	[cP], [mPa s]	[ft^2/s*10^{-5}]
32.02	0.9506	3.7414	4.3336	1.7914	1.9287
34	0.0962	3.6047	4.1752	1.7259	1.8579
39.2	0.1180	3.2801	3.7992	1.5705	1.6906
40	0.1217	3.2340	3.7458	1.5484	1.6668
50	0.1781	2.7276	3.1593	1.3060	1.4063
60	0.2563	2.3405	2.7109	1.1206	1.2075
70	0.3634	2.0337	2.3556	0.9737	1.0503
80	0.5076	1.7888	2.0719	0.8565	0.9250
90	0.6992	1.5896	1.8411	0.7611	0.8234
100	0.9506	1.4243	1.6497	0.6820	0.7392
110	1.277	1.2847	1.4880	0.6151	0.6682
120	1.695	1.1652	1.3496	0.5579	0.6075
130	2.226	1.0620	1.2300	0.5085	0.5551
140	2.893	0.9733	1.1273	0.4660	0.5102
150	3.723	0.8950	1.0366	0.4285	0.4706
160	4.747	0.8279	0.9589	0.3964	0.4367
170	6.000	0.7698	0.8916	0.3686	0.4074
180	7.520	0.7192	0.8330	0.3444	0.3820
190	9.349	0.6745	0.7813	0.3230	0.3596
200	11.537	0.6300	0.7297	0.3016	0.3371
212	14.710	0.5881	0.6812	0.2816	0.3163

https://sciencing.com/flow-rate-vs-pipe-size-7270380.html
https://courses.lumenlearning.com/physics/chapter/12-4-viscosity-and-laminar-flow-poiseuilles-law/

- Flow rate is affected by pressure and viscosity. If a direct relationship between velocity and viscosity is found, the change in velocity can be plugged into friction and pipe bend equations to find loss in energy. Need a way to find the loss in velocity due to an increase in viscosity.
- Poiseuille's law for flow in a tube is

$$Q = \frac{(P_2 - P_1)\pi r^4}{8\eta l}$$

where (P2-P1) represents the pressure difference, r is the radius of the pipe, l is the length of the pipe and n is the viscosity in mPas

https://www.engineeringtoolbox.com/reynold-number-water-flow-pipes-d_574.html
https://www.engineeringtoolbox.com/reynolds-number-d_237.html

- Reynold's number can be used to relate the velocity and viscosity but a "turbulence value" (Reynold's number) must be assumed. The average is 2300. Pipe bends will affect this and water may not always be in laminar flow.

https://www.engineeringtoolbox.com/fluid-velocities-pipes-d_1885.html

- Typical flow velocity is 0.9-2.4 m/s for a temperature range of 50-100 degrees F. Is this difference in velocity going to get scaled up as velocity increases or will the difference be constant? If constant, the difference is negligible.

https://www.engineeringtoolbox.com/thermal-expansion-pipes-d_283.html

- The following equation relates temperature to pipe length expansion (which thus affects flow rate and energy lost)

$dl = \alpha L_o dt$ (1)

where

dl = expansion (m, inches)

L_o = length of pipe (m, inches)

dt = temperature difference (^oC, ^oF)

α = linear expansion coefficient (m/m^{o}K, in/in^{o}F)

- temperature expansion calculator

Note that the mean expansion coefficient may vary with temperature:

Mean Expansion Coefficient 10^{-6} (in/in ^{o}F, (m/(m C)))								
Material	Temperature Range (^{o}F) (deg C)							
	- 32	32 - 212	32 - 400	32 - 600	32 - 750	32 - 900	32 - 1100	32 - 1300
Alloy Steel (1% Cr. 1/2% Mo)	7.7	8.0	8.4	8.8	9.2	9.6	9.8	
Mild Steel (0.1 - 0.2% C)	7.1	7.8	8.3	8.7	9.0	9.5	9.7	
Stainless Steel (18% Cr. 8% Ni)	10.8	11.1	11.5	11.8	12.1	12.4	12.6	12.8

- Regardless of material, within a 100 degree F range in temperature, change in pipe length can vary from 0 to 0.8 inches per 30.5m and thus is very negligible.

Team Meeting

Date: November 28 2020 **Location**: Discord **Time**: 1:00 PM EST

Meeting Facilitator:

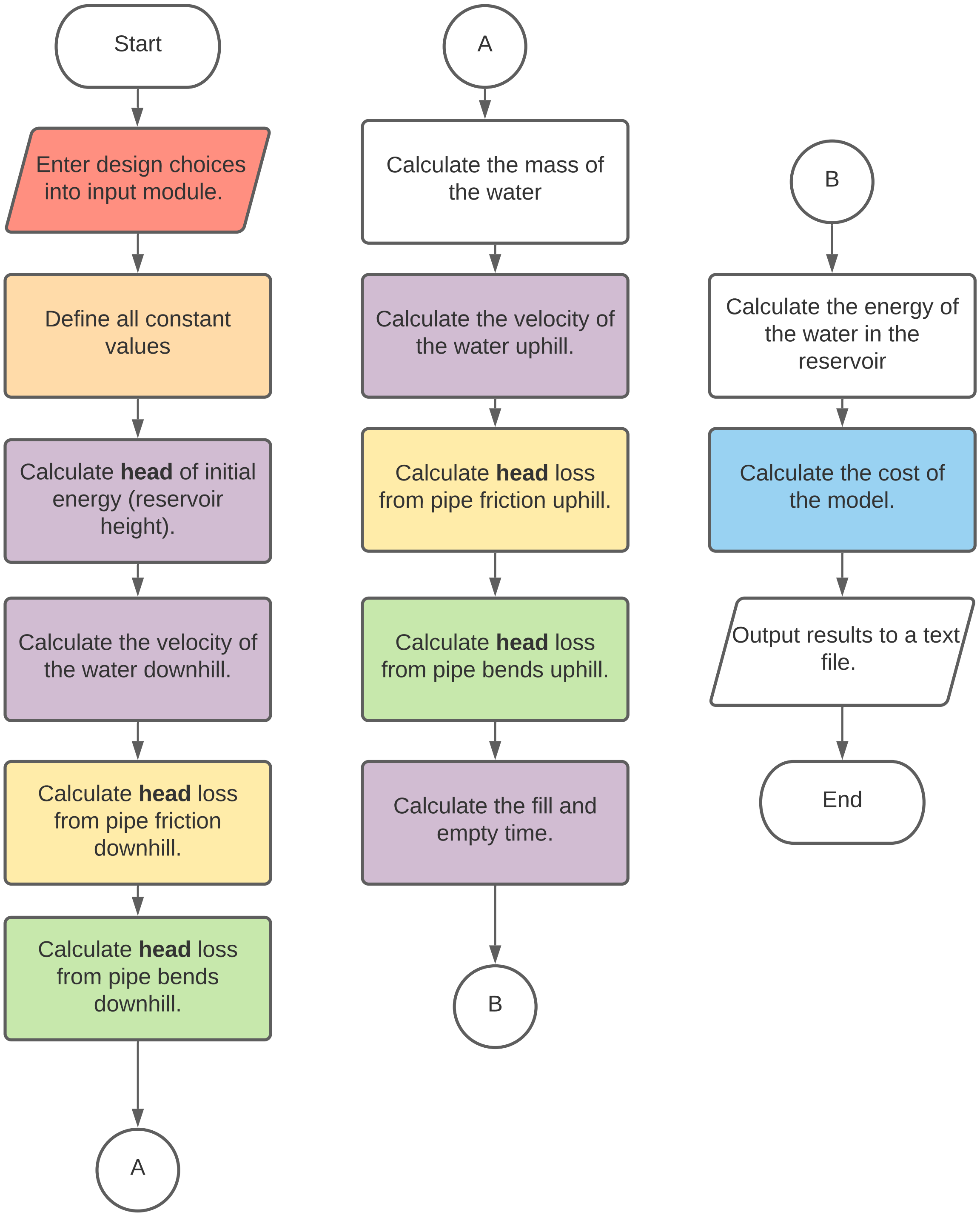
Electronic Signatures of Attendees:
- Matthew Stuber
- Natalie Harvey
- Agathiya Tharun

Agenda			
Item	**Item Description**	**Presenter**	**Minutes**
1	Flowchart Code		60
2			
3			
4			
5			

Notes	Action Items

• Write flowchart - very basic
  ○ Not including individual inputs, just calculations the program needs to make
• Add cost analysis in code if there is time
  ○ Not necessary, only for ease for the team
• Maybe physically calculate the model using physics calculations before programming - help us understand it better
	• Finish research
	• Meet at 3 on Sun

Team 30
Project 2 - Flowchart 1
Section 2
Natalie Harvey
Matthew Stuber
Alyssa Devincenzi
Agathiya Tharun



Team Meeting

Date: November 29 2020 **Location**: Discord **Time**: 3:00 PM EST

Meeting Facilitator:

Electronic Signatures of Attendees:

- Natalie Harvey
- Alyssa Devincenzi
- Matthew Stuber
- Agathiya Tharun

Agenda			
Item	**Item Description**	**Presenter**	**Minutes**
1	Discuss Research		15
2	Write and Discuss Physics Model		110
3	Begin Coding		60
4			
5			

Notes	Action Items
• Make text file or module for inputs when coding • Output results to a text file • 2 equations • Each module should return a value • Don’t include mass	Work on modules Note: Do NOT include mass in consumed energy modules Natalie: Matt: MISC functions Alyssa:Friction Aggy:

Physics Model

Up the hill

$$E_{con} = E_{pump,loss} + \sum E_{bend,loss} + E_{pipe fric, up}$$

$$E_{con} = (1-n_p)E_{in} + \sum M\left(\xi \frac{V_{up}^2}{2}\right) + M\left(\frac{fLV_{up}^2}{2D}\right)$$

UAE

$$E_f - \cancel{E_i} = E_{in} - \cancel{E_{out}} + \cancel{E_{gen}} - E_{con}$$

$$Mg\left(H+\frac{d}{2}\right) = E_{in} - E_{con}$$

$$Mg\left(H+\frac{d}{2}\right) = E_{in} - (1-n_p)E_{in} - \sum M\left(\xi \frac{V_{up}^2}{2}\right) - M\left(\frac{fLV_{up}^2}{2D}\right)$$

$$E_{in} + (n_p - 1)E_{in}$$

$$(n_p - 1 + 1)E_{in}$$

$$n_p E_{in}$$

$$\boxed{Mg\left(H+\frac{d}{2}\right) = n_p E_{in} - \sum M\left(\xi \frac{V_{up}^2}{2}\right) - M\left(\frac{fLV_{up}^2}{2D}\right)}$$

$$n_p E_{in} = Mg\left(H+\frac{d}{2}\right) + \sum M\left(\xi \frac{V_{up}^2}{2}\right) + M\left(\frac{fLV_{up}^2}{2D}\right)$$

$$\boxed{E_{in} = \frac{M}{n_p}\left(g\left(H+\frac{d}{2}\right) + \sum\left(\xi \frac{V_{up}^2}{2}\right) + \frac{fLV_{up}^2}{2D}\right)}$$

d = depth $\quad D$ = diameter

Down the hill

UAE

$$E_{out} = 120\,\text{MWh} = 4.32 \times 10^{11}\,\text{J}$$

$$E_{con} = E_{turbine,loss} + E_{pipe fric, down} + \sum E_{bend,loss}$$

$$E_{con} = E_{out}\left(\frac{1}{n_t} - 1\right) + M\left(\frac{f'LV_{down}^2}{2D'}\right) + \sum M\left(\xi \frac{V_{down}^2}{2}\right)$$

$$\cancel{E_f} - E_i = \cancel{E_{in}} - E_{out} + \cancel{E_{gen}} - E_{con}$$

$$Mg\left(H+\frac{d}{2}\right) = E_{out} + E_{con}$$

$$Mg\left(H+\frac{d}{2}\right) = E_{out} + E_{out}\left(\frac{1}{n_t} - 1\right) + M\left(\frac{f'LV_{down}^2}{2D'}\right) + \sum M\left(\xi \frac{V_{down}^2}{2}\right)$$

$$Mg\left(H+\frac{d}{2}\right) - M\left(\frac{f'LV_{down}^2}{2D'}\right) - \sum M\left(\xi \frac{V_{down}^2}{2}\right) = E_{out}\left(1 + \frac{1}{n_t} - 1\right)$$

$$M\left(g\left(H+\frac{d}{2}\right) - \frac{f'LV_{down}^2}{2D'} - \sum\left(\xi \frac{V_{down}^2}{2}\right)\right) = E_{out}\left(\frac{1}{n_t}\right)$$

$$\boxed{M = \frac{E_{out}\left(\frac{1}{n_t}\right)}{g\left(H+\frac{d}{2}\right) - \frac{f'LV_{down}^2}{2D'} - \sum\left(\xi \frac{V_{down}^2}{2}\right)}}$$

d = depth $\quad D'$ = diameter

Time

$$M_f - \cancel{M_i}^{\,0} = \rho Q_{pump} \Delta t$$

$$M = \rho Q_{pump} \Delta t$$

$$\boxed{\Delta t = \frac{M}{\rho Q_{pump}}}$$

Velocity Up

$$Q_{pump} = AV_{up}$$

$$V_{up} = \frac{Q_{pump}}{\pi\left(\frac{D}{2}\right)^2}$$

D = pipe diameter up

Velocity Down

$$Q_{turbine} = AV_{down}$$

$$V_{down} = \frac{Q_{turbine}}{\pi\left(\frac{D'}{2}\right)^2}$$

D' = pipe diameter down

Down the hill

UAE

$$E_{out} = 120\,\text{MWh} = 4320005$$

$$E_{con} = E_{turbine,loss} + E_{pipe\,fric,down} + \sum E_{bend,loss}$$

$$E_{con} = E_{out}\left(\frac{1}{n_t} - 1\right) + M\left(\frac{f' L V_{down}^2}{2D'}\right) + \sum M\left(\xi \frac{V_{down}^2}{2}\right)$$

$$\cancel{E_f} - E_i = \cancel{E_{in}} - E_{out} + \cancel{E_{gen}} - E_{con}$$

$$Mg\left(H + \frac{d}{2}\right) = E_{out} + E_{con}$$

$$Mg\left(H + \frac{d}{2}\right) = E_{out} + E_{out}\left(\frac{1}{n_t} - 1\right) + M\left(\frac{f' L V_{down}^2}{2D'}\right) + \sum M\left(\xi \frac{V_{down}^2}{2}\right)$$

$$Mg\left(H + \frac{d}{2}\right) - M\left(\frac{f' L V_{down}^2}{2D'}\right) - \sum M\left(\xi \frac{V_{down}^2}{2}\right) = E_{out}\left(1 + \frac{1}{n_t} - 1\right)$$

$$M\left(g\left(H + \frac{d}{2}\right) - \frac{f' L V_{down}^2}{2D'} - \sum\left(\xi \frac{V_{down}^2}{2}\right)\right) = E_{out}\left(\frac{1}{n_t}\right)$$

$$\boxed{M = \frac{E_{out}\left(\frac{1}{n_t}\right)}{g\left(H + \frac{d}{2}\right) - \frac{f' L V_{down}^2}{2D'} - \sum\left(\xi \frac{V_{down}^2}{2}\right)}}$$

Up the hill

UAE

$$E_{con} = E_{pump,loss} + \sum E_{bend,loss} + E_{pipe fric, up}$$

$$E_{con} = (1-n_p)E_{in} + \sum M\left(\xi \frac{V_{up}^2}{2}\right) + M\left(\frac{fLV_{up}^2}{2D}\right)$$

$$E_f - \cancel{E_i} = E_{in} - \cancel{E_{out}} + \cancel{E_{gen}} - E_{con}$$

$$Mg\left(H + \frac{d}{2}\right) = E_{in} - E_{con}$$

$$Mg\left(H+\frac{d}{2}\right) = E_{in} - (1-n_p)E_{in} - \sum M\left(\xi \frac{V_{up}^2}{2}\right) - M\left(\frac{fLV_{up}^2}{2D}\right)$$

$$\boxed{Mg\left(H+\frac{d}{2}\right) = n_p E_{in} - \sum M\left(\xi \frac{V_{up}^2}{2}\right) - M\left(\frac{fLV_{up}^2}{2D}\right)}$$

$$E_{in} + (n_p - 1)E_{in}$$

$$(n_p - 1 + 1)E_{in}$$

$$n_p E_{in}$$

$$n_p E_{in} = Mg\left(H+\frac{d}{2}\right) + \sum M\left(\xi \frac{V_{up}^2}{2}\right) + M\left(\frac{fLV_{up}^2}{2D}\right)$$

$$\boxed{E_{in} = \frac{M}{n_p}\left(g\left(H+\frac{d}{2}\right) + \sum\left(\xi \frac{V_{up}^2}{2}\right) + \frac{fLV_{up}^2}{2D}\right)}$$

Velocity Up

$$Q_{pump} = AV_{up}$$

D = pipe diameter up

$$V_{up} = \frac{Q_{pump}}{\pi\left(\frac{D}{2}\right)^2}$$

Velocity Down

$$Q_{turbine} = AV_{Down}$$

D' = pipe diameter down

$$V_{Down} = \frac{Q_{turbine}}{\pi\left(\frac{D'}{2}\right)^2}$$

Time

$$M_f - \cancel{M_i}^{\,0} = \rho Q_{pump} \Delta t$$

$$M = \rho Q_{pump} \Delta t$$

$$\boxed{\Delta t = \frac{M}{\rho Q_{pump}}}$$

Team Meeting

Date: November 30 2020 **Location**: Discord **Time**: 10:15 AM EST

Meeting Facilitator:

Electronic Signatures of Attendees:

- Matthew Stuber
- Alyssa Devincenzi
- Natalie Harvey
- Agathiya Tharun

Agenda			
Item	**Item Description**	**Presenter**	**Minutes**
1	Compile Code		100
2			
3			
4			
5			

Notes	Action Items
• Make sure code works by the end • Output file should include inputs, intermediate calcs, and outputs	• Meet Tuesday at 3:00 EST ○ Optimize

```python
# Project 2 Constants
# File: constants.py
# Date: 1 September 2014
# By: Natalie Harvey
# harve115
# Matthew Stuber
# mjstuber
# Alyssa Devincenzi
# adevinc
# Agathiya Tharun
# atharun
# Section: 2
# Team: 30
#
# ELECTRONIC SIGNATURE
# Natalie Harvey
# Matthew Stuber
# Alyssa Devincenzi
# Agathiya Tharun
#
# The electronic signatures above indicate that the program
# submitted for evaluation is the combined effort of all
# team members and that each member of the team was an
# equal participant in its creation. In addition, each
# member of the team has a general understanding of
# all aspects of the program development and execution.
#
# This program contains the constants used throughout the main program

import math as m

Eout = 4.32 * m.pow(10,11) # Required energy output (Joules)
g = 9.81 # Gravity (m/s^2)
density = 1000 # Water density (kg/m^3)
```

```python
# Project 2 Inputs
# File: inputs.py
# Date: 1 September 2014
# By: Natalie Harvey
# harve115
# Matthew Stuber
# mjstuber
# Alyssa Devincenzi
# adevinc
# Agathiya Tharun
# atharun
# Section: 2
# Team: 30
#
# ELECTRONIC SIGNATURE
# Natalie Harvey
# Matthew Stuber
# Alyssa Devincenzi
# Agathiya Tharun
#
# The electronic signatures above indicate that the program
# submitted for evaluation is the combined effort of all
# team members and that each member of the team was an
# equal participant in its creation. In addition, each
# member of the team has a general understanding of
# all aspects of the program development and execution.
#
# This module contains the input values for the main program

filename = "demoOutput" # name of the output file

#BEGIN
zone          = 1           # Zone choice
nPump         = 0.92        # Pump efficiency
Qpump         = 38          # Pump flow volume (m^3/s)
dPipe         = 3           # Pipe diameter (m)
lPipe         = [67.08203932]  # Pipe length (m) (list form)
fPipe         = 0.002       # Pipe friction coefficient
dWater        = 7           # Depth of water reservoir (m)
hWater        = 30          # Elevation of water reservoir (m)
bendCoefficient = [0.15, 0.15]   # Pipe bend coefficient (list form)
nTurbine      = 0.94        # Turbine efficiency
Qturbine      = 38          # Turbine flow volume (m^3/s)
totalPipes    = 1           # total number of pipes (up & down)
nPipe         = 0           # number of pipes in one direction (0 if 1 total pipe)
areaUnderPipes  = 0           # area below raised pipe instalation (user defined)
```

```python
# Project 2
# File: Proj2_SolarHydro_Team30.py
# Date: 1 September 2014
# By: Natalie Harvey
# harve115
# Matthew Stuber
# mjstuber
# Alyssa Devincenzi
# adevinc
# Agathiya Tharun
# atharun
# Section: 2
# Team: 30
#
# ELECTRONIC SIGNATURE
# Natalie Harvey
# Matthew Stuber
# Alyssa Devincenzi
# Agathiya Tharun
#
# The electronic signatures above indicate that the program
# submitted for evaluation is the combined effort of all
# team members and that each member of the team was an
# equal participant in its creation. In addition, each
# member of the team has a general understanding of
# all aspects of the program development and execution.
#
# This program uses the inputs from inputs.py to calculate
# the efficiency of the system and the cost of the system.
# It will append the results to a data file specified in inputs.py
#-----------------------------------------------------------
from BendLoss import bendloss
import constants as k
from headLoss import headLoss
import inputs as i
import MISC
import math
import costFinder as cost
import time
#--------------------------------------------------------------------------------
bendLossDown = 0        # Head loss from bends in the pipe going down
frictionLossDown = 0    # Head loss from friction in the pipe going down
bendLossUp = 0          # Head loss from bends in the pipe going up
frictionLossUp = 0      # Head loss from friction in the pipe going up
output = []             # Output results to text file

# Intermediate Calculations
pipeAreaUp = math.pi * ((i.dPipe / 2) ** 2)
pipeAreaDown = math.pi * ((i.dPipe / 2) ** 2)
velocityUp = MISC.WaterVelocity(i.Qpump, pipeAreaUp)
velocityDown = MISC.WaterVelocity(i.Qturbine, pipeAreaDown)
waterHead = MISC.WaterHead(i.hWater + i.dPipe, i.dWater - i.dPipe)

# Finding summation of head loss from pipe friction and pipe bends
for length in i.lPipe:
   frictionLossDown += headLoss(i.dPipe, i.fPipe, length, velocityDown)

for bend in i.bendCoefficient:
   bendLossDown += bendloss(velocityDown, bend)

for length in i.lPipe:
   frictionLossUp += headLoss(i.dPipe, i.fPipe, length, velocityUp)

for bend in i.bendCoefficient:
   bendLossUp += bendloss(velocityUp, bend)
```

```
#Key Intermediate Calculation
Mass = k.Eout * (1 / i.nTurbine)
Mass = Mass / (k.g * waterHead - (i.totalPipes - i.nPipe) * (frictionLossDown + bendLossDown))

# Calculate surface area of reservoir
area = (Mass / k.density) / (i.dWater - i.dPipe)

# Calculate Ein
Ein = Mass / i.nPump
Ein = Ein * (k.g * waterHead + (i.totalPipes - i.nPipe) * (bendLossUp + frictionLossUp))
Ein = MISC.JoulestoMWH(Ein)

# Calculate efficiency
efficiency = MISC.JoulestoMWH(k.Eout) / Ein

# Calculate fill and drain time
fillTime = MISC.Time(Mass, i.Qpump)
drainTime = MISC.Time(Mass, i.Qturbine)

# Calculate total cost of project
cost = cost.findCost(i.zone, i.nPump, i.Qpump, i.dPipe, i.lPipe, i.fPipe, i.totalPipes, i.dWater, i.hWater, area, i.bendCoefficient,
i.nTurbine, i.Qturbine, i.areaUnderPipes)

# Calculate cost per percent efficiency
unitCost = cost / (100 * efficiency)

ins = open("inputs.py", "r")
while (ins.readline().rstrip("\n") != "#BEGIN"): # read in inputs
    pass
for line in ins.readlines():
    output.append(line)

out = open(i.filename + ".txt","a")
out.write("--------------------------------------------------------------------------------\n")
out.write("Inputs:\n")
out.writelines(output)
out.write("\n\nOutput:\n")
out.write("Surface Area (m^2) = " + str(area) + "\n")
out.write("Ein          (MWh) = " + str(Ein) + "\n")
out.write("Efficiency         = " + str(efficiency) + "\n")
out.write("Fill Time  (hours) = " + str(fillTime) + "\n")
out.write("Drain Time (hours) = " + str(drainTime) + "\n")
out.write("Cost           ($)  = " + str(cost) + "\n")
out.write("$ / %Efficiency    = " + str(unitCost))
out.write("\n")

out.close()
ins.close()
```

```
---------------------------------------------------------------------------
Inputs:
zone           = 1             # Zone choice
nPump          = 0.92          # Pump efficiency
Qpump          = 38            # Pump flow volume (m^3/s)
dPipe          = 3             # Pipe diameter (m)
lPipe          = [67.08203932]  # Pipe length (m) (list form)
fPipe          = 0.002         # Pipe friction coefficient
dWater         = 7             # Depth of water reservoir (m)
hWater         = 30            # Elevation of water reservoir (m)
bendCoefficient = [0.15, 0.15]   # Pipe bend coefficient (list form)
nTurbine       = 0.94          # Turbine efficiency
Qturbine       = 38            # Turbine flow volume (m^3/s)
totalPipes     = 1             # total number of pipes (up & down)
nPipe          = 0             # number of pipes in one direction (0 if 1 total pipe)
areaUnderPipes  = 0            # area below raised pipe instalation (user defined)

Output:
Surface Area (m^2) = 339551.527339109
Ein          (MWh) = 142.84591401266854
Efficiency         = 0.8400660307956557
Fill Time  (hours) = 9.92840723213769
Drain Time (hours) = 9.92840723213769
Cost          ($)  = 517288.3113453164
$ / %Efficiency    = 6157.710136849299
```

```python
# Project 2
# File: costFinder.py
# Date: 1 September 2014
# By: Natalie Harvey
# harve115
# Matthew Stuber
# mjstuber
# Alyssa Devincenzi
# adevinc
# Agathiya Tharun
# atharun
# Section: 2
# Team: 30
#
# ELECTRONIC SIGNATURE
# Natalie Harvey
# Matthew Stuber
# Alyssa Devincenzi
# Agathiya Tharun
#
# The electronic signatures above indicate that the program
# submitted for evaluation is the combined effort of all
# team members and that each member of the team was an
# equal participant in its creation. In addition, each
# member of the team has a general understanding of
# all aspects of the program development and execution.
#
# This module accepts design choices, calculates the total cost and returns
# it to the calling function.
import partsTables as unit
import WallConstructionCosts as wall

grateCost = 0


def findCost(zone, nPump, QPump, dPipe, lPipe, fPipe, tPipes, depth, head, area, bend, nTurbine, QTurbine, pipeArea):

    # price of pump
    cost = QPump * unit.priceLookup(nPump, depth + head, "pump")


    #price of pipes
    for length in lPipe:
        cost += tPipes * (length * (500 + unit.priceLookup(fPipe, dPipe, "pipe") + pipeArea * 250))

    #price of pipe bends
    for b in bend:
        cost += tPipes * unit.priceLookup(b, dPipe, "bend")

    #price of wall and other zone costs
    cost += wall.Cost(depth, area, zone)

    #price of turbine
    cost += QTurbine * unit.priceLookup(nTurbine, depth + head, "turbine")

    #misc extra costs
    cost += grateCost * tPipes

    return cost
```

```
# Project 2
# File: WallConstructionCosts.py
# Date: 1 September 2014
# By: Natalie Harvey
# harve115
# Matthew Stuber
# mjstuber
# Alyssa Devincenzi
# adevinc
# Agathiya Tharun
# atharun
# Section: 2
# Team: 30
#
# ELECTRONIC SIGNATURE
# Natalie Harvey
# Matthew Stuber
# Alyssa Devincenzi
# Agathiya Tharun
#
# The electronic signatures above indicate that the program
# submitted for evaluation is the combined effort of all
# team members and that each member of the team was an
# equal participant in its creation. In addition, each
# member of the team has a general understanding of
# all aspects of the program development and execution.
#
# This module deals with the wall construction costs. zoneDiction contains
# data associated with each individual size which can be accessed as needed by
# the funcitons in this module.
import math as m

zoneDict = { # contains data for each zone to be used in cost analysis

    "1": [150000, .25, 282743, 360000, "2 * m.pi * m.sqrt(area / m.pi)", "4 * m.sqrt(area)"],

    "2": [208000, .5, 20000, 25617.37, "3 * m.sqrt(2 * area)", "2 * m.sqrt(m.pow(area / 100, 2) + 10000) + 200"],

    "3": [250000, .62, 39760.78, 39760.78, "2 * m.pi * m.sqrt(area / m.pi)", "2 * m.pi * m.sqrt(area / m.pi)"]
} #"zone" [fixed cost, variable cost, optimum area cutoff, max area, optimum perimeter, secondary optimum perimter]


# returns linear cost of the wall / m
def WallUnitCost (depth):
    prices = [30,60,95,135,180,250,340]
    heights = [5,7.5,10,12.5,15,17.5,20]

    if depth in heights:
        return prices[heights.index(depth)]
    else: # use linear interpolation
        for x in range(len(heights)):
            if depth > heights[x] and depth < heights[x+1]:
                #print("between",heights[x],"and",heights[x+1])
                y2 = prices[x+1]
                y1 = prices[x]
                x2 = heights[x+1]
                x1 = heights[x]
                M = (y2 - y1) / (x2 - x1)
                return ((M * depth) - y1)

def Cost(depth, area, zone): # returns total cost of reservoir + zone fixed costs
    linearCost = WallUnitCost(depth)
    if (area <= zoneDict[str(zone)][2]):
        perimeter = eval(zoneDict[str(zone)][4])
    else:
        perimeter = eval(zoneDict[str(zone)][5])
```

```
cost = perimeter * linearCost + zoneDict[str(zone)][0] + zoneDict[str(zone)][1] * area
return cost
```

Team Meeting

Date: 12/1/20 **Location**:Discord **Time**: 3:00

Meeting Facilitator:

Electronic Signatures of Attendees:

- Natalie Harvey
- Alyssa Devincenzi
- Matthew Stuber
- Agathiya Tharun

Agenda			
Item	**Item Description**	**Presenter**	**Minutes**
1	Discuss Optimization Options		20
2	Test different values in code		90
3			
4			
5			

Notes	Action Items
• Use all research in documentation • Minimize mass • Focus on flow rate and diameter ○ Have highest effect on mass equation ○ Larger diameter, smaller flow rate • Optimize each site • Zone 1 has lowest extra costs • Depth - maximize ○ Area decreases ○ Ein slightly decreases • Depth effects are very minimal - not a main focus • Focus on cost or efficiency? ○ Find a balance between the two	Plan next meeting time

```
--------------------------------------------------------------------------
Inputs:
zone            = 1             # Zone choice
nPump           = 0.92          # Pump efficiency
Qpump           = 78            # Pump flow volume (m^3/s)
dPipe           = 3             # Pipe diameter (m)
lPipe           = [67.3]        # Pipe length (m)
fPipe           = 0.002         # Pipe friction coefficient
dWater          = 8             # Depth of water reservoir (m)
hWater          = 60            # Elevation of water reservoir (m)
bendCoefficient = [0.15, 0.15]  # Pipe bend coefficient
nTurbine        = 0.92          # Turbine efficiency
Qturbine        = 33            # Turbine flow volume (m^3/s)
totalPipes      = 1             # total number of pipes (up & down)
nPipe           = 0             # number of pipes in one direction (0 if 1 total pipe)

Output:
Surface Area (m^2) = 147015.5284415723
Ein         (MWh) = 147.27108314221675
Efficiency        = 0.8148239113860417
Fill Time  (hours) = 2.617797871110618
Drain Time (hours) = 6.187522240806915
Cost          ($)  = 446077.0981385484
$ / %Efficiency    = 5474.521450650078
--------------------------------------------------------------------------
Inputs:
zone            = 1             # Zone choice
nPump           = 0.92          # Pump efficiency
Qpump           = 78            # Pump flow volume (m^3/s)
dPipe           = 3             # Pipe diameter (m)
lPipe           = [67.3]        # Pipe length (m)
fPipe           = 0.002         # Pipe friction coefficient
dWater          = 8             # Depth of water reservoir (m)
hWater          = 60            # Elevation of water reservoir (m)
bendCoefficient = [0.15, 0.15]  # Pipe bend coefficient
nTurbine        = 0.92          # Turbine efficiency
Qturbine        = 33            # Turbine flow volume (m^3/s)
totalPipes      = 1             # total number of pipes (up & down)
nPipe           = 0             # number of pipes in one direction (0 if 1 total pipe)

Output:
Surface Area (m^2) = 147015.5284415723
Ein         (MWh) = 147.27108314221675
Efficiency        = 0.8148239113860417
Fill Time  (hours) = 2.617797871110618
Drain Time (hours) = 6.187522240806915
Cost          ($)  = 446077.0981385484
$ / %Efficiency    = 5474.521450650078
--------------------------------------------------------------------------
Inputs:
zone            = 1             # Zone choice
nPump           = 0.92          # Pump efficiency
Qpump           = 78            # Pump flow volume (m^3/s)
dPipe           = 2.75          # Pipe diameter (m)
```

lPipe = [67.3] # Pipe length (m)
fPipe = 0.002 # Pipe friction coefficient
dWater = 11 # Depth of water reservoir (m)
hWater = 60 # Elevation of water reservoir (m)
bendCoefficient = [0.15, 0.15] # Pipe bend coefficient
nTurbine = 0.94 # Turbine efficiency
Qturbine = 33 # Turbine flow volume (m^3/s)
totalPipes = 1 # total number of pipes (up & down)
nPipe = 0 # number of pipes in one direction (0 if 1 total pipe)

Output:
Surface Area (m^2) = 85614.86180568184
Ein (MWh) = 146.32578487088043
Efficiency = 0.820087861520028
Fill Time (hours) = 2.515393909889157
Drain Time (hours) = 5.945476514283461
Cost ($) = 443826.9048814967
$ / %Efficiency = 5411.943350299883

Inputs:
zone = 1 # Zone choice
nPump = 0.92 # Pump efficiency
Qpump = 70 # Pump flow volume (m^3/s)
dPipe = 2.75 # Pipe diameter (m)
lPipe = [67.08203932] # Pipe length (m)
fPipe = 0.002 # Pipe friction coefficient
dWater = 13 # Depth of water reservoir (m)
hWater = 30 # Elevation of water reservoir (m)
bendCoefficient = [0.15, 0.15] # Pipe bend coefficient
nTurbine = 0.94 # Turbine efficiency
Qturbine = 29 # Turbine flow volume (m^3/s)
totalPipes = 1 # total number of pipes (up & down)
nPipe = 0 # number of pipes in one direction (0 if 1 total pipe)

Output:
Surface Area (m^2) = 122039.20067787184
Ein (MWh) = 149.48392911109215
Efficiency = 0.802761880247471
Fill Time (hours) = 4.963896059318199
Drain Time (hours) = 11.981818074216346
Cost ($) = 458026.31206479046
$ / %Efficiency = 5705.631063642342

Inputs:
zone = 1 # Zone choice
nPump = 0.92 # Pump efficiency
Qpump = 70 # Pump flow volume (m^3/s)
dPipe = 2.75 # Pipe diameter (m)
lPipe = [67.08203932] # Pipe length (m)
fPipe = 0.002 # Pipe friction coefficient
dWater = 13 # Depth of water reservoir (m)
hWater = 30 # Elevation of water reservoir (m)
bendCoefficient = [0.15, 0.15] # Pipe bend coefficient
nTurbine = 0.94 # Turbine efficiency

```
Qturbine      = 29           # Turbine flow volume (m^3/s)
totalPipes    = 1            # total number of pipes (up & down)
nPipe         = 0            # number of pipes in one direction (0 if 1 total pipe)

Output:
Surface Area (m^2) = 122038.5730599078
Ein        (MWh) = 149.47900257007512
Efficiency        = 0.8027883377382352
Fill Time  (hours) = 4.963870531206567
Drain Time (hours) = 11.98175645463654
Cost         ($)  = 457747.0681996484
$ / %Efficiency   = 5701.964598654967
-----------------------------------------------------------------------------
Inputs:
zone          = 1            # Zone choice
nPump         = 0.92         # Pump efficiency
Qpump         = 70           # Pump flow volume (m^3/s)
dPipe         = 2.75         # Pipe diameter (m)
lPipe         = [67.08203932]      # Pipe length (m)
fPipe         = 0.002        # Pipe friction coefficient
dWater        = 13           # Depth of water reservoir (m)
hWater        = 30           # Elevation of water reservoir (m)
bendCoefficient = [0.15, 0.15]  # Pipe bend coefficient
nTurbine      = 0.94         # Turbine efficiency
Qturbine      = 29           # Turbine flow volume (m^3/s)
totalPipes    = 1            # total number of pipes (up & down)
nPipe         = 0            # number of pipes in one direction (0 if 1 total pipe)

Output:
Surface Area (m^2) = 122038.5730599078
Ein        (MWh) = 149.47900257007512
Efficiency        = 0.8027883377382352
Fill Time  (hours) = 4.963870531206567
Drain Time (hours) = 11.98175645463654
Cost         ($)  = 457747.0681996484
$ / %Efficiency   = 5701.964598654967
-----------------------------------------------------------------------------
Inputs:
zone          = 1            # Zone choice
nPump         = 0.92         # Pump efficiency
Qpump         = 70           # Pump flow volume (m^3/s)
dPipe         = 2.75         # Pipe diameter (m)
lPipe         = [67.08203932]      # Pipe length (m)
fPipe         = 0.002        # Pipe friction coefficient
dWater        = 13           # Depth of water reservoir (m)
hWater        = 30           # Elevation of water reservoir (m)
bendCoefficient = [0.15, 0.15]  # Pipe bend coefficient
nTurbine      = 0.94         # Turbine efficiency
Qturbine      = 29           # Turbine flow volume (m^3/s)
totalPipes    = 1            # total number of pipes (up & down)
nPipe         = 0            # number of pipes in one direction (0 if 1 total pipe)

Output:
Surface Area (m^2) = 122038.5730599078
```

```
Ein         (MWh) = 149.47900257007512
Efficiency        = 0.8027883377382352
Fill Time  (hours) = 4.963870531206567
Drain Time (hours) = 11.98175645463654
Cost          ($)  = 457747.0681996484
$ / %Efficiency    = 5701.964598654967
-----------------------------------------------------------------------------
Inputs:
zone            = 1              # Zone choice
nPump           = 0.92           # Pump efficiency
Qpump           = 70             # Pump flow volume (m^3/s)
dPipe           = 2.75           # Pipe diameter (m)
lPipe           = [67.13267284]       # Pipe length (m)67.08203932
fPipe           = 0.002          # Pipe friction coefficient
dWater          = 13             # Depth of water reservoir (m)
hWater          = 30             # Elevation of water reservoir (m)
bendCoefficient = [0.15, 0.15]   # Pipe bend coefficient
nTurbine        = 0.94           # Turbine efficiency
Qturbine        = 29             # Turbine flow volume (m^3/s)
totalPipes      = 1              # total number of pipes (up & down)
nPipe           = 0              # number of pipes in one direction (0 if 1 total pipe)

Output:
Surface Area (m^2) = 122038.71885859621
Ein         (MWh) = 149.48014702937982
Efficiency        = 0.8027821913796647
Fill Time  (hours) = 4.963876461510361
Drain Time (hours) = 11.981770769162942
Cost          ($)  = 457811.93815578875
$ / %Efficiency    = 5702.816318944388
-----------------------------------------------------------------------------
Inputs:
zone            = 1              # Zone choice
nPump           = 0.92           # Pump efficiency
Qpump           = 70             # Pump flow volume (m^3/s)
dPipe           = 2.75           # Pipe diameter (m)
lPipe           = [67.08203932]  # Pipe length (m) (list form)
fPipe           = 0.002          # Pipe friction coefficient
dWater          = 13             # Depth of water reservoir (m)
hWater          = 30             # Elevation of water reservoir (m)
bendCoefficient = [0.15, 0.15]   # Pipe bend coefficient (list form)
nTurbine        = 0.94           # Turbine efficiency
Qturbine        = 29             # Turbine flow volume (m^3/s)
totalPipes      = 1              # total number of pipes (up & down)
nPipe           = 0              # number of pipes in one direction (0 if 1 total pipe)
areaUnderPipes  = 0              # area below raised pipe instalation (user defined)

Output:
Surface Area (m^2) = 122038.5730599078
Ein         (MWh) = 149.47900257007512
Efficiency        = 0.8027883377382352
Fill Time  (hours) = 4.963870531206567
Drain Time (hours) = 11.98175645463654
Cost          ($)  = 457747.0681996484
```

$ / %Efficiency = 5701.964598654967

Team Meeting

Date: December 2 2020 **Location**: Discord **Time**: 10:15 AM EST

Meeting Facilitator:

Electronic Signatures of Attendees:
- Natalie Harvey
- Matthew Stuber
- Alyssa Devincenzi
- Agathiya Tharun

Agenda			
Item	**Item Description**	**Presenter**	**Minutes**
1	Adjust Code		120
2	Test Values		30
3	Discuss Results		10
4	Begin Poster		60
5			

Notes	Action Items
Poster Presentation: - Look over poster presentation module on Brightspace - Content: present everything that can defend our decision - Treat it as a pitch to a company about the model Decision matrices	Continue to work on poster

```
# Project 2 Inputs for Analysis
# File: inputsAnalysis.py
# Date: 1 September 2014
# By: Matthew Stuber
# mjstuber
# Natalie Harvey
# harve115
# Alyssa Devincenzi
# adevinc
# Agathiya Tharun
# atharun
# Section: 2
# Team: 30
#
# ELECTRONIC SIGNATURE
# Natalie Harvey
# Matthew Stuber
# Alyssa Devincenzi
# Agathiya Tharun
#
# The electronic signatures above indicate that the program
# submitted for evaluation is the combined effort of all
# team members and that each member of the team was an
# equal participant in its creation. In addition, each
# member of the team has a general understanding of
# all aspects of the program development and execution.
#
# This program contains the input values for the main program

#BEGIN
zone            = 3                                  # Zone choice
nPump           = [0.89, .92]                        # Pump efficiency
Qpump           = [x for x in range(25, 51, 1)]      # Pump flow volume (m^3/s)
dPipe           = [float(d/100) for d in range(75, 325, 25)]   # Pipe diameter (m)
lPipe           = [43.84412423,70.71380809]          # Pipe length (m)
fPipe           = [.05,.03,.02,.01,.005,.002]        # Pipe friction coefficient
dWater          = [d for d in range(5, 21)]          # Depth of water reservoir (m)
hWater          = 65                                 # Elevation of water reservoir (m)
bendCoefficient = [.1, .1, .2, .2]                   # Pipe bend coefficient
nTurbine        = [0.89, .92, .94]                   # Turbine efficiency
Qturbine        = [x for x in range(10, 40, 1)]      # Turbine flow volume (m^3/s)
totalPipes      = [1,2,4,6]                          # total number of pipes (up & down)
nPipe           = [0,1,2,3]                          # number of pipes in one direction (0 if 1 total pipe)
areaUnderPipes  = 0                                  # area below raised pipe instalation (user defined)
```

```
# Analysis Function
from BendLoss import bendloss
import constants as k
from headLoss import headLoss
import inputsAnalysis as i
import MISC
import math
import costFinder as cost
import WallConstructionCosts as wall
#---------------------------------------------------------------------------
case = 1
filename = "zone {} analysis".format(str(i.zone))
out = open("./" + filename + ".txt", "w")
out.write("Zone = {},Elevation = {},Length = ".format(i.zone, i.hWater))
for length in i.lPipe:
    out.write("-{}".format(length))
out.write(",Bends = ")
for b in i.bendCoefficient:
    out.write("-{}".format(b))
out.write(",\n")
out.write("Case,Pump Efficiency, Pump Flow Rate, Pipe Diameter, Pipe friction Coefficient, Turbine Efficiency, Turbine Flow Rate, Total Pipes, Depth, Ein, Cost, $ / %Efficiency\n")

def analyze(case, zone, nPump, QPump, dPipe, lPipe, fPipe, tPipes, nPipes, depth, head, bend, nTurbine, QTurbine):

    bendLossDown = 0
    frictionLossDown = 0
    bendLossUp = 0
    frictionLossUp = 0

    # Intermediate Calculations
    pipeAreaUp = math.pi * ((dPipe / 2) ** 2)
    pipeAreaDown = math.pi * ((dPipe / 2) ** 2)
    velocityUp = MISC.WaterVelocity(QPump, pipeAreaUp)
    velocityDown = MISC.WaterVelocity(QTurbine, pipeAreaDown)
    waterHead = MISC.WaterHead(head + dPipe, depth - dPipe)

    for length in lPipe:
        frictionLossDown += headLoss(dPipe, fPipe, length, velocityDown)

    for b in bend:
        bendLossDown += bendloss(velocityDown, b)

    for length in lPipe:
        frictionLossUp += headLoss(dPipe, fPipe, length, velocityUp)

    for b in bend:
        bendLossUp += bendloss(velocityUp, b)

    #Key Intermediate Calculation
    Mass = k.Eout * (1 / nTurbine)
    Mass = Mass / (k.g * waterHead - (tPipes - nPipes) * (frictionLossDown + bendLossDown))

    #Other Main Calculations
    area = (Mass / k.density) / (depth - dPipe)
    fillTime = MISC.Time(Mass, QPump)
    drainTime = MISC.Time(Mass, QTurbine)
    Ein = Mass / nPump
    Ein = Ein * (k.g * waterHead + (tPipes - nPipes) * (bendLossUp + frictionLossUp))
    Ein = MISC.JoulestoMWH(Ein)
    efficiency = MISC.JoulestoMWH(k.Eout) / Ein

    if (area < 0 or area > wall.zoneDict[str(zone)][3] or fillTime > 5 or drainTime > 12 or efficiency < .6):
        return

    tCost = cost.findCost(zone, nPump, QPump, dPipe, lPipe, fPipe, tPipes, depth, head, area, bend, nTurbine, QTurbine,
```

```
i.areaUnderPipes)
    unitCost = tCost / (100 * efficiency)

    out.write("{},{},{},{},{},{},{},{},{},{},{},{}\n".format(case,nPump, QPump, dPipe, fPipe, nTurbine, QTurbine, tPipes, depth, Ein,
tCost, unitCost))
    return

#analyze(1, .89,25,2,[115.47],.01,1,0,16,100,[.22,.22],.92,11)
for nP in i.nPump:
    for QP in i.Qpump:
        for diameter in i.dPipe:
            for friction in i.fPipe:
                for depth in i.dWater:
                    for nT in i.nTurbine:
                        for QT in i.Qturbine:
                            for x in range(0,len(i.totalPipes)):
                                analyze(case, i.zone, nP, QP, diameter, i.lPipe, friction, i.totalPipes[x], i.nPipe[x], depth, i.hWater,
i.bendCoefficient, nT, QT)
                                case += 1
out.close()
print("\aDone!")
```

```
partDict = {      # Part Catalogue, first row and column represent design choices.
    "pump":      [[  0,  .8, .83, .86, .89, .92],
                 [ 20, 200, 240, 288, 346, 415],
                 [ 30, 220, 264, 317, 380, 456],
                 [ 40, 242, 290, 348, 418, 502],
                 [ 50, 266, 319, 383, 460, 552],
                 [ 60, 293, 351, 422, 506, 607],
                 [ 70, 322, 387, 464, 557, 668],
                 [ 80, 354, 425, 510, 612, 735],
                 [ 90, 390, 468, 561, 673, 808],
                 [100, 429, 514, 617, 741, 889],
                 [110, 472, 566, 679, 815, 978],
                 [120, 519, 622, 747, 896, 1076]],

    "pipe":      [[   0,  .05,  .03,  .02,  .01, .005, .002],
                 [  .1, 1.00,  1.2, 1.44, 2.16, 2.70, 2.97],
                 [ .25, 1.20, 1.44, 1.77, 2.58, 3.23, 3.55],
                 [  .5, 2.57, 3.08, 3.70, 5.55, 6.97, 7.64],
                 [ .75, 6.30, 7.56, 9.07,   14,   17,   19],
                 [   1,   14,   16,   20,   29,   37,   40],
                 [1.25,   26,   31,   37,   55,   69,   76],
                 [ 1.5,   43,   52,   63,   94,  117,  129],
                 [1.75,   68,   82,   98,  148,  185,  203],
                 [   2,  102,  122,  146,  219,  274,  302],
                 [2.25,  144,  173,  208,  311,  389,  428],
                 [ 2.5,  197,  237,  284,  426,  533,  586],
                 [2.75,  262,  315,  378,  567,  708,  779],
                 [   3,  340,  408,  490,  735,  919, 1011]],

    "bend":      [[   0,   .1,  .15,   .2,  .22,  .27,   .3],
                 [ 0.1, 1.00, 1.05, 1.10, 1.16, 1.22, 1.28],
                 [0.25, 1.49, 1.57, 1.64, 1.73, 1.81, 1.90],
                 [0.50, 4.93, 5.17, 5.43, 5.70, 5.99,    7],
                 [0.75,   14,   15,   16,   16,   17,   18],
                 [1.00,   32,   34,   36,   38,   39,   41],
                 [1.25,   62,   65,   69,   72,   76,   80],
                 [1.50,  107,  112,  118,  124,  130,  137],
                 [1.75,  169,  178,  187,  196,  206,  216],
                 [2.00,  252,  265,  278,  292,  307,  322],
                 [2.25,  359,  377,  396,  415,  436,  458],
                 [2.50,  492,  516,  542,  569,  598,  628],
                 [2.75,  654,  687,  721,  757,  795,  835],
                 [3.00,  849,  892,  936,  983, 1032, 1084]],

    "turbine":     [[    0, 0.83, 0.86, 0.89, 0.92, 0.94],
                   [   20,   360,  432,  518,  622,  746],
                   [   30,   396,  475,  570,  684,  821],
                   [   40,   436,  523,  627,  753,  903],
                   [   50,   479,  575,  690,  828,  994],
                   [   60,   527,  632,  759,  911, 1093],
                   [   70,   580,  696,  835, 1002, 1202],
                   [   80,   638,  765,  918, 1102, 1322],
                   [   90,   702,  842, 1010, 1212, 1455],
                   [  100,   772,  926, 1111, 1333, 1600],
                   [  110,   849, 1019, 1222, 1467, 1760],
                   [  120,   934, 1120, 1345, 1614, 1936]]
}
# Looks up price for part from partDict
# column and row represnt design choices
# kw specifies which part table to acces
def priceLookup(col, row, kw):
    if ((kw == "pump" or kw == "turbine") and row % 10 != 0): # round up height
        while (row % 10 != 0):
            row += 1

    col = partDict[kw][0].index(col)
    for element in partDict[kw]:
```

```
        if (element[0] == row):
            row = partDict[kw].index(element)
            break

    return partDict[kw][row][col]
```

zone 1 analysis_min_depth.xlsx

Open with

	A	B	C	D	E	F	G	H	I	J	K	L
1	Zone = 1	Elevation = 30	Length = -67.08203932	Bends = -0.15-0.15								
2	**Case**	**Pump Efficiency**	**Pump Flow Rate**	**Pipe Diameter**	**Pipe friction Coefficient**	**Turbine Efficiency**	**Turbine Flow Rate**	**Total Pipes**	**Depth**	**Ein**	**Cost**	**$ / %Efficiency**
3	9552417	0.92	70	2.75	0.002	0.94	29	1	13	149.4790026	457747.0682	5701.964599
4	9851937	0.92	71	2.75	0.002	0.94	29	1	13	149.7422532	458299.0682	5718.894593
5	9572385	0.92	70	3	0.01	0.94	29	1	13	150.0915934	457243.5938	5719.034964
6	9552421	0.92	70	2.75	0.002	0.94	30	1	13	149.5977603	458813.999	5719.795554
7	9567397	0.92	70	3	0.02	0.94	30	1	13	155.2004414	442393.1353	5721.634155
8	10151457	0.92	72	2.75	0.8028	0.94	29	1	13	150.0092379	458851.0682	5735.991588
9	9871905	0.92	71	3	0.01	0.94	29	1	13	150.3698894	457795.5938	5736.556067
10	9851941	0.92	71	2.75	0.002	0.94	30	1	13	149.8612201	459365.999	5736.762424
11	9572389	0.92	70	3	0.01	0.94	30	1	13	150.2176586	458315.837	5737.260996
12	9552425	0.92	70	2.75	0.002	0.94	31	1	13	149.7207423	459883.5037	5737.841628
13	9567401	0.92	70	3	0.02	0.94	31	1	13	155.3940717	443503.8412	5743.155643
14	9866917	0.92	71	3	0.02	0.94	30	1	13	155.6001043	442945.1353	5743.52577
15	9547429	0.92	70	2.75	0.005	0.94	30	1	13	151.9050132	454285.2437	5750.683828
16	10450977	0.92	73	2.75	0.002	0.94	29	1	13	150.2799567	459403.0682	5753.256099
17	10151461	0.92	72	2.75	0.002	0.94	30	1	13	150.1284169	459917.999	5753.896758
18	10171425	0.92	72	3	0.01	0.94	29	1	13	150.6521328	458347.5938	5754.253549
19	9871909	0.92	71	3	0.01	0.94	30	1	13	150.4961884	458867.837	5754.821703
20	9851945	0.92	71	2.75	0.002	0.94	31	1	13	149.9844187	460435.5037	5754.845946
21	10150837	0.92	72	2.75	0.002	0.94	30	1	11	150.4405407	459091.2382	5755.494507
22	9572393	0.92	70	3	0.01	0.94	31	1	13	150.3482201	459390.8479	5755.716358
23	9552429	0.92	70	2.75	0.002	0.94	32	1	13	149.8479689	460955.5925	5756.104941
24	9821989	0.92	71	2.5	0.002	0.94	30	1	13	155.3936865	444720.5399	5758.897011
25	9567405	0.92	70	3	0.02	0.94	32	1	13	155.5945585	444618.6288	5765.019936
26	9866921	0.92	71	3	0.02	0.94	31	1	13	155.7942333	444055.8412	5765.111609
27	10166437	0.92	72	3	0.02	0.94	30	1	13	156.0054362	443497.1353	5765.663669
28	9846949	0.92	71	2.75	0.005	0.94	30	1	13	152.2245631	454837.2437	5769.783391
29	9547433	0.92	70	2.75	0.005	0.94	31	1	13	152.0565049	455370.992	5770.176789
30	10750497	0.92	74	2.75	0.002	0.94	29	1	13	150.5544095	459955.0682	5770.68864
31	10450981	0.92	73	2.75	0.002	0.94	30	1	13	150.3993507	460469.999	5771.199074
32	10151465	0.92	72	2.75	0.002	0.94	31	1	13	150.2518351	460987.5037	5772.0182
33	10470945	0.92	73	3	0.01	0.94	29	1	13	150.9383237	458899.5938	5772.127955
34	10171429	0.92	72	3	0.01	0.94	30	1	13	150.7786689	459419.837	5772.55929
35	10450357	0.92	73	2.75	0.002	0.94	30	1	11	150.7189133	459643.2382	5773.077448

fx Zone = 2

	A	B	C	D	E	F	G	H	I	J	K	L
1	Zone = 2	Elevation = 100	Length = -130-200	Bends = -0.22-0.22								
2	Case	Pump Efficienc	Pump Flow Rat	Pipe Diameter	Pipe friction Coefficien	Turbine Efficienc	Turbine Flow Rat	Total Pipe:	Deptl	Ein	Cost	$ / %Efficienc
3	5777221	0.92	24	1.75	0.002	0.94	10	1	20	144.8906071	737654.8507	8906.604928
4	5777225	0.92	24	1.75	0.002	0.94	11	1	20	145.0897287	739807.0111	8944.866541
5	5973061	0.92	25	1.75	0.002	0.94	10	1	20	145.3349559	738730.8507	8946.9513
6	5757637	0.92	24	1.5	0.002	0.94	10	1	20	151.0851098	712160.1089	8966.399021
7	5777229	0.92	24	1.75	0.002	0.94	12	1	20	145.3084431	741980.4899	8984.669152
8	5973065	0.92	25	1.75	0.002	0.94	11	1	20	145.5346881	740883.0111	8985.348164
9	6168901	0.92	26	1.75	0.002	0.94	10	1	20	145.7974414	739806.8507	8988.495497
10	5757641	0.92	24	1.5	0.002	0.94	11	1	20	151.5031375	714528.3972	9021.107837
11	5973069	0.92	25	1.75	0.002	0.94	12	1	20	145.7540734	743056.4899	9025.292512
12	5777233	0.92	24	1.75	0.002	0.94	13	1	20	145.5469254	744175.4909	9026.037886
13	6168905	0.92	26	1.75	0.002	0.94	11	1	20	145.9978092	741959.0111	9027.032513
14	6364741	0.92	27	1.75	0.002	0.94	10	1	20	146.2780636	740882.8507	9031.242396
15	5953477	0.92	25	1.5	0.002	0.94	10	1	20	151.9784684	713236.1089	9033.044288
16	5973073	0.92	25	1.75	0.002	0.94	13	1	20	145.993287	745251.4909	9066.809565
17	6168909	0.92	26	1.75	0.002	0.94	12	1	20	146.2178926	744132.4899	9067.123707
18	5777237	0.92	24	1.75	0.002	0.94	14	1	20	145.805367	746392.237	9068.999503
19	6364745	0.92	27	1.75	0.002	0.94	11	1	20	146.4790919	743035.0111	9069.924475
20	6560581	0.92	28	1.75	0.002	0.94	10	1	20	146.7768225	741958.8507	9075.196876
21	5757645	0.92	24	1.5	0.002	0.94	12	1	20	151.9636397	716940.8088	9079.077897
22	5953481	0.92	25	1.5	0.002	0.94	11	1	20	152.398968	715604.3972	9088.1143
23	6149317	0.92	26	1.5	0.002	0.94	10	1	20	152.9082907	714312.1089	9102.020299
24	5773957	0.92	24	1.75	0.005	0.94	10	1	20	149.1817421	732429.8859	9105.430527
25	6168913	0.92	26	1.75	0.002	0.94	13	1	20	146.4578674	746327.4909	9108.794393
26	5973077	0.92	25	1.75	0.002	0.94	14	1	20	146.2525212	747468.237	9109.926181
27	6364749	0.92	27	1.75	0.002	0.94	12	1	20	146.6999008	745208.4899	9110.167629
28	5777241	0.92	24	1.75	0.002	0.94	15	1	20	146.0839767	748630.9712	9113.582446

fx Zone = 3

	A	B	C	D	E	F	G	H	I	J	K	L
1	Zone = 3	Elevation = 65	Length = -43.84412423-70.7138080	Bends = -0.1-0.1-0.2-0.2								
2	Case	Pump Efficienc	Pump Flow Rate	Pipe Diameter	Pipe friction Coefficien	Turbine Efficienc	Turbine Flow Rat	Total Pipe:	Deptl	Ein	Cost	$ / %Efficiency
3	12735261	0.92	35	2.75	0.02	0.94	15	1	20	144.2484971	653592.8143	7856.648432
4	12729501	0.92	35	2.75	0.03	0.94	15	1	20	145.8563826	646618.9198	7859.458049
5	12700701	0.92	35	2.5	0.02	0.94	15	1	20	147.3055078	640871.4567	7866.99128
6	13080861	0.92	36	2.75	0.02	0.94	15	1	20	144.5172243	654400.8143	7881.015771
7	12735265	0.92	35	2.75	0.02	0.94	16	1	20	144.3705725	655163.0315	7882.188495
8	12764061	0.92	35	3	0.03	0.94	15	1	20	143.4683196	659502.4533	7884.809062
9	12729505	0.92	35	2.75	0.03	0.94	16	1	20	146.0160223	648223.0771	7887.579606
10	13075101	0.92	36	2.75	0.03	0.94	15	1	20	146.2038407	647426.9198	7888.02519
11	12700705	0.92	35	2.5	0.02	0.94	16	1	20	147.4997021	642504.9463	7897.44068
12	13046301	0.92	36	2.5	0.02	0.94	15	1	20	147.7239231	641679.4567	7899.283892
13	12683421	0.92	35	2.25	0.002	0.94	15	1	20	144.7810531	654740.9706	7899.507268
14	13426461	0.92	37	2.75	0.02	0.94	15	1	20	144.7935212	655208.8143	7905.832614
15	13080865	0.92	36	2.75	0.02	0.94	16	1	20	144.6395271	655971.0315	7906.611648
16	13109661	0.92	36	3	0.03	0.94	15	1	20	143.698845	660310.4533	7907.154122
17	12758301	0.92	35	3	0.05	0.94	15	1	20	145.5437333	652028.958	7908.227396
18	12735269	0.92	35	2.75	0.02	0.94	17	1	20	144.5007509	656740.8518	7908.295521
19	12764065	0.92	35	3	0.03	0.94	16	1	20	143.572462	661056.9799	7909.131511
20	12648861	0.92	35	2	0.002	0.94	15	1	20	148.6375035	638776.2127	7912.175127
21	13075105	0.92	36	2.75	0.03	0.94	16	1	20	146.3638607	649031.0771	7916.224512
22	12729509	0.92	35	2.75	0.03	0.94	17	1	20	146.1863457	649837.1415	7916.443084
23	13420701	0.92	37	2.75	0.03	0.94	15	1	20	146.5610864	648234.9198	7917.167841
24	12677661	0.92	35	2.25	0.005	0.94	15	1	20	146.1021253	650470.1232	7919.588951
25	12706461	0.92	35	2.5	0.01	0.94	15	1	20	144.7062551	656748.5216	7919.634925
26	13029021	0.92	36	2.25	0.002	0.94	15	1	20	145.0758571	655548.9706	7925.360733
27	12683425	0.92	35	2.25	0.002	0.94	16	1	20	144.9154799	656320.5752	7925.917595
28	12700709	0.92	35	2.5	0.02	0.94	17	1	20	147.7069885	644150.3605	7928.792489
29	13046305	0.92	36	2.5	0.02	0.94	16	1	20	147.918669	643312.9463	7929.832895
30	13455261	0.92	37	3	0.03	0.94	15	1	20	143.935864	661118.4533	7929.887983
31	13772061	0.92	38	2.75	0.02	0.94	15	1	20	145.077388	656016.8143	7931.10049
32	13426465	0.92	37	2.75	0.02	0.94	16	1	20	144.9160579	656779.0315	7931.485678
33	13109665	0.92	36	3	0.03	0.94	16	1	20	143.8031547	661864.9799	7931.522676
34	13391901	0.92	37	2.5	0.02	0.94	15	1	20	148.1541247	642487.4567	7932.263899
35	13080869	0.92	36	2.75	0.02	0.94	17	1	20	144.769948	657548.8518	7932.776092
36	12671901	0.92	35	2.25	0.01	0.94	15	1	20	148.3125494	641863.808	7933.038142
37	12764069	0.92	35	3	0.03	0.94	17	1	20	143.6834896	662618.0532	7933.939511
38	12735273	0.92	35	2.75	0.02	0.94	18	1	20	144.6390758	658326.3072	7934.975719
39	12758305	0.92	35	3	0.05	0.94	16	1	20	145.6960047	653627.5884	7935.910682
40	13103901	0.92	36	3	0.05	0.94	15	1	20	145.8758823	652836.958	7936.097273
41	12769821	0.92	35	3	0.02	0.94	15	1	20	142.434194	668738.4137	7937.601415
42	12694941	0.92	35	2.5	0.03	0.94	15	1	20	149.9198146	635879.1808	7944.240739
43	12648865	0.92	35	2	0.002	0.94	16	1	20	148.8640447	640434.7717	7944.809206
44	13075109	0.92	36	2.75	0.03	0.94	17	1	20	146.5345898	650645.1415	7945.168245

Team Meeting

Date: December 3 2020 **Location**: Discord **Time**: 9:30 AM EST

Meeting Facilitator:

Electronic Signatures of Attendees:

- Natalie Harvey
- Matthew Stuber
- Alyssa Devincenzi
- Agathiya Tharun

Agenda			
Item	**Item Description**	**Presenter**	**Minutes**
1	Prepare Poster		240
2			
3			
4			
5			

Notes	Action Items
• Start with description of the problem • Discuss all factors that cause energy loss (class slides) • Discuss extraneous factors that were researched • Discuss physics model • Discuss the code • Discuss social factors and final solution	• Present the poster ○ Each column about 4 mins

Engineering Specifications

Customer Need	Technical Need	Technical Requirement	Target Value
High Energy Efficiency	Efficiency (Eout/Ein)	Efficiency > 70%	Same
Low Cost to Build/Operate	Total Cost of Project	Total Cost < $750,000	Cost < $500,000
Few Environmental/Cultural Concerns	Potenial to Postpone Project Indefinitely	No	No

Zone Characteristics

Characteristics	Zone 1	Zone 2	Zone 3
Fixed Cost ($)	$150,000	$208,000	$250,000
Site Preparation Cost ($/m^2)	$0.25	$0.50	$0.62
Elevation from River (m)	30	100	65
Horizonal Distance from River (m)	60	130	91.2
Maximum Area (m^2)	360000	27500	39760
Max Fill / Drain Time (h)	5 / 12	5 / 12	5 / 12
Eout (MWh)	120	120	120
Max Wall Height (m)	20	20	20
Potential Social Factors	Hazardous chemicals exist in the soil.	Potential ancestrial burial site for local indigenous population.	Need for long-term erosion monitoring.

Top Results From Each Zone

Results	Zone 1	Zone 2	Zone 3
Efficency	80.28%	82.82%	83.19%
Cost	$457,747.07	$737,654.85	$653,592.82
Cost / % Efficiency	$5,701.88	$8,906.72	$7,856.63

Decision Matrix

				Potential Sites		
Customer Need	Technical Need	Normalizing Function	Weight	Zone 1	Zone 2	Zone 3
High Energy Efficiency	Efficiency (Eout/Ein)	efficency/100	0.35	0.281	0.29	0.291
Low Cost to Build/Operate	Total Cost of Project	1- (cost/800,000)	0.45	0.427	0.0779	0.183
Few Environmental/Cultural Concerns	Potenial to Postpone Project Indefinitely	1- (potential/1)	0.2	0.2	0	0.2
			Total:	**0.908**	**0.368**	**0.674**

Final Design Choices

Optimal Zone 1 Design Choices	
Pump Efficency	0.92
Pump Flow Rate	70 m^3/s
Pipe Friction Coefficent	0.002
Number of Pipes	1
Length of Pipes	67.08 m
Pipe Diamiter	2.75 m
Turbine Efficency	0.94
Turbine Flow Rate	29 m^3/s
Bend Angles	2 at 30°
Bend Coefficients	2 at 0.15
Wall Height	13 m

Picture of Final Design