Modeling a Solar-Hydro Storage System - Project 2

Team 30

Alyssa Devincenzi, Natalie Harvey, Matthew Stuber, Agathiya Tharun

Seymour Crystals, Inc.
Potter Engineering Center, Room 318
500 Central Drive
West Lafayette, IN 47907-2022

December 6, 2020

To Dr. Seymour Glass,

As the need for sustainable energy grows, and solar fields become common, it is necessary to design a method to reliably store the energy generated by the solar fields; solar fields cannot operate when the sun is not present. Pumped hydro storage is one of the methods already used in practice to store energy generated by the sun. However, the process of designing a pumped hydro storage system is highly complex and involves numerous design choices. In order to simplify the process, a condensed model is necessary to evaluate performance tradeoffs in addition to cost.

We have created a model for a solar-hydro storage system that successfully accomplishes this task, and we have used it to help us develop a proposed design for such a system. Our design will effectively store excess solar energy for future use at a low cost with a high efficiency. Our model has found that the most optimized system has an efficiency of 80.28% at a cost of $457,747.07.

We found that energy lost from the friction in the pipes, bends in the pipe, and turbine or pump efficiency were significant; thus, variables affecting these factors need to be carefully considered in our model when optimizing our system. Conversely, we found factors such as pipe leaks and the expansion of pipes due to temperature to be insignificant and thus do not need to be accounted for in the model.

Our model ensures that all possibilities for systems and their respective costs and efficiencies are output, thus aiding you in making the best decision for your company.

Thank you.

Sincerely,
Team 30 - Alyssa Devincenzi, Natalie Harvey, Matthew Stuber, Agathiya Tharun

Table of Contents

Executive Summary

The process used to develop a final solution started with developing a physics based model that allowed the team to conceptualize the transfer of energies within the system. Next, two programs were developed using the equations from the physics model to assist in the computation of quantitative information for the given set of design parameters.

The first program utilizes an easily modifiable input file and outputs the surface area of the reservoir, input energy, efficiency, fill time, drain time, cost, and the cost to efficiency ratio. The outputs and the inputs used to obtain the results are written to a text file for ease of access and comparison. This program is compatible with all three zones.

A second program is designed to iterate through every possible combination of parameters and output the design parameters, input energy, cost, and cost to efficiency ratio for each combination to a CSV file. This can then be converted to an Excel file and sorted to highlight the best cost to efficiency ratio. This program also excludes all combinations of values that do not meet the desired fill and drain time, maximum reservoir wall height, or site surface area requirements. Once the highest cost to efficiency combination was found for each site, a decision matrix that considered cost, efficiency, and social factors was used to determine the most optimal design.

However, the model does have some limitations. First, the model does not account for the water lost from being absorbed into the ground. In addition, the model does not take the long term costs into account, such as any maintenance costs that may arise as a result of the normal wear and tear on the structure. Lastly, the program's efficiency outputs are limited by fill time. If the user had the option to input their own maximum fill time, more outputs and possibilities for efficiencies could exist.

The optimal design uses Zone 1, with a total cost of \$457,747.07, and efficiency of 80.28%. This is achieved with a pump efficiency of 0.92, a pump flow rate of 70 m^3/s, a friction coefficient of 0.002, one pipe with a length of 67.06m and a diameter of 2.75m, a turbine efficiency of 0.94, a turbine flow rate of m^3/s, two bends of 30°, and a wall height of 13m.

Cost Impact Analysis

Assumptions:

Site Layout

Given the numerous options for parts and site setups, it was necessary to make a few generalized assumptions about how the site would be laid out to simplify the model's development. First, it was decided to keep the pipe diameter consistent across all pipes per simulation. Second, the reservoir had to have a height less than 20m with a minimum depth greater than or equal to the pipe diameter to cover the pipe opening. Finally, it was assumed that the reservoir can allow for multiple pipes.

Time Limitations

To store generated energy from the solar field and be able to support the power grid, the system needed to account for the fill and drain time. On the worst case day, the time of peak solar energy generation averages 5 hours (Sengupta et al., 2018). Furthermore, the project proposal asked that the facility generate 120 MWh, or 10 MW for 12 hours. In order to meet these requirements, the maximum fill time and drain time were set at 5 and 12 hours respectively. To calculate the drain time, the equation in Figure 1 was used.

$$\Delta t = \frac{M}{\rho Q}$$

Figure 1: Fill/Drain Time Equation

Physics Model:

General Equations

The major portions of the physics model were derived from the Universal Accounting Equation, as this made it easier to account for energy loss that arises from the movement of water through the system.

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

Figure 2: Universal Accounting Equation

The water started in the river, a reference point, and traveled up into the reservoir via the pump before going back downhill, through the turbine, and back into the river. Energy is lost as the water travels through the system due to several different factors shown in Figure 3. In addition, a general energy-consumed equation was used for each part of the system.

$$E_{con} = E_{pump,\,loss} + E_{turbine,\,loss} + \Sigma E_{pipe\,fric} + \Sigma E_{bend,\,loss}$$
$$E_{con} = (1 - n_p)E_{in} + E_{out}(\frac{1}{n_t} - 1) + \Sigma M(\frac{fLV^2}{2D}) + \Sigma M(\xi \frac{V^2}{2})$$

Figure 3: Energy-Consumed Equation

Since velocity is not an input, a separate calculation was required to find it. It was based on the cross-sectional area of the pipe and the water's flow rate.

$$V = \frac{Q}{\pi(0.5D)^2}$$

Figure 4: Velocity Equation

Downhill

Knowing the final energy to be 120 MWh, the model solves for the initial energy of this portion of the system in terms of gravitational potential energy of the water's mass at the top of the hill. Once the mass of the water is known, it can be used to calculate the input energy needed to move the mass from the river to the top of the hill. The consumed energy for the downhill portion did not include the energy lost to the pump, as the water does not pass through the pump in this section.

$$Mg(H + \frac{d}{2}) = E_{out} + E_{con}$$
$$Mg(H + \frac{d}{2}) = E_{out} + E_{out}(\frac{1}{n_t} - 1) + \Sigma M(\frac{f'LV^2_{down}}{2D'}) + \Sigma M(\xi \frac{V^2_{down}}{2})$$
$$Mg(H + \frac{d}{2}) - \Sigma M(\frac{f'LV^2_{down}}{2D'}) - \Sigma M(\xi \frac{V^2_{down}}{2}) = E_{out}(1 + \frac{1}{n_t} - 1)$$
$$M(g(H + \frac{d}{2}) - \Sigma(\frac{f'LV^2_{down}}{2D'}) - \Sigma(\xi \frac{V^2_{down}}{2})) = E_{out}(\frac{1}{n_t})$$
$$M = \frac{E_{out}(\frac{1}{n_t})}{g(H + \frac{d}{2}) - \Sigma(\frac{f'LV^2_{down}}{2D'}) - \Sigma(\xi \frac{V^2_{down}}{2})}$$

Figure 5: Mass Equation Derivation

The input energy for this equation is the average potential energy of the water at the top of the hill, as this is the water's initial position for the downhill section.

Uphill

The uphill calculation was completed last, as it required intermediate calculations to be completed. The consumed energy did not include the energy lost to the turbine, as the water does not pass through the turbine when going up the hill. The final energy consisted of the water's average potential energy at the top of the hill as that is the water's final location in this section.

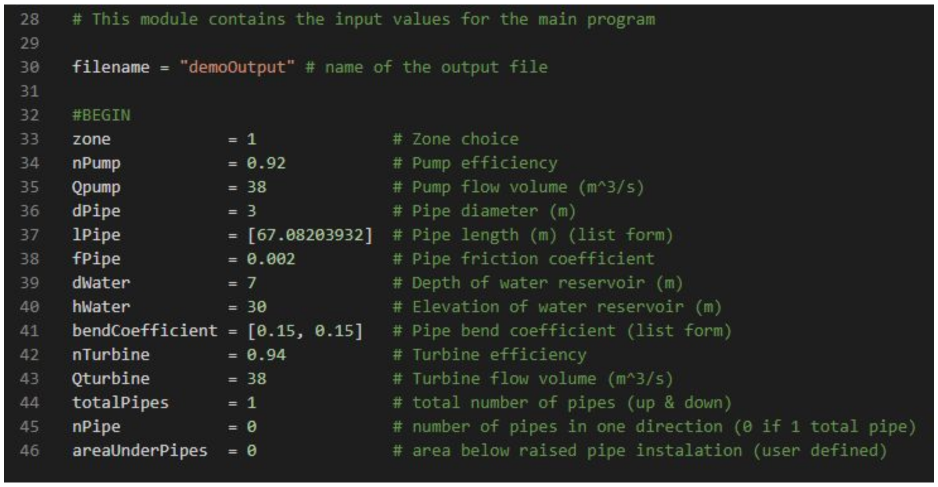
$$Mg(H + \tfrac{d}{2}) = E_{in} - E_{con}$$
$$Mg(H + \tfrac{d}{2}) = E_{in} - (1 - n_p)E_{in} - \Sigma M(\frac{fLV_{up}^2}{2D}) - \Sigma M(\xi \frac{V_{up}^2}{2})$$
$$Mg(H + \tfrac{d}{2}) = n_p E_{in} - \Sigma M(\frac{fLV_{up}^2}{2D}) - \Sigma M(\xi \frac{V_{up}^2}{2})$$
$$n_p E_{in} = Mg(H + \tfrac{d}{2}) + \Sigma M(\frac{fLV_{up}^2}{2D}) + \Sigma M(\xi \frac{V_{up}^2}{2})$$
$$E_{in} = \frac{M}{n_p}(g(H + \tfrac{d}{2}) + \Sigma(\frac{fLV_{up}^2}{2D}) + \Sigma(\xi \frac{V_{up}^2}{2}))$$

Figure 6: Energy Input Derivation

Program Development:

Inputs

The program used to compute efficiency and costs of the system accepts inputs from an input file titled "inputs.py". A screen capture of the input is shown in Figure 7. Here, the user can edit all the system parameters before running the code. This method is more efficient since inputs do not have to be reentered for every run.

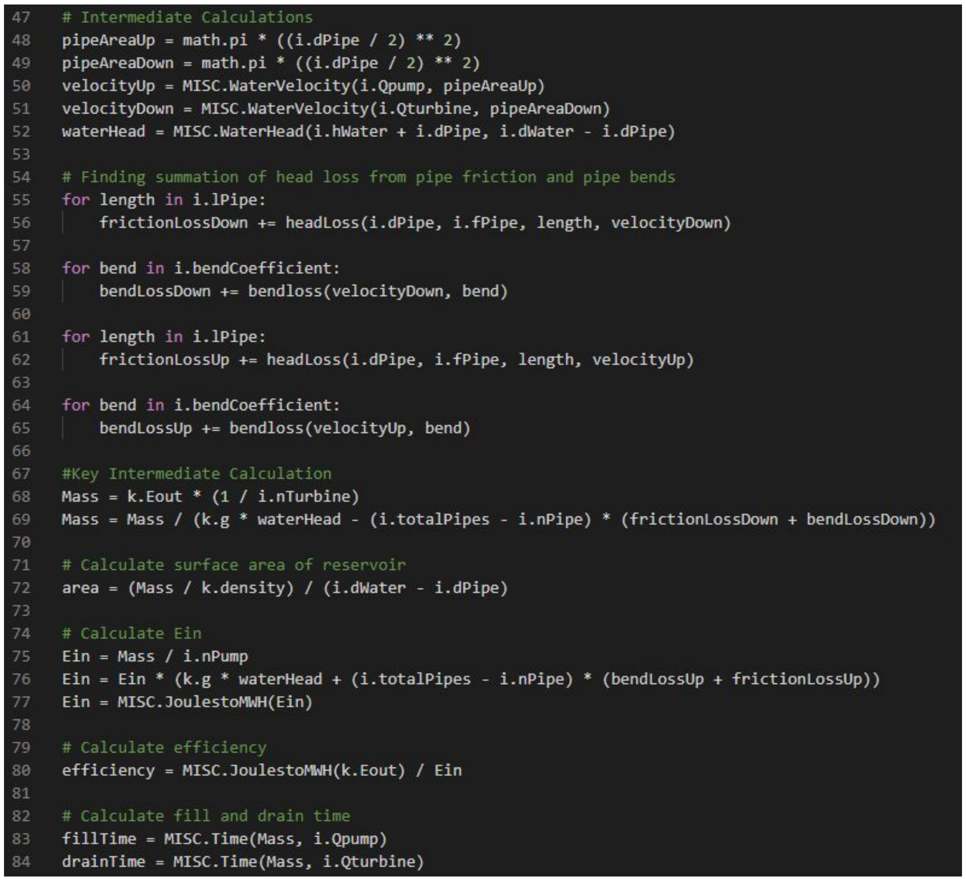


```
# This module contains the input values for the main program

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)
```

Figure 7: Input Module

Energy Calculation

Using the equations from the physics model, the program uses the values from the input module to compute the required input energy for the system. In addition, it also calculates the surface area of the reservoir, the fill and drain time, and the efficiency of the system.

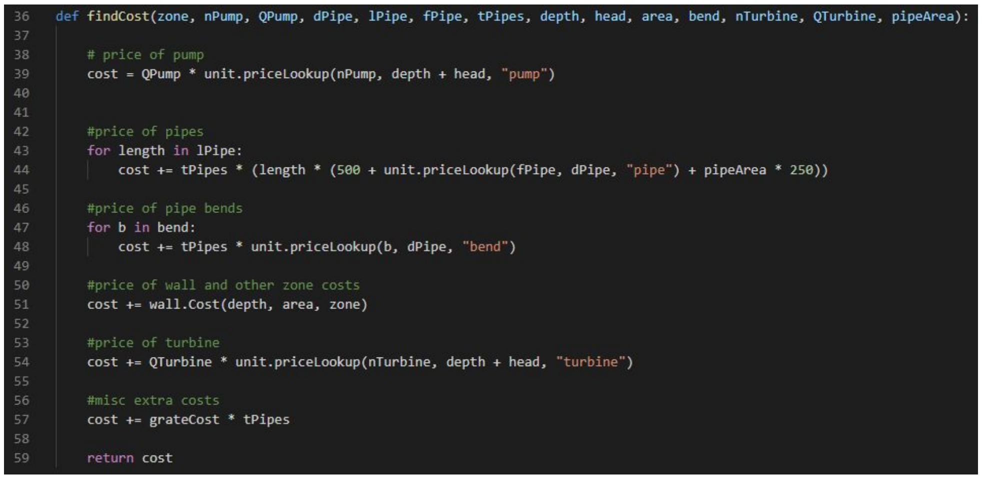


```
# Intermediate Calculations
pipeAreaUp = math.pi * ((i.dPipe / 2) ** 2)
pipeAreaDown = math.pi * ((i.dPipe / 2) ** 2)
velocityUp = MISC.WaterVelocity(i.Qpump, pipeAreaUp)
velocityDown = MISC.WaterVelocity(i.Qturbine, pipeAreaDown)
waterHead = MISC.WaterHead(i.hWater + i.dPipe, i.dWater - i.dPipe)

# 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)
```

Figure 8: Ein, Surface Area, Drain/Fill Time, and Efficiency Calculations

Cost Calculation and Cost to Efficiency Ratio

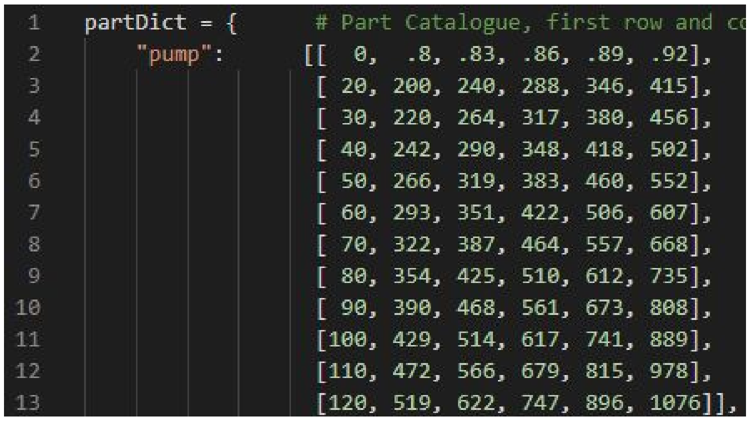
Once the energy calculation is complete, the program then calculates the total cost of the project. This includes the cost of the pumphouse, access road, site preparation, and materials used. To accomplish this task, the program makes use of the cost finder function shown in Figure 9, which uses the parameters passed in the calling function to determine the total cost.



```
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
```

Figure 9: Cost Funder Function

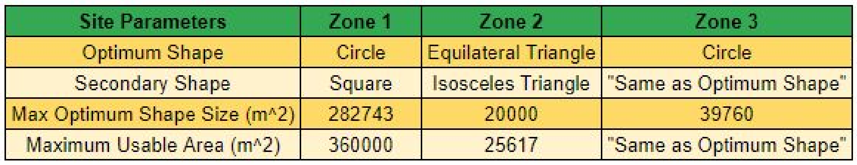
To determine the cost of parts, a price lookup function accesses the parts tables shown in Figure 10 and returns the cost of the part via the indices of the table by using the first row and column to represent design choices, such as pump efficiency, effective performance rating and more.



```
partDict = {      # Part Catalogue, first row and co
    "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]],
```

Figure 10: Example Part Table for Pump Costs

For the site costs, another function uses the surface area and wall height to calculate the cost of the reservoir and zone. The program considers how the shape of the reservoir affects the cost of the walls. In Zone 1, for example, the usable area is in the shape of a square. A square has a perimeter about 1.13 times that of the perimeter of a circle with the same area. Therefore, by using a circular shaped reservoir, the wall cost is minimized while keeping the area the same. However, the reservoir must fit within the usable area of the zone and the biggest circle that can be inscribed in Zone 1 has a maximum area of 282743 m^2. The program recognizes this, and will use the optimum shape to calculate the perimeter until the area of the reservoir exceeds that of the maximum optimal shape. In this case, it will use the secondary shape. This information is summarized below in Figure 11. Once the program finishes calculating the cost of the program, it will calculate the cost to efficiency ratio.

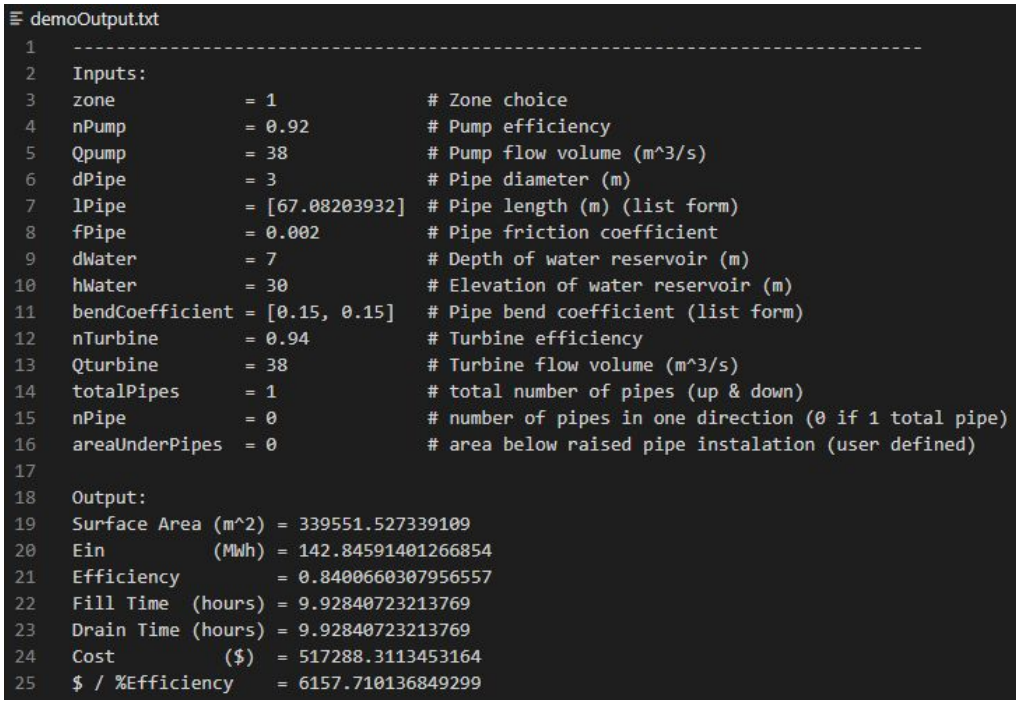


Site Parameters	Zone 1	Zone 2	Zone 3
Optimum Shape	Circle	Equilateral Triangle	Circle
Secondary Shape	Square	Isosceles Triangle	"Same as Optimum Shape"
Max Optimum Shape Size (m^2)	282743	20000	39760
Maximum Usable Area (m^2)	360000	25617	"Same as Optimum Shape"

Figure 11: Optimum Reservoir Shape Summaries

Output

Before terminating, the program will output its findings to a text file. The name of the text file is an input and can be altered in the input module and it will append the results rather than creating a new file, and thus allowing the user to easily compare runs without needing to navigate between files. Furthermore, the output will include the inputs, input energy, efficiency, cost, and cost to efficiency ratio. By including the inputs, the user can easily associate the combination of design parameters the program used with its results. An example of the output text is found in Figure 12.



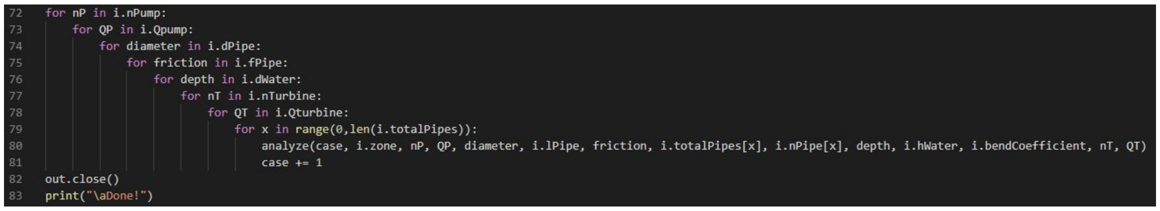
demoOutput.txt

```
--------------------------------------------------------------------------------
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
```

Figure 12: Example Output

Optimization

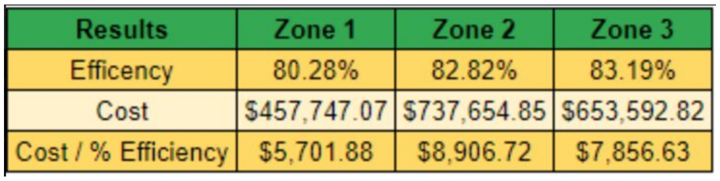
In order to determine the best combination of inputs, the logic for the main program was altered to create an additional program called "Analysis.py" (Figure 13) which accepts lists of possible inputs. Once ran, it loops through all the possible combinations, checks to make sure the model meets all the site specifications, including fill/drain time, max area, wall height, and more, and then outputs its findings to a CSV file that the user can import to Excel to perform additional analysis, such as sorting by input energy, cost, or cost to efficiency ratio.



```python
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!")
```

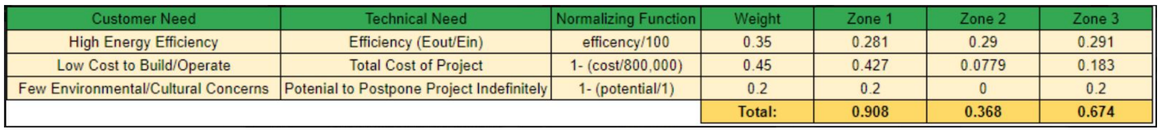
Figure 13: Nested For-Loops in "Analysis.py"

The team ran the "Analysis.py" program mentioned above once for each zone and found the results from each zone with the lowest cost to efficiency ratio. These are considered the three optimized designs and the outputs are summarized in Figure 14. To objectively compare the three optimal designs, the team utilized the decision matrix shown in Figure 15.



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

Figure 14: Zone Results

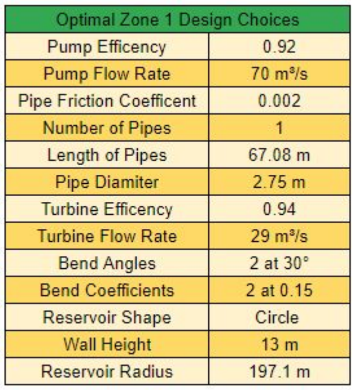


Customer Need	Technical Need	Normalizing Function	Weight	Zone 1	Zone 2	Zone 3
High Energy Efficiency	Efficiency (Eout/Ein)	efficency/100	0.35	0.281	0.29	0.291
Low Cost to Build/Operate	Total Cost of Project	1- (cost/800,000)	0.45	0.427	0.0779	0.183
Few Environmental/Cultural Concerns	Potenial to Postpone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

Figure 15: Decision Matrix

The detailed findings and a depiction for the proposed system located in Zone 1 are shown in Figures 16 and 17. Comparing the figures for efficiency to real world studies, the team was able to validate the model, as the results are close to the 87%

efficiencies of real world systems (Hawaiian Electric, 2013). In addition, the model matches the sample inputs and outputs provided to the team.



Optimal Zone 1 Design Choices	
Pump Efficency	0.92
Pump Flow Rate	70 m³/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 m³/s
Bend Angles	2 at 30°
Bend Coefficients	2 at 0.15
Reservoir Shape	Circle
Wall Height	13 m
Reservoir Radius	197.1 m

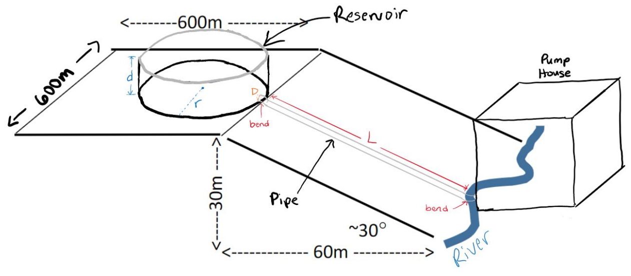


Figure 16 and 17: Optimal Zone 1 Design Choices and Final Design Diagram

Discussion of Other Factors

Leaky Pipes

One factor that was considered was the potential for leaks in pipes. The equation used to determine how much water would be lost is shown in Figure 18 (LMNO Engineering). In this equation, Q is the leak rate in m^3/s, A is the area of the crack, P is the change in pressure from inside the pipe to outside, S is the specific gravity of the liquid, and $P_{w,std}$ is the density of water under standard conditions. Reinforced concrete was evaluated, as concrete poses the most threats for leaks out of all other materials used for above ground water transport ("Cracks, Pipe Breaks, Collapse" 2004; Gur, 2019). Pipe cracks exceeding a width of 0.100 inches are repaired, meaning the worst case leak would be a width of 0.100 in (Busba, Sagues, & Mullins, 2011). 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 worn out and leaking. After plugging in all worst case values, each potential leak could result in a mass loss of 0.34% per day. Even with multiple leaks, this value is insignificant.

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

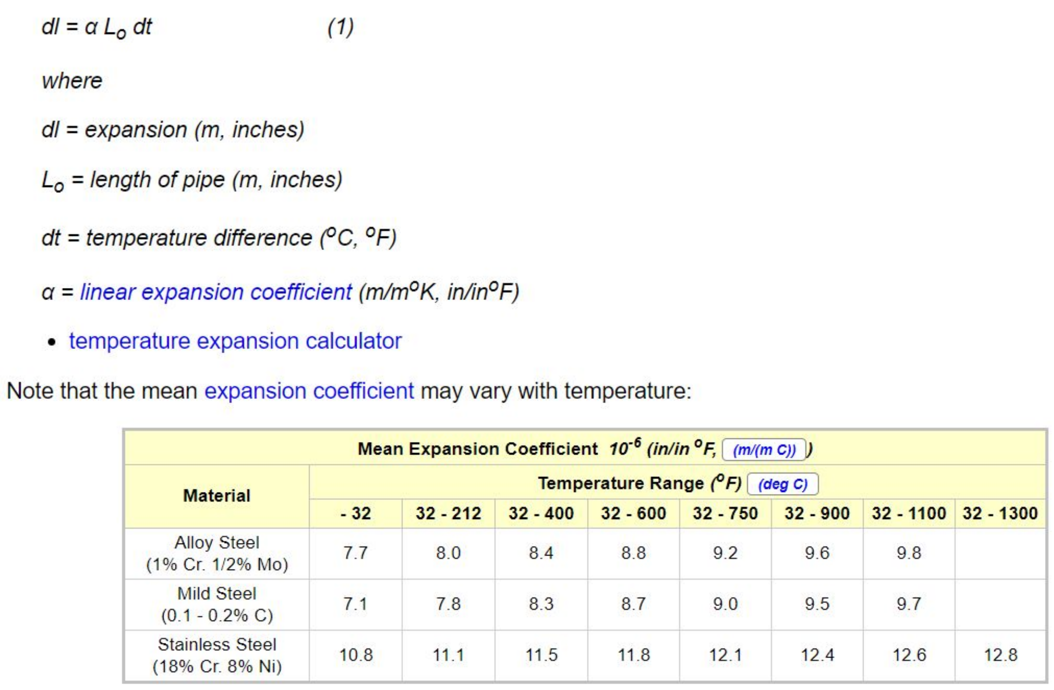
Figure 18: Pipe Leak Rate Equation

Objects Falling into Reservoir

Another factor considered was the water lost by humans, animals, or debris falling into the reservoir. When humans fall into a pool of water, some water, approximately equal to the surface area of a human, adheres to them. Taking that into account, it is possible to find the volume of water that would adhere to a human. Using the surface area of a human, which is approximately $1.9m^2$, and the volume of a drop of water ($5.0*10^{-8}$ m^3), the amount of water that adheres to a human is approximately 0.0069997 m^3 (Shiel Jr., 2018). This volume of water is approximately 0.007% of the reservoir, which is negligible. Animals would cause the reservoir to lose about the same amount of water, as they are mostly smaller than humans, and thus are also considered negligible. Moreover, debris falling into the reservoir would not be an issue unless it started clogging the pipe system.

Pipe Length Expansion/Contraction due to Temperature

Another factor worth considering was the possibility of the pipe length changing due to changes in temperature of the surrounding environment. A change in pipe length affects pipe friction which could potentially impact the energy loss. The following equation can be used to model the relationship between temperature and pipe expansion:



$$dl = \alpha L_o dt \qquad (1)$$

where

dl = expansion (m, inches)

L_o = length of pipe (m, inches)

dt = temperature difference (^{o}C, ^{o}F)

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

Figure 19: Pipe Length Expansion Equation

As seen above, the expansion coefficients vary with material and temperature. For temperature, the team assumed that the range was between 32°F (freezing temperature) and 130°F (≅highest recorded temperature) (“Pipes and Tubes”, 2020).

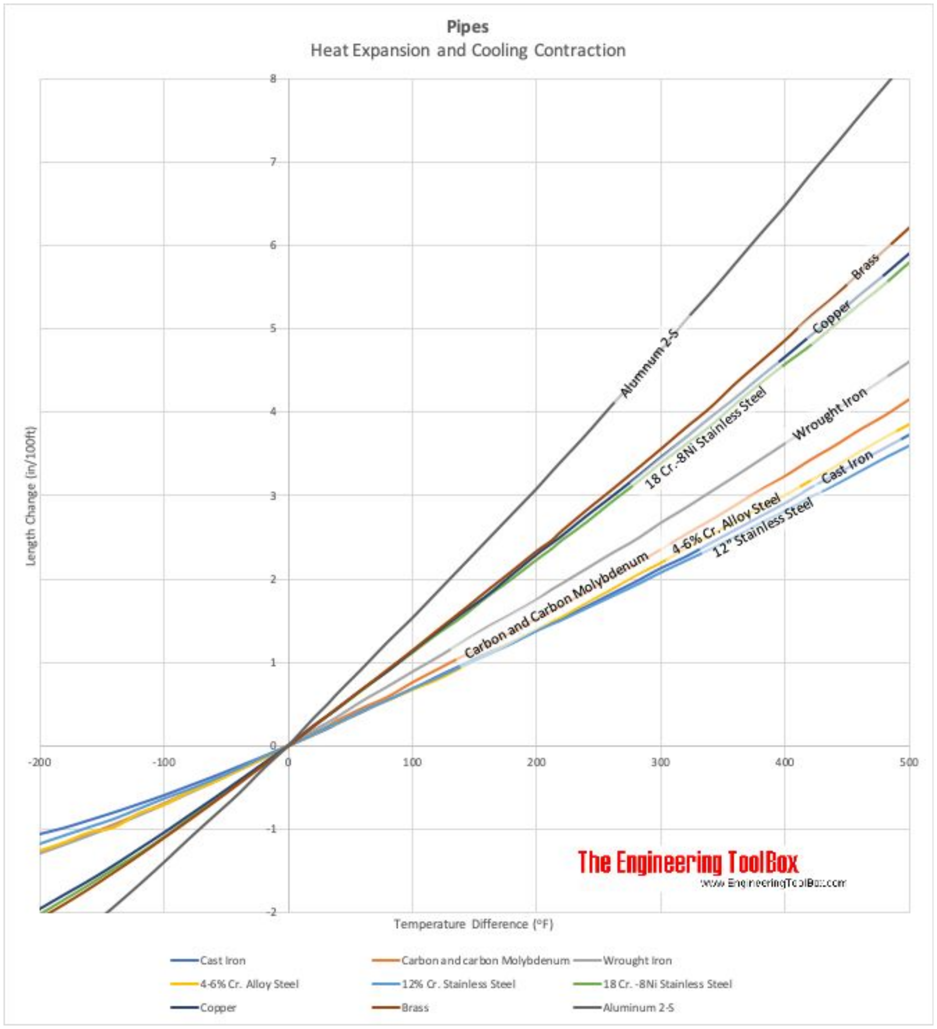


Figure 20: Expansion as a Function of Temperature

The figure above indicates that regardless of material, within a 0-130°F range, the maximum change in length is 2in per 100ft ("Pipes", 2020). Since Stainless Steel 18% has a trend line wedged between the rest and would thus provide a ballpark figure and an average estimate, the team computed the values for this material and found that this factor decreases the efficiency by 0.0006%. The team found this to be negligible.

Social and Cultural Concerns

Social and cultural concerns are an important part of any construction project. The social factors at play in this case study are unique to each zone. Zone 1 contains hazardous chemicals, which could pose an issue to the health and safety of the humans building the system. Zone 2 contained possible ancestral burial grounds, which could create tension with local people and lead to legal concerns. Zone 3 poses a challenge because of soil erosion. Trees would have to be removed, which could result in lawsuits from environmentalists. Additionally, continued erosion could create an unsafe working environment. These factors were evaluated in the decision matrix in Figure 15, accounting for 20% of the final decision. Each zone was assigned a value of 1 or 0 based on if it had the potential to delay the project indefinitely. It was decided Zone 1 did not have this potential, as hazardous chemicals could be removed. Zone 2 could shut the project down, as burial grounds are often sacred and moving them may not be an option. Zone 3 did not have a potential to delay the project indefinitely as new vegetation could be planted so there is no net loss of forested area and long term erosion monitoring for this area is feasible.

Conclusions and Recommendations

For all the possible systems checked, it was assumed that the pipes were located within 2m of the ground and ran parallel to it at all times. Using a different approach, one can possibly improve the efficiency of the system by having pipes run directly from the pumphouse to the reservoir. However, doing so raises the installation costs of the system. To keep the analysis simple and equal across all zones, the team decided to have all pipes in the system run along the ground.

There remains the possibility that a different system could yield better results. The team recommends experimenting with the raised installation of pipes. Doing so increases the cost, but there is a chance this increases efficiency as it cuts down on bends and the overall length of the pipe.

In the proposed design, the wall height is said to be 13 meters. In order to simplify the model, it was assumed that the reservoir's maximum fill height was at the same level as its wall. It was also assumed that the reservoir allows for multiple pipe connections. Lastly, it was assumed that the Earth's ground was a sufficient base for the reservoir and that a separate floor does not need to be built.

By addressing the various limitations in our model, the model can be improved for greater accuracy. By using an equation to account for water being absorbed into the ground, more accurate results can be provided. Secondly, equations that account for long term effects, such as maintenance costs or wear and tear, can also help make the model more accurate. Lastly, allowing the user to input a maximum threshold value for the reservoir filling time will allow the analysis algorithm to output more system combinations, some which may have greater efficiencies.

After inter-zone and intra-zone comparisons of various outputs, the team decided that the most optimal outputs ranged anywhere from 83.1% efficiency for \$653,592.82 to 80.28% for \$457,747.07. The change in price was more prominent than the difference in efficiency and thus it was decided that the optimized system would have 80.28% efficiency.

References

Busba, E., Sagues, A., &amp; Mullins, G. (2011). REINFORCED CONCRETE PIPE CRACKS – ACCEPTANCE CRITERIA (Tech.). University of South Florida College of Engineering. Retrieved December 05, 2020, from https://www.researchgate.net/publication/311355262_Reinforced_Concrete_Pipe_Cracks-_Acceptance_Criteria_see_link

Cracks, Pipe Breaks, Collapse. (2004). Retrieved December 05, 2020, from https://www.unitracc.com/technical/books/rehabilitation-and-maintenance-of-drains-and-sewers/damage-its-causes-and-its-consequences/cracks-pipe-breaks-collapse-en/inhalt

Gur, E. (2019, May 16). Water Distribution Pipes. Retrieved December 05, 2020, from https://sswm.info/sswm-university-course/module-2-centralised-and-decentralised-systems-water-and-sanitation-1/water-distribution-pipes

Hawaiian Electric: Energy Storage. (2013). Retrieved 6 December 2020, from https://web.archive.org/web/20151118171429/http://www.heco.com/portal/site/heco/menuitem.508576f78baa14340b4c0610c510b1ca/?vgnextoid=94600420af0db110VgnVCM1000005c011bacRCRD&vgnextchannel=ab020420af0db110VgnVCM1000005c011bacRCRD&vgnextfmt=default&vgnextrefresh=1&level=0&ct=article

LMNO Engineering. (n.d.). Leak Rate for Liquid in Pipe or Tank. Retrieved December 05, 2020, from https://www.lmnoeng.com/Flow/LeakRate.php

Pipes and Tubes - Temperature Expansion. (2020). Retrieved 6 December 2020, from https://www.engineeringtoolbox.com/thermal-expansion-pipes-d_283.html

Pipes - Heat Expansion and Cooling Contraction. (2020), from https://www.engineeringtoolbox.com/thermal-expansion-steam-pipes-d_358.html

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.

William C. Shiel Jr., M. (2018, December 21). Definition of Body surface area. Retrieved December 05, 2020, from https://www.medicinenet.com/body_surface_area/definition.htm