

Optimization of Hydroelectric Pumped Storage: An Extension of Optimal Switching

Anthony Kuehne
tkuehne@umich.edu

July 1, 2007

1 Introduction

Commodities unlike stocks, bonds, and other paper assets require storage. Yet, commodities like electricity cannot be stored in the same sense as other commodities such as oil or natural gas. The storage of electricity requires storing the means to produce electricity with the ability to quickly generate electricity from the stored fuels such as coal. Hydroelectric pumped storage presents a new way to store electricity (by storing electricity we mean as above in storing the means to generate electricity). Hydroelectric pumped storage facilities operate differently than commonly known hydroelectric damming plants. A pumped storage plant consists of two reservoirs on different elevations and a hydroelectric dam. Water is moved from the top reservoir through the dam to the lower reservoir turning turbines that generate electricity, which is then sold to the power grid in a similar fashion to hydroelectric damming.

The difference lies in that the pumped storage facility's pumps which push water through the dam can be reversed to move water from the lower reservoir back up to refill the upper reservoir. Since electricity prices are random and fluctuate throughout the day and also seasonally, pumped storage facilities allow electric companies to take advantage of these fluctuations. Following the standard model of 'buy low and sell high' the pumped storage facility can be filled with water from the lower reservoir, which may be a large lake or other body of water, during times of low demand for electricity such as late at night or weekends. Then during times of high demand for electricity the pumped storage facility can open the dam and generate electricity, which is then sold. The profit is realized in the fact that the price of electricity, which is again random, is lower when the demand is low and higher when demand is high following the standard rules of supply and demand. This

means that it costs less to fill the upper reservoir than the revenue made from selling the electricity generated from the dam and the water in the upper reservoir.

After examining the mechanics of pumped storage, it is natural to ask how one could maximize the profit of this process over a certain period. This paper sets out to create a stochastic model for the optimization of hydroelectric pumped storage. In this paper, we take the place of the owner or manager of a pumped storage facility whose goal is to maximize profits over a certain period of time. It seems that the most intuitive argument to maximize the value of the facility would be to refill the upper reservoir when the price of electricity is low such as nights and weekends and switch to generating energy when the price of electricity is high such as evenings. This then becomes a problem of optimal switching, and finding the optimal time to switch from refilling the reservoir to generating electricity. Since the plant also stores the potential to switch and generate electricity quickly, this breaks down to a problem of energy storage and optimal switching. Ludkovski and Carmona (2007) developed a model for gas storage with respect to optimal switching which can be used as a basis for the model this paper presents. Hydroelectric pumped storage can then be viewed as an extension of this model in which additions and changes must be made to move from natural gas to pumped storage electricity generation.

2 Our Model

We extended the model Ludkovski and Carmona (2007) with the addition of a new variable (precipitation), the addition of a more complex pricing process (electricity price jumps), and change of parameters to change from a model for natural gas to that of pumped storage. The variables that appear in the value equation and algorithm developed for the facility are:

1) Inventory, c_t , is the current amount of water that is held in the upper reservoir. This given amount depends on previous actions and inventory, which is known. For the sake of simplicity, we assume that the lower reservoir has an infinite amount of water and for the pumped storage facility that is modeled this simplification has no bearing on results.

2) Price of electricity, g_t , is a random variable that can be estimated using a suitable model. We used the model of Carlea et al (2005). This is "a mean-reverting jump diffusion model for the electricity spot price". The model states that the spot price of electricity, g_t , can be represented by:

$$g_t = e^{Y_t} \tag{1}$$

and Y_t is a stochastic process defined by:

$$dY_t = -\alpha \cdot (\theta - Y_t) \cdot dt + \sigma(t) \cdot dZ_t + \ln J \cdot dq_t \quad (2)$$

Where Z_t is a standard Brownian motion and q_t is a Poisson process with intensity J jumps per month which is independent of Z_t , α is the speed of mean revision, θ is the log of the mean price level, and σ is the volatility (Cartea et al 2005). We encourage the reader to refer to this paper for a more complete and in-depth explanation of this pricing model. For the example in this paper, the prices of electricity are on a log scale and are in dollars per megawatt-hour (MWh).

3) Precipitation, l_t , is a random variable which was added to account for the impact of rain, snow, and runoff on the current inventory c_t of the upper reservoir. These factors increase current inventory and therefore increase the amount of electricity that can be generated and sold. Cao et al (2004) present a good model for precipitation that is thorough yet not overly complex for the needs of the model. The model makes l_t a Bernoulli random variable that measures the probability that it starts or stops raining (transitional probabilities) conditional on the current weather state. The probabilities are seasonal to account for the changes in precipitation that occur seasonally such as wet springs and dry summers. The effect or rain, R , is a rate in billions of gallons per month, similar to the flow rates explained below, which accounts for the additional inventory that precipitation adds to the upper reservoir, c_t , (Cao et al 2004). We again encourage the reader to refer to this paper to gain a stronger understanding of the transitional rain properties.

4) Time, t , which is known and ranges from 0 to T

5) Action, i_t , This is a known parameter which takes 3 values either -1,0,1, referring to pumping in water, storing water, or generating electricity respectively. The choice of action has a direct effect on both c_t and g_t which will be explained more thoroughly below.

Other variables that appear in or impact the equation are:

6) Flow rates, f , are the rates at which water flows in and out of the reservoir computed over a certain time scale such as daily or monthly. These are given amounts determined from information on pumped storage such as Ludington (2007). For the examples below the flow rates are in billions of gallons per month.

7) α rates, α_i , and $\alpha_{rain,i}$, are the rates at which water is either pumped, depleted, or stored in the upper reservoir, over a certain change in time Δt . The i value corresponds to the same numbered actions, i_t , as above. The α_i rates that are determined from from f accounting for change in time, Δt , and change in inventory, Δc :

$$\alpha_{-1} = \frac{f \cdot \Delta t}{\Delta c} \quad (3)$$

$$\alpha_0 = 0 \quad (4)$$

$$\alpha_1 = -\frac{f \cdot \Delta t}{\Delta c} \quad (5)$$

The $\alpha_{rain,i}$ rates are used to take into account the effects of rain:

$$\alpha_{rain,-1} = \alpha_{-1} + \frac{R \cdot \Delta t}{\Delta c} \quad (6)$$

$$\alpha_{rain,0} = \alpha_0 + \frac{R \cdot \Delta t}{\Delta c} \quad (7)$$

$$\alpha_{rain,1} = \alpha_{-1} + \frac{R \cdot \Delta t}{\Delta c} \quad (8)$$

Using the above parameters we can create a generic value function for the pumped storage facility, $V(g, c, l, t, i)$, which is the expected value of the facility from t until T assuming optimal behavior.

Another important aspect which needs to be incorporated to make the model more complete is jumps in the price of electricity. The price of electricity is often apt to price jumps which play an important role in optimization. The method for the addition of price jumps is discussed below.

Two other factors play important roles in the model for pumped storage. These are the inefficiency of pumping water into the upper reservoir and switching costs. The inefficiency of pumping water into the upper reservoir is important because pumping water into the upper reservoir uses electricity that the plant must pay for while the plant is trying to generate electricity at the same time. This creates inefficiency because electricity must be used in order to generate electricity, which reduces the efficiency of the facility. The switching costs are important because when the facility switches actions its costs money to change from one regimen to another. Both of these factors play an important role in the optimization and maximization of pumped storage and play an important role in deriving the equation below.

3 Our Method

Our goal is to maximize the value function $V(g, c, l, t, i)$ subject to the given constraints. In order to maximize the value at time T (the end of the scenario) we must act optimally at time $t = 0$. The action made at $t = 0$ effects the rest of the actions due to the fact that the choice of action changes one or more variables such as g_t and c_t in the value equation. Therefore it is necessary to find the optimal action at $t = 0$ in order to optimize the total value. From this we get the below example.

$$E[V(g, c, l, t, i)] = \max(E[V(g_0, c_0, l_0, 0, -1)], E[V(g_0, c_0, l_0, 0, 0)], E[V(g_0, c_0, l_0, 0, 1)]) \quad (9)$$

If the dam were to be opened today the following would occur:

Open dam: ψ_{out}

Receive: $\alpha_1 \cdot g$

New Inventory: $c - \alpha_1$

New Price: $N(g + \mu \cdot \Delta t + \sigma^2 \cdot \Delta t)$

Where N is a normal random variable, μ is the mean price of electricity, and σ is the volatility of electricity prices. The expected value is then taken over remaining time $t + \Delta t$ to T :

$$E[V(g_{t+1}, c - \alpha_1, l, t + \Delta t, 1)] \quad (10)$$

The goal is to maximize the value $V(g, c, l, t, i)$ and to solve this maximization problem it is more efficient to solve backwards from T to $t = 0$. We also assume that we are only interested in the time up to and including T . From this we get the following equation.

$$V(g, c, l, T, i) = 0 \quad (11)$$

Solving back in time and applying Ito's equation to the stochastic differential equations for the state variables we arrive at a Hamilton-Jacobi-Bellman (HJB) equation:

$$V_g + 1/2 \cdot V_{gg} - \alpha_i \cdot V_c + \psi_{out} - r \cdot V = 0 \quad (12)$$

where V_n represents the partial derivative of $V(g, c, l, t, i)$ with respect to the parameter n and r is the risk free rate determined using a suitable measure.

The factor to account for price jumps of electricity must also be added to the model. The following term must be added to the HJB equation to account for price jumps.

$$J \cdot dt \cdot (V((1 + j)g, c, l, t, i) - V(g, c, l, t, i)) \quad (13)$$

Where J is the probability of a price jump, $1 + j$ is the magnitude of a price jump, with the rest of the parameters are as described above. Adding this factor to the HJB equation accounts for jumps in the price of electricity and creates a more realistic model.

In order to solve this partial differential equation, partial derivatives with respect to the correct parameter must be computed. To do this manually would be nearly impossible and therefore an alternative method must be implemented. In the case of this model the method of finite difference methods was used to compute the partial derivatives at every Δt . The method of finite difference methods involves computing the change in values with respect to a small change of a certain variable over a small time step, Δt . If the time step and variable change are small enough the computed change in value becomes close enough to approximate the corresponding partial derivative, which can then be used in the algorithm. An example is the partial derivative V_g which is computed by:

$$V_g = \frac{(V(g + \Delta g, c, l, t, i) - V(g - \Delta g, c, l, t, i)) \cdot \Delta t}{2\Delta g} \quad (14)$$

The process for computing the remaining partial derivatives is similar. Finite difference methods present a good estimation of partial derivatives but also present problems such as the massive amount of calculations that must be computed in order to give quality results. The need for such a large number of calculations causes the computation of these values to be slow and extremely long. Other methods of estimation exist but are out of the scope of this paper (Wilmott et al 1995).

4 Numerical Results

This section provides numerical examples of the algorithm and examines the impact of different variables on the optimal action and value of the facility. Parameter values in this section were obtained from a brochure on the Ludington pumped storage plant in northern Michigan. The facility is situated on Lake Michigan, which makes it ideal for our model due to the almost infinite amount of water in the lower reservoir. As a note, all electricity prices are on a log scale and inventory levels are in billions of gallons of water. (Ludington 2005).

Solving the partial differential equation, (12), requires using the above method of finite difference methods. In order to do this a grid must be implemented for Δc and Δg to compute the partial derivatives. In the code, which is attached in the appendix, a 200 by 200 grid was created with c ranging from 1.35 to 27 (billion gallons) and with g ranging from .3120 to 7.9120 (log price in dollars per MWh). This gave $\Delta c = .135$ and $\Delta g = .004$. The code and program were then used to evaluate the partial derivatives and solve the partial differential equation.

4.1 Example 1

This section provides a numerical example of the model, which was explained above and will be used as a comparison for other examples. The capacity of the pumped storage facility is 27 (billion gallons), and $T = 3$ months. The flow rates, f , are 1473.12 (billion gallons per month) and the impact of rain, R , is 500 (billion gallons per month). We use the electricity price model explained in (1) and (2), with θ equal to $\log(50)$ dollars and σ equal to .3839379. A time step, Δt , of 1 minute or .000022 of a month was used. The risk free rate, r , was set to 6 percent. Other variable parameters include:

$$\alpha_{-1} = .24444$$

$$\alpha_1 = -.24444$$

$$\alpha_{rain,-1} = .327413$$

$$\alpha_{rain,1} = -.161476$$

The seasonal rain probabilities used were:

Days	Probability Precipitation Starting	Probability of Precipitation Stopping
1-25	.7125	.4325
25-50	.7225	.4025
50-75	.7475	.385
75-93	.7775	.4025

For the sake of unanimity in comparison, we choose 'open' or generating electricity as our final regimen. The status of 'open' plays little impact on the results and thus does not play a role in the final value or optimal action. This is due to the fact that we are only interested in up to and including time T . For the first example the current state of precipitation is set to 'rain', the impact of this choice will be examined in Example 2.

Figure 1 represents the final value of the facility at $T = 3$, precipitation set to 'rain', and the action set to 'open' or generate electricity. The results show that the higher initial inventory and the higher the initial price of electricity the higher the value of the facility.

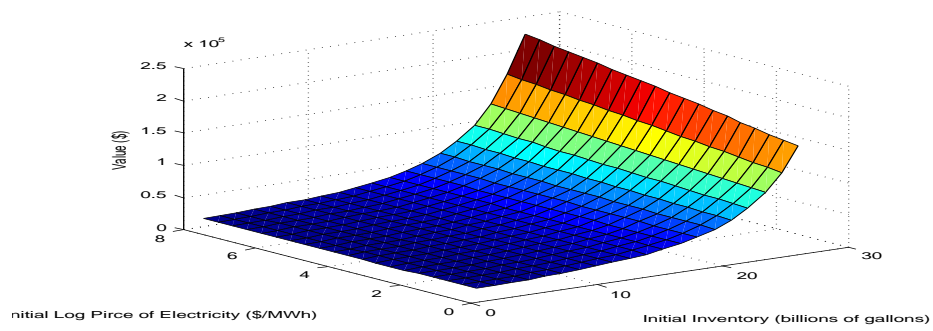


Figure 1: Value function surface showing $V(g, c, rain, 3, open : T = 3)$

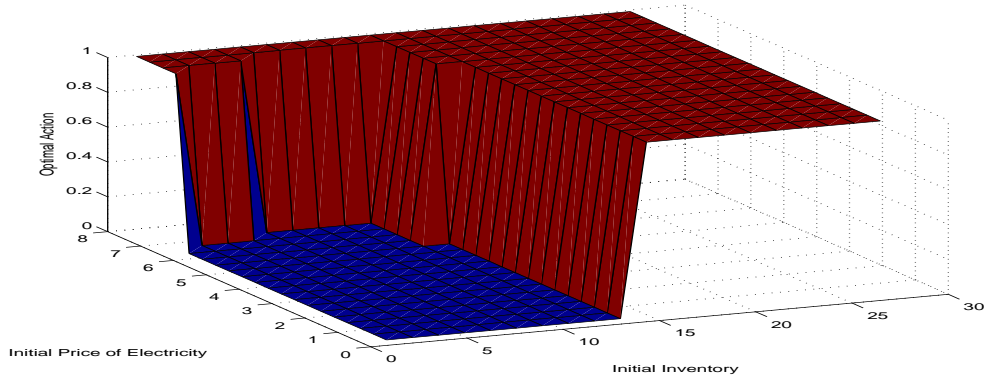


Figure 2: Optimal action at time $t = 0$ for $V(g, c, rain, 3, open : T = 3)$

Another piece to examine is that at low levels of initial inventory the value of the facility is low regardless of the initial price of electricity. This can be explained by the fact that at low levels of inventory water must be pumped into the upper reservoir in order for the facility to have water to generate electricity. Operating the pumps uses electricity that the plant must pay for at the same price that generated energy could be sold for. Once power is generated, the plant makes a profit off the random changes in the price of electricity. Unlike large initial values, which have the immediate potential to generate large amounts of electricity, lower initial inventory levels must rely on changes in price to make profits.

After the value of the facility is analyzed the next logical piece to examine is the optimal beginning action at time $t = 0$. The graph below depicts the optimal actions, the numbers -1,0,1 on the graph refer to the same numbered action as above.

From figure 2 it appears that the optimal action at $t= 0$ takes only two of the three possible values. When either the initial inventory level or price of electricity is high, it is optimal to generate electricity, and when the initial inventory and price of electricity are low, the

optimal action is to store. This may seem strange, with both low initial inventory and low initial price of electricity it seems the optimal action should be to refill the upper reservoir and then when the price rises generate electricity. There are many possible reasons that storing water, as opposed to pumping in water may be optimal. One reason is that the model presented includes a penalty or 20 percent for injecting water into the upper reservoir and switching costs. These are included to account for the costs incurred by changing the plant from one regimen to the other and the inefficiency of pumping water into the upper reservoir. Another factor may be the large effect of rain on inventory. In our example the effect of rain, R , is high to have a noticeable effect on the value. Together these effects may cause the facility to have a larger value by waiting for rain than to incur the costs to increase inventory.

Another aspect to examine is the effect of rain on the value of the facility. The value with rain should be larger due to the fact that rain increases inventory without the costs of pumping water into the upper reservoir. In order to view the effects of rain it suits to examine the difference when the initial l_t is 'rain and 'no rain'.

Figure 3 displays that precipitation has a larger effect on the value of the facility when the initial capacity is high. The figure also points out that the increases in value from rain do depend on the initial price of electricity. This may be attributed to the fact that both figures ($V(g, c, rain, 3, open : T = 3)$ and $V(g, c, norain, 3, open : T = 3)$) have shapes similar to figure 1, which are more dependent on initial inventory than on the initial price of electricity. Another factor on the lack of dependence on initial price of electricity is the randomness of the price of electricity. The model above uses a mean reverting model for electricity. This means electricity prices are random but stay close to and fluctuate around the mean. The randomness of electricity prices may causes the dependence on initial price of electricity to be small as opposed to the dependence on initial inventory, which is directly increased by rain.

4.2 Example 2

In this example, the effect of the inclusion of jumps in the price of electricity on our model is examined. The addition of price jumps is removed from the model to examine the effect of their initial inclusion. In the model above $J = .1$ where J is the probability of a price jump and $1 + j \approx 1.16$ is the amount the price of electricity will increase due to a price jump and using the same parameters as Example 1 the model is implemented with the omission of (13). The values for the removal of price jumps are then compared to the values when the full model (Example 1) is used.

Figure 4 shows that the inclusion of price jumps has a positive effect on the value of the

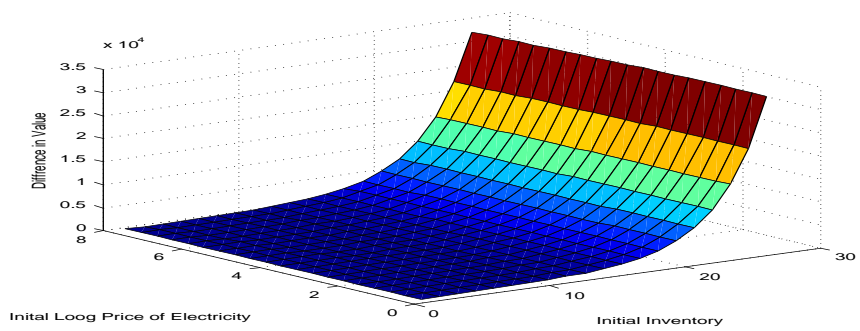


Figure 3: The difference in value between for $V(g, c, rain, 3, open : T = 3)$ and $V(g, c, norain, 3, open : T = 3)$

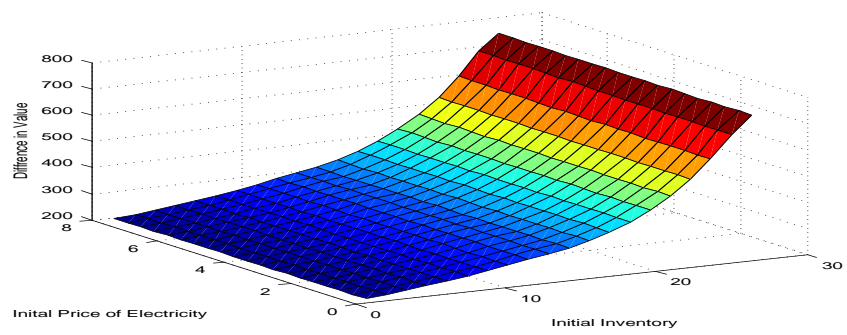


Figure 4: The difference in value between for $V(g, c, rain, 3, open : T = 3)$ when the algorithm includes and does not include price jumps (eq. (13))

facility. Similar to figure 3, figure 4 shows that the effect of price jumps have a stronger dependence on initial inventory than on the initial price of electricity. This is reasonable based on the fact that price jumps increase the price of electricity which makes the water in the upper reservoir and the facility more valuable. The initial price of electricity has less dependence due to the fact that the model for price jumps, (13), only allows the price of electricity to increase. While price jumps make the value of the initial inventory higher they also causes the price of pumping water to increase and the two effects are negated. The figure shows that the price jumps have a positive effect on the value of the facility even though jumps in the price of electricity also have a negative effect on the facility because of the need to pump water into the upper reservoir. The overall advantages of the addition of price jumps outweigh the negative effects experienced from refilling the upper reservoir.

4.3 Example 3

In this example we examine the effect of the flow rates, f , on the value of the facility. In the previous examples the flow rates were equal for both pumping up and down. The flow rate for pumping down was determined from the brochure on the Ludington Pumped Storage Plant and the same flow rate was then used for pumping up, under the assumption that the pumps were reversible. In reality this is not a realistic assumption due to the fact that pumping water into the higher reservoir requires pumping up a large elevation, 35 stories in the case of the Ludington facility (Ludington 2005). In order to observe the effects of decreasing flow rates the f rate for pumping water into the reservoir was decreased by $1/2$. Therefore it takes twice as long to fill the upper reservoir as it does to deplete it.

Examining figure 5, the only time which reducing the rate of pumping water into the upper reservoir affects the value is when the initial price and inventory are low. This implies that the rate at which water is pumped into the upper reservoir does not have a large effect on the value of the facility unless the initial inventory and price are low. After looking at the best response at time 0 there was no difference between the original flow rates and the new flow rates. This may be attributed to the large value for rain, R , which may overshadow the effects of the flow rates. The large value for R which give the facility more inventory with out the need to pump, coupled with pumping costs (switching costs and the penalty) may cause pumping water into the reservoir to not be optimal. The impact of the amount or rain, R , will be discussed below.

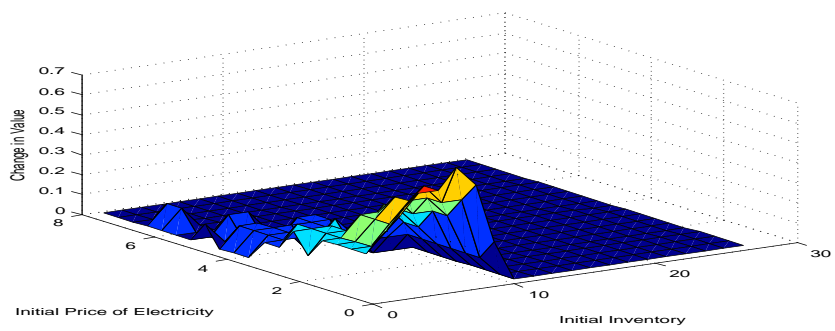


Figure 5: The difference in value between $V(g,c,rain,3,open: T=3)$ when the flow rates are equal and when the flow rate of pumping water up is reduced by $1/2$

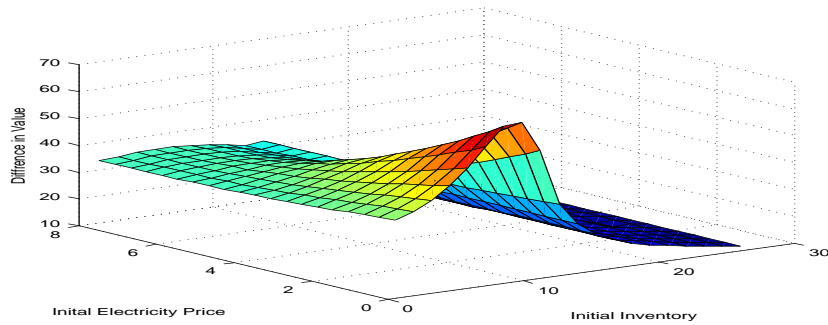


Figure 6: The difference in value between for $V(g, c, rain, 3, open : T = 3)$ when the penalty for pumping water into the upper reservoir is removed

4.4 Example 4

The effects on changing the penalty for pumping water into the reservoir are now examined. In the model presented and in Example 1 a penalty of 20 percent is charged when water is pumped into the upper reservoir to account for the inefficiency of pumping. In this example the penalty is removed so that there is no excess cost for pumping water into the upper reservoir besides switching costs and electricity. The data is then compared to the original data of Example 1 to create a comparison.

Removing the penalty for pumping has the largest effect when the initial inventory is low and spikes when both the initial inventory and initial price are both low. The changes can be attributed to the fact that at low initial inventories pumping water up is necessary in order to be able to generate and sell electricity later on. The effect of the relaxed penalty is largest when the initial inventory is low and decreases as the initial inventory increases. The best initial response seems to agree with the earlier best response graph presented in

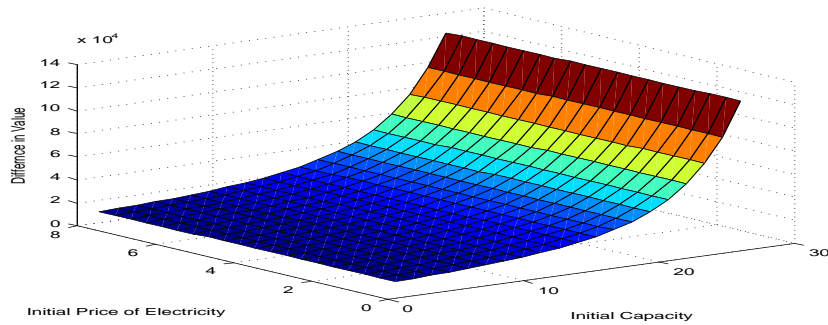


Figure 7: Difference in value for $V(g, c, rain, 3, open : T = 3 : R = 500)$ -
 $V(g, c, rain, 3, open : T = 3 : R = 100)$

Example 1, but the area where store or 0 is optimal was reduced around the edges. The lack of change in the optimal initial response will be addressed later.

4.5 Example 5

In this final example we explore the effect that the amount of rain, R , had on the value and optimal action of the facility. The initial R for the above examples was 500. This is a high number chosen so rain would have a noticeable effect on the value functions. In the examples above the amount of rain, R , was approximately 34 percent the size of the flow rates. In this example R is decreased to 100 which is about 6 percent of the flow rates to examine if the larger R was overly effecting the optimal actions and value .

As we can see from figure 7 changing the impact of rain effected the value but in a similar way as the other factors. The value change is more dependent on the initial inventory than on the initial price of electricity. The optimal action graph for the decreased rain

was identical to the optimal action graph for Example 1. Therefore, the effect of rain is larger when there is more inventory and again does not have a large dependence on the initial price of electricity. This may be because rain affects the inventory throughout the model and along with the model for electricity prices causes initial inventory have a large dependence. The reasons for this interesting dependence on inventory levels for all of the examples will be discussed in the conclusion.

5 Conclusion

The model presented in this paper provides a good extension to the model of Ludkovski and Carmona (2007). With the addition of a variable for the effect of precipitation and the addition of price jumps, a more robust and realistic model was created. The examples included above provided good numerical examples to the problem of the optimization of hydroelectric pumped storage the authors set out to solve. The examples determined the dependence of the model on particular variables that were changed or imposed. The numerical examples also allowed us to examine how the factors that were added and changed in the model affected the new algorithm. The two in particular are the addition of precipitation and the addition of electricity price jumps and how these how these resulted in an increased value for the new model.

We found the addition of rain and price jumps increased the value of the facility and both had a strong dependence on the initial inventory with a lesser dependence on the initial price of electricity. Reducing the impact of rain decreased the value of the facility with a strong dependence on the initial inventory. Decreasing the flow rate for pumping water up had an interesting effect on the value. The value decreased only at places with low initial inventory and low initial price of electricity while everywhere else stayed the same. Removing the penalty for pumping water into the reservoir increased the value of the facility. Changing the penalty for pumping had a large dependence on the initial inventory but unlike the other variables it also had a dependence on the initial price of electricity especially when both initial price and inventory were low. Overall, most variables displayed a strong dependence on the initial inventory and a weaker dependence on the initial price of electricity. Another interesting fact to note is that changing variables had little or no effect on the initial optimal action. This may be because regardless of the variables the optimal initial action is the same and only depends on the initial inventory and initial price of electricity. While the initial optimal action stayed the same as variables changed there was a large change in the value of the facility accompanied with these variable changes.

While the model developed in this paper is good there are a few extensions that could be added to create an even more realistic model. The first addition could be seasonality and time dependence on the price of electricity. The price of electricity is higher in the evening than it is late at night. Adding this factor to the model would help create a more realistic model since water could be pumped up during the night and then pumped out during the day increasing the margin of profit. Another addition could be a different approach to solving the partial differential equation. While the finite difference methods gave good results getting the results was slow. Another method could be implemented to increase the speed of the algorithm and therefore allow for more changes to be made and parameters to be varied. Finally the assumption of an infinite bottom reservoir could be relaxed to create a model that can work on pump storage facilities situated on smaller bodies of water.

This paper presented a complex yet computationally simple model that extended the work of Ludkovski and Carmona (2005). The solutions found gave insight into the optimization of hydroelectric pump storage with respect to optimal switching. The paper developed an extended model that was able to complete the task of creating a more realistic model under the framework of hydroelectric pumped storage.

6 Acknowledgments

This paper and the research covered by the paper were done as part of the Research Experience for Undergraduates (REU) program through the National Science Foundation and The University of Michigan. I would especially like to thank Professor Mike Ludkovski for the countless hours and effort he put in while guiding me through the entire process of the research and paper. His support and help allowed me to learn and understand research methods and present this paper.

References

- [1] Cao, Melanie, Anlong Li, and Jason Wei. "Precipitation Model and Contract Valuation: a Frontier in Weather Derivatives." *Journal of Alternative Investment* 7.2 (2004): 93-99.
http://www.yorku.ca/mcao/cao_li_wei_precipitation.pdf
- [2] Cartea, Alvaro, and Marcelo Figueroa. "Pricing in Electricity Markets: a Mean Reverting Jump Diffusion Model with Seasonality." *Taylor and Francis Journals* 12 (2005): 313-335.
<http://129.3.20.41/eps/fin/papers/0501/0501011.pdf>

- [3] *Ludington Pumped Storage Plant*. DTE and Consumers Energy, 2005.
<http://www.consumersenergy.com/apps/pdf/LudingtonPumpedStorage.pdf>
- [4] Ludkovski, Mike, and Carmona Rene. "As Storage and Supply Guarantees: an Optimal Switching Approach." Submitted (2007).
http://www-personal.umich.edu/~mludkov/CL_Storage.pdf
- [5] Wilmott, Paul, Sam Howison, and Jeff Dewaynne. *The Mathematics of Financial Derivative*. New York: Cambridge UP, 1995.

7 Appendix

7.1 Code

The following is the code is the code for the first example.

```
//Variable Metric
#include <math.h>
#include <iostream>
#include <stdlib.h>
#include <fstream>

using namespace std;

#define max(a,b) (((a)>(b))?(a):(b))
#define min(a,b) (((a)<(b))?(a):(b))

//#define M_PI 4*atan(1.0)

int main(int argc, char *argv[])
{
    ofstream outDataW1, outDataV1, outDataBRW1, outDataBRV1, outDataValues;

    double x0, c0,r;
    double thetaX, kx, gx, alpha, alphaRain; // sde parameters
    int steps = 133920;
    int days = 93;
    int increment = 36000; // increment for rain probabilities
    int scaleFactor = 1.15556;
```

```

int moneyFactor = 1;

double dt, dx, dc;          // Step size of time/space
int i, j, j1, jj1, j2, k, kk, nStates, N1, N2;

double T;                   // The horizon in years
double R;

double ***V1, ***V2, ***W1, ***W2, ***bestActionV1, ***bestActionW1; //function values
double dV_dx, d2V_dx2, curx, curc, dV_dxRain, d2V_dx2Rain; //pde derivatives
double driftx ,driftxx;
int inorder[550];

nStates = 3;
c0 = (double)13.5; // 1/2 storage capacity
x0 = (double)log(50); // Log price of electricity
r = (double)0.06000;
kx = 1.425; thetaX = x0;
T = 3;
R= 500; // impact of rain
gx = .3839379;
double scaleP[] = {-1.2, 1, 0};
double curCost, curMax, curMaxRain;
double CAP[] = {1473.12, 1473.12, 0}; // flow rates
double P;
double Q;
double J = .1; //probability or price jump

dt = (double)T/steps;          /* controls how fine is the mesh */
dc = (double)0.135;
dx = (double)0.04;
N2 = 100; N1 = 100;
double COST[] = {0, 0.25, 0.25, 0.25, 0, 0.25, 0.25, 0.25, 0}; // Switching costs

inorder[0] = 1; inorder[1] = 0;
for (i=2; i <= 549; i++)
inorder[i] = i;

double inRate = CAP[0]*dt/dc;
double outRate = CAP[1]*dt/dc;
double storeRate = 0; // storage cost

```

```

double BIG = 100;
double inRateRain = CAP[0]*dt/dc + R*dt/dc;
double outRateRain = CAP[1]*dt/dc - R*dt/dc;
double storeRateRain = R*dt/dc;

outDataW1.open("LudingtonDataW1Open");
outDataV1.open("LudingtonDataV1Open");
outDataBRW1.open("LudingtonDataBRW1");
outDataBRV1.open("LudingtonDataBRV1");
outDataValues.open("CurValues");

// Allocate memory for value functions
V1 = new double**[2*N1+1];
V2 = new double**[2*N1+1];
for (j1=0; j1<=2*N1; j1++)
{
V1[j1] = new double*[2*N2+1];
V2[j1] = new double*[2*N2+1];
for (j2=0; j2 <=2*N2; j2++)
{
V1[j1][j2] = new double[nStates];
V2[j1][j2] = new double[nStates];
}
}

W1 = new double**[2*N1+1];
W2 = new double**[2*N1+1];
for (j1=0; j1<=2*N1; j1++)
{
W1[j1] = new double*[2*N2+1];
W2[j1] = new double*[2*N2+1];
for (j2=0; j2 <=2*N2; j2++)
{
W1[j1][j2] = new double[nStates];
W2[j1][j2] = new double[nStates];
}
}

```

```

bestActionV1 = new double**[2*N1+1];
bestActionW1 = new double**[2*N1+1];
for (j1=0; j1<=2*N1; j1++)
{
bestActionV1[j1] = new double*[2*N2+1];
bestActionW1[j1] = new double*[2*N2+1];
for (j2=0; j2 <=2*N2; j2++)
{
bestActionV1[j1][j2] = new double[nStates];
bestActionW1[j1][j2] = new double[nStates];
}
}

//-----
for (j1 = 0; j1 <= 2*N1; j1++)
{
curx = (x0 + (j1-N1)*dx);
for(j2=0; j2<=2*N2; j2++) // Initialize the values of V for t=N*dt
{
curc = (c0 + (j2-N2)*dc);
for (k = 0; k < nStates; k++)
{
V1[j1][j2][k] = 0;
V2[j1][j2][k] = 0;
W1[j1][j2][k] = 0;
W2[j1][j2][k] = 0;
}
}
}

/*****/
// Compute the value of V for t=i*dt from values for t=(i+1)*dt
for(i=steps-1; i>=0; i--)
{

if(i<= 0 && i < increment)

```

```

{
P = .7125;
Q = .4325;
}
else if(i <= 2*increment)
{
P = .7225;
Q = .4075;
}
else if(i <= 3*increment)
{
P = .7475;
Q = .386;
}
else
{
P = .7775;
Q = .4025;
}

//sw = 0;
for (k = 0; k < nStates; k++)
{

for (jj1=0; jj1<=2*N1; jj1++)
{
j1 = inorder[jj1]; // correct boundary condition for j1 = 0

curx = (x0 + (j1-N1)*dx); // This is the LOG-price

driftx = kx*(thetaX - curx);
driftxx = gx*gx/2;

for(j2=0; j2<=2*N2; j2++) // loop through inventory values
{
// No injection with maximum capacity, no withdrawal with min capacity
if ((j2 == 0 && k == 1) || (j2 == 2*N2 && k == 0) || (j2 == 2*N2 && k ==2))
{
V2[j1][j2][k] = -BIG;
W2[j1][j2][k] = -BIG;
continue;
}
}
}
}

```

```

}

/* Compute partial derivatives in x */
if (j1 > 0 && j1 < 2*N1)
{
dV_dx = (V1[j1+1][j2][k] - V1[j1-1][j2][k])*(dt/(2*dx));
d2V_dx2 = (V1[j1+1][j2][k] - 2*V1[j1][j2][k] +V1[j1-1][j2][k])*(dt/dx/dx);
dV_dxRain = (W1[j1+1][j2][k] - W1[j1-1][j2][k])*(dt/(2*dx));
d2V_dx2Rain = (W1[j1+1][j2][k] - 2*W1[j1][j2][k] +W1[j1-1][j2][k])*(dt/dx/dx);

}
else if (j1 == 0) // at the bottom boundary
{
dV_dx = (V1[1][j2][k] - V1[0][j2][k])*(dt/dx);
d2V_dx2 = 0;
dV_dxRain = (W1[1][j2][k] - W1[0][j2][k])*(dt/dx);
d2V_dx2Rain = 0;
}
else // top boundary
{
dV_dx = (V1[2*N1][j2][k] - V1[2*N1-1][j2][k])*(dt/dx);
d2V_dx2 = 0;
dV_dxRain = (W1[2*N1][j2][k] - W1[2*N1-1][j2][k])*(dt/dx);
d2V_dx2Rain = 0;
}

//Compute continuation value, interpolate based on tomorrow's inventory
curc = (c0 + (j2-N2)*dc);
if (k == 0) // inject
{
alpha = inRate*V1[j1][j2+1][k] + (1-inRate)*V1[j1][j2][k];
alphaRain = inRateRain*W1[j1][j2+1][k] + (1-inRateRain)*W1[j1][j2][k];
}
else if (k == 1) // eject
{
alpha = outRate*V1[j1][j2-1][k] + (1-outRate)*V1[j1][j2][k];
alphaRain = outRateRain*W1[j1][j2-1][k] + (1-outRateRain)*W1[j1][j2][k];
}
else //store
{

```

```

alpha = V1[j1][j2][k];
alphaRain = storeRateRain*W1[j1][j2+1][k] + (1-storeRateRain)*W1[j1][j2][k];
}

// HJB equation
V2[j1][j2][k] = (1- r*dt)*((Q/(steps/days))*alphaRain
+ (1-(Q/(steps/days)))*alpha) - curc*dt*storeRate
+ driftx*dV_dx + driftxx*d2V_dx2
+ CAP[k]*dt*(exp(curx)*scaleP[k])*(scaleFactor)/moneyFactor
+ J*dt*(V1[min(j1+5,2*N1)][j2][k]-V1[j1][j2][k]);
W2[j1][j2][k] = (1- r*dt)*((1-(P/(steps/days)))*alphaRain
+ (P/(steps/days))*alpha) - curc*dt*storeRate
+ driftx*dV_dxRain + driftxx*d2V_dx2Rain
+ CAP[k]*dt*(exp(curx)*scaleP[k])*(scaleFactor)/moneyFactor
+ J*dt*(W1[min(j1+5,2*N1)][j2][k]-W1[j1][j2][k]);

}
}
}

// Select optimal action, from k=0,1,2:
for (k = 0; k < nStates; k++)
{
for(j2=0; j2<=2*N2; j2++)
{
for (j1 = 0; j1 <= 2*N1; j1++)
{
curCost = COST[k*nStates];
curMax = V2[j1][j2][0] - curCost;
for (kk = 0; kk < nStates; kk++)
{
curCost = COST[k*nStates + kk];
if (V2[j1][j2][kk] - curCost > curMax)
{
curMax = V2[j1][j2][kk] - curCost;
bestActionV1[j1][j2][k]= kk;
}
}
}
}
}

V1[j1][j2][k] = curMax; // Curmax is the best continuation value
}

```

```

}
}
}

for (k = 0; k < nStates; k++)
{
for(j2=0; j2<=2*N2; j2++)
{
for (j1 = 0; j1 <= 2*N1; j1++)
{
curCost = COST[k*nStates];
curMaxRain = W2[j1][j2][0] - curCost;
for (kk = 0; kk < nStates; kk++)
{
curCost = COST[k*nStates + kk];
if (W2[j1][j2][kk] - curCost > curMaxRain)
{
curMaxRain = W2[j1][j2][kk] - curCost;
bestActionW1[j1][j2][k]= kk;
}
}

W1[j1][j2][k] = curMaxRain; // Curmax is the best continuation value
}
}
}
}

if (i % 2000 == 0)
{
cout << i << " " << V1[N1][N2][0] << " " << V1[N1][N2][1] << endl;
cout << i << " " << W2[N1][N2][0] << " " << W2[N1][N2][1] << endl;
}

} //end main loop

// Print value function for different initial inventory levels
for (i=0; i < (N2/5); i++)
{
for(j=0; j < (N1/5); j++)

```

```

{
outDataW1 << W1[2*N1-10*j][2*N2-10*i][1] << "\t";
}
outDataW1 << endl;
}

outDataW1.close();

for (i=0; i < (N2/5); i++)
{
for(j=0; j < (N1/5); j++)
{
outDataV1 << V1[2*N1-10*j][2*N2-10*i][1] << "\t";
}
outDataV1 << endl;
}

outDataV1.close();

for (i=0; i < (N2/5); i++)
{
for(j=0; j < (N1/5); j++)
{
outDataBRV1 << bestActionV1[2*N1-10*j][2*N2-10*i][1] << "\t";
}
outDataBRV1 << endl;
}

outDataBRV1.close();

for (i=0; i < (N2/5); i++)
{
for(j=0; j < (N1/5); j++)
{
outDataBRW1 << bestActionW1[2*N1-10*j][2*N2-10*i][1] << "\t";
}
outDataBRW1 << endl;
}

outDataBRW1.close();

```

```

for (i=0; i < (N2/5); i++)
{
outDataValues << (c0 + (N2-10*i)*dc) << "\t";
for(j=0; j < (N1/5); j++)
{
outDataValues << (x0 + (N1-10*j)*dx) << "\t";
}
outDataValues << endl;
}

```

```

outDataValues.close();

```

```

// Deallocate memory
for (j1=0; j1<=2*N1; j1++)
{
for (j2=0; j2 <=2*N2; j2++)
{
delete[] V1[j1][j2];
delete[] V2[j1][j2];
}
delete[] V1[j1];
delete[] V2[j1];
}

```

```

for (j1=0; j1<=2*N1; j1++)
{
for (j2=0; j2 <=2*N2; j2++)
{
delete[] W1[j1][j2];
delete[] W2[j1][j2];
}
delete[] W1[j1];
delete[] W2[j1];
}

```

```

for (j1=0; j1<=2*N1; j1++)
{
for (j2=0; j2 <=2*N2; j2++)

```

```
{
delete[] bestActionV1[j1][j2];
delete[] bestActionW1[j1][j2];
}
delete[] bestActionV1[j1];
delete[] bestActionW1[j1];
}

return 0;

}
```