A METHODOLOGY FOR APPROXIMATING UNIT GENERATION FUNCTION AND OPTIMIZING ENERGY FUNCTION FOR HYDROPOWER RESERVOIRS

A Thesis
Presented to the Faculty of the Graduate School
of Cornell University
In Partial Fulfillment of the Requirements for the Degree of
Master of Science

by
Liang Chen
August 2014

ABSTRACT

Within a hydropower station, by optimizing the allocation of water among heterogeneous hydropower generation units, the power output could increase using same amount of water that had been used historically which maximums the benefits.

The purpose of this project is to propose a method for optimizing powerhouse function within one reservoir and give examples for 10 federal hydropower station.

Firstly, two unit generation function approximation methods were developed and compared using Least Square Cubic Spline with Specified End Conditions and Least Square Quadratic Spline. Second derivative at two ends were estimated using a quadratic Lagrange polynomial fit to the three smallest points to generate an approximate of the first derivative. To adjust the approximation method on different reservoirs, different number of intervals was applied and a few data points were deleted. In this way, the approximation function is smooth.

After getting the generation function of each unit type, the loading order of different types was identified according to the efficiency at EAOP. Allocation of water among heterogeneous hydropower generation units required both the powerhouse function and its first derivative to be continuous. It also required the operation of hydropower units to be most efficient. The economic dispatch algorithm was implemented on two reservoirs. The results indicated that the powerhouse function

was smooth and the first derivative of it was non-increasing. It is convenience using this optimization function to do further calculation.

The Fish Passage Plan (FPP) was described in the end of this thesis. FPP is developed annually by the U.S. Army Corps of Engineers and it has special requirement with allocation of water among different units. Modification of the loading order was made according to FPP.

Turbine generation data used in the report were acquired from Bonneville Power Administration.

BIOGRAPHICAL SKETCH

Liang Chen was born in Chengdu City, China. She received her Bachelor of Science degree in Environmental Engineering from Tsinghua University in Beijing, China in 2012. Liang started her Master of Science degree in Department of Civil and Environmental Engineering at Cornell University in Fall 2012 advised by Prof. Christine A. Shoemaker.

ACKNOWLEDGMENTS

In completion of my thesis, I would like to express my gratitude to my advisor Prof. Christine A. Shoemaker for her continued support and guidance. And I want to thank to Prof. Jery Russell Stedinger from the Civil and Environmental Engineering Department at Cornell for his help with algorithm and programming used in this research.

Also, I'd like to thank to my seniors Sue Nee Tan and Jonathan Lamontagne for sharing their experiences and their patience in helping me find navigation work and life at Cornell.

This research is a culmination of the work of the abovementioned people, as well as previous graduate students of Prof. Shoemaker and was made possible through financial support from BPA, and the Civil and Environmental Engineering Department at Cornell.

GLOSSARY OF BPA TERMS

- STS: submersible traveling screen.
- ESBS: extended submersible barrier screen.
- NS: No Screens.
- FPP: fish passage plan.
- PH: powerhouse.
- Dispatch: determines order in which units are started.
- Loading: determines how much water to run through each turbine that is running.

LIST OF SYMBOLS

- G: generation in MW.
- q: flow in cfs.
- H: head in ft.
- e: efficiency. The fraction of the potential energy in the falling water that is converted by the turbines to electrical energy.
- n: interval numbers when discretizing the flow domain of least-squares cubic spline.
- b(i): nodal points of the evenly spaced flow intervals, i=1,2, … ,n+1.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

Optimization covers a broad variety of topics when considering the performance of hydroelectric plants [Cook, 2008]. Various types of river and plant models [*Yeh*, 1985; *Wurbs*, 1993] that deal with parameters having a system wide scope are necessary for such optimization. Optimal operation of multi-reservoir systems is a complicated optimization problem with high- dimensional, dynamic, nonlinear, and computationally expensive characteristics. This problem [*Labadie*, 2004; *Yeh et al.*, 1992; *Cai et al.* , 2001] can be formulated as:

Maximize Expectation $\sum_t \sum_r U(Q_{r,t}, H_{r,t}, P_{r,t})$ (1)

Such that $W(Q, H, P) \geq 0$ (2)

Where the decision variables in the optimization in equation (1) are release $Q_{r,t}$ and power output $P_{r,t}$ over all reservoirs and time periods. The heads $H_{r,t}$ are functions of the decisions variables $Q_{r,t}$. $U(\,)$ in equation (1) is the utility function (which can be measured in terms of income or other metrics of hydropower output). This utility function is summed over all time period (t=1 to T) and over all reservoirs (r=1 to R). There are a number of constraints that are included in the equation (2) including minimum and maximum flow rates (e.g., fish-flush, ancillary service operations, etc.), the relationship between a release Q_r from reservoir r and the future head at r and its downstream reservoirs, etc. The $P_{r,t}$ is computed by deciding on the allocation of the flow ($q_{j,k}$) sent to each of the heterogeneous hydropower generating units.

My focus in this research is to obtain improved simulator of the power P produced by release Q given a head H within one time period and one reservoir [*Shawwash et al.* , 2000]. The simulator is coded in Matlab to model the hydraulic system. This value will be computed in advance for many values of Q and H, and stored for use in a multi period, multi reservoir model like equation (1) and (2). In particular, it addresses the optimal dispatching and loading of units (economic dispatch) within an individual plant to minimize that plant's discharge for a given load requirement or to maximize that plant's power for a given discharge requirement.

There are typically multiple units of the same type. Different kind of units are heterogeneous [*Li et al.* , 2012; *Li et al.* , 2013; *Arce et al.* , 2002]. This means that the amount of power generated for a given head and flow will be different from different types. Moreover, the operation of hydroelectric plants can be improved when these differences are known with sufficient precision. So it is crucial to get the approximated generation function for different kinds of units.

This paper gives a quick and easy method to get the input for short-term operation problem of hydropower, including fitting data to a model to calculate power output for each type of unit, deciding how to allocate water within one reservoir among its various units, and creating a piecewise linear function to simulate powerhouse function. The methodology has the advantage of: 1) generating a smooth, non-concave and continuous unit generation function fast and easily; 2) allowing further multi-reservoir, multi-time optimization based on the non-concave, non-decreasing powerhouse function for one reservoir, this pre-computed piecewise linear

powerhouse function can be used in designing a user-friendly, flexible, computational efficient and fast real time operational tool which uses linear programming instead of dynamic programming in solving optimal hydro scheduling problem.

Generation of unit is a function of the gross head and the discharge of this unit [*Shawwash et al.* , 2000]. Figure 1 shows a typical input-output curve for hydroelectric plant where the net hydraulic head is constant [*Wood and Wollenberg*, 1996].

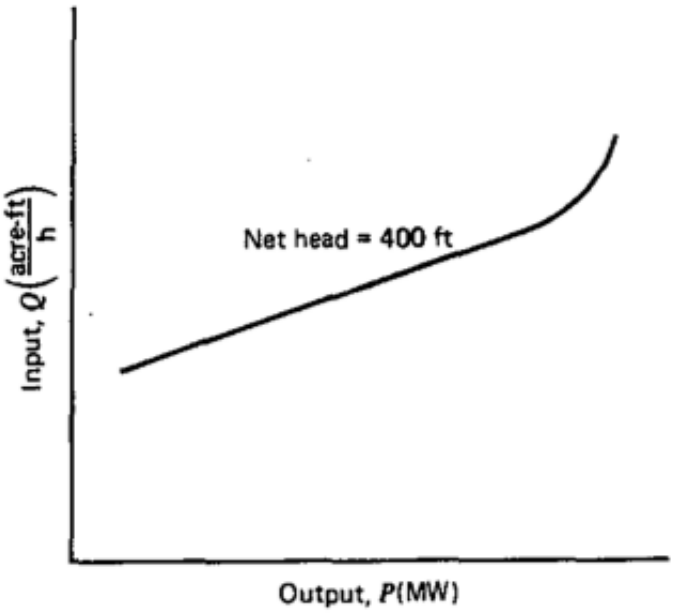


Figure 1 Hydroelectric unit input-output curve

Figure 2 shows the input-output characteristics of a hydroelectric plant with variable head. The volume of water required for a given power output decreases as the head increases.

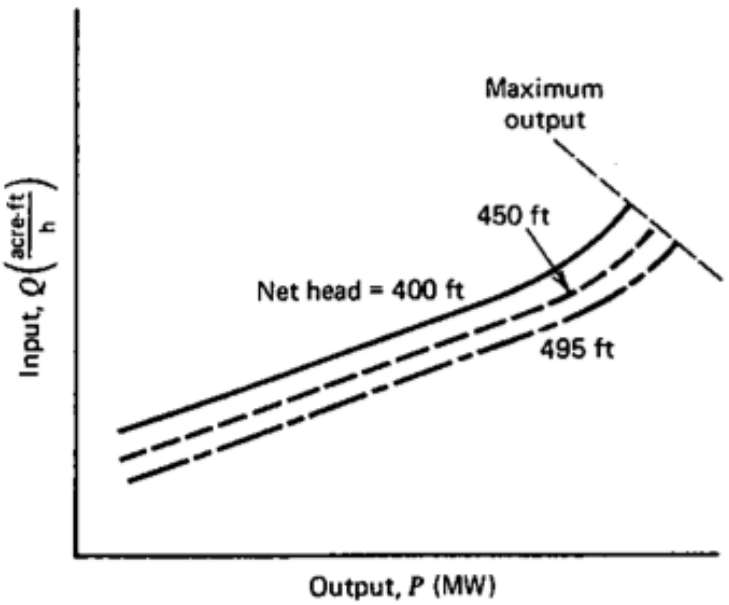


Figure 2 Input-output curves for hydroelectric plant with a variable head

This paper proposals two methods to approximate unit generation function. A modified least square cubic splines fitting in Matlab function and a least-square quadratic splines fitting. The second method uses a novel constrained least square method for solving the piecewise local curve fitting problem with global continuity constraint [*Hou et al.* , 2011].

Furthermore, this paper implements powerhouse function algorithm for ordering the dispatch of water to different type of units with the goal of obtaining the most power for a given amount of water dispatched based on economic dispatch. Economic dispatch is defined as the process of allocating generation levels to the generation units in the mix, so that the system load may be supplied entirely and most economically. Various methods were in use such as: the base load method and best point loading [*Chowdhury and Rahman*, 1990].

To summarize, the purpose of this paper is to propose a method for optimizing powerhouse function within one reservoir and give examples for 10 federal hydropower station.

This report addresses the following:

- Unit generation characteristics for each reservoir and each type,
- Approximation method of unit generation function,
- Adjustment of approximation method used on different cases,
- Ordering and loading turbine criteria,
- Optimization powerhouse function for one reservoir,
- Modification of powerhouse function for Fish Passage Plan.

Turbine generation data used in the report were acquired from Bonneville Power Administration.

CHAPTER 2 METHODOLOGY

2.1 General framework

The purpose of this project is to propose different methods to approximate hydroelectric unit generation function. This function is then used to obtain improved relationship (which is called powerhouse function in this thesis) between the generation G and release q given a head H within one time period for one reservoir. In particular, this analysis includes the aspect that in order to produce the power, from q, the water needs to be sent to hydropower generating units. There are typically different kinds of units and multiple units of the same kind. This value G will be computed in advance for many values of q and H, and stored for use in a multi period model.

The framework for generating powerhouse function for one reservoir can be described as a three-step process: 1) develop approximation generation function for hydroelectric unit; 2) with given Q and H, allocate water release among all the units of different kinds within one reservoir; 3) obtain optimized powerhouse function for one reservoir. Detailed steps of the framework are as follows:

Step 1: Develop approximation generation function for hydroelectric unit

> *Step 1.1*: With known generation and flow data, fit them to a unit generation function using Least-Squares Cubic Splines with Specified End Conditions (This can be found at: http://www.mathworks.com/help/curvefit/least-

squares-approximation-by-natural-cubic-splines.html) and Least-Square Quadratic Splines [*Hou et al.* , 2011] respectively.

Step 1.2: With the generation function G(q) obtained by previous step, calculate the Efficient Average Operating Point (EAOP, will be discussed in Section 2.2.1.1), which is the flow q at which G(q)/q is maximum, and unit efficiency achieved around EAOP.

Step 2: Allocate water release among all the units

Step 2.1: Unit efficiencies achieved around EAOP, generated by previous calculation, are used to decide units operating order according to them.

Step 2.2: Loading units with flow of their EAOP until power target (PT, will be discussed in Section 2.3) is reached.

Step 3: Obtain optimized powerhouse function

Step 3.1: According to water loading criteria, calculate power output G with given H and q using the unit generation function in Step 1.

Step 3.2: Store the value G computed in advance for many values of q and H for future use in a multi period model.

2.2 Unit generation function

The ultimate goal of this study is to optimize powerhouse function for one reservoir with given observed generation-flow data points for each type of unit. In order to do

this, the very first step is to study the characteristic of each unit type, and get the functions G(q), G'(q), G"(q), and G(q)/q. Therefore, fitting data to a model to calculate power output for each type of unit is crucial.

2.2.1 Unit generation function characteristics

2.2.1.1 Basic function of interest

Unit efficiency:

To compute the efficiency of the turbine at release q, we have:

$$e = 11.81 \times 10^3 \times \frac{G(q)}{q \times H}$$

Where e represents the fraction of the potential energy in the falling water that is converted by the turbines to electrical energy. Here G is in megawatts, q in cfs and H in feet.

(This can be found at: http://www.canyonhydro.com/micro/formulas.html)

G(q), G'(q), G"(q), and G(q)/q

G(q), G'(q), G"(q), and G(q)/q represents turbine generation function, first derivative of generation function, second derivative of generation function and average generation respectively. And the following states several matters of fact: G(q) should always be increasing. G'(q) means generation increase when adding one unit of flow at a certain base flow. G"(q) should change from positive to negative.

In my study, for a given head H, turbine efficiency is proportional to average generation G(q)/q. Maximizing efficiency is equivalent to maximizing average generation. Moreover, to order the turbine of different characteristics, it is important to know to what degree generation could increase given the increasing of certain amount of flow. Therefore, first derivative of generation function G'(q) should be studied. Also a monotonic second derivative G"(q) is a good indicate of an appropriate approximation of generation function, otherwise not.

Efficient Average Operating Point (EAOP)

According to *Stedinger et al.* (2013), the Efficient Average Operating Point (EAOP) can be used to determine distribution of water among hydropower generating units. The EAOP is the flow q where G(q)/q is maximum. Thus, running a turbine at that rate results in the maximum energy per release.

To find the maximum of the average generation rate G(q)/q, we can set to zero its derivative with respect to q. Thus we seek the q = EAOP where:

$$0 = \frac{d}{dq}\left(\frac{G(q)}{q}\right) = \frac{G'(q)}{q} - \frac{G(q)}{q^2}$$

which yields the desired relationship:

$$G'(EAOP) = \frac{G(EAOP)}{EAOP}$$

Thus, at the point where q = EAOP, the marginal generation per unit release G'(EAOP) equals the average generation rate G(EAOP)/EAOP. We use this relationship elsewhere to compute EAOP for given head. Using the fitted spline function, EAOP was obtained by evaluating G(q)/q over the range of q values in the original data to find the maximum.

Around the most efficient operating point, turbines often achieve efficiencies of 80-90%. Greater efficiencies are generally achieved with higher heads.

2.2.1.2 Two Turbine Generation Shapes

Turbine characteristics could be dramatically different from type to type. Basically, two shapes, S-shape and concave-shape, could describe a large proportion of them.

For the classic S-shape turbine generation function, the first derivative increases first, and becomes decreasing and the second derivative starts positive, and then becomes negative.

Meanwhile the first derivative of the concave-shape generation function is always decreasing and the second derivative of it is always negative. In most case with concave-shape, G'(q) > G(q)/q for all q, and EAOP is the largest allowable flow because G(q)/q continues to increase monotonically until the maximum allowable flow rate is reached.

The following two figures Figure 3 and Figure 4 are typical examples of these two shapes. These curves are based data produced by Steve Barton, BPA, and described in *Stedinger et al.* , 2013 for Grand Coulee Dam.

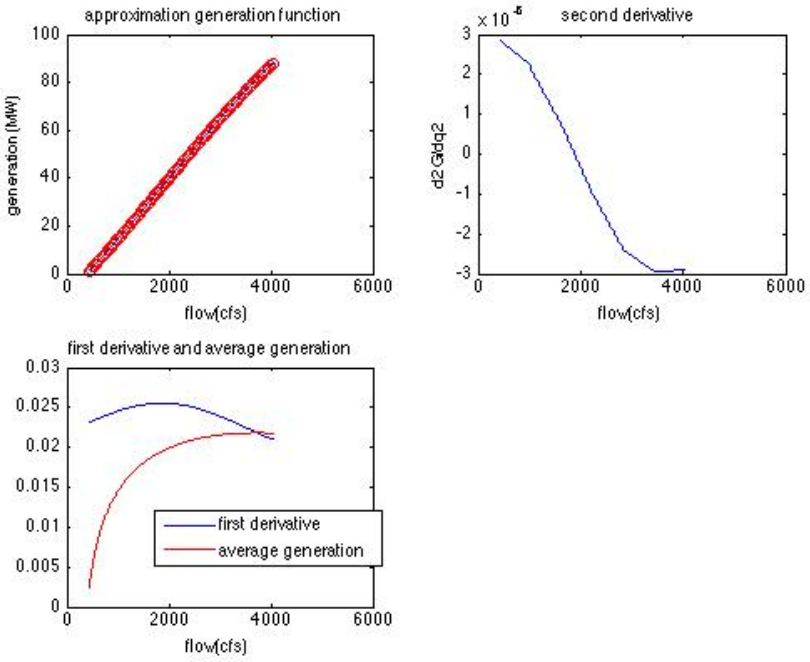


Figure 3 Grand Coulee unit type 1 Head=280 (S-Shape)--Upper left: generation versus flow G(q). Upper right: the second derivative of generation function G"(q). Lower: the first derivative G'(q) and average generation G(q)/q

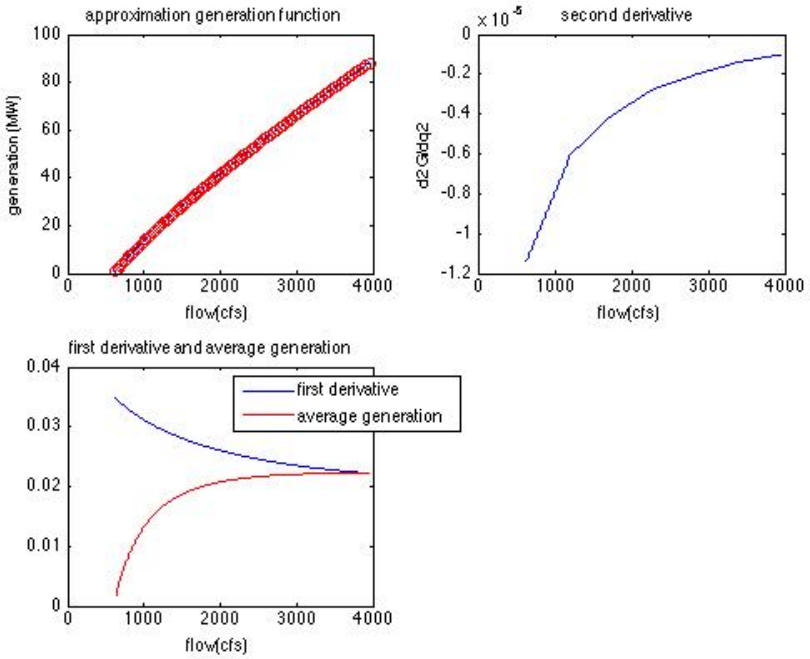


Figure 4 Grand Coulee unit type 2 Head=280 (Concave-Shape)--Upper left: generation versus flow G(q). Upper right: the second derivative of generation function G"(q). Lower: the first derivative G'(q) and average generation G(q)/q

2.2.2 Introduce and define diagnostic tools

Several simple statistics are usually being used to evaluate the fitting of approximation curve.

Mean generation is the mean of the observed generation.

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

R^2 is the square of the sample correlation coefficient between the outcomes and their predicted values.

$SS_{res} = \sum_i (y_i - f_i)^2$, the residual sum of squares

$SS_{tot} = \sum_i (y_i - \bar{y})^2$, the total sum of squares

$$R^2 = 1 - \frac{SS_{tot}}{SS_{res}}$$

Standard error is the magnitude of the average error.

$$SE = \sqrt{SS_{res}/n}$$

Residual plot is the plot of $[G_i - G(q_i)]$ versus q_i.

A good fit model of the data should have R^2 close to 1 and SE close to 0. Residual plot is a visual check of which part of the model doesn't fit the data well.

2.2.3 Least-Squares Cubic Splines with Specified End Conditions

2.2.3.1 Least-Squares Cubic Splines

Since the two shapes of generation function that introduced in section 2.2.1.2 could cover most of the cases, it is reasonable to think of using a polynomial to fit the observed data. However, it is hard to decide on the order of the polynomial. Higher order might result in capturing too much noise in the data and in turn lower order might result in losing lots of information. Another way to fit the data is using spline. Also fit is not satisfactory because even though the function is smooth, the second derivative is not monotonic due to the noise in the observed data.

Finally, *Stedinger et al.* (2013) suggests splines that are piecewise polynomial functions which combine ease of handling in a computer with great flexibility. Least-

squares cubic spline allows us to construct the least-squares approximation to given data (x, y) from the space S of "natural" cubic splines with given breaks b(1) < ...< b(l+1).

The construction of a least-squares cubic spline usually requires that we discretize the flow domain into n evenly spaced intervals that are separated by the nodes $q_i=q_1+(i-1)*\Delta q$, where i=2, … , n+1, $\Delta q= (q_{n+1}-q_1)/n$, and apply cubic polynomial fitting on each interval to minimize sum of square error.

(This can be found at: http://www.mathworks.com/help/curvefit/least-squares-approximation-by-natural-cubic-splines.html)

2.2.3.2 End Conditions with Finite Difference

The available Matlab package for least squares splines works very well. However, it employs the assumption of the cubic splines that the second derivatives are zero at the end point. This assumption did not always work well with the BPA turbine data. So *Stedinger et al.* , 2013 developed an extension of their procedure that results in a least squares spline that has a specified second derivative at each end point. Reasonable second derivative values were computed using finite difference approximations with the turbine power curve data.

Suppose the second derivatives at the two ends are s”(0) and s”(1), respectively, for a spline on [0,1]. In order to use the Matlab LS-Spline function rather than write a new function ourselves, we construct a cubic polynomial for the first and last interval with s”(0) and s”(1) equal to the specified non-zero values; furthermore, the

constructed splines has f(x) = f'(x) = f''(x) = 0 at the second largest knot, and the second smallest knot, respectively. Then the least square cubic spline was obtained by fitting the original data minus the two constructed splines function, and then adding the constructed splines to the LS-Spline to obtain our final LS-Spline approximation. Because adding two cubic splines with the same knots yields a cubic splines over those knots, adding the specialized and constructed splines to the least squares splines yields a spline function that is both a LS fit to the data and has the desired end conditions.

2.2.3.3 Least-Squares Cubic Splines with Specified End Conditions

This section describes the three key steps in the development of a LS-Spline with reasonable second-derivative end conditions.

- *Calculate second derivative at the two end points using numerical differentiation*

Using a quadratic Lagrange polynomial fit to the three smallest points to generate an approximate of the first derivative yields the approximation [Chapra *and* Canale, 2006]:

$$f'(x) = f(x_{i-1})\frac{2x - x_i - x_{i+1}}{(x_{i-1} - x_i)(x_{i-1} - x_{i+1})} + f(x_i)\frac{2x - x_{i-1} - x_{i+1}}{(x_i - x_{i-1})(x_i - x_{i+1})}$$
$$+ f(x_{i+1})\frac{2x - x_{i-1} - x_i}{(x_{i+1} - x_{i-1})(x_{i+1} - x_i)}$$

which is applicable for unequally spaced point.

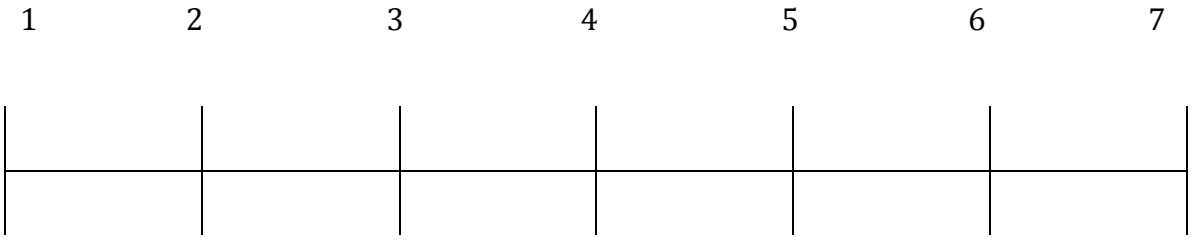
The second derivative of this linear approximation of the first derivative is:

$$f''(x) = \frac{2f(x_{i-1})}{(x_{i-1} - x_i)(x_{i-1} - x_{i+1})} + \frac{2f(x_i)}{(x_i - x_{i-1})(x_i - x_{i+1})} + \frac{2f(x_{i+1})}{(x_{i+1} - x_{i-1})(x_{i+1} - x_i)}$$

This approximation is used to obtain finite difference estimates of the second derivative of the turbine power curves at both end points.

- *Construct specialized spline function for first and last intervals*

Assume there are 6 intervals for the cubic spline corresponding to 7 knots.



Then for interval between 1 and 2, the approximation function is:

$$G = a\times(q - q_1)^3 + b\times(q - q_1)^2 + c\times(q - q_1) + d$$

In step one, the second derivative of point 1 (s"(0)) and point 7 (s"(1)) is calculated. We constructed a cubic function between 1 and 2 and another cubic function between 6 and 7 that have the desired s"(0) and s"(1).

Function 1 (between nodal points 1 and 2):

We require:

$$f''(x_1)=s''(0);\ f(x_2)=0;\ f'(x_2) = 0;\ f''(x_2) = 0$$

Suppose:

$$f(x) = a\times(x - x_2)^3 + b\times(x - x_2)^2 + c\times(x - x_2) + d$$

Then the conditions at $x = x_2$ require that

$$b = c = d = 0$$

The second derivative of this polynomial at $x = x_1$ is

$$f"(x) = 6\times a\times(x_1 - x_2)$$

which must equal to s”(0). Thus we obtain

$$a = \frac{s"(0)}{6\times(x_1 - x_2)}$$

$$f(x) = \frac{s"(0)\times(x - x_2)^3}{6\times(x_1 - x_2)}$$

Function 2 (between nodal points 6 and 7):

Use the same method as above which yields:

$$f(x) = \frac{s"(1)\times(x - x_6)^3}{6\times(x_7 - x_6)}$$

- *Fit a Least square cubic spline to the original data minus the constructed spline functions.*

There might be more than 1 data point between knot 1 and 2. For all of the data points, calculate the value of the constructed function, then use original data minus

constructed function value. Followed, apply least square cubic spline over the whole range use the updated value. So, the approximation function is the cubic spline function plus the constructed function.

- *Calculate coefficients for the generation function*

Combine the coefficients of the special functions for the two end intervals with the LS spline for the whole interval so that our approximation of the turbine power function is described by one single cubic spline that satisfied specified second derivatives at the two end points. The equations for this conversion are relatively easy to obtain. For the last interval, $s''(1)(x-x_6)^3/[6(x_7 - x_6)]$ is of exactly the correct form to add the coefficients of $(x-x_6)^3$ for the least squares spline and for f(x). For the first interval one can expand $s''(0)(x-x_2)^3/[6(x_1 - x_2)]$ in powers of $(x-x_1)$ to get the coefficients needed to add the two cubic polynomials in powers of $(x-x_1)$ over the first interval.

The plots below (Figure 5, 6, 7, 8) display the First and Second derivatives of cubic spline approximations derived from Grand Coulee turbines type 1 with a head equal to 302 ft using different methods. We are trying to have smaller end intervals to minimize the distortion and thus tried 1/4 and 1/64 length of other intervals as end interval to see how they would work. The analysis showed that with 1/4 length of others we could still see distortion in the first and last panels. With 1/64, the largest and smallest panels were so small, we could hardly see the distortion in f'(q), but it is still there as shown by the f"(q) plot. Thus we proceeded to modify the least

squares spline procedure to allow non-zero second derivatives at the first and last knot.

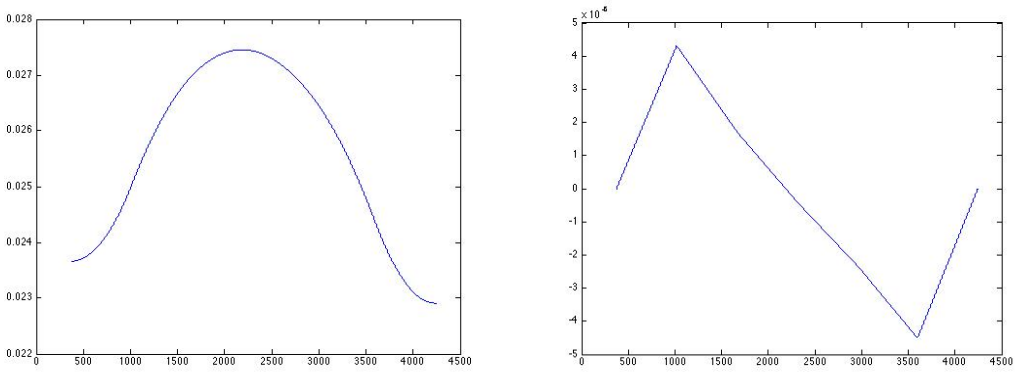


Figure 5 These two plots show least square cubic spline with zero second-derivative end conditions and six intervals of equal length

From Figure 5 it's noted that making the second derive of the spline go to zero is both unnatural, and distorts the first derivative function at the end points. Attempts were made on making the first and last intervals smaller to decrease the error. The result is displayed in the figures below. This did not solve the problem because one can still see the distortion in the first and last intervals.

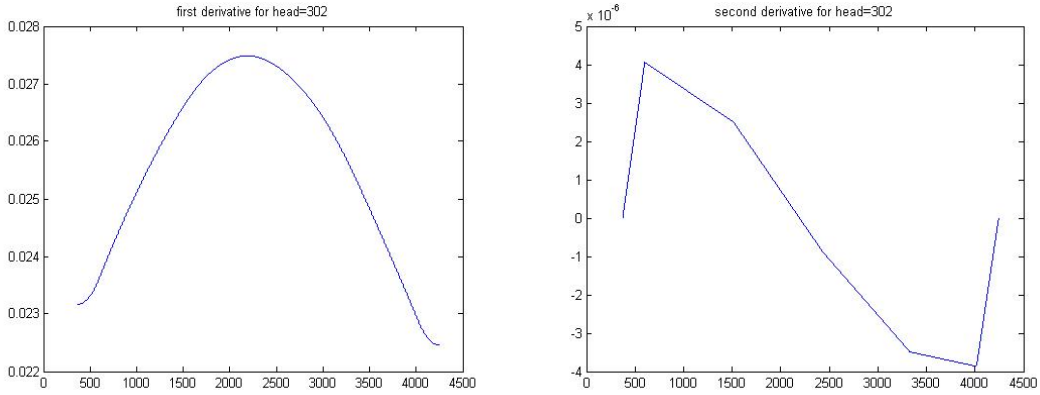


Figure 6 These two plots show least square cubic spline with zero second-derivative end conditions with first and last intervals that are short 1/4 length

The following two plots are least square cubic spline with zero second-derivative end conditions and using 1/64 length of the other interval as the first and last ones. The fitted spline function improves a lot. Compare this pair with the former ones, even though the trend of the second derivative is the same, the first derivative doesn't have visible flat tails at two ends because the distortion is restricted to a small interval. The second derivative function does behave poorly at the end points.

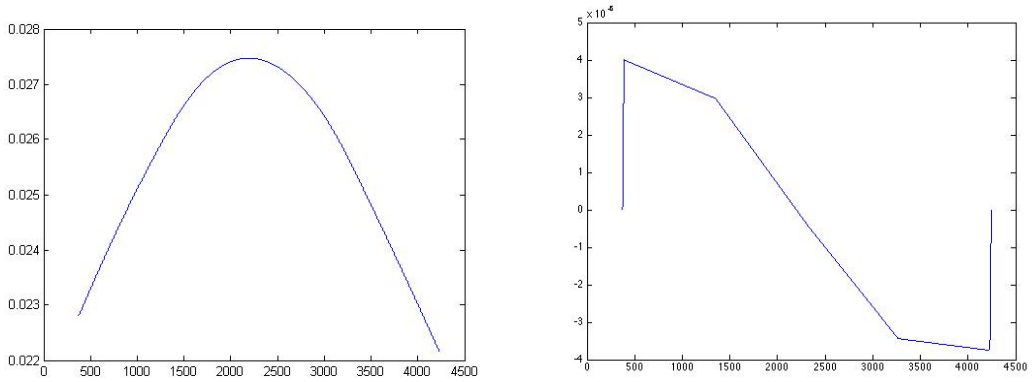


Figure 7 These two plots show least square cubic spline with zero second-derivative end conditions with first and last intervals that are short 1/64 length

The last two plots are least square cubic spline plus constructed spline functions for the two ends to achieve specified second derivatives. It provides a much better second derivative model because it doesn't have big jumps at the end. The first derivative function is very smooth.

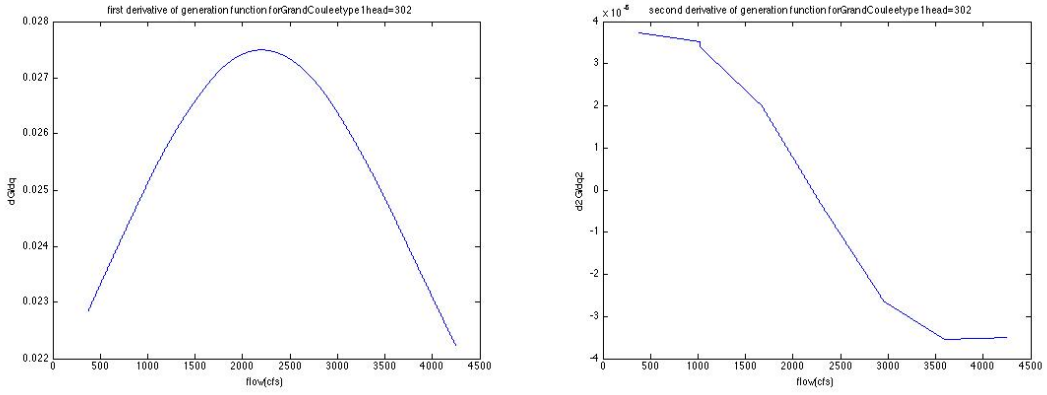


Figure 8 These two plots show least square cubic spline with specified non-zero second-derivative with equally spaced intervals

Also, the following table compares the residual sum of square and R^2 of each method with $\bar{y} = 50.5$ MW.

Table 1 residual sum of square and R^2 of each method

Method	Equal	1/4	1/64	Specified second derivative
SE	0.0192	0.0033	0.0034	0.0088
R^2	1	1	1	1

According to table 1, using smaller length as the end intervals improves the approximation function a lot. Also with the specified second derivative, the distortions in the first and last intervals disappear without significant increase of the residual sum of square.

2.2.4 Least-Squares Quadratic Splines

2.2.4.1 Method Description

In order to get approximations of generation function for certain head and certain type of unit, assume we have flow-generation data (x_i, y_i), i=1, 2, … , n, in which x refers to flow q, y refers to generation G and n is the number of data points.

Divide the data into k group:

$$S_1 = \left\{\left(x_i^{(1)}, y_i^{(1)}\right), i = 1,2, \dots, n_1\right\}$$

$$S_2 = \left\{\left(x_i^{(2)}, y_i^{(2)}\right), i = 1,2, \dots, n_2\right\}$$

…

$$S_k = \left\{\left(x_i^{(k)}, y_i^{(k)}\right), i = 1,2, \dots, n_k\right\}$$

in which,

$$b_0 = x_1 \le x_i^{(1)} \le x_{n_1} \le b_1$$

$$b_1 \le x_{n_1+1} \le x_i^{(2)} \le x_{n_1+n_2} \le b_2$$

…

$$b_{k-1} \le x_{n_1+n_2+\cdots+n_{k-1}+1} \le x_i^{(k)} \le x_{n_1+n_2+\cdots+n_k} = b_k$$

and

$$n_1 + n_2 + \cdots + n_k = n$$

The way to calculate nodal points b_i, i=0, 1, 2, … , k, are as following. Assume the max flow is $flow_{max}$, the min flow is $flow_{min}$, then the interval between groups is:

$$\text{interval} = \frac{1}{k} \times (\text{flow}_{max} - \text{flow}_{min})$$

the nodal points are b_0, b_1, … ,b_k, respectively with

$$b_0 = \text{flow}_{min}, \qquad b_k = \text{flow}_{max}, \qquad b_{i+1} = b_i + \text{interval}$$

For quadratic spline, assume basis function:

$$h_1(x) = 1, \qquad h_2(x) = x, \qquad h_3(x) = x^2$$

and

$$h(x) = [h_1(x), h_2(x), h_3(x)]$$

$\left\{\alpha_j^{(i)}\right\}$, j = 1,2,3 and i = 1,2, … , k are the coefficients

the approximated function is:

$$f(x) = \begin{cases} f_1(x) = \sum_{j=1}^{3} \alpha_j^{(1)} h_j(x), & b_0 \le x \le b_1 \\ f_2(x) = \sum_{j=1}^{3} \alpha_j^{(2)} h_j(x), & b_1 \le x \le b_2 \\ \cdots & \\ f_k(x) = \sum_{j=1}^{3} \alpha_j^{(k)} h_j(x), & b_{k-1} \le x \le b_k \end{cases}$$

Using least square approximation, we need to optimize all the coefficients $\left\{\alpha_j^{(i)}\right\}$ to minimize the error between the approximated function and the given data, such that the approximated function f(x) and the first derivative of it f′(x) are continuous, the equations are as follow:

$$\min_{\alpha_j^{(1)},\ldots,\alpha_j^{(k)}}\left[\sum_{i=1}^{n_1}\left(f_1\left(x_i^{(1)}\right)-y_i^{(1)}\right)^2+\sum_{i=1}^{n_2}\left(f_2\left(x_i^{(2)}\right)-y_i^{(2)}\right)^2+\cdots+\sum_{i=1}^{n_k}\left(f_k\left(x_i^{(k)}\right)-y_i^{(k)}\right)^2\right]$$

$$\text{s.t.}\quad f_1(b_1)=f_2(b_1), f_2(b_2)=f_3(b_2), \ldots, f_{k-1}(b_{k-1})=f_k(b_{k-1})$$

$$f_1'(b_1)=f_2'(b_1), f_2'(b_2)=f_3'(b_2), \ldots, f_{k-1}'(b_{k-1})=f_k'(b_{k-1})$$

In which the constraint is to make sure consistency of the approximation function and its first derivative in the spline function.

2.2.4.2 Implementation in Matlab

The Matlab function fmincon is used to find minimum of constrained nonlinear multivariable function. The equation looks like:

$$\min_x f(x) \text{ such that } \begin{cases} c(x)\le 0 \\ ceq(x)=0 \\ A\cdot x\le b \\ Aeq\cdot x=beq \\ lb\le x\le ub, \end{cases}$$

b and beq are vectors, A and Aeq are matrices, c(x) and ceq(x) are functions that return vectors, and f(x) is a function that returns a scalar. f(x), c(x), and ceq(x) can be nonlinear functions. x, lb, and ub can be passed as vectors or matrices.

The syntax looks like:

```
[x,fval] = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)
```

(This can be found at: http://www.mathworks.com/help/optim/ug/fmincon.html)

To solve this problem in Matlab using function fmincon:

$$X_i = \begin{bmatrix} h(x_1^{(i)}) \\ h(x_2^{(i)}) \\ \dots \\ h(x_{n_i}^{(i)}) \end{bmatrix}, \qquad y_k = \begin{bmatrix} y_1^{(i)} \\ y_2^{(i)} \\ \dots \\ y_{n_i}^{(i)} \end{bmatrix}, \qquad \alpha_k = \begin{bmatrix} \alpha_1^{(i)} \\ \alpha_2^{(i)} \\ \alpha_3^{(i)} \end{bmatrix}, \qquad i = 1,2,\dots,k$$

$$y = \left[y_1^T, y_2^T, \dots, y_k^T\right]^T, \qquad \alpha = \left[\alpha_1^T, \alpha_2^T, \dots, \alpha_k^T\right]^T$$

$$X = \begin{bmatrix} X_1 & 0 & \dots & 0 \\ 0 & X_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & X_k \end{bmatrix}$$

$$Z = \begin{bmatrix} h(b_1) & -h(b_1) & 0 & \dots & 0 & 0 \\ 0 & h(b_2) & -h(b_2) & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & -h(b_{k-2}) & 0 \\ 0 & 0 & 0 & \dots & h(b_{k-1}) & -h(b_{k-1}) \end{bmatrix}$$

$$Y = \begin{bmatrix} \alpha_2 - \alpha_1 & 0 & \dots & 0 \\ 0 & \alpha_3 - \alpha_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \alpha_k - \alpha_{k-1} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 2 \\ 0 & -1 & 0 \\ 2 & 0 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} B & 0 & \dots & 0 \\ 0 & B & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & B \end{bmatrix}_{[3(k-1)]^2}$$

The optimization equation turns to be [*Gluss*, 1964]:

$$\min_{\alpha} \|X\alpha - y\|^2$$

$$\text{s.t. } Z\alpha = 0, \qquad Y^T(AY) = 0$$

That is because:

$$\begin{cases} f_1(b_1) = f_2(b_1) \\ f_1'(b_1) = f_2'(b_1) \end{cases}$$

$$\rightarrow \begin{cases} \alpha_1^{(1)} + \alpha_2^{(1)} b_1 + \alpha_3^{(1)} b_1^2 = \alpha_1^{(2)} + \alpha_2^{(2)} b_1 + \alpha_3^{(2)} b_1^2 & (1) \\ \alpha_2^{(1)} + 2\alpha_3^{(1)} b_1 = \alpha_2^{(2)} + 2\alpha_3^{(2)} b_1 & (2) \end{cases}$$

from (2), we get

$$b_1 = \frac{\alpha_2^{(1)} - \alpha_2^{(2)}}{2\times(\alpha_3^{(2)} - \alpha_3^{(1)})}$$

plug into (1), we get

$$4\times\left(\alpha_3^{(1)} - \alpha_3^{(2)}\right)\times\left(\alpha_1^{(1)} - \alpha_1^{(2)}\right) = (\alpha_2^{(1)} - \alpha_2^{(2)})^2$$

In my problem, the independent variable is α. Aeq is Z, beq is 0. ceq(x) is $Y^T(AY)$. The function f(x) is $\|X\alpha - y\|^2$. In options, I choose 'interior-point' as my algorithm.

In this optimization problem, there are two very strong linear and nonlinear equation constraints. Therefore if the initial point x_0 is not picked properly, it is very likely that the optimization will stop with a local minimum.

This conjecture was verified when I first set the initial point x0 to be 0 in this problem. It turned out that the error blew out to be 10 to the 8th and the fmincon stopped because the size of the current step is less than the selected value of the step size tolerance.

In order to solve for the better initial point x_0, within every interval, three points were picked, (xi, yi), i=1, 2, 3, which represents the first, the last and the middle one respectively. And the coefficients of the quadratic polynomial that went through these three points were calculated.

$$x\alpha = y \quad \text{and} \quad \alpha = \frac{y}{x}$$

In which,

$$x = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix}, \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}, \qquad \alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}$$

Then take α as the initial points for this interval.

2.3 Powerhouse function based on economic dispatch

In *Stedinger et al.* (2013), it described a method for ordering the dispatch of water to different type of turbines with the goal of obtaining the most power for a given amount of water dispatched. That method is described here.

Step a: Order turbines such that turbine 1 has greatest efficiency $G(EAOP_1)/q$ at its $EAOP_1$, turbine 2 is second,..., and the last turbine k has the lowest $G(EAOP_k)/q$. This ignores turbine size, and the convenience of loading and shutting down turbines.

Step b: For power target PT, optimal solution starts by loading turbines of type 1 with flow $EAOP_1$ until PT is reached, if possible. If there are n_1 turbines of type 1, the load can be met by loading only turbine of type 1 at flow $EAOP_1$ if

$$PT \leq n_1 \times G_1(EAOP_1)$$

Step c: If the inequality above is not met, then before loading units of type 2, the efficient type 1 units are run beyond their most efficient operating point until their efficiency drops to that of type 2 units.

Increase the average flow q/n_1 through all type 1 units, until either PT is met, or the following equations is satisfied:

$$G_1'(q_1) = G_2'(EAOP_2)$$

where $q_1 = q/n_1 \leq q_1^{max}$ (flow through type 1 units).

At that point it becomes optimal to bring on type 2 units for part of all of the period, while leaving the flow through type 1 units at $q_1{}^{[2]}$ where

$$G_1'\left(q_1{}^{[2]}\right) = G_2'(EAOP_2)$$

unless the solution is constrained by $q_1 \leq q_1^{max}$.

Continue to load type 2 units until all are operating the full period at flow $EAOP_2$.

Step d: When type 2 units are fully loaded at $EAOP_2$ and

$$PT \geq n_1 \times G_1\left(q_1{}^{[2]}\right) + n_2 \times G_2(EAOP_2)$$

It is time to increase the flow in all units until their marginal efficiency equals $G_3'(EAOP_3)$. So increase q_1 and q_2 so that total flow:

$q = n_1 \times q_1 + n_2 \times q_2$ attempts to yield generation PT while maintaining

$G_1'(q_1) = G_2'(q_2) > G_3'(EAOP_3)$ and $q_i \leq q_i^{max}$ for i = 1,2

at all times to maximize efficiency with all of the $n_1 + n_2$ units running.

The following is an explanation of $G_1'(q_1) = G_2'(q_2)$.

$$G = n_1 \times G_1(q_1) + n_2 \times G_2(q_2) = n_1 \times G_1(q_1) + n_2 \times G_2\left(\frac{q - n_1 \times q_1}{n_2}\right)$$

$$G' = n_1 \times G_1'(q_1) + n_2 \times G_2'(q_2) \times \left(-\frac{n_1}{n_2}\right) = n_1 \times (G_1'(q_1) - G_2'(q_2))$$

In order to maximize G, we need the first derivative of G to be 0, so:

$$G_1'(q_1) = G_2'(q_2)$$

Step e: The previous step stops when $G_1'(q_1) = G_2'(q_2) = G_3'(EAOP_3)$. Then it is time to maintain the current flow rates in units type 1 and 2 at $q_1^{[3]}$ and $q_2^{[3]}$, and to begin to bring on units of type 3 for all or part of the period. This means that all operating units will have the same efficiency:

$$G_1'\left(q_1^{[3]}\right) = G_2'\left(q_2^{[3]}\right) = G_3'(EAOP_3)$$

unless constrained by $q_i \leq q_i^{max}$ for i = 1,2.

And type 3 units are loaded only when their average efficiency is at least that of unit types 1 and 2.

When all of the unit type 3 units are running at q_3 = $EAOP_3$ we need to continue with step d above to increase the generation of all three types of units so that:

$q = n_1 \times q_1 + n_2 \times q_2 + n_3 \times q_3$ attempts to yield generation PT while maintaining

$$G_1'(q_1) = G_2'(q_2) = G_3'(q_3) > G_4'(EAOP_4) \text{ and } q_i \leq q_i^{max} \text{ for i = 1,2,3}$$

Step f: When we reach

$$G_1'\left(q_1^{[4]}\right) = G_2'\left(q_2^{[4]}\right) = G_3'\left(q_3^{[4]}\right) = G_4'(EAOP_4)$$

with

$$PT \geq n_1 \times G_1\left(q_1^{[4]}\right) + n_2 \times G_2\left(q_2^{[4]}\right) + n_3 \times G_3\left(q_3^{[4]}\right)$$

it is time to add units of type 4 at flow rate $EAOP_4$.

Step g: Algorithm continues as necessary. When no new units can be loaded, and if all units reach their maximum turbine release rate and q_i^{max} for a given head, the maximum powerhouse generation has been reached, which may be less than PT.

As for the case with three different turbine types, the powerhouse function will have 6 different parts, three constant slope parts with two transition parts between the first two and the last two and a final part. The illustration is as below.

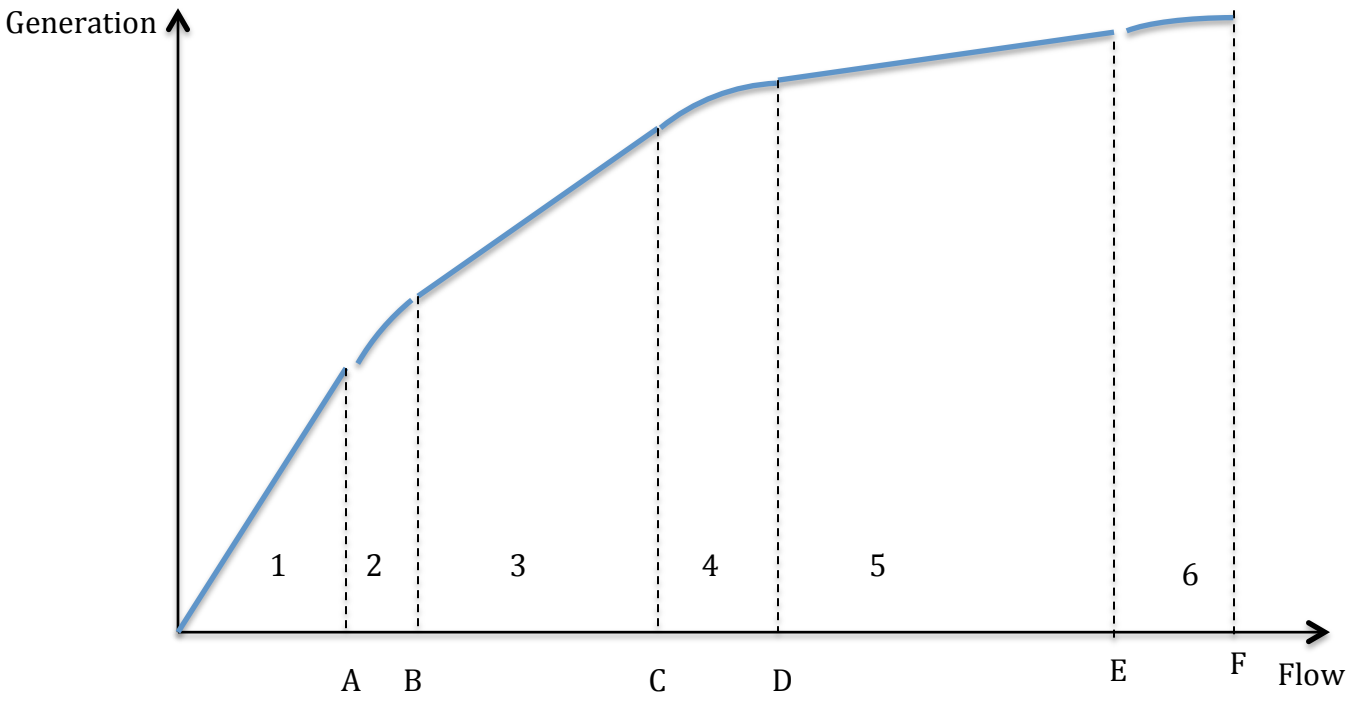


Figure 9 powerhouse curve example with three different turbine types: number in the figure refer to part, character refer to total flow of certain point

Table 2 summary of how to calculate flow for every important point in Figure 9

Point	Equation for computation
A	$n_1 \times EAOP_1$

B	$n_1 \times q_1^{[2]}$ ($q_1^{[2]}$ is defined in step c)
C	$n_1 \times q_1^{[2]} + n_2 \times EAOP_2$
D	$n_1 \times q_1^{[3]} + n_2 \times q_2^{[3]}$ ($q_1^{[3]}$ and $q_2^{[3]}$ are defined in step e)
E	$n_1 \times q_1^{[3]} + n_2 \times q_2^{[3]} + n_3 \times EAOP_3$
F	$n_1 \times q_1^{max} + n_2 \times q_2^{max} + n_3 \times q_3^{max}$

I developed with Prof. Shoemaker the equations in Table 3 to calculate powerhouse function and amount of flow going through each type of turbine.

Table 3 summary of how to calculate powerhouse function

Part	Range	Equation for powerhouse function	Equation for amount of flow going through each type of turbine
1	[0, A]	$\frac{q}{EAOP_1} \times G_1(EAOP_1)$	$q < EAOP_1$, all through the 1st unit of type 1 $EAOP_1 < q < 2 \times EAOP_1$, 1st unit full time, 2nd unit of type 1 a fraction of time…
2	[A, B]	$n_1 \times G_1(\frac{q}{n_1})$	All through type 1 with same flow for all the units
3	[B, C]	$n_1 \times G_1({q_1}^{[2]}) + \frac{q - n_1 \times {q_1}^{[2]}}{EAOP_2} \times G_2(EAOP_2)$	Type 1: $n_1 \times {q_1}^{[2]}$ Type 2: $q - n_1 \times {q_1}^{[2]}$ ${q_1}^{[2]}$ is defined in step c
4	[C, D]	$Max: n_1 \times G_1(q_1) + n_2 \times G_2(q_2)$ $s.t. n_1 \times q_1 + n_2 \times q_2 = q$ $q_1 \geq {q_1}^{[2]}, \quad q_2 \geq EAOP_2$	Type 1: $n_1 \times q_1$ all units with same flow Type 2: $n_2 \times q_2$ all units with same flow

5	[D, E]	$n_1 \times G_1(q_1^{[3]}) + n_2 \times G_2(q_2^{[3]}) + \frac{q - n_1 \times q_1^{[3]} - n_2 \times q_2^{[3]}}{EAOP_3} \times G_3(EAOP_3)$	Type 1: $n_1 \times q_1^{[3]}$ Type 2: $n_2 \times q_2^{[3]}$ Type 3: $q - n_1 \times q_1^{[3]} - n_2 \times q_2^{[3]}$ $q_1^{[3]}$ and $q_2^{[3]}$ are defined in step e
6	[E, F]	$Max: n_1 \times G_1(q_1) + n_2 \times G_2(q_2) + n_3 \times G_3(q_3)$ $s.t. n_1 \times q_1 + n_2 \times q_2 + n_3 \times q_3 = q$ $q_1^{max} \geq q_1 \geq q_1^{[3]}, q_2^{max} \geq q_2 \geq q_2^{[3]}, q_3^{max} \geq q_3 \geq EAOP_3$	Type 1: $n_1 \times q_1$ all units with same flow Type 2: $n_2 \times q_2$ all units with same flow Type 3: $n_3 \times q_3$ all units with same flow

CHAPTER 3 RESULTS AND DISCUSSION

3.1 Case description

This project is based on 10 federal hydropower plants on the Columbia basin which belong to BPA (Bonneville Power Administration) system. BPA's service territory includes Idaho, Oregon, Washington, western Montana and small parts of eastern Montana, California, Nevada, Utah and Wyoming. This project focuses on 10 plants as shown in Figure 10 and Figure 11.

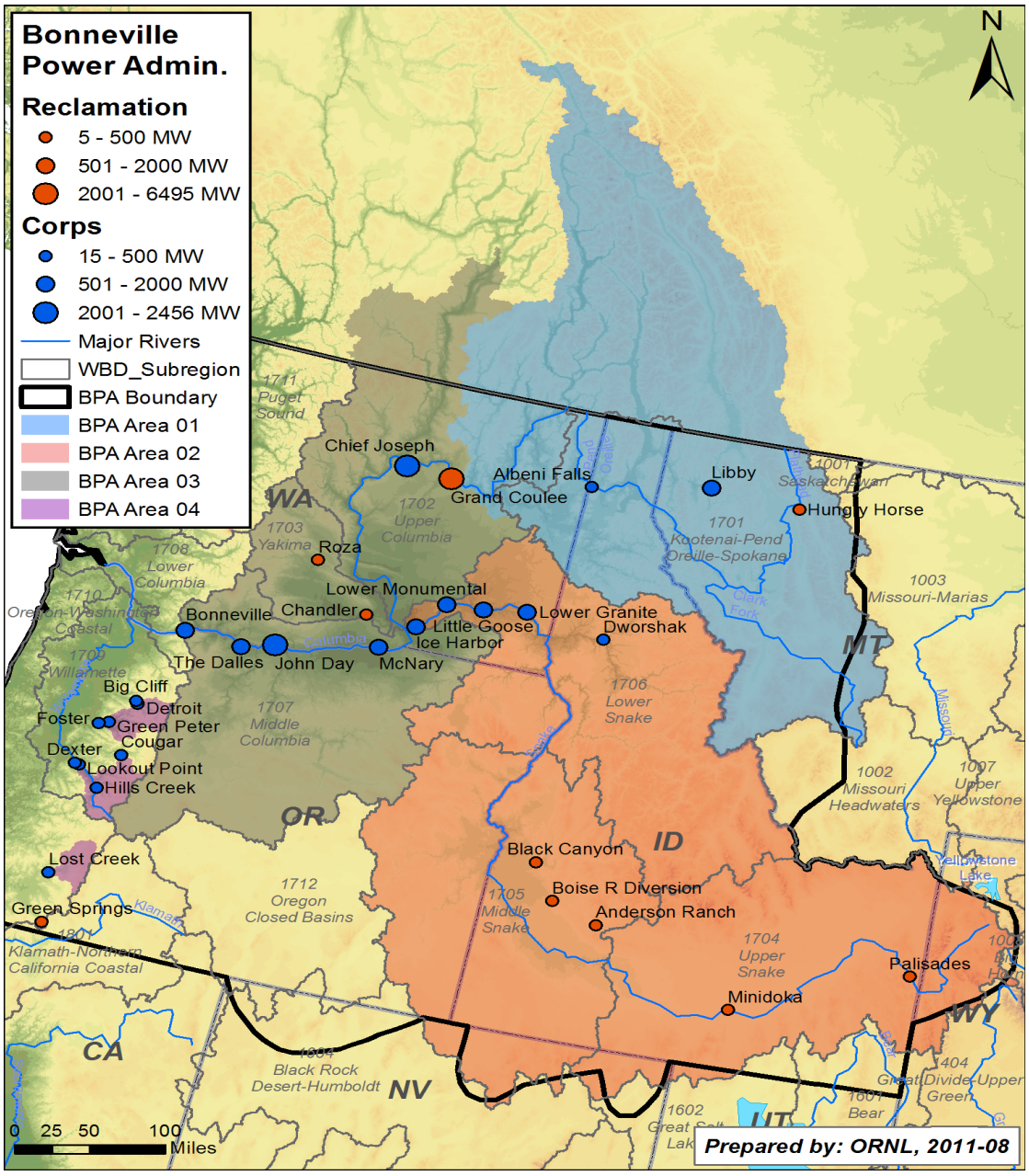


Figure 10 BPA's service territory

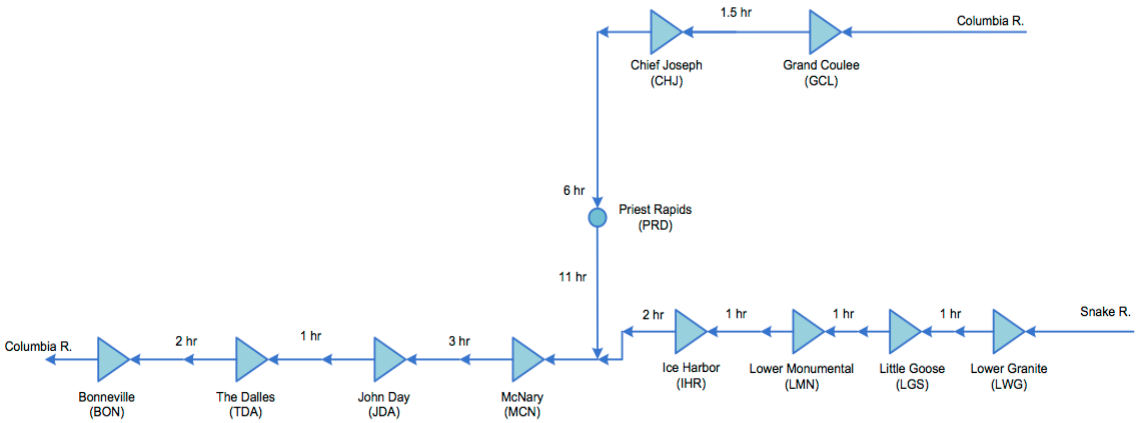


Figure 11 Reservoir Network Topology: 10-Project Schematic and Hydraulic Lag Times for the CV-STR Hydro Modeling Pilot Sow

3.2 Generation Data and Spline Analysis

There are totally 10 federal projects: Grand Coulee, Chief Joseph, Lower Granite, Little Goose, Lower Monumental, Ice Harbor, Mc Nary, John Day, The Dalles, and Bonneville. And 7 of them have 2 different operation situations: with and without screens, so there are 17 different reservoirs to study. The figures below compare the least squares cubic spline function (with end-point second derivatives specified) with the original data, report the first derivative of the spline function and average generation G(q)/q, the second derivative , and the error of the spline function.

3.2.1 Grand Coulee

Grand Coulee has 4 different turbine types. The following table summaries the unit characteristics of different turbines.

Table 4 Grand Coulee turbine generation characteristics

	Unit 1-9 (TYPE 1)	Unit 10-18 (TYPE 2)	Unit 19-21 (TYPE 3)	Unit 22-24 (TYPE 4)
Maximum Generation (MW)	119	125	705	805
Minimum Generation (MW)	0	0	0	0
Maximum Flow (cfs)	4481	4416	29748	33391
Minimum Flow (cfs)	277	461	4584	6549

To approximate the generation function for Grand Coulee, 6 equally spaced intervals and specified second derivative calculated using first 3 and last 3 points at two ends for the least square cubic spline were used. The following 4 figures are the 4 types respectively (take head=280ft for example).

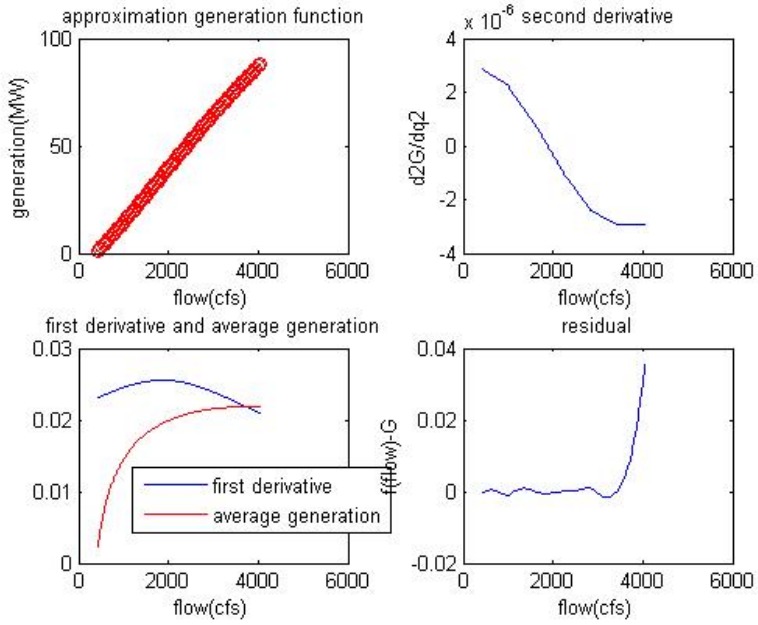


Figure 12 Grand Coulee unit type 1 Head=280, $\bar{y} = 44.5$, SE=0.0068, R^2=1 (S-Shape)--Upper left: the original generation versus flow G(q) shown as circles and the fitted least-square spline with specified end condition. Upper right: the second derivative of spline generation function G"(q). Lower left: the

first derivative G'(q) and average generation G(q)/q of the spline. Lower right: the error for the fitted function

As discussed before, the maximum of G(q)/q occurs at the flow EAOP where G'(q) = G(q)/q. The lower left plot in Figure 12 shows that EAOP occurs very close to the maximum turbine flow. G'(q) and G(q)/q are both very flat. Here the second derivative starts positive, and then becomes negative. So this is the classic S-shaped turbine generation function.

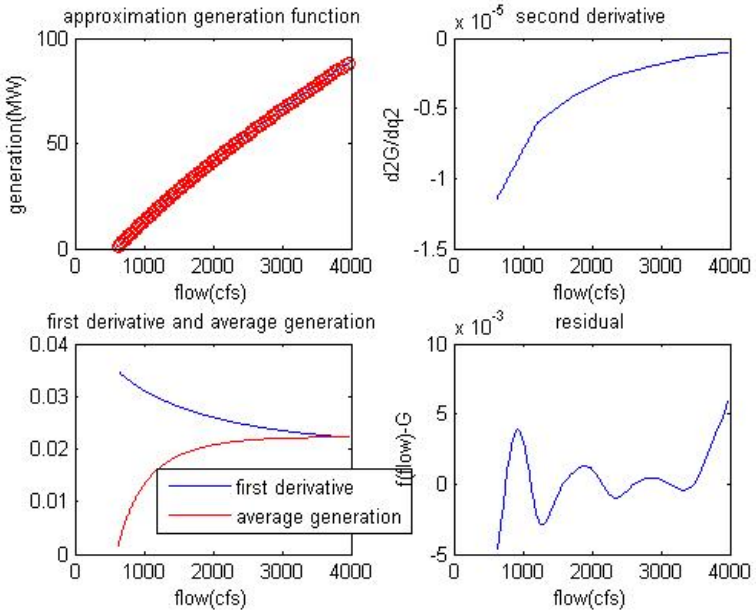


Figure 13 Grand Coulee unit type 2 Head=280, $\bar{y} = 44.5$, SE=0.0019, R^2=1 (Concave)--Upper left: the original generation versus flow G(q) shown as circles and the fitted least-square spline with specified end condition. Upper right: the second derivative of spline generation function G"(q). Lower left: the first derivative G'(q) and average generation G(q)/q of the spline. Lower right: the error for the fitted function

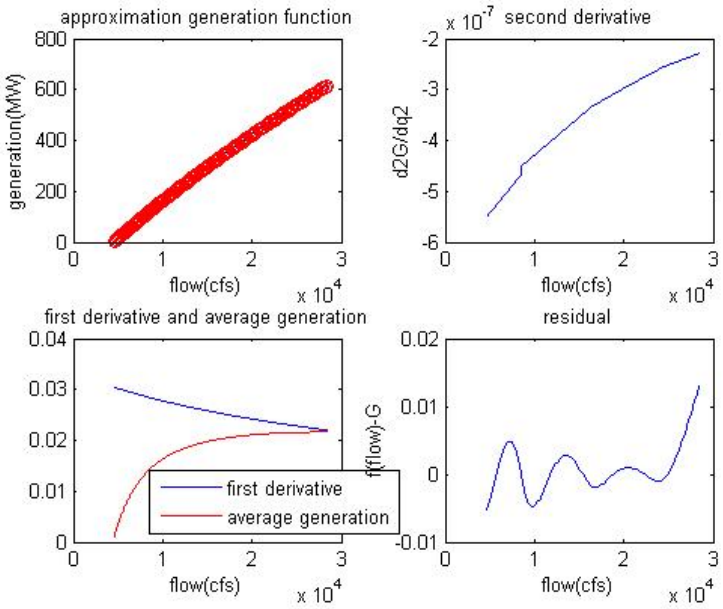


Figure 14 Grand Coulee unit type 3 Head=280, $\bar{y} = 310$, SE=0.0034, R^2=1 (Concave)--Upper left: the original generation versus flow G(q) shown as circles and the fitted least-square spline with specified end condition. Upper right: the second derivative of spline generation function G"(q). Lower left: the first derivative G'(q) and average generation G(q)/q of the spline. Lower right: the error for the fitted function

Note that in these two cases (Grand Coulee TYPE 2 and TYPE 3) the second derivative is always negative, and G'(q) > G(q)/q for all q. In this instance, EAOP is the largest allowable flow because G(q)/q continues to increase monotonically until the maximum allowable flow rate is reached.

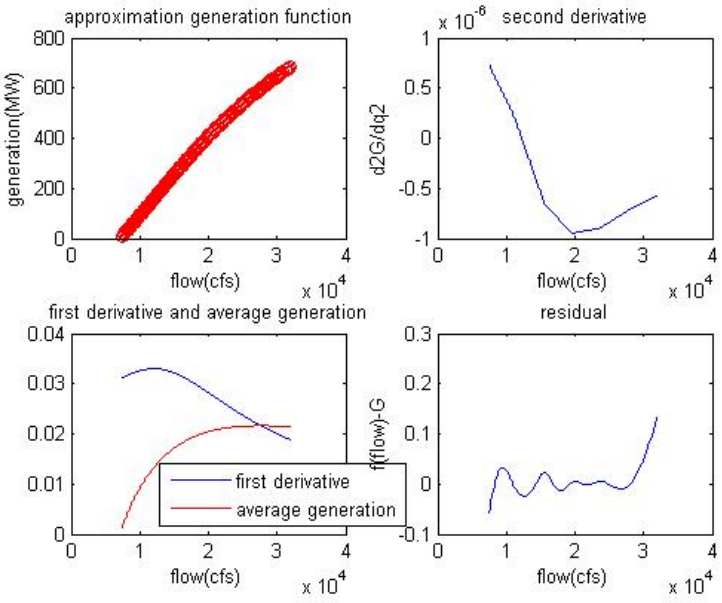


Figure 15 Grand Coulee unit type 4 Head=280, $\bar{y} = 347.5$, SE=0.0284, R^2=1 (S-shape & Concave)--Upper left: the original generation versus flow G(q) shown as circles and the fitted least-square spline with specified end condition. Upper right: the second derivative of spline generation function G"(q). Lower left: the first derivative G'(q) and average generation G(q)/q of the spline. Lower right: the error for the fitted function

Grand Coulee Type 4 is a combination of both shapes with S-shape in the beginning and concave shape coming after. In this case, the second derivative changes from positive to negative but decreases first and increases after.

All the approximations for Grand Coulee have R^2=1 which is a good indication of good fit.

3.2.2 Chief Joseph

Chief Joseph has 3 different turbine types. The following table summaries the unit characteristics of different turbines.

Table 5 Chief Joseph turbine generation characteristics

	Unit 1-4, 15, 16 (TYPE 1)	Unit 5-14 (TYPE 2)	Unit 17-27 (TYPE 3)

Maximum Generation (MW)	96	98	103
Minimum Generation (MW)	28.5	27.5	33.5
Maximum Flow (cfs)	8547	7992	8689
Minimum Flow (cfs)	2813	2665	3288

To approximation the generation function for Chief Joseph, 6 equally spaced intervals and specified second derivative calculated using first 3 and last 3 points at two ends for the least square cubic spline were used. The results for type 1 are shown in Figure 16 (take head=170ft for example).

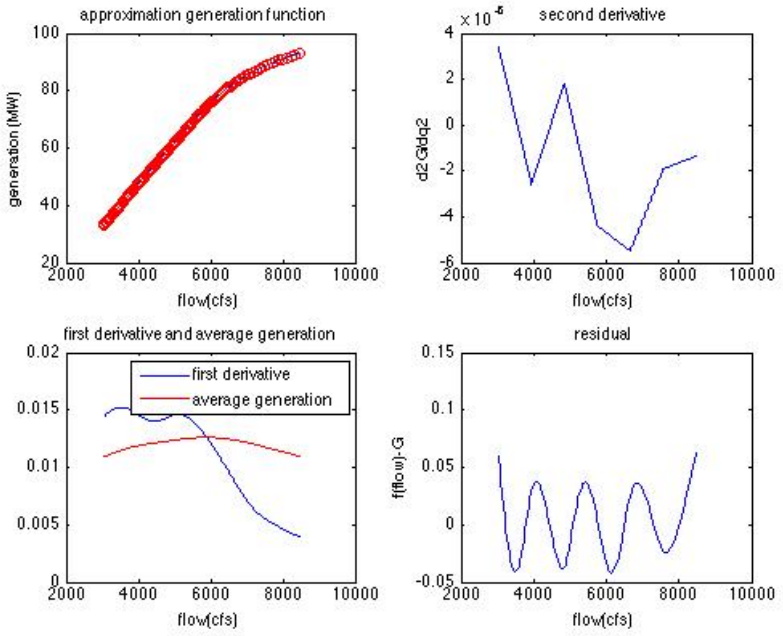


Figure 16 Chief Joseph unit type 1 Head=170, $\bar{y} = 63$, SE=0.0280, R^2=1 --Upper left: the original generation versus flow G(q) shown as circles and the fitted least-square spline with specified end condition. Upper right: the second derivative of spline generation function G"(q). Lower left: the first derivative G'(q) and average generation G(q)/q of the spline. Lower right: the error for the fitted function

A Concern is that we can see that in Figure 16 the second derivative for Chief Joseph has too many fluctuations. As described in section 2.2.3.3, we used the first three

points to calculate second derivative. But in this case, too many fluctuations may occur due to the specified second derivate at two ends. So instead, we use the first, the third, and the fifth or 1st , 4th , 7th ...points to calculate second derivative using the below equation:

$$f''(x) = \frac{2f(x_{i-1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})} + \frac{2f(x_i)}{(x_i-x_{i-1})(x_i-x_{i+1})} + \frac{2f(x_{i+1})}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)}$$

We assume that the last of these three points is the kth point, so the three points are 1st , (k+1)/2th, kth respectively. Figure 17-19 are three examples of Chief Joseph turbine type 1 while k=3, 7, 21 respectively.

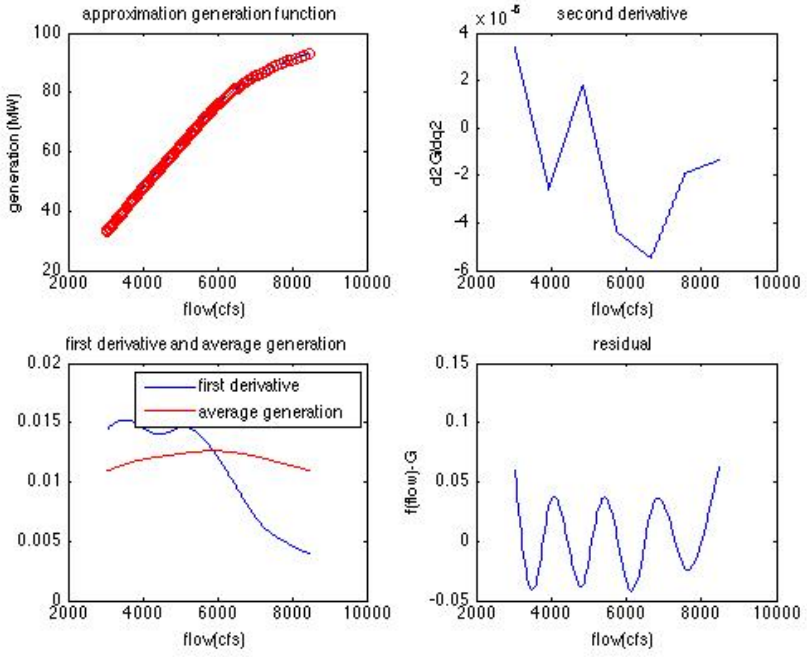


Figure 17 k=3 for Chief Joseph unit type 1 Head=170, $\bar{y} = 63$, SE=0.0280, R^2=1 --Upper left: the original generation versus flow G(q) shown as circles and the fitted least-square spline with specified end condition. Upper right: the second derivative of spline generation function G"(q). Lower left: the first derivative G'(q) and average generation G(q)/q of the spline. Lower right: the error for the fitted function

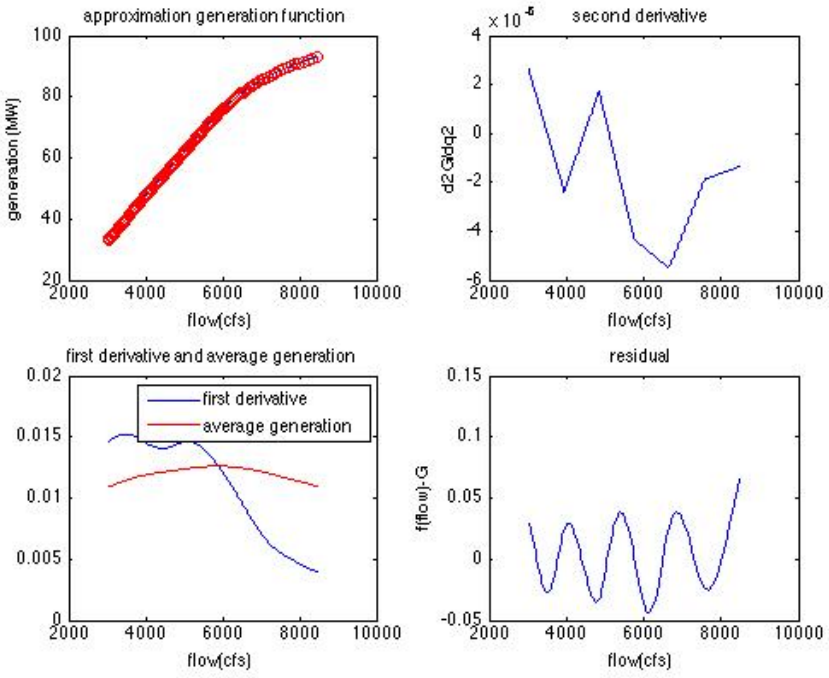


Figure 18 k=7 for Chief Joseph unit type 1 Head=170, $\bar{y} = 63$, SE=0.0253, R^2=1 --Upper left: the original generation versus flow G(q) shown as circles and the fitted least-square spline with specified end condition. Upper right: the second derivative of spline generation function G"(q). Lower left: the first derivative G'(q) and average generation G(q)/q of the spline. Lower right: the error for the fitted function

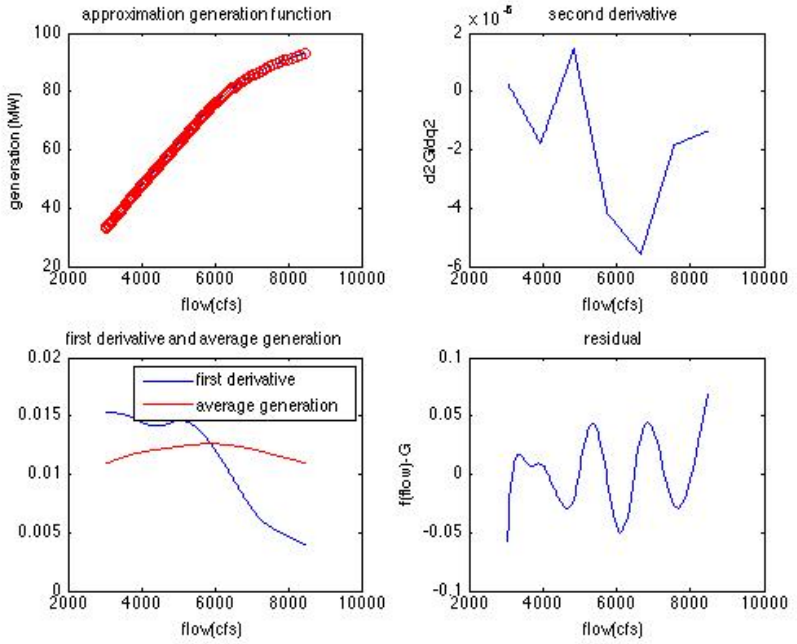


Figure 19 k=21 for Chief Joseph unit type 1 Head=170, $\bar{y} = 63$, SE=0.0265, R^2=1 --Upper left: the original generation versus flow G(q) shown as circles and the fitted least-square spline with specified end condition. Upper right: the second derivative of spline generation function G"(q). Lower left: the first derivative G'(q) and average generation G(q)/q of the spline. Lower right: the error for the fitted function

From Figure 17-19 we could see, it is disappointing that cubic spline fit doesn't improve much, the fluctuations in the second derivative are still there. This may due to noise in the data. To verify, we tried using 12 spline intervals to see clear the trend of the data. The result (figure 20) were very similar to those using 6 intervals which proves that the wiggle is in the original data. Fewer number of intervals is preferable to obtain smooth function. Thus to get a more consistent fit we tried 4 intervals, meaning the spline has 5 knots with 4 intervals (earlier examples had 6 intervals). This makes the spline somewhat smoother. See the results in figure 21. With 4 intervals we got a good fit that smoothed out small variations in the data. The problem is in part that the turbine generation curves are very close to linear with a second derivative close to zero. Thus a little noise in the original data results in a

second derivative function that fluctuates around zero. This would cause havoc with an optimization algorithm, and does not correspond to anything but a little noise around an almost straight line. By providing a smooth approximation, we improve the ability of our operations optimizer to find a reasonable solution and to provide stable results (not sensitive to small perturbations of the data or initial conditions).

We could see from figure 17, 20 and 21 that even though SE for using 4 intervals, 6 intervals and 12 intervals are 0.2226, 0.0280 and 0.0012, the R^2 for them are all very close to zero. It confirms us that by using 4 intervals, we only give up a little precision with a lot progress in second derivative.

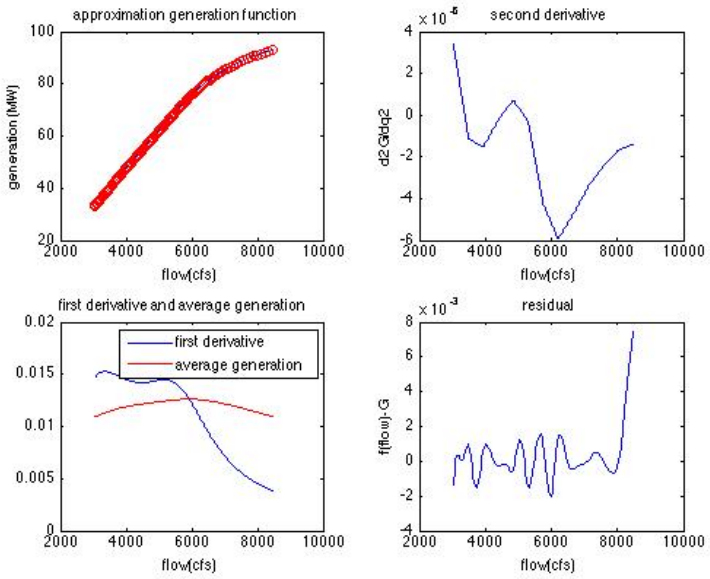


Figure 20 12 intervals cubic spline for Chief Joseph unit type 1 Head=170, $\bar{y} = 63$, SE=0.0012, R^2=1 -- Upper left: the original generation versus flow G(q) shown as circles and the fitted least-square spline with specified end condition. Upper right: the second derivative of spline generation function G"(q). Lower left: the first derivative G'(q) and average generation G(q)/q of the spline. Lower right: the error for the fitted function

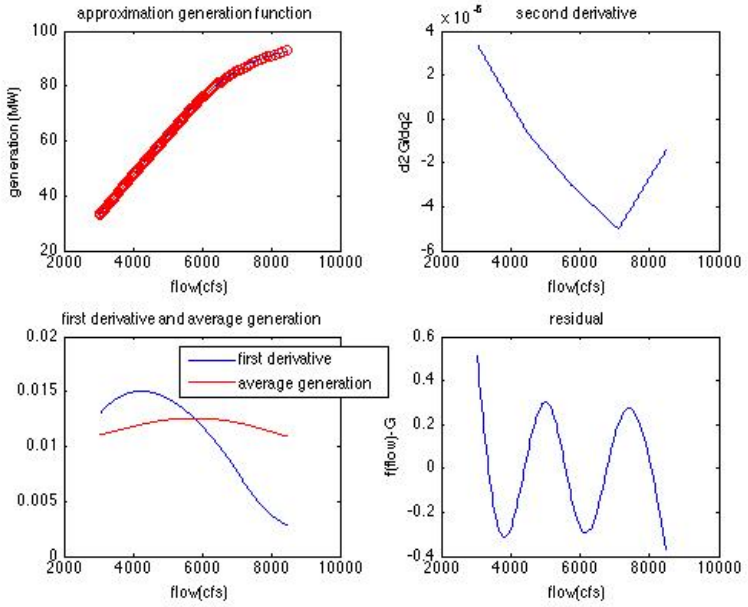


Figure 21 4 intervals cubic spline for Chief Joseph unit type 1 Head=170, $\bar{y} = 63$, SE=0.2226, R^2=0.9998--Upper left: the original generation versus flow G(q) shown as circles and the fitted least-square spline with specified end condition. Upper right: the second derivative of spline generation function G"(q). Lower left: the first derivative G'(q) and average generation G(q)/q of the spline. Lower right: the error for the fitted function

In this research, Least-squares spline functions with non-zero second derivatives at the end point will be fit to the turbine data for all turbine types at all ten federal dams in the Columbia-Snake system. In some cases larger intervals will be adopted to smooth the original generation data and to yield convex turbine generation curves over the relevant turbine operating range. Those spline curves will serve as the basis of powerhouse functions used in the Cornell SHOA short-term scheduling model. In particular they will provide Expected Average Operating Points for the turbine units, and the marginal generation functions needed to construct powerhouse functions.

We also recommend that BPA consider using these improved turbine powerhouse functions in other models of system operations.

Using the same method to determine the optimal numbers of intervals we should use for least-square cubic spline. The following two figures (Figure 23 and 24) are for Chief Joseph type 2 and type 3 and they use 6 and 4 intervals respectively.

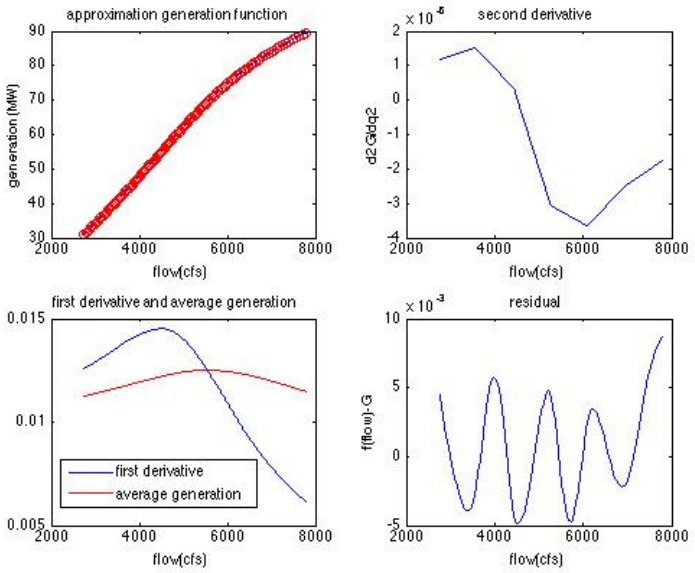


Figure 22 6 intervals cubic spline for Chief Joseph unit type 2 Head=170, $\bar{y} = 60.25$, SE=0.0035, R^2=0.9998--Upper left: the original generation versus flow G(q) shown as circles and the fitted least-square spline with specified end condition. Upper right: the second derivative of spline generation function G"(q). Lower left: the first derivative G'(q) and average generation G(q)/q of the spline. Lower right: the error for the fitted function

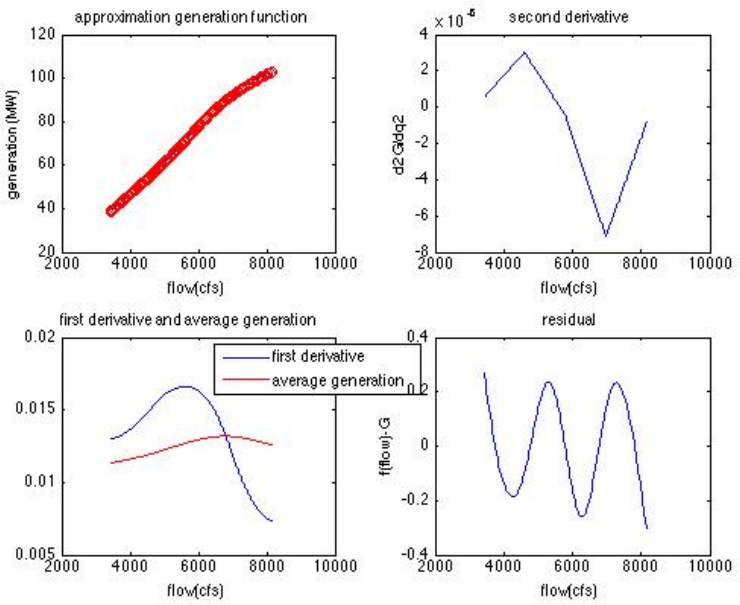


Figure 23 4 intervals cubic spline for Chief Joseph unit type 3 Head=170, $\bar{y} = 70.75$, SE=0.1647, R^2=0.9999--Upper left: the original generation versus flow G(q) shown as circles and the fitted least-square spline with specified end condition. Upper right: the second derivative of spline generation function G"(q). Lower left: the first derivative G'(q) and average generation G(q)/q of the spline. Lower right: the error for the fitted function

3.2.3 Lower Monumental STS

Lower Monumental has 2 different turbine types. The following table summaries the unit characteristics of different turbines.

Table 6 Lower Monumental STS turbine generation characteristics

	Unit 1-3 (TYPE 1)	Unit 4-6 (TYPE 2)
Maximum Generation (MW)	160.5	160.5
Minimum Generation (MW)	55.5	63.5
Maximum Flow (cfs)	25191	25686
Minimum Flow (cfs)	9604	11528

To approximation the generation function for Lower Monumental STS, 6 equally spaced intervals were used and specified second derivative was calculated using first 3 and last 3 points at two ends for the least square cubic spline. The results for type 1 are shown in Figure 24 (take head=95ft for example).

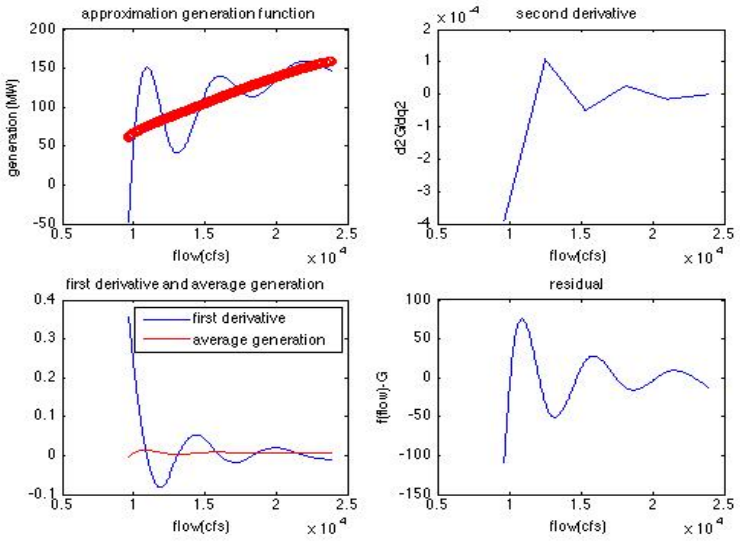


Figure 24 6 intervals cubic spline for Lower Monumental STS unit type 1 Head=95, $\bar{y} = 110.25$, SE=34.247, R^2=0.4194--Upper left: the original generation versus flow G(q) shown as circles and the fitted least-square spline with specified end condition. Upper right: the second derivative of spline generation function G"(q). Lower left: the first derivative G'(q) and average generation G(q)/q of the spline. Lower right: the error for the fitted function

In figure 24, the approximation generation function of Lower Monumental STS head=95ft type 1 has many wiggles. The wiggles were unlikely to exist when using least square cubic spline method. As discussed in Section 3.5, the second step in least-square cubic spline with specified second derivative is to construct specialized spline function for first and last intervals. The wiggles in the generation plot of figure 4-13 is most likely due to the constructed cubic function at two. To confirm this assumption, I plot the original generation data minus the constructed

generation function which is shown in Figure 25. The below table 7 shows the parameters of the constructed function:

$$G = a\times(q - q_0)^3 + b\times(q - q_0)^2 + c\times(q - q_0) + d$$

Because there is no constrain on the value of the first point when constructing the cubic function, it turned out to be -1883 MW of the first point which is much bigger than the average generation 100MW.

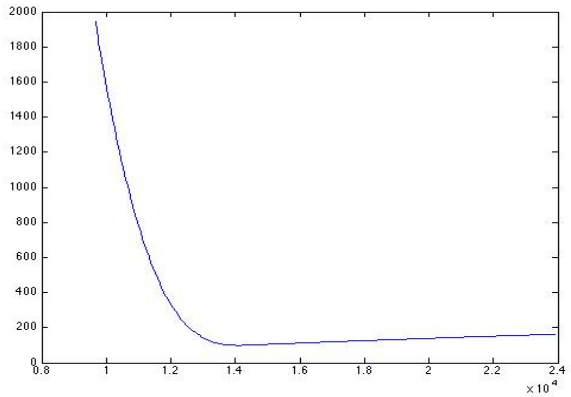


Figure 25 the original generation data minus the constructed generation function for the first and last intervals of Lower Monumental STS head=95 ft unit type 1

Table 7 the parameters of the constructed generation function for the first interval

a	b	c	d
$1.732*10^{-8}$	$-2.48*10^{-4}$	1.1836	-1883

Since this situation only happens for a few times in all the reservoirs, I guess that it is due to the data itself. In the below figure, I plot the original data we get from BPA. It shows clearly that the trend of the first few points has big difference with that of the rest points. To better visualize the trend of the data, I plot only the first 50 points.

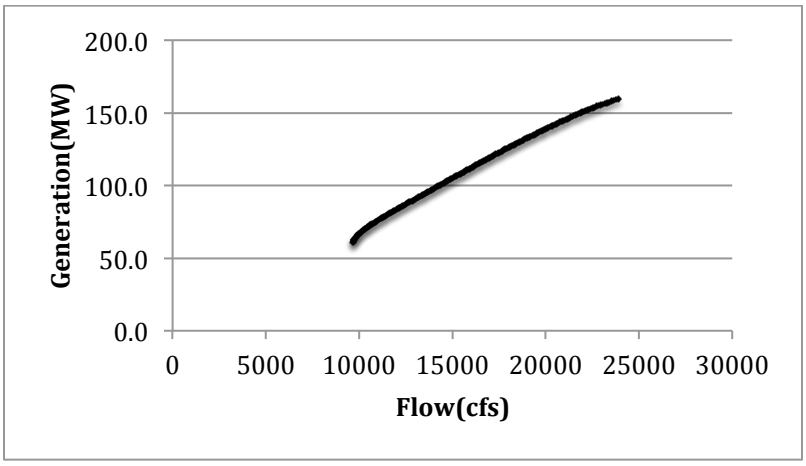


Figure 26 Original Generation-Flow data of Lower Monumental STS head= 95 ft unit type 1

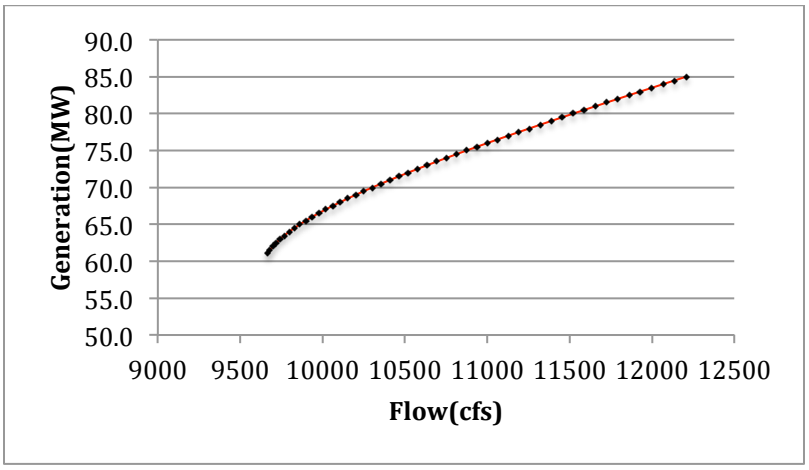


Figure 27 Original Generation-Flow data of Lower Monumental STS head= 95 ft unit type 1(first 50 points)

When I construct the cubic function for the two ends, I use the first 3 points which cannot represent the rest points. So I try to delete the first 15 points and run the program again. The following figure shows the new result using 6 intervals. Compare figure 24 with figure 28, we could see the second derivative of the first point change from $-4*10^{-4}$ to $-10*10^{-5}$, only ¼ of the former value. Also the approximation function is much better. Even though in the first interval, the first derivative has a wired wiggle which causes the second derivative changes from negative to positive, the max second derivative is much less than 10^{-5} which is

negligible. So the approximation generation function for Lower Monumental STS type 1 could be seen as a concave shape.

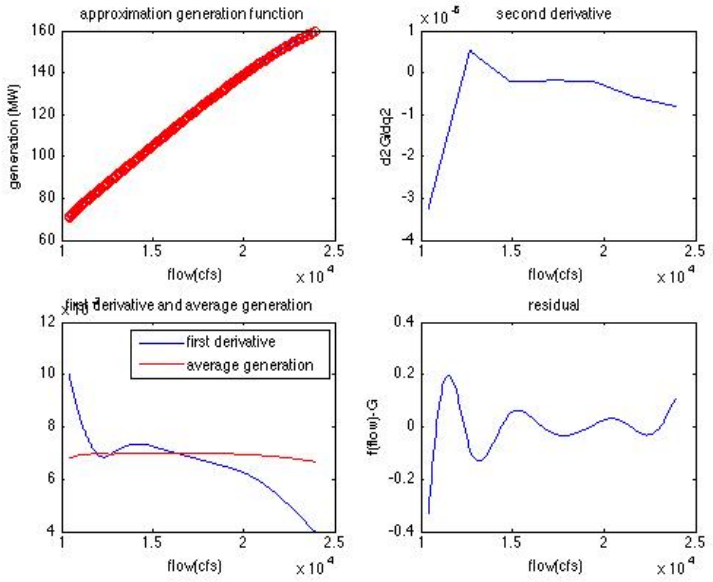


Figure 28 Lower Monumental STS unit type 1 Head=95, $\bar{y} = 115.25$, SE=0.0806, R^2=0.9999 (delete first 15 points)--Upper left: the original generation versus flow G(q) shown as circles and the fitted least-square spline with specified end condition. Upper right: the second derivative of spline generation function G"(q). Lower left: the first derivative G'(q) and average generation G(q)/q of the spline. Lower right: the error for the fitted function

The figure 29 is for Lower Monumental STS type 2 with 3 intervals to get rid of the noise in the original data.

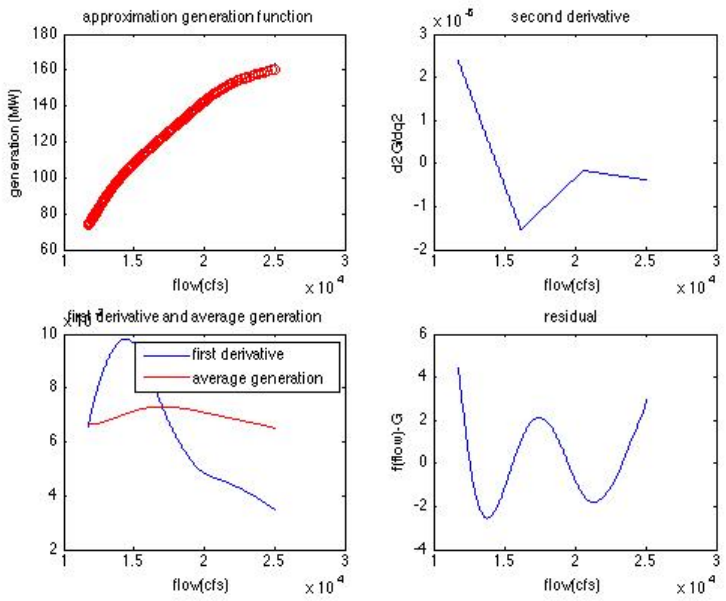


Figure 29 Lower Monumental STS unit type 2 Head=95, $\bar{y} = 117$, SE=1.7427, R^2=0.9952 (S-shape)--Upper left: the original generation versus flow G(q) shown as circles and the fitted least-square spline with specified end condition. Upper right: the second derivative of spline generation function G"(q). Lower left: the first derivative G'(q) and average generation G(q)/q of the spline. Lower right: the error for the fitted function

3.2.4 Bonneville STS

Bonneville STS has 3 different turbine types. The following table summaries the unit characteristics of different turbines.

Table 8 Bonneville STS turbine generation characteristics

	Unit 9 (TYPE 1)	Unit 1-8,10 (TYPE 2)	Unit 11-18 (TYPE 3)
Maximum Generation (MW)	62.5	63	80
Minimum Generation (MW)	8	13.5	24
Maximum Flow (cfs)	17392	13814	22893
Minimum Flow (cfs)	3056	5011	10335

To approximation the generation function for Bonneville STS, 4 equally spaced intervals were used and specified second derivative was calculated using first 3 and last 3 points at two ends for the least square cubic spline. The results for 3 different types are shown in Figure 30, 31 and 32 (take head=50ft for example).

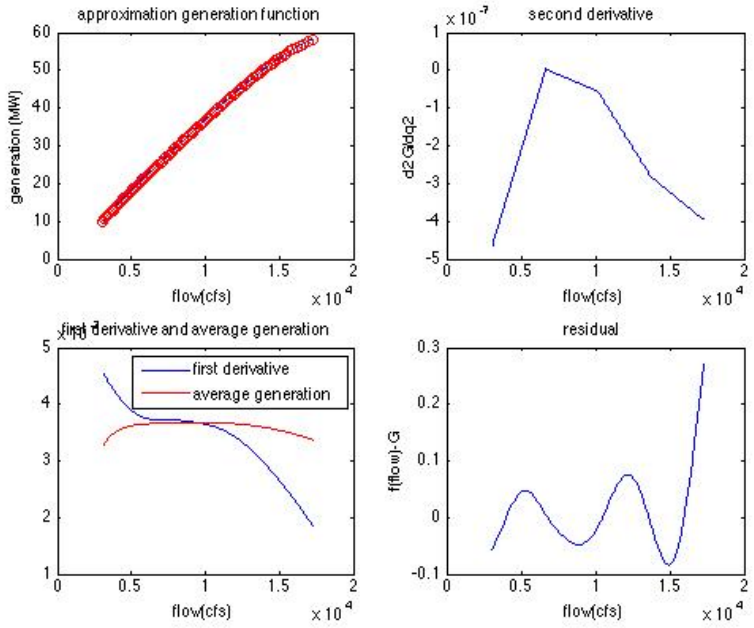


Figure 30 Bonneville STS unit type 1 Head=50, $\bar{y} = 34$, SE=0.0565, R^2=0.9999 (Concave)--Upper left: the original generation versus flow G(q) shown as circles and the fitted least-square spline with specified end condition. Upper right: the second derivative of spline generation function G"(q). Lower left: the first derivative G'(q) and average generation G(q)/q of the spline. Lower right: the error for the fitted function

For Bonneville STS type 2 and 3, the slope of the first few points is quite different from the rest points, just like the case for Lower Monumental STS described in Section 3.2.3. To approximate the generation function of these two types, first 8 points were deleted just like the way dealing with Lower Monumental STS. The following figure 31 and 32 are for type 2 and 3 respectively.

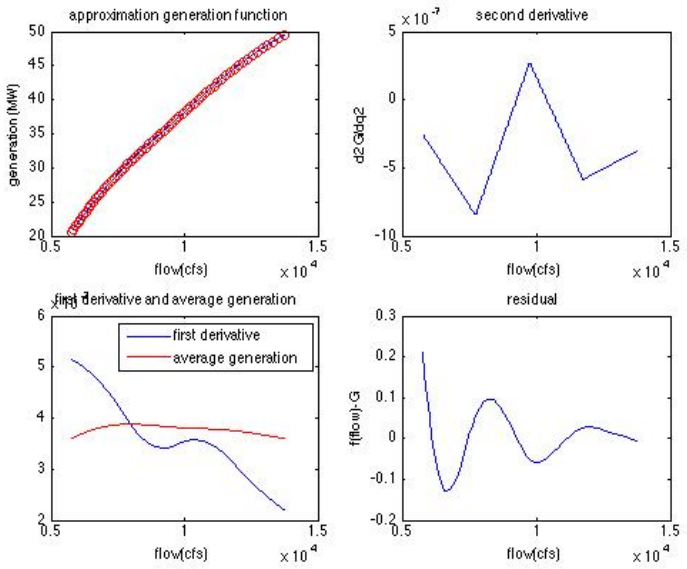


Figure 31 Bonneville STS unit type 2 Head=50, $\bar{y} = 35$, SE=0.0690, R^2=0.9999 (Concave)(delete first 8 points)--Upper left: the original generation versus flow G(q) shown as circles and the fitted least-square spline with specified end condition. Upper right: the second derivative of spline generation function G"(q). Lower left: the first derivative G'(q) and average generation G(q)/q of the spline. Lower right: the error for the fitted function

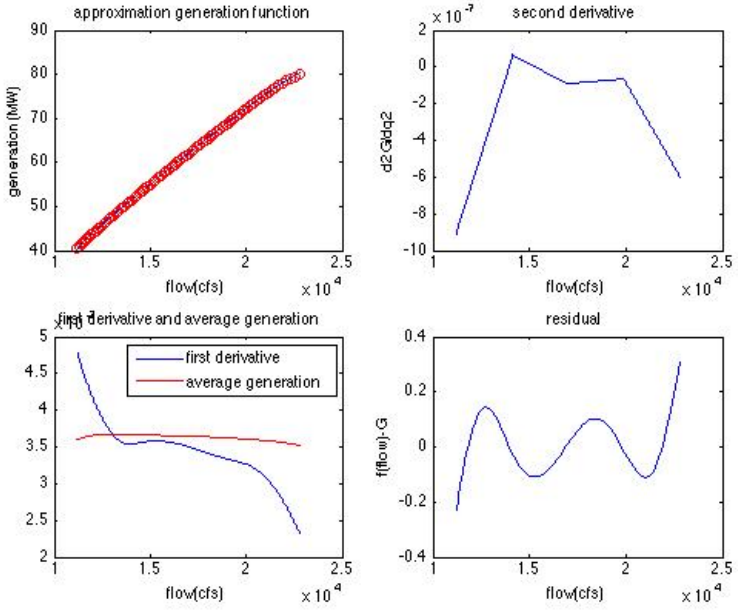


Figure 32 Bonneville STS unit type 3 Head=50, $\bar{y} = 60.25$, SE=0.0973, R^2=0.9999 (Concave)(delete first 8 points)--Upper left: the original generation versus flow G(q) shown as circles and the fitted least-square spline with specified end condition. Upper right: the second derivative of spline

generation function G"(q). Lower left: the first derivative G'(q) and average generation G(q)/q of the spline. Lower right: the error for the fitted function

Notice that, for these 3 types, a range of flow for each of them with the second derivative close to 0 exists. Also the second derivative of the whole flow range is no bigger than 10 to the minus 6. It means that the slope is almost constant in the whole range. It also shows in the average generation plot for each type, it is almost a straight line, which means the efficiency doesn't change a lot in the whole range. It adds difficulty for us to find EAOP and the same time makes the result less accurate.

3.3 Comparison between cubic splines and quadratic splines

Take Chief Joseph type 1 head = 170ft for example and set the interval numbers to be six. Figure 33 and 34 are the results of least square curve fitting using cubic and quadratic splines respectively.

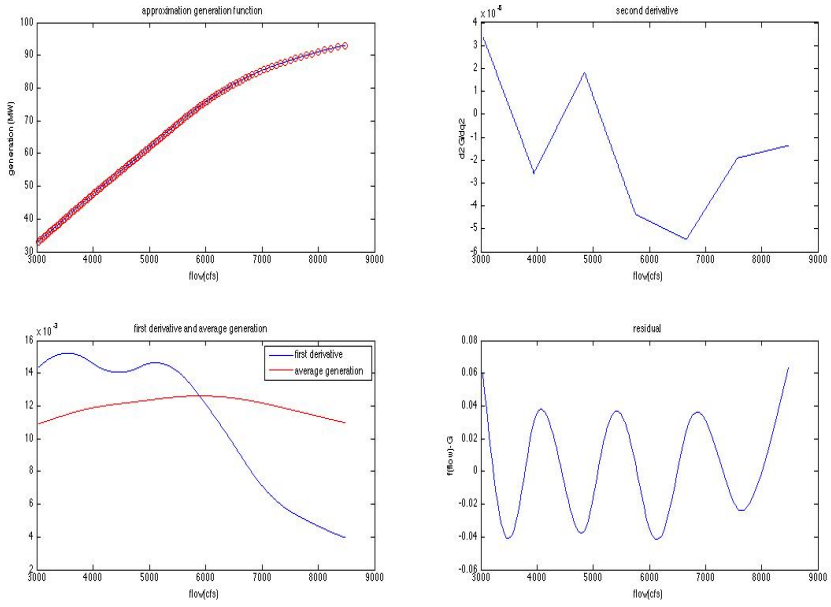


Figure 33 Cubic spline approximation for Chief Joseph unit type 1 head = 170ft. Upper left: approximation function and original data points. Upper right: second derivative of approximation

generation function. Lower left: the first derivative and average generation. Lower right: the error for the fitted function. Lower right: the error for the fitted function.

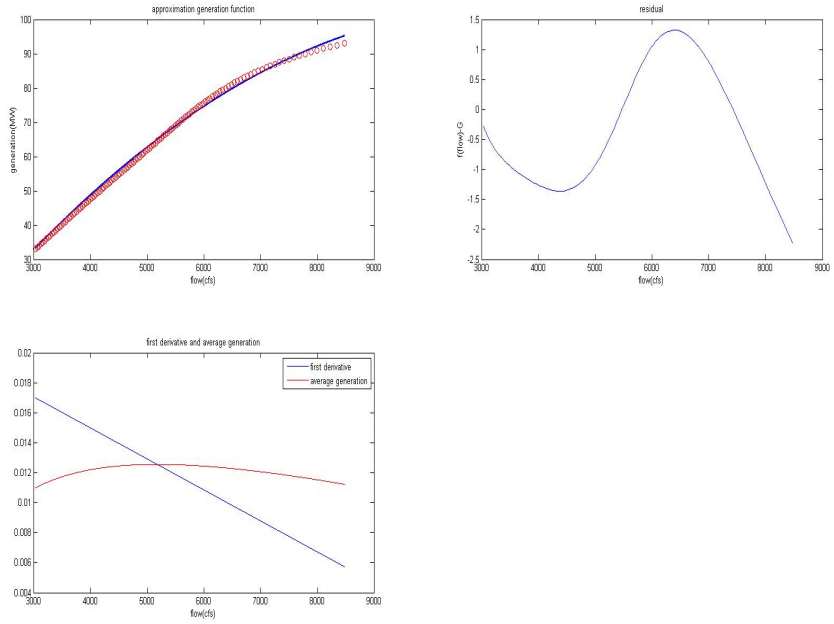


Figure 34 Quadratic spline approximation for Chief Joseph unit type 1 head = 170ft. Upper left: approximation function and original data points. Upper right: the error for the fitted function. Lower: the first derivative and average generation.

As we could see, the fitted function using quadratic spline goes away from the original data at the end part, also the magnitude for residual using quadratic spline is much larger than that using cubic spline, so it is clear that the error for cubic spline is smaller than that of quadratic spline, in this aspect we could conclude that cubic spline is a 'better' fit for the data. However, as the data itself has some strange wiggles which accounts for the negative to positive jump in the second derivative of the cubic spline approximation function. That is to say that cubic spline takes into consideration too much unimportant information in the given raw data. On the contrary, the first derivative of the quadratic spline approximation function decreases from 0.017 to 0.006 without unnecessary fluctuations. In this aspect, we

could conclude that quadratic spline is a 'better' fit which seizes the principal trend and identifies its primary and secondary aspect.

Also, for Chief Joseph type 1, head from 155 ft to 185 ft, the efficient average operating point plot as a function of head using both methods are shown in Figure 35 and 36.

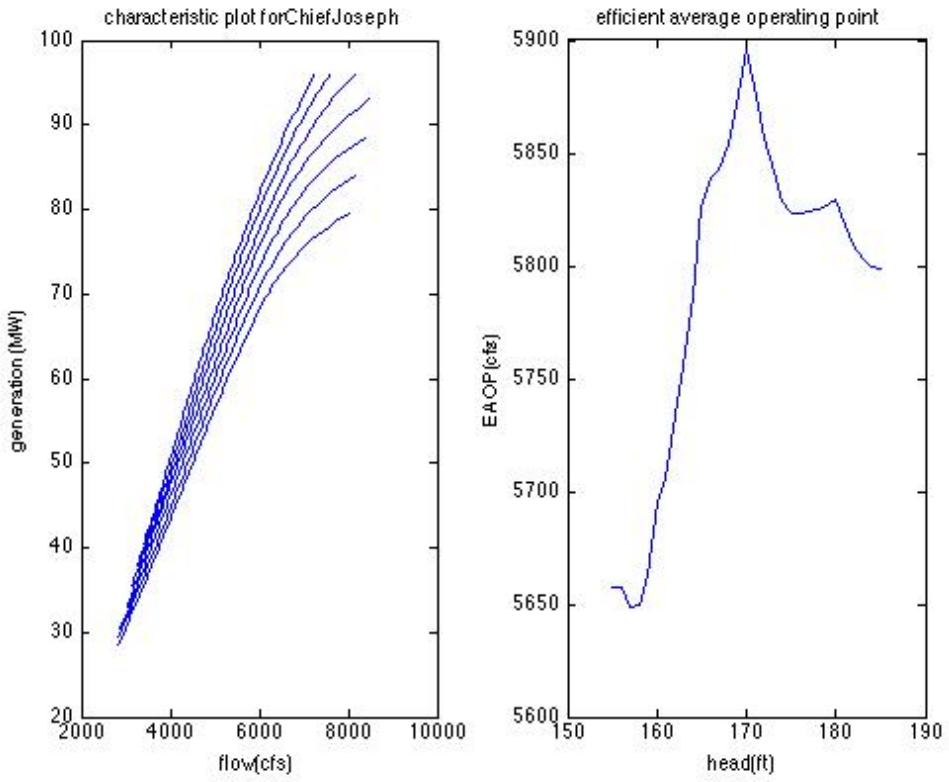


Figure 35 Generation as a function of flow for head from 155 ft to 185 ft and EAOP plot using cubic spline approximation for Chief Joseph unit type 1.

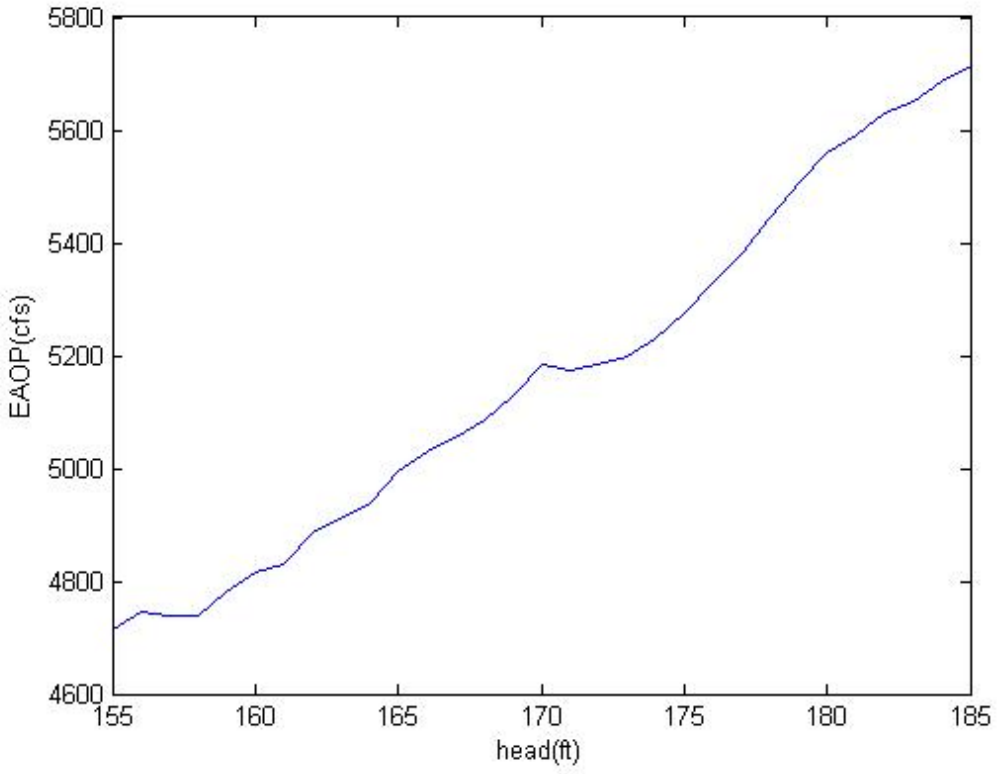


Figure 36 Efficient average operating point plot using quadratic spline approximation for Chief Joseph unit type 1.

As we could see, EAOP plot is another way to prove that quadratic spline approximation is much more helpful in getting rid of useless wiggle.

In order to see clearly the trend of the original data, empirical first derivative, average generation and EAOP are calculated and shown in the following figure. When comparing figure 37 and 38 with figure 33 to 36, it is surprising to see that figure 37 is very similar with the upper left part in figure 33 and figure 38 is very similar with the right part in figure 35. On the whole, we could conclude that cubic spline is much better.

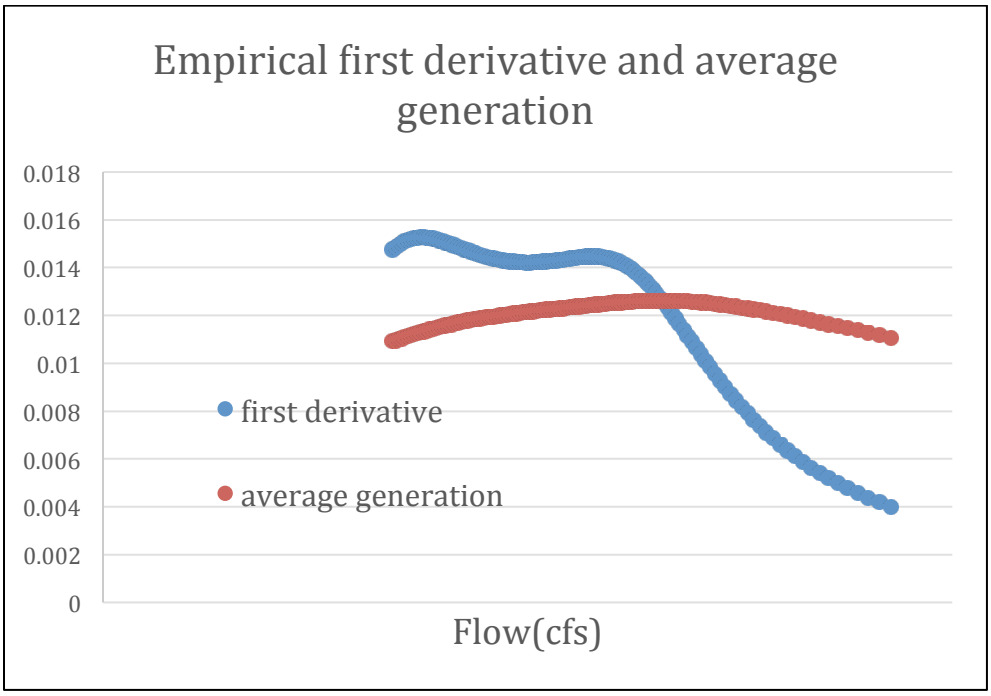


Figure 37 Empirical first derivative and average generation for Chief Joseph unit type 1 head = 170ft.

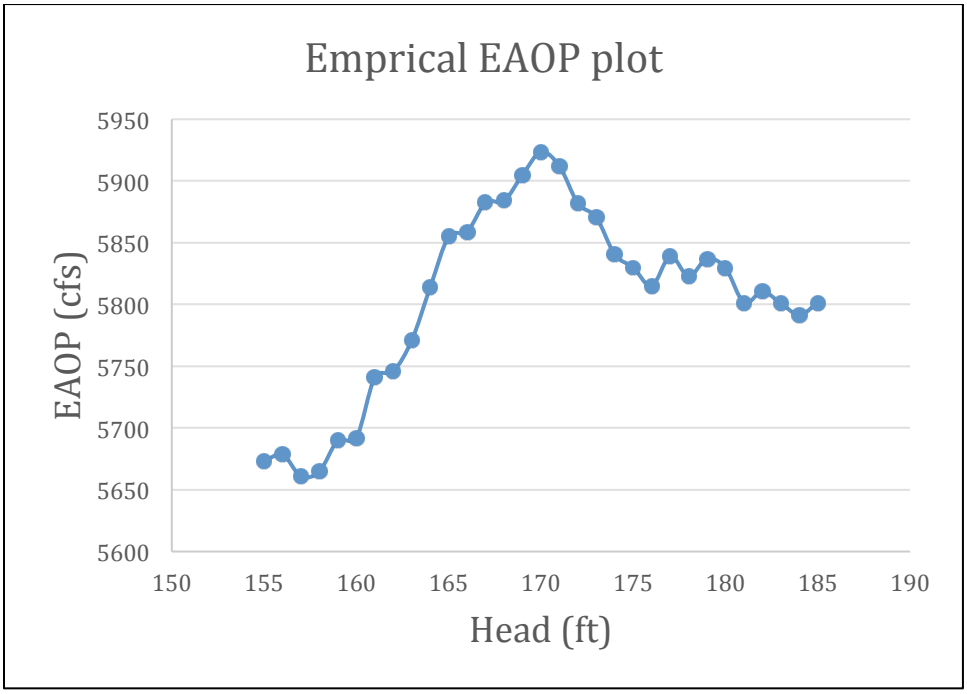


Figure 38 Empirical EAOP plot for Chief Joseph unit type 1.

3.4 Powerhouse generation function

3.4.1 Economic Dispatch Algorithm Using Polynomial Generation Function

Generation function:

Assume the generation function for a unit is of S shape, which is:

$$G(q) = a \times q^2 \times (q_0 - q)$$

We also assume there are 3 different turbine type and each type has 4 different units, also the efficiency at EAOP is ordered from big to small, 1>2>3.

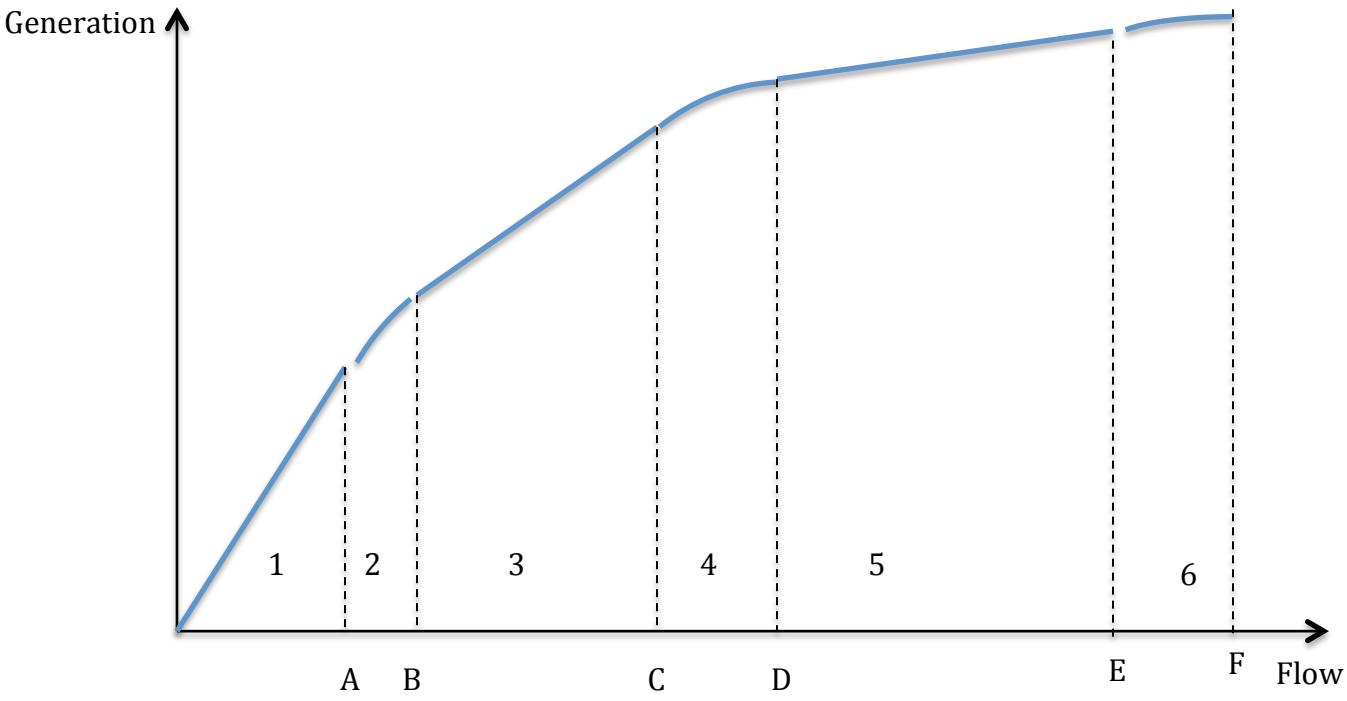


Figure 39 powerhouse curve example: number in the figure refer to part, character refer to total flow of certain point

Powerhouse function:

As for the case with three different turbine types, the powerhouse function will have 6 different parts, three constant slope parts with two transition parts between the first two and the last two and a final part (Figure 37).

In part 1 and 2, only type 1 units are operating, but only in part 1 they are operating at its $EAOP_1$, so A is 4 times $EAOP_1$.

Part 1: when the flow is less than $EAOP_1$, only one unit of type 1 is operating a fraction of time at its $EAOP_1$. When the flow is between $EAOP_1$ and 2 times $EAOP_1$, one unit of type 1 is operating full time and another unit of type 1 is operating a fraction of time both at their $EAOP_1$…Things go on till 4 units operate full time at their $EAOP_1$.

Part 2: this is the transition part from type 1 to type 2. In this part, type 1 will operate over its $EAOP_1$, till ${q_1}^{[2]}$, which is

$$G_1'\left({q_1}^{[2]}\right) = \frac{G(EAOP_2)}{EAOP_2}$$

as shown in the below figure. So all the units of type 1 will operate full time with flow increasing from $EAOP_1$ to ${q_1}^{[2]}$. And the powerhouse curve will follow the generation function of turbine type 1.

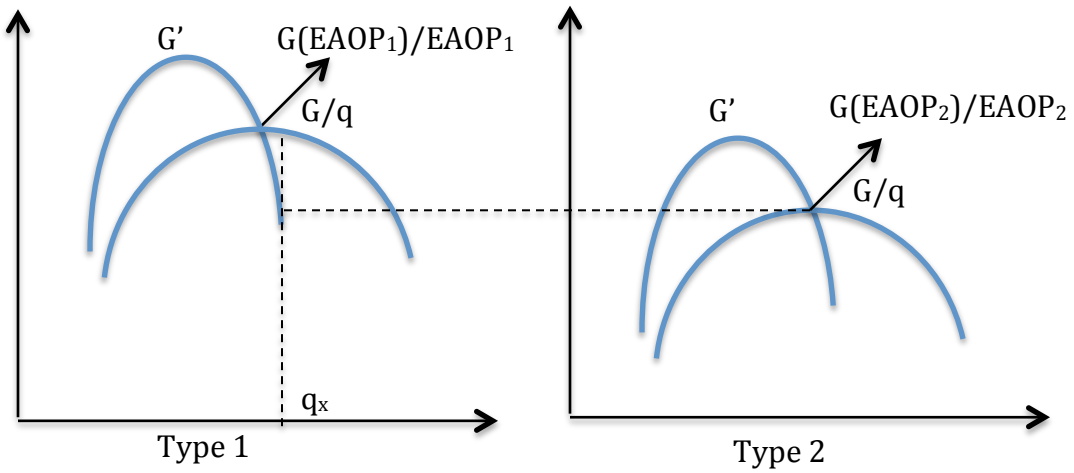


Figure 40 first derivative and average generation in one plot for type 1 and 2

Part 3: all the units of type 1 will operate full time at $q_1^{[2]}$. When the flow is less than k_1 times $q_1^{[2]}$ plus $EAOP_2$, only one unit of type 2 is operating a fraction of time at its $EAOP_2$. When the flow is between k_1 times $q_1^{[2]}$ plus $EAOP_2$ and k_1 times $q_1^{[2]}$ plus 2 times $EAOP_2$, one unit of type 2 is operating full time and another unit of type 2 is operating a fraction of time both at their $EAOP_2$…Things go on till n_2 units of type 2 operate full time at their $EAOP_2$.

Part 4: this is the transition part from type 2 to type 3. In this part, both type 1 and type 2 will operate over its EAOP, till $q_1^{[3]}$ and $q_2^{[3]}$, respectively, which is

$$G_1'(q_1^{[3]}) = G_2'(q_2^{[3]}) = \frac{G(EAOP_3)}{EAOP_3}$$

as shown in the below figure. So all the units of type 1 and 2 will operate full time with flow increasing from $q_1^{[2]}$ to $q_1^{[3]}$, $EAOP_2$ to $q_2^{[3]}$. And the powerhouse curve will follow:

$$Max: 4\times G_1(q_1) + 4\times G_2(q_2)$$

$$s.t.\ 4\times q_1 + 4\times q_2 = current\ flow$$

here, q_1 is the flow through one unit of type 1 and q_2 is the flow through one unit of type 2.

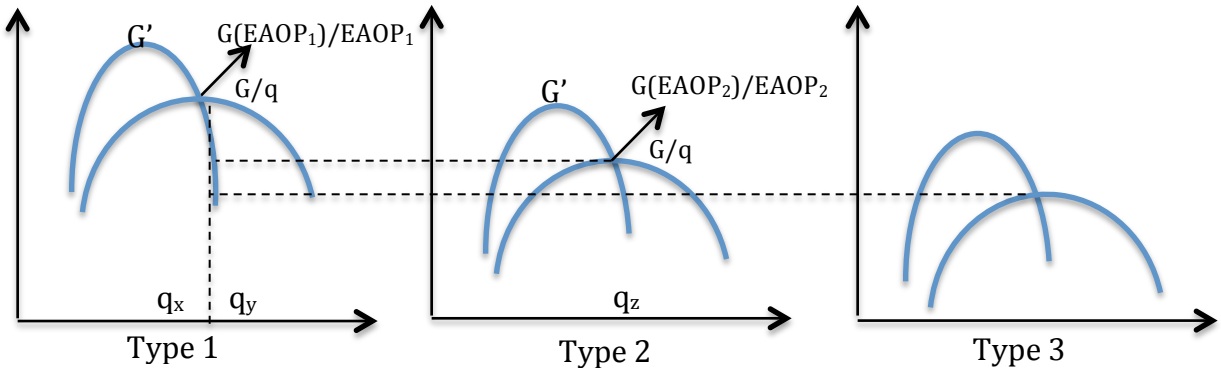


Figure 41 first derivative and average generation in one plot for type 1, 2 and 3

Part 5: all the units of type 1 and 2 will operate full time at q_2 and q_3, respectively. When the flow is less than

$$4\times {q_1}^{[3]} + 4\times {q_2}^{[3]} + EAOP_3$$

only one unit of type 3 is operating a fraction of time at its $EAOP_3$. When the flow is between

$$4\times {q_1}^{[3]} + 4\times {q_2}^{[3]} + EAOP_3 \ \text{ and } \ 4\times {q_1}^{[3]} + 4\times {q_2}^{[3]} + 2\times EAOP_3$$

one unit of type 3 is operating full time and another unit of type 3 is operating a fraction of time both at their $EAOP_3$…Things go on till 4 units of type 3 operate full time at their $EAOP_3$.

Part 6: all the units of the three types will operate over its EAOP till all of them reach their capacity.

Calculation:

The following table summarizes EAOP for each turbine type:

Table 9 summary of EAOP for each turbine type

	Type 1(4)	Type 2(4)	Type 3(4)
EAOP(cfs)	13000	9000	7000
Max flow(cfs)	17000	12000	10000

We assumes their generation function are:

$$G_1(q) = 0.03 \times q^2 \times (26000 - q)/10^9$$

$$G_2(q) = 0.05 \times q^2 \times (18000 - q)/10^9$$

$$G_3(q) = 0.07 \times q^2 \times (14000 - q)/10^9$$

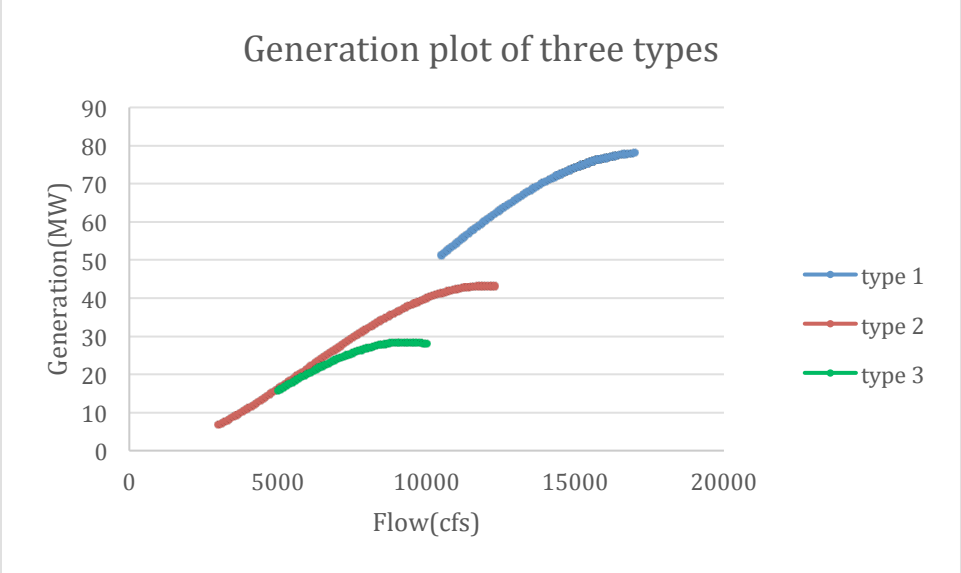


Figure 42 generation plot for the three different types

If only Type 1 is operating

Generation for one unit = $0.03\times13000^2\times(26000-13000)/10^9$= 65.91 MW.

So, a =$(4\times13000, 4\times65.91)$ should be on power function curve.

The Switch Point (SP) of unit 1 to unit 2 should be:

$$G_1' = \frac{G_2}{q}$$

$$\frac{2\times0.03\times q\times26000-3\times0.03\times q^2}{10^9} = (0.05\times18000^2)/(4\times10^9)$$

Using excel Solver, get answer for the above equation, $q_1{}^{[2]}$ = 14154 cfs.

The corresponding power should be:

$0.03\times14154^2\times(26000-14154)/10^9$= 71.20 MW.

So, b =$(4\times14154, 4\times71.20)$ should be on power function curve.

Between a and b, the curve follows the Generation function

$$\mathrm{G} = 4\times\mathrm{G}_1 = 4\times0.03\times\mathrm{q}^2\times(26000-\mathrm{q})/10^9$$

In which, q equal to current flow divided by 4.

After Type 2 is brought on line

Generation for one unit = $0.05\times9000^2\times(18000-9000)/10^9$= 36.45 MW.

So, c = $(x_b + 4\times9000, y_b + 4\times36.45)$ should be on power function curve.

The SP of unit 2 to unit 3 should be:

$$G_2' = G_1' = \frac{G_3}{q}$$

$$\frac{2\times0.05\times q_2\times18000 - 3\times0.05\times q_2^2}{10^9} = \frac{2\times0.03\times q_1\times26000 - 3\times0.03\times q_1^2}{10^9}$$

$$= (0.07\times14000^2)/(4\times10^9)$$

Using excel Solver, get answer for the above equation, which is q_2[3] = 9624 cfs, q_1[3]=14749 cfs.

The corresponding power should be:

$0.03\times14749^2\times(26000 - 14749)/10^9$= 73.42 MW.

$0.05\times9624^2\times(18000 - 9624)/10^9$ = 38.79 MW.

So, d = $(4\times14749 + 4\times9624, 4\times73.42 + 4\times38.79)$ should be on power function curve.

Between c and d, the curve maximizes the Generation function

$$Max:\ G = 4\times G_2 + 4\times G_1$$

$$= 4\times0.05\times q_2^2\times\frac{18000 - q_2}{10^9} + 0.03\times q_1^2\times(26000 - q_1)/10^9$$

in which, $4\times q_2 + 4\times q_1$ equal to the current flow.

After Type 3 is brought on line

Generation for one unit = $0.07\times7000^2\times(14000 - 7000)/10^9$= 24.01 MW.

So, e = $(x_d + 4\times7000, y_d + 4\times24.01)$ should be on power function curve.

So, the power function curve should look like that in figure 41:

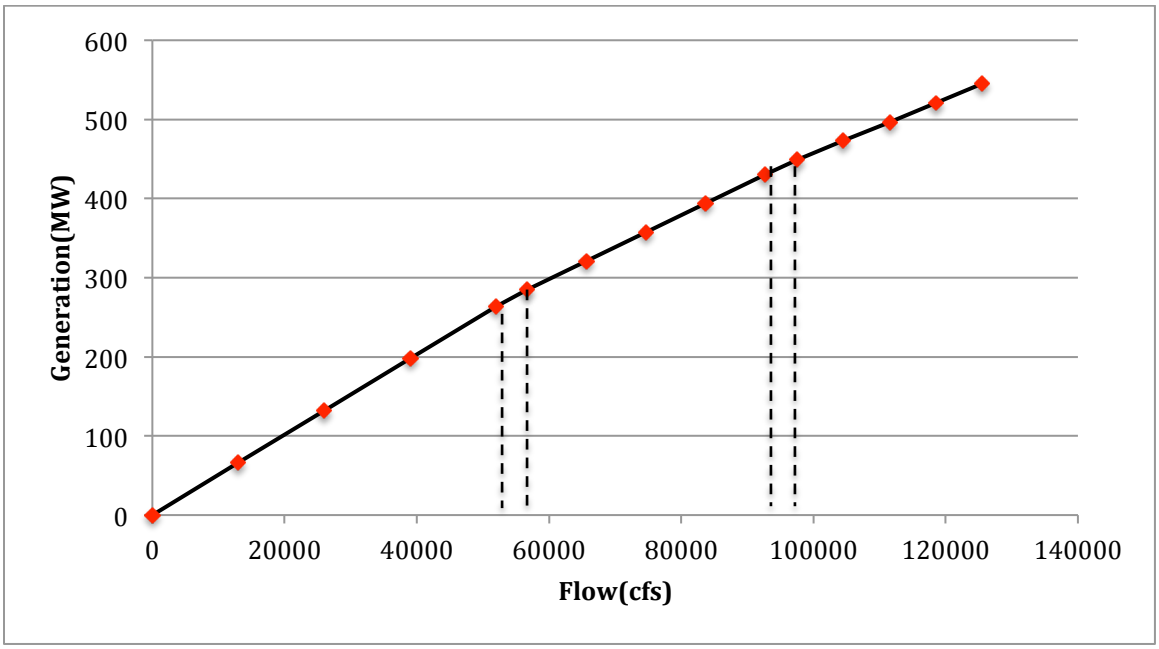


Figure 43 powerhouse function of this simple numeric example

The transitions are very short. For practical purposes, we could almost use linear line connectors and solve the short-term scheduling problem with an Linear Programming.

3.4.2 Powerhouse Generation Functions for Grand Coulee and Chief Joseph

In this section, two examples using economic dispatch algorithm are displayed.

For Grand Coulee, there are 4 different turbine types, 2 big ones and 2 small ones. The following table summarizes EAOP and efficiency for each type (take 250ft for examples).

Table 10 EAOP and efficiency at EAOP of each type for Grand Coulee head=250 ft (the number in the bracket is unit number)

	Type 1(9)	Type 2(9)	Type 3(3)	Type 4(3)
EAOP(cfs)	3585.23	3491.95	25954.93	27700.89
efficiency	0.9094	0.9335	0.9191	0.9076

Since there are 4 turbine types, it should have 8 parts in the powerhouse function plot as shown in Figure 42. The 8 parts contain 4 straight parts and 3 transition parts and a final part. The slope of each straight part should be proportion to related efficiency. As shown in the table, the turbine was ordered according to efficiency, so type 2 should be operated first at its EAOP (first part in the figure). After the flow reaches EAOP times unit number of type 2, type 2 units will operate full time with flow over its EAOP and the powerhouse curve will follow the generation function of turbine type 2(second part in the firgure). Then type 3 units begin to operate since the efficiency of type 3 is the second highest. It first operates at its EAOP which is third part in the figure then the flow through both type 2 and type 3 will increase before the next turbine type is brought on line(the second transition part). Type 1 goes after type 3 then is type 4. After all the turbine types are operated, the flow will increase to the sum of capacity of all the units(final part in the figure).

We could see that the second and fourth straight parts are longer than the other two, it is because these two parts are for two bigger unit types. Also the three transition parts are all very small, it is due to the small difference of efficiency between every two adjacent types, never up to 1.5%.

This powerhouse function is smooth and the first derivative of it is non-increasing. It is convenience using this optimization function to do further calculation.

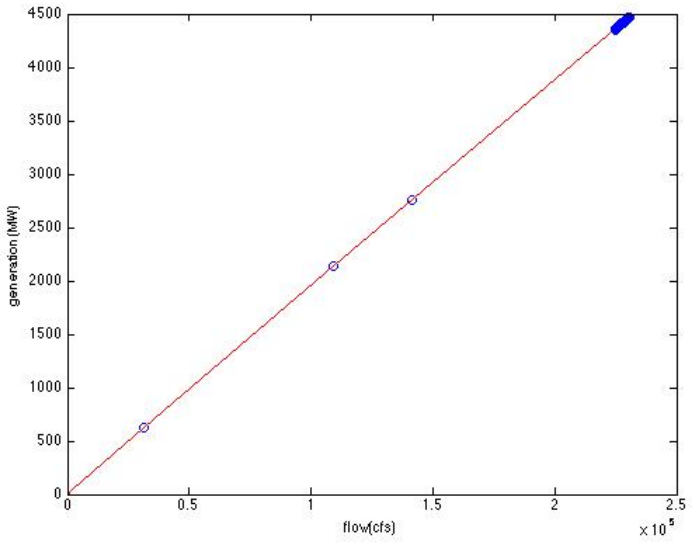


Figure 44 Powerhouse function of Grand Coulee Head= 250 ft (Blue 'o' is the transfer parts and red line is the straight part)

For Chief Joseph, there are 3 different turbine types and all of then do not have too much difference in their sizes. The following table summarizes EAOP and efficiency for each type. (take 170ft for examples).

Table 11 EAOP and efficiency at EAOP of each type for Chief Joseph head=170 ft (the number in the bracket is unit number)

	Type 1(6)	Type 2(10)	Type 3(11)
EAOP(cfs)	5895.90	5544.35	6680.16
efficiency	0.8764	0.8692	0.9170

Same as Grand Coulee, the powerhouse function for Chief Joseph also has straight parts, transition parts and a final part. The obvious difference between these two

powerhouse functions is that the transition part for Chief Joseph is much longer. It is because that the difference of efficiency between type 1 and type 3 is 4%, much bigger than 1.5% for Grand Coulee. So in order to make the first derivative of powerhouse function smooth, when switching turbine type from one to another, the first derivative of the generation function of the first type should drop to that of the second type at its EAOP. 4% is a relatively long way to go compared to 1.5%.

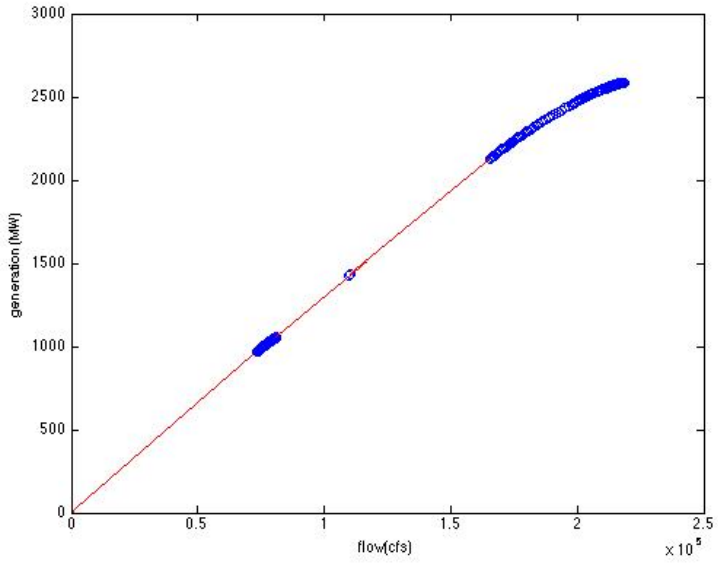


Figure 45 Powerhouse function of Chief Joseph Head= 170 ft (Blue 'o' is the transfer parts and red line is the straight part)

3.4.3 Powerhouse Generation Functions with Fish Dispatch

For Bonneville, most of the time, the turbines are operating according to Table BON-14 PH2(Powerhouse2)-Priority (give PH-2 with units 11-18 priority; the PH-1 with units 1-10) so the order is[Fish passage plan, 2012]:

11, 18, 12, 17, 13, 14, 15, 16, 1, 10, 3, 6, 2, 4, 5, 8, 7, 9.

However, when adult and jack salmonid counts equal or exceed 30,000 fish per day before August 31 or when adult and jack salmonid counts equal or exceed 25,000 fish per day after August 31, then the operating order is (Operate first 1 & 10 from PH-1, then use PH-2 units, and finally use the remaining PH-1 units):

1, 10, 11, 18, 12, 17, 13, 14, 15, 16, 3, 6, 2, 4, 5, 8, 7, 9.

Also, if there is special instruction about PH1 priority (use first PH-1, and then PH-2), so the operating order is:

1, 10, 3, 6, 2, 4, 5, 8, 7, 9, 11, 18, 12, 17, 13, 14, 15, 16.

From April 1 and October 31, we always need to use the +/- 1% efficiency rule.

So the economic dispatch needs to be modified to adjust the fish dispatch.

CHAPTER 4 CONCLUSION

The methodology discussed in this paper provides two piecewise local curve fitting with global continuity constraint methods, both of them can be easily programmed in numerical calculation. Using piecewise least square approximation separately on each interval is a much better fit for the trend of the actual data. For the Modified Least Square Cubic Splines, adding the constraint of specified end conditions doesn't change the standard error very much. However, it provides a much better second derivative model because it doesn't have big jumps at the end. For the Quadratic Splines, it has easy calculation formula. Also it neglects unimportant information in the given raw data and seizes the principal trend. The performance of these two approximation methods were evaluated and compared in this paper and Cubic Splines fit the data much better. It is important to note that the user of this methodology is not limited to approximate hydropower generation unit discussed here.

Moreover, the paper, dealing with individual plant optimization issues, implemented economic dispatch algorithm for ordering the dispatch of water to different type of units with the goal of obtaining the most power for a given amount of water dispatched and generated the powerhouse function for one reservoir. This powerhouse function is smooth, continuous and the first derivative of it is non-increasing. It is convenience using this optimization function to do further calculation. So when it is used in further multi-reservoir optimization, it could prevent termination at local optimized results.

Further studies on this topic are warranted. This paper neglects fish dispatch which is discussed in Fish Passage Plan. The Fish Passage Plan describes year-round operation and maintenance activities at Corps mainstem hydroelectric projects in the Federal Columbia River Power System that are coordinated through the Corps' Fish Passage Operations and Maintenance so as to protect and enhance anadromous and resident fish species listed as endangered or threatened under the Endangered Species Act, as well as other resident and migratory fish species. Considering this, the economic dispatch needs to be modified. Even though ordering the dispatch of water is changed, the idea of power target discussed in this paper could still be used.

Further investigation into the economic dispatch algorithm used in this paper can be expended upon. In this paper, we assume flexibility in operation is allowed which means generation unit could operate for part of the time, based on what gives the most efficient operation. However, if flexibility in operation is not allowed, if we must consider shut down and start up problems of the turbine, we will certainly get gaps within which the total flow cannot be passed through the units. How to deal with this issue and how to generate powerhouse function based on this new assumption needs to be studied.

With the ever increasing complexity of hydropower models, the demand for fast and accurate approximation of unit generation function and optimization of powerhouse function for one reservoir will surely increase. The methodology provided in this paper will be a valuable tool not only in optimizing the hydropower plant

considered in isolation but also in providing useful input in multi-reservoir, multi-time optimization problems.

APPENDIX

Matlab Code:

```
%command file
clear;
clc;
inputfile=xlsread('input file.xlsx');
[row,column]=size(inputfile);
[reservoir,n,index]=get_restype(inputfile);
data=xlsread('Unit_Characteristics_v3_Matlab.xlsx',reservoir);

%%delete the row with NaN
[R,C]=size(data);
datanew=data(1,:);
r=2;
while r<=R
    c=1;
    while isnan(data(r,c))==1 && c<C
        c=c+1;
    end
    if c<C
        datanew=[datanew;data(r,:)];
    end
    r=r+1;
end

typeno=(row-4)/2;%%type number
parange=zeros(inputfile(2,1)*(n+2),4);%%parameter+range+SE+R2+ybar
headno=zeros(1,typeno);%%head number
eaop=zeros(3*typeno,C-1);

for num=1:typeno%%loop through all type
    %%choose type
    type=inputfile(5+(num-1)*2,1);

    %%find the head for the indicated type in row r and end in row r1
    G=datanew(:,1);
    R=length(datanew(:,1));
    r=1;t=0;
    while isnan(G(r))==0 || t<type-1
        if isnan(G(r))==0
            r=r+1;
        else if isnan(G(r))==1
                t=t+1;r=r+1;
            end
        end
    end
    r1=r+1;
    while r1<=R && isnan(G(r1))==0
        r1=r1+1;
    end
    r1=r1-1;
```

```
    H=datanew(r,2:C);
    generation=datanew(r+1:r1,1);
    F=datanew(r+1:r1,2:C);

    %%delete NaN
    head=H(1);
    for h=2:C-1
        if isnan(H(h))==0
            head=[head,H(h)];
        end
    end
    h=length(head);
    flow=F(:,1:h);

    %%output: characteristic plot + EAOP(ans)
    EAOP=char_eaopplot(head,flow,generation,n,index,reservoir);

    headpara=inputfile(6+(num-1)*2,:);%%read in head for certain type
    headno(num)=1;% initialize number of head for each turbine type
    while headno(num)<=column && isnan(headpara(headno(num)))==0
        headno(num)=headno(num)+1;
    end
    headno(num)=headno(num)-1;
    [r,c]=size(EAOP);
    eaop((num-1)*3+1:(num-1)*3+3,1:c)=EAOP;

    for I=1:headno(num)
        Head=inputfile(6+(num-1)*2,I);

[range,ybar,SE,R2,para]=para_fourplot(head,Head,generation,flow,index,n
);
        if num==4
            caseno=headno(1)+headno(2)+headno(3)+I;
        else if num==3
                caseno=headno(1)+headno(2)+I;
            else if num==2
                    caseno=headno(1)+I;
                else if num==1
                        caseno=I;
                    end
                end
            end
        end
        parange(caseno*(n+2)-1,1)=range(1);
        parange(caseno*(n+2)-1,2)=range(2);
        parange(caseno*(n+2),1)=ybar;
        parange(caseno*(n+2),2)=SE;
        parange(caseno*(n+2),3)=R2;
        for j=1:n
            for k=1:4
                parange((caseno-1)*(n+2)+j,k)=para(j,k);
            end
        end
    end
end
```

```
xlswrite('EAOP.xlsx',eaop);
xlswrite('parameter+range.xlsx',parange);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
function [reservoir,n,index]=get_restype(inputfile)
%%choose reservoir, interval from input file
%%reservoir is in string, n is the number of interval, index is for
interval equally spaced or small interval at two ends

%%choose reservoir
reservoirindex=inputfile(1,1);
reservoir_array =
char('GrandCoulee','ChiefJoseph','LowerGraniteSTS','LowerGraniteNS','Li
ttleGooseSTS','LittleGooseNS','LowerMonumentalSTS','LowerMonumentalNS',
'IceHarborSTS','IceHarborNS','McNaryESBS','McNaryNS','JohnDaySTS','John
DayNS','TheDallesNS','BonnevilleSTS','BonnevilleNS');
reservoir = deblank(reservoir_array(reservoirindex,:));

%%choose interval
n=inputfile(3,1);
%%interval equally spaced or small interval at two ends
index=inputfile(4,1);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
function EAOP=char_eaopplot(head,flow,generation,n,index,reservoir)
%%input: reservoir name + unit type + interval number + interval
equally spaced or not
%%output: characteristic plot + EAOP(EAOP)

figure;
subplot(1,2,1);
%% characteristic plot
h=length(head);
for i=1:5:h
    [l,j]=delete_nan(flow(:,i),generation);
    plot(flow(l+1:j,i),generation(l+1:j),'linewidth', 0.5)
    hold on;
end
xlabel('flow(cfs)')
ylabel('generation(MW)')
tit=strcat('characteristic plot for ',' ',reservoir);
title(tit)

%% calculate second derivative at two end, G0 first point, G1 last
point
G0=zeros(1,h);
```

```
G1=zeros(1,h);
for i=1:h
    [l,j]=delete_nan(flow(:,i),generation);
    %%G0 calculates left end, G1 calculate right using finite
difference
    [G0(i),G1(i)]=secondderi(flow(l+1:j,i),generation(l+1:j));
end

%% calculat EAOP

if index==1
    %% 1.equally spaced
    maxflow=zeros(1,h);
    e=zeros(1,h);
    eff=zeros(1,h);
    for i=1:h % for each of the head values for the turbine type
        % read flow and generation data
        [l,j]=delete_nan(flow(:,i),generation);

[maxflow(i),e(i),eff(i),para,ybar,SE,R2,residual]=eaop_index1(generatio
n(l+1:j),flow(l+1:j,i),head(h),G0(i),G1(i),n);
    end
    EAOP=[head;maxflow;eff];
    subplot(1,2,2);
    plot(head,maxflow)
    xlabel('head(ft)')
    ylabel('EAOP(cfs)')
    tit=strcat('efficient average operating point');
    title(tit)

    %%2.small interval at two ends
else if index==2
        maxflow=zeros(1,h);
        e=zeros(1,h);
        eff=zeros(1,h);
        for i=1:h
            [l,j]=delete_nan(flow(:,i),generation);

[maxflow(i),e(i),eff(i),para,ybar,SE,R2,residual]=eaop_index2(generatio
n(l+1:j),flow(l+1:j,i),head(h),G0(i),G1(i),n);
        end
        EAOP=[head;maxflow;eff];
        subplot(1,2,2);
        plot(head,maxflow)
        xlabel('head(ft)')
        ylabel('EAOP(cfs)')
        tit=strcat('efficient average operating point');
        title(tit)
    end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
```

```
function [l,j]=delete_nan(flow,generation)
%delete the first and last few flow with no value
l=0;
while isnan(flow(l+1))==1
    l=l+1;
end
j=l;
while j<length(generation)&&isnan(flow(j+1))==0
    j=j+1;
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
function [G0,G1]=secondderi(Flow,Generation)
%calculate second derivate at two ends
j=length(Flow);
G0=2*Generation(1)/((Flow(1)-Flow(2))*(Flow(1)-
Flow(3)))+2*Generation(2)/((Flow(2)-Flow(1))*(Flow(2)-
Flow(3)))+2*Generation(3)/((Flow(3)-Flow(1))*(Flow(3)-Flow(2)));
G1=2*Generation(j-2)/((Flow(j-2)-Flow(j-1))*(Flow(j-2)-
Flow(j)))+2*Generation(j-1)/((Flow(j-1)-Flow(j-2))*(Flow(j-1)-
Flow(j)))+2*Generation(j)/((Flow(j)-Flow(j-2))*(Flow(j)-Flow(j-1)));



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
function
[maxflow,e,eff,para,ybar,SE,R2,residual]=eaop_index1(Generation,Flow,He
ad,G0,G1,n)
%calculate EAOP for equally spaced interval
j=length(Flow);
ybar=0;
SSE=0;
SST=0;
residual=zeros(1,j);
for r=1:j
    ybar=ybar+Generation(r)/j;
end
Gnew=Generation;
interval=(Flow(j)-Flow(1))/n;
b=[Flow(1):interval:Flow(j)];
para=zeros(n,4);
%%solve for the formula between the first two points
Flow1=Flow(1)+interval;
m1=1;
while Flow(m1)<Flow1
    m1=m1+1;
end
g=zeros(m1,1);
A=[0,2,0,0;
```

```
   6*(Flow(m1)-Flow(1)),2,0,0;
   3*(Flow(m1)-Flow(1))^2,2*(Flow(m1)-Flow(1)),1,0;
   (Flow(m1)-Flow(1))^3,(Flow(m1)-Flow(1))^2,(Flow(m1)-Flow(1)),1];
B=[G0;0;0;0];
para1=A\B;
for r=1:m1
    g(r)=para1(1)*(Flow(r)-Flow(1))^3+para1(2)*(Flow(r)-
Flow(1))^2+para1(3)*(Flow(r)-Flow(1))+para1(4);
    Gnew(r)=Generation(r)-g(r);
end

%%solve for the formula between the last two points
Flow2=Flow(j)-interval;
m2=j;
while Flow(m2)>=Flow2
    m2=m2-1;
end
f=zeros(j-m2+1,1);
A=[6*(Flow(j)-Flow(m2)),2,0,0;
   0,2,0,0;
   0,0,1,0;
   0,0,0,1];
B=[G1;0;0;0];
para2=A\B;
for r=1:j-m2+1
    f(r)=para2(1)*(Flow(r+m2-1)-Flow(m2))^3+para2(2)*(Flow(r+m2-1)-
Flow(m2))^2+para2(3)*(Flow(r+m2-1)-Flow(m2))+para2(4);
    Gnew(r+m2-1)=Generation(r+m2-1)-f(r);
end

s = csape(b, ...
Gnew'/fnval(csape(b,eye(length(b)),'var'),Flow'),'var');

msl=zeros(1,n);
mf=zeros(1,n);


for k=1:n %k is the interval index

    x=b(k):0.1:b(k+1);
    N=length(b(k):0.1:b(k+1));
    if k==1
        for i=1:4
            para(k,i)=s.coefs(k,i)+para1(i);
        end
    else if k==n
            for i=1:4
                para(k,i)=s.coefs(k,i)+para2(i);
            end
        else
            for i=1:4
                para(k,i)=s.coefs(k,i);
            end
        end
    end
    for m=1:N
        eval=evaluation(x(m),b(k),para(k,:));
```

```
        sl(m)=eval/x(m);
    end
    [msl(k),index1]=max(sl);
    mf(k)=x(index1);
    sl=zeros(1,length(sl));
end

sl=zeros(1,length(sl));
[e,index2]=max(msl);
maxflow=mf(index2);
eff=11.81*1000*e/Head;


for i=1:m1
    [eval(i)]=evaluation(Flow(i),b(1),para(1,:));
    residual(i)=eval(i)-Generation(i);
    SSE=SSE+(eval(i)-Generation(i))^2;
    SST=SST+(eval(i)-ybar)^2;
end

for i=m1+1:m2-1
    residual(i)=fnval(s,Flow(i))-Generation(i);
    SSE=SSE+(fnval(s,Flow(i))-Generation(i))^2;
    SST=SST+(fnval(s,Flow(i))-ybar)^2;
end

for i=m2:j
    eval(i)=evaluation(Flow(i),b(n),para(n,:));
    residual(i)=eval(i)-Generation(i);
    SSE=SSE+(eval(i)-Generation(i))^2;
    SST=SST+(eval(i)-ybar)^2;
end

SE=sqrt(SSE/j);
R2=1-SSE/SST;



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
function
[maxflow,e,eff,para,ybar,SE,R2,residual]=eaop_index2(Generation,Flow,He
ad,G0,G1,n)
%calculate EAOP for small interval at two ends
j=length(Flow);
ybar=0;
SSE=0;
SST=0;
residual=zeros(1,j);
for r=1:j
    ybar=ybar+Generation(r)/j;
end

Gnew=Generation;
```

```
interval=(Flow(j-1)-Flow(2))/(n-2);
b=[Flow(1),Flow(2):interval:Flow(j-1),Flow(j)];

%%solve for the formula between the first two points
g=zeros(2,1);
A=[0,2,0,0;
   6*(Flow(2)-Flow(1)),2,0,0;
   3*(Flow(2)-Flow(1))^2,2*(Flow(2)-Flow(1)),1,0;
   (Flow(2)-Flow(1))^3,(Flow(2)-Flow(1))^2,(Flow(2)-Flow(1)),1];
B=[G0;0;0;0];
para1=A\B;
for r=1:2
    g(r)=para1(1)*(Flow(r)-Flow(1))^3+para1(2)*(Flow(r)-
Flow(1))^2+para1(3)*(Flow(r)-Flow(1))+para1(4);
    Gnew(r)=Generation(r)-g(r);
end

%%solve for the formula between the last two points
f=zeros(2,1);
A=[6*(Flow(j)-Flow(j-1)),2,0,0;
   0,2,0,0;
   0,0,1,0;
   0,0,0,1];
B=[G1;0;0;0];
para2=A\B;
for r=1:2
    f(r)=para2(1)*(Flow(r+j-2)-Flow(j-1))^3+para2(2)*(Flow(r+j-2)-
Flow(j-1))^2+para2(3)*(Flow(r+j-2)-Flow(j-1))+para2(4);
    Gnew(r+j-2)=Generation(r+j-2)-f(r);
end

s = csape(b, ...
Gnew'/fnval(csape(b,eye(length(b)),'var'),Flow'),'var');


msl=zeros(1,n);
mf=zeros(1,n);

sl=zeros(1,round(10*interval));

for k=1:n %k is the interval index
    x=b(k):0.1:b(k+1);
    N=length(b(k):0.1:b(k+1));
    if k==1
        para(k,:)=s.coefs(k,:)+para1;
    else if k==n
            para(k,:)=s.coefs(k,:)+para2;
        else para(k,:)=s.coefs(k,:);
        end
    end
    for m=1:N
        [eval(m)]=evaluation(x(m),b(k),parameter);
        sl(m)=eval(m)/x(m);
    end
    [msl(k),index1]=max(sl);
    mf(k)=x(index1);
    sl=zeros(1,length(sl));
```

```
end

sl=zeros(1,length(sl));
[e,index2]=max(msl);
maxflow=mf(index2);
eff=11.81*1000*e/Head;


for i=1:2
    [eval(i)]=evaluation(Flow(i),b(1),para(1,:));
    residual(i)=eval(i)-Generation(i);
    SSE=SSE+(eval(i)-Generation(i))^2;
    SST=SST+(eval(i)-ybar)^2;
end

for i=3:j-2
    residual(i)=fnval(s,Flow(i))-Generation(i);
    SSE=SSE+(fnval(s,Flow(i))-Generation(i))^2;
    SST=SST+(fnval(s,Flow(i))-ybar)^2;
end

for i=j-1:j
    [eval(i)]=evaluation(Flow(i),b(n),para(n,:));
    residual(i)=eval(i)-Generation(i);
    SSE=SSE+(eval(i)-Generation(i))^2;
    SST=SST+(eval(i)-ybar)^2;
end

SE=sqrt(SSE/j);
R2=1-SSE/SST;



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
function eval=evaluation(x,x0,parameter)
%evaluate cubic function value
eval=parameter(1)*(x-x0)^3+parameter(2)*(x-x0)^2+parameter(3)*(x-
x0)+parameter(4);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
function
[range,ybar,SE,R2,para]=para_fourplot(head,Head,generation,flow,index,n
)
%%input: reservoir name + unit type + head + interval number + interval
equally spaced or not
%%output: plot of G(q) + G'(q) + G"(q) + G/q, parameter for cubic
spline in each interval(para), EAOP(ans) and 1% efficiency
point(range).

%%find flow for indicated head and corresponding generation
```

```
h=length(head);
h1=1;
while head(h1)~=Head && h1<h
    h1=h1+1;
end
i=h1;
[l,j]=delete_nan(flow(:,i),generation);
Flow=flow(l+1:j,i);
Generation=generation(l+1:j);

%%calculate second derivative at two end, G0 first point, G1 last point
j=length(Generation);

ybar=0;
SSE=0;
SST=0;
residual=zeros(1,j);
for r=1:j
    ybar=ybar+Generation(r)/j;
end
[G0,G1]=secondderi(Flow,Generation);

%%EAOP
%%1.equally spaced
if index==1

[maxflow,eff,e,para,SE,R2,ybar,residual]=eaop_index1(Generation,Flow,He
ad,G0,G1,n);
    interval=(Flow(j)-Flow(1))/n;
    b=[Flow(1):interval:Flow(j)];
    no=0;
    for i=1:n
        x=b(i):0.1:b(i+1)-0.1;
        N=length(x);
        for j=1:N
            or(j+no)=para(i,1)*(x(j)-b(i))^3+para(i,2)*(x(j)-
b(i))^2+para(i,3)*(x(j)-b(i))+para(i,4);
            fd(j+no)=3*para(i,1)*(x(j)-b(i))^2+2*para(i,2)*(x(j)-
b(i))+para(i,3);
            sd(j+no)=6*para(i,1)*(x(j)-b(i))+2*para(i,2);
            avegen(j+no)=(para(i,1)*(x(j)-b(i))^3+para(i,2)*(x(j)-
b(i))^2+para(i,3)*(x(j)-b(i))+para(i,4))/x(j);
        end
        no=no+N;
     end
    x=b(1):0.1:b(1)+(no-1)*0.1;

    figure;
    subplot(2,2,1);
    plot(x,or)
    xlabel('flow(cfs)')
    ylabel('generation(MW)')
    tit=strcat('approximation generation function');
    title(tit)
    hold on;
    plot(Flow,Generation,'or')
    hold off;
```

```
    subplot(2,2,3);
    plot(x,fd)
    xlabel('flow(cfs)')
    tit=strcat('first derivative and average generation');
    title(tit)
    hold on;
    plot(x,avegen,'r')
    hold off;
    legend('first derivative','average generation')

    subplot(2,2,2);
    plot(x,sd)
    xlabel('flow(cfs)')
    ylabel('d2G/dq2')
    tit=strcat('second derivative');
    title(tit)

    subplot(2,2,4);
    plot(Flow,residual)
    xlabel('flow(cfs)')
    ylabel('f(flow)-G')
    tit=strcat('residual');
    title(tit)

    %%find 1% point
    [range]=point_1(e,maxflow,Flow,avegen,b,no);

    ans=[Head;maxflow;eff];

    %%2.small interval at two ends
else if index==2

[maxflow,eff,e,para,ybar,SE,R2,residual]=eaop_index2(Generation,Flow,He
ad,G0,G1,n);
        interval=(Flow(j-1)-Flow(2))/(n-2);
        b=[Flow(1),Flow(2):interval:Flow(j-1),Flow(j)];
        ans=[Head;maxflow;eff];

        no=0;
        i=1;
        y=b(i):0.1:b(i+1)-0.1;
        N=length(y);
        for j=1:N
            or(j+no)=para(i,1)*(y(j)-b(i))^3+para(i,2)*(y(j)-
b(i))^2+para(i,3)*(y(j)-b(i))+para(i,4);
            fd(j+no)=3*para(i,1)*(y(j)-b(i))^2+2*para(i,2)*(y(j)-
b(i))+para(i,3);
            sd(j+no)=6*para(i,1)*(y(j)-b(i))+2*para(i,2);
            avegen(j+no)=(para(i,1)*(y(j)-b(i))^3+para(i,2)*(y(j)-
b(i))^2+para(i,3)*(y(j)-b(i))+para(i,4))/y(j);
        end
        no=no+N;
        for i=2:n-1
            x=b(i):0.1:b(i+1)-0.1;
            N=length(x);
            for j=1:N
```

```
                or(j+no)=para(i,1)*(x(j)-b(i))^3+para(i,2)*(x(j)-
b(i))^2+para(i,3)*(x(j)-b(i))+para(i,4);
                fd(j+no)=3*para(i,1)*(x(j)-b(i))^2+2*para(i,2)*(x(j)-
b(i))+para(i,3);
                sd(j+no)=6*para(i,1)*(x(j)-b(i))+2*para(i,2);
                avegen(j+no)=(para(i,1)*(x(j)-b(i))^3+para(i,2)*(x(j)-
b(i))^2+para(i,3)*(x(j)-b(i))+para(i,4))/x(j);
            end
            no=no+N;
        end
        i=n;
        z=b(i):0.1:b(i+1)-0.1;
        N=length(z);
        for j=1:N
            or(j+no)=para(i,1)*(z(j)-b(i))^3+para(i,2)*(z(j)-
b(i))^2+para(i,3)*(z(j)-b(i))+para(i,4);
            fd(j+no)=3*para(i,1)*(z(j)-b(i))^2+2*para(i,2)*(z(j)-
b(i))+para(i,3);
            sd(j+no)=6*para(i,1)*(z(j)-b(i))+2*para(i,2);
            avegen(j+no)=(para(i,1)*(z(j)-b(i))^3+para(i,2)*(z(j)-
b(i))^2+para(i,3)*(z(j)-b(i))+para(i,4))/z(j);
        end
        no=no+N;
        x=b(1):0.1:b(1)+(no-1)*0.1;

        figure;
        subplot(2,2,1);
        plot(x,or)
        xlabel('flow(cfs)')
        ylabel('generation(MW)')
        tit=strcat('approximation generation function');
        title(tit)
        hold on;
        plot(Flow,Generation,'or')
        hold off;

        subplot(2,2,3);
        plot(x,fd)
        xlabel('flow(cfs)')
        tit=strcat('first derivative and average generation');
        title(tit)
        hold on;
        plot(x,avegen,'r')
        hold off;
        legend('first derivative','average generation')

        subplot(2,2,2);
        plot(x,sd)
        xlabel('flow(cfs)')
        ylabel('d2G/dq2')
        tit=strcat('second derivative');
        title(tit)

        subplot(2,2,4);
        plot(Flow,residual)
        xlabel('flow(cfs)')
        ylabel('f(flow)-G')
```

```
        tit=strcat('residual');
        title(tit)

        %%find 1% point
        [range]=point_1(e,maxflow,Flow,avegen,b,no);

    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
%powerhouse function of economic dispatch
clear;
clc;
inputfile=xlsread('PH input file.xlsx');
[reservoir,n,index]=get_restype(inputfile);
[unit_no]=get_unitno(inputfile);
Head=inputfile(2,1);
data=xlsread('Unit_Characteristics_v3_Matlab.xlsx',reservoir);

%%delete the row with NaN
[R,C]=size(data);
datanew=data(1,:);
r=2;
while r<=R
    c=1;
    while isnan(data(r,c))==1 && c<C
        c=c+1;
    end
    if c<C
        datanew=[datanew;data(r,:)];
    end
    r=r+1;
end

type_no=type_no(datanew);
%result store all the useful parameters(EAOP,eff,para,b)
result=zeros(type_no*(n+2),5);
efficiency=zeros(1,type_no);
for type=1:type_no%%loop through all type

    %%find the head for the indicated type in row r and end in row r1
    G=datanew(:,1);
    R=length(datanew(:,1));
    r=1;t=0;
    while isnan(G(r))==0 || t<type-1
        if isnan(G(r))==0
            r=r+1;
        else if isnan(G(r))==1
                t=t+1;r=r+1;
            end
        end
    end
    r1=r+1;
    while r1<=R && isnan(G(r1))==0
```

```
        r1=r1+1;
    end
    r1=r1-1;

    H=datanew(r,2:C);
    generation=datanew(r+1:r1,1);
    F=datanew(r+1:r1,2:C);

    %%delete NaN
    head=H(1);
    for h=2:C-1
        if isnan(H(h))==0
            head=[head,H(h)];
        end
    end
    h=length(head);
    flow=F(:,1:h);

    %%find flow for indicated head and corresponding generation
    h1=1;
    while head(h1)~=Head && h1<h
        h1=h1+1;
    end
    i=h1;
    [l,j]=delete_nan(flow(:,i),generation);
    Flow=flow(l+1:j,i);
    Generation=generation(l+1:j);

    %%calculate second derivative at two end, G0 first point, G1 last
point
    [G0,G1]=secondderi(Flow,Generation);
    %%EAOP
    %%1.equally spaced
    if index==1

[maxflow,e,eff,para,SE,R2,ybar,residual]=eaop_index1(Generation,Flow,He
ad,G0,G1,n);
        interval=(Flow(j-l)-Flow(1))/n;
        b=[Flow(1):interval:Flow(j-l)];
        %%2.small interval at two ends
    else if index==2

[maxflow,e,eff,para,ybar,SE,R2,residual]=eaop_index2(Generation,Flow,He
ad,G0,G1,n);
            interval=(Flow(j-l-1)-Flow(2))/(n-2);
            b=[Flow(1),Flow(2):interval:Flow(j-l-1),Flow(j-l)];
        end
    end
    result((n+2)*type,1)=type;
    result((n+2)*type,2)=maxflow;
    result((n+2)*type,3)=eff;
    result((n+2)*type,4)=e;
    result((n+2)*(type-1)+1:(n+2)*(type-1)+n+1,1)=b';
    result((n+2)*(type-1)+1:(n+2)*(type-1)+n,2:5)=para;
    efficiency(type)=eff;
end
% operating order of different type
```

```
order=zeros(1,type_no);
[effnew,o]=sort(efficiency);
for i=1:type_no
    order(i)=o(type_no+1-i);
end
%switch point
SP_no=0;
if type_no~=1
    for i=1:type_no-1
        SP_no=SP_no+i;
    end
    SP=zeros(1,SP_no);
    for SP_index=1:SP_no
        if SP_index==1
            result1=result((order(1)-1)*(n+2)+1:order(1)*(n+2),:);
            result2=result((order(2)-1)*(n+2)+1:order(2)*(n+2),:);
        elseif SP_index<=3
            result1=result((order(SP_index-1)-
1)*(n+2)+1:order(SP_index-1)*(n+2),:);
            result2=result((order(3)-1)*(n+2)+1:order(3)*(n+2),:);
        elseif SP_index<=6
            result1=result((order(SP_index-3)-
1)*(n+2)+1:order(SP_index-3)*(n+2),:);
            result2=result((order(4)-1)*(n+2)+1:order(4)*(n+2),:);

        end
        SP(SP_index)=switchpoint(result1,result2,n);
    end
end
%powerhouse function
PH_part=2*type_no;

%part 1
result1=result((order(1)-1)*(n+2)+1:order(1)*(n+2),:);
[x1,y1]=straigut_part(1,result1,order,unit_no,n);

%part 2
if type_no==1
    [x2,y2]=part_2noSP(result1,order,unit_no,n);
    x=[x1,x2];
    y=[y1,y2];
    plot(x1,y1,'r')
    hold on;
    plot(x2,y2,'bo')
    xlabel('flow(cfs)')
    ylabel('generation(MW)')
    tit=strcat('powerhouse function');
    hold off;
else
    [x2,y2]=part_2(SP,result1,order,unit_no,n);
    %part 3
    result2=result((order(2)-1)*(n+2)+1:order(2)*(n+2),:);
    [x3,y3]=straigut_part(3,result2,order,unit_no,n);

    %part 4
    if type_no==2
        [x4,y4]=part_4end(SP,result1,result2,order,unit_no,n);
```

```
        [x3new,y3new]=change_xy(x2,y2,x3,y3);
        plot(x1,y1,'r')
        hold on;
        plot(x2,y2,'bo')
        plot(x3new,y3new,'r')
        plot(x4,y4,'bo')
        xlabel('flow(cfs)')
        ylabel('generation(MW)')
        tit=strcat('powerhouse function');
        hold off;
        x=[x1,x2,x3new,x4];
        y=[y1,y2,y3new,y4];
    else
        result3=result((order(3)-1)*(n+2)+1:order(3)*(n+2),:);
        [x4,y4]=part_4(SP,result1,result2,result3,order,unit_no,n);
        %part 5

        [x5,y5]=straigut_part(5,result3,order,unit_no,n);
        %part 6
        if type_no==3

[x6,y6]=part_6end(SP,result1,result2,result3,order,unit_no,n);
            [x3new,y3new]=change_xy(x2,y2,x3,y3);
            [x5new,y5new]=change_xy(x4,y4,x5,y5);
            plot(x1,y1,'r')
            hold on;
            plot(x2,y2,'bo')
            plot(x3new,y3new,'r')
            plot(x4,y4,'bo')
            plot(x5new,y5new,'r')
            plot(x6,y6,'bo')
            xlabel('flow(cfs)')
            ylabel('generation(MW)')
            tit=strcat('powerhouse function');
            hold off;
            x=[x1,x2,x3new,x4,x5new,x6];
            y=[y1,y2,y3new,y4,y5new,y6];
        else
            result4=result((order(4)-1)*(n+2)+1:order(4)*(n+2),:);

[x6,y6]=part_6(SP,result1,result2,result3,result4,order,unit_no,n);
            %part 7
            [x7,y7]=straigut_part(7,result4,order,unit_no,n);
            %part 8

[x8,y8]=part_8end(SP,result1,result2,result3,result4,order,unit_no,n);
            [x3new,y3new]=change_xy(x2,y2,x3,y3);
            [x5new,y5new]=change_xy(x4,y4,x5,y5);
            [x7new,y7new]=change_xy(x6,y6,x7,y7);
            plot(x1,y1,'r')
            hold on;
            plot(x2,y2,'bo')
            plot(x3new,y3new,'r')
            plot(x4,y4,'bo')
            plot(x5new,y5new,'r')
            plot(x6,y6,'bo')
            plot(x7new,y7new,'r')
            plot(x8,y8,'bo')
```

```
            xlabel('flow(cfs)')
            ylabel('generation(MW)')
            tit=strcat('powerhouse function');
            hold off;
            x=[x1,x2,x3new,x4,x5new,x6,x7new,x8];
            y=[y1,y2,y3new,y4,y5new,y6,y7new,y8];
        end
    end
end
figure;
plot(x,y)
xlabel('flow(cfs)')
ylabel('generation(MW)')
tit=strcat('powerhouse function');
```

REFERENCES

Arce, A. (2002), Optimal dispatch of generating units of the Itaipu hydroelectric plant, *Power Systems*, 17(1), 154-158, doi: 10.1109/59.982207.

Cai, X. M., D. C. Mckinney, and L. S. Lasdon (2001), Solving nonlinear water management models using a combined genetic algorithm and linear programming approach, *Advances in Water Resources*, 24(6), 667-676, doi: 10.1016/S0309-1708(00)00069-5.

Chapra, S. C., R. Canale (2006), Numerical Methods for Engineers, McGraw-Hill Companies, 637.

Chowdhury, B. H., S. Rahman (1990), a review of recent advances in economic dispatch, *Power Systems*, 5(4), 1248-1259, doi: 10.1109/59.99376.

Cook, J. and J. Walsh (2008), Optimization of Hydro-Power Plants for Generation.

Gluss, B. (1964), An alternative method for continuous line segment curve-fitting, *Information and Control*, 7(2), 200-206, doi: 10.1016/S0019-9958(64)90109-3.

Hou C., Y. Ceng, D. Wu and Z. Yang (2011), Global Continuous Curve Fitting Method of Piecewise Least Square Fitting with Global Continuity, *Journal of Chongqing Normal University*, 28(6), 44-48, doi: CNKI: 50-1165 /N. 20111110. 1503. 009.

Labadie, J. W. (2004), Optimal operation of multireservoir systems: state of the art review, *Water Resources Planning and Management*, 130(2), 93-111, doi: 10.1061/(ASCE)0733-9496(2004)130:2(93).

Li, F., C. A. Shoemaker, J. Wei and X. Fu (2012), A New Reservoir Operation Policy Generation Method (SOSM) Using Scenarios Optimization (SO) and Surrogate Model (SM), *Advances in Intelligent and Soft Computing*, 136, 679-688, doi: 10.1007/978-3-642-27711-5_90.

Li, F., C. A. Shoemaker, J. Wei and X. Fu (2013), Estimating Maximal Annual Energy Given Heterogeneous Hydropower Generating Units with Application to the Three Gorges System, *Journal of Water Resources Planning and Management*, 139(3), 265–276, doi: 10.1061/(ASCE)WR.1943-5452.0000250.

Oliveira, R., D. P. Loucks (1997), Operating Rules for Multi-reservoir Systems, *Water resources research*, 33(4), 839-852, doi: 10.1029/96WR03745.

Shawwash, Z. K., T. K. Siu and S.O.D. Russell (2000), The B.C. Hydro Short Term Hydro Scheduling Optimization Model, *Power Systems*, 15(3), 1125-1131, doi: 10.1109/59.871743.

Stedinger, J., C. A. Shoemaker, S. N. Tan, L. Chen. “Progress Report to BPA on Optimal Operation of Hydropower System under Uncertainty.”

Wood, A. J. and B. F. Wollenberg (1996), Power generation, operation, and control, second edition, John Wiley & sons, Inc.

Wurbs, R. (1993), Reservoir-System Simulation and Optimization Models, *Journal of Water Resources Planning and Management*, 19(4), 455–472, doi: 10.1061/(ASCE)0733-9496(1993)119:4(455).

Yeh, W. G.(1985), Reservoir Management and Operation Models: a state-of-the art review, *Water Resources Research*, 21(12), 1797-1818, doi: 10.1029/WR021i012p01797.

Yeh, W. G., L. Becker, S. Q. Hua, D. P. Wen, and D. P. Liu (1992), Optimization of real time hydrothermal system operation, *Water Resources Planning and Management*, 118(6), 636-653, doi: 10.1061/(ASCE)0733-9496(1992)118:6(636).

US Army Corps of Engineers, Fish passage plan, March 2012.