2011

Modeling Direct Runoff Hydrographs with the Surge Function

Denis Voytenko
University of South Florida, dvoytenk@mail.usf.edu

Modeling Direct Runoff Hydrographs with the Surge Function

by

Denis Voytenko

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
Department of Geology
College of Arts and Sciences
University of South Florida

Major Professor: H.L. Vacher, Ph.D.
Mark C. Rains, Ph.D.
Mark Stewart, Ph.D.

Date of Approval:
May 17, 2011

Keywords: streamflow, probability distribution function, optimization, parameter estimation, gamma distribution, Weibull distribution, hydrologic modeling

Table of Contents

List of Tables

List of Figures

Abstract

A surge function is a mathematical function of the form $f(x) = ax^p e^{-bx}$. We simplify the surge function by holding p constant at 1 and investigate the simplified form as a potential model to represent the full peak of a stream discharge hydrograph. The previously studied Weibull and gamma distributions are included for comparison. We develop an analysis algorithm which produces the best-fit parameters for every peak for each model function, and we process the data with a MATLAB script that uses spectral analysis to filter year-long, 15-minute, stream-discharge data sets. The filtering is necessary to locate the concave-upward inflection points used to separate the data set into its constituent, individual peaks. The Levenberg-Marquardt algorithm is used to iteratively estimate the unknown parameters for each version of the modeled peak by minimizing the sum of squares of residuals. The results allow goodness-of-fit comparisons between the three model functions, as well as a comparison of peaks at the same gage through the year of record. Application of these methods to five rivers from three distinct hydrologic regions shows that the simple surge function is a special case of the gamma distribution, which is known to be useful as a modeling function for a full-peak hydrograph. The study also confirms that the Weibull distribution produces good fits to 15-minute hydrograph data.

1. Introduction

Hydrographs are often used by engineers and hydrologists to predict flooding and to allocate surface water resources. Therefore, knowing more about how hydrographs vary with time at the same stream gage, and how they differ across watesheds, will improve efforts to solve water resource management problems.

An individual peak of the hydrograph is generally associated with a storm event. The shape of the peak can be described by a variety of mathematical functions. No previous work, however, has been published on obtaining model coefficients for event peaks over a time period at the same gage station, with a subsequent comparison of the coefficients across hydrologic regions.

The plot of the function, $y = axe^{-bx}$, resembles a hydrograph peak, with the ordinate y representing the discharge or stage in some way, and the abscissa x representing time or its proxy. The function, which will later be called the simple surge function, can be fitted to a number of peaks to obtain the desired model coefficients, a and b.

Because we wish to compare extended streamflow periods across multiple rivers, we develop an automated approach to fitting each peak. Our approach has two technical goals: hydrograph discretization and peak modeling; both require the methods to be repeatable from river to river.

Discretizing hydrographs requires separating the 15-minute data into individual peaks before they are modeled. We will use methods of spectral analysis, specifically, frequency-based filtering in which we construct a crude periodogram and use

the endpoint of the highest frequency range with the highest spectral density as the corner frequency.

We will model the peaks with the iterative Levenberg-Marquardt Algorithm, which is used to estimate the best-fit coefficients for the simple surge function and to compare them with the best-fit coefficients from the gamma and Weibull distributions.

We present a background explaining the types of hydrographs and their applications, along with descriptions of each of the functions used to model the peaks. This is followed by a summary of the technical methods used to process the data; detailed descriptions of the methods are in the appendices. The results are shown primarily as figures and tables, and are followed by a discussion about the utility of the automated methods, the relationship between functions, and the interpretation of the function parameters.

2. Background

2.1 Types of Hydrographs

A streamflow hydrograph is a "graphical representation of instantaneous discharge at a given location with respect to time during and after a storm or a snowmelt event" (Fang et al., 2005).

The streamflow hydrograph can be separated into three components: baseflow, interflow and runoff. Baseflow is defined as "flow in a channel that exists even before the occurence of rainfall" (Fang et al., 2005). Interflow describes the lateral movement of water in the soil and is generally allocated to either baseflow or runoff. To be more precise, quick interflow is combined with runoff, while delayed interflow is combined with baseflow (Brodie and Hostetler, 2005). Runoff can be defined as the flow of water along the ground surface and the rainfall onto the stream (Ramirez, 2000).

The first modified hydrograph is the direct runoff hydrograph (DRH), which is produced after baseflow ordinates are subtracted from the total discharge ordinates. It should be noted that the amount of effective rainfall (i.e. volume of rainfall that did not infiltrate and is available for runoff) is the measure used to account for the volume of runoff. Thus, the amount of rainfall (depth) multiplied by the drainage basin area produces the total volume underneath the direct runoff hydrograph curve. Therefore, it is possible to reduce the DRH to a probability distribution function (pdf), which can subsequently be modeled using a parameter estimation algorithm, like the Levenberg-Marquardt algorithm.

However, instead of accounting for the total discharge volume, it is also possible to account for the amount of rainfall, producing a unit hydrograph (UH). Unit

hydrographs describe the stream discharge, given a normalized amount of effective precipitaion across a watershed. For example, a one-hour unit hydrograph can describe the stream discharge given a one-hour event of 1 inch of effective rainfall across the watershed area.

If the duration of effective rainfall is known, each unit hydrograph can be described as an n-hour hydrograph. Then, if the duration of effective rainfall is accounted for, the hydrograph becomes an instantaneous unit hydrograph (IUH). The instantaneous unit hydrograph represents the response of the stream to an instantaneous impluse of rainfall.

The advantage of unit hydrographs is that they can be varied by convolution. Convolution is a "mathematical technique to combine two signals into one" (Smith, 1999) and deconvolution is its reverse process. Unit hydrographs can be convolved with measurements of effective rainfall on the same timestep to produce a hydrograph which accounts for the duration and amount of rainfall. Conversely, a DRH can be deconvolved into a unit hydrograph and an effective hyetograph, as long as both have the same timestep measurements and either the unit hydrograph or the hyetograph is provided. All of the aforementioned methods are applied to gaged basins.

For ungaged basins, synthetic unit hydrographs (SUH) are developed based upon watershed characteristics, such as drainage basin area, time to base, and time to peak. SUH methods, pioneered by Snyder (1938) and Espey and Winslow (1968), rely on calculating a set of coefficients, like time to peak, and then estimating a small number of points on the hydrograph.

The dimensionless unit hydrograph (DUH), introduced by the Soil Conservation Service (SCS), describes an average synthetic unit hydrograph (USDA-SCS, 1972; He, 2004). The shape of the peak is shown by normalizing both axes. The y-axis plots instantaneous discharge divided by the peak discharge; the range becomes

0 to 1. The x-axis plots instantaneous time divided by the time to peak; the range is usually from 0 to 5.

Bhunya et al. (2011) compared SUH methods and suggested that methods based on probability distribution functions, such as the gamma, Weibull and beta distributions, are preferable, because they are not subjective and the area underneath the curve hydrograph is always equal to one.

2.1.1 Assumptions of hydrograph theory.

Dooge (1959) suggested that there are two underlying principles behind unit hydrographs: invariance and superposition. Invariance implies that basin properties do not change with time - if multiple events of the same volume of effective rainfall and duration were to occur at the gaging station, the produced unit hydrograph would always be the same. Superposition implies proportionality - if starting with a 1-inch unit hydrograph, meaning the hydrograph produced by a 1-inch rainfall, and we needed to produce a 2-inch hydrograph, all discharge ordinates would be multiplied by 2. Additionally, lagging one 1-inch hydrograph by 1-hour after another, and summing the ordinates would produce a 1-inch, 2-hour hydrograph.

Ramirez (2000) summarized the fundamental assumptions to be the following: watersheds behave like linear systems (i.e., Dooge's assumption); effective rainfall intensity is uniform across the drainage basin; rainfall excess is of constant intensity; and the duration of the DRH depends only on rainfall duration.

2.2 The Surge Function

2.2.1 General concepts.

A surge function is a function that can be used to model the behavior of a system when the response to a stimulus increases rapidly in the beginning and then drops off slowly. Mathematically, a function is a surge function when it has the

following form (Gordon, 2006):

$$f(x) = ax^p e^{-bx} \tag{1}$$

Applying the surge function to model a physical process is not new - variations of these functions have been used to model both the concentration-time behavior of drugs and the response to an advertisement campaign (Gordon, 2006). The case of the surge function with $p = 1$ also appears in biology as the Ricker Curve (Ricker, 1954), which describes the behavior of a fish population where the spawning population begins to affect the juvenile population before recruitment (Quinn and Deriso, 1999). Although he did not call it a surge function, Edson (1951) suggested using its full form (1) as an equation for a hydrograph. An explanation of his derivation follows in the next section.

Both Edson (1951) and Gordon (2006) noticed that this function has one maximum and two inflection points, which can be derived via calculus. If the derivative of this function is set to be zero, the maximum is $\frac{p}{b}$. Additionally, setting the second derivative of the function to be zero finds the two inflection points, $\frac{p}{b} \pm \frac{\sqrt{p}}{b}$.

In the case of this thesis, only the case where $p = 1$ will be considered, giving the form:

$$f(x) = axe^{-bx} \tag{2}$$

This function will be referred to as the simple surge function.

2.2.2 Edson's derivation of the surge function.

To derive the full surge function (1), Edson (1951) related two distinct functions. The first describes the relationship between the watershed area and the time needed for the effective rainfall to reach the stream as a parabola-like power function of the following form:

$$A \propto T^x \qquad x > 1 \tag{3}$$

where A is watershed area; T is time; and x is a constant relating time to area. Now, since effective rainfall is proportional to the watershed area, (3) can be rewritten in terms of discharge as:

$$Q \propto T^x \qquad x > 1 \tag{4}$$

The second function describes the behavior of the watershed as a reservoir with exponentially declining reserves:

$$Q \propto e^{-yT} \qquad y > 0 \tag{5}$$

where Q is discharge; T is time; and y is a constant controlling the rate of decline. Now, since both relationships relate discharge to time, they can be combined as:

$$Q \propto T^x e^{-yT} \tag{6}$$

Because Q is proportional to the combined relationship, another constant, B, is introduced to produce an equation:

$$Q = BT^x e^{-yT} \tag{7}$$

This equation is exactly analogous to the surge function. Edson then used the equation to solve for maximum discharge and time to maximum discharge, which led him to derive a version of the two-parameter gamma distribution.

2.3 The Gamma and Weibull Distributions

2.3.1 The gamma distribution.

The gamma distribution is:

$$f(x; k, \theta) = x^{k-1} \frac{e^{-x/\theta}}{\theta^k \Gamma(k)} \tag{8}$$

where k is generally known as the shape parameter, and θ is known as the scale parameter.

This distribution has a long history of being applied to hydrograph models, starting as early as the Edson paper (1951). When Dooge (1959) derived the primary unit hydrograph theory, he related the gamma distribution to a linear reservoir model. The reservoir model was also corroborated by Nash (1959).

Singh and Chowdhury (1985) compared twelve different parameter-fitting methods for the gamma distribution to model parameters for four experimental event peaks. They also suggested using the gamma distribution and multiplying it by the total volume of discharge, which is also the total volume the effective rainfall distributed across the basin area, to model either unit or direct runoff hydrographs.

In Singh's and Chowdhury's approach (1985), the difference between the UH and the DRH is not particularly significant, as their pdf and the modeled coefficients are exactly the same. The two kinds of hydrographs are scaled versions of one another. In the UH case, the pdf is multiplied by the total discharge volume given a unit-depth of rainfall. The DRH case, on the other hand, is multiplied by the total discharge volume of the event, and not the unit depth.

2.3.2 The Weibull distribution.

Another probability distribution function that has been used to describe the shape of a hydrograph (Ahmed, 1990; Bhunya et al., 2006; Nadarajah, 2007) is the Weibull distribution:

$$f(x;a,b) = \frac{a}{b}\left(\frac{x}{b}\right)^{a-1} e^{-\left(\frac{x}{b}\right)^{a}} \tag{9}$$

Bhunya et al. (2006) preferred the Weibull distribution to the gamma distribution for peak modeling due to its flexibility. Hence, both distributions are compared to the simple surge function in this study.

2.4 Hydrology

2.4.1 Hydrologic landscape regions.

Winter (2001) introduced the concept of a hydrologic landscape unit to describe the flow of water in different hydrologic environments which differ from one another due to variations in "land-surface form, geology and climate". Wolock et al. (2004) expanded on Winter's concepts by discretizing the United States into 43,931 watersheds and analyzing their hydrologic landscape unit properties with statistical methods and GIS. Their results subdivided the United States into twenty types of hydrologic units called hydrologic landscape regions (HLRs). These regions address differences in hydrologic properties between watersheds instead of differences in geographic location.

2.4.2 The five rivers.

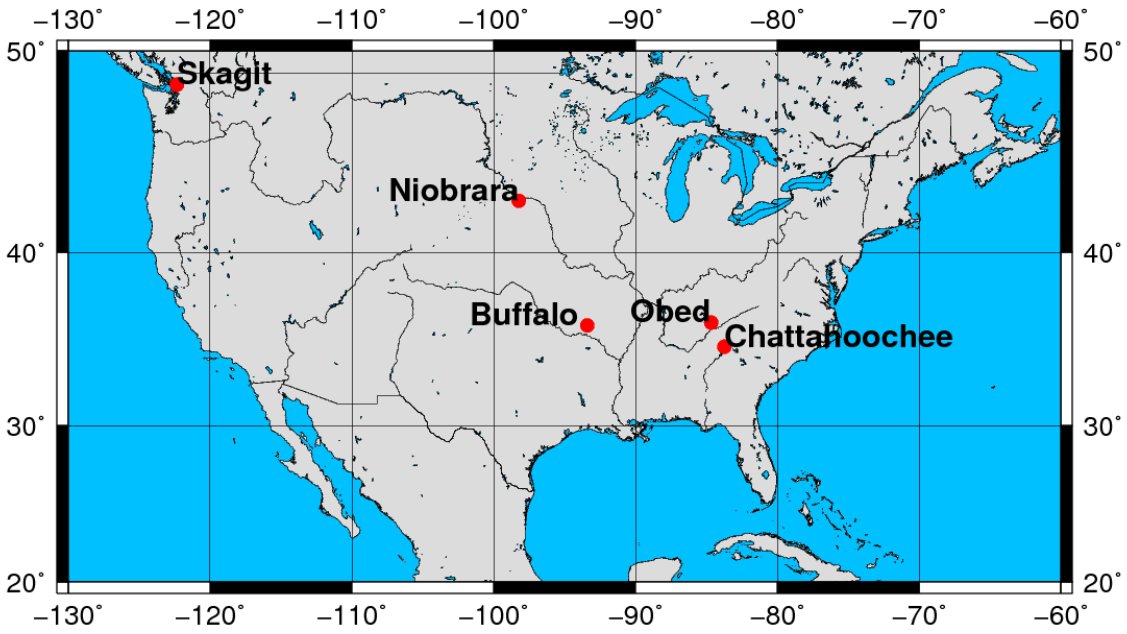

Figure 2.1: Locations of the studied rivers

Because we wished to use rivers that were distributed across the country and were located in a number of different environments, we selected the Buffalo, Chatta-

hoochee, Niobrara, Obed and Skagit rivers for this study. We also wanted to work with rivers that were not significantly influenced by human activity. All rivers except for the Skagit have segments that are designated to either be wild, scenic or recreational by Congress, and as such, should not be heavily controlled.

The Buffalo River flows east across the Arkansas Ozarks and discharges into the White River. It is undammed and 135 miles long. The gage from which the data were obtained is the USGS gage 07055646 located near Broxley, AR. This gage is located in HLR 16, which represents "humid mountains with permeable soils and impermeable bedrock" and where the primary hydrologic flow paths are in the shallow groundwater (Wolock et al., 2004). The drainage basin area for this gage is 57.4 square miles.

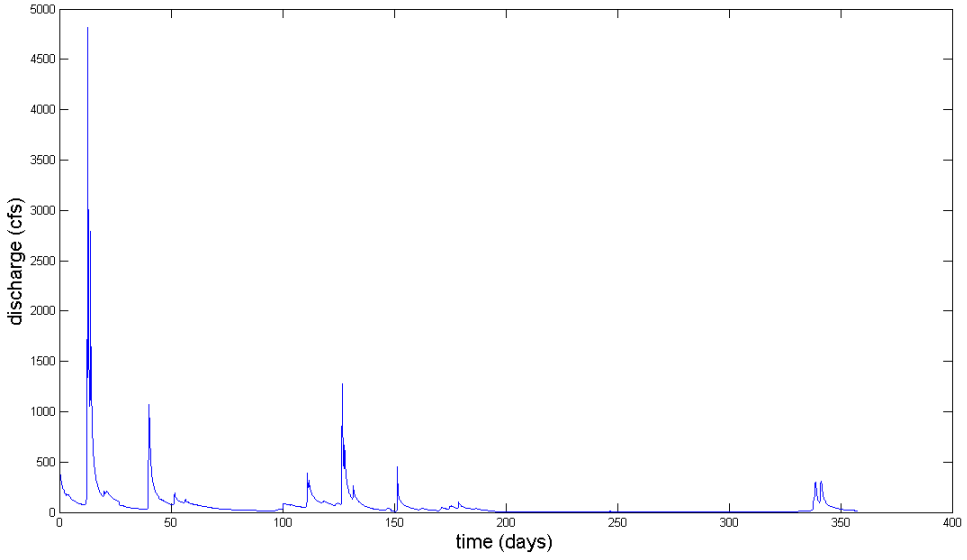

Figure 2.2: Unprocessed Hydrograph for the Buffalo River

The Chattahoochee River originates in the Appalachian Mountains in northern Georgia and flows southward until it forms the Appalachicola River. The USGS gage used is 02330450, located in Helen, GA. The gage site is close to the beginning of the river and does not appear to be regulated upstream, but is regulated downstream. This gage is located in HLR 16, which represents "humid mountains with permeable soils and impermeable bedrock" and where the primary hydrologic flow paths are in the shallow groundwater (Wolock et al., 2004). The drainage basin area for this gage is 44.7 square miles.

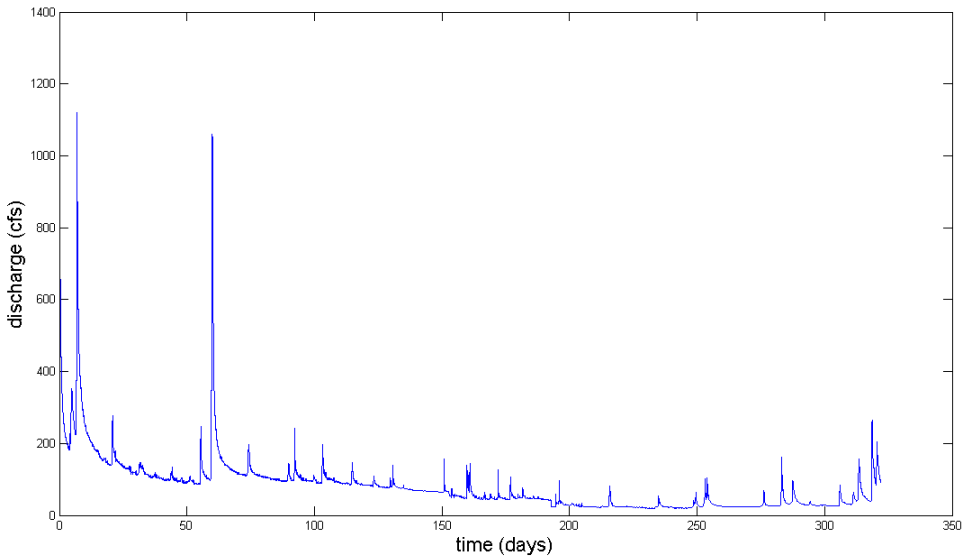

Figure 2.3: Unprocessed Hydrograph for the Chattahoochee River

The Obed River is a wild and scenic river which originates in the Cumberland Plateau in Tennessee and flows in a north, and then east, direction until it reaches the Emory River. The gage, 03539800, is located near Lancing, TN. This gage is located in HLR 16, which represents "humid mountains with permeable soils and impermeable bedrock" and where the primary hydrologic flow paths are in the shallow groundwater (Wolock et al., 2004). The drainage basin area for this gage is 518 square miles.

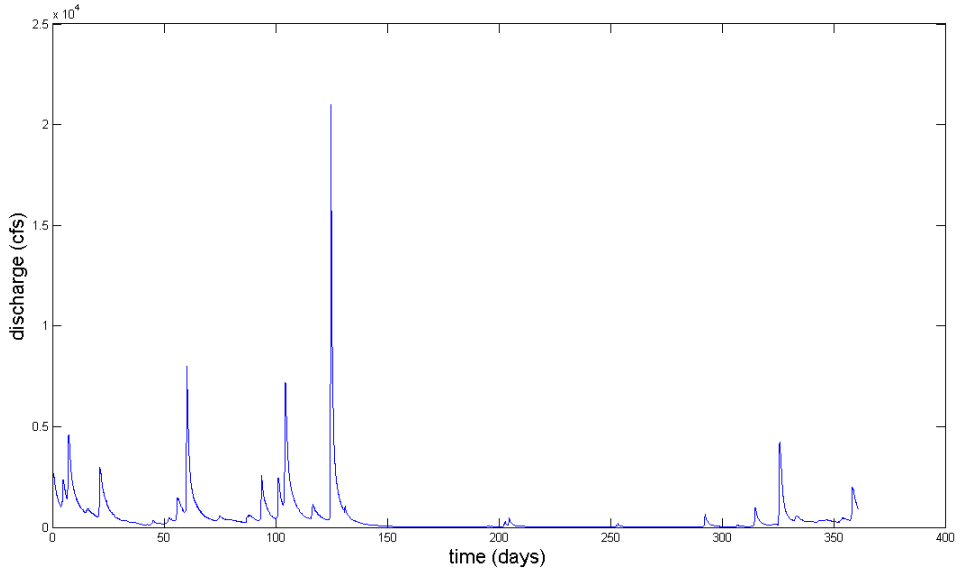

Figure 2.4: Unprocessed Hydrograph for the Obed River

The Niobrara River originates in the High Plains of Wyoming and flows in an eastward direction through Nebraska until it discharges into the Missouri River. A 76-mile portion of the Niobrara is designated as a scenic river. The USGS gage 06465500, is located near Verdel, NE, outside of the scenic river boundaries. This gage is located in HLR 13, which represents "semiarid plateaus with impermeable soils and bedrock" and where the primary hydrologic flow paths are via overland flow (Wolock et al., 2004). The drainage basin area for this gage is $11,580$ square miles.

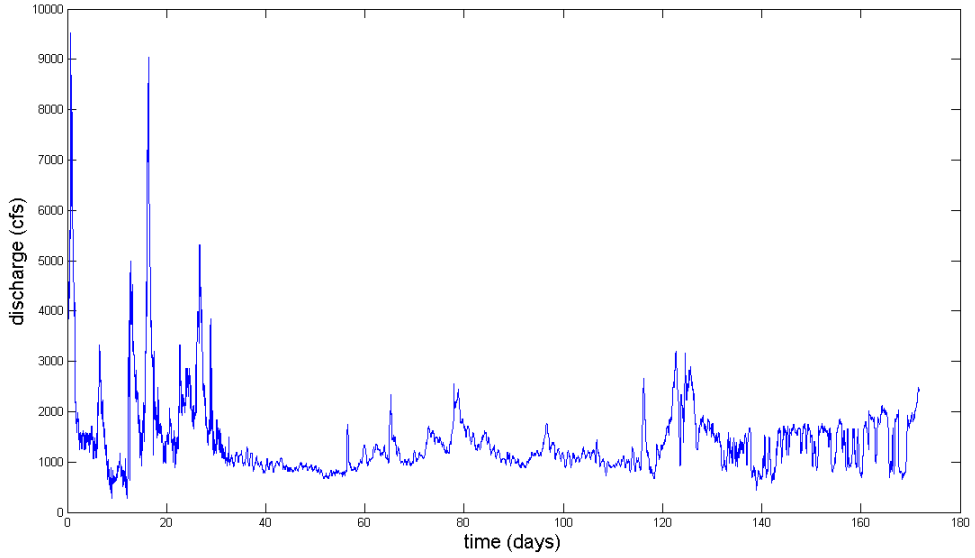

Figure 2.5: Unprocessed Hydrograph for the Niobrara River

The Skagit River originates in the Canadian part of the Cascade Range and flows in southwest direction towards Puget Sound. Some parts of the river are considered scenic and recreational, while others are heavily controlled. The gage 12200500 is located near Mount Vernon, WA, where it has been subject to control structures upstream. This gage is located in HLR 3, which represents "subhumid plains with impermeable soils and permeable bedrock" and where the primary hydrologic flow paths are via overland flow and in the deep groundwater systems (Wolock et al., 2004). The drainage basin area for this gage is $3,093$ square miles.

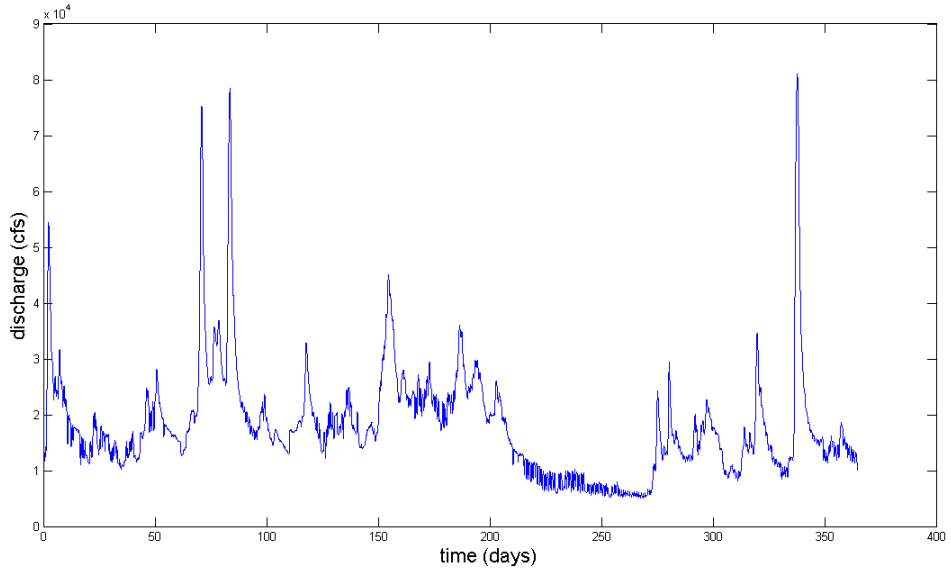

Figure 2.6: Unprocessed Hydrograph for the Skagit River

3.0.3 Obtaining and loading the data.

The method for analyzing the data was developed ad hoc as a part of the study. The procedure started with 15-minute data downloaded from the USGS Water Data site, and the product of the method development was a MATLAB script. More details about the script are found in appendices A and B, and the script itself is in appendix C.

One of the data processing hurdles lies in loading downloaded files into MATLAB. They contain more data than the script requires. The script relies only on two columns of data - a column of dates and a column of discharge values. In fact, the downloaded data files contain 67 header rows and 6 columns of unnecessary data, such as the gage number, time zone and precision. The extra rows and columns are deleted using Excel, and the generated file is saved into a working MATLAB folder as a tab-delimited text file.

Once loaded into MATLAB, the file is checked for missing and redudnant values. Redundant instances of dates and times, and the associated discharges are removed so that only one instance remains. If values are missing, the script expands the data set by adding the missing dates and times, and linearly interpolates discharge values to fill in the gaps.

3.0.4 Filtering and separating the data.

The 15-minute data are plagued with a high-frequency noise problem. The unfiltered data sets contain many locations where the derivative changes sign far

more often than at the turning points between peaks. Thus, to develop an automated discretization method, we need to filter out the noise.

We did not achieve satisfactory filter results with moving averaging or simple exponential smoothing. Both required using an arbitrary coefficient - the window length for the moving average, and the smoothing coefficient for the simple exponential smoothing. Moving averaging also required multiple passes, making it a slow and unreliable way of smoothing the data. Frequency-based filtering, however, produced much better results.

We discretize the peaks by analyzing a crude periodogram to determine the corner frequency to be used in a lowpass Butterworth filter. The Butterworth filter is used because it has a maximally flat passband, meaning that data to be kept are minimally affected by the filter. Once the data are filtered, they are cross-correlated with the original data to determine the amount of lag and are shifted accordingly. It is to be noted that the filtered data are not modeled; they are used only for determining the cutoff points between new peaks.

Our post-filtering discretization approach is to consider that the data behave like a smooth and continuous function and examine the signs of the first derivative. The points where the sign of the derivative becomes positive, after being negative or zero, are the points at which the data set is separated.

We do, however, simplify the modeling process by not considering double peaks. If they are kept by the automated filtering method, then they are modeled as individual ones. If the filtering smooths out their effect, then they are included in some other, larger peak.

3.0.5 Modeling the data.

After separating the peaks, the script uses the turning points to extract data for each one from the original data set (Appendix A) and removes baseflow with the simplest method - by assuming that it is the minimum flow in each of the event

peaks. The removal method is similar to the constant discharge method of Linsley et al. (1958), which assumes uniform baseflow equal to the discharge at the start of the peak. However, in order to account for uncertainty in the filtering, our version of the baseflow separation method uses the minimum discharge from each selected peak, which is not always the starting point (Appendix B).

Because hyetographs are not available for our data, we modeled direct runoff hydrographs by converting them to the pdf form, to account for the volume, but not duration of rainfall (Appendix B).

Three versions of the Levenberg-Marquardt algorithm are applied to the data; one for each model. The LMA is an iterative algorithm which starts out using initial guesses for the parameters to be estimated and updates them at each iteration step while converging on a best fit solution (Levenberg, 1944; Marquardt, 1963; More, 1978; Roweis, 1996; Ranganathan, 2004; Bun, 2009; Nielsen and Madsen, 2010). Further details are provided in Appendices A and B.

Additionally, after simplifying the simple surge function to a 1-variable gamma distribution, the hydrographs are processed again with the LMA to obtain the best-fit times to peak. The best-fit times to peak are later compared with actual times-to-peak from the original data (Appendix B).

The script generates plots of every peak and the best-fit model peak for visual comparison and notes the estimated parameters along with a goodness of fit parameter. All generated results are saved as delimited text files for further analysis and access to combined results (Appendix B).

3.0.6 Measuring goodness of fit.

Coefficients of determination (R^2 values) are not to be interpreted in the same manner for nonlinear regression cases as they are for linear regressions. The nonlinear R^2 is not the squared value of the correlation coefficient, as the $SS_{tot} \neq SS_{reg} + SS_{resid}$.

Instead, we use the Nash-Sutcliffe efficiency coefficient (NSE), which is a modified version of the R^2 calculation, to quantify the goodness-of-fit of our models. The NSE is a criterion commonly used in hydrologic models (Arnold et al., 1993; Peterson and Hamlett, 1998). It has been recommended by the American Society of Civil Engineers (1993) and has been used for measuring goodness of fit in hydrograph models (Servat and Dezetter, 1991; Criss and Winston, 2008).

Moriasi et al. (2007) described the NSE as a normalized statistic that compares the amount of residual variance, or noise, with the measured information. The formula for the NSE is:

$$NSE = 1 - \frac{\sum_{i=1}^{n}(Y_i^{obs} - Y_i^{sim})^2}{\sum_{i=1}^{n}(Y_i^{obs} - Y^{mean})^2} \tag{10}$$

The range of the NSE is from $-\infty$ to 1. An NSE of 1 suggests that the model fits all points perfectly, while an NSE of ≤ 0 suggests that a horizontal mean value line is preferable to the model.

3.0.7 Interpreting results.

First, we analyzed two sets of the processed results. The first was filtered by the difference in discharge, peak duration (12 hours) and the NSE (>0.6). The second was filtered only by the difference in discharge (10 cfs) in order to better account for the filtering algorithm picking insignificant peaks.

After obtaining the coefficient results from the script, we examined them in Excel. We plotted each of the two variables against each other (b vs a, k vs θ) to observe any possible relationships between the variables themselves. If we noticed any relationships, such as in the case of the simple surge function and the gamma distribution, we used our peak-fitting algorithm to evaluate the simpifying function.

We created histograms with 5-hour bin intervals for the one-variable gamma distribution to see the differences in the distributions from river to river. We also visually compared the modeled histograms with histograms from the time-to-peak

values determined by looking at the index of the maximum discharge value of each separated peak.

4. Results

4.1 Overview

To avoid working with poorly processed results we considered whether or not to filter the results by time, goodness-of-fit and discharge. We created Figures 4.1 and 4.2, which show that for the time to peak, the results are not substantially different. Only an increase in frequency is observed. Hence, the data were filtered only by the difference in discharge. Only the results where the NSE > 0 are shown.

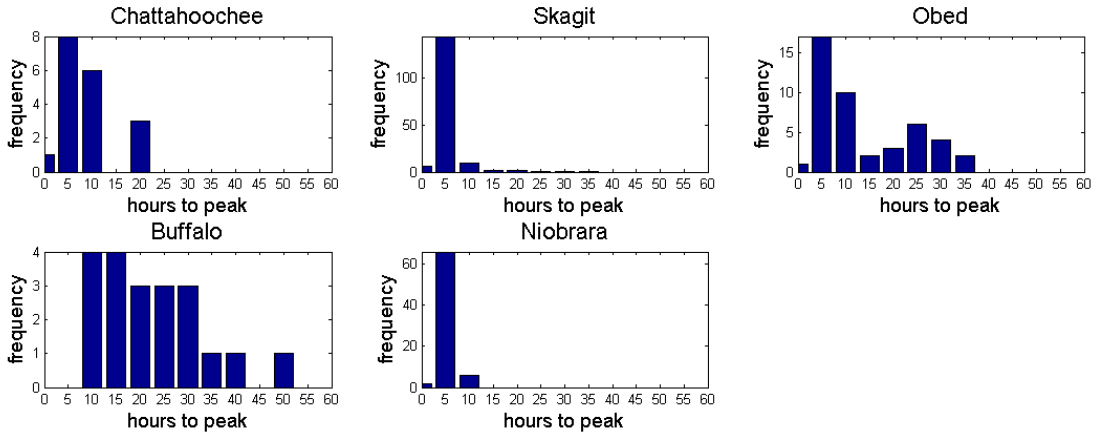

Figure 4.1: Filtered Histogram

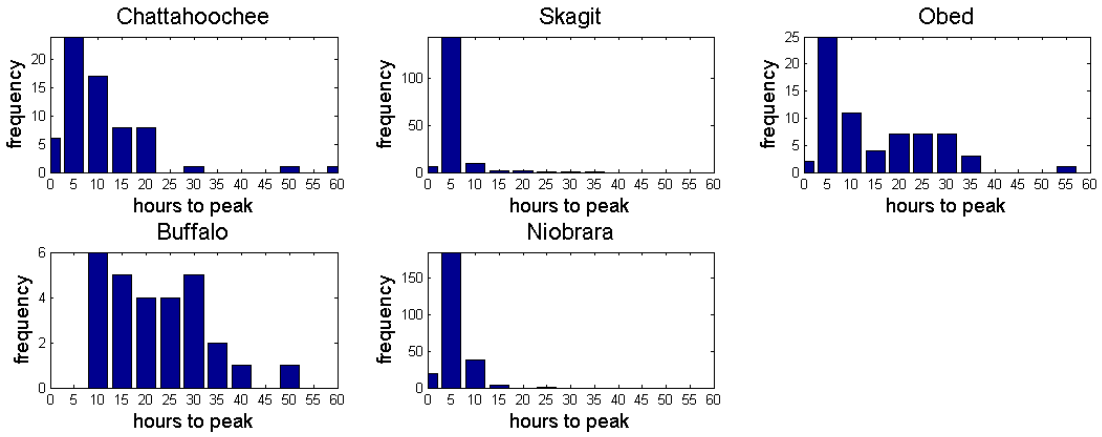

Figure 4.2: Unfiltered Histogram

Figure 4.3 is an example of the peaks that are separated by the algorithm. The asterisks indicate the turning points. Figure 4.4 is an example of the plot of the best-fit model and the actual data generated by the script.

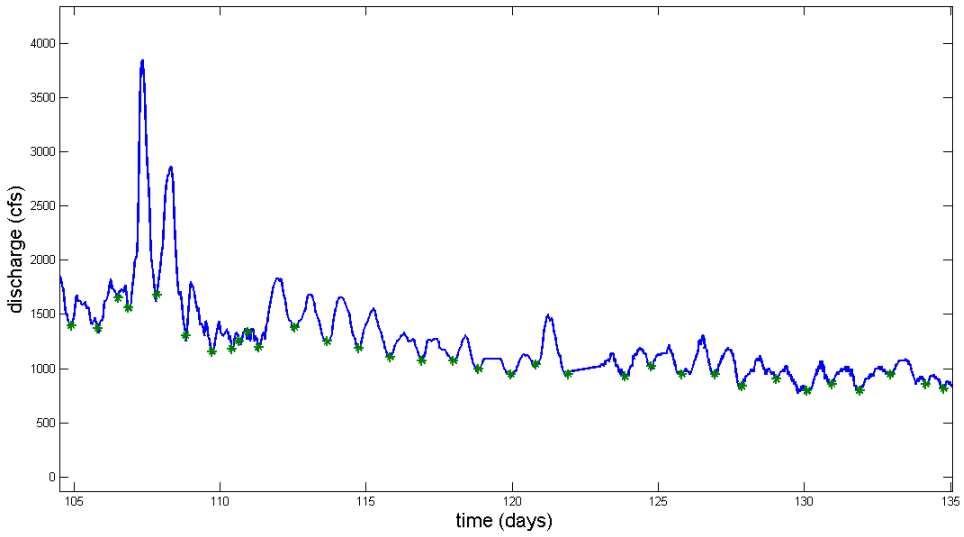

Figure 4.3: Separated Hydrograph Example

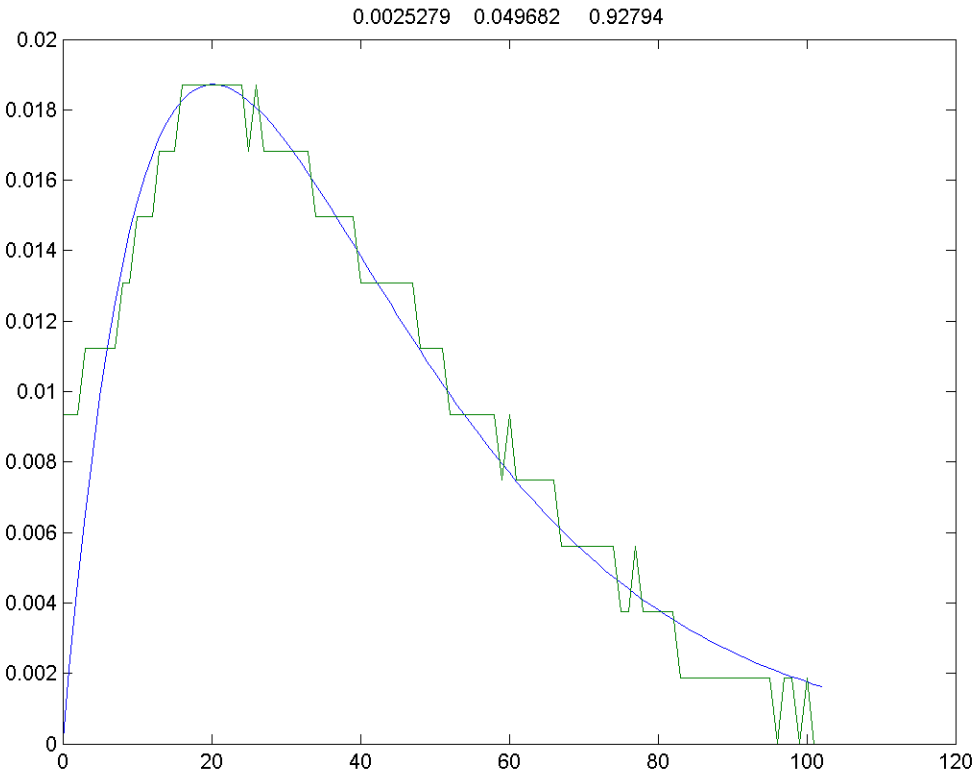

Figure 4.4: Modeled Peak Example

4.2 The Simple Surge Function

Figures 4.5-4.9 show the plots of b vs a for all rivers and table 4.1 shows the processed results. The Buffalo River had the highest mean NSE value, 0.736, and the lowest NSE standard deviation, 0.144. The Chattahoochee River had the lowest NSE of 0.522 and the highest NSE standard deviation of 0.189. For all five rivers, the a-parameters ranged from 3.12E-6 to 2.89E-2. The b-parameters ranged from 1.02E-4 to 1.64E-1. Plots of b vs a for all rivers show that b and a are not independent.

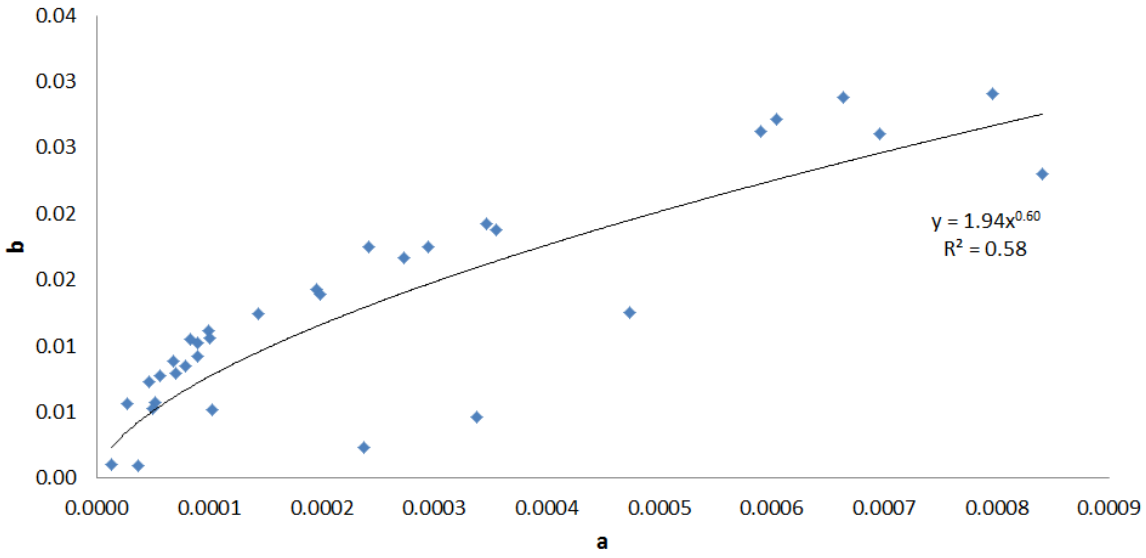

Figure 4.5: Simple Surge a vs b for the Buffalo River

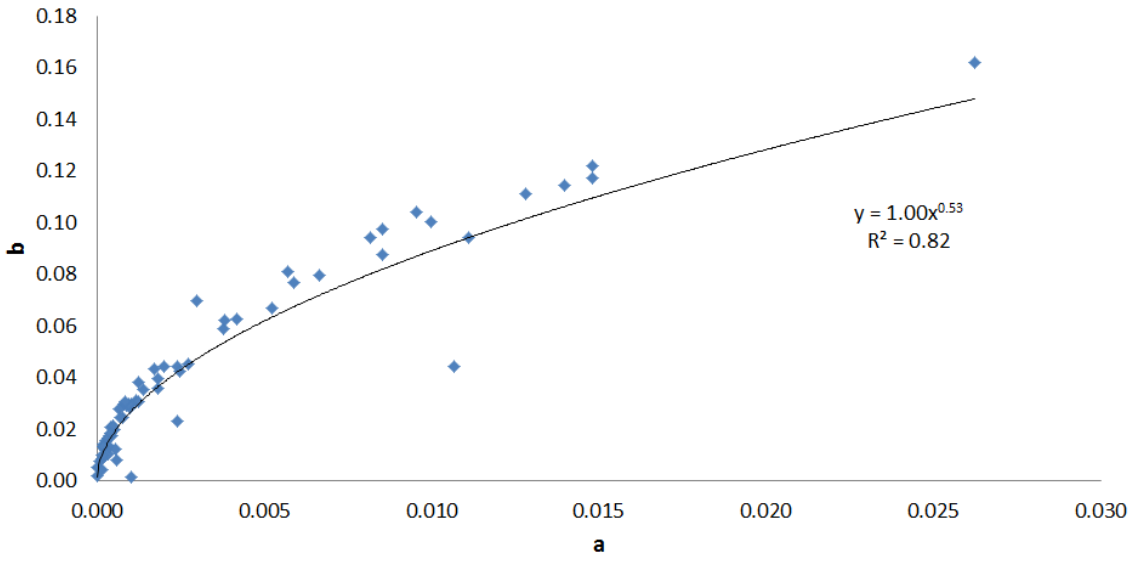

Figure 4.6: Simple Surge a vs b for the Chattahoochee River

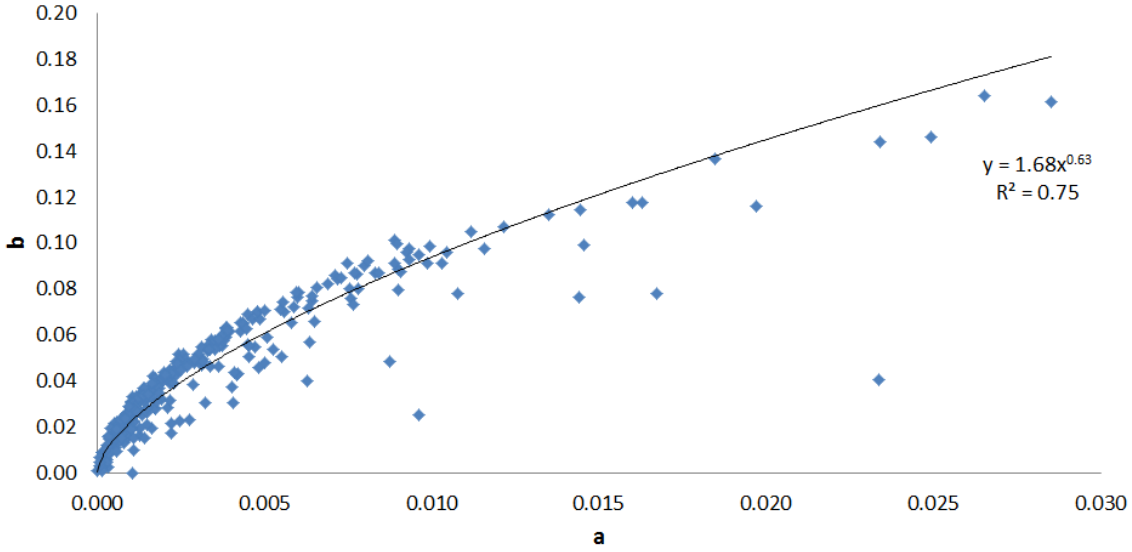

Figure 4.7: Simple Surge a vs b for the Niobrara River

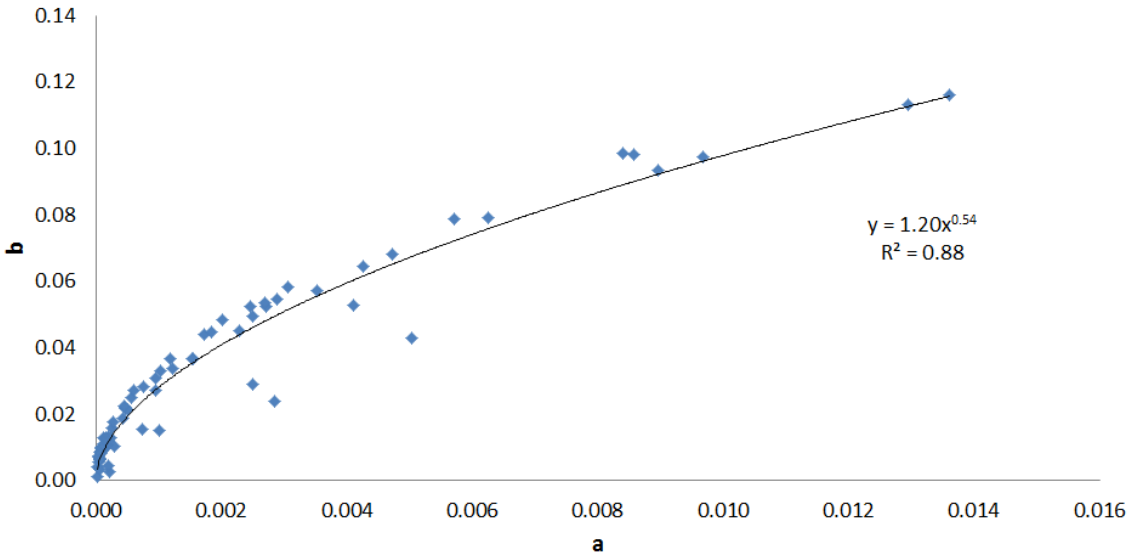

Figure 4.8: Simple Surge a vs b for the Obed River

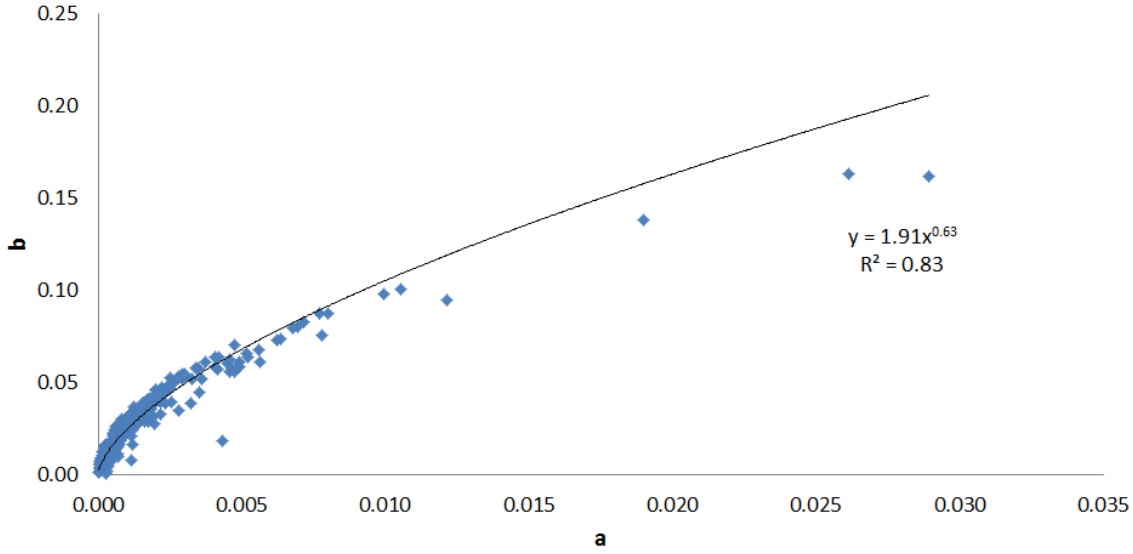

Figure 4.9: Simple Surge a vs b for the Skagit River

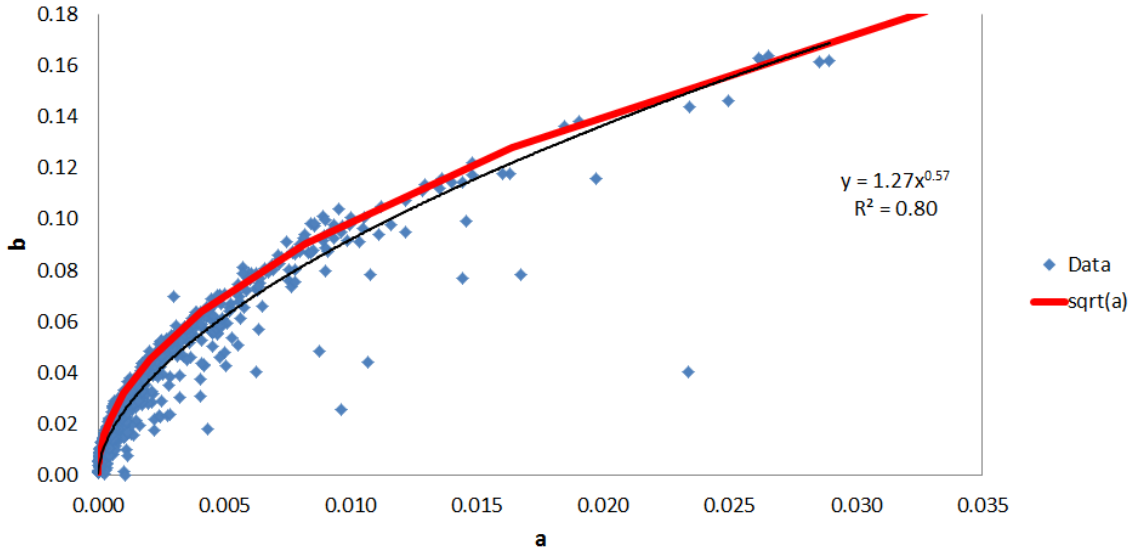

Figure 4.10: Simple Surge a vs b for all rivers

Table 4.1: Results Table for the Simple Surge Function

	Buffalo	Chattahoochee	Niobrara	Obed	Skagit
min a	1.37E-05	3.12E-06	5.40E-06	2.17E-05	3.29E-05
max a	8.41E-04	2.62E-02	2.85E-02	1.36E-02	2.89E-02
mean a	2.53E-04	3.16E-03	3.69E-03	1.90E-03	1.94E-03
stdev a	2.44E-04	4.84E-03	4.49E-03	2.95E-03	2.92E-03
min b	9.62E-04	1.67E-03	1.02E-04	1.23E-03	5.90E-04
max b	2.91E-02	1.62E-01	1.64E-01	1.16E-01	1.63E-01
mean b	1.29E-02	4.05E-02	4.48E-02	3.13E-02	3.33E-02
stdev b	8.19E-03	3.57E-02	2.97E-02	2.87E-02	2.17E-02
min NSE	3.86E-01	1.24E-01	1.09E-01	3.23E-01	2.50E-03
max NSE	9.41E-01	9.02E-01	9.78E-01	9.79E-01	9.79E-01
mean NSE	7.36E-01	5.22E-01	6.49E-01	7.11E-01	6.87E-01
stdev NSE	1.44E-01	1.89E-01	1.71E-01	1.48E-01	1.50E-01

4.3 The Weibull Distribution

Figures 4.11-4.15 show the plots of b vs a for all rivers and table 4.2 shows the processed results. The Skagit River had the highest mean NSE, 0.883, and the lowest NSE standard deviation, 0.110. The Chattahoochee River had the lowest mean NSE of 0.746, while the Niobrara River had the highest standard deviation, 0.143. For all five rivers, the a-parameters ranged from 1.04 to 6.34. The b-parameters ranged from 25.4 to 313. The plots of b vs a for all rivers suggest that the parameters are independent.

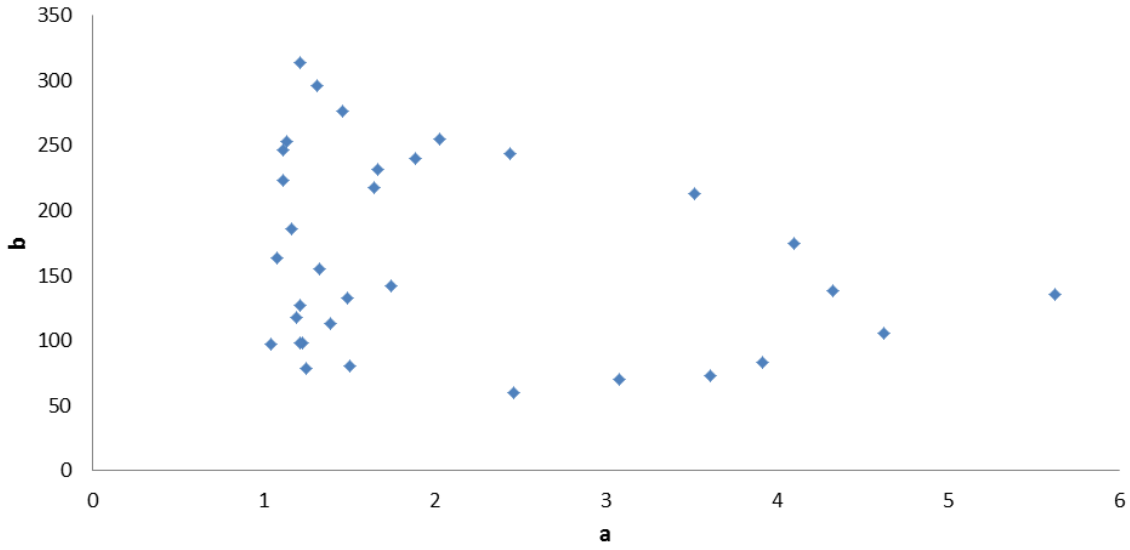

Figure 4.11: Weibull a vs b for the Buffalo River

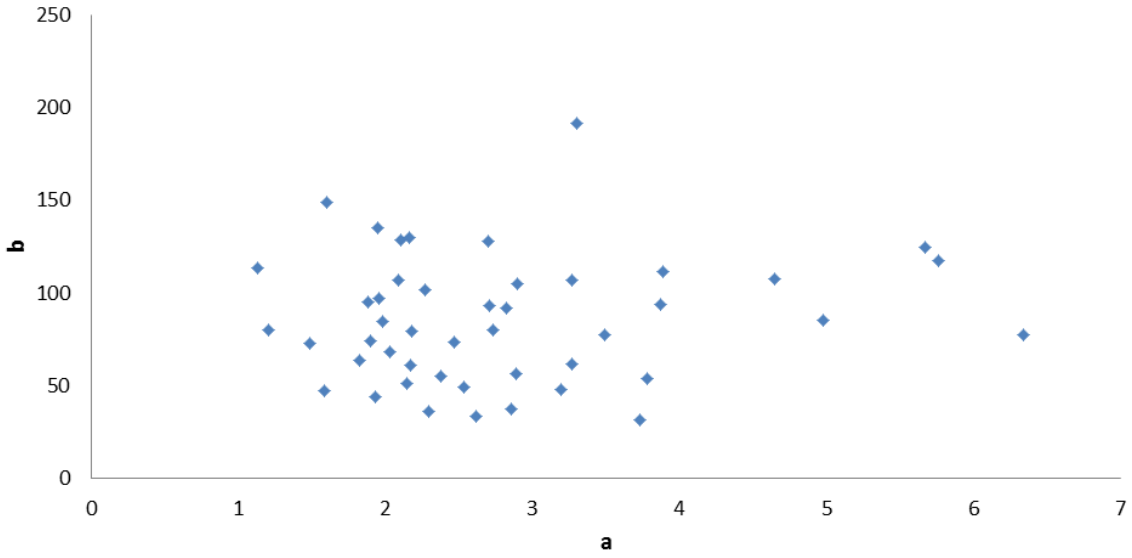

Figure 4.12: Weibull a vs b for the Chattahoochee River

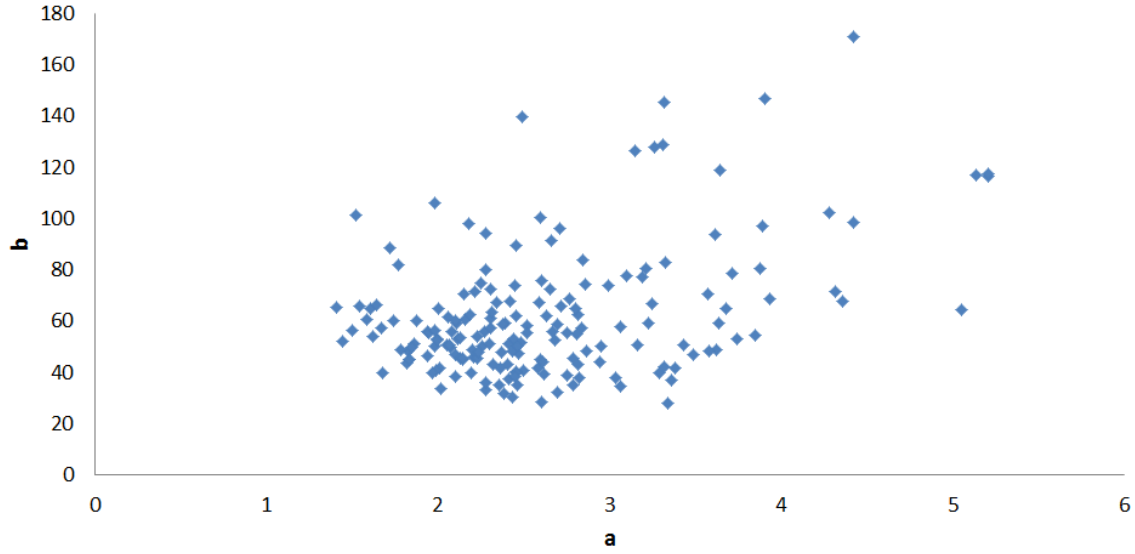

Figure 4.13: Weibull a vs b for the Niobrara River

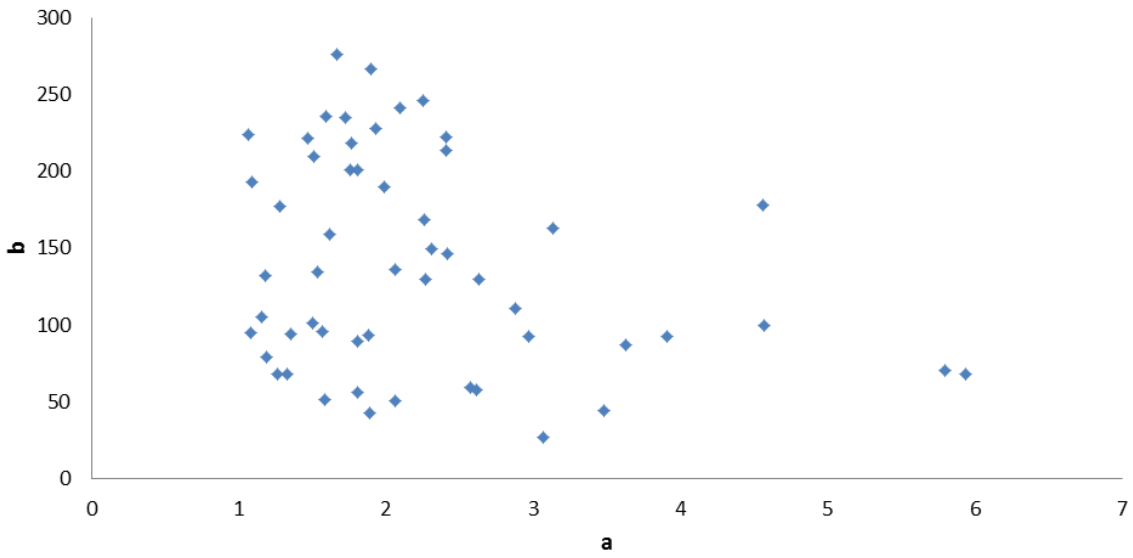

Figure 4.14: Weibull a vs b for the Obed River

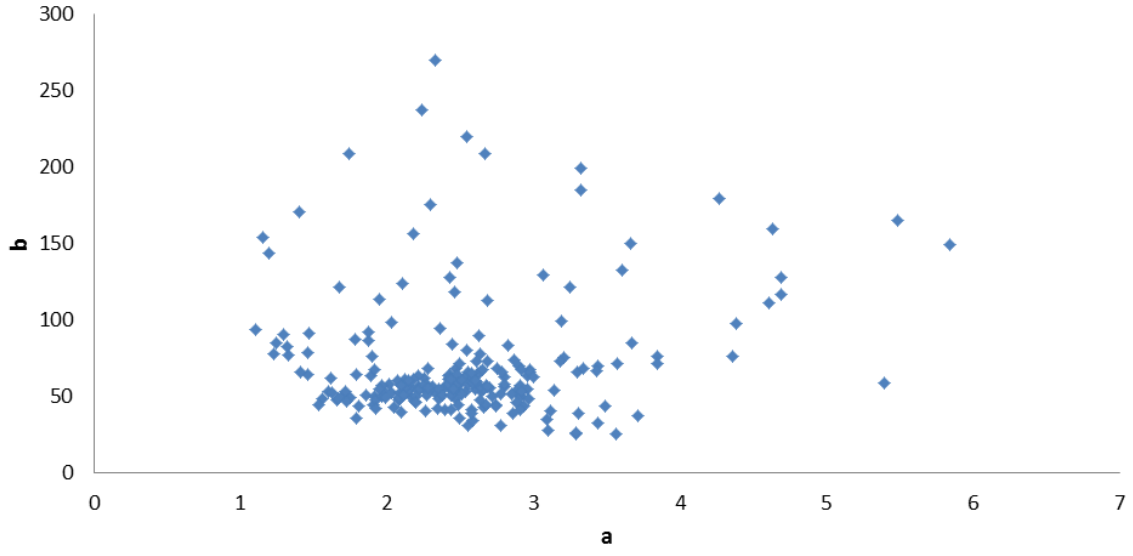

Figure 4.15: Weibull a vs b for the Skagit River

Table 4.2: Results Table for the Weibull distribution

	Buffalo	Chattahoochee	Niobrara	Obed	Skagit
min a	1.04	1.13	1.41	1.07	1.10
max a	5.62	6.34	5.21	5.94	5.84
mean a	2.09	2.80	2.63	2.23	2.53
stdev a	1.26	1.18	0.75	1.09	0.73
min b	59.40	31.80	28.10	26.90	25.40
max b	313.00	191.00	171.00	276.00	270.00
mean b	164.00	85.10	61.70	139.00	70.50
stdev b	74.20	34.20	24.70	68.80	38.70
min NSE	0.51	0.38	0.10	0.41	0.24
max NSE	0.98	0.95	0.99	0.98	1.00
mean NSE	0.87	0.75	0.81	0.84	0.88
stdev NSE	0.13	0.13	0.14	0.11	0.11

4.4 The Gamma Distribution

Figures 4.16-4.20 show the plots of k vs θ for all rivers and table 4.3 shows the processed results. The Buffalo River had the highest mean NSE value, 0.594. The Skagit River had the lowest NSE standard deviation, 0.131 The lowest mean NSE value, 0.521, is attributed to the Chattahoochee River. The Niobrara River had highest NSE standard deviation, 0.189. For all rivers, the k values range from 1.04 to 4.30 and the θ values range from 3.38 to 379. Plots of k vs θ for all rivers suggest that the parameters are independent.

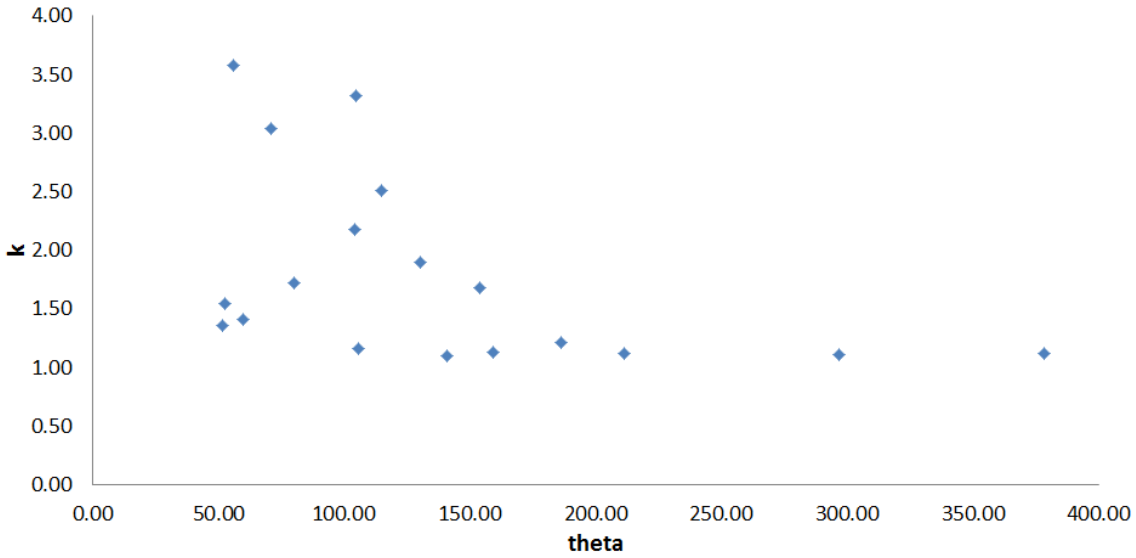

Figure 4.16: Gamma k vs θ for the Buffalo River

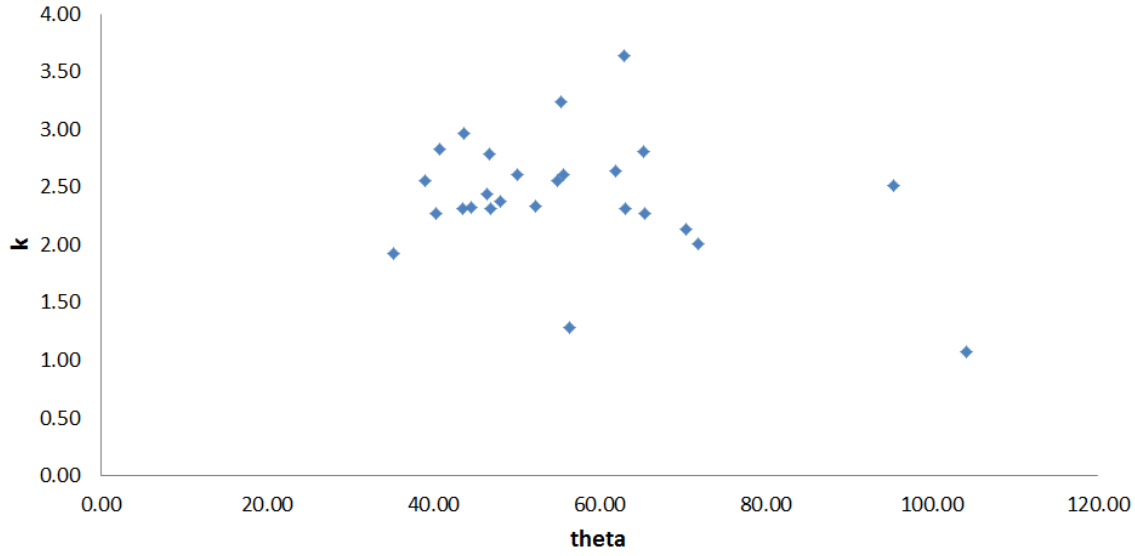

Figure 4.17: Gamma k vs θ for the Chattahoochee River

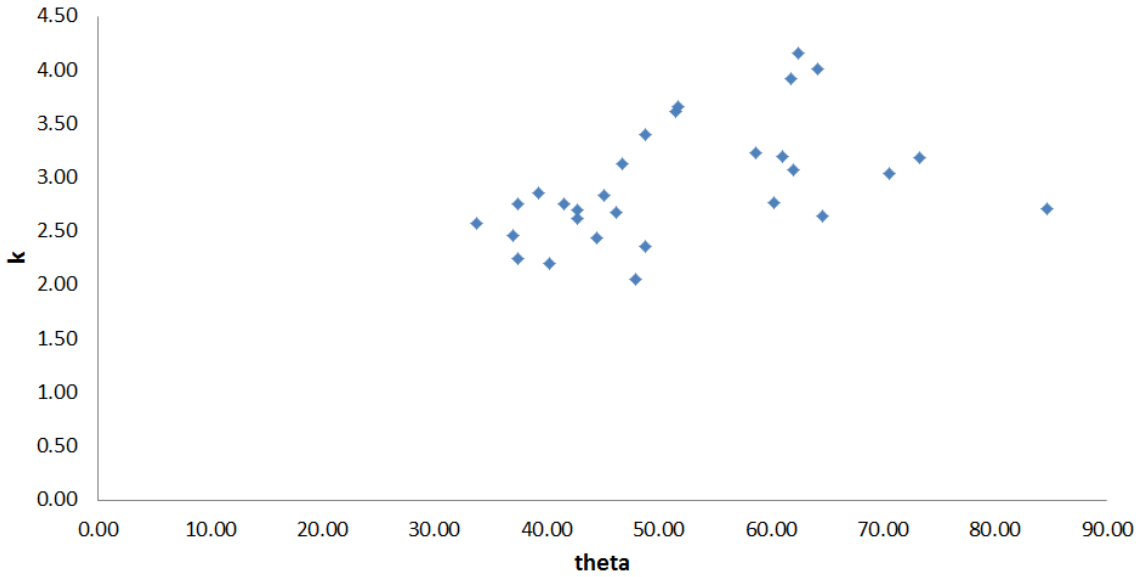

Figure 4.18: Gamma k vs θ for the Niobrara River

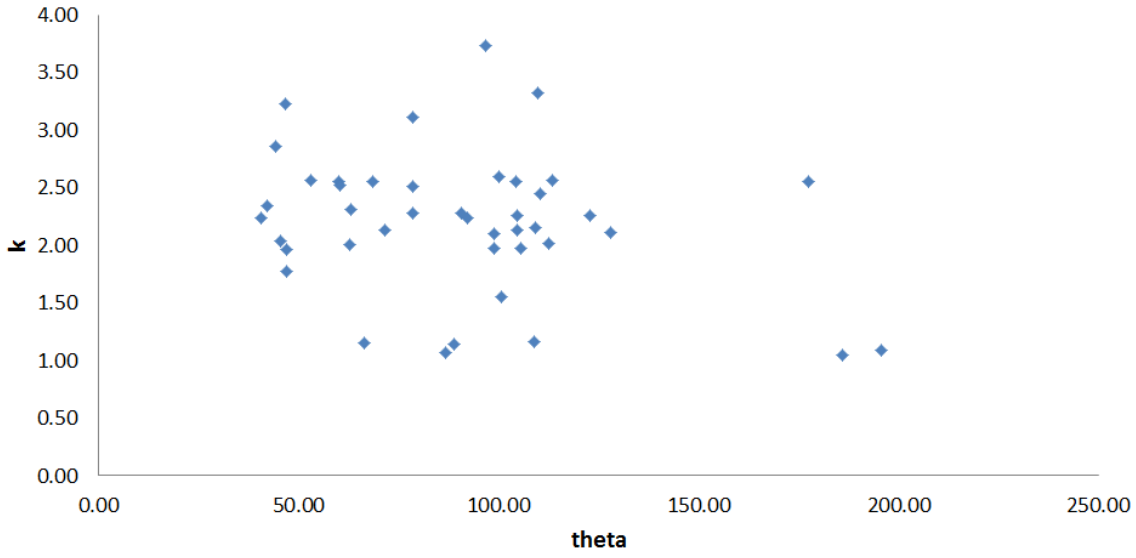

Figure 4.19: Gamma k vs θ for the Obed River

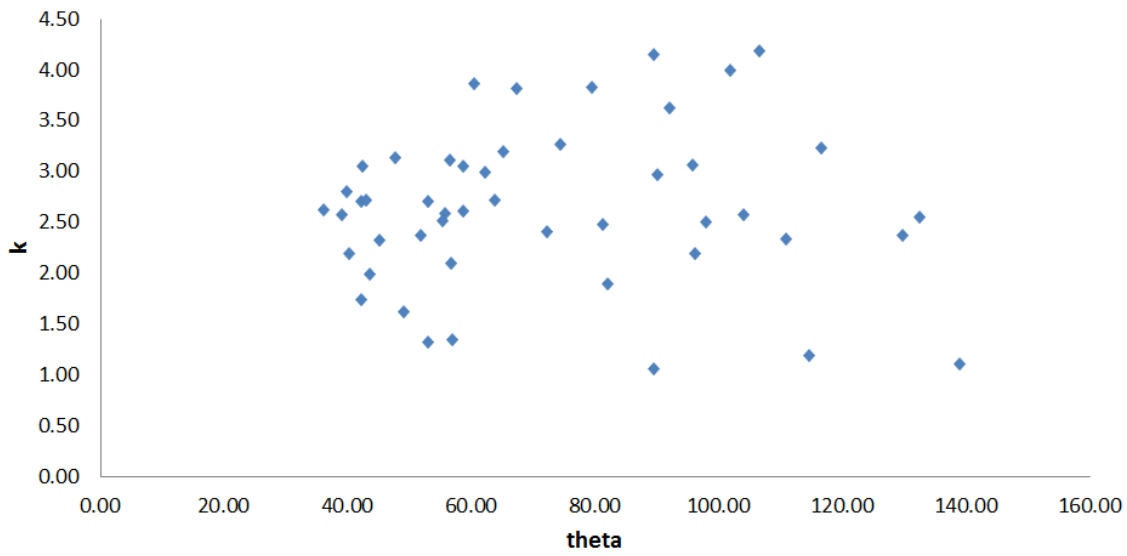

Figure 4.20: Gamma k vs θ for the Skagit River

Table 4.3: Results Table for the Gamma Distribution

	Buffalo	Chattahoochee	Niobrara	Obed	Skagit
min k	1.06	1.08	2.05	1.04	1.05
max k	4.30	3.63	4.16	3.73	4.18
mean k	2.00	2.43	2.94	2.20	2.64
stdev k	0.91	0.52	0.55	0.61	0.78
min θ	51.70	35.30	33.80	40.70	36.20
max θ	379.00	104.00	84.60	196.00	139.00
mean θ	126.00	56.20	52.00	90.70	72.60
stdev θ	75.00	16.30	12.60	36.70	28.00
min NSE	0.51	0.22	0.18	0.39	0.38
max NSE	0.99	0.90	0.96	0.98	0.98
mean NSE	0.84	0.59	0.74	0.82	0.84
stdev NSE	0.14	0.18	0.19	0.13	0.13

4.5 The One-Variable Gamma Distribution

Table 4.4 shows the results of the data processing. The mean one-variable gamma distribution best-fit times to peak ranged from 1.75E-3 to 1.72E-1. The highest mean NSE, 0.651, was for the Buffalo River. The lowest mean NSE, 0.453 and highest standard deviation, 0.243 were for the Niobrara River. The lowest standard deviation, 0.207, was for the Obed River.

Table 4.4: Results for the One-Variable Gamma Distribution

	Buffalo	Chattahoochee	Niobrara	Obed	Skagit
min β	5.04E-03	1.75E-03	9.12E-03	4.59E-03	6.21E-03
max β	3.04E-02	1.62E-01	1.69E-01	1.17E-01	1.72E-01
mean β	1.42E-02	4.47E-02	5.83E-02	3.45E-02	4.19E-02
stdev β	7.32E-03	3.55E-02	2.98E-02	2.92E-02	2.14E-02
min NSE	3.74E-02	9.39E-03	8.33E-04	6.12E-03	6.24E-03
max NSE	9.05E-01	8.95E-01	9.12E-01	9.79E-01	9.41E-01
mean NSE	6.51E-01	4.57E-01	4.53E-01	6.46E-01	4.88E-01
stdev NSE	2.22E-01	2.23E-01	2.43E-01	2.07E-01	2.24E-01

4.6 Comparisons

4.6.1 Model comparison.

Table 4.5 shows the mean NSE values and NSE standard deviations for all five rivers. The Weibull distribution had the highest mean and lowest standard deviation NSE values for all rivers. The full gamma and the 1-variable gamma distributions had the lowest NSE values and the highest NSE standard deviations.

4.6.2 One-variable gamma and time to peak comparison.

Figures 4.21 and 4.22 show histograms of the hours to peak for the modeled and observed peaks. The best-fit modeled one-variable gamma distribution values, which are inverses of the time to peak, were compared against actual times to peak for the same hydrographs. Qualitatively, there does not appear to be substantial variation between the two approaches.

Table 4.5: Results for the NSE Values for All Models

	Buffalo	Chattahoochee	Niobrara	Obed	Skagit
Simple mean NSE	0.74	0.52	0.65	0.71	0.69
Simple stdev NSE	0.14	0.19	0.17	0.15	0.15
Weibull mean NSE	0.87	0.75	0.81	0.84	0.88
Weibull stdev NSE	0.13	0.13	0.14	0.11	0.11
Gamma mean NSE	0.84	0.59	0.74	0.82	0.84
Gamma stdev NSE	0.14	0.18	0.19	0.13	0.13
1-Var Gamma mean NSE	0.65	0.46	0.45	0.65	0.49
1-Var Gamma stdev NSE	0.22	0.22	0.24	0.21	0.22

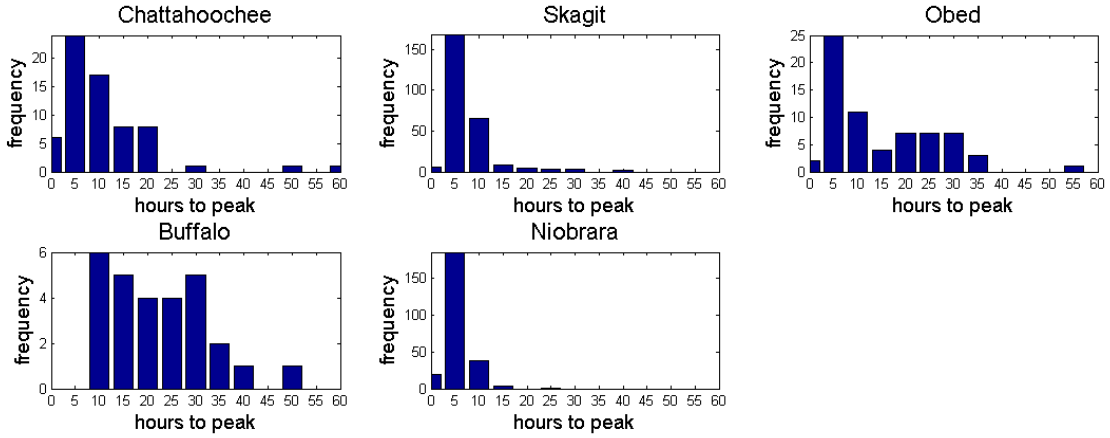

Figure 4.21: Histogram of One-Variable Gamma Times to Peak

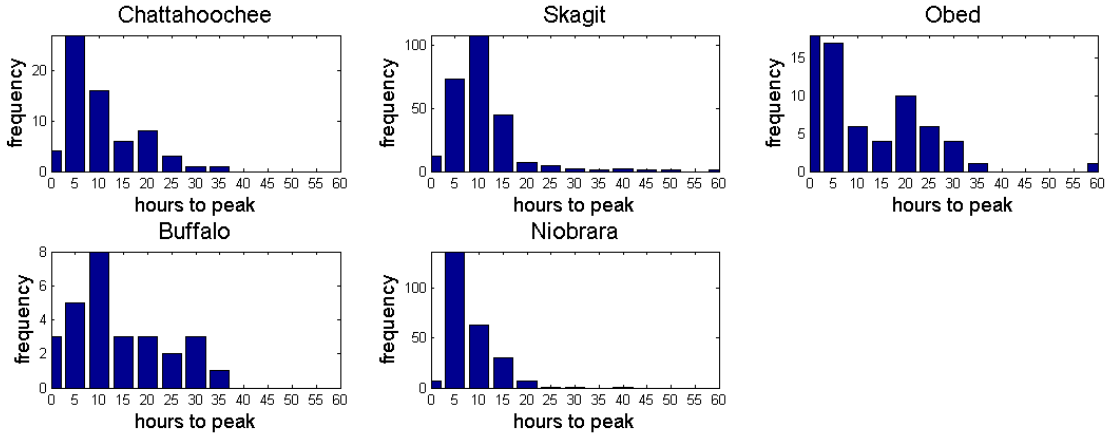

Figure 4.22: Histogram of Observed Times to Peak

5. Discussion

5.1 Model Comparsion

We can judge the quality of the models in two ways. First is by examining the NSE values and their standard deviations. Second is by plotting the estimated parameters against each other. Plots of the parameters show whether or not one of the variables can be represented as a function of the other. If the data do not appear independent, then it may be possible to reduce the two-parameter model to a one-parameter model, simplifying the problem.

The Weibull distribution had independent parameters (Figures 4.11-4.15) and the highest mean NSEs along with the lowest NSE standard deviations for all rivers (Table 4.5). Therefore, the Weibull distribution is the distribution that best represented hydrograph peaks for this study.

The gamma distribution had independent parameters (Figures 4.16-4.20) and lower mean NSE and higher NSE standard deviations (Table 4.5) than the Weibull distribution. The reason for poor fits is that the Levenberg-Marquardt algorithm does not work well for functions which are extremely sensitive to initial guesses.

The simple surge function mean NSE and NSE standard deviaton values ranged between those of the Weibull distribution and those of the gamma (Table 4.5), but had non-randomly distributed parameters (Figures 4.5-4.9). In the next section we show that the simple surge function can be reduced to a 1-variable form of the gamma distribution.

5.2 Simple Surge Function

Plotting the b vs a values of the simple surge function for all rivers led to an unexpected result.

We assumed that a and b would characterize basin properties and, therefore, would be independent of each other. The results showed otherwise. All b versus a plots showed a clear power function trend. The set of coefficients and exponents of the fitted power function (Figure 4.10) falls within a simple band of $b = a^{1/2}$. An overlay of this simplified square root curve is shown in figure 4.10.

Because the variation in the second parameter could be reasonably explained by the first parameter, the simple surge function can be reduced to

$$f(x) = \beta^2 x e^{-\beta x} \tag{11}$$

Now, recall that for the surge function, the time to peak (t_p) is the $\frac{p}{b}$. In this case, p is always 1, so the time to peak is reduced to $\frac{1}{b}$, which is also equal to β in (3). We therefore decided to model this function as a one-variable gamma distribution with the parameter-estimation algorithm to determine its suitability for modeling hydrograph peaks.

By carefully examining the newly derived function, we noticed similarities with the gamma distribution itself. Consider a case of the gamma distribution where there are still two parameters k and θ. However, the k parameter is always held constant as 2. This substitution reduces the gamma distribution to:

$$f(x) = \frac{1}{\theta^2} x e^{-\frac{1}{\theta}x}, \tag{12}$$

exactly the same as the reduced form of the simple surge function (2), where $\beta = \frac{1}{\theta}$. The simple surge function, therefore, is related to the gamma distribution.

Reducing the surge function to a version of the gamma distribution explains the shape of the hydrograph with only one parameter, the time to peak. Because the modeled data are normalized only by the discharge, the hydrographs are direct runoff hydrographs and not instantaneous unit hydrographs. By assuming uniform rainfall across the whole watershed, it appears that the differences between times to peak for the same river depend only on the duration of effective rainfall, since watershed parameters are invariant. The rainfall volume has already been accounted for when converting the direct runoff hydrograph to its pdf form. The uniform rainfall assumption however, has been shown not to be robust, especially for very large watersheds (Ramirez, 2000). Therefore, it is more likely that the time to peak value at a single station is influenced by the duration and distribution of rainfall across the watershed.

5.3 Modeling Methods

There are three main pitfalls to modeling hydrographs using the presented methods: the filtering method, the initial parameter guesses, and the staring Levenberg parameter. Although the initial parameter guesses and the starting Levenberg parameter are all in the same part of the algorithm, they appear to be independent. The initial guesses represent the starting point of the algorithm with respect to the original data, while the Levenberg parameter influences the initial scaling of the impact of the curvature on the on the solution.

Unfortunately, there is not a unique set of starting initial guesses and Levenberg parameters for each of the model functions. So, some of the higher order functions are extremely sensitive to both, and therefore, each model function does not comprehensively model the same number of peaks as another model function. Certain instances of the functions can lead to a divergent solution, which would be indicated by the peak not being modeled. Another function, however, can model that same peak without any problems. This appears to be the reason behind the gamma

distribution having high standard deviations (Table 4.3). Considering its previous use for hydrograph models, it is therefore likely that bad initial guesses are to blame.

We can consider the one-variable version of the gamma distribution as a daughter function, and the complete gamma distribution as a mother function. From this, it would be expected that the mother functions will always produce better fitting parameters since they can be subject to more variation due to the presence of the extra parameter. However, the observed higher NSE values for the gamma distribution only occured for three out of the five rivers (Table 4.5). As mentioned previously, the extreme sensitivity of the gamma distribution to initial guesses is the likely factor.

Because both the full gamma and Weibull distributions have an extra parameter over the simple surge function (which has two parameters, but where one parameter is related to the other by a power function and a scaling factor) and the one-variable form of the gamma distribution, it is justified to expect that they would produce better fitting parameters. However, if the the goodness of fit of a one-parameter model is not significantly less than the goodness of fit for a two-parameter model, then for the sake of model parsimony, the one-parameter model may be preferred. Such is the case with the presented one-variable form of the gamma distribution, which only depends on the time to peak, and which produced better average NSE values for two out of the five rivers than the full gamma distribution (Table 4.5).

Altough we are not suggesting that the 1-variable gamma distribution is a superior model than the Weibull distribution, which had the highest mean NSE values and the lowest standard deviations, the fact that it only relies on one intuitive parameter to generate the pdf form of a hydrograph, which can then be converted into a DRH or UH, makes it an attractive function for estimation purposes.

5.4 Applications to Rivers

We observed that rivers have distinct distributions of time to peak values (Figure 4.21). The similarities of distributions and their possible relationship to some basin properties ought to be examined.

An advantage of the automated method is that it can obtain the best-fit parameters for any well-behaving function and therefore, break down any duration of discharge data into individual peaks. These results could then be subject to statistical analysis, such as generating histograms of the estimated parameters.

While our comparsion of the modeled times to peak and the actual ones could have easily been extracted via a single MATLAB command, parameter estimation would still be required for determining the distributions of multiple-parameter functions.

5.5 Dimensional Analysis

The non-pdf form of the simple surge function (2) accounts for the volume of water flowing past the gage per unit time and can be described in terms of two fundamental dimensions: length [L] and time [T].

$$Q\left[\frac{L^3}{T}\right] = a\left[\frac{L^3}{T^2}\right] x\,[T]\, e^{-b\left[\frac{1}{T}\right]x[T]} \tag{13}$$

Whenever we normalize the simple surge function into its pdf form, which we use for modeling, we account for the total total volume of water passing by the gage throughout the recorded event interval. Accounting for the total volume of water removes the $[L^3]$ term from both sides of the equation.

$$Q\left[\frac{1}{T}\right] = a\left[\frac{1}{T^2}\right] x\,[T]\, e^{-b\left[\frac{1}{T}\right]x[T]} \tag{14}$$

We can determine the units of the estimated parameters a and b by inspection. The units of a are: $\left[\frac{1}{T^2}\right]$, and the units of b are $\left[\frac{1}{T}\right]$.

The relationship between the units of a and b parallels the relationship between the θ parameters in the gamma distribution when the k term is held constant at 2 (12). Additionally, the relationship of the units between a and b is similar to the trend seen when we calculated the best-fit power functions between the a and b coefficients for the simple surge function (Table 5.1).

Table 5.1: Best-fit Power Functions of b vs a for the Simple Surge Function

River	Best-fit equation
Buffalo	$1.9x^{0.60}$
Chattahoochee	$1.0x^{0.53}$
Niobrara	$1.7x^{0.63}$
Obed	$1.2x^{0.54}$
Skagit	$1.9x^{0.63}$

5.6 Future Work

It would be useful to improve the filtering and LMA algorithms to model the gamma (with an implementation of a guess-testing algorithm), Weibull and beta distributions across the hydrologic regions of the United States. Additonally, considering that there are multiple relatively simple ways of separating baseflow along with a number of digital filter methods, it would be of interest to evaluate the impact of baseflow separation methods on the resulting best-fit parameters.

6. Conclusion

It has been shown that parameter estimation methods can be applied to obtain best-fit parameters for functions modeling hydrographs, and that automation is possible, given a robust filtering algorithm and a reasonable starting Levenberg parameter and initial guesses. We verified the superiority of the Weibull distribution for modeling hydrograph peaks. We also showed that the surge function is actually a scaled version of the two-parameter gamma distribution, which can be reduced to a one-variable version, with the variable being the time to peak discharge. We discovered that modeling the gamma distribution with the Levenberg-Marquardt algorithm is extremely sensitive to initial guesses, which would require modifying the modeling method to obtain better results.

References

Ahmed, N. (1990). Weibull distribution and natural hydrograph. In *Hydraulic Engineering*, pp. 227–232. ASCE.

Arnold, J., P. Allen, and G. Bernhardt (1993). A comprehensive surface-groundwater flow model. *Journal of Hydrology 142*(1-4), 47–69.

ASCE Task Committee on Definition of Criteria for Evaluation of Watershed Models of the Watershed Management Committee and Irrigation Drainage Division (1993). Criteria for evaluation of watershed models. *Journal of Irrigation and Drainage Engineering 119*(3), 429–442. http://link.aip.org/link/?QIR/119/429/1.

Bhunya, P., R. Berndtsson, C. Ojha, and S. Mishra (2006). Suitability of gamma, chi-square, weibull, and beta distributions as synthetic UH. *J. of Hydro 334*, 28–38.

Bhunya, P., S. Panda, and M. Goel (2011). Synthetic Unit Hydrograph Methods: A Critical Review. *Open Hydrology Journal 5*, 1–8.

Brodie, R. and S. Hostetler (2005). A review of techniques for analysing baseflow from stream hydrographs. In *Proceedings of the NZHS-IAH-NZSSS 2005 conference*, Volume 28.

Bun, M. (2009). Applications of the Levenberg-Marquardt Algorithm to the Inverse Problem. http://www.math.washington.edu/∼reu/papers/2009/mark/reu paper.pdf.

Criss, R. and W. Winston (2008). Properties of a diffusive hydrograph and the interpretation of its single parameter. *Mathematical Geosciences 40*(3), 313–325.

Dooge, J. (1959). A general theory of the unit hydrograph. *Journal of Geophysical Research 64*(2), 241–256.

Edson, C. (1951). Parameters for relating unit hydrographs to watershed characteristics. *Trans. Am. Geophys. Union 32*(4), 591–596.

Espey, W. and D. Winslow (1968). The effects of urbanization on Unit Hydrograph, in effects of watershed, Houston, Texas, 1964-67. *Tracor, Inc. Austin, Texas. Tracor Documents*.

Fang, X., W. H. Asquith, C. A. Garcia, T. G. Cleveland, D. Thompson, and R. Malla (2005). Literature Review on Timing Parameters for Hydrographs. Technical report, Lamar University. http://hdl.handle.net/2346/22806.

Gordon, S. (2006). Fitting Surge Functions to Data. *PRIMUS 16*(1), 81–96.

He, X. (2004). Comparison of Gamma, Rayleigh, Weibull and NRCS models with observed runoff data for central Texas small watersheds. Master's thesis, University of Houston.

Levenberg, K. (1944). A method for the solution of certain nonlinear problems in least squares. *Quart. Appl. Math 2*(2), 164–168.

Linsley, R., M. Kohler, J. Paulhus, and J. Wallace (1958). *Hydrology for Engineering*. McGraw Hill, New York.

Marquardt, D. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics 11*(2), 431–441.

More, J. (1978). The Levenberg-Marquardt algorithm: implementation and theory. *Numerical analysis*, 105–116.

Moriasi, D., J. Arnold, M. Van Liew, R. Bingner, R. Harmel, and T. Veith (2007). Model evaluation guidelines for systematic quantification of accuracy in watershed simulations. *Transactions of the ASAE 50*(3), 885–900.

Nadarajah, S. (2007). Probability models for unit hydrograph derivation. *Journal of Hydrology 344*(3-4), 185–189.

Nash, J. (1959). Systematic determination of unit hydrograph parameters. *Journal of Geophysical Research 64*(1), 111–115.

Nielsen, H. B. and K. Madsen (2010, August). *Introduction to Optimization and Data Fitting*. Informatics and Mathematical Modelling, Technical University of Denmark, DTU. http://www2.imm.dtu.dk/pubdb/p.php?5938.

Peterson, J. and J. Hamlett (1998). Hydrologic calibration of the swat model in a watershed containing fragipan soils. *JAWRA Journal of the American Water Resources Association 34*(3), 531–544.

Quinn, T. and R. Deriso (1999). *Quantitative fish dynamics*. Oxford University Press, USA.

Ramirez, J. (2000). Prediction and modeling of flood hydrology and hydraulics. *Inland Flood Hazards: Human, Riparian and Aquatic Communities. Cambridge University Press, Cambridge*.

Ranganathan, A. (2004). The Levenberg-Marquardt Algorithm. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.10.2258&rep=rep1&type=pdf.

Ricker, W. (1954). *Stock and recruitment*. Fisheries Research Board of Canada.

Roweis, S. (1996). Levenberg-Marquardt optimization. http://www.cs.toronto.edu/~roweis/notes.html.

Servat, E. and A. Dezetter (1991). Selection of calibration objective fonctions in the context of rainfall-ronoff modelling in a sudanese savannah area/sélection de fonctions critères dans le cadre d'une modélisation pluie-débit en zone de savane soudanaise. *Hydrological Sciences Journal 36*(4), 307–330.

Singh, V. and P. Chowdhury (1985). On Fitting Gamma Distrubution to Synthetic Runoff Hydrographs. *Nordic Hydrology 16*(3), 177–192.

Smith, S. (1999). *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing.

Snyder, F. (1938). Synthetic unit hydrographs. *Trans. Amer. Geophysical Union 19*, 447–454.

USDA-SCS (1972). National Engineering Handbook, Section 4: Hydrology. *US Dept of Agriculture, Soil Conservation Service (USDA SCS), Washington DC, USA*.

Winter, T. (2001). The concept of hydrologic landscapes. *Journal of the American Water Resources Association 37*(2), 335–349.

Wolock, D., T. Winter, and G. McMahon (2004). Delineation and evaluation of hydrologic-landscape regions in the united states using geographic information system tools and multivariate statistical analyses. *Environmental Management 34*, 71–88.

Appendices

Appendix A: The Levenberg-Marquardt Algorithm

This section of the appendix explains the mathematical concepts of numerical optimization and its uses. The section also explains the derivation of the Levenberg-Marquardt algorithm and the algorithms on which it is based - the gradient descent method and the Gauss-Newton method.

Overview.

Nonlinear problems are ubiquitous in science and engineering. Solving nonlinear problems generally requires iterative algorithms, whose goal is to determine better-fitting parameters at each iteration step while being driven by some objective function.

Usually the goal is to minimize the objective function, which is done by the method of least-squares. Minimizing the squared differences of the residuals yields a result which maximally reduces the error between the two data sets, so the best-fitting model parameters are those that lead to the minimum value of the residual-square sums.

The Levenberg-Marquardt algorithm is a standard algorithm used to solve nonlinear optimization problems. The main reasons for its success include its efficiency when dealing with small sized data sets, and it being an optimal combination between the gradient descent and the Gauss-Newton methods (Roweis, 1996; Ranganathan, 2004). Therefore, the gradient descent and Gauss-Newton algorithms are derived as the predecessor algorithms to emphasize the advantages of the LMA.

Nonetheless, the algorithm is not perfect. Its effectiveness greatly varies depending on the size of the matrix to be inverted. It is efficient for problems with low numbers of parameters and very inefficient for large problems requiring complicated matrix inversions (Roweis, 1996).

Some general remarks.

The goal of the algorithms presented below is to calculate an updated set of parameters to be optimized. Usually this is done by solving for an iterative update step, δ (Bun, 2009). The δ is a vector with the same dimensions as the number of parameters to be estimated. It's computed by going through the algorithm using parameters from the end of the last iteration step. The values of δ are then added to the old parameters to make an updated set of improved parameters.

The derivations and explanations presented in the subsections below are of Madsen and Nielsen (2010). Their notation, with some slight modifications, has been used for its concision. Additionally, since the aim of all of the methods is to solve for δ, most of the equations will be written in such a form as opposed to solving for the new set of optimized points, as presented in some of the literature (Roweis, 1996; Ranganathan, 2004).

It should also be noted that sometimes, as is impemented in the MATLAB script for this project, the Jacobian matrices are presented as the transposes of the Jacobians in the explanation below. This has no impact on the outcome of the algorithm, just on the operations within the script.

Concepts of optimization.

The objective function is usually written as:

$$f(x) = \sum_{i=1}^{n} r_i^2 = ||r(x)||^2 \tag{1}$$

Where: $r(x)$ is the residual vector between the model and the actual data, and $||\bullet||$ represents the L2-norm.

Differentiating this equation will produce a term of 2 in many of the equations derived below (Nielsen and Madsen, 2010), so another common notation is to rewrite

the objective function as :

$$f(x) = \frac{1}{2}||r(x)||^2 \tag{2}$$

Now, we linearize the objective function, so that

$$r = y - F(x) \tag{3}$$

Where: y is the original data vector and $F(x)$ is the output of the model function with the estimated parameters at the last iteration step. The matrix of first-order partial derivatives with regard to each of the parameters to be optimized is the Jacobian matrix:

$$(\nabla f(x))_j = \frac{\partial f}{\partial x_j}(x) = \sum_{i=1}^{m} r_i(x)\frac{\partial r_i}{\partial x_j}(x) = J(x) = J \tag{4}$$

The Jacobian is an $m \times n$ matrix, where m represents the number of parameters and n represents the number of needed data points.

Then, the gradient is:

$$\nabla f(x) = J(x)^T r(x) = J^T r \tag{5}$$

Then, the matrix of second-order mixed partial derivatives is the Hessian matrix:

$$\frac{\partial^2 f}{\partial x_j \partial x_k}(x) = \sum_{i=1}^{m}\left(\frac{\partial r_i}{\partial x_j}(x)\frac{\partial r_i}{\partial x_k} + r_i(x)\frac{\partial^2 r_i}{\partial x_j \partial x_k}(x)\right) \tag{6}$$

Which is equivalent to:

$$\nabla^2 f(x) = J(x)^T J(x) + \sum_{i=1}^{m} r_i(x)\nabla^2 r_i(x) \approx J^T J \tag{7}$$

The Hessian ends up being a square $m \times m$ matrix.

For the presented methods, only the local minimizers are truly considered, as the algorithms have no definite way of knowing whether a minimizer is global or local.

In order to verify if the minimizer is indeed global, multiple runs of the algorithm would need to be conducted across the possible solution space.

Taylor series approximations are used in the derivation of the presented methods. A first-order Taylor series approximation, used for the steepest descent method, assumes that the function has continuous second derivatives and is given by:

$$f(x+h) \approx f(x) + h^T \nabla f(x) \tag{8}$$

A second-order Taylor series approximation, used for the Gauss-Newton and the LMA method, assumes that the function has continuous third derivatives. It is given by:

$$f(x+h) \approx f(x) + h^T \nabla f(x) + \frac{1}{2} h^T \nabla^2 f(x) h \tag{9}$$

The steepest descent method.

The method of steepest descent, also called gradient descent, is the simplest method for iteratively converging on a solution. Its derivation can be shown from a first-order Taylor series approximation (8).

Recall that a function can be a minimizer at a point where $\nabla f = 0$. This now yields:

$$\delta = -\lambda \nabla f = -\lambda J^T r \tag{10}$$

This method is very straightforward - it starts with an initial guess of a minimum and then updates the parameters for the next iteration step by subtracting a constant times the gradient of the function, δ, from the values from the previous iteration step. The constant, λ, can be either a specified step value which remains the same throughout the iterative process, or it can be adjusted at each iteration step for quicker convergence.

Unfortunately, the implementation of this method is the opposite of being optimal. The method takes large steps where the gradients are high, when in reality, small steps are desired at high gradients to prevent overshooting. The converse of this is true as well - when the gradients are small, the method takes very small steps and requires long convergence times (Roweis, 1996).

Newton and Gauss-Newton algorithms.

Newton's method is derived from the second-order Taylor approximation (9), to obtain:

$$\delta = H^{-1}\nabla f = H^{-1}J^T r \tag{11}$$

In reality, the true Hessian matrix (6) is generally never calculated. Instead, an assumption is made about the area where the linearization of f is considered to hold true. This changes the value of H from the full Hessian (6) to an approximated version, defined by $J^T J$, the Jacobian squared, which due to the nature of matrix multiplication, approximates the mixed partial derivatives desired in the Hessian. If this replacement is made, Newton's Method becomes the Gauss-Newton Method (Bun, 2009), which yields:

$$\delta = (J^T J)^{-1}\nabla f = (J^T J)^{-1}J^T r \tag{12}$$

The Levenberg-Marquardt algorithm.

Levenberg (1944) combined the gradient descent method and the Gauss-Newton method into one equation by adding one more parameter. He replaced the $(J^T J^{-1})$ by $(J^T J + \lambda I)$, where λ is known as the Levenberg parameter and I is the identity matrix. The result of this change affects the magnitude of importance of the Hessian (Ranganathan, 2004).

Levenberg's Method can be written as:

$$\delta = (J^T J + \lambda I)^{-1} \nabla f = (J^T J + \lambda I)^{-1} J^T r \tag{13}$$

The λ is a scalar value that is adjusted depending on the behavior of the error between the iteration steps. If the error increased from the previous step, then the algorithm needs to rely on the gradient descent method to quickly return to a level of more reasonable convergence. To do this, λ must increase to reduce the effect of H.

If λ becomes very large, it is apparent that (13) becomes the gradient descent method (10).

Conversely, if the error is reduced, then the algorithm needs to take advantage of the Gauss-Newton method to gradually approach the solution. To do this, λ must become very small, negating the effect of I and relying only on $J^T J$ instead. One can note that if λ becomes negligible, (13) becomes (12), the Gauss-Newton method (Roweis, 1996; Ranganathan, 2004).

On a side note, the starting λ and its multipliers, depend on the function to be modeled, so that a bad initial parameter could cause the model to not converge on a reasonable solution. Additionally, Levenberg's method may not be optimal - if λ is large, the impact of the approximated Hessian matrix, which requires the more complicated second derivatives, becomes negligible. Since the approximated Hessian needs to be calculated anyways and it is proportional to the curvature of the function, it could be beneficial to extract information from the second derivatives (Roweis, 1996; Ranganathan, 2004). This was done by Marquardt (1963).

Marquardt (1963) replaced I with the diagonal of $J^T J$ to ensure that large steps are taken in the direction with low curvature and small steps are taken in the direction with high curvature. This is supposed to prevent the algorithm from becoming trapped in an error valley and gives rise to the Levenberg-Marquardt algorithm

(More, 1978; Ranganathan, 2004)

$$\delta = (J^T J + \lambda diag[J^T J])^{-1} \nabla f = (J^T J + \lambda diag[J^T J])^{-1} J^T r \tag{14}$$

Appendix B: Code Explanation

This section explains the MATLAB commands and arguments used to fit the peaks.

thesis_loader

This script loads the desired text file containing the data into a variable containing two columns and n rows, where n is the number of measurements. Then, it assigns the first column, the one containing the date and time, to one variable, q_date_bad, and the second column, discharge, to another variable, q_bad. The reason for the "bad" being appended to the names is that the script must first check to see if there are any missing or redundant data values before proceeding.

The script calls another custom MATLAB function called date_vector. This function is written specifically to convert a numeric date from the format used by the USGS Instantaneous Data Archive (http://ida.water.usgs.gov/ida/), which combines the year, month, day, hour and miniute values, into a single number containing twelve characters (e.g. 20070101001500) into a format recognized by MATLAB.

First, the script converts the large number into a string, so that it can be parsed. Parsing splits the string into 5 columns. The first column contains four integer values for the year. The second column contains two values for the month, third column for the day, fourth for the hour and fifth for the minute. Then, the script uses the str2num command to convert the string values back into numbers. The script output is a matrix of values for the year, month, date, hour, minute and second. Each part of the date and time is stored as a separate column. The size of this matrix is n by 6.

The script stores the output of this function as the vectordate variable and converts it into a matrix of computer time values using the datenum command.

The script subtracts the starting date-time from all other date-times so that the first entry is zero. Then, the script checks for redundant values (i.e. multiple values which have the same exact date and time) using the unique command. The generated output is three data vectors: b, i and j. The b vector contains the data points from serialdate without redundant values. The i vector contains the indices of the non-redundant values, and the j vector contains a numerical sequence stating whether or not a number is unique by giving index of each one. If the number already exists, a previous index value is repeated.

The next step is to define a variable, iorig, to be a vector starting at one, incremented by one, and ending at the maximum value of vector i (the last index of non-redundant values). This creates a vector of indices which are supposed to have values.

Then, the script defines a vector called baddata, containing the points which are in iorig but are not in i. This is accomplished by treating i and iorig as sets and finding the intersection between them by using the setxor command in MATLAB. The intersection is between iorig, which has all indices in the data set, and the the i vector, which has the important values. The intersection is a vector of values which can be removed.

The time vector is a new vector called q_date, which takes all values from the serialdate vector. Then, the indices of the bad points (the baddata vector) are removed from q_date using the [] command.

The script repeates the above for the discharge vector, where a new vector, q, has the values of q_bad. The script removes the baddata values.

The next step is to find out how many data points are actually missing. A new vector, q_date_diff, is differentiated version (diff command) of the q_date vector.

Unfortunately, whenever MATLAB subtracts computer time, it introduces roundoff error, so that the differences between measurements cannot be divided exactly by 96 (the number of fifteen-minute measurements in one day). Instead, we must account for the roundoff error by checking whether or not the gap is within a 10% range; if it is, then the gap is 1 interval and no interpolation is necessary. If the gap is outside, then there is a missing time point, and the location of the point is written in a date_gap vector.

The script uses the nonzero values of date_gap to find the variable, num_missing, which shows how many points are missing in each gap of data.

The expected_date_vec vector is a vector of data of the necessary length to account for all the available data points and for the points to be interpolated. The expected_q_vec is a vector of zeros of the same dimensions as expected_date_vec, and is used to incorporate the available discharge values and the interpolated values.

Because we still havent corrected for the roundoff error, y is q_date multiplied by 1e7, rounded, and then divided by 1e7. The script repeates the previous command for the expected_date_vec, defining it to be z. Since expected_date_vec contains all possible indices of missing and present values, the setdiff command finds the locations of missing values.

The ismember function checks to see whether or not values of y are located in z. The output is two vectors, tf and loc. Loc infills the proper index values of a properly sized data vector. The non-infilled points are those to be interpolated later. The linear interpolation function, interp1, uses the q_date, q, and expected_date_vec vectors to generate q_interp - a vector where missing values have been replaced by interpolated ones.

thesis_ft

The first step is a preparation step for building the periodogram; q_detrended is the detrend command applied to the q_interp vector. Detrending reduces the the intensity of low frequency components.

Then, MATLAB's periodogram command generates two vectors: estm and f. Estm contains the power spectral density estimate and f is a list of frequencies with power spectral density estimates. The arguments of the periodogram function are q_detrended, [], 64 and 1. The first argument means that the periodogram will be calculated from the q_detrended vector. The closed square brackets denote that the default (rectangular) filtering window is used. Filtering windows are generally used to reduce spectral leakage, but the rectangular window is effectively no window. The 64 represents how many points are used to calculate the fast Fourier transform.

By examining multiple kinds of periodograms of the data, we determined that crude periodograms generated by the above method exhibit a triangular shaped distribution of spectral density estimates. Therefore, if the frequency at the top of the triangle could be determined, and multiplied by 2, the whole range of the undesirable high frequencies could be filtered out.

The range is calculated from the first point where the derivative of the crude power spectral density estimate changes sign from being positive to being negative. The estderivsign variable calculates the numerical derivative of estm. A vector of zeros, crossing, is defined for memory allocation. The for loop writes locations into the crossing vector where the derivative changes sign. The nonzeros of crossing leave only the index locations of the turning points. Then, the script extracts the first turning point index location from the f vector to determine the frequency associated with the peak of the triangle. Consequently, the two-poles of the Butterworth lowpass filter with the corner frequency of twice the associated frequency are the variables A and B.

Lowpass filtering is used because frequencies higher than the corner frequency need to be filtered out while preserving the desired frequency components as best as possible. The Butterworth filter accomplishes that by having a maximally flat passband. MATLAB's filter function uses A and B, along with q_interp, to produce q_filtered.

The final filtering step involves calculating the lag between the actual and filtered time series. The lag is calculated with a cross-correlation function, xcorr. Its arguments are q_filtered and q_interp, and its outputs are two variables, C, a vector of the cross correlation sequence, and lags, a vector of lag indices. Next, the max function finds the maximum value in C and its index. The calculated index value is maxlag and is used to extract the highest value from lags.

thesis_separator

The derivsign vector stores the values of the sign of the derivative of q_filtered - the sign command assigns a value of -1 if the number is negative and a 1 if it is positive. Then, k, a vector of zeros with the same length as the derivsign, is allocated into memory. A for loop determines the locations within the vector of values where the sign of the derivative becomes positive. This is written to k, and indicates that the hydrograph is beginning to increase, which can be explained by the beginning of a new peak.

The variable, lagged, is the maxlag value from the previous script is subtracted from all of the turning point locations to account for the lag between the actual data series and the filtered one (which is subject to lag).

A vector called timeindex stores the values of the lag-corrected locations along with placeholders for the first and last value in the data set. This is done so that a loop can be used later do determine which ranges of the discharge data set to extract for each peak.

Now, two vectors of zeros, lowrange and highrange, with dimensions of one less than the length of the timeindex vector, are predefined for memory allocation. A new n-by-2 matrix, ranges, consisting of peak beginning and end indices is created with the first column containing the lowrange vector and the second column containing the highrange vector.

thesis_md_***

This section will go over the modeling of the parameters with the simple surge function. The thesis_md_gamm, thesis_md_weib and thesis_md_sqrt scripts are essentially the same, except different functions are used to calculate forward model values and the partial derivatives. Thus, they are not mentioned in this section, but the code is present in appendix C.

The script closes all the opened figures (which were generated throughout the previous scripts for test purposes and have been commented out). Then, to improve performance, it clears out some of the unnecessary variables using the clearvars command. The .png extension for the plots is defined as the variable ext. A new directory is created which combines the name variable defined in thesis_loader and appends a _simple to it.

From here on, a long, nested for loop with multiple logic functions begins. The for loop runs from 1 to the number of rows present in the ranges matrix, which is equivalent to the number of separated peaks.

The first step extracts the values from the q_interp vector using the data from ranges. The row with the same number as the iteration step is extracted and is defined to be the vector Q. A vector T is defined to be from 0 to one less than the length of Q.

An if statement checks to see if the difference in Q is less than 10cfs. If it is, then the iteration step values for coefficient vectors adata_simp, bdata_simp and nashsutcliffe_simp are all -1, which is a flag value for removing bad results. If it is

the case that the range of Q is less than 10, the peak is not modeled and the i is increased by one, to model the next peak.

Next, it's necessary to subtract the minimum value of the data set this serves as the simplest way to account for, and remove, baseflow. The discharge values without baseflow are the Qa vector. Then, a new vector, Qb, contains each element of Qa divided by the total sum of Qa. This normalizes the discharge values so that the area under the hydrograph curve is 1 reducing the hydrograph to a probability distribution function.

The Levenberg parameter, which was λ in the introduction, is the variable m. The script contains the initial guesses for a and b. The parameters were determined through trial and error and ensuring that the algorithm converged on a correct solution. For the surge function, a high value for a and a low value for b, leads to LMA convergence. In this case, a is 1e6 and b is 1e-6.

A variable x the same as T, and variable ynz is Qb (this was done in out of simplicity, only in order to incorporate a piece of code that was written long before the automation procedure). A variable called iter, which serves as an iteration counter is 0. A variable called maxiter, which specifies the maximum number of iterations is defined to be 15.

Two separate, custom-written, MATLAB functions estimate the partial derivative of the model function with respect to each variable, or parameter, to be estimated. The function dfda evaluates $\frac{df}{da}$ and dfdb evaluates $\frac{df}{db}$. The functions estimate the partial derivative by a forward finite difference equation with a specified delta step.

The jacobian variable consists of the evaluation of dfda and dfdb using the iteration step a and b values, and the non-changing x vector. The jacobiant variable is the transpose of the jacobian. Another function, qq, obtains a set of modeled values. Qq evaluates the surge function at the a and b values of the current iteration step, and x. Then, the sum of residual squares is calculated and defined as rsq. The

calculation subtracts the modeled values from the actual values, squares each one, and adds the result.

Now, three more logic functions follow. The first if statement checks to see if the iteration step is 0, if it is, then m, the Levenberg parameter, is unchanged. Then, if the iteration step is greater than zero, and if the residual sum of squares decreased at this step from the previous one, indicating that the algorithm is moving in the right direction, m is divided by 100. If the opposite is true, m is multiplied by 100.

We now begin to implement the LMA. The first step is to calculate the residual vector, E, by subtracting qq from ynz. We define the rhs vector to be the jacobian multiplied by E (this is the right-hand side of the equation to be solved). JTJ is a vector of the jacobian multiplied by its transpose. Another logic function examines the condition of the JTJ matrix using the rcond comman for computational efficiency. An infinite rcond implies that the solution did not converge, and there is no point to iterate further. The isnan command checks for an ill-conditioned matrix. It produces a 1 when the number is infinite and a zero when it is finite. Thus, when isnan(rcond(JTJ)) is not 1, the script can continue. If rcond is infinite, the iter counter becomes the maxiter value and the script moves on to the next peak.

The adjpar variable is m, multiplied by the double use of MATLABs diag command applied to JTJ. The diag command is used twice to create a diagonal matrix instead of a column vector. The lhs vector is the JTJ matrix plus the adjustment parameters. The δ vector is d, and is solved by using MATLABs backslash divide command, which is similar to multiplying by the inverse, but is more efficient and allows for pseudoinverse multiplication (though it is not used in this case).

Now, since d is a two value vector, the first value is added to the a value, to come up with a better estimate for the next iteration, and the second value is added to b. The reason why a is first and b is second is because in the jacobian, the dfda

function was in the first column and dfdb was in the second. If the order is reversed, then so would the parameters of d.

The script increases the iteration counter by 1 and the rsqold variable becomes the value of rsq. We do this so that rsqold can be used in the next iteration of the algorithm to compare the goodness of fit. The script moves on to plotting the peaks.

The first thing to verify is that the peak has actually been modeled, as it could have not met the pre-LMA criteria. The way to check for this is to see if both a and b values are greater than zero (the filter variables were all -1). The script closes all previously created figures to save memory.

The nashsutcliffe_simp vector uses the Nash-Sutcliffe equation (10) to calculate the fit parameter. A variable, titul, is a row vector of a, b, and nashsutcliffe values. Then, the num2str command converts titul into a string, which becomes the title of each plot. The script navigates to the directory created in the beginning. We define sar to be an invisible figure to prevent it from plotting it on the screen, saving processing time. Namevar becomes the peaks variable converted into a string using num2str.

Next, we plot the x, qq and x, ynz vectors as the modeled and original values. The plot title is the titul variable, evaluated the eval command, which runs a string as an expression. The print command ssaves the figure sar as a png file, with the name being namevar. The script closes the figure and exits out of the directory.

The final step generates a delimited text file of all of the results. First we define, a matrix, dmt, to have: a vector of 1 to the length of the ranges vector - this is done to show the peak numbers; the ranges matrix - to show locations of peak in the data set; the adata and bdata vectors - for the a and b coefficients of the best fitting parameters; the nashsutcliffe values - goodness of fit parameters. Then, a for loop with logic statments checks to see if any of the a or b values are -1, which was the flag used when the peak did not cover a sufficient change in discharge. If both

values are -1, then the index is written into the rmv vector. If not, then the rmv vector entry is zero. If bad data points exist, they are removed using the [] command, and the script saves dmt matrix in the ascii text format as the name variable with the extension simp.dat appended to it.

Appendix C: The MATLAB code

This section contains all of the MATLAB codes used for this project.

thesis_shell

```
clear all
thesis_loader
thesis_ft
thesis_separator
thesis_md_simp
thesis_md_weib
thesis_md_gamm
thesis_md_sqrt
```

thesis_loader

```
name=input('Filename: ');
flow=load([name '.txt']);
q_date_bad=flow(:,1);
q_bad=flow(:,2);

vectordate=date_vector(q_date_bad);
serialdate=datenum(vectordate);
serialdate=serialdate-serialdate(1);

[b,i,j]=unique(serialdate);
iorig=1:max(i);
baddata=setxor(i,iorig);

q_date=serialdate;
q_date(baddata)=[];
q=q_bad;
q(baddata)=[];

q_date_diff=diff(q_date);

for i=1:length(q_date_diff)
    if q_date_diff(i)>1.1*(1/96)
```

```
        date_gap(i)=i;
    end
    if q_date_diff(i)<.9*(1/96)
        date_gap(i)=i;
    end
end

date_gap=nonzeros(date_gap);

num_missing=round(q_date_diff(date_gap)/(1/96));
num_missing=num_missing-1;

expected_date_vec=(0:(1/96):serialdate(end))';

expected_q_vec=zeros(length(expected_date_vec),1);

y=round(q_date*1e7)/1e7;
z=round(expected_date_vec*1e7)/1e7;

[c,zeroindex]=setdiff(z,y);

[tf,loc]=ismember(y,z);

expected_q_vec(loc)=q;

q_interp=interp1(q_date,q,expected_date_vec);
```



```
q_detrended=detrend(q_interp);

[estm,f]=periodogram(q_detrended,[],64,1);

estderivsign=sign(diff(estm));
crossing=zeros(1,length(estderivsign)-1);
for i=2:length(estderivsign)
    if estderivsign(i-1)~=estderivsign(i)
        crossing(i)=i;
    end
end
crossing=nonzeros(crossing);
corner=f(crossing(1));

[B,A]=butter(2,2*corner);

q_filtered=filter(B,A,q_interp);
```

```
[C,lags]=xcorr(q_filtered,q_interp);
[mx,mind]=max(abs(C));
maxlag=lags(mind);
```

thesis_separator

```
deriv=diff(q_filtered);
derivsign=sign(deriv);

k=zeros(1,length(derivsign));
for i=2:length(derivsign)
if derivsign(i) == 0
k(i)=0;
elseif derivsign(i) > derivsign(i-1)
k(i)=(i);
else
k(i)=0;
end
end

lagged=(nonzeros(k)-round(maxlag))';

timeindex=[1 lagged length(q_filtered)];

dateattimeindex=expected_date_vec(timeindex);

dischargeatindex=zeros(1,length(timeindex));
for i=1:length(timeindex)
dischargeatindex(i)=q_filtered(timeindex(i));
end

lowrange=zeros(1,length(timeindex)-1);
highrange=zeros(1,length(timeindex)-1);
for i=1:length(timeindex)-1
lowrange(i)=timeindex(i);
highrange(i)=timeindex(i+1)-1;
end
ranges=[(lowrange)',(highrange)'];
```

thesis_md_simp

```
close all
clearvars -except q_interp expected_date_vec ranges name ;
ext='.png';
mkdir([name '_simple'])
```

```
for peaks=1:length(ranges)
    close all
    Q=q_interp(ranges(peaks,1):ranges(peaks,2));
    T=0:length(Q)-1;
    if range(Q)<10;
        adata_simp(peaks)=-1;
        bdata_simp(peaks)=-1;
        nashsutcliffe_simp(peaks)=-1;
    else
        Qa=Q-min(Q);
        Qb=Qa/sum(Qa);
        m=1;
        a=1e6;
        b=1e-6;
        x=T;
        ynz=Qb;
        iter=0;
        maxiter=15;
            while iter<maxiter
                jacobian=[dfda(a,b,x);dfdb(a,b,x)];
                jacobiant=jacobian';
                qq=s(a,b,x)';
                lsqsum=sum((ynz-qq).^2);
                rsq=lsqsum;
                if iter ==0
                m=m;
                elseif rsq<rsqold
                m=m/100;
                else
                m=m*100;
                end
                E=ynz-qq;
                rhs=jacobian*E;
                JTJ=jacobian*jacobiant;
                if isnan(rcond(JTJ))~=1;
                    adjpar=m.*diag(diag(JTJ));
                    lhs=JTJ+adjpar;
                    d=lhs\ rhs;
                    a=a+d(1);
                    b=b+d(2);
                    iter=iter+1;
                    rsqold=rsq;
                else
                    iter=maxiter;
                end
```

```
            end
            nashsutcliffe=1-(sum((ynz-qq).^2)/
sum((ynz-mean(ynz)).^2));
            if a>0 && b>0 && nashsutcliffe>0
            close all
            nashsutcliffe_simp(peaks)=1-(sum((ynz-qq).^2)/
sum((ynz-mean(ynz)).^2));
            adata_simp(peaks)=a;
            bdata_simp(peaks)=b;

            titul=[a,b,nashsutcliffe_simp(peaks)];
            titul=num2str(titul);

            cd([name '_simple']);
            sar=figure('visible','off');
            namevar=num2str(peaks);
            plot(x,qq,x,ynz);
            title(eval('titul'));
            print(sar,'-dpng',namevar);
            close all
            cd ..
            clear titul x qq ynz;
            else
                 adata_simp(peaks)=-1;
                 bdata_simp(peaks)=-1;
                 nashsutcliffe_simp(peaks)=-1;
            end
    end
end

dmt=[(1:length(ranges))' ranges adata_simp' bdata_simp'
nashsutcliffe_simp'];
for i=1:length(dmt)
    if dmt(i,4)==-1 && dmt(i,5)==-1 && dmt(i,6)==-1
        rmv(i)=i;
    else
        rmv(i)=0;
    end
end

if sum(rmv)~=0
dmtt=dmt;
rmv=nonzeros(rmv);
dmt(rmv,:)=[];
    save([name,'simp.dat'],'dmt','-ascii')
```

```
else
    save([name,'simp.dat'],'dmt','-ascii')
end

clear dmt dmtt rmv;
```

thesis_md_weib

```
clearvars -except q_interp expected_date_vec ranges name ;
ext='.png';
mkdir([name '_weibull'])
for peaks=1:length(ranges)
    close all
    Q=q_interp(ranges(peaks,1):ranges(peaks,2));
    T=(0:length(Q)-1);
    if range(Q)<10;
        nashsutcliffe_weib(peaks)=-1;
        adata_weib(peaks)=-1;
        bdata_weib(peaks)=-1;
    else
        Qa=Q-min(Q);
        Qb=Qa/sum(Qa);
        m=1000000;
        a=1;
        b=100;
        x=T;
        ynz=Qb;
        iter=0;
        maxiter=15;
        while iter<maxiter
            jacobian=[dfdk(a,b,x);dfdtheta(a,b,x)]';
            jacobiant=jacobian';
            qq=wb(a,b,x)';
            lss= sum((ynz-qq).^2);
                if iter ==0
                m=m;
                elseif lss>lssold
                m=m*10;
                else
                m=m/10;
                end
            E=ynz-qq;
            rhs=jacobiant*E;
            JTJ=jacobiant*jacobian;
            if isnan(rcond(JTJ))~=1
                adjpar=m.*diag(diag(JTJ));
```

```
                lhs=JTJ+adjpar;
                d=lhs\ rhs;
                a=a+d(1);
                b=b+d(2);
                lssold=lss;
                disp (m);
            else
                iter=maxiter
            end
            iter=iter+1;
        end
            nashsutcliffe=1-(sum((ynz-qq).^2)/
sum((ynz-mean(ynz)).^2));
            if isnan(a)~=1 && b>0 && nashsutcliffe>0
                nashsutcliffe_weib(peaks)=nashsutcliffe;
                clear rsq jacobian jacobiant E adjpar
lhs d m lsqsum rsqold iter;
                adata_weib(peaks)=a;
                bdata_weib(peaks)=b;
                titul=[a,b,nashsutcliffe_weib(peaks)];
                titul=num2str(titul);
                cd([name '_weibull']);
                sar=figure('visible','off');
                namevar=num2str(peaks);
                plot(x,qq,x,ynz);
                title(eval('titul'));
                print(sar,'-dpng',namevar);
                close
                cd ..
                clear titul x qq ynz;
            else
                            nashsutcliffe_weib(peaks)=-1;
                              adata_weib(peaks)=-1;
                            bdata_weib(peaks)=-1;
            end
    end
end
dmt=[(1:length(ranges))' ranges adata_weib'
bdata_weib' nashsutcliffe_weib'];
for i=1:length(dmt)
    if dmt(i,4)==-1 && dmt(i,5)==-1 && dmt(i,6)==-1
        rmv(i)=i;
    else
        rmv(i)=0;
    end
```

```
end

if sum(rmv)~=0
dmtt=dmt;
rmv=nonzeros(rmv);
dmt(rmv,:)=[];
    save([name,'weib.dat'],'dmt','-ascii')
else
    save([name,'weib.dat'],'dmt','-ascii')
end

clear dmt dmtt rmv;
```

thesis_md_gamm

```
clearvars -except q_interp expected_date_vec ranges name ;
ext='.png';
mkdir([name '_gamma'])
for peaks=1:length(ranges)
    close all
    Q=q_interp(ranges(peaks,1):ranges(peaks,2));
    T=0:length(Q)-1;
    if range(Q)<10;
        kdata_gamm(peaks)=-1;
        thetadata_gamm(peaks)=-1;
        nashsutcliffe_gamm(peaks)=-1;

    else
        Qa=Q-min(Q);
        Qb=Qa/sum(Qa);
        m=1;
        k=10;
        theta=10;
        x=T;
        ynz=Qa;
        iter=0;
        maxiter=15;
            while iter<maxiter
                jacobian=[ddag(k,theta,x); ddbg(k,theta,x)];
                jacobiant=jacobian';
                qq=gm(k,theta,x)';
                lss=sum((ynz-qq).^2);
                    if iter ==0
                    m=m;
                    elseif lss>lssold
                    m=m*100;
```

```
                    else
                    m=m/100;
                    end
                E=ynz-qq;
                rhs=jacobiant'*E;
                JTJ=jacobian*jacobiant;
                if isnan(rcond(JTJ))~=1
                    adjpar=m.*diag(diag(JTJ));
                    lhs=JTJ+adjpar;
                    d=lhs'\ rhs;
                    k=k+d(1);
                    theta=theta+d(2);
                    iter=iter+1;
                    lssold=lss;
                    disp (m);
                else
                    iter=maxiter;
                end
            end
            nashsutcliffe=1-(sum((ynz-qq).^2)/
sum((ynz-mean(ynz)).^2));
            if k>0 && theta>0 && nashsutcliffe >0
                nashsutcliffe_gamm(peaks)=nashsutcliffe;
                kdata_gamm(peaks)=k;
                thetadata_gamm(peaks)=theta;
                titul=[k,theta,nashsutcliffe];
                titul=num2str(titul);
                cd([name '_gamma']);
                sar=figure('visible','off');
                namevar=num2str(peaks);
                plot(x,qq,x,ynz);
                title(eval('titul'));
                print(sar,'-dpng',namevar);
                close
                cd ..
                clear titul x qq ynz;
            else
                nashsutcliffe_gamm(peaks)=-1;
                kdata_gamm(peaks)=-1;
                thetadata_gamm(peaks)=-1;
            end
    end
end

dmt=[(1:length(ranges))' ranges kdata_gamm'
```

```
thetadata_gamm' nashsutcliffe_gamm'];
for i=1:length(dmt)
    if dmt(i,4)==-1 && dmt(i,5)==-1 && dmt(i,6)==-1
        rmv(i)=i;
    else
        rmv(i)=0;
    end
end

if sum(rmv)~=0
dmtt=dmt;
rmv=nonzeros(rmv);
dmt(rmv,:)=[];
     save([name,'gamm.dat'],'dmt','-ascii')
else
    save([name,'gamm.dat'],'dmt','-ascii')
end

clear dmt dmtt rmv;
```

thesis_md_sqrt

```
clearvars -except q_interp  ranges name ;
ext='.png';
mkdir([name '_sqr'])
for peaks=1:length(ranges)
    close all
    Q=q_interp(ranges(peaks,1):ranges(peaks,2));
    T=(0:length(Q)-1);
    if range(Q)<10;
        nashsutcliffe_sqt(peaks)=-1;
        kdata_sqt(peaks)=-1;
        ttpk(peaks)=-1;
    else
        Qa=(Q-min(Q));
        Qb=Qa/sum(Qa);
        m=1;
        beta=1e-3;
        x=T;
        [zzzzz,ttp]=max(Qb);
        ttpk(peaks)=(1/ttp(1));
        ynz=Qb;
        iter=0;
        maxiter=15;
            while iter<maxiter
                jacobian=[dsqtdx(beta,x)];
```

```
                jacobiant=jacobian';
                qq=sqt(beta,x)';
                lss=sum((ynz-qq).^2);
                    if iter ==0
                    m=m;
                    elseif lss>lssold
                    m=m*10;
                    else
                    m=m/10;
                    end
                E=ynz-qq;
                rhs=jacobiant'*E;
                JTJ=jacobian*jacobiant;
                if isnan(rcond(JTJ))~=1
                    adjpar=m.*diag(diag(JTJ));
                    lhs=JTJ+adjpar;
                    d=lhs'\ rhs;
                    beta=beta+d;
                    iter=iter+1;
                    lssold=lss;
                    disp (m);
                else
                    iter=maxiter;
                end

            end
                nashsutcliffe=1-(sum((ynz-qq).^2)/
sum((ynz-mean(ynz)).^2));
            if isnan(beta)~=-1 && nashsutcliffe>0
                nashsutcliffe_sqt(peaks)=nashsutcliffe;
                kdata_sqt(peaks)=beta;

                titul=[beta,nashsutcliffe];
                titul=num2str(titul);

                cd([name '_sqr']);
                sar=figure('visible','off');
                namevar=num2str(peaks);
                plot(x,qq,x,ynz);
                title(eval('titul'));
                print(sar,'-dpng',namevar);
                close
                cd ..
                clear titul x qq ynz;
            else
```

```
                nashsutcliffe_sqt(peaks)=-1;
                kdata_sqt(peaks)=-1;
            end

    end
end
dmt=[(1:length(ranges))' ranges kdata_sqt'
nashsutcliffe_sqt' ttpk'];
for i=1:length(dmt)
    if dmt(i,4)==-1 && dmt(i,5)==-1
        rmv(i)=i;
    else
        rmv(i)=0;
    end
end

if sum(rmv)~=0
dmtt=dmt;
rmv=nonzeros(rmv);
dmt(rmv,:)=[];
     save([name,'sqr.dat'],'dmt','-ascii')
else
    save([name,'sqr.dat'],'dmt','-ascii')
end

close all
```

wb

```
function wb=wb(a,b,x)
    wb=(a/b)*((x/b).^(a-1)).*exp(-(((x/b).^a)));
end
```

sqt

```
function sqt=sqt(beta,x)
sqt=(beta^2)*x.*exp(-beta.*x);
end
```

s

```
function s=s(a,b,x)
s=a*x.*exp(-b.*x);
end
```

gm

```
function gm=gm(k,theta,x)
    gm=((x.^(k-1)).*((exp(-x/theta)/((theta^k)*1))));
end
```

dsqtdx

```
function dsqtdx=dsqtdx(beta,x)
h=.00000001;
dsqtdx=(sqt(beta+h,x)-sqt(beta-h,x))/(2*h);
end
```

dfdtheta

```
function dfdtheta=dfdtheta(k,theta,x)
h=.000001;
dfdtheta=(wb(k,theta+h,x)-wb(k,theta,x))/h;
end
```

dfdk

```
function dfdk=dfdk(k,theta,x)
h=.000001;
dfdk=(wb(k+h,theta,x)-wb(k,theta,x))/h;
end
```

dfdb

```
function dfdb=dfdb(a,b,x)
h=.00000001;
dfdb=(s(a,b+h,x)-s(a,b,x))/h;
end
```

dfda

```
function dfda=dfda(a,b,x)
h=.00000001;
dfda=(s(a+h,b,x)-s(a,b,x))/h;
end
```

ddbg

```
function ddbg=ddbg(k,theta,x)
h=1e-6;
ddbg=(gm(k,theta+h,x)-gm(k,theta,x))/(h);
end
```

ddag

```
function ddag=ddag(k,theta,x)
h=1e-12;
ddag=(gm(k+h,theta,x)-gm(k,theta,x))/h;
end
```

date_vector

```
function date_conv=date_vector(discharge_date)
tts=int2str((discharge_date(:,1)));
year_string=tts(:,1:4);
month_string=tts(:,5:6);
day_string=tts(:,7:8);
hour_string=tts(:,9:10);
minute_string=tts(:,11:12);

yr=str2num(year_string);
mo=str2num(month_string);
da=str2num(day_string);
ho=str2num(hour_string);
mi=str2num(minute_string);

date_conv=[yr, mo, da, ho, mi,zeros(length(mi),1)];
end
```