

University of Nevada, Reno

**A Vector Generalized Linear Model for
Trivariate Stochastic Episodes with Pareto and
Geometric Marginals**

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy in
Statistics and Data Science

by

Erick Luerken

Anna K. Panorska, Ph.D., Dissertation Advisor

August, 2025

Copyright by Erick Luerken 2025

All Rights Reserved



THE GRADUATE SCHOOL

We recommend that the dissertation
prepared under our supervision by

Erick Luerken

entitled

**A Vector Generalized Linear Model for Trivariate Stochastic
Episodes with Pareto and Geometric Marginals**

be accepted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

Anna K. Panorska, Ph.D.
Advisor

Tomasz J. Kozubowski, Ph.D.
Committee Member

Marek Arendarczyk, Ph.D.
Committee Member

Emily M. Hand, Ph.D.
Committee Member

Minggen Lu, Ph.D.
Graduate School Representative

Markus Kemmelmeier, Ph.D., Dean
Graduate School

August, 2025

Abstract

Extreme weather, climate, and financial events are frequently featured in the news due to their significant impact on both human life and the environment. This work presents statistical modeling tools for analyzing such extreme events. We propose a multivariate model to describe events like storms, floods, heatwaves, and financial market crashes. The model incorporates covariates that influence the size or probability of these extreme events. Specifically, we implement a vector generalized linear model (VGLM), an extension of standard regression methods. We also develop parameter estimators, including theorems on their existence and uniqueness. To facilitate practical use, we have created a freely available computational package. In addition, we offer recommendations on best practices for handling common computational challenges in applying the model to real-world data. We demonstrate the effectiveness of our model by applying it to precipitation data in California and Nevada, achieving strong results.

ACKNOWLEDGEMENTS

This has been the culmination of years of hard work and dedication, but would have been impossible without an even greater amount of love, support and encouragement from a truly wonderful group of people. Unfortunately the list of people to whom I have learned and grown from is too long to list, but to all the friends, teammates, colleagues, professors and family that did not see your names here, know that your contributions have nevertheless been felt and truly appreciated by me more than I could ever articulate.

I would like to start by thanking the professors and office staff at the Department of Mathematics and Statistics at UNR for their continuous support. I owe an even greater debt to my committee members, Professor Minggen Lu, Professor Emily Hand & Professor Marek Arendarczyk for their wonderful suggestions and guidance, and for always being so willing to offer me your

time.

I would also like to thank Dr. Grant Schissler, my first PhD advisor for believing in me and pushing me towards excellence. Your friendship in the years since has been the best part of our collaboration. I would also like to single out Professor Tomasz Kozubowski (aka “Dr. K”) for the wonderful guidance he has offered to me from the first time we met as a prospective Grad Student 20 years ago, throughout my career and now at the conclusion of this degree. I couldn’t imagine doing this without you as part of the journey.

I owe my deepest debt to my advisor, Professor Anna Panorska. You put so much effort into all your students and will never get the recognition for this that you deserve. But I hope you know that your impact in my life has been infinitely positive ($\mu - a.e.$). It is not exaggeration to say that I would not be here, were it not for your kindness and support. *Dziękuję z całego serca.*

To the family I gained in Australia, particularly ‘Nonna’ and ‘Pa’, thank you so much for welcoming me into your family with open arms and always being

supportive and proud of me. I won the in-law lottery with you all and am grateful for this every day.

I want to thank two special family members that didn't live to see this come to fruition. To Grandma, I know that you would have thought this "took long enough" but can also picture the twinkle in your eyes as you smiled as you said it. I only wish I could see it in person one more time.

Für "Vats": Es gibt nicht genug Tinte und Papier auf der Welt, um dir zu sagen, wie sehr ich dich vermisse und mir wünsche, du wärst jetzt bei mir. Du hast mir den Wert von Bildung beigebracht und in mir die Liebe zum Lernen geweckt. Du hast mich gelehrt, unermüdlich für meine Träume zu kämpfen – und dies ist ein weiterer, den ich erreicht habe. Was gäbe ich darum, wenn wir diesen Weg gemeinsam hätten gehen können. Wie immer hoffe ich, dass ich dein Vertrauen in mich gerechtfertigt und mir den Stolz verdient habe, den du mir stets entgegengebracht hast. In unendlicher Liebe, dein Sohn.

To Tom: You are the greatest brother and best friend I could ever have wished for, and I'm grateful every day that I get to call you my brother.

You have never shied away from any challenge, and when fully engaged, you never give up. What would be a ‘defeat’ for most, is merely a minor setback that you’ll overcome ‘next time’. I don’t know if I’ve ever told you this but I’ve always admired you for that, and have tried my best to learn from your example. I could never have achieved this, without thinking of you and how you would “find a way”. You’re my favorite human, and I love you.

To Mom: Like with Vati, it’s impossible to put into ink and paper how much you mean to me. You instilled in us the value of hard work and the sense of pride that can only be felt by giving full measure. You have never stopped encouraging Tom & I to pursue our dreams and sacrificed so much to help us achieve them. You taught me the importance of inner strength and resolve and I’ve needed all of it as I’ve pursued this. Thank you so much for always supporting me and helping me push forward throughout my life. With endless love, your son.

And finally, to my brilliant, humble and beautiful partner and wife, Alexandra, and our three greatest adventures: Mathilda, Willa and Tobi. You are the greatest joys of my life; every day that I’ve had with you has been a trea-

sure, and all the memories I have with you are my most valuable possessions. You've all been invaluable to me in helping me complete this endeavor, from your jokes, your songs and your beautiful pictures and creations. Of course I owe a special thanks to you, my love, for all the sacrifices you've made to make sure I could accomplish this, and for your infinite support, caring and love. I love you all more than there are stars in the multiverse.

Contents

List of Tables	xii
List of Figures	xiii
1 Introduction and Motivation	1
1.1 Overview	1
1.2 Motivation and Background	2
1.3 Stochastic Events and Mathematical Framework	5
1.4 Related Work	9
1.4.1 The TETLG Model	9
1.4.2 The GSMP Model	10
1.4.3 Vector Generalized Linear Models	11
1.5 Contributions & Outline	11

2	Theoretical Derivations of the GSMP-VGLM Model	14
2.1	Historical Development of GSMP Models	15
2.2	The VGLM model	19
2.3	Estimation of parameters	23
2.3.1	Case 1: Known α	26
2.3.2	Case 2: Known $\boldsymbol{\eta}$	33
2.3.3	Case 3: α and $\boldsymbol{\eta}$ Unknown	43
2.4	Establishing Goodness-of-Fit Properties	51
2.4.1	Deriving Goodness-of-Fit for X	53
2.4.2	Deriving Goodness-of-Fit for Y	56
3	Computational Estimation of GSMP-VGLM Model	61
3.1	Overview of Estimation Approaches	61
3.2	Exact Solutions via Normal Equations	62
3.2.1	Theoretical Motivations & Derivation	63
3.2.2	Normal Equation Solution	64
3.2.3	Applying in Practice	65
3.2.4	Analysis of Avg. Exact Solutions vs. Normal Equation	66
3.3	Numerical Solvers for Systems of Nonlinear Equations	70
3.3.1	Solving the Maximum Likelihood Equations	70

3.3.2	The <code>fsolve</code> Algorithm and Implementation	71
3.3.3	Issues with Implementation	73
3.4	Direct Maximization of Likelihood Functions	74
3.4.1	The Maximum Likelihood Objective	74
3.4.2	Taylor Series Approximation Background	76
3.4.3	Implementation Details and Practical Considerations	83
3.4.4	Advanced Gradient-Based Methods	87
3.4.5	Conclusions on Likelihood Maximization	91
3.5	Telescopic Grid Search for α Parameter Estimation	93
3.5.1	The Challenge of α Estimation	94
3.5.2	Conceptual Framework of Telescopic Grid Search	95
3.5.3	Detailed Algorithm Specification	97
3.5.4	TGS Details	97
3.5.5	Visual Demonstration of Telescopic Grid Search	104
3.5.6	Comparison to Other Exploratory Searches	105
3.5.7	Practical Recommendations	107
3.5.8	Conclusions on Telescopic Grid Search	108
3.6	Bayesian Optimization for α Parameter Estimation	109
3.6.1	Theoretical Foundations of Bayesian Optimization	110

3.6.2	Gaussian Process Surrogate Functions	112
3.6.3	Acquisition Functions for Bayesian Optimization	114
3.6.4	Implementation of Bayesian Optimization for GSMP- VGLM	116
3.6.5	Visual Demonstration of Telescopic Grid Search	120
3.6.6	Practical Recommendations	121
3.6.7	Conclusions on Bayesian optimization	122
4	Applications	124
4.1	Overview of Data	124
4.2	Addressing the Seasonality of Precipitation Data	125
4.3	Case Study: South Lake Tahoe	127
4.4	Case Study: Reno Airport	129
4.5	Discussion and Extensions	133
5	Additional Considerations	135
5.1	Review of Hurdle Models	136
5.2	Hurdle Model Implementation in GSMP-VGLM	137
6	Python Package	141
6.1	Python and the Scientific Computing Ecosystem	141

6.2	High-Level Overview of the GSMP-VGLM Package	143
6.3	<code>data_generation.py</code>	144
6.4	<code>grad_desc_sem_functions.py</code>	147
6.5	<code>bayes_alpha_opt.py</code>	149
6.6	<code>data_generation.py</code>	150
6.7	Conclusion	151
7	Summary	153
7.1	Main Results	153
7.2	Future Work Directions	155
8	Bibliography	158

List of Tables

- 2.1 The table shows the proportion of times (out of 100 simulations) that the statistic $A < 0$ for different true values of α (rows) and sample sizes (columns). 42

List of Figures

- 1.1 Daily closing prices of Bitcoin (BTC) in USD(\$). Data is from Yahoo! Finance and covers the period from July 31, 2024 to December 31, 2024. Date is on the x-axis and the closing price is on the y-axis. 3
- 1.2 Daily stock closing prices of Tesla (TSLA) from January 1, 2019 to December 31, 2024 (x-axis). The closing price is shown in USD (\$) on the y-axis. 4
- 1.3 Example of stochastic episodes around a threshold value of zero. The points mark the values of the process' observations. The shaded areas show the episodes/events. 6

- 2.1 Boxplots of parameter estimates of each element of $\boldsymbol{\eta}$ (x-axis) when $\alpha = 0.001$. For each parameter, there are six boxplots associated with the smallest sample size (red) to the largest sample (yellow). The y-axis shows the actual values observed and the dashed horizontal lines show the true value of the data. 31
- 2.2 Boxplots of parameter estimates of each element of $\boldsymbol{\eta}$ (x-axis) when $\alpha = 1.1$. For each parameter, there are six boxplots associated with the smallest sample size (red) to the largest sample (yellow). The y-axis shows the actual values observed and the dashed horizontal lines show the true value of the data. 32
- 2.3 Boxplots of parameter estimates of each element of $\boldsymbol{\eta}$ (x-axis) when $\alpha = 2.1$. For each parameter, there are six boxplots associated with the smallest sample size (red) to the largest sample (yellow). The y-axis shows the actual values observed and the dashed horizontal lines show the true value of the data. 33
- 2.4 The log-likelihood function $v(\alpha)$ when $\alpha = 0.001$. The x-axis shows a range of possible α values from 0.001 to 0.01, while the y-axis shows the value of $v(\alpha)$ 38

- 2.5 Similar to 2.4, but with the x-axis now from 0.001 to 0.3. The y-axis is scaled accordingly. 39
- 2.6 Varying values of α on the x-axis between 0.001 to 200 for $v(\alpha)$ when the true value of $\alpha = 1$. The y-axis shows the resulting value of the function $v(\alpha)$ 40
- 2.7 Boxplots of $\hat{\alpha}$ estimates when all parameters are unknown. Each boxplot is associated with a different value of α from $\alpha = 0$ (TETLG) to $\alpha = 5$. The y-axes in each boxplot vary for each value of α 47
- 2.8 Boxplots for each specific value of $\hat{\eta}$. Each value of η has five different values that are based on the different values of α tested with $\alpha = 0$ on the left and increasing to $\alpha = 5$ on the right. The dashed black line shows the true value of each parameter, while the y-axis shows the values of the estimator. 48
- 2.9 Boxplots for each specific value of $\hat{\beta}$. Each value of β has five different values that are based on the different values of α tested with $\alpha = 0$ on the left and increasing to $\alpha = 5$ on the right. The dashed black line shows the true value of each parameter, while the y-axis shows the values of the estimator. 49

2.10	QQ-Plot of the observed data with fitted parameters against a theoretical $\mathcal{U}(0,1)$ distribution. The red line is the theoretical perfect fit between observed and theoretical quantiles.	55
2.11	QQ-Plot of the observed data with fitted parameters against a theoretical $\mathcal{U}(0,1)$ distribution. The red line is the theoretical perfect fit between observed and theoretical quantiles.	60
3.1	Boxplots of the estimated values of each component of $\boldsymbol{\eta}$ using averages of 2x2 matrices (left two boxplots) and the normal equations (right two boxplots). For each method and value of $\boldsymbol{\eta}$, there are five plots for different values of α ranging from 1×10^{17} (leftmost) to 5 (rightmost). The y-axis shows the full-range of parameter estimates found scaled to include the largest value across all parameters and methods.	67
3.2	The same plot as in Figure 3.1, but "zoomed" to y-values between ± 10	68

- 3.3 KDE plots of the observed computation time for each method where the “average” method is in blue and the “Normal Equation” method in red. The dashed vertical lines show the mean time across all simulations. The x-axis is the time taken and the y-axis is the density value of the KDE estimator. 69
- 3.4 Three iterations of Telescopic Grid Search. The x-axis shows values of α while the y-axis shows the log-likelihood values associated with the points. Only points with dots represent actually estimated values; the connecting lines are linear interpolations between them. The blue line represents the first set of estimated values which spans the width of the x-axis. The orange line shows the next iteration and the green represents the final iteration. The red dot is the actual value of α used to generate the data. 104

- 3.5 The results of 30 steps of Bayesian Optimization to find α .
The x-axis is the range of potential α values that can be explored and each dot represents a tested value of α and the y-axis is the associated log-likelihood value. Only blue dots were evaluated and the lines connecting them are linear interpolations between known points. The red dot between a blue dot at the top is the true value of α used to generate the data. 120
- 4.1 Annual cycle of precipitation events at South Lake Tahoe Airport. (a) Estimated daily probability of above-trace precipitation. (b–d) Points represent observed event properties; red curves denote theoretical percentiles (50th, 75th, 95th, 99th) under the HSMP-VGLM. 129
- 4.2 Annual cycle of precipitation events at South Reno Airport. (a) Estimated daily probability of above-trace precipitation. (b–d) Points represent observed event properties; red curves denote theoretical percentiles (50th, 75th, 95th, 99th) under the HSMP-VGLM. 130

4.3 Monthly counts of severe weather reports (2004–2018) from NOAA for western Nevada counties. Event types are categorized as “Explicitly Convective”, “Winter Storm”, and “Any Flooding”. 132

Chapter 1

Introduction and Motivation

1.1 Overview

In recent years, the world has witnessed an unprecedented increase in the frequency and intensity of extreme events across various domains. From prolonged droughts and devastating floods to volatile market fluctuations, these events significantly impact human lives, infrastructure, and financial systems. The ability to accurately model and quantify such extreme behaviors has become increasingly crucial for effective risk management, informed decision-making, and proactive planning across disciplines.

Though several models for these phenomena exist (see, e.g., Weyant et al., 2025, or Arendarczyk et al., 2018 and the references therein), in this work, we focus on the introduction and development of a new Vector Generalized

Linear Model (VGLM) for a trivariate vector generated by stochastic events with Pareto marginals. This novel approach extends previous work by allowing the parameters of the marginal distributions to depend on covariates. This allows for leveraging vast amounts of external data that may exist for each stochastic episode and induce heterogeneity, which is often present in real-world data.

1.2 Motivation and Background

Now, more than ever, the world around us appears to be experiencing increased extremes across numerous domains. The most striking examples occur in climate and weather science, where we witness an increased frequency and severity in extreme weather events. These range from prolonged droughts to deadly floods, record-breaking heat waves, and destructive high winds, causing untold devastation in both human lives and the economy, see World Weather and Climate Extremes Archive, World Meteorological Organization.

Financial markets also exhibit substantial volatility, exemplified by the rise of cryptocurrencies. The advent of digital currencies has fostered a new spec-

ulative arena with dramatic shifts in valuations and unprecedented boom-and-bust patterns. Figure 1.1 shows an example of the cryptocurrency price movement exhibiting large jumps.



Figure 1.1: Daily closing prices of Bitcoin (BTC) in USD(\$). Data is from Yahoo! Finance and covers the period from July 31, 2024 to December 31, 2024. Date is on the x-axis and the closing price is on the y-axis.

As can be seen in this limited plot, there is a large jump in price in November 2024. Even in traditional markets, log-returns—a barometer of market sentiment—show large swings with massive shocks. Figure 1.2 contains Tesla stock prices also showing large volatility.



Figure 1.2: Daily stock closing prices of Tesla (TSLA) from January 1, 2019 to December 31, 2024 (x-axis). The closing price is shown in USD (\$) on the y-axis.

Again, large swings in price changes are evident, which sharp upticks leading to sudden declines even among broader macro trends. Being able to accurately model and quantify such extreme behavior is of paramount importance. Consider an insurance company reviewing daily claims lodged against it. Should these totals exceed their available capital, they could face ruin without alternative support. Being able to quantify the total excess claims, maximum daily claim, and expected duration of the overages would enable the company to purchase the appropriate amount of protection to minimize their risk while maintaining profitability.

Another topic we will delve into more deeply is precipitation. The United Nations reports: "*A 2022 World Bank study estimated that 1.81 billion people – nearly a quarter of the world’s population – are directly exposed to 1-in-100-year flood events, with 89 per cent living in low- and middle-income countries.*" Being able to understand and predict more accurate rain/snowfalls would allow regions and towns to properly prepare for flood and mud slide events or droughts. Lacking this insight could lead to critical infrastructure failure, loss of life and severe economic impact.

1.3 Stochastic Events and Mathematical Framework

A stochastic event such as flood is characterized mathematically as a trivariate vector:

$$(X, Y, N) = \left(\sum_{i=1}^N X_i, \bigvee_{i=1}^N X_i, N \right), \quad (1.1)$$

where X represents the *magnitude* (sum of observations), Y represents the *maximum* observation, and N represents the *duration* of the event, measured as the number of consecutive observations. The random variables X_1, X_2, \dots are independent and identically distributed (IID) and represent observations

from the process of interest, while N is a positive integer-valued random variable independent of the $\{X_i\}$ sequence. In the flood example the process of interest may be the daily maximum river stage or daily total precipitation in a given location. See Arendarczyk et al. 2018 for reference.

To better understand the concept of stochastic events, consider Figure 1.3, which shows observations crossing above and below a threshold of zero.

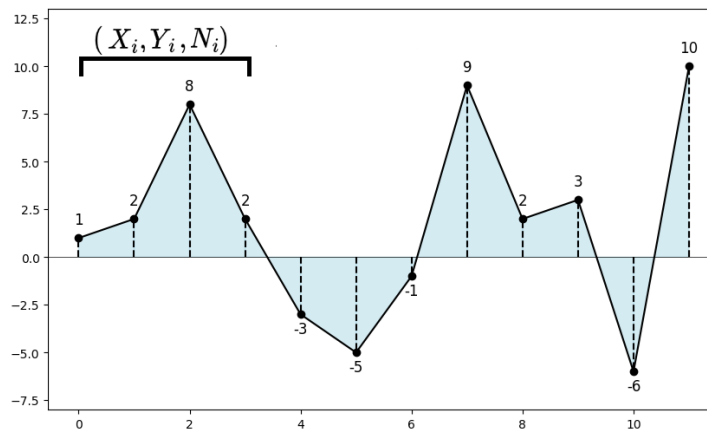


Figure 1.3: Example of stochastic episodes around a threshold value of zero. The points mark the values of the process' observations. The shaded areas show the episodes/events.

When consecutive observations cross above or below a threshold, they form distinct episodes characterized by their total magnitude, maximum value, and duration.

This visualization demonstrates how stochastic events naturally arise in threshold exceedance analysis. For example, the first episode in this hypothetical process results in the trivariate vector $(X = 13, Y = 8, N = 4)$, representing an event with a total magnitude of 13, a maximum value of 8, and a duration of 4 time units.

Returning to the notion of possible applications, we can consider the following domains:

- **Climate Science:** Heat waves can be modeled as periods of consecutive days with maximum temperatures exceeding a high threshold. Similarly, precipitation events can be characterized as consecutive wet days, and floods as consecutive observations of water levels above a flood threshold. See Weyant et al., 2025.
- **Hydrology:** Drought events can be defined as consecutive days with precipitation below a critical threshold, providing insights into their severity (magnitude), intensity (maximum), and persistence (duration).
- **Energy Demand:** Heating or cooling events can be defined as consecutive days classified as "heating/cooling" days, where the definition

involves a threshold of daily temperature, helping energy companies plan resource allocation.

- **Finance:** Episodes of growth for financial assets can be modeled as periods of consecutive positive log-returns, while market downturns represent periods of consecutive negative returns.
- **Insurance:** Claim events exceeding certain thresholds can be modeled to better understand their potential magnitude, maximum severity, and duration for improved risk assessment.
- **Epidemiology:** Disease outbreaks can be characterized by episodes where infection rates exceed endemic thresholds, providing insights into the total case burden, peak intensity, and duration of epidemic waves.

In each of these contexts, the trivariate structure of stochastic events provides a comprehensive and holistic characterization of the phenomenon of interest, capturing not just whether a threshold is exceeded, but also the extent, intensity, and persistence of the exceedance.

1.4 Related Work

The study of stochastic events, as defined in Equation (1.1), has a rich history in statistical literature. Previous research has explored various distributional assumptions for the underlying observations and duration.

1.4.1 The TETLG Model

One prominent model assumes that the observations of interest are independent and identically distributed (IID) from an Exponential distribution with mean $1/\lambda$, denoted: $X_i \stackrel{iid}{\sim} \mathcal{Exp}(\lambda)$, with probability density function (PDF) as follows:

$$g(x) = \lambda e^{-\lambda x}, \quad x \in \mathbb{R}^+ \quad (1.2)$$

The distribution of duration N is typically modeled using a Geometric distribution with probability mass function as follows:

$$P(N = k) = p(1 - p)^{k-1}, \quad k \in \mathbb{N} = \{1, 2, \dots\}, \quad (1.3)$$

and denoted by $Geo(p)$. This combination leads to the **T**rivariate distribution with **E**xponential, **T**runcated **L**ogistic, and **G**eometric marginals, or $\mathcal{TETLG}(\lambda, p)$, as detailed by Kozubowski et al. (2011) which provides closed-

form expressions for the joint density function and establishes the existence and uniqueness of the maximum likelihood estimators for the parameters λ and p .

1.4.2 The GSMP Model

A significant extension to the TETLG model addresses the light-tailed nature of the Exponential distribution, which may be inappropriate for modeling heavy-tailed phenomena common in extreme events. The **G**eometric **S**um and **M**aximum of **P**areto variables (GSMP) model, introduced by Arendarczyk et al. (2018), assumes that the vector of observed durations follows a multivariate Pareto distribution with the following stochastic representation:

$$[X_1, X_2, \dots, X_n]^\top \stackrel{d}{=} \left[\frac{E_1}{Z}, \frac{E_2}{Z}, \dots, \frac{E_n}{Z} \right]^\top \quad (1.4)$$

where the $E_i \stackrel{iid}{\sim} \mathcal{Exp}(\lambda)$ and Z is a mixing variable with $E_i \perp Z$ (i.e. Z is independent from X_i). The GSMP model introduces a dependence structure among observations. In this framework, the X_i are identically distributed but no longer independent, providing greater flexibility in modeling complex dependencies.

1.4.3 Vector Generalized Linear Models

Traditional stochastic event models assume that the observed events are independent and identically distributed. However, in many real-world applications, the parameters governing these events may depend on covariates that vary over time or across different contexts. For a thorough treatment on the subject, refer to Yee (2015). To address the specific limitations of the TETLG model, Zuniga et al. (2021) introduced a Vector Generalized Linear Model (VGLM), allowing the TETLG parameters to depend on covariates through appropriate link functions.

This dissertation extends the VGLM approach to the GSMP model, developing a VGLM framework for stochastic events arising from a multivariate Pareto distribution. This extension accommodates both the heavy-tailed nature of many extreme phenomena and the non-stationarity and heterogeneity often present in real-world data.

1.5 Contributions & Outline

This dissertation makes several significant contributions to the statistical modeling of stochastic events, including the following:

1. Development of a Vector Generalized Linear Model (VGLM) for the GSMP distribution, allowing the parameters to depend on covariates.
2. Derivation of maximum likelihood estimation procedures for the model parameters under various scenarios.
3. Investigation of the statistical properties of the estimators, including consistency and asymptotic behavior.
4. Implementation of the model in efficient computational algorithms.
5. Application of the model to real-world data, demonstrating its utility in climate science, hydrology, and finance.
6. An open-source statistical library, `gsmp_vglm`, that fits the GSMP-VGLM and TETLG-VGLM models. This includes goodness-of-fit methods as well. See Luerken 2025.

The remainder of this dissertation is organized as follows:

- **Chapter 2** presents the theoretical derivations of the GSMP-VGLM model, including the likelihood function, parameter estimation, and statistical properties including Goodness-of-Fit methods for validation.

- **Chapter 3** discusses computational methods for implementing the GSMP-VGLM, including optimization algorithms, numerical challenges, and software implementation.
- **Chapter 4** presents an application of the GSMP-VGLM model to precipitation data.
- **Chapter 5** includes some additional features of the model, specifically parameterizations as a Hurdle model to handle over-dispersion of $\text{duration} = 1$.
- **Chapter 6** reviews the open-source Python library we wrote, `gsmp_vglm`, including discussions of underlying libraries used, class structures and reviews of the primary code used for sample generation, parameter estimation and goodness-of-fit as well as example files.
- **Chapter 7** summarizes our work and lays out several interesting future directions for this work going forward.

By addressing both the theoretical foundations and practical applications of the GSMP-VGLM, this dissertation provides a comprehensive framework for modeling complex stochastic events with broad applicability across disciplines.

Chapter 2

Theoretical Derivations of the GSMP-VGLM Model

In this chapter, we begin by reviewing the “genealogy” of our work by discussing the foundational models that deal with the sum, maxima and duration of stochastic events. After a review we establish the formulation of our primary contribution, the GSMP-VGLM model and proceed to establish the statistical properties of maximum likelihood estimators (MLE’s) of the parameters in various cases before leading to a situation where all parameters are unknown. We show the efficacy of these estimators through numerous simulation studies. Finally, we conclude the chapter by deriving the Goodness-of-Fit procedures and reviewing their performance through simulation studies.

2.1 Historical Development of GSMP Models

The modeling of stochastic episodes characterized by random sums, maxima, and durations has evolved through a series of increasingly sophisticated frameworks addressing specific aspects of the problem we study, with each motivated by diverse scientific applications.

The earliest such work dates back to of Kozubowski and Panorska (2005) and Biondi et al. (2005) that sought to model the joint distribution of the duration and magnitude of events built from observations $X_1, X_2, \dots \stackrel{iid}{\sim} \mathcal{Exp}(\lambda)$ with $N \sim \mathcal{Geo}(p)$ and independent of the sequence X_i . The resulting bivariate stochastic vector $(N, \sum_{i=1}^N X_i)$ was named **Bivariate Exponential-Geometric (BEG)** model, for its **exponential** and **geometric** marginals. These two papers established the statistical properties of the estimators and demonstrated the utility of these approaches in finance in Kozubowski and Panorska (2005) and in climate and hydrology in Biondi et al. (2005).

The next step came with the development of the **Bivariate Truncated Logistic-Geometric (BTLG)** model. This work defined a bivariate vector $(N, Y =$

$\max_{i=1,\dots,n} X_i$) where $X_1, X_2, \dots \stackrel{iid}{\sim} \mathcal{Exp}(\lambda)$ and $N \sim \mathcal{Geo}(p)$ and independent of the sequence $\{X_i\}$. Statistical properties and estimation for the BTLG mode were developed in Kozubowski and Panorska (2007). A subsequent paper by Biondi et al. (2007) applied this model to a 2300-year long dendroclimatic record from the Eastern Sierra Nevada headwaters. Another paper by Saito et al. (2008) extended this framework to streamflow reconstruction using tree-ring data, showing how the BTLG model could capture peak-flow magnitudes and durations in paleohydrological contexts.

Next significant development was the *exact* distribution of the sum and maximum of n iid exponential random variables in Qaedan et al. (2012). This paper was a major step forward for two reasons. First, it linked the magnitude of episodes to peak intensity. Secondly, it laid the final piece needed to unite duration, magnitude and maximum into one trivariate model. This was achieved in Kozubowski et al. (2011) with the publication of the **T**rivariate distribution with **E**xponential, **T**runcated **L**ogistic, and **G**eometric marginals (TETLG) model (see Section 1.4.1 for the mathematical formulation). This significant work established the trivariate stochastic vector composed of the magnitude, maximum and duration of events built from N iid exponential

observations X_1, X_2, \dots , where $N \sim \text{Geo}(p)$ was independent of the sequence $\{X_i\}$. The paper developed properties of the TETLG vector, its marginal and conditional distributions, as well as MLEs of the parameters with their asymptotic properties.

Within the field of climatology, the work of Cavanaugh et al. (2015) demonstrated that daily precipitation data at many weather stations around the world followed a power law distribution. This finding meant that the work we described earlier that assumed exponential observations would fail to capture the true distribution of total daily precipitation. To address this, Arendarczyk et al. (2018a) constructed a bivariate model of the sum and duration of stochastic events similar to the BEG model. However, rather than *iid* Exponential observations, their new model, called **Bivariate distribution with Lomax and Geometric marginals (BLG)**, had the observations X_1, \dots, X_n within an event of duration n follow a multivariate Pareto Type II (Lomax) distribution. Next, Arendarczyk et al. (2018b) developed a trivariate stochastic vector called **Geometric Sum and Maximum of Pareto variables (GSMP)** model, where the distribution of the observations within an event, given the duration of an events $N = n$, followed the multivariate Lomax

distribution, and N was geometric and independent of $\{X_i\}$. The GSMP model was an extension of the TETLG model to events built from heavy tailed observations. This paper developed properties of the GSMP model (also described in Section 1.4.2), MLE estimators of its parameters, and presented an application of the GSMP model to stock returns.

The last paper closely connected to our work was the application of a Vector Generalized Linear Model (VGLM) to the TETLG framework, demonstrated in Zuniga et al. (2021). This allowed different events to have unique parameterizations rather than rely on a restrictive assumption that all parameters must remain static from event to event. This paper established the MLE estimators of the parameters and presented a Goodness-of-Fit procedure for the TETLG-VGLM model.

This leads us to our work, where we will combine the flexibility of the VGLM framework with the heavy tail GSMP model, making the resulting model appropriate for the motivating episodes of heavy precipitation.

2.2 The VGLM model

The foundation of our work is the Geometric Sum and Maximum of Pareto Variables (GSMP) model, introduced and studied in Arendarczyk et al. (2018). We extend the GSMP model to a Vector Generalized Linear Model (VGLM) framework. We begin with the definition of the base GSMP model from Arendarczyk et al. (2018).

Let (X_1, \dots, X_n) be a random vector with a Pareto II (or Lomax) distribution with the probability density function (PDF):

$$f_{\alpha, \lambda, n}(x_1, \dots, x_n) = \frac{(\alpha\lambda)^n \Gamma\left(\frac{1}{\alpha} + n\right)}{\Gamma\left(\frac{1}{\alpha}\right)} \left(1 + \alpha\lambda \sum_{i=1}^n x_i\right)^{-\frac{1}{\alpha} - n}, \quad x_i \in \mathbb{R}^+, \quad (2.1)$$

where $i = 1, \dots, n$, $\lambda > 0$ is a scale parameter and $\alpha \geq 0$ is a tail parameter.

The one-dimensional marginal distributions of Pareto II are univariate Lomax distributions (see, e.g., Arnold, 1983), with the survival function (SF)

$$P(X_i > x) = (1 + \alpha\lambda x)^{-\frac{1}{\alpha}}, \quad x \in \mathbb{R}^+. \quad (2.2)$$

Note that parameter α controls the tail of the SF, that is the larger the α , the *heavier* the tail of X_i . When $\alpha = 0$ we understand the Lomax distribution

as the (limiting) exponential distribution.

The vector (X_1, \dots, X_n) has the following stochastic representation:

$$(X_1, \dots, X_n) \stackrel{d}{=} \left(\frac{E_1}{Z}, \dots, \frac{E_n}{Z} \right), \quad (2.3)$$

where the $\{E_i\}$ are independent and identically distributed (IID) exponential variables with common PDF:

$$f(x) = \lambda e^{-\lambda x}, \quad x \in \mathbb{R}^+, \quad (2.4)$$

and the random variable Z , independent of the $\{E_i\}$, has a Gamma distribution with both shape and scale parameters equal to $1/\alpha$ and the PDF given by:

$$g_\alpha(x) = \frac{(1/\alpha)^{1/\alpha}}{\Gamma(1/\alpha)} x^{\frac{1}{\alpha}-1} e^{-\frac{1}{\alpha}x}, \quad x \in \mathbb{R}^+. \quad (2.5)$$

Z is often called the *mixing* random variable as it introduces a dependence structure to the vector (X_1, \dots, X_n) . In other words, the X_i are identically distributed, but not independent. Note that all univariate marginals of (X_1, \dots, X_n) have Pareto II (Lomax) distributions.

The GSMP vector (X, Y, N) with parameters α , λ and p , denoted as $(X, Y, N) \sim \mathcal{GSMP}(\alpha, \lambda, p)$, is defined by:

$$(X, Y, N) \stackrel{d}{=} \left(\sum_{i=1}^N X_i, \bigvee_{i=1}^N X_i, N \right), \quad (2.6)$$

where, conditionally on $N = n$, the vector (X_1, \dots, X_n) has the multivariate Pareto distribution defined above and N is a geometric random variable independent of the $\{X_i\}$ with probability mass function:

$$p_n = P(N = n) = p(1 - p)^{n-1}, \quad n \in \mathbb{N},$$

where $0 < p < 1$ is the probability of success. The PDF of the vector $(X, Y, N) \sim \mathcal{GSMP}(\alpha, \lambda, p)$ is given by:

$$g_{\alpha, \lambda, p}(x, y, n) = \frac{\Gamma\left(n + \frac{1}{\alpha}\right)}{\Gamma\left(\frac{1}{\alpha}\right)} (\alpha\lambda)^n H(x, y, n) (1 + \alpha\lambda x)^{-\frac{1}{\alpha} - n} p(1 - p)^{n-1}, \quad (2.7)$$

with

$$H(x, y, n) = \begin{cases} \sum_{s=1}^k \frac{a_n(s)}{(n-1)!} (x - sy)^{n-2} & \text{for } n \geq 2, (x, y) \in \mathcal{S}_k, k = 1, \dots, n-1, \\ 1 & \text{for } n = 1, (x, y) \in \mathcal{S}_0, \\ 0 & \text{otherwise,} \end{cases} \quad (2.8)$$

and

$$a_n(s) = n(n-1) \binom{n-1}{s-1} (-1)^{s+1}, \quad n \geq 2, \quad s = 1, \dots, n-1, \quad (2.9)$$

where $\mathcal{S}_0 = \{(x, y) \in \mathbb{R}^2 : x = y, x > 0\}$, and

$$\mathcal{S}_k = \left\{ (x, y) \in \mathbb{R}^2 : 0 \leq \frac{1}{k+1}x \leq y \leq \frac{1}{k}x \right\}, \quad k = 1, 2, \dots, n-1. \quad (2.10)$$

In the limiting case when $\alpha = 0$, the GSMP model reduces to the TETLG model introduced in Kozubowski et al. (2011), where the random variables (X_1, \dots, X_n) are IID exponential with PDF given by (2.4).

The goal of this research is to develop a VGLM for the random vector $(X, Y, N) \sim \mathcal{GSMP}(\alpha, \lambda, p)$ so that its parameters may depend on covariates. We begin by formulating λ and p as functions of covariates.

Let $(X_1, Y_1, N_1), (X_2, Y_2, N_2), \dots, (X_\ell, Y_\ell, N_\ell)$ be independent observations from a $\mathcal{GSMP}(\alpha, \lambda_i, p_i)$ distribution with $i = 1, 2, \dots, \ell$, respectively.

Since both λ and p have restricted values, we use the following link functions:

$$p_i = \frac{e^{\beta^\top \mathbf{z}_i}}{1 + e^{\beta^\top \mathbf{z}_i}}, \quad i = 1, 2, \dots, \ell, \quad (2.11)$$

where $\boldsymbol{\beta} = [\beta_0, \beta_1, \dots, \beta_m]^\top \in \mathbb{R}^{m+1}$ is a vector of unknown coefficients and $\mathbf{Z}_i = [1, Z_{i,1}, \dots, Z_{i,m}]^\top \in \mathbb{R}^{m+1}$ is a vector of covariates for p_i , and

$$\lambda_i = e^{\boldsymbol{\eta}^\top \mathbf{W}_i}, \quad i = 1, 2, \dots, \ell, \quad (2.12)$$

where $\boldsymbol{\eta} = [\eta_0, \eta_1, \dots, \eta_k]^\top \in \mathbb{R}^{k+1}$ is a vector of unknown coefficients and $\mathbf{W}_i = [1, W_{i,1}, \dots, W_{i,k}]^\top \in \mathbb{R}^{k+1}$ is a vector of covariates for λ_i .

We call this VGGLM $\mathcal{V}\mathcal{G}\mathcal{S}\mathcal{M}\mathcal{P}(\alpha, \boldsymbol{\eta}, \boldsymbol{\beta})$. In the next section, we describe the estimation of parameters for this model.

2.3 Estimation of parameters

We employ the maximum likelihood approach (MLE) for parameter estimation. Let $(X_1, Y_1, N_1), (X_2, Y_2, N_2), \dots, (X_\ell, Y_\ell, N_\ell)$ be independent observations from $\mathcal{G}\mathcal{S}\mathcal{M}\mathcal{P}(\alpha, \lambda_i, p_i)$, $i = 1, 2, \dots, \ell$, respectively.

The log-likelihood function of this sample takes the following form:

$$\begin{aligned} \mathcal{L}(\alpha, \boldsymbol{\lambda}, \mathbf{p}) = \log \left\{ \prod_{i=1}^{\ell} \frac{\Gamma(n_i + \frac{1}{\alpha})}{\Gamma(\frac{1}{\alpha})} \alpha^{n_i} \lambda_i^{n_i} (1 + \alpha \lambda_i x_i)^{-\frac{1}{\alpha} - n_i} \right\} \\ + \log \prod_{i=1}^{\ell} p_i (1 - p_i)^{n_i - 1} + \sum_{i=1}^n \log(H(x_i, y_i, n_i)). \end{aligned} \quad (2.13)$$

After substituting equations (2.11) and (2.12) into (2.13), we obtain:

$$\mathcal{L}(\alpha, \boldsymbol{\eta}, \boldsymbol{\beta}) = g(\alpha, \boldsymbol{\eta}) + h(\boldsymbol{\beta}) + C, \quad (2.14)$$

where

$$g(\alpha, \boldsymbol{\eta}) = \log \left\{ \prod_{i=1}^{\ell} \frac{\Gamma(n_i + \frac{1}{\alpha})}{\Gamma(\frac{1}{\alpha})} \alpha^{n_i} e^{(\boldsymbol{\eta}^\top \mathbf{w}_i) n_i} \left(1 + \alpha e^{\boldsymbol{\eta}^\top \mathbf{w}_i} x_i\right)^{-\frac{1}{\alpha} - n_i} \right\}, \quad (2.15)$$

$$h(\boldsymbol{\beta}) = \boldsymbol{\beta}^\top \sum_{i=1}^{\ell} \mathbf{z}_i - \sum_{i=1}^{\ell} n_i \log \left(1 + e^{\boldsymbol{\beta}^\top \mathbf{z}_i}\right), \quad (2.16)$$

and

$$C = \sum_{i=1}^{\ell} \log(H(x_i, y_i, n_i)). \quad (2.17)$$

Note that the left-hand part of Equation 2.15 can be simplified which will lead to a different formulation used later in this dissertation. Specifically, we have:

$$\begin{aligned} \log \left\{ \prod_{i=1}^{\ell} \frac{\Gamma(n_i + \frac{1}{\alpha})}{\Gamma(\frac{1}{\alpha})} \alpha^{n_i} \right\} &= \sum_{i=1}^{\ell} \log \left\{ \frac{\Gamma(n_i + \frac{1}{\alpha})}{\Gamma(\frac{1}{\alpha})} \alpha^{n_i} \right\} \\ &= \sum_{i=1}^{\ell} \log \left\{ \frac{\left(\prod_{j=0}^{n_i-1} (j + \frac{1}{\alpha})\right) \Gamma(\frac{1}{\alpha})}{\Gamma(\frac{1}{\alpha})} \alpha^{n_i} \right\} \\ &= \sum_{i=1}^{\ell} \log \left\{ \frac{\left(\prod_{j=0}^{n_i-1} (j\alpha + 1)\right) \alpha^{n_i}}{\alpha^{n_i}} \right\} \\ &= \sum_{i=1}^{\ell} \log \left\{ \prod_{j=0}^{n_i-1} (j\alpha + 1) \right\}. \end{aligned}$$

Therefore, we can rewrite Equation 2.15 as

$$g(\alpha, \boldsymbol{\eta}) = \log \left\{ \prod_{i=1}^{\ell} \left(\prod_{j=0}^{n_i-1} (j\alpha + 1) \right) e^{(\boldsymbol{\eta}^\top \mathbf{w}_i)n_i} \left(1 + \alpha e^{\boldsymbol{\eta}^\top \mathbf{w}_i x_i} \right)^{-\frac{1}{\alpha} - n_i} \right\}. \quad (2.18)$$

In order to establish the MLEs, we will proceed by considering several cases wherein some parameters are assumed to be known while others require estimation. The three scenarios are as follows:

1. First, we find MLE of parameters α and $\boldsymbol{\beta}$ under assumption that $\boldsymbol{\eta}$ are known.
2. In the second scenario we assume the knowledge of parameter α and find MLE for parameters $\boldsymbol{\eta}$ and $\boldsymbol{\beta}$.
3. In the last considered scenario we examine the most general model, where α , $\boldsymbol{\eta}$ and $\boldsymbol{\beta}$ are unknown parameters.

Note, that C does not depend on the parameters α , $\boldsymbol{\beta}$ and $\boldsymbol{\eta}$. Furthermore, the structure of the log-likelihood function (2.14) implies that we can estimate $\boldsymbol{\beta} = [\beta_0, \beta_1, \dots, \beta_m]^\top$ separately from α and $\boldsymbol{\eta} = [\eta_0, \eta_1, \dots, \eta_k]^\top$. Therefore, to find MLEs we separately maximize function $g(\alpha, \boldsymbol{\eta})$ with respect to arguments α and $\boldsymbol{\eta}$, and function $h(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$.

The estimation of $\boldsymbol{\beta}$ in our VGLM is the same as the estimation of $\boldsymbol{\beta}$ in the VGLM for the TETLG model discussed in Zuniga (2021). Thus, we use their result that under certain conditions, the MLEs $[\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_k]^\top$ of $[\beta_0, \beta_1, \dots, \beta_k]^\top$ exist and are unique, see Theorem 2.2 in Zuniga et. al (2021). We cite this result below for completeness of the exposition.

Theorem 2.3.1 *Suppose that in the above setting we have $n > m$ and there are at least $m + 1$ data points (x_i, y_i, n_i) where $n_i > 1$. Further, assume that the sub-matrix of the $n \times (m + 1)$ matrix $\mathbf{Z} = [z_{i,j}]$ whose rows correspond to the i values where $n_i > 1$ is of full rank. Then the function $h(\cdot)$ in (2.16) attains a unique maximum value at $\hat{\boldsymbol{\beta}} \in \mathbb{R}^{m+1}$, which satisfies the system of equations*

$$\sum_{i=1}^{\ell} z_{i,j} = \sum_{i=1}^{\ell} n_i z_{i,j} \frac{e^{\hat{\boldsymbol{\beta}}^\top \mathbf{z}_i}}{1 + e^{\hat{\boldsymbol{\beta}}^\top \mathbf{z}_i}}, j = 0, \dots, m. \quad (2.19)$$

2.3.1 Case 1: Known α

In the scenario considered in this section we assume the knowledge of parameter $\alpha \in [0, \infty)$ and find MLE for parameters $\boldsymbol{\eta} \in \mathbb{R}^{k+1}$ and $\boldsymbol{\beta} \in \mathbb{R}^{m+1}$.

Let $\alpha \in [0, \infty)$. In order to find $\hat{\boldsymbol{\eta}} \in \mathbb{R}$ that maximizes the function $g_\alpha(\boldsymbol{\eta}) := g(\alpha, \boldsymbol{\eta})$, $\boldsymbol{\eta} \in \mathbb{R}$ first observe that

$$g_\alpha(\boldsymbol{\eta}) = \log \left\{ \prod_{i=1}^{\ell} \frac{\Gamma(n_i + \frac{1}{\alpha})}{\Gamma(\frac{1}{\alpha})} \alpha^{n_i} \right\} + f_\alpha(\boldsymbol{\eta}), \quad (2.20)$$

where

$$f_\alpha(\boldsymbol{\eta}) = \boldsymbol{\eta}^\top \sum_{i=1}^{\ell} n_i \mathbf{w}_i - \sum_{i=1}^{\ell} \left(\frac{1}{\alpha} + n_i \right) \log \left(1 + \alpha e^{\boldsymbol{\eta}^\top \mathbf{w}_i} x_i \right) \quad \boldsymbol{\eta} \in \mathbb{R}^{k+1}. \quad (2.21)$$

Thus, in the case considered, we shall find $\hat{\boldsymbol{\eta}} \in \mathbb{R}^k$ that maximizes $f_\alpha(\boldsymbol{\eta})$.

Theorem 2.3.2 *Consider $\mathcal{VGSMP}(\alpha, \boldsymbol{\eta}, \boldsymbol{\beta})$ with α known. Moreover, suppose that $n > k$ and $n \times (k + 1)$ matrix $\mathbf{W} = [w_{i,l}]$, $i \in \{1, 2, \dots, n\}$, $l \in \{0, 1, \dots, k\}$ is of full rank, so that there is a subset of $k + 1$ linearly independent vectors among the \mathbf{w}_i . Then the MLE $\hat{\boldsymbol{\beta}} \in \mathbb{R}^{m+1}$ of $\boldsymbol{\beta}$ exists, is unique, and satisfies the system of equations (2.19). The MLE and $\hat{\boldsymbol{\eta}} \in \mathbb{R}^{k+1}$ of $\boldsymbol{\eta}$ also exists, is unique and satisfies the following system of equations*

$$\sum_{i=1}^n n_i w_{i,l} = \sum_{i=1}^n (1 + \alpha n_i) \left[\frac{w_{i,l} x_i e^{\boldsymbol{\eta}^\top \mathbf{w}_i}}{1 + \alpha x_i e^{\boldsymbol{\eta}^\top \mathbf{w}_i}} \right], \quad l = 0, 1, \dots, k. \quad (2.22)$$

The proof of Theorem 2.3.2 is based on the following lemmas.

Lemma 1 *For any $\alpha, n, x > 0$, the function*

$$f_{\alpha,n,x}(y) = ny - \left(\frac{1}{\alpha} + n \right) \log(1 + \alpha x e^y), \quad y \in \mathbb{R}, \alpha, n, x > 0. \quad (2.23)$$

strictly concave on \mathbb{R} with the limit of $-\infty$ as $y \rightarrow \pm\infty$.

Proof. Since $\alpha, n, x > 0$ and since $(\alpha x e^y) > 0$ for all $y \in \mathbb{R}$, then $\frac{d^2}{dy^2} f_{\alpha,n,x}(y) = - \left(\frac{1}{\alpha} + n \right) \left(\frac{\alpha x e^y}{(1 + \alpha x e^y)^2} \right) < 0$ for all $y \in \mathbb{R}$ and $f_{\alpha,n,x}(y)$ is therefore strictly concave. The limits at $\pm\infty$ can be obtained by standard calcu-

lations.

□

The following result that is Lemma 6.3 in Zuniga et al. (2021), is used in the proof of Theorem 2.3.2 and is included here for convenience of the reader.

Lemma 2 *Suppose that $k, n \in \mathbb{N}$ with $n \geq k$ and let $\{\mathbf{w}_i\}$ where $\mathbf{w}_i = [W_{i,1}, \dots, W_{i,k}]^\top$ and $i \in A_n = \{1, \dots, n\}$ be a set of n vectors in \mathbb{R}^k . Further, suppose that the $n \times k$ matrix $\mathbf{W} = [w_{i,j}]$ is of full rank, so that there is a subset of k linearly independent vectors among the \mathbf{w}_i . Then, there exists $\delta > 0$ such that for each non-zero $\boldsymbol{\eta} = [\eta_1 \dots, \eta_k]^\top \in \mathbb{R}^k$ there exists $i \in A_n$ such that*

$$|\boldsymbol{\eta}^\top \mathbf{w}_i| \geq \delta \|\boldsymbol{\eta}\| = \left(\sum_{j=1}^k \eta_j^2 \right)^{1/2} \quad (2.24)$$

Proof of Theorem 2.3.2 The existence and uniqueness of $\hat{\boldsymbol{\beta}}$ is guaranteed by Theorem 2.3.1. In order to prove the existence and uniqueness of $\hat{\boldsymbol{\eta}}$, we shall proceed as follows. To prove the existence, we shall show that there exists $\hat{\boldsymbol{\eta}}$ that maximizes $f_\alpha(\boldsymbol{\eta})$ in 2.21, i.e., for any $M \in \mathbb{R}$ there exists $r > 0$ such that for any $\boldsymbol{\eta} \in \mathbb{R}^{k+1}$, that satisfies $\|\boldsymbol{\eta}\| > r$, we have $f_\alpha(\boldsymbol{\eta}) \leq M$.

Let $M \in \mathbb{R}$. First, observe that for a given $\boldsymbol{\eta} \in \mathbb{R}^{k+1}$, we have

$$f_\alpha(\boldsymbol{\eta}) = \sum_{i=1}^{\ell} f_{\alpha, n_i, x_i}(\boldsymbol{\eta}^\top \mathbf{w}_i). \quad (2.25)$$

where $f_{\alpha, n_i, x_i}(\cdot)$, $i \in A_n := \{1, 2, \dots, n\}$ is defined in 2.23. Due to the Lemma

1 there exists $c > 0$ such that

$$f_{\alpha, n_i, x_i}(\boldsymbol{\eta}^\top \mathbf{w}_i) \leq c \quad (2.26)$$

for all $\boldsymbol{\eta}$ and $i \in A_n$. Furthermore, let $L = M - (n - 1)c$.

Since the limit of each $f_{\alpha, n_i, x_i}(y)$ is $-\infty$, as $y \rightarrow \pm\infty$, then by Lemma 1 there exists $t > 0$ such that for all $i \in A_n$ and $y \in \mathbb{R}$, that satisfy $|y| \geq t$, we have $f_{\alpha, n_i, x_i}(y) \leq L$.

Set $r = t/\delta$ where $\delta > 0$ is a constant which existence is guaranteed by Lemma 2. Then, due to Lemma 2, for a given $\boldsymbol{\eta} \in \mathbb{R}^{k+1}$, that satisfies $\|\boldsymbol{\eta}\| \geq r$ there exists $j \in A_n$ such that $|\boldsymbol{\eta}^\top \mathbf{w}_j| \geq \delta \|\boldsymbol{\eta}\| \geq t$, which follows that

$$f_{\alpha, n_j, x_j}(\boldsymbol{\eta}^\top \mathbf{w}_j) \leq L. \quad (2.27)$$

Finally, combining 2.27 and 2.26 with 2.25, for a given $\boldsymbol{\eta} \in \mathbb{R}^{k+1}$, that satisfies $\|\boldsymbol{\eta}\| \geq r$, we obtain

$$f_\alpha(\boldsymbol{\eta}) = f_{\alpha, n_j, x_j}(\boldsymbol{\eta}^\top \mathbf{w}_j) + \sum_{i \neq j}^{\ell} f_{\alpha, n_i, x_i}(\boldsymbol{\eta}^\top \mathbf{w}_i) \leq M + (n - 1)c + (n - 1)c = M.$$

This proves the existence of $\hat{\boldsymbol{\eta}} \in \mathbb{R}^{k+1}$ such that

$$f_{\alpha}(\hat{\boldsymbol{\eta}}) = S = \sup_{\boldsymbol{\eta} \in \mathbb{R}^{k+1}} f_{\alpha}(\boldsymbol{\eta}). \quad (2.28)$$

To show uniqueness, assume by contradiction that there exist $\hat{\boldsymbol{\eta}}_1 \neq \hat{\boldsymbol{\eta}}_2$ that satisfy 2.28 and let $\hat{\boldsymbol{\eta}}_{\lambda} = \lambda \hat{\boldsymbol{\eta}}_1 + (1 - \lambda) \hat{\boldsymbol{\eta}}_2$ for some $\lambda \in (0, 1)$. By Lemma 1 all the functions $f_{\alpha, n_i, x_i}(y)$ are strictly concave, hence $f_{\alpha, n_i, x_i}(\boldsymbol{\eta}^{\top} \mathbf{w})$ is strictly concave as a function of $\boldsymbol{\eta} \in \mathbb{R}^{k+1}$ for each $i \in A_n$, which by representation 2.25 follows that $f_{\alpha}(\boldsymbol{\eta})$ is strictly concave. Thus,

$$f_{\alpha}(\hat{\boldsymbol{\eta}}_{\lambda}) > \lambda f_{\alpha}(\hat{\boldsymbol{\eta}}_1) + (1 - \lambda) f_{\alpha}(\hat{\boldsymbol{\eta}}_2) = S,$$

which contradicts 2.28. This completes the proof.

Evaluation of Estimators

To assess the performance of the MLEs in the case where α is known, we performed a simulation study where several different sample sizes were generated from various \mathcal{VGSMP} distributions where α was known but varied. Doing so would allow us to understand the impact of both sample size, ℓ and α on the quality of estimation. Specifically, we would repeat 500 draws from a $\mathcal{VGSMP}(\alpha, \boldsymbol{\eta} = [1, 0.5, -0.25]^{\top}, \boldsymbol{\beta} = [2, -2, 2]^{\top})$ distribution where the sample size ranged between $\{20, 50, 100, 500, 1000, 5000\}$ and

$\alpha \in \{0.001, 1.1, 2.1\}$. Covariates were drawn as *iid* realizations from a $\mathcal{U}(0, 1)$ distribution. For each draw, the estimated parameter vector, $\hat{\boldsymbol{\eta}}$ was stored. Finally, the stored estimates $\hat{\boldsymbol{\eta}}$ were plotted in boxplots. The resulting boxplots are shown in Figures 2.1 ($\alpha = 0.001$), 2.2 ($\alpha = 1.1$) and 2.3 ($\alpha = 2.1$)

:

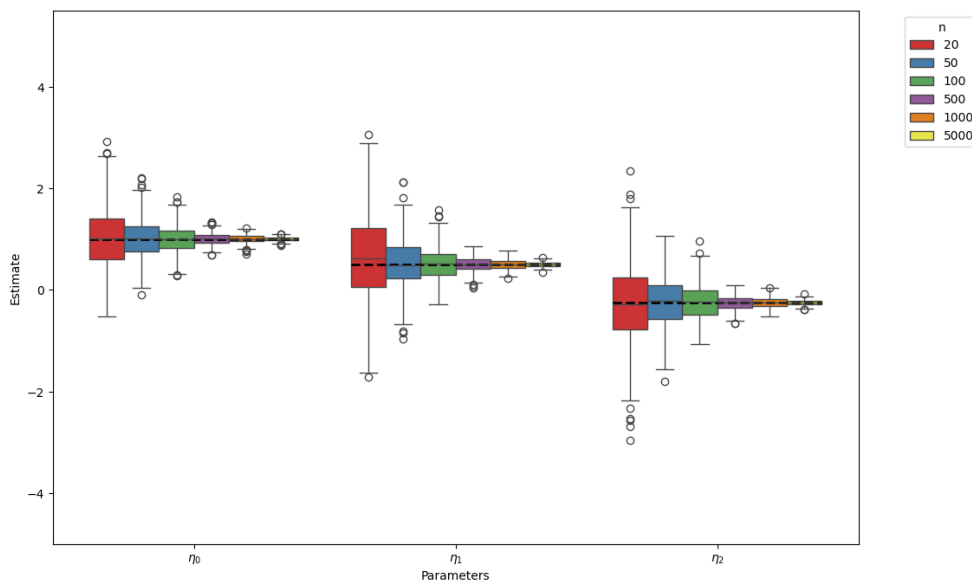


Figure 2.1: Boxplots of parameter estimates of each element of $\boldsymbol{\eta}$ (x-axis) when $\alpha = 0.001$. For each parameter, there are six boxplots associated with the smallest sample size (red) to the largest sample (yellow). The y-axis shows the actual values observed and the dashed horizontal lines show the true value of the data.

As can be seen in Figure 2.1, there is a common theme in the efficiency of the estimators: namely that, as the sample size increased, the variability of the estimator diminished, as one would expect. We also see that all estimators

appear fairly well centered around the true parameter estimates.

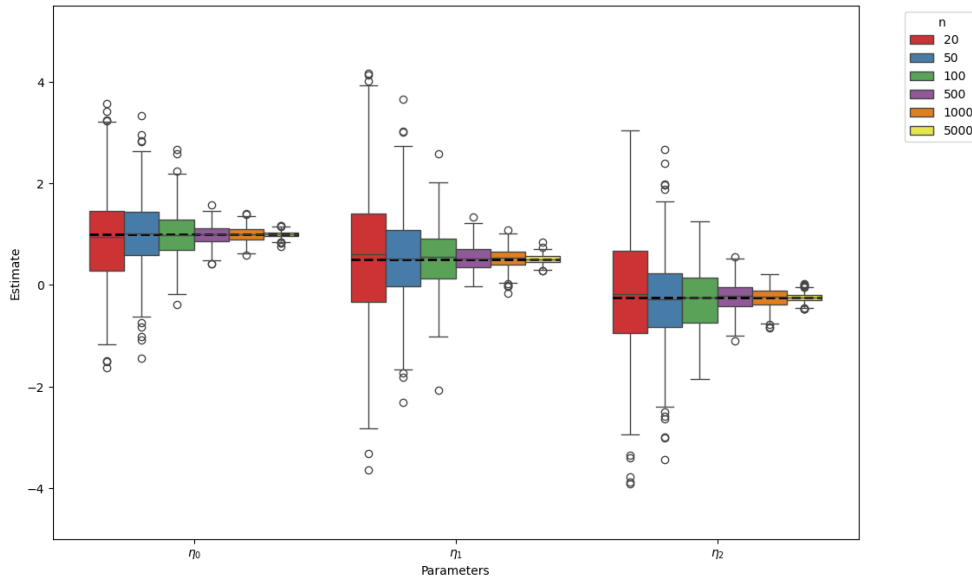


Figure 2.2: Boxplots of parameter estimates of each element of $\boldsymbol{\eta}$ (x-axis) when $\alpha = 1.1$. For each parameter, there are six boxplots associated with the smallest sample size (red) to the largest sample (yellow). The y-axis shows the actual values observed and the dashed horizontal lines show the true value of the data.

When α increased from 0.0001 to 1.1, the overall trend of estimator efficiency with increased sample size persists, but now the overall error of the estimators increases. Since the volatility of the underlying process X_i increases as α increases, this finding is consistent with expectation.

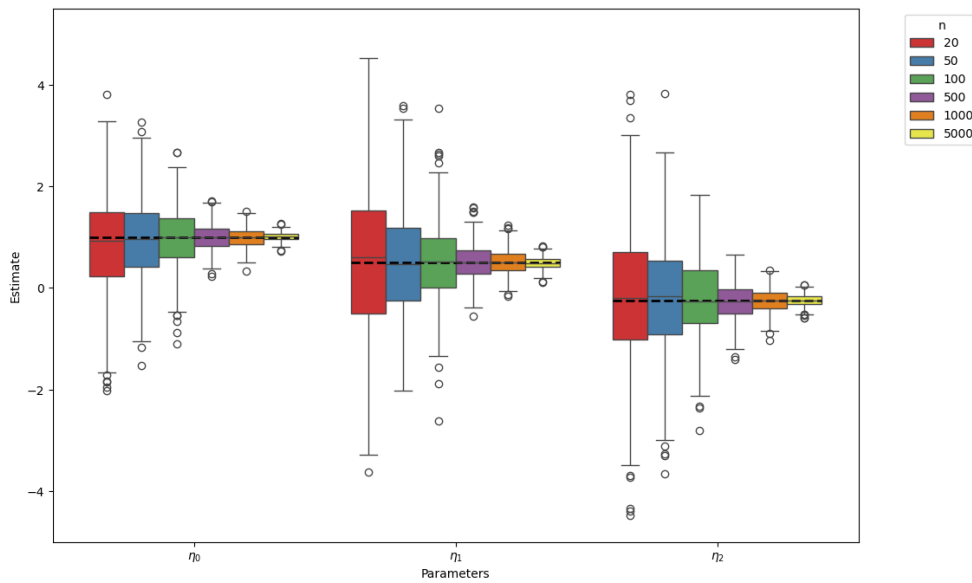


Figure 2.3: Boxplots of parameter estimates of each element of $\boldsymbol{\eta}$ (x-axis) when $\alpha = 2.1$. For each parameter, there are six boxplots associated with the smallest sample size (red) to the largest sample (yellow). The y-axis shows the actual values observed and the dashed horizontal lines show the true value of the data.

Lastly, when $\alpha > 2$, we still have unbiased estimates of the true parameters, but now with increased estimator error for a given sample size when compared to Figures 2.1 and 2.2. These findings agree with intuition that the estimator is more volatile when α grows large and the sample size decreases.

2.3.2 Case 2: Known $\boldsymbol{\eta}$

In this section, we examine the scenario where the parameter vector $\boldsymbol{\eta}$ is known, and we need to estimate $\alpha \in [0, \infty)$. This analysis provides valuable

insights into the behavior of the GSMP-VGLM model and the role of the tail parameter α .

When $\boldsymbol{\eta}$ is known, we can define $g_{\boldsymbol{\eta}}(\alpha) := g(\alpha, \boldsymbol{\eta})$ and study its behavior with respect to α . To maximize the log-likelihood function, we need to analyze how $g_{\boldsymbol{\eta}}(\alpha)$ behaves as α changes, particularly in the neighborhoods of $\alpha = 0$ and as $\alpha \rightarrow \infty$.

Proposition 2.3.3 *Let $g_{\boldsymbol{\eta}}(\alpha) := g(\alpha, \boldsymbol{\eta})$, where $g(\alpha, \boldsymbol{\eta})$ is given in 2.18 and define statistic S_n as follows:*

$$S_n := \frac{\sum_{i=1}^{\ell} (n_i - \lambda_i x_i)^2}{\sum_{i=1}^{\ell} n_i}.$$

Then

- (i) $\lim_{\alpha \rightarrow 0} g_{\boldsymbol{\eta}}(\alpha) = \sum_{i=1}^{\ell} [n_i \log(\lambda_i) - \lambda_i x_i]$.
- (ii) *If $S_n > 1$ then $g_{\boldsymbol{\eta}}(\alpha)$ is increasing in the right neighborhood of $\alpha = 0$.
If $S_n < 1$ then $g_{\boldsymbol{\eta}}(\alpha)$ is decreasing in the right neighborhood of $\alpha = 0$.*
- (iii) $\lim_{\alpha \rightarrow \infty} g_{\boldsymbol{\eta}}(\alpha) = -\infty$.

The proof of the proposition is as follows:

Proof. To prove (i) observe that $g(\alpha) = w_1(\alpha) + w_2(\alpha) + \sum_{i=1}^{\ell} n_i \log \lambda_i$,

where

$$\lim_{\alpha \rightarrow 0} w_1(\alpha) = \lim_{\alpha \rightarrow 0} \left[\sum_{i=1}^{\ell} \sum_{j=1}^{n_i-1} \log(j\alpha + 1) - \sum_{i=1}^{\ell} n_i \log(1 + \alpha \lambda_i x_i) \right] = 0,$$

and

$$\lim_{\alpha \rightarrow 0} w_2(\alpha) = \lim_{\alpha \rightarrow 0} \left[- \sum_{i=1}^{\ell} \frac{1}{\alpha} \log(1 + \alpha \lambda_i x_i) \right] = \sum_{i=1}^{\ell} [n_i \log(\lambda_i) - \lambda_i x_i],$$

where the last limit is obtained by applying L'Hopital's rule. This completes the proof of part (i).

To prove (ii) observe that $\frac{d}{d\alpha} g(\alpha) = a_1(u) + a_2(u)$, where

$$\begin{aligned} \lim_{\alpha \rightarrow 0} a_1(u) &= \lim_{\alpha \rightarrow 0} \sum_{i=1}^{\ell} \sum_{j=1}^{n_i-1} \frac{j}{j\alpha + 1} - \sum_{i=1}^{\ell} n_i \frac{\lambda_i x_i}{1 + \alpha \lambda_i x_i} \\ &= \sum_{i=1}^{\ell} \frac{n_i(n_i - 1)}{2} - \sum_{i=1}^{\ell} n_i \lambda_i x_i, \end{aligned} \tag{2.29}$$

and

$$\lim_{\alpha \rightarrow 0} a_2(u) = \lim_{\alpha \rightarrow 0} \frac{1}{\alpha^2} \sum_{i=1}^{\ell} \left[\log(1 + \alpha \lambda_i x_i) - \frac{\alpha \lambda_i x_i}{1 + \alpha \lambda_i x_i} \right] = \frac{1}{2} \sum_{i=1}^{\ell} (\lambda_i x_i)^2, \tag{2.30}$$

where the limit of $a_2(u)$ is obtained by applying L'Hopital's rule. Combining

the limits in (2.29) and (2.30) we obtain the following limit of the derivative of $g(\alpha)$ at zero:

$$\begin{aligned}
\lim_{\alpha \rightarrow 0} \frac{d}{d\alpha} g(\alpha) &= \lim_{\alpha \rightarrow 0} [a_1(u) + a_2(u)] = \sum_{i=1}^{\ell} \frac{n_i(n_i - 1)}{2} - \sum_{i=1}^{\ell} n_i \lambda_i x_i + \frac{1}{2} \sum_{i=1}^{\ell} (\lambda_i x_i)^2 \\
&= \frac{1}{2} \left[\sum_{i=1}^{\ell} (n_i - \lambda_i x_i)^2 - \sum_{i=1}^{\ell} n_i \right] \\
&= \frac{1}{2} (S_n - 1) \sum_{i=1}^{\ell} n_i.
\end{aligned} \tag{2.31}$$

Since $n_i > 0$ for all $i = 1, \dots, \ell$, then $\lim_{\alpha \rightarrow 0} \frac{d}{d\alpha} g(\alpha) > 0$ when $S_n > 1$, and $\lim_{\alpha \rightarrow 0} \frac{d}{d\alpha} g(\alpha) < 0$ when $S_n < 1$. This completes the proof of part (ii).

To prove (iii), observe that

$$g(\alpha) = u_1(\alpha) + u_2(\alpha) + u_3(\alpha), \tag{2.32}$$

where

$$\lim_{\alpha \rightarrow \infty} u_1(\alpha) = \lim_{\alpha \rightarrow \infty} \sum_{i=1}^{\ell} \sum_{j=0}^{n_i-1} \log \left(\frac{1}{\alpha} + j \right) = -\infty,$$

and

$$\lim_{\alpha \rightarrow \infty} u_2(\alpha) = \lim_{\alpha \rightarrow \infty} \left[- \sum_{i=1}^{\ell} n_i \log \left(\frac{1}{\alpha \lambda} + x_i \right) \right] = - \sum_{i=1}^{\ell} n_i \log x_i,$$

and

$$\lim_{\alpha \rightarrow \infty} u_3(\alpha) = \lim_{\alpha \rightarrow \infty} \left[-\frac{1}{\alpha} \sum_{i=1}^{\ell} \log(1 + \alpha \lambda_i x_i) \right] = 0,$$

where the last limit is obtained by applying L'Hospital's rule. The result follows.

□

The following Theorem is a straightforward consequence of Proposition 2.3.3 (iii).

Theorem 2.3.4 *Assume that $\boldsymbol{\eta}$ is known in the $\mathcal{VGSMP}(\alpha, \boldsymbol{\eta}, \boldsymbol{\beta})$ model. Then the MLEs $\hat{\alpha} \in [0, \infty)$ and $\hat{\boldsymbol{\beta}} \in \mathbb{R}^{m+1}$ of α and $\boldsymbol{\beta}$, respectively, always exist. Moreover, if condition $S_n > 1$ in Proposition 2.3.3 (ii) is satisfied, then $\hat{\alpha} > 0$.*

Though Theorem 2.3.4 establishes the existence of an MLE for α , it does *not* establish uniqueness. To better understand the behavior of the log-likelihood function $v(\alpha) = g_{\boldsymbol{\eta}}(\alpha)$ for different values of α , we conducted several simulation experiments. The goal was to see the behavior of $v(\alpha)$ close to zero and more globally.

For the first simulation we used a very small $\alpha = 0.001$. We plotted function $v(\alpha)$ close to zero in Figure 2.4 and then *zoomed out* to plot $v(\alpha)$ on a larger interval of α values in Figure 2.5.

Figure 2.4 shows the behavior of the log-likelihood function for data generated with $\alpha = 0.001$, focusing on a narrow range of values around the true value. This plot shows a clear peak near $\alpha = 0.001$, which supports the hypothesis of uniqueness of the MLE of α .

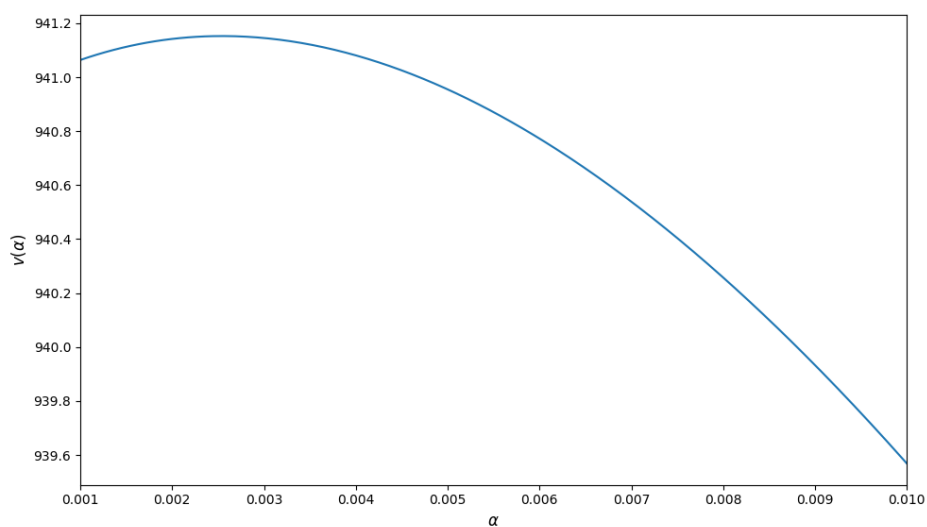


Figure 2.4: The log-likelihood function $v(\alpha)$ when $\alpha = 0.001$. The x-axis shows a range of possible α values from 0.001 to 0.01, while the y-axis shows the value of $v(\alpha)$

When we *zoom out* to examine a wider range of α values, as shown in Figure 2.5, we can see the global behavior of the function $v(\alpha)$. The graph shows that the maximum of $v(\alpha)$ is close to zero and that it is unique, which supports our hypothesis.

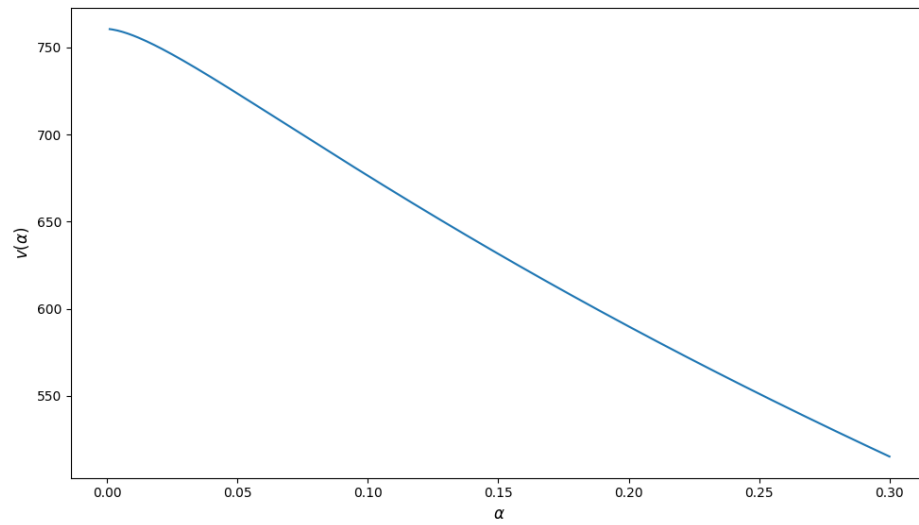


Figure 2.5: Similar to 2.4, but with the x-axis now from 0.001 to 0.3. The y-axis is scaled accordingly.

For data generated with a larger value of $\alpha = 1$, the behavior is similar but with the maximum occurring at a larger value of α , as demonstrated in Figure 2.6.

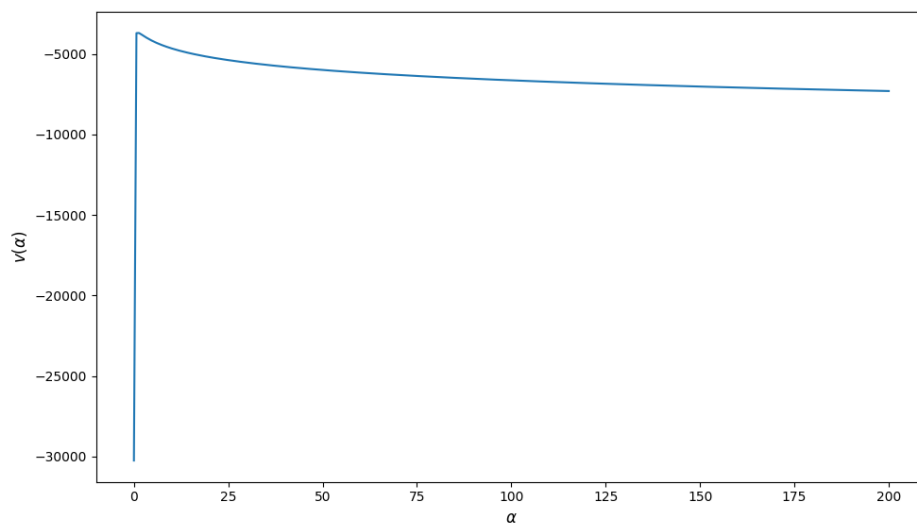


Figure 2.6: Varying values of α on the x-axis between 0.001 to 200 for $v(\alpha)$ when the true value of $\alpha = 1$. The y-axis shows the resulting value of the function $v(\alpha)$.

The graph shows one maximum, again supporting our hypotheses of unique MLE of α . We performed more simulation studies for various values of α and obtained the same behavior of $v(\alpha)$.

While these plots do not prove that the MLE of α is unique, they collectively not only provide strong empirical evidence for the theoretical results in Proposition 2.3.3, but also demonstrate support for our hypothesis that the log-likelihood function has a *unique* maximum.

The practical value of part (ii) of Proposition 2.3.3 is in using statistic S_n to determine if the underlying process follows an exponential or Pareto II distribution with $\alpha > 0$. Since the risk of large events for exponentially distributed random variable is qualitatively different (smaller) from that for a power-law tailed Pareto II (with $\alpha > 0$) determining whether the data comes from an exponential or heavy-tailed distribution is of primary practical importance. The idea here is to use statistic S_n in testing the null hypothesis that $\alpha = 0$ versus an alternative that $\alpha > 0$. While we do not present a full derivation of the testing process, we did check empirically how often using S_n would provide correct answer to the question whether the underlying distribution of the process generating events is exponential. Namely, we generated samples from the TETLG ($\alpha = 0$) or GSMP ($\alpha > 0$) VGLMs and computed the proportion of times that a statistic A defined below is negative. Specifically, we calculate:

$$A = \sum_{i=1}^n (n_i - \lambda_i x_i)^2 - \sum_{i=1}^n n_i \quad (2.33)$$

If $A > 0$, then $S_n > 1$ and the MLE of α is positive. If $A < 0$, then $S_n < 1$ and the MLE of α is 0, which corresponds to the TETLG model.

We summarized the results of the simulation study in Table 2.1. The first column in 2.1 contains α values for the generated samples, which varied from $\alpha = 0$ (exponential or TETLG case) to $\alpha = 0.5$ through increasing values of α . The first row of the table provides the sample sizes which varied from 20 to 1000. For each combination of α and sample size we generated 100 values of statistic A , that is we generates 100 samples from either TETLG or GSMP models with 2 covariates. We drew the values of covariates from uniform distribution on $(0, 1)$. The elements of the table are the proportions of time that statistic $A < 0$, that is the proportion of times statistic A (or S_n) suggests that the sample comes from the exponential distribution.

α	sample Size				
	20	50	100	500	1000
0.0000	0.64	0.56	0.48	0.44	0.40
0.0001	0.60	0.59	0.50	0.40	0.36
0.0010	0.56	0.52	0.52	0.43	0.35
0.0100	0.51	0.34	0.29	0.05	0.00
0.1000	0.12	0.00	0.00	0.00	0.00
0.2000	0.01	0.01	0.00	0.00	0.00
0.5000	0.00	0.00	0.00	0.00	0.00

Table 2.1: The table shows the proportion of times (out of 100 simulations) that the statistic $A < 0$ for different true values of α (rows) and sample sizes (columns).

The results in Table 2.1 show that as the sample size increases and as α

increases, the proportion of $A < 0$ signaling exponential model approaches zero. In fact, we have reasonable results for relatively small $\alpha = 0.1$ and relatively small sample size of 50. The simulation results demonstrate that the statistic A becomes increasingly effective at correctly identifying the TETLG model ($\alpha = 0$) versus the GSMP model ($\alpha > 0$) as the sample size and α increase. For $\alpha = 0$, the proportion of times that $A < 0$ remains relatively high even for large sample size of 1000, which means that statistic A is not very effective when the data are drawn from the exponential model.

We did not pursue testing using statistic A or S_n as there is a very good test for exponential versus Pareto distribution presented in Kozubowski et al. (2009).

2.3.3 Case 3: α and η Unknown

In this section, we address the most general and challenging case of parameter estimation in the GSMP-VGLM model, where all parameters— α , η , and β —are unknown and must be estimated simultaneously from the observed data. This scenario is of particular practical importance, as it represents the typical situation encountered in real-world applications where no prior knowledge of parameter values is available.

To maximize $g(\alpha, \boldsymbol{\eta})$, we first show that for a fixed $\alpha \geq 0$, there exists a unique $\boldsymbol{\eta}(\alpha)$ that maximizes $g(\alpha, \boldsymbol{\eta})$ with respect to $\boldsymbol{\eta}$. Therefore, our goal is to maximize $f_\alpha(\boldsymbol{\eta})$ defined in 2.21 with respect to $\boldsymbol{\eta}$. Due to Theorem 2.3.2, under the condition of a fixed $\alpha \geq 0$ and some additional assumptions on the matrix \mathbf{W} , there exists a unique $\boldsymbol{\eta}(\alpha)$ that maximizes $f_\alpha(\boldsymbol{\eta})$. The next step is to maximize

$$v(\alpha) = g(\alpha, \boldsymbol{\eta}(\alpha)), \quad (2.34)$$

with respect to $\alpha \geq 0$, where $\boldsymbol{\eta}(\alpha)$ satisfies the system of equations 2.22. To accomplish this, in the following Proposition we study the behavior of the function $v(\cdot)$ in the right neighborhood of 0 and at ∞ .

Proposition 2.3.5 *Suppose that we have $n > k$ and $n \times (k + 1)$ matrix $\mathbf{W} = [w_{i,l}], i \in \{1, 2, \dots, n\}, l \in \{0, 1, \dots, k\}$ is of full rank, so that there is a subset of $k + 1$ linearly independent vectors among the \mathbf{w}_i . Then,*

(i)

$$v(0) = \boldsymbol{\eta}(0)^\top \sum_{i=1}^{\ell} n_i \mathbf{w}_i - \sum_{i=1}^{\ell} n_i x_i e^{\boldsymbol{\eta}(0)^\top \mathbf{w}_i}, \quad (2.35)$$

where $\boldsymbol{\eta}(0)$ satisfies the following system of equations

$$\sum_{i=1}^{\ell} n_i w_{i,l} = \sum_{i=1}^{\ell} w_{i,l} x_i e^{\boldsymbol{\eta}(0)^\top \mathbf{w}_i} \quad l = 0, 1, \dots, k. \quad (2.36)$$

(ii) $\lim_{\alpha \rightarrow \infty} v(\alpha) = -\infty$.

The proof of Proposition 2.3.5 is as follows:

Proof. To prove (i), note that due to Theorem 2.3.2, $\boldsymbol{\eta}(\alpha)$ that maximizes $g(\alpha, \boldsymbol{\eta}(\alpha))$, for a given $\alpha \in [0, \infty)$, satisfies the following system of equations

$$\sum_{i=1}^{\ell} n_i w_{il} = \sum_{i=1}^{\ell} (1 + \alpha n_i) \left[\frac{w_{il} x_i e^{\boldsymbol{\eta}^\top \mathbf{w}_i}}{1 + \alpha x_i e^{\boldsymbol{\eta}^\top \mathbf{w}_i}} \right], \quad l = 0, 1, \dots, k.$$

Converging with $\alpha \rightarrow 0$ we obtain 2.36. In order to find the limit of $v(\alpha)$ as $\alpha \rightarrow 0$ observe that $\sum_{i=1}^{\ell} \sum_{j=1}^{n_i-1} \log(j\alpha + 1)$ vanishes at $\alpha = 0$. Moreover, by applying L'Hopital's rule we obtain

$$\lim_{\alpha \rightarrow 0} \sum_{i=1}^{\ell} \left(\frac{1}{\alpha} + n_i \right) \log \left(1 + \alpha x_i e^{\boldsymbol{\eta}(\alpha)^\top \mathbf{w}_i} \right) = \sum_{i=1}^{\ell} n_i x_i e^{\boldsymbol{\eta}(0)^\top \mathbf{w}_i},$$

where $\boldsymbol{\eta}(0)$ satisfies 2.36. To prove (ii) consider the following decomposition $v(\alpha) = v_1(\alpha) + v_2(\alpha) + v_3(\alpha)$, where each of the elements $v_1(\alpha), v_2(\alpha), v_3(\alpha)$ is investigated separately. First, observe that

$$v_1(\alpha) = \sum_{i=1}^{\ell} \sum_{j=0}^{n_i-1} \log \left(\frac{1}{\alpha} + j \right) \rightarrow -\infty, \quad \text{as } \alpha \rightarrow \infty,$$

Moreover,

$$v_2(\alpha) = - \sum_{i=1}^{\ell} n_i \log \left(\frac{1}{\alpha e^{\boldsymbol{\eta}(\alpha)^\top \mathbf{w}_i}} + x_i \right) < - \sum_{i=1}^{\ell} n_i \log(x_i) < \infty,$$

where the first inequality holds by observation that $\frac{1}{\alpha e^{\boldsymbol{\eta}(\alpha)^\top \mathbf{w}_i}} > 0$, and the

second inequality is due to the fact that $(x_1, x_2, \dots, x_\ell)$ is a sample from multivariate Lomax distribution and hence $x_i > 0$, for all $i = 1, 2, \dots, \ell$ which follows that $\log(x_i)$ is a finite constant, for all $i = 1, 2, \dots, \ell$.

Analogously, for $\alpha > 0$, we have

$$v_3(\alpha) = - \sum_{i=1}^{\ell} \frac{1}{\alpha} \log \left(1 + \alpha e^{\boldsymbol{\eta}(\alpha)^\top \mathbf{w}_i} x_i \right) \leq - \sum_{i=1}^{\ell} \frac{1}{\alpha} \log(1) = 0.$$

This completes the proof. \square

The following Theorem is a straightforward consequence of Proposition 2.3.5 (ii).

Theorem 2.3.6 *Assume that $\alpha, \boldsymbol{\eta}, \boldsymbol{\beta}$ are unknown in the $\mathcal{VGLM}(\alpha, \boldsymbol{\eta}, \boldsymbol{\beta})$ model. Suppose that we have $n > k$ and $n \times (k + 1)$ matrix $\mathbf{W} = [w_{i,l}], i \in \{1, 2, \dots, n\}, l \in \{0, 1, \dots, k\}$ is of full rank, so that there is a subset of $k + 1$ linearly independent vectors among the \mathbf{w}_i . Then the MLEs $\hat{\alpha} \in [0, \infty), \hat{\boldsymbol{\eta}} \in \mathbb{R}^{k+1}$, and $\hat{\boldsymbol{\beta}} \in \mathbb{R}^{m+1}$ of $\alpha, \boldsymbol{\eta}$, and $\boldsymbol{\beta}$, respectively, always exist.*

Evaluation of Estimators

To assess the performance of the MLEs in the general case where all parameters are unknown, we conducted extensive simulation studies. We generated $\ell = 1000$ observations from a $\mathcal{VGLM}(\alpha, \boldsymbol{\eta} = [1, 0.5, -0.25]^\top, \boldsymbol{\beta} = [-1, -2, 0.5]^\top)$ and covariates generated as *iid* observations from a $\mathcal{U}(0, 1)$

distribution. To see the effect of different α , we varied α between 0 (TETLG) as well as $\{0.001, 0.1, 1, 5\}$. This process was repeated 100 times where all parameter estimates $(\hat{\alpha}, \hat{\eta}, \hat{\beta})$ were stored to create boxplots which are shown in Figures 2.7 (change with α), 2.8 (change with sample size for estimates of η_i) and 2.9 (change with sample size for estimates of β_i). Figure 2.7 shows the distribution of $\hat{\alpha}$ for different true values of α :

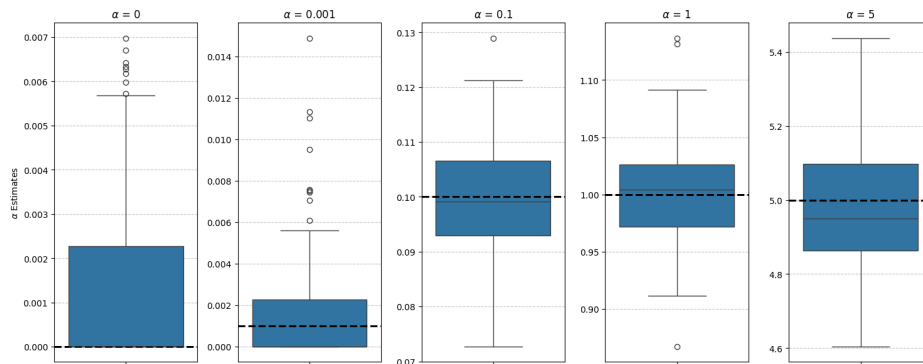


Figure 2.7: Boxplots of $\hat{\alpha}$ estimates when all parameters are unknown. Each boxplot is associated with a different value of α from $\alpha = 0$ (TETLG) to $\alpha = 5$. The y-axes in each boxplot vary for each value of α .

Figure 2.7 shows that the model is generally unbiased and that the volatility increases as α grows larger as expected. Note that when the true $\alpha = 0$, the solver cannot actually reach zero and often hits the lower bound of the search range. Figures 2.8 and 2.9 show the distributions of $\hat{\eta}$ and $\hat{\beta}$, respectively:

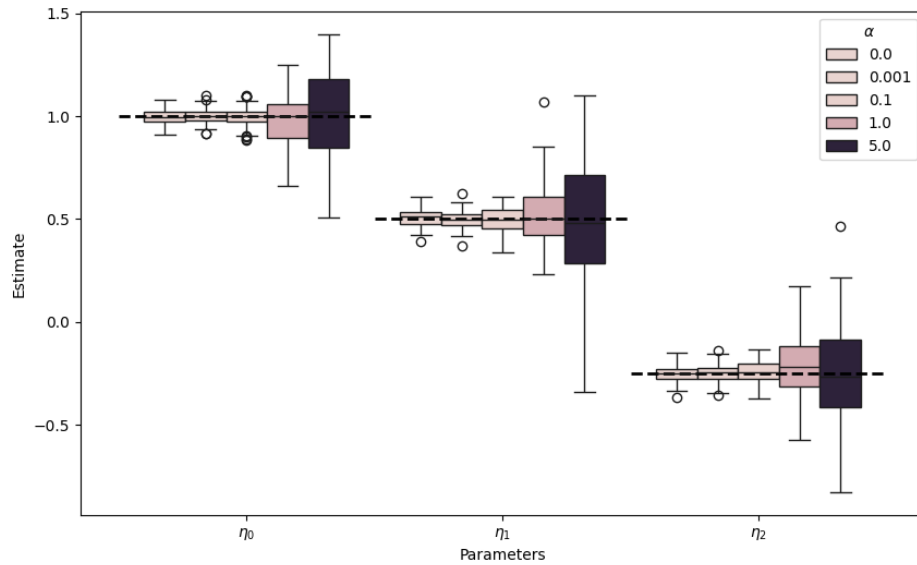


Figure 2.8: Boxplots for each specific value of $\hat{\eta}$. Each value of η has five different values that are based on the different values of α tested with $\alpha = 0$ on the left and increasing to $\alpha = 5$ on the right. The dashed black line shows the true value of each parameter, while the y-axis shows the values of the estimator.

Figure 2.8 shows a pattern we observed in Section 2.3.1 as we varied α : namely that the estimators' uncertainty increases as α increases. Regardless of the value of α , the parameter estimates remain unbiased. Figure 2.9 shows a similar plot for β . Change of α does not affect variability of the estimator of β because the estimator is based on the observed values of N only.

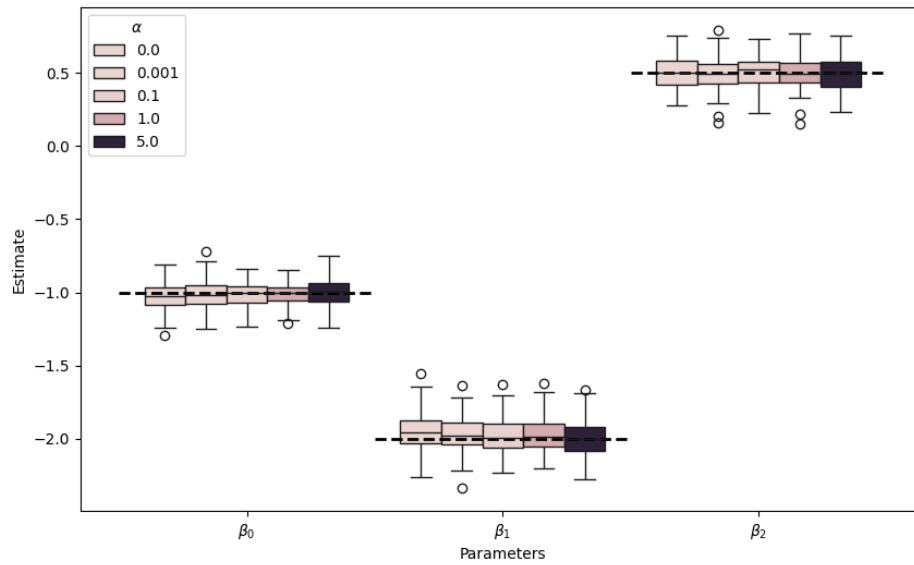


Figure 2.9: Boxplots for each specific value of $\hat{\beta}$. Each value of β has five different values that are based on the different values of α tested with $\alpha = 0$ on the left and increasing to $\alpha = 5$ on the right. The dashed black line shows the true value of each parameter, while the y-axis shows the values of the estimator.

These simulation results reveal several important patterns:

1. The MLEs generally converge to the true parameter values, with the variability decreasing as the sample size increases.
2. The accuracy of $\hat{\alpha}$ improves as the true value of α increases, reflecting the increasing distinctiveness of the GSMP model from the TETLG model.
3. The estimates of η show a systematic pattern where their magnitudes

tend to increase as the true value of α increases. This pattern reflects the complex interplay between α and $\boldsymbol{\eta}$ in determining the scale parameter λ of the GSMP model.

4. In contrast, the estimates of $\boldsymbol{\beta}$ remain stable and relatively unaffected by the value of α . This stability is consistent with the structure of the log-likelihood function, which allows $\boldsymbol{\beta}$ to be estimated independently of α and $\boldsymbol{\eta}$.

The observed pattern where the magnitude of $\hat{\boldsymbol{\eta}}$ increases with α while $\hat{\boldsymbol{\beta}}$ remains unaffected is particularly noteworthy. This behavior is a direct consequence of the decomposition of the log-likelihood function into separate components for $(\alpha, \boldsymbol{\eta})$ and $\boldsymbol{\beta}$. As α increases, the Pareto distribution becomes increasingly heavy-tailed, requiring larger values of $\boldsymbol{\eta}$ to maintain the same scale of observations. However, since the distribution of N (governed by $\boldsymbol{\beta}$) is independent of the distribution of the individual observations, the estimation of $\boldsymbol{\beta}$ remains stable across different values of α .

These additional comparisons confirm our theoretical understanding of the GSMP-VGLM model and provide valuable guidance for practical applications. When interpreting the results of a GSMP-VGLM analysis, it is important to consider the estimated parameters jointly rather than in isolation,

particularly for α and $\boldsymbol{\eta}$, which exhibit a complex interdependence.

2.4 Establishing Goodness-of-Fit Properties

As with any model, we would like to have means of checking if the model fits the data reasonably well. The difficulty in checking the goodness of fit for a VGLM is that each observation is (potentially) coming from a given distribution but with different parameters. That is we have observations coming from a given model which are independent but not identically distributed. Thus, the typical goodness of fit strategy of regression which is standardization of the residuals and check if they form IID observations from a standard normal distribution does not work in this case.

Our idea is to modify the "typical standardization" process so that the independent observations become also identically distributed. The idea is to transform the observations from VGSMF using estimated parameters, so they become identically distributed.

The mathematics of this method rests on the fact that for any continuous random variable X with CDF F , the random variable $Y = F(X)$ has a stan-

dard uniform distribution. Therefore, if we can find F we can transform an observation X from a given distribution to an observation from $U(0, 1)$. We use this idea on the marginal distributions of the VGLM-GSMP vector. We explain the steps on the marginal of magnitude X .

Since the magnitude has a Pareto II (Lomax) distribution with parameters α and λ , the CDF of X is

$$\mathbf{F}_X(x) = 1 - \Pr(X_i > x) = 1 - (1 + \alpha\lambda x)^{-\frac{1}{\alpha}}, \quad x \in \mathbb{R}^+. \quad (2.37)$$

Suppose X_1, X_2, \dots, X_k is a sample of independent observations from the Lomax distribution, each X_i having parameters α_i and β_i and CDF $F_i(\cdot)$. Then, the variables $Y_i = F_i(X_i)$ are IID Uniform on the interval $(0, 1)$.

In practice we do not know the true values of the parameters, but we do have their estimates: $(\hat{\alpha}_i, \hat{\beta}_i)$ for all the observations. We then estimate the true CDF F with \hat{F} where the true parameters are replaced by their estimates. If the estimates are close to the true parameters, then we have good fit of the model to the data and the $Y_{e,i} = \hat{F}_i(X_i)$ are close to an IID sample from $U(0, 1)$. We can check this fit using standard graphical methods and/or use

tests of goodness-of-fit such as the Kolmogorov-Smirnov test.

Although the idea of checking the fit of the maximum Y is the same, the difference is in the implementation. The CDF of magnitude has an explicit representation. The PDF and CDF of maximum do not have an explicit form. The PDF of maximum has an integral representation as follows:

$$f_Y(y) = \frac{\alpha \lambda p}{\Gamma\left(\frac{1}{\alpha}\right)} \int_0^\infty \frac{z^{\frac{1}{\alpha}} e^{-z(1+\alpha \lambda y)}}{\{p + (1-p)e^{-\alpha \lambda z y}\}^2} dz, \quad y \in \mathbb{R}^+.$$

Therefore we have to numerically approximate the CDF and the PDF in order to use this idea of checking goodness-of-fit.

2.4.1 Deriving Goodness-of-Fit for X

From Arendarczyk et al. (2018), we know that the marginal density of a single (X_i, Y_i, N_i) triple from a $\mathcal{GSM}\mathcal{P}(\alpha, \lambda_i, p_i)$, where we observed $X_i = x_i$, is:

$$f_X(x) = p_i \lambda_i (1 + \alpha p_i \lambda_i x_i)^{-\left(\frac{1+\alpha}{\alpha}\right)} \quad (2.38)$$

To find the CDF, we must integrate this over $[0, x]$:

$$F_X(x) = \int_0^x p_i \lambda_i (1 + \alpha p_i \lambda_i x)^{-\left(\frac{1+\alpha}{\alpha}\right)} dx$$

Setting $u = 1 + \alpha p_i \lambda_i x_i$ and replaced dx with $\frac{du}{\alpha p_i \lambda_i}$ and simplifying, we have:

$$\begin{aligned} F_X(x) &= F_X(u) \\ &= \frac{p_i \lambda_i}{\alpha p_i \lambda_i} \int_1^{1+\alpha p_i \lambda_i x_i} u^{-\left(\frac{1+\alpha}{\alpha}\right)} du \\ &= 1 - (1 + p_i \lambda_i x_i)^{-\frac{1}{\alpha}} \end{aligned}$$

Therefore, after fitting the $\mathcal{VGSMP}(\alpha, \boldsymbol{\eta}, \boldsymbol{\beta})$ model, we can construct $\boldsymbol{\lambda}$ and \mathbf{p} and use the resulting values of \mathbf{x} to derive the vector of, what should be, $\mathcal{U}(0, 1)$ random variables.

Simulation Study Example

To show how this works in practice, we drew 1000 triples from a $\mathcal{VGSMP}(\alpha, \boldsymbol{\eta}, \boldsymbol{\beta})$ where $\alpha = 0.1$, $\boldsymbol{\eta} = [1.0, 0.5, -0.25]^\top$, $\boldsymbol{\beta} = [2.0, -2.0, 1.0]^\top$, \mathbf{W} & \mathbf{Z} were drawn from a standard uniform distribution. We estimated the parameters $\boldsymbol{\eta}$ and $\boldsymbol{\beta}$ and then used these to construct $\boldsymbol{\lambda}$ and \mathbf{p} . Finally, we found $\hat{F}_{X_i}(x_i)$ for all 1000 observations and then plotted these in a QQ-Plot of the Standard Uniform distribution in Figure 2.10. The results were quite encouraging, as the fit proved to be very good.

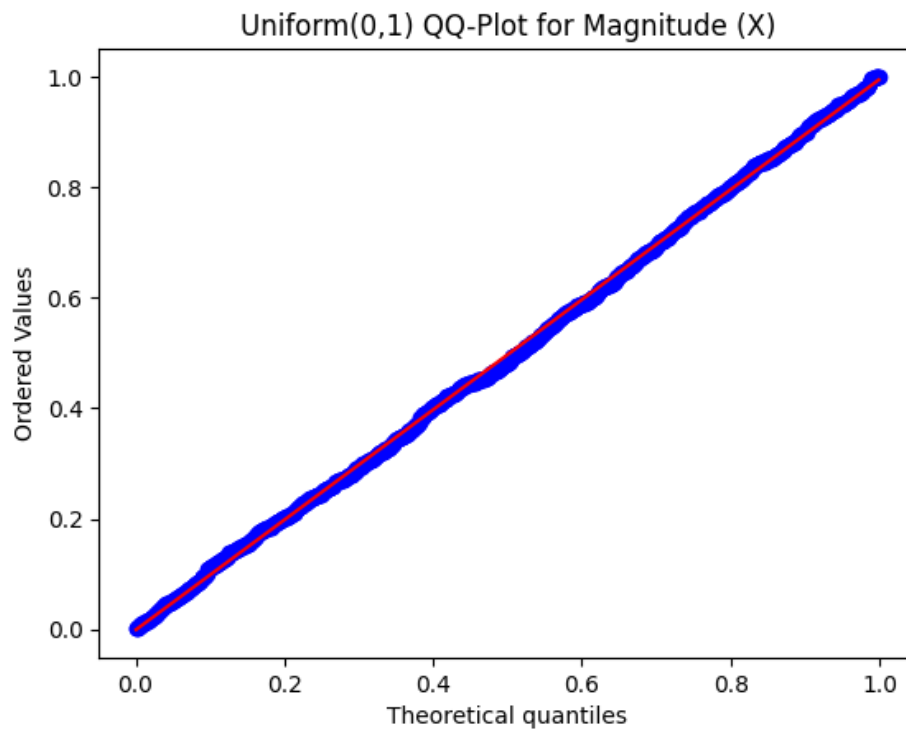


Figure 2.10: QQ-Plot of the observed data with fitted parameters against a theoretical $\mathcal{U}(0, 1)$ distribution. The red line is the theoretical perfect fit between observed and theoretical quantiles.

Additionally, we also ran the Kolmogorov-Smirnov Test comparing the observed values and the Standard Uniform CDF on the 5% significance level.

The tested hypotheses were:

$$H_o : \hat{F}_{X_i}(x_i) \text{ come from } U(0, 1) \text{ vs. } H_a : \hat{F}_{X_i}(x_i) \text{ do not come from } U(0, 1),$$

with $i = 1, 2, \dots, n$. The results were: the test-statistic of 0.01928 with a

resulting p-value of 0.8439. Given this, we do not reject the Null Hypothesis that the data we returned is consistent with a Standard Uniform distribution.

2.4.2 Deriving Goodness-of-Fit for Y

Consider an arbitrary triple, (X, Y, N) from a $\mathcal{GSM}\mathcal{P}(\alpha, \lambda, p)$ distribution. Our goal is to derive $F_Y(y) = \Pr(Y \leq y), y \in \mathbb{R}^+$. We begin by doing standard conditioning:

$$F_Y(y) = \sum_{n=1}^{\infty} \Pr(Y \leq y | N = n) \cdot p_n \quad (2.39)$$

where $p_n = \Pr(N = n)$ which is geometrically distributed with parameter p by assumption. Recall that since $Y = \bigvee_{i=1}^N L_i$ where, once we condition on $N = n$, (L_1, \dots, L_n) follows a multivariate Pareto II (Lomax) distribution, and that we can express this vector stochastically as:

$$(L_1, \dots, L_n) \stackrel{d}{=} \left(\frac{E_1}{Z}, \dots, \frac{E_n}{Z} \right)$$

where $E_i \stackrel{iid}{\sim} \text{Exp}(\lambda)$ and $Z \sim \Gamma(1/\alpha, 1/\alpha)$. Using this, we can express Equation (2.39) as

$$\begin{aligned} \Pr(Y \leq y | N = n) &= \Pr\left(\frac{E_1}{Z} \leq y, \dots, \frac{E_n}{Z} \leq y\right) \\ &= \int_0^{\infty} [F_E(yz)]^n f_Z(z) dz \end{aligned}$$

where $F_E(x) = 1 - e^{-\lambda x}$, $x \in \mathbb{R}^+$ is the CDF of an $Exp(\lambda)$ distribution and $f_Z(x)$ is the density of the $\Gamma(1/\alpha, 1/\alpha)$ distribution. This means that

$$\begin{aligned} F_Y(y) &= \int_0^\infty \left\{ \sum_{n=1}^\infty [F_E(yz)]^n \cdot f_Z(z) \cdot p_n \right\} dz \\ &= \int_0^\infty G(F_E(yz)) \cdot f_Z(z) dz \\ &= \mathbb{E}[G(F_E(yZ))] \end{aligned}$$

where $G(F_E(yz)) = \sum_{n=1}^\infty [F_E(yz)]^n \cdot p_n$ and $G(\cdot)$ is the probability generating function (PGF) of N :

$$\begin{aligned} G(s) &= \sum_{n=1}^\infty s^n p_n \\ &= \sum_{n=1}^\infty s^n p (-p)^{n-1} \\ &= \frac{ps}{1 - (1-p)s} \end{aligned}$$

where $s \in [0, 1]$. We may also define $Z = \alpha T$, where $T \sim \Gamma(1/\alpha, 1)$, which means we can express

$$F_E(yZ) = 1 - e^{-\alpha \lambda y T}$$

Therefore, we can express $F_Y(y)$ as the expected value of a function of the random variable $T \sim \Gamma(1/\alpha, 1)$ as follows:

$$F_Y(y) = \mathbb{E} \left[\frac{p \cdot (1 - e^{-\alpha \lambda y T})}{1 - (1-p)(1 - e^{-\alpha \lambda y T})} \right] \quad (2.40)$$

We estimate the expected value in equation (2.40) using Monte Carlo technique as the sample mean of k observations of the values of that function, see equation (2.41).

$$F_Y(y) \approx \frac{1}{k} \sum_{j=1}^k k \frac{p \cdot (1 - e^{-\alpha \lambda y T_j})}{1 - (1 - p)(1 - e^{-\alpha \lambda y T_j})} \quad (2.41)$$

where k is some large number of samples from a $\Gamma(1/\alpha, 1)$ distribution.

This derivation was for a single triple from a $\mathcal{GSM}\mathcal{P}$ distribution, but we can easily extend this to our $\mathcal{V}\mathcal{GSM}\mathcal{P}(\alpha, \boldsymbol{\eta}, \boldsymbol{\beta})$ distribution. In this case, we assume that each triple now has a unique λ_i and p_i , and that these parameters are linked to $\boldsymbol{\eta}$ and $\boldsymbol{\beta}$ using the associated link functions and covariates. Thus, unique λ_i and p_i can be found given \mathbf{Z} , \mathbf{W} and estimates of $\boldsymbol{\eta}$ & $\boldsymbol{\beta}$. Furthermore, Monte Carlo simulation can easily be done at scale by drawing (ℓ, k) matrices of $\Gamma(1/\alpha, 1)$ random variables and applying the function to those.

Simulation Study Example

To demonstrate this in practice, we kept the same set up as in Section 2.4.1, where we draw 1000 triples from a $\mathcal{V}\mathcal{GSM}\mathcal{P}(\alpha, \boldsymbol{\eta}, \boldsymbol{\beta})$ where $\alpha = 0.1$, $\boldsymbol{\eta} =$

$[1.0, 0.5, -0.25]^\top$, $\boldsymbol{\beta} = [2.0, -2.0, 1.0]^\top$, \mathbf{W} & \mathbf{Z} were drawn from a standard uniform distribution. Again, we estimated the parameters $\boldsymbol{\eta}$ and $\boldsymbol{\beta}$ and then used these to construct $\boldsymbol{\lambda}$ and \mathbf{p} . In this case, we performed a Monte Carlo approximation for each of our 1,000 values of Y . This means, for all 1,000 values of Y , we generated 10,000 $\Gamma(1/\alpha, 1)$ variables, put them into Equation (2.40) and returned the sample means. Lastly, we plotted these in a QQ-Plot of the Standard Uniform distribution, and once again, the results were very good, as seen in Figure 2.11.

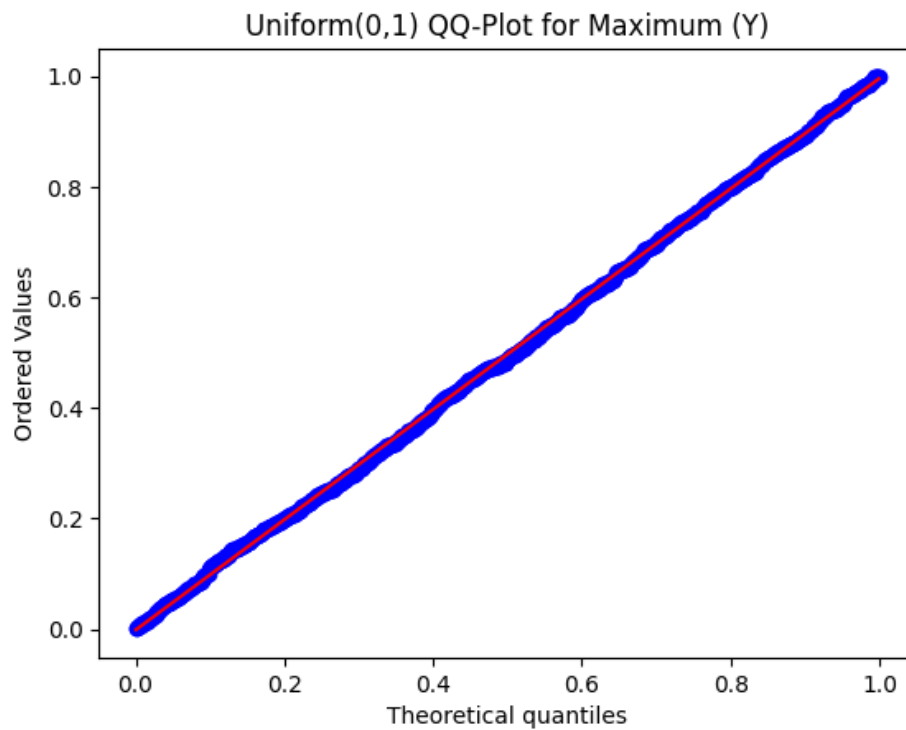


Figure 2.11: QQ-Plot of the observed data with fitted parameters against a theoretical $\mathcal{U}(0, 1)$ distribution. The red line is the theoretical perfect fit between observed and theoretical quantiles.

Additionally, we also ran the Kolmogorov-Smirnov Test comparing the observed values and the Standard Uniform CDF. The results were a test-statistic of 0.01547 with a resulting p-value of 0.9675. Given this, we do not reject the Null Hypothesis that the data we returned is consistent with a Standard Uniform distribution.

Chapter 3

Computational Estimation of GSMP-VGLM Model

Chapter 2 established the theoretical results we need to perform estimation and perform Goodness-of-fit checks in practice. However, there are several computational issues that need to be overcome before this method can be put into practice. In this chapter, we review various ways we can estimate η and β , including issues we uncovered and a preferred solution which was able to overcome this issue. We also review applications of Machine Learning techniques which we leveraged in order to find α in practice.

3.1 Overview of Estimation Approaches

Parameter estimation in the GSMP-VGLM framework presented significant computational challenges due to the complex interaction between model pa-

rameters and the structure of the likelihood function. This section explores three distinct computational approaches to estimating the parameter vectors $\boldsymbol{\eta}$ and $\boldsymbol{\beta}$:

1. Exact solutions via Normal Equations
2. Solving systems of nonlinear equations based on derivatives of the log-likelihood equation
3. Directly optimizing the likelihood function

Each approach offers unique advantages and limitations that affect the efficiency, accuracy, and reliability of the estimation process, and we present our recommendations at the end of the chapter.

3.2 Exact Solutions via Normal Equations

This section provides a quick derivation of a matrix solution that can be solved to provide a direct estimate of the parameters. We present two potential ways this solution can be applied in practice and challenges that impact both methods.

3.2.1 Theoretical Motivations & Derivation

Recall that the exact solution of $\boldsymbol{\eta}$ can be found by solving Equation 2.22:

$$\sum_{i=1}^n n_i \mathbf{w}_i = \sum_{i=1}^n (1 + \alpha n_i) \left[\frac{\mathbf{w}_i x_i e^{\boldsymbol{\eta}^\top \mathbf{w}_i}}{1 + \alpha x_i e^{\boldsymbol{\eta}^\top \mathbf{w}_i}} \right]$$

This equation can be reformulated as follows:

$$\begin{aligned} \sum_{i=1}^n \alpha n_i \mathbf{w}_i &= \sum_{i=1}^n (1 + \alpha n_i) \left[\frac{\mathbf{w}_i x_i e^{\boldsymbol{\eta}^\top \mathbf{w}_i}}{1 + \alpha x_i e^{\boldsymbol{\eta}^\top \mathbf{w}_i}} \right] \\ \sum_{i=1}^n \alpha n_i \mathbf{w}_i &= \sum_{i=1}^n (1 + \alpha n_i) \mathbf{w}_i \left[1 - \frac{1}{1 + \alpha x_i e^{\boldsymbol{\eta}^\top \mathbf{w}_i}} \right] \\ \sum_{i=1}^n \alpha n_i \mathbf{w}_i &= \sum_{i=1}^n \mathbf{w}_i + \sum_{i=1}^n \alpha n_i \mathbf{w}_i - \sum_{i=1}^n (1 - \alpha n_i) \left[\frac{1}{1 + \alpha x_i e^{\boldsymbol{\eta}^\top \mathbf{w}_i}} \right] \mathbf{w}_i \\ \sum_{i=1}^n \mathbf{w}_i &= \sum_{i=1}^n \left[\frac{1 - \alpha n_i}{1 + \alpha x_i e^{\boldsymbol{\eta}^\top \mathbf{w}_i}} \right] \mathbf{w}_i \end{aligned}$$

The equation is always true if $\left[\frac{1 - \alpha n_i}{1 + \alpha x_i e^{\boldsymbol{\eta}^\top \mathbf{w}_i}} \right] = 1$. This condition is equivalent to the following statement:

$$\mathbf{W}\boldsymbol{\eta} = \mathbf{r} \tag{3.1}$$

where $\mathbf{r} = \ln(\mathbf{n}/\mathbf{x})$ and \mathbf{W} is the covariate matrix. From the earlier requirements on this equation outlined in Theorem 2.3.2, \mathbf{W} would be invertible as long as it is square, i.e. $\ell = k + 1$. This means that the solution $\hat{\boldsymbol{\eta}} = \mathbf{W}^{-1}\mathbf{r}$ would exist and provide a direction solution.

In a specific case, if $2 = \ell = k + 1$, we can further show that the specific

values of $\hat{\boldsymbol{\eta}}$ would be:

$$\hat{\eta}_0 = \frac{w_2 \ln(n_1/x_1) - w_1 \ln(n_2/x_2)}{w_2 - w_1}$$

$$\hat{\eta}_1 = \frac{\ln(n_2/x_2) - \ln(n_1/x_1)}{w_2 - w_1}$$

where $\mathbf{W} = \begin{bmatrix} 1 & w_1 \\ 1 & w_2 \end{bmatrix}$, $\mathbf{n} = [n_1, n_2]^\top$ and $\mathbf{x} = [x_1, x_2]^\top$

3.2.2 Normal Equation Solution

In practice it is highly unlikely that $\ell = k + 1$, and this condition puts an unnaturally tight constraint on the problem. However, it is possible to extend this solution to the general case where $\ell \geq k + 1$ by using a Normal Equation approach as done in OLS regression. In other words, we define

$$\hat{\boldsymbol{\eta}} = (\mathbf{W}^\top \mathbf{W})^{-1} \mathbf{W}^\top \mathbf{r} \quad (3.2)$$

where we use the Moore-Penrose pseudoinverse. It can be shown that in the special case of $2 = \ell = k + 1$, these two solutions are identical. Further we can derive the same estimates for $\boldsymbol{\beta}$. Given

$$h(\boldsymbol{\beta}) = \boldsymbol{\beta} \sum_{i=1}^{\ell} \mathbf{z}_i - \sum_{i=1}^{\ell} n_i \ln \left(1 + e^{\boldsymbol{\beta}^\top \mathbf{z}_i} \right),$$

the derivative of $h(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$ is

$$\frac{\partial h(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{\ell=1}^{\ell} \mathbf{z}_i - \sum_{i=1}^{\ell} n_i \left[\frac{\mathbf{z}_i e^{\boldsymbol{\beta}^\top \mathbf{z}_i}}{1 + e^{\boldsymbol{\beta}^\top \mathbf{z}_i}} \right].$$

This value is zero if $n_i \left[\frac{e^{\boldsymbol{\beta}^\top \mathbf{z}_i}}{1 + e^{\boldsymbol{\beta}^\top \mathbf{z}_i}} \right] = 1 \forall i$.

$$\boldsymbol{\beta}^\top \mathbf{z} = -\ln(n_i - 1) \quad \text{for all } i = 1, \dots, n.$$

This can only be satisfied when $n_i > 1$. Let \mathbf{n}_\star be the values of \mathbf{n} where $n_i > 1$, and let \mathbf{Z}_\star be the corresponding rows of \mathbf{Z} . Then

$$\mathbf{Z}_\star \boldsymbol{\beta} = -\ln(\mathbf{n}_\star - 1), \text{ thus } \mathbf{W}_\star \boldsymbol{\beta} = \mathbf{W}_\star^\top \ln(\mathbf{n}_\star - 1),$$

$$\text{and so } \hat{\boldsymbol{\beta}} = -(\mathbf{W}_\star^\top \mathbf{W}_\star)^{-1} \mathbf{W}_\star^\top \ln(\mathbf{n}_\star - 1).$$

3.2.3 Applying in Practice

Though we have potential exact solutions, how we can apply this in practice could be done in two possible ways in the case of $\ell \geq k + 1$. For conciseness, we focus on the case where we are estimating $\boldsymbol{\eta}$ but this reasoning could easily be applied to $\boldsymbol{\beta}$. The two solutions are:

1. **Average estimates $\mathbf{W}^{-1}\mathbf{r}$ applied to $\ell \times \ell$ subsets of the data.:**

In this approach, we can either bootstrap ρ samples of $\ell \times \ell$ blocks from the full data. The estimates $\hat{\boldsymbol{\eta}}_1, \dots, \hat{\boldsymbol{\eta}}_\rho$ are stored and the final estimate is $\hat{\boldsymbol{\eta}}_{avg} = \frac{1}{\rho} \sum_{i=1}^{\rho} \hat{\boldsymbol{\eta}}_i$.

2. **Solve using the Normal Equations.:** Apply equation 3.2 directly, which we denote $\hat{\boldsymbol{\eta}}_{NE}$.

3.2.4 Analysis of Avg. Exact Solutions vs. Normal Equation

To study the performance of these approaches, we simulated data from a GSMP-VGLM distribution with true parameter vectors $\boldsymbol{\eta} = [1, 0.5]^\top$ and $\boldsymbol{\beta} = [-1, -2]$. 1000 rows of data were drawn, i.e. $\ell = 1000$ and we utilized one covariate and intercept term, i.e. $k = 1$. We also selected different α values in $\{1e^{-17}, 0.0001, 0.1, 1, 5\}$. We then estimated $\hat{\boldsymbol{\eta}}_{NE}$ and saved the results, as well as constructing $\hat{\boldsymbol{\eta}}_{avg}$ from averaging the 500 2×2 submatrices. The time needed to perform each operation was also logged. This entire process was repeated 300 times. Figures 3.1 & 3.2 show the results.

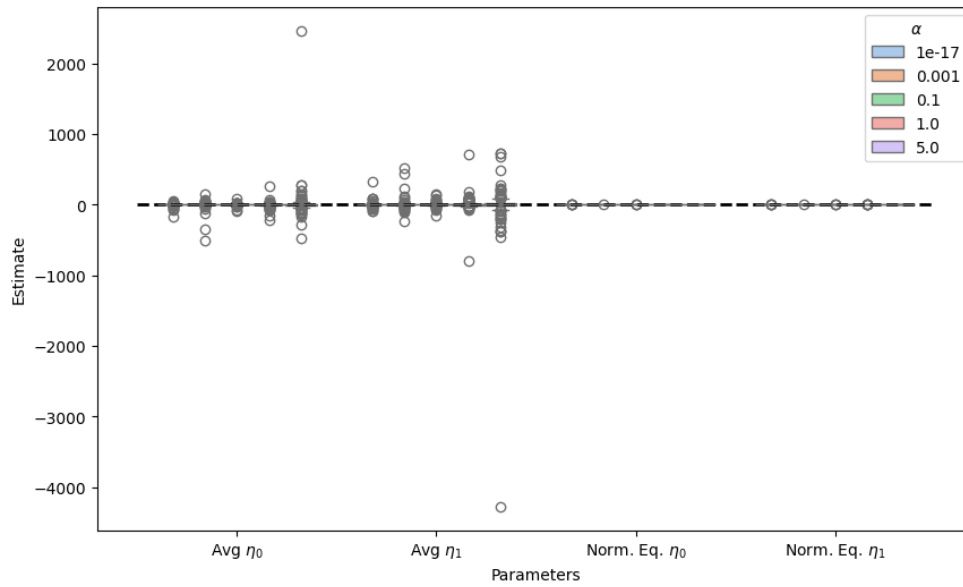


Figure 3.1: Boxplots of the estimated values of each component of $\boldsymbol{\eta}$ using averages of 2x2 matrices (left two boxplots) and the normal equations (right two boxplots). For each method and value of $\boldsymbol{\eta}$, there are five plots for different values of α ranging from 1×10^{17} (leftmost) to 5 (rightmost). The y-axis shows the full-range of parameter estimates found scaled to include the largest value across all parameters and methods.

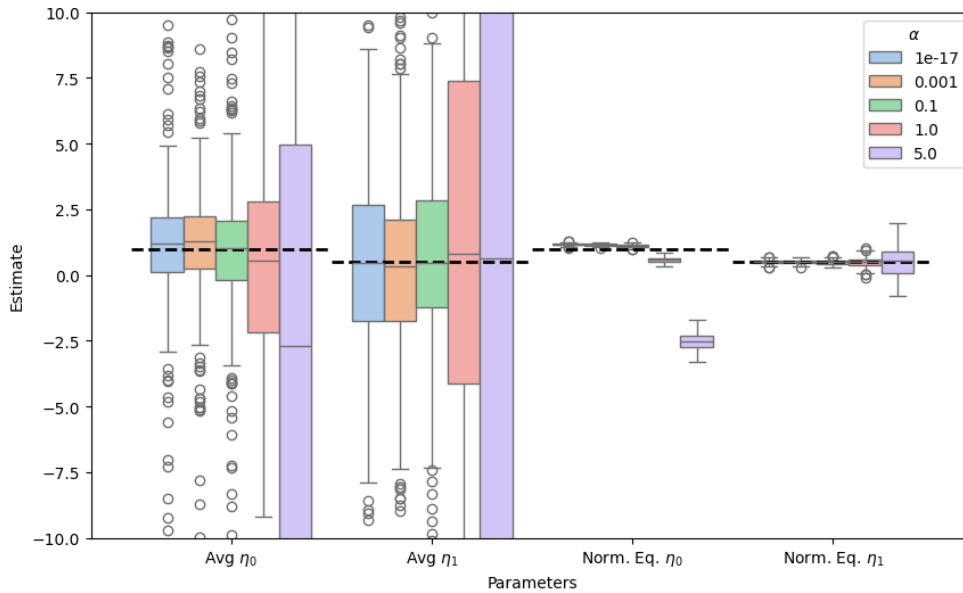


Figure 3.2: The same plot as in Figure 3.1, but "zoomed" to y-values between ± 10 .

As can be clearly seen in the figures, the boxplots of data are impossible to view when plotting all results due to large volatility in the $\hat{\eta}_{avg}$ estimates. When zoomed in however, we can observe that the average estimates are far more volatile than those obtained using normal equations, but that the median value (horizontal black dash in boxplots) align across $\hat{\eta}_{avg}$ and $\hat{\eta}_{NE}$. This is most clear in the case when $\alpha = 5$ and there is large bias in the $\hat{\eta}_0$ term.

We also present KDE plots of the runtime in the following figure:

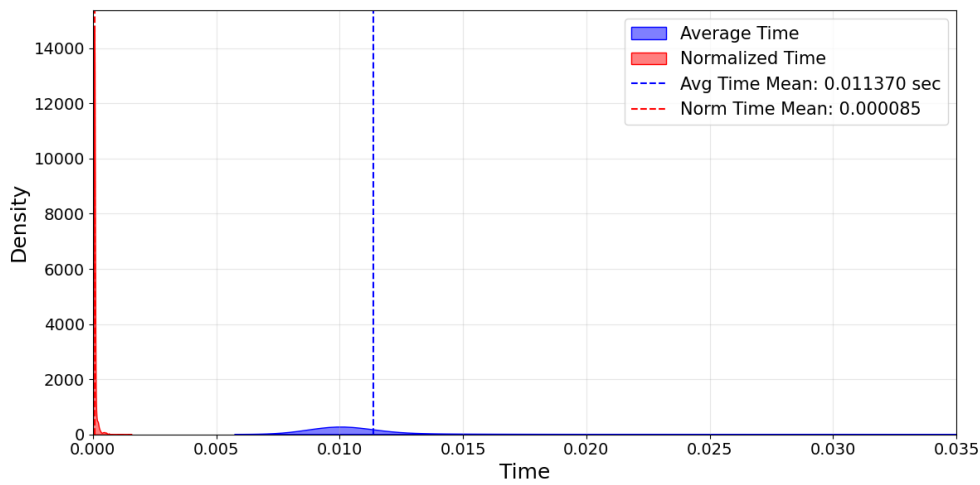


Figure 3.3: KDE plots of the observed computation time for each method where the “average” method is in blue and the “Normal Equation” method in red. The dashed vertical lines show the mean time across all simulations. The x-axis is the time taken and the y-axis is the density value of the KDE estimator.

Here we observe that the average computation time of $\hat{\eta}_{avg}$ is several orders larger than $\hat{\eta}_{NE}$. Increasing the size of ℓ and k would cause these estimates to vary, but it should be clear that $\hat{\eta}_{avg}$ would always take more time to compute due to the nature of the estimation process.

Based on these two findings, we prefer $\hat{\eta}_{NE}$ as the estimation function, but note that this estimation becomes increasingly unstable as α increases, and that the intercept term becomes increasingly biased. These findings highlight the major pro and con of the Exact solution: Extremely fast and direct

computation without the need for iterative solutions contrast with unstable estimation as α increases. Since, in practice, we cannot generally state what α is we cannot recommend using this as the sole estimation approach for practitioners. However, *we do* believe this approach can add a great deal of value as providing an informed initialization to other methods which we'll discuss shortly.

3.3 Numerical Solvers for Systems of Nonlinear Equations

The next potential solution to estimating parameters in the GSMP-VGLM framework involves solving the systems of nonlinear equations derived from setting the derivatives of the log-likelihood function to zero. This section examines the application of numerical solvers to this problem, with particular focus on the challenges and limitations encountered when using these methods.

3.3.1 Solving the Maximum Likelihood Equations

The maximum likelihood estimation of $\boldsymbol{\eta}$ in our GSMP-VGLM model requires solving the following system of equations:

$$\sum_{i=1}^n n_i w_{i,l} = \sum_{i=1}^n (1 + \alpha n_i) \left[\frac{w_{i,l} x_i e^{\boldsymbol{\eta}^\top \mathbf{w}_i}}{1 + \alpha x_i e^{\boldsymbol{\eta}^\top \mathbf{w}_i}} \right], \quad l = 0, 1, \dots, k. \quad (3.3)$$

Similarly, for $\boldsymbol{\beta}$, we need to solve:

$$\sum_{i=1}^n z_{i,j} = \sum_{i=1}^n n_i z_{i,j} \frac{e^{\boldsymbol{\beta}^\top \mathbf{z}_i}}{1 + e^{\boldsymbol{\beta}^\top \mathbf{z}_i}}, \quad j = 0, \dots, m \quad (3.4)$$

In principle, solving these equations should provide the maximum likelihood estimates directly, though there is no closed-form solution. Instead, computational methods must be used to provide the solution.

3.3.2 The `fsolve` Algorithm and Implementation

The most direct solution is to solve the system of equations shown earlier directly. In `Python`, this can be done using several packages, but we used the scientific computing package `scipy` and its `fsolve` function to accomplish this. `fsolve` is a wrapper around a library called `MINPACK` which typically uses two variations of Powell's Hybrid method for root-finding in non-linear systems of equations.

Powell's hybrid method combines quasi-Newtonian solutions with gradient

based methods using a trust-region approach. Given a system of equations

$$\mathbf{F}(\boldsymbol{\theta}) = \begin{bmatrix} f_1(\mathbf{x}; \boldsymbol{\theta}) \\ \vdots \\ f_n(\mathbf{x}; \boldsymbol{\theta}) \end{bmatrix} = \mathbf{0},$$

the objective is to find a root $\boldsymbol{\theta}^*$ such that $\mathbf{F}(\boldsymbol{\theta}^*) = \mathbf{0}$. The component functions, $f_i(\mathbf{x}; \boldsymbol{\theta})$ are parameterized by $\boldsymbol{\theta}$ and have observations \mathbf{x} . At each iteration, Powell's method performs the following high-level procedure:

1. Compute or approximate the Jacobian, \mathbf{J}_F
2. Determine the Gauss-Newton step using the Jacobian. If this step lands inside of a “trust region”, Δ , then the parameters update based on the Gauss-Newton step.
3. If the Gauss-Newton step is outside of the trust region, the function calculates the gradient in the direction of steepest descent with a dynamically calculated step-size. If the dynamic step size is also out of the trust region, then the step-size is truncated to land exactly on the boundary of the trust region.
4. If the Gradient step is inside the trust region and the Gauss-Newton step is outside the trust region, then the line between the two steps is

found and the step is taken where this line intersects the trust region.

`fsolve` employs two variations of this method; one which calculates the Jacobian directly and the other which approximates the Jacobian using Broyden's method, though the fundamental mechanics remain the same. By blending both Newtonian methods and gradient-based methods, this system can be highly effective and rapidly converge. However, there is no guarantee that this approach will work and there are known situations where this method can fail.

3.3.3 Issues with Implementation

Though this method is quite powerful and yielded good performance, we were able to uncover cases during simulation studies where the estimation became suddenly quite poor, with estimates varying wildly (e.g. ± 100 where other studies had intervals less than 0.1 for the same ℓ and α). The ultimate issue will be detailed in Section 3.4 momentarily, but of critical importance was that this system of linear equations was based on the derivative of the log-likelihood function, meaning that the Jacobian of the system of equations in this case is equivalent to the Hessian of the log-likelihood function. Issues which we can show impact the Hessian led to the instability of this method.

Due to the observed instability that could be induced, this method cannot be universally trusted and we continued our search to find a more robust method.

3.4 Direct Maximization of Likelihood Functions

An alternative approach to parameter estimation in the GSMP-VGLM framework involves directly maximizing the log-likelihood function rather than solving the system of equations that results from its derivative set to zero. This section discusses the objective function, standard optimizing procedures used in practice, reviewing why these methods can fail and then showing how a simpler gradient-based solver can overcome these limitations. The remainder of the section reviews additional implementation details and possible extensions.

3.4.1 The Maximum Likelihood Objective

The log-likelihood function for our GSMP-VGLM model, as derived in Chapter 2, takes the form:

$$\mathcal{L}(\alpha, \boldsymbol{\eta}, \boldsymbol{\beta}) = g(\alpha, \boldsymbol{\eta}) + h(\boldsymbol{\beta}) + C \quad (3.5)$$

where:

$$g(\alpha, \boldsymbol{\eta}) = \log \left\{ \prod_{i=1}^n \frac{\Gamma(n_i + \frac{1}{\alpha})}{\Gamma(\frac{1}{\alpha})} \alpha^{n_i} e^{(\boldsymbol{\eta}^\top \mathbf{w}_i) n_i} \left(1 + \alpha e^{\boldsymbol{\eta}^\top \mathbf{w}_i} x_i\right)^{-\frac{1}{\alpha} - n_i} \right\}, \quad (3.6)$$

$$h(\boldsymbol{\beta}) = \boldsymbol{\beta}^\top \sum_{i=1}^n \mathbf{z}_i - \sum_{i=1}^n n_i \log \left(1 + e^{\boldsymbol{\beta}^\top \mathbf{z}_i}\right), \quad (3.7)$$

and C is a constant independent of the parameters. Direct maximization seeks to find:

$$(\hat{\boldsymbol{\eta}}, \hat{\boldsymbol{\beta}}) = \arg \max_{\substack{\boldsymbol{\eta}, \boldsymbol{\beta} \\ \alpha \text{ fixed}}} \mathcal{L}(\alpha, \boldsymbol{\eta}, \boldsymbol{\beta}). \quad (3.8)$$

It is clear that $h(\boldsymbol{\beta})$ is independent of $g(\alpha, \boldsymbol{\eta})$, and we also showed that, when α was fixed, we could determine $g(\alpha, \boldsymbol{\eta}(\alpha))$. Then in practice, what we attempt to achieve in this approach is to directly search the parameter space for values that maximize the log-likelihood function rather than trying to solve a system of nonlinear equations.

Of course both approaches will rely on (at least) the first derivative and,

depending on the likelihood maximization function, the second derivative. Thus, the issues that were shown in Section 3.3 can and do impact the estimation process, but in a much more direct way.

3.4.2 Taylor Series Approximation Background

When dealing with concave/convex function optimization, which we established is the case for $h(\boldsymbol{\beta})$ and $g(\alpha, \boldsymbol{\eta})$, a common approach is to utilize Taylor Series approximation of the function. This is a method where a continuous and differentiable function, f , can be well-approximated locally using polynomials centered at some fixed point, \mathbf{x} .

Though this approximation could be of any order, in practice f is most commonly approximated using a first-order approximation (i.e. using lines/hyperplanes to approximate f) or a second-order approximation (i.e. using parabolas/hyperparabolas to approximate f). We can express the first-order approximation mathematically as:

$$f(\mathbf{x} + \boldsymbol{\Delta}) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^\top \boldsymbol{\Delta}, \quad (3.9)$$

where $\boldsymbol{\Delta}$ is some vector distance, typically assumed to have small magnitude

(e.g. $\|\Delta\|_2 < \epsilon$. Similarly, the second-order approximation is:

$$f(\mathbf{x} + \Delta) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^\top \Delta + \frac{1}{2} \Delta^\top \mathbb{H}_f(\mathbf{x}) \Delta \quad (3.10)$$

where \mathbb{H}_f is the Hessian of f .

Before discussing how to use this in likelihood optimization, we need to make an explicit statement that most optimization methods are build around minimizing convex functions, so rather than maximizing the log-likelihood, $\mathcal{L}(\cdot)$, we seek to minimize the negative log-likelihood function, $\mathcal{N}\mathcal{L}(\cdot) := -\mathcal{L}(\cdot)$.

Gradient Descent

Given a convex function to minimize, $\mathcal{N}\mathcal{L}(\theta)$, we assume that the function behaves like the first order approximation:

$$\mathcal{N}\mathcal{L}(\theta) + \nabla_{\theta} \mathcal{N}\mathcal{L}(\theta)^\top \Delta, \quad (3.11)$$

and the goal is to find Δ that minimizes the function. It can be shown that the optimal value is

$$\Delta := -\nabla_{\theta} \mathcal{N}\mathcal{L}(\theta)$$

and that the value of $\mathcal{N}\mathcal{L}(\theta) \geq \mathcal{N}\mathcal{L}([\theta - \nabla_{\theta}\mathcal{N}\mathcal{L}(\theta)])$.

Of course we don't assume that the first-order approximation is perfect and adjust the parameter θ exactly by the value of the gradient, $-\nabla_{\theta}\mathcal{N}\mathcal{L}(\theta)$. Instead, we only adjust the parameter vector by some small fraction of the gradient: $-\gamma \cdot \nabla_{\theta}\mathcal{N}\mathcal{L}(\theta)$, where $\gamma \in \mathbb{R}^+$ is some small value known as the “step size” or “learning rate”. The larger the value of γ , the larger the “step” taken to adjust θ . Thus, over some number of iterations, we define the updated parameter vector as

$$\theta^{(t+1)} := \theta^{(t)} - \gamma \cdot \nabla_{\theta^{(t)}}\mathcal{N}\mathcal{L}(\theta^{(t)})$$

If the step size is appropriate, given enough steps, the gradient descent algorithm will reach the global minimum. Of course unless it lands exactly at the optimal value, the gradient will be nonzero and could oscillate in the neighborhood of the minimum. To force convergence, there are several practices, but a schedule to reduce the step size is often introduced to make the value of γ converge to 0. Other variants will be discussed later, but a common approach is setting $\gamma_{(t+1)} = c \cdot \gamma_{(t)}$ where $c \in (0, 1)$.

We'll discuss this in subsequent sections, but though Gradient Descent *does* converge, the major weakness is the dependence on $\gamma_{(t)}$ for the function to converge. It can be very slow and inefficient compared to other methods and for this reason, Gradient Descent is not typically preferred in solving convex optimization problems.

Newtonian Methods

Given the objective defined in the section above, rather than assuming that $\mathcal{N}\mathcal{L}(\theta)$ behaves like a line/hyperplane, we can instead assume that the function behaves like a parabola/hyperparabola:

$$\mathcal{N}\mathcal{L}(\theta) + \nabla_{\theta}\mathcal{N}\mathcal{L}(\theta)^{\top}\Delta + \frac{1}{2}\Delta^{\top}\mathbb{H}_{\mathcal{N}\mathcal{L}}(\theta)\Delta.$$

Though calculating a full Hessian matrix at each step is slower and much more computationally intensive, there is a significant advantage to this. Because it is a parabola/hyperparabola, *the function achieves a unique global minimum*. So rather than defining a “step size”, Newtonian methods determine the specific step location by minimizing the parabola. Put mathematically, we set Δ as

$$\Delta := \arg \min_{\Delta} \mathcal{N}\mathcal{L}(\theta) + \nabla_{\theta} \mathcal{N}\mathcal{L}(\theta)^{\top} \Delta + \frac{1}{2} \Delta^{\top} \mathbb{H}_{\mathcal{N}\mathcal{L}}(\theta) \Delta,$$

which can easily be found by setting the derivative of the hyperparabola with respect to Δ equal to $\mathbf{0}$ and solving for Δ . The result is

$$\Delta := -[\mathbb{H}_{\mathcal{N}\mathcal{L}}(\theta)]^{-1} \nabla_{\theta} \mathcal{N}\mathcal{L}(\theta). \quad (3.12)$$

Because $\mathcal{N}\mathcal{L}(\theta)$ is convex, the Hessian is an invertible square matrix, so this can always be calculated.

Summarizing, even though the Newtonian methods are more computationally intensive, they don't require a parameter that needs tuning (though this can be added). If the approximation of $\mathcal{N}\mathcal{L}(\cdot)$ with a hyperparabola is good, then convergence using these approaches can be extremely fast and requires only a few iterations. Furthermore, there are numerous solvers that do not build the entire Hessian, but rather approximate it with a quasi-Newtonian method, such as BFGS (Broyden-Fletcher-Goldfarb-Shanno) or L-BFGS (Limited-memory BFGS).

Newtonian Issues and the Fundamental Advantage of Gradient Descent

In our implementation, we initially explored these Newtonian approaches to minimize the negative log-likelihood. Though this could be a successful solver, we found situations in simulation studies where the solver exhibited highly divergent behavior, particularly when we used skewed distributions for our covariates that were on different scales.

As a last resort, we implemented a very basic Gradient Descent implementation with lots of iterations and extremely small step sizes. The results were dramatically improved approximations of the function of interest. This raised the question: why should a generally poor solver best the Newtonian solvers? After more analysis, we determined that the principal cause was a known issue with Hessian-based solvers: nearly flat dimensions of the function of interest.

Newtonian and Quasi-Newtonian solvers rely on the Hessian to set the step size, which can be highly useful when the shape of the function is well approximated by the Hessian. However, when $\mathcal{N}\mathcal{L}(\cdot)$ has at least one nearly flat

dimension, the Hessian provides a poor approximation and results in wildly large steps in the nearly flat dimension(s).

Gradient Descent solvers, by contrast, do not have to suffer in these cases and can still make progress in cases where the Hessian is nearly singular. It may converge slowly in flat dimensions, but there are several best practices and more advanced variants that can be implemented to ensure progress is achieved.

Before we discuss implementation details, we do acknowledge that a step size limiter can be implemented within Newtonian methods, i.e.

$$\Delta := -\gamma \cdot [\mathbb{H}_{\mathcal{N}\mathcal{L}}(\theta)]^{-1} \nabla_{\theta} \mathcal{N}\mathcal{L}(\theta), \quad \gamma \in (0, 1).$$

This can help control the cases of extremely large step sizes, but we did not find this to be a practical solution for several reasons:

1. **The Step Size is Variable:** The step size in Newtonian Methods is adaptive, and can vary greatly from one step to the next. Decay schedules tend to be monotonic and cannot adapt. Even if it were possible to vary the step size limiter adaptively, it cannot distinguish

when the large step is “good” versus “bad”.

2. **No Guaranteed Convergence with Decay Schedules:** Unlike Gradient Descent with Decay Schedules which will converge to the minimum given enough iterations, we cannot impose this condition with Newtonian solvers, as the step sizes can be extremely large and push the solver into a bad zone even as the step-size goes to zero.
3. **Extremely Small Steps Defeats the Purpose:** *Even if* it were possible to restrict the step size of the Newtonian solver to be very cautious and decay uniformly, the ultimate output is to have implemented a Gradient Descent style solver (many iterations, small, controlled step sizes) but in a highly computationally inefficient manner as you would be computing or at least approximating the Hessian with each step.

3.4.3 Implementation Details and Practical Considerations

Our implementation of gradient descent for maximizing the GSMP-VGLM log-likelihood incorporates several practical refinements beyond the basic algorithm outlined above. These refinements address specific challenges encountered in our numerical experiments and significantly improve the relia-

bility and efficiency of the estimation procedure.

Coordinate-wise Optimization

Rather than optimizing all parameters simultaneously, we found that a coordinate-wise approach often yielded better results. Specifically, we alternated between:

1. Optimizing β with fixed α and η .
2. Optimizing η with fixed α and β .
3. For the full model, optimizing α with fixed η and β .

This approach leverages the partial separability of the log-likelihood function and allows for different step size schedules tailored to the specific properties of each parameter group.

Initialization Strategies

The success of gradient descent can depend significantly on the choice of initial parameter values. We explored several initialization strategies:

1. **Zero Initialization:** Setting all parameters to zero (or small random values) is a simple baseline approach but can lead to slow convergence if the true parameters are far from zero.

2. **Normal Equations Approximation:** As discussed in Section 3.2, the normal equations can provide a closed-form approximation that, while not exactly satisfying the maximum likelihood equations, often serves as a reasonable starting point.
3. **Multiple Random Starts:** In some cases, we employed multiple random initializations and selected the solution with the highest final likelihood value, helping to address potential concerns about local maxima.

Our empirical investigations suggested that the normal equations approximation generally provided the most reliable initialization, significantly reducing the number of iterations required for convergence compared to simpler initialization schemes.

Gradient Computation and Numerical Stability

Accurate and numerically stable computation of the gradient is essential for successful optimization. In our implementation, all parameters were cast as `float64` or `int64` so that we could utilize double-precision floating-point operations to help prevent ‘underflow’ and ‘overflow’. Though these precautions increase the memory consumed by the program, this turned out to be particularly useful when trying to fit the model in situations where the mag-

nitude of the durations of excess were extremely large or when the likelihood function has near-zero values in the Hessian along the main diagonal.

Convergence Criteria

Determining when and how to terminate the gradient descent algorithm requires careful consideration. In our current implementation, we stopped the optimization only after running a pre-specified number of iteration steps. Since we utilize step size decay, in theory, enough steps in conjunction with a good decay schedule should ensure that the optimization stops on or near the optimal value.

Going forward, there are additional methods that we could instantiate that would allow for early termination. These include:

1. **Parameter Precision Thresholds:** The optimization will terminate before the maximum number of iterations if the change in all parameters is less than some pre-defined threshold over successive iterations, e.g. if all parameters do not change above the third decimal for 10 iterations.
2. **Likelihood Precision Thresholds:** The optimization will terminate before the maximum number of iterations if the change in the log-

likelihood function is less than some pre-defined threshold over successive iterations, e.g. if the log-likelihood function doesn't change above the third decimal for 10 iterations.

In practice, we found that utilizing the solution from the Normal Equations (Equations 3.2 & 3.2.2) as an initialization point, coupled with step decay will ensure consistent convergence in most cases with about 100-2000 steps, though this does depend on the number of observations, the number of total covariates used in the estimation process, and the curvature of $\mathcal{N}\mathcal{L}(\cdot)$.

3.4.4 Advanced Gradient-Based Methods

Though our current Gradient-based solution has provided a consistent solver across all situations we've studied, it is not the most efficient form of gradient descent. Thanks to their application in tuning Deep Learning models, there have been several advanced forms of gradient-based optimization techniques that could potentially offer improvements in speed and robustness. We'll discuss two major forms, Momentum and Adam.

Momentum Methods

In the standard "vanilla" Gradient Descent, the gradient is calculated at each iteration. The information is used to adjust the parameter vector and

then that information is completely discarded. However, if the gradient in successive steps had been pointing in generally the same direction (at least in some dimensions), then this information could be used to help increase the convergence speed. Momentum methods augment standard gradient descent by incorporating the prior gradient information from previous updates to help speed up convergence. The idea is to calculate

$$\mathbf{m}^{(t+1)} = \mu \mathbf{m}^{(t)} + \gamma_t \nabla f(\boldsymbol{\theta}^{(t)}) \quad (3.13)$$

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \mathbf{m}^{(t+1)} \quad (3.14)$$

where

- $\mu \in [0, 1)$ is the momentum coefficient and dictates how much influence prior steps get.
- $\mathbf{m}^{(t)}$ is the momentum vector at iteration t , and
- γ_t is the step size at time t .

Momentum methods can accelerate convergence, particularly in regions where the gradient consistently points in similar directions across iterations, but can

be problematic if the momentum weight is too large and the step sizes are too large.

Adam Optimizer

We also explored implementing the Adam (Adaptive Moment Estimation) optimizer, which extends the notion of momentum discussed earlier while also providing adaptive step sizes. It computes adaptive step for each parameter based on the first and second moments of the gradients. It first calculates a term very similar to the moment value we saw before:

$$\mathbf{m}^{(t)} = \beta_1 \mathbf{m}^{(t-1)} + (1 - \beta_1) \nabla f(\boldsymbol{\theta}^{(t)}) \quad (3.15)$$

However, the primary difference is that this is now a convex combination of prior gradients (i.e. ‘momentum’) and the current gradient, which increases stability. It then calculates a new term $\mathbf{v}^{(t)}$

$$\mathbf{v}^{(t)} = \beta_2 \mathbf{v}^{(t-1)} + (1 - \beta_2) (\nabla f(\boldsymbol{\theta}^{(t)}))^2, \quad (3.16)$$

which is a convex combination of squares of prior gradients and the current gradient. This is done to provide a notion of variance around each of the gradient dimensions which will be used to adjust the step in the next iteration. As currently written, both terms are biased, so the following correction

is applied:

$$\hat{\mathbf{m}}^{(t)} = \frac{\mathbf{m}^{(t)}}{1 - \beta_1^t} \quad (3.17)$$

$$\hat{\mathbf{v}}^{(t)} = \frac{\mathbf{v}^{(t)}}{1 - \beta_2^t} \quad (3.18)$$

Finally, the gradient step is now taken using the following equation:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \gamma \frac{\hat{\mathbf{m}}^{(t)}}{\sqrt{\hat{\mathbf{v}}^{(t)} + \epsilon}} \quad (3.19)$$

where:

- β_1 and β_2 weight how important prior gradients and their variance are (respectively),
- γ is the step size, and
- ϵ is a small constant for numeric stability when the denominator is small.

The primary addition is the $\frac{\hat{\mathbf{m}}^{(t)}}{\sqrt{\hat{\mathbf{v}}^{(t)} + \epsilon}}$ term. If the momentum term in a particular parameter is highly variable, then $\hat{\mathbf{v}}^{(t)}$ will be large and will result in a smaller contribution in that dimension relative to another dimension

which has been very stable. This helps the solver converge quickly like in Momentum solvers, but better handles noisy or sparse gradients. This also helps it overcome some of the issues that momentum suffers in flat portions of the likelihood, though it is by no means immune.

We had attempted to implement Adam as part of a migration of the code base to PyTorch in an effort to speed up our performance. However, our estimates were never as good as “vanilla” Gradient Descent. Though flat gradients could have been problematic, it is also possible that a mistake was made in the implementation that we were never able to resolve.

3.4.5 Conclusions on Likelihood Maximization

Our comprehensive investigation of likelihood maximization approaches for the GSMP-VGLM model leads to several important conclusions:

1. **Superiority of Gradient-Based Methods:** Gradient Descent is generally a slower and more inefficient solver in general. However, Gradient Descent and its variants consistently outperformed Hessian-based methods in terms of reliability and robustness, directly addressing the potential issues caused by near zero values of the Hessian in the likeli-

hood function.

2. **Improved Initialization:** Being able to leverage the solutions used from the Normal Equations 3.2.2 as a starting point provides an excellent initialization, particularly when dealing with small values of α . This can allow for smaller step sizes and maximum iterations.
3. **Step Size Control as Key Factor:** Proper step size management, particularly through decaying schedules, has been instrumental in producing robust optimization results.
4. **Practical Implementation:** Our current gradient descent implementation with enhanced initialization strategies and step-size management offers a practical solution that can be applied on generic problems without requiring specialized skills and lots of tuning work.

These findings fundamentally changed our approach to how we approached estimation and implementation of the GSMP-VGLM in practice. Though our initial attempts utilized more traditional Hessian-based methods (either directly to maximize the likelihood function or indirectly as part of `fsolve`), our thorough analysis uncovered that known issues with Hessian-based solvers could easily apply in this case. Since we wanted our solution to

be robust to all possible inputs, we opted towards a more reliable, if traditionally more inefficient solver. Our work shows that our current approach, augmented as it is with initialization from the Normal Equations and with step-size decay schedules offers a practical solution that can be applied to any generic problem without requiring specialized skills on behalf of the users or a lot of time spent tuning.

3.5 Telescopic Grid Search for α Parameter Estimation

The estimation of the shape parameter α in the GSMP-VGLM framework presents unique challenges that necessitate specialized approaches. While the methods discussed in previous sections provide effective strategies for estimating the parameter vectors $\boldsymbol{\eta}$ and $\boldsymbol{\beta}$, the estimation of α requires additional consideration due to its distinct role in the model and its complex interaction with other parameters. This section introduces and analyzes the Telescopic Grid Search method, a systematic approach to determining the optimal value of α that combines the efficiency of directed search with the reliability of global exploration.

3.5.1 The Challenge of α Estimation

As we showed in Chapter 2, *an* MLE for α exists, but 1) we cannot prove that it is unique or 2) provide a mechanism to find it. A natural consideration would be if we could apply our solutions in Section 3.5 and use gradient-based solvers to estimate $g(\alpha, \boldsymbol{\eta})$ directly. This proved to be difficult to implement for several major reasons:

1. **Bounded Parameter Space:** Unlike $\boldsymbol{\eta}$ and $\boldsymbol{\beta}$, which can take any real values, α is constrained to be positive ($\alpha > 0$). This bounded nature of the parameter space introduces a large challenge for gradient-based methods, which may attempt to “step” into infeasible regions without controls that can be difficult to implement.
2. **Interactions between α and $\boldsymbol{\eta}$:** As demonstrated in Chapter 2, the value of $\boldsymbol{\eta}$ is tied to the value of α , i.e. $\boldsymbol{\eta}(\alpha)$. This means that changes in α have an effect on the estimation of $\boldsymbol{\eta}$ that is not going to be detected in the gradient of $\boldsymbol{\eta}$ alone and can cause bad steps in the $\boldsymbol{\eta}$ solver. This means that there would have to be some sort of nested optimization procedure.
3. **Volatility in $v(\alpha)$:** As we showed in Section 2.3.2, the form of $v(\alpha)$

can change dramatically depending on the true value of α . Since this isn't known in advance, it would be very difficult to find parameters that would generalize well across all problems.

These issues make direct application of standard optimization techniques extremely challenging for estimating α in practice and would require a good deal of fine-tuning. To address this, we sought to find other optimization approaches that could meet these challenges while still being robust and practical to implement.

3.5.2 Conceptual Framework of Telescopic Grid Search

Telescopic Grid Search (TGS) is a specialized global optimization approach used commonly in Machine Learning application to find optimal hyperparameters or parameters by searching through the resulting parameter spaces. TGS is an extension of Exhaustive Grid Search where *all* possible combinations of parameters are checked to find the optimal response. TGS extends this idea by using multiple iterations of grids that successively narrow around ideal candidates after each iteration, much like how viewing stars through a telescope begins with a wide focus to find the field of interest before continuing to zoom in until a clear picture of the region of interest is achieved.

In this way, Telescopic Grid Search creates a more efficient search procedure that can still find accurate estimates of the global optima while reducing the overall time spent in less-optimal regions.

At a high level, the TGS algorithm for α estimation proceeds as follows:

1. **Initial Coarse Grid:** Define a wide, coarse grid of candidate α values spanning the feasible range.
2. **Profile Likelihood Evaluation:** For each candidate α value, find η and β with fixed α and evaluate the resulting log-likelihood.
3. **Region Identification:** Identify the region(s) containing the highest log-likelihood values. However, from our analytical study, we believe this to be a single region in $v(\alpha)$.
4. **Grid Refinement:** Generate a finer grid concentrated within the identified high-likelihood region.
5. **Iterative Refinement:** Repeat the evaluation and refinement process until some kind of stopping criteria is reached (more to this discussion later).

6. **Final Output:** Return the value of α that produced the highest log-likelihood and associated $\boldsymbol{\eta}$ and $\boldsymbol{\beta}$.

This approach effectively decomposes the complex optimization problem of three parameters, α , $\boldsymbol{\eta}$ and $\boldsymbol{\beta}$, into a series of nested optimizations.

3.5.3 Detailed Algorithm Specification

We now present a formal specification of the Telescopic Grid Search algorithm for α estimation in the GSMP-VGLM framework with complete pseudo-code.

Note: Though this is a common technique, the specific implementation below is mine, and includes early termination based on changes in the log-likelihood, though this could be changed/augmented with estimate precision or removed entirely to just force the maximum number of iterations be run.

3.5.4 TGS Details

Several key aspects of this algorithm warrant further elaboration:

Grid Generation

The function `GenerateGrid` creates a grid of values that are spaced according to specified bounds. In practice this takes one of two forms:

Algorithm 1 Telescopic Grid Search for α Estimation

```

1: Input: Data  $\mathbf{X}, \mathbf{N}, \mathbf{W}, \mathbf{Z}$ ; initial range  $[\alpha_{\min}, \alpha_{\max}]$ ; grid size  $k$ ; convergence threshold  $\epsilon$ ; maximum iterations  $M$ 
2: Output: Estimated parameters  $\hat{\alpha}, \hat{\boldsymbol{\eta}}, \hat{\boldsymbol{\beta}}$ 
3: Initialize iteration counter  $t \leftarrow 0$ 
4:  $\alpha_{\text{grid}} \leftarrow \text{GenerateGrid}(\alpha_{\min}, \alpha_{\max}, k)$   $\triangleright$  Generate a uniformly spaced grid
5:  $\mathcal{L}_{\text{best}} \leftarrow -\infty$   $\triangleright$  Initialize best log-likelihood
6: while  $t < M$  do
7:    $\mathcal{L}_{\text{grid}} \leftarrow \{\emptyset\}$   $\triangleright$  Initialize empty list for log-likelihood values
8:   for each  $\alpha_i \in \alpha_{\text{grid}}$  do
9:     Optimize  $\boldsymbol{\eta}_i, \boldsymbol{\beta}_i$  with fixed  $\alpha_i$  using gradient descent
10:    Compute log-likelihood  $\mathcal{L}(\alpha_i, \boldsymbol{\eta}_i, \boldsymbol{\beta}_i)$ 
11:     $\mathcal{L}_{\text{grid}} \leftarrow \mathcal{L}_{\text{grid}} \cup \{\mathcal{L}(\alpha_i, \boldsymbol{\eta}_i, \boldsymbol{\beta}_i)\}$ 
12:   end for
13:    $i_{\text{best}} \leftarrow \arg \max_i \mathcal{L}_{\text{grid}}$   $\triangleright$  Index of best  $\alpha$  value
14:    $\alpha_{\text{best}} \leftarrow \alpha_{\text{grid}}[i_{\text{best}}]$ 
15:   if  $|\mathcal{L}_{\text{grid}}[i_{\text{best}}] - \mathcal{L}_{\text{best}}| < \epsilon$  then
16:     break  $\triangleright$  Convergence achieved
17:   end if
18:    $\mathcal{L}_{\text{best}} \leftarrow \mathcal{L}_{\text{grid}}[i_{\text{best}}]$ 
19:    $i_{\min} \leftarrow \max(0, i_{\text{best}} - 1)$ 
20:    $i_{\max} \leftarrow \min((k - 1), i_{\text{best}} + 1)$ 
21:    $\alpha'_{\min} \leftarrow \alpha[i_{\min}]$   $\triangleright$  Refine bounds
22:    $\alpha'_{\max} \leftarrow \alpha[i_{\max}]$ 
23:    $\alpha_{\text{grid}} \leftarrow \text{GenerateGrid}(\alpha'_{\min}, \alpha'_{\max}, k)$   $\triangleright$  Generate refined grid
24:    $t \leftarrow t + 1$ 
25: end while
26: Perform final optimization of  $\boldsymbol{\eta}, \boldsymbol{\beta}$  with fixed  $\alpha_{\text{best}}$ 
27: return  $\alpha_{\text{best}}, \boldsymbol{\eta}, \boldsymbol{\beta}$ 

```

1. **Linear-Scale Spacing:** Here each point in the space is a fixed distance apart from the subsequent values. This is used when there is not a large range in potential candidate values.
2. **Log-scale Spacing:** In this regime, points are spaced uniformly in the log-scale. This is used when there is a broad range of values that span multiple orders of magnitude.

In practice, the linear-scale, a spaced grid with k points between α_{min} and α_{max} is generated as:

$$\alpha_i = \alpha_{min} + i(\alpha_{max} - \alpha_{min}), \quad i = 0, 1, \dots, k - 1, \quad (3.20)$$

while the logarithmically spaced grid is generated as:

$$\alpha_i = \alpha_{min} \cdot \left(\frac{\alpha_{max}}{\alpha_{min}} \right)^{i/(k-1)}, \quad i = 0, 1, \dots, k - 1 \quad (3.21)$$

Refinement Strategy

A common approach used in TGS is to define a “magnification/refinement factor” which defines a neighborhood around α_{best} and uses this to restrict the search space in coming intervals. The value of this factor dictates how aggressively the space is focused.

However, in our analysis, we found this to be a problematic detail in practice due to two issues:

1. **Lack of Optimal Magnification Factor Guidance:** The inability to set a robust magnification scale. Since the magnification factor determines how fast the function can converge around the optimum, there is a direct correlation between the function being optimized, $v(\alpha)$, the magnification factor, and the maximum number of iterations, and without knowledge of $v(\alpha)$, it isn't possible to state an ideal magnification factor that would be robust to all situations that doesn't involve having to spend a large number of iteration steps.
2. **Poor Behavior When True α is Near Zero:** In the case where the real value of α was either 0 or very small, the magnification factor could wind up "zooming away" from the true value, with successive iterations getting further and further away from the optimal value.

Instead, we chose to anchor the successive bounds to the value to the left and right of the optimal value found in practice (unless the optimal value were in the boundary). This effectively defined the shrinkage bound and generally

ensured a good rate of enhancement while also preventing a large number of iterations and poor behavior when the true value of α was small. An additional benefit was that at least two points were re-evaluated and if the gradient descent procedures have been dramatically adjusted, this can provide clear evidence of bad tuning if the repeated points exhibit high volatility.

Convergence Criteria

The algorithm employs two convergence criteria:

1. **Grid-Width Threshold:** The algorithm terminates when the width of the telescopic grid is no larger than some pre-defined threshold δ . This criterion is analogous to imposing a numeric precision constraint on α that must be met.
2. **Function Improvement Threshold:** The algorithm terminates when the improvement in log-likelihood between successive iterations falls below a pre-specified threshold ϵ . This criterion identifies when further refinement is unlikely to yield significant improvements.
3. **Maximum Iterations:** A maximum number of magnification steps is defined and after the final step is run, the α which produced the largest log-likelihood at any step is returned. This provides no guarantees on

This parallel implementation can reduce wall-clock time substantially, particularly as the number of cores increases. In particular, if the number of estimation points in each round is a multiple of cores, then it becomes possible to explore more candidate α values in each iteration without increasing run time. This typically means that a smaller maximum iteration size may be chosen and/or increased likelihood of early termination. This makes TGS far more robust and competitive even for larger initial search spaces.

3.5.5 Visual Demonstration of Telescopic Grid Search

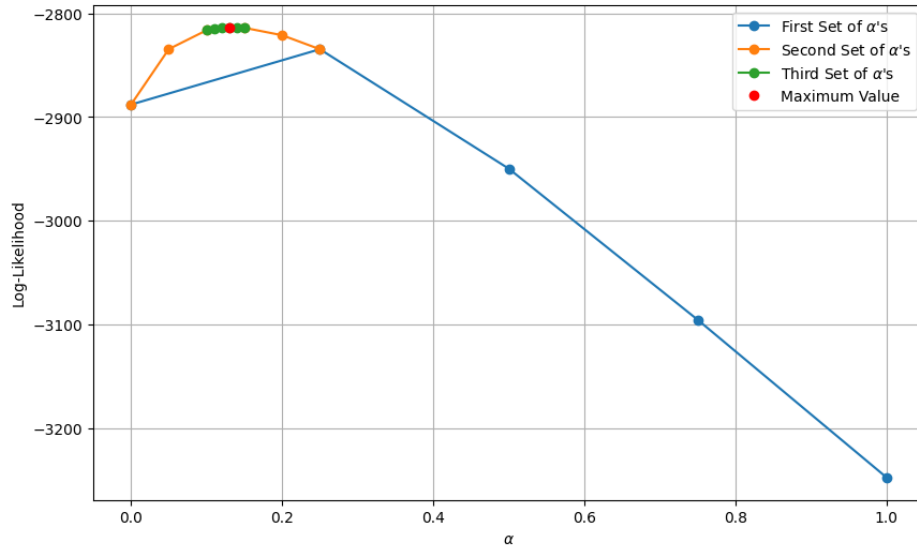


Figure 3.4: Three iterations of Telescopic Grid Search. The x-axis shows values of α while the y-axis shows the log-likelihood values associated with the points. Only points with dots represent actually estimated values; the connecting lines are linear interpolations between them. The blue line represents the first set of estimated values which spans the width of the x-axis. The orange line shows the next iteration and the green represents the final iteration. The red dot is the actual value of α used to generate the data.

To provide intuition for the Telescopic Grid Search process, Figure 3.4 demonstrates how the algorithm progressively narrows its focus to identify the optimal α value. This example illustrates the algorithm's behavior on a dataset generated from a GSMP-VGLM model with true $\alpha = 0.13$.

Several key observations can be made from this visualization:

1. **Initial Exploration:** The first iteration (blue curve) covers a wide

range of α values, allowing a large swath to be explored.

2. **Progressive Refinement:** The next iteration (orange curve) explores only a region with a greater likelihood of containing the true value. This reviews a smaller domain, but with increased likelihood of containing the optimal α .
3. **Final Refinement:** The final iteration (green curve) provides fine-grained exploration around the maximum likelihood estimate, correctly identifying the maximum value of $\alpha = 0.13$.

This visualization demonstrates the efficiency of TGS in navigating from a broad initial search to a precise final estimate, by systematically narrowing the search space based on the results of the prior round.

3.5.6 Comparison to Other Exploratory Searches

Though another more complicated approach exists (and will be discussed in the next section), for comparison, we can contrast this against two other exploratory methods:

- **Random Search (RS):** This method allows a user to specify the number of samples to be drawn from a potential search space. Each

point is evaluated and the randomly drawn α that reveals the largest log-likelihood value is selected as the optimal value.

- **Exhaustive Grid Search (EGS):** This approach foregoes iterative searching that begins broadly before narrowing in on the regions more likely to contain the true maximum value. Instead, a uniformly spaced array of points is proposed and evaluated, where the α producing the largest log-likelihood is selected as the optimal value. In practice, this grid tends to sample points with very small spacing between points, meaning many points need to be sampled.

Both of these alternatives suffer significant drawbacks compared to TGS. In particular, with RS, there is no guarantee that the true optimum is reached, and any information learned in the sampling is not used to improve subsequent estimation. GGS can generally provide good results if the spacing is fine-grained enough. Unfortunately, this means that a large number of points needs to be sampled, and many points are in regions that clearly are suboptimal. Instead, TGS provides a simple mechanism that will learn from earlier iterations and spend time searching in domains with high likelihood of containing the true optimum with minimal tuning or computational complexity.

3.5.7 Practical Recommendations

Based on our extensive experimentation with Telescopic Grid Search for α estimation, we offer the following practical recommendations for implementing this approach:

1. **Broad Initial Grid Bounds:** Unless there is strong evidence for a specific range of α , we recommend making the initial grid fairly broad, containing a value close to 0 and a reasonably large upper bound, such as 10. A log-scale grid could be used here, but so could a linear-scale grid. It is also possible to combine both approaches, with initial searches at the log-scale before moving to a linear-scale grid when the optimal grid is between consecutive powers.
2. **Be Skeptical of Optimal Values Near the Boundary:** If the resulting optimal value is very near the initial boundary, consider computing the A test statistic from Equation 2.33 and fitting a TETLG-VGLM model as well. If the result is near the upper bound, consider rerunning the analysis with a larger upper bound.
3. **Parallel Implementation:** This approach may still require a large number of test points. Fortunately, each magnification step can be

parallelized for improved speed, and this could make the approach quite practical.

These recommendations provide a practical guide for implementing Telescopic Grid Search in diverse applications, ensuring reliable and efficient estimation of the α parameter in the GSMP-VGLM framework.

3.5.8 Conclusions on Telescopic Grid Search

The Telescopic Grid Search approach represents a refinement of exhaustive parameter searching that lends itself easily to α estimation in the GSMP-VGLM framework. By combining global exploration with progressive refinement, TGS effectively navigates the shape of $v(\alpha)$ which we defined in Equation 2.34, to identify the optimal α value with high reliability and computational efficiency.

Our simulations found that TGS outperforms alternative approaches across multiple dimensions of performance—accuracy, reliability, efficiency, and ease of implementation. The method’s natural parallelizability further enhances its practical utility, making it particularly well-suited for applications involving larger datasets or more complex model specifications.

Though less efficient than other models which we will discuss briefly, the simplicity of the approach combined with the ability to parallelize each estimation point makes TGS a practical alternative to α estimation. In the subsequent section, we will explore an alternative approach to global optimization for α estimation based on Bayesian optimization principles, providing a complementary approach to handle the estimation process.

3.6 Bayesian Optimization for α Parameter Estimation

While the Telescopic Grid Search method described in the previous section provides a robust approach to estimating the α parameter in our GSMP-VGLM framework, it does possess a few significant drawbacks. While it does learn from prior attempts, the estimation process is hardly intelligent and merely repeats a coarse search pattern with increasing precision. In cases where limiting computation may be at a premium, this approach will waste time spending too much time exploring areas that couldn't be valid. An alternative solution that can improve on this is Bayesian Optimization (BO). Unlike other exhaustive and random search methods, this approach attempts

to construct a surrogate of the underlying function being optimized given the observed data. Defining a function on the surrogate function that blends observed data and model uncertainty, BO can suggest new candidate values to test. Bayesian Optimization can often achieve comparable performance to other exhaustive search algorithms with far fewer evaluations. In this section, we explore the theoretical foundations of Bayesian Optimization, how it can be implemented into our GSMP-VGLM estimation framework.

3.6.1 Theoretical Foundations of Bayesian Optimization

Bayesian Optimization can be especially useful in cases where

1. There is a multi-stage approach in parameter estimation.
2. Testing points may be complicated, expensive or slow.
3. The function of interest may not have gradients or have complicated bounding of parameters.

Bayesian Optimization attempts to overcome these issues by building a probabilistic model of the function of interest. Fundamentally there are three components to Bayesian Optimization:

1. **Surrogate Model:** A probabilistic model that approximates the function of interest and quantifies uncertainty about its predictions.

2. **Acquisition Function:** A function derived from the surrogate model that guides the selection of points for evaluation by balancing exploration (sampling in regions of high uncertainty) and exploitation (sampling in regions with promising expected values).
3. **Sequential Sampling Strategy:** An iterative process that alternates between updating the surrogate model based on observed function evaluations and selecting new points to evaluate based on the acquisition function. This will not get much attention here as we utilized a very simple sampling strategy that alternated between selecting a single new candidate and then evaluating it.

In our case, we seek to find the α that optimizes Equation (2.14):

$$\mathcal{L}(\alpha) = \max_{\boldsymbol{\eta}(\alpha), \boldsymbol{\beta}} \mathcal{L}(\alpha, \boldsymbol{\eta}(\alpha), \boldsymbol{\beta}) \quad (3.22)$$

Since this involves bounds on α ($\alpha \in (0, \infty)$) and optimizing a $\boldsymbol{\eta}(\alpha)$ for a given α , GSMP-VGLM estimation is an ideal candidate for Bayesian Optimization.

3.6.2 Gaussian Process Surrogate Functions

The heart and soul of the probabilistic model is the surrogate model, from which all other aspects of the model follow. In practice, the most commonly used form of probabilistic model used as the surrogate is the Gaussian Process, in large part due to their closed-form, flexibility and ability to capture the uncertainty. They are quite popular and many libraries exist to train and fit Gaussian Processes computationally, including several that utilize GPU's for enhanced performance.

A Gaussian Process is a stochastic process where any finite collection of points from the process have a multivariate Gaussian distribution, see Rasmussen and Williams (2006). More formally, given a function, f , and a set of points, $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, if $\{f(\mathbf{x}_1), f(\mathbf{x}_2), \dots, f(\mathbf{x}_n)\} \sim \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}), \mathbf{K})$, then f is a Gaussian Process, $\mathcal{GP}(\boldsymbol{\mu}(\mathbf{x}), \mathbf{K})$. $\boldsymbol{\mu}(\mathbf{x})$ is the mean function and \mathbf{K} is an $n \times n$ matrix with elements $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ for $i, j \in \{1, 2, \dots, n\}$ where $\kappa(\cdot, \cdot)$ is a covariance function. The latter function is also known as a kernel function, and this plays the key role in the \mathcal{GP} , since it constrains the way that functions can behave between points. Fortunately, covariance functions need only meet a few basic criteria, and covariance functions may even be combined to

make new covariance functions, meaning there is a great deal of flexibility in practice.

Another feature we alluded to earlier was that there is a closed-form solution afforded by this model. In particular, given a set of inputs, $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ and corresponding outputs, $\mathbf{F} = \{f(\mathbf{x}_1), f(\mathbf{x}_2), \dots, f(\mathbf{x}_n)\}$, we have an analytical solution for the posterior predictive distribution around any new point, \mathbf{x}_* :

$$f_* | \mathbf{x}_*, \mathbf{X}, \mathbf{F} \sim \mathcal{N}(\mu_*, \sigma_*)$$

where

$$\mu_* = \boldsymbol{\mu}(\mathbf{x}_*) + \kappa(\mathbf{x}_*, \mathbf{X}) \mathbf{K}^{-1} (\mathbf{F} - \boldsymbol{\mu}(\mathbf{X})) \quad \text{and}$$

$$\sigma_*^2 = \kappa(\mathbf{x}_*, \mathbf{x}_*) - \kappa(\mathbf{x}_*, \mathbf{X}) \mathbf{K}^{-1} \kappa(\mathbf{X}, \mathbf{x}_*)$$

Gaussian Processes for GSMP-VGLM

For our purpose, we want to approximate $\mathcal{L}(\alpha)$, i.e.

$$\mathcal{L}(\alpha) \sim \mathcal{GP}(\boldsymbol{\mu}(\alpha), \mathbf{K}_\alpha)$$

where the elements of \mathbf{K}_α consist of the covariance function in the form $\kappa(\alpha, \alpha')$. We replace \mathbf{X} and \mathbf{F} with $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \dots, \alpha_n]^\top$, the tested values

of α and $\mathcal{L} = [\mathcal{L}(\alpha_1), \mathcal{L}(\alpha_2), \dots, \mathcal{L}(\alpha_n)]^\top$, the corresponding log-likelihood function values.

3.6.3 Acquisition Functions for Bayesian Optimization

Having established our surrogate model, we need to be able to convert this information into a different function (the Acquisition function) that, when maximized, will suggest a new candidate value of α to try. There are several forms of Acquisition function which we'll discuss below.

Upper Confidence Bound (UCB)

The UCB acquisition function combines the mean functions of the Gaussian Process with a multiple of the size of the uncertainty at that point:

$$\text{UCB}(\mathbf{x}) = \boldsymbol{\mu}(\mathbf{x}) + \gamma \cdot \sigma(\mathbf{x})$$

where $\gamma > 0$ is a defined parameter that changes how much the function examines unexplored regions rather than trying to continue examining regions that have large values of $\boldsymbol{\mu}(\mathbf{x})$. UCB effectively looks at vertical “slices” of the surrogate function to determine the most likely regions containing the maximum value. This will be in contrast to the next two acquisition functions.

Probability of Improvement (PI)

Unlike UCB which looks at vertical “slices”, the PI acquisition function maximizes the probability of improving over the current best observation, $f_{\max} := \{f(\mathbf{x}_1), f(\mathbf{x}_2), \dots, f(\mathbf{x}_n)\}$. This is akin to predicting the probability of being above a threshold:

$$\text{PI}(\mathbf{x}) = P(f(\mathbf{x}) \geq f_{\max} + \gamma) = \Phi\left(\frac{\boldsymbol{\mu}(\mathbf{x}) - f_{\max} - \gamma}{\sigma(\mathbf{x})}\right),$$

where $\Phi(\cdot)$ is the CDF of the $\mathcal{N}(0, 1)$ distribution and $\gamma > 0$ is a constant to encourage that unexplored regions are examined.

Expected Improvement (EI)

In practice, PI can be extremely greedy and produce poor results. This is because PI only cares about high probability of improvement and does not try to consider *the magnitude* of the improvement. The EI acquisition function tries to quantify the expected gain relative to the current maximum:

$$\text{EI}(\mathbf{x}) = \mathbb{E}[\max(f(\mathbf{x}) - f_{\max} - \gamma, 0)],$$

which has a closed-form expression:

$$\text{EI}(\mathbf{x}) = (\boldsymbol{\mu}(\mathbf{x}) - f_{\max} - \gamma) \Phi(Z) + \sigma(\mathbf{x}) \phi(Z),$$

where

$$Z = \frac{\mu(\mathbf{x}) - f_{\max} - \gamma}{\sigma(\mathbf{x})},$$

and $\phi(\cdot)$ is the PDF of the $\mathcal{N}(0, 1)$ distribution and $\gamma > 0$ has the same role in PI.

3.6.4 Implementation of Bayesian Optimization for GSMP-VGLM

With this foundation, it is quite straight-forward to implement this in the GSMP-VGLM case. The core algorithm for Bayesian Optimization of the α parameter is outlined below:

As can be clearly seen in the algorithm, you provide initial bounds on potential ranges for α , the number of points to evaluate, an initial testing point and an acquisition function. An initial GP is fit to the evaluation point and then the acquisition function is applied. This process repeats iteratively until some notion of convergence has been achieved or until the maximum number of samples has been evaluated.

In stark contrast to the Telescopic Grid Search algorithm described in Section 3.6, this approach does not blindly evaluate all points in a space; rather it only looks at areas that are deemed to have high potential at including superior

Algorithm 2 Bayesian Optimization for α Estimation

- 1: **Input:** Data $\mathbf{X}, \mathbf{N}, \mathbf{W}, \mathbf{Z}$; bounds $[\alpha_{\min}, \alpha_{\max}]$; initial test point, α_0 ; total sample size n_{\max} ; acquisition function $a(\alpha)$
 - 2: **Output:** Estimated parameters $\hat{\alpha}, \hat{\boldsymbol{\eta}}, \hat{\boldsymbol{\beta}}$.
 - 3: Generate initial design $\mathbf{A} = [\alpha_0]$
 - 4: Evaluate profile log-likelihood at initial points: $\mathbf{L} = [\mathcal{L}(\alpha_0)]$
 - 5: Find current best: $\alpha^+ \leftarrow \arg \max_i \mathcal{L}(\alpha_i), \mathcal{L}^+ \leftarrow \mathcal{L}(\alpha^+)$
 - 6: **for** $t = 1$ to n_{\max} **do**
 - 7: Fit Gaussian Process model to \mathbf{A} and \mathbf{L}
 - 8: Compute acquisition function $a(\alpha)$ using the GP model
 - 9: Find next point: $\alpha_t \leftarrow \arg \max_{\alpha \in [\alpha_{\min}, \alpha_{\max}]} a(\alpha)$
 - 10: $\mathbf{A} = \mathbf{A} \cup \{\alpha_t\}$ ▷ Update \mathbf{A}
 - 11: Evaluate profile log-likelihood at α_t : $\mathcal{L}_t \leftarrow \mathcal{L}(\alpha_t)$
 - 12: $\mathbf{L} = \mathbf{L} \cup \{\mathcal{L}(\alpha_t)\}$ ▷ Update \mathbf{L}
 - 13: **if** $\mathcal{L}_t > \mathcal{L}^+$ **then**
 - 14: **if** $|\mathcal{L}^+ - \mathcal{L}_t| < \epsilon$ **then** ▷ Convergence achieved
 - 15: Update current best: $\alpha^+ \leftarrow \alpha_t, \mathcal{L}^+ \leftarrow \mathcal{L}_t$
 - 16: **break**
 - 17: **end if**
 - 18: Update current best: $\alpha^+ \leftarrow \alpha_t, \mathcal{L}^+ \leftarrow \mathcal{L}_t$
 - 19: **end if**
 - 20: **end for**
 - 21: Perform final optimization of $\boldsymbol{\eta}, \boldsymbol{\beta}$ with fixed α^+
 - 22: **return** $\alpha^+, \boldsymbol{\eta}, \boldsymbol{\beta}$
-

solutions. In practice, this method tends to not only converge faster, but to do so in far fewer steps.

Advanced Implementation Details

The version presented above is a minimum working example, though there are several ways to boost performance of this model in practice, depending on what additional information and resources are available. These are:

1. **Including Previous Measurements:** It is possible that there have been some initial studies done where known values of **A** and **L** have already been established. These values can be passed into the optimizer which allows users to go directly to the optimization `for`-loop.
2. **Provide range of α Values:** Rather than passing a simple value, it is possible to pass a grid of candidate values through, as in the first step of Telescopic Grid Search. This can be done if there is strong intuition where the maximum may lie, or to prevent over-exploration in suboptimal areas, though this tends to defeat the purpose if the range of possible values is large.
3. **Parallel Implementation:** As in Telescopic Grid Search, it is also possible to parallelize the search process. To accomplish this, rather

than returning the optimal value of the acquisition function, the top k values are returned and each thread is used to evaluate these points, where k , the number of available processing cores, is a factor of n_{\max} . This can generally yield faster convergence.

Benefits Over Telescopic Grid Search

In our analysis, we found that Bayesian Optimization tended to yield superior estimates in fewer steps. The Acquisition function tends to do a quite good job in practice at evaluating optimal values and ignoring suboptimal areas. In this way, it overcomes one of the two biggest issues we discussed in Telescopic Grid Search: the lack of optimal magnification factor guidance.

This was an issue because, depending on the true value of α , the shape of $v(\alpha)$ has different shapes and prevents robust measures to magnify the search space. Since this fundamentally restricts the size of the bounds of α values being considered, a bad magnification can cause irreparable harm to the estimation process and wind up excluding the optimal value. However, Bayesian Optimization never restricts the original bounds of α . So as long as the true value of α lies in the bounds, it is always possible to discover it at any stage.

The other related issue of Telescopic Grid Search was how zooming in when the true value of α was near zero is also not a problem in Bayesian Optimization for the same reason discussed as above: it never restricts the original bounds and can always explore the boundary. This was borne out in simulation studies, even when the true α was 0.

3.6.5 Visual Demonstration of Telescopic Grid Search

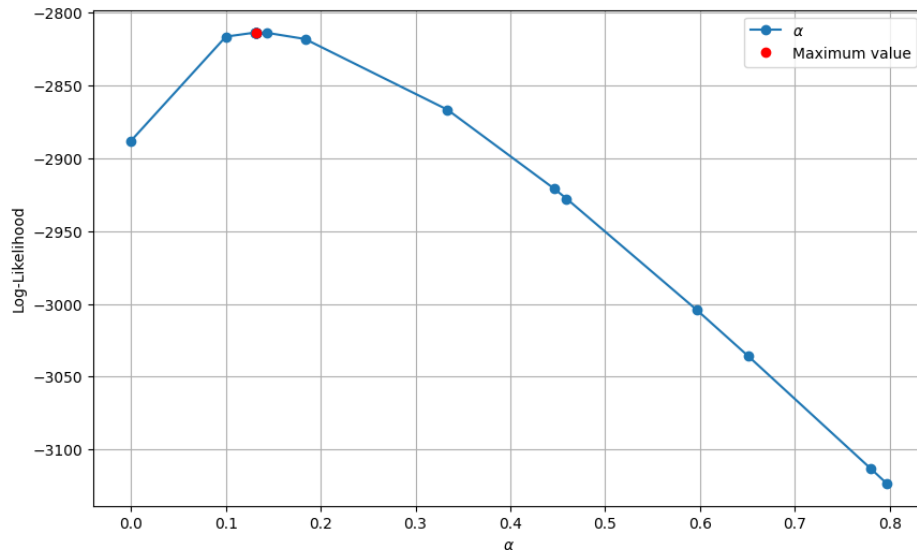


Figure 3.5: The results of 30 steps of Bayesian Optimization to find α . The x-axis is the range of potential α values that can be explored and each dot represents a tested value of α and the y-axis is the associated log-likelihood value. Only blue dots were evaluated and the lines connecting them are linear interpolations between known points. The red dot between a blue dot at the top is the true value of α used to generate the data.

Using the same setup as in Section 3.5.5, data was drawn from a GSMP-VGLM with the true value of $\alpha = 0.13$. 30 points were evaluated without any early-stopping for convergence. The initial value provided was the right boundary value of the range. Figure 3.5 presents the results of the BO process. The key observations are:

1. **Non-Uniform Searching:** Unlike Telescopic Grid Search, points are non-uniformly sampled, so the general shape of $v(\alpha)$ is not as clearly defined as in Section 3.5.5.
2. **Generally Exploring Optimal Ranges:** Though not shown, Bayesian Optimization approached the optimal value with the second iteration and all values after the 10th point were effectively converged. TGS, in contrast, needed more iterations to arrive at the optimal value, and was only halted in this case because we knew the optimal value had been achieved. In practice, more iterations would have been conducted.

3.6.6 Practical Recommendations

Based on our research about estimating α with Bayesian Optimization, we offer the following practical recommendations for implementing BO in practice:

1. **Broad Initial Grid Bounds:** As in TGS, unless there is strong evidence for a specific range of α (e.g. earlier studies), we recommend passing a broad range of potential values of α , including a value very close to 0 and some reasonably large upper bound, such as 10.
2. **Pick an Initial Point near the Largest Bound:** This is a point that will generally be explored anyway, but if the true value of α is reasonably far away (i.e. 5 or less), the shape of the Gaussian Process surrogate function with $\boldsymbol{\mu}(\mathbf{x}) = 0$ will force subsequent steps to be closer to 0.

3.6.7 Conclusions on Bayesian optimization

Bayesian Optimization provided the most robust and performant method of parameter estimation in the GSMP-VGLM model in the course of our examinations. This was due to the ease of implementation, data-driven candidate proposal and robust performance over other exhaustive parameter search methods we trialed.

In our research, we found that BO outperformed TGS in both accuracy of parameter estimates as well as the average number of steps needed to converge.

Further, the two most glaring issues of TGS (poor boundary performance and unclear guidance when magnifying the α parameter range), are not issues for BO, which always keeps the original α estimation bounds. Though potentially slower (fitting a Gaussian Process and optimizing the acquisition function are added steps to consider), the ability to parallelize computation combined with intelligent candidate selection overcame the additional computation costs.

Chapter 4

Applications

In this chapter, we apply the GSMP-VGLM model to daily precipitation data from two NOAA sites close to Reno, NV. In this area precipitation is highly seasonal with heavier precipitation in the winter, as well as inherently episodic, where large periods of no precipitation are punctuated with highly variable wet periods. Any statistical model for these would need the ability to overcome these challenges.

4.1 Overview of Data

The data analyzed comes from two weather stations of the National Oceanic and Atmospheric Administration (NOAA): South Lake Tahoe Airport, which is in the Sierra Nevada mountain range, and the nearby Reno Airport which is in the western Great Basin, at the foot of the Sierras, see Menne et al.

(2012). The observations (data) were total daily precipitation values on record. Though the two locations are less than fifty miles apart, they experience markedly different precipitation regimes: one heavily influenced by Pacific frontal systems, the other by arid continental conditions.

The high quality data, was obtained from Alexander Weyant (SCRIPPS Institution of Oceanography, UCSD). For detailed information about data selection and preprocessing see Weyant et al. (2025). Next, to construct the observations of Magnitude, Maximum and Duration, we used sequences of nonzero values that exceeded the 75th percentile of nonzero daily precipitation totals for each station, respectively. This thresholding was done to ensure that we focus on moderate to extreme events, which are most relevant for hydrological and societal impact assessments.

4.2 Addressing the Seasonality of Precipitation Data

As mentioned earlier, precipitation at the two locations of interest exhibits a strong seasonal pattern that needs to be accounted for. The VGLM structure provides a natural mechanism to do so: encoding the seasonal variation as a

covariate in the matrices \mathbf{W} and \mathbf{Z} . This was done by providing a Fourier Basis Function expansion to the link parameters $\boldsymbol{\lambda}$ and \mathbf{p} :

$$\begin{aligned}\boldsymbol{\eta}^\top \mathbf{w}_i &= \hat{I} + \sum_{h=1}^H \left(\underbrace{\eta_{i,h}^c \cos\left(t \frac{2h\pi}{365.25}\right)}_{w_{i,h}^c} + \underbrace{\eta_{i,h}^s \sin\left(t \frac{2h\pi}{365.25}\right)}_{w_{i,h}^s} \right) \\ \boldsymbol{\beta}^\top \mathbf{z}_i &= \hat{I} + \sum_{h=1}^H \left(\underbrace{\beta_{i,h}^c \cos\left(t \frac{2h\pi}{365.25}\right)}_{z_{i,h}^c} + \underbrace{\beta_{i,h}^s \sin\left(t \frac{2h\pi}{365.25}\right)}_{z_{i,h}^s} \right)\end{aligned}\tag{4.1}$$

where \hat{I} is a bias term (i.e. ‘‘Intercept’’), H is the number of harmonic terms we wish to include ($H = 2$ in our application) and t is the integer day of year, where January 1 is $t = 1$. Since $H = 2$, we have two sine harmonics and two cosine harmonics for $\boldsymbol{\eta}$ and $\boldsymbol{\beta}$ respectively, including the bias term, meaning \mathbf{W} and \mathbf{Z} each consist of five columns. This parameterization allows the model to capture complex seasonal patterns without adding the risk of significant overfitting compared to other methods (e.g. adding dummy variables for individual days) or degraded performance (e.g. using dummies for specific months or weeks). Finally, a single α parameter was estimated across all observations for each station

4.3 Case Study: South Lake Tahoe

We used data from 1968-2024 to fit a GSMP-VGLM¹ model to the observed total daily precipitation values. Figure 4.1 illustrates the fit of the model to data from the South Lake Tahoe Airport in addition to observed values. The plot in the upper left shows the probability of observing total daily precipitation over the 75th (local) percentile threshold for every day of the year. As expected, this probability is maximized during the winter (t either small or very large). The remaining three plots show the theoretical 50th, 75th, 95th and 99th observed and fitted model (red curves) components quantiles. The points on the graphs represent values of the duration, daily mean and precipitation event total (magnitude) all precipitation events registered on a given day of the year over the period of record. In these plots, a model that fits well should have percentiles following the observed percentiles. That means the red curves (model percentiles for every day of the year) should be close to the observed percentiles of the data. Although percentiles of the data are not marked on the graphs, a model that fits well should estimate percentiles close to the observed ones. That can be seen if the percent of

¹The model fit to the precipitation data used hurdle geometric distribution for duration of events. This allowed for better fit to events of length one. More on hurdle geometric model in the next section.

observation below the red curve on a given day is approximately equal to the percentile given by the red curve. The plot in the upper right shows the event durations, and again we see a seasonal pattern in the data and in the model. The two bottom plots show the daily average as well as the total precipitation of events and here it is easier to see how well the model fits the observed data. The lowest red line represents the 50th percentile from the model, and it should separate the lower half from the upper half of the observed data on that day. As the red lines move up (showing higher percentiles), the fit continues to be very reasonable, even as we approach the 95th and 99th percentiles.

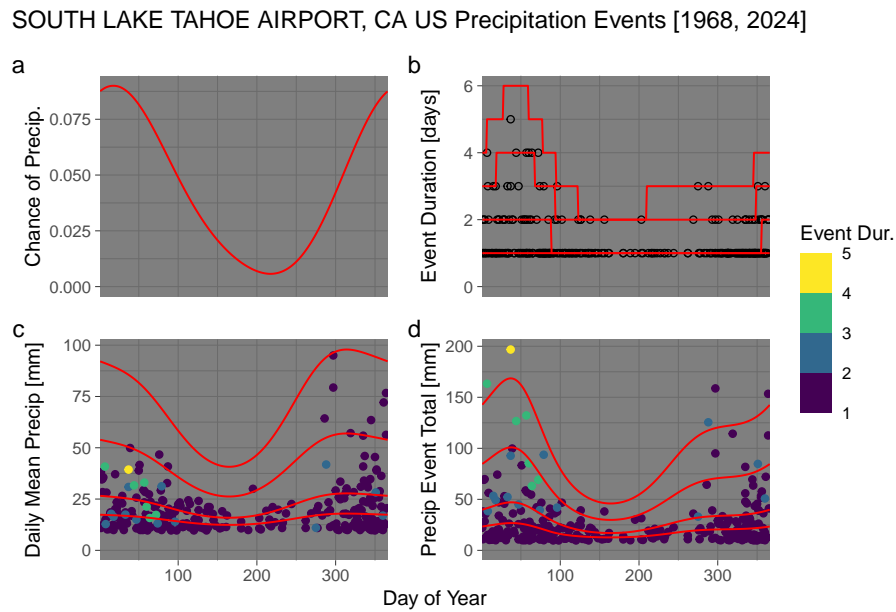


Figure 4.1: Annual cycle of precipitation events at South Lake Tahoe Airport. (a) Estimated daily probability of above-trace precipitation. (b–d) Points represent observed event properties; red curves denote theoretical percentiles (50th, 75th, 95th, 99th) under the HSMP-VGLM.

4.4 Case Study: Reno Airport

Repeating the analysis for the South Lake Tahoe airport, we also fit a model to observations from the Reno Airport, using data from 1937-2024. Though there are several commonalities, such as the clear seasonal pattern and generally strong performance of the model in representing the data, there are some clear differences that can be seen in Figure 4.2.

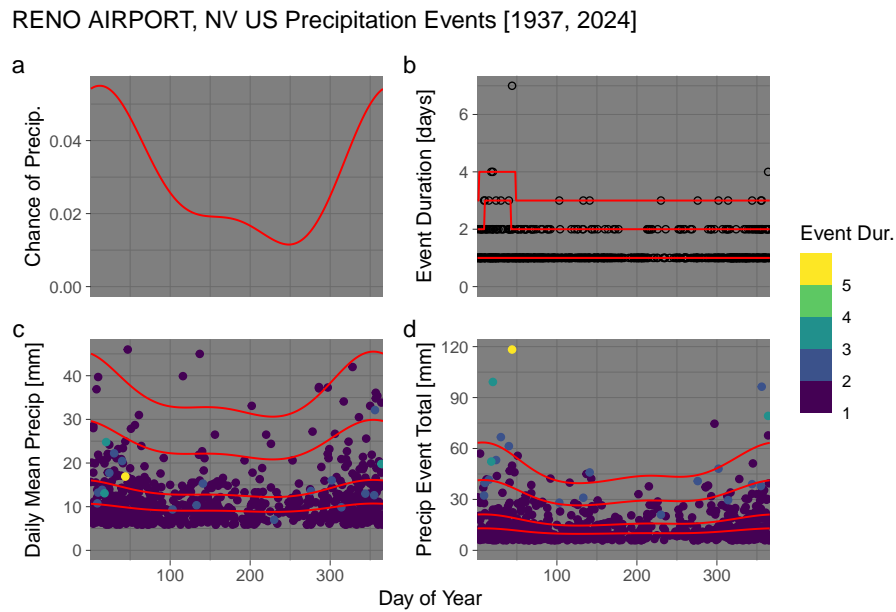


Figure 4.2: Annual cycle of precipitation events at South Reno Airport. (a) Estimated daily probability of above-trace precipitation. (b–d) Points represent observed event properties; red curves denote theoretical percentiles (50th, 75th, 95th, 99th) under the HSMP-VGLM.

The most notable differences are:

1. The seasonal pattern in Reno precipitation is different from that of South Lake Tahoe. The model traces the new pattern reasonably well.
2. The generally smaller precipitation probability and precipitation totals in Reno.
3. Though the durations of precipitation events are shorter in Reno in general, the longest event in Reno is longer than the longest event in

south Lake Tahoe.

4. The duration and magnitude values in Reno are typically more “flat” and do not exhibit the same sharp seasonal variation as those in South Lake Tahoe.

In summary, we see that the GSMP-VGLM model fits the precipitation events reasonably well in both Reno and South Lake Tahoe. However, the model based on daily total precipitation does not represent all precipitation events very well. One-day events are dominant year-round, and multi-day storms—common at Lake Tahoe in February—are rare. While the 2-harmonic model suggests limited seasonal variation, additional data sources reveal otherwise. NOAA severe weather records NOAA Events Database [16] show a distinct summertime risk from convective storms, which bring not just rain but also lightning, strong winds, and wildfires (Figure 4.3).

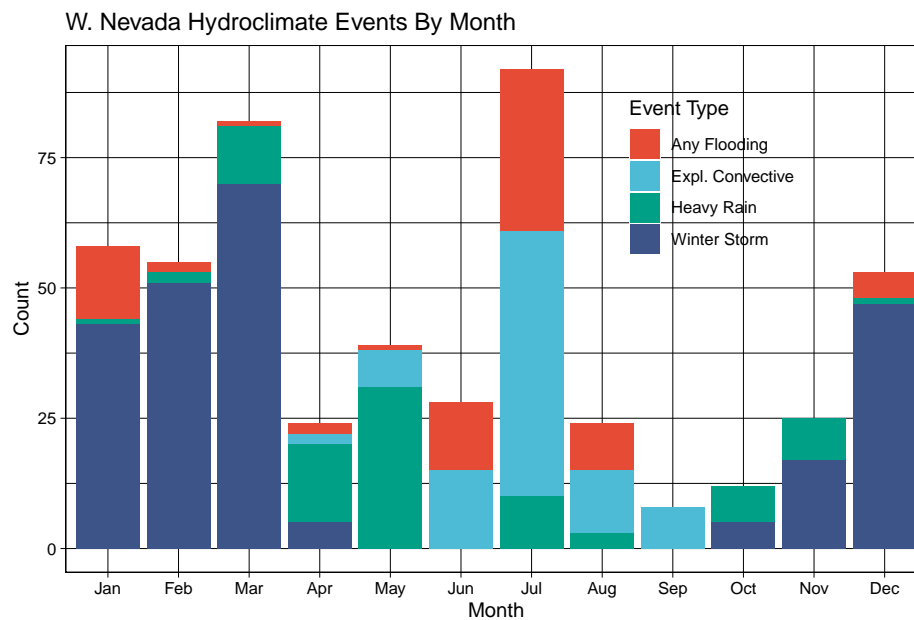


Figure 4.3: Monthly counts of severe weather reports (2004–2018) from NOAA for western Nevada counties. Event types are categorized as “Explicitly Convective”, “Winter Storm”, and “Any Flooding”.

Because our model operates at a daily resolution, high-intensity short-duration events (e.g., flash floods caused by 1 inch of rain in 30 minutes) are conflated with longer, lower-intensity events. Thus, while the model captures seasonal shifts in the statistics of daily data, it may under-represent flash flood risks.

Disclaimer: All climatological and meteorological information was collected in personal communications with Alexander Weyant.

4.5 Discussion and Extensions

In this chapter, we were able to demonstrate the effectiveness of the GSMP-VGLM model to provide reasonable fits to precipitation data from South Lake Tahoe Airport and the nearby Reno Airport. Using only a few small harmonics as covariates, the models were able to capture complex seasonality and extreme values on precipitation that can occur throughout the year. This was demonstrated via validation plots for each airport which showed generally strong fit for each model at various quantiles.

Though model fit was generally good, there are is room for improvement:

1. Being able to make α a function of covariates, as $\boldsymbol{\lambda}$ and \mathbf{p} are should provide enhanced model performance. We believe that this would be particularly useful if the data were available at even lower levels of temporal granularity.
2. As discussed above, we believe that looking at data more granular than daily would help distinguish between the impact of the short, intense events that cause flash floods.
3. Additional covariate data may also exist that can further account for

model volatility.

Ultimately, we do believe that the current GSMP-VGLM model provides a useful framework to modeling these kinds of events and may provide highly useful and reliable data to planners dealing with these kinds of precipitation events.

Chapter 5

Additional Considerations

In our work, we assumed that the marginal distribution on N is Geometric. However, in practice, we often find this assumption challenged when many events have duration equal to one. It is the case with precipitation events we considered. It is partially the consequence of the climate and meteorology, and partially of the fact that we look at "large" events, that is events over a threshold. Since the daily precipitation we work with has to be larger than a given threshold to be considered, there will be more observations with duration 1, especially as the threshold used to determine the underlying events increases. This excess of "1's" creates an issue with the geometric assumption for duration. In order to address this, we turn to Hurdle models which can help deal with the excess observations of $N = 1$.

5.1 Review of Hurdle Models

We consider a setting where the observed duration variable N exhibits excess of the values $N = 1$. That means, that the empirical probability of $N = 1$ is much larger than the $P(N = 1)$ for a geometrically distributed N . In practice, if we fit a geometric distribution to such durations, the resulting fit will tend to under-predict the number of events with duration $N = 1$, and be a poor fit for durations $N > 1$. One potential solution is to modify the geometric model to a *Hurdle geometric* model. A hurdle model puts a point mass at $N = 1$ and then distributes the remaining geometric probabilities among $N > 1$. It is a mixture model that puts a point mass at $N = 1$ with some probability π , and a truncated geometric distribution for $N > 1$. Another popular solution is the *zero-inflated model*. A comprehensive review of both hurdle and zero-inflated models can be found in Zuniga (2020).

Mathematically, for a counting variable N with values $1, 2, \dots$ with probability mass function $f(n)$, a hurdle model that allows for $P(N = 1)$ to be larger

than $f(1)$ has the form:

$$\begin{aligned} P(N = 1) &= \pi, \\ P(N = k) &= (1 - \pi) \cdot \frac{f(k)}{1 - f(1)} \quad \text{for } n \geq 1, \end{aligned} \tag{5.1}$$

where $0 < \pi < 1$. In the hurdle model, the base variable N can be any counting variable, such as Geometric, Poisson, or Negative Binomial.

A mixture representation of the hurdle distribution, with adjusted probability of 1s, based on a random variable N with PMF f , has PMF as follows.

$$f_H(n) = \pi I_{\{1\}}(n) + (1 - \pi) f_T(n), \quad n = 1, 2, \dots, \tag{5.2}$$

where $0 < \pi < 1$, $I_{\{1\}}(n)$ is the indicator of $N = 1$, and

$$f_T(n) = \frac{f(n)}{1 - f(1)}, \quad n = 2, 3, \dots$$

is the PMF of the random variable T which is N truncated at 1.

5.2 Hurdle Model Implementation in GSMP-VGLM

In this section we give a brief overview of the implementation of the hurdle geometric model instead of the geometric model for the duration N in the

GSMP-VGLM. Detailed work on this model is outside the scope of this dissertation.

Since in practice with the precipitation data we saw more GSMP events of length one than allowed by geometrically distributed duration N , we first implement hurdle geometric model for N in the GSMP framework. We first consider GSMP model, where the distribution of N follows a hurdle geometric distribution.

Let N have the 1-inflated hurdle PMF given by (5.2) with the base geometric distribution and let N be independent of X_1, X_2, \dots . Note that N has two parameters: π and p . Let $(X = \sum_{i=1}^N X_i, Y = \max_{i=1}^N X_i, N)$ be an extension of the GSMP vector to one with hurdle geometric N denoted as HGSMP. Then, the PDF of HGSMP is as follows:

$$f_{\alpha, \lambda, p, \pi}(x, y, n) = \begin{cases} \pi \lambda \left(\frac{1}{1 + \alpha \lambda x} \right), & \text{if } n = 1 \\ \left(\frac{(1 - \pi) \lambda}{1 + \alpha \lambda x} \right) \left[\prod_{j=0}^{n-1} (j \alpha + 1) \right] p (1 - p)^{n-2}, & \text{if } n \geq 2. \end{cases} \quad (5.3)$$

Let $(X_1, Y_1, N_1), \dots, (X_l, Y_l, N_l)$ be a sample of l IID observations from the HGSMMP distribution. It is convenient to split the observations into two sets: one with events of duration one, and the other of events with duration greater than one. Let ℓ_1 be the number of observations with $n_i = 1$. Further, let ℓ_2 being the number of observations where $n_i \geq 2$ so that $\ell_1 + \ell_2 = \ell$. Then the likelihood function is as follows:

$$\begin{aligned}
L(\alpha, \lambda, p, \pi; \vec{x}, \vec{y}, \vec{n}) &= \pi^{\ell_1} \lambda^{\ell_1} \prod_{i=1}^{\ell_1} \left(\frac{1}{1 + \alpha \lambda x_i} \right) \cdot (1 - \pi)^{\ell_2} \lambda^{\ell_2} \prod_{i=1}^{\ell_2} \left(\frac{1}{1 + \alpha \lambda x_i} \right) \\
&\quad \cdot \prod_{i=1}^{\ell_2} \left[\prod_{j=0}^{n_i-1} (j\alpha + 1) \right] p^{\ell_2} (1 - p)^{\sum_{i=1}^{\ell_2} n_i - 2\ell_2} \\
&= \lambda^\ell \prod_{i=1}^{\ell} \left(\frac{1}{1 + \alpha \lambda x_i} \right) \cdot \pi^{\ell_1} \cdot (1 - \pi)^{\ell_2} \prod_{i=1}^{\ell_2} \left[\prod_{j=0}^{n_i-1} (j\alpha + 1) \right] \\
&\quad \cdot p^{\ell_2} (1 - p)^{\sum_{i=1}^{\ell} n_i - (2\ell_2 + \ell_1)}
\end{aligned}$$

Next, the log-likelihood function is:

$$\begin{aligned}
\ell(\alpha, \lambda, p, \pi; \vec{x}, \vec{y}, \vec{n}) &= \underbrace{\ell \ln(\lambda) + \sum_{i=1}^{\ell} \left[\sum_{j=0}^{n_i-1} \ln(j\alpha + 1) \right] - \sum_{i=1}^{\ell} \ln(1 - \alpha \lambda x_i)}_{\text{Function of } \alpha \text{ \& } \lambda} \\
&\quad + \underbrace{\ell_1 \ln(\pi) + \ell_2 \ln(1 - \pi)}_{\text{Function of } \pi} \\
&\quad + \underbrace{\ell_2 \ln(p) + \left(\left[\sum_{i=1}^{\ell} n_i \right] - (\ell_2 \cdot 2 + \ell_1) \right) \ln(1 - p)}_{\text{Function of } p}
\end{aligned}$$

Thus, the log-likelihood function is a sum of three separate terms where the only “new” term is the middle term which is a function of π . One may use the same link function for π covariates as that used for the p covariates with different set of covariates for π and p . Since the log-likelihood function is a sum of three functions we can maximize each term separately, adding maximization for the term corresponding to π . The maximization procedures would be very similar to those for GSMP-VGLM and are beyond the scope of this dissertation.

Chapter 6

Python Package

In this chapter, we review the scientific computing library, `gsmp_vglm`, we created in Python to support fitting this model in practice. We detail the underlying libraries used to construct this library, the overall structure of the library, and the functions of each of the core components of the library. These functions allow users to generate data from a \mathcal{VGSMP} and \mathcal{VTETLG} distribution, fit the parameters and also supports goodness of fit procedures for the \mathcal{VGSMP} Distribution. Though not shown here, example notebooks are included in our Git repo to demonstrate how to fit this model in practice.

6.1 Python and the Scientific Computing Ecosystem

Since our equations don't yield closed-form solutions, to facilitate estimating parameters in practice, we created a scientific library in Python. Python

is versatile, high-level programming language that has become one of the most popular languages used in scientific computing, statistical analysis, data science and machine learning. We selected Python to write our library, due to its popularity, simple syntax and vast ecosystem of libraries. Our package, `gsmv_vglm`, leverages several well-maintained and popular scientific libraries, namely:

- **NumPy**: Efficient numerical and linear algebra operations and array handling.
- **SciPy**: Advanced mathematical functions, distributions and optimization tools.
- **Matplotlib**: Visualization and plotting capabilities.
- **Pandas**: Data manipulation and analysis; creates objects similar to the R-language `data.frame` object.
- **Scikit-Optimize**: Bayesian optimization algorithms for hyperparameter tuning.

By building our library on top of this ecosystem, we have produced a scalable and computationally efficient system that integrates well within other

standard Python libraries. By building on libraries that are well-maintained and actively developed, we ensure that our package will be easy to maintain for future users and development.

6.2 High-Level Overview of the GSMP-VGLM Package

The current instantiation of our package is very straightforward and intuitive:

```

gsmv_vglm
├── gsmv_vglm
│   ├── __init__.py
│   ├── bayes_alpha_opt.py
│   ├── data_generation.py
│   ├── grad_desc_sem_functions.py
│   └── goodness_of_fit.py
├── Example Notebooks
│   ├── Ex - Estimating alpha with GPO.ipynb
│   ├── Ex - Estimating eta_vec w alpha known.ipynb
│   └── Ex - Goodness of fit w alpha known.ipynb
├── setup.py
└── requirements.txt

```

The library contains some generic matter for a Python package (e.g. ‘setup.py’), a folder of example iPython Notebooks and the core library which consists of three main files which we discuss in details below. But a high-level summary is:

- `data_generation.py` - Creates data from the TETLG & GSMP distributions where the specific values of α , β and η are specified by the

user.

- `grad_desc_sem_functions.py` - Given data, there are methods to estimate $\boldsymbol{\eta}$ and $\boldsymbol{\beta}$ for GSMP and TETLG distributions using Gradient Descent.
- `bayes_alpha_opt.py` - In the GSMP distribution, α needs to be estimated using other methods and this file provides the means to do so using Bayesian Optimization.
- `goodness_of_fit.py` - Create QQ-plots for goodness-of-fit tests, as well as Kolmogorov-Smirnov statistics for the observed data.

6.3 `data_generation.py`

This module implements a flexible and extensible framework for simulating synthetic event-based data. This is essential for several use cases, including:

- Evaluating estimation procedures
- Establishing properties of the values of $v(\alpha)$
- Conducting broader simulation studies.

In all of these cases, knowing the mechanisms that generate the data give the user the ability to see what happens when the true values of α , $\boldsymbol{\eta}$ and $\boldsymbol{\beta}$

are known.

The module adheres to the principles of object-oriented programming, utilizing class inheritance to encapsulate model-specific behavior while providing a unified interface. Our base class consists of a single method: `make_data`. We can construct any number of subclasses that inherit from this base class, which makes this extensible in the future as other models are added to the research. At present there are two subclasses:

1. `TETLG_GLMData`
2. `GSMP_GLMData`

Each subclass encapsulates the parameters and stochastic mechanisms specific to the corresponding model while reusing common logic, such as covariate matrix generation. The common applications in both subclasses are:

- **Covariate Generation** - Both `TETLG_GLMData` and `GSMP_GLMData` implement `make_W()` & `make_Z()` methods. This allows for covariates to be drawn from several probability distributions: Uniform, Gaussian and Exponential. An intercept term can also be added into the models.
- **Model Parameterization** - The covariate matrices are combined with

the parameter vectors to define $\boldsymbol{\eta}$ and $\boldsymbol{\beta}$ for the event durations and magnitudes: the vectors \mathbf{p} and $\boldsymbol{\lambda}$, respectively.

- **Data Generation** - Given \mathbf{p} , $\boldsymbol{\lambda}$ and the user-provided parameter α (for GSMP), then simulated event triples can be created.

1. First, ℓ Geometric observations are drawn using parameter vector \mathbf{p} to create the N array.
2. Next, for each $N_i \in N$, N_i Exponential variables are drawn using $\lambda_i \in \boldsymbol{\lambda}$.
 - If this is GSMP-GLM data, then a single Gamma($1/\alpha$, $1/\alpha$) is drawn and the vector of N_i exponential random variables is divided by the mixing parameter, α , to create an N_i -dimensional Pareto Type II random vector.
3. Lastly, the sum and maximum of the N_i Exponential/Pareto observations is found and used to create X and Y . X, Y, N, W, Z are returned to the user.

6.4 `grad_desc_sem_functions.py`

The second module provides a structured framework for estimating parameters in TETLG and GSMP models using gradient descent and matrix-based solvers. Like the previous module, this code is designed with extensibility in mind through the use of object-oriented programming and class inheritance.

The base class, `BaseEstimator`, is an abstract class that consists of two principal methods: `fit()` and `predict()`. These two methods are included as these are core requirements in order to be able to work within the popular Python library, `scikit-learn`. At present, we do not actually perform any “`predict()`” functions, though this could easily return the expected values of (X, Y, N) given some covariate matrices and estimated parameters. Therefore, we will spend most of our time discussing the “`fit()`” method.

Once again, there are two subclasses:

1. `TETLG_GLM`
2. `GSMP_GLM`

Though the specific mechanics of the subclasses is unique to the distributions

being used (i.e. the gradients depend on the respective likelihood functions of each distribution), there are generic commonalities in both methods:

- Estimation of β and η parameters are disentangled. As the likelihood equations are linear combinations of functions of η and β , the estimation can safely be linearized.
- For each parameter, we do the following procedures:
 1. Initialize estimates of either η or β using the “Normal Equations” approach discussed in Section 3.2.2. These represent the starting points when the gradient descent starts
 2. Defining the gradient of the respective parameter. This calculation uses the current estimates of the parameter along with observed data (and a provided value of α in the GSMP case).
 3. The actual gradient descent function for the respective parameter. This takes in the initialized parameter vector value and then takes a pre-defined number of steps with an initial length that decays with each step.
 4. After the gradient descent process is completed, the estimated parameter vector is returned.

- This process is completed for $\boldsymbol{\eta}$ and $\boldsymbol{\beta}$ and the final parameter vectors are returned.

6.5 `bayes_alpha_opt.py`

This module is specific to the GSMP-GLM case, as it is used to determine an estimated value of α . For this reason, this module is very specific and will import the “GSMP_GLM” directly. The overall implementation is quite straightforward and relies heavily on using the `scikit-optimize` library. To use this package, all we need to do is to be able to

1. Have an “Objective function” to optimize, which is used to tell the model how well it is performing. In this case, it is the log-likelihood function of $g(\alpha, \boldsymbol{\eta})$, where α is to be provided by the Bayesian Optimization process.
2. Given a candidate value of α , calculate $\boldsymbol{\eta}(\alpha)$. This is achieved using the `GSMP_GLM` module defined earlier in Section 6.4.
3. Implement the required data, objective function and estimation procedure inside of the `scikit-optimize` module.

With these functions, we are able to implement the Bayesian optimization

process to determine an optimal alpha. All a user needs to do is to provide the appropriate covariate matrices, (X, Y, N) triples, and bounds to explore for α . More advanced users could manipulate the `scikit-optimize` procedure more (e.g. specify the Gaussian process kernels and hyperparameters) but the major implementation runs on defaults and has been highly performative even when we attempted to fine-tune the results when the answers were known.

6.6 `data_generation.py`

This module creates several Goodness-of-Fit outputs based on the observed data and estimated/known GSMP parameters. In particular, the module assumes that you have triplets of $\{(X_1, Y_1, N_1), \dots, (X_\ell, Y_\ell, N_\ell)\}$, \mathbf{Z} , \mathbf{W} , estimated parameter vectors $\boldsymbol{\eta}$ & $\boldsymbol{\beta}$ and either known or estimated α . Given this, we replicate the work shown in Chapter 5. The class is instantiated with these values. The primary method is `run_analysis()` which

- Uses $\boldsymbol{\eta}, \boldsymbol{\beta}, \mathbf{W}, \mathbf{Z}$ to construct $\boldsymbol{\lambda}, \mathbf{p}$
- Using the CDF of X and a Monte Carlo approximation of Y, find F_X^{-1} and F_Y^{-1}
- Using these values, create QQ-Plots against $\mathcal{U}(0, 1)$ and returning the

test statistics and p-values of the Kolmogorov-Smirnov tests.

This provides visual and statistical means to compare the quality of the fit of X and Y for the \mathcal{VGSMP} models.

6.7 Conclusion

This chapter presented a comprehensive overview of the Python package `gsmp_vglm`, developed to support simulation and estimation procedures for the GSMP and TETLG models introduced in earlier chapters. Motivated by the absence of closed-form solutions for the relevant parameter estimation problems, we have implemented a robust, modular, and extensible software suite within the scientific computing ecosystem of Python.

The package is constructed using modern software design principles, particularly object-oriented programming and class inheritance, to ensure clarity, reusability, and extensibility. The core modules—`data_generation.py`, `grad_desc_sem_functions.py`, and `bayes_alpha_opt.py`—encapsulate distinct aspects of the modeling process:

- Synthetic data generation for controlled evaluation of estimation techniques,

- Parameter estimation via gradient-based optimization procedures,
- Hyperparameter tuning (specifically, α) using Bayesian optimization.

Through well-documented and structured code, this package enables both novice and advanced users to engage with complex event-based models with minimal overhead. Furthermore, its modular design anticipates future enhancements, including support for alternative distributions, additional model families, and deeper integration with machine learning workflows. One such simple example is using the library to bootstrap parameter estimates on actual data to gauge parameter uncertainty.

This computational implementation not only serves as a practical engine for empirical studies and simulation experiments but also reinforces the theoretical framework developed earlier.

Chapter 7

Summary

The research in this dissertation has presented a comprehensive theoretical and computational framework that extends the existing literature to model stochastic events by applying Vector-GLM's with Pareto marginals. Doing so addressed some gaps in previous research, namely static parameters for all observations and/or the lack of heavy-tailed marginals for the values above a threshold. In this chapter, we summarize the key findings of this work and provide future research directions.

7.1 Main Results

The main results of this work are:

- A new vector generalized linear model for stochastic events characterized by the \mathcal{GSMP} distribution. The VGLM links parameters to

covariates. We have results on existence for all parameters's estimators as well as uniqueness for two parameter estimators' vectors $(\boldsymbol{\eta}, \boldsymbol{\beta})$. We also demonstrate simulation studies that strongly imply uniqueness for the MLE of α . We concluded multiple simulation studies that demonstrate the robustness and properties of the estimator in various cases. We also demonstrate goodness-of-fit methods for the estimated maxima and magnitudes.

- A statistical library to fit data with a *VGSMP* or *VTETLG* as well as generating observations from these distributions. This library also implemented a solver which showed itself to be quite robust to some of the issues that can lead to divergences when using other standard solver methods that ultimately depend on Newtonian solvers. We show a theoretical result to find optimal parameters, illustrate how this becomes biased when the data becomes heavy-tailed, but how this can still be used to improve convergence to the best estimates in practice. Since one parameter does not have an MLE, we discuss two potential solutions commonly used in Machine Learning and implemented Bayesian Optimization in the library and demonstrate the efficacy of this solver.

- We briefly discuss extension of the GSMP-VGLM model to one where duration has the Hurdle Geometric distribution. This extension allows better fit to events with a large frequency of duration = 1.
- We illustrate the utility of our new model with fit to precipitation data at the South Lake Tahoe and Reno airports and demonstrate how well the model fits the underlying data.

7.2 Future Work Directions

1. **Covariate-Dependent α :** Developing theoretical and computational frameworks for allowing the tail parameter to vary with covariates would significantly enhance model flexibility and more accurately reflect the data-generating process.
2. **Full Development of Hurdle Models:** Many events showed duration= 1. Full development of the estimation for the hurdle-geometric duration in the GSMP-VGLM model and numerical implementation of this model would be a good future project.
3. **Incorporating Regularization of Covariates:** Our GSMP-VGLM framework allows for any number of covariates to be employed, but

there are no mechanisms to control for overfitting natively within the model context as in other standard regularized regression models (e.g. LASSO, Ridge Regression). Implementing an intrinsic means to apply regularization within the models will be very valuable, especially as the amount of available data increases.

4. **Machine Learning Integration:** Exploring the integration of the GSMP-VGLM framework with modern machine learning approaches, could yield better model performance, where non-linear relationships may be captured better.
5. **Developing a Framework for Model Comparison:** As the literature in this field grows, we are being inundated with numerous models that could be fit to the same kind of data. The question arises: "Which model should be used?" We believe that there is a great deal of research possible in this domain and in particular think that methods leveraging Machine Learning approaches focusing on prediction using cross-validation can provide valuable data that will make optimal model selection that generalize well to new data.
6. **Bayesian Framework:** Development of fully Bayesian estimation pro-

cedures for \mathcal{VGSMP} and \mathcal{VTETLG} would enable proper uncertainty quantification and model averaging, and possibly model comparisons.

7. **Exploring Covariate Construction:** Through the application of the models to applied data, we believe that there is a promising area of research in how covariates can be constructed in ways that can reduce estimator uncertainty, similar to the field of Variance Reduction.

Chapter 8

Bibliography

- [1] Arendarczyk, M., Kozubowski, T. J., Panorska, A. K. (2018a).
A bivariate distribution with Lomax and geometric margins.
Journal of the Korean Statistical Society, 47(4), 405–422.
<https://doi.org/10.1016/j.jkss.2018.04.006>.
- [2] Arendarczyk, M., Kozubowski, T. J., Panorska, A. K. (2018b) The Joint
Distribution of the Sum and the Maximum of Dependent Pareto Risks.
Journal of Multivariate Analysis, 167, 136 - 156.
- [3] Arnold, B.C. (1983). Pareto Distributions, International Co-operative
Publishing House, Burtonsville, MD.
- [4] Biondi, F., Kozubowski, T. J., and Panorska, A. K. (2005). A new model
for quantifying climate episodes. *International Journal of Climatology*,

- 25, 1253-1264.
- [5] Biondi, F., Kozubowski, T.J., Panorska, A.K., and Saito, L. (2007). A new stochastic model of episode peak and duration for eco-hydro-climatic applications. *Ecological Modelling*, 211(3), 383-395.
- [6] Cavanaugh, N., Gershunov, A., Panorska, A. K., Kozubowski, T. J. (2015). The probability distribution of intense daily precipitation. *Geophysical Research Letters*, 42(5), 1560–1567.
- [7] EPA. (2012). Revised Air Quality Standards for Particle Pollution and Updates to the Air Quality Index(AQI). U.S. Environmental Protection Agency. https://www.epa.gov/sites/default/files/2016-04/documents/2012_aqi_factsheet.pdf.
- [8] FEMA. (n.d.). Flood Maps. U.S. Department of Homeland Security. <https://www.fema.gov/flood-maps>
- [9] Kozubowski, T. J. and Panorska, A. K. (2005). A mixed bivariate distribution with exponential and geometric marginals. *Journal of Statistical Planning and Inference*, 134, 501-520. <https://doi.org/10.1016/j.jspi.2004.04.010>.

- [10] Kozubowski, T. J., Panorska, A. K., Qeadan, F., with Gershunov, A. and Rominger, D. (2009). Testing exponentiality versus Pareto distribution via likelihood ratio. *Communications in Statistics: Simulation and Computation*, 38 (1), 118-139.
- [11] Kozubowski, T. J., Panorska, A. K. (2007). A mixed bivariate distribution connected with geometric maxima of exponential variables. *Communications in Statistics – Theory and Methods*, 37, 2903–2923.
- [12] Kozubowski, T. J., Panorska, A. K., Qeadan, F. (2011). A new multivariate model involving geometric sums and maxima of exponentials. *Journal of Statistical Planning and Inference*, 141, 2353 – 2367.
- [13] Luerken, E. (2025). *gsmp_vglm: Generalized Semi-Markov Process for VGLMs* [Python library]. GitHub. https://github.com/Schlagmichtot/gsmp_vglm.
- [14] Menne, M.J., Durre, I., Vose, R.S., Gleason, B.E., and Houston, T.G. (2012). An Overview of the Global Historical Climatology Network-Daily Database, *Journal of Atmospheric and Oceanic Technology* 29(7), 897-910, doi: <https://doi.org/10.1175/JTECH-D-11-00103.1>

- [15] NOAA. (2013). NOAA Atlas 14, Precipitation-Frequency Atlas of the United States, Volume 8 Version 2.0: Midwestern States. National Oceanic and Atmospheric Administration. https://www.weather.gov/media/owp/oh/hdsc/docs/Atlas14_Volume8.pdf.
- [16] NOAA: Storm Events Database, <https://www.ncdc.noaa.gov/stormevents/>, accessed Nov. 2024
- [17] Qeadan, F., Kozubowski, T. J., Panorska, A. K. (2012). The joint distribution of the sum and the maximum of iid exponential random variables. *Communications in Statistics – Theory and Methods*, 41(3), 544–569.
- [18] Rasmussen, C. E., Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press. ISBN 026218253X. <https://www.gaussianprocess.org/gpml/>
- [19] Saito, L., Biondi, F., Salas, J. D., Panorska, A. K., and Kozubowski, T. J. (2008) A watershed modeling approach to stream flow reconstruction from tree-ring records. *Environmental Research Letters* 3(2). 024006-. <https://doi.org/10.1088/1748-9326/3/2/024006>.

- [20] Weyant, A. and Gershunov, A. and Panorska, A. K. and Kozubowski, T. J. and Kalansky, J. (2025) A Holistic Stochastic Model for Precipitation Events. *Nature: Scientific Reports* <https://www.nature.com/articles/s41598-024-77031-3>
- [21] Yee, T. W. (2015). *Vector Generalized Linear and Additive Models: With an Implementation in R*. Springer Series in Statistics. Springer New York. DOI: 10.1007/978-1-4939-2818-7.
- [22] Zuniga, F., Kozubowski, T. J., Panorska, A. K. (2021) A generalized linear model for multivariate events. *Journal of Computational and Applied Mathematics*, 398, 113655.
- [23] Zuniga, F., Kozubowski, T. J., Panorska, A. K. (2021) A new trivariate model for stochastic episodes, *Journal of Statistical Distributions and Applications*, 8, <https://doi.org/10.1186/s40488-021-00114-3>.
- [24] Yahoo Finance. (2024). *Bitcoin (BTC) stock historical data*. <https://finance.yahoo.com/quote/BTC/history> (Accessed: Jan 3, 2025).

- [25] Yahoo Finance. (2024). *Tesla, Inc. (TSLA) stock historical data*.
<https://finance.yahoo.com/quote/TSLA/history> (Accessed: Jan 3, 2025).
- [26] Yee, T. W., Wild, C. J. (1996). Vector generalized additive models.
Journal of the Royal Statistical Society, Series B. 58(3), 481–493.
- [27] Yee, T. W. (2015). *Vector Generalized Linear and Additive Models: With an Implementation in R*. Springer, New York. ISBN 978-1-4939-2817-0.