Advancements on the Interface of Computer Experiments and Survival Analysis

Yueyao Wang

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Statistics

Yili Hong, Chair
Xinwei Deng
Inyoung Kim
Laura Freeman

June 24, 2022
Blacksburg, Virginia

Keywords: Computer Experiment; Degradation Index; Survival Analysis; System Health Monitoring; Variational Inference

Advancements on the Interface of Computer Experiments and Survival Analysis

Yueyao Wang

Abstract

Design and analysis of the computer experiment is an area focusing on efficient data collection (e.g., space-filling designs), surrogate modeling (e.g., Gaussian process models), and uncertainty quantification. Survival analysis focuses on modeling the period of time until a certain event happens. Data collection, prediction, and uncertainty quantification are also fundamental in survival models. In this dissertation, the proposed methods are motivated by a wide range of real world applications, including high-performance computing (HPC) variability data, jet engine reliability data, Titan GPU lifetime data and pine tree survival data. This dissertation is to explore interfaces on computer experiments and survival analysis with the above applications.

Chapter 1 provides a general introduction of computer experiments and survival analysis. Chapter 2 focuses on the HPC variability management application. We investigate the applicability of space-filling designs and statistical surrogates in the HPC variability management setting, in terms of design efficiency, prediction accuracy, and scalability. A comprehensive comparison for the design strategies and predictive methods is conducted to study the combinations' performance in prediction accuracy.

Chapter 3 focuses on the reliability prediction application. With the availability of multi-channel sensor data, a single degradation index is needed to be compatible with most existing models. We propose a flexible framework with multi-sensory data to model the nonlinear relationship between sensors and the degradation process. We also involve the automatic variable selection to exclude sensors that have no effect on the underlying degradation process.

Chapter 4 investigates inference approaches for spatial survival analysis under the Bayesian framework. The Markov chain Monte Carlo (MCMC) approaches and variational inferences performance are studied for two survival models, the cumulative exposure model and the proportional hazard (PH) model. The Titan GPU data and pine tree survival data are used to illustrate the capability of variational inference on spatial survival models. Chapter 5 provides some general conclusions.

Advancements on the Interface of Computer Experiments and Survival Analysis

Yueyao Wang

General Audience Abstract

This dissertation focus on three projects related to computer experiments and survival analysis. Design and analysis of the computer experiment is an area focusing on efficient data collection, building predictive models, and uncertainty quantification. Survival analysis focuses on modeling the period of time until a certain event happens. Data collection, prediction, and uncertainty quantification are also fundamental in survival models. Thus, this dissertation aims to explore interfaces between computer experiments and survival analysis with real world applications.

High performance computing systems aggregate a large number of computers to achieve high computing speed. The first project investigates the applicability of space-filling designs and statistical predictive models in the HPC variability management setting, in terms of design efficiency, prediction accuracy, and scalability. A comprehensive comparison of the design strategies and predictive methods is conducted to study the combinations' performance in prediction accuracy.

The second project focuses on building a degradation index that describes the product's underlying degradation process. With the availability of multi-channel sensor data, a single degradation index is needed to be compatible with most existing models. We propose a flexible framework with multi-sensory data to model the nonlinear relationship between sensors and the degradation process. We also involve the automatic variable selection to exclude sensors that have no effect on the underlying degradation process.

The spatial survival data are often observed when the survival data are collected over a spatial region. The third project studies inference approaches for spatial survival analysis under the Bayesian framework. The commonly used inference method, Markov chain Monte Carlo (MCMC) approach and the approximate inference approach, variational inference's performance are studied for two survival models. The Titan GPU data and pine tree survival data are used to illustrate the capability of variational inference on spatial survival models.

To my family, my love, and Otis.

Acknowledgments

First of all, I would like to express my deepest gratitude to my advisors, Dr. Yili Hong and Dr. Xinwei Deng. Dr. Hong is always very patient to teach me and guide me for my research. He can always find shining points in every student and I really appreciate his encouragements and praises that made me confident. I also want to express my great thanks to Dr. Deng for his generous help during the pandemic. He not only provides me precious advice and suggestions on my research but also encourages students to live healthy and be happy during this special period. I can not accomplish this without their guidance, encouragement, and tolerance.

I would like to thank my committee members, Dr. Inyoung Kim and Dr. Laura Freeman for their inspiring suggestions and valuable advice. I also want to express my thanks to Dr. Xiaowei Wu, Dr. Laura Sands and Dr. Kirk Cameron for the amazing collaboration opportunities. Furthermore, I greatly appreciate the National Science Foundation (NSF) funding and the Advanced Research Computing at Virginia Tech for providing computational resources. I am so fortunate to study with all the faculty members, collaborators, and colleagues at Virginia Tech. Their guidance, kindness, and supports made my Ph.D. study a fruitful and wonderful journey.

I also want to appreciate all my group members for their generous help, Dr. Zhongnan Jin, Dr. Li Xu, Dr. Ichen, Lee, Dr. Hung-Ping Tung, Jie Min, Zhengzhi Lin, Xinyi Song, Zhihao Hu, Yanran Wei, Jiayi Lian and Qing Guo. I am grateful for all my friends and roommates during my life at Blacksburg. I am especially thankful to my dear friend Beibei. I can not go so far without anyone of you.

I also want to thank Otis, my little fur baby, who always supports and accompanies me by stepping, sitting, and lying down on my computer keyboards. I know he values my study progress because not even once he presses the "delete" key on my computer. Besides, I would like to thank my boyfriend, Shengchao, an overfitting version of me in this world. He is always so witty and intelligent. He accompanies me along this journey and always understands me, encourages me, and supports me. Last but not least, I would like to express my appreciation for my mother. She provides me all the opportunities that she ever had but had to give up. She made all of this possible.

Contents

List of Figures

List of Tables

Chapter 1 General Introduction

Analysis of computer experiments has been employed under many objectives, such as exploring the relationship between the response and input factors, predicting the response of interest, and quantifying the uncertainty in the computer simulation system. To achieve these objectives, computer experiments often interact with predictive models. Survival analysis covers a branch of statistical models that focus on various topics that are related to observations' survival. Examples of these topics include understanding the survival rates of observations over time, characterizing factors that contribute to the failures of observations, and predicting the failure status of products at certain times. To address these problems, survival models often involve estimation, prediction as well as uncertainty quantification. Both fields have a wide range of applications in real world. Thus, it is interesting to investigate the intersection of the two areas through engineering applications. Motivated by the high-performance computing (HPC) variability prediction, reliability prediction applications, Titan GPU lifetime data, and pine tree survival analysis, this dissertation aims to explore three interesting research projects regarding computer experiments and survival analysis interfaces.

For the HPC variability project, computer scientists are interested in accurate HPC variability predictions using statistical prediction methods, while computer experiments provide an efficient way to collect data that the predictive models can be built based on. In the reliability prediction project, engineers are interested in building an accurate degradation prediction model based on complex sensor data with automatic feature selection for the sensor channels. The capability of variational inference on the spatial survival model is studied in the third project. In the following, a more detailed introduction to the three projects is provided, highlighting the interaction between computer experiments and survival analysis.

1.1 Computer Experiments

Many physical and engineering processes are difficult to study by classical experiments. Advances in computing power allow computer simulations to model these complicated processes. Experiments used to study such computer simulations become known as computer experiments (Santner et al., 2003). Computer experiments have wide applications in computational physics and other similar disciplines. In this dissertation, the application of computer experiments in the HPC variability management is of interest as well as its interfaces with predictive models, such as the Gaussian process.

1.1.1 Space-filling Designs

There are many types of designs for computer experiments such as space-filling designs (SFDs) and sequential designs. When little information is known about the phenomenon to be studied, SFDs are widely used for experimental designs. This dissertation focuses on SFDs. As the name suggests, SFDs spread out design points evenly in the experiment region in order to gather thorough information from the whole experiment region. SFDs can be constructed based on sampling methods, such as the uniform design (Fang et al., 2000) and Latin hypercube designs (McKay et al., 1979). SFDs can also be constructed based on the distance measure, such as the maximin and minimax designs (Johnson et al., 1990). There are also several designs constructed based on the variants or combinations of the aforementioned designs, such as the maximin Latin hypercube design (Morris and Mitchell, 1995) and the maximum projection design (Joseph et al., 2015).

1.1.2 Surrogate Modeling

Surrogate modeling is an important area in computer experiments. The computer simulation is sophisticated and it is usually time-consuming to conduct a single simulation. When an outcome of interest cannot be easily obtained, a surrogate model is often used to approximate the behavior of the simulation model. There are many popular surrogate modeling approaches,

such as response surface models, kriging methods, kernel estimation, and neural networks. Response surface methodology (Box and Wilson, 1951), is a traditional statistical technique for modeling the response variables given input variables. Gaussian process (GP) regression (e.g., Sacks et al., 1989, Currin et al., 1991) is also commonly used to build predictive models based on computer experiments. Some numerical techniques can also be used as surrogates, such as linear Shepard model (Thacker et al., 2010) and Delaunay triangulation model (Chang et al., 2020).

1.1.3 High Performance Computing Variability

One motivating application of investigating the computer experiments performance in data collection efficiency and building predictive models is high performance computing (HPC) variability data. HPC system aggregates computing powers to process complex tasks at high speed. The advancements in modern technologies and scientific areas result in the increasing of computing scale and complexity, which make HPC important. However, many researches have observed performance variability, which brings challenges in the research of HPC systems (Giampapa et al., 2010, Akkan et al., 2012, Cameron et al., 2019). High variability in HPC systems can lead to unstable system performance and potentially high energy costs. Therefore, variability management is crucial for system performance optimization. To understand the performance variability in HPC, data collection and building prediction models based on the collected data are important steps in HPC variability modeling. The performance variability is affected by many complicated interactions of factors in the system. In this study, I focus on input/output (I/O) performance variability. We are interested in the relationship between I/O performance variability and four system factors: CPU frequency, file size, record size, and the number of I/O threads. In our experiment, the file size needs to be larger than or equal to the record size.

To collect data for this application, all combinations of the four system factors are enumerated. At each system configuration, the IOzone benchmark (2016) is run multiple times and the throughputs are gathered as an HPC performance measurement. The configurations

are collected under 6 I/O operation modes. In total, there are 2658 configurations and each configuration has 300 replications to capture the performance variability. Because the data collection procedure is time-consuming and motivated by the need for efficient data collection, Chapter 2 provides a thorough study about the interactions of design strategies and prediction models in the HPC setting.

1.2 Survival Analysis

1.2.1 Reliability Prediction

Reliability analysis uses survival analysis techniques in engineering problems. In reliability applications, designing qualified products and maintaining quality over time are the primary objective. There are various data provide us information about the system reliability, including failure time data, recurrent events data, and degradation data. Motivated by the jet engine data, Chapter 3 proposed a statistical model that characterizes product degradation paths.

Degradation data have been widely used for reliability and system health assessment (e.g., Meeker and Escobar, 1998). There are many examples of products and systems provide degradation data, for example, the loss of light output from a light-emitting diode (LED) array, the power output decrease of photovoltaic (PV) arrays, and the vibration from a worn bearing in a wind turbine. The data type is typically a repeated measurement of the degradation index (e.g., the depth of tire tread). The degradation data usually has a failure threshold, the product fails if its degradation index exceeds the threshold. The degradation-induced failure is considered as a soft failure.

Most existing research on degradation data modeling assumes that the degradation index for a product or system is well defined. Measurements can be made for such degradation index. Modern sensor technology allows one to collect multi-channel sensor data that are related to an underlying degradation process. For example, in the jet engine simulation data, there are 16 multi-channel sensors after removing those channels with constant signals. However, in multi-channel sensory data, any single channel may not be sufficient to represent the underlying

degradation process. Therefore, it is important to construct an appropriate degradation index with multi-channel sensory data in degradation modeling. Since not all sensory information contributes to the underlying degradation process, variable selection procedure is desirable to appropriately construct the degradation index. Chapter 3 introduces a flexible framework to build a degradation index using multi-channel sensory data with automatic variable selection.

1.2.2 Spatial Survival Models under Bayesian Framework

Spatial survival models receive great attention in recent decades. Generally, spatial effects are represented by hidden location's effect on observations' survival time. Regarding model inferences, the Bayesian framework is one popular approach in spatial survival models. In literature, many applications in biomedical and social science areas involve spatial location effect in survival time. Examples include the leukemia survival (Henderson et al., 2002), infant mortality (Banerjee et al., 2003), political event processes (Darmofal, 2009), breast cancer studies (Zhao et al., 2009), and Titan GPU lifetime (Min and Hong, 2022).

The Bayesian framework is one popular approach that conducts inference for survival models based on posterior samples. Due to high dimensionality of spatial survival models, Markov Chain Monte Carlo (MCMC) techniques can be used to draw samples from the posterior. The Hamiltonian Monte Carlo (HMC) is often used to obtain the ground truth from the posterior. However, HMC can suffer from a low mixing rate. Variational methods (Blei et al., 2017) are often used as alternative methods to obtain an approximation distribution for the posterior.

1.2.3 Variational Inference

Variational inference (VI) is an optimization-based approach to approximate intractable distributions (Blei et al., 2017). VI approximates the complicated posterior with a simple variational distribution by minimizing the metrics that quantify the closeness between two distributions. The Kullback-Leibler (KL) divergence and the $\alpha-$divergence are two common metrics to measure distributions distance. Minimizing the KL divergence between the posterior and a variational distribution that belong to a mean-field family is a standard VI procedure in

many studies. But the parameter independence assumption of the mean-field family limits the flexibility of variational distribution, and the KL divergence can lead to underestimation of the posterior variance. Structured VI that allows dependencies between parameters is one direction to introduce flexibility to variational distributions. Another direction is to use different divergence metrics, such as the α-divergence, to allow flexible behavior of the variational distribution. In Chapter 4, the performance of KL divergence, $\alpha-$divergence, and MCMC approach is compared for spatial survival models under the Bayesian framework.

1.3 Overview

The rest of this dissertation is organized as follows. Chapter 2 presents a comprehensive comparison for the combination of design strategies and approximation methods under the setting of HPC variability management. The design strategies are tailored so that they are suitable to the HPC constraint and the approximation methods. Chapter 2 is mainly based on Wang et al. (2022c). Chapter 3 proposes a flexible model to build the degradation index with multi-sensory data. The variable selection is also incorporated to exclude covariates that do not contribute to the degradation process. Chapter 3 is mainly based on Wang et al. (2022b). Chapter 4 studies the performance of VI and MCMC when conducting model inference of spatial survival models. Two survival models, cumulative exposure model and proportional hazard model are used to illustrate the capability of VI. Chapter 4 is mainly based on Wang et al. (2022a).

During my Ph.D. study, I also worked on several other projects besides my dissertation research. I co-authored one journal paper about the HPC prediction model (Xu et al., 2020), one journal paper about genome-wide association study (Wang et al., 2020), one journal paper about Poisson multinomial distribution (Lin et al., 2022), one review paper about reliability of artificial intelligence systems (Hong et al., 2021), two conference papers (Lux et al., 2020b, Lux et al., 2020a) in computer science, and one book chapter (Wang et al., 2021) about modern reliability analysis.

Bibliography

H. Akkan, M. Lang, and L. M. Liebrock. Stepping towards noiseless Linux environment. In *Ross' 12 Proceedings of the 2nd International Workshop on Runtime and Operating Systems for Supercomputers*, number 7, 2012. doi:10.1145/2318916.2318925.

S. Banerjee, M. M. Wall, and B. P. Carlin. Frailty modeling for spatially correlated survival data, with application to infant mortality in Minnesota. *Biostatistics*, 4(1):123–142, 2003.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017. doi: 10.1080/01621459.2017.1285773.

G. E. Box and K. B. Wilson. On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 13(1):1–38, 1951.

K. W. Cameron, A. Anwar, Y. Cheng, L. Xu, B. Li, U. Ananth, J. Bernard, C. Jearls, T. Lux, Y. Hong, L. T. Waston, and A. R. Butt. MOANA: Modeling and analyzing I/O variability in parallel system experimental design. *IEEE Transactions on Parallel and Distributed Systems*, 30(8):1843–1856, 2019.

T. H. Chang, L. T. Watson, T. C. Lux, A. R. Butt, K. W. Cameron, and Y. Hong. Algorithm 1012: Delaunaysparse: Interpolation via a sparse subset of the Delaunay triangulation in medium to high dimensions. *ACM Transactions on Mathematical Software (TOMS)*, 46(4): 1–20, 2020.

C. Currin, T. Mitchell, M. Morris, and D. Ylvisaker. Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *Journal of the American Statistical Association*, 86(416):953–963, 1991.

D. Darmofal. Bayesian spatial survival models for political event processes. *American Journal of Political Science*, 53(1):241–257, 2009.

K.-T. Fang, D. K. Lin, P. Winker, and Y. Zhang. Uniform design: theory and application. *Technometrics*, 42(3):237–248, 2000.

M. Giampapa, T. Gooding, T. Inglett, and R. W. Wisniewski. Experiences with a lightweight supercomputer kernel: Lessons learned from blue gene's CNK. In *SC'10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–10. IEEE, 2010.

R. Henderson, S. Shimakura, and D. Gorst. Modeling spatial variation in leukemia survival data. *Journal of the American Statistical Association*, 97(460):965–972, 2002.

Y. Hong, J. Lian, L. Xu, J. Min, Y. Wang, L. J. Freeman, and X. Deng. Statistical perspectives on reliability of artificial intelligence systems. *arXiv preprint arXiv:2111.05391*, 2021.

IOzone projects contributors. IOzone filesystem benchmark, 2016. http://www.iozone.org/.

M. E. Johnson, L. M. Moore, and D. Ylvisaker. Minimax and maximin distance designs. *Journal of Statistical Planning and Inference*, 26(2):131–148, 1990.

V. R. Joseph, E. Gul, and S. Ba. Maximum projection designs for computer experiments. *Biometrika*, 102(2):371–380, 2015.

Z. Lin, Y. Wang, and Y. Hong. The poisson multinomial distribution and its applications in voting theory, ecological inference, and machine learning. *arXiv preprint arXiv:2201.04237*, 2022.

T. C. Lux, L. T. Watson, T. H. Chang, L. Xu, Y. Wang, J. Bernard, Y. Hong, and K. W. Cameron. Effective nonparametric distribution modeling for distribution approximation applications. In *2020 SoutheastCon*, volume 2, pages 1–6. IEEE, 2020a.

T. C. H. Lux, L. T. Watson, T. H. Chang, L. Xu, Y. Wang, and Y. Hong. An algorithm for constructing monotone quintic interpolating splines. In *Proceedings of the 2020 Spring Simulation Conference*, SpringSim '20, 2020b.

M. D. McKay, R. J. Beckman, and W. J. Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

W. Q. Meeker and L. A. Escobar. *Statistical Methods for Reliability Data*. John Wiley & Sons, 1998.

J. Min and Y. Hong. Spatially correlated time-to-event model for Titan GPU failures data under competing risks, in progress. 2022.

M. D. Morris and T. J. Mitchell. Exploratory designs for computational experiments. *Journal of Statistical Planning and Inference*, 43(3):381–402, 1995.

J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423, 1989.

T. J. Santner, B. J. Williams, W. I. Notz, and B. J. Williams. *The design and analysis of computer experiments*, volume 1. Springer, 2003.

W. I. Thacker, J. Zhang, L. T. Watson, J. B. Birch, M. A. Iyer, and M. W. Berry. Algorithm 905: Sheppack: Modified Shepard algorithm for interpolation of scattered multivariate data. *ACM Transactions on Mathematical Software (TOMS)*, 37(3):1–20, 2010.

Y. Wang, D. Bandyopadhyay, J. R. Shaffer, and X. Wu. Gene-based association mapping for dental caries in the geneva consortium. *Journal of Dentistry and Dental Medicine*, 3, 2020.

Y. Wang, I. Lee, L. Lu, and Y. Hong. *Springer Handbook of Engineering Statistics, 2nd ed. (in press)*, chapter Statistical Analysis of Modern Reliability Data. 2021. doi: arXiv:1908.09729.

Y. Wang, Y. Hong, X. Deng, and L. Freeman. Variational inference for spatial survival model under Bayesian framework, in progress. 2022a.

Y. Wang, I. Lee, Y. Hong, and X. Deng. Building degradation index with variable selection for multivariate sensory data. *Journal of Reliability Engineering & System Safety (in revision)*, 2022b.

Y. Wang, L. Xu, Y. Hong, R. Pan, T. Chang, T. Lux, J. Bernard, L. T. Watson, and K. Cameron. Design strategies and approximation methods for high-performance computing variability management. *Journal of Quality Technology (in press)*, 2022c. doi: 10.1080/00224065.2022.2035285.

L. Xu, Y. Wang, T. C. Lux, T. Chang, J. Bernard, B. Li, Y. Hong, L. T. Watson, and K. W. Cameron. Modeling I/O performance variability in high-performance computing systems using mixture distributions. *Journal of Parallel and Distributed Computing*, 139:87–98, 2020.

L. Zhao, T. E. Hanson, and B. P. Carlin. Mixtures of Polya trees for flexible spatial frailty survival modelling. *Biometrika*, 96(2):263–276, 2009.

Chapter 2 Design Strategies and Approximation Methods for High-Performance Computing Variability Management

Abstract

Problem: Performance variability management is an active research area in high-performance computing (HPC). In this chapter, we focus on input/output (I/O) variability, which is a complicated function that is affected by many system factors. To study the performance variability, computer scientists often use grid-based designs (GBDs) which are equivalent to full factorial designs to collect I/O variability data, and use mathematical approximation methods to build a prediction model. Mathematical approximation models, as deterministic methods, could be biased particularly if extrapolations are needed. In statistics literature, space-filling designs (SFDs) and surrogate models such as Gaussian process (GP) are popular for data collection and building predictive models. The applicability of SFDs and surrogates in HPC variability management setting, however, needs investigation. In this case study, we investigate their applicability in the HPC setting in terms of design efficiency, prediction accuracy, and scalability.

Approach: We first customize the existing SFDs so that they can be applied in the HPC setting. We conduct a comprehensive investigation of design strategies and the prediction ability of approximation methods. We use both synthetic data simulated from three test functions and the real data from the HPC setting. We then compare different methods in terms of design efficiency, prediction accuracy, and scalability.

Results: In our synthetic and real data analysis, GP with SFDs outperforms in most scenarios. With respect to the choice of approximation models, GP is recommended if the data are collected by SFDs. If data are collected in a grid-based way, both GP and Delaunay can be considered. With the best choice of approximation method, the performance of SFDs and GBD depends on the property of the underlying surface. For the cases in which SFDs perform better, the number of design points needed for SFDs is about half of or less than that of the GBD to achieve the same prediction accuracy. Although we observe that the GBD can also outperform SFDs for smooth underlying surface, GBD is not scalable to high-dimensional experimental regions. Therefore, SFDs that can be tailored to high dimension and non-smooth surface are recommended especially when large numbers of input factors need to be considered in the model.

Key Words: Computer Experiment; Delaunay Triangulation; Gaussian Process; Linear Shepard's Method; MARS; Space-Filling Design.

2.1 Problem Description

The computing scale and complexity in modern technologies and scientific areas make high-performance computing (HPC) increasingly important. Performance variability, however, is an important challenge in the research of HPC systems, which has been observed for a long time (e.g., Giampapa et al., 2010, Akkan et al., 2012, Cameron et al., 2019). High variability in HPC systems can lead to unstable system performance and potentially high energy costs. Therefore, variability management is crucial for system performance optimization. The performance variability is affected by many complicated interactions of factors in the system. In this study, we focus on input/output (I/O) performance variability. The relationship between system configurations (e.g., CPU frequency, file size, record size, the number of I/O threads, and I/O operation modes) and the I/O performance variability is of interest.

One framework that has been used for HPC variability management involves three steps (Cameron et al., 2019), which are data collection on performance variability for a set of system configurations, building an approximation model to make predictions for new configurations, and using the prediction to do optimization for future designs. Statistical research is usually involved in the data collection and prediction steps. Although there is vast research on designs for data collections and approximation models for predictions, the applicability of those statistical methods in the setting of HPC performance management needs investigation, which motivates us to do a case study based on the HPC performance data.

In the data collection stage, researchers identify HPC system settings for which I/O throughput data should be collected. Computer scientists often use grid-based designs (GBDs) to collect data under numerous possible system configurations, when the number of factors is relatively small (Cameron et al., 2019, Xu et al., 2020b). Note that the GBDs are equivalent to full factorial designs. In statistics literature, space-filling designs (SFDs) are often used, which assign design points far apart to fill the whole experiment region. In the prediction

stage, an approximation model is built based on collected data for various system configurations. Mathematical approximation methods such as the linear Shepard's method (Shepard, 1968) and Delaunay triangulation (Delaunay, 1934) have been used. In statistics literature, Gaussian process (GP) models are popular for building approximation models.

From an HPC application point of view, there are several questions that need to be addressed. First, due to HPC system constraints, the design region could be irregular. For example, in our experiments, the file size needs to be larger than or equal to the record size, causing complexity in the experimental design. Therefore, existing SFDs need to be tailored so that they are suitable for the HPC setting. Second, it is desirable to demonstrate SFDs can collect data more efficiently than the GBD in a way that is accessible to computer scientists. Third, there is little research on the interaction between design strategies and approximation methods, especially in the setting of HPC variability management. However, it is possible that the prediction accuracy of a design may depend on the chosen approximation method.

Motivated by the needs in HPC performance variability management, we perform thorough comparisons between the prediction abilities of different approximation methods under different design strategies. We aim to recommend some efficient and scalable ways for computer scientists to collect performance data and provide a few practical guidelines in the HPC setting. Note that in this study, our goal is to model and analyze HPC performance variability data collected from HPC experiments, rather than using HPC to do statistical analysis.

In literature, SFDs are widely used for experimental design when little information is known about the phenomenon to be studied. The uniform design proposed by Fang et al. (2000) has the natural idea of placing design points uniformly in the experimental region. Latin hypercube designs (McKay et al., 1979) ensure good one-dimensional projection properties. Some design strategies are based on the distance measure, such as the maximin and minimax designs (Johnson et al., 1990). There are also several designs constructed based on the variants or combinations of the aforementioned designs, such as the maximin Latin hypercube design (Morris and Mitchell, 1995) and the maximum projection design (Joseph et al., 2015).

Non-regular and constrained design regions are quite common in industrial experiments

and physical sciences. Some efforts have been made to allocate design points within such regions. Draguljić et al. (2012) introduce an algorithm to construct noncollapsing space-filling designs for bounded input regions. The design in Draguljić et al. (2012) can be applied to regions with constraints, but it can be time-consuming, especially when the design size is in hundreds or thousands. Lekivetz and Jones (2015) propose the fast flexible space filling algorithm, which constructs designs based on hierarchical clustering. Pratola et al. (2017) map the original dimension to a higher dimensional space to convert the geodesic distance to Euclidean distance. Golchi and Loeppky (2015) adopt the idea of sampling from constrained distributions and propose a sequential constrained Monte Carlo algorithm to sample design points uniformly from the constrained input region. Hung et al. (2010) propose probability-based Latin hypercube design for slid-rectangular region with the ability to achieve optimal design criteria.

When the physical experiment or the corresponding computer simulation model is complex and time-consuming, a surrogate model is needed to describe the underlying process. Many smooth techniques, such as response surface models, Kriging methods, kernel estimation, and neural networks, can be used to approximate the true surface. Response surface methodology, originally introduced by Box and Wilson (1951), is a traditional technique for modeling the response variables given input variables. Another commonly-used statistical approximation model is Gaussian process (GP) regression (e.g., Sacks et al., 1989, Currin et al., 1991), which can generate a smooth surface and be capable of dealing with the heteroscedasticity (Goldberg et al., 1998) in the response variable. In the HPC community, mixture models have been used to study the multimodal behavior of the throughput distribution (Xu et al., 2020b). Some novel numerical techniques, including max box mesh, iterative box mesh, and Voronoi mesh methods for interpolation, are investigated by Lux et al. (2018b).

In previous work, the SFDs and approximation models have been compared, separately, in terms of prediction performance. The prediction accuracy of SFDs is studied by Johnson et al. (2011) with Gaussian process surrogates. Multiple approximation methods are compared under GBDs by Lux et al. (2018a) and Cameron et al. (2019). Those compared methods

include regression methods, such as multivariate adaptive regression spline model, support vector regression, multilayered perceptron regression, and some numerical methods, such as linear Shepard's method and Delaunay triangulation. However, little study has been done regarding the interaction between design strategies and approximation methods in the HPC performance setting. This paper aims to investigate the prediction ability of different kinds of design strategies and approximation methods as a case study. So, when a particular design strategy is given, HPC engineers can choose the best approximation method to achieve a higher prediction accuracy, or vice versa.

The rest of the paper is presented as follows. Section 2.2 provides a detailed background of the motivating example and introduces the underlying problem that we are interested in. Sections 2.3.1 and 2.3.2 briefly summarize the SFDs and approximation methods that are under investigation in this paper. Section 2.3.3 conducts synthetic data analyses where the performance of all combinations of the five designs and five approximation methods are compared under three test functions. In Section 2.3.4, an HPC variability experiment is introduced and real-world comparison results are presented. Section 2.4 provides conclusions of the comparisons, practical guidelines, and areas for future work.

2.2 Data Collection and Preparation

We first introduce some notation for the HPC data. To define a general experiment process, let $\mathbb{X}$ be a d-dimensional space of input factors with a set of constraints. Let $\boldsymbol{Y}$ be the random vector of the experiment outputs. The first step to start exploring the relationship between the input factors and the output is to efficiently allocate design points within $\mathbb{X}$, at which the corresponding output vectors will be obtained. Suppose $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ is the set of selected design points, where $\boldsymbol{x}_i \in \mathbb{X}$ is the ith design point. Let y_i be the corresponding observed output at the ith design point. Then $\boldsymbol{y} = (y_1, \ldots, y_n)$ is the collected output vector containing the observed outputs at all $\boldsymbol{x}_i \in \mathcal{D}$.

In the HPC data, there are four numeric factors, including one hardware factor: CPU

frequency (GHz); and three application factors: the number of I/O threads, the file size (KB), and the record size (KB). For our experiments, we consider CPU frequencies in the range $[2.0, 3.5]$ GHz, numbers of I/O threads in the range $[1, 64]$, and file sizes and record sizes in the range $[4, 16384]$ KB. Additionally, we have the constraint that for each system configuration, the file size must be greater than or equal to the record size, and the file size and record size must be of the form of $\sum_{l \in \mathcal{L}} 2^l$, where $\mathcal{L}$ is a set of positive integers. To make the design region uniform, we apply a log transformation to the file size and record size: *log file size* $=$ $\log_2$(*file size*), *log record size* $= \log_2$(*record size*). Then the experiment region becomes $\mathbb{X} = [2.0, 3.5] \times [1, 64] \times [2, 14] \times [2, 14]$ with the constraint that *log file size* $\geq$ *log record size*. In this application, we define the response of interest $\boldsymbol{Y}$ as the performance variability measurement (PVM). The PVM is given by the standard deviation of I/O throughputs (in the units of KB/s) at each input configuration (Cameron et al., 2019).

To collect data for this application, the I/O throughput of hard disk is collected in a grid-based pattern, using the IOzone benchmark (2016) to produce the workload with each system setting. Each factor is divided into k_i levels, $i = 1, \ldots, 4$ and a configuration is obtained by taking a possible combination of levels in each factor. Figures 2.1(a) and 2.1(b) show the two-dimensional projections of the real 4-dimension grid space where data are collected. At each configuration, the IOzone benchmark is run multiple times and the throughputs are gathered as an HPC performance measurement. The configurations are collected under 6 IO operation modes: initial_writers, rewriters, readers, re_readers, random_readers, random_writers. In total, we have 2658 configurations and each configuration has 300 replications to capture the performance variability. Because the data collection procedure is time-consuming, we are interested in whether SFDs can select points more efficiently than the GBDs.

After obtaining the data, another problem that we are interested in is the problem of accurately predicting HPC performance variability at a new configuration. With the collected dataset $\{\mathcal{D}, \boldsymbol{y}\}$, we want to build models that describe the relationship between system factors and performance variability. Based on the previous literature and studies, we adopt both statistical models and numerical models in computer science to explore the underlying

Table 2.1: Illustration of the HPC performance variability data structure.

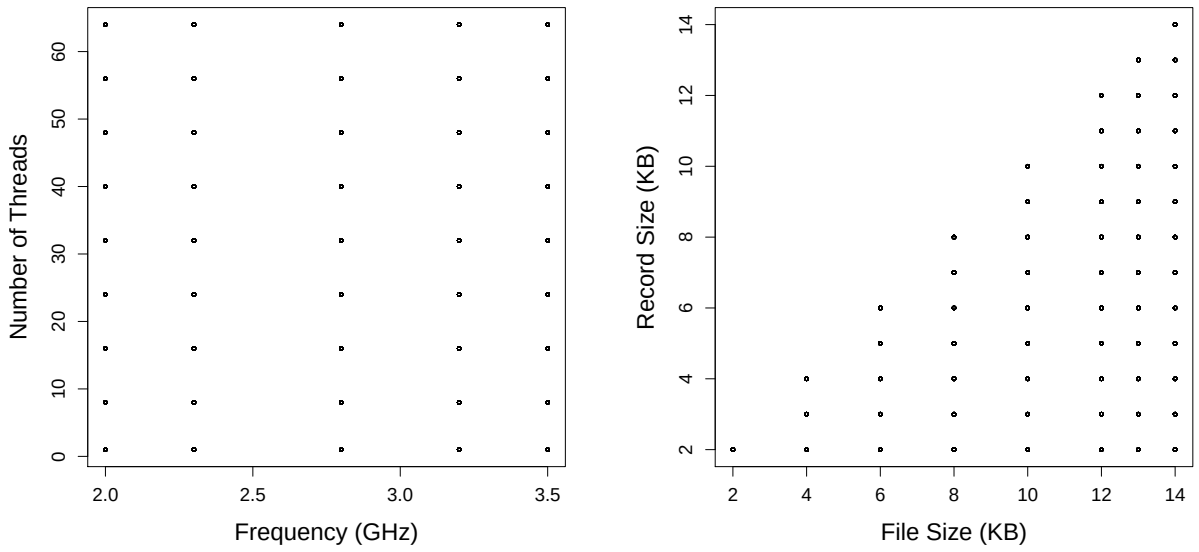
Frequency	No. of Threads	File Size	Record Size	PVM
2.0	1	2	2	17411.29
2.0	1	4	2	58014.19
2.0	1	4	3	46393.96
2.0	1	4	4	43238.60
2.0	1	6	2	49839.42
2.0	1	6	3	109721.24

relationship.

An example of the data structure is presented in Table 2.1. The 2D surface plots of the PVM under two pairs of factors are shown in Figure 2.2. From the surface plot, we gain a rough idea of the relationship between response and input factors. In Figure 2.2(a), we can see that the number of I/O threads and frequency have a positive relationship with the throughput variability; the larger the number of threads and frequency, the larger the PVM. Figure 2.2(b) shows that when the file size and record size are closer to each other (i.e., near the boundary of the constraint in the plot), the variability is relatively small. When the file size is much larger than the record size, the performance varies a lot. These relationships are consistent with our intuition.

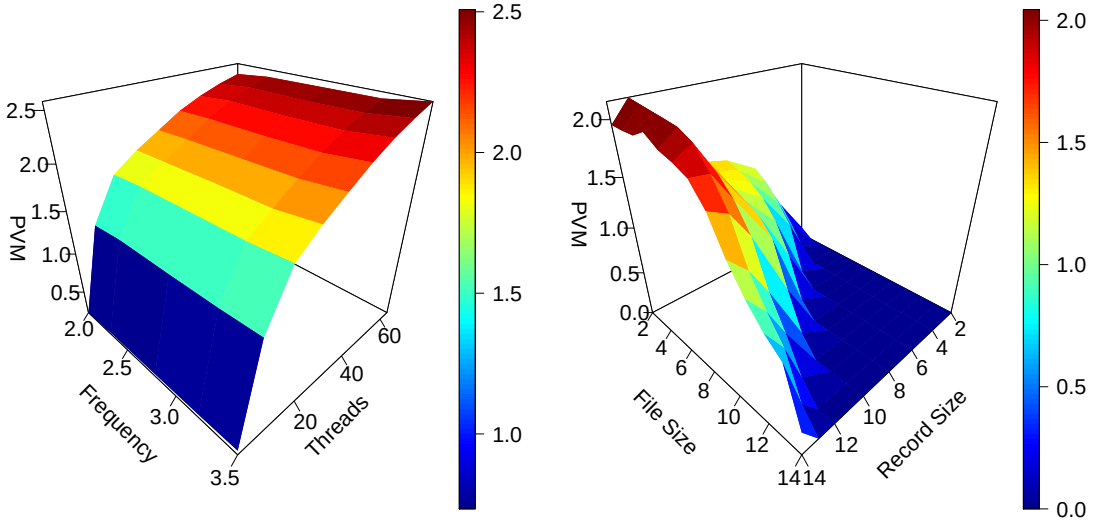
2.3 Analysis and Interpretation

In this section, we conduct analyses on the effectiveness of designs and approximation methods in HPC setting. We first give brief descriptions on the design strategies and approximations in Sections 2.3.1 and 2.3.2, respectively. We then examine the effectiveness of designs and approximation methods using synthetic data simulated from three different test functions. Finally, we study the designs and approximation methods using the real data from the HPC study.



(a) Frequency & Number of Threads (b) File Size & Record Size

Figure 2.1: Two-dimensional projection grids of design points where the real data were collected.



(a) Frequency & Threads (b) File Size & Record Size

Figure 2.2: 3D surface plots of performance variability measure (PVM) on the magnitude of 10^6 under two pairs of factors.

2.3.1 Design Strategies

Space-filling designs (SFDs), as the name suggests, spread out design points evenly in the experiment region in order to gather information from the whole experiment region. SFDs can assign points based on distance measures or sampling strategies. One reason to propose SFDs as an alternative to GBDs is that spreading points evenly across the entire design region is ideal when prediction accuracy is our primary goal. This is because the prediction error at a particular point depends on its location relative to the design points. If a design allocates points only near the center of the experiment region, a large error may result in prediction for inputs on the boundary of the experiment region. For convenience, we first assume that the experiment region is a unit hypercube $\mathbb{X} = [0, 1]^d$. This assumption can easily be relaxed with linear transformations. This section introduces several common criteria for constructing SFDs, as well as a list of designs we will study in this paper.

2.3.1.1 SFDs Constructed by Sampling Strategies

Several sampling strategies to construct SFDs are discussed in this section. An intuitive idea for a spread-out design is to scatter points uniformly in the design region. Designs built in this way are referred to as uniform designs (UDs), as described in Fang et al. (2000). UDs can gather a sufficient amount of information to explore the relationship between the response variable and the input factors with relatively small runs (Li et al., 2004). UDs consider the whole design region equally important. However, when some portions of the domain are of more interest than others, stratified random sampling (SRS) can be used to enhance the design performance. Suppose n design points are desired. The SRS partitions the design region $\mathbb{X}$ into s strata and in stratum j, n_j points are selected based on a certain input distribution, where $j = 1, \ldots, s$ and $\sum_{j=1}^{s} n_j = n$. The size and position of each stratum depends on different experiment scenarios. When we know that some input factors are important to the response, we also want the design points' projections on to those factors to be spread out. This can be achieved by the Latin hypercube design (LHD) (McKay et al., 1979), which is a popular

SFD and has been combined with various other designs. To construct an LHD of size n, the range of each input factor is equally divided into n intervals $[0, 1/n), \ldots, [(n-1)/n, 1]$. For each of the d coordinates, exactly one design point projection is sampled from each interval. In this way, it can be guaranteed that the design points are spread out across the range of each input factor. One favorable property of LHD is that any lower dimension projection of an LHD is also an LHD. Although LHDs have good properties for projection, it is not guaranteed that the design points will be spread out evenly in the entire experiment region. For example, it is possible that all design points could be located along the diagonal of the d-dimensional unit hypercube. To overcome this drawback, Latin hypercube sampling is often used together with other designs such as maximin design (Johnson et al., 1990) or maximum projection design (Joseph et al., 2015). These SFDs are based on a distance metric, which will be discussed in the next section.

2.3.1.2 SFDs Based on Distance Measures

In this section, design strategies based on distance measures and metrics are briefly introduced. One idea for SFDs is that no point in the experiment region $\mathbb{X}$ should be too far from its nearest neighbor in $\mathcal{D}$. Let

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(\sum_{k=1}^{d} |x_{ik} - x_{jk}|^s \right)^{1/s}$$

be the distance metric. Usually, the Euclidean distance, (i.e., $s = 2$) is used. For an arbitrary point $\boldsymbol{x}$ in $\mathbb{X}$, we determine its closest design point $\boldsymbol{x}_i$ and the minimum distance $\min_i d(\boldsymbol{x}, \boldsymbol{x}_i)$. To guarantee that no point is too far from the design points, we choose the point $\boldsymbol{x} \in \mathbb{X}$ with the maximum distance to its closest design point, which is $\max_{\boldsymbol{x} \in \mathbb{X}} \min_i d(\boldsymbol{x}, \boldsymbol{x}_i)$. Then we find the design to minimize this distance,

$$\min_{\mathcal{D}} \max_{\boldsymbol{x} \in \mathbb{X}} \min_i d(\boldsymbol{x}, \boldsymbol{x}_i),$$

which is called the minimax distance design (Johnson et al., 1990). The second way to spread out points in $\mathcal{D}$ is to allocate the design points as far apart as possible. This can be realized by finding the design that maximizes the minimum distance between two design points, which is the criterion of the maximin distance design,

$$\max_{\mathcal{D}} \min_{i,j} d(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

The maximin distance design ensures that the design points are spread as far apart from each other as possible in the full dimension but does not guarantee that the design is space filling for each projection on a subspace. Morris and Mitchell (1995) propose the maximin Latin hypercube (MmLh) design, which incorporates the structure of an LHD with the maximin design. The criterion is

$$\min_{\mathcal{D}} \left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{d^m(\boldsymbol{x}_i, \boldsymbol{x}_j)} \right]^{1/m}. \tag{2.1}$$

The MmLh design ensures good projection properties when projecting into one dimension, but fails to consider the projection onto other subdimensions. Joseph et al. (2015) propose a maximum projection design (MaxPro) that ensures good space filling on all subspaces. It considers a weighted Euclidean distance

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{\theta}) = \left[\sum_{k=1}^{d} \theta_k (x_{ik} - x_{jk})^2 \right]^{1/2}, \tag{2.2}$$

where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d)$ is a vector of weight factors. Let $\theta_k = 1$ for those factors that construct the subspace and $\theta_k = 0$ for other factors, then (2.2) calculates the distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ after projection into subspaces. The criterion of maximum projection design can be modified based on (2.1) as

$$\min_{\mathcal{D}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{d^m(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{\theta})}.$$

Here, the weight factors satisfy $\sum_{k=1}^{d} \theta_k = 1$. To properly choose the weight factor $\boldsymbol{\theta}$, Joseph et al. (2015) adopt the Bayesian framework. A prior distribution is assigned to $\boldsymbol{\theta}$ and the

expected value of the objective function is minimized. The criterion can be further simplified with uniform prior and $m = 2d$:

$$\min_{\mathcal{D}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{\prod_{k=1}^{d} (x_{ik} - x_{jk})^2}. \tag{2.3}$$

It is clear from (2.3) that any two design points in $\mathcal{D}$ can not have the same value in any dimension. Otherwise, $x_{ik} - x_{jk} = 0$, which will cause the objective function to be infinite. The criterion automatically guarantees that the design also has the LHD property.

Another design strategy, the maximum entropy design is also included in this section, although generally it is considered to be a model-based design. The maximum entropy design (Shewry and Wynn, 1987) maximizes the negative entropy function, which captures the amount of information gained from an experiment. For a random vector $\boldsymbol{X}$ with support $\mathbb{X}$ and density function $f(\boldsymbol{X})$, the entropy is defined as $H(\boldsymbol{X}) = -\mathrm{E}_{\boldsymbol{X}}[\log f(\boldsymbol{X})] = -\int_{\mathbb{X}} f(\boldsymbol{x}) \log[f(\boldsymbol{x})] d\boldsymbol{x}$. The information gained is $I(\boldsymbol{X}) = -H(\boldsymbol{X})$. We want the experimental design that causes the largest change in the information, which is equivalent to maximizing the entropy:

$$\max_{\mathcal{D}} H(\boldsymbol{X}).$$

This can be simplified as $\max_{\mathcal{D}} \log |\Sigma_n|$ if we assume that the underlying surface is a Gaussian process, where Σ_n is the variance-covariance matrix of $\mathcal{D}$. Then the construction of the maximum entropy design depends on the choice of the correlation function. In this study, we adopt the format that is implemented in the R package *DiceDesign* (Dupuy et al., 2015), where the correlation function is $r(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1 - 1.5d/a - 0.5(d/a)^3$ if $d > a$ and 0 otherwise. Here, d is the Euclidean distance between the two points ($\boldsymbol{x}_i$ and $\boldsymbol{x}_j$) and a denotes the range of the variogram.

2.3.1.3 Customizing SFDs for HPC Setting

Four of the above designs are considered in this paper. They are uniform sampling, maximum Latin hypercube design (MnLh), maximum entropy design (MaxEnt), and maximum projec-

tion LHD (MaxPro). In the HPC performance variability modeling application, the additional constraint is that the file size needs to be larger than or equal to the record size. To deal with the constraint in the real application, we want to use an approach that can tailor all different designs. So above designs must be adjusted in order to accommodate the constrained region. In this study, a rejection sampling strategy, as a simple way that can be easily applied across different design strategies, is used. Suppose we would like to generate a design with size n. If points in the initial size n design fail to meet the constraint, then we generate designs with size $2n$, $3n$, ..., until there are at least n points in the design that satisfy the constraint. Then we randomly select n points from the large design, which satisfy our constraint.

Note that this idea is different from rejection sampling, because we reject points that are obtained from a specific optimization problem. After discarding the points that do not meet the constraint, the remaining design points may not hold all the original properties of that type of design. However, this is a uniform, simple approach that can be applied to any type of design. We do not need to solve extra optimization problems or alter the existing sampling algorithm.

Besides tailoring SFDs for the constraint in the HPC problem, we also need to adjust designs to include boundary points so that we will not have bad extrapolation problems when we using the data to build numerical prediction models. The details are explained in Section 2.3.2.1. In this study, we use central composite designs (CCDs) to augment origin SFDs. A CCD consists of a factorial (fractional factorial) design with factors of two levels. That is, the vertices of the experiment region, a set of center points, and a set of axial points. We add the vertex points, the center point of the axis, and the center point of the hypercube to the SFDs. If the experiment region is irregular, then the augmented points that do not meet the constraint are excluded.

2.3.2 Approximation Methods

To conduct the prediction accuracy comparison, we need to select several representative surrogates to approximate the underlying function. The approximation methods often used in

both computer science and statistics are briefly introduced here.

In applied mathematics, numerical methods are often used to approximate the true surface. In this section, two numerical methods are used, which are Delaunay triangulation and Linear Shepard's method.

2.3.2.1 Delaunay Triangulation Interpolation

Delaunay triangulation interpolation is a numerical method that approximates the underlying function $f : \mathbb{R}^d \to \mathbb{R}$ based on the values $f(\boldsymbol{p})$ for a given vertex set $\mathcal{P}$ and the corresponding Delaunay triangulation. A Delaunay triangulation (Delaunay, 1934) is given by any triangulation that satisfies the Delaunay properties. Suppose $\mathcal{P} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$ is a set of n points in $\mathbb{R}^d$. Then a d-dimensional triangulation of $\mathcal{P}$, denoted $\mathrm{T}(\mathcal{P})$, is a set of d-simplices that satisfies the following criteria: 1) the vertex set of $\mathrm{T}(\mathcal{P})$ is $\mathcal{P}$; 2) the union of all simplices in $\mathrm{T}(\mathcal{P})$ is the convex hull of $\mathcal{P}$, denoted as $\mathrm{CH}(\mathcal{P})$; and 3) the d-simplices are disjoint besides their common boundaries (vertices, edges and facets). A Delaunay triangulation, denoted as $\mathrm{DT}(\mathcal{P})$, is the geometric dual of the Voronoi diagram (Watson, 1981). A Voronoi diagram divides $\mathbb{R}^d$ into n regions with each containing one Delaunay vertex in $\mathcal{P}$, such that all points within each region are closer to the vertex in their region than to any other vertex. The Delaunay triangulation is given by connecting all vertices whose Voronoi cells share a boundary with an edge.

Let $\boldsymbol{x} \in \mathrm{CH}(\mathcal{P})$ be an interpolation point and $S \in \mathrm{DT}(\mathcal{P})$ be the simplex with vertices $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{d+1}$ that contains $\boldsymbol{x}$. One finds the weights $w_1, \ldots, w_{d+1}$ that satisfy $\sum_{i=1}^{d+1} w_i = 1$, $w_i \geq 0$, for $i = 1, \ldots, d+1$ and $\boldsymbol{x} = \sum_{i=1}^{d+1} w_i \boldsymbol{s}_i$. Then the estimated function value $\widehat{f}_{\mathrm{DT}}(\boldsymbol{x})$ based on $\mathrm{DT}(\mathcal{P})$ is

$$\widehat{f}_{\mathrm{DT}}(\boldsymbol{x}) = w_1 f(\boldsymbol{s}_1) + w_2 f(\boldsymbol{s}_2) + \cdots + w_{d+1} f(s_{d+1}).$$

In this paper, the Fortran 2003 package DELAUNAYSPARSE (Chang et al., 2020) is used to perform the interpolations. In order to achieve computational efficiency, the algorithm in

DELAUNAYSPARSE only computes a necessary, sparse subset of the Delaunay triangulation given pre-specified interpolation points (Chang et al., 2018).

In the HPC variability management application, in order to compute the predictions, we need to approximate the response values for a test set based on the Delaunay triangulation $\mathrm{DT}(\mathcal{D})$ of the proposed design $\mathcal{D}$. However, the points in the test set might not always fall inside $\mathrm{CH}(\mathcal{D})$, which results in an extrapolation problem instead of an interpolation problem. DELAUNAYSPARSE can handle extrapolation problems by projecting each test point onto $\mathrm{CH}(\mathcal{D})$ when the test point is close to $\mathrm{CH}(\mathcal{D})$, but this solution can perform badly if the test point is far outside of $\mathrm{CH}(\mathcal{D})$. Since we do not know whether a test point is inside $\mathrm{CH}(\mathcal{D})$ beforehand, necessary adjustments of the design strategies need to be made to avoid bad extrapolation problems.

In order to avoid bad extrapolation problems when using the Delaunay method to approximate the true surface, we need to augment the proposed SFDs so that the convex hull of each augmented SFD covers the entire experimental region. In this way, wherever a test point falls in the experimental region, it always results in an interpolation problem for the Delaunay method. To realize this idea, intuitively one can augment each SFD with those boundary points. In this study, we choose to use CCD to augment the proposed SFDs.

2.3.2.2 Linear Shepard Method

Shepard's method is a form of inverse distance weighting, originally proposed by Shepard (1968). It is an interpolation method based on the weighted average of basis functions, each centered on a point in the data set $\mathcal{P} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$, where the weight is calculated in terms of inverse distance from the interpolation point to points in $\mathcal{P}$. Several variations of Shepard's method are available such as, quadratic and cubic Shepard's method. However, in our application, linear Shepard's method is used for the sake of efficiency. Given the same setting as in Section 2.3.2.1, the original Shepard approximation of $f(\boldsymbol{x})$ at point $\boldsymbol{x}$ is:

$$\widehat{f}(\boldsymbol{x}) = \frac{\sum_{i=1}^{n} W_i(\boldsymbol{x}) f(\boldsymbol{p}_i)}{\sum_{i=1}^{n} W_i(\boldsymbol{x})},$$

where $W_i(\boldsymbol{x}) = 1/\|\boldsymbol{x} - \boldsymbol{p}_i\|_2^2$. This weight is nonzero for all the data points even for those points that are far away from the interpolation point $\boldsymbol{x}$. In order to achieve better approximation performance, a modified linear Shepard's method considers only the local points within an R-sphere of $\boldsymbol{x}$ and replaces the original $f(\boldsymbol{p}_i)$ with a linear approximation function $B(\boldsymbol{p}_i)$. The modified linear Shepard's method has the form

$$\widehat{f}(\boldsymbol{x}) = \frac{\sum_{i=1}^{n} W_i(\boldsymbol{x}) B(\boldsymbol{p}_i)}{\sum_{i=1}^{n} W_i(\boldsymbol{x})},$$

where the modified version weights are given by

$$W_i(\boldsymbol{x}) = \left[\frac{\max\left(0, R_i - d(\boldsymbol{x}, \boldsymbol{p}_i)\right)}{R_i d(\boldsymbol{x}, \boldsymbol{p}_i)} \right]^2.$$

Here, R_i is the radius of a sphere centered at $\boldsymbol{x}_i$ that reflects influence scope of $\boldsymbol{x}_i$. The Fortran package SHEPPACK (Thacker et al., 2010) is used in our study to perform the linear Shepard interpolation.

Besides the numerical methods, three statistical models are considered in this study: response surface methodology (RSM), multivariate adaptive regression splines (MARS) and Gaussian processes (GPs). Compared to numerical models, one advantage of statistical models is that one can quantify the prediction uncertainty in a relatively easy manner.

2.3.2.3 Response Surface Methodology

Response surface methodology (RSM) is a method that investigates the relationship between a response variable and input factors through design experiments. It is a traditional method for studying computer experiments (e.g., Myers, 1999, Myers et al., 2004). Low-order polynomial models such as first- or second-degree polynomial models are commonly used in applications. In this paper, we use a second-order model to approximate the true response surface.

2.3.2.4 Multivariate Adaptive Regression Splines

Multivariate adaptive regression splines (MARS) were introduced by Friedman (1991). They are nonparametric regression models formed by spline basis product expansion. MARS can automatically capture nonlinearity and interaction effects. The MARS model is given by:

$$\widehat{f}(x) = \sum_{m=1}^{M} c_m B_m(x),$$

where $B_m(x)$ are the basis functions. These basis functions can be constants, hinge functions, or products of hinge functions, where each hinge function is of the form $\max(0, x - c)$ or $\max(0, c - x)$, where c is a constant. As a flexible nonparametric model, MARS tends to overfit without pruning. The algorithm for constructing MARS model is based on a modification of recursive partition trees that requires a forward and backward pass. In the forward pass, the MARS model is initialized as a constant valued function (whose value is the intercept), then basis functions are gradually added to the model until the maximum number of terms is reached or the loss in sum of squared residuals is small. Next, a backward pass is used to prune the model based on the generalized cross validation (GCV) criterion, which is a trade off between goodness-of-fit and model complexity. In this study, we use the *earth* package in R to build the MARS model (Milborrow, 2019). A grid of hyper-parameter: the number of maximum terms in the model and the maximum degree of interactions, is used to train the model in order to the select model with the highest R-squared value.

2.3.2.5 Gaussian Process

Gaussian process (GP) interpolation is a commonly used approximation method in computer experiments. GP is often defined as a stochastic process where every finite collection of n observations follows a multivariate normal (MVN) distribution. A GP is determined by its mean function $\mu(\boldsymbol{x})$ and its covariance function $C(\boldsymbol{x}, \boldsymbol{x}')$. When a zero-mean GP is assumed, it can be completely determined by the covariance function, which is also referred

to as the kernel function. There are several common kernel functions, including Gaussian $C(\boldsymbol{x}, \boldsymbol{x}') = \exp[-(\boldsymbol{x} - \boldsymbol{x}')^2/\theta]$ and Matérn kernels. Let $\boldsymbol{Y}_n = \{y_1, \ldots, y_n\}$ be the n observations at the proposed design points $\boldsymbol{X}_n = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ and $\boldsymbol{Y} \sim \mathrm{N}_n(0, \Sigma_n)$, where Σ_n is the covariance matrix with covariance elements $C(\boldsymbol{x}, \boldsymbol{x}')$. Then for an arbitrary point $\boldsymbol{x}$, the value of $Y(\boldsymbol{x})$ given the design and corresponding observations can be obtained by the conditional distribution $Y(\boldsymbol{x})|\{\boldsymbol{Y}_n, \boldsymbol{X}_n\}$. This can be calculated based on the conditional distribution of MVN:

$$Y(\boldsymbol{x})|\{\boldsymbol{Y}_n, \boldsymbol{X}_n\} \sim \mathrm{N}[\mu(\boldsymbol{x}), \sigma^2(\boldsymbol{x})],$$

where $\mu(\boldsymbol{x}) = \Sigma(\boldsymbol{x}, \boldsymbol{X}_n)\Sigma_n^{-1}\boldsymbol{Y}_n$ and $\sigma^2(\boldsymbol{x}) = \Sigma(\boldsymbol{x}, \boldsymbol{x}) - \Sigma(\boldsymbol{x}, \boldsymbol{X}_n)\Sigma_n^{-1}\Sigma(\boldsymbol{X}_n, \boldsymbol{x})$. Here, $\Sigma(\boldsymbol{x}, \boldsymbol{X}_n)$ is a $1 \times n$ matrix with elements $C(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, C(\boldsymbol{x}, \boldsymbol{x}_n)$. It is obvious that the mean function is a linear combination of $\boldsymbol{Y}_n$ while the covariance function does not involve information of observations. A maximum likelihood estimator can be used for parameter estimation. We use the R package *laGP* (Gramacy, 2016) to implement the local Gaussian process approximation. In the local GP model, if one wants to predict at $\boldsymbol{x}$, instead of using the whole design $\mathcal{D}$, a subset of $\mathcal{D}$ close to $\boldsymbol{x}$ is selected sequentially to increase the computing speed. Note that when building the GP model, both the input variables and the response are normalized to range $[0, 1]$.

2.3.3 Synthetic Data Analyses

In order to investigate the performance of each design strategy and approximation method combination, we conduct synthetic data analyses using three test functions. Specifically, if we denote y_i as the true response for the ith design point and $\hat{y}_i$ is the predicted value, $i = 1, \ldots, n$, we compare the root mean squared error (RMSE) $\sqrt{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2/n}$ and the mean absolute percentage error (MAPE) $\sum_{i=1}^{n}\left(|\hat{y}_i - y_i|/y_i\right)/n$ of the predictions for each combination with each test function.

We have two goals when conducting the synthetic data analyses. First, we want to perform simulations with test functions that are representative of the HPC performance. Since this

application has four input factors, we choose two test functions that have four input variables each, and for each of these test functions, we apply the same linear constraint function as in the HPC performance variance problem. Second, we would like to explore the prediction behavior and computing time of GBD and SFD with a high-dimension test function. So the eight-dimensional Borehole function is adopted to illustrate the design performance when the experiment region is of high dimension.

2.3.3.1 Test Functions

The Colville function is a four-dimensional function with the formula:

$$f(\boldsymbol{x}) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + \\ 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1),$$

where $\boldsymbol{x} = (x_1, \ldots, x_4)$. The Colville function's input domain is $x_i \in [-10, 10]$, $i = 1, \ldots, 4$. We apply the constraint $x_3 \geq x_4$ to the input domain to emulate the HPC performance variability problem.

The Friedman function was proposed by Friedman et al. (1983). It is a five-dimensional function, and in our study, we map the first four variables to our experiment region and fix the last variable x_5. The function is:

$$f(\boldsymbol{x}) = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5,$$

where $\boldsymbol{x} = (x_1, \ldots, x_4)$. This function's domain is the unit hypercube: $x_i \in [0, 1]$, $i = 1, \ldots, 4$ with $x_5 = 0.5$. The same constraint as with Colville function was applied here.

The Borehole function is an eight-dimensional function that models water flow rate through a borehole. Let $\boldsymbol{x} = (r_w, r, T_u, H_u, T_l, H_l, L, K_w)$ be the input variables, then the water flow rate is:

$$f(\boldsymbol{x}) = \frac{2\pi T_u(H_u - H_l)}{\log(r/r_w)\left(1 + \frac{T_u}{T_l} + \frac{2LT_u}{\log(r/r_w)K_w r_w^2}\right)}.$$

Table 2.2: Ranges of parameters for the Borehole function.

Variable	Minimum	Maximum
r_w	0.05	0.15
r	100	50000
T_u	63070	115600
H_u	990	1110
T_l	63.1	116
H_l	700	820
L	1120	1680
K_w	9855	12045

The input ranges are listed in Table 2.2. For this test function, no constraint is applied because the goal of this test function is to investigate the high-dimensional performance of the design strategy and the approximation method combinations.

2.3.3.2 Comparison Procedures

Using the above test functions, we want to compare the prediction accuracy of GBDs with that of proposed SFDs for each test function using various design sizes and approximation methods. Since the GBD is built by selecting n levels on each factor and then enumerate all possible combinations of d factors, the design size needs to be n^d. In a real application, it is possible that the numbers of levels at each factor are different. However, since our test functions are continuous functions, we assume that if we can take n levels on one factor, we can also take the same number of levels on all other factors, which means we only consider the fine "regular grid" in the synthetic data analyses. After deciding the size of GBD, we can generate SFDs correspondingly. The simulation procedure is as follow:

1. Choose a test function. Uniformly select n_g random points within the input region as the test set g.

2. For $n = 3, \ldots, 7$,

   (a) Create a GBD $\mathcal{D}_g$ by choosing n points in each dimension and expanding into a grid via the Cartesian product. Exclude design points that do not satisfy the constraint

and denote the size of GBD as N_g.

(b) To generate the SFDs, create designs $\mathcal{D}_{maximin}$, $\mathcal{D}_{maxpro}$, $\mathcal{D}_{maxent}$, and $\mathcal{D}_{uniform}$ each of size $N = N_g - n_a$, where n_a is the size of the augmented design.

(c) For each test function, find the corresponding values of points in the above designs.

(d) Use five approximation methods to generate five predictive surfaces for each design:

i. Linear regression: use backward selection to determine the second-order linear regression model with minimum Bayesian information criterion (BIC).

ii. Delaunay triangulation: use the DELAUNAYSPARSE package to build the model.

iii. Linear Shepard: use the SHEPPACK package to build the model.

iv. MARS: use cross validation to tune the MARS model on a hyper-parameter grid, and select the model with the highest R-squared value.

v. Gaussian process: use the separable Gaussian kernel with a nugget effect included in the model.

(e) Repeat Steps 2c - 2d B times.

(f) Compute RMSE and MAPE over repetitions for each SFD and approximation method.

Although in Section 2.3.1 we introduced that the SFDs are obtained by solving different optimization problems, those optimization problems often do not have analytic solutions if large number of points are desired in a high-dimensional experiment region. Numerical algorithms are usually used to obtain SFDs, which lead to non-unique solutions for a certain type of SFD. In order to understand the overall prediction performance for a certain type of SFD, we repeat the step of generating designs and making predictions for B times. In the above procedures, the test set size n_g and the repetition number B is changed according to the dimension of the test function and the approximation method. For relatively smooth test functions, (e.g., Friedman function) or stable approximation methods, (e.g., Delaunay

and MARS), we do not require a large number of replications or a large test set to obtain a stable and representative result. However for other models, such as using the linear Shepard's method under a non-smooth test function, we need to increase the replication number in order to get a reliable result. The summary of the test sizes n_g and replication numbers B are listed in Table 2.3.

For the Borehole function, since the dimension is relatively high, it is difficult to compute the prediction performance for a series of GBDs within a reasonable time and with reasonable computational resources. Therefore, we skip the step of generating multiple GBDs of the same size as each SFD. Instead, we consider one GBD of size $3^8 = 6561$ GBD and compare its performance with other SFDs varying sizes from $\{500, 600, \ldots, 2000\}$.

For each test function, we plot the average RMSE and MAPE versus design size for each combination of approximation methods and design strategies. The results are shown in Figures 2.3, 2.4, and 2.5. The results show that the overall error decreases as the design size increases, but at different rates for different methods and problems. For the two four-dimensional test functions, we can see an interaction effect between the approximation method and design strategy. Under numerical approximation methods, GBD has a smaller prediction error as size increasing compared to the SFDs. However, under statistical models, the trend is the opposite. One possible explanation for this behavior is that GBDs have good geometric properties and both numerical approximation methods, Delaunay and linear Shepard, rely on geometric properties to make predictions. For the eight-dimensional Borehole function, the 3^8 GBD is shown as a horizontal line in Figure 2.5. In general for the Borehole function, the GBD does not perform as well as the SFDs, despite the fact that each SFD has a smaller design size.

2.3.4 HPC Data Analysis

In this section, we conduct a data analysis using the real data as described in Section 2.2.

Table 2.3: Test set size n_g and replication time B in the synthetic data analyses.

Surrogate Model	Friedman		Colville		Borehole	
	n_g	B	n_g	B	n_g	B
RSM	10000	30	10000	30	5000	60
Delaunay	10000	30	10000	30	5000	60
LSP	10000	30	10000	30	5000	180
MARS	10000	30	10000	30	5000	60
GP	10000	30	10000	30	5000	60

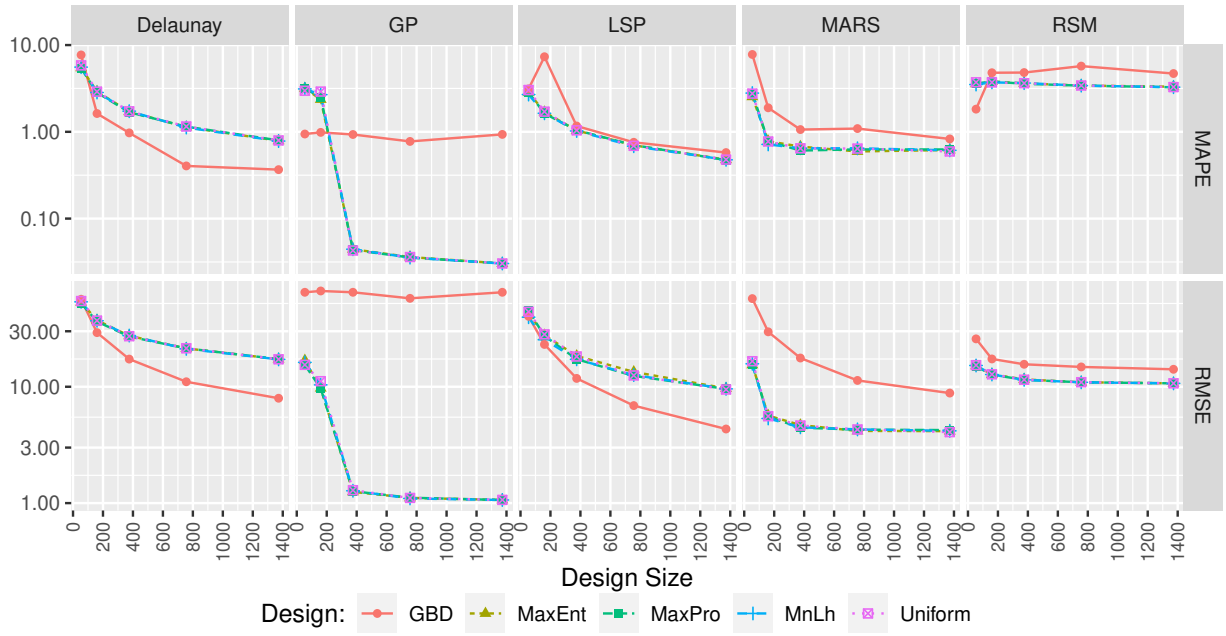


Figure 2.3: Plot of RMSE and MAPE as functions of design size under the Colville test function. RMSE is on the magnitude of 10^4.

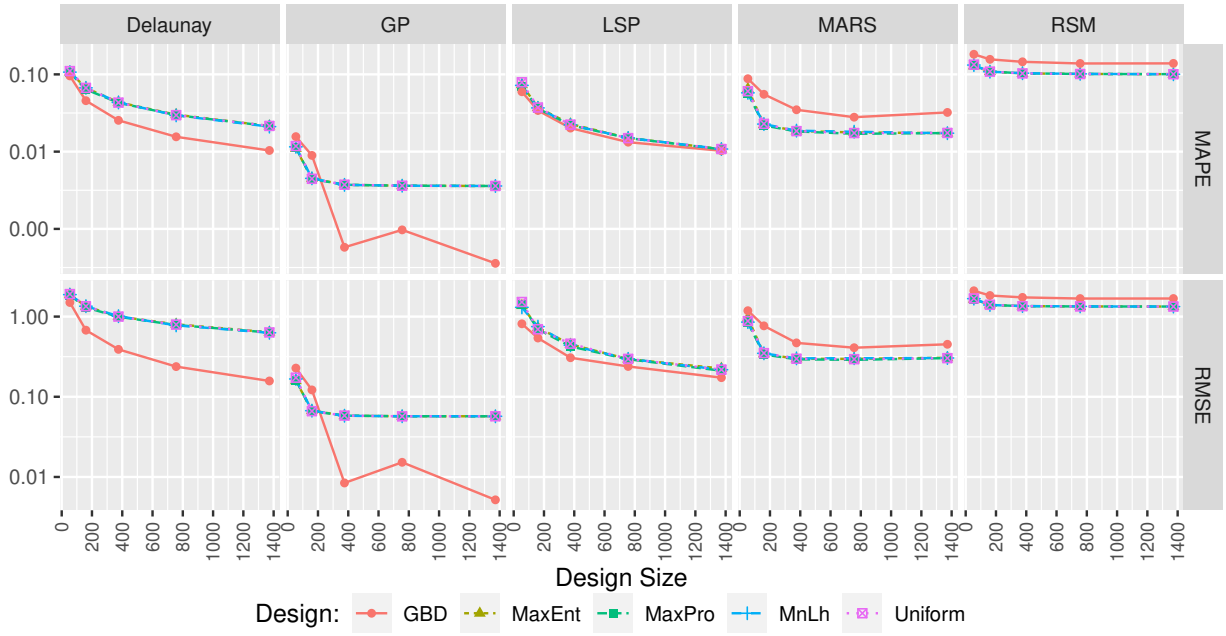


Figure 2.4: Plot of RMSE and MAPE as functions of design size under the Friedman test function.

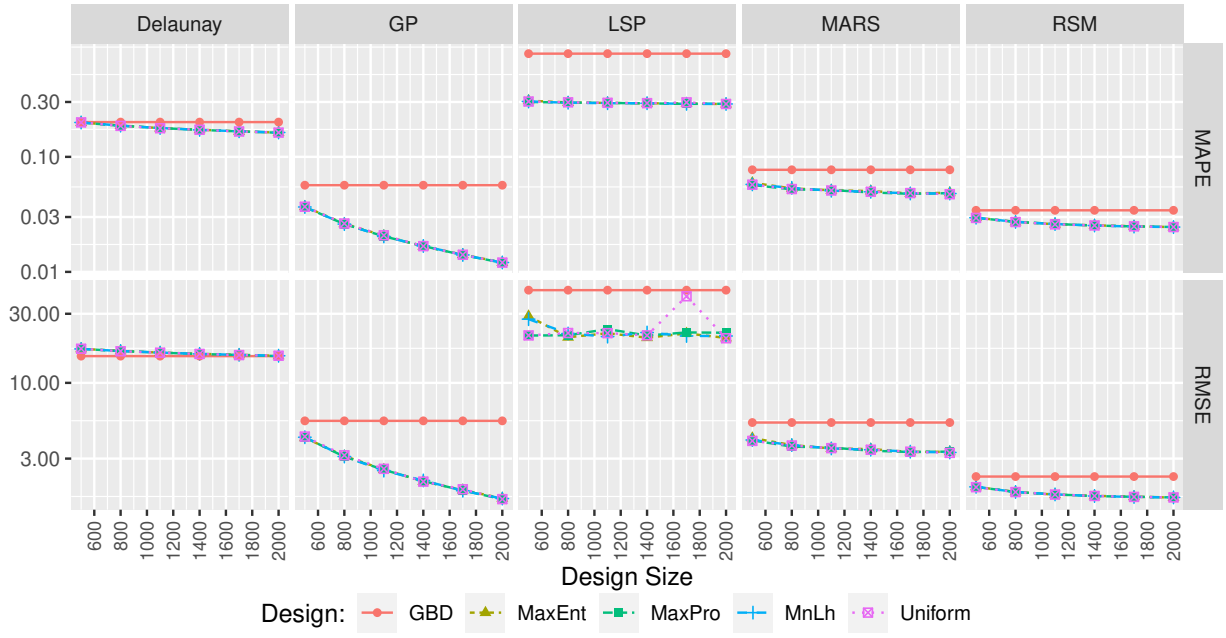


Figure 2.5: Plot of RMSE and MAPE as functions of design size under the Borehole test function. Here, the GBD has a fixed size of $3^8 = 6561$ due to the computational limit.

2.3.4.1 Model Fittings

In HPC application, the data were collected using a full factorial design. Each of the four input factors has several unique levels and the throughput data were collected at each combination of levels. In total there are 2658 possible configurations. Since the real data itself is a GBD, we construct SFDs with sizes increased from 100 to 2700 with an increment of 200, and compare them to the real data. In this way, we investigate whether SFDs can achieve the same prediction accuracy as the GBD with a smaller design size.

A similar procedure as in Section 2.3.3.2 is used. Note that we did not consider the log transformation when building statistical models using the real data, although the response PVM is non-negative. Because the performance variability in our study is quite large, which is far away from 0. The responses are not highly skewed around zero. Thus, the non-negativity is of less concern. One possible side effect of the log transformation is that it can result in extremely large predictions after taking the anti-log to obtain the prediction in the original scale.

There are a few modifications in the comparison procedure when analyzing the HPC data. Since the experiment region in the HPC performance variability problem is a discrete space, the SFD points are binned to the nearest feasible value after generation. Also because we do not know the true underlying surface in the real application as in the synthetic data analysis, we need to decide an underlying surface that can describe the real data well and also suitable for conducting comparisons. We choose to use a fitted model with methods in Section 2.3.2 using the real data as the truth. In order to choose the most appropriate model, we first use a 10-fold cross validation (CV) to see the average CV prediction errors for different models. The result is summarized in Table 2.4. Choosing the model that has the smallest CV prediction error is a natural idea. In this case, it is either Delaunay which has the smallest MAPE or GP with the smallest RMSE. However, one property of Delaunay method is that the fitted model traverses every training data point. This means if we use the fitted Delaunay from the real data as the underlying truth, the response value we take from the fitted surface for the

Table 2.4: Average 10-folds CV error for the HPC data under different models.

Model	RSM	MARS	Delaunay	LSP	GP
MAPE	0.47	0.39	0.20	0.26	0.22
RMSE	89476.99	80468.44	63394.17	76488.13	54854.81

GBD, is exactly the real collected PVM. This benefits the GBD since the GBD has exactly the same data we used to build the true surface. When building approximation models to make predictions, the GBD has advantages over SFDs. Especially if we use the Delaunay approximation method, we will obtain 0 prediction error for the GBD. In this way, we can not make a fair comparison for the performance of the GBD and the SFD. If we consider the GP model, without the nugget effect, the GP model also goes through every training point, resulting in the same situation as using the Delaunay method. Even though in our analysis, we include the nugget effect so that the model does not go through every training point, and the GBD will not benefit as much as in the above situation. Using the fitted GP model is still not a fair comparison. That is because the variance-covariance matrix of the fitted GP model trained with the real data is featured by the evenly spaced design points. Potentially, using fitted GP model as the truth also benefits the GBD, which has good geometric properties. Similarly, LSP model also has the same problem as the Delaunay. Therefore, in order to have a fair comparison, we decided to use the MARS fitted model. It will not return exactly the same PVM for the GBD, also is not affected by the geometric property. Although its ability to describe the real data is not as good as the above three methods, we choose it as the truth in the real data analysis for a fair comparison.

2.3.4.2 Comparisons

Figure 2.6 describes the MAPE and RMSE of different methods and design combinations when using MARS fitted model as the underlying truth. The error trends of all SFDs are similar to each other under every approximation method. GP with SFDs can achieve the smallest prediction error under both criteria. Like most cases in synthetic analysis, the GBD outperforms SFDs with Delaunay method. While with GP, MARS and RSM models, SFDs

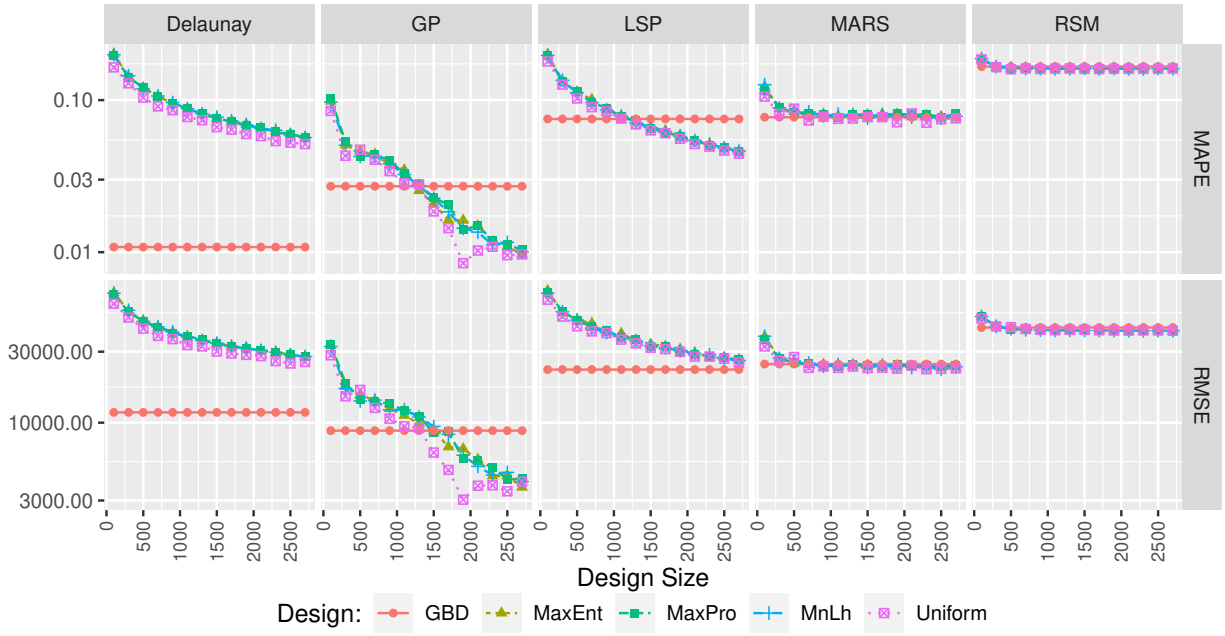


Figure 2.6: Plot of RMSE and MAPE as functions of design size in the HPC application. The GBD is given by the locations where true data were collected. The true underlying surface is the one based on the MARS fit.

can achieve the same prediction error as the GBD with smaller budget. For the LSP method, the SFDs outperform the GBD under MAPE criterion and are quite close with the GBD under the RMSE criterion.

One may notice that in Figures 2.3 to 2.6, the averaged prediction errors of SFDs are similar. One reason is that SFDs share similar properties, which are trying to fill out the whole design region to gain thorough information, so the average performance of various SFDs may perform similarly. Actually there do exist differences among different designs prediction ability if we look into each replication. However, these differences are hard to distinguish in the magnitude of those figures (i.e., if we do not zoom in).

2.4 Conclusions and Recommendations

In this paper, comparisons are conducted for synthetic data analysis and HPC variability management application to explore the prediction accuracy of GBDs and SFDs under different approximation method. Overall the prediction error decreases as the design size increases. We

find that no design outperforms all others uniformly. The GP with SFD, however, generates the best results under most scenarios. In previous work, the Delaunay method with GBD tends to have the smallest relative error (Lux et al., 2018a). In our analysis, with the maximum design budget, the best approximation method and design combination under each test function and real application is summarized in Table 2.5. From this table we can see that GP outperforms other models under both error criteria. This is consistent with the CV prediction ability analysis with the real HPC data in Table 2.4. Besides, the GP model can quantify the prediction uncertainty, which allows a better understanding of the data and model. Therefore, GP model is recommended when choosing the approximation method. If the data are collected in a grid-based manner, Delaunay method can also be considered. However, as a numerical predictive model, the Delaunay method can not provide uncertainty quantifications at predictions.

If we fix the approximation methods and look at the performance of designs, we notice that the GBD outperforms the SFD under the two numerical approximation methods (i.e., Delaunay and linear Shepard's method) for most cases. One possible reason is that GBDs have better geometric properties and these two methods depend on those properties. For the two statistical methods, MARS and RSM methods, SFDs have higher prediction accuracy compared to the GBD. One thing interesting with these two statistical methods is that, for SFDs, the increase of design size brings small improvement in prediction accuracy after the design size exceeds a certain value. That is, with MARS and RSM method, the prediction accuracy does benefit from the increase of design size in the early stage but after the design size achieves a relatively small budget, increasing design size can not guarantee a decrease in prediction error. While in contrast, under the GP surrogate, increasing the design size of SFDs always results in an obvious decrease in prediction errors.

With the best approximation method for each scenario, the design budget (i.e., the number of design points) of SFDs and GBDs to achieve the same or higher prediction accuracy are summarized in Table 2.6. We allow the design type that has weaker performance to take the maximum design budget in our analysis, and see how many runs the other type needs to have

the same or higher precision. Then we can obtain Table 2.6 by comparing the error trends from Figures 2.3 to 2.6. When making comparisons, we only consider the design sizes that we computed the RMSE and MAPE in our analysis. For example, under the four-dimensional test functions, the design sizes of GBDs and SFDs take values from $54, 160, 375, 756, 1372$ (i.e., the sizes of GBDs with design budgets $3^4 = 81, 4^4 = 256, \ldots, 7^4 = 2401$ after adjusting for the constraint $x_3 \geq x_4$). So when we fix the design type that has weaker performance at the maximum design budget 1372, we only consider the prediction accuracy of the other type with size $54, 160, ..., 1372$. Similarly for Borehole function and the HPC application. From Table 2.6 we can see that by choosing the right design type, we can save a large amount of time and cost. For the cases where SFDs are better, the number of design points needed for SFDs is about half of or less than that of the GBD to achieve the same prediction accuracy. Under the Friedman function, where the underlying surface is quite smooth, we observed that the GBD can also beat SFDs. However, GBD is not scalable to high-dimensional experiment regions. When the experiment region is of high dimension, the size of the GBD increases exponentially and building prediction models based on GBD will be time consuming even if we only consider a few unique points in each dimension. In practice, we can not guarantee that the underlying mechanism is a smooth function. So SFDs that are able to accommodate high-dimension and non-smooth functions are recommended when large numbers of input factors need to be considered in the model.

Despite the non-scalability of GBDs, they do have computational advantages when building models. Some matrix operations techniques can be used for GBDs because of their regular structure. To see the computational cost in our numerical studies, we report the running time for various designs to construct and build different models in the Appendix 2.A. We notice that GBD almost costs no time to construct, while the construction time of SFDs increases with the size increases. Except for RSM, which requires very little computational resource, GBDs do perform with advantages compared to SFDs with the same design size in building predictive models across all other four models.

In the future, it will be interesting to investigate designs that can maximize the prediction

Table 2.5: Best design and approximation method combination with different test functions under the two criteria.

Test Function	RMSE		MAPE	
	Best Method	Best Design	Best Method	Best Design
Colville	GP	SFD	GP	SFD
Friedman	GP	GBD	GP	GBD
Borehole	GP	SFD	GP	SFD
HPC application	GP	SFD	GP	SFD

Table 2.6: Design budget (i.e., the number of design points) of the GBD and SFDs to achieve a certain precision under both criteria.

Test Function and Method	RMSE		MAPE	
	SFD	GBD	SFD	GBD
Colville with GP	54	1372	375	1372
Friedman with GP	1372	375	1372	375
Borehole with GP	500	6561	500	6561
HPC application with GP	1500	2658	1300	2658

accuracy based on a good choice of surrogates in the HPC setting. For example, G-optimal designs aim to minimize the maximum element on the diagonal of the hat matrix, which has the effect of minimizing the maximum variance among the predicted values. V-optimal designs minimize the average prediction variance among a set of points. G-/V-optimal designs could be considered because they minimize variance predictions. Another direction is that in our study, we used SFDs for continuous inputs and using a heuristic way to exclude points that do not satisfy the application constraint. A more refined approach that can propose discretized designs that maintain the space filling properties and also meet the constraint are desirable for solving real application. One further step after obtaining desirable designs can be determining the system configuration that optimizes the HPC performance. For example, in Xu et al. (2020a) work, the optimal system configuration is determined as the configuration that can minimize the HPC variability while maintaining the HPC performance (i.e., the computing speed). This can provide insights in choosing system configurations in real HPC applications.

Appendix 2.A Computing Time of Different Designs and Models

Here we report the running time for various designs to construct and build different models in Figures 2.7 and 2.8, respectively. From Figure 2.7, we can see that GBD almost costs no time to construct, while the construction time of SFDs increases with the size increases. In Figure 2.8, except for RSM, which requires very little computational resource, GBDs do perform with advantages compared to SFDs with the same design size in building predictive models across all other four models.

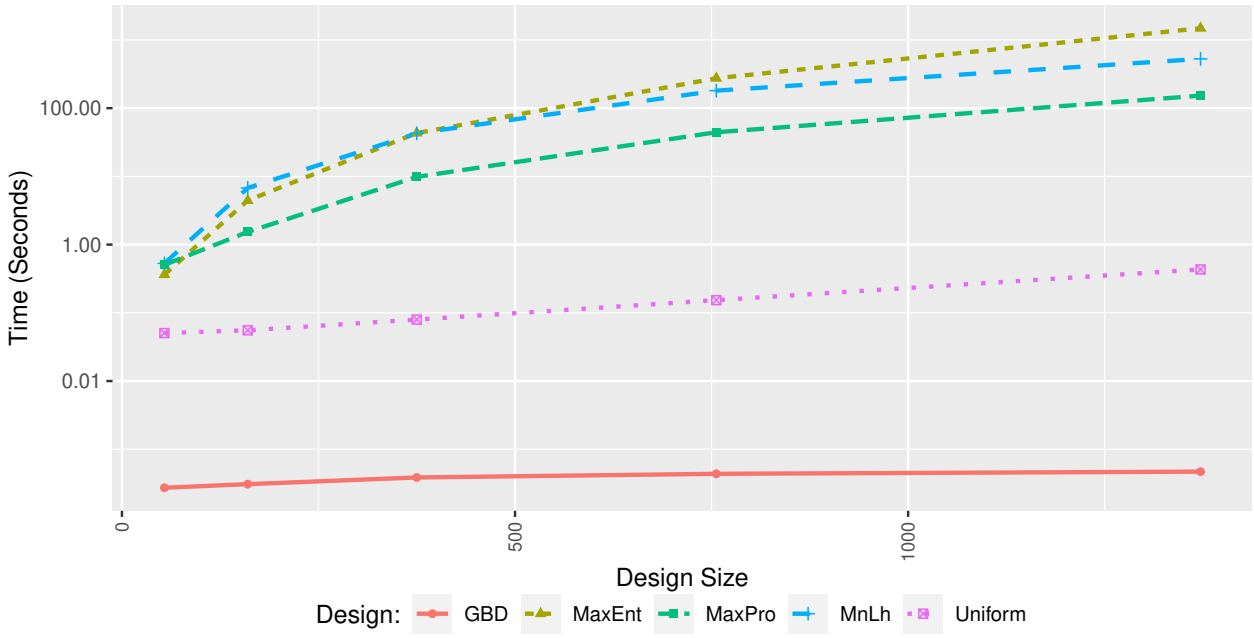


Figure 2.7: The running time versus design size to construct various designs. The y-axis is on the log scale.

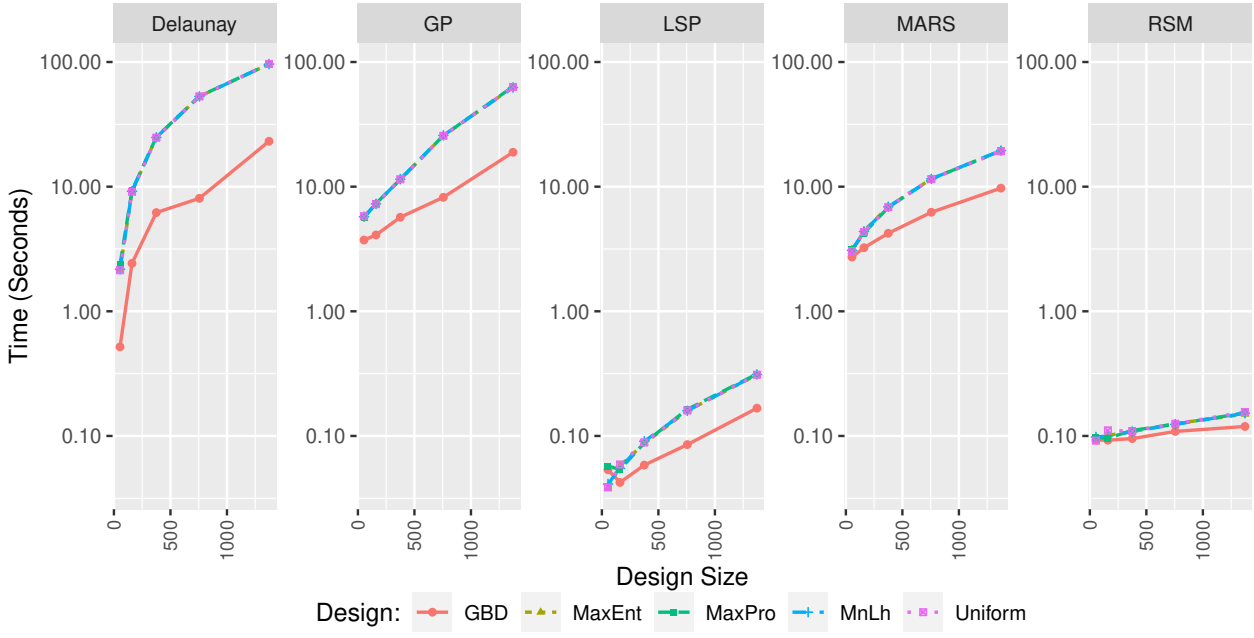


Figure 2.8: The running time versus design size for various designs to build each predictive model under the Friedman test function. The y-axis is on the log scale.

Bibliography

H. Akkan, M. Lang, and L. M. Liebrock. Stepping towards noiseless Linux environment. In *Ross' 12 Proceedings of the 2nd International Workshop on Runtime and Operating Systems for Supercomputers*, number 7, 2012. doi:10.1145/2318916.2318925.

G. E. Box and K. B. Wilson. On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 13(1):1–38, 1951.

K. W. Cameron, A. Anwar, Y. Cheng, L. Xu, B. Li, U. Ananth, J. Bernard, C. Jearls, T. Lux, Y. Hong, L. T. Waston, and A. R. Butt. MOANA: Modeling and analyzing I/O variability in parallel system experimental design. *IEEE Transactions on Parallel and Distributed Systems*, 30(8):1843–1856, 2019.

T. H. Chang, L. T. Watson, T. C. Lux, B. Li, L. Xu, A. R. Butt, K. W. Cameron, and Y. Hong. A polynomial time algorithm for multivariate interpolation in arbitrary dimension via the Delaunay triangulation. In *Proceedings of the ACMSE 2018 Conference*, number 12. ACM, 2018.

T. H. Chang, L. T. Watson, T. C. Lux, A. R. Butt, K. W. Cameron, and Y. Hong. Algorithm 1012: Delaunaysparse: Interpolation via a sparse subset of the Delaunay triangulation in medium to high dimensions. *ACM Transactions on Mathematical Software (TOMS)*, 46(4): 1–20, 2020.

C. Currin, T. Mitchell, M. Morris, and D. Ylvisaker. Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *Journal of the American Statistical Association*, 86(416):953–963, 1991.

B. Delaunay. Sur la sphère vide. a la mémoire de georges voronoï. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na no. 6*, pages 793–800, 1934.

D. Draguljić, T. J. Santner, and A. M. Dean. Noncollapsing space-filling designs for bounded nonrectangular regions. *Technometrics*, 54(2):169–178, 2012.

D. Dupuy, C. Helbert, and J. Franco. DiceDesign and DiceEval: Two R packages for design and analysis of computer experiments. *Journal of Statistical Software*, 65(11):1–38, 2015. URL `http://www.jstatsoft.org/v65/i11/`.

K.-T. Fang, D. K. Lin, P. Winker, and Y. Zhang. Uniform design: theory and application. *Technometrics*, 42(3):237–248, 2000.

J. H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–67, 1991.

J. H. Friedman, E. Grosse, and W. Stuetzle. Multidimensional additive spline approximation. *SIAM Journal on Scientific and Statistical Computing*, 4(2):291–301, 1983.

M. Giampapa, T. Gooding, T. Inglett, and R. W. Wisniewski. Experiences with a lightweight supercomputer kernel: Lessons learned from blue gene's CNK. In *SC'10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–10. IEEE, 2010.

S. Golchi and J. L. Loeppky. Monte Carlo based designs for constrained domains. *arXiv:1512.07328*, 2015.

P. W. Goldberg, C. K. Williams, and C. M. Bishop. Regression with input-dependent noise: A Gaussian process treatment. In *Advances in Neural Information Processing Systems*, pages 493–499, 1998.

R. B. Gramacy. laGP: Large-scale spatial modeling via local approximate Gaussian processes in R. *Journal of Statistical Software*, 72(1):1–46, 2016. doi: 10.18637/jss.v072.i01.

Y. Hung, Y. Amemiya, and C.-F. J. Wu. Probability-based Latin hypercube designs for slid-rectangular regions. *Biometrika*, 97(4):961–968, 2010.

IOzone projects contributors. IOzone filesystem benchmark, 2016. http://www.iozone.org/.

M. E. Johnson, L. M. Moore, and D. Ylvisaker. Minimax and maximin distance designs. *Journal of Statistical Planning and Inference*, 26(2):131–148, 1990.

R. T. Johnson, D. C. Montgomery, and J. Bradley. An empirical study of the prediction performance of space-filling designs. *International Journal of Experimental Design and Process Optimisation*, 2(1):1–18, 2011.

V. R. Joseph, E. Gul, and S. Ba. Maximum projection designs for computer experiments. *Biometrika*, 102(2):371–380, 2015.

R. Lekivetz and B. Jones. Fast flexible space-filling designs for nonrectangular regions. *Quality and Reliability Engineering International*, 31(5):829–837, 2015.

R. Li, D. K. Lin, and Y. Chen. Uniform design: design, analysis and applications. *International Journal of Materials and Product Technology*, 20(1-3):101–114, 2004.

T. C. Lux, L. T. Watson, T. H. Chang, J. Bernard, B. Li, L. Xu, G. Back, A. R. Butt, K. W. Cameron, and Y. Hong. Predictive modeling of I/O characteristics in high performance computing systems. In *Proceedings of the High Performance Computing Symposium*, number 8. Society for Computer Simulation International, 2018a.

T. C. Lux, L. T. Watson, T. H. Chang, J. Bernard, B. Li, X. Yu, L. Xu, G. Back, A. R. Butt, and K. W. Cameron. Novel meshes for multivariate interpolation and approximation. In *Proceedings of the ACMSE 2018 Conference*, number 13. ACM, 2018b.

M. D. McKay, R. J. Beckman, and W. J. Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

S. Milborrow. earth: Multivariate adaptive regression splines. 2019. R package version 5.1.2.

M. D. Morris and T. J. Mitchell. Exploratory designs for computational experiments. *Journal of Statistical Planning and Inference*, 43(3):381–402, 1995.

R. H. Myers. Response surface methodology—current status and future directions. *Journal of Quality Technology*, 31(1):30–44, 1999.

R. H. Myers, D. C. Montgomery, G. G. Vining, C. M. Borror, and S. M. Kowalski. Response surface methodology: a retrospective and literature survey. *Journal of quality technology*, 36(1):53–77, 2004.

M. T. Pratola, O. Harari, D. Bingham, and G. E. Flowers. Design and analysis of experiments on nonconvex regions. *Technometrics*, 59(1):36–47, 2017.

J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423, 1989.

D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM National Conference*, pages 517–524. ACM, 1968.

M. C. Shewry and H. P. Wynn. Maximum entropy sampling. *Journal of Applied Statistics*, 14(2):165–170, 1987.

W. I. Thacker, J. Zhang, L. T. Watson, J. B. Birch, M. A. Iyer, and M. W. Berry. Algorithm 905: Sheppack: Modified Shepard algorithm for interpolation of scattered multivariate data. *ACM Transactions on Mathematical Software (TOMS)*, 37(3):1–20, 2010.

D. F. Watson. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *The Computer Journal*, 24(2):167–172, 1981.

L. Xu, T. Lux, T. Chang, B. Li, Y. Hong, L. Watson, A. Butt, D. Yao, and K. Cameron. Prediction of high-performance computing input/output variability and its application to optimization for system configurations. *Quality Engineering, in press*, 2020a. arXiv:2012.07915.

L. Xu, Y. Wang, T. C. Lux, T. Chang, J. Bernard, B. Li, Y. Hong, L. T. Watson, and K. W. Cameron. Modeling I/O performance variability in high-performance computing systems using mixture distributions. *Journal of Parallel and Distributed Computing*, 139:87–98, 2020b.

Chapter 3 Building Degradation Index with Variable Selection for Multivariate Sensory Data

Abstract

The modeling and analysis of degradation data have been an active research area in reliability engineering for reliability assessment and system health management. As the sensor technology advances, multivariate sensory data are commonly collected for the underlying degradation process. However, most existing research on degradation modeling requires a univariate degradation index to be provided. Thus, constructing a degradation index for multivariate sensory data is a fundamental step in degradation modeling. In this section, we propose a novel degradation index building method for multivariate sensory data with censoring. Based on an additive nonlinear model with variable selection, the proposed method can handle censored data, and can automatically select the informative sensor signals to be used in the degradation index. The penalized likelihood method with adaptive group penalty is developed for parameter estimation. We demonstrate that the proposed method outperforms existing methods via both simulation studies and analyses of the NASA jet engine sensor data.

Key Words: Adaptive LASSO; General Path Model; Prognostics; Sensor Selection; Splines; System Health Monitoring.

3.1 Introduction

3.1.1 Background

Degradation data have been widely used in reliability engineering for reliability and system health assessment. There are many examples of products and systems that provide degradation data, such as the loss of light output from a light-emitting diode (LED) array, the power output decrease of photovoltaic arrays, and the vibration from a worn bearing in a wind turbine. The data type is typically a repeated measurement of the degradation index (e.g., the loss of light output from an LED array) with a monotone increasing (decreasing) trend. The general path model (GPM) is one class of methods for degradation modeling. In the typically modeling framework of the GPM, a soft failure occurs when the degradation level reaches a predefined failure threshold. The stochastic process model (SPM) framework is also popular in the degradation literature. In the SPM framework, the distribution of the degradation incremental levels is modeled by a Gaussian distribution or other distributions such as the gamma or inverse Gaussian distribution. In these two frameworks, most existing research on degradation modeling assumes that the degradation index is well defined and can be measured over time.

Different from traditional degradation data, modern sensor technology allows one to collect multi-channel sensor data that are related to an underlying degradation process. This is very common in many modern engineering systems. One example is the motivating data in our study, the multi-channel jet engine data (Saxena and Goebel, 2008). In the jet engine data, multiple sensors such as temperatures and pressures of module parts are recorded while the engine is operating. The details about the dataset are introduced in Section 3.5.1. In such multi-channel sensor data, any single channel may not be sufficient to represent the underlying degradation process. Without a degradation index, most existing methods in the aforementioned frameworks will not be applicable in the analysis of such sensor signal data. Thus, building a degradation index is an important step in utilizing the sensor data in

degradation analysis.

There are several key considerations when building the degradation index based on multi-sensory data. In real engineering applications, it is typically complicated how each sensor signal reflects the overall degradation. In such a case, a linear form for the effect of each sensor signal may not be adequate, which motivates us to consider nonlinear functional forms of the individual sensor signal. In most cases, not all sensors collected are useful in representing the underlying degradation process. Thus, it is important to automatically select more useful and relevant sensors to build the degradation index. Besides, censored data are quite common in reliability engineering applications. It is ideal to use both exact failure and censored time data in the training of the model. In addition, when considering the prediction accuracy, the risks of being false positive and false negative are quite different especially for failures of important systems such as jet engines. An asymmetric loss function is desirable during the training of the degradation index model.

Motivated by the these considerations, this section aims to develop a flexible method for constructing a degradation index from multi-channel signals with automatic variable selection while accounting for censored data and nonlinear relationships.

3.1.2 Literature Review and Contribution of This Work

Regarding GPM for degradation data, the classic reference book is Meeker and Escobar (1998). For the SPM for degradation data, the Wiener process, gamma process, and the inverse-Gaussian process have been used (e.g., Ye and Chen, 2014). Ye and Xie (2015) provided a comprehensive review of degradation models. These models are suitable for the cases of one-dimensional degradation data. On the other hand, Meeker and Hong (2014) and Hong et al. (2018) outlined some opportunities for using sensor data in reliability modeling and analysis.

Regarding the recent development of degradation modeling, Zheng et al. (2021) considered the joint modeling of degradation data and lifetime data using the proportional hazards model. Wang et al. (2021) developed a Wiener process model to describe heterogeneity in degradation data. Chen et al. (2022) developed an integration method of multi-source accel-

erated degradation testing for reliability evaluation. Duan et al. (2022) proposed an adaptive monitoring scheme based on the hidden Markov model to predict the faults of systems with hidden degradation processes. Kumar et al. (2022) proposed a health indicator based on the state-space model to access the degradation process. Wang et al. (2022) constructed a stochastic multi-phase model for multi-component systems. In summary, the modeling and analysis of degradation data is an active area, which is carried out with the availability of degradation measurements.

In the area of degradation index building, Liu et al. (2013) proposed a data-level fusion model for developing composite health indices for degradation modeling and prognostic analysis. Follow-up work includes Fang et al. (2017); Song et al. (2018); Chehade et al. (2018); Song and Liu (2018). Recently, Kim et al. (2019) proposed a latent linear model that constructs a health index via multiple sensors and selects informative sensors. Wei et al. (2021) proposed a dynamic conditional variational autoencoder to learn the health index. Kim et al. (2021) developed a sensor selection framework that can be applied to neural network-based models and improved in interpretability of neural network models. Li et al. (2022) developed a data-model interactive prediction method for multi-sensor monitored stochastic degrading devices. However, existing methods can not handle censored data and conduct variable selection at the same time, which is the gap that this section aims to fill.

Regarding variable selections, the least absolute shrinkage and selection operator (LASSO) penalty for regression-type problems was studied in Tibshirani (1996). Zou (2006) developed the adaptive LASSO to ensure variable selection consistency, and Yuan and Lin (2006) considered the group LASSO for efficient variable selections with meaningful interpretation. Under the context of this section, we focus on the adaptive group LASSO to select the important sensors.

This section proposes a novel framework based on the cumulative exposure model to build the degradation index with multivariate sensors, which is applicable to many engineering systems equipped with sensors. The contributions of this section are unique from existing literature. Specifically, the proposed framework can include censored failure time information

to train the model, which can preserve the information provided by the data. Besides, the proposed framework can automatically select the most informative sensors related to the degradation process using the adaptive group LASSO penalty. To enable sufficient flexibility on the nonlinear relationship between sensors and degradation path, spline-based methods are used to describe the contribution of each sensor signal in the cumulative exposure. In addition, the proposed model can automatically guarantee the monotonicity of the degradation index.

3.1.3 Overview

The rest of this section is organized as follows. Section 3.2 introduces the framework for degradation index building based on time-to-event data with multivariate signals. Section 3.3 presents the parameter estimation with variable selection. Section 3.4 uses simulation to study the performance of the proposed methods. The motivating example is used to illustrate the developed method in Section 3.5. Section 3.6 contains conclusions and discusses areas for future research.

3.2 Building Degradation Index

3.2.1 Degradation Index

Consider sensor data with p degradation signals. Let $\boldsymbol{x}(t) = \{[x_1(s), \cdots, x_p(s)]^\top : 0 \leq s \leq t\}$ be the collection of information for the p signals from a unit, where $x_j(s)$ is the jth the dynamic covariate information at time s, $j = 1, \ldots, p$. For the jet engine data, those $x_j(\cdot)$'s can be signals from various sensors and recorded operating conditions. We use the cumulative exposure model (aka, the cumulative damage model) to construct the degradation index (e.g., see Hong and Meeker, 2013). The cumulative exposure model is useful when the damage is accumulative, which is the case for many engineering systems. For example, the damage to an automobile tire is accumulative due to wear out.

The cumulative exposure $u(t)$ for the covariate history $\boldsymbol{x}(t)$ is defined as,

$$u(t) = \int_0^t h\left\{\sum_{j=1}^{p} f_j[x_j(s); \boldsymbol{\beta}_j]\right\} ds, \tag{3.1}$$

where $f_j[x_j(t); \boldsymbol{\beta}_j]$ represents the nonlinear effect function of the signal $x_j(t)$ on the degradation index and $\boldsymbol{\beta}_j$ are parameters that represent the influence of the covariate on the cumulative exposure. Note that $f_j(\cdot; \boldsymbol{\beta}_j)$ is defined on the range of the covariate, not on time t. Here, $h(z)$ maps the effect to a positive exposure, and the integral is from 0 to t, which guarantees that $u(t)$ is monotonically increasing and utilizes all the information in the history up to time t. In this section, we use the function $h(z) = \log[1 + \exp(z)]/\log(2)$, which maps the input $z \in (-\infty, \infty)$ to an output that takes value $h(z) \in (0, \infty)$. In literature, functions like $h(z) = \exp(z)$ are used, but the function $h(z)$ used here is numerically more stable.

When modeling the nonlinear effect of the jth signal $f_j(\cdot; \boldsymbol{\beta}_j)$ in $u(t)$, it is desirable to make the function form flexible enough to capture potential nonlinearity in sensors' effect. Therefore, we use a non-negative spline function, called M-splines (e.g., Ramsay, 1988). For the jth signal, let

$$f_j[x_j(t); \boldsymbol{\beta}_j] = \sum_{k=1}^{m} \beta_{jk} \gamma_{jk}[x_j(t)],$$

where $\{\gamma_{jk}[x_j(t)] : k = 1, \ldots, m\}$ are spline basis of the M-spline of order three with $(m-3)$ interior knots and $\boldsymbol{\beta}_j = (\beta_{j1}, \ldots, \beta_{jm})'$ are the coefficients of the basis. Let $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^\top, \ldots, \boldsymbol{\beta}_p^\top)^\top$ be the parameters for all p signals. Figure 3.1(a) shows the basis of the M-spline of order 3 with 7 interior knots ($m = 10$). The magnitude of $\boldsymbol{\beta}$ can be used to identify which signals are more important for the degradation index $u(t)$. Although we used the M-splines for implementation, other splines such as the B-splines can also be used, because most spline basis functions are quite flexible.

Note that $u(0) = 0$, and $u(t)$ is always monotonically increasing, which are the two properties of $u(t)$ that satisfy the characteristics of a degradation index as introduced in Section 3.1.1. Thus, in this section, we propose to use $u(t)$ as a degradation index.

3.2.2 Modeling Time to Failure and Degradation Index

Based on the cumulative exposure model, a unit fails at time T when the cumulative exposure reaches a random threshold U (e.g., Hong and Meeker, 2013). That is

$$U = u(T), \tag{3.2}$$

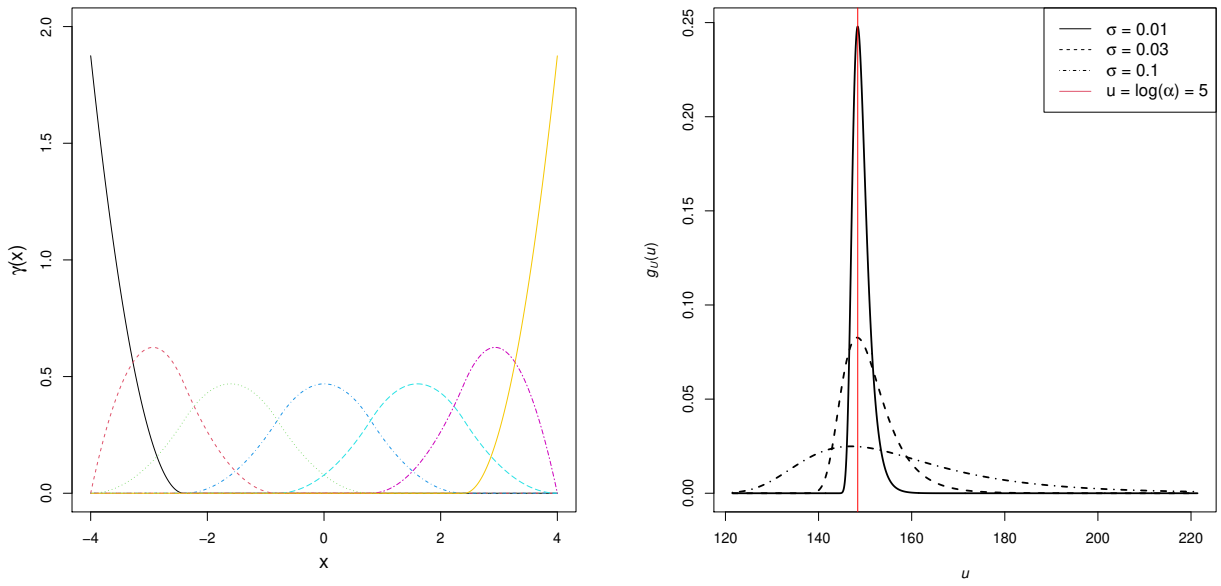
where the function $u(\cdot)$ is defined in (3.1). We model $\log(U)$ by the largest extreme value (LEV) distribution (e.g., Meeker and Escobar, 1998) with the location parameter $\log(\alpha)$ and the scale parameter $\sigma > 0$. The cumulative distribution function (cdf) and the probability density function (pdf) of U can be expressed as

$$G_U(u;\alpha,\sigma) = \Phi_{\text{LEV}}\left[\frac{\log(u)-\log(\alpha)}{\sigma}\right] \text{ and } g_U(u;\alpha,\sigma) = \frac{1}{\sigma u}\phi_{\text{LEV}}\left[\frac{\log(u)-\log(\alpha)}{\sigma}\right],$$

where $\Phi_{\text{LEV}}(x) = \exp[-\exp(-x)]$, and $\phi_{\text{LEV}}(x) = \exp[-x-\exp(-x)]$. As an illustration, Figure 3.1(b) shows the pdf of LEV distributions with $\alpha = \exp(5)$ and $\sigma = 0.01$, 0.03, and 0.1.

The parameter α in the LEV distribution can be used as the *target failure threshold* for the degradation index in (3.1), which is pre-fixed. This is because we want those failed units with their degradation indexes $u(t)$ centered around the failure threshold when they fail. The scale parameter σ serves as a measurement of how small the difference between $u(t)$ and the threshold α is if the unit is failed. Because we do not observe U, in practice, we use the following threshold rule. That is, if $u(t) \geq \alpha$, we say a unit fails, and if $u(t) < \alpha$, we say the unit is surviving (i.e., the operation status is normal). Through re-scaling, we can map the degradation index to any range that is desirable for the particular application. For example, using $100u(t)/\alpha$, one can map the normal range of the degradation index into $[0, 100]$.

For building the degradation index, a suitable property of the LEV distribution is its skewness to the right as shown in Figure 3.1(b). Thus, we allow the larger difference between $u(t)$ and α on the positive side so that when making predictions, we can potentially reduce



(a) M-spline Basis (b) Probability Density Function

Figure 3.1: The plots show the construction of the M-spline of order 3 with 7 interior knots (a), and the pdf of LEV distributions with $\alpha = \exp(5)$ and various values of σ (b).

false negative error (i.e., falsely predicting failed units as censored). This is desirable because, in our study, we focus on important equipment like jet engine. Falsely predicting a failed unit as censored (i.e., false negative) can result in great losses or even severe accidents. Through the relationship between T and U as shown in (3.2), the cdf and the pdf of T are, $G_T(t;\boldsymbol{\beta}) = \Phi_{\text{LEV}}\left(\{\log[u(t)] - \log(\alpha)\}/\sigma\right)$, and $g_T(t;\boldsymbol{\beta}) = \{u'(t)/[\sigma u(t)]\}\phi_{\text{LEV}}\left(\{\log[u(t)] - \log(\alpha)\}/\sigma\right)$, where the derivative of of $u(t)$ is $u'(t) = h\left(\sum_{j=1}^{p}\sum_{k=1}^{m}\beta_{jk}\gamma_{jk}[x_j(t)]\right)$.

3.3 Parameter Estimation

3.3.1 Log-likelihood with Adaptive Group LASSO Penalty

We first introduce some notation for the data. From the sensor data, there are n units with p degradation signals. Three sets of observable data are taken into consideration, which include failure-time data, censoring indicator, and multivariate degradation signals. Let t_i be the failure time of the ith unit and let $\boldsymbol{x}_i(t) = \{[x_{i1}(s), \cdots, x_{ip}(s)]^\top : 0 \le s \le t\}$ be the collection of p signals for the ith unit prior to and at time t. Here $x_{ij}(s)$ is the jth observed dynamic

covariate information at time s, $i = 1, \ldots, n$, $j = 1, \ldots, p$, and $0 \leq s \leq t_i$. The observed censoring indicator of the ith unit is denoted by δ_i, where δ_i equals to 1 if the ith unit fails and 0 otherwise. Then, the collected information from unit i is denoted by $\{t_i, \delta_i, \boldsymbol{x}_i(t_i)\}$, where $i = 1, \ldots, n$, and let $\mathcal{H} = \{\{t_i, \delta_i, \boldsymbol{x}_i(t_i)\} : i = 1, \ldots, n\}$ be the collection of the data.

To estimate the parameters $\boldsymbol{\beta}$, the likelihood for the ith unit is expressed as,

$$L_i(\boldsymbol{\beta}) = \left(\left[\frac{u'(t_i)}{\sigma u(t_i)} \right] \phi_{\text{LEV}} \left\{ \frac{\log[u(t_i)] - \log(\alpha)}{\sigma} \right\} \right)^{\delta_i} \times \left(1 - \Phi_{\text{LEV}} \left\{ \frac{\log[u(t_i)] - \log(\alpha)}{\sigma} \right\} \right)^{1-\delta_i}.$$

Note that this is not an actual likelihood, because the "response" $u(t_i)$ is unknown. Instead, the log-likelihood function serves as a loss function for the estimation purpose. The log-likelihood function for the ith unit is

$$\begin{aligned} l_i(\boldsymbol{\beta}) = \quad & \delta_i \left\{ \log[u'(t_i)] - \log(\sigma) - \log[u(t_i)] - \log \left[\frac{\alpha}{u(t_i)} \right]^{1/\sigma} - \left[\frac{\alpha}{u(t_i)} \right]^{1/\sigma} \right\} \\ & + (1 - \delta_i) \log \left(1 - \exp \left\{ - \left[\frac{\alpha}{u(t_i)} \right]^{1/\sigma} \right\} \right). \end{aligned} \tag{3.3}$$

Hence, the overall log-likelihood function is $l(\boldsymbol{\beta}|\boldsymbol{H}) = \sum_{i=1}^{n} l_i(\boldsymbol{\beta})$. Then, we can obtain the maximum likelihood (ML) estimates of $\boldsymbol{\beta}$ by maximizing the overall log-likelihood function.

Note that $\boldsymbol{\beta}$ are the unknown parameters, and α and σ are set as constants because the "response" $u(t_i)$ is unknown. From Figure 3.1(b) and the perspective of the likelihood function, the smaller value of σ is, the larger value of the pdf of a failure is. Besides, inside the $\phi_{\text{LEV}}(\cdot)$ and $\Phi_{\text{LEV}}(\cdot)$, we have the $\log(u) - \log(\alpha) = \log(u/\alpha)$ term. In this case, α and $\boldsymbol{\beta}$ can take different combinations of values so that u/α keeps the same, which yields the same likelihood. Thus, it is necessary to let α and σ be given constants to avoid identifiability issues.

As discussed in Section 3.2.2 and from (3.3), it can be seen that $u(t_i) \approx \alpha$ if the ith unit is a failure, and $u(t_i) < \alpha$ if the ith unit is censored. That is, the role of α is to set the target failure threshold. Then, the role of σ is to measure how close the difference between $u(t_i)$ and α. So for the value of σ, ideally, we want σ to be a small enough value to allow the

degradation index of failed units to end close to the target failure threshold α. For the value of α, in theory, we can set any value for the target failure threshold α. Then $\boldsymbol{\beta}$ can adjust correspondingly to provide a degradation index between 0 and α. In applications, setting α close to the mean of failure times helps the convergence of the estimation algorithm.

Although multiple sensors are available to assess the degradation process, not every sensor collected has a significant contribution. So we integrate variable selection in the model to find out informative sensors. Since there are multiple M-splines basis to represent one sensor, we want to penalize coefficient parameters associated with one sensor simultaneously when that sensor does not contribute. Therefore, we adopt the adaptive group LASSO method in Huang et al. (2010) to conduct variable selection. The adaptive group LASSO approach penalizes parameters in the same group simultaneously. In our model, the parameters in M-splines for the same variable are treated to be in the same group. That is, $\boldsymbol{\beta}_i$ and $\boldsymbol{\beta}_j$ are in different groups for any $i \neq j$. From the perspective of variable selection, the jth sensor variable has no effect on $u(t)$ if all elements in $\boldsymbol{\beta}_j$ are significantly small. The adaptive group LASSO considers the penalties on different grouped parameters have different effects. Let ω_j be a given weight of the penalty for the jth variable, where $\omega_j \geq 0$ and $j = 1, \ldots, p$. Then, the penalized negative log-likelihood function is

$$\mathcal{L}(\boldsymbol{\beta}; \lambda) = -l(\boldsymbol{\beta}|\boldsymbol{H}) + \lambda \sum_{j=1}^{p} \omega_j ||\boldsymbol{\beta}_j||_2, \tag{3.4}$$

where $\lambda \geq 0$ is a tuning parameter and $||\boldsymbol{\beta}_j||_2 = \sqrt{\sum_{k=1}^{m} \beta_{jk}^2}$ is the L_2 norm of the vector $\boldsymbol{\beta}_j$. Typically, the weights are given by setting

$$\omega_j = \begin{cases} ||\widetilde{\boldsymbol{\beta}}_j||_2^{-\gamma} & \text{if } ||\widetilde{\boldsymbol{\beta}}_j||_2 > 0 \\ \infty & \text{if } ||\widetilde{\boldsymbol{\beta}}_j||_2 = 0, \end{cases} \tag{3.5}$$

where $\widetilde{\boldsymbol{\beta}}_j$ is an estimate of $\boldsymbol{\beta}_j$ and $\gamma \geq 0$ is a hyper-parameter. Here we follow the practice in Huang et al. (2010) and define $\infty \cdot 0 = 0$. That means the model does not select sensor j if

its coefficient estimates L_2 norm is zero (i.e., $||\widetilde{\boldsymbol{\beta}}_j||_2 = 0$).

3.3.2 Optimization of Objective Function

The optimization of (3.4) is challenging. Here we discuss some strategies used in the optimization of the objective functions. To optimize the objective function, we use the Nelder-Mead algorithm in the R package *nloptr* (Johnson, 2020). Due to the model complexity and non-convexity, there exist multiple local optimal of the coefficients. However, in our study, the main focus is to predict the status accurately. Therefore, even though there are local optimal points, it is of less concern as long as the optimization allows us to predict units' status accurately.

One thing to notice is the influence of σ value in the optimization procedure. If σ is prefixed at a small value (e.g., 0.01) at the beginning of the optimization, the algorithm could be easily trapped at local optima. As shown in Figure 3.1(b), when σ is small, the pdf of LEV is highly concentrated around the location parameter $\log(\alpha)$. That means a unit with $\log[u(t_i)]$ at the event time that is close to the threshold $\log(\alpha)$ has a high probability. While a unit which degradation index at the event time is far away from the threshold has almost zero probability, thus, its contribution to the likelihood function is small. During the optimization process, with a small σ, it is possible that the $\boldsymbol{\beta}$ is updated to an estimation that some units' degradation paths get almost 0 probability. So the contribution of these units to the objective function is neglected in the following updates of the $\boldsymbol{\beta}$ estimation. Only units with $\log[u(t_i)]$ that are relatively close to $\log(\alpha)$ have the chance to further move close to the threshold.

One approach to avoid the local optima is to set a larger value for σ at the beginning of the optimization process, and then decrease it gradually to the prefixed lower bound. By setting σ to a large value, say $\sigma = 1$, the information of all units is equally treated, regardless of the distance between the value of $\log[u(t_i)]$ and the location parameter. After a certain number of iterations, the unit's $\log[u(t_i)]$ moves closer to $\log(\alpha)$, then we can decrease the value of σ by a small amount and update $\boldsymbol{\beta}$ estimation. Repeating this step until σ decreased to the fixed constant can help to avoid the local optima problem. Instead of manually determining

a sequence of σ to decrease, we add it to the optimization parameters.

Let σ_l be the prefixed lower bound of σ. We consider the transformation $\log(\sigma^*) = \log(\sigma - \sigma_l)$. Thus, $\sigma = \exp[\log(\sigma^*)] + \sigma_l$. Then we optimize $\log(\sigma^*)$ and $\boldsymbol{\beta}$ simultaneously. This transformation can always impose a lower bound for the estimation of σ. Although we include $\log(\sigma^*)$ in the parameter estimation, the purpose is not to obtain an estimation of $\log(\sigma^*)$. The reason is that σ is not identifiable and it always becomes smaller to allow a larger likelihood. Via the iterations, it will get to its lower bound eventually. Thus, we include σ in the parameter estimation so that it can smoothly decrease and help to avoid the local optima of $\boldsymbol{\beta}$ estimation.

With a larger value of σ in the early stage of the iterations, the benefit of the asymmetric property of the LEV distribution is not evident. We introduce the following remedy to ensure the estimates of $u(t_i)$ are moving towards the right direction during the early stage of the optimization iterations. As discussed above, when building a degradation index, it is desirable that $u(t_i) = \alpha$ if ith unit fails and $u(t_i) < \alpha$ if ith unit is censored. Thus, we further impose those two constraints on the objective function. That is, we modify the objective function as,

$$\mathcal{M}(\boldsymbol{\beta}, \lambda) = \mathcal{L}(\boldsymbol{\beta}; \lambda) + \eta \sum_{i=1}^{n} \delta_i \{[\alpha - u(t_i)]\}^2 + (1 - \delta_i)\{[u(t_i) - \alpha]_+\}^2. \tag{3.6}$$

The positive part function is $[u(t_i) - \alpha]_+ = \max(u(t_i) - \alpha, 0)$ and the penalty η is non-negative. Therefore, to encourage the estimated $u(t)$ satisfying the degradation index characteristic (i.e., $u(t_i) = \alpha$ if ith unit fails and $u(t_i) < \alpha$ if ith unit is censored) as well as perform variable selection, we work with the objective function $\mathcal{M}(\boldsymbol{\beta}, \lambda)$ as shown in (3.6).

3.3.3 Determining Tuning Parameter

In parameter estimation, we need to determine the tuning parameter λ in the objective function (3.6). The k-fold cross-validation (k-CV) approach is used. Because the main goal of our degradation index model is to accurately predict the status of testing units, especially for the failed units, we use both the false negative error rate and total error rate as the criterion to select the tuning parameters. Let FNR and FPR be the averaged false negative

and positive error rates across the k folds, respectively. Then the averaged total error rate is $\mathrm{TER} = \mathrm{FNR} + \mathrm{FPR}$. For a sequence of values for λ, denoted by $\{\lambda_1, \ldots, \lambda_q\}$, the corresponding error rates are $\{\mathrm{TER}_1, \ldots, \mathrm{TER}_q\}$, and $\{\mathrm{FNR}_1, \ldots, \mathrm{FNR}_q\}$. Let $k_f = \arg\min_b \mathrm{FNR}_b$ be the index of tuning parameter that minimizes FNR. We want to select the tuning parameter λ so that FNR is minimized, while TER is kept at a relatively low level. That means we do not want to sacrifice FPR to achieve the smallest FNR. Therefore, the selected tuning parameter is λ_s, of which the index s is determined by

$$s = \begin{cases} \arg\min_b \mathrm{FNR}_b & \text{if } \mathrm{TER}_{k_f} \leq 0.2, \\ \arg\min_b \mathrm{TER}_b & \text{otherwise.} \end{cases} \tag{3.7}$$

In this way, when a λ minimizes FNR at the cost of FPR, we will switch to the λ that minimizes TER to achieve a balance between FPR and FNR.

The parameters η, γ, and σ_l are treated as hyper-parameters. We do not tune η because the role of the η penalty term is to help $u(t)$ move towards α at the beginning stage. After $u(t)$ is close to α, the effect of the asymmetric property of LEV kicks in and the shrinkage of σ towards σ_l serves the same role. Therefore, we only include η penalty term to help the algorithm converge and set a moderate large value for η. In particular, η is set to be 5. We fix the hyper-parameter $\gamma = 2$ in the calculation of the weights in (3.5), which is as a common practice.

The lower bound of scale parameter σ_l is set to be 0.01 in this section. In practice, the σ_l can be set by users. Ideally, the smaller value of σ_l has better performance of the degradation index. However, the smaller value takes more computation time. In our cases, we set it as $\sigma_l = 0.01$, because when $\alpha = \exp(5)$, the 99% tolerance interval of the LEV distribution will be $(145.9, 156.5)$, using the quantile of the LEV distribution. This interval is $(-2\%, 5\%)$ around the target $\exp(5)$, which is narrow enough. Regarding the selection of the number of splines, the strategy is to set it be large enough to allow flexibility. Then we use the adaptive group LASSO penalty to regularize the estimation and let the penalty shrink those less important

sensors.

3.3.4 Parameter Estimation Procedure

In this section, we describe how to obtain the estimates $\widehat{\boldsymbol{\beta}}$ based on the training set, using the adaptive group LASSO procedure. With initial estimates $\widetilde{\boldsymbol{\beta}}_j$, we obtain the weights ω_j, as in (3.5) for $j = 1, \cdots, p$. We first apply the k-CV in Section 3.3.3 to obtain the tuning parameter λ_s as in (3.7). The average of the $\boldsymbol{\beta}$ estimates from each fold in k-CV can also provide a warm starting point for the final estimate of $\boldsymbol{\beta}$. Similar ideas are also used in literature (e.g., Mazumder et al., 2011). With the best selected λ_s, the warm starting point for $\boldsymbol{\beta}$, and the entire training set, we apply the adaptive group LASSO procedure to obtain the final estimates of $\boldsymbol{\beta}$ in (3.6), which is denoted by $\widehat{\boldsymbol{\beta}}$. The statistical inference and variable selection results can be obtained based on $\widehat{\boldsymbol{\beta}}$.

The question remains how to find the initial estimates $\widetilde{\boldsymbol{\beta}}_j$'s. Huang et al. (2010) suggested that one can use the group LASSO procedure (i.e., without adaptive weights) to find the initial estimates of parameters. For the group LASSO estimates, the objective function in (3.6) is simplified by setting $\omega_j = 1, j = 1, \cdots, p$. Similarly, we apply the k-CV to find the best tuning parameter $\widetilde{\lambda}_s$ for the group LASSO, and use the averaged estimates as the warm starting points for the final estimates of the group LASSO procedure, denoted by $\widetilde{\boldsymbol{\beta}}_j, j = 1, \cdots, p$.

For the initial values of the k-CV of the group LASSO procedure, we have to use cold starting points as we do not have much information about the parameters at this step. We randomly select the starting points that satisfy some desirable properties in the degradation scenario. For example, the starting points need to allow the minimum of $\log[u(t_i)]$'s of the failed units to be larger than the maximum of $\log[u(t_i)]$'s of the censored units.

3.4 Simulation Study

In order to evaluate the performance of the proposed degradation index building method, we use various simulated scenarios to compare our model with the model without variable

selection model assumes a linear relationship between sensors and degradation index. We investigate the average times that our model properly selects variables and the average prediction accuracy (i.e., correctly predicts the status of a unit as a failure or censored one).

3.4.1 Simulation Setup and Procedure

In this simulation study, we want to generate datasets similar to the jet engine data to demonstrate the performance of our degradation index building framework. To generate the simulation data, we first consider the signal sensors. In the simulation study, we generate 10 sensor signals similar to the jet engine signals within time interval $[0, 350]$. We assume each signal is a function of time with some variations. That is $X_j(t) = g_j(t) + \epsilon_j(t), j = 1, \ldots, 10$. The function $g_j(t)$ can take forms such as constant, linear, quadratic, log functions, and the error term $\epsilon_j(t)$ follows a normal or a uniform distribution. Figure 3.2 presents the example of simulated signals for 10 units. The signals can be increasing, decreasing, or randomly fluctuate over time. After obtaining the sensor information, the basis functions of the M-spline with 2 interior knots are constructed based on the signals. So we have 50 coefficient parameters for the simulated data. To test the variable selection capability of our method, we assume that 5 out of 10 signals cause the units to fail, and the rest 5 have no effect on the failure process. Hence, we set the values of parameters of the first 5 signals to have effects on the degradation index. Among the five signals with effect, we assume the second and fourth are linear functions of time, the first and the fifth are quadratic functions of time, and the third one is assumed to follow a normal distribution. With the true parameter coefficients $\boldsymbol{\beta}$ and the simulated sensor signal history, we can compute $u(t)$ using (3.1).

The next step is to generate the failure-time data. The failure-time data are generated by the following steps:

(a) **Generate the censoring time:** Let the time to failure be $T = \min\{C, 350\}$, where C follows a Weibull distribution. The shape and scale parameters are determined to ensure the proportion of failed units does not exceed 90%. The shape and scale parameters can be varied with different coefficient parameters. Here we set the upper bound for T to be

350 to mimic the jet engine data.

(b) **Determine the status of each unit:** Set the failure threshold as $\alpha = \exp(5)$, and the censoring indicator is defined as $\delta = 1$ if $u(T) \geq \exp(5)$ and $\delta = 0$ if $u(T) < \exp(5)$.

To test the procedure under different situations, we consider various n and $\boldsymbol{\beta}$ to control the number of total units and effects degree of covariates. In particular,

1. Number of units $n = 50, 100, 150, 200, 250, 300$.

2. Degree of effects on the degradation process. Because we have 5 splines for each covariate, the corresponding $\boldsymbol{\beta}_j$ is a vector of length 5 for $j = 1, \ldots, 10$. Assume that the last 5 signals do not affect the degradation process, so $\boldsymbol{\beta} = \left(\boldsymbol{\beta}_1^\top, \boldsymbol{\beta}_2^\top, \boldsymbol{\beta}_3^\top, \boldsymbol{\beta}_4^\top, \boldsymbol{\beta}_5^\top, \mathbf{0}_{25}^\top\right)^\top$. We consider four scenarios and the coefficients are listed in Table 3.1. In particular,

   (A) Contributions of effective covariates $(x_1(t), \ldots, x_5(t))$ are on the same magnitude;

   (B) The signals with quadratic function forms (i.e., $x_1(t)$ and $x_5(t)$) have larger effects;

   (C) The signals with linear function forms (i.e., $x_2(t)$ and $x_4(t)$) have larger effects;

   (D) Only the random term (i.e., $x_3(t)$) has larger effect.

3.4.2 Method Comparisons

In order to better understand the performance of the proposed model and procedure, we compare the proposed model with the model without variable selection, and with the model considering only linear relationship for the covariate effect.

For the model without variable selection, λ is set to 0 in (3.6). The objective function is,

$$\mathcal{M}(\boldsymbol{\beta}) = -l(\boldsymbol{\beta}|\boldsymbol{H}) + \eta \sum_{i=1}^{n} \delta_i \{[\alpha - u(t_i)]\}^2 + (1 - \delta_i)\{[u(t_i) - \alpha]_+\}^2.$$

If we assume the covariate effect is in a linear form, then degradation index becomes $\widetilde{u}(t) = \int_0^t h\left\{\sum_{j=1}^p x_j(s)\beta_j\right\} ds$. We also use adaptive LASSO to perform variable selection and the objective function is the same as (3.6) but with $u(t)$ replaced by $\widetilde{u}(t)$.

Table 3.1: The true $\boldsymbol{\beta}$ under four simulation scenarios.

	$\boldsymbol{\beta}_1$	$\boldsymbol{\beta}_2$	$\boldsymbol{\beta}_3$	$\boldsymbol{\beta}_4$	$\boldsymbol{\beta}_5$
	10.61	−2.49	−18.00	−3.07	−4.86
	1.39	0.24	−3.31	−16.09	11.55
Scenario A: $\boldsymbol{\beta}_a$	2.75	−4.93	−0.47	1.32	1.53
	4.35	0.17	−1.06	−0.13	1.97
	−0.38	−16.67	2.31	7.90	−6.59
	2.35	0.08	0.29	−0.12	−2.70
	1.76	−0.04	−0.28	0.00	−2.09
Scenario B: $\boldsymbol{\beta}_b$	2.04	−0.18	0.30	0.14	−2.16
	1.59	−0.06	−0.11	−0.08	−2.49
	1.74	−0.09	−0.24	−0.04	−1.72
	−0.06	2.42	−0.42	−2.64	0.02
	0.16	1.44	0.48	−1.37	0.08
Scenario C: $\boldsymbol{\beta}_c$	−0.04	2.40	−0.49	−1.59	0.10
	0.11	1.32	0.04	−2.54	0.07
	−0.09	1.79	0.47	−1.94	−0.14
	−0.32	−0.13	3.25	−0.75	0.79
	−0.46	0.24	3.05	−0.77	0.89
Scenario D: $\boldsymbol{\beta}_d$	0.02	−0.55	3.68	0.55	0.68
	−0.49	−0.05	2.73	−0.18	0.63
	−0.47	0.28	2.99	0.78	0.66

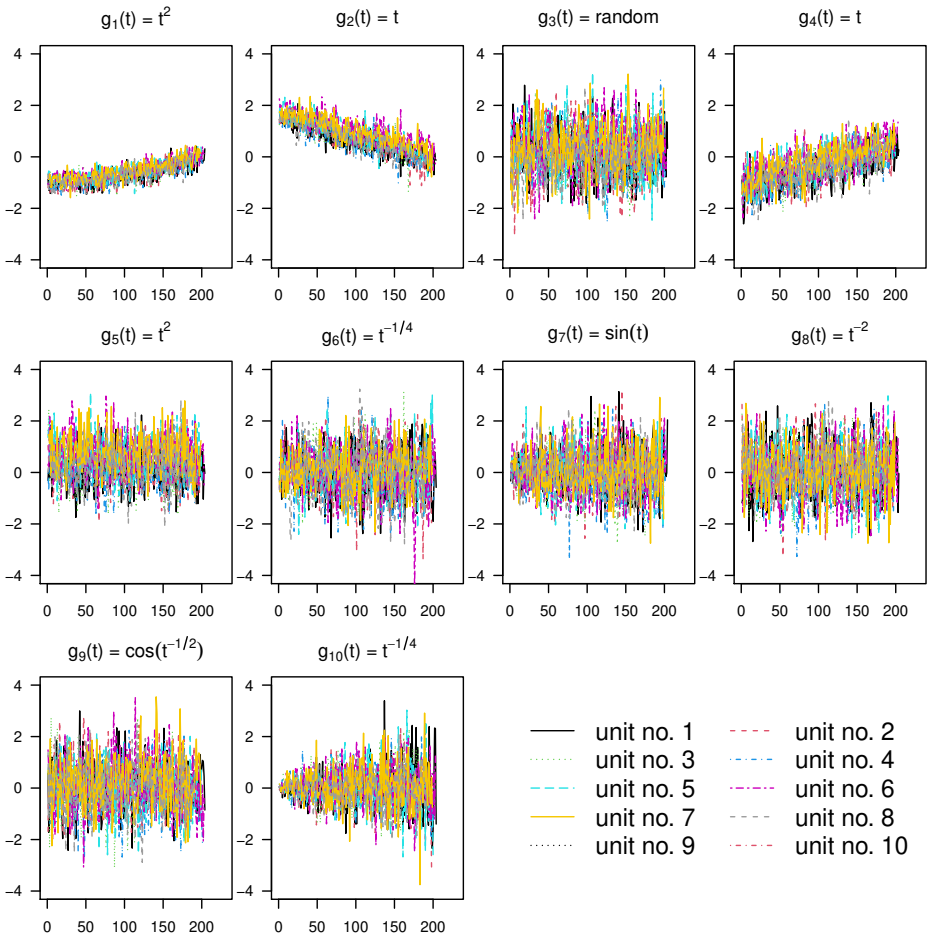


Figure 3.2: Plot of a subset of units with 10 simulated signals. Each panel represents one signal. The x-axis shows the time and the y-axis shows the signal value.

3.4.3 Simulation Results

We generate simulation data based on the procedure described in Section 3.4.1 and apply three different models to the simulated data. We denote our proposed model as DI-VS, the model without variable selection as DI-NVS, and the model assumes linear sensor effect as DI-VSL. In this simulation, the target failure threshold is set to be $\alpha = \exp(5)$. For each sample size n and coefficient parameter vector $\boldsymbol{\beta}$, we repeat the trial 200 times. For each trial, we split the simulated data into 80% training set and 20% testing set. The tuning parameter is selected by 5-fold cross-validation on the training set.

Regrading to the predictions of unit status, although the target failure threshold is α, in practice, due to training errors. If we use the target failure threshold α as the threshold, some failures with $u(t) < \alpha$ are reported to be surviving units. In practice, one can use a threshold that is slightly smaller than α, which typically yields better classification results. We call this the *practical threshold* $\widetilde{\alpha}$. One can choose $\widetilde{\alpha}_p = \exp\left[\log(\alpha) + z_p\sigma\right]$, where z_p is the quantile function of the standard LEV distribution. In the simulation study, the classification uses the $\widetilde{\alpha}_{0.01}$ threshold. The practical threshold separates the two categories better than the target failure threshold. For the purpose of prediction, we suggest using the practical threshold for predicting the status of testing units.

The simulation results are summarized in Figures 3.3 to 3.4. Figure 3.3 shows the average FNR, FPR, and TER as the number of units increases. Although in general FNR decreases for all three models, DI-VS has the most consistent performance across various scenarios and the number of units. When the number of units is small (i.e., less than 100), the FNR of DI-NVS is the largest among the three models, which shows the benefit to consider variable selection, especially when n is relatively small. The FNR for DI-VSL is large in Scenarios A and D, even when the number of units is large. For FPR, in Scenarios B, C, and D, FPR does not change a lot with the size increases. In Scenario A, DI-NVS has the smallest FPR across the number of units while DI-VSL has the largest. The results show that ignoring nonlinear relationships can lead to larger errors in some scenarios. With respect to the TER, DI-VS has

the smallest errors for almost all scenarios.

Regrading the variable selection capability, Figure 3.4 shows the average number of correctly specified variables, effect variables, and no effect variables identified by the model versus the number of units (n) for different methods and scenarios. The actual number of effective variables is 5, no-effect variables in the model is 0, and correctly specified variables are 10. Ideally, we want the model to include all 5 effective variables and zero no-effect variable. The number of correctly specified variables includes the number of effective variables that remained in the model and no-effect variables excluded from the model, which should be 10. Compared to DI-VSL, DI-VS tends to include more effective signals in the model when the number of units is large in Scenarios A and D. For Scenarios B and C, the number of effect signals remained in the model is close to DI-VSL when the number of units is large. For signals that have no effect on the underlying degradation process, DI-VS can exclude more signals across all four scenarios and various numbers of units. In general, the DI-VS model has more correctly specified variables across different simulation scenarios compared to the DI-VSL model.

Overall, the simulation results show that our proposed model has better accuracy in predicting a unit status for various scenarios and numbers of units. It can also exclude signals with no effect from the model. Appendix Section 4.A also contains some further simulation results which show that the proposed method can select the most informative sensors and is robust to different censoring rates.

3.5 Application

3.5.1 The Illustrative Application

In this section, we apply the developed methods to the jet engine sensor data (Saxena and Goebel, 2008). We first provide a brief introduction to the jet engine data. For illustration, we use a subset that contains 200 units with 100 failures and 100 surviving units. The failure-time data give the cycles to failure for failed units and time in service for surviving units. The

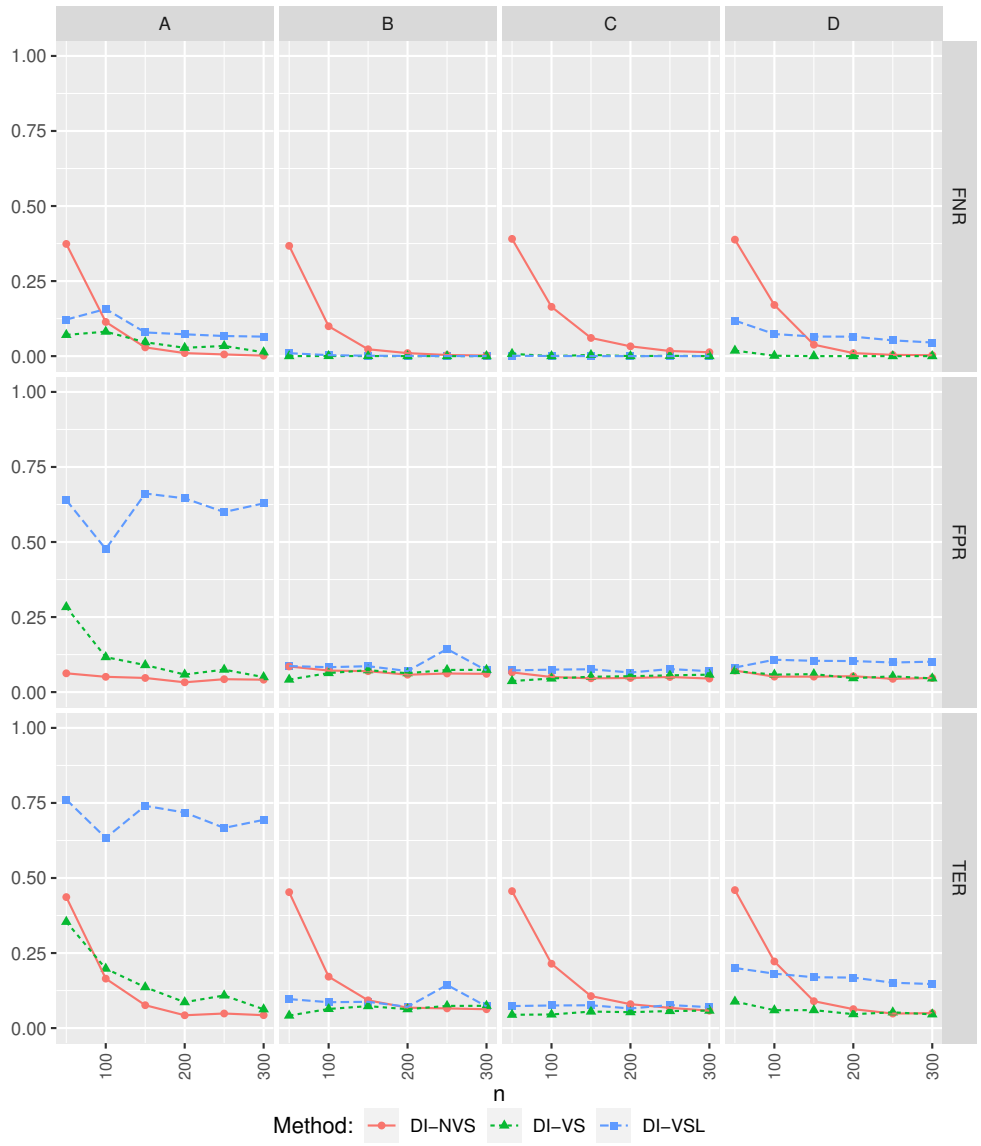


Figure 3.3: Average FNR, FPR and TER versus number of units (n) for different methods and scenarios.

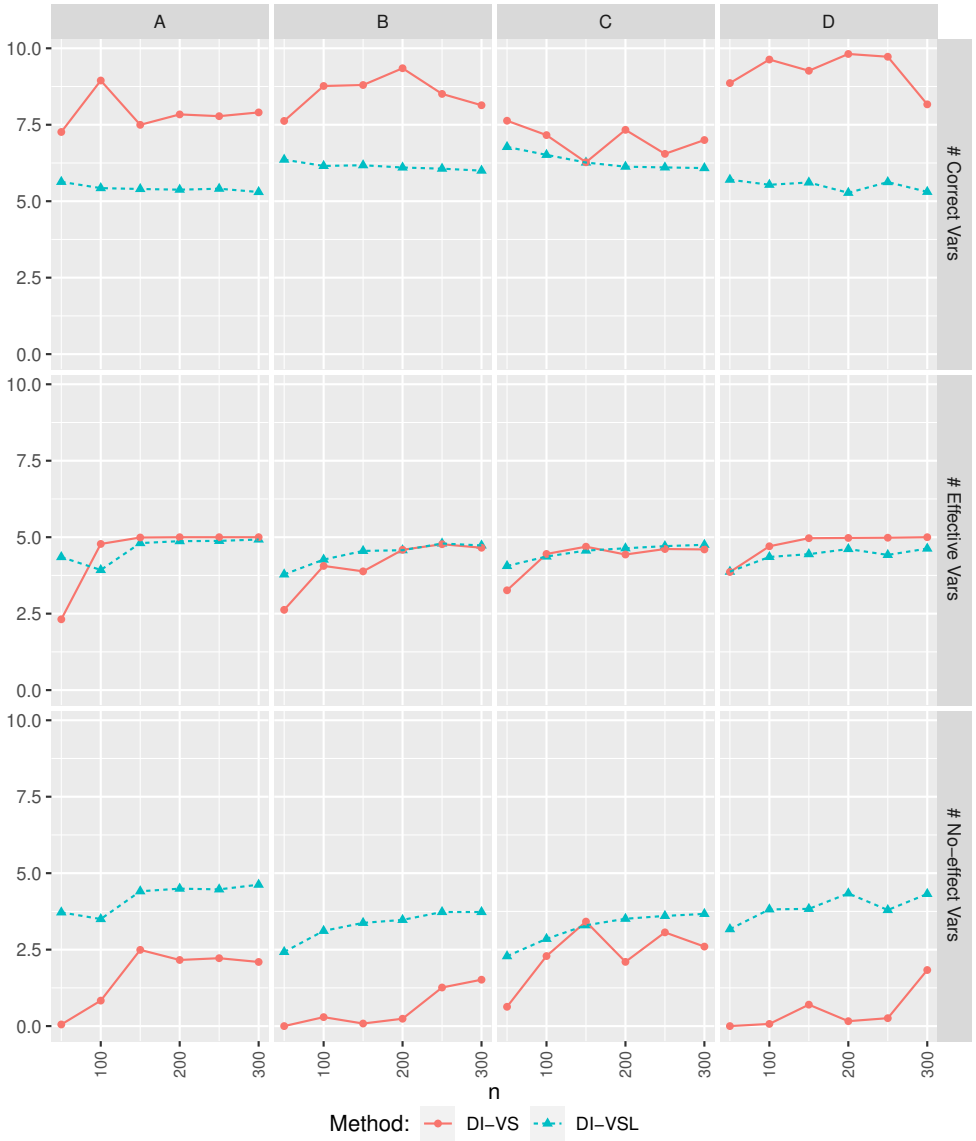


Figure 3.4: Average number of correctly specified variables, effect variables, and no effect variables identified by the model versus number of units (n) for different methods and scenarios.

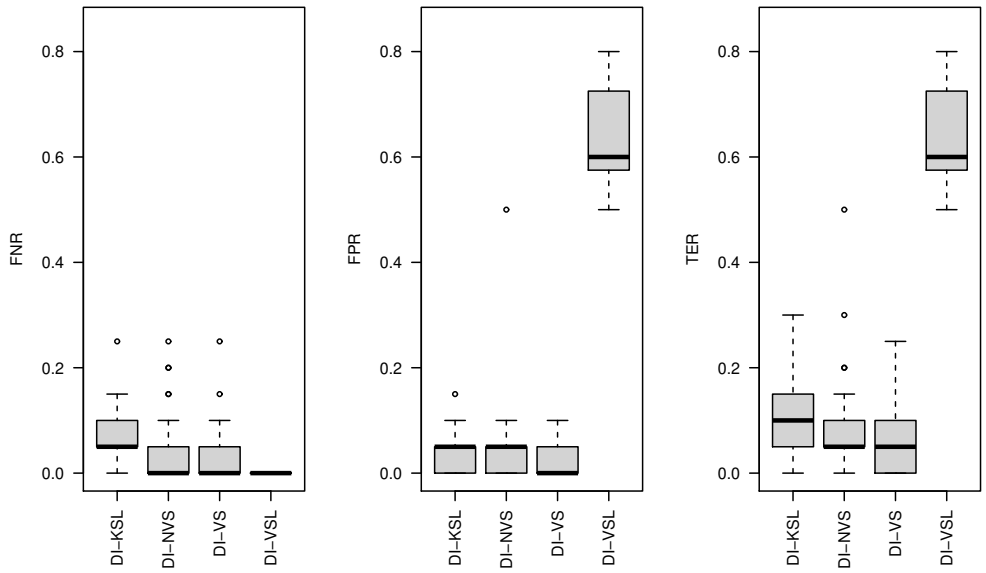
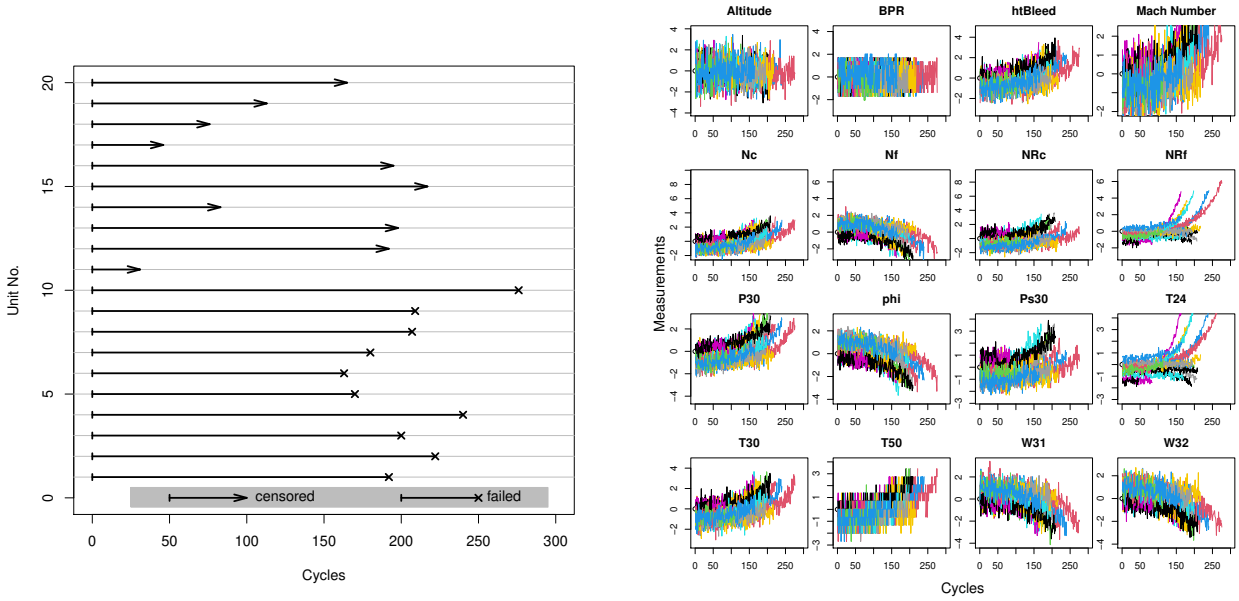


Figure 3.5: The boxplot of error rates with jet engine data with 50 replicates for different methods with $\widetilde{\alpha}_{0.01}$ practical threshold.

multi-channel sensor data give time-varying signals with cycles for all 200 units. The dataset includes 21 sensor outputs that measure the system's physical and functional conditions. There are 8 sensors that record temperature and pressure at the fan inlet and different outlets and 8 sensors that capture various fan speeds and coolant bleed in the simulation model. Besides, the measurements of pressure ratio, fuel flow ratio, bypass ratio, burner fuel-air ratio, and bleed enthalpy are also provided in the dataset. In the jet engine simulation data, there are 16 multi-channel sensors after removing those with constant signals. For demonstration, Figure 3.6 visualizes the cycles to failure for 20 units with their corresponding 16 multi-channel signals.

Except for the model introduced in Section 3.4.2, in the real application, we also compare our model with existing models in literature Kim et al. (2019), which will be introduced in Section 3.5.2. To investigate the prediction ability of each model, we split the data into 80% training set and 20% testing set. We train the degradation model with the training units and test the parameter estimation with the test units. For each of the comparison methods, we repeat the splitting 50 times and the results are summarized in Section 3.5.3.



(a) Time to Event

(b) Multi-channel Signals

Figure 3.6: Plot of a subset of the time to event (a), and multi-channel signals of the 20 units presented by different colors (b).

3.5.2 Existing Health Index Model

Kim et al. (2019) proposed a latent linear model to construct a health index using multiple sensors and involve variable selection. The health index (i.e., degradation index) is a linear combination of the sensors, as well as a linear combination of basis functions with measurement error, which is denoted as DI-KSL ("KSL" are the authors' last name initials). For unit i, the health index at time t is $h_i(t) = \boldsymbol{x}_i(t)\boldsymbol{\omega}_0 = \boldsymbol{\psi}(t)\boldsymbol{\tau}_i + \epsilon_i(t)$, where $\boldsymbol{x}_i(t)$ denotes the sensor vector, $\boldsymbol{\omega}_0$ represents the model parameter vector to be estimated, $\boldsymbol{\psi}(t)$ are the basis functions, $\boldsymbol{\tau}_i$ is the coefficient vector for the basis functions, and $\epsilon_i(t)$ follows a normal distribution. To obtain the estimation of $\boldsymbol{\omega}_0$, the property $\boldsymbol{\psi}(t_i)\boldsymbol{\tau}_i = \alpha$ of failed units is used, where α is the failure threshold and t_i is the time to failure for the ith failed unit. Therefore, when training the health index model, only the failed units are used. The maximum likelihood method was used to estimate $\boldsymbol{\omega}_0$. The variable selection is done by the adaptive LASSO. To ensure a monotonic trend (i.e., decreasing/increasing) of the health index, the section also introduced strategies to resolve practice issues. In the following comparison, the choice of basis and tuning parameters

is set to be the same as in Kim et al. (2019).

3.5.3 Result Comparisons

For the model comparison, we randomly split the dataset into 80% training set and 20% testing set while keeping the proportion of failed units as 50% in both training and testing sets. We apply each of the four models DI-VS, DI-NVS, DI-VSL, and DI-KSL to the training set and then use the trained model to predict unit status in the test set. In DI-VS and DI-NVS model, we use 10 spline basis to represent each of the 16 sensors, so in total, we have 160 coefficient parameters. The target failure threshold is set as $\alpha = \exp(5)$. The tuning parameter is selected by 5-fold cross-validation. For DI-KSL, we only use the failed units in the training set to build the model due to the model property. For each method, we repeat the splitting 50 times and the results are summarized as follows.

Figure 3.5 presents the boxplot of the prediction errors over 50 splits on the testing set of four models and Table 3.2 provides the average prediction error over repetitions. We can see that DI-VS has the lowest averaged total error and FPR among the four models. The DI-VSL has zero FNR. However, its FPR is unusually large. One potential reason is that DI-VSL fails to capture the nonlinear trend in the data can cause the estimation of $\log(\sigma^*)$ fail to shrink as expected, which results in the model having errors on one side. The DI-KSL model has the second smallest average FPR. However, its FNR is 4% larger than our proposed model DI-VS. One possible reason is that the DI-KSL model neglects the censored unit's information when training the model. Except for the DI-VSL model, the other three models have a similar inter quantile range for prediction errors.

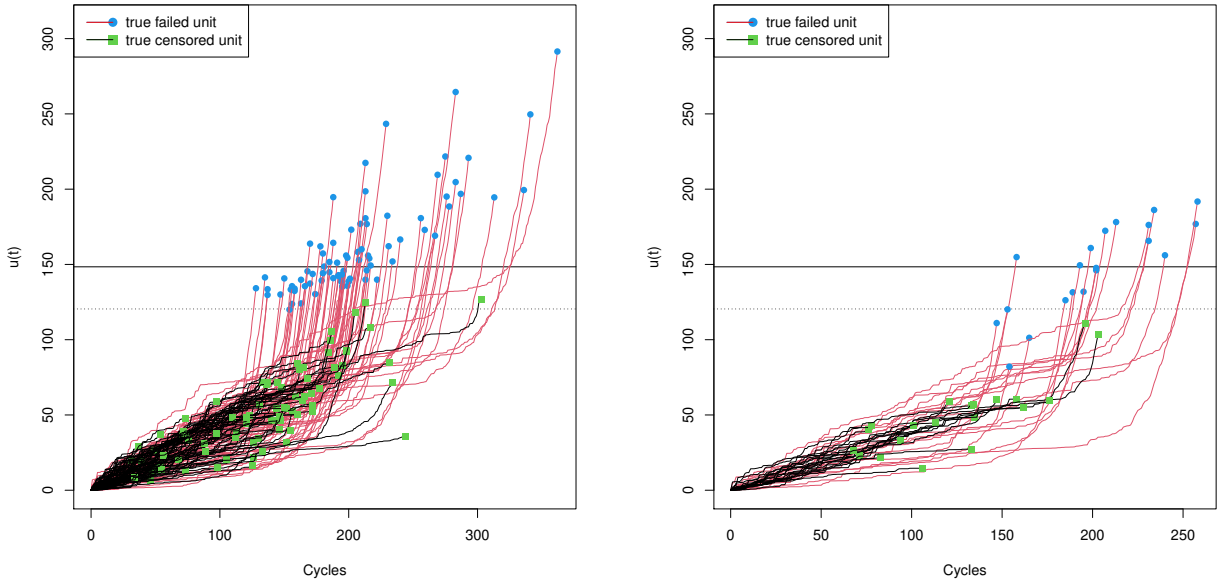
Substituting the parameter estimates into (3.1), the degradation index over time can be obtained for an individual unit. Figure 3.7 presents the degradation index built with our proposed framework DI-VS from one split. In this plot, in the testing set, all censored units are below $\widetilde{\alpha}_{0.01}$ threshold and most failed units are over that threshold. We can see from the plot that using a practice failure threshold helps to allow more true failed units' $u(t)$ to reach the threshold.

Regrading to the variable selection, in the above jet engine data analysis, all 16 sensors are kept by the DI-VS model. One possible reason is that the jet engine sensors can be highly correlated to each other. The adaptive group LASSO penalty term is not good at dealing with highly correlated covariates. It will be interesting to consider the adaptive group elastic net penalty term under the highly correlated covariates situations. Besides, as we have already seen in the simulation study, when the number of units in the model becomes larger, DI-VS tends to keep more variables in the model even some of them are no-effect. Therefore, we take a small subset of the jet engine data and test the model's variable selection ability when n is small. Appendix Figure 3.12(a) shows the proportion of each variable in the model excluded over 20 replicates when the number of units changes from 40 to 80 with 10 increment. We can see that when the number of units is relatively small, some sensors such as NRf, altitude, and Nf are excluded from the model with high probability. When $n = 40$, each variable is excluded from the model at least once. However, as n increases, there are variables that can be remained the model for all repetitions.

To better understand the sensors' effects on the degradation path, we use the accumulated local effects (ALE) plot in Apley and Zhu (2020) for visualization. ALE plot is a visualization approach to present the predictors' main and secondary-order effects in complex black box supervised learning models. We provide the technical details and the ALE plots in Appendix 3.B. The ALE plot in Appendix Figure 3.12(b) shows that the temperature at LPT outlet (T50) and pressure at HPC outlet tend to have a constant influence when the measurements are low and a larger effect on the damage level when measurements increase, while the coolant bleed (W31) has decreasing effect to the damage level before a certain point then the effect becomes constant.

3.6 Conclusions and Areas for Future Research

In this section, motivated by the jet engine multi-channel sensory data, we propose a new framework to build the degradation index based on the cumulative exposure model. The



(a) The $u(t)$ on Training Set (b) The $u(t)$ on Testing Set

Figure 3.7: The degradation index for training and testing set with one split of the jet engine data. The horizontal lines represent different thresholds. The solid line is $\alpha = \exp(5)$ and the dotted line is $\widetilde{\alpha}_{0.01}$.

Table 3.2: The average error rate over 50 splits with practical threshold $\widetilde{\alpha}_{0.01}$ of four methods.

Method	FNR	FPR	TER
DI-VS	0.030	0.026	0.057
DI-KSL	0.070	0.034	0.104
DI-VSL	0.000	0.645	0.645
DI-NVS	0.040	0.040	0.080

framework can handle censored data and conduct variable selection automatically. The comprehensive simulation studies and jet engine data analysis show that our approach has flexibility and advantages. It is also demonstrated that the performance of the proposed framework is more robust than other models. The DI-VS model has consistently good prediction accuracy regardless of the dataset size and scenarios.

The flexibility of the proposed framework makes it applicable to many complex engineering systems equipped with multiple sensors. In practice, based on the knowledge about different applications, the way to model the impact of sensors $f_j(x_j)$, and how to conduct the nonlinear transformation $h(z)$ can be adjusted accordingly. Regarding the usage of the proposed degradation index, this section focuses on the product status prediction. Other potential usage includes the product health assessment and prediction for the remaining useful life based on existing models.

There are a few limitations of the proposed framework. One of the limitations is that the model tends to keep no-effect sensors when the number of units n is large. A possible reason is that we use the log-likelihood function in the objective function. When the number of units is large, as long as the sensor can provide a little contribution to the likelihood, adding up these contributions of n units can lead to large minimization of the objective function. It will be interesting to study the adaptive elastic net penalties or involve the number of units in the penalty term. Another limitation is that the proposed model can only handle numerical covariates. It will be useful to extend the current framework to deal with categorical variables.

There are some other future directions for the proposed model. In this section, we consider an additive way to model all sensors' impact. However, in reality, the way that sensors influence the degradation process can be much more complex. It will be interesting to allow sensors interactions in the model. Besides, this proposed framework requires strict monotonicity of the degradation index. If the Wiener process is used, then there needs not to be such a strict requirement for monotonicity as long as there is an overall trend. It will be interesting to study degradation index building with less requirement on monotonicity. Multiple degradation characteristics modeling has been popular (e.g., Fang et al., 2020; Saberzadeh and

Razmkhah, 2022). Building multiple degradation indexes for data with multiple degradation characteristics is also worth investigating.

Appendix 3.A Additional Results in Simulation Study

3.A.1 Most Informative Sensors Selected

Besides the prediction error rates and the number of correctly selected variables, the most informative sensors selected are also of interest. In the simulation study, there are four scenarios and 10 sensors in each scenario. For all scenarios, sensors No. 1 to 5 have impacts on the underlying degradation index and sensors No. 6 to 10 have no effect on the degradation process. The results are summarized to find which are the top 5 most informative sensors selected under different scenarios by the two methods: DI-VS and DI-VSL. Here the most informative sensor is referred to as the sensor with the highest average proportion remaining in the model. Table 3.3 reports the top 5 sensors and their corresponding proportion remaining in the models under various scenarios and different methods. We can see that under all scenarios, the top 5 most informative sensors selected by the proposed DI-VS model are the 5 sensors that have impacts on the degradation index. However, in the DI-VSL model, except for scenario A, the model includes sensors that do not impact the degradation index as the top 5 most important sensors. This indicates that on average, our proposed model can correctly select all useful sensors as the most important sensors while the DI-VSL model can not always correctly select all useful sensors.

We also summarized the proportion of each sensor being selected by the two models. Figures 3.8 to 3.9 show the boxplot of the proportion of a sensor remaining in the model under each scenario. We can see that for those truly effective sensors (No. 1 to 5), the proposed model has a higher chance to select them for most cases compared to the DI-VSL model. For those no-effect sensors (No. 6 to 10), our proposed model has a lower chance to keep them in the model compared to the DI-VSL model.

Table 3.3: The selected top 5 most informative sensors under various scenarios and different methods.

Scenario	DI-VS		DI-VSL	
	Sensor No.	Proportion	Sensor No.	Proportion
A	1	1.00	5	0.99
	5	0.91	4	0.93
	2	0.90	2	0.92
	4	0.89	1	0.92
	3	0.82	3	0.87
B	1	1.00	1	1.00
	5	0.99	2	0.98
	2	0.94	5	0.93
	4	0.73	4	0.82
	3	0.44	9	0.72
C	4	1.00	2	1.00
	1	0.99	1	1.00
	2	0.99	4	0.98
	3	0.77	5	0.87
	5	0.59	6	0.67
D	1	1.00	2	1.00
	2	0.99	1	0.97
	4	0.98	3	0.84
	5	0.91	9	0.84
	3	0.86	8	0.83

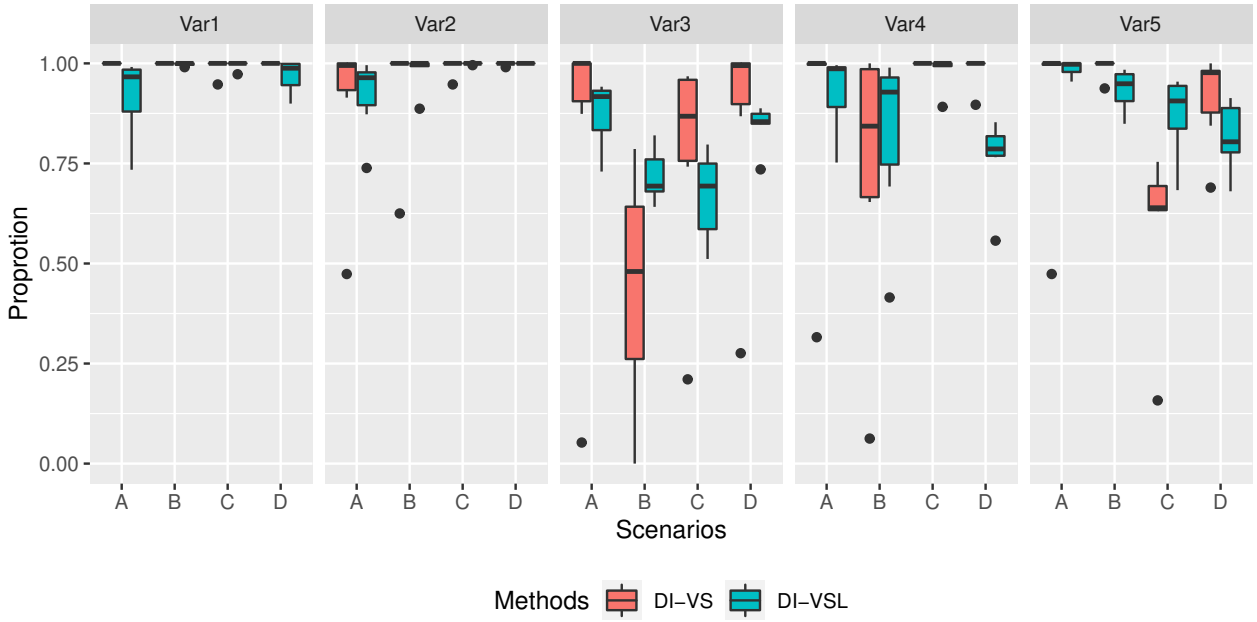


Figure 3.8: The boxplot for proportions of truly effective sensors selected by the two models under various scenarios.

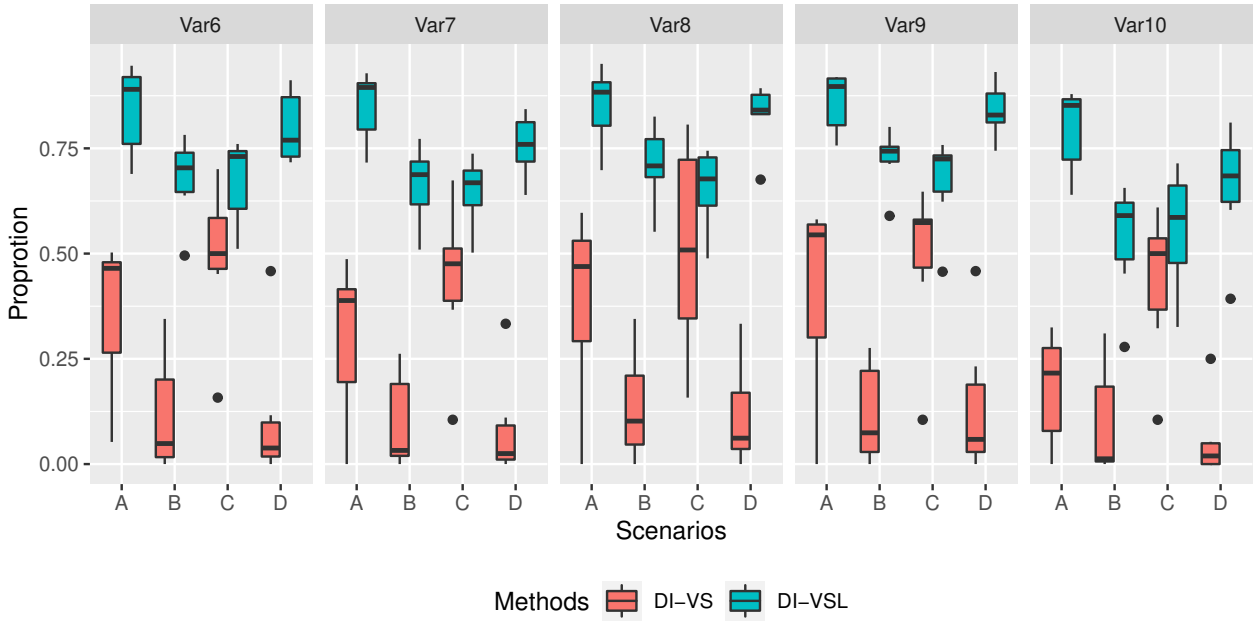


Figure 3.9: The boxplot for proportions of truly no effect sensors selected by the two models under various scenarios.

Table 3.4: The average proportions of failed units in the simulated datasets under different scenarios and sample sizes.

n	Scenario			
	A	B	C	D
50	0.70	0.75	0.71	0.65
100	0.68	0.74	0.72	0.62
150	0.68	0.74	0.72	0.63
200	0.68	0.73	0.72	0.61
250	0.67	0.74	0.72	0.60
300	0.67	0.73	0.72	0.60

3.A.2 Details on Censoring Rates

The effects of the censoring rate on the model performance are worth investigating. We summarize the average censoring rates in the simulation study. Table 3.4 shows the average proportions of the failed units under different scenarios and sample sizes. Overall, the failure proportions of different samples sizes are close to each other when the scenarios are the same. So the censoring rate can be considered fixed in each scenario in the simulation study.

Due to the failure mechanism and randomness in the generated observation time and sensors in our study, it is hard to precisely control the censoring rate. We considered a small case study under scenario A with a sample size of 100. We generated simulated datasets with low, medium, and high censoring rates and applied our method to see the performance. On average the censoring rates are 0.22, 0.34, and 0.52 over replications. Figures 3.10 to 3.11 present the average error rates and variable selection versus censoring rate over 100 replications. We can see that for the proposed DI-VS model, changing the censoring rate does not have much impact on the error rates and the number of correctly selected variables. While for the DI-NVS model, a higher censoring rate leads to higher FNR and TER. For the DI-VSL method, a higher censoring rate reduces the number of correctly selected variables. Therefore, we can see that the proposed DI-VS model performance is robust with respect to the censoring rate.

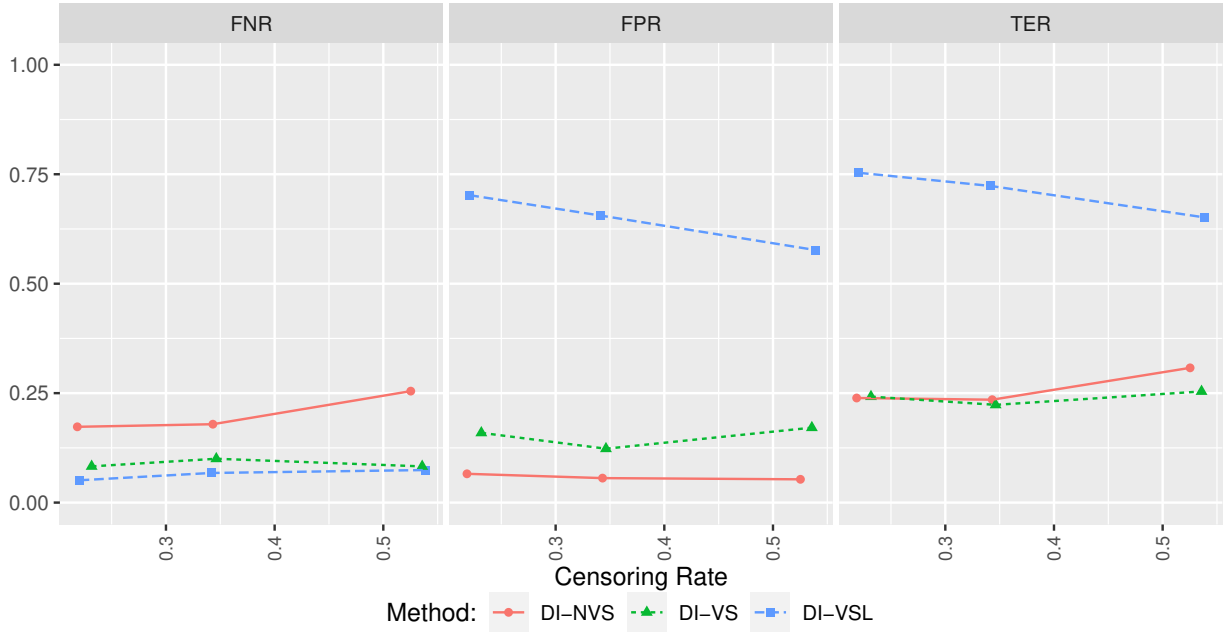


Figure 3.10: The error rates versus censoring rate under scenario A with a sample size of 100.

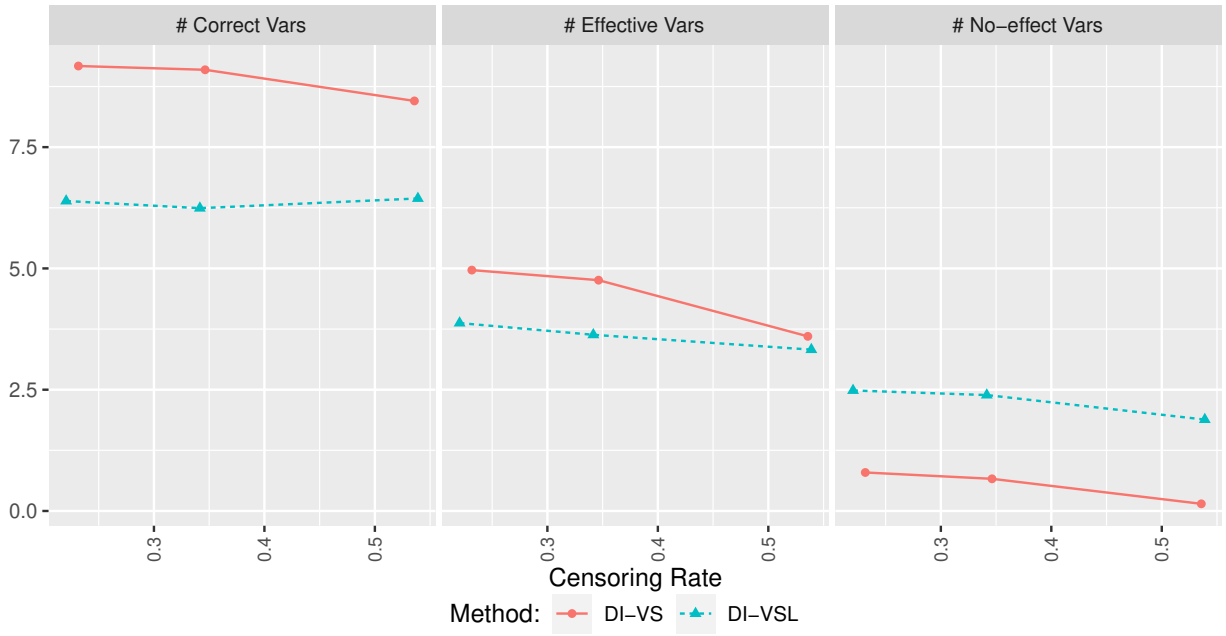


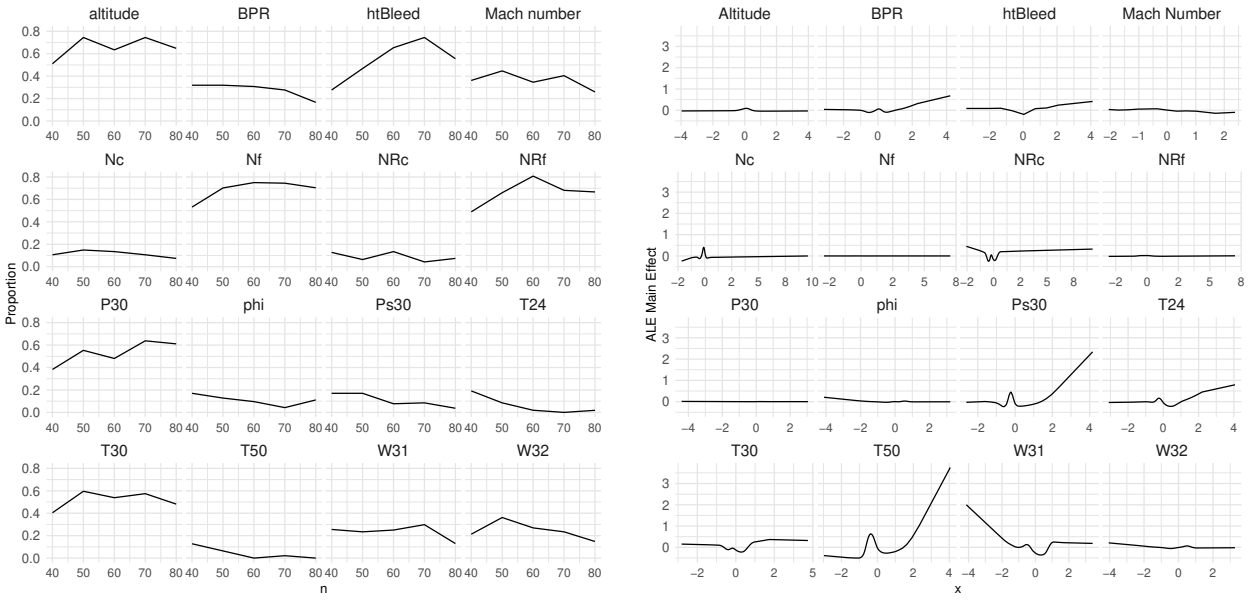
Figure 3.11: The variable selection results versus censoring rate under scenario A with a sample size of 100.

Appendix 3.B Visualization of Signal Effects on the Degradation Process

To better understand sensors' effects on the degradation path, we use the accumulated local effects (ALE) plot proposed by Apley and Zhu Apley and Zhu (2020) for visualization. ALE plot is a visualization approach to present the predictors' main and secondary-order effects in complex black box supervised learning models. Here, we focus on the main effect of the sensors. Suppose the model has p predictor variables $\boldsymbol{X} = (X_1, \ldots, X_p)'$ and the $f(\boldsymbol{x}) = f(x_1, \ldots, x_p)$ is the fitted model that predicts the response as a function of observed variables $\boldsymbol{x}$. The ALE main effect is defined as

$$g_{j,\mathrm{ALE}}(x_j) = \int_{x_{\min,j}}^{x_j} \mathbb{E}[f^j(X_j, \boldsymbol{X}_{\setminus j})|X_j = z_j]dz_j, \tag{3.8}$$

where $\boldsymbol{X}_{\setminus j}$ denotes the $p-1$ predictors excluding X_j and $f^j(X_j, \boldsymbol{X}_{\setminus j}) = \partial f(\boldsymbol{X})/\partial X_j$ represents the local effect of X_j on $f(\boldsymbol{X})$. Here we consider visualizing the effect of sensors to the damage exposure for interpretability. That means we use the pre-integration damage level $h\left\{\sum_{i=1}^{p} f_j[x_j(s); \boldsymbol{\beta}_j]\right\}$ instead of $u(t)$ as the predictive model $f(\boldsymbol{x})$ in (3.8) to reduce the complexity in interpretation introduced by integration. The ALE plot of each sensor is shown in Figure 3.12(b), where the x-axes in Figure 3.12(b) are the standardized measurements. The ALE plot shows that the temperature at LPT outlet (T50) and pressure at HPC outlet tend to have a constant influence when the measurements are low and a larger effect on the damage level when measurements increase, while the coolant bleed (W31) has a decreasing effect to the damage level before a certain point then the effect becomes constant. In addition, Figure 3.12(a) shows the proportion of sensors being removed from the model when the number of units is small. It demonstrates that the sensors with no obvious effect (i.e., the ALE main effects are constant as the measurements change) on the degradation process are more likely removed from the model, such as altitude and NRf.



(a) Proportion of Sensors Removed (b) Accumulated Local Effects

Figure 3.12: The proportion of sensors be removed from the model when the number of units is small with the jet engine data (a) and accumulated local effects of each signal on the degradation process (b). The x-axes in (b) are the standardized measurements.

Bibliography

D. W. Apley and J. Zhu. Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82:1059–1086, 2020.

A. Chehade, C. Song, K. Liu, A. Saxena, and X. Zhang. A data-level fusion approach for degradation modeling and prognostic analysis under multiple failure modes. *Journal of Quality Technology*, 50:150–165, 2018.

W.-B. Chen, X.-Y. Li, and R. Kang. Integration for degradation analysis with multi-source adt datasets considering dataset discrepancies and epistemic uncertainties. *Reliability Engineering & System Safety*, 222:108430, 2022.

C. Duan, Y. Li, H. Pu, and J. Luo. Adaptive monitoring scheme of stochastically failing systems under hidden degradation processes. *Reliability Engineering & System Safety*, 222: 108322, 2022.

G. Fang, R. Pan, and Y. Hong. Copula-based reliability analysis of degrading systems with dependent failures. *Reliability Engineering & System Safety*, 193:106618, 2020.

X. Fang, K. Paynabar, and N. Gebraeel. Multistream sensor fusion-based prognostics model for systems with single failure modes. *Reliability Engineering & System Safety*, 159:322–331, 2017.

Y. Hong and W. Q. Meeker. Field-failure predictions based on failure-time data with dynamic covariate information. *Technometrics*, 55:135–149, 2013.

Y. Hong, M. Zhang, and W. Q. Meeker. Big data and reliability applications: The complexity dimension. *Journal of Quality Technology*, 50:135–149, 2018.

J. Huang, J. L. Horowitz, and F. Wei. Variable selection in nonparametric additive models. *Annals of Statistics*, 38:2282–2313, 2010.

S. G. Johnson. The nlopt nonlinear-optimization package. 2020. https://cran.r-project.org/web/packages/nloptr/index.html.

M. Kim, C. Song, and K. Liu. A generic health index approach for multisensor degradation modeling and sensor selection. *IEEE Transactions on Automation Science and Engineering*, 16:1426–1437, 2019.

M. Kim, J.-R. C. Cheng, and K. Liu. An adaptive sensor selection framework for multisensor prognostics. *Journal of Quality Technology*, 53:566–585, 2021.

A. Kumar, C. Parkash, G. Vashishtha, H. Tang, P. Kundu, and J. Xiang. State-space modeling and novel entropy-based health indicator for dynamic degradation monitoring of rolling element bearing. *Reliability Engineering & System Safety*, 222:108356, 2022.

T. Li, X. Si, H. Pei, and L. Sun. Data-model interactive prognosis for multi-sensor monitored stochastic degrading devices. *Mechanical Systems and Signal Processing*, 167:108526, 2022.

K. Liu, N. Z. Gebraeel, and J. Shi. A data-level fusion model for developing composite health indices for degradation modeling and prognostic analysis. *IEEE Transactions on Automation Science and Engineering*, 10:652–664, 2013.

R. Mazumder, J. H. Friedman, and T. Hastie. Sparsenet: Coordinate descent with nonconvex penalties. *Journal of the American Statistical Association*, 106:1125–1138, 2011.

W. Q. Meeker and L. A. Escobar. *Statistical Methods for Reliability Data*. John Wiley & Sons, 1998.

W. Q. Meeker and Y. Hong. Reliability meets big data: Opportunities and challenges, with discussion. *Quality Engineering*, 26:102–116, 2014.

J. O. Ramsay. Monotone regression splines in action. *Statistical Science*, 3:425–441, 1988.

Z. Saberzadeh and M. Razmkhah. Reliability of degrading complex systems with two dependent components per element. *Reliability Engineering & System Safety*, 222:108398, 2022.

A. Saxena and K. Goebel. PHM08 challenge data set. Technical report, NASA Ames Prognostics Data Repository, Moffett Field, CA, 2008.

C. Song and K. Liu. Statistical degradation modeling and prognostics of multiple sensor signals via data fusion: A composite health index approach. *IISE Transactions*, 50:853–867, 2018.

C. Song, K. Liu, and X. Zhang. Integration of data-level fusion model and kernel methods for degradation modeling and prognostic analysis. *IEEE Transactions on Reliability*, 67: 640–650, 2018.

R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996.

H. Wang, H. Liao, and X. Ma. Stochastic multi-phase modeling and health assessment for systems based on degradation branching processes. *Reliability Engineering & System Safety*, 222:108412, 2022.

Z. Wang, Q. Zhai, and P. Chen. Degradation modeling considering unit-to-unit heterogeneity-a general model and comparative study. *Reliability Engineering & System Safety*, 216:107897, 2021.

Y. Wei, D. Wu, and J. Terpenny. Learning the health index of complex systems using dynamic conditional variational autoencoders. *Reliability Engineering & System Safety*, 216:108004, 2021.

Z. Ye and N. Chen. The inverse Gaussian process as a degradation model. *Technometrics*, 56:302–311, 2014.

Z. Ye and M. Xie. Stochastic modeling and analysis of degradation for highly reliable products. *Applied Stochastic Models in Business and Industry*, 31:16–32, 2015.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68:49–67, 2006.

H. Zheng, X. Kong, H. Xu, and J. Yang. Reliability analysis of products based on proportional hazard model with degradation trend and environmental factor. *Reliability Engineering & System Safety*, 216:107964, 2021.

H. Zou. The adaptive Lasso and its oracle properties. *Journal of the American Statistical Association*, 101:1418–1429, 2006.

Chapter 4 Variational Inference for Spatial Survival Models under Bayesian Framework

Abstract

In the modern survival analysis, the geo-graphically referenced time-to-event data are often collected for analysis. For such spatial survival data, the spatial dependence among survival times needs to be properly accounted in the model. In the literature, spatial random effect models, such as the cumulative exposure model and the proportional hazards model, are often used for analysis with the Bayesian approach as model inference. However, the inference problem is often high dimensional with respect to the number of spatial locations. Consequently, the conventional Markov Chain Monte Carlo (MCMC) methods for sampling the posterior can be very time-consuming when the number of spatial locations is large. In this chapter, we investigate the capability of variational inference (VI) for the inference of spatial survival models and aim for a good balance between the estimation accuracy and computational efficiency. Specifically, two divergence metrics, $\alpha-$divergence and the Kullback Leibler (KL) divergence are used in the VI methods for the spatial survival models. In the numerical study, we compare the MCMC and VI methods under two spatial survival datasets. The comparison results show that the VI method has a comparable performance as the MCMC approach but with much more efficient computational time.

Key Words: $\alpha-$divergence, Accelerated Failure Time Model, Bayesian Inference, Hamiltonian Monte Carlo, Proportional Hazards Model.

4.1 Introduction

4.1.1 Background

Spatial survival data are commonly observed in medical, biostatistics, and engineering fields, when the survival data are collected over spatial locations, such as clinical sites and geographical regions. For example, in the Titan GPU lifetime study (e.g., Min et al., 2022), the GPUs of Titan supercomputer are regularly arranged in a server room and the lifetime of these GPUs across various spatial locations is of interest. In the pine tree survival study (Li et al., 2015), the survival data of pine trees in 182 geographical sites are collected, and the main focus is to understand the effect of thinning treatment on the survival rates of pine trees across regions. In these studies, there exist unobserved heterogeneities in observations among different locations. Besides, the survival data in adjacent locations may be correlated, because closer locations can share similar environmental characteristics. Thus, modeling the spatial correlation is crucial for understanding the time-to-event data across locations.

In literature, spatial survival models incorporate the spatial effect by extending existing survival models with spatially correlated random effects. The spatial random effects are assumed to follow a certain distribution and the correlations between random effects are described by the variance-covariance matrix of the distribution. The Bayesian framework is one popular approach for spatial survival model inference. In Bayesian approaches, priors are assigned over model parameters and inferences are conducted by obtaining samples from posteriors. Because the number of spatial locations can be large, the posterior is often intractable. Markov chain Monte Carlo (MCMC) approaches are used in many studies to generate samples from the posterior. Recently, Hamiltonian Monte Carlo (HMC), as an efficient MCMC method in dealing with high-dimensional distributions, receives great attention. Various algorithms such as No-U-Turn Sampler (NUTS) are deployed for HMC (Hoffman et al., 2014).

However, in problems with a massive amount of data and complex models, the computational cost of HMC can still limit its application. Thus, we are motivated to study an

approximation inference approach, variational inference (VI) for spatial survival model inferences. VI is an optimization-based approach to approximate intractable distributions. VI approximates the complicated posterior with a simple variational distribution by minimizing the metrics that quantify the closeness between two distributions. The Kullback-Leibler (KL) divergence and the $\alpha-$divergence are two commonly used metrics.

In this chapter, motivated by computational efficiency, we are interested in investigating the performance of VI for spatial survival models. Specifically, we study two types of spatial survival models: the accelerated failure time (AFT) model, which is a special case of the cumulative exposure model, and the proportional hazards (PH) model. The capability of VI for spatial survival model inference is illustrated with the Titan GPU lifetime data and pine tree survival data. Besides, we compare the performance of VI and HMC regarding computing time and negative log-likelihood (NLL).

4.1.2 Related Literature

In literature, spatial survival models have been studied in many fields. Henderson et al. (2002) investigate possible spatial variation in survival of adult leukemia patients based on Bayesian hierarchical proportional hazards (PH) and proportional odds (PO) models. The study of Banerjee et al. (2003) employs the PH model with gamma frailties to explain the pattern of infant mortality among counties while accounting for important covariates and spatial correlations. Diva et al. (2008) study the joint modeling of multiple cancers while accounting for spatial correlations. Darmofal (2009) studies political event processes using survival models that incorporate spatial correlation via a conditional auto-regression (CAR) prior. Zhao et al. (2009) illustrate a flexible spatial frailty survival modeling with an analysis of spatially oriented breast cancer data. Li et al. (2015) present a semi-parametric proportional hazards model for the survival rates of pine trees across 182 sites in the United States. Min et al. (2022) use mixture distributions within the lifetime regression model to study the potential effect of spatial locations on the Titan GPU lifetime with competing risks.

In the above studies, MCMC is often used for statistical inferences. However, for massive

data, MCMC can be time-consuming. VI as an efficient alternative to MCMC receives much research attention. Blei et al. (2017) provide a thorough review of VI. Minimizing the KL divergence between the posterior and a variational distribution that belong to a mean-field family is a standard VI procedure in many studies. But the parameter independence assumption of the mean-field family limits the flexibility of variational distribution, and the KL divergence can lead to underestimation of the posterior variance. Structured VI that allows dependencies between parameters is one direction to introduce flexibility to variational distributions. For example, Hoffman and Blei (2015) allow arbitrary dependencies between global and local hidden variables for better parameter estimations. Kucukelbir et al. (2017) considers Gaussian variational families with non-diagonal covariance. Ranganath et al. (2016) propose hierarchical Gaussian process priors to allow the correlation between parameters through a more flexible and structured approximation distribution. The other direction is to use different divergence metrics, such as the α-divergence as in Hernandez-Lobato et al. (2016) to allow flexible behavior of the variational distribution.

There are some works incorporating VI with Bayesian survival analysis. Kim and Pavlovic (2018) apply VI to a Gaussian process survival analysis model, which is an extension of the Cox proportional hazards model. Boškoski et al. (2021) use variational Bayes to estimate parameters in an unemployment problem. Jung and Gerstung (2021) propose a population level inference in electronic health records with stochastic VI. Xiu et al. (2020) studies the variational inference capability of model fitting with individual survival analysis. However, these applications did not consider survival models with spatial correlations. For the two applications in this study, Li et al. (2015) and Min et al. (2022) use expectation-maximization (EM) algorithm and HMC respectively for model inference. Both methods can be time-consuming. The capability and time efficiency of VI with the KL/$\alpha-$divergence in survival analysis and the performance comparison between HMC and variational inference are also worth investigating. Therefore, this chapter aims to compare the performance of KL divergence, $\alpha-$divergence, and HMC in spatial survival models under the Bayesian framework.

4.1.3 Overview

The rest of this chapter is organized as follows. Section 4.2 introduces two commonly used spatial survival models: the cumulative exposure model and the proportional hazards model under the Bayesian framework. Section 4.3 describes two metrics, KL and $\alpha-$divergence as well as the parameter estimation in VI. Section 4.4 uses two real datasets, the GPU data and pine tree data to compare VI and HMC performance. Section 4.5 provides some conclusion and future directions.

4.2 Bayesian Framework for Spatial Survival Models

In this section, two specific Bayesian spatial survival models are introduced. The first one is the cumulative exposure model, which is a generalized model for accelerated failure time (AFT) model for time-dependent covariates. The second one is the proportional hazards model (PH) with time-dependent covariates.

Suppose there are m distinct locations $s_1, \ldots, s_m$. Let t_{ij} be the event time for the jth unit in the ith location s_i, where $i = 1, \ldots, m, j = 1, \ldots, n_i$. Here n_i is the number of units in location s_i and the total number of units is $n = \sum_{i=1}^{m} n_i$. Let δ_{ij} be the corresponding censoring indicator. Set $\delta_{ij} = 1$ if the unit is failed, otherwise $\delta_{ij} = 0$. Denote $\boldsymbol{x}_{ij}(t)$ to be the p-dimensional vector of related explanatory variables at time t. Here we consider a general notation of the covariates vector $\boldsymbol{x}_{ij}(t)$, which is a function of time t. For a time-invariant variable, the corresponding elements of $\boldsymbol{x}_{ij}(t)$ can be set to constants. For categorical variables, a dummy variable coding can be implemented. The time-to-event data collection can be denoted by $\mathcal{D} = \{t_{ij}, \delta_{ij}, \boldsymbol{x}_{ij}(t) : t \leq t_{ij}, i = 1, \ldots, m, j = 1, \ldots, n_i\}$

4.2.1 Cumulative Exposure Model

The cumulative exposure model describes the cumulative damage level of the product by a certain time t given time-varying covariates $\boldsymbol{x}(t)$. Let $\boldsymbol{\beta}$ be the coefficient vector for covariates.

The cumulative exposure $u_{ij}(t)$ of unit j in location i by time t is defined as

$$u_{ij}(t) = \int_0^t \exp\left[-\boldsymbol{x}_{ij}(s)^\top \boldsymbol{\beta}\right] ds.$$

The unit fails at time T when the amount of cumulative exposure reaches a random threshold U. Let T_{ij} be the random event time of jth unit in ith location. Usually, the log cumulative exposure $\log[u_{ij}(T_{ij})]$ is assumed to follow a location-scale distribution. When considering the spatial random effect, the model can be written as

$$\log\left[u_{ij}(T_{ij})\right] = \mu + \gamma_i + \sigma\epsilon_{ij}, \tag{4.1}$$

where γ_i is the spatial random parameter that represents the random effect of ith location and ϵ_{ij} is independent and identically distributed and follows the standard location-scale distribution. Let $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_m)^\top$ be the spatial random parameter vector. Under the Bayesian framework, the random effect vector $\boldsymbol{\gamma}$ is assumed to follow a multivariate normal distribution $\boldsymbol{\gamma} \sim \text{MVN}(\mathbf{0}, \Sigma)$, where $\Sigma = \sigma_\gamma^2\Omega$. Here, σ_γ^2 describes the overall spatial variability and $\Omega = (\rho_{i,i'})_{m\times m}$ is the correlation matrix. The (i, i')th element of the correlation matrix $\rho_{i,i'}$ describes the spatial correlation between the random effect of location s_i and $s_{i'}$. Commonly used correlation function includes exponential, Gaussian, and Matèrn correlations. In this chapter, we use the exponential correlation function

$$\rho_{i,i'} = \exp[-d(s_i, s_{i'})/\nu], \nu > 0,$$

where $d(s_i, s_{i'})$ is the Euclidean distance between locations s_i and $s_{i'}$.

Denote the model parameter vector as $\boldsymbol{\theta} = \left(\mu, \boldsymbol{\beta}^\top, \boldsymbol{\gamma}^\top, \sigma, \sigma_\gamma^2, \nu\right)^\top$. Given the distribution

assumptions, the likelihood of the model parameters is

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) &= L(\sigma_\gamma^2, \nu|\boldsymbol{\gamma}) L(\mu, \boldsymbol{\beta}, \sigma|\mathcal{D}, \boldsymbol{\gamma}) \\ &= (2\pi)^{-m/2} |\Sigma|^{-1/2} \exp\left(\frac{1}{2}\boldsymbol{\gamma}^\top \Sigma^{-1} \boldsymbol{\gamma}\right) \prod_{ij} \left[\frac{1}{\sigma t_{ij}} \phi(z_{ij})\right]^{\delta_{ij}} \left[1 - \Phi(z_{ij})\right]^{1-\delta_i},\end{aligned} \tag{4.2}$$

where

$$z_{ij} = \frac{\log(u_{ij}) - \gamma_i}{\sigma}.$$

Here $u_{ij} = u(t_{ij})$ is the cumulative exposure level at the event time t_{ij} of jth unit in ith location, $\phi(\cdot)$ and $\Phi(\cdot)$ are the probability density function (pdf) and cumulative distribution function (cdf) of standard location-scale distribution, respectively.

Note that when the covariates are time-invariant, the cumulative exposure becomes

$$u(t) = \int_0^t \exp\left(-\boldsymbol{x}_{ij}^\top \boldsymbol{\beta}\right) ds = \exp\left(-\boldsymbol{x}_{ij}^\top \boldsymbol{\beta}\right) t.$$

The model in (4.1) can be simplified to a accelerated failure time (AFT) model:

$$\log\left[u_{ij}(T_{ij})\right] = -\boldsymbol{x}_{ij}^\top \boldsymbol{\beta} + \log(T_{ij}) = \mu + \gamma_i + \sigma\epsilon_{ij}, \tag{4.3}$$

which is equivalent to

$$\log(T_{ij}) = \mu + \boldsymbol{x}_{ij}^\top \boldsymbol{\beta} + \gamma_i + \sigma\epsilon_{ij}.$$

The z_{ij} term in the likelihood function (4.2) becomes

$$z_{ij} = \frac{\log(t_{ij}) - \boldsymbol{x}_{ij}^\top \boldsymbol{\beta} - \gamma_i}{\sigma}.$$

For statistical inferences of the model parameter $\boldsymbol{\theta}$, priors are assigned to model parameters $\boldsymbol{\theta}$ under the Bayesian framework. Denote the priors for $\boldsymbol{\theta}$ as $p(\boldsymbol{\theta})$, then posterior function is proportion to the joint density function of the data collection $\mathcal{D}$ and model parameters $\boldsymbol{\theta}$,

which is

$$p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}|\mathcal{D})p(\boldsymbol{\theta}).$$

For spatial AFT model parameters, when assigning priors, since we know $\sigma_\gamma^2 > 0$ and $\nu > 0$, inverse-gamma priors are considered. Non-informative priors are considered for μ, $\boldsymbol{\beta}$, and σ. We consider the following Bayesian framework for spatial AFT model:

$$\begin{aligned}
\boldsymbol{\gamma} &\propto \text{MVN}(\mathbf{0}, \sigma_\gamma^2\Omega),\\
\mu \propto 1, \quad \boldsymbol{\beta}_p &\propto \mathbf{1}_p,\\
\sigma_l \propto 1, &\text{where } \sigma_l = \log(\sigma),\\
\sigma_\gamma^2 &\propto \text{IGAM}(a_\sigma, b_\sigma),\\
\nu &\propto \text{IGAM}(a_\nu, b_\nu).
\end{aligned}$$

Here, MVN stands for multivariate normal distribution and IGAM stands for inverse-gamma distribution. The parameter a_σ, b_σ and a_ν, b_ν are the shape and scale parameters for σ_γ^2 and ν, respectively. Then the log posterior can be derived by plugging in the likelihood function and priors' pdf as follows:

$$\begin{aligned}
\log\left[p(\boldsymbol{\theta}|\mathcal{D})\right] =& \text{Constant} + \sum_{ij}\left(\delta_i\left\{-\log(t_{ij}) - \sigma_l + \log\left[\phi(z_{ij})\right]\right\} + (1-\delta_i)\log\left[1-\Phi(z_{ij})\right]\right)\\
&- \frac{1}{2}\left[\log(\sigma_\gamma^2\Omega) + \frac{\boldsymbol{\gamma}^\top\Omega^{-1}\boldsymbol{\gamma}}{\sigma_\gamma^2}\right] - \frac{1}{2}(a_\sigma+1)\log(\sigma_\gamma^2)\\
&- \frac{b_\sigma}{\sigma_\gamma^2} - \frac{1}{2}(a_\nu+1)\log(\nu) - \frac{b_\nu}{\nu}.
\end{aligned}$$

4.2.2 Proportional Hazard Model with Time-dependent Covariates

Proportional hazards (PH) model is an alternative approach to model survival data. The hazard function of the jth unit in ith location is modeled as

$$h_{ij}(t) = h_0(t) \exp\left[\boldsymbol{x}_{ij}(t)^\top \boldsymbol{\beta} + \gamma_i\right], \tag{4.4}$$

where $h_0(t)$ is the baseline hazard function, $\boldsymbol{\beta}$ is the coefficient vector for covariates, and γ_i is the spatial random effect at location s_i. The cumulative hazard function is

$$H_{ij}(t) = \int_0^t h_{ij}(s)ds = \exp(\gamma_i) \int_0^t h_0(t) \exp\left[\boldsymbol{x}_{ij}(t)^\top \boldsymbol{\beta}\right] ds.$$

The survival function is obtained by $S_{ij}(t) = \exp\left[-H_{ij}(t)\right]$. In this chapter, we consider a parametric form of the baseline hazard $h_0(t; \boldsymbol{\theta}_h)$, where $\boldsymbol{\theta}_h$ is the parameter vector. Then the model parameter can be written as $\boldsymbol{\theta} = \left(\boldsymbol{\beta}^\top, \boldsymbol{\gamma}^\top, \boldsymbol{\theta}_h^\top, \sigma_\gamma^2, \nu\right)^\top$. The likelihood of model parameter is

$$\begin{aligned} L(\boldsymbol{\theta}) &= L(\sigma_\gamma^2, \nu|\boldsymbol{\gamma}) L(\boldsymbol{\beta}, \boldsymbol{\theta}_h|\mathcal{D}, \boldsymbol{\gamma}) \\ &= (2\pi)^{-m/2} |\Sigma|^{-1/2} \exp\left(\frac{1}{2}\boldsymbol{\gamma}^\top \Sigma^{-1} \boldsymbol{\gamma}\right) \prod_{ij} \left[h_{ij}(t_{ij})\right]^{\delta_{ij}} S_{ij}(t_{ij}). \end{aligned}$$

Similar to Section 4.2.1, we can assign priors to model parameter and derive the posterior of $\boldsymbol{\theta}$.

4.3 Variational Inference for Bayesian Spatial Survival

In most Bayesian spatial survival models, due to the high-dimensional parameter space and model complexity, the exact posterior $p(\boldsymbol{\theta}|\mathcal{D})$ is usually intractable. MCMC approach is the most common way to obtain samples from the posterior distribution. However, MCMC methods are often time-consuming.

In this section, we provide a detailed introduction to variational inference. The key idea of VI is to use a variational distribution $q(\boldsymbol{\theta}|\boldsymbol{\eta})$ to approximate the exact posterior. Here $\boldsymbol{\eta}$ is the parameter vector in the variational probability distribution. Often the case, variational distribution $q(\boldsymbol{\theta}|\boldsymbol{\eta})$ is relative simple for future inference. Then a metric that evaluates the distance between two distributions $p(\boldsymbol{\theta}|\mathcal{D})$ and $q(\boldsymbol{\theta}|\boldsymbol{\eta})$ is optimized to obtain the estimate of $\boldsymbol{\eta}$. Given the estimated $\boldsymbol{\eta}$, we can sample model parameters from variational distribution $q(\boldsymbol{\theta}|\boldsymbol{\eta})$ and conduct inferences as in MCMC methods.

4.3.1 Two Types of Divergence in Variational Inference

In this section, we introduce the KL divergence and $\alpha-$divergence, which are commonly used to measure the closeness between the exact posterior $p(\boldsymbol{\theta}|\mathcal{D})$ and variational distribution $q(\boldsymbol{\theta}|\boldsymbol{\eta})$. In order to approximate the true posterior, we want to find the variational parameters $\boldsymbol{\eta}$ that minimize these divergences.

4.3.1.1 KL divergence

The KL divergence between $p(\boldsymbol{\theta}|\mathcal{D})$ and $q(\boldsymbol{\theta}|\boldsymbol{\eta})$ is defined as:

$$\begin{aligned}\mathrm{KL}\left[q(\boldsymbol{\theta}|\boldsymbol{\eta})||p(\boldsymbol{\theta}|\mathcal{D})\right] &= \int q(\boldsymbol{\theta}|\boldsymbol{\eta}) \log\left[\frac{q(\boldsymbol{\theta}|\boldsymbol{\eta})}{p(\boldsymbol{\theta}|\mathcal{D})}\right] d\boldsymbol{\theta} \\ &= \mathrm{E}_{q(\boldsymbol{\theta}|\boldsymbol{\eta})}\{\log[q(\boldsymbol{\theta}|\boldsymbol{\eta})]\} - \mathrm{E}_{q(\boldsymbol{\theta}|\boldsymbol{\eta})}\{\log\left([p(\mathcal{D},\boldsymbol{\theta})]\right)\} + \log[p(\mathcal{D})].\end{aligned} \tag{4.5}$$

Because $\log[p(\mathcal{D})]$ is intractable, we can not directly minimize KL divergence. An equivalent optimization problem is to maximize the evidence lower bound (ELBO):

$$\mathcal{L}_{\mathrm{VI}} = \log[p(\mathcal{D})] - \mathrm{KL}(q(\boldsymbol{\theta}|\boldsymbol{\eta})||p(\boldsymbol{\theta}|\mathcal{D})) = \mathrm{E}_{q(\boldsymbol{\theta}|\boldsymbol{\eta})}\left\{\log\left[\frac{p(\boldsymbol{\theta},\mathcal{D})}{q(\boldsymbol{\theta}|\boldsymbol{\eta})}\right]\right\}. \tag{4.6}$$

One characteristic of the KL divergence is that it is an asymmetric measurement. That means $\mathrm{KL}\left[q(\boldsymbol{\theta}|\boldsymbol{\eta})||p(\boldsymbol{\theta}|\mathcal{D})\right] \neq \mathrm{KL}\left[p(\boldsymbol{\theta}|\mathcal{D})||p(\boldsymbol{\theta}|\boldsymbol{\eta})\right]$. The choice of KL divergence direction affects the behavior of the variational distribution. Besides, the variational distribution often

belongs to mean-field family for optimization simplicity. That is, the parameters are assumed to be independent in $q(\boldsymbol{\theta}|\boldsymbol{\eta})$. These characteristics of KL divergence potentially lead to underestimate of posterior variance.

4.3.1.2 Rèyi's $\alpha-$Divergence

Rèyi's $\alpha-$divergence is an alternative metric that measures the closeness of $p(\boldsymbol{\theta}|\mathcal{D})$ and $q(\boldsymbol{\theta}|\boldsymbol{\eta})$. It is defined as

$$D\left[q(\boldsymbol{\theta}|\boldsymbol{\eta})||p(\boldsymbol{\theta}|\mathcal{D})\right] = \frac{1}{\alpha-1}\log\left[\int q(\boldsymbol{\theta}|\boldsymbol{\eta})^{\alpha}p(\boldsymbol{\theta}|\mathcal{D})^{1-\alpha}d\boldsymbol{\theta}\right],$$

where $\alpha > 0, \alpha \neq 1$. When $\alpha \to 1$, the $\alpha-$divergence approaches the KL divergence.

Similar to the KL divergence, we can not compute the $\alpha-$divergence directly. Alternatively, we can maximize the variational Rènyi (VR) bound, which is defined as

$$\mathcal{L}_{\alpha} = \log[p(\mathcal{D})] - D\left[q(\boldsymbol{\theta}|\boldsymbol{\eta})||p(\boldsymbol{\theta}|\mathcal{D})\right] = \frac{1}{\alpha-1}\log\left\{\mathrm{E}_{q(\boldsymbol{\theta}|\boldsymbol{\eta})}\left[\left(\frac{p(\boldsymbol{\theta},\mathcal{D})}{q(\boldsymbol{\theta}|\boldsymbol{\eta})}\right)^{1-\alpha}\right]\right\}. \tag{4.7}$$

Unlike the original Rèyi's $\alpha-$divergence's positive support, the VR lower bound can be extended to $\alpha \leq 0$. Besides, the objective $\mathcal{L}_{\alpha}$ is continuous and non-increasing on α (Li and Turner, 2016). When $\alpha \to -\infty$, it allows the approximation mass-covering behavior. That means $\boldsymbol{\theta}$ is encouraged to explore a wider range of possible values. When $\alpha \to \infty$, it encourages the zero-forcing behavior. That means the approximation tends to return an over-confident $\boldsymbol{\theta}$ estimate with a small variance. By adjusting α value, we can encourage flexible variational distribution.

4.3.2 Parameter Estimation in Variational Inference

In this section, we focus on how to obtain estimates of variational parameters by optimizing KL/$\alpha-$divergence. Since KL divergence is a special case of $\alpha-$divergence, in the following of this section, we will focus on the $\alpha-$divergence.

The VR bound $\mathcal{L}_\alpha$ is often intractable due to the model complexity and high-dimensional integration over $\boldsymbol{\theta}$. Therefore, a Monte Carlo (MC) method is used to evaluate the VR bound. Specifically, we can randomly sample from the variational distribution $\boldsymbol{\theta}_k \sim q(\boldsymbol{\theta}|\boldsymbol{\eta}), k = 1, \ldots, h$ and $\mathcal{L}_\alpha$ can be approximated by

$$\widehat{\mathcal{L}}_\alpha = \frac{1}{\alpha - 1} \log \frac{1}{h} \left[\left(\frac{p(\boldsymbol{\theta}_k, \mathcal{D})}{q(\boldsymbol{\theta}_k|\boldsymbol{\eta})} \right)^{1-\alpha} \right]. \tag{4.8}$$

Then $\widehat{\mathcal{L}}_\alpha$ is optimized and updated iteratively in order to obtain variational parameter estimates $\boldsymbol{\eta}^*$. We optimize $\widehat{\mathcal{L}}_\alpha$ using the Adam algorithm (Kingma and Ba, 2014), which is an extension to stochastic gradient descent method. The derivation for the gradient of ELBO and VR bound are shown in Appendix 4.A. The following procedure describes a general approach to obtain $\boldsymbol{\eta}^*$:

1. Determine the variational distribution for $\boldsymbol{\theta}$ in a specific spatial survival model, denoted as $q(\boldsymbol{\theta}|\boldsymbol{\eta})$.

2. Set the initial value for variational parameter to $\boldsymbol{\eta}_0$.

3. At step r, $r = 1, \ldots, g$:

   (a) Take h samples from the variational distribution with the $(r-1)$th step parameter estimate $\boldsymbol{\theta}_k \sim q(\boldsymbol{\theta}|\boldsymbol{\eta}_{r-1}), k = 1, \ldots, h$ and compute a stochastic estimate of VR bound $\widehat{\mathcal{L}}_\alpha$.

   (b) Take a gradient descent step in Adam algorithm (Kingma and Ba, 2014) to obtain the updated parameter estimate $\boldsymbol{\eta}_r$.

4. After getting the variational parameter estimate $\boldsymbol{\eta}^*$, we can sample model parameters from $q(\boldsymbol{\theta}|\boldsymbol{\eta}^*)$ and then make inferences about the spatial survival model.

4.4 Methods Comparisons with Real Datasets

In this section, we investigate the performance of HMC and VI using the Titan GPU lifetime data and the pine tree survival data. The statistical inference and computing time of HMC and VI are compared. The HMC algorithm used in this chapter is implemented in Stan (Carpenter et al., 2017). We use the default HMC algorithm in Stan, NUTS for model inferences.

4.4.1 Titan GPU Lifetime Data

4.4.1.1 Data Description

The Titan GPU dataset is from the study by Ostrouchov et al. (2020). The Titan supercomputer is on service for over 6 years and the data collected during this period allows us to understand more about GPU's failure mechanism. Since supercomputers are a collection of massive regular arranged GPUs, it is interesting to study whether the location of each GPU influences its lifetime. Specifically, GPU nodes are placed in 8×25 cabinets. There are 3 cages within each cabinet, 8 slots within each cage, and 4 nodes within each slot. Figure 4.1 shows the physical structure of Titan supercomputer. Min et al. (2022) provide more details regarding the dataset and data cleaning.

One main failure type of GPU is Off the Bus (OTB). In this chapter, we are interested to study the effect of spatial location on the OTB failure mode. We use a subset of the data, which includes the units in row number 0-7 and 1-13 with OTB failure mode. The row and column positions of each unit are considered as the location information. The node, slot, and cage information are considered as covariates that can affect GPU's lifetime. Because the covariates in the GPU dataset are invariant over time, we are interested to build a spatial AFT model to study the failures of GPU.

4.4.1.2 Bayesian AFT model for GPU data

In GPU dataset, the total number of units is n = 10,101 and there are m = 104 spatial locations. The covariates cage, slot and node are categorical variables. Consider a dummy

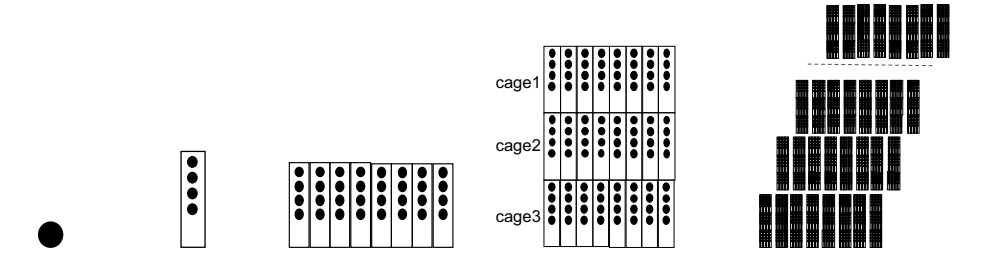


Figure 4.1: The physical organization of Titan supercomputer.

variable encoding, then there are 12 dummy variable covariates. Denote covariate as $\boldsymbol{x} = \left(1, \boldsymbol{x}_{cage}^\top, \boldsymbol{x}_{slot}^\top, \boldsymbol{x}_{node}^\top\right)^\top$, where $\boldsymbol{x}_{cage}$ represents the vector of dummy variable encoding of cage variable and similar for $\boldsymbol{x}_{slot}$ and $\boldsymbol{x}_{node}$. The spatial AFT model of jth GPU in ith location can be written as

$$\log(T_{ij}) = \boldsymbol{x}_{ij}^\top \boldsymbol{\beta} + \boldsymbol{\gamma}_i + \sigma\epsilon_{ij}.$$

For the GPU data, we assume the error term ϵ_{ij} follows the standard smallest extreme value (SEV) distribution. According to Section 4.2.1, the spatial random effect $\boldsymbol{\gamma}$ is assumed to follow a multivariate normal distribution $\boldsymbol{\gamma} \sim \text{MVN}(\mathbf{0}, \Sigma)$, where $\Sigma = \sigma_\gamma^2 \Omega$ and $(\Omega)_{ii'} = (\exp[-d(s_i, s_{i'})/\nu])$. The model parameter is $\boldsymbol{\theta} = \left(\boldsymbol{\beta}^\top, \boldsymbol{\gamma}^\top, \sigma_l, \sigma_\gamma^2, \nu\right)^\top$. We set a weak prior for σ_γ^2 by letting $a_\sigma = 1$, $b_\sigma = 1$ and strong prior for ν by setting $a_\nu = 5$, $b_\nu = 0.1$.

For the variation inference of model parameters, we assume the following variational distribution assumptions:

$$\begin{aligned}
\boldsymbol{\beta} &\sim \text{MVN}(\boldsymbol{\mu}_{\boldsymbol{\beta}}, \Sigma_{\boldsymbol{\beta}}), \text{where } \Sigma_{\boldsymbol{\beta}} = \text{Diag}(\boldsymbol{\sigma}_{\boldsymbol{\beta}}^2), \\
\boldsymbol{\gamma} &\sim \text{MVN}(\boldsymbol{\mu}_{\boldsymbol{\gamma}}, \Sigma_{\boldsymbol{\gamma}}), \text{where } \Sigma_{\boldsymbol{\gamma}} = \text{Diag}(\boldsymbol{\sigma}_{\boldsymbol{\gamma}}^2), \\
\sigma_l &\sim \text{N}(\mu_\sigma, \sigma_\sigma^2), \\
\sigma_\gamma^2 &\propto \text{IGAM}(c_\sigma, d_\sigma), \\
\nu &\propto \text{IGAM}(c_\nu, d_\nu),
\end{aligned}$$

where $\Sigma_{\boldsymbol{\beta}}$ and $\Sigma_{\boldsymbol{\gamma}}$ are diagonal matrices with diagonal elements being $\boldsymbol{\sigma}_{\boldsymbol{\beta}}^2$ and $\boldsymbol{\sigma}_{\boldsymbol{\gamma}}^2$. The varia-

tional distributions are independent to each other. In the above framework, c_σ, d_σ and c_σ, d_σ are the scale and shape parameters in inverse-gamma distribution for σ_γ^2 and ν, respectively. Let $\boldsymbol{\sigma}_{l\boldsymbol{\beta}} = \log(\boldsymbol{\sigma}_{\boldsymbol{\beta}})$, $\boldsymbol{\sigma}_{l\boldsymbol{\gamma}} = \log(\boldsymbol{\sigma}_{\boldsymbol{\gamma}})$ and $\sigma_{l\sigma} = \log(\sigma_\sigma)$. Then the variational parameter vector can be denoted as $\boldsymbol{\eta} = \left(\boldsymbol{\mu}_{\boldsymbol{\beta}}^\top, \boldsymbol{\sigma}_{l\boldsymbol{\beta}}^\top, \boldsymbol{\mu}_{\boldsymbol{\gamma}}^\top, \boldsymbol{\sigma}_{l\boldsymbol{\gamma}}^\top, \mu_\sigma, \sigma_{l\sigma}, c_\sigma, d_\sigma, c_\nu, d_\nu\right)^\top$. The log density of joint variational distribution is

$$\begin{aligned}
\log\left[q(\boldsymbol{\theta}|\boldsymbol{\eta})\right] =& \text{Constant} - \frac{1}{2}\sum_j \left[2\sigma_{l\boldsymbol{\beta},j} + \frac{(\beta_j - \mu_{\boldsymbol{\beta},j})^2}{\exp(2\sigma_{l\boldsymbol{\beta},j})}\right] \\
& - \frac{1}{2}\sum_j \left[2\sigma_{l\boldsymbol{\gamma},j} + \frac{(\gamma_j - \mu_{\boldsymbol{\gamma},j})^2}{\exp(2\sigma_{l\boldsymbol{\gamma},j})}\right] - \frac{1}{2}\left[2\sigma_{l\sigma} + \frac{(\sigma_l - \mu_\sigma)^2}{\exp(2\sigma_{l\sigma})}\right] \\
& + c_\sigma \log(d_\sigma) - \log[\Gamma(c_\sigma)] - (c_\sigma + 1)\log(\sigma_\gamma^2) - \frac{d_\sigma}{\sigma_\gamma^2} \\
& + c_\nu \log(d_\nu) - \log[\Gamma(c_\nu)] - (c_\nu + 1)\log(\nu) - \frac{d_\nu}{\nu},
\end{aligned}$$

where $\mu_{\boldsymbol{\beta},j}$ is the jth element of mean vector $\boldsymbol{\mu}_{\boldsymbol{\beta}}$ and $\sigma_{l\boldsymbol{\beta},j}$ is the jth element of the log diagonal variance vector $\boldsymbol{\sigma}_{\boldsymbol{\beta}}$ for $\boldsymbol{\beta}$. Similarly for $\mu_{\boldsymbol{\gamma},j}$ and $\sigma_{l\boldsymbol{\gamma},j}$. And $\Gamma(\cdot)$ denotes the gamma function. Plugging in $\log\left[p(\boldsymbol{\theta}, \mathcal{D})\right]$ and $\log\left[q(\boldsymbol{\theta}|\boldsymbol{\eta})\right]$ in (4.6) and (4.7) we can obtain the objective functions $\mathcal{L}_\alpha$ and $\mathcal{L}_{\text{VI}}$ in variational inference. The procedures in Section 4.3.2 can be used to obtain estimates $\boldsymbol{\eta}^*$. Then we can sample $\boldsymbol{\theta}$ from $q(\boldsymbol{\theta}|\boldsymbol{\eta}^*)$ and conduct inferences for the spatial AFT model.

4.4.1.3 Analysis Results

For the GPU data, we compare three inference methods, HMC, KL divergence and $\alpha-$divergence with $\alpha = 0.8$. To understand how well the posterior samples from each inference approach fit the model, we use residual plot to conduct diagnostics. The censored Cox-Snell residual of spatial AFT model is an extension of the standardized residual, which is defined as

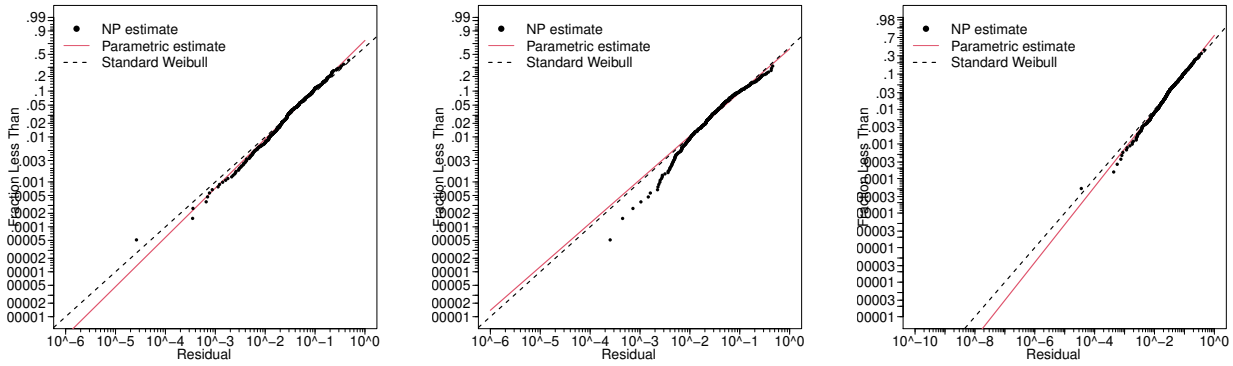
$$\widehat{\epsilon}_{ij} = \exp\left[\frac{\log(t_{ij}) - \boldsymbol{x}_{ij}^\top \widehat{\boldsymbol{\beta}} - \gamma_i}{\widehat{\sigma}}\right].$$

For a censored unit, the corresponding residual is also censored. With the SEV distribution

assumption, the residuals should approximately follow a Weibull distribution. The probability plots of censored Cox-Snell residual of three inference methods are presented in Figure 4.2. We can see that for $\alpha-$divergence, its probability plot of residual is almost aligned with the standard Weibull distribution. While the KL divergence has some deviation at the tail of residuals.

The negative log-likelihood of posterior samples and computational time are summarized in Table 4.1. We can see that VI with $\alpha-$divergence has the smallest negative log-likelihood value. Besides, the computational time of $\alpha-$divergence is also the shortest. Thus, the estimated result of $\alpha-$divergence is presented. The estimated parameters and corresponding credible intervals (CIs) are shown in Table 4.2. We can see that when other variables are fixed, cage 0-1 and slot 0-6 accelerate a unit's failure compared to the baseline unit with cage 2 and slot 7, while node 0-2 decelerates a unit's failure compared to node 3. Figure 4.3 presents the heat map of random effect at each location as well as its histogram. We can see from the heat map that the position in the upper left corner tends to decelerate the GPU's failure. Figure 4.4 shows the estimated spatial correlation as the distance increase. When fitting the spatial AFT model, we standardized the distance matrix between each GPU location and the standardized distance is in the range of $(0.083, 1.414)$. The correlation plot shows that there exists spatial correlation between adjacent locations.

It is also interesting to examine whether the number of spatial locations impacts the computing time for each of the three inference method. A series of subsets with the number of locations $m = 56, 104, 160, 200$ are selected. Each of the three methods is applied and the computing time versus the number of locations is presented in Figure 4.5. We can see that the computing time difference between HMC and $\alpha-$divergence increases as the number of locations increases. The computational advantage of $\alpha-$divergence is more obvious when the number of locations is large.



(a) $\alpha = 0.8$ (b) KL (c) HMC

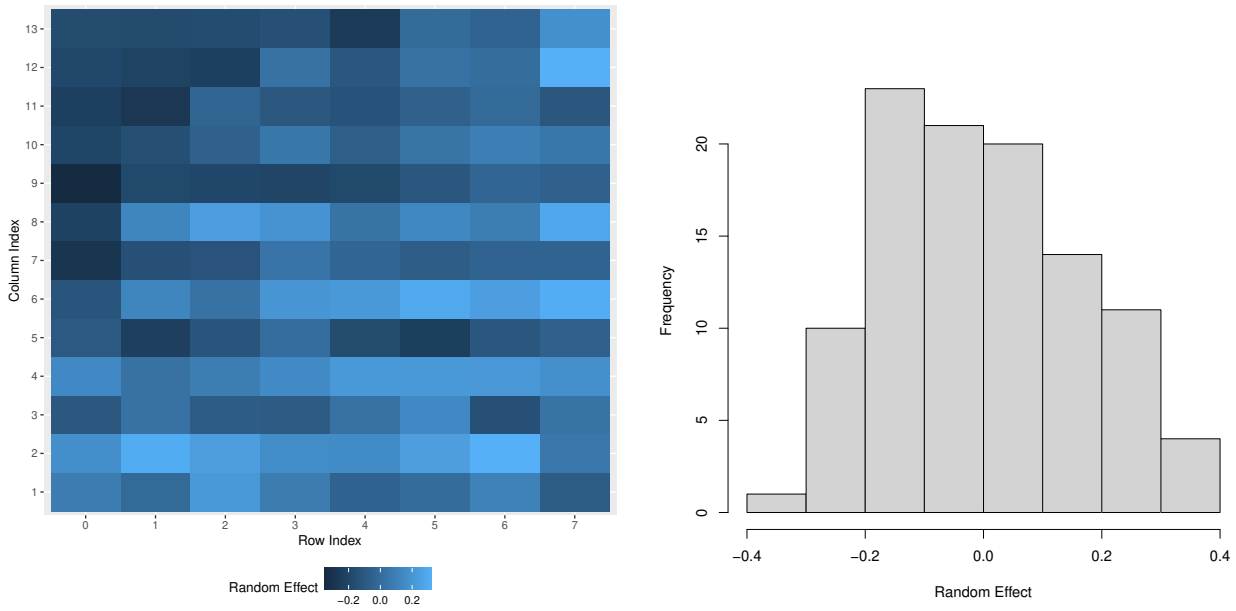
Figure 4.2: Weibull probability plot of residuals for $\alpha-$divergence, KL divergence and HMC with the GPU data.

Table 4.1: The negative log likelihood and computing time of $\alpha-$divergence, KL divergence and HMC inferences with the GPU data.

	$\alpha = 0.8$	KL	HMC
NLL	2024.96	2411.86	2040.68
Time (minutes)	17.80	96.05	20.89

Table 4.2: Parameter estimates and 95% CIs in the AFT model for the GPU data.

Parameter	Est	SE	CI Lower	CI Upper
Constant	1.925	0.015	1.896	1.953
Cage 0	0.701	0.040	0.626	0.785
Cage 1	0.275	0.018	0.240	0.311
Slot 0	0.109	0.030	0.048	0.166
Slot 1	0.066	0.026	0.019	0.117
Slot 2	0.108	0.027	0.053	0.159
Slot 3	0.113	0.027	0.061	0.165
Slot 4	0.151	0.031	0.089	0.214
Slot 5	0.048	0.026	−0.002	0.096
Slot 6	0.038	0.025	−0.011	0.087
Node 0	−0.268	0.019	−0.305	−0.228
Node 1	−0.288	0.021	−0.330	−0.248
Node 2	−0.044	0.023	−0.090	0.001
σ	0.199	0.005	0.190	0.208
σ_γ^2	0.052	0.014	0.032	0.091
ν	0.037	0.010	0.024	0.061



(a) Heat Map (b) Histogram

Figure 4.3: The heat map and histogram of estimated random effects in the AFT model for the GPU data.

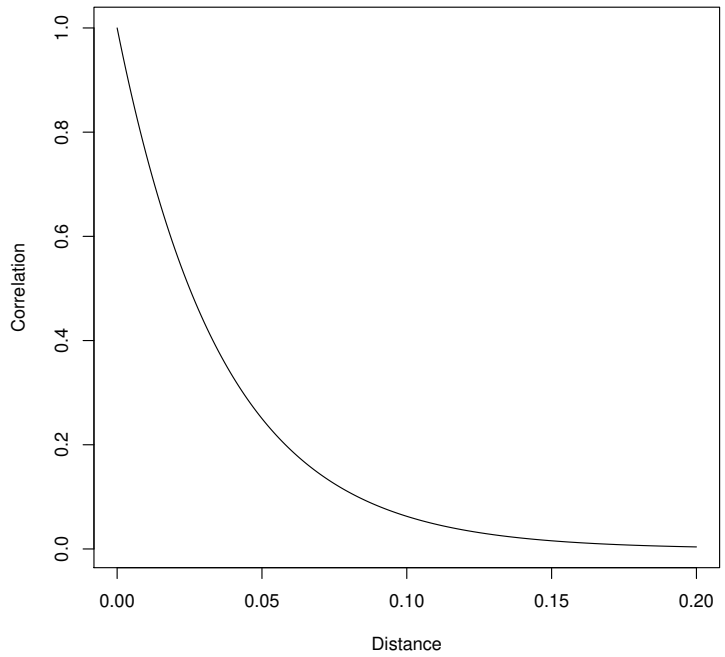


Figure 4.4: Estimated spatial correlation versus standardized distance in the GPU data.

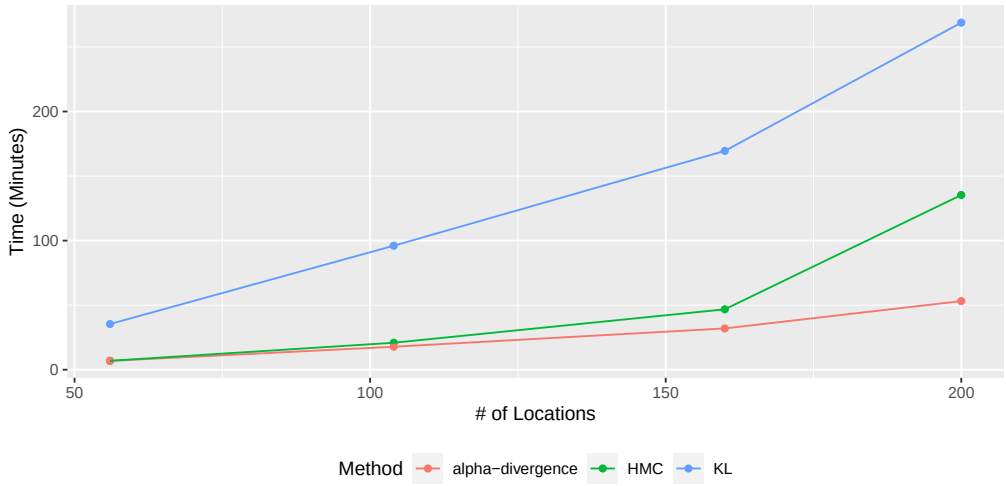


Figure 4.5: The computing time of three inference methods versus the number of locations with the GPU data.

4.4.2 Pine Trees Survival Data

4.4.2.1 Data Description

The pine tree dataset is from Li et al. (2015). In this dataset, 182 pine tree sites are selected throughout the Piedmont and Atlantic Coastal Plain regions. The survival and growth of trees with different living conditions and treatments are of interest. During each tree's lifetime, variables such as total height (TH), diameter at breast height (DBH), and crown class are recorded every three years up to 7 times. Other time-invariant variables such as physical region and thinning treatment (control, light thinning, and heavy thinning) applied to the tree are also collected. Table 4.3 presents the exploratory variables in the dataset. In total there are 42,525 pine trees at 182 sites. The event of interest is the death of a tree. A tree is recorded as censored if it survives till the 7th follow-up period. We are interested in studying the thinning treatment effect and spatial effect on the survival of pine trees. Due to the computational limitation, in this chapter, we randomly select 60 sites from the original dataset with 13,911 trees and conduct the analysis.

4.4.2.2 Bayesian PH model for Tree Data

For the pine tree dataset, since modeling the survival rate is of interest, we consider the spatial PH model with time-dependent covariates. We use a Weibull hazard with function form $h_0(t) = at^b$ as the baseline function. The Bayesian framework for spatial PH model is as follows:

$$\begin{aligned}
h_{ij}(t) &= at^b \exp\left[\boldsymbol{x}_{ij}(t)\boldsymbol{\beta} + \gamma_i\right], \\
\gamma &\propto \text{MVN}(\mathbf{0}, \sigma_\gamma^2 \Omega), \\
\boldsymbol{\beta}_p &\propto \mathbf{1}_p \quad a_l \propto 1, \text{where } a_l = \log(a), \\
b_l &\propto 1, \text{where } b_l = \log(b), \\
\sigma_\gamma^2 &\propto \text{IGAM}(a_\sigma, b_\sigma), \\
\nu &\propto \text{IGAM}(a_\nu, b_\nu).
\end{aligned}$$

We set weak priors $a_\sigma = b_\sigma = 1$ for σ_γ^2 and strong priors $a_\nu = 5, b_\nu = 0.1$ for ν. The log posterior can be derived as

$$\begin{aligned}
\log\left[p(\boldsymbol{\theta}|\mathcal{D})\right] =& \text{Constant} + \sum_{ij} \left\{\delta_{ij}\left[\gamma_i + a_l + b_l \log(t) + \boldsymbol{x}_{ij}(t_{ij})\boldsymbol{\beta}\right] - H_{ij}(t_{ij})\right\} \\
&- \frac{1}{2}\left[\log(\sigma_\gamma^2 \Omega) + \frac{\boldsymbol{\gamma}^\top \Omega^{-1} \boldsymbol{\gamma}}{\sigma_\gamma^2}\right] \\
&- \frac{1}{2}(a_\sigma + 1)\log(\sigma_\gamma^2) - \frac{b_\sigma}{\sigma_\gamma^2} - \frac{1}{2}(a_\nu + 1)\log(\nu) - \frac{b_\nu}{\nu}.
\end{aligned}$$

We assume the following variational distribution assumptions:

$$
\begin{aligned}
\boldsymbol{\beta} &\sim \text{MVN}(\boldsymbol{\mu}_{\boldsymbol{\beta}}, \Sigma_{\boldsymbol{\beta}}), \text{where } \Sigma_{\boldsymbol{\beta}} = \text{Diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\beta}}), \\
\boldsymbol{\gamma} &\sim \text{MVN}(\boldsymbol{\mu}_{\boldsymbol{\gamma}}, \Sigma_{\boldsymbol{\gamma}}), \text{where } \Sigma_{\boldsymbol{\gamma}} = \text{Diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\gamma}}), \\
a_l &\sim \text{N}(\mu_a, \sigma_a^2), \\
b_l &\sim \text{N}(\mu_b, \sigma_b^2), \\
\sigma_\gamma^2 &\propto \text{IGAM}(c_\sigma, d_\sigma), \\
\nu &\propto \text{IGAM}(c_\nu, d_\nu),
\end{aligned}
$$

where $\Sigma_{\boldsymbol{\beta}}$ and $\Sigma_{\boldsymbol{\gamma}}$ are diagonal matrices with diagonal elements being $\boldsymbol{\sigma}^2_{\boldsymbol{\beta}}$ and $\boldsymbol{\sigma}^2_{\boldsymbol{\gamma}}$. The variational distributions are independent to each other. Let $\boldsymbol{\sigma}_{l\boldsymbol{\beta}} = \log(\boldsymbol{\sigma}_{\boldsymbol{\beta}})$, $\boldsymbol{\sigma}_{l\boldsymbol{\gamma}} = \log(\boldsymbol{\sigma}_{\boldsymbol{\gamma}})$, $\sigma_{la} = \log(\sigma_a)$ and $\sigma_{lb} = \log(\sigma_b)$. Then the variational parameter vector can be denoted as $\boldsymbol{\eta} = \left(\boldsymbol{\mu}_{\boldsymbol{\beta}}^\top, \boldsymbol{\sigma}_{l\boldsymbol{\beta}}^\top, \boldsymbol{\mu}_{\boldsymbol{\gamma}}^\top, \boldsymbol{\sigma}_{l\boldsymbol{\gamma}}^\top, \mu_a, \sigma_{la}, \mu_b, \sigma_{lb}, c_\sigma, d_\sigma, c_\nu, d_\nu\right)^\top$. The log density of joint variational distribution is

$$
\begin{aligned}
\log\left[q(\boldsymbol{\theta}|\boldsymbol{\eta})\right] =& \text{Constant} - \frac{1}{2}\sum_j \left[2\sigma_{l\boldsymbol{\beta},j} + \frac{(\beta_j - \mu_{\boldsymbol{\beta},j})^2}{\exp(2\sigma_{l\boldsymbol{\beta},j})}\right] \\
&- \frac{1}{2}\sum_j \left[2\sigma_{l\boldsymbol{\gamma},j} + \frac{(\gamma_j - \mu_{\boldsymbol{\gamma},j})^2}{\exp(2\sigma_{l\boldsymbol{\gamma},j})}\right] \\
&- \frac{1}{2}\left[2\sigma_{la} + \frac{(a_l - \mu_a)^2}{\exp(2\sigma_{la})}\right] - \frac{1}{2}\left[2\sigma_{lb} + \frac{(b_l - \mu_b)^2}{\exp(2\sigma_{lb})}\right] \\
&+ c_\sigma \log(d_\sigma) - \log[\Gamma(c_\sigma)] - (c_\sigma + 1)\log(\sigma_\gamma^2) - \frac{d_\sigma}{\sigma_\gamma^2} \\
&+ c_\nu \log(d_\nu) - \log[\Gamma(c_\nu)] - (c_\nu + 1)\log(\nu) - \frac{d_\nu}{\nu}.
\end{aligned}
$$

Similar to Section 4.4.1, we can apply VI based on the unnormalized posterior $\log\left[p(\boldsymbol{\theta}, \mathcal{D})\right]$ and variational distribution $\log\left[q(\boldsymbol{\theta}|\boldsymbol{\eta})\right]$.

4.4.2.3 Analysis Results

For a PH model with time-dependent covariates, the Cox-Snell residual is defined as

$$\widehat{\epsilon}_{ij} = \int_0^{T_{ij}} \widehat{h}_0(t) \exp[\boldsymbol{x}_{ij}(t)^\top \widehat{\boldsymbol{\beta}} + \widehat{\gamma}_i] dt,$$

where $\widehat{H}_0(T_{ij})$ is the estimated cumulative baseline hazard rate by plugging in $\widehat{a}$ and $\widehat{b}$. If the model is correct, then $\widehat{\epsilon}_{ij}$ approximately follows exponential distribution with $\lambda = 1$ and censoring, which is a special case of Weibull distribution (Klein and Moeschberger, 2003). Therefore, we can use Weibull probability plot of residuals to check three inference methods posterior samples.

Figure 4.6 shows the probability plot of residuals for $\alpha-$divergence, KL divergence and HMC. We can see although there exists curvature at the tail of $\alpha-$divergence residual, it is close to the HMC method. The KL divergence residual far more deviates from the standard Weibull distribution. Table 4.4 summarizes the NLL and computational time of three methods. We can see the $\alpha-$divergence method is close to the NLL of HMC while taking much less computational time.

Since the statical inference of the $\alpha-$divergence is very close to HMC and it takes less computing time, its inference results are presented. Table 4.5 shows the parameter estimates in the spatial PH model and 95% CIs. We can see that if fixed other covariates, control and light thinning treatment decrease the survival rates compared to heavy thinning. Pine trees in the Piedmont region have higher survival rates compared to Coastal Plain and other types. DBH tends to extend trees' lifetime while TH tends to shorten trees' lifetime. Figure 4.7(a) shows the estimated spatial correlation versus distance. Since the distance matrix is standardized when fitting the model, now the range of the distance changes to $(0.0018, 1.2882)$. This indicates that for sites that are near each other, spatial correlations are strong. Figure 4.7(b) shows the histogram of estimated random effect, which are in the range of $(-2, 2)$. Figure 4.8 shows the spatial random effect at each site on the map. For Virginia and North Carolina areas, the spatial random effect increases when moving from the coastal to the inland area,

Table 4.3: The description and summary statistics of explanatory variables in the dataset.

Variable	Description	N (%)
SN	A serial number assigned to each tree	13911 (100%)
Thinning Intensity	1-control	3351 (24.10%)
	2-light thinning	5647 (40.59%)
	3-heavy thinning	4913 (35.32%)
Physical Region	1-Coastal Plain	7786 (55.97%)
	2-Piedmont	4839 (34.79%)
	3-other types	1286 (9.24%)
Crown Class	1-dominant	3548 (25.50%)
	2-codominant	6583 (47.32%)
	3-intermediate	2584 (18.58%)
	4-suppressed	1196 (8.60%)
Variable	Description	Mean (SD)
DBH	Diameter at breast height (cm)	7.87 (2.68)
TH	Total height of the tree (meters)	56.28 (16.25)

Table 4.4: The negative log likelihood and computing time of $\alpha-$divergence, KL divergence and HMC inferences with pine tree data.

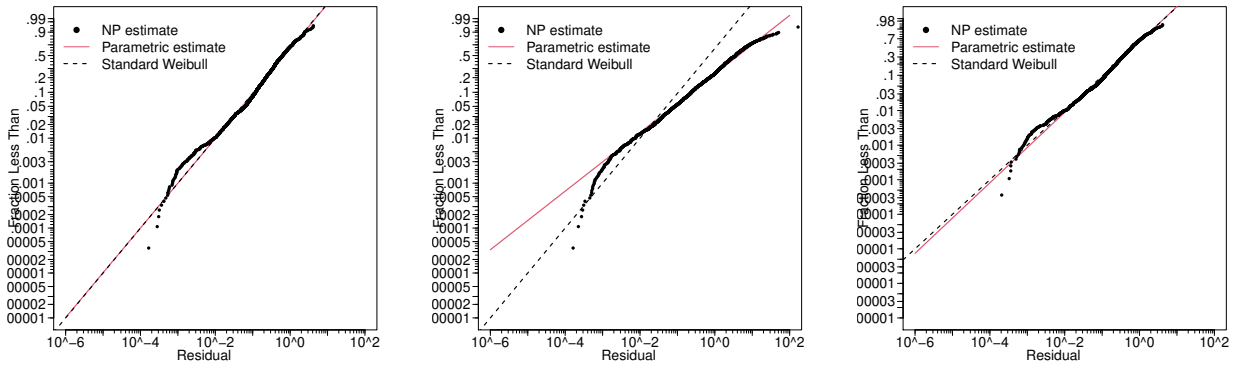
	$\alpha = 0.8$	KL	HMC
NLL	12341.71	18985.41	12332.84
Time (hours)	7.97	8.52	16.29

which indicates the survival rate becomes lower.

4.5 Conclusion and Future Directions

In this study, we compare the performance of VI and HMC for spatial survival models with two real applications. The Titan GPU lifetime data is used to illustrate the spatial AFT model and the pine tree survival data is used to illustrate the spatial PH model. The computing time and NLL of $\alpha-$divergence, KL divergence, and HMC are compared. The statistical inferences using the $\alpha-$divergence posterior samples are also presented.

In the Titan GPU lifetime application, $\alpha-$divergence has the smallest NLL and shortest computing time, indicating its capability on spatial survival models. Although the computing

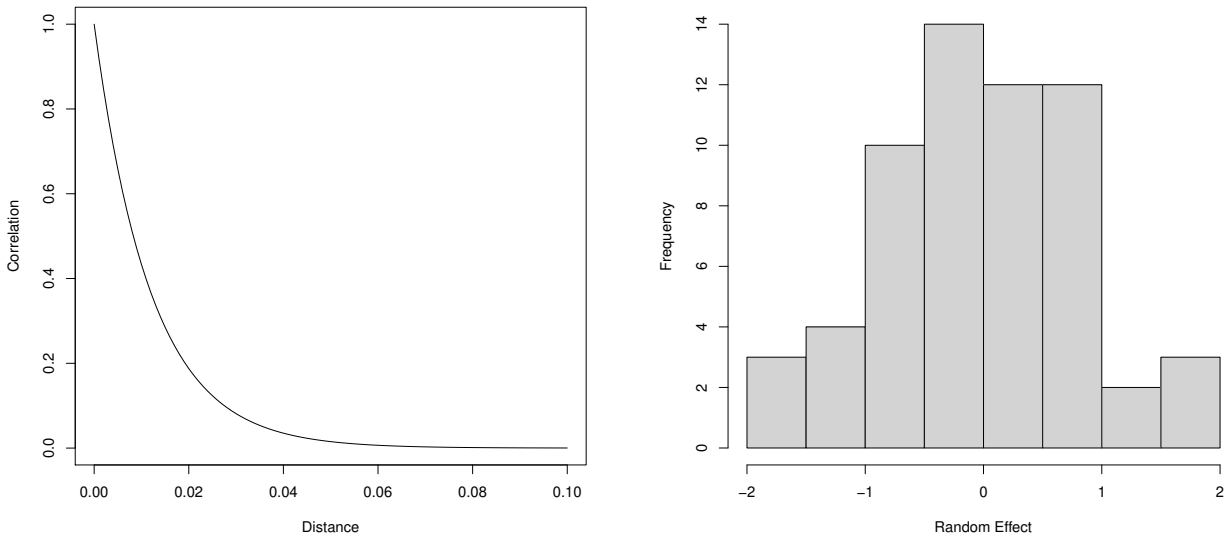


(a) $\alpha = 0.8$ (b) KL (c) HMC

Figure 4.6: Weibull probability plot of residuals for $\alpha-$divergence, KL divergence and HMC with the pine tree data.

Table 4.5: Parameter estimates and 95% CIs for pine tree data.

Parameter	Est	SE	CI Lower	CI Upper
Treatment (Control)	0.969	0.058	0.852	1.088
Treatment (Light)	0.152	0.059	0.043	0.271
Physical region (Coastal Plain)	−0.537	0.048	−0.630	−0.441
Physical region (Piedmont)	−0.894	0.067	−1.028	−0.758
Crown class (dominant)	−2.385	0.063	−2.503	−2.263
Crown class (codominant)	−2.312	0.052	−2.409	−2.209
Crown class (intermediate)	−1.208	0.045	−1.300	−1.117
DBH	−0.277	0.018	−0.314	−0.241
TH	0.004	0.003	−0.000	0.009
a	0.043	0.003	0.037	0.049
b	0.929	0.016	0.900	0.960
σ^2_γ	1.309	0.321	0.856	2.096
ν	0.012	0.002	0.009	0.016



(a) Spatial Correlation (b) Random Effects

Figure 4.7: Estimated spatial correlation versus distance and histogram of random effects for pine tree data.

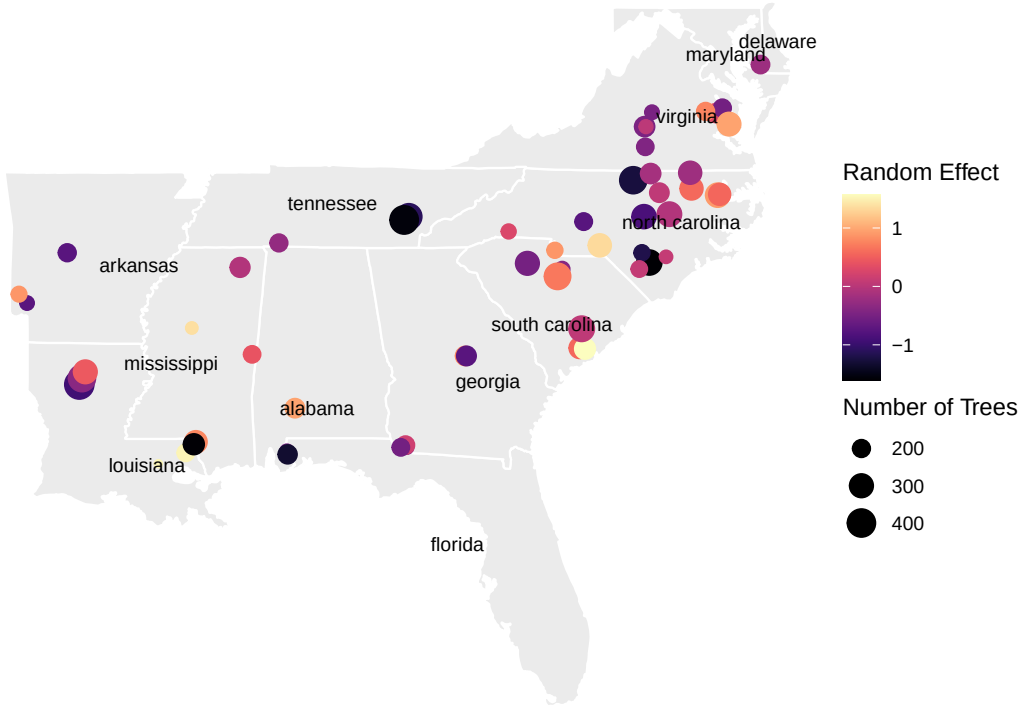


Figure 4.8: Estimated random effects at different locations for pine tree data.

time is only a few minutes shorter than HMC, the code of VI is implemented with R while the source code of HMC in Stan is implemented with C++. In the future, it will be useful to also implement VI with C++ for a more fair computing time comparison. In the pine tree dataset, the NLL of $\alpha-$divergence is close to HMC while the computing time is about half of HMC. Compared to KL divergence, $\alpha-$divergence encourages a more flexible variational distribution, thus it has better performance regarding statistical inference. Besides, $\alpha-$divergence is also easier to converge compared to KL divergence. Based on these two applications, we find $\alpha-$divergence with $\alpha < 1$ has comparable performance as HMC but with better computational efficiency.

In the real data analysis, we want the $\alpha-$divergence to encourage a flexible variational distribution behavior so we set $\alpha = 0.8$. However, it will be interesting to study how the statistical inference performance changes with different α values. It is also desirable to study how to choose a reasonable α value with various applications and models. Besides, it will be interesting to investigate the accuracy of the uncertainty quantification of VI for spatial survival models. It is also desirable to consider using a mixture variational distribution or other flexible structures to capture the possible multi-mode behavior of posterior distributions in spatial survival models.

Appendix 4.A The Gradient of ELBO and VR Bound

The derivative of ELBO $\mathcal{L}_{\mathrm{VI}}$ with respect to variational parameter $\boldsymbol{\eta}$ is:

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\mathrm{VI}}}{\partial \boldsymbol{\eta}} &= \frac{\partial \mathcal{L}_{\mathrm{VI}}}{\partial \log[q(\boldsymbol{\theta}|\boldsymbol{\eta})]} \frac{\partial \log[q(\boldsymbol{\theta}|\boldsymbol{\eta})]}{\partial \boldsymbol{\eta}} \\
&= \left\{ \frac{\partial}{\partial \log[q(\boldsymbol{\theta}|\boldsymbol{\eta})]} \int q(\boldsymbol{\theta}|\boldsymbol{\eta}) \log[p(\boldsymbol{\theta}, \mathcal{D})] - q(\boldsymbol{\theta}|\boldsymbol{\eta}) \log[q(\boldsymbol{\theta}|\boldsymbol{\eta})] d\boldsymbol{\theta} \right\} \frac{\partial \log[q(\boldsymbol{\theta}|\boldsymbol{\eta})]}{\partial \boldsymbol{\eta}} \\
&= \int q(\boldsymbol{\theta}|\boldsymbol{\eta}) \log[p(\boldsymbol{\theta}, \mathcal{D})] \frac{\partial \log[q(\boldsymbol{\theta}|\boldsymbol{\eta})]}{\partial \boldsymbol{\eta}} d\boldsymbol{\theta} - \int q(\boldsymbol{\theta}|\boldsymbol{\eta}) \left\{ \log[q(\boldsymbol{\theta}|\boldsymbol{\eta})] + 1 \right\} \frac{\partial \log[q(\boldsymbol{\theta}|\boldsymbol{\eta})]}{\partial \boldsymbol{\eta}} d\boldsymbol{\theta} \\
&= \int q(\boldsymbol{\theta}|\boldsymbol{\eta}) \left\{ \log \left[\frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta}|\boldsymbol{\eta})} - 1 \right] \right\} \frac{\partial \log[q(\boldsymbol{\theta}|\boldsymbol{\eta})]}{\partial \boldsymbol{\eta}} d\boldsymbol{\theta}.
\end{aligned}$$

The derivative of VR bound $\mathcal{L}_{\alpha}$ with respect to variational parameter $\boldsymbol{\eta}$ is:

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\alpha}}{\partial \boldsymbol{\eta}} &= \frac{\partial \mathcal{L}_{\alpha}}{\partial \log[q(\boldsymbol{\theta}|\boldsymbol{\eta})]} \frac{\partial \log[q(\boldsymbol{\theta}|\boldsymbol{\eta})]}{\partial \boldsymbol{\eta}} \\
&= \left(\frac{1}{1-\alpha} \frac{\partial}{\partial \log[q(\boldsymbol{\theta}|\boldsymbol{\eta})]} \log \left\{ \int q(\boldsymbol{\theta}|\boldsymbol{\eta}) \left[\frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta}|\boldsymbol{\eta})} \right]^{1-\alpha} d\boldsymbol{\theta} \right\} \right) \frac{\partial \log[q(\boldsymbol{\theta}|\boldsymbol{\eta})]}{\partial \boldsymbol{\eta}} \\
&= \frac{\alpha}{1-\alpha} \frac{\int q(\boldsymbol{\theta}|\boldsymbol{\eta}) \left[\frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta}|\boldsymbol{\eta})} \right]^{1-\alpha} \frac{\partial \log[q(\boldsymbol{\theta}|\boldsymbol{\eta})]}{\partial \boldsymbol{\eta}} d\boldsymbol{\theta}}{\int q(\boldsymbol{\theta}|\boldsymbol{\eta}) \left[\frac{p(\boldsymbol{\theta}, \mathcal{D})}{q(\boldsymbol{\theta}|\boldsymbol{\eta})} \right]^{1-\alpha} d\boldsymbol{\theta}}.
\end{aligned}$$

Bibliography

S. Banerjee, M. M. Wall, and B. P. Carlin. Frailty modeling for spatially correlated survival data, with application to infant mortality in Minnesota. *Biostatistics*, 4(1):123–142, 2003.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

P. Boškoski, M. Perne, M. Rameša, and B. M. Boshkoska. Variational Bayes survival analysis for unemployment modelling. *Knowledge-Based Systems*, 229:107335, 2021.

B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software*, 76(1), 2017.

D. Darmofal. Bayesian spatial survival models for political event processes. *American Journal of Political Science*, 53(1):241–257, 2009.

U. Diva, D. K. Dey, and S. Banerjee. Parametric models for spatially correlated survival data for individuals with multiple cancers. *Statistics in Medicine*, 27(12):2127–2144, 2008.

R. Henderson, S. Shimakura, and D. Gorst. Modeling spatial variation in leukemia survival data. *Journal of the American Statistical Association*, 97(460):965–972, 2002.

J. Hernandez-Lobato, Y. Li, M. Rowland, T. Bui, D. Hernández-Lobato, and R. Turner. Black-box alpha divergence minimization. In *International Conference on Machine Learning*, pages 1511–1520. PMLR, 2016.

M. D. Hoffman and D. M. Blei. Structured stochastic variational inference. In *Artificial Intelligence and Statistics*, pages 361–369, 2015.

M. D. Hoffman, A. Gelman, et al. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

A. W. Jung and M. Gerstung. Bayesian Cox regression for population-scale inference in electronic health records. *arXiv preprint arXiv:2106.10057*, 2021.

M. Kim and V. Pavlovic. Variational inference for Gaussian process models for survival analysis. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 435–445, 2018.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

J. P. Klein and M. L. Moeschberger. *Survival analysis: techniques for censored and truncated data*, volume 2. Springer, 2003.

A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 2017.

J. Li, Y. Hong, R. Thapa, and H. E. Burkhart. Survival analysis of Loblolly pine trees with spatially correlated random effects. *Journal of the American Statistical Association*, 110 (510):486–502, 2015.

Y. Li and R. E. Turner. Rényi divergence variational inference. *Advances in Neural Information Processing Systems*, 29, 2016.

J. Min, Y. Hong, and M. William. Spatially correlated time-to-event model for Titan GPU failures data under competing risks, in progress. 2022.

G. Ostrouchov, D. Maxwell, R. A. Ashraf, C. Engelmann, M. Shankar, and J. H. Rogers. GPU lifetimes on Titan supercomputer: Survival analysis and reliability. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–14. IEEE, 2020.

R. Ranganath, D. Tran, and D. Blei. Hierarchical variational models. In *International Conference on Machine Learning*, pages 324–333. PMLR, 2016.

Z. Xiu, C. Tao, and R. Henao. Variational learning of individual survival distributions. In *Proceedings of the ACM Conference on Health, Inference, and Learning*, pages 10–18, 2020.

L. Zhao, T. E. Hanson, and B. P. Carlin. Mixtures of Polya trees for flexible spatial frailty survival modelling. *Biometrika*, 96(2):263–276, 2009.

Chapter 5 General Conclusions and Areas for Future Research

5.1 Conclusions

In this dissertation, I have focused on several advancements in the interface of computer experiments and survival analysis with real world applications. Specifically, Chapter 2 shows the usage of the computer experiments to enhance the data collection in high performance computing setting to manage the variability. Chapter 3 focuses on the reliability analysis, which uses survival analysis techniques in engineering areas and proposes a flexible framework for building degradation index. Chapter 4 studies the capability of VI on the spatial survival analysis under the Bayesian framework.

In Chapter 2, comprehensive comparisons are conducted for synthetic data analysis and HPC variability management application to explore the prediction accuracy of GBDs and SFDs under different approximation methods. The comparison shows that the GP model with SFD generates the best results under most scenarios.

In Chapter 3, a new framework to build the degradation index based on the cumulative exposure model is proposed, motivated by the jet engine multi-channel sensory data. The framework can handle censored data and conduct variable selection automatically. The comprehensive simulation studies and jet engine data analysis show that the framework has flexibility and advantages. It is also demonstrated that the performance of the proposed framework is more robust than other models.

In Chapter 4, the performance of VI and HMC for spatial survival models are compared with two real applications. The Titan GPU lifetime data is used to illustrate the spatial AFT model and the pine tree survival data is used to illustrate the spatial PH model. The results show that $\alpha-$divergence has comparable performance as HMC but with less computing time.

5.2 Areas for Future Research

In the future, it will be interesting to investigate the accuracy of uncertainty quantification of VI for spatial survival models in Chapter 4. For some spatial survival models, the posterior distributions may potentially have multi-mode behavior. It is desirable to extend the standard VI procedure by considering mixture variational distribution or other flexible structures that allow dependencies between parameters to capture the potential multi-mode behavior of posteriors.