Maximum Penalized Likelihood Estimation For Semi-parametric Regression Models With Partly Interval-Censored Failure Time Data

Jinqing Li

Bachelor of Actuarial studies, Macquarie University

Master of Applied statistics, Macquarie University

This thesis is presented for the degree of Doctor of Philosophy

Department of Statistics

Faculty of Science

Macquarie University , Australia

August 2014

Contents

List of Figures

List of Tables

Statement of Candidate

I certify that the work in this dissertation entitled "Maximum Penalized Likelihood Estimation For Semi-parametric Regression Models With Partly Interval-Censored Failure Time Data" has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree to any other university or institution other than Macquarie University.

I also certify that the dissertation is an original piece of research and it has been written by me. Any assistance that I have received in my research work and the preparation of the thesis itself have been appropriately acknowledged.

In addition, I certify that all information sources and literature used are indicated in the dissertation.

Signature:

Jinqing Li (41305728)

August, 2014

Acknowledgments

I would like to express my heartfelt gratitude to my principal supervisor A/Prof Jun Ma for his great encouragement and patience, and guiding and training me towards being an independent researcher. I have benefited too much from our discussions.

I would like to thank A/Prof Gillian Heller and Dr Ken Beath for their willingness to help and give best suggestions for writing tasks.

I would like to thank Jing Xu, Zhipeng Hao, Andrew Grant, Madawa Weerasinghe and Kasun Rathnayake for their valuable comments when editing this dissertation.

I warmly thank the entire Department of Statistics at Macquarie University for their constant support and for providing excellent academic atmosphere for doing research.

It is a pleasure to acknowledge the companionship from the graduate students who have crossed my path during my academic life at Macquarie University.

Finally, I would also like to thank my parents and girl friend for their belief in me, and for supporting me and encouraging me with their best wishes.

Abstract

Interval-censored failure time data arise in many areas including demographical, financial, actuarial, medical and sociological studies. By interval censoring, we mean that the failure time is not always exactly observed and we can only observe an interval within which the failure event has occurred. The goal of this dissertation is to develop maximum penalized likelihood (MPL) methods for proportional hazard (PH), additive hazard (AH) and accelerated failure time (AFT) models with partly interval-censored failure time data, which contains exactly observed, left-censored, finite interval-censored and right-censored data. We fit these three semi-parametric regression models by estimating the underlying non-parametric baseline hazard functions and regression coefficients. For the PH and AFT models, we compute these estimates simultaneously using the Newton and multiplicative iterative (Newton-MI) algorithm with line search steps, where the non-negativity of baseline hazard functions is imposed in a direct way. For the AH model, we obtain the estimates using the primal-dual interior point algorithm, with which the baseline hazard function and hazard function are constrained to be non-negative simultaneously. The MPL methods provide smoothness for the baseline hazard estimates, which can clearly show the trend of how the baseline hazard estimates are changing over time. The asymptotic properties of these MPL estimators are studied.

We investigate the performance of our proposed MPL methods by conducting simulation studies, and the simulation results demonstrate that our methods work well. In addition, we also make comparisons between our MPL methods and some existing methods. In a real data analysis, the proposed MPL methods are applied to the AIDS example provided by Lindsey and Ryan (1998).

Keywords: Interval-censored failure time data; Proportional hazard model; Additive hazard model; Accelerated failure time model; Maximum penalized likelihood; Newton-MI algorithm; Primal-dual interior point algorithm; Cross validation.

Chapter 1

Introduction

1.1 Failure time data

Failure time data concerns positive random variables representing times to a certain event, such as death, the onset of a disease or the failure of a mechanical component of a machine. Failure time data arises extensively in medical studies, but there are many other areas where it may also appear. These areas include biological studies, demographic studies, actuarial studies and sociological studies.

Censoring is the main feature of failure time data. By censoring, we mean that the failure time is observed only to fall into a certain range, instead of being known exactly. Hence censored data only provides partial information. Truncation is another feature existing in failure time data. This refers to cases where a subject is included in a study only if the corresponding failure time satisfies certain conditions. For example, in a cohort study, subjects are included in the study only if they experience an initial event prior to the failure event, hence their failure times are greater than the occurrence time of the initial event. This type of truncation is commonly known as left-truncation. In our dissertation, we only analyze failure time data with censoring, and do not consider truncation. Next

we give definitions of four types of censoring.

Suppose there are n independent subjects in a survival study. For each subject i, $i = 1, \cdots, n$, we denote T_i as the failure time, C_i as the censoring time if T_i is not exactly observed, and $\boldsymbol{X}_i$ as a $p \times 1$ vector of covariates. The definition of right censoring is that the failure time T_i is not exactly observed, but is known to be greater than the censoring time C_i. Left censoring is defined as when T_i is known only to be less than C_i. Current status data arises when each subject is observed only once and the corresponding failure time T_i can only be determined to lie above or below a censoring time C_i. In other words, we can only observe

$$\{C_i, I(T_i \leq C_i), \boldsymbol{X}_i; i = 1, \cdots, n\}, \tag{1.1}$$

where $I(\mathrm{A})$ is an indicator function for event A. Current status data is commonly encountered in biomedicine, economics, sociology and other scientific areas. For example, in carcinogenicity experiments, animals are randomly assigned to various doses of a suspected carcinogen and are examined at sacrifice or death time for evidence of a malignancy. Here the time to tumour onset is of interest, but not directly observable. Rather, we only know at the age of death whether the tumour is present or not.

Interval censoring, which is the focus of the thesis, is another type of censoring. By interval censoring, we mean that the failure time T_i is not always exactly observed and we can only observe an interval within which the failure event has occurred. A typical example of interval censoring is in medical or health studies that entail periodic follow-ups. Many clinical trials and longitudinal studies fall into this category. In such situations, interval-censored data may arise in several ways. For instance, an individual may miss one or more observation times that have been scheduled to clinically observe possible changes in disease status, and then return with a changed status. Alternatively, individuals may choose convenient times to visit clinical centers rather than visiting at predetermined

observation times. Under both situations, the data on status changes is interval-censored.

1.2 Formulations for interval-censored data

In this section, we define some notation for interval-censored data. From the definition of interval censoring given in Section 1.1, we know that, for each subject i, there is an observed interval $(L_i, R_i]$ such that the corresponding failure time T_i satisfies

$$T_i \in (L_i, R_i], \quad i = 1, \cdots, n \tag{1.2}$$

with $L_i < R_i$. Note that, instead of $(L_i, R_i]$, we can also present interval-censored data using $[L_i, R_i]$, $[L_i, R_i)$, or (L_i, R_i) (Peto, 1973; Turnbull, 1976). However, since we assume that T_i is continuous throughout the thesis, there is no difference among these expressions and they all represent the same observed information about T_i. We say the failure time T_i is finite interval-censored if the observed interval $(L_i, R_i]$ satisfies $0 < L_i < R_i < \infty$. In the case where $L_i = R_i$, we mean T_i is exactly observed, $L_i = 0$ means T_i is left-censored and $R_i = \infty$ means T_i is right-censored. Therefore, interval-censored data contains left-censored, right-censored and finite interval-censored data. We can regard current status data (1.1) as a special case of interval-censored data. We define partly interval-censored data as a combination of exactly observed and interval-censored data, and we analyze partly interval-censored data throughout the thesis.

Another way to represent interval-censored data by Sun (2006) is based on the assumption that each subject i in a survival study is observed twice, so we have

$$\{C_{i1}, C_{i2}, \xi_{i1} = I(T_i \leq C_{i1}), \xi_{i2} = I(C_{i1} < T_i \leq C_{i2}), \xi_{i3} = 1 - \xi_{i1} - \xi_{i2}; i = 1, \cdots, n\}, \tag{1.3}$$

where we call C_{i1} and C_{i2} the monitoring random variables, satisfying $C_{i1} < C_{i2}$. With the formulation (1.3), we can easily obtain the corresponding data with representation (1.2).

Specifically, for the indicator functions in (1.3), if $\xi_{i1} = 1$, the failure time of subject i is left-censored, and we have $L_i = 0$ and $R_i = C_{i1}$; if $\xi_{i2} = 1$, it is finite interval-censored, and we have $L_i = C_{i1}$ and $R_i = C_{i2}$; and if $\xi_{i3} = 1$, it is right-censored, and we have $L_i = C_{i2}$ and $R_i = \infty$.

We can also obtain a more generalized form of (1.3) by assuming that each subject i is observed more than twice, so that there exists a sequence of monitoring time points, say $C_{i1} \leq C_{i2} \leq \cdots \leq C_{iM_i}$, where M_i is the number of monitoring time points for subject i. Then the observed information is expressed in the form of

$$\{M_i, C_{ij}, \xi_{ij} = I(C_{ij-1} < T_i \leq C_{ij}), i = 1, \cdots, n; j = 1, \cdots, M_i\}, \tag{1.4}$$

where $C_{i0} = 0$. It is apparent that the above formulation provides a natural representation of interval-censored data arising from longitudinal studies with periodic follow-up. For data given in the formulation (1.4), if $\xi_{ij} = 1$, then we have $L_i = C_{ij-1}$ and $R_i = C_{ij}$. If, on the other hand, $\xi_{ij} = 0$ for all j, then T_i is right-censored and we have $L_i = C_{iM_i}$ and $R_i = \infty$. The formulations (1.2)-(1.4) will be used in Chapter 2 for literature review.

1.3 Some functions

In this section, we introduce some basic functions used in survival studies. The hazard function $h(t)$ is defined as the instantaneous probability that a subject fails at time t given that the subject has not failed before t (Kalbfeisch and Prentice, 2002). It is expressed as

$$h(t) = \lim_{\Delta t \to 0^+} \frac{Pr(t \leq T < t + \Delta t | T \geq t)}{\Delta t}. \tag{1.5}$$

It is also called the force of mortality in demography and actuarial science. It must be non-negative, i.e. $h(t) \geq 0$. It may be increasing, decreasing or non-monotonic, depending on the risk characteristics of subjects. We denote $f(t)$ and $S(t)$ as the density and survival functions of failure time at t, respectively. There is a one-to-one relationship among the

survival, density and hazard functions. To be specific, given the density and survival functions, we have

$$h(t) = \frac{f(t)}{S(t)} = -\frac{d \log S(t)}{dt}.$$

On the other hand, given the hazard function, we have

$$S(t) = e^{-\int_0^t h(s)ds} = e^{-H(t)} \tag{1.6}$$

and

$$f(t) = h(t)e^{-H(t)}, \tag{1.7}$$

where $H(t) = \int_0^t h(s)ds$, which is called the cumulative hazard function of T at time t.

1.4 Three semi-parametric regression models

Semi-parametric regression analysis is used to to assess covariate effects on the failure time or the hazard function. In this section, we describe three semi-parametric regression models analyzed throughout the thesis. They are the proportional hazard (PH), additive hazard (AH) and accelerated failure time (AFT) models.

1.4.1 Proportional Hazards (PH) Model

For each subject i, the PH model specifies the hazard function, $h(t|\boldsymbol{X}_i)$, at time point t according to

$$h(t|\boldsymbol{X}_i) = h_0(t) \exp\left\{\boldsymbol{X}_i^T \boldsymbol{\beta}\right\}, \tag{1.8}$$

where $h_0(\cdot)$ is an unspecified baseline hazard function, $\boldsymbol{\beta}$ is a $p \times 1$ vector of regression coefficients, usually of primary interest in the model fitting, and $\boldsymbol{X}_i$ is a covariate vector. The PH model relates covariate effects multiplicatively to the hazard. Now consider two subjects with their covariate vectors denoted by $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$. Then the ratio of their

hazards at time t is

$$\frac{h(t|\boldsymbol{X}_i)}{h(t|\boldsymbol{X}_j)} = \frac{h_0(t)\exp\left\{\boldsymbol{X}_i^T\boldsymbol{\beta}\right\}}{h_0(t)\exp\left\{\boldsymbol{X}_j^T\boldsymbol{\beta}\right\}} = \exp\left\{(\boldsymbol{X}_i - \boldsymbol{X}_j)^T\boldsymbol{\beta}\right\}.$$

Hence we conclude that the hazards are proportional to each other and do not depend on time, i.e., $h(t|\boldsymbol{X}_i) \propto h(t|\boldsymbol{X}_j)$. From (1.8), the corresponding cumulative hazard function is

$$H(t|\boldsymbol{X}_i) = \int_0^t h(s|\boldsymbol{X}_i)ds = H_0(t)\exp\left\{\boldsymbol{X}_i^T\boldsymbol{\beta}\right\}, \tag{1.9}$$

where $H_0(t) = \int_0^t h_0(s)ds$, which is called the cumulative baseline hazard function. Using equations (1.6) and (1.7), the conditional survival and density functions of T_i given $\boldsymbol{X}_i$ have the form

$$\begin{aligned} S(t|\boldsymbol{X}_i) &= \exp\left\{-H_0(t)\exp(\boldsymbol{X}^T\boldsymbol{\beta})\right\} \\ &= [S_0(t)]^{\exp(\boldsymbol{X}_i^T\boldsymbol{\beta})} \end{aligned}$$

and

$$f(t|\boldsymbol{X}_i) = h_0(t)\exp(\boldsymbol{X}_i^T\boldsymbol{\beta})\exp\left\{-H_0(t)\exp(\boldsymbol{X}_i^T\boldsymbol{\beta})\right\},$$

where $S_0(\cdot)$ is the baseline survival function.

1.4.2 Additive Hazards (AH) Model

The AH model specifies the hazard function by summation of the baseline hazard and the regression function of covariates. To be specific, it is assumed that given $\boldsymbol{X}_i$, the hazard function of T_i has the additive form given by

$$h(t|\boldsymbol{X}_i) = h_0(t) + \boldsymbol{X}_i^T\boldsymbol{\beta}. \tag{1.10}$$

Hence the effect of covariates is to additively increase or decrease the hazard function. The corresponding cumulative hazard function is

$$H(t|\boldsymbol{X}_i) = H_0(t) + \boldsymbol{X}_i^T\boldsymbol{\beta}t. \tag{1.11}$$

Under the AH model, the conditional survival and density functions of T_i given $\boldsymbol{X}_i$ are

$$S(t|\boldsymbol{X}_i) = S_0(t)\exp\{-\boldsymbol{X}_i^T\boldsymbol{\beta}t\}$$

and

$$f(t|\boldsymbol{X}_i) = [h_0(t) + \boldsymbol{X}_i^T\boldsymbol{\beta}]\exp\{-[H_0(t) + \boldsymbol{X}_i^T\boldsymbol{\beta}t]\}.$$

Under the AH model, $\boldsymbol{\beta}$ can be understood as the risk difference. In particular, for two subjects with $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$, we have

$$h(t|\boldsymbol{X}_i) - h(t|\boldsymbol{X}_j) = (\boldsymbol{X}_i - \boldsymbol{X}_j)^T\boldsymbol{\beta}.$$

1.4.3 Accelerated Failure Time (AFT) Model

The AFT model assumes that the logarithm of failure time is associated linearly with covariates,

$$\log T_i = \boldsymbol{X}_i^T\boldsymbol{\beta} + \epsilon_i, \tag{1.12}$$

where ϵ_i is an error variable with an unknown distribution function. Under the AFT model, the effect of covariates is also considered to be multiplicative as in the PH model, but on the failure time T_i instead of the hazard function. In other words, the effect is to change the timescale and therefore to accelerate or decelerate the time to failure.

We define $\epsilon_i^* = e^{\epsilon_i}$. Let $h_{\epsilon_i^*}(\cdot)$ denote the hazard function of ϵ_i^*, which is independent of $\boldsymbol{\beta}$. Using definition (1.5) and model (1.12), we can get a relationship between the hazard function of failure time T_i and the hazard function of ϵ_i^*, that is

$$h(t|\boldsymbol{X}_i) = e^{-\boldsymbol{X}_i^T\boldsymbol{\beta}}h_{\epsilon_i^*}(te^{-\boldsymbol{X}_i^T\boldsymbol{\beta}}). \tag{1.13}$$

The relationship (1.13) will be used in estimation for the AFT model in Chapter 5. The corresponding cumulative hazard function is then given by

$$H(t|\boldsymbol{X}_i) = \int_0^t h(s|\boldsymbol{X}_i)ds = H_{\epsilon_i^*}(te^{-\boldsymbol{X}_i^T\boldsymbol{\beta}}), \tag{1.14}$$

where $H_{\epsilon^*}(t) = \int_0^t h_{\epsilon_i^*}(s)ds$. The conditional survival and density functions of T_i given $\boldsymbol{X}_i$ are expressed as

$$S(t|\boldsymbol{X}_i) = e^{-H_{\epsilon_i^*}(te^{-\boldsymbol{X}_i^T\boldsymbol{\beta}})}$$

and

$$f(t|\boldsymbol{X}_i) = e^{-\boldsymbol{X}_i^T\boldsymbol{\beta}}h_{\epsilon_i^*}(te^{-\boldsymbol{X}_i^T\boldsymbol{\beta}})e^{-H_{\epsilon_i^*}(te^{-\boldsymbol{X}_i^T\boldsymbol{\beta}})}.$$

1.5 Existing methods

There exists an extensive literature for fitting the three semi-parametric regression models, discussed in Section 1.4, with right-censored data. Cox (1975) develops a partial likelihood (PL) approach to estimate the regression coefficient $\boldsymbol{\beta}$ under the PH model, where the baseline hazard $h_0(\cdot)$ is considered as a nuisance quantity and is not estimated. But the baseline hazard can be obtained using the Breslow estimator (Breslow, 1972), and smoothed estimates of $h_0(\cdot)$ can be derived by kernel (Gray, 1990), or penalized likelihood (Anderson and Senthilselvan, 1980; Ma et al., 2014). For the AH model, Lin and Ying (1994) construct an estimating function for $\boldsymbol{\beta}$, which mimics the martingale feature of the partial likelihood score function in the PH model. The estimator for the cumulative baseline hazard function parallels the Breslow estimator for the corresponding quantity under the PH model. For the estimation in the AFT model, Miller(1976), Buckley and James (1979), and Jin et al. (2006) apply a least squares method. Tsiatis (1990), Ritov (1990), Jin et al. (2003), Park and Wei (2003), and Gay (2000) develop linear rank statistics as estimation functions for the regression coefficient $\boldsymbol{\beta}$. Zeng and Lin (2007) propose a likelihood-based method, where they construct a smooth approximation to a profile likelihood function using a kernel function, and $\boldsymbol{\beta}$ is estimated by maximizing the kernel-smoothed profile log-likelihood function through a Quasi-Newton search algorithm. In this dissertation, we devote ourselves to analyzing interval-censored data. Below we

summarize existing methodologies for interval-censored data, and detail will be given in Chapter 2.

To fit the PH model with interval-censored data, Pan (2000) proposes an imputation method. The imputation approach generates the failure time T_i from a conditional distribution function given T_i is left-censored or finite interval-censored, while right-censored observations are kept unchanged. Then, existing inference procedures for right-censored data can be applied for estimation. Satten (1996), Goggins et al. (1998) and Satten et al. (1998) develop a rank-based likelihood method, where estimations of the baseline hazard are not involved. The method first generates a ranking of the failure times from a ranking probability function. Based on the generated ranking, a score function for $\boldsymbol{\beta}$ is constructed, then $\boldsymbol{\beta}$ is computed by solving roots of the score function. Finkelstein (1986), Pan (1999), and Goetghebeur and Ryan (2000) use a maximum likelihood (ML) method to estimate $\boldsymbol{\beta}$. Finkelstein (1986) considers estimations of the baseline survival $S_0(\cdot)$ and imposes the constraint $0 \leq S_0(\cdot) \leq 1$ by parametrization. Pan (1999) considers estimations of the baseline distribution function $F_0(\cdot)$ with the constraint $0 \leq F_0(\cdot) \leq 1$ by using an iterative convex minorant (ICM) algorithm (Groeneboom and Wellner, 1992).Geotghebeur and Ryan (2000) produce a non-negative baseline hazard estimate using the EM algorithm, but the baseline estimate is not smooth, and hence the trend of changing the baseline hazard over time may not be clear. Betensky et al. (2002) obtain a smooth estimate of the baseline hazard by a local likelihood method. The method starts with approximating $\log h_0(\cdot)$ at local time points near some estimation times by a polynomial function. Then a local likelihood function is constructed, where kernel functions are included to ensure smoothness of the baseline hazard estimate. Estimates of $\boldsymbol{\beta}$ and $h_0(\cdot)$ are computed by maximizing the local likelihood using the EM algorithm. Cai and Betensky (2003), and Joly et al. (1998) also obtain the smooth estimate of baseline hazard by

using a maximum penalized likelihood (MPL) method, where both methods model $h_0(\cdot)$ by splines, and impose the non-negativity constraint on $h_0(\cdot)$ in indirect ways. Joly et al. (1998) use squared coefficients for spline, which may cause convergence issues when some coefficients are zero. Cai and Betensky (2003) impose the constraint on $h_0(\cdot)$ by approximating the log-baseline hazard using a linear spline, using the log function may lead to unstable estimation procedures when the baseline hazard estimate approaches zero, and give rise to difficulties in obtaining the cumulative baseline hazard in closed form for spline orders higher than linear. In the MPL method, estimates are computed by maximizing a penalized log-likelihood function, where a roughness penalty function is included for smoothness of the estimate of $h_0(\cdot)$.

For the AH model, Lin et al. (1998) and Wang et al. (2010) propose a counting process method, where a partial likelihood type score function for $\boldsymbol{\beta}$ is constructed, and $\boldsymbol{\beta}$ is estimated by solving the root of the score function. In this method, the baseline hazard is regarded as a nuisance parameter and not estimated. Ghosh (2001) and Zeng et al. (2006) develop a ML method. Ghosh (2001) estimates $\boldsymbol{\beta}$ and the baseline cumulative hazard $H_0(\cdot)$ using a primal-dual interior point algorithm (Wright, 1997), with constraints of non-negativity and monotonic increasing imposed on $H_0(\cdot)$ directly. Zeng et al. (2006) fit the AH model by estimating $\boldsymbol{\beta}$ and $S_0(\cdot)$ using the Newton algorithm, and constraints on $S_0(\cdot)$ are imposed indirectly by using a log function. Farrington (1996) applies a generalized linear model (GLM) approach for estimations. The GLM approach regards the occurrences of left, right and finite interval-censored observations as independent Bernoulli trials, and relate the occurrence probability to a linear predictor through a negative log function. $\boldsymbol{\beta}$ and the baseline hazard are considered as covariate coefficients in this linear predictor, and estimated by fitting the GLM using the statistical package GLIM. However, the GLM method does not guarantee the smoothness and non-negativity

constraint for the baseline hazard estimate.

For the AFT model, Rabinowitz et al. (1995) and Betensky et al. (2001) propose using linear rank statistics as estimating functions for $\boldsymbol{\beta}$, and estimate $\boldsymbol{\beta}$ by solving roots of the estimating functions. However, as the dimension of $\boldsymbol{\beta}$ is high, implementation of the method becomes numerically difficult. The method involves estimations of the distribution function for the error variable ϵ, and the distribution function is estimated by a ML method. A rank statistic derived by Li and Pu (2003) can only be applied to estimate $\boldsymbol{\beta}$ in one dimension. It is not straightforward to generalize the statistic to high-dimensional covariates. Komárek et al. (2005) develop a MPL method for estimations of $\boldsymbol{\beta}$ and the density function of standardized ϵ. Denote $\tilde{\epsilon}$ to be the standardized ϵ and let $f_{\tilde{\epsilon}}(t)$ be the density function of $\tilde{\epsilon}$. The MPL method starts with approximating $f_{\tilde{\epsilon}}(t)$ by a linear combination of Gaussian density functions, and imposes the constraints, $f_{\tilde{\epsilon}}(t) > 0$ and $\int f_{\tilde{\epsilon}}(t)dt = 1$, indirectly by transforming $f_{\tilde{\epsilon}}(t)$. MPL estimates of $\boldsymbol{\beta}$ and $f_{\tilde{\epsilon}}(t)$ are obtained by maximizing a penalized log-likelihood function, where a roughness penalty function is used for smoothness of the estimate of $f_{\tilde{\epsilon}}(t)$.

1.6 Goal

In this dissertation, we assume that (i) the observations from different subjects are independent, (ii) covariates are time-independent and the distribution of covariates do not involve regression coefficients, and (iii) censoring time is independent of failure time. We propose novel methods to fit the PH model, AH model and AFT model for partly interval-censored data, which contains fully observed, left-censored, finite interval-censored and right-censored data.

We fit the PH model by estimating the regression coefficient $\boldsymbol{\beta}$ and the baseline hazard $h_0(\cdot)$. We compute the estimates of $\boldsymbol{\beta}$ and $h_0(\cdot)$ simultaneously by maximizing a penalized

log-likelihood function, where the non-negative constraint on $h_0(\cdot)$ is imposed in a direct way. We solve this constrained optimization problem by using the Newton-MI algorithm (Ma et al., 2014). This algorithm ensures the non-negativity for the baseline hazard, and also avoids using the second derivative of the penalized log-likelihood when estimating the baseline hazard. To produce a smoothed estimate of the baseline hazard so that its trend of changing over time can be clearly observed, we use a penalty function. We derive the asymptotic properties of the MPL estimates, and the asymptotic properties can be used to perform hypothesis testing and to obtain variance estimates for the MPL estimates.

Under the AH model, since effects of the covariates are additive on the hazard function, $h(\cdot|\boldsymbol{X}_i)$, we have to constrain both $h(\cdot|\boldsymbol{X}_i)$ and the baseline hazard $h_0(\cdot)$ to be non-negative in the estimation procedure. For this purpose, we use a primal-dual interior point algorithm (Wright, 1997). These two constraints can be imposed simultaneously and directly in this algorithm. This algorithm has also been applied by Ghosh (2001) in studying current status data. We obtain the MPL estimates of $\boldsymbol{\beta}$ and $h_0(\cdot)$ simultaneously by maximizing a penalized log-likelihood function with a penalty function included for smoothness of the baseline hazard estimate.

To fit the AFT model, we also develop a MPL method for estimation. We estimate $\boldsymbol{\beta}$ and the hazard function of ϵ^*, where $\epsilon^* = e^{\epsilon}$ and ϵ is the error variable in the AFT model, which has been defined in Section 1.4.3. Again we obtain estimates of $\boldsymbol{\beta}$ and the hazard of ϵ^* simultaneously by maximizing a penalized log-likelihood function using the Newton-MI algorithm, where the non-negative constraint on the hazard is imposed in a direct way. For smoothness of the hazard estimate, we use a roughness penalty function, which relates to the value of its second derivative. The asymptotic properties of the MPL estimates are derived.

In summary, the goal of this dissertation is to develop MPL methods to estimate

the PH model, AH model and AFT model with partly interval-censored data. The performance of the proposed methods is investigated through simulation studies, and the proposed methods are applied in analyzing a real data set. The statistical package R is used for both the simulation and application studies.

1.7 Outline

Chapter 2 reviews some existing methods to fit the PH, AH and AFT models.

Chapter 3 proposes an estimation method for the PH model with partly interval-censored data using MPL and Newton-MI algorithm. The asymptotic properties of the MPL estimators are derived and a simulation study is conducted. The MPL method is also applied to analyze a set of real data.

Chapter 4 proposes an estimation method for the AH model in the case of partly interval-censored data, using MPL and a primal-dual interior point algorithm. The asymptotic properties of the MPL estimates are derived. A simulation study is conducted to evaluate the proposed method, and results from real data analysis are reported.

Chapter 5 fits the AFT model with partly interval-censored data by MPL and Newton-MI algorithm. The asymptotic properties of the MPL estimates are presented. A simulation study is conducted to investigate the performance of the proposed method, and real data is analyzed.

Conclusions are drawn in Chapter 6.

Chapter 2

Literature Review

In this chapter, we will review some existing methodologies for semi-parametric regression analysis of interval-censored failure time data under the proportional hazard (PH), additive hazard (AH) and accelerated failure time (AFT) models. Semi-parametric regression models assume some relationships between covariate effects and the hazard function or failure time, but the underlying distribution functions for failure time are unspecified and need to be estimated. Hence the semi-parametric regression models have both parametric and non-parametric parts. In this dissertation, we concentrate ourselves on estimations for the semi-parametric regression models. Before reviewing methods for the semi-parametric regression models, we briefly summarize some approaches for parametric and non-parametric estimations.

Parametric inference methods assume parametric models for the failure time T, and there are three models commonly used, which are the Weibull, log-normal and log-logistic models. One can refer to Kalbfleisch and Prentice (2002), and Lawless (2011) for more parametric models. The advantage of parametric inference approaches is that their implementation is straightforward in principle and standard maximum likelihood theory can be easily applied. The disadvantage is that there often does not exist enough prior

information or data to suggest or verify a parametric distribution for the failure time T.

Non-parametric regression models cannot specify the underlying failure time distributions, and the underlying distribution functions are estimated without considering covariate effects. With interval-censored data, non-parametric estimations of survival functions for T have been analyzed by Peto (1973), Turnbull (1976), Gentleman and Geyer (1994), Li et al. (1997), Yu et al. (2000), Groeneboom and Wellner (1992), and Wellner and Zhan (1997). For non-parametric estimations of density functions for T, methods have been developed by Keiding (1991), and Kooperberg and Stone (1992).

The chapter is organized as follows. In Section 2.1, we review some existing methods for the proportional hazard (PH) model. In Section 2.2, we review methods for the additive hazard (AH) model. In Section 2.3, we review methods for the accelerated failure time (AFT) model. In Section 2.4, we give a brief summary of the methods that will be developed in this dissertation for the PH, AH and AFT models.

2.1 Proportional Hazard (PH) Model

In this section, we review existing approaches to fit the proportional hazard (PH) model with interval-censored data. Recall the definition of interval-censored data given in Chapter 1, which contains left-censored data, finite interval-censored data and right-censored data. Suppose there are n subjects in a survival study, and for each subject $i = 1, \cdots, n$, the PH model specifies the hazard function $h(t|\boldsymbol{X}_i)$ according to $h(t; \boldsymbol{X}_i) = h_0(t) \exp\{\boldsymbol{X}_i^T \boldsymbol{\beta}\}$, where $h_0(\cdot)$ is an unspecified baseline hazard function, $\boldsymbol{X}_i$ is a $p \times 1$ covariate vector and assumed to be time-independent, and $\boldsymbol{\beta}$ is a $p \times 1$ regression coefficient vector.

2.1.1 Imputation Approaches

Imputation approaches are somtimes used to transform the problem of analyzing interval-censored data to that of analyzing right-censored data, so that one can apply existing inference procedures and statistical software developed for right-censored data. The simplest imputation method modifies the left-censored and finite interval-censored data. For instance, it is assumed that the corresponding unobserved failure time T lies at the middle-point of the observed interval $(L, R]$, i.e., $T = (L + R)/2$, while the right censored data is kept unchanged. Therefore, the data set now consists of imputed failure time data $(L + R)/2$ and the right censored data, and the data set is known as imputed right-censored data (Pan, 2000). This method is called mid-point imputation. Alternatively, it can also be assumed that the corresponding unobserved T is uniformly distributed over $(L, R]$ for the left-censored and finite interval-censored data.

With the imputed right-censored data, one can apply the partial likelihood method. Assuming there are no tied imputed failure times, the regression coefficients $\boldsymbol{\beta}$ are estimated by maximizing the partial likelihood given by

$$L_p(\boldsymbol{\beta}) = \prod_{i=1}^{K} \frac{\exp(\boldsymbol{X}_{(i)}^T \boldsymbol{\beta})}{\sum_{j \in R(t_{(i)})} \exp(\boldsymbol{X}_j^T \boldsymbol{\beta})}, \tag{2.1}$$

where $t_{(i)}$ is the ith ordered statistic of the imputed failure times, K is the number of the imputed failure times, $\boldsymbol{X}_{(i)}$ is $\boldsymbol{X}_i$ corresponding to $t_{(i)}$ and $R(t_{(i)})$ is the risk set of individuals at $t_{(i)}^-$. Law and Brookmeyer (1992) assess the performance of the mid-point imputation method and conclude that wide intervals lead to biased estimates and underestimated standard errors. In addition, they show that the statistical properties of the $\boldsymbol{\beta}$ estimate depend strongly on the width of the interval $(L, R]$ and on the distribution of the failure time T. Therefore, it is not reliable to apply the mid-point imputation method when the observed intervals are wide.

To overcome this shortcoming, Pan (2000) applies a multiple imputation approach,

which is originally proposed by Tanner and Wong (1987) for grouped right censored data. The multiple imputation method improves the accuracy by generating multiple sets of imputed right-censored data, each of which is analyzed separately as in the mid-point imputation approach, and then the results are combined. Specifically, the multiple imputation method iterates between an imputation step and an estimation step. In the imputation step, it is supposed to generate M sets of imputed right-censored data from the observations $\{L_i, R_i, \boldsymbol{X}_i, i = 1, \cdots, n\}$ using the poor man's data augmentation (PMDA) (Wei and Tanner, 1991) or asymptotic normal data augmentation (ANDA) (Wei and Tanner, 1991). Here we describe the kth iteration of the PMDA algorithm for demonstration. For each subject i and each imputed data set j, $j = 1, \cdots, M$, let $\delta^{(k)}_{(j),i}$ be an indicator, where $\delta^{(k)}_{(j),i} = 0$ if the failure time of subject i is right censored, and $\delta^{(k)}_{(j),i} = 1$ if left censored or finite interval censored. For $\delta^{(k)}_{(j),i} = 0$, set $R_i = \infty$ and define $T^{(k)}_{(j),i}$ to be the corresponding censored time, i.e., $T^{(k)}_{(j),i} = L_i$. While for $\delta^{(k)}_{(j),i} = 1$, $T^{(k)}_{(j),i}$ is defined to be the corresponding imputed failure time and is generated from the distribution conditional on $T^{(k)}_{(j),i} \in (L_i, R_i]$ using the current estimates of $\boldsymbol{\beta}^{(k)}$ and $S^{(k)}_0(\cdot)$, that is to solve

$$\Pr(T^{(k)}_{(j),i} \geq t | T^{(k)}_{(j),i} \in (L_i, R_i]) = \frac{[S^{(k)}_0(L_i)]^{\exp(\boldsymbol{X}_i^T \boldsymbol{\beta}^{(k)})} - [S^{(k)}_0(t)]^{\exp(\boldsymbol{X}_i^T \boldsymbol{\beta}^{(k)})}}{[S^{(k)}_0(L_i)]^{\exp(\boldsymbol{X}_i^T \boldsymbol{\beta}^{(k)})} - [S^{(k)}_0(R_i)]^{\exp(\boldsymbol{X}_i^T \boldsymbol{\beta}^{(k)})}}$$

by an inversion method. Repeating this step M times gives rise to M sets of imputed right-censored data denoted by

$$\left\{ T^{(k)}_{(j),i}, \delta^{(k)}_{(j),i}, \boldsymbol{X}_i, i = 1, \cdots, n, j = 1, \cdots, M \right\}.$$

In the estimation step, for each of the M data sets, an estimate of $\boldsymbol{\beta}^{(k)}_{(j)}$ is obtained by applying the partial likelihood approach, and its corresponding covariance estimate, $\Sigma^{(k)}_{(j)}$, is computed by the observed Fisher information matrix that can be derived from the partial likelihood (2.1). Finally, the Breslow estimate of $S^{(k)}_{0,(j)}(\cdot)$ is calculated. Then the estimates are updated by $\boldsymbol{\beta}^{(k+1)} = \sum_{j=1}^{M} \boldsymbol{\beta}^{(k)}_{(j)}/\mathrm{M}$, $S^{(k+1)}_0 = \sum_{j=1}^{M} S^{(k)}_{0,(j)}/M$ and $\Sigma^{(k+1)} =$

$\sum_{j=1}^{M} \Sigma_{(j)}^{(k)}/M + (1+\frac{1}{M})\sum_{j=1}^{M}[\boldsymbol{\beta}_{(j)}^{(k)} - \boldsymbol{\beta}^{(k+1)}][\boldsymbol{\beta}_{(j)}^{(k)} - \boldsymbol{\beta}^{(k+1)}]^T/(M-1)$. Note the updated covariance estimate is the sum of the within-imputation and between-imputation variance estimates. The initial values of the iteration procedure can be computed via the mid-point imputation approach.

2.1.2 Rank-based approach

Satten (1996) proposes a rank-based marginal likelihood approach, in which data comprises a set of all possible rankings of the failure times, postulated from their corresponding censored intervals $\{(L_i, R_i], i = 1, \cdots, n\}$. Since the failure times are unobserved, there may be different failure time ranking results when some censored intervals overlap. Note that when none of them overlap, there will be only one ranking result. Denote the set of all possible rankings by $\mathcal{R}$, and each element of this set by $\boldsymbol{r} = [r_1, \cdots, r_n]^T$. That is, every element is a permutation of the integers from 1 to n. For example, if we take $n = 3$, then $\boldsymbol{r} = [1,2,3]^T$, $[1,3,2]^T$, $[2,1,3]^T$, $[2,3,1]^T$, $[3,1,2]^T$ or $[3,2,1]^T$. Now define $\Pr(\boldsymbol{r}|\boldsymbol{\beta}, \boldsymbol{X})$ to be the probability of ranking $\boldsymbol{r}$ under the PH model given $\boldsymbol{\beta}$ and $\boldsymbol{X}_{n\times p}$. Based on the ranking set $\mathcal{R}$, one can derive the following marginal log-likelihood

$$\ell(\boldsymbol{\beta}) = \ln \sum_{\boldsymbol{r}\in\mathcal{R}} \Pr(\boldsymbol{r}|\boldsymbol{\beta}, \boldsymbol{X}). \tag{2.2}$$

Assuming there are no tied failure times, the ranking probability $\Pr(\boldsymbol{r}|\boldsymbol{\beta}, \boldsymbol{X})$ is

$$\Pr(\boldsymbol{r}|\beta, \boldsymbol{X}) = \prod_{i=1}^{n} \frac{e^{\boldsymbol{X}_i^T\boldsymbol{\beta}}}{\sum_{j\in R(i)} e^{\boldsymbol{X}_j^T\boldsymbol{\beta}}},$$

where $R(i)$ is the risk set of subjects at the ith ranked failure time under $\boldsymbol{r}$. The corresponding score function is obtained by taking the first derivative of $\ell(\boldsymbol{\beta})$ with respective to $\boldsymbol{\beta}$, that is

$$S(\boldsymbol{\beta}) = \sum_{\boldsymbol{r}\in\mathcal{R}} w(\boldsymbol{r}|\boldsymbol{\beta}, \boldsymbol{X}) S_{\boldsymbol{\beta}}(\boldsymbol{r}|\boldsymbol{X}), \tag{2.3}$$

where $w(\boldsymbol{r}|\boldsymbol{\beta}, X)$ is the weight given by

$$w(\boldsymbol{r}|\boldsymbol{\beta}, \boldsymbol{X}) = \frac{\Pr(\boldsymbol{r}|\boldsymbol{\beta}, \boldsymbol{X})}{\sum_{\boldsymbol{r}\in\mathcal{R}} \Pr(\boldsymbol{r}|\boldsymbol{\beta}, \boldsymbol{X})}, \tag{2.4}$$

and $S_{\boldsymbol{\beta}}(\boldsymbol{r}|\boldsymbol{X})$ is the score function based on the rank $\boldsymbol{r}$, which can be calculated by $S_{\boldsymbol{\beta}}(\boldsymbol{r}|\boldsymbol{X}) = \partial \log \Pr(\boldsymbol{r}|\boldsymbol{\beta}, \boldsymbol{X})/\partial\boldsymbol{\beta}$. Note that $w(\boldsymbol{r}|\boldsymbol{\beta}, \boldsymbol{X})$ can also be understood as a probability distribution on the set $\mathcal{R}$, so that $S(\boldsymbol{\beta})$ can be considered to be the expected value of the score $S_{\boldsymbol{\beta}}(\boldsymbol{r}|\boldsymbol{X})$ with respect to the distribution $w(\boldsymbol{r}|\boldsymbol{\beta}, \boldsymbol{X})$. Therefore, equation (2.3) can be rewritten as

$$S(\boldsymbol{\beta}) = E[S_{\boldsymbol{\beta}}(\boldsymbol{r}|\boldsymbol{X})]. \tag{2.5}$$

Let $\hat{\boldsymbol{\beta}}$ be the maximum marginal likelihood estimates of $\boldsymbol{\beta}$, obtained by solving $S(\boldsymbol{\beta}) = 0$. Satten (1996) calculates $\hat{\boldsymbol{\beta}}$ by implementing a stochastic approximation method, which can be considered as a variant of the Newton method. As a consequence of (2.5), $S_{\boldsymbol{\beta}}(\boldsymbol{r}|\boldsymbol{X})$ is an unbiased estimator of $S(\boldsymbol{\beta})$ if a random element $\boldsymbol{r}$ of $\mathcal{R}$ is selected with the distribution $w(\boldsymbol{r}|\boldsymbol{\beta}, \boldsymbol{X})$. Therefore, the kth iteration of the stochastic approximation involves two steps: (i) generating rankings $\boldsymbol{r}$ using the probability $w(\boldsymbol{r}|\boldsymbol{\beta}^{(k)}, \boldsymbol{X})$ which can be achieved by a Gibbs sampling scheme, and (ii) computing $\boldsymbol{\beta}^{(k+1)}$ by estimating the root of $S(\boldsymbol{\beta})$ using a Robbins-Munro process (Ruppert et al., 1984). The stochastic approximation scheme is run until $k = k^*$, where k^* is determined by a termination criterion.

One drawback of this method is that maximization of the log-likelihood (2.2) can be difficult when there are many overlapping censoring intervals, as the ranking set $\mathcal{R}$ will be large in this case. To overcome this difficulty, Goggins et al. (1998) maximize (2.2) using an EM algorithm, where the complete data is the ranking of the unobserved failure times of all subjects. Let $\boldsymbol{r}$ be the true ranking and $\boldsymbol{\beta}^{(k)}$ denote the estimate of $\boldsymbol{\beta}$ at the kth iteration of the EM algorithm. The E step is to find the conditional expectation of

the complete data log-likelihood, given by

$$\sum_{\boldsymbol{r}\in\mathcal{R}} \log[L(\boldsymbol{\beta}|\boldsymbol{r})] w(\boldsymbol{r}|\boldsymbol{\beta}^{(k)}, \boldsymbol{X}), \tag{2.6}$$

where $L(\boldsymbol{\beta}|\boldsymbol{r}) \propto \Pr(\boldsymbol{r}|\boldsymbol{\beta}, \boldsymbol{X})$, and $w(\boldsymbol{r}|\boldsymbol{\beta}^{(k)}, \boldsymbol{X})$ is the conditional distribution of $\boldsymbol{r}$ that is given in (2.4). Then the M step maximizes (2.6) to update $\boldsymbol{\beta}$. However, it is difficult to maximize (2.6) if the ranking set $\mathcal{R}$ is large. Goggins et al. (1998) propose using Monte Carlo in the E step. Specifically, a sample of N rankings, $\{\boldsymbol{r}_j, \boldsymbol{r}_j \in \mathcal{R}, j = 1, \cdots, N\}$, are drawn from $w(\boldsymbol{r}|\boldsymbol{\beta}^{(k)}, \boldsymbol{X})$ given the current estimate $\boldsymbol{\beta}^{(k)}$, where an MCMC algorithm (Hastings, 1970) is applied for the drawing. In the M-step, the average of the individual log-likelihood, given by

$$\frac{1}{N}\sum_{j=1}^{N} \log[L(\boldsymbol{\beta}|\boldsymbol{r}_j)],$$

is maximized with respective to $\boldsymbol{\beta}$ to give $\boldsymbol{\beta}^{(k+1)}$. These two steps are repeated until the $\boldsymbol{\beta}$ estimates converge.

Alternative to the approach by Satten (1996), Satten et al. (1998) consider a different estimating equation that is also an expectation of the partial likelihood score function, but now with respect to the distribution of the imputed right-censored data. The definition of imputed right-censored has been given in Section 2.1.1. For interval-censored data $\{(L_i, R_i], i = 1, \cdots, n\}$, let $\boldsymbol{\delta}$ be an indicator vector with components $\delta_i = 0$ if failure time T_i is right-censored and $\delta_i = 1$ otherwise. Define $\boldsymbol{t}^* = [t_1^*, \cdots, t_n^*]^T$ to be the vector of imputed right-censored data. The distribution function of $\boldsymbol{t}^*$ is

$$F(\boldsymbol{t}^*|\boldsymbol{\delta}; \boldsymbol{\beta}, \boldsymbol{\theta}) = \prod_{i=1}^{n} F(t_i^*|L_i, R_i; \boldsymbol{\beta}, \boldsymbol{\theta})^{\delta_i} I(L_i \le t_i^*)^{1-\delta_i},$$

where

$$F(t_i^*|L_i, R_i; \boldsymbol{\beta}, \boldsymbol{\theta}) = \frac{F(t_i^*|\boldsymbol{\beta}, \boldsymbol{\theta}) - F(L_i|\boldsymbol{\beta}, \boldsymbol{\theta})}{F(R_i|\boldsymbol{\beta}, \boldsymbol{\theta}) - F(L_i|\boldsymbol{\beta}, \boldsymbol{\theta})} I(t_i^* \in (L_i, R_i])$$

and $I(\cdot)$ is an indicator function. Now let $S_{\boldsymbol{\beta}}(\boldsymbol{t}^*|\boldsymbol{X})$ denote the partial likelihood score

function, then the score function for $\boldsymbol{\beta}$ is

$$S(\boldsymbol{\beta}, \boldsymbol{\theta}) = \int S_{\boldsymbol{\beta}}(\boldsymbol{t}^*|\boldsymbol{X}) dF(\boldsymbol{t}^*|\boldsymbol{\delta}; \boldsymbol{\beta}, \boldsymbol{\theta}). \tag{2.7}$$

Estimates of $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ are computed alternatively. Specifically, with a fixed value of $\boldsymbol{\beta}$, the estimate of $\boldsymbol{\theta}$ is obtained by maximizing the observed log-likelihood function with respect to $\boldsymbol{\theta}$, where the observed log-likelihood is

$$\ell(\boldsymbol{\beta}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \{\delta_i \ln[F(R_i|\boldsymbol{\beta}, \boldsymbol{\theta}) - F(L_i|\boldsymbol{\beta}, \boldsymbol{\theta})] + (1 - \delta_i) \ln[1 - F(L_i|\boldsymbol{\beta}, \boldsymbol{\theta})]\}.$$

Denote the resulting estimator by $\hat{\boldsymbol{\theta}}$. With $\hat{\boldsymbol{\theta}}$, $\boldsymbol{\beta}$ is then estimated by solving $S(\boldsymbol{\beta}, \hat{\boldsymbol{\theta}}) = 0$. To estimate the root of $S(\boldsymbol{\beta}, \hat{\boldsymbol{\theta}})$, the imputed right-censored data $\boldsymbol{t}^*$ is firstly generated from $F(\boldsymbol{t}^*|\boldsymbol{\delta}; \boldsymbol{\beta}, \hat{\boldsymbol{\theta}})$, and then a recursive stochastic approximation method is applied, which is similar to that in Satten (1996). Although the generation of $\boldsymbol{t}^*$ depends on the underlying distribution, simulation studies conducted by Satten et al. (1998) show that the estimation method performs well even when the distribution is misspecified. This can be explained by the fact that the partial likelihood score function $S_{\boldsymbol{\beta}}(\boldsymbol{t}^*|\boldsymbol{X})$ is independent of the distribution and depends only on the ranks of the imputed times.

2.1.3 Maximum likelihood (ML) approach

Finkelstein (1986) proposes a maximum likelihood (ML) approach to fit the PH model with interval-censored data. In this method, the log-likelihood function is constructed in terms of the regression coefficient $\boldsymbol{\beta}$ and the baseline survival $S_0(\cdot)$ by

$$\ell(\boldsymbol{\beta}, S_0) = \sum_{i=1}^{n} \log \left\{ S_0(L_i)^{\exp(\boldsymbol{X}_i^T \boldsymbol{\beta})} - S_0(R_i)^{\exp(\boldsymbol{X}_i^T \boldsymbol{\beta})} \right\}. \tag{2.8}$$

Let $0 = u_0 < u_1 < u_2 < \cdots < u_m < u_{m+1} = \infty$ be the ordered distinct time points of all observed interval end points $\{L_i, R_i; i = 1, \cdots, n\}$. Here $S_0(\cdot)$ is only estimated at these ordered distinct time points. In order to impose the constraints $0 \leq S_0(\cdot) \leq 1$,

$S_0(\cdot)$ is transformed by $\alpha_j = \log[-\log S_0(u_j)]$, $j = 0, \cdots, m+1$, where clearly $\alpha_0 = -\infty$ and $\alpha_{m+1} = \infty$. Let $\boldsymbol{\alpha} = [\alpha_0, \cdots, \alpha_{m+1}]^T$. Now define indicator variables $\zeta_{ij} = I(u_j \in (L_i, R_i])$. Then (2.8) can be re-expressed in terms of $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ by

$$\ell(\boldsymbol{\beta}, \boldsymbol{\alpha}) = \sum_{i=1}^{n} \log \Big\{ \sum_{j=1}^{m+1} \zeta_{ij} \big[e^{-\exp(\alpha_{j-1} + \boldsymbol{X}_i^T \boldsymbol{\beta})} - e^{-\exp(\alpha_j + \boldsymbol{X}_i^T \boldsymbol{\beta})} \big] \Big\}. \tag{2.9}$$

Finally, estimates of $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ are computed by maximizing (2.9) using the Newton algorithm.

Pan (1999) reformulates an iterative convex minorant (ICM) algorithm (Groeneboom and Wellner, 1992) as a special case of the generalized gradient projection (GGP) method (Bertsekas, 1982) to maximize (2.9), where (2.9) is rewritten using the baseline distribution vector $\boldsymbol{F}_0 = [F_0(u_1), \cdots, F_0(u_m)]^T$,

$$\ell(\boldsymbol{\beta}, \boldsymbol{F}_0) = \sum_{i=1}^{n} \log \left\{ \sum_{j=1}^{m+1} \zeta_{ij} \big[(1 - F_0(u_{j-1}))^{\exp(\boldsymbol{X}_i^T \boldsymbol{\beta})} - (1 - F_0(u_j))^{\exp(\boldsymbol{X}_i^T \boldsymbol{\beta})} \big] \right\}. \tag{2.10}$$

In contrast to the Newton algorithm applied by Finkelstein (1986), which uses the inverse Hessian matrix at each iteration, the ICM algorithm only uses the diagonal elements of the Hessian matrix. Let $\nabla_1(\boldsymbol{F}_0, \boldsymbol{\beta}) = \partial \ell(\boldsymbol{\beta}, \boldsymbol{F}_0)/\partial \boldsymbol{\beta}$ and $\nabla_2(\boldsymbol{F}_0, \boldsymbol{\beta}) = \partial \ell(\boldsymbol{\beta}, \boldsymbol{F}_0)/\partial \boldsymbol{F}_0$. Let $D_1(\boldsymbol{F}_0, \boldsymbol{\beta})$ and $D_2(\boldsymbol{F}_0, \boldsymbol{\beta})$ be diagonal matrices with diagonal elements of $-\partial \ell^2(\boldsymbol{\beta}, \boldsymbol{F}_0)/\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T$ and $-\partial \ell^2(\boldsymbol{\beta}, \boldsymbol{F}_0)/\partial \boldsymbol{F}_0 \partial \boldsymbol{F}_0^T$, respectively. To have a proper baseline distribution function, the constraints $\boldsymbol{F}_0 \in \mathcal{C} = \{\boldsymbol{F}_0 : 0 \le F_0(u_1) \le \cdots \le F_0(u_m) \le 1\}$ must be imposed in each iteration of the ICM algorithm. Then $\boldsymbol{F}_0$ and $\boldsymbol{\beta}$ are updated simultaneously by

$$\boldsymbol{F}_0^{(k+1)} = \text{Proj}[\boldsymbol{F}_0^{(k)} + \omega_{1j} D_2(\boldsymbol{F}_0^{(k)}, \boldsymbol{\beta}^{(k)})^{-1} \nabla_2(\boldsymbol{F}_0^{(k)}, \boldsymbol{\beta}^{(k)}), D_2(\boldsymbol{F}_0^{(k)}, \boldsymbol{\beta}^{(k)}), \mathcal{C}] \tag{2.11}$$

and

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \omega_{2j} D_1(\boldsymbol{F}_0^{(k)}, \boldsymbol{\beta}^{(k)})^{-1} \nabla_1(\boldsymbol{F}_0^{(k)}, \boldsymbol{\beta}^{(k)}), \tag{2.12}$$

where Proj is a projection operation defined by

$$\text{Proj}[\boldsymbol{F}_0, D_2(\boldsymbol{F}_0, \boldsymbol{\beta}), \mathcal{C}] = \arg\min_{\boldsymbol{F}} \big\{ (\boldsymbol{F} - \boldsymbol{F}_0)^T D_2(\boldsymbol{F}_0, \boldsymbol{\beta}) (\boldsymbol{F} - \boldsymbol{F}_0) : \boldsymbol{F} \in \mathcal{C} \big\}, \tag{2.13}$$

and ω_{1j} and ω_{2j} are step sizes used to ensure the increasing of the log-likelihood. The projection (2.13) is just an isotonic least squares regression problem and can be solved by the pool-adjacent-violators algorithm (Robertson et al., 1988) , for example.

Huang and Wellner (1995,1997) also adopt the ICM algorithm. But they estimate $\boldsymbol{\beta}$ and $\boldsymbol{F}_0$ alternatively, where $\ell(\boldsymbol{\beta}^{(k)}, \boldsymbol{F}_0)$ is firstly maximized with respect to $\boldsymbol{F}_0$ by the ICM algorithm to get $\boldsymbol{F}_0^{(k+1)}$, and then $\ell(\boldsymbol{\beta}, \boldsymbol{F}_0^{(k+1)})$ is maximized with respect to $\boldsymbol{\beta}$ by the Newton algorithm to give $\boldsymbol{\beta}^{(k+1)}$.

Lindsey and Ryan (1998), and Goetghebeur and Ryan (2000) consider fitting the PH model for partly interval-censored data by estimating the baseline hazard $h_0(\cdot)$ and $\boldsymbol{\beta}$. Recall the definition of partly interval-censored data given in Chapter 1, which contains exactly observed failure time data and interval-censored data. In their method, the baseline hazard is assumed to be piecewise constant. Suppose there are m intervals, $\{(\tau_{j-1}, \tau_j], j = 1, \cdots, m\}$, which divide up the time scale, then $h_0(t) = \theta_j$ for $t \in (\tau_{j-1}, \tau_j]$. Let $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_m]^T$. For simplicity, assume that each of the m intervals has unit width, and each observed failure time or right censoring time is rounded up to the endpoint of the interval within which it occurs. Let D_{ij} indicate whether the failure occurs in the jth interval for subject i, and Y_{ij} indicate whether subject i is at risk in that interval. Estimates of $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ are obtained by using an EM algorithm, where the exactly observed failure time data, and the unobserved failure time that is left-, right- and finite interval-censored are considered to be the complete data. The complete-data log-likelihood is then given by

$$\ell(\boldsymbol{\beta}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{m} \left\{ \log(\theta_j)\ D_{ij} + \boldsymbol{X}_i^T \boldsymbol{\beta} D_{ij} - \theta_j \exp(\boldsymbol{X}_i^T \boldsymbol{\beta}) Y_{ij} \right\}. \tag{2.14}$$

In the E-step, the conditional expectation of (2.14) is derived, which simply involves calculating expectations of D_{ij} and Y_{ij} at τ_j. The expectations of D_{ij} and Y_{ij} are denoted by ρ_{ij} and ν_{ij} respectively. For left- and finite interval-censored data, with current estimates

$\boldsymbol{\beta}^{(k)}$ and $\boldsymbol{\theta}^{(k)}$,

$$\rho_{ij}^{(k)} = \frac{p_{ij}^{(k)}}{\sum_{\tau_u \in (L_i, R_i]} p_{iu}^{(k)}}, \tag{2.15}$$

where $p_{ij}^{(k)} = \theta_j^{(k)} \exp(\boldsymbol{X}_i^T \boldsymbol{\beta}^{(k)}) \exp\left\{-\sum_{u=1}^{j} \theta_u^{(k)} \exp(\boldsymbol{X}_i^T \boldsymbol{\beta}^{(k)})\right\}$. For exactly observed failure time data, $\rho_{ij}^{(k)} = 1$, and for right-censored data, $\rho_{ij}^{(k)} = 0$. Now let d_i be an indicator taking value one if the failure time T_i is left- or finite interval-censored, and zero if T_i is right- censored. Then the conditional expectation of Y_{ij} is given by

$$\nu_{ij}^{(k)} = \sum_{u \geq j} \rho_{iu}^{(k)} d_i + I(\tau_j \leq L_i)(1 - d_i). \tag{2.16}$$

In the M-step, estimates of $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ are updated by maximizing the expectation of (2.14). The solution with fixed $\boldsymbol{\beta}^{(k)}$ takes the form of

$$\theta_j^{(k+1)} = \frac{\sum_{i=1}^{n} \rho_{ij}^{(k)}}{\sum_{i=1}^{n} \exp(\boldsymbol{X}_i^T \boldsymbol{\beta}^{(k)}) \nu_{ij}^{(k)}}. \tag{2.17}$$

Equation (2.17) shows that $\theta_j^{(k+1)}$ is non-negative, and hence the non-negativity of the baseline hazard estimate is automatically guaranteed. With fixed $\theta_j^{(k+1)}$, $\boldsymbol{\beta}^{(k+1)}$ is computed by maximization of the expectation of (2.14). In this method, smoothness of the estimated baseline hazard is not guaranteed.

2.1.4 Local likelihood approach

Betensky et al. (1999) and Betensky et al. (2002) obtain a smooth estimate of the baseline hazard $h_0(\cdot)$ by a local likelihood method with partly interval-censored data. Let $\{t_j; j = 1, \cdots, m\}$ be the time points at which $h_0(\cdot)$ is estimated. The local likelihood estimate of $h_0(\cdot)$ at t_j starts with approximating $\log h_0(\cdot)$ in a smoothing window with bandwidth b at local points near t_j using a polynomial function, that is

$$\log h_0(s) \approx \theta_{0j} + \theta_{1j}(s - t_j) + \cdots + \theta_{qj}(s - t_j)^q = \boldsymbol{\theta}_j^T \boldsymbol{P}_{j,q}(s) \quad \text{for } |s - t_j| \leq b,$$

where $\boldsymbol{\theta}_j = [\theta_{0j}, \theta_{1j}, \cdots, \theta_{qj}]^T$ and $\boldsymbol{P}_{j,q}(s) = [1, (s - t_j), \cdots, (s - t_j)^q]^T$. Note that the log function guarantees the positivity of $h_0(\cdot)$.

In this local likelihood approach, parameters $\boldsymbol{\beta}$ and $\boldsymbol{\theta}_j$ are estimated alternatively. Given the observed data $\{(L_i, R_i], \boldsymbol{X}_i; i = 1, \cdots, n\}$ and the current estimates $\boldsymbol{\beta}^{(k)}$ and $\boldsymbol{\theta}_j^{(k)}$, $\boldsymbol{\theta}_j$ is firstly updated by using an EM algorithm with $\boldsymbol{\beta}$ fixed at $\boldsymbol{\beta}^{(k)}$, where the exactly observed failure time data is viewed as the incomplete data and the failure time that is left, right, or finite interval-censored is considered to be the missing data. Note that $h_0^{(k)}(t_j) = e^{\theta_{0j}^{(k)}}$, and for $|u - t_j| \leq b$, $h_0^{(k)}(u) = \exp\{(\boldsymbol{\theta}_j^{(k)})^T \boldsymbol{P}_{j,q}(u)\}$, $S_0^{(k)}(u) = \exp\{-\int_0^u h_0^{(k)}(v)dv\}$ and $f_0^{(k)}(u) = h_0^{(k)}(u) S_0^{(k)}(u)$. For each subject i, set $S_i^{(k)}(u) = [S_0^{(k)}(u)]^{\exp(\boldsymbol{X}_i^T \boldsymbol{\beta}^{(k)})}$ and $f_i^{(k)}(u) = h_0^{(k)}(u) \exp(\boldsymbol{X}_i^T \boldsymbol{\beta}^{(k)}) S_i^{(k)}(u)$. The E-step computes the conditional expectation of the complete-data local log-likelihood, that is

$$\mathrm{E}\Big[\ell_i(u, t_j|\boldsymbol{\beta}, \boldsymbol{\theta}_j)|T_i \in (L_i, R_i]\Big] = \frac{\int_{L_i}^{R_i} \ell_i(u, t_j|\boldsymbol{\beta}^{(k)}, \boldsymbol{\theta}_j) f_i^{(k)}(u)du}{\int_{L_i}^{R_i} f_i^{(k)}(u)du}, \tag{2.18}$$

where

$$\begin{aligned} \ell_i(u, t_j|\boldsymbol{\beta}, \boldsymbol{\theta}_j) &= \boldsymbol{X}_i^T \boldsymbol{\beta} + K(\frac{u - \tau_j}{b}) \boldsymbol{\theta}_j^T \boldsymbol{P}_{j,q}(u) \\ &\quad - \int_0^u \exp\left\{\boldsymbol{X}_i^T \boldsymbol{\beta} + \boldsymbol{\theta}_j^T \boldsymbol{P}_{j,q}(s)\right\} K(\frac{s - \tau_j}{b}) ds. \end{aligned} \tag{2.19}$$

In (2.19), $K(\cdot)$ is a kernel function used for smoothness of the baseline hazard estimate. In the M step, $\boldsymbol{\theta}_j$ is updated by maximizing (2.18) with respect to $\boldsymbol{\theta}_j$. Let $\boldsymbol{\theta}_j^{(k+1)}$ be the resulting estimate. With fixed $\boldsymbol{\theta}_j^{(k+1)}$, $\boldsymbol{\beta}^{(k+1)}$ is then obtained by maximizing

$$\sum_{i=1}^{n} \left[\ln\{(S_0^{(k+1)}(L_i))^{\exp(\boldsymbol{X}_i^T \boldsymbol{\beta})} - (S_0^{(k+1)}(R_i))^{\exp(\boldsymbol{X}_i^T \boldsymbol{\beta})}\} \right]$$

using, for example, the Newton algorithm.

2.1.5 Maximum penalized likelihood (MPL) approach

The smooth estimate of $h_0(\cdot)$ can also be produced by the maximum penalized likelihood (MPL) approach. In the MPL method, $\boldsymbol{\beta}$ and the smooth estimate of $h_0(\cdot)$ are obtained by maximizing a penalized log-likelihood function,

$$\Phi(\boldsymbol{\beta}, h_0) = \ell(\boldsymbol{\beta}, h_0) - \tilde{\gamma} J(h_0), \tag{2.20}$$

where $\ell(\boldsymbol{\beta}, h_0)$ is the log-likelihood given by

$$\ell(\boldsymbol{\beta}, h_0) = \sum_{i=1}^{n} \log \Big\{ \exp\big[- e^{\boldsymbol{X}_i^T \boldsymbol{\beta}} \int_0^{L_i} h_0(s) ds \big] - \exp\big[- e^{\boldsymbol{X}_i^T \boldsymbol{\beta}} \int_0^{R_i} h_0(s) ds \big] \Big\},$$

$\tilde{\gamma}$ is a non-negative smoothing parameter controlling the trade-off between the goodness of fit to data and smoothness of the estimated baseline hazard, and $J(h_0)$ is a penalty function measuring the roughness of $h_0(\cdot)$. Common choices for the penalty function can be

$$J(h_0) = \int_s [h_0'(s)]^2 ds$$

or

$$J(h_0) = \int_s [h_0''(s)]^2 ds,$$

where $h_0'(s)$ and $h_0''(s)$ are the first and second derivatives of $h_0(s)$ with respect to s. Selecting a suitable smoothing value is crucial for good fitting. The smoothing value can be determined empirically, or automatically from the data using the cross validation method, i.e., see (O'Sullivan, 1988).

The MPL estimation procedure starts with modeling $h_0(\cdot)$. Ma et al. (2014) use a linear combination of general basis functions to approximate $h_0(\cdot)$. Therefore, estimating $h_0(\cdot)$ is equivalent to estimating coefficients of the basis functions. Let $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_m]^T$ be the basis coefficients. The parameters $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ are estimated alternatively by using the Newton and Multiplicative Iterative (Newton-MI) algorithm (Ma et al., 2014). At each iteration of the algorithm, $\boldsymbol{\beta}$ is updated by the Newton algorithm, while $\boldsymbol{\theta}$ is updated by the Multiplicative Iterative (MI) algorithm (Ma, 2010). The MI algorithm ensures $\boldsymbol{\theta} \geq \mathbf{0}_{m \times 1}$ so that the non-negativity of $h_0(\cdot)$ can be constrained directly. We also adopt the Newton-MI algorithm for estimations in Chapters 3 and 5.

Cai and Betensky (2003) use a linear spline to approximate the log baseline hazard,

$$\log h_0(t) = \alpha_0 + \alpha_1 t + \sum_{j=1}^{m} \theta_j (t - \mu_j)^+,$$

where $x^+ = \max(0, x)$ and $\{\mu_1, \cdots, \mu_m\}$ are the knots, see also Kooperberg and Clarkson (1997) for the ML estimation. The disadvantage of the log transformation is that when the estimated baseline hazard approaches zero, the estimation procedure will become unstable. The penalty function in this method is taken in the form of $J(h_0) = \frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\theta}$. In this context, $\boldsymbol{\theta}$ is treated as a random effect,

$$\boldsymbol{\theta} \sim N(0, \sigma^2 \boldsymbol{I}),$$

where $\boldsymbol{I}$ is an identity matrix and variance σ^2, whose reciprocal is acting as $\tilde{\gamma}$ in (2.20), controls the amount of smoothness.

Alternative to estimating $\log h_0(\cdot)$, Joly et al. (1998) directly model $h_0(\cdot)$ by using a linear combination of non-negative q-order M-splines with squared coefficients,

$$h_0(t) = \sum_{j=1}^{m} \theta_j^2 M_j(t|q). \tag{2.21}$$

The M-spline of order q in (2.21) can be calculated recursively by

$$M_j(t|q) = \frac{q[(t-\mu_j)M_j(t|q-1) + (\mu_{j+q} - t)M_{j+1}(t|q-1)]}{(q-1)(\mu_{j+q} - \mu_j)}$$

for $t \in (\mu_j, \mu_{j+q}]$, and

$$M_j(t|1) = \frac{1}{\mu_{j+1} - \mu_j}$$

for $t \in (\mu_j, \mu_{j+1}]$, where $\{\mu_j, ; j = 1, \cdots, m\}$ are the knots. The M-splines are positive, and hence the positivity of $h_0(\cdot)$ is satisfied. However, the way the constraint is imposed on $h_0(\cdot)$ may cause convergence issues, especially when some elements of $\boldsymbol{\theta}$ are zero. In this method, the penalty function takes the form $J(h_0) = \int [\sum_{j=1}^{m} \theta_j M_j''(u|q)]^2 du$. The MPL estimates of $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ are obtained by using a combination of the Newton algorithm and a steepest descent algorithm.

2.2 Additive Hazard (AH) Model

In this section, we review three approaches to fit the additive hazard (AH) model with interval-censored data. Suppose there are n subjects in a survival study, for each subject $i = 1, \cdots, n$, the AH model specifies the hazard function according to

$$h(t_i|\boldsymbol{X}_i) = h_0(t_i) + \boldsymbol{X}_i^T\boldsymbol{\beta}, \tag{2.22}$$

where $h_0(\cdot)$ is an unspecified baseline hazard function, $\boldsymbol{X}_i$ is a $p \times 1$ covariate vector and assumed to be time-independent, and $\boldsymbol{\beta}$ is a $p \times 1$ regression coefficient vector.

2.2.1 Counting process approach

We first review a counting process method given by Lin et al. (1998) for current status data. Later Wang et al. (2010) adopted this counting process method to study general interval-censored data. Recall the definition of current status data given in Chapter 1, where the exact value of the failure time T_i is never known, but only observed below or above the monitoring time variable C_i. Therefore, we can only observe C_i and $\delta_i = I(C_i \leq T_i)$, where $I(\cdot)$ is an indicator function. In this context, we denote the observation by $\{C_i, \delta_i, \boldsymbol{X}_i\}$. Lin et al. (1998) study the AH model under two cases: (i) the monitoring time C_i is independent of $\boldsymbol{X}_i$, and (ii) the monitoring time C_i is dependent on $\boldsymbol{X}_i$. They also conduct analysis to compare estimation results between these two cases. The conclusion reveals that estimates of $\boldsymbol{\beta}$ obtained in case (ii) have smaller standard errors than those in case (i).

Here we demonstrate the counting process method under case (ii). Lin et al. (1998) formulate the dependence of C_i on $\boldsymbol{X}_i$ through the proportional hazard (PH) model, and model the hazard function of C_i at time t by

$$d\bar{H}(t|\boldsymbol{X}_i) = e^{\boldsymbol{X}_i^T\boldsymbol{\alpha}} d\bar{H}_0(t), \tag{2.23}$$

where $\bar{H}_0(\cdot)$ is an unspecified baseline cumulative hazard function, and $\boldsymbol{\alpha}$ is a $p \times 1$ vector of unknown regression parameters representing the effect of $\boldsymbol{X}_i$ on C_i. Note that $\boldsymbol{\alpha} = \mathbf{0}$ means that C_i is independent of $\boldsymbol{X}_i$, which is case (i).

Now define a counting process $N_i(t) = \delta_i I(t \geq C_i)$. The process $N_i(t)$ jumps by one unit at time t if and only if $t = C_i$ and $C_i \leq T_i$. Subject i is at risk for the event at t if $t \leq C_i$. Let $Y_i(t) = I(t \leq C_i)$, the hazard of $N_i(t)$ takes the form of

$$d\bar{\Lambda}_i(t) = Y_i(t) e^{-\boldsymbol{X}_i^T \boldsymbol{\beta} t + \boldsymbol{X}_i^T \boldsymbol{\alpha}} d\bar{\Lambda}_0(t), \tag{2.24}$$

where $d\bar{\Lambda}_0(t) = e^{-H_0(t)} d\bar{H}_0(t)$ with $H_0(t) = \int_0^t h_0(s) ds$. Note that equation (2.24) is just the PH model. Lin et al. (1998) prove that the process

$$M_i(t) = N_i(t) - \int_0^t Y_i(s) e^{-\boldsymbol{X}_i^T \boldsymbol{\beta} s + \boldsymbol{X}_i^T \boldsymbol{\alpha}} d\bar{\Lambda}_0(s)$$

is a martingale. One can then apply the partial likelihood method to model (2.24) for inference about $\boldsymbol{\beta}$, yielding the partial likelihood score function for $\boldsymbol{\beta}$ given $\boldsymbol{\alpha}$

$$U_{\boldsymbol{\beta}}(\boldsymbol{\beta}, \boldsymbol{\alpha}) = \sum_{i=1}^{n} \int_0^{\infty} \Big[t\boldsymbol{X}_i - \frac{W_{\boldsymbol{\beta}}^{(1)}(t; \boldsymbol{\beta}, \boldsymbol{\alpha})}{W_{\boldsymbol{\beta}}^{(0)}(t; \boldsymbol{\beta}, \boldsymbol{\alpha})} \Big] dN_i(t), \tag{2.25}$$

where, for $j = 0, 1$,

$$W_{\boldsymbol{\beta}}^{(j)}(t; \boldsymbol{\beta}, \boldsymbol{\alpha}) = \sum_{i=1}^{n} Y_i(t) (t\boldsymbol{X}_i)^{(j)} e^{-t\boldsymbol{X}_i^T \boldsymbol{\beta} + \boldsymbol{X}_i^T \boldsymbol{\alpha}},$$

with $(t\boldsymbol{X}_i)^{(0)} = 1$ and $(t\boldsymbol{X}_i)^{(1)} = t\boldsymbol{X}_i$. Since the monitoring time C_i is always observed, for estimation of $\boldsymbol{\alpha}$, Lin et al. (1998) apply the partial likelihood approach to model (2.23). This gives the partial likelihood score function for $\boldsymbol{\alpha}$,

$$U_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \int_0^{\infty} \Big[\boldsymbol{X}_i - \frac{W_{\boldsymbol{\alpha}}^{(1)}(t; \boldsymbol{\alpha})}{W_{\boldsymbol{\alpha}}^{(0)}(t; \boldsymbol{\alpha})} \Big] dI(C_i \leq t),$$

where, for $j = 0, 1$, $W_{\boldsymbol{\alpha}}^{(j)}(t; \boldsymbol{\alpha}) = \sum_{i=1}^{n} Y_i(t) \boldsymbol{X}_i^{(j)} e^{\boldsymbol{X}_i^T \boldsymbol{\alpha}}$ with $\boldsymbol{X}_i^{(0)} = 1$ and $\boldsymbol{X}_i^{(1)} = \boldsymbol{X}_i$. Let $\hat{\boldsymbol{\alpha}}$ be the solution to $U_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}) = \mathbf{0}$. Then the estimate of $\boldsymbol{\beta}$ is computed by finding a root of $U_{\boldsymbol{\beta}}(\boldsymbol{\beta}, \hat{\boldsymbol{\alpha}})$. Both $\hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{\beta}}$ can be obtained by the Newton algorithm. Although the

approach requires C_i to follow the PH model, which could result in biased estimates of $\boldsymbol{\beta}$ if the assumption is incorrect, one can easily verify the PH assumption because C_i is always observed. On the other hand, the dependence of C_i on $\boldsymbol{X}_i$ through the PH model may cause dependent censoring, since T_i is also dependent on $\boldsymbol{X}_i$, but through the AH model.

Motivated by Lin et al. (1998), Wang et al. (2010) propose an estimating equation approach for interval-censored data $\{(L_i, R_i], \boldsymbol{X}_i : i = 1, \cdots, n\}$. In this approach, it is assumed that there exist only two monitoring times for each subject i, denoted by C_{i1} and C_{i2} with $C_{i1} < C_{i2}$. Assume that T_i is independent of C_{i1} and C_{i2}. Define indicator functions by $\xi_{1i} = I(T_i < C_{i1})$, $\xi_{2i} = I(C_{i1} \leq T_i < C_{i2})$ and $\xi_{3i} = 1 - \xi_{1i} - \xi_{2i}$. From these indicator functions, one can determine whether failure of subject i has occurred before C_{i1}, during the interval $(C_{i1}, C_{i2}]$, or after C_{i2}. In other words, for $\xi_{1i} = 1$, we have $L_i = 0$ and $R_i = C_{i1}$; for $\xi_{2i} = 1$, $L_i = C_{i1}$ and $R_i = C_{i2}$; and for $\xi_{3i} = 1$, $L_i = C_{i2}$ and $R_i = +\infty$. Similar to the analysis by Lin et al. (1998), Wang et al. (2010) model C_{i1} and C_{i2} respectively through the PH model,

$$d\bar{H}_{i1}(t|\boldsymbol{X}_i) = e^{\boldsymbol{X}_i^T\boldsymbol{\alpha}} d\bar{H}_{01}(t) \tag{2.26}$$

and

$$d\bar{H}_{i2}(t|C_{i1}, \boldsymbol{X}_i) = I(t > C_{i1}) e^{\boldsymbol{X}_i^T\boldsymbol{\alpha}} d\bar{H}_{02}(t), \tag{2.27}$$

where $\bar{H}_{01}(\cdot)$ and $\bar{H}_{02}(\cdot)$ are unspecified cumulative baseline hazard functions, and $\boldsymbol{\alpha}$ is a $p \times 1$ vector of unknown regression parameters. For each subject i, define counting processes $N_{i1}(t) = (1 - \xi_{1i}) I(t \geq C_{i1})$ and conditional C_{i1},

$$N_{i2}(t) = \begin{cases} \xi_{3i} I(t \geq C_{i2}) & \text{if } t \geq C_{i1} \\ 0 & \text{if } t < C_{i1}. \end{cases}$$

Define $Y_{i1}(t) = I(t \leq C_{i1})$ and $Y_{i2}(t) = I(C_{i1} < t \leq C_{i2})$. Following similar arguments as those in Lin et al. (1998), under models (2.22), (2.26) and (2.27), the hazard functions

for $N_{i1}(t)$ and $N_{i2}(t)$ are respectively derived by

$$d\bar{\Lambda}_{i1}(t) = Y_{i1}(t)e^{-\boldsymbol{X}_i^T\boldsymbol{\beta}t+\boldsymbol{X}_i^T\boldsymbol{\alpha}}d\bar{\Lambda}_{01}(t) \tag{2.28}$$

and

$$d\bar{\Lambda}_{i2}(t) = Y_{i2}(t)e^{-\boldsymbol{X}_i^T\boldsymbol{\beta}t+\boldsymbol{X}_i^T\boldsymbol{\alpha}}d\bar{\Lambda}_{02}(t), \tag{2.29}$$

where $d\bar{\Lambda}_{01}(t) = e^{-H_0(t)}d\bar{H}_{01}(t)$ and $d\bar{\Lambda}_{02}(t) = e^{-H_0(t)}d\bar{H}_{02}(t)$. It is clear that models (2.28) and (2.29) are the PH models, and (2.29) is a conditional model since the starting time point is C_{i1}. Given $\boldsymbol{\alpha}$, the score function for $\boldsymbol{\beta}$ is given by

$$U_{\boldsymbol{\beta}}(\boldsymbol{\beta},\boldsymbol{\alpha}) = \sum_{i=1}^{n}\left[\int_0^\infty\left\{t\boldsymbol{X}_i - \frac{W_{1,\boldsymbol{\beta}}^{(1)}(t,\boldsymbol{\beta},\boldsymbol{\alpha})}{W_{1,\boldsymbol{\beta}}^{(0)}(t,\boldsymbol{\beta},\boldsymbol{\alpha})}\right\}dN_{i1}(t)+\int_0^\infty\left\{t\boldsymbol{X}_i - \frac{W_{2,\boldsymbol{\beta}}^{(1)}(t,\boldsymbol{\beta},\boldsymbol{\alpha})}{W_{2,\boldsymbol{\beta}}^{(0)}(t,\boldsymbol{\beta},\boldsymbol{\alpha})}\right\}dN_{i2}(t)\right],$$

where, for $j = 0, 1$,

$$W_{1,\boldsymbol{\beta}}^{(j)}(t,\boldsymbol{\beta},\boldsymbol{\alpha}) = \sum_{i=1}^{n}Y_{i1}(t)e^{-t\boldsymbol{X}_i^T\boldsymbol{\beta}+\boldsymbol{X}_i^T\boldsymbol{\alpha}}(t\boldsymbol{X}_i)^{(j)}$$

and

$$W_{2,\boldsymbol{\beta}}^{(j)}(t,\boldsymbol{\beta},\boldsymbol{\alpha}) = \sum_{i=1}^{n}Y_{i2}(t)e^{-t\boldsymbol{X}_i^T\boldsymbol{\beta}+\boldsymbol{X}_i^T\boldsymbol{\alpha}}(t\boldsymbol{X}_i)^{(j)}$$

with $(t\boldsymbol{X}_i)^{(0)} = 1$ and $(t\boldsymbol{X}_i)^{(1)} = t\boldsymbol{X}_i$. Define counting processes $\tilde{N}_{i1}(t) = I(t \geq C_{i1})$ and, conditional on C_{i1},

$$\tilde{N}_{i2}(t) = \begin{cases} I(t \geq C_{i2}) & \text{if } t \geq C_{i1} \\ 0 & \text{if } t < C_{i1}. \end{cases}$$

For estimation of $\boldsymbol{\alpha}$, one can apply the partial likelihood approach to (2.26) and (2.27) to obtain the score function for $\boldsymbol{\alpha}$

$$U_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\left[\int_0^\infty\left\{\boldsymbol{X}_i - \frac{W_{1,\boldsymbol{\alpha}}^{(1)}(t,\boldsymbol{\alpha})}{W_{1,\boldsymbol{\alpha}}^{(0)}(t,\boldsymbol{\alpha})}\right\}d\tilde{N}_{i1}(t) + \int_0^\infty\left\{\boldsymbol{X}_i - \frac{W_{2,\boldsymbol{\alpha}}^{(1)}(t,\boldsymbol{\alpha})}{W_{2,\boldsymbol{\alpha}}^{(0)}(t,\boldsymbol{\alpha})}\right\}d\tilde{N}_{i2}(t)\right],$$

where, for $j = 0, 1$,

$$W_{1,\boldsymbol{\alpha}}^{(j)}(t,\boldsymbol{\alpha}) = \sum_{i=1}^{n}Y_{i1}(t)e^{\boldsymbol{X}_i^T\boldsymbol{\alpha}}\boldsymbol{X}_i^{(j)}$$

and

$$W_{2,\boldsymbol{\alpha}}^{(j)}(t,\boldsymbol{\alpha}) = \sum_{i=1}^{n}Y_{i2}(t)e^{\boldsymbol{X}_i^T\boldsymbol{\alpha}}\boldsymbol{X}_i^{(j)}$$

with $\boldsymbol{X}_i^{(0)} = 1$ and $\boldsymbol{X}_i^{(1)} = \boldsymbol{X}_i$. Let $\hat{\boldsymbol{\alpha}}$ be the root of $U_{\boldsymbol{\alpha}}(\boldsymbol{\alpha})$, then $\boldsymbol{\beta}$ is estimated by solving $U_{\boldsymbol{\beta}}(\boldsymbol{\beta}, \hat{\boldsymbol{\alpha}}) = \mathbf{0}$. Wang et al. (2010) show that the resulting estimators $\hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{\beta}}$ are consistent and asymptotically normal, and the covariance matrix for $\hat{\boldsymbol{\beta}}$ can be consistently estimated.

The advantage of the counting process estimation approach is that it does not involve estimation of any baseline hazard function. However, the method assumes the PH model for the monitoring time variables, which requires the user to test the validity of this assumption before applying it.

2.2.2 Maximum likelihood (ML) approach

Alternative to the counting process estimation methods, Ghosh (2001) and Zeng et al. (2006) develop maximum likelihood (ML) approaches for the AH model. Ghosh (2001) considers current status data, and fits the AH model by estimating $\boldsymbol{\beta}$ and a cumulative baseline hazard function $H_0(\cdot)$. These estimates are computed by using a primal-dual interior point algorithm (Wright, 1997), where the algorithm imposes constraints of positivity and monotonic increasing on $H_0(\cdot)$ and the cumulative hazard $H_i(\cdot)$. The resulting maximum likelihood estimator of $\boldsymbol{\beta}$ is shown to be consistent and converge to a multivariate normal distribution. Details of the primal-dual interior point algorithm will be given in Chapter 4, where we will also use this algorithm for estimations in the AH model with partly interval-censored data.

Zeng et al. (2006) fit the AH model with interval-censored data, where the log-likelihood function is expressed in terms of $S_0(\cdot)$ and $\boldsymbol{\beta}$ as

$$\ell(\boldsymbol{\beta}, S_0) = \sum_{i=1}^{n} \log \left\{ S_0(L_i) e^{-\boldsymbol{X}_i^T \boldsymbol{\beta} L_i} - S_0(R_i) e^{-\boldsymbol{X}_i^T \boldsymbol{\beta} R_i} \right\}. \tag{2.30}$$

Similar to Section 2.1.3, let $0 = u_0 < u_1 < u_2 < \cdots < u_m < u_{m+1} = \infty$ denote the unique ordered time points of all observed interval end points $\{L_i, R_i; i = 1, \cdots, n\}$. Let

$\boldsymbol{S}_0 = [S_0(u_0), S_0(u_1), \cdots, S_0(u_{m+1})]^T$, where $S_0(u_0) = 1$ and $S_0(u_{m+1}) = 0$. Estimation of $S_0(\cdot)$ is considered at these ordered distinct time points. Define indicator functions $\zeta_{ij} = I(u_j \in (L_i, R_i])$ for $j = 1, \cdots, m+1$ and $i = 1, \cdots, n$. Then the log-likelihood (2.30) can be rewritten as

$$\ell(\boldsymbol{\beta}, \boldsymbol{S}_0) = \sum_{i=1}^{n} \log \left\{ \sum_{j=1}^{m+1} \zeta_{ij} \left[S_0(u_{j-1}) e^{-\boldsymbol{X}_i^T \boldsymbol{\beta} u_{j-1}} - S_0(u_j) e^{-\boldsymbol{X}_i^T \boldsymbol{\beta} u_j} \right] \right\}. \tag{2.31}$$

In order to ensure the positivity and monotonic decreasing of $S_0(\cdot)$, which is that $0 < S_0(u_m) < \cdots < S_0(u_1) < 1$, transformation is performed on $S_0(\cdot)$ according to

$$\omega_j = \begin{cases} \log S_0(u_1) & \text{for } j = 1 \\ \log[S_0(u_{j-1}) - S_0(u_j)] & \text{for } j = 2, \cdots, m. \end{cases}$$

Let $\boldsymbol{\omega} = [\omega_1, \cdots, \omega_m]^T$. Estimates of $\boldsymbol{\beta}$ and $\boldsymbol{\omega}$ are obtained by maximizing the log-likelihood $\ell(\boldsymbol{\beta}, \boldsymbol{S}_0(\boldsymbol{\omega}))$ that is given by replacing $\boldsymbol{S}_0$ in (2.31) by $\boldsymbol{\omega}$. The Newton algorithm can be used to solve this optimization problem. However, this method may be time consuming when the sample size is large, because of the need for estimation of the baseline survival at each distinct observed time point.

2.2.3 Generalized linear model (GLM) approach

Farrington (1996) fits the AH model for interval censored data using a generalized linear model (GLM) approach. Before giving descriptions of the method, we define some notation. Let n_L be the number of subjects with their failure times left-censored, n_I the number of subjects finite interval-censored and n_R the number of subjects right-censored, then we have $n = n_L + n_R + n_I$. Denote $S(\cdot|\boldsymbol{X}_i)$ as the survival function conditional on $\boldsymbol{X}_i$. The likelihood is expressed in terms of $S(\cdot|\boldsymbol{X}_i)$ by

$$L = \prod_{i=1}^{n_L} [1 - S_i(R_i|\boldsymbol{X}_i)] \prod_{i=n_L+1}^{n_L+n_R} S_i(L_i|\boldsymbol{X}_i) \prod_{i=n_L+n_R+1}^{n} S_i(L_i|\boldsymbol{X}_i) \Big[1 - \frac{S_i(R_i|\boldsymbol{X}_i)}{S_i(L_i|\boldsymbol{X}_i)} \Big], \tag{2.32}$$

where the first product term is the likelihood for the left-censored observations, the second for the right-censored observations and the third for the finite interval-censored observations.

The GLM approach considers (2.32) as a particular realization of an associated model involving $n + n_I$ independent Bernoulli trials with response r_i and probability π_i, where $i = 1, \cdots, n+n_I$. The associated model is defined as follows. Consider the ith observation, if it is left-censored, it contributes one Bernoulli trial with $\pi_i = 1 - S(R_i|\boldsymbol{X}_i)$, $r_i = 1$ and $D_i = (0, R_i]$. If, on the other hand, the ith observation is right-censored, it also contributes one Bernoulli trial with $\pi_i = 1 - S(L_i|\boldsymbol{X}_i)$, $r_i = 0$ and $D_i = (0, L_i]$. Finally, if the ith observation is finite interval-censored (with i from $n - n_I + 1$ to n), it contributes two Bernoulli trails. The first has $\pi_i = 1 - S(L_i|\boldsymbol{X}_i)$, $r_i = 0$ and $D_i = (0, L_i]$, and the second has $\pi_{i+n_I} = 1 - \frac{S(R_i|\boldsymbol{X}_i)}{S(L_i|\boldsymbol{X}_i)}$, $r_i = 1$ and $D_{i+n_I} = (L_i, R_i]$. Then (2.32) is rewritten as

$$L = \prod_{i=1}^{n+n_I} \pi_i^{r_i}(1-\pi_i)^{1-r_i}, \tag{2.33}$$

where

$$\pi_i = 1 - \exp\left\{-\int_{D_i} h(t|\boldsymbol{X}_i)dt\right\}. \tag{2.34}$$

Therefore, maximizing (2.32) is equivalent to maximizing (2.33). In this method, the baseline hazard $h_0(\cdot)$ is assumed to be piecewise constant over some intervals, $(\tau_{k-1}, \tau_k]$, $k = 1, \cdots, m$, which divide up the time line. Then $h_0(t) = \theta_k$ for $t \in (\tau_{k-1}, \tau_k]$. Let $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_m]^T$. Now denote the length of D_i by a_i, and the length of $D_i \cap (\tau_{k-1}, \tau_k]$ by b_{ik}. Then based on (2.34), the probability π_i is related to a linear predictor through

$$\eta_i = -\ln(1-\pi_i) = \boldsymbol{X}_i^T\boldsymbol{\beta}a_i + \sum_{k=1}^{m}\theta_k b_{ik}. \tag{2.35}$$

Then $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ can be estimated by fitting the generalized linear model. However, in the estimation procedure, one has to ensure all the linear predictors in (2.35) remain positive, which may cause some difficulties. In addition, neither non-negativity nor smoothness for

the baseline hazard estimate can be guaranteed.

2.3 Accelerated Failure Time (AFT) Model

In this section, we review existing methods for estimations in the accelerated failure time (AFT) model. For each subject i, the AFT model specifies that

$$\log T_i = \boldsymbol{X}_i^T \boldsymbol{\beta} + \epsilon_i, \tag{2.36}$$

where $\boldsymbol{X}_i$ is a covariate vector with dimension p, assumed to be time-independent, $\boldsymbol{\beta}$ is a $p \times 1$ regression coefficient vector and ϵ_i is an error variable with its distribution function unknown.

2.3.1 Least squares approach

The estimation procedures in the AFT model have been studied extensively in the literature for right-censored data, and several estimators have been proposed. For example, Miller (1976), and Buckley and James (1979) apply a least squares method. However, the computation of the least squares estimator is based on an unstable iterative algorithm, which may lead to multiple limiting values and hence the resulting solution may not converge to the least squares estimator (Currie, 1996).

2.3.2 Linear rank estimation approach

Alternative to the least squares estimator, rank estimators have been studied by several authors. Among them, Tsiatis (1990) uses linear rank statistics, developed by Prentice (1978) as estimation functions for $\boldsymbol{\beta}$ with right-censored data. The linear rank statistics are usually defined by

$$S(\boldsymbol{\beta}) = \sum_{i=1}^{n} c_i(\boldsymbol{\beta}) \boldsymbol{X}_i, \tag{2.37}$$

where $c_i(\cdot)$ can be either assigned or estimated. As discussed below, one can choose or estimate $c_i(\cdot)$ such that $S(\boldsymbol{\beta})$ approximates the score function for $\boldsymbol{\beta}$.

Rabinowitz et al. (1995) propose a class of linear rank statistics for estimating $\boldsymbol{\beta}$ with interval-censored data, $\{(L_i, R_i]; i = 1, \cdots, n\}$. To present this method, we first give some notations. Denote $F_\epsilon(\cdot)$ as the distribution function of ϵ. Define $L_i(\boldsymbol{\beta}) = \log L_i - \boldsymbol{X}_i^T\boldsymbol{\beta}$ and $R_i(\boldsymbol{\beta}) = \log R_i - \boldsymbol{X}_i^T\boldsymbol{\beta}$. Clearly, for $L_i = 0$, $L_i(\boldsymbol{\beta}) = -\infty$ and for $R_i = +\infty$, $R_i(\boldsymbol{\beta}) = +\infty$. Define a vector $\boldsymbol{F}_\epsilon$ by $\boldsymbol{F}_\epsilon = [F_\epsilon(L_1(\boldsymbol{\beta})), F_\epsilon(R_1(\boldsymbol{\beta})), \cdots, F_\epsilon(L_n(\boldsymbol{\beta})), F_\epsilon(R_n(\boldsymbol{\beta}))]^T$, then the log-likelihood function is

$$\ell_R(\boldsymbol{\beta}, \boldsymbol{F}_\epsilon) = \sum_{i=1}^{n} \log[F_\epsilon(R_i(\boldsymbol{\beta})) - F_\epsilon(L_i(\boldsymbol{\beta}))]. \tag{2.38}$$

Based on (2.38), Rabinowitz et al. (1995) choose $c_i(\boldsymbol{\beta})$ in (2.37) by

$$c_i(\boldsymbol{\beta}) = \frac{g[F_\epsilon(R_i(\boldsymbol{\beta}))] - g[F_\epsilon(L_i(\boldsymbol{\beta}))]}{F_\epsilon(R_i(\boldsymbol{\beta})) - F_\epsilon(L_i(\boldsymbol{\beta}))},$$

which gives rise to a set of statistics

$$S_R(\boldsymbol{\beta}) = \sum_{i=1}^{n} \frac{g[F_\epsilon(R_i(\boldsymbol{\beta}))] - g[F_\epsilon(L_i(\boldsymbol{\beta}))]}{F_\epsilon(R_i(\boldsymbol{\beta})) - F_\epsilon(L_i(\boldsymbol{\beta}))} \boldsymbol{X}_i \tag{2.39}$$

with $\boldsymbol{F}_\epsilon$ replaced by the maximum likelihood estimate $\hat{\boldsymbol{F}}_\epsilon$ derived by maximizing (2.38) with a fixed $\boldsymbol{\beta}$. In the statistics (2.39) , $g(\cdot)$ is a weight function. For example, if selecting $g(\cdot) = f_\epsilon \circ F_\epsilon^{-1}(\cdot)$, then $S_R(\boldsymbol{\beta})$ becomes the score function for $\boldsymbol{\beta}$. An advantage of the statistics given in (2.39) is that the weight function $g(\cdot)$ can be selected in such a way that the resulting estimate of $\boldsymbol{\beta}$ will have the minimum asymptotic variance under suitable regularity conditions. However, the computational effort involved can be heavy for the method to be applied in practice (Betensky et al., 2001).

Alternative to this computationally demanding procedure, Betensky et al. (2001) propose a method based on the assumption that, for each subject i, there is a sequence of monitoring times which are independent of T_i. Recall the definition (1.4) given in Chapter 1, where we formulate the monitoring times by $\{M_i, C_{ij}, \xi_{ij}, i = 1, \cdots, n; j = 1, \cdots, M_i\}$,

where M_i is the number of monitoring times for subject i, $C_{i1} < \cdots < C_{iM_i}$ are the monitoring times, and ξ_{ij} is an indicator function defined by $\xi_{ij} = I(C_{ij-1} < T_i \leq C_{ij})$. In order to choose $c_i(\cdot)$ in (2.37), Betensky et al. (2001) treat all the C_{ij}'s as if they arise from $M_1 + \cdots + M_n$ independent subjects. Define $I_{ij} = I(T_i \leq C_{ij})$, then a set of current status data is obtained and denoted by $\{C_{ij}, I_{ij}; j = 1, \cdots, M_i, i = 1, \cdots, n\}$. Based on the new set of current status data, the log-likelihood function is

$$\ell_B(\boldsymbol{\beta}, \boldsymbol{F}_\epsilon) = \sum_{i=1}^{n} \sum_{j=1}^{M_i} \Big\{ I_{ij} \log \big[F_\epsilon(C_{ij}(\boldsymbol{\beta}))\big] + (1 - I_{ij}) \log \big[1 - F_\epsilon(C_{ij}(\boldsymbol{\beta}))\big] \Big\}, \tag{2.40}$$

where $\boldsymbol{F}_\epsilon = [F_\epsilon(C_{11}(\boldsymbol{\beta})), \cdots, F_\epsilon(C_{nM_n}(\boldsymbol{\beta}))]^T$, and $C_{ij}(\boldsymbol{\beta}) = \log C_{ij} - \boldsymbol{X}_i^T \boldsymbol{\beta}$. The first derivative of $\ell_B(\boldsymbol{\beta}, \boldsymbol{F}_\epsilon)$ with respective to $\boldsymbol{\beta}$ gives

$$\frac{\ell_B(\boldsymbol{\beta}, \boldsymbol{F}_\epsilon)}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{n} \boldsymbol{X}_i \sum_{j=1}^{M_i} \frac{-f_\epsilon(C_{ij}(\boldsymbol{\beta}))}{F_\epsilon(C_{ij}(\boldsymbol{\beta}))\big[1 - F_\epsilon(C_{ij}(\boldsymbol{\beta}))\big]} \big[I_{ij} - F_\epsilon(C_{ij}(\boldsymbol{\beta}))\big].$$

Since $\mathrm{E}(I_{ij}) = F_\epsilon(C_{ij}(\boldsymbol{\beta}))$, Betensky et al (2001) suggest taking

$$c_i(\boldsymbol{\beta}) = \sum_{j=1}^{M_i} \big[I_{ij} - \hat{F}_\epsilon(C_{ij}(\boldsymbol{\beta}))\big],$$

which results in a linear rank statistic

$$S_B(\boldsymbol{\beta}) = \sum_{i=1}^{n} \sum_{j=1}^{M_i} \big[I_{ij} - \hat{F}_\epsilon(C_{ij}(\boldsymbol{\beta}))\big] \boldsymbol{X}_i, \tag{2.41}$$

where $\hat{F}_\epsilon(\cdot)$ is the MLE of $F_\epsilon(\cdot)$ obtained by maximizing (2.40) with a fixed $\boldsymbol{\beta}$. Betensky et al. (2001) show that asymptotically $S_B(\boldsymbol{\beta})$ has expectation of zero, hence the estimate of $\boldsymbol{\beta}$ can be calculated by searching for the root of $S_B(\boldsymbol{\beta})$.

Li and Pu (2003) derive a rank statistic by directly ranking the observed censored intervals $\{(L_i, R_i]; i = 1, \cdots, n\}$. The proposed statistic is given by

$$S_{LP}(\beta) = \sum_{i<k} [I(X_i < X_k) - I(X_i > X_k)][I(R_i(\beta) < L_k(\beta)) - I(L_i(\beta) > R_k(\beta))], \tag{2.42}$$

where $I(\cdot)$ is the indicator function. However, in this proposed statistic, it is assumed that there exists only one covariate, which means that the X_i's and β are scalars. It is not straightforward to generalize this method to multiple covariates.

2.3.3 Penalized likelihood approach

Komárek et al.(2005) propose a penalized likelihood method for the AFT model with partly interval-censored data. In this method, they first standardize the error variable ϵ_i by reformatting (2.36) as

$$\log T_i = \alpha + \boldsymbol{X}_i^T \boldsymbol{\beta} + \sigma \epsilon_i,$$

where $\mathrm{E}(\epsilon_i) = 0$, $\mathrm{Var}(\epsilon_i) = 1$, and α and σ are the scale parameters. The method starts with modeling the density function of the standardized ϵ by a linear combination of Gaussian density functions, that is

$$f_\epsilon(t) = \sum_{j=1}^{m} \theta_j \varphi(t; \mu_j, \sigma_0^2), \tag{2.43}$$

where $\varphi(t; \mu_j, \sigma_0^2)$ is the Gaussian density with mean μ_j and variance σ_0^2, and $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_m]^T$ is a coefficient vector. Values of μ_j's and σ_0 are fixed by design. Komárek et al.(2005) fit the AFT model by estimating α, $\log \sigma$, $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$, with three constraints imposed on $\boldsymbol{\theta}$, which are

$$\sum_{j=1}^{m} \theta_j = 1, \quad \theta_j > 0 \text{ for } j = 1, \cdots, m, \tag{2.44}$$

$$\sum_{j=1}^{m} \theta_j \mu_j = 0 \tag{2.45}$$

and

$$\sum_{j=1}^{m} \theta_j (\mu_j^2 + \sigma_0^2) = 1. \tag{2.46}$$

The constraint (2.44) guarantees $f_\epsilon(t)$ is a proper density function, i.e. $f_\epsilon(t) > 0$ and $\int f_\epsilon(t) dt = 1$. The constraints (2.45) and (2.46) ensure $\mathrm{E}(\epsilon_i) = 0$ and $\mathrm{Var}(\epsilon_i) = 1$ respectively. To impose the constraint (2.44), $\boldsymbol{\theta}$ is reparametrized by

$$\theta_j(\boldsymbol{c}) = \frac{\exp(c_j)}{\sum_{l=1}^{m} \exp(c_l)} \quad \text{for } j = 1, \cdots, m,$$

where $\boldsymbol{c} = (c_1, \cdots, c_m)^T$. The equality constraints (2.45) and (2.46) can be imposed by expressing, for example, c_{m-2} and c_{m-1} as functions of the remaining c_i's. Now let

$\boldsymbol{d} = (c_1, \cdots, c_{m-3}, c_m)^T$ and rewrite (2.43) as

$$f_\epsilon(t) = \sum_{j=1}^{m} \theta_j(\boldsymbol{d}) \varphi(t; \mu_j, \sigma_0^2). \tag{2.47}$$

The model parameters to be estimated are $\boldsymbol{\omega} = [\alpha, \log \sigma, \boldsymbol{\beta}^T, \boldsymbol{d}^T]$. Denote $\ell(\boldsymbol{\omega})$ as the log-likelihood. To ensure the estimate of $f_\epsilon(t)$ is smooth, a penalty function $J(f_\epsilon)$ is subtracted from $\ell(\boldsymbol{\omega})$, resulting in a penalized log-likelihood function

$$\Phi(\boldsymbol{\omega}) = \ell(\boldsymbol{\omega}) - \tilde{\lambda} J(f_\epsilon),$$

where $\tilde{\lambda}$ is a smooth parameter. Komárek et al.(2005) adopt a squared second-order difference penalty function. The estimate of $\boldsymbol{\omega}$ is computed by maximizing $\Phi(\boldsymbol{\omega})$ using the Newton algorithm, for example. In chapter 5, we will make comparisons between our proposed penalized likelihood method and the method by Komárek et al.(2005).

2.4 The proposed methods

In this section, we summarize the methods that will be developed in this dissertation to fit the PH, AH and AFT models. Our methods can be applied to partly interval-censored data which contains exactly observed, left-censored, finite interval-censored and right-censored failure time data.

We fit the PH model by estimating simultaneously the regression coefficient $\boldsymbol{\beta}$ and the baseline hazard function $h_0(\cdot)$. Our method attempts to maximize a penalized log-likelihood function, which is constructed with the assumption that the baseline hazard is approximated by piecewise constant functions. We obtain a smoothed baseline hazard estimate through a penalty function and a smoothing parameter. Choosing an appropriate smoothing parameter can substantially reduce the standard deviation, without significantly increasing the bias of the baseline hazard estimate. For estimations, we use the

Newton-MI algorithm, which combines the Newton algorithm and the multiplicative iterative (MI) algorithm of Ma (2010). The MI algorithm is used because (i) it guarantees non-negativity for the baseline hazard estimate, and (ii) it only demands the first derivative of the penalized log-likelihood with respect to the baseline hazard for estimating the baseline hazard, which makes it easy to derive and implement. The Newton-MI algorithm has been adopted by Ma et al. (2014) for fitting the PH model with right-censored data. We also develop asymptotic properties for the resulting maximum penalized log-likelihood (MPL) estimators.

For the AH model, we estimate $\boldsymbol{\beta}$ and $h_0(\cdot)$, where $h_0(\cdot)$ is assumed to be piecewise constant. We produce a smoothed baseline hazard estimate again by using a penalty function. Since the AH model assumes covariate effects act additively on the hazard function $h(\cdot|\boldsymbol{X}_i)$, we have to impose the non-negativity constraint on both $h_0(\cdot)$ and $h(\cdot|\boldsymbol{X}_i)$ when maximizing the penalized log-likelihood function. We obtain the MPL estimates of $\boldsymbol{\beta}$ and $h_0(\cdot)$ by using a primal-dual interior point algorithm (Wright, 1997), with which $h(\cdot|\boldsymbol{X}_i)$ and $h_0(\cdot)$ are constrained to be non-negative simultaneously. This algorithm has also been applied by Ghosh (2001) for estimations in the AH model with current status data.

In fitting the AFT model, we first model the hazard function of ϵ^*, where $\epsilon^* = e^{\epsilon}$, using a linear combination of Gaussian basis functions. Then a penalized log-likelihood function is derived, with a roughness penalty included for smoothness of the hazard estimate. The regression coefficient $\boldsymbol{\beta}$ and the hazard are estimated simultaneously by maximizing the penalized log-likelihood function. In the estimation procedure, we must ensure non-negativity of the hazard estimate. To solve this constrained optimization problem, we again use the Newton-MI algorithm.

Chapter 3

Maximum penalized log-likelihood approach for proportional hazard model with partly interval-censored failure time data

3.1 Introduction

In this chapter, we develop a penalized likelihood method to fit the proportional hazard (PH) model with partly interval-censored failure time data. The PH model assumes that the effect of covariates acts multiplicatively on the hazard function. For subject i, let T_i be the failure time, $\boldsymbol{X}_i$ the covariate vector and $h(t|\boldsymbol{X}_i)$ the hazard function at time t conditional on $\boldsymbol{X}_i$. The PH model then specifies that

$$h(t|\boldsymbol{X}_i) = h_0(t) \exp\{\boldsymbol{X}_i^T \boldsymbol{\beta}\},$$

where $\boldsymbol{\beta}$ is a regression coefficient vector and $h_0(\cdot)$ is an unspecified baseline hazard function (Kalbfleisch and Prentice, 2002). The PH model has been studied widely by

researchers with right-censored failure time data. For instance, Cox (1975) develops a partial likelihood method. The main advantage of this method is that it only involves estimating $\boldsymbol{\beta}$, avoiding estimations of the baseline hazard $h_0(\cdot)$. However, if failure times involve other censoring types, such as interval censoring, the partial likelihood method cannot be applied directly (Sun, 2006).

There exist some methodologies to fit the PH model with interval-censored data, which have been reviewed in Section 2.1. Pan (2000) considers a multiple imputation approach. This method iterates between two steps. Firstly, the failure time is imputed from left and finite interval censored data based on current estimates of $\boldsymbol{\beta}$ and the baseline survival $S_0(\cdot)$, with the right-censored data kept unchanged. Secondly, the Cox partial likelihood method is applied to the imputed data for updating the estimates. Satten (1996), Goggins et al. (1998) and Satten et al. (1998) propose a rank-based method, where the estimation of $h_0(\cdot)$ is not involved. Finkelstein (1986) develops a maximum likelihood (ML) approach to estimate $\boldsymbol{\beta}$ and $S_0(\cdot)$, where the constraint $0 \leq S_0(\cdot) \leq 1$ is imposed indirectly by using the transformation $\log[-\log S_0(\cdot)]$. Betensky et al. (2002), Joly et al. (1998), and Cai and Betensky (2003) fit the PH model by estimating $\boldsymbol{\beta}$ and $h_0(\cdot)$, and the resulting estimated baseline hazard is smooth. Specifically, Betensky et al. (2002) develop a local likelihood method, where $\log h_0(\cdot)$ is expressed in terms of a polynomial function. Cai and Betensky (2003) develop a penalized likelihood approach and model $\log h_0(\cdot)$ by a linear spline. Although the logarithm transformation assures non-negativity of the estimated baseline hazard, these estimation procedures will become unstable as the estimated baseline hazard tends to zero. Joly et al (1998) develop a penalized likelihood method, where $h_0(\cdot)$ is modeled by a linear combination of M-splines, and the non-negativity of the estimated baseline hazard is constrained by using squared spline coefficients. However, the way the constraint on $h_0(\cdot)$ imposed may cause convergence issues, particularly when some

coefficients are zero.

Our method proposed in this chapter is different from the methods described above in four aspects: (i) we develop a maximum penalized log-likelihood (MPL) method for partly interval-censored data, which contains exactly observed, left-, right-, and finite interval-censored failure time data; (ii) we model the baseline hazard $h_0(\cdot)$ by a linear combination of piecewise constant functions, and guarantee non-negativity of the estimated baseline hazard in a direct way by imposing a non-negativity constraint on the coefficients of the piecewise constant functions, (iii) we estimate $\boldsymbol{\beta}$ and $h_0(\cdot)$ simultaneously by using the Newton-MI algorithm, which combines the Newton algorithm and the Multiplicative Iterative (MI) algorithm (Ma, 2010), and (iv) we ensure smoothness of the estimated baseline hazard by using a penalty function. Under certain conditions, we show that the resulting MPL estimators are asymptotically normal and consistent.

The chapter is organized as follows. Section 3.2 constructs a penalized log-likelihood function with partly interval-censored data under the PH model. Section 3.3 develops the Newton-MI algorithm to compute the MPL estimators. Section 3.4 describes an automatic method of selecting the smoothing parameter. Asymptotic properties of the MPL estimators are presented in Section 3.5. Section 3.6 reports numerical results via simulations. Section 3.7 applies the MPL method to a real life example. Section 3.8 concludes with some discussions.

3.2 Penalized log-likelihood functions under the Proportional Hazard (PH) model

We first introduce some notations which will be used in this and the following chapters. Suppose there are n subjects in a survival study with their failure times denoted as

$\{T_i : i = 1, \ldots, n\}$, and some of them are censored and the corresponding censoring times are observed. Different censoring types are considered in our analysis. For subject i, its related subscript will generally be denoted by ic, where c takes one of the following values: L – if T_i is left censored, R – if right censored, I – if finite interval censored and O – if exactly observed (i.e., uncensored). However, in the case where there is no need to specify censoring types, we simply use the single subscript i. Let $\boldsymbol{X}_i$ be a $p \times 1$ vector of covariates, and we assume $\boldsymbol{X}_i$ is time-independent throughout the thesis. Then observations for subject i regarding T_i fall into one of the following four exclusive categories:

$$
\begin{array}{ll}
\{t_{iO}, \boldsymbol{X}_{iO}\} & \text{if } T_i \text{ is observed at } t_{iO},\ T_i = t_{iO}; \\
\{t_{iL}, \boldsymbol{X}_{iL},\} & \text{if } T_i \text{ is left censored at } t_{iL},\ T_i \in (0, t_{iL}]; \\
\{t_{iIL}, t_{iIR}, \boldsymbol{X}_{iI},\} & \text{if } T_i \text{ is interval censored within an interval } (t_{iIL}, t_{iIR}],\ T_i \in (t_{iIL}, t_{iIR}]; \\
\{t_{iR}, \boldsymbol{X}_{iR},\} & \text{if } T_i \text{ is right censored at } t_{iR},\ T_i \in (t_{iR}, +\infty).
\end{array}
\tag{3.1}
$$

Let n_O be the number of subjects with their failure times exactly observed, n_L the number of subjects with their failure times left-censored, n_I the number of subjects with their failure times finite interval-censored, and n_R the number of subjects with their failure times right-censored.

For each subject i, the PH model specifies a linear relationship between the logarithm of $h(t|\boldsymbol{X}_i)$ and $\boldsymbol{X}_i$ in the form of

$$
h(t|\boldsymbol{X}_i) = h_0(t) \exp\{\boldsymbol{X}_i^T \boldsymbol{\beta}\}. \tag{3.2}
$$

Since the form of $h_0(\cdot)$ is unknown, model (3.2) is a semi-parametric regression model. The corresponding cumulative hazard function is

$$
H(t|\boldsymbol{X_i}) = \int_0^t h(s|X_i)ds = H_0(t) \exp\{\boldsymbol{X}_i^T \boldsymbol{\beta}\} \tag{3.3}
$$

where $H_0(\cdot)$ is the cumulative baseline hazard function. We fit the PH model by estimating $\boldsymbol{\beta}$ and $h_0(\cdot)$.

We start the estimation procedure with approximating the baseline hazard $h_0(\cdot)$. Direct estimation of $h_0(\cdot)$ is ill-conditioned, since this is a problem of estimating an infinite dimensional parameter from finite observations. However, we can simplify the problem by discretizing $h_0(\cdot)$, or equivalently $h_0(\cdot)$ is taken to be piecewise constant. To be specific, assume all observations fall in a finite interval

$$\mathcal{I} = [t_{(1)}, t_{(n+n_I)}], \tag{3.4}$$

where $t_{(1)} = \min\{t_{iv}, i = 1, \cdots, n + n_I, v = O, L, IL, IR, R\}$ and $t_{(n+n_I)} = \max\{t_{iv}, i = 1, \cdots, n+n_I, v = O, L, IL, IR, R\}$. Suppose there are m bins, $\{B_1, \cdots, B_m\}$, partitioning the interval $\mathcal{I}$ with edge points $t_{(1)} = \tau_0 < \tau_1 < \cdots < \tau_m = t_{(n+n_I)}$, where

$$B_u = \{t : \tau_{u-1} < t \leq \tau_u\} \quad \text{for } u = 1, \cdots, m, \tag{3.5}$$

$\cup_{u=1}^{m} B_u = \mathcal{I}$ and $B_u \cap B_v = \emptyset$ if $u \neq v$. The baseline hazard is assumed to be piecewise constant over the bins (3.5). Then, at time t, it takes the form

$$h_0(t) = \sum_{u=1}^{m} \theta_u I(\tau_{u-1} < t \leq \tau_u), \tag{3.6}$$

where $I(\cdot)$ is an indicator function, and $\{\theta_u : u = 1, \cdots, m\}$ are basis coefficients. We define $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_m]^T$. Clearly, after discretization, estimating $h_0(\cdot)$ is equivalent to estimating the vector $\boldsymbol{\theta}$, subject to $\theta_u \geq 0$ for all u. We comment that although $h_0(\cdot)$ is discretized, its non-parametric nature is still somewhat preserved as there is no restriction on the number of bins m. The corresponding cumulative baseline hazard function is

$$H_0(t) = \sum_{u=1}^{m} \Big[(t - \tau_{u-1}) I(\tau_{u-1} < t \leq \tau_u) + \delta_u I(t > \tau_u)\Big]\theta_u, \tag{3.7}$$

where $\delta_j = \tau_j - \tau_{j-1}$ is the width of bin B_j. From the equations (3.2), (3.3), (3.6) and (3.7), the hazard, cumulative hazard and survival functions concerning $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ are then

expressed respectively as

$$h(t|\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{X}_i) = \Big[\sum_{u=1}^{m}\theta_u I(\tau_{u-1} < t \le \tau_u)\Big]\exp(\boldsymbol{X}_i^T\boldsymbol{\beta}), \tag{3.8}$$

$$H(t|\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{X}_i) = \left[\sum_{u=1}^{m}\Big((t-\tau_{u-1})I(\tau_{u-1} < t \le \tau_u) + \delta_u I(t > \tau_u)\Big)\theta_u\right]\exp(\boldsymbol{X}_i^T\boldsymbol{\beta}) \tag{3.9}$$

and

$$S(t|\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{X}_i) = \exp\left\{-\left[\sum_{u=1}^{m}\Big((t-\tau_{u-1})I(\tau_{u-1} < t \le \tau_u) + \delta_u I(t > \tau_u)\Big)\theta_u\right]\exp(\boldsymbol{X}_i^T\boldsymbol{\beta})\right\}. \tag{3.10}$$

Now we define $\mathcal{O}$ as the index set for exactly observed data, $\mathcal{L}$ the index set for left censored data, $\mathcal{I}$ the index set for finite interval-censored data and $\mathcal{R}$ the index set for right censored data. We assume that the censoring time is independent of the failure time T_i and that the distribution of censoring does not involve the regression coefficient vector $\boldsymbol{\beta}$. Then, using equations (3.8)-(3.10) and the notation (3.1), the log-likelihood function involving $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ is

$$\begin{aligned}\ell(\boldsymbol{\beta},\boldsymbol{\theta}) =& \sum_{i\in\mathcal{L}}\ln\Big[1 - S(t_{iL}|\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{X}_{iL})\Big] + \sum_{i\in\mathcal{I}}\ln\Big[S(t_{iIL}|\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{X}_{iI}) - S(t_{iIR}|\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{X}_{iI})\Big] \\ &- \sum_{i\in\mathcal{R}} H(t_{iR}|\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{X}_{iR}) + \sum_{u=1}^{m} N_u^O\cdot\log\theta_u + \sum_{i\in\mathcal{O}}\Big[\boldsymbol{X}_{iO}^T\boldsymbol{\beta} - H(t_{iO}|\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{X}_{iO})\Big],\end{aligned} \tag{3.11}$$

where N_u^O is the number of subjects with their failure times exactly observed and lying in bin B_u. To produce a smooth estimate of the baseline hazard, i.e., neighboring θ_u's are similar, a penalty function for the baseline hazard is subtracted from the log-likelihood (3.11), giving rise to a penalized log-likelihood function

$$\Phi(\boldsymbol{\beta},\boldsymbol{\theta}) = \ell(\boldsymbol{\beta},\boldsymbol{\theta}) - \tilde{\gamma}J(h_0), \tag{3.12}$$

where $\tilde{\gamma} \ge 0$ is a smoothing parameter used to control the trade-off between the goodness of fit to data and smoothness of the estimated baseline hazard, and $J(h_0)$ is a penalty

function. The penalty $J(h_0)$ in (3.12) can be, for instance, a roughness penalty, i.e., $J(h_0) = \int h_0''(u)du$, measuring total curvature of the baseline hazard. Due to discretization of $h_0(\cdot)$, we adopt a penalty representing the square of the second order differences, that is

$$J(h_0) = J(\boldsymbol{\theta}) = \sum_{j=2}^{m-1} (\theta_{j-1} - 2\theta_j + \theta_{j+1})^2 ,$$

This penalty can be rewritten in a quadratic form of

$$J(\boldsymbol{\theta}) = \boldsymbol{\theta}^T \boldsymbol{R} \boldsymbol{\theta}, \tag{3.13}$$

where $\boldsymbol{R} = \boldsymbol{C}^T\boldsymbol{C}$ with $\boldsymbol{C}$ being an $m \times m$ matrix given by the requirement that $\boldsymbol{C\theta}$ represents the second order difference of $\boldsymbol{\theta}$. Therefore, the matrix $\boldsymbol{R}$ is given by

$$\boldsymbol{R} = \begin{pmatrix} 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ -2 & 5 & -4 & 1 & 0 & 0 & 0 \\ 1 & -4 & 6 & -4 & 1 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 1 & -4 & 6 & -4 & 1 \\ 0 & 0 & 0 & 1 & -4 & 5 & -2 \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 \end{pmatrix}.$$

The penalty (3.13) helps to penalize the discrepancy between θ_u and the average of its neighborhoods. Smoothness of the estimated baseline hazard function is achieved when the neighbor of θ_u's are similar.

3.3 Maximum penalized likelihood estimation

We obtain the MPL estimators of $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ by maximizing the penalized log-likelihood function (3.12) with respect to $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$, with $\boldsymbol{\theta}$ non-negative, namely

$$(\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\theta}}) = \arg\max_{\boldsymbol{\beta},\boldsymbol{\theta}} \{\Phi(\boldsymbol{\beta}, \boldsymbol{\theta})\} \tag{3.14}$$

subject to $\boldsymbol{\theta} \geq \mathbf{0}_{m\times 1}$.

We combine the Newton Algorithm and the MI algorithm (Ma, 2010), named as the Newton-MI algorithm, to solve the constrained optimization problem (3.14). Before proceeding, we first give some notations. In deriving the first and second derivatives of $\Phi(\boldsymbol{\beta}, \boldsymbol{\theta})$, for simplicity, we denote $S(\cdot)$ to be $S(\cdot|\boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{X}_i)$ defined in (3.10), and $H(\cdot)$ to be $H(\cdot|\boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{X}_i)$ defined in (3.9). We define

$$\bar{\eta}_{iI} = H(t_{iIR})S(t_{iIR}) - H(t_{iIL})S(t_{iIL})$$

and

$$\zeta_{iI} = H^2(t_{iIR})S(t_{iIR}) - H^2(t_{iIL})S(t_{iIL}).$$

The first derivative of the objective function $\Phi(\boldsymbol{\beta}, \boldsymbol{\theta})$ with respective to $\boldsymbol{\beta}$ is

$$\frac{\partial \Phi(\boldsymbol{\beta}, \boldsymbol{\theta})}{\partial \boldsymbol{\beta}} = \boldsymbol{X}_L^T \mathcal{A} \mathbf{1}_{n_L} + \boldsymbol{X}_I^T \mathcal{B} \mathbf{1}_{n_I} - \boldsymbol{X}_R^T \mathcal{C} \mathbf{1}_{n_R} + \boldsymbol{X}_O^T \mathcal{D} \mathbf{1}_{n_O}, \tag{3.15}$$

where $\boldsymbol{X}_L$ is the covariate matrix for subjects with their failure times left-censored, $\boldsymbol{X}_I$ the covariate matrix for subjects with their failure times finite interval-censored, $\boldsymbol{X}_R$ for subjects with their failure times right-censored and $\boldsymbol{X}_O$ for subjects with their failure times exactly observed. In (3.15), the matrices $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$ are diagonal matrices defined by

$$\mathcal{A} = \text{diag}\left\{ \frac{H(t_{iL})S(t_{iL})}{1 - S(t_{iL})} \right\},$$

$$\mathcal{B} = \text{diag}\left\{ \frac{\bar{\eta}_{iI}}{S(t_{iIL}) - S(t_{iIR})} \right\},$$

$$\mathcal{C} = \text{diag}\{H(t_{iR})\},$$

and

$$\mathcal{D} = \text{diag}\{1 - H(t_{iO})\}.$$

For convenience in deriving the first and second derivatives of $\Phi(\boldsymbol{\beta}, \boldsymbol{\theta})$ with respect to θ_u, $u = 1, \cdots, m$, we introduce some notations. For subject i whose observed data

t_{iv} is in B_u, its t_{iv} is then denoted to be t_{iv_u} to reflect the dependence on bin B_u, and its corresponding covariate X_{iv} is denoted to be X_{iv_u}, where $v = \{L, IL, IR, R, O\}$. For subject i with its failure time finite interval-censored, if its censoring times t_{iIL} and t_{iIR} are in bins $t_{iIL} \in B_u$ and $t_{iIR} \in B_k$, then they are denoted respectively to be t_{iIL_u} and t_{iIR_k}, and its corresponding X_{iI} is denoted to be X_{iIL_u} or X_{iIR_k}. Let $\mathcal{O}_u$ be the index set for exactly observed data falling in B_u, $\mathcal{L}_u$ the index set for left censored data falling in B_u, $\mathcal{IL}_u$ the index set for finite interval-censored data with its left limit falling in B_u, i.e., $\mathcal{IL}_u = \{i : t_{iIL} \in B_u\}$, $\mathcal{IR}_u$ the index set for finite interval-censored data with its right limit falling in B_u, i.e., $\mathcal{IR}_u = \{i : t_{iIR} \in B_u\}$, and $\mathcal{R}_u$ the index set for right censored data falling in B_u. Let N_u^O be the number of subjects with their failure times exactly observed and lying in B_u. Define

$$a_{iv_u} = (t_{iv_u} - \tau_{u-1}) \exp(\boldsymbol{X}_{iv_u}^T \boldsymbol{\beta}). \tag{3.16}$$

Then the first derivative of $\Phi(\boldsymbol{\beta}, \boldsymbol{\theta})$ with respective to θ_u is given by

$$\begin{aligned}
\frac{\partial \Phi(\boldsymbol{\beta}, \boldsymbol{\theta})}{\partial \theta_u} = & \sum_{i \in \mathcal{L}_u} \left[\frac{a_{iL_u} S(t_{iL_u})}{1 - S(t_{iL_u})} \right] + \delta_u \sum_{w>u} \sum_{i \in \mathcal{L}_w} \left[\frac{S(t_{iL_w}) \exp(\boldsymbol{X}_{iL_w}^T \boldsymbol{\beta})}{1 - S(t_{iL_w})} \right] \\
& + \sum_{v<u} \sum_{i \in \mathcal{IL}_v} \left[\sum_{i \in \mathcal{IR}_u} \frac{a_{iIR_u} S(t_{iIR_u})}{S(t_{iIL_v}) - S(t_{iIR_u})} + \delta_u \sum_{w>u} \sum_{i \in \mathcal{IR}_w} \frac{\exp(\boldsymbol{X}_{iIR_w}^T \boldsymbol{\beta}) S(t_{iIR_w})}{S(t_{iIL_v}) - S(t_{iIR_w})} \right] \\
& + \sum_{i \in \mathcal{IL}_u} \left[\sum_{i \in \mathcal{IR}_u} \frac{a_{iIR_u} S(t_{iIR_u}) - a_{iIL_u} S(t_{iIL_u})}{S(t_{iIL_u}) - S(t_{iIR_u})} \right] \\
& + \sum_{i \in \mathcal{IL}_u} \sum_{w>u} \sum_{i \in \mathcal{IR}_w} \left[\frac{\delta_u S(t_{iIR_w}) \exp(\boldsymbol{X}_{iIR_w}^T \boldsymbol{\beta}) - a_{iIL_u} S(t_{iIL_u})}{S(t_{iIL_u}) - S(t_{iIR_w})} \right] \\
& - \delta_u \sum_{w \geq v > u} \sum_{i \in \mathcal{IL}_v} \sum_{i \in \mathcal{IR}_w} \exp(\boldsymbol{X}_{iIR_w}^T \boldsymbol{\beta}) - \sum_{i \in \mathcal{R}_u} a_{iR_u} - \delta_u \sum_{w>u} \sum_{i \in \mathcal{R}_w} \exp(\boldsymbol{X}_{iR_w}^T \boldsymbol{\beta}) \\
& - \sum_{i \in \mathcal{O}_u} a_{iO_u} - \delta_u \sum_{w>u} \sum_{i \in \mathcal{O}_w} \exp(\boldsymbol{X}_{iO_w}^T \boldsymbol{\beta}) + \frac{N_u^O}{\theta_u} - 2\tilde{\gamma} \boldsymbol{R}_u \boldsymbol{\theta},
\end{aligned} \tag{3.17}$$

where $\boldsymbol{R}_u$ is the u-th row of $\boldsymbol{R}$.

The Karush-Kuhn-Tucker (KKT) necessary conditions for the constrained MPL esti-

mations of $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ are

$$\frac{\partial \Phi(\boldsymbol{\beta}, \boldsymbol{\theta})}{\partial \beta_j} = 0 \tag{3.18}$$

for $j = 1, \cdots, p$, and

$$\frac{\partial \Phi(\boldsymbol{\beta}, \boldsymbol{\theta})}{\partial \theta_u} \begin{cases} = 0 & \text{if } \theta_u > 0 \\ < 0 & \text{if } \theta_u = 0 \end{cases} \tag{3.19}$$

for $u = 1, \cdots, m$. We compute MPL estimators of $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ by solving the equations (3.18) and (3.19) iteratively using the Newton-MI algorithm (Ma et al., 2014). The algorithm is outlined below.

1. We first choose initial values $\boldsymbol{\beta}^{(0)}$ and $\boldsymbol{\theta}^{(0)}$, where $\theta_u^{(0)} > 0$ for $u = 1, \cdots, m$.

2. Let $\boldsymbol{\beta}^{(k)}$ and $\boldsymbol{\theta}^{(k)}$ be the estimates of $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ at iteration k, respectively. By fixing $\boldsymbol{\theta}$ at $\boldsymbol{\theta}^{(k)}$, we maximize $\Phi(\boldsymbol{\beta}, \boldsymbol{\theta}^{(k)})$ with respect to $\boldsymbol{\beta}$ to obtain $\boldsymbol{\beta}^{(k+1)}$ using a modified Newton algorithm. A line search step is included in the procedure to guarantee increment of the penalized log-likelihood function $\Phi(\boldsymbol{\beta}, \boldsymbol{\theta}^{(k)})$ when $\boldsymbol{\beta}$ moves from $\boldsymbol{\beta}^{(k)}$ to $\boldsymbol{\beta}^{(k+1)}$, namely $\Phi(\boldsymbol{\beta}^{(k+1)}, \boldsymbol{\theta}^{(k)}) \geq \Phi(\boldsymbol{\beta}^{(k)}, \boldsymbol{\theta}^{(k)})$, where the equality holds only if the iterations have converged.

3. With $\boldsymbol{\beta}$ fixed at $\boldsymbol{\beta}^{(k+1)}$, we then maximize $\Phi(\boldsymbol{\beta}^{(k+1)}, \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ to obtain $\boldsymbol{\theta}^{(k+1)}$ using the multiplicative iterative (MI) algorithm (Ma, 2010). A line search step is included to guarantee $\Phi(\boldsymbol{\beta}^{(k+1)}, \boldsymbol{\theta})$ increases when $\boldsymbol{\theta}$ moves from $\boldsymbol{\theta}^{(k)}$ to $\boldsymbol{\theta}^{(k+1)}$, namely $\Phi(\boldsymbol{\beta}^{(k+1)}, \boldsymbol{\theta}^{(k+1)}) \geq \Phi(\boldsymbol{\beta}^{(k+1)}, \boldsymbol{\theta}^{(k)})$, where the equality holds only if the iterations have converged. The MI procedure respects the non-negativity constraint on $\boldsymbol{\theta}^{(k+1)}$.

4. Go to step 2 and the process is repeated until both $\boldsymbol{\beta}^{(k)}$ and $\boldsymbol{\theta}^{(k)}$ have converged satisfying the criterion $\max_j |\beta_j^{(k+1)} - \beta_j^{(k)}| < 10^{-5}$ and $\max_u |\theta_u^{(k+1)} - \theta_u^{(k)}| < 10^{-5}$.

The MI algorithm in step 3 has the advantage that it only demands the first derivative

of the penalized log-likelihood function with respect to the discretized baseline hazard, which leads to its straightforward derivation and implementation.

In Step 1 of the Newton-MI algorithm, one iteration of the Newton algorithm for updating $\boldsymbol{\beta}^{(k)}$ is

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \alpha^{(k)} \left[\frac{\partial^2 \Phi(\boldsymbol{\beta}^{(k)}, \boldsymbol{\theta}^{(k)})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} \right]^{-1} \frac{\partial \Phi(\boldsymbol{\beta}^{(k)}, \boldsymbol{\theta}^{(k)})}{\partial \boldsymbol{\beta}}, \tag{3.20}$$

where $\alpha^{(k)} \in (0, 1]$ is a line search step size. The step size can be computed by using the Armijo's rule (Luenberger, 2003). Firstly, from (3.20), we express $\boldsymbol{\beta}^{(k+1)}$ in terms of the step size α as

$$\boldsymbol{\beta}^{(k+1)}(\alpha) = \boldsymbol{\beta}^{(k)} + \alpha \boldsymbol{d}_N^{(k)}, \tag{3.21}$$

where

$$\boldsymbol{d}_N^{(k)} = - \left[\frac{\partial^2 \Phi(\boldsymbol{\beta}^{(k)}, \boldsymbol{\theta}^{(k)})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} \right]^{-1} \frac{\partial \Phi(\boldsymbol{\beta}^{(k)}, \boldsymbol{\theta}^{(k)})}{\partial \boldsymbol{\beta}}.$$

The line search starts with $\alpha = 1$, and then tests whether the following condition is satisfied:

$$\Phi(\boldsymbol{\beta}^{(k+1)}(\alpha), \boldsymbol{\theta}^{(k)}) \geq \Phi(\boldsymbol{\beta}^{(k)}, \boldsymbol{\theta}^{(k)}) + c\alpha \left[\frac{\partial \Phi(\boldsymbol{\beta}^{(k)}, \boldsymbol{\theta}^{(k)})}{\partial \boldsymbol{\beta}} \right]^T \boldsymbol{d}_N^{(k)}, \tag{3.22}$$

where $0 < c < 1$ is a fixed parameter and usually chosen to be small (e.g. $c = 10^{-2}$). If α satisfies (3.22), then stop; otherwise, reset $\alpha = \rho\alpha$, where $\rho \in (0, 1]$ (e.g. $\rho = 0.6$) and then re-valuate the condition (3.22). The second derivative $\partial^2 \Phi(\boldsymbol{\beta}, \boldsymbol{\theta}) / \partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T$ in (3.20) is derived by

$$\frac{\partial^2 \Phi(\boldsymbol{\beta}, \boldsymbol{\theta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} = -\boldsymbol{X}_L^T \mathcal{E} \boldsymbol{X}_L - \boldsymbol{X}_I^T \mathcal{F} \boldsymbol{X}_I - \boldsymbol{X}_R^T \mathcal{C} \boldsymbol{X}_R - \boldsymbol{X}_O^T \mathcal{G} \boldsymbol{X}_O, \tag{3.23}$$

where the matrices $\mathcal{E}$, $\mathcal{F}$ and $\mathcal{G}$ are diagonal matrices defined by

$$\mathcal{E} = \text{diag} \left\{ \frac{H(t_{iL}) S(t_{iL}) \Big[S(t_{iL}) - 1 + H(t_{iL}) \Big]}{\Big[1 - S(t_{iL}) \Big]^2} \right\},$$

$$\mathcal{F} = \text{diag} \left\{ \Big[\frac{\bar{\eta}_{iI}}{S(t_{iIL}) - S(t_{iIR})} \Big]^2 + \frac{\zeta_{iI} - \bar{\eta}_{iI}}{S(t_{iIL}) - S(t_{iIR})} \right\},$$

and

$$\mathcal{G} = \text{diag}\{H(t_{iO})\}.$$

To implement the MI algorithm in Step 2, we first define

$$\begin{aligned}\varsigma_u(\boldsymbol{\beta},\boldsymbol{\theta}) = &\sum_{i\in\mathcal{L}_u}\left[\frac{a_{iL_u}S(t_{iL_u})}{1-S(t_{iL_u})}\right] + \delta_u\sum_{w>u}\sum_{i\in\mathcal{L}_w}\left[\frac{S(t_{iL_w})\exp(\boldsymbol{X}_{iL_w}^T\boldsymbol{\beta})}{1-S(t_{iL_w})}\right] + \frac{N_u^O}{\theta_u}\\ &+\sum_{v<u}\sum_{i\in\mathcal{IL}_v}\left[\sum_{i\in\mathcal{IR}_u}\frac{a_{iIR_u}S(t_{iIR_u})}{S(t_{iIL_v})-S(t_{iIR_u})} + \delta_u\sum_{w>u}\sum_{i\in\mathcal{IR}_w}\frac{\exp(\boldsymbol{X}_{iIR_w}^T\boldsymbol{\beta})S(t_{iIR_w})}{S(t_{iIL_v})-S(t_{iIR_w})}\right]\\ &+\sum_{i\in\mathcal{IL}_u}\sum_{i\in\mathcal{IR}_u}\left[\frac{a_{iIR_u}S(t_{iIR_u})}{S(t_{iIL_u})-S(t_{iIR_u})}\right] + \delta_u\sum_{i\in\mathcal{IL}_u}\sum_{w>u}\sum_{i\in\mathcal{IR}_w}\left[\frac{S(t_{iIR_w})\exp(\boldsymbol{X}_{iIR_w}^T\boldsymbol{\beta})}{S(t_{iIL_u})-S(t_{iIR_w})}\right]\end{aligned}$$

and

$$\begin{aligned}\xi_u(\boldsymbol{\beta},\boldsymbol{\theta}) = &\sum_{i\in\mathcal{IL}_u}\sum_{i\in\mathcal{IR}_u}\left[\frac{a_{iIL_u}S(t_{iIL_u})}{S(t_{iIL_u})-S(t_{iIR_u})}\right] + \sum_{i\in\mathcal{IL}_u}\sum_{w>u}\sum_{i\in\mathcal{IR}_w}\left[\frac{a_{iIL_u}S(t_{iIL_u})}{S(t_{iIL_u})-S(t_{iIR_w})}\right]\\ &+\delta_u\sum_{w\geq v>u}\sum_{i\in\mathcal{IL}_v}\sum_{i\in\mathcal{IR}_w}\exp(\boldsymbol{X}_{iIR_w}^T\boldsymbol{\beta}) + \sum_{i\in\mathcal{R}_u}a_{iR_u} + \sum_{i\in\mathcal{O}_u}a_{iO_u}\\ &+\delta_u\sum_{w>u}\left[\sum_{i\in\mathcal{R}_w}\exp(\boldsymbol{X}_{iR_w}^T\boldsymbol{\beta}) + \sum_{i\in\mathcal{O}_w}\exp(\boldsymbol{X}_{iO_w}^T\boldsymbol{\beta})\right].\end{aligned}$$

For any constant c, we define $[c]^+ = cI(c \geq 0)$ and $[c]^- = cI(c \leq 0)$, so that $c = [c]^+ + [c]^-$. Then we rearrange (3.17) into an equation, of which both sides are non-negative, that is

$$\xi_u(\boldsymbol{\beta},\boldsymbol{\theta}) + 2\tilde{\gamma}[\boldsymbol{R}_u\boldsymbol{\theta}]^+ = \varsigma_u(\boldsymbol{\beta},\boldsymbol{\theta}) - 2\tilde{\gamma}[\boldsymbol{R}_u\boldsymbol{\theta}]^-. \tag{3.24}$$

Multiplying the equation (3.24) by θ_u, we have

$$\theta_u\{\xi_u(\boldsymbol{\beta},\boldsymbol{\theta}) + 2\tilde{\gamma}[\boldsymbol{R}_u\boldsymbol{\theta}]^+\} = \theta_u\{\varsigma_u(\boldsymbol{\beta},\boldsymbol{\theta}) - 2\tilde{\gamma}[\boldsymbol{R}_u\boldsymbol{\theta}]^-\}. \tag{3.25}$$

From the equation (3.25), we suggest updating θ_u by two steps. In the first step, a temporary estimate $\boldsymbol{\theta}^{(k+1/2)}$ is computed from $\boldsymbol{\theta}^{(k)}$ via

$$\theta_u^{(k+1/2)} = \theta_u^{(k)}\frac{\varsigma_u(\boldsymbol{\beta}^{(k+1)},\boldsymbol{\theta}^{(k)}) - 2\tilde{\gamma}[\boldsymbol{R}_u\boldsymbol{\theta}^{(k)}]^- + \epsilon_u}{\xi_u(\boldsymbol{\beta}^{(k+1)},\boldsymbol{\theta}^{(k)}) + 2\tilde{\gamma}[\boldsymbol{R}_u\boldsymbol{\theta}^{(k)}]^+ + \epsilon_u} \tag{3.26}$$

for $u = 1, \cdots, m$, where ϵ_u is a small constant (such as 10^{-5}) used to avoid a zero denominator. It is clearly seen from (3.26) that $\boldsymbol{\theta}^{(k+1/2)}$ satisfies the non-negativity constraint if

$\boldsymbol{\theta}^{(k)} > 0$. However, this $\boldsymbol{\theta}^{(k+1/2)}$ may fail to increase $\Phi(\boldsymbol{\beta}^{(k+1)}, \boldsymbol{\theta})$ when $\boldsymbol{\theta}$ moves from $\boldsymbol{\theta}^{(k)}$ to $\boldsymbol{\theta}^{(k+1/2)}$. Hence a line search step is required. We first rewrite the equation (3.26) as a gradient algorithm:

$$\boldsymbol{\theta}^{(k+1/2)} = \boldsymbol{\theta}^{(k)} + S(\boldsymbol{\theta}^{(k)}) \frac{\partial \Phi(\boldsymbol{\beta}^{(k+1)}, \boldsymbol{\theta}^{(k)})}{\partial \boldsymbol{\theta}}, \tag{3.27}$$

where $S(\boldsymbol{\theta}^{(k)})$ is a diagonal matrix with diagonal elements $s_u^{(k)}$, $u = 1, \cdots, m$, given by

$$s_u^{(k)} = \frac{\theta_u^{(k)}}{\xi_u(\boldsymbol{\beta}^{(k+1)}, \boldsymbol{\theta}^{(k)}) + 2\gamma[\boldsymbol{R}_u \boldsymbol{\theta}^{(k)}]^+ + \epsilon_u}. \tag{3.28}$$

Here $s_u^{(k)}$ is non-negative due to the fact that $\theta_u^{(k)}$ is constrained to be non-negative in the estimation procedure. Then, from equation (3.27), we know that the MI iteration given by (3.26) proceeds along the gradient direction with a non-negative step size. Let $\omega^{(k)}$ be the step size of the line search. Then, in the second step of the MI algorithm, $\boldsymbol{\theta}^{(k+1)}$ is computed by

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \omega^{(k)} \boldsymbol{d}_M^{(k)}, \tag{3.29}$$

where $\boldsymbol{d}_M^{(k)} = \boldsymbol{\theta}^{(k+1/2)} - \boldsymbol{\theta}^{(k)}$ is the search direction. In order to guarantee that $\boldsymbol{\theta}^{(k+1)} \geq 0$, we restrict $\omega^{(k)} \in (0, 1]$. The line search step size $\omega^{(k)}$ is computed using the Armijo's rule (Luenberger, 2003). Following similar arguments as in Ma et al. (2014), we can show that, under certain regular conditions, (i) the Newton-MI algorithm converges, and (ii) it converges to the solution satisfying the KKT conditions (3.18) and (3.19).

3.4 Selection of the smoothing parameter

Using suitable values of the smoothing parameter $\tilde{\gamma}$ is crucial in providing good balance between the fit to the data and the smoothness of the MPL baseline hazard estimates. Although we can select these values empirically, automatic determination of them by data is less subjective.

The automatic selection of $\tilde{\gamma}$ can be implemented by the cross-validation method (Silverman, 1984). Let $\boldsymbol{\eta} = [\boldsymbol{\beta}^T, \boldsymbol{\theta}^T]^T$. The standard cross-validation score (CVS), which is minimized to obtain the optimal smoothing value, is

$$\text{CVS}(\tilde{\gamma}) = -\sum_{i=1}^{n} \ell_i(\hat{\boldsymbol{\eta}}_{\tilde{\gamma}}^{(-i)}),$$

where $\hat{\boldsymbol{\eta}}_{\tilde{\gamma}}^{(-i)}$ is the MPL estimator of $\boldsymbol{\eta}$ computed with $\tilde{\gamma}$ using the sample in which the subject i is removed, and $\ell_i(\cdot)$ is the log-likelihood for the subject i. However, minimizing $\text{CVS}(\tilde{\gamma})$ involves a great deal of computation, since it performs a minimization for each subject i and different values of $\tilde{\gamma}$. O'Sullivan (1988) suggests a one-step Newton-Raphson expansion for approximations, and the resulting approximate CVS is given by

$$\text{CVS}(\tilde{\gamma}) = -\ell(\hat{\boldsymbol{\eta}}_{\tilde{\gamma}}) + \text{trace}([\Phi''(\hat{\boldsymbol{\eta}}_{\tilde{\gamma}})]^{-1}\ell''(\hat{\boldsymbol{\eta}}_{\tilde{\gamma}})), \tag{3.30}$$

where $\Phi''(\boldsymbol{\eta})$ and $\ell''(\boldsymbol{\eta})$ are respectively the second derivatives of $\Phi(\boldsymbol{\eta})$ and $\ell(\boldsymbol{\eta})$ with respect to $\boldsymbol{\eta}$, and $\hat{\boldsymbol{\eta}}_{\tilde{\gamma}}$ is the MPL estimate of $\boldsymbol{\eta}$ calculated with $\tilde{\gamma}$. The expression (3.30) is indeed equivalent to the AIC criterion if we interpret its second term as the model degrees of freedom (O'Sullivan, 1988). We extend the score (3.30) to our case and select the optimal smoothing value $\tilde{\gamma}_{op}$ according to

$$\tilde{\gamma}_{op} = \arg\min_{\tilde{\gamma}} \{\text{CVS}(\tilde{\gamma})\}. \tag{3.31}$$

We can choose $\tilde{\gamma}_{op}$ by computing $\text{CVS}(\tilde{\gamma})$ for a grid of $\tilde{\gamma}$ values and searching for the one that minimizes $\text{CVS}(\tilde{\gamma})$. The method (3.31) has also been used by other researchers for the baseline hazard estimation, see Joly et al. (1998) for instance.

3.5 Asymptotic properties of the MPL estimators

In this section, we analyze the asymptotic consistency and asymptotic distribution of the MPL estimators. Recall that $\boldsymbol{\eta} = [\boldsymbol{\beta}^T, \boldsymbol{\theta}^T]^T$ is the model parameter vector with dimension

$p+m$. We define $\boldsymbol{\Gamma}$ as the parameter space of $\boldsymbol{\eta}$, i.e., $\boldsymbol{\eta} \in \boldsymbol{\Gamma}$. Let $\hat{\boldsymbol{\eta}} = [\hat{\boldsymbol{\beta}}^T, \hat{\boldsymbol{\theta}}^T]^T$ be the MPL estimator of $\boldsymbol{\eta}$. Let $\boldsymbol{\eta}_0 = [\boldsymbol{\beta}_0^T, \boldsymbol{\theta}_0^T]^T$ be the true value of $\boldsymbol{\eta}$. For convenience in the asymptotic study, we re-express the partly interval-censored data (3.1), arising from n i.i.d subjects, as

$$\{(L_i, R_i], \boldsymbol{X}_i, \xi_i : i = 1, \cdots, n\}, \tag{3.32}$$

where $\xi_i = 1$ if the failure time for subject i is exactly observed so that we have $L_i = R_i = T_i$, and $\xi_i = 0$ if censored. Clearly, for left-censored data, $L_i = 0$ and for right-censored data, $R_i = +\infty$. Let $u(L_i, R_i)$ and $k(\boldsymbol{X}_i)$ denote respectively the joint density function of the censoring bivariate random variable $(L_i, R_i]$ and the density function of $\boldsymbol{X}_i$. We assume both the censoring times and covariates are independent of the failure time T_i. Then, the joint density function of (3.32) is given by

$$g(L_i, R_i, \boldsymbol{X}_i, \xi_i|\boldsymbol{\eta}) = \{[S(L_i|\boldsymbol{\eta}) - S(R_i|\boldsymbol{\eta})]u(L_i, R_i)\}^{1-\xi_i} f(L_i|\boldsymbol{\eta})^{\xi_i} k(\boldsymbol{X}_i), \tag{3.33}$$

where $S(\cdot|\boldsymbol{\eta})$ is the survival function of failure time defined in (3.10), and $f(\cdot|\boldsymbol{\eta})$ is the density function of failure time given by $f(t|\boldsymbol{\eta}) = h(t|\boldsymbol{\eta})S(t|\boldsymbol{\eta})$, where $h(\cdot|\boldsymbol{\eta})$ is defined in (3.8). The joint density function (3.33) can be simplified by

$$g(L_i, R_i, \xi_i|\boldsymbol{\eta}) = [S(L_i|\boldsymbol{\eta}) - S(R_i|\boldsymbol{\eta})]^{1-\xi_i} f(L_i|\boldsymbol{\eta})^{\xi_i}. \tag{3.34}$$

Now we can rewrite the log-likelihood function (3.11) as

$$\ell(\boldsymbol{\eta}) = \sum_{i=1}^{n} \ell_i(\boldsymbol{\eta}) = \sum_{i=1}^{n} \log g(L_i, R_i, \xi_i|\boldsymbol{\eta}).$$

We replace the notation of the penalized log-likelihood function $\Phi(\boldsymbol{\eta})$ in (3.12) by $\Phi_n(\boldsymbol{\eta})$ to reflect its dependence on sample size n, and re-express $\Phi_n(\boldsymbol{\eta})$ as

$$\Phi_n(\boldsymbol{\eta}) = \sum_{i=1}^{n} \{\ell_i(\boldsymbol{\eta}) - \tilde{\gamma}_n J(\boldsymbol{\eta})\},$$

where $J(\boldsymbol{\eta}) = J(\boldsymbol{\theta})$ and $\tilde{\gamma}_n = \tilde{\gamma}/n$.

In the analysis of asymptotic properties of $\hat{\boldsymbol{\eta}}$, we fix the number of bins m, although potentially large, and assume $\tilde{\gamma}_n$ is fixed at a value independent of sample size n. To develop the asymptotic properties, additional assumptions are required and stated below.

Assumptions:

3.1 The parameter space $\boldsymbol{\Gamma}$ of $\boldsymbol{\eta}$ is compact.

3.2 There exists a measurable function $m(L_i, R_i, \boldsymbol{X}_i, \xi_i)$ with $E[m(L_i, R_i, \boldsymbol{X}_i, \xi_i)] < \infty$, which satisfies $|\log g(L_i, R_i, \xi_i|\boldsymbol{\eta})| \leq m(L_i, R_i, \boldsymbol{X}_i, \xi_i)$ for all $\boldsymbol{\eta} \in \boldsymbol{\Gamma}$.

3.3 $E_{\boldsymbol{\eta}_0}[n^{-1}\Phi_n(\boldsymbol{\eta})]$ exists and has a unique maximum at $\boldsymbol{\eta}^* \in \text{int}(\boldsymbol{\Gamma})$, where $\boldsymbol{\eta}^*$ is not necessarily equal to $\boldsymbol{\eta}_0$ due to the penalty term.

3.4 $\Phi_n(\boldsymbol{\eta})$ is continuous over $\boldsymbol{\Gamma}$ and is twice differentiable in a neighborhood of $\boldsymbol{\eta}^*$. Also, the matrices

$$G_0(\boldsymbol{\eta}) = E_{\boldsymbol{\eta}_0}\left[n^{-1}\frac{\partial\Phi_n(\boldsymbol{\eta})}{\partial\boldsymbol{\eta}}\frac{\partial\Phi_n(\boldsymbol{\eta})}{\partial\boldsymbol{\eta}^T}\right]$$

and

$$F_0(\boldsymbol{\eta}) = -E_{\boldsymbol{\eta}_0}\left[n^{-1}\frac{\partial^2\Phi_n(\boldsymbol{\eta})}{\partial\boldsymbol{\eta}\partial\boldsymbol{\eta}^T}\right]$$

exist and are positive definite in a neighborhood of $\boldsymbol{\eta}^*$.

3.5 The penalty function $J(\boldsymbol{\eta})$ is continuous and bounded over $\boldsymbol{\Gamma}$. Both $\partial J(\boldsymbol{\eta})/\partial\boldsymbol{\eta}$ and $\partial^2 J(\boldsymbol{\eta})/\partial\boldsymbol{\eta}\partial\boldsymbol{\eta}^T$ exist for all $\boldsymbol{\eta} \in \boldsymbol{\Gamma}$, and $\partial^2 J(\boldsymbol{\eta})/\partial\boldsymbol{\eta}\partial\boldsymbol{\eta}^T$ is bounded in a neighborhood of $\boldsymbol{\eta}^*$.

The asymptotic results are presented in the following theorem.

Theorem 3.1. *Suppose that Assumptions 3.1-3.5 are satisfied. Then, when $n \to \infty$, the MPL estimator $\hat{\boldsymbol{\eta}}$ is consistent for $\boldsymbol{\eta}^*$ and*

$$\sqrt{n}(\hat{\boldsymbol{\eta}} - \boldsymbol{\eta}^*) \xrightarrow{\mathcal{D}} N(\mathbf{0}_{(p+m)\times 1}, V(\boldsymbol{\eta}^*)),$$

where

$$V(\boldsymbol{\eta}^*) = F_0(\boldsymbol{\eta}^*)^{-1} G_0(\boldsymbol{\eta}^*) F_0[(\boldsymbol{\eta}^*)^{-1}]^T \tag{3.35}$$

Proof. Detail of the proof is omitted here, since it is a simple modification of that in Yu and Ruppert (2002) for penalized spline estimation in partially linear single-index models. □

Note that if we let $\tilde{\gamma}_n \to 0$ as $n \to \infty$, with Assumption 3.5, the penalty term in the penalized log-likelihood will disappear in proving asymptotic normality of the MPL estimators, and the analysis will follow the same ways with those in studying the maximum likelihood estimator (MLE). In practice, $\boldsymbol{\eta}^*$ is generally unavailable, but we can replace it by $\hat{\boldsymbol{\eta}}$ due to the consistency result. The variance formula (3.35) for the asymptotic normal distribution is called the sandwich formula, and can be estimated by replacing $F_0(\boldsymbol{\eta}^*)$ and $G_0(\boldsymbol{\eta}^*)$ by their empirical versions, with $\boldsymbol{\eta}^*$ replaced by $\hat{\boldsymbol{\eta}}$. The asymptotic confidence interval for each element of the true parameter vector $\boldsymbol{\eta}_0$ is calculated based on a finite-size sample. Specifically, the approximate 95% asymptotic confidence interval of β_j is constructed by

$$\hat{\beta}_j \pm z_{0.025} \sqrt{\hat{\mathrm{Var}}(\hat{\beta}_j)}, \quad j = 1, \cdots, p, \tag{3.36}$$

where $z_{0.025} = 1.96$ and $\hat{\mathrm{Var}}(\hat{\beta}_j)$ is obtained from the estimate of the asymptotic variance (3.35). By defining a vector $\boldsymbol{A}(t) = [I(\tau_0 < t \leq \tau_1), \cdots, I(\tau_{m-1} < t \leq \tau_m)]^T$, the baseline hazard (3.6) can be rewritten in a matrix form as $h_0(t) = \boldsymbol{A}(t)^T \boldsymbol{\theta}$. Then we have $\hat{h}_0(t) = \boldsymbol{A}(t)^T \hat{\boldsymbol{\theta}}$ and $h_0^*(t) = \boldsymbol{A}(t)^T \boldsymbol{\theta}^*$. Define a function $g_t(\boldsymbol{\theta})$ by $g_t(\boldsymbol{\theta}) = \boldsymbol{A}(t)^T \boldsymbol{\theta}$. Then we have $g_t(\hat{\boldsymbol{\theta}}) = \hat{h}_0(t)$ and $g_t(\boldsymbol{\theta}^*) = h_0^*(t)$. Since $g_t(\boldsymbol{\theta})$ is a continuously differentiable function in $\boldsymbol{\theta}$ with $dg_t(\boldsymbol{\theta})/d\boldsymbol{\theta} = \boldsymbol{A}_t$, according to Theorem 3.3 and the Delta Theorem, at time t, we have

$$\sqrt{n}(\hat{h}_0(t) - h_0^*(t)) \xrightarrow{\mathcal{D}} \mathrm{N}(0, \boldsymbol{A}(t)^T \mathrm{Var}(\boldsymbol{\theta}^*) \boldsymbol{A}(t)), \tag{3.37}$$

where $\mathrm{Var}(\boldsymbol{\theta}^*)$ is obtained from the asymptotic covariance matrix (3.35), and its estimate is obtained from the estimate of the asymptotic covariance matrix (3.35). Hence the approximate 95% asymptotic confidence interval for $h_0(t)$ at time t is given by

$$\hat{h}_0(t) \pm z_{0.025}\sqrt{\boldsymbol{A}(t)^T\hat{\mathrm{Var}}(\hat{\boldsymbol{\theta}})\boldsymbol{A}(t)}, \tag{3.38}$$

where $z_{0.025} = 1.96$.

Analyzing the asymptotic variance of $\hat{\boldsymbol{\eta}}$ requires that $\boldsymbol{\eta}$ is interior to its parameter space $\boldsymbol{\Gamma}$, i.e., $\theta_u > 0$ for all u. However, it is possible that some elements of $\boldsymbol{\theta}$ take zero values, causing difficulties to develop asymptotic properties for $\hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\theta}}$. In this case, the bootstrapping method can be applied to approximate the mean and variance. However, this can be computationally intensive, since large numbers of estimations of the parameters are required. The simulation study in Section 3.6 indicates that the sandwich formula (3.35) for the asymptotic variance is generally accurate.

Note that the second derivative of the log-likelihood function with respective to $\boldsymbol{\eta}$ is given by

$$\frac{\partial^2\ell(\boldsymbol{\eta})}{\partial\boldsymbol{\eta}\partial\boldsymbol{\eta}^T} = \begin{bmatrix} \frac{\partial^2\ell(\boldsymbol{\beta},\boldsymbol{\theta})}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^T} & \frac{\partial^2\ell(\boldsymbol{\beta},\boldsymbol{\theta})}{\partial\boldsymbol{\beta}\partial\boldsymbol{\theta}^T} \\ \frac{\partial^2\ell(\boldsymbol{\beta},\boldsymbol{\theta})}{\partial\boldsymbol{\theta}\partial\boldsymbol{\beta}^T} & \frac{\partial^2\ell(\boldsymbol{\beta},\boldsymbol{\theta})}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^T} \end{bmatrix},$$

where,

$$\begin{aligned}
\frac{\partial^2\ell(\boldsymbol{\eta})}{\partial\theta_u^2} &= -\sum_{i\in\mathcal{L}_u}\left[\frac{(a_{iL_u})^2S(t_{iL_u})}{(1-S(t_{iL_u}))^2}\right] - \delta_u^2\sum_{w>u}\sum_{i\in\mathcal{L}_w}\left[\frac{S(t_{iL_w})(\exp(\boldsymbol{X}_{iL_w}^T\boldsymbol{\beta}))^2}{(1-S(t_{iL_w}))^2}\right] \\
&\quad -\sum_{i\in\mathcal{IL}_u}\sum_{i\in\mathcal{IR}_u}\left[\frac{(a_{iIL_u}-a_{iIR_u})^2S(t_{iIL_u})S(t_{iIR_u})}{(S(t_{iIL_u})-S(t_{iIR_u}))^2}\right] \\
&\quad -\sum_{w>u}\sum_{i\in\mathcal{IL}_u}\sum_{i\in\mathcal{IR}_w}\left[\frac{(a_{iIL_u}-\delta_u\exp(\boldsymbol{X}_{iIR_w}^T\boldsymbol{\beta}))^2S(t_{iIL_u})S(t_{iIR_w})}{(S(t_{iIL_u})-S(t_{iIR_w}))^2}\right] \\
&\quad -\sum_{v<u}\sum_{i\in\mathcal{IL}_v}\sum_{i\in\mathcal{IR}_u}\left[\frac{(a_{iIR_u})^2S(t_{iIL_v})S(t_{iIR_u})}{(S(t_{iIL_v})-S(t_{iIR_u}))^2}\right] \\
&\quad -\delta_u^2\sum_{v<u}\sum_{w>u}\sum_{i\in\mathcal{IL}_v}\sum_{i\in\mathcal{IR}_w}\left[\frac{(\exp(\boldsymbol{X}_{iIR_w}^T\boldsymbol{\beta}))^2S(t_{iIL_v})S(t_{iIR_w})}{(S(t_{iIL_v})-S(t_{iIR_w}))^2}\right] \\
&\quad -\frac{N_u^O}{\theta_u^2},
\end{aligned}$$

for $u > k$, we have

$$\begin{aligned}
\frac{\partial^2 \ell(\boldsymbol{\eta})}{\partial\theta_k \partial\theta_u} &= -\delta_k \sum_{i\in\mathcal{L}_u} \left[\frac{a_{iL_u} \exp(\boldsymbol{X}_{iL_u}^T \boldsymbol{\beta}) S(t_{iL_u})}{(1 - S(t_{iL_u}))^2} \right] - \delta_u \delta_k \sum_{w>u} \sum_{i\in\mathcal{L}_w} \left[\frac{(\exp(\boldsymbol{X}_{iL_w}^T \boldsymbol{\beta}))^2 S(t_{iL_w})}{(1 - S(t_{iL_w}))^2} \right] \\
&\quad -\delta_k \sum_{v<k} \sum_{i\in\mathcal{IL}_v} \sum_{i\in\mathcal{IR}_u} \left[\frac{a_{iIR_u} \exp(\boldsymbol{X}_{iIR_u}^T \boldsymbol{\beta}) S(t_{iIL_v}) S(t_{iIR_u})}{(S(t_{iIL_v}) - S(t_{iIR_u}))^2} \right] \\
&\quad - \sum_{i\in\mathcal{IL}_k} \sum_{i\in\mathcal{IR}_u} \left[\frac{a_{iIR_u} (\delta_k \exp(\boldsymbol{X}_{iIR_u}^T \boldsymbol{\beta}) - a_{iIL_k}) S(t_{iIL_k}) S(t_{iIR_u})}{(S(t_{iIL_k}) - S(t_{iIR_u}))^2} \right] \\
&\quad -\delta_u \delta_k \sum_{v<k} \sum_{w>u} \sum_{i\in\mathcal{IL}_v} \sum_{i\in\mathcal{IR}_w} \left[\frac{(\exp(\boldsymbol{X}_{iIR_w}^T \boldsymbol{\beta}))^2 S(t_{iIL_v}) S(t_{iIR_w})}{(S(t_{iIL_v}) - S(t_{iIR_w}))^2} \right] \\
&\quad -\delta_u \sum_{w>u} \sum_{i\in\mathcal{IL}_k} \sum_{i\in\mathcal{IR}_w} \left[\frac{\exp(\boldsymbol{X}_{iIR_w}^T \boldsymbol{\beta}) (\delta_k \exp(\boldsymbol{X}_{iIR_w}^T \boldsymbol{\beta}) - a_{iIL_k}) S(t_{iIL_k}) S(t_{iIR_w})}{(S(t_{iIL_k}) - S(t_{iIR_w}))^2} \right]
\end{aligned}$$

and $\partial^2 \ell(\boldsymbol{\eta}) / \partial\theta_u \partial\theta_k = \partial^2 \ell(\boldsymbol{\eta}) / \partial\theta_k \partial\theta_u$. We define

$$b_{iv_u} = S(t_{iv_u}) \exp(\boldsymbol{X}_{iv_u}^T \boldsymbol{\beta}),$$

then we have

$$
\begin{aligned}
&\frac{\partial^2 \ell(\boldsymbol{\eta})}{\partial \boldsymbol{\beta} \partial \theta_u} \\
= &\sum_{i \in \mathcal{L}_u} \left[\frac{-[H(t_{iL_u}) + S(t_{iL_u}) - 1] a_{iL_u} S(t_{iL_u})}{(1 - S(t_{iL_u}))^2} \boldsymbol{X}_{iL_u} \right] \\
+ &\delta_u \sum_{w>u} \sum_{i \in \mathcal{L}_w} \left[\frac{-[H(t_{iL_w}) + S(t_{iL_w}) - 1] b_{iL_w}}{(1 - S(t_{iL_w}))^2} \boldsymbol{X}_{iL_w} \right] \\
+ &\sum_{v<u} \sum_{i \in \mathcal{IL}_v} \sum_{i \in \mathcal{IR}_u} \left[\frac{a_{iIR_u} S(t_{iIR_u}) [S(t_{iIL_v}) - S(t_{iIR_u}) - S(t_{iIL_v})(H(t_{iIR_u}) - H(t_{iIL_v}))]}{(S(t_{iIL_v}) - S(t_{iIR_u}))^2} \boldsymbol{X}_{iIR_u} \right] \\
+ &\delta_u \sum_{v<u} \sum_{w>u} \sum_{i \in \mathcal{IL}_v} \sum_{i \in \mathcal{IR}_w} \left[\frac{b_{iIR_w} [S(t_{iIL_v}) - S(t_{iIR_w}) - S(t_{iIL_v})(H(t_{iIR_w}) - H(t_{iIL_v}))]}{(S(t_{iIL_v}) - S(t_{iIR_w}))^2} \boldsymbol{X}_{iIR_w} \right] \\
+ &\sum_{i \in \mathcal{IL}_u} \sum_{i \in \mathcal{IR}_u} \left[\frac{a_{iIR_u} S(t_{iIR_u}) - a_{iIL_u} S(t_{iIL_u})}{S(t_{iIL_u}) - S(t_{iIR_u})} \boldsymbol{X}_{iIR_u} \right] \\
+ &\sum_{i \in \mathcal{IL}_u} \sum_{i \in \mathcal{IR}_u} \left[\frac{[H(t_{iIL_u}) - H(t_{iIR_u})] S(t_{iIL_u}) S(t_{iIR_u}) (a_{iIR_u} - a_{iIL_u})}{(S(t_{iIL_u}) - S(t_{iIR_u}))^2} \boldsymbol{X}_{iIR_u} \right] \\
+ &\sum_{w>u} \sum_{i \in \mathcal{IL}_u} \sum_{i \in \mathcal{IR}_w} \left[\frac{\delta_u \exp(\boldsymbol{X}_{iIR_w}^T \boldsymbol{\beta}) S(t_{iIR_w}) - a_{iIL_u} S(t_{iIL_u})}{S(t_{iIL_u}) - S(t_{iIR_w})} \boldsymbol{X}_{iIR_w} \right] \\
+ &\sum_{w>u} \sum_{i \in \mathcal{IL}_u} \sum_{i \in \mathcal{IR}_w} \left[\frac{[H(t_{iIL_u}) - H(t_{iIR_w})] S(t_{iIL_u}) S(t_{iIR_w}) (\delta_u \exp(\boldsymbol{X}_{iIR_w}^T \boldsymbol{\beta}) - a_{iIL_u})}{(S(t_{iIL_u}) - S(t_{iIR_w}))^2} \boldsymbol{X}_{iIR_w} \right] \\
- &\delta_u \sum_{w \geq v > u} \sum_{i \in \mathcal{IL}_v} \sum_{i \in \mathcal{IR}_w} \exp(\boldsymbol{X}_{iIR_w}^T \boldsymbol{\beta}) \boldsymbol{X}_{iIR_w} - \sum_{i \in \mathcal{R}_u} a_{iR_u} \boldsymbol{X}_{iR_u} \\
- &\sum_{i \in \mathcal{O}_u} a_{iO_u} \boldsymbol{X}_{iO_u} - \delta_u \sum_{w>u} \left[\sum_{i \in \mathcal{R}_w} \exp(\boldsymbol{X}_{iR_w}^T \boldsymbol{\beta}) \boldsymbol{X}_{iR_w} + \sum_{i \in \mathcal{O}_w} \exp(\boldsymbol{X}_{iO_w}^T \boldsymbol{\beta}) \boldsymbol{X}_{iO_w} \right].
\end{aligned}
$$

These expressions are used to estimate the asymptotic variance.

3.6 Simulation studies

In this section, a simulation study is conducted to evaluate the MPL method in fitting the PH model with partly interval-censored data. All computations are done by using the R language and the R code is provided in Appendix A. The objectives of the simulation study are

1. to investigate effects of the censoring proportion and sample size on the MPL estimators of regression coefficient $\boldsymbol{\beta}$ and baseline hazard $h_0(\cdot)$,

2. to compare the asymptotic standard deviations with the Monte Carlo standard deviations of the MPL estimators, and

3. to compare our proposed MPL method with the ML method by Pan (1999) which has been reviewed in Section 2.1.3. The ML method is available in CRAN.

The first objective attempts to analyze the sensitivity of the MPL estimators of $\boldsymbol{\beta}$ and $h_0(\cdot)$ to the censoring proportion π_c and the sample size n. We use $n = 100$, 500 and 1000 as small, intermediate and large sample sizes respectively, with approximate censoring proportions of $\pi_c = 20\%$, 50% and 80% for each value of the sample sizes. Results are presented in Tables 3.1-3.3 and Figures 3.1-3.3.

For the second objective of the simulation study, we investigate whether the asymptotic standard deviations computed by the sandwich formula (3.35) are accurate for the MPL estimators, and this is achieved by comparing them with the Monte Carlo standard deviations. The results are reported in Tables 3.1-3.3, and Figures 3.1-3.3.

For the third objective, we demonstrate improvements of our MPL method in estimating $\boldsymbol{\beta}$ and $h_0(\cdot)$ compared with the ML method (Pan, 1999). Results can be viewed in Table 3.4 and Figure 3.4.

We generate a random sample of data, $\{(L_i, R_i], \boldsymbol{X}_i : i = 1, 2, \cdots, n\}$, by the following two steps:

1. For each subject $i = 1, \cdots, n$, the failure time T_i is simulated from the Weibull distribution with the hazard function

$$h(t_i) = 3t_i^2 \exp\{X_{i1} - 0.3X_{i2} + 0.5X_{i3}\}. \tag{3.39}$$

Model (3.39) is the PH model with the baseline hazard function

$$h_0(t_i) = 3t_i^2. \tag{3.40}$$

We set regression coefficients $\boldsymbol{\beta} = [\beta_1, \beta_2, \beta_3]^T$ to be $\beta_1 = 1$, $\beta_2 = -0.3$ and $\beta_3 = 0.5$. For the covariate vector $\boldsymbol{X}_i = (X_{i1}, X_{i2}, X_{i3})^T$, X_{i1} is generated from a Bernoulli random variable with parameter of 0.5, and X_{i2} and X_{i3} are generated according to $X_{i2} \sim \text{Unif}(0, 3)$ and $X_{i3} \sim \text{Unif}(0, 5)$ respectively.

An inversion method is applied to generate T_i based on the relationship $u_i = F_T(t_i)$, where u_i is a standard uniform random variable and $F_T(t_i)$ is the distribution function of T_i, given by

$$F_T(t_i) = 1 - \exp\bigg[- t_i^3 \exp(\boldsymbol{X}_i^T \boldsymbol{\beta}) \bigg].$$

2. For each subject i, independently of T_i, we generate two monitoring times according to $C_{i1} \sim \text{Unif}(0, 1)$ and $C_{i2} = C_{i1} + \text{Unif}(0, 1)$. Then we generate one standard uniform random variable u_i. If $u_i \geq \pi_c$, the failure time T_i is deemed to be exactly observed and we set $L_i = R_i = T_i$. If $u_i \leq \pi_c$, the failure time is censored and there are three cases: if $T_i \leq C_{i1}$, it is left-censored and we set $L_i = 0$ and $R_i = C_{i1}$; if $C_{i1} < T_i \leq C_{i2}$, it is finite interval-censored and $L_i = C_{i1}$ and $R_i = C_{i2}$ and if $T_i > C_{2i}$, it is right-censored and $L_i = C_{i2}$ and $R_i = +\infty$. We can manipulate the censoring proportion by adjusting π_c.

In discretising the baseline hazard, the bins are selected in such a way that the number of observations in each bin, n_c, is approximately the same. Some preliminary tests have been done to indicate that generally the MPL estimator $\hat{\boldsymbol{\beta}}$ is not very sensitive to the choice of n_c, as long as n_c is not too large and the smoothing value $\tilde{\gamma}$ is appropriate. We set $n_c = 2$ for $n = 100$, $n_c = 5$ for $n = 500$ and $n_c = 8$ for $n = 1000$. We set the smoothing parameter $\tilde{\gamma}$ by $\tilde{\gamma}(\lambda) = \lambda/(1 - \lambda)$, so that we can select $\tilde{\gamma}$ by selecting the tuning parameter λ in the range $[0, 1)$. The value of λ is determined according to the criterion (3.31). The convergence criterion of the Newton-MI algorithm is defined as

the absolute value of differences of both $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ updates between consecutive iterations being less than 10^{-5}. We obtain the MPL estimates when Newton-MI converges or the maximum of 5000 iterations are reached, whichever occurs first.

For each combination of n and π_c, we perform Monte Carlo simulations using $N = 300$ repeated samples, and thus obtain 300 MPL estimates of $\boldsymbol{\beta}$ and $h_0(\cdot)$ by our MPL method. From the 300 estimates of $\boldsymbol{\beta}$, we can compute the average estimate (AEST), the bias (BIAS), the Monte Carlo standard deviation (MCSD), the average asymptotic standard deviation (AASD) and the mean squared error (MSE) of $\hat{\boldsymbol{\beta}}$. Specifically, let $\hat{\beta}_{jk}$ be the estimate of β_j, $j = 1, 2, 3$, from the kth sample, $k = 1, \cdots, N$, then we have

$$\text{AEST}(\hat{\beta}_j) = \sum_{k=1}^{N} \hat{\beta}_{jk}/N, \tag{3.41}$$

$$\text{BIAS}(\hat{\beta}_j) = \beta_j - \text{AEST}(\hat{\beta}_j), \tag{3.42}$$

$$\text{MCSD}(\hat{\beta}_j) = \sqrt{\frac{1}{N-1}\sum_{k=1}^{N}(\hat{\beta}_{jk} - \text{AEST}(\hat{\beta}_j))^2}, \tag{3.43}$$

$$\text{AASD}(\hat{\beta}_j) = \sum_{k=1}^{N} \text{ASD}(\hat{\beta}_{jk})/N, \tag{3.44}$$

and

$$\text{MSE}(\hat{\beta}_j) = [\text{BIAS}(\hat{\beta}_j)]^2 + [\text{MCSD}(\hat{\beta}_j)]^2, \tag{3.45}$$

where $\text{ASD}(\hat{\beta}_{jk})$ is the asymptotic standard deviation of the estimator of β_{jk} obtained by the sandwich formula (3.35).

Let $\hat{h}_{0k}(t)$ be the MPL estimate of the true baseline hazard $h_0(t)$ at time t in the kth sample, $k = 1, \cdots, N$. From the $N = 300$ estimates of $h_0(t)$, we calculate the AEST, MCSD and AASD of $\hat{h}_0(t)$ by

$$\text{AEST}(\hat{h}_0(t)) = \sum_{k=1}^{N} \hat{h}_{0k}(t)/N, \tag{3.46}$$

$$\text{MCSD}(\hat{h}_0(t)) = \sqrt{\frac{1}{N-1}\sum_{k=1}^{N}(\hat{h}_{0k}(t) - \text{AEST}(\hat{h}_0(t)))^2}, \tag{3.47}$$

and

$$\text{AASD}(\hat{h}_0(t)) = \sum_{k=1}^{N} \text{ASD}(\hat{h}_{0k}(t))/N, \tag{3.48}$$

where $\text{ASD}(\hat{h}_{0k}(t))$ is the asymptotic standard deviation of the estimator $\hat{h}_{0k}(t)$ at time t and obtained from the formula (3.37). For each sample k, we calculate the distances between the true baseline hazard and the MPL estimate. The distance used is the integrated squared error (ISE) and is given by

$$\text{ISE}_k = \int_J \left[h_0(t) - \hat{h}_{0k}(t)\right]^2 dt, \tag{3.49}$$

where J is a time interval over which the ISE_k is calculated. Hence the average integrated squared error (AISE) of the 300 baseline hazard estimates is

$$\text{AISE} = \sum_{k=1}^{N} \text{ISE}_k/N. \tag{3.50}$$

Tables 3.1-3.3 summarize the AEST, BIAS, MCSD, AASD and MSE values for the MPL estimates of $\boldsymbol{\beta}$ with different censoring proportions and sample sizes. We observe that (i) with a fixed sample size n, the MSE increases with censoring proportion, and the MCSD, AASD and absolute value of BIAS follow the same trend, (ii) with a fixed censoring proportion, all of these four quantities are decreasing as sample size increases, and (iii) comparison between MCSD and AASD demonstrates that the sandwich formula (3.35) is generally accurate in approximating the variance of the MPL estimates of $\boldsymbol{\beta}$, particularly when sample size becomes larger or censoring proportion becomes smaller.

Figures 3.1-3.3 exhibit plots for the true baseline hazard, AEST of the MPL estimates, the corresponding 95% Monte Carlo piecewise confidence intervals (PWCI) and the corresponding mean of the 95% asymptotic piecewise confidence intervals (PWCI). We detect that (i) AESTs are all very close to the true baseline hazard under different sample sizes and censoring proportions, (ii) the 95% Monte Carlo PWCI is close to the mean of the 95% asymptotic PWCI, hence the sandwich formula (3.35) gives a good variance approxi-

mation for the MPL estimates of $\hat{h}_0(t)$, and (iii) both the 95% Monte Carlo PWCI and the mean of the 95% asymptotic PWCI become wider as the censoring proportion increases, but narrower when the sample size increases.

Tables 3.1-3.3 also give AISEs for the MPL estimates of $h_0(t)$ plotted in Figures 3.1-3.3 respectively. It is observed that the AISEs exhibit an increasing trend as the censoring proportion increases, but a decreasing trend as the sample size increases.

To make comparison between our MPL method and the ML method by Pan (1999), we have to set the censoring proportion π_c to be one in our method, since the ML method only considers interval-censored data with no considerations to exactly observed failure time data. The two methods are evaluated based on the same data sets. The simulated data for both methods have sample sizes $n = 100$, 500 and 1000. Monte Carlo simulations are done with 300 repeated samples. The covariates $\boldsymbol{X}_i = [X_{i1}, X_{i2}, X_{i3}]^T$ and regression coefficients $\boldsymbol{\beta} = [\beta_1, \beta_2, \beta_3]^T$ are the same as above. In our MPL method, for the equal count of observations in each bin, we set $n_c = 2$ for $n = 100$, $n_c = 5$ for $n = 500$ and $n_c = 8$ for $n = 1000$. Estimates from the ML method are computed by using the R package, intcox. In this package, asymptotic standard deviations are not available, hence we only compute Monte Carlo standard deviations. Table 3.4 reports estimation results for $\boldsymbol{\beta}$ from the two methods. We observe that, under each of the three sample sizes, the biases of estimates in MPL are smaller than those in ML, although they yield similar standard deviations. Thus MPL achieves lower MSE than ML. Figure 3.4 displays baseline hazard estimations for the two methods. Clearly, the MPL estimate is closer to the true baseline hazard than the ML estimate. Table 3.4 also report values of AISE for the baseline hazard estimate. We observe that our MPL method gives much smaller AISE than ML.

3.7 AIDS example

In this section, we apply our MPL method to fit the PH model using a data set from the AIDS example given by Lindsey and Ryan (1998). This example concerns the study of development of drug resistance (measured using a plaque reduction assay) to zidovudine in patients. The patients are enrolled in four clinical trials for the treatment of AIDS and samples are collected at scheduled visit times dictated by the four protocols. This data has very wide intervals and a high proportion of right censoring, since there were few assessments on each patient due to the high cost of the resistance assays. There are four covariates that may have an effect on the time to development of resistance: stage of disease, dose of zidovudine and CD4 lymphocyte counts at time of randomization (CD4: 100-399 and CD4: ≥ 400). We define T_i to be the time to development of resistance for patient i, $i = 1, \cdots, 31$, and also define X_{1i} for the covariate variable of stage of disease, X_{2i} for dose of zidovudine, X_{3i} for CD4: 100-399 and X_{4i} for CD4: ≥ 400. For this data set, we assume that the observation times are independent of T_i's, and assume the T_i's follow the PH model (3.2). In fitting the PH model to this data set, we choose the number of observations in each bin as $n_c = 3$, and select the smoothing value $\lambda = 1 - 10^{-4}$. Based on the consistency and asymptotic normality properties of $\hat{\boldsymbol{\beta}}$, we perform a hypothesis test of the null hypothesis, H_0: $\beta_j = 0$ versus alternative hypothesis, H_a: $\beta_j \neq 0$, $j = 1, 2, 3, 4$. We use a z-test. Results are summarized in Table 3.5 and Figure 3.5.

Table 3.5 summarizes the MPL estimates of regression coefficients $\boldsymbol{\beta}$, asymptotic standard deviations, the p-values and 95% confidence intervals. We conclude that none of the covariates has significant effects on the time to develop resistance. Figure 3.5 displays the MPL estimates of the baseline hazard with its 95% PWCI, and the MPL estimate of the baseline survival function. We observe that the estimated baseline hazard rises sharply at month 2, and then is monotonically decreasing afterwords.

3.8 Conclusion

In this chapter, we have developed a MPL method to fit the semi-parametric PH model with partly interval-censored data, where a quadratic penalty term is added to the log-likelihood function to assure smoothness of the estimated baseline hazard. To obtain MPL estimators of the regression coefficients and baseline hazard, we use the Newton-MI algorithm which combines the Newton algorithm and the MI algorithm. The Newton-MI algorithm is easily implemented and successful in imposing the non-negativity constraint on the estimated baseline hazard. In addition to piecewise constant approximation to the baseline hazard in our method, the Newton-MI algorithm can also be easily applied in other approximation approaches, such as spline or kernel. The simulation study demonstrates that our MPL method works well, and the sandwich formula given in the asymptotic analysis provides accurate variance approximations for both the regression coefficients and baseline hazard.

In developing the MPL method, we assume the covariate variables are time independent and the censoring time is independent of the failure time. However, we can extend our method to fit the PH model with time-dependent covariates and dependent censoring. In addition to the PH model, our MPL approach can also be adapted to fit other semi-parametric regression models, such as additive hazard model and accelerated failure time model, which will be studied in depth in the next two chapters.

		n=100		
π_c		20%	50%	80%
n_c		2	2	2
$\beta_1 = 1$	AEST	0.9695	0.9710	0.9425
	BIAS	0.0305	0.0290	0.0575
	MCSD	0.2444	0.2919	0.2959
	AASD	0.2385	0.2860	0.2973
	MSE	(0.0607)	(0.0861)	(0.0909)
$\beta_1 = -0.3$	AEST	-0.3012	-0.3160	-0.2989
	BIAS	0.0012	0.0160	-0.0011
	MCSD	0.1366	0.1376	0.1442
	AASD	0.1312	0.1461	0.1570
	MSE	(0.0187)	(0.0192)	(0.0208)
$\beta_2 = 0.5$	AEST	0.5012	0.5086	0.4830
	BIAS	-0.0012	-0.0086	0.0170
	MCSD	0.0789	0.1095	0.1148
	AASD	0.0799	0.0949	0.1104
	MSE	(0.0062)	(0.0121)	(0.0135)
$h_0(t)$	AISE	0.2861	0.3964	0.4386

Table 3.1: AEST, BIAS, MCSD, AASD and MSE for the estimate $\hat{\boldsymbol{\beta}}$, and average integrated squared error (AISE) for the baseline hazard estimate $\hat{h}_0(t)$ with equal count in each bin $n_c = 2$, sample size $n = 100$ and censoring proportions $\pi_c = 20\%$, 50% and 80%.

		n=500		
π_c		20%	50%	80%
n_c		5	5	5
$\beta_1 = 1$	AEST	1.0057	1.0211	0.9762
	BIAS	-0.0057	-0.0211	0.0238
	MCSD	0.1141	0.1186	0.1256
	AASD	0.1060	0.1126	0.1301
	MSE	(0.0131)	(0.0145)	(0.0163)
$\beta_1 = -0.3$	AEST	-0.2989	-0.3052	-0.3058
	BIAS	-0.0011	0.0052	-0.0058
	MCSD	0.0618	0.0682	0.0690
	AASD	0.0553	0.0627	0.0730
	MSE	(0.0038)	(0.0047)	(0.0048)
$\beta_2 = 0.5$	AEST	0.5042	0.5048	0.4861
	BIAS	-0.0042	-0.0048	0.0139
	MCSD	0.0353	0.0401	0.0459
	AASD	0.0340	0.0424	0.0477
	MSE	(0.0013)	(0.0016)	(0.0023)
$h_0(t)$	AISE	0.1428	0.231	0.2671

Table 3.2: AEST, BIAS, MCSD, AASD and MSE for the estimate $\hat{\boldsymbol{\beta}}$, and average integrated squared error (AISE) for the baseline hazard estimate $\hat{h}_0(t)$ with equal count in each bin $n_c = 5$, sample size $n = 500$ and censoring proportions $\pi_c = 20\%$, 50% and 80%.

		n=1000		
π_c		20%	50%	80%
n_c		8	8	8
$\beta_1 = 1$	AEST	0.9943	0.9941	0.9899
	BIAS	0.0057	0.0059	0.0101
	MCSD	0.0619	0.0730	0.0821
	AASD	0.0686	0.0783	0.0887
	MSE	(0.0039)	(0.0054)	(0.0068)
$\beta_1 = -0.3$	AEST	-0.3034	-0.3050	-0.2954
	BIAS	0.0034	0.0050	-0.0046
	MCSD	0.0380	0.0423	0.0534
	AASD	0.0396	0.0435	0.0578
	MSE	(0.0015)	(0.0018)	(0.0029)
$\beta_2 = 0.5$	AEST	0.5005	0.5029	0.4978
	BIAS	-0.0005	-0.0029	0.0022
	MCSD	0.0242	0.0292	0.0348
	AASD	0.0242	0.0280	0.0376
	MSE	(0.0006)	(0.0009)	(0.0012)
$h_0(t)$	AISE	0.0498	0.0800	0.1004

Table 3.3: AEST, BIAS, MCSD, AASD and MSE for the estimate $\hat{\boldsymbol{\beta}}$, and average integrated squared error (AISE) for the baseline hazard estimate $\hat{h}_0(t)$ with equal count in each bin $n_c = 8$, sample size $n = 1000$ and censoring proportions $\pi_c = 20\%$, 50% and 80%.

		n=100		n=500		n=1000	
		MPL	ML	MPL	ML	MPL	ML
$\beta_1 = 1$	AEST	0.9735	1.0844	1.0141	0.9467	0.9870	0.8638
	BIAS	0.0265	-0.0844	-0.0141	0.0533	0.0130	0.1362
	MCSD	0.2865	0.3161	0.1980	0.1704	0.0901	0.0760
	AASD	0.3160		0.1901		0.0894	
	MSE	(0.0828)	(0.1070)	(0.0395)	(0.0319)	(0.0083)	(0.0243)
$\beta_2 = -0.3$	AEST	-0.2926	-0.2079	-0.3160	-0.1454	-0.2912	-0.0681
	BIAS	-0.0074	-0.0921	0.0160	-0.1546	-0.0088	-0.2319
	MCSD	0.1931	0.2011	0.0910	0.0890	0.05534	0.0416
	AASD	0.1861		0.1040		0.0503	
	MSE	(0.0373)	(0.0489)	(0.0085)	(0.0318)	(0.0031)	(0.0555)
$\beta_3 = 0.5$	AEST	0.4876	0.5013	0.4966	0.4302	0.4970	0.3867
	BIAS	0.0124	-0.0013	0.0034	0.0698	0.0030	0.1133
	MCSD	0.1078	0.1092	0.0711	0.0612	0.0398	0.0322
	AASD	0.1267		0.0747		0.0401	
	MSE	(0.0118)	(0.0119)	(0.0051)	(0.0086)	(0.0016)	(0.0139)
$h_0(t)$	AISE	0.5887	1.2588	0.3633	1.5292	0.1966	1.6820

Table 3.4: Comparisons of regression coefficient estimates and the baseline hazard estimates between the MPL method and the ML method by Pan (1999) using simulated samples, with sample size of $n = 100$, 500 and 1000.

research/R and matlab/R for PH model Phd/100cp.eps

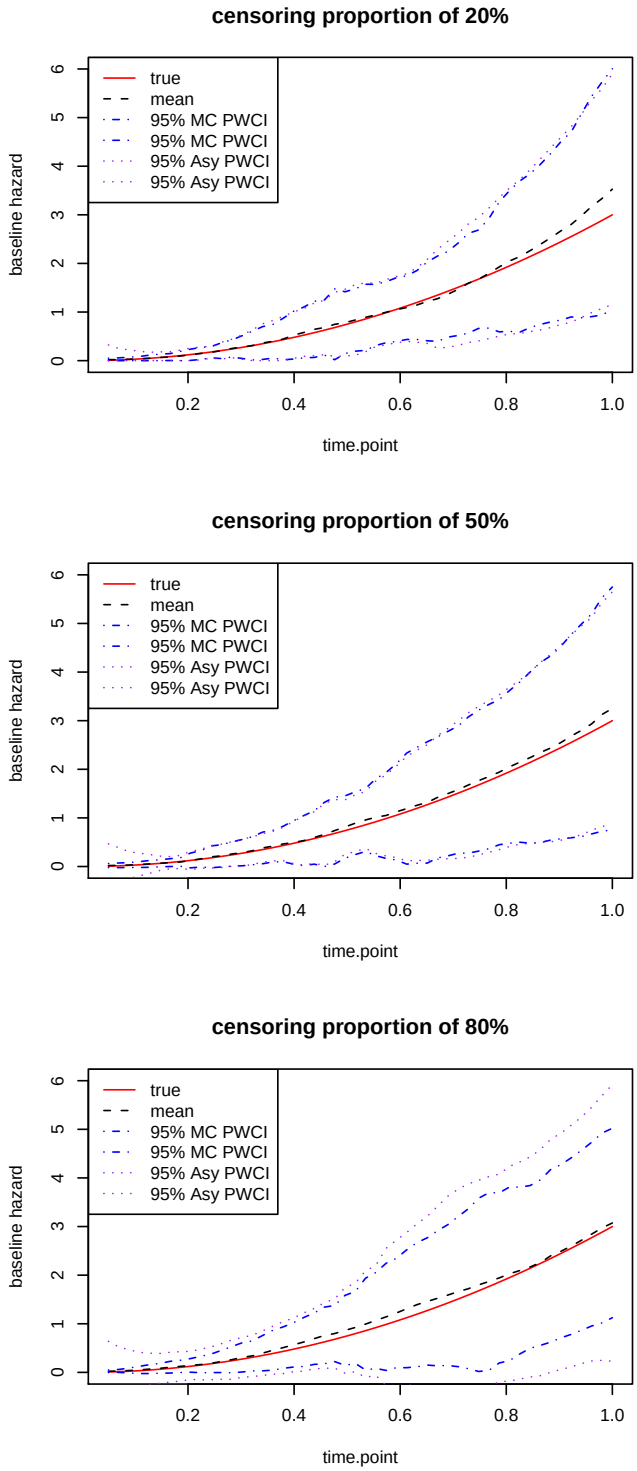

Figure 3.1: Plots of the true baseline hazard $h_0(t)$ (solid), the average MPL estimates of $h_0(t)$ (dash), the 95% Monte Carlo piecewise confidence interval (PWCI) (dot-dash), and the average 95% asymptotic PWCI (dots), assuming sample size $n = 100$, equal count in each bin $n_c = 2$, and censoring proportions of 20%, 50% and 80% corresponding to top, middle and bottom plots respectively.

research/R and matlab/R for PH model Phd/500cp.eps

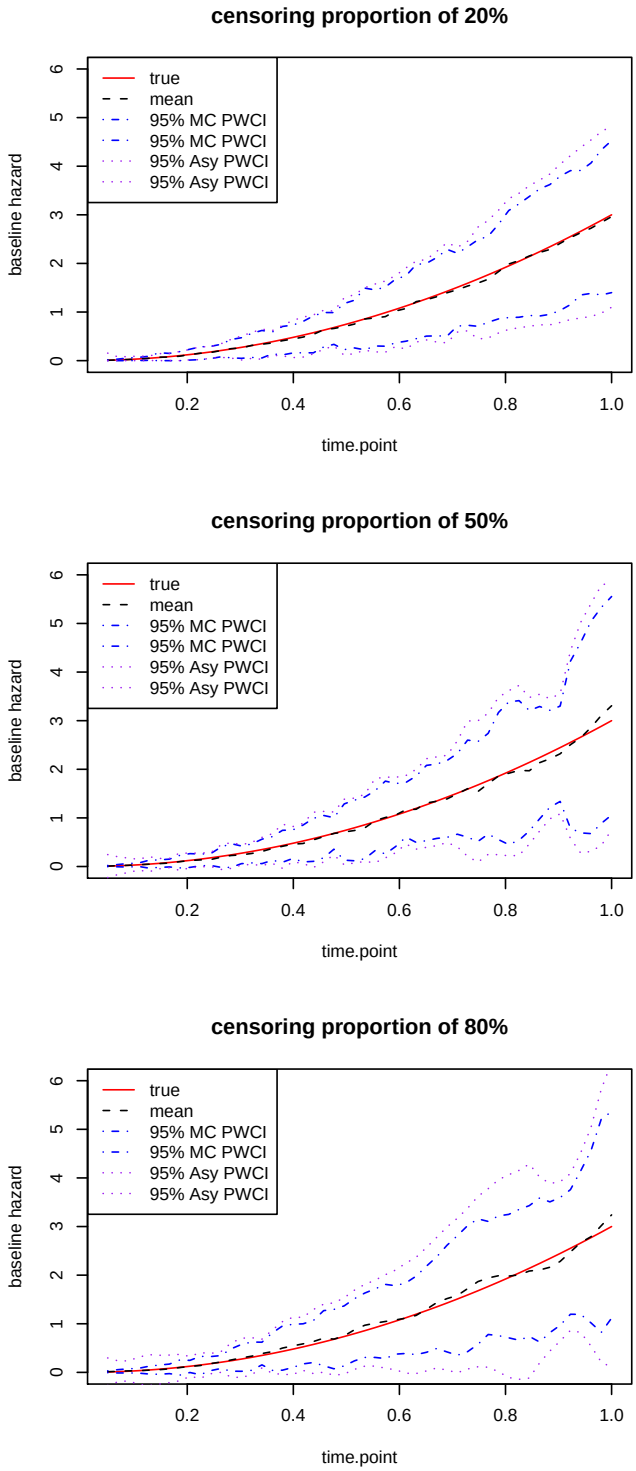

Figure 3.2: Plots of the true baseline hazard $h_0(t)$ (solid), the average MPL estimates of $h_0(t)$ (dash), the 95% Monte Carlo piecewise confidence intervals (PWCI) (dot-dash), and the average 95% asymptotic PWCI (dots), assuming sample size $n = 500$, equal count in each bin $n_c = 5$, and censoring proportions of 20%, 50% and 80% corresponding to top, middle and bottom plots respectively.

research/R and matlab/R for PH model Phd/1000cp.eps

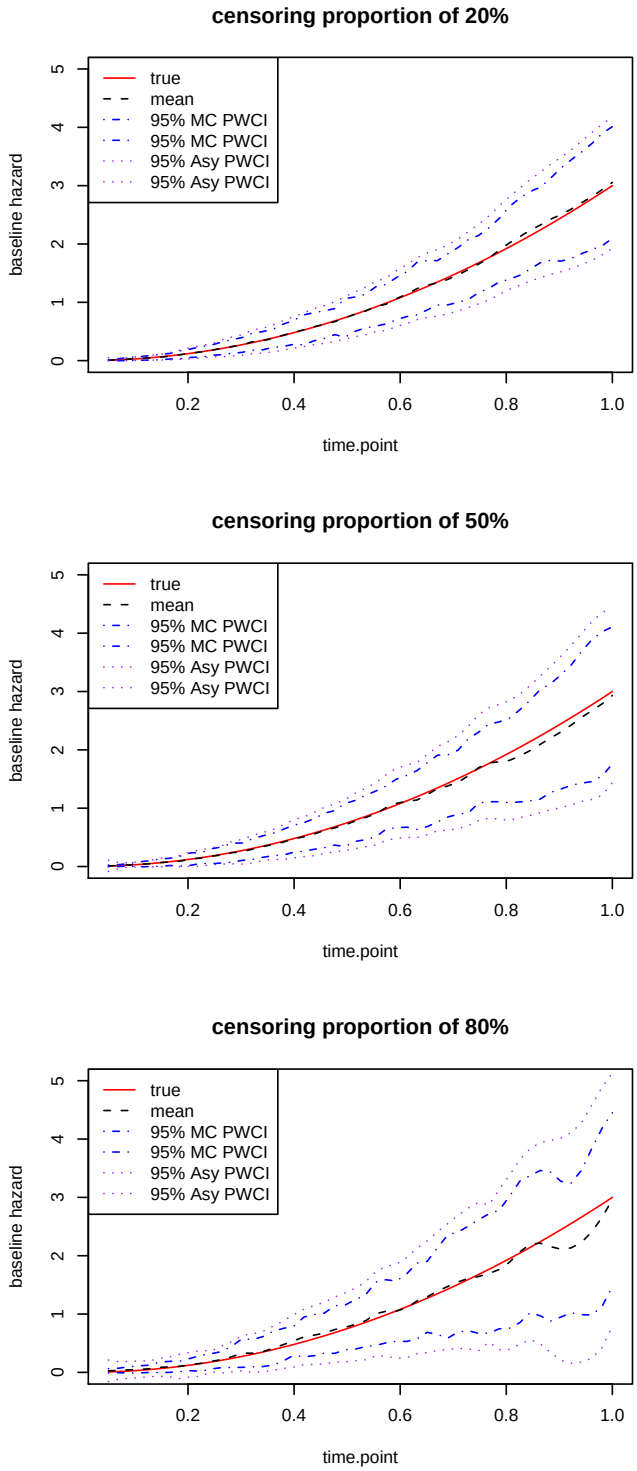

Figure 3.3: Plots of the true baseline hazard $h_0(t)$ (solid), the average MPL estimates of $h_0(t)$ (dash), the 95% Monte Carlo piecewise confidence intervals (PWCI) (dot-dash), and the average 95% asymptotic PWCI (dots), assuming sample size $n = 1000$, equal count in each bin $n_c = 8$, and censoring proportions of 20%, 50% and 80% corresponding to top, middle and bottom plots respectively.

research/R and matlab/R for PH model Phd/mc.eps

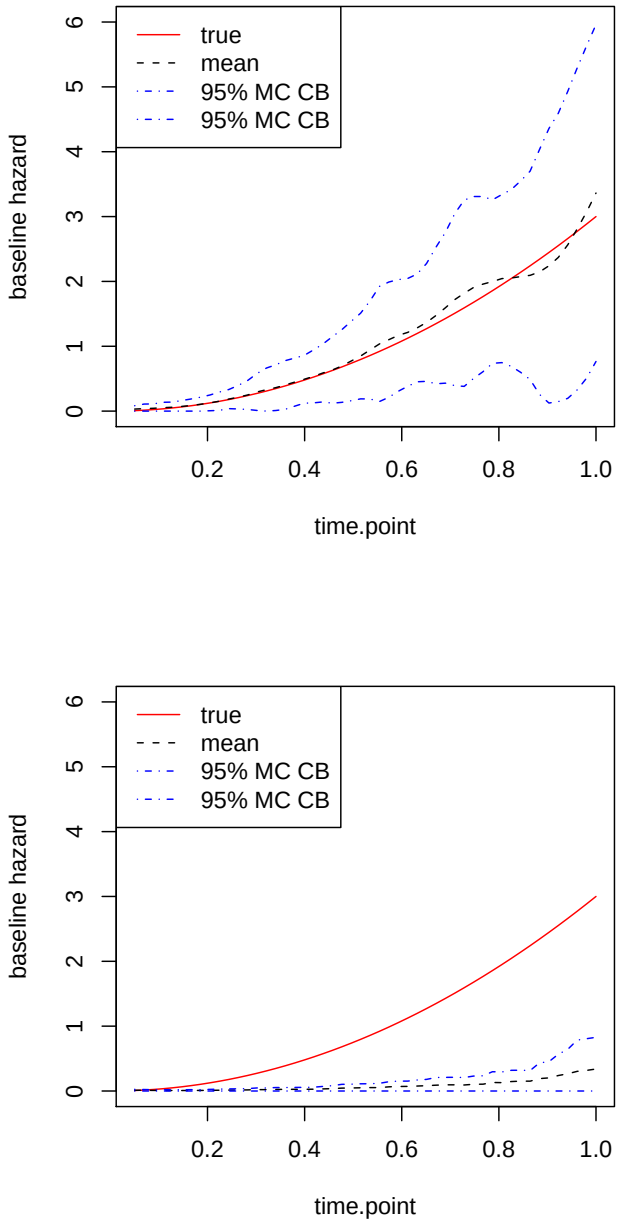

Figure 3.4: Plots of the baseline hazard estimates with its 95% piecewise confidence intervals (PWCI) for the MPL method and the ML method, with sample sizes of $n = 100$, 500 and 1000, and censoring proportion $\pi_c = 1$

research/R and matlab/R for PH model Phd/realdata.eps

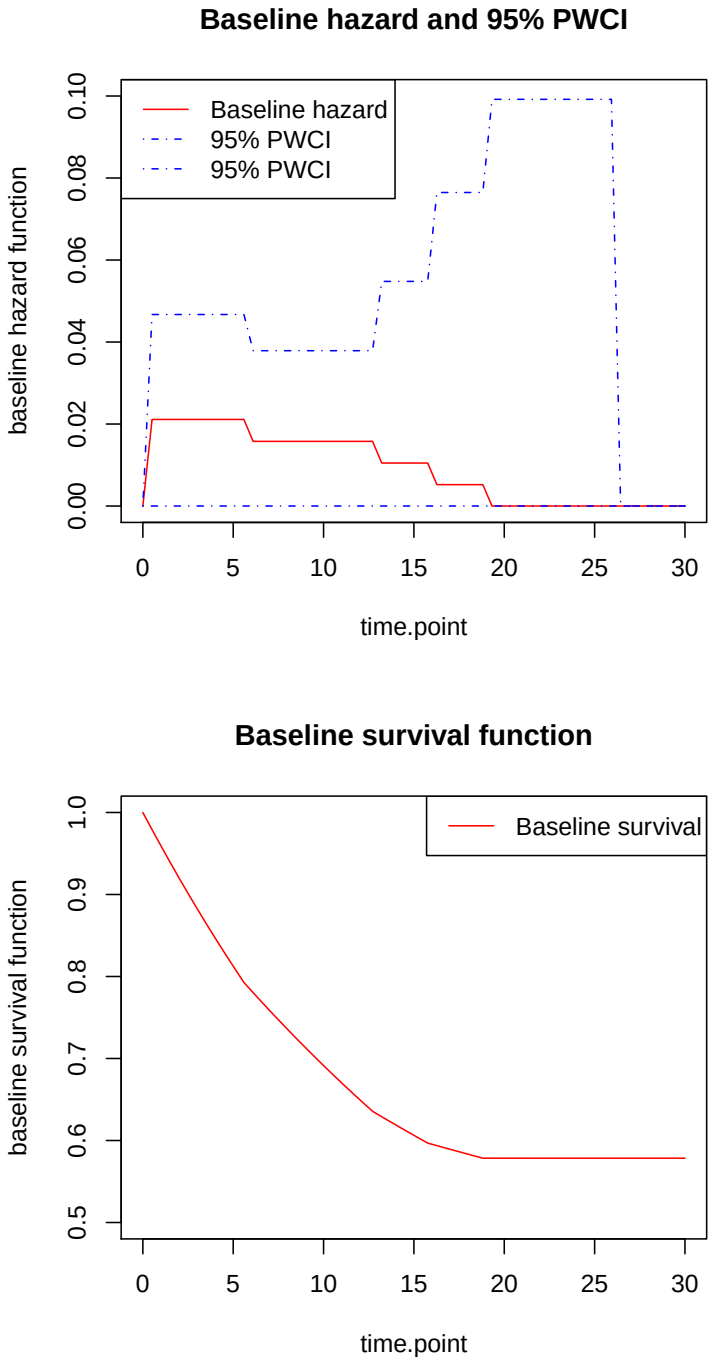

Figure 3.5: Plots of the estimates for the AIDS example, top, the baseline hazard and its 95% PWCI; bottom, the survival function.

Effects	$\hat{\boldsymbol{\beta}}$	astd	p-value	95% C.I
stage	1.1341	0.7665	0.1390	(-0.3682, 2.6364)
dose	0.8662	0.9173	0.3450	(-0.9317, 2.6641)
CD4: 100-399	-0.0356	0.9790	0.9710	(-1.9544, 1.8832)
CD4: ≥ 400	0.1146	1.0571	0.9137	(-1.9573, 2.1865)

Table 3.5: Regression coefficient estimates given by the MPL method with asymptotic standard deviations (astd), p-values and 95% confidence intervals.

Chapter 4

Penalized Likelihood Methods for Additive Hazard Model with Partly Interval-Censored Failure Time Data

4.1 Introduction

In this chapter, we develop a penalized log-likelihood method to fit the additive hazard (AH) model with partly interval-censored failure time data. In contrast to the proportional hazard (PH) model where the covariates are assumed to act multiplicatively on the baseline hazard function, the AH model assumes that the effect of covariates is to additively increase or decrease the hazard function. For subject i, let $h(t|\boldsymbol{X}_i)$ be the hazard function at time t conditional on the covariate vector $\boldsymbol{X}_i$. The AH model specifies that

$$h(t|\boldsymbol{X}_i) = h_0(t) + \boldsymbol{X}_i^T\boldsymbol{\beta}, \tag{4.1}$$

where $\boldsymbol{\beta}$ is a $p \times 1$ regression coefficient vector and $h_0(\cdot)$ is an unspecified baseline hazard function (Kalbfleisch and Prentice, 2002). Since the form of $h_0(\cdot)$ is unknown, the model (4.1) is indeed a semi-parametric regression model. The AH model may be more

appropriate when we concentrate on the absolute change in the hazard, instead of the hazard ratio, or when the proportional hazard assumption for the PH model is violated. In addition, the AH model has the attractive feature in that it provides a simple structure for modeling failure time data when latent variables or frailties exist (Sun, 2006).

The approach to fitting the AH model developed in this chapter differs from that of Wang et al. (2010), Farrington (1996), Ghosh (2001) and Zeng et al. (2006). Wang et al. (2010) develop a counting process estimation approach with interval-censored data, where the baseline hazard is considered to be a nuisance parameter and is not estimated. The method requires the PH model assumption for monitoring time variables. Therefore, we have to test the validity of this assumption before applying it. Zeng et al. (2006) apply a maximum likelihood (ML) method to estimate $\boldsymbol{\beta}$ and the baseline survival function $S_0(\cdot)$ with interval-censored data. The method constrains positivity and monotonic decreasing on $S_0(\cdot)$ by using a logarithm transformation. However, the way the constraints are imposed can make the estimation procedure unstable when the baseline survival estimate approaches zero. The ML approach is also adopted by Ghosh (2001) in studying current status data, where estimation of the cumulative baseline hazard is involved with constraints of monotonic increasing and non-negativity on it. Farrington (1996) develops a generalized linear model (GLM) approach with interval-censored data, where the occurrences of left-, right- and finite interval-censored observations are assumed to be from independent Bernoulli trials, and the occurrence probability is related to a linear predictor by a negative log link function. The regression coefficients and the baseline hazard in the AH model are considered to be covariate coefficients in the linear predictor, and are estimated by fitting this GLM model. However, the resulting estimated baseline hazard is not guaranteed to be smooth.

We fit the AH model by estimating the regression coefficients $\boldsymbol{\beta}$ and the baseline hazard

$h_0(\cdot)$. Our method contributes in three perspectives: (i) we consider all censoring types of data which contain exactly observed, left-, right-, and finite interval-censored failure time data; (ii) our method is developed based on maximizing a penalized log-likelihood function with a penalty function included for smoothness of the estimated baseline hazard, hence the smoothness of the estimated baseline is guaranteed; (iii) and similar to Chapter 3, we model the baseline hazard through assuming it is piecewise constant. In the estimation procedure, we impose the non-negativity constraints both on $h_0(\cdot)$ and the hazard $h(\cdot|\boldsymbol{X}_i)$ in a direct way, and obtain the estimates of $\boldsymbol{\beta}$ and $h_0(\cdot)$ simultaneously by using a primal-dual interior-point algorithm (Wright, 1997). Under certain conditions, we show that the resulting maximum penalized likelihood (MPL) estimators are asymptotically consistent and normally distributed.

This chapter is organized as follows. In Section 4.2, we develop a penalized log-likelihood function with partly interval-censored data under the AH model. In Section 4.3, we analyze the convexity of the negative penalized log-likelihood function. In Section 4.4, we propose the primal-dual interior-point algorithm to compute the MPL estimators, and present its convergence result. Asymptotic results of the MPL estimators are given in Section 4.5. In Section 4.6, we investigate the performance of our proposed method by conducting simulation studies. The MPL method is also illustrated using a set of real data in Section 4.7. Finally, we draw conclusions in Section 4.8.

4.2 Penalized log-likelihood function under the Additive Hazard (AH) model

Using the data formulation defined in (3.1), for each subject i, we express its observed data as $(t_{iO}, \boldsymbol{X}_{iO})$ or $(t_{iL}, \boldsymbol{X}_{iL})$ or $(t_{iIL}, t_{iIR}, \boldsymbol{X}_{iI})$ or $(t_{iR}, \boldsymbol{X}_{iR})$ depending on censoring status

of its failure time. The AH model specifies a hazard function according to $h(t|\boldsymbol{X}_i) = h_o(t) + \boldsymbol{X}_i^T\boldsymbol{\beta}$, where the covariate vector $\boldsymbol{X}_i$ is assumed to be time-independent. The cumulative hazard function is

$$H(t|\boldsymbol{X}_i) = \int_0^t h(s|\boldsymbol{X}_i)ds = H_0(t) + \boldsymbol{X}_i^T\boldsymbol{\beta}t, \tag{4.2}$$

where $H_0(\cdot)$ is the cumulative baseline hazard function. We fit the AH model by estimating the regression coefficient vector $\boldsymbol{\beta}$ and the baseline hazard $h_0(\cdot)$. The baseline hazard $h_0(\cdot)$ belongs to an infinite dimensional parameter space subject to the non-negativity constraint. Similar to Section 3.2, we model $h_0(\cdot)$ using a linear combination of indicator functions, that is

$$h_0(t) = \sum_{u=1}^{m} \theta_u I(\tau_{u-1} < t \leq \tau_u), \tag{4.3}$$

where $I(\cdot)$ is an indicator function. Therefore, estimating the baseline hazard is equivalent to estimating the parameter vector $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_m]^T \in \mathbf{R}_+^m$, where $\mathbf{R}_+^m = \{\boldsymbol{\theta} \in \mathbf{R}^m | \boldsymbol{\theta} \geq \mathbf{0}_{m\times 1}\}$. The corresponding cumulative baseline hazard function is then expressed as

$$H_0(t) = \sum_{u=1}^{m} \theta_u[(t - \tau_{u-1})I(\tau_{u-1} < t \leq \tau_u) + \delta_u I(t > \tau_u)], \tag{4.4}$$

where $\delta_u = t_u - t_{u-1}$ is the width of B_u, where B_u has been defined in (3.5).

Similar to Section 3.5, let $\boldsymbol{\eta} = [\boldsymbol{\beta}^T, \boldsymbol{\theta}^T]^T$ and let $\boldsymbol{\Gamma}$ be the parameter space of $\boldsymbol{\eta}$, i.e., $\boldsymbol{\eta} \in \boldsymbol{\Gamma}$. We define two vectors $\boldsymbol{C}(t, \boldsymbol{X}_i)$ and $\boldsymbol{D}(t, \boldsymbol{X}_i)$ by

$$\boldsymbol{C}(t, \boldsymbol{X}_i) = [x_{i1}, \cdots, x_{ip}, I(\tau_0 < t \leq \tau_1), \cdots, I(\tau_{m-1} < t \leq \tau_m)]^T$$

and

$$\begin{aligned} \boldsymbol{D}(t, \boldsymbol{X}_i) &= [x_{i1}t, \cdots, x_{ip}t, (t - \tau_0)I(\tau_0 < t \leq \tau_1) + \delta_1 I(t > \tau_1), \\ &\quad \cdots, (t - \tau_{m-1})I(\tau_{m-1} < t \leq \tau_m) + \delta_m I(t > \tau_m)]^T. \end{aligned}$$

Using equations (4.3) and (4.4), the hazard, cumulative hazard and survival functions concerning the model parameter $\boldsymbol{\eta}$ are given respectively by

$$h(t|\boldsymbol{\eta}, \boldsymbol{X}_i) = \boldsymbol{C}(t, \boldsymbol{X}_i)^T\boldsymbol{\eta}, \tag{4.5}$$

$$H(t|\boldsymbol{\eta}, \boldsymbol{X}_i) = \boldsymbol{D}(t, \boldsymbol{X}_i)^T \boldsymbol{\eta} \tag{4.6}$$

and

$$S(t|\boldsymbol{\eta}, \boldsymbol{X}_i) = \exp\{-\boldsymbol{D}(t, \boldsymbol{X}_i)^T \boldsymbol{\eta}\}. \tag{4.7}$$

We assume the censoring time is independent of the failure time conditional on the observable covariates, and the distribution of the censoring time does not depend functionally on the regression coefficients. Then using equations (4.5)-(4.7), the log-likelihood function involving $\boldsymbol{\eta}$ is

$$\begin{aligned} \ell(\boldsymbol{\eta}) &= \sum_{i\in\mathcal{L}} \log\Big[1 - S(t_{iL}|\boldsymbol{\eta}, \boldsymbol{X}_{iL})\Big] + \sum_{i\in\mathcal{I}} \log\Big[S(t_{iIL}|\boldsymbol{\eta}, \boldsymbol{X}_{iI}) - S(t_{iIR}|\boldsymbol{\eta}, \boldsymbol{X}_{iI})\Big] \\ &\quad + \sum_{i\in\mathcal{O}} \Big[\log h(t_{iO}|\boldsymbol{\eta}, \boldsymbol{X}_{iO}) - H(t_{iO}|\boldsymbol{\eta}, \boldsymbol{X}_{iO})\Big] \\ &\quad - \sum_{i\in\mathcal{R}} H(t_{iR}|\boldsymbol{\eta}, \boldsymbol{X}_{iR}). \end{aligned} \tag{4.8}$$

Note that the first sum term in (4.8) is the log-likelihood for left-censored observations, the second for finite interval-censored observations, the third for exactly observed failure time observations and the fourth for right-censored observations. To assure smoothness of the estimated baseline hazard, a penalty function is subtracted from the log-likelihood function (4.8), resulting in a penalized log-likelihood function,

$$\Phi(\boldsymbol{\eta}) = \ell(\boldsymbol{\eta}) - \tilde{\gamma} J(h_0), \tag{4.9}$$

where $\tilde{\gamma} > 0$ is a smoothing parameter used to balance fidelity of the fitted model to the data and smoothness of the estimated baseline hazard, and $J(h_0)$ is a penalty function. In this chapter, the penalty function $J(h_0)$ is taken to be of the same form as that in Section 3.2, i.e., $J(h_0) = J(\boldsymbol{\theta}) = \boldsymbol{\theta}^T \boldsymbol{R} \boldsymbol{\theta}$.

4.3 Convexity of negative penalized log-likelihood function

In this section, we analyze convexity of the negative penalized log-likelihood function (4.9). The convexity implies that a local minimizer of $-\Phi(\boldsymbol{\eta})$ will also be a global minimizer, and the global minimizer is unique if the convexity is strict.

We first re-express the negative penalized log-likelihood $-\Phi(\boldsymbol{\eta})$ as

$$\begin{aligned} -\Phi(\boldsymbol{\eta}) &= -\sum_{i\in\mathcal{L}} \log\Big[1 - S(t_{iL}|\boldsymbol{\eta}, \boldsymbol{X}_{iL})\Big] \\ &\quad -\sum_{i\in\mathcal{I}} \Big[\log S(t_{iIL}|\boldsymbol{\eta}, \boldsymbol{X}_{iI}) + \log\Big(1 - \frac{S(t_{iIR}|\boldsymbol{\eta}, \boldsymbol{X}_{iI})}{S(t_{iIL}|\boldsymbol{\eta}, \boldsymbol{X}_{iI})}\Big)\Big] \\ &\quad -\sum_{i\in\mathcal{O}} \Big[\log h(t_{iO}|\boldsymbol{\eta}, \boldsymbol{X}_{iO}) - H(t_{iO}|\boldsymbol{\eta}, \boldsymbol{X}_{iO})\Big] \\ &\quad +\sum_{i\in\mathcal{R}} H(t_{iR}|\boldsymbol{\eta}, \boldsymbol{X}_{iR}) + \tilde{\gamma}\boldsymbol{\theta}^T\boldsymbol{R}\boldsymbol{\theta} \\ &= \Phi_{\mathcal{L}}(\boldsymbol{\eta}) + \Phi_{\mathcal{I}}(\boldsymbol{\eta}) + \Phi_{\mathcal{O}}(\boldsymbol{\eta}) + \Phi_{\mathcal{R}}(\boldsymbol{\eta}) + \tilde{\gamma}\boldsymbol{\theta}^T\boldsymbol{R}\boldsymbol{\theta}, \end{aligned} \tag{4.10}$$

where

$$\Phi_{\mathcal{L}}(\boldsymbol{\eta}) = \sum_{i\in\mathcal{L}} -\log\Big[1 - \exp\{-\boldsymbol{D}(t_{iL}, \boldsymbol{X}_{iL})^T\boldsymbol{\eta}\}\Big],$$

$$\Phi_{\mathcal{I}}(\boldsymbol{\eta}) = \sum_{i\in\mathcal{I}} \Big[-\log\Big(1 - \exp\{-(\boldsymbol{D}(t_{iIR}, \boldsymbol{X}_{iI}) - \boldsymbol{D}(t_{iIL}, \boldsymbol{X}_{iI}))^T\boldsymbol{\eta}\}\Big) + \boldsymbol{D}(t_{iIL}, \boldsymbol{X}_{iI})^T\boldsymbol{\eta}\Big],$$

$$\Phi_{\mathcal{O}}(\boldsymbol{\eta}) = \sum_{i\in\mathcal{O}} \Big[-\log\Big(\boldsymbol{C}(t_{iO}, \boldsymbol{X}_{iO})^T\boldsymbol{\eta}\Big) + \boldsymbol{D}(t_{iO}, \boldsymbol{X}_{iO})\boldsymbol{\eta}\Big]$$

and

$$\Phi_{\mathcal{R}}(\boldsymbol{\eta}) = \sum_{i\in\mathcal{R}} \boldsymbol{D}(t_{iR}, \boldsymbol{X}_{iR})^T\boldsymbol{\eta}.$$

To prove the convexity of $-\Phi(\boldsymbol{\eta})$, we need the following Lemma.

Lemma 4.1. *Let $\boldsymbol{V}_i$ be vectors with $m + p$ dimensions, such that $\boldsymbol{V}_i^T\boldsymbol{\eta} > 0$, $i = 1, 2, 3$. Then with any strictly convex function $G(x)$ in $x \in \mathbf{R}^+$, where $\mathbf{R}^+$ is the positive real number space, the functions $G_{\boldsymbol{V}_1,\boldsymbol{V}_2}(\boldsymbol{\eta})$ and $G_{\boldsymbol{V}_3}(\boldsymbol{\eta})$ defined by*

$$G_{\boldsymbol{V}_1,\boldsymbol{V}_2}(\boldsymbol{\eta}) = G(\boldsymbol{V}_1^T\boldsymbol{\eta}) + \boldsymbol{V}_2^T\boldsymbol{\eta} \tag{4.11}$$

and

$$G_{\boldsymbol{V}_3}(\boldsymbol{\eta}) = G(\boldsymbol{V}_3^T\boldsymbol{\eta}) \tag{4.12}$$

are strictly convex in $\boldsymbol{\eta} \in \boldsymbol{\Gamma}$.

Let $\boldsymbol{W}_1, \cdots, \boldsymbol{W}_r$ *be vectors with dimension of* $m+p$*, such that* $\boldsymbol{W}_j^T\boldsymbol{\eta} > 0$ *for* $j = 1, \cdots, r$*, and let* $K(\boldsymbol{\eta})$ *be a strictly convex function in* $\boldsymbol{\eta}$*, then the function* $Q(\boldsymbol{\eta})$ *defined by*

$$Q(\boldsymbol{\eta}) = K(\boldsymbol{\eta}) + \sum_{j=1}^{r} \boldsymbol{W}_j^T\boldsymbol{\eta} \tag{4.13}$$

is strictly convex in $\boldsymbol{\eta} \in \boldsymbol{\Gamma}$.

Proof. The proof is omitted here since it is a simple modification of the proof in Theorem 5.7 of Rockafellar (1997). □

It can be easily verified that both functions $-\log\big[1-\exp(-x)\big]$ and $-\log x$ are strictly convex in $x \in \mathbf{R}^+$. Since the sum of strictly convex functions is a strictly convex function, and $(\boldsymbol{D}(t_{iIR}, \boldsymbol{X}_{iI}) - \boldsymbol{D}(t_{iIL}, \boldsymbol{X}_{iI}))^T\boldsymbol{\eta} > 0$, $\boldsymbol{D}(t_{iL}, \boldsymbol{X}_{iL})^T\boldsymbol{\eta} > 0$ and $\boldsymbol{D}(t_{iIL}, \boldsymbol{X}_{iI})^T\boldsymbol{\eta} > 0$, using Lemma 4.1 where we set $G(x) = -\log\big[1-\exp(-x)\big]$, $\boldsymbol{V}_1 = \boldsymbol{D}(t_{iIR}, \boldsymbol{X}_{iI}) - \boldsymbol{D}(t_{iIL}, \boldsymbol{X}_{iI})$ and $\boldsymbol{V}_2 = \boldsymbol{D}(t_{iIL}, \boldsymbol{X}_{iI})$ in equation (4.11), and $\boldsymbol{V}_3 = \boldsymbol{D}(t_{iL}, \boldsymbol{X}_{iL})^T\boldsymbol{\eta}$ in equation (4.12), we conclude that both $\Phi_{\mathcal{L}}(\boldsymbol{\eta})$ and $\Phi_{\mathcal{I}}(\boldsymbol{\eta})$ are strictly convex in $\boldsymbol{\eta}$. Since $\boldsymbol{C}(t_{iO}, \boldsymbol{X}_{iO})^T\boldsymbol{\eta} > 0$ and $\boldsymbol{D}(t_{iO}, \boldsymbol{X}_{iO})^T\boldsymbol{\eta} > 0$, the strict convexity of $\Phi_{\mathcal{O}}(\boldsymbol{\eta})$ follows simply by setting $G(x) = -\log x$, $\boldsymbol{V}_1 = \boldsymbol{C}(t_{iO}, \boldsymbol{X}_{iO})$ and $\boldsymbol{V}_2 = \boldsymbol{D}(t_{iO}, \boldsymbol{X}_{iO})$ in equation (4.11). Since $\Phi_{\mathcal{L}}(\boldsymbol{\eta}) + \Phi_{\mathcal{I}}(\boldsymbol{\eta}) + \Phi_{\mathcal{O}}(\boldsymbol{\eta})$ is strictly convex in $\boldsymbol{\eta}$, and $\boldsymbol{D}(t_{iR}, \boldsymbol{X}_{iR})^T\boldsymbol{\eta} > 0$, by setting $K(\boldsymbol{\eta}) = \Phi_{\mathcal{L}}(\boldsymbol{\eta}) + \Phi_{\mathcal{I}}(\boldsymbol{\eta}) + \Phi_{\mathcal{O}}(\boldsymbol{\eta})$, $\boldsymbol{W}_i = \boldsymbol{D}(t_{iR}, \boldsymbol{X}_{iR})$ and $r = n_R$ in equation (4.13), where n_R is the number of right-censored observations, we conclude that $\Phi_{\mathcal{L}}(\boldsymbol{\eta}) + \Phi_{\mathcal{I}}(\boldsymbol{\eta}) + \Phi_{\mathcal{O}}(\boldsymbol{\eta}) + \Phi_{\mathcal{R}}(\boldsymbol{\eta})$ is strictly convex in $\boldsymbol{\eta}$. The penalty term in (4.10) is strictly convex because it is a quadratic function of $\boldsymbol{\theta}$. Therefore, the negative penalized log-likelihood function is strictly convex in $\boldsymbol{\eta} \in \boldsymbol{\Gamma}$.

4.4 Maximum penalized likelihood (MPL) estimation

Recall that the AH model specifies the hazard function by $h(t|\boldsymbol{X}_i) = h_0(t) + \boldsymbol{X}_i^T\boldsymbol{\beta}$. Since the covariate effects act additively on the baseline hazard function, both the baseline hazard function and the hazard function have to be constrained to be non-negative in computing the MPL estimators of $\boldsymbol{\eta}$, i.e., $h_0(t) \geq 0$ and $h(t|\boldsymbol{X}_i) \geq 0$. Hence the proper constrained optimization problem is to solve

$$\hat{\boldsymbol{\eta}} = \text{argmax}_{\boldsymbol{\eta}}\Phi(\boldsymbol{\eta}), \tag{4.14}$$

subject to the following two conditions,

$$\theta_u \geq 0 \quad \text{for } u = 1, \cdots, m \tag{4.15}$$

and

$$\boldsymbol{C}(t_{iv}, \boldsymbol{X}_{iv})^T\boldsymbol{\eta} \geq 0 \quad \text{for } i = 1, \cdots, n + n_I \text{ and } v = \{O, L, IL, IR, R\}, \tag{4.16}$$

where n_I is the number of subjects with their failure times finite interval-censored.

4.4.1 Constrained optimization by the primal-dual interior-point algorithm

We first define a function f by

$$f(\boldsymbol{\eta}) = [f_1(\boldsymbol{\eta}), f_2(\boldsymbol{\eta}), \ldots, f_{m+n+n_I}(\boldsymbol{\eta})]^T, \tag{4.17}$$

where $f_i(\boldsymbol{\eta}) = -\theta_i$, $i = 1, 2, \cdots, m$ and $f_{m+i}(\boldsymbol{\eta}) = -\boldsymbol{C}(t_{iv}, \boldsymbol{X}_{iv})^T\boldsymbol{\eta}$, $i = 1, 2, \cdots, n + n_I$. We define a matrix $\boldsymbol{M}_1$ by $\boldsymbol{M}_1 = \left[\boldsymbol{0}_{m\times p}, \boldsymbol{I}_{m\times m}\right]$, where $\boldsymbol{0}_{m\times p}$ is a zero matrix and $\boldsymbol{I}_{m\times m}$ is an identity matrix, and define $\boldsymbol{M}_2$ to be a $(n + n_I) \times (m + p)$ matrix with row vectors given by $\boldsymbol{C}(t_{iv}, \boldsymbol{X}_{iv})^T$. By combining $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$, we define $\bar{\boldsymbol{M}} = \left[\boldsymbol{M}_1^T, \boldsymbol{M}_2^T\right]^T$. Then the function (4.17) can be written equivalently by

$$f(\boldsymbol{\eta}) = -\bar{\boldsymbol{M}}\boldsymbol{\eta}. \tag{4.18}$$

The constrained optimization problem (4.14)-(4.16) can now be reformulated to find $\hat{\boldsymbol{\eta}} \in \mathcal{F}$ such that

$$\hat{\boldsymbol{\eta}} = \arg \min_{\boldsymbol{\eta} \in \mathcal{F} \subset \boldsymbol{\Gamma}} \{-\Phi(\boldsymbol{\eta})\}, \tag{4.19}$$

In (4.19), $\mathcal{F}$ is a feasible set and defined by

$$\mathcal{F} = \left\{\boldsymbol{\eta} | f_i(\boldsymbol{\eta}) = -\bar{\boldsymbol{M}}_i \boldsymbol{\eta} \leq 0, i = 1, \cdots, m+n+n_I\right\},$$

where $\bar{\boldsymbol{M}}_i$ is the ith row vector of matrix $\bar{\boldsymbol{M}}$. The set $\mathcal{F}$ is also called a polyhedral set, and the polyhedral is a closed convex set; see Example 1.3.3 in Sun and Yuan (2006). Since the objective function $-\Phi(\boldsymbol{\eta})$ is strictly convex in $\boldsymbol{\eta} \in \boldsymbol{\Gamma}$, we conclude that (4.19) is a convex optimization problem.

Let $-\nabla\Phi(\boldsymbol{\eta})$ be the gradient of $-\Phi(\boldsymbol{\eta})$. The constrained optimization problem (4.19) is solved by the following theorem.

Theorem 4.2. *Let $\boldsymbol{\eta} \in \boldsymbol{\Gamma}$, such that $-\Phi(\boldsymbol{\eta}) < \infty$, then $\boldsymbol{\eta}$ solves the constrained optimization problem (4.19) if and only if there exist vectors $\boldsymbol{\lambda}$ and $\boldsymbol{s}$ satisfying the following Karush-Kuhn-Tucker (KKT) conditions:*

$$-\nabla\Phi(\boldsymbol{\eta}) + \nabla f^T(\boldsymbol{\eta})\boldsymbol{\lambda} = \mathbf{0}_{(m+p)\times 1}, \tag{4.20}$$

$$f(\boldsymbol{\eta}) + \boldsymbol{s} = \mathbf{0}_{(m+n+n_I)\times 1}, \tag{4.21}$$

$$\lambda_i s_i = 0, \quad i = 1, \cdots, m+n+n_I, \tag{4.22}$$

and

$$(\boldsymbol{\lambda}, \boldsymbol{s}) \geq \mathbf{0}_{(m+n+n_I)\times 1}, \tag{4.23}$$

where $\nabla f(\boldsymbol{\eta}) = -\bar{\boldsymbol{M}}$ by equation (4.18), $\boldsymbol{\lambda}$ is a vector of Lagrange multipliers, and $\boldsymbol{s}$ is a vector of slack variables for the constraints.

Proof. Detail of the proof of the theorem is omitted here since it is a simple modification of the proof given in Chapter 8 of Sun and Yuan (2006). □

Based on Theorem 4.2, we compute the MPL estimators of $\boldsymbol{\eta}$ by using a primal-dual interior point algorithm. The algorithm is known as primal-dual for the reason that we solve simultaneously the primal problem for the vector $\boldsymbol{\eta}$ and the dual problem for the vectors $\boldsymbol{\lambda}$ and $\boldsymbol{s}$. In each iteration, the algorithm searches for solutions of the KKT system (4.20)-(4.23) firstly by applying Newton's method to the three equality conditions (4.20)-(4.22), so that a Newton direction is obtained. Then a line search is performed along the Newton direction to guarantee that the inequality condition (4.23) is satisfied strictly for new updates. However, we can only take a small step along the Newton direction before violating the inequality condition (Wright, 1997). Hence the Newton direction does not allow us to make much progress toward a solution. But we can modify the Newton procedure by requiring that each pairwise product $\lambda_i s_i$ has the same positive value and setting $\lambda_i s_i = \sigma\mu$. The parameter σ is called a centering parameter satisfying $\sigma \in [0, 1]$, and μ is called a duality measure defined by

$$\mu = \frac{\boldsymbol{\lambda}^T \boldsymbol{s}}{m + n + n_I}, \tag{4.24}$$

which measures the average value of $\lambda_i s_i$. Since the modified Newton step is toward the interior of the non-negative orthant $(\boldsymbol{\lambda}, \boldsymbol{s}) \geq \mathbf{0}_{m+n+n_I}$, where the pairwise product $\lambda_i s_i$ is kept strictly positive, we can take longer steps before violating the positivity condition. During each iteration of the algorithm, as μ goes to zero, $\lambda_i s_i$ decreases to zero at the same rate, and condition (4.22) is satisfied.

The algorithm has been studied in depth by Wright (1997) and Sun and Yuan (2006), and one can refer to them for more detail. Ghosh (2001) has applied the algorithm in studying the AH model with current status data. Before presenting the algorithm, we introduce some definitions. We define $r_p(\boldsymbol{\eta}, \boldsymbol{s})$ and $r_d(\boldsymbol{\eta}, \boldsymbol{\lambda})$ by

$$r_p(\boldsymbol{\eta}, \boldsymbol{s}) = f(\boldsymbol{\eta}) + \boldsymbol{s} \tag{4.25}$$

and

$$r_d(\boldsymbol{\eta}, \boldsymbol{\lambda}) = -\nabla\Phi(\boldsymbol{\eta}) + \nabla f(\boldsymbol{\eta})^T\boldsymbol{\lambda}, \tag{4.26}$$

and define $\phi_{\boldsymbol{\lambda}}(\boldsymbol{\eta})$ by $\phi_{\boldsymbol{\lambda}}(\boldsymbol{\eta}) = -\Phi(\boldsymbol{\eta}) + \boldsymbol{\lambda}^T f(\boldsymbol{\eta})$. Let $\nabla_{\eta\eta}\phi_\lambda(\boldsymbol{\eta})$ be the Hessian matrix of $\phi_{\boldsymbol{\lambda}}(\boldsymbol{\eta})$ with respect to $\boldsymbol{\eta}$. Let $\boldsymbol{I}$ be an identity matrix with dimension $(m+n+n_I)\times(m+n+n_I)$. Define $\boldsymbol{\Lambda}$ and $\boldsymbol{S}$ respectively as diagonal matrices with diagonal elements λ_i and s_i, $i = 1, \cdots, m+n+n_I$. For a fixed $\rho > 0$, we define $\mathcal{N}(\mu)$ as

$$\begin{aligned}\mathcal{N}(\mu) \;=\; & \{(\boldsymbol{\eta}, \boldsymbol{\lambda}, \boldsymbol{s}) |\, \|r_p(\boldsymbol{\eta}, \boldsymbol{s})\| \le \rho\mu, \|r_d(\boldsymbol{\eta}, \boldsymbol{\lambda})\| \le \rho\mu, (\lambda_i, s_i) \ge 0, \\ & \lambda_i s_i \ge \mu/2, i = 1, \cdots, m+n+n_I\},\end{aligned}$$

where $\|.\|$ is the Euclidean norm. With these definitions, we outline the computational procedure of the algorithm below:

1. We select $\boldsymbol{\eta}^{(0)}$ by $\boldsymbol{\beta}^{(0)} = \mathbf{0}_{p\times 1}$ and $\boldsymbol{\theta}^{(0)} = \mathbf{1}_{m\times 1}$, leading to $f(\boldsymbol{\eta}^{(0)}) < 0$. We take $\boldsymbol{\lambda}^{(0)} = \boldsymbol{s}^{(0)} = \mathbf{1}_{(m+n+n_I)\times 1}$.

2. At the kth iteration of the algorithm, we have $\boldsymbol{\eta}^{(k)}$, $\boldsymbol{\lambda}^{(k)}$ and $\boldsymbol{s}^{(k)}$. Then the duality measure is given by $\mu_k = (\boldsymbol{\lambda}^{(k)})^T\boldsymbol{s}^{(k)}/(m+n+n_I)$. The Newton direction, $(d\boldsymbol{\eta}^T, d\boldsymbol{\lambda}^T, d\boldsymbol{s}^T)^T$, is obtained by solving the following equation:

$$\begin{pmatrix} \nabla_{\eta\eta}\phi_\lambda(\boldsymbol{\eta}^{(k)}) & \nabla f(\boldsymbol{\eta}^{(k)})^T & \mathbf{0} \\ \nabla f(\boldsymbol{\eta}^{(k)}) & \mathbf{0} & \boldsymbol{I} \\ \mathbf{0} & \boldsymbol{S}^{(k)} & \boldsymbol{\Lambda}^{(k)} \end{pmatrix} \begin{pmatrix} d\boldsymbol{\eta}^{(k)} \\ d\boldsymbol{\lambda}^{(k)} \\ d\boldsymbol{s}^{(k)} \end{pmatrix} = \begin{pmatrix} -r_d(\boldsymbol{\eta}^{(k)}, \boldsymbol{\lambda}^{(k)}) \\ -r_p(\boldsymbol{\eta}^{(k)}, \boldsymbol{s}^{(k)}) \\ -\boldsymbol{\Lambda}^{(k)}\boldsymbol{S}^{(k)}\boldsymbol{e} + \sigma\mu^{(k)}\boldsymbol{e} \end{pmatrix}. \tag{4.27}$$

   where σ is set to be $\sigma = 0.5$. Then the updated iterate is computed according to

$$(\boldsymbol{\eta}^{(k+1)}, \boldsymbol{\lambda}^{(k+1)}, \boldsymbol{s}^{(k+1)}) = (\boldsymbol{\eta}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{s}^{(k)}) + \alpha_k(d\boldsymbol{\eta}^{(k)}, d\boldsymbol{\lambda}^{(k)}, d\boldsymbol{s}^{(k)}), \tag{4.28}$$

   and the updated duality measure is given by $\mu_{k+1} = (\boldsymbol{\lambda}^{(k+1)})^T\boldsymbol{s}^{(k+1)}/(m+n+n_I)$. The step length α_k in (4.28) is chosen to be the first element in the sequence

$$\{1, \kappa, \kappa^2, \kappa^3, \cdots\},$$

where $\kappa \in (0,1)$, such that $(\eta^{(k+1)}, \lambda^{(k+1)}, s^{(k+1)}) \in \mathcal{N}(\mu_k)$ and the following condition holds,

$$\mu_{k+1} \leq (1 - 0.01\alpha_k)\mu_k. \tag{4.29}$$

3. Go to step 2 and repeat the process until μ_k is below a certain tolerance criterion, such as 10^{-5} or 10^{-10}.

The algorithm requires the first and second derivatives of $\Phi(\boldsymbol{\eta})$ with respect to $\boldsymbol{\eta}$. For simplicity, we denote $S(\cdot)$ to be $S(\cdot|\boldsymbol{\eta}, \boldsymbol{X}_i)$ defined in (4.7), and $h(\cdot)$ to be $h(\cdot|\boldsymbol{\eta}, \boldsymbol{X}_i)$ defined in (4.5). Then the first derivative of $\Phi(\boldsymbol{\eta})$ with respect to $\boldsymbol{\beta}$ is

$$\frac{\partial \Phi(\boldsymbol{\eta})}{\partial \boldsymbol{\beta}} = \boldsymbol{X}_L^T \mathcal{A} \mathbf{1}_{n_L} + \boldsymbol{X}_I^T \mathcal{B} \mathbf{1}_{n_I} - \boldsymbol{X}_R^T \mathcal{C} \mathbf{1}_{n_R} + \boldsymbol{X}_O^T \mathcal{D} \mathbf{1}_{n_O},$$

where n_L, n_I, n_R and n_O are all defined in Chapter 3. The second derivative of $\Phi(\boldsymbol{\eta})$ with respect to $\boldsymbol{\beta}$ is

$$\frac{\partial^2 \Phi(\boldsymbol{\eta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} = -\boldsymbol{X}_L^T \mathcal{E} \boldsymbol{X}_L - \boldsymbol{X}_I^T \mathcal{G} \boldsymbol{X}_I - \boldsymbol{X}_O^T \mathcal{L} \boldsymbol{X}_O,$$

where $\boldsymbol{X}_L$, $\boldsymbol{X}_I$ and $\boldsymbol{X}_O$ are defined in Chapter 3, and matrices $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$, $\mathcal{E}$, $\mathcal{G}$ and $\mathcal{L}$ are diagonal matrices defined by

$$\mathcal{A} = \text{diag}\left\{ \frac{t_{iL} S(t_{iL})}{1 - S(t_{iL})} \right\},$$

$$\mathcal{B} = \text{diag}\left\{ \frac{t_{iIR} S(t_{iIR}) - t_{iIL} S(t_{iIL})}{S(t_{iIL}) - S(t_{iIR})} \right\},$$

$$\mathcal{C} = \text{diag}\left\{ t_{iR} \right\},$$

$$\mathcal{D} = \text{diag}\left\{ \frac{1}{h(t_{iO})} - t_{iO} \right\},$$

$$\mathcal{E} = \text{diag}\left\{ \frac{(t_{iL})^2 S(t_{iL})}{[1 - S(t_{iL})]^2} \right\},$$

$$\mathcal{G} = \text{diag}\left\{ \frac{(t_{iIR} - t_{iIL})^2 S(t_{iIR}) S(t_{iIL})}{[S(t_{iIL}) - S(t_{iIR})]^2} \right\}$$

and

$$\mathcal{L} = \text{diag}\left\{ \frac{1}{h^2(t_{iO})} \right\}.$$

For convenience in deriving the first and second derivatives of $\Phi(\boldsymbol{\eta})$ with respect to θ_u, we adopt the notation defined in Page 49 in Section 3.3. Then the first derivative of $\Phi(\boldsymbol{\eta})$ with respect to θ_u is

$$
\begin{aligned}
\frac{\partial \Phi(\boldsymbol{\eta})}{\partial \theta_u} &= \sum_{i \in \mathcal{L}_u} \left[\frac{(t_{iL_u} - \tau_{u-1}) S(t_{iL_u})}{1 - S(t_{iL_u})} \right] + \delta_u \sum_{w>u} \sum_{i \in \mathcal{L}_w} \left[\frac{S(t_{iL_w})}{1 - S(t_{iL_w})} \right] \\
&+ \sum_{v<u} \sum_{i \in \mathcal{IL}_v} \sum_{i \in \mathcal{IR}_u} \left[\frac{(t_{iIR_u} - \tau_{u-1}) S(t_{iIR_u})}{S(t_{iIL_v}) - S(t_{iIR_u})} \right] \\
&+ \delta_u \sum_{v<u} \sum_{w>u} \sum_{i \in \mathcal{IL}_v} \sum_{i \in \mathcal{IR}_w} \left[\frac{S(t_{iIR_w})}{S(t_{iIL_v}) - S(t_{iIR_w})} \right] \\
&+ \sum_{i \in \mathcal{IL}_u} \sum_{i \in \mathcal{IR}_u} \left[\frac{(t_{iIR_u} - \tau_{u-1}) S(t_{iIR_u}) - (t_{iIL_u} - \tau_{u-1}) S(t_{iIL_u})}{S(t_{iIL_u}) - S(t_{iIR_u})} \right] \\
&+ \sum_{w>u} \sum_{i \in \mathcal{IL}_u} \sum_{i \in \mathcal{IR}_w} \left[\frac{\delta_u S(t_{iIR_w}) - (t_{iIL_u} - \tau_{u-1}) S(t_{iIL_u})}{S(t_{iIL_u}) - S(t_{iIR_w})} \right] - \delta_u \sum_{w>u} N_w^{IL} \\
&+ \sum_{i \in \mathcal{O}_u} \left[\frac{1}{\theta_u + \boldsymbol{X}_{iO_u}^T \boldsymbol{\beta}} - (t_{iO_u} - \tau_{u-1}) \right] - \sum_{i \in \mathcal{R}_u} (t_{iR_u} - \tau_{u-1}) \\
&- \delta_u \sum_{w>u} (N_w^O + N_w^R) - 2\gamma \boldsymbol{R}_u \boldsymbol{\theta},
\end{aligned}
$$

where N_w^{IL} is the number of subjects whose failure times are finite interval-censored with the corresponding observed left limit t_{iIL} in B_w, N_w^R is the number of subjects whose failure times are right-censored with the corresponding observed time t_{iR} in B_w, and N_w^O is the number of subjects whose failure times are exactly observed with the corresponding

failure time t_{iO} in B_w. The second derivative of $\Phi(\boldsymbol{\eta})$ with respect to θ_u is

$$\begin{aligned}
\frac{\partial^2\Phi(\boldsymbol{\eta})}{\partial\theta_u^2} &= -\sum_{i\in\mathcal{L}_u}\left[\frac{(t_{iL_u}-\tau_{u-1})^2S(t_{iL_u})}{(1-S(t_{iL_u}))^2}\right]-\delta_u^2\sum_{w>u}\sum_{i\in\mathcal{L}_w}\left[\frac{S(t_{iL_w})}{(1-S(t_{iL_w}))^2}\right]\\
&\quad-\sum_{v<u}\sum_{i\in\mathcal{IL}_v}\sum_{i\in\mathcal{IR}_u}\left[\frac{(t_{iIR_u}-\tau_{u-1})^2S(t_{iIL_v})S(t_{iIR_u})}{(S(t_{iIL_v})-S(t_{iIR_u}))^2}\right]\\
&\quad-\delta_u^2\sum_{v<u}\sum_{w>u}\sum_{i\in\mathcal{IL}_v}\sum_{i\in\mathcal{IR}_w}\left[\frac{S(t_{iIL_v})S(t_{iIR_w})}{(S(t_{iIL_v})-S(t_{iIR_w}))^2}\right]\\
&\quad-\sum_{i\in\mathcal{IL}_u}\sum_{i\in\mathcal{IR}_u}\left[\frac{(t_{iIL_u}-t_{iIR_u})^2S(t_{iIL_u})S(t_{iIR_u})}{(S(t_{iIL_u})-S(t_{iIR_u}))^2}\right]\\
&\quad-\sum_{w>u}\sum_{i\in\mathcal{IL}_u}\sum_{i\in\mathcal{IR}_w}\left[\frac{(t_{iIL_u}-\tau_{u-1}-\delta_u)^2S(t_{iIL_u})S(t_{iIR_w})}{(S(t_{iIL_u})-S(t_{iIR_w}))^2}\right]\\
&\quad-\sum_{i\in\mathcal{O}_u}\left[\frac{1}{(\theta_u+\boldsymbol{X}_{iO_u}^T\boldsymbol{\beta})^2}\right]-2\tilde{\gamma}r_{uu},
\end{aligned}$$

where r_{uu} is the u^{th} diagonal entry of the matrix $\boldsymbol{R}$. For $u>k$, we have

$$\begin{aligned}
\frac{\partial^2\Phi(\boldsymbol{\eta})}{\partial\theta_k\partial\theta_u} &= -\delta_k\sum_{i\in\mathcal{L}_u}\left[\frac{(t_{iL_u}-\tau_{u-1})S(t_{iL_u})}{(1-S(t_{iL_u}))^2}\right]-\delta_u\delta_k\sum_{w>u}\sum_{i\in\mathcal{L}_w}\left[\frac{S(t_{iL_w})}{(1-S(t_{iL_w}))^2}\right]\\
&\quad-\delta_k\sum_{v<k}\sum_{i\in\mathcal{IL}_v}\sum_{i\in\mathcal{IR}_u}\left[\frac{(t_{iIR_u}-\tau_{u-1})S(t_{iIR_u})S(t_{iIL_v})}{(S(t_{iIL_v})-S(t_{iIR_u}))^2}\right]\\
&\quad-\sum_{i\in\mathcal{IL}_k}\sum_{i\in\mathcal{IR}_u}\left[\frac{(t_{iIR_u}-\tau_{u-1})(\delta_k+\tau_{k-1}-t_{iIL_k})S(t_{iIL_k})S(t_{iIR_u})}{(S(t_{iIL_k})-S(t_{iIR_u}))^2}\right]\\
&\quad-\delta_k\delta_u\sum_{v<k}\sum_{w>u}\sum_{i\in\mathcal{IL}_v}\sum_{i\in\mathcal{IR}_w}\left[\frac{S(t_{iIR_w})S(t_{iIL_v})}{(S(t_{iIL_v})-S(t_{iIR_w}))^2}\right]\\
&\quad-\delta_u\sum_{w>u}\sum_{i\in\mathcal{IL}_k}\sum_{i\in\mathcal{IR}_w}\left[\frac{(\delta_k+\tau_{k-1}-t_{iIL_k})S(t_{iIR_w})S(t_{iIL_k})}{(S(t_{iIL_k})-S(t_{iIR_w}))^2}\right]-2\tilde{\gamma}r_{ku},
\end{aligned}$$

where r_{ku} is the entry in the k^{th} row and u^{th} column of the matrix $\boldsymbol{R}$, and

$$\begin{aligned}
\frac{\partial^2\Phi(\boldsymbol{\eta})}{\partial\boldsymbol{\beta}\partial\theta_u} &= -\sum_{i\in\mathcal{L}_u}\left[\frac{t_{iL_u}(t_{iL_u}-\tau_{u-1})S(t_{iL_u})}{(1-S(t_{iL_u}))^2}\boldsymbol{X}_{iL_u}\right]-\delta_u\sum_{w>u}\sum_{i\in\mathcal{L}_w}\left[\frac{t_{iL_w}S(t_{iL_w})}{(1-S(t_{iL_w}))^2}\boldsymbol{X}_{iL_w}\right] \\
&\quad -\sum_{v<u}\sum_{i\in\mathcal{IL}_v}\sum_{i\in\mathcal{IR}_u}\left[\frac{(t_{iIR_u}-t_{iIL_v})(t_{iIR_u}-\tau_{u-1})S(t_{iIR_u})S(t_{iIL_v})}{(S(t_{iIL_v})-S(t_{iIR_u}))^2}\boldsymbol{X}_{iIR_u}\right] \\
&\quad -\delta_u\sum_{v<u}\sum_{w>u}\sum_{i\in\mathcal{IL}_v}\sum_{i\in\mathcal{IR}_w}\left[\frac{(t_{iIR_w}-t_{iIL_v})S(t_{iIR_w})S(t_{iIL_v})}{(S(t_{iIL_v})-S(t_{iIR_w}))^2}\boldsymbol{X}_{iIR_w}\right] \\
&\quad -\sum_{i\in\mathcal{IL}_u}\sum_{i\in\mathcal{IR}_u}\left[\frac{(t_{iIL_u}-t_{iIR_u})^2S(t_{iIL_u})S(t_{iIR_u})}{(S(t_{iIL_u})-S(t_{iIR_u}))^2}\boldsymbol{X}_{iIR_u}\right] \\
&\quad -\sum_{w>u}\sum_{i\in\mathcal{IL}_u}\sum_{i\in\mathcal{IR}_w}\left[\frac{(t_{iIR_w}-t_{iIL_u})(\delta_u+\tau_{u-1}-t_{iIL_u})S(t_{iIR_w})S(t_{iIL_u})}{(S(t_{iIL_u})-S(t_{iIR_w}))^2}\boldsymbol{X}_{iIR_w}\right] \\
&\quad -\sum_{i\in\mathcal{O}_u}\left[\frac{1}{(\theta_u+\boldsymbol{X}_{iO_u}^T\boldsymbol{\beta})^2}\boldsymbol{X}_{iO_u}\right].
\end{aligned}$$

4.4.2 Convergence of the algorithm

In this section, we analyze global convergence of the primal-dual interior point algorithm. We first give a brief description of two types of convergence rate. The convergence rate is a local characterization of an algorithm and can be used to measure the effectiveness of an optimization method. Suppose there is an iterative sequence $\{a^{(k)}\}$ generated by an algorithm and converging to a^* in some norm, i.e.,

$$\lim_{k\to\infty}\|a^{(k)}-a^*\|=0.$$

If there is a constant $b\in(0,1)$, which is independent of the iteration number k, and satisfies

$$\lim_{k\to\infty}\frac{\|a^{(k+1)}-a^*\|}{\|a^{(k)}-a^*\|}=b,$$

then we say that the sequence $\{a^{(k)}\}$ converges Q-linearly. The sequence $\{a^{(k)}\}$ is said to be R-linearly convergent to a^* if there is a sequence of non-negative scalars $\{c^{(k)}\}$ such

that

$$\|a^{(k+1)} - a^*\| \leq c^{(k)} \quad \text{for all } k,$$

with $\{c^{(k)}\}$ converging Q-linearly to zero. Now we state the global convergence of the primal-dual interior algorithm in the following theorem.

Theorem 4.3. *The sequence of iterates $(\boldsymbol{\eta}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{s}^{(k)})$ generated by the primal-dual interior point algorithm converges to a solution satisfying the KKT conditions (4.20)-(4.23). Furthermore, the sequence of the duality measure $\{\mu_k\}$ converges Q-linearly to zero, and the sequences of residual norms $\|r_p(\boldsymbol{\eta}^{(k)}, \boldsymbol{s}^{(k)})\|$ in (4.25) and $\|r_d(\boldsymbol{\eta}^{(k)}, \boldsymbol{\lambda}^{(k)})\|$ in (4.26) converge R-linearly to zero.*

Proof. The proof is omitted here, since it is a simple modification of that given in Wright (1997), and Ralph and Wright (1997). □

4.5 Asymptotic properties of the MPL estimators

The methods for analyzing asymptotic properties of the MPL estimators under the PH model in Section 3.5 can be applied to the MPL estimators under the AH model. In the analysis of the asymptotic properties, we restrict ourselves to the situation of fixed dimension of $\boldsymbol{\theta}$ (i.e., fixed m). Let $\boldsymbol{\eta}_0$ be the true value of $\boldsymbol{\eta}$ and $\hat{\boldsymbol{\eta}}$ be the MPL estimator of $\boldsymbol{\eta}$. For convenience, we use the observation formulation defined in (3.32), $\{(L_i, R_i], \boldsymbol{X}_i, \xi_i\}$, $i = 1, \cdots, n$. We denote $g(L_i, R_i, \xi_i|\boldsymbol{\eta})$ as a density function of $\{(L_i, R_i], \boldsymbol{X}_i, \xi_i\}$, and express it by

$$g(L_i, R_i, \xi_i|\boldsymbol{\eta}) = \big[S(L_i|\boldsymbol{\eta}) - S(R_i|\boldsymbol{\eta})\big]^{1-\xi_i} f(t_i|\boldsymbol{\eta})^{\xi_i}, \tag{4.30}$$

where ξ_i is an indicator with $\xi_i = 1$ if the failure time of subject i is exactly observed, or $\xi_i = 0$ if censored. In the density function (4.30), $S(\cdot|\boldsymbol{\eta})$ is the survival function of the

failure time given by (4.7), and $f(\cdot|\boldsymbol{\eta})$ is the density function of the failure time given by

$$f(t_i|\boldsymbol{\eta}) = h(t_i|\boldsymbol{\eta})S(t_i|\boldsymbol{\eta}) = \boldsymbol{C}(t_i, \boldsymbol{X}_i)^T\boldsymbol{\eta}\exp\{-\boldsymbol{D}(t_i, \boldsymbol{X}_i)^T\boldsymbol{\eta}\},$$

where $h(\cdot|\boldsymbol{\eta})$ is given by (4.5). Let $\ell_i(\boldsymbol{\eta}) = \log g(L_i, R_i, \xi_i|\boldsymbol{\eta})$. Then the log-likelihood from the $g(L_i, R_i, \xi_i|\boldsymbol{\eta})$ is identical to (4.8). We denote the penalized log-likelihood function in (4.9) by $\Phi_n(\boldsymbol{\eta})$ to indicate its dependence on the sample size, and re-express it as

$$\Phi_n(\boldsymbol{\eta}) = \sum_{i=1}^{n}\{\ell_i(\boldsymbol{\eta}) - \tilde{\gamma}_n J(\boldsymbol{\eta})\},$$

where $J(\boldsymbol{\eta}) = J(\boldsymbol{\theta})$ and $\tilde{\gamma}_n = \tilde{\gamma}/n$. The following assumptions are needed to develop the asymptotic properties of $\hat{\boldsymbol{\eta}}$.

Assumptions:

4.1 $\{(L_i, R_i], \boldsymbol{X}_i, \xi_i : i = 1, \cdots, n\}$ are independently and identically distributed, and the distribution of covariate $\boldsymbol{X}_i$ is independent of $\boldsymbol{\eta}$.

4.2 The censoring time is independent of the failure time conditionally on the covariates, and the distribution of censoring is independent of $\boldsymbol{\eta}$.

4.3 The parameter space $\boldsymbol{\Gamma}$ of $\boldsymbol{\eta}$ is compact.

4.4 The density function $g(L_i, R_i, \xi_i|\boldsymbol{\eta})$ is greater than zero. There exists a measurable function $m(L_i, R_i, \boldsymbol{X}_i, \xi_i)$ with $E[m(L_i, R_i, \boldsymbol{X}_i, \xi_i)] < \infty$, satisfying $|\log g(L_i, R_i, \xi_i|\boldsymbol{\eta})| \leq m(L_i, R_i, \boldsymbol{X}_i, \xi_i)$ for all $\boldsymbol{\eta} \in \boldsymbol{\Gamma}$.

4.5 $E_{\boldsymbol{\eta}_0}[n^{-1}\Phi_n(\boldsymbol{\eta})]$ exists and has a unique maximum at $\boldsymbol{\eta}^* \in \text{int}(\boldsymbol{\Gamma})$, which is not necessarily equal to $\boldsymbol{\eta}_0$ due to the penalty function.

4.6 $\Phi_n(\boldsymbol{\eta})$ is continuous over $\boldsymbol{\Gamma}$ and is twice differentiable in a neighborhood of $\boldsymbol{\eta}^*$. Also, the matrices

$$G_0(\boldsymbol{\eta}) = E_{\boldsymbol{\eta}_0}\left[n^{-1}\frac{\partial\Phi_n(\boldsymbol{\eta})}{\partial\boldsymbol{\eta}}\frac{\partial\Phi_n(\boldsymbol{\eta})}{\partial\boldsymbol{\eta}^T}\right]$$

and

$$F_0(\boldsymbol{\eta}) = -E_{\boldsymbol{\eta}_0}\left[n^{-1}\frac{\partial^2\Phi_n(\boldsymbol{\eta})}{\partial\boldsymbol{\eta}\partial\boldsymbol{\eta}^T}\right]$$

exist and are positive definite in a neighborhood of $\boldsymbol{\eta}^*$.

4.7 The penalty function $J(\boldsymbol{\eta})$ is continuous and bounded over $\boldsymbol{\Gamma}$. Both $\partial J(\boldsymbol{\eta})/\partial\boldsymbol{\eta}$ and $\partial^2 J(\boldsymbol{\eta})/\partial\boldsymbol{\eta}\partial\boldsymbol{\eta}^T$ exist for all $\boldsymbol{\eta} \in \boldsymbol{\Gamma}$, and $\partial^2 J(\boldsymbol{\eta})/\partial\boldsymbol{\eta}\partial\boldsymbol{\eta}^T$ is bounded in a neighborhood of $\boldsymbol{\eta}^*$.

The asymptotic results are stated in Theorem 4.4 below. Their proof is the same as that in Chapter 3 and hence is omitted here.

Theorem 4.4. *Assume that Assumptions 4.1-4.7 are satisfied, then the MPL estimator $\hat{\boldsymbol{\eta}}$ is consistent for $\boldsymbol{\eta}^*$ and the distribution of $\sqrt{n}(\hat{\boldsymbol{\eta}}_n - \boldsymbol{\eta}^*)$ converges, when $n \to \infty$, to a multivariate normal distribution $N(0_{(p+m)\times 1}, V(\boldsymbol{\eta}^*))$, where*

$$V(\boldsymbol{\eta}^*) = F_0(\boldsymbol{\eta}^*)^{-1}G_0(\boldsymbol{\eta}^*)F_0(\boldsymbol{\eta}^*)^{-1} \tag{4.31}$$

In developing the asymptotic properties of $\hat{\boldsymbol{\eta}}$, we keep $\tilde{\gamma}_n$ fixed as $n \to \infty$. In the case where $\tilde{\gamma}_n \to 0$ as $n \to \infty$, the asymptotic results and proof follow from those in the maximum likelihood estimation (MLE) method. In practice, $\boldsymbol{\eta}^*$ can be replaced by $\hat{\boldsymbol{\eta}}$ due to the strong consistency property. The simulation study in Section 4.6 shows that the asymptotic sandwich formula (4.31) is generally accurate.

4.6 Simulation studies

In this section, simulation studies are conducted to evaluate the performance of our proposed MPL method in fitting the AH model with partly interval-censored data. We concentrate on the following three objectives:

1. Investigating effects of the censoring proportion and sample size on the MPL estimators of regression coefficients $\boldsymbol{\beta}$ and the baseline hazard $h_0(\cdot)$. The results are summarized in Tables 4.1-4.3 and Figures 4.1-4.3.

2. Evaluating whether the asymptotic standard deviations computed by the sandwich formula, $F_0(\hat{\boldsymbol{\eta}})^{-1}G_0(\hat{\boldsymbol{\eta}})F_0(\hat{\boldsymbol{\eta}})^{-1}$, are accurate for the MPL estimators. This is achieved by comparing them with the Monte Carlo standard deviations. Results can be viewed in Tables 4.1-4.3, and Figures 4.1-4.3.

3. Comparing our proposed MPL method with an estimation method based on the counting process by Lin and Ying (1994) to demonstrate improvement of the estimations of $\boldsymbol{\beta}$ and $h_0(\cdot)$ by our MPL method. Results are reported in Table 4.4 and Figure 4.4. R code for the counting process method is available in CRAN.

All the simulation results are computed using the R program, and the relevant R codes are provided in Appendix B.

We choose $n = 100$, 500 and 1000 respectively as small, intermediate and large sample sizes, with approximate censoring proportions of $\pi_c = 20\%$, 50% and 80% for each value of n. For each subject i, $i = 1, \cdots, n$, we generate the covariate vector $\boldsymbol{X}_i = [X_{i1}, X_{i2}, X_{i3}]^T$ in the same way as in the simulation study in Chapter 3, where X_{i1} follows a Bernoulli distribution with parameter 0.5, $X_{i2} \sim \text{Unif}(0, 3)$ and $X_{i3} \sim \text{Unif}(0, 5)$. For regression coefficient vector $\boldsymbol{\beta} = [\beta_1, \beta_2, \beta_3]^T$, we set $\beta_1 = 1$, $\beta_2 = -0.3$ and $\beta_3 = 0.5$, which are the same as those in Chapter 3. We simulate the failure time T_i using the hazard function

$$h(t_i) = 3t_i^2 + X_{i1} - 0.3X_{i2} + 0.5X_{i3},$$

which is the AH model with the baseline hazard

$$h_0(t_i) = 3t_i^2.$$

An inversion method is applied to obtain T_i based on the relationship $u_i = F_T(t_i)$, where u_i is a standard uniform random variable, and $F_T(\cdot)$ is the distribution function of T_i given by

$$F_T(t_i) = 1 - \exp\left[- t_i^3 - (X_{i1} - 0.3X_{i2} + 0.5X_{i3})t_i \right].$$

For each subject i, independently of T_i, we generate two monitoring times in the same way as in Chapter 3, which are $C_{i1} \sim \text{Unif}(0,1)$ and $C_{i2} = C_{i1} + \text{Unif}(0,1)$. Methods to simulate left-censored data, finite interval-censored data and right-censored data in Chapter 3 are also applied here.

In discretising the baseline hazard, bins are selected in such a way that each bin contains an approximately equal number of observations, denoted by n_c. We set $n_c = 2$ for $n = 100$, $n_c = 5$ for $n = 500$, and $n_c = 8$ for $n = 1000$. The smoothing value is selected according to the method described in Chapter 3. The convergence criterion of the primal-dual interior-point algorithm is defined as when the duality measure becomes less than 10^{-5}, i.e., $\mu_k < 10^{-5}$.

For each combination of n and π_c, 300 repeated samples are generated. Based on the 300 Monte Carlo samples, we compute the average estimate (AEST), the bias (BIAS), the Monte Carlo standard deviation (MCSD), the average asymptotic standard deviation (AASD) and the mean squared error (MSE) for the estimate of $\boldsymbol{\beta}$ by the formulas (3.41)-(3.45). For the estimate of $h_0(\cdot)$, we compute its AEST, MCSD and AASD by the formulas (3.46)-(3.48) and average integrated squared error (AISE) by the formula (3.50). We assess the performance of different $\boldsymbol{\beta}$ estimates by examining the BIAS and MCSD or MSE. We assess the performance of the $h_0(\cdot)$ estimates by inspecting plots of its AEST, 95% Monte Carlo piecewise confidence interval (PWCI) and 95% asymptotic piecewise confidence interval (PWCI), and examining its AISE.

Tables 4.1-4.3 provide the MPL estimates of $\boldsymbol{\beta}$ with different sample sizes n and censoring proportions π_c. We observe that, for the same sample size, MCSD, AASD, MSE and absolute value of BIAS all decrease as censoring proportion decreases, and for the same censoring proportion, a decreasing trend for the four quantities exists with increasing sample size. Comparing MCSD against AASD indicates that the sandwich formula (4.31) generally gives accurate variance approximation for the $\boldsymbol{\beta}$ estimates, especially with large sample size.

Figures 4.1-4.3 exhibit plots for the true baseline hazard, its AESTs, the corresponding 95% Monte Carlo PWCIs and the corresponding mean of the 95% asymptotic PWCIs for different sample sizes and censoring proportions. We detect that the AESTs of $h_0(\cdot)$ become increasingly close to $h_0(\cdot)$ as the sample size increases. It is also shown that both the 95% Monte Carlo PWCIs and mean of the 95% asymptotic PWCIs become wider with increasing censoring proportion but narrower with increasing sample size. By inspecting the closeness of the 95% Monte Carlo PWCIs to the mean of 95% asymptotic PWCIs, we deduce that the formula (4.31) approximates the variance of baseline hazard estimates well, especially when the sample size is large or the censoring proportion is small. Tables 4.1-4.3 also provide the values of AISE for the MPL estimates of $h_0(\cdot)$ plotted in Figures 4.1-4.3. We observe that the values of AISE decrease with sample size but increase with censoring proportion.

The counting process (CP) method by Lin and Ying (1994) is applied to study right-censored data, rather than interval-censored data, under the AH model. However, our proposed MPL method can deal with any censoring type of data. Therefore, for comparison, we only generate right-censored data. The censoring proportion is chosen approximately as $\pi_c = 0.4$. The sample sizes are taken as $n = 100$, 500 and 1000. In our MPL method, we select equal counts of observations in each bin to be $n_c = 2$ for $n = 100$,

$n_c = 5$ for $n = 500$, and $n_c = 8$ for $n = 1000$. The covariates $\boldsymbol{X}_i = [X_{i1}, X_{i2}, X_{i3}]^T$ and regression coefficients $\boldsymbol{\beta} = [\beta_1, \beta_2, \beta_3]^T$ are the same as above. 300 Monte Carlo samples are generated for each sample size. We use the R package, ahaz, to compute estimates from the method of Lin and Ying (1994). These two methods are evaluated based on the same data sets. Results are reported in Table 4.4. We observe that, for each of the sample sizes, our MPL estimates for the three regression parameters have lower MCSDs and MSEs. By making comparisons between MCSDs and AASDs separately in these two methods, we conclude that the asymptotic standard deviation formulas from the two methods are working accurately in variance approximations. Figure 4.4 displays estimates of the baseline hazard together with 95% PWCIs. Since the asymptotic variance of the baseline hazard estimate is not available from the package ahaz, we only compute its 95% Monte Carlo PWCIs. From Figure 4.4, it can be seen clearly that our MPL method has better estimates for the baseline hazard, and the 95% Monte Carlo PWCIs are very close to the mean of 95% asymptotic PWCIs. In addition, our MPL method gives smaller values of AISE for the MPL estimates of $h_0(\cdot)$.

4.7 A real data example

In this section, we apply the primal-dual interior point algorithm to fit the AH model, using the data from the AIDS study (Lindsey and Ryan, 1998). The data is also analyzed in Chapter 3 under the PH model. Suppose that the failure times T_i, $i = 1, \cdots, 31$, can be reasonably described by the AH model. We use our MPL method to fit the model using a smoothing value of $1 - 10^{-4}$, and equal number of observations $n_c = 1$ in each bin. Based on the consistency and asymptotic normality properties of $\hat{\boldsymbol{\beta}}$, we perform hypothesis tests on $\boldsymbol{\beta}$, i.e., H_0: $\beta_j = 0$ versus H_a: $\beta_j \neq 0$ for $j = 1, 2, 3, 4$. Table 4.5 reports the MPL estimates of $\boldsymbol{\beta}$ with the corresponding asymptotic standard deviations

(astd), p-values and 95% confidence intervals. We observe that none of the covariates has significant effects on the hazard of the time to development of drug resistance (i.e., $p > 0.05$). Figure 4.5 displays the baseline survival estimate and baseline hazard estimate with its 95% asymptotic PWCI. The baseline hazard plot shows that the baseline risk of time to development peaks after a few months, then monotonically decreases until month 25, and finally falls sharply on month 26.

4.8 Conclusion

This chapter develops a maximum penalized log-likelihood (MPL) method to fit the additive hazard (AH) model with partly interval-censored failure time data, where a penalty function is included to ensure the smoothness of the estimated baseline hazard function. In the estimation procedure, the baseline hazard is assumed to be piecewise constant, and the estimates of hazard and baseline hazard are constrained to be non-negative simultaneously and directly by using the primal-dual interior point algorithm. The algorithm provides simultaneously the MPL estimates of the baseline hazard and the regression coefficient vector $\boldsymbol{\beta}$. $\sqrt{n}$-consistency and asymptotic normality of the MPL estimates are shown. The asymptotic standard deviations generally give accurate estimates of the standard deviations for the MPL estimates.

Although we assume independent censoring and time-independent covariates in developing the MPL method for the AH model, we can relax these assumptions and extend the method to cases of dependent censoring or time-dependent covariates. Our MPL method can also be adopted in fitting models which are generalizations of the AH model (4.1) in this chapter. For instance, Lin and Ying (1995) consider an additive-multiplicative hazard model which combines the PH model and the AH model together, but only right-censored data is analyzed. Analysis of interval-censored data is possible by using our method.

		n=100		
π_c		20%	50%	80%
n_c		2	2	2
$\beta_1 = 1$	AEST	0.9657	1.0368	1.1204
	BIAS	0.0343	-0.0368	-0.1204
	MCSD	0.3662	0.4301	0.4409
	AASD	0.3877	0.4176	0.4492
	MSE	(0.1353)	(0.1863)	(0.2089)
$\beta_2 = -0.3$	AEST	-0.4150	-0.4556	-0.5513
	BIAS	0.1150	0.1556	0.2513
	MCSD	0.2199	0.2191	0.2457
	AASD	0.2186	0.2333	0.2509
	MSE	(0.0616)	(0.0722)	(0.1235)
$\beta_3 = 0.5$	AEST	0.5166	0.4713	0.5373
	BIAS	-0.0166	0.0287	-0.0373
	MCSD	0.1355	0.1374	0.1409
	AASD	0.1425	0.1499	0.1609
	MSE	(0.0186)	(0.0197)	(0.0212)
$h_0(t)$	AISE	0.2612	0.2786	0.3724

Table 4.1: AEST, BIAS, MCSD, AASD and MSE for the estimate of $\hat{\boldsymbol{\beta}}$, and average integrated squared error (AISE) for the baseline hazard estimate $\hat{h}_0(t)$ with equal count in each bin $n_c = 2$, sample size $n = 100$ and censoring proportions $\pi_c = 20\%$, 50% and 80%.

		n=500		
π_c		20%	50%	80%
n_c		5	5	5
$\beta_1 = 1$	AEST	0.9881	1.0327	0.9620
	BIAS	0.0119	-0.0327	0.0380
	MCSD	0.1675	0.1848	0.1852
	AASD	0.1571	0.1746	0.1868
	MSE	(0.0282)	(0.0352)	(0.0357)
$\beta_2 = -0.3$	AEST	-0.3151	-0.3194	-0.2538
	BIAS	0.0151	0.0194	-0.0462
	MCSD	0.0970	0.1071	0.1108
	AASD	0.0852	0.0951	0.1015
	MSE	(0.0096)	(0.0119)	(0.0144)
$\beta_3 = 0.5$	AEST	0.5131	0.4812	0.4690
	BIAS	-0.0131	0.0188	0.0310
	MCSD	0.0584	0.0598	0.0553
	AASD	0.0602	0.0649	0.0681
	MSE	(0.0036)	(0.0039)	(0.0040)
$h_0(t)$	AISE	0.0714	0.0913	0.0985

Table 4.2: AEST, BIAS, MCSD, AASD and MSE for the estimate of $\hat{\boldsymbol{\beta}}$, and average integrated squared error (AISE) for the baseline hazard estimate $\hat{h}_0(t)$ with equal count in each bin $n_c = 5$, sample size $n = 500$ and censoring proportions $\pi_c = 20\%$, 50% and 80%.

		n=1000		
π_c		20%	50%	80%
n_c		8	8	8
$\beta_1 = 1$	AEST	0.9915	0.9899	0.9711
	BIAS	0.0085	0.0101	0.0289
	MCSD	0.1112	0.1248	0.1292
	AASD	0.1129	0.1215	0.1309
	MSE	(0.0124)	(0.0157)	(0.0175)
$\beta_2 = -0.3$	AEST	-0.3005	-0.2953	-0.2787
	BIAS	0.0005	-0.0047	-0.0213
	MCSD	0.0733	0.0734	0.0737
	AASD	0.0611	0.0669	0.0715
	MSE	(0.0053)	(0.0054)	(0.0059)
$\beta_3 = 0.5$	AEST	0.5073	0.4824	0.4772
	BIAS	-0.0073	0.0176	0.0228
	MCSD	0.0460	0.0461	0.0462
	AASD	0.0422	0.0449	0.0474
	MSE	(0.0022)	(0.0024)	(0.0027)
$h_0(t)$	AISE	0.0385	0.0529	0.0825

Table 4.3: AEST, BIAS, MCSD, AASD and MSE for the estimate of $\hat{\boldsymbol{\beta}}$, and average integrated squared error (AISE) for the baseline hazard estimate $\hat{h}_0(t)$ with equal count in each bin $n_c = 8$, sample size $n = 1000$ and censoring proportions $\pi_c = 20\%$, 50% and 80%.

		n=100		n=500		n=1000	
		MPL	CP	MPL	CP	MPL	CP
$\beta_1 = 1$	AEST	0.9529	0.9501	0.9667	0.9963	0.9733	0.9766
	BIAS	0.0471	0.0499	0.0333	0.0037	0.0267	0.0234
	MCSD	0.4254	0.4273	0.2370	0.2477	0.1127	0.1189
	AASD	0.4549	0.4195	0.2468	0.2348	0.1311	0.1273
	MSE	(0.1832)	(0.1851)	(0.0573)	(0.0614)	(0.0134)	(0.0147)
$\beta_2 = -0.3$	AEST	-0.4295	-0.2844	-0.3106	-0.2890	-0.3001	-0.2945
	BIAS	0.1295	-0.0156	0.0106	-0.0110	0.0001	-0.0055
	MCSD	0.2251	0.2581	0.1209	0.1255	0.0636	0.0666
	AASD	0.2514	0.2363	0.1381	0.1293	0.0730	0.0702
	MSE	(0.0674)	(0.0668)	(0.0147)	(0.0159)	(0.0040)	(0.0045)
$\beta_3 = 0.5$	AEST	0.5206	0.5186	0.4589	0.4864	0.4853	0.4862
	BIAS	-0.0206	-0.0156	0.0411	0.0136	0.0147	0.0138
	MCSD	0.1369	0.2581	0.0696	0.0822	0.0381	0.0406
	AASD	0.1775	0.2363	0.0628	0.0827	0.0496	0.0449
	MSE	(0.0192)	(0.0668)	(0.0065)	(0.0069)	(0.0017)	(0.0018)
$h_0(t)$	AISE	0.3436	1.7654	0.1507	1.7884	0.1130	1.7963

Table 4.4: Comparisons of regression coefficient estimates and the baseline hazard estimates between the MPL method and the counting process (CP) method by Lin and Ying (1994) using simulated samples with sample sizes of $n = 100$, 500 and 1000, and approximate censoring proportion of $\pi_c = 0.4$.

research/R and matlab/R for AH model Phd/100cp.eps

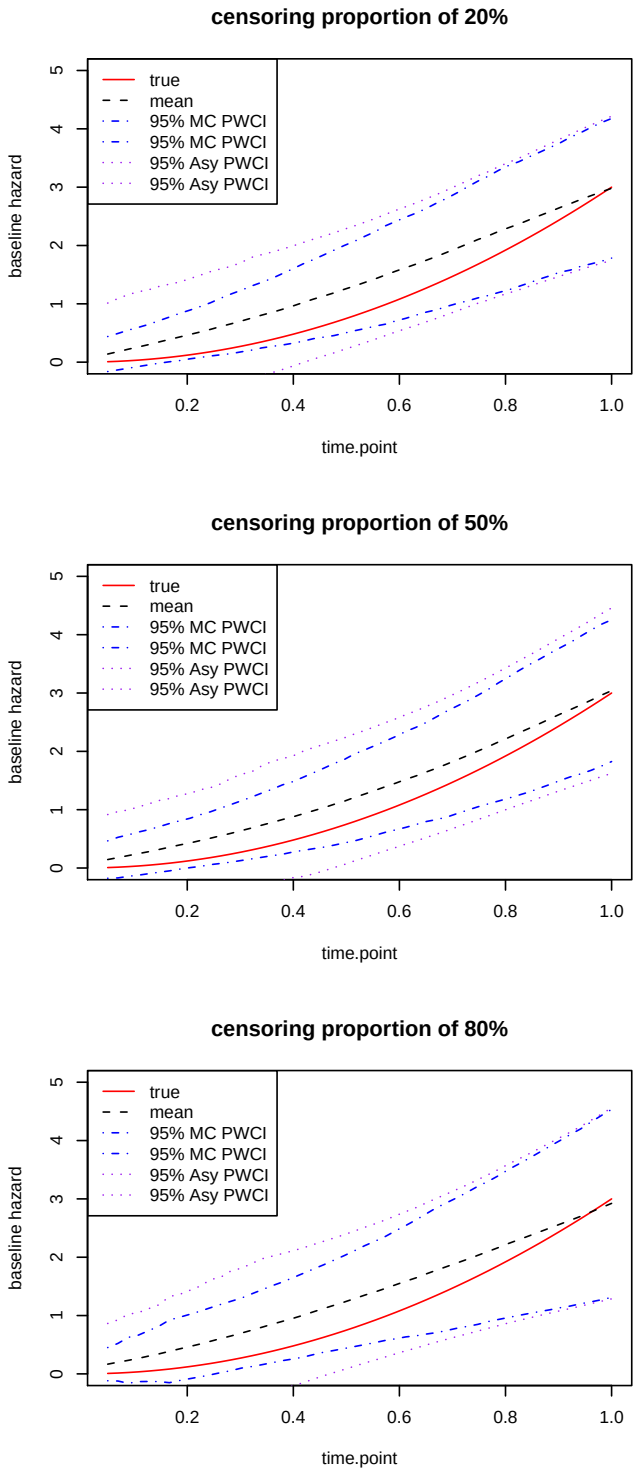

Figure 4.1: Plots of the true hazard $h_0(t)$ (solid), the average MPL estimate of $h_0(t)$ (dash), the 95% Monte Carlo PWCI (dot-dash), and the average 95% asymptotic PWCI (dots), assuming sample size $n = 100$, equal count in each bin $n_c = 2$ and censoring proportions of 20%, 50% and 80% respectively.

research/R and matlab/R for AH model Phd/500cp.eps

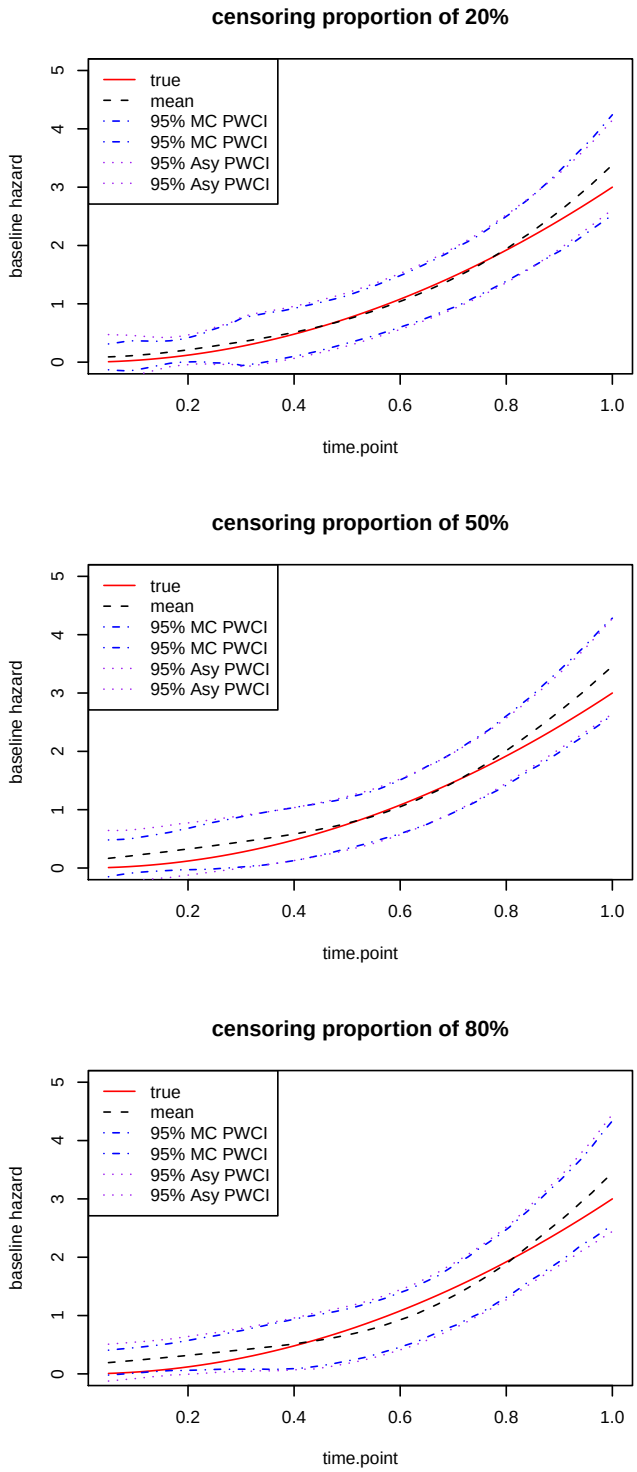

Figure 4.2: Plots of the true hazard $h_0(t)$ (solid), the average MPL estimate of $h_0(\cdot)$ (dash), the 95% Monte Carlo PWCI (dot-dash), and the average 95% asymptotic PWCI (dots), assuming sample size $n = 500$, equal count in each bin $n_c = 5$ and censoring proportions of 20%, 50% and 80% respectively.

research/R and matlab/R for AH model Phd/1000cp.eps

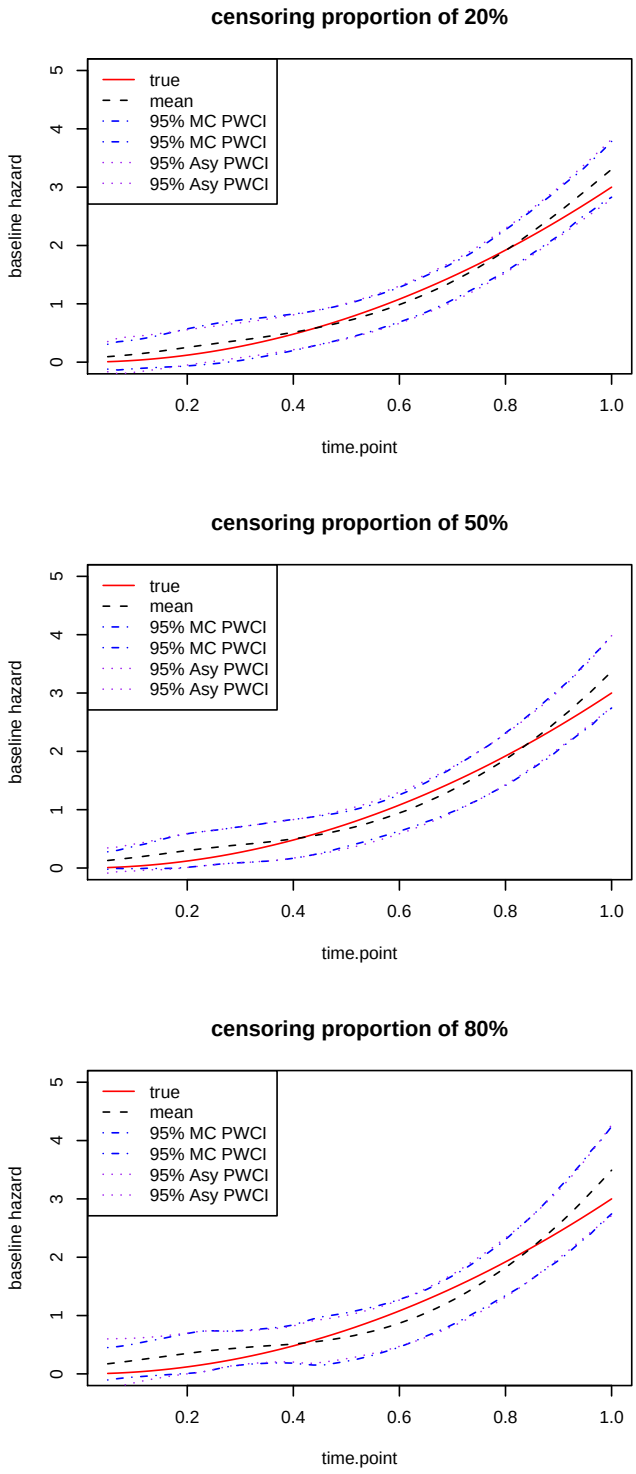

Figure 4.3: Plots of the true hazard $h_0(t)$ (solid), the average MPL estimate of $h_0(t)$ (dash), the 95% Monte Carlo PWCI (dot-dash), and the average 95% asymptotic PWCI (dots), assuming sample size $n = 1000$, equal count in each bin $n_c = 8$ and censoring proportions of 20%, 50% and 80% respectively.

research/R and matlab/R for AH model Phd/method comparison.eps

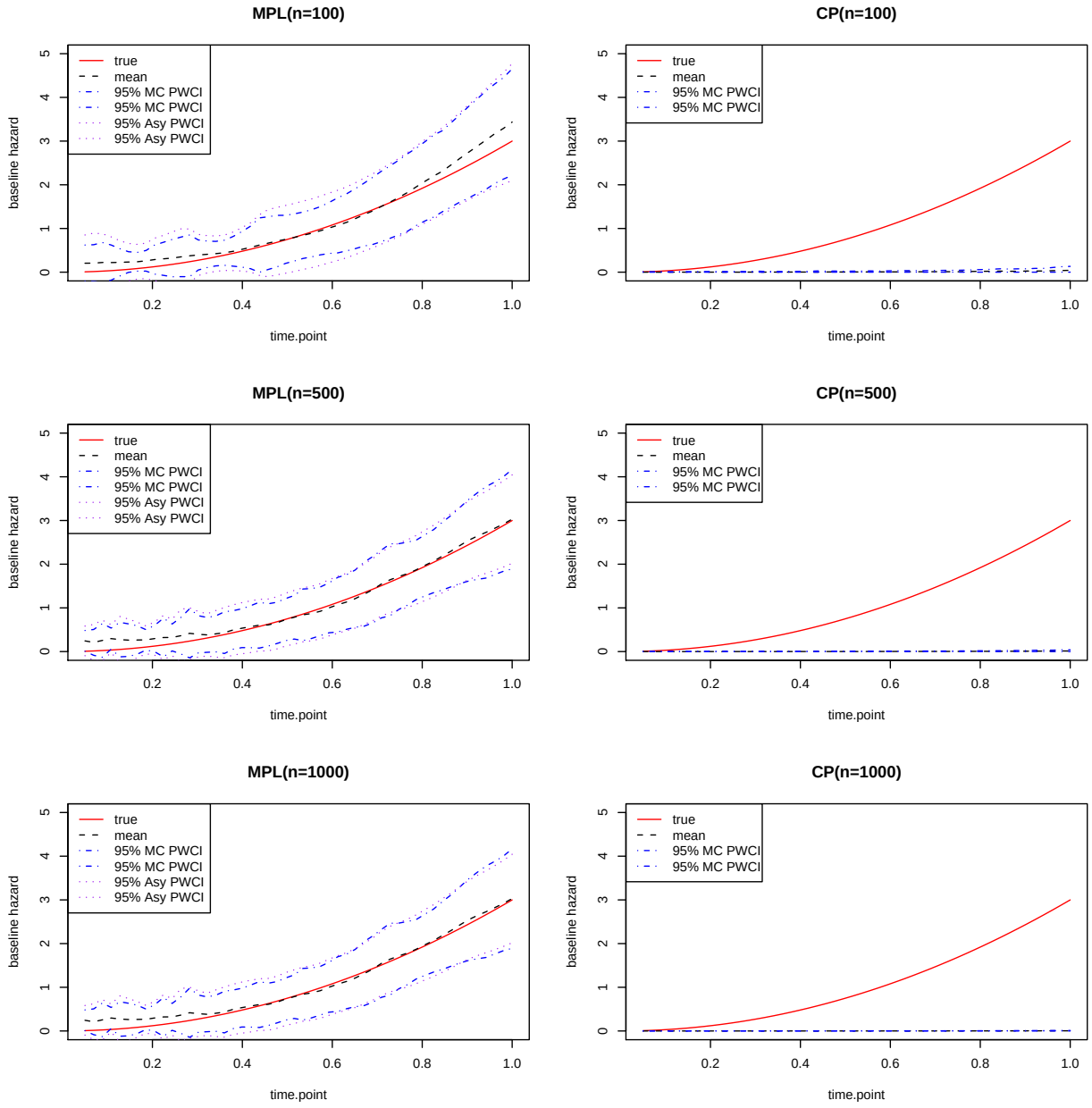

Figure 4.4: Plots of the baseline hazard estimate with its 95% PWCI for the MPL method and the counting process (CP) method, with sample sizes of $n = 100$, 500 and 1000 and censoring proportion of 40%.

research/R and matlab/R for AH model Phd/realdata.eps

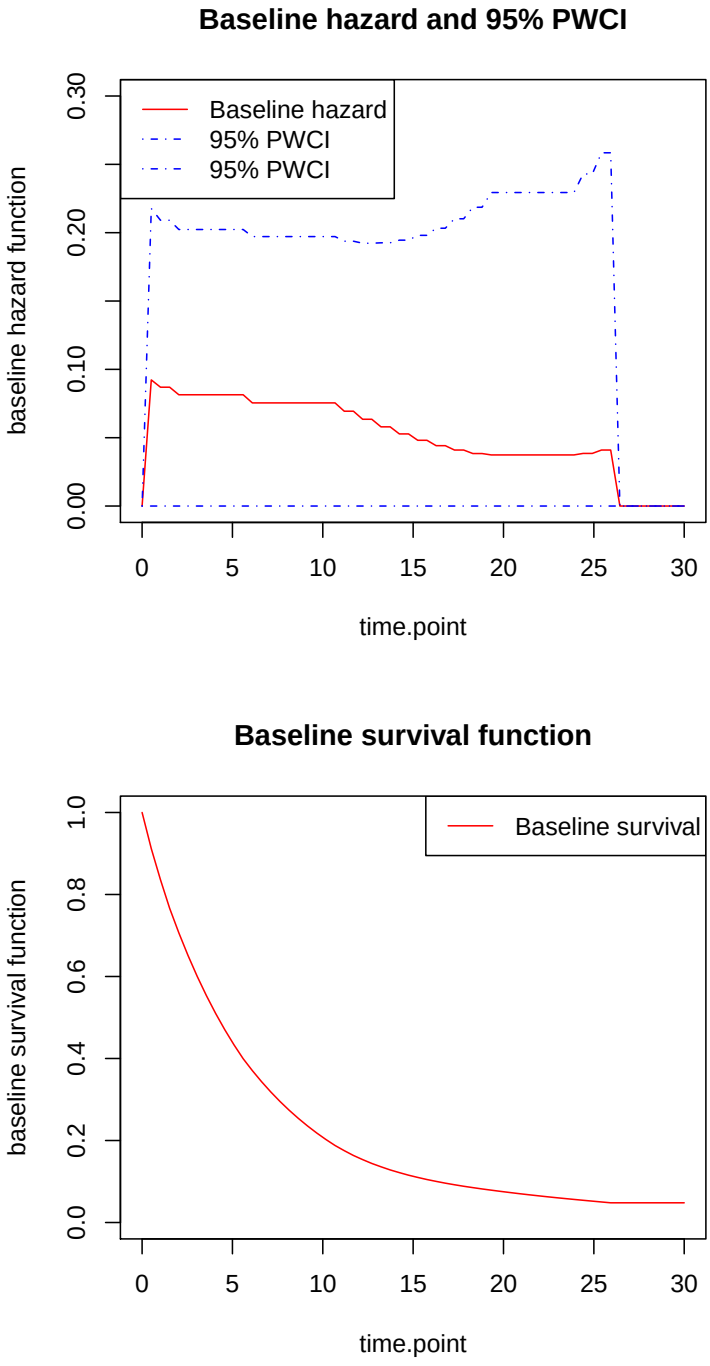

Figure 4.5: Plots of the estimates for the AIDS example, top, the baseline hazard estimate and its 95% PWCI; bottom, the baseline survival estimate.

Effects	$\hat{\boldsymbol{\beta}}$	astd	p-value	95%C.I
stage	0.0290	0.0321	0.3663	(-0.0339, 0.0919)
does	0.0000	0.0252	0.9999	(-0.0494, 0.0494)
CD4: 100-399	-0.0374	0.0626	0.5502	(-0.1601, 0.0853)
CD4: ≥ 400	-0.0410	0.0576	0.4766	(-0.1539, 0.0719)

Table 4.5: Regression coefficient estimates given by the MPL method with asymptotic standard deviations (astd), p-values and 95% confidence intervals.

Chapter 5

Accelerated Failure Time (AFT) Model with Partly Interval-Censored Failure Time Data

5.1 Introduction

In this chapter, we develop a penalized likelihood procedure to fit an accelerated failure time (AFT) model with partly interval-censored failure time data. Alternative to a PH model where the effect of covariates is assumed to be multiplicative on the hazard function, the AFT model assumes that the effect of covariates is to accelerate or decelerate the time to failure. Specifically, let T_i be the failure time and $\boldsymbol{X}_i$ the covariate vector for subject i, then the AFT model specifies that

$$\log T_i = \boldsymbol{X}_i^T \boldsymbol{\beta} + \epsilon_i, \tag{5.1}$$

where $\boldsymbol{\beta}$ is the regression coefficient vector and ϵ_i is an error variable independent of (Kalbfleisch and Prentice, 2002). The distribution of ϵ_i is unknown, hence the model (5.1) is a semi-parametric regression model. The AFT model has some advantages when

compared with the PH model: (i) it does not require the assumption of proportional hazard; and (ii) it models directly the covariate effects on the failure time, so that the interpretation of the estimate results is more easily understood in terms of effects on the mean failure time. Therefore, it is of interest to develop estimation procedures for the AFT model.

The presence of censoring causes major challenges in the semi-parametric analysis of the AFT model. However, there exist some methods to fit the AFT model with interval-censored failure time data, which have been reviewed in Section 2.3. Rabinowitz et al. (1995) and Betensky et al. (2001) propose a linear rank estimation method. This method involves estimations of the distribution function of the error variable ϵ, but the resulting distribution estimate is not guaranteed to be smooth. The major disadvantage of this method is that the implementation becomes numerically difficult for high-dimensional covariate variables. Komárek et al.(2005) develop a maximum penalized likelihood (MPL) method with partly interval-censored data. This method starts by standardizing ϵ, and then fits the AFT model by estimating the regression coefficient $\boldsymbol{\beta}$ and the density function of the standardized ϵ, where the density is approximated by a linear combination of Gaussian density functions. Let $\tilde{\epsilon}$ denote the standardized ϵ. Let $\hat{f}_{\tilde{\epsilon}}(t)$ be the estimated density of $\tilde{\epsilon}$. In the estimation procedure, there are three constraints imposed: namely (i) $\hat{f}_{\tilde{\epsilon}}(t) > 0$ and $\int \hat{f}_{\tilde{\epsilon}}(t)dt = 1$; (ii) $\mathrm{E}(\tilde{\epsilon}) = 0$; and (iii) $\mathrm{Var}(\tilde{\epsilon}) = 1$. The three constraints are imposed indirectly by transforming coefficients of the Gaussian density basis. The smoothness of $\hat{f}_{\tilde{\epsilon}}(t)$ is controlled by a penalty term based on squared second-order differences between adjacent basis coefficients. Finally, the estimates of the regression coefficients and the density of $\tilde{\epsilon}$ are obtained by using the sequential quadratic programming algorithm and the Newton algorithm.

We adopt the MPL method for the AFT model, allowing for exactly observed, left-,

right-, and finite interval-censored failure time data. In our approach, we consider $\epsilon^* = e^{\epsilon}$, and let $h_{\epsilon^*}(\cdot)$ denote the hazard function of ϵ^*. Our method differs from the method by Komárek et al.(2005) in four aspects: (i) we fit the AFT by estimating $\boldsymbol{\beta}$ and the hazard $h_{\epsilon^*}(\cdot)$, with the hazard modeled by a linear combination of Gaussian basis functions; (ii) in the estimation procedure, we only have one constraint, which is the non-negativity of the hazard estimate, and the constraint is imposed in a direct way by imposing non-negativity on the Gaussian basis coefficients; (iii) we guarantee the smoothness of the hazard estimate by a roughness penalty function, which involves its second derivative; and (iv) we obtain the MPL estimators of $\boldsymbol{\beta}$ and $h_{\epsilon^*}(\cdot)$ simultaneously by the Newton-MI algorithm, which has been used in Chapter 3. We show that, under certain conditions, the MPL estimators are asymptotically consistent and multivariate normal.

The chapter is organized as follows. In Section 5.2, we construct a penalized log-likelihood function with partly interval-censored data under the AFT model. In section 5.3, we develop the Newton-MI algorithm to compute the MPL estimators, and study the convergence of the algorithm. In Section 5.4, we present asymptotic results of the MPL estimators. In Section 5.5, we investigate the performance of our proposed method by simulation studies. In Section 5.6, we report results from real data analysis. Finally, conclusions and some discussions are given in Section 5.7.

5.2 Penalized log-likelihood functions under the AFT model

Consider a random sample of n subjects. For each subject i, $i = 1, \cdots, n$, its observed data is denoted by $(t_{iO}, \boldsymbol{X}_{iO})$ or $(t_{iL}, \boldsymbol{X}_{iL})$ or $(t_{iIL}, t_{iIR}, \boldsymbol{X}_{iI})$ or $(t_{iR}, \boldsymbol{X}_{iR})$ depending on the censoring type of its failure time. These formulations for the observed data have been

used in Chapter 3. The AFT model specifies a linear relationship between the logarithm of the failure time T_i and the covariate vector $\boldsymbol{X}_i$ in the form of $\log T_i = \boldsymbol{X}_i^T\beta + \epsilon_i$, where $\boldsymbol{X}_i$ is assumed to be time-independent. Recall that $\epsilon^* = e^{\epsilon}$ and $h_{\epsilon^*}(\cdot)$ is the hazard function of ϵ^*. Let $H_{\epsilon^*}(\cdot)$ be the cumulative hazard function of ϵ^*. Let $h(\cdot|\boldsymbol{X}_i)$ and $H(\cdot|\boldsymbol{X}_i)$ denote the conditional hazard and cumulative hazard functions of the failure time T_i given $\boldsymbol{X}_i$. In Section 1.4.3, we have obtained the relationship between the hazard function of T_i and the hazard function of ϵ^*,

$$h(t|\boldsymbol{X}_i) = e^{-\boldsymbol{X}_i^T\boldsymbol{\beta}} h_{\epsilon^*}(te^{-\boldsymbol{X}_i^T\boldsymbol{\beta}}). \tag{5.2}$$

Based on (5.2), the cumulative hazard function of T_i is given by

$$H(t|\boldsymbol{X}_i) = \int_0^t h(s|\boldsymbol{X}_i)ds = H_{\epsilon^*}(te^{-\boldsymbol{X}_i^T\boldsymbol{\beta}}), \tag{5.3}$$

where $H_{\epsilon^*}(t) = \int_0^t h_{\epsilon^*}(s)ds$. By letting $\boldsymbol{X}_i = \mathbf{0}_{p\times 1}$ in equation (5.2), we get a direct relationship between the baseline hazard function of T_i and the hazard function of ϵ^*, that is $h_0(t) = h_{\epsilon^*}(t)$. Therefore, equation (5.2) can be rewritten as $h(t|\boldsymbol{X}_i) = e^{-\boldsymbol{X}_i^T\boldsymbol{\beta}} h_0(te^{-\boldsymbol{X}_i^T\boldsymbol{\beta}})$, and estimating $h_0(\cdot)$ is equivalent to estimating $h_{\epsilon^*}(\cdot)$. We fit the AFT model by estimating the regression coefficient vector $\boldsymbol{\beta}$ and the hazard function $h_{\epsilon^*}(\cdot)$.

We start to fit the AFT model by approximating the hazard function $h_{\epsilon^*}(\cdot)$. The approximation is given by

$$h_{\epsilon^*}(t) = \sum_{j=1}^{m} \theta_j \cdot \varphi(t; \mu_j, \sigma_j^2), \tag{5.4}$$

where $\varphi(t; \mu_j, \sigma_j^2) = e^{-\frac{(t-\mu_j)^2}{2\sigma_j^2}}$ is an unscaled Gaussian basis function, $\{\mu_j : j = 1, \cdots, m\}$ are knots satisfying $\mu_1 < \cdots < \mu_m$, m is the number of knots, $\{\sigma_j : j = 1, \cdots, m\}$ are non-negative parameters, and $\{\theta_j : j = 1, \cdots, m\}$ are basis coefficients. We write $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_m]^T$. Values of $\{\mu_1, \cdots, \mu_m\}$ and $\{\sigma_1, \cdots, \sigma_m\}$ are fixed by design and will be determined in Section 5.5. The corresponding cumulative hazard function of ϵ^* is

$$H_{\epsilon^*}(t) = \int_0^t h_{\epsilon^*}(s)ds = \sqrt{2\pi}\sum_{j=1}^{m} \theta_j\sigma_j \Big[\Phi(t; \mu_j, \sigma_j^2) - \Phi(0; \mu_j, \sigma_j^2)\Big], \tag{5.5}$$

where $\Phi(\cdot;\mu_j,\sigma_j^2)$ is the cumulative distribution function of the normal distribution with mean μ_j and variance σ_j^2. For simplicity, we write $\varphi_j(\cdot) = \varphi(\cdot;\mu_j,\sigma_j^2)$ and $\Phi_j(\cdot) = \Phi(\cdot;\mu_j,\sigma_j^2)$. From equations (5.2), (5.3), (5.4) and (5.5), the hazard, cumulative hazard and survival functions of the failure time T_i concerning the model parameters $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ are expressed respectively as

$$h(t|\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{X}_i) = e^{-\boldsymbol{X}_i^T\boldsymbol{\beta}}\sum_{j=1}^{m}\theta_j\cdot\varphi_j(te^{-\boldsymbol{X}_i^T\boldsymbol{\beta}}), \tag{5.6}$$

$$H(t|\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{X}_i) = \sqrt{2\pi}\sum_{j=1}^{m}\theta_j\sigma_j\Big[\Phi_j(te^{-\boldsymbol{X}_i^T\boldsymbol{\beta}}) - \Phi_j(0)\Big], \tag{5.7}$$

and

$$S(t|\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{X}_i) = \exp\Big\{-\sqrt{2\pi}\sum_{j=1}^{m}\theta_j\sigma_j\Big[\Phi_j(te^{-\boldsymbol{X}_i^T\boldsymbol{\beta}}) - \Phi_j(0)\Big]\Big\}. \tag{5.8}$$

We make the standard assumptions that the censoring time is independent of the failure time conditional on covariates, and that the distribution of the censoring time does not involve $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$. Then, using equations (5.6)-(5.8), the log-likelihood function involving $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ is

$$\begin{aligned}\ell(\boldsymbol{\beta},\boldsymbol{\theta}) &= \sum_{i\in\mathcal{L}}\log\Big[1 - S(t_{iL}|\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{X}_{iL})\Big] - \sum_{i\in\mathcal{R}}H(t_{iR}|\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{X}_{iR})\\ &\quad+\sum_{i\in\mathcal{I}}\log\Big[S(t_{iIL}|\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{X}_{iI}) - S(t_{iIR}|\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{X}_{iI})\Big]\\ &\quad+\sum_{i\in\mathcal{O}}\Big[\log\Big(h(t_{iO}|\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{X}_{iO})\Big) - H(t_{iO}|\boldsymbol{\beta},\boldsymbol{\theta},\boldsymbol{X}_{iO})\Big].\end{aligned} \tag{5.9}$$

Note that the first sum term is the log-likelihood for left-censored observations, the second for right-censored observations, the third for finite interval-censored observations and the fourth for exactly observed failure time observations. To guarantee smoothness of the estimate of $h_{\epsilon^*}(\cdot)$, we subtract a penalty term $J(h_{\epsilon^*})$, which is a function of $h_{\epsilon^*}(\cdot)$, from the log-likelihood $\ell(\boldsymbol{\beta},\boldsymbol{\theta})$, resulting in a penalized log-likelihood function,

$$\ell_P(\boldsymbol{\beta},\boldsymbol{\theta}) = \ell(\boldsymbol{\beta},\boldsymbol{\theta}) - \tilde{\gamma}J(h_{\epsilon^*}), \tag{5.10}$$

where $\tilde{\gamma}$ is a smoothing parameter that controls balance between smoothness of the estimated hazard and fit to the data. The optimal smooth value of $\tilde{\gamma}$ can be selected using the method similar to that developed in Section 3.4. We assume that the hazard function $h_{\epsilon^*}(\cdot)$ is continuous and twice differentiable, and that its second derivative is square integrable. Then the smooth aspect of $h_{\epsilon^*}(\cdot)$ can be related to the value of its second derivative, which leads to a roughness penalty given by

$$J(h_{\epsilon^*}) = \int_{\mathcal{I}(\boldsymbol{\beta})} [h''_{\epsilon^*}(s)]^2 ds. \tag{5.11}$$

In equation (5.11), $\mathcal{I}(\boldsymbol{\beta})$ is an interval varying with the value of $\boldsymbol{\beta}$ and defined by $\mathcal{I}(\boldsymbol{\beta}) = [t_{\min}(\boldsymbol{\beta}), t_{\max}(\boldsymbol{\beta})]$, where $t_{\min}(\boldsymbol{\beta}) = \min\{t_{ic}e^{-\boldsymbol{X}_{ic}^T\boldsymbol{\beta}} : i = 1, \cdots, n\}$, $t_{\max}(\boldsymbol{\beta}) = \max\{t_{ic}e^{-\boldsymbol{X}_{ic}^T\boldsymbol{\beta}} : i = 1, \cdots, n\}$ and $c = \{L, R, IL, IR, O\}$. The penalty term (5.11) can also be expressed in a quadratic form, i.e., $J(h_{\epsilon^*}) = \boldsymbol{\theta}^T\boldsymbol{R}\boldsymbol{\theta}$, where $\boldsymbol{R}$ is a $m \times m$ symmetric matrix with elements $r_{uv} = \int_{\mathcal{I}(\boldsymbol{\beta})} \varphi''_u(s)\varphi''_v(s)ds$, and

$$\varphi''_u(s) = e^{-\frac{(s-\mu_u)^2}{2\sigma_u^2}} \Big[(\frac{s-\mu_u}{\sigma_u^2})^2 - \frac{1}{\sigma_u^2}\Big].$$

5.3 Constrained optimization by the Newton-MI algorithm

We fit the AFT model by estimating the regression coefficient vector $\boldsymbol{\beta}$ and the basis coefficient vector $\boldsymbol{\theta}$. Since the basis functions $\{\varphi_j(\cdot) : j = 1, \cdots, m\}$, given in (5.4), are non-negative, it suffices to impose the non-negativity constraint on $\boldsymbol{\theta}$ to guarantee that the estimate of $h_{\epsilon^*}(\cdot)$ is non-negative. The problem can now be stated as maximizing the penalized log-likelihood function $\ell_P(\boldsymbol{\beta}, \boldsymbol{\theta})$, given in (5.10), with respect to $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$, with $\boldsymbol{\theta}$ non-negative, namely

$$(\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\theta}}) = \arg\max_{\boldsymbol{\beta},\boldsymbol{\theta}} \{\ell_P(\boldsymbol{\beta}, \boldsymbol{\theta})\} \tag{5.12}$$

subject to $\boldsymbol{\theta} \geq \mathbf{0}_{m\times 1}$.

Similar to Chapter 3, we use the Newton-MI algorithm to solve the constrained optimization problem (5.12). To describe the algorithm in the present setting, we first introduce some notations. For simplicity in deriving the first and second derivatives of $\ell_P(\boldsymbol{\beta}, \boldsymbol{\theta})$, we denote $S(\cdot)$ to be $S(\cdot|\boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{X}_i)$ defined in (5.8). We define

$$t_{ic}^* = t_{ic} \exp\left\{-\boldsymbol{X}_{ic}^T \boldsymbol{\beta}\right\},$$

$$\bar{\eta}(t_{ic}^*) = h_{\epsilon^*}(t_{ic}^*) t_{ic}^*,$$

$$\xi_{iI} = \bar{\eta}(t_{iIL}^*) S(t_{iIL}) - \bar{\eta}(t_{iIR}^*) S(t_{iIR}),$$

$$w_j(t_{ic}^*) = \frac{\theta_j (t_{ic}^* - \mu_j)}{\sigma_j^2},$$

$$A(t_{ic}^*) = \sum_{j=1}^{m} w_j(t_{ic}^*) \varphi_j(t_{ic}^*),$$

$$\rho_{ic,j} = \sqrt{2\pi} S(t_{ic}) \sigma_j \Big[\Phi_j(t_{ic}^*) - \Phi_j(0)\Big],$$

$$q_{iI} = \bar{\eta}(t_{iIL}^*)^2 S(t_{iIL}) - \bar{\eta}(t_{iIR}^*)^2 S(t_{iIR}),$$

$$\bar{H}(t_{ic}^*) = t_{ic}^* h_{\epsilon^*}(t_{ic}^*) - (t_{ic}^*)^2 A(t_{ic}^*),$$

$$\bar{h}_{iI} = \bar{H}(t_{iIL}^*) S(t_{iIL}) - \bar{H}(t_{iIR}^*) S(t_{iIR}),$$

$$z_j(t_{iO}^*) = \frac{t_{iO}^* - \mu_j}{\sigma_j^2} w_j(t_{iO}^*),$$

$$a_j = \frac{\theta_j}{\sigma_j^2},$$

$$B(t_{iO}^*) = \sum_{j=1}^{m} z_j(t_{iO}^*) \varphi_j(t_{iO}^*),$$

$$C(t_{iO}^*) = \sum_{j=1}^{m} a_j \varphi_j(t_{iO}^*),$$

$$\kappa_{1u}(\boldsymbol{\beta}, \boldsymbol{\theta}) = \sum_{i\in\mathcal{L}} \left[\frac{\rho_{iL,u}}{1 - S(t_{iL})}\right] + \sum_{i\in\mathcal{I}} \left[\frac{\rho_{iIR,u}}{S(t_{iIL}) - S(t_{iIR})}\right] + \sum_{i\in\mathcal{O}} \left[\frac{\varphi_u(t_{iO}^*)}{h_{\epsilon^*}(t_{iO}^*)}\right]$$

and

$$\kappa_{2u}(\boldsymbol{\beta},\boldsymbol{\theta}) = \sum_{i\in\mathcal{I}}\left[\frac{\rho_{iIL,u}}{S(t_{iIL})-S(t_{iIR})}\right]+\sqrt{2\pi}\sigma_u\sum_{i\in\mathcal{R}}\Big[\Phi_u(t^*_{iR})-\Phi_u(0)\Big] +\sqrt{2\pi}\sigma_u\sum_{i\in\mathcal{O}}\Big[\Phi_u(t^*_{iO})-\Phi_u(0)\Big].$$

Then the first derivative of $\ell_P(\boldsymbol{\beta},\boldsymbol{\theta})$ with respect to $\boldsymbol{\beta}$ is

$$\frac{\partial\ell_P(\boldsymbol{\beta},\boldsymbol{\theta})}{\partial\boldsymbol{\beta}} = \boldsymbol{X}_L^T\mathcal{M}_1\mathbf{1}_{n_L}+\boldsymbol{X}_I^T\mathcal{M}_2\mathbf{1}_{n_I}+\boldsymbol{X}_R^T\mathcal{M}_3\mathbf{1}_{n_R}+\boldsymbol{X}_O^T\mathcal{M}_4\mathbf{1}_{n_O}, \tag{5.13}$$

where matrices $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$ and $\mathcal{M}_4$ are diagonal matrices defined by

$$\mathcal{M}_1 = \mathrm{diag}\left\{\frac{-\bar\eta(t^*_{iL})S(t_{iL})}{1-S(t_{iL})}\right\},$$

$$\mathcal{M}_2 = \mathrm{diag}\left\{\frac{\xi_{iI}}{S(t_{iIL})-S(t_{iIR})}\right\},$$

$$\mathcal{M}_3 = \mathrm{diag}\Big\{\bar\eta(t^*_{iR})\Big\}$$

and

$$\mathcal{M}_4 = \mathrm{diag}\left\{-1+\frac{t^*_{iO}A(t^*_{iO})}{h_{\epsilon^*}(t^*_{iO})}+\bar\eta(t^*_{iO})\right\}.$$

The Hessian matrix of $\boldsymbol{\beta}$ is

$$\frac{\partial^2\ell_P(\boldsymbol{\beta},\boldsymbol{\theta})}{\partial\beta\partial\beta^T} = \boldsymbol{X}_L^T\mathcal{M}_5\boldsymbol{X}_L+\boldsymbol{X}_I^T\mathcal{M}_6\boldsymbol{X}_I-\boldsymbol{X}_R^T\mathcal{M}_7\boldsymbol{X}_R+\boldsymbol{X}_O^T\mathcal{M}_8\boldsymbol{X}_O, \tag{5.14}$$

where matrices $\mathcal{M}_5$, $\mathcal{M}_6$, $\mathcal{M}_7$ and $\mathcal{M}_8$ are diagonal matrices defined by

$$\mathcal{M}_5 = \mathrm{diag}\left\{\frac{[\bar H(t^*_{iL})-\bar\eta(t^*_{iL})^2]S(t_{iL})}{1-S(t_{iL})}-\Big[\frac{\bar\eta(t^*_{iL})S(t_{iL})}{1-S(t_{iL})}\Big]^2\right\},$$

$$\mathcal{M}_6 = \mathrm{diag}\left\{\frac{-\bar h_{iI}+q_{iI}}{S(t_{iIL})-S(t_{iIR})}-\Big[\frac{\xi_{iI}}{S(t_{iIL})-S(t_{iIR})}\Big]^2\right\},$$

$$\mathcal{M}_7 = \mathrm{diag}\Big\{\bar H(t^*_{iR})\Big\}$$

and

$$\mathcal{M}_8 = \mathrm{diag}\left\{\frac{-t^*_{iO}A(t^*_{iO})+(t^*_{iO})^2(B(t^*_{iO})-C(t^*_{iO}))}{h_{\epsilon^*}(t^*_{iO})}-\Big[\frac{t^*_{iO}A(t^*_{iO})}{h_{\epsilon^*}(t^*_{iO})}\Big]^2-\bar H(t^*_{iO})\right\}.$$

The first derivative of $\ell_P(\boldsymbol{\beta}, \boldsymbol{\theta})$ with respect to θ_j, $j = 1, \cdots, m$, is

$$\begin{aligned}\frac{\partial \ell_P(\boldsymbol{\beta}, \boldsymbol{\theta})}{\partial \theta_j} &= \sum_{i \in \mathcal{L}} \left[\frac{\rho_{iL,j}}{1 - S(t_{iL})} \right] + \sum_{i \in \mathcal{I}} \left[\frac{\rho_{iIR,j} - \rho_{iIL,j}}{S(t_{iIL}) - S(t_{iIR})} \right] \\ &\quad - \sqrt{2\pi}\sigma_j \sum_{i \in \mathcal{R}} \left[\Phi_j(t^*_{iR}) - \Phi_j(0) \right] \\ &\quad + \sum_{i \in \mathcal{O}} \left[\frac{\varphi_j(t^*_{iO})}{h_{\epsilon^*}(t^*_{iO})} - \sqrt{2\pi}\sigma_j(\Phi_j(t^*_{iO}) - \Phi_j(0)) \right] - 2\tilde{\gamma}\boldsymbol{R}_j\boldsymbol{\theta}, \end{aligned} \tag{5.15}$$

where $\boldsymbol{R}_j$ is the jth row of matrix $\boldsymbol{R}$. The KKT necessary conditions for the constrained MPL estimations of $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ are

$$\frac{\partial \ell_P(\boldsymbol{\beta}, \boldsymbol{\theta})}{\partial \beta_j} = 0 \quad \text{for } j = 1, \cdots, p \tag{5.16}$$

and

$$\frac{\partial \ell_P(\boldsymbol{\beta}, \boldsymbol{\theta})}{\partial \theta_u} \begin{cases} = 0 & \text{if } \theta_u > 0 \\ < 0 & \text{if } \theta_u = 0 \end{cases} \tag{5.17}$$

for $u = 1, \cdots, m$.

We obtain the MPL estimators of $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ by solving equations (5.16) and (5.17) iteratively using the Newton-MI algorithm. In this algorithm, $\boldsymbol{\beta}$ is updated using the Newton-Raphson algorithm, while $\boldsymbol{\theta}$ is updated, due to its positivity constraint, by a Multiplicative Iterative (MI) algorithm of Ma (2010). Details of the algorithm are provided below:

1. Choose initial values of $\boldsymbol{\beta}^{(0)}$ and $\boldsymbol{\theta}^{(0)}$, with $\theta_j^{(0)} > 0$ for $j = 1, \cdots, m$.

2. Let $\boldsymbol{\beta}^{(k)}$ and $\boldsymbol{\theta}^{(k)}$ be the estimates of $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ at iteration k. With fixed $\boldsymbol{\theta}^{(k)}$, $\boldsymbol{\beta}^{(k)}$ is updated by

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \alpha^{(k)} \left[\frac{\partial^2 \ell_P(\boldsymbol{\beta}^{(k)}, \boldsymbol{\theta}^{(k)})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} \right]^{-1} \frac{\partial \ell_P(\boldsymbol{\beta}^{(k)}, \boldsymbol{\theta}^{(k)})}{\partial \boldsymbol{\beta}}, \tag{5.18}$$

   where $\alpha^{(k)}$ is a line search step determined by the Armijo's rule, similar to Chapter 3.

3. By fixing $\boldsymbol{\beta}$ at $\boldsymbol{\beta}^{(k+1)}$, $\boldsymbol{\theta}^{(k)}$ is obtained by running one iteration of the MI algorithm.

$$\theta_u^{(k+1/2)} = \theta_u^{(k)} \cdot \frac{\kappa_{1u}(\boldsymbol{\beta}^{(k+1)}, \boldsymbol{\theta}^{(k)}) - 2\tilde{\gamma}[\boldsymbol{R}_u\boldsymbol{\theta}^{(k)}]^- + \nu_u}{\kappa_{2u}(\boldsymbol{\beta}^{(k+1)}, \boldsymbol{\theta}^{(k)}) + 2\tilde{\gamma}[\boldsymbol{R}_u\boldsymbol{\theta}^{(k)}]^+ + \nu_u} \tag{5.19}$$

and

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \omega^{(k)}(\boldsymbol{\theta}^{(k+1/2)} - \boldsymbol{\theta}^{(k)}), \tag{5.20}$$

where ν_u is a small constant (e.g., 10^{-5}) included to avoid a zero denominator, which will not affect where the MI algorithm converges to, $\omega^{(k)}$ is the step size determined by the Armijo's rule, similar to Chapter 3.

4. Go to Step 2 and repeat the process until both $\boldsymbol{\beta}^{(k)}$ and $\boldsymbol{\theta}^{(k)}$ satisfy certain termination criteria, such as $\max_j |\beta_j^{(k+1)} - \beta_j^{(k)}| < 10^{-5}$ and $\max_u |\theta_u^{(k+1)} - \theta_u^{(k)}| < 10^{-5}$.

Following the same arguments as those in Chapter 3, we can show that the sequence $\left\{\boldsymbol{\beta}^{(k)}, \boldsymbol{\theta}^{(k)}\right\}$ generated by the Newton-MI algorithm converges to a solution satisfying the KKT conditions defined in (5.16) and (5.17).

5.4 Asymptotic properties

Asymptotic properties for the MPL estimator of $\boldsymbol{\eta} = [\boldsymbol{\beta}^T, \boldsymbol{\theta}^T]^T$ are analyzed in the same ways as those in Chapter 3. Define $\boldsymbol{\Gamma}$ as parameter space of $\boldsymbol{\eta}$. Let $\boldsymbol{\eta}_0 = [\boldsymbol{\beta}_0^T, \boldsymbol{\theta}_0^T]^T$ be the true model parameter. Let $\hat{\boldsymbol{\eta}} = [\hat{\boldsymbol{\beta}}^T, \hat{\boldsymbol{\theta}}^T]$ be the MPL estimator of $\boldsymbol{\eta}$. In analyzing the asymptotic properties, we fix the dimension of $\boldsymbol{\theta}$. For convenience, we use the notation for observations $\{(L_i, R_i], \boldsymbol{X}_i, \xi_i : i = 1, \cdots, n\}$, which has been used in Chapter 3. We denote the penalized log-likelihood function (5.10) as $\ell_{P,n}(\boldsymbol{\eta})$ to reflect its dependence on n, and rewrite $\ell_{P,n}(\boldsymbol{\eta})$ as $\ell_{P,n}(\boldsymbol{\eta}) = \sum_{i=1}^n \{\ell_i(\boldsymbol{\eta}) - \tilde{\gamma}_n J(\boldsymbol{\eta})\}$, where $J(\boldsymbol{\eta}) = J(h_{\epsilon^*})$ and $\tilde{\gamma}_n = \tilde{\gamma}/n$. Here we have $\ell_i(\boldsymbol{\eta}) = \log g(L_i, R_i, \xi_i|\boldsymbol{\eta})$, where $g(L_i, R_i, \xi_i|\boldsymbol{\eta})$ denotes the density function for $\{(L_i, R_i], \boldsymbol{X}_i, \xi_i\}$. We assume that both the censoring time and covariate variable are

independent of failure time, and their distribution functions are free of $\boldsymbol{\eta}$. Then the density function $g(L_i, R_i, \xi_i|\boldsymbol{\eta})$ is given by $g(L_i, R_i, \xi_i|\boldsymbol{\eta}) = [S(L_i|\boldsymbol{\eta}) - S(R_i|\boldsymbol{\eta})]^{1-\xi_i} f(L_i|\boldsymbol{\eta})^{\xi_i}$, where $S(t|\boldsymbol{\eta})$ is the survival function of the failure time given in (5.8), and $f(t|\boldsymbol{\eta})$ is the density function given by $f(t|\boldsymbol{\eta}) = h(t|\boldsymbol{\eta})S(t|\boldsymbol{\eta})$ with $h(t|\boldsymbol{\eta})$ given by (5.6).

In order to develop the asymptotic properties of the MPL estimate $\hat{\boldsymbol{\eta}}$, we require additional assumptions similar to those in Section 3.5 on the model parameter space, penalized log-likelihood function and penalty function.

Assumptions:

5.1 The parameter space $\boldsymbol{\Gamma}$ of $\boldsymbol{\eta}$ is compact.

5.2 There exists a measurable function $m(L_i, R_i, \boldsymbol{X}_i, \xi_i)$ with $E[m(L_i, R_i, \boldsymbol{X}_i, \xi_i)] < \infty$, which satisfies $|\log g(L_i, R_i, \xi_i|\boldsymbol{\eta})| \leq m(L_i, R_i, \boldsymbol{X}_i, \xi_i)$ for all $\boldsymbol{\eta} \in \boldsymbol{\Gamma}$.

5.3 $E_{\boldsymbol{\eta}_0}[n^{-1}\ell_{P,n}(\boldsymbol{\eta})]$ exists and has a unique maximum at $\boldsymbol{\eta}^* \in \text{int}(\boldsymbol{\Gamma})$, where $\boldsymbol{\eta}^*$ is not necessarily equal to $\boldsymbol{\eta}_0$ due to the penalty term.

5.4 $\ell_{P,n}(\boldsymbol{\eta})$ is continuous over $\boldsymbol{\Gamma}$ and is twice differentiable in a neighborhood of $\boldsymbol{\eta}^*$. The matrices

$$G_0(\boldsymbol{\eta}) = E_{\boldsymbol{\eta}_0}\left[n^{-1}\frac{\partial \ell_{P,n}(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}}\frac{\partial \ell_{P,n}(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}^T}\right]$$

and

$$F_0(\boldsymbol{\eta}) = -E_{\boldsymbol{\eta}_0}\left[n^{-1}\frac{\partial^2 \ell_{P,n}(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}\partial \boldsymbol{\eta}^T}\right]$$

exist and are positive definite in a neighborhood of $\boldsymbol{\eta}^*$.

5.5 The penalty function $J(\boldsymbol{\eta})$ is continuous and bounded over $\boldsymbol{\Gamma}$. Both $\partial J(\boldsymbol{\eta})/\partial \boldsymbol{\eta}$ and $\partial^2 J(\boldsymbol{\eta})/\partial \boldsymbol{\eta}\partial \boldsymbol{\eta}^T$ exist for all $\boldsymbol{\eta} \in \boldsymbol{\Gamma}$, and $\partial^2 J(\boldsymbol{\eta})/\partial \boldsymbol{\eta}\partial \boldsymbol{\eta}^T$ is bounded in a neighborhood of $\boldsymbol{\eta}^*$.

Theorem 5.1. *Assume that Assumptions 5.1-5.5 are satisfied, when $n \to \infty$, the MPL estimator $\hat{\boldsymbol{\eta}}$ is consistent for $\boldsymbol{\eta}^*$ and the distribution of $\sqrt{n}(\hat{\boldsymbol{\eta}} - \boldsymbol{\eta}^*)$ converges to the multivariate normal distribution $N(\mathbf{0}_{(p+m)\times 1}, V(\boldsymbol{\eta}^*))$, where*

$$V(\boldsymbol{\eta}^*) = F_0(\boldsymbol{\eta}^*)^{-1} G_0(\boldsymbol{\eta}^*) F_0(\boldsymbol{\eta}^*)^{-1}. \tag{5.21}$$

In practice, we can estimate the sandwich formula (5.21) consistently by replacing $F_0(\boldsymbol{\eta}^*)$ and $G_0(\boldsymbol{\eta}^*)$ by their empirical versions, with $\boldsymbol{\eta}^*$ replaced by $\hat{\boldsymbol{\eta}}$ due to its consistency property.

We denote the MPL estimator of $h_{\epsilon^*}(t)$ at time t by $\hat{h}_{\epsilon^*}(t)$. We define a vector $\boldsymbol{W}(t) = [\varphi_1(t), \cdots, \varphi_m(t)]^T$. From (5.4), we can write $\hat{h}_{\epsilon^*}(t) = \boldsymbol{W}(t)^T \hat{\boldsymbol{\theta}}$ and $h^*_{\epsilon^*}(t) = \boldsymbol{W}(t)^T \boldsymbol{\theta}^*$. Define a function $K_t(\boldsymbol{\theta}) = \boldsymbol{W}(t)^T \boldsymbol{\theta}$, then we have $K_t(\hat{\boldsymbol{\theta}}) = \hat{h}_{\epsilon^*}(t)$ and $K_t(\boldsymbol{\theta}^*) = h^*_{\epsilon^*}(t)$. Since the function $K_t(\boldsymbol{\theta})$ is a continuously differentiable function of $\boldsymbol{\theta}$ with $dK_t(\boldsymbol{\theta})/d\boldsymbol{\theta} = \boldsymbol{W}(t)$, according to Theorem 5.3 and the Delta Theorem, we have

$$\sqrt{n}(\hat{h}_{\epsilon^*}(t) - h^*_{\epsilon^*}(t)) \xrightarrow{\mathcal{D}} \mathrm{N}(0, \boldsymbol{W}(t)^T \mathrm{Var}(\boldsymbol{\theta}^*) \boldsymbol{W}(t)),$$

where $\mathrm{Var}(\boldsymbol{\theta}^*)$ is the covariance matrix of $\boldsymbol{\theta}^*$ obtained from the asymptotic variance matrix (5.21). Its estimate, $\hat{\mathrm{Var}}(\hat{\boldsymbol{\theta}})$, can be obtained from the estimate of (5.21). Hence the approximate 95% asymptotic pointwise confidence interval for $h_{\epsilon^*}(t)$, at time t, is

$$\hat{h}_{\epsilon^*}(t) \pm z_{0.025} \sqrt{\boldsymbol{W}(t)^T \hat{\mathrm{Var}}(\hat{\boldsymbol{\theta}}) \boldsymbol{W}(t)},$$

where $z_{0.025} = 1.96$. The approximate 95% asymptotic confidence intervals of β_j, $j = 1, \cdots, p$, are constructed in a similar way to that in Chapter 3.

Note that the second derivatives of $\ell_P(\boldsymbol{\beta}, \boldsymbol{\theta})$ (5.10) with respect to θ_j, $j = 1, \cdots, m$, is

$$\begin{aligned}
\frac{\partial^2 \ell_P(\boldsymbol{\beta}, \boldsymbol{\theta})}{\partial \theta_j^2} \;=\; & -\sum_{i \in \mathcal{L}} \left[\frac{\tau_{iL,j}}{1 - S(t_{iL})} \right] - \sum_{i \in \mathcal{L}} \left[\frac{\rho_{iL,j}}{1 - S(t_{iL})} \right]^2 - \sum_{i \in \mathcal{I}} \left[\frac{\tau_{iIR,j} - \tau_{iIL,j}}{S(t_{iIL}) - S(t_{iIR})} \right] \\
& - \sum_{i \in \mathcal{I}} \left[\frac{\rho_{iIR,j} - \rho_{iIL,j}}{S(t_{iIL}) - S(t_{iIR})} \right]^2 - \sum_{i \in \mathcal{O}} \left[\frac{\varphi_j(t^*_{iO})}{h_{\epsilon^*}(t^*_{iO})} \right]^2 - 2\tilde{\gamma} r_{jj},
\end{aligned}$$

where

$$\tau_{ic,j} = 2\pi\sigma_j^2 S(t_{ic})\Big[\Phi_j(t_{ic}^*) - \Phi_j(0)\Big]^2,$$

and, for $k \neq j$,

$$\begin{aligned}\frac{\partial^2 \ell_P(\boldsymbol{\beta},\boldsymbol{\theta})}{\partial\theta_k\partial\theta_j} &= -\sum_{i\in\mathcal{L}}\left[\frac{\rho_{iL,j}\rho_{iL,k}/S(t_{iL})}{1-S(t_{iL})}\right] - \sum_{i\in\mathcal{L}}\left[\frac{\rho_{iL,j}\rho_{iL,k}}{(1-S(t_{iL}))^2}\right] \\ &\quad -\sum_{i\in\mathcal{I}}\left[\frac{\rho_{iIR,j}\rho_{iIR,k}/S(t_{iIR}) - \rho_{iIL,j}\rho_{iIL,k}/S(t_{iIL})}{S(t_{iIL}) - S(t_{iIR})}\right] \\ &\quad -\sum_{i\in\mathcal{I}}\left[\frac{(\rho_{iIR,j}-\rho_{iIL,j})(\rho_{iIR,k}-\rho_{iIL,k})}{(S(t_{iIL}) - S(t_{iIR}))^2}\right] \\ &\quad -\sum_{i\in\mathcal{O}}\left[\frac{\varphi_j(t_{iO}^*)\varphi_k(t_{iO}^*)}{h_{\epsilon^*}^2(t_{iO}^*)}\right] - 2\tilde{\gamma}r_{jk}.\end{aligned}$$

The second derivative of $\ell_P(\boldsymbol{\beta},\boldsymbol{\theta})$ with respect to $\boldsymbol{\beta}$ and θ_j, $j = 1,\cdots,m$, are

$$\begin{aligned}\frac{\partial^2 \ell_P(\boldsymbol{\beta},\boldsymbol{\theta})}{\partial\boldsymbol{\beta}\partial\theta_j} &= \sum_{i\in\mathcal{L}}\left[\frac{\Omega_{iL,j}}{1-S(t_{iL})} + \frac{\rho_{iL,j}\bar{\eta}(t_{iL}^*)S(t_{iL})}{(1-S(t_{iL}))^2}\right]\boldsymbol{X}_{iL} + \sum_{i\in\mathcal{R}}\Big[\varphi_j(t_{iR}^*)t_{iR}^*\Big]\boldsymbol{X}_{iR} \\ &\quad +\sum_{i\in\mathcal{O}}\left[\frac{(t_{iO}^*-\mu_j)\varphi_j(t_{iO}^*)}{\sigma_j^2}\frac{t_{iO}^*}{h_{\epsilon^*}(t_{iO}^*)} - \varphi_j(t_{iO}^*)\frac{t_{iO}^* A(t_{iO}^*)}{h_{\epsilon^*}^2(t_{iO}^*)} + \varphi_j(t_{iO}^*)t_{iO}^*\right]\boldsymbol{X}_{iO} \\ &\quad +\sum_{i\in\mathcal{I}}\left[\frac{\Omega_{iIR,j}-\Omega_{iIL,j}}{S(t_{iIL})-S(t_{iIR})} - \frac{(\rho_{iIR,j}-\rho_{iIL,j})\xi_{iI}}{(S(t_{iIL})-S(t_{iIR}))^2}\right]\boldsymbol{X}_{iI}^T,\end{aligned}$$

where

$$\Omega_{ic,j} = -\varphi_j(t_{ic}^*)t_{ic}^* S(t_{ic}) + \rho_{ic,j}\bar{\eta}(t_{ic}^*).$$

5.5 Simulation studies

In this section, we conduct simulation studies to evaluate the performance of our proposed MPL method in fitting the AFT model with partly interval-censored failure time data. The main objectives are listed below:

1. Investigating effects of sample size and censoring proportion on the MPL estimates of $\boldsymbol{\beta}$ and $h_{\epsilon^*}(\cdot)$.

2. Comparing the asymptotic standard deviations with the Monte Carlo standard deviations of the MPL estimates, and

3. Comparing our method with the method by Komárek et al.(2005). The method by Komárek et al.(2005) can be found in CRAN.

All the results are obtained using R, and the relevant R codes are provided in Appendix C.

Objective 1 is to investigate sensitivities of our method to sample sizes and censoring proportions for the MPL estimates $\hat{\boldsymbol{\beta}}$ and $\hat{h}_{\epsilon^*}(\cdot)$. Results are presented in Tables 5.1-.5.3 and Figures 5.1-5.3. Objective 2 is to test whether the asymptotic standard deviations computed by the sandwich formula (5.21) are accurate for the MPL estimates, and this is achieved by comparing them with the Monte Carlo standard deviations. Results are reported in Tables 5.1-5.3 and Figures 5.1-5.3. For the third objective, we want to investigate how our method differs from the method by Komárek et al.(2005). The comparative results are reported in Table 5.4 and Figures 5.4-5.6.

We use samples with sizes of $n = 100$, 500, and 1000 corresponding to small, intermediate and large sample sizes respectively. For each sample size, we consider censoring proportions of $\pi_c = 20\%$, 50% and 80%. The data is generated as follows.

1. We use the same regression coefficient vector as that in Chapter 3, namely, $\boldsymbol{\beta} = [1, -0.3, 0.5]^T$. For each subject $i = 1, \cdots, n$, we generate covariate variables in the same ways as those in Chapter 3: the covariate X_{i1} is from a Bernoulli distribution with the parameter 0.5; X_{i2} is generated by $X_{i2} \sim \text{Unif}(0, 3)$; and X_{i3} generated by $X_{i3} \sim \text{Unif}(0, 5)$. The logarithm of failure time T_i, $i = 1, \cdots, n$ is then generated according to the model

$$\log T_i = 1 \cdot X_{i1} - 0.3 \cdot X_{i2} + 0.5 \cdot X_{i3} + \epsilon_i.$$

Therefore, the true failure time is obtained by

$$T_i = \exp\{1 \cdot X_{i1} - 0.3 \cdot X_{i2} + 0.5 \cdot X_{i3}\} \cdot \epsilon_i^*,$$

where $\epsilon_i^* = e^{\epsilon_i}$. We generate the error variable ϵ_i from an extreme value distribution with location parameter equal to 0 and scale parameter equal to $\frac{1}{3}$, which has the density function given by $f_{\epsilon_i}(t) = 3\exp\{3t - e^{3t}\}$. Therefore, the random variable ϵ_i^* follows the Weibull distribution with shape parameter equal to 3 and scale parameter equal to 1, i.e., $\epsilon_i^* \sim \text{WEB}(3, 1)$. The hazard function of ϵ_i^* is given by

$$h_{\epsilon_i^*}(t) = 3t^2.$$

2. Independently of T_i, we generate two monitoring times, C_{i1} and C_{i2}, for each subject i, which are generated in the same ways as those in Chapter 3, i.e., C_{i1} is from a standard uniform $\text{Unif}(0, 1)$ and C_{i2} is obtained by $C_{i2} = C_{i1} + \text{Unif}(0, 1)$. Then we generate the censoring times L_i and R_i following the same steps as those in Chapter 3.

For each combination of n and π_c, we generate 300 Monte Carlo samples. Based on the 300 Monte Carlo samples, we compute the average estimate (AEST), bias (BIAS), Monte Carlo standard deviation (MCSD), average asymptotic standard deviation (AASD) and mean squared error (MSE) for the MPL estimate $\hat{\boldsymbol{\beta}}$ by the formulas (3.41)-(3.45). For the MPL estimate $\hat{h}_{\epsilon^*}(t)$ at time t, we compute its AEST, MCSD, and AASD by the formula (3.46)-(3.48), and AISE by the formula (3.50).

Recall the Gaussian basis used to approximate the hazard function $h_{\epsilon^*}(\cdot)$ in (5.4), i.e., $\varphi(t; \mu_j, \sigma_j^2) = e^{-\frac{(t-\mu_j)^2}{2\sigma_j^2}}$, $j = 1, \cdots, m$. We select the knots μ_j's in the interval $\mathcal{I}(\boldsymbol{\beta}) = [t_{\min}(\boldsymbol{\beta}), t_{\max}(\boldsymbol{\beta})]$, where $t_{\min}(\boldsymbol{\beta}) = \min\{t_{ic}e^{-\boldsymbol{X}_{ic}^T\boldsymbol{\beta}} : i = 1, \cdots, n\}$, $t_{\max}(\boldsymbol{\beta}) = \max\{t_{ic}e^{-\boldsymbol{X}_{ic}^T\boldsymbol{\beta}} : i = 1, \cdots, n\}$ and $c = \{L, R, IL, IR, O\}$. Following the arguments by Komárek et al.(2005), we select the knots in such a way that the distance between two

consecutive knots is constant, i.e., $\delta = \mu_{j+1} - \mu_j$, and we choose σ_j by $\sigma_j = 2/3(\mu_{j+1} - \mu_j) = 2/3\delta$. Note that the knots depend on $\boldsymbol{\beta}$, and hence they change with respect to the estimated values of $\boldsymbol{\beta}$ during the iterative estimation algorithm. But in any case, there must be a knot before or at the point $t_{\min}(\boldsymbol{\beta})$ and after or at the point $t_{\max}(\boldsymbol{\beta})$. For number of knots m, we set $m = 8$ for $n = 100$, $m = 12$ for $n = 500$ and $m = 18$ for $n = 1000$. We choose the smoothing value according to the method described in Chapter 3.

Tables 5.1-5.3 provide the AEST, BIAS, MCSD, AASD and MSE values for the MPL estimates of $\boldsymbol{\beta}$ assuming different sample sizes and censoring proportions. We observe that the absolute value of BIAS, the MCSD, AASD and MSE all increase when censoring proportion increases, but decrease when sample size increases. Comparing MCSD with AASD in these three tables demonstrates that the sandwich formula (5.21) generally gives accurate variance approximation for the MPL estimates of $\boldsymbol{\beta}$, particularly with large sample size or small censoring proportion.

Figures 5.1-5.3 exhibit plots of the true hazard function $h_{\epsilon^*}(\cdot)$, its AESTs, the corresponding 95% Monte Carlo piecewise confidence intervals (PWCIs) and the corresponding average of 95% asymptotic PWCIs under different sample sizes and censoring proportions. We detect that: (i) the AEST of hazard becomes closer to the true hazard as sample size increases or censoring proportion decreases; (ii) both the 95% Monte Carlo PWCI and the average of 95% asymptotic PWCI become wider as censoring proportion increases but narrower with increasing sample size, and they all cover the true hazard; and (iii) the 95% Monte Carlo PWCI is close to the average of 95% asymptotic PWCI, and they become indistinguishable from each other when sample size increases or censoring proportion decreases. The closeness indicates that the sandwich formula (5.21) generally works well in approximating variance for the MPL estimate of $h_{\epsilon^*}(\cdot)$.

Tables 5.1-5.3 also report values of AISE for the MPL estimates of $h_{\epsilon^*}(\cdot)$ plotted in Figures 5.1-5.3 under different sample sizes and censoring proportions. We observe that the AISE exhibits increasing trend when the censoring proportion increases and decreasing trend when the sample size increases.

To compare our method with the method by Komárek et al.(2005), we use sample sizes $n = 100$, 500 and 1000, and censoring proportion $\pi_c = 0.5$. We generate $N = 300$ Monte Carlo samples. For both methods, we use the same number of knots. We select $m = 8$ for $n = 100$, $m = 12$ for $n = 500$ and $m = 18$ for $n = 1000$. The regression coefficients $\boldsymbol{\beta} = [\beta_1, \beta_2, \beta_3]^T$ and the covariates $\boldsymbol{X}_i = [X_{i1}, X_{i2}, X_{i3}]^T$ are the same as above. The two methods are evaluated based on the same data sets. Let $f_\epsilon(\cdot)$ be the density function of ϵ. Although Komárek et al.(2005) fit the AFT model involving estimation of the density of standardized ϵ, we can obtain the estimates of $f_\epsilon(\cdot)$ and $h_{\epsilon^*}(\cdot)$. For comparison, we also compute the estimate of $f_\epsilon(\cdot)$. We use the R package, smoothSurv, to obtain estimates of $\boldsymbol{\beta}$, $f_\epsilon(\cdot)$ and $h_{\epsilon^*}(\cdot)$ from the method by Komárek et al.(2005), where asymptotic standard deviation (astd) for the $\boldsymbol{\beta}$ estimate is computed based on the asymptotic theory for the penalized method and the corresponding astd formula is similar to our sandwich formula (5.21). Comparison between these two methods are accomplished by examining values of BIAS, MCSD, AASD and MSE for estimates of $\boldsymbol{\beta}$, and values of AISE for estimates of $f_\epsilon(\cdot)$ and $h_{\epsilon^*}(\cdot)$. Results are presented in Table 5.4 and Figures 5.4-5.6. For each sample size, we observe that our method gives higher absolute values of BIAS but lower values of MCSD for estimates of $\boldsymbol{\beta}$. Overall these two methods achieve quite similar MSE values. In each of the two methods, we compare values of MCSD with values of AASD for $\boldsymbol{\beta}$ estimates, and conclude that the astd formulas under these two methods generally approximate variance accurately. Figures 5.4-5.6 exhibit plots of the estimates for $f_\epsilon(\cdot)$ and $h_{\epsilon^*}(\cdot)$ with their true values and corresponding 95% Monte Carlo PWCIs

under different sample sizes. In our method, we also include the corresponding average of 95% asymptotic PWCIs for estimates of $h_{\epsilon^*}(\cdot)$. We observe that these two methods give good estimates both for $f_\epsilon(\cdot)$ and $h_{\epsilon^*}(\cdot)$, especially when the sample size is large, since the estimates are very close to their true values. In addition, all the 95% Monte Carlo PWCIs cover the true values, and they become narrower when the sample size increases. For sample sizes $n = 500$ and 1000, these two methods give similar values of AISE for the estimates of $f_\epsilon(\cdot)$ and $h_{\epsilon^*}(\cdot)$. However, for sample size $n = 100$, our method achieves lower values of AISE.

5.6 Real data analysis

In this section, we re-analyze the data from the AIDS study (Lindsey and Ryan, 1998) described in Chapter 3, using the Newton-MI estimation procedure. Let the T_i's and $\boldsymbol{X}_i$'s be defined as in Section 3.7 and assume that the distribution of T_i can be described by the AFT model (5.1). We set the number of knots $m = 8$ and smoothing value of 0.5. To test if the four covariates, stage of disease, dose of zidovudine, CD4: $100 - 399$ and CD4: ≥ 400, affect the time to development of resistance, based on the consistency and asymptotic normality properties of $\boldsymbol{\beta}$ (see Section 5.4), we perform a hypothesis test of the null hypothesis, H_0: $\beta_j = 0$ versus alternative hypothesis, H_a: $\beta_j \neq 0$, $j = 1, 2, 3$. Table 5.5 summarizes the MPL estimates $\hat{\beta}$ with the corresponding asymptotic standard deviations, p-values and 95% confidence intervals. We deduce that only the covariate of stage has significant effect on the time to development of resistance. Figure 5.7 shows plots for the MPL estimates of baseline hazard and baseline survival with their 95% asymptotic piecewise confidence intervals (PWCIs). We observe that the estimated baseline hazard remains stable over time and its 95% asymptotic PWCI becomes narrower until month 10, and then wider afterward.

5.7 Conclusions

This chapter has developed a MPL method to fit the AFT model with partly interval-censored failure time data, where the hazard function $h_{\epsilon^*}(\cdot)$ is modeled by a linear combination of Gaussian basis functions. We obtain the MPL estimates of $h_{\epsilon^*}(\cdot)$ and the regression coefficient vector $\boldsymbol{\beta}$ simultaneously by using a pioneer algorithm. The algorithm constitutes two alternative steps in each estimation iteration. The first step updates $\boldsymbol{\beta}$ by the modified Newton algorithm and the second step updates the Gaussian basis coefficients $\boldsymbol{\theta}$ by the MI algorithm. The resulting estimated hazard function is smooth and satisfies the non-negativity constraints. The sandwich formula (5.21) provides accurate estimates of the standard deviations for the MPL estimates.

Although we have developed the MPL method assuming independent censoring and time-independent covariates, it can be easily extended to fit the AFT model with dependent censoring or time-dependent covariates. For example, Lin and Ying (1995) have developed estimation methods for the AFT model with time-dependent covariates, but only considering right-censored data. In addition, our MPL approach can also be adapted to fit other models, such as an accelerated failure time partial linear model (AFT-PLM). The model incorporates a nonlinear structure of covariate into the AFT model, that is

$$\log T_i = \boldsymbol{X}_i^T \boldsymbol{\beta} + g(Z_i) + \epsilon_i,$$

where $\boldsymbol{\beta}$ is the p-dimensional vector of regression coefficients, $\boldsymbol{X}_i$ is the p-dimensional vector of covariates, Z_i is a 1-dimensional covariate, ϵ_i is an error variable with an unknown distribution, and $g(\cdot)$ is an unknown function. Zou et al. (2011) have proposed an estimation procedure for this model with right-censored data.

		n=100		
π_c		20%	50%	80%
m		8	8	8
$\beta_1 = 1$	AEST	0.9950	0.9757	0.9558
	BIAS	0.0050	0.0243	0.0442
	MCSD	0.0901	0.1018	0.1605
	AASD	0.0846	0.0978	0.1325
	MSE	(0.0081)	(0.0109)	(0.0277)
$\beta_2 = -0.3$	AEST	-0.3238	-0.3234	-0.3479
	BIAS	0.0238	0.0234	0.0479
	MCSD	0.0440	0.0616	0.0866
	AASD	0.0374	0.0536	0.0698
	MSE	(0.0025)	(0.0043)	(0.0097)
$\beta_3 = 0.5$	AEST	0.4865	0.4867	0.4706
	BIAS	0.0135	0.0133	0.0294
	MCSD	0.0356	0.0359	0.0555
	AASD	0.0275	0.0274	0.0404
	MSE	(0.0014)	(0.0015)	(0.0039)
$h_{\epsilon^*}(t)$	AISE	0.1627	0.1995	0.3284

Table 5.1: AEST, BIAS, MCSD, AASD and MSE for the estimates $\hat{\boldsymbol{\beta}}$, and average integrated squared error (AISE) for the hazard estimates $\hat{h}_{\epsilon^*}(t)$ with number of knots $m = 8$, sample size $n = 100$ and censoring proportions $\pi_c = 20\%$, 50% and 80% respectively.

		n=500		
π_c		20%	50%	80%
m		12	12	12
$\beta_1 = 1$	AEST	0.9963	0.9919	0.9926
	BIAS	0.0037	0.0081	0.0074
	MCSD	0.0338	0.0386	0.0556
	AASD	0.0318	0.0407	0.0582
	MSE	(0.0012)	(0.0016)	(0.0031)
$\beta_2 = -0.3$	AEST	-0.3025	-0.3050	-0.3160
	BIAS	0.0025	0.0050	0.0160
	MCSD	0.0178	0.0243	0.0317
	AASD	0.0172	0.0220	0.0300
	MSE	(0.0003)	(0.0006)	(0.0013)
$\beta_3 = 0.5$	AEST	0.4988	0.4938	0.4937
	BIAS	0.0012	0.0062	0.0063
	MCSD	0.0109	0.0139	0.0210
	AASD	0.0104	0.0132	0.0199
	MSE	(0.0001)	(0.0002)	(0.0005)
$h_{\epsilon^*}(t)$	AISE	0.0486	0.0648	0.0851

Table 5.2: AEST, BIAS, MCSD, AASD and MSE for the estimates $\hat{\boldsymbol{\beta}}$, and average integrated squared error (AISE) for the hazard estimates $\hat{h}_{\epsilon^*}(t)$ with number of knots $m = 12$, sample size $n = 500$ and censoring proportions $\pi_c = 20\%$, 50% and 80% respectively.

		n=1000		
π_c		20%	50%	80%
m		18	18	18
$\beta_1 = 1$	AEST	0.9960	1.0044	0.9928
	BIAS	0.0040	-0.0044	0.0072
	MCSD	0.0266	0.0279	0.0457
	AASD	0.0230	0.0274	0.0394
	MSE	(0.0007)	(0.0008)	(0.0021)
$\beta_2 = -0.3$	AEST	-0.3006	-0.3047	-0.3072
	BIAS	0.0006	0.0047	0.0072
	MCSD	0.0127	0.0160	0.0251
	AASD	0.0124	0.0150	0.0200
	MSE	(0.0002)	(0.0003)	(0.0007)
$\beta_3 = 0.5$	AEST	0.4990	0.4971	0.4983
	BIAS	0.0010	0.0029	0.0017
	MCSD	0.0097	0.0098	0.0148
	AASD	0.0090	0.0090	0.0119
	MSE	(0.0001)	(0.0001)	(0.0002)
$h_{\epsilon^*}(t)$	AISE	0.0202	0.0340	0.0715

Table 5.3: AEST, BIAS, MCSD, AASD and MSE for the estimates $\hat{\boldsymbol{\beta}}$, and average integrated squared error (AISE) for the hazard estimates $\hat{h}_{\epsilon^*}(t)$ with number of knots $m = 18$, sample size $n = 1000$ and censoring proportions $\pi_c = 20\%$, 50% and 80% respectively.

		n=100		n=500		n=1000	
		MPL_1	MPL_2	MPL_1	MPL_2	MPL_1	MPL_2
$\beta_1 = 1$	AEST	0.9677	0.9955	0.9947	0.9983	0.9977	0.9994
	BIAS	0.0323	0.0045	0.0053	0.0017	0.0023	0.0006
	MCSD	0.0908	0.1173	0.0502	0.0494	0.0311	0.0329
	AASD	0.0839	0.1026	0.0404	0.0401	0.0290	0.0292
	MSE	(0.0093)	(0.0138)	(0.0025)	(0.0024)	(0.0010)	(0.0011)
$\beta_2 = -0.3$	AEST	-0.3357	-0.2986	-0.3063	-0.3001	-0.3021	-0.3000
	BIAS	0.0357	-0.0014	0.0063	0.0001	0.0021	0.0000
	MCSD	0.0557	0.0704	0.0257	0.0245	0.0148	0.0148
	AASD	0.4991	0.0601	0.0234	0.0231	0.0161	0.0162
	MSE	(0.0044)	(0.0050)	(0.0007)	(0.0006)	(0.0002)	(0.0002)
$\beta_3 = 0.5$	AEST	0.4812	0.4999	0.4966	0.4996	0.4985	0.4994
	BIAS	-0.0188	0.0001	0.0034	0.0004	0.0015	0.0006
	MCSD	0.0304	0.0381	0.0127	0.0122	0.0102	0.0106
	AASD	0.0289	0.0254	0.0136	0.0137	0.0097	0.0096
	MSE	(0.0013)	(0.0015)	(0.0002)	(0.0001)	(0.0001)	(0.0001)
$h_{\epsilon^*}(t)$	AISE	0.2065	0.8785	0.0722	0.0852	0.0303	0.0350
$f_{\epsilon}(t)$	AISE	0.0570	0.0828	0.0105	0.0096	0.0051	0.0045

Table 5.4: Comparisons of estimates between our MPL method, denoted by MPL_1, and the MPL method by Komárek et al.(2005), denoted by MPL_2, using simulated samples with sample sizes of $n = 100$, 500 and 1000, and censoring proportion of $\pi_c = 0.5$.

research/R and matlab/R for AFT model Phd/100all.eps

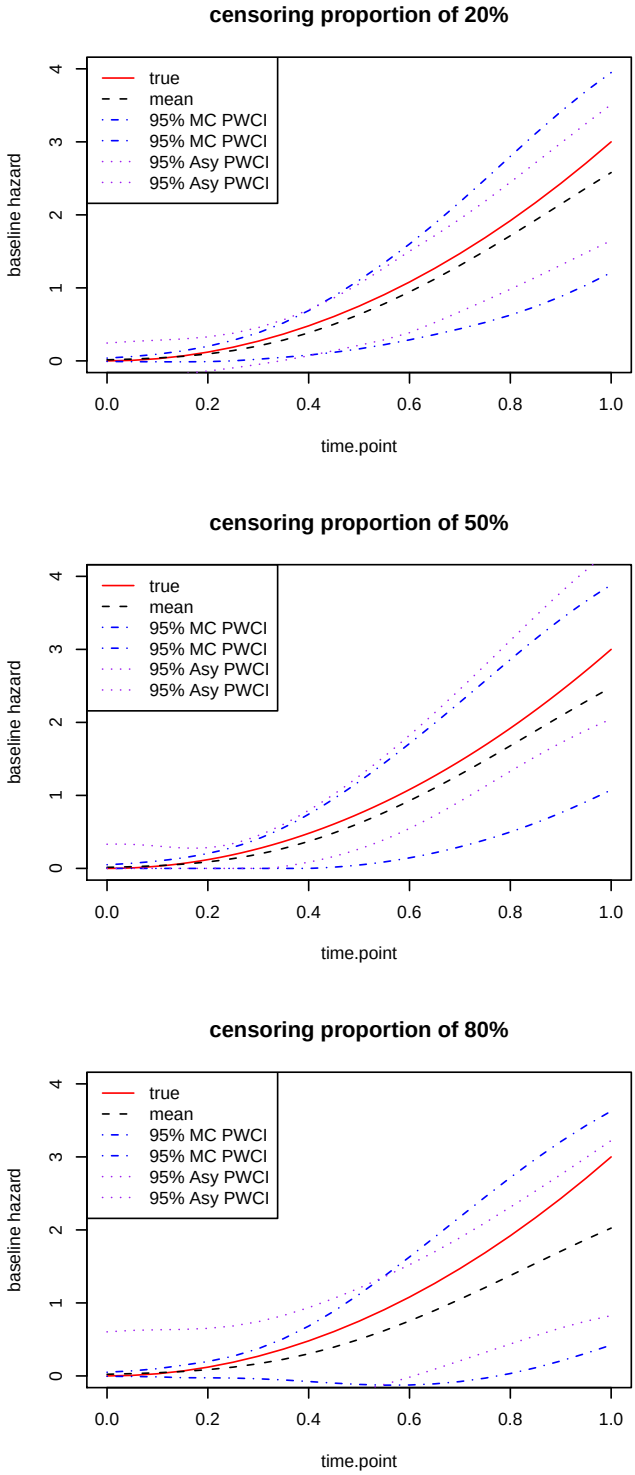

Figure 5.1: Plots of the true hazard $h_{\epsilon^*}(t)$ (solid), the average MPL estimates of $h_{\epsilon^*}(t)$ (dash), the 95% Monte Carlo piecewise confidence interval (PWCI) (dot-dash), and the average 95% asymptotic PWCI (dots), assuming sample size $n = 100$, number of knots $m = 8$ and censoring proportions of 20%, 50% and 80% respectively.

research/R and matlab/R for AFT model Phd/500all.eps

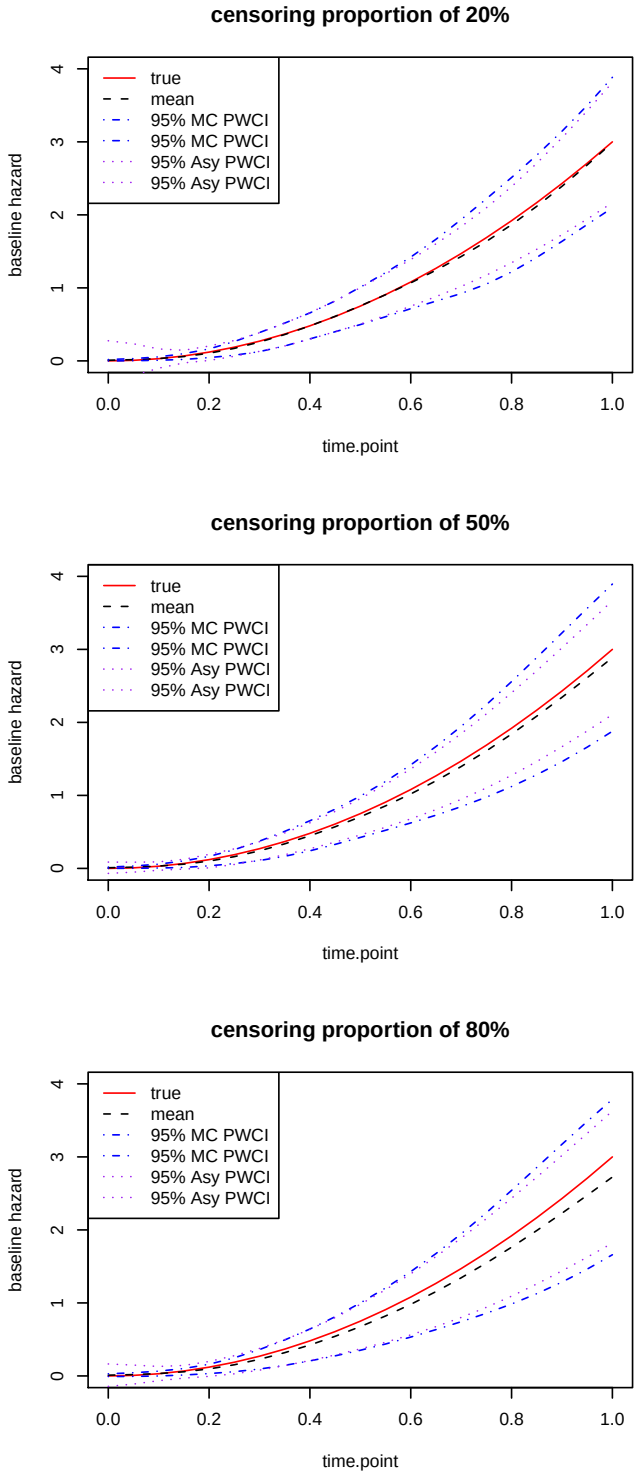

Figure 5.2: Plots of the true hazard $h_{\epsilon^*}(t)$ (solid), the average MPL estimates of $h_{\epsilon^*}(t)$ (dash), the 95% Monte Carlo piecewise confidence interval (PWCI) (dot-dash), and the average 95% asymptotic PWCI (dots), assuming sample size $n = 500$, number of knots $m = 12$ and censoring proportions of 20%, 50% and 80% respectively.

research/R and matlab/R for AFT model Phd/1000all.eps

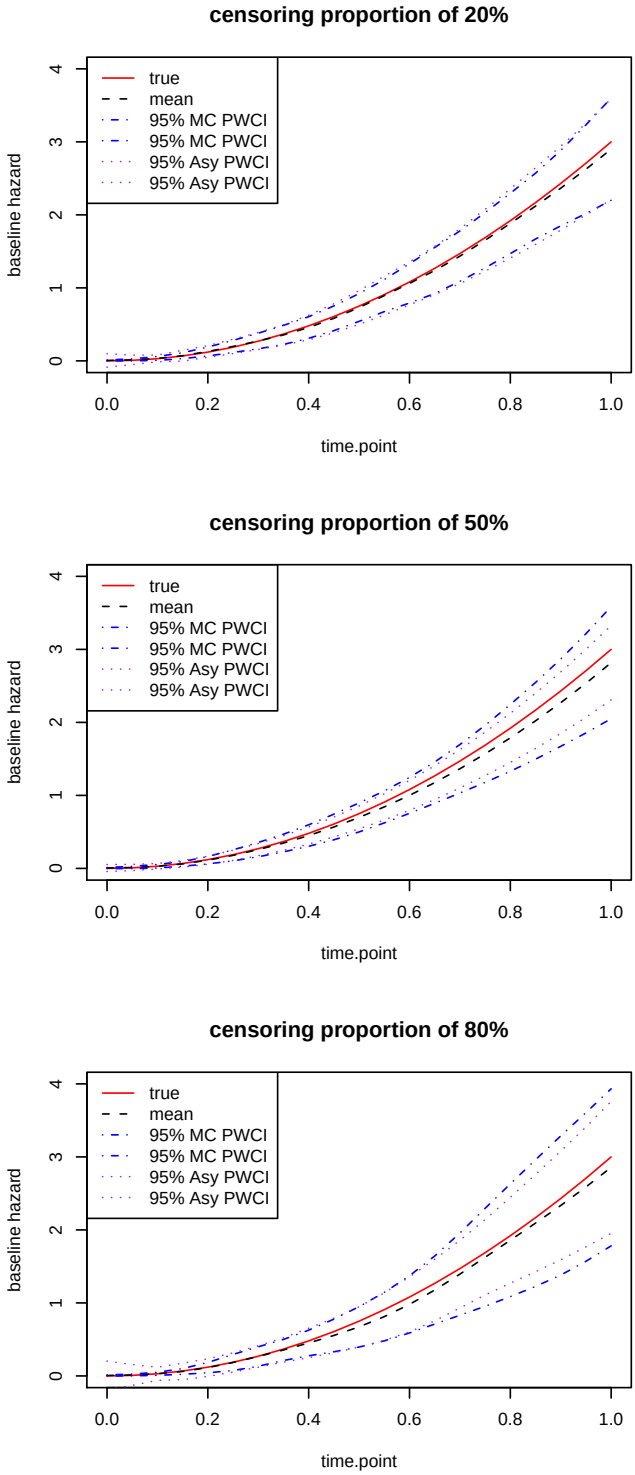

Figure 5.3: Plots of the true hazard $h_{\epsilon^*}(t)$ (solid), the average MPL estimates of $h_{\epsilon^*}(t)$ (dash), the 95% Monte Carlo piecewise confidence interval (PWCI) (dot-dash), and the average 95% asymptotic PWCI (dots), assuming sample size $n = 1000$, number of knots $m = 18$ and censoring proportions of 20%, 50% and 80% respectively.

research/R and matlab/R for AFT model Phd/100mc1.eps

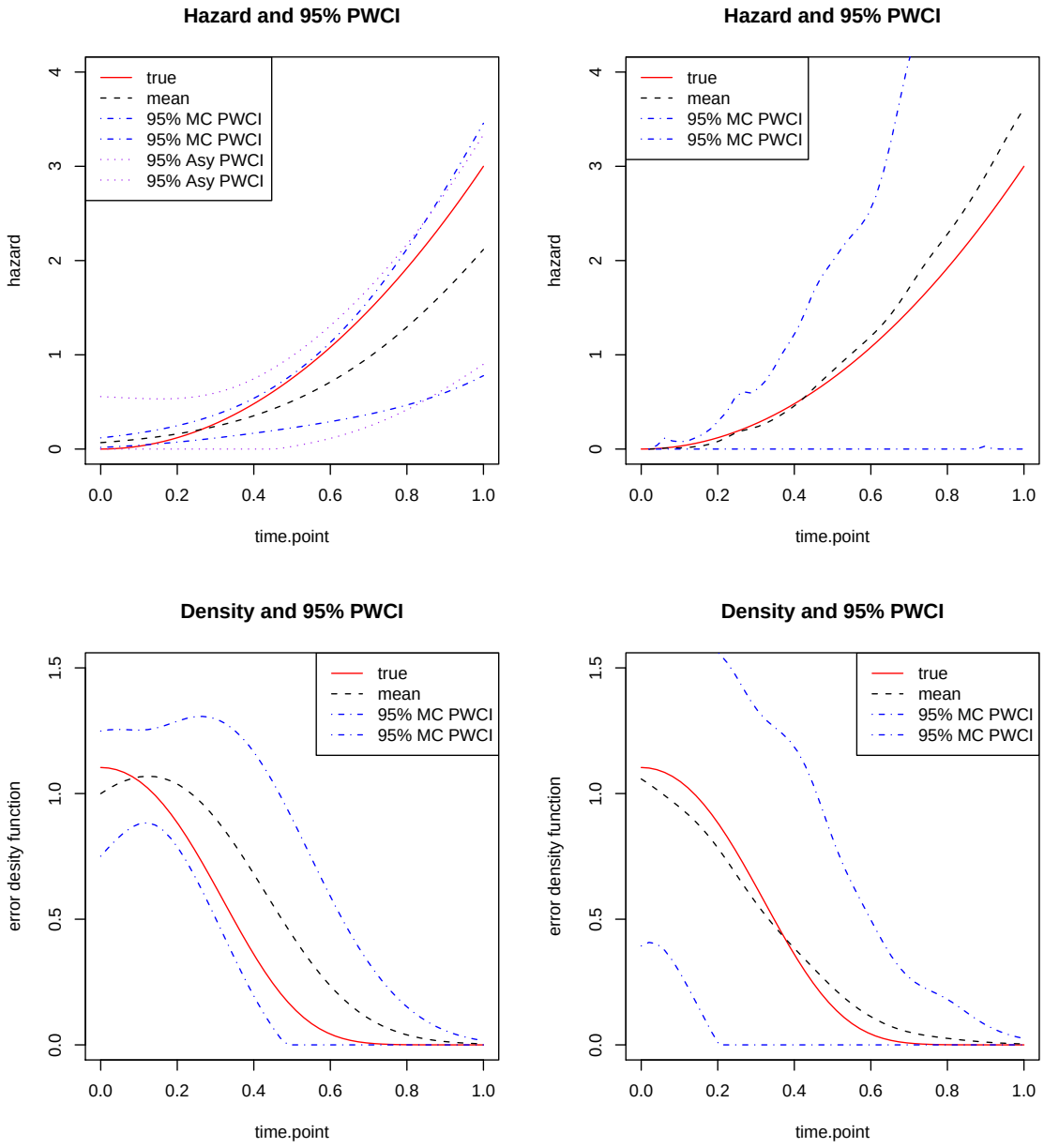

Figure 5.4: Plots of the estimates for the two methods. Left side: our MPL method for the hazard estimate of ϵ^* and the density estimate of ϵ. Right side: the MPL method by Komárek et al.(2005) for the hazard estimate of ϵ^* and the density estimate of ϵ. Sample size is $n = 100$ and censoring proportion is $\pi_c = 50\%$.

research/R and matlab/R for AFT model Phd/500mc1.eps

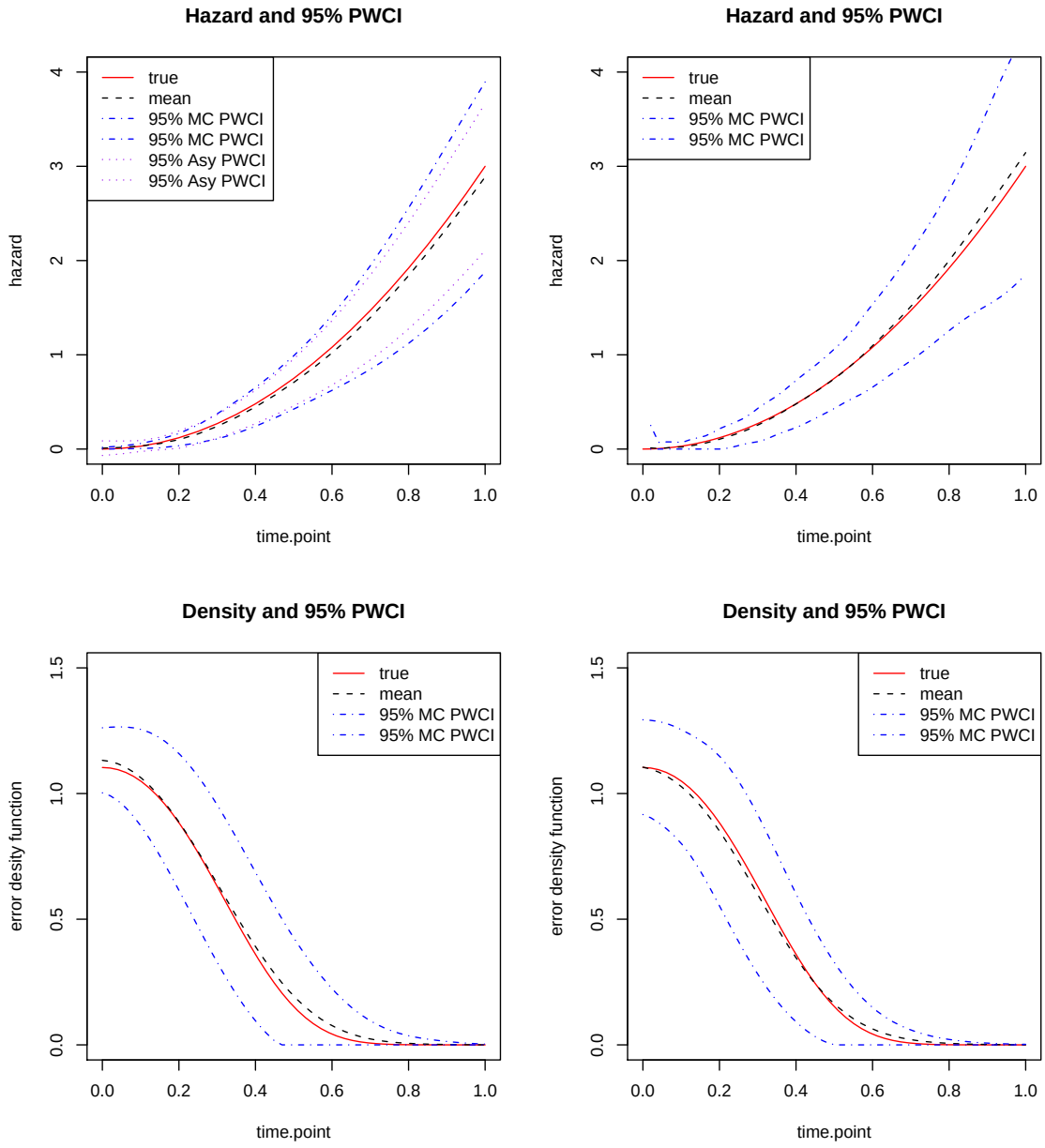

Figure 5.5: Plots of the estimates for the two methods. Left side: our MPL method for the hazard estimate of ϵ^* and the density estimate of ϵ. Right side: the MPL method by Komárek et al.(2005) for the hazard estimate of ϵ^* and the density estimate of ϵ. Sample size is $n = 500$ and censoring proportion is $\pi_c = 50\%$.

research/R and matlab/R for AFT model Phd/1000mc1.eps

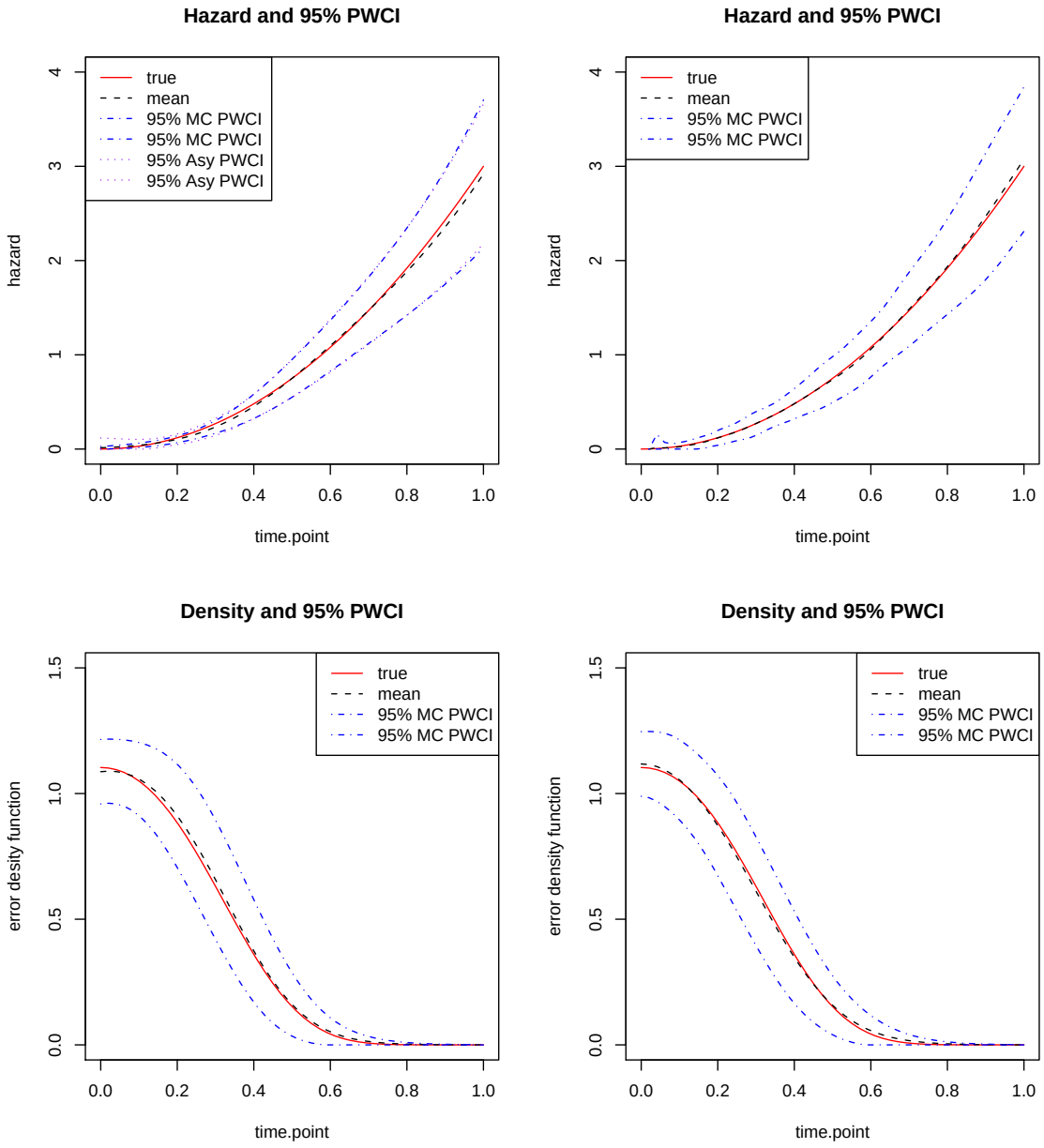

Figure 5.6: Plots of the estimates for the two methods. Left side: our MPL method for the hazard estimate of ϵ^* and the density estimate of ϵ. Right side: the MPL method by Komárek et al.(2005) for the hazard estimate of ϵ^* and the density estimate of ϵ. Sample size is $n = 1000$ and censoring proportion is $\pi_c = 50\%$.

research/R and matlab/R for AFT model Phd/realdata.eps

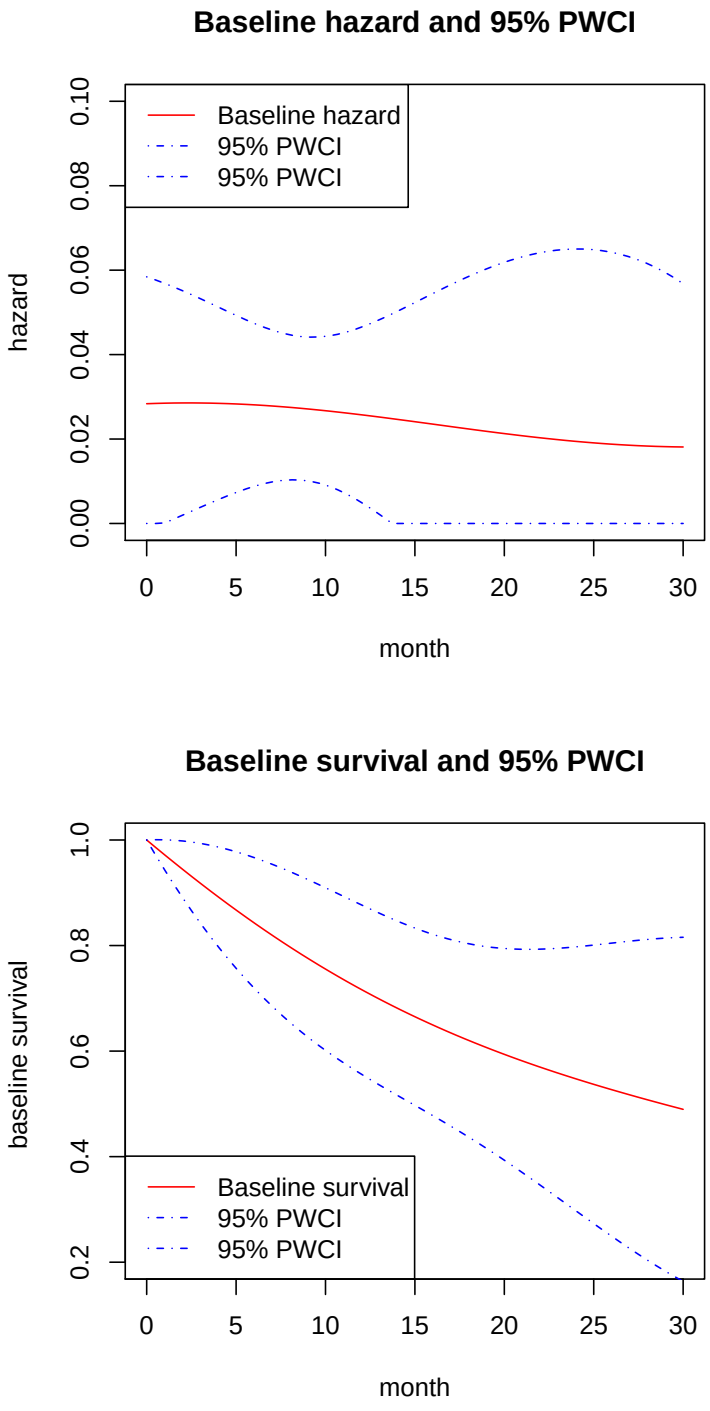

Figure 5.7: Plots of the estimates for the AIDS study. Top: the baseline hazard and its 95% piecewise confidence interval (PWCI). Bottom: the baseline survival function with its 95% PWCI.

Effects	$\hat{\boldsymbol{\beta}}$	astd	p-value	95%C.I
stage	-1.2068	0.4833	0.0125	(-2.1541, -0.2595)
dose	-0.8508	0.4833	0.0783	(-1.7981, 0.0965)
CD4: 100-399	0.6826	0.6156	0.2675	(-0.5240, 1.8892)
CD4: ≥ 400	0.7156	0.4747	0.1317	(-0.2148,1.6460)

Table 5.5: Regression coefficient estimates given by the MPL method with asymptotic standard deviations (astd), p-values and 95% confidence intervals.

Chapter 6

Conclusions and future work

6.1 Conclusions

This dissertation has developed novel estimation methods for the PH, AH and AFT models with partly interval-censored data, which contains exactly observed, left-censored, finite interval-censored and right-censored data. These methods attempt to maximize the penalized log-likelihood functions. We assumed that the data were from independent subjects. For each subject, the covariate vector was assumed to be time-independent, and censoring time was independent of failure time.

We fitted the PH model by estimating the regression coefficients and baseline hazard. We first assumed the baseline hazard to be piecewise constant, and then expressed it in terms of piecewise constant functions. We obtained estimates of the regression coefficients and baseline hazard simultaneously by maximizing the penalized log-likelihood function using the Newton-MI algorithm, which combines the Newton algorithm and the MI algorithm (Ma, 2006). In the estimation procedure, the regression coefficients are updated by the Newton algorithm, while the baseline hazard is updated by the MI algorithm. The MI algorithm not only guarantees the non-negative constraint for the baseline hazard,

but also avoids requiring the second derivatives of the penalized log-likelihood with respect to the baseline hazard. The convergence of this algorithm have been proved by Ma et al. (2014). We produced the smoothed estimate of the baseline hazard through the penalty function which represents the square of the second order differences for the basis coefficients of the baseline hazard. Our MPL method differs from those of Joly et al. (1998) and Cai and Betensky (2003) in the way the non-negative constraint on the baseline hazard is handled. They imposed the constraint in some indirect ways which may cause estimation or convergence issues. On the other hand, we imposed the constraints directly by constraining non-negativity on the basis coefficients of the baseline hazard. In addition, the algorithms they proposed also differ from the Newton-MI algorithm.

For the AH model, we estimated the regression coefficients and the baseline hazard by maximizing the penalized log-likelihood function, where the penalty function was included for smoothness of the baseline hazard estimate. In the estimation procedure, we not only imposed a non-negative constraint on the baseline hazard, but also on the hazard. The constraint on these two quantities are imposed simultaneously and directly by the primal-dual interior point algorithm. This algorithm has also been used by Ghosh (2001) in studying current status data, but only for the maximum likelihood estimation which may not yield a smoothed baseline hazard estimate.

Under the AFT model, we simultaneously computed the estimates for the regression coefficients and the hazard of the exponential function of the error variable by maximizing the penalized log-likelihood function, where we used the roughness penalty function for smoothness of the hazard estimate. The roughness penalty function relates the smoothness of the hazard estimate to its second derivative. We approximated the hazard using a linear combination of Gaussian basis functions, and constrained the hazard to be non-negative directly by imposing the non-negativity on the basis coefficients. The Newton-

MI algorithm was applied for the estimations. This can be compared with Komárek et al.(2005), who proposed a penalized likelihood method that produced a smoothed estimate for the density function of the standardized error variable, but there are three constraints imposed in the estimation procedure which are imposed indirectly. Moreover, the Newton-MI algorithm we used is not like that used by Komárek et al.(2005).

We fit the PH and AFT models using the Newton-MI algorithm. While, for the AH model, we compute estimators using the primal-dual interior point algorithm. The reason is that, under the AH model, it is easier to prove convexity of the penalized log-likelihood function in $\boldsymbol{\eta}$, where $\boldsymbol{\eta} = [\boldsymbol{\beta}^T, \boldsymbol{\theta}^T]^T$, and the estimator of $\boldsymbol{\eta}$ can be obtained just by solving the KKT equations for $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ simultaneously during the estimation procedure of the primal-dual interior point algorithm. Under the PH and AFT models, it is harder to prove concavity of the penalized log-likelihood function in $\boldsymbol{\eta}$. Using the Newton-MI algorithm, we can solve the KKT equations for $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$. Under the AFT model, we model the baseline hazard function by a linear combination of Gaussian basis functions instead of piecewise constant functions. This is because the penalized log-likelihood function constructed based on the assumption of piecewise constant on the baseline hazard is not differential with respect to $\boldsymbol{\beta}$ with fixed $\boldsymbol{\theta}$.

We investigated our proposed MPL method under the three models by conducting simulation studies. For all three models, the results of the simulation studies showed that the bias and standard deviation for the MPL estimate of regression coefficients increased with the censoring proportion but decreased with the sample size. The MPL estimate for the hazard also exhibited the same trend. The sandwich formula derived using asymptotic theory provided a good estimate for the standard deviation, especially when the sample size was large or the censoring proportion was small. Comparisons between our method and other methods were also made in the simulation studies. Under the PH model, we

compared our method with the method by Pan (1999), and the results showed that our method gave lower mean squared error (MSE) for the estimate of the regression coefficients and lower average integrated squared error (AISE) for the baseline hazard estimate. Under the AH model, we compared our method with that produced by Lin and Ying (1994). The results demonstrated that although the estimates of the regression coefficients were similar, our MPL method gave better estimates for the baseline hazard. For the AFT model, we compared with the method by Komárek et al.(2005). Estimates under the two methods were similar. For a real data analysis, we observed that the smoothed baseline hazard estimate, produced by our MPL method, gave clear patterns of hazard over time.

6.2 Future work

We can extend the proposed MPL methods of this dissertation further based on the following future research directions.

In developing the MPL methods, we assume the covariate variables are time independent and the censoring time is independent of the failure time. However, we can extend our methods to estimate the PH, AH and AFT models with partly interval-censored data under time-dependent covariates and dependent censoring. Dependent censoring can be modeled using either a copula function or a frailty model, where a copula can be represented as a function of the marginal distributions of the failure and censoring times, which is the joint distribution function, and a frailty is the common random variable shared between the failure and censoring times, and the joint distribution function of the failure and censoring times is the integral of the conditional joint distribution function given the frailty value.

In addition, our MPL methods can also be applied in fitting more general models. For example, we can fit an additive-multiplicative hazard model, which combines the

PH and AH models together, with partly interval-censored data. Lin and Ying (1995) considered some estimation procedures for this model, but only with right-censored data. Alternatively, we can use our MPL methods to study an accelerated failure time partial linear model (AFT-PLM), which incorporates a nonlinear structure of covariates into the AFT model.

Bibliography

Anderson, J. and Senthilselvan, A. (1980), 'Smooth estimates for the hazard function' *Journal of the Royal Statistical Society. Series B (Methodological)* **42**(3), 322-327.

Bertsekas, D. (1982), 'Projected Newton methods for optimization problems with simple constraints', *SIAM Journal on Control and Optimization* **20**(2), 221-246.

Betensky, R. A., Lindsey, J. C., Ryan, L.M. and Wand, M. (1999), 'Local EM estimation of the hazard function for interval-censored data', *Biometrics* **55**(1), 238-245.

Betensky, R. A., Lindsey, J.C., Ryan, L.M. and Wand, M. (2002), 'A local likelihood proportional hazards model for interval-censored data', *Statistics in Medicine* **21**(2), 263-275.

Betensky, R. A., Rabinowitz, D. and Tsiatis, A. A. (2001), 'Computationally simple accelerated failure time regression for interval-censored data', *Biometrika* **88**(3), 703-711.

Breslow, N.E. (1972), 'Contribution to the discussion of the paper by Dr Cox', *Journal of the Royal Statistical Society, Series B* **34**(2), 216-217.

Buckley, J. and James, I. (1979), 'Linear regression with censored data', *Biometrika* **66**(3), 429-436.

Cai, T. and Betensky, R. A. (2003), 'Hazard regression for interval-censored data with penalized spline', *Biometrics* **59**(3), 570-579.

Cox, D. R. (1975), 'Partial likelihood', *Biometrika* **62**(2), 269-276.

Currie, I.D. (1996), 'A note on Buckley-James estimators for censored data', *Biometrika* **83**(4), 912-915.

Farrington, C. (1996), 'Interval-censored survival data: a generalized linear modeling approach', *Statistics in Medicine* **15**(3), 283-292.

Finkelstein, D. M. (1986), 'A proportional hazards model for interval-censored failure time data', *Biometrics* **42**(4), 845-854.

Gentleman, R. and Geyer, C. J. (1994), 'Maximum likelihood for interval-censored data: Consistency and computation', *Biometrika* **81**(3), 618-623.

Ghosh, D. (2001), 'Efficiency considerations in the additive hazards model with current status data', *Statistica Neerlandica* **55**(3), 367-376.

Goetghebeur, E. and Ryan, L. (2000), 'Semi-parametric regression analysis of interval-censored data', *Biometrics* **56**(4), 1139-1144.

Goggins, W. B., Finkelstein, D. M., Schoenfeld, D. A. and Zaslavsky, A. M. (1998), 'A Markov Chain Monte Carlo EM algorithm for analyzing interval-censored data under the Cox proportional hazards model', *Biometrics* **54**(4), 1498-1507.

Gray, R. J. (1990), 'Some diagnostic methods for Cox regression models through hazard smoothing', *Biometrics* **46**(1), 93-102.

Gray, R. J. (2000), 'Estimation of regression parameters and the hazard function in transformed linear survival models', *Biometrics* **56**(2), 571-576.

Groeneboom, P. and Wellner, J.A. (1992), *Information bounds and non-parametric maximum likelihood estimation*, Vol. 19, Springer.

Hastings, W. K. (1970), 'Monte Carlo sampling methods using Markov Chains and their applications', *Biometrika* **57**(1), 97-109.

Jin, Z., Lin, D., Wei, L. and Ying, Z. (2003), 'Rank-based inference for the accelerated failure time model', *Biometrika* **90** (2), 341-353.

Jin, Z., Lin, D. and Ying, Z. (2006), 'On least-squares regression with censored data', *Biometrika* **93**(1), 147-161.

Joly, P., Commenges, D. and Letenneur, L. (1998), 'A penalized likelihood approach for arbitrarily censored and truncated data: application to age-specific incidence of dementia', *Biometrics* **54**(1), 185-194.

Kalbfleisch, J. D. and Prentice, R. L. (2002), *The statistical analysis of failure time data*, John Wiley & Sons.

Keiding, N. (1991), 'Age-specific incidence and prevalence: a statistical perspective', *Journal of the Royal Statistical Society. Series A (Statistics in Society)* **154**(3), 371-412.

Komárek, A., Lesaffre, E. and Hilton, J. F. (2005), 'Accelerated Failure Time Model for Arbitrarily Censored Data With Smoothed Error Distribution', *Journal of Computational and graphical Statistics* **14**(3), 726-745.

Kooperberg, C. and Clarkson, D. B. (1997), 'Hazard regression with interval-censored data', *Biometrics* **53**(4), 1485-1494.

Kooperberg, C. and Stone, C. J. (1992), 'Logspline density estimation for censored data', *Journal of Computational and Graphical Statistics* **1**(4), 301-328.

Law, C.G. and Brookmeyer, R. (1992), 'Effects of mid-point imputation on the analysis of doubly censored data', *Statistics in Medicine* **11**(12), 1569-1578.

Lawless, J. F. (2011), *Statistical models and methods for lifetime data*, Vol. 362, John Wiley & Sons.

Li, L. and Pu, Z. (2003), 'Rank estimation of log-linear regression with interval-censored data', *Lifetime Data Analysis* **9**(1), 57-70.

Li, L., Watkins, T. and Yu, Q. (1997), 'An EM algorithm for smoothing the self-consistent estimator fo survival functions with interval-censored data', *Scandinavian Journal of Statistics* **24**(4), 531-542.

Lin, D., Oakes, D. and Ying, Z. (1998), 'Additive hazards regression with current status data', *Biometrika* **85**(2), 289-298.

Lin, D and Ying, Z. (1994), 'Semiparametric analysis of the additive risk model', *Biometrika* **81**(1), 61-71.

Lin, D and Ying, Z. (1995), 'Semiparametric inference for the accelerated life model with time-dependent covariates', *Journal of statistical planning and inference* **44**(1), 47-63.

Lindsey, J. C. and Ryan, L. M. (1998), 'Methods for interval-censored data', *Statistics in Medicine* **17**(2), 219-238.

Luenberger, D. G. (2003) *Linear and nonlinear programming*, Springer.

Ma, J. (2006), 'Multiplicative algorithms for maximum penalized likelihood inversion with non-negative constraints and generalized error distributions', *Communications in Statistics Theory and Methods* **35**(5), 831-848.

Ma, J. (2010), 'Positively constrained multiplicative iterative algorithm for maximum penalized likelihood tomographic reconstruction', *Nuclear Science, IEEE Transactions* **57**(1), 181-192.

Ma, J., Heritier, S. and Lô, S.N. (2014), 'On the maximum penalized likelihood approach for proportional hazard models with right censored survival data', *Computational Statistics and Data Analysis* **74**(0), 142-156.

Miller, R. G. (1976), 'Least squares regression with censored data', *Biometrika* **63**(3), 449-464.

O'Sullivan, F. (1988), 'Fast computation of fully automated log-density and log-hazard estimators', *SIAM Journal on scientific and statistical computing* **9**(2), 363-379.

Pan, W. (1999), 'Extending the iterative convex minorant algorithm to the Cox model for interval-censored data', *Journal of Computational and Graphical Statistics* **8**(1), 109-120.

Pan, W. (2000), 'A multiple imputation approach to Cox regression with interval-censored data', *Biometrics* **56**(1), 199-203.

Park, Y. and Wei, L. (2003), 'Estimating subject-specific survival functions under the accelerated failure time model', *Biometrika* **90**(3), 717-723.

Peto, R. (1973), 'Experimental survival curves for interval-censored data', *Applied Statistics* **22**(1), 86-91.

Prentice, R. L. (1978), 'Linear rank tests with right censored data', *Biometrika* **65**(1), 167-179.

Rabinowitz, D., Tsiatis, A. and Aragon, J. (1995), 'Regression with interval-censored data', *Biometrika* **82**(3), 501-513.

Ralph, D. and Wright, S. J. (1997), 'Superlinear convergence of an interior-point method for monotone variational inequalities', *Complementarity and Variational Problems: State of the Art*, 345-385.

Ritov, Y. (1990), 'Estimation in a linear regression model with censored data', *The Annals of Statistics* **18**(1), 303-328.

Robertson, T., Wright, F. T. and Dykstra, R. L. (1988), *Order Restricted Statistical Inference*, John Wiley & Sons.

Rockafellar, R. T. (1997), *Convex analysis*, Vol. 28, Princeton university press.

Ruppert, D., Reish, R. L., Dersio, R. B. and Carroll, R. J. (1984), 'Optimization using stochastic approximation and Monte Carlo simulation (with application to harvesting of Atlantic Menhaden)', *Biometrics* **40**(2), 535-545.

Satten, G. A. (1996), 'Rank-based inference in the proportional hazards model for interval censored data', *Biometrika* **83**(2), 355-370.

Satten, G.A., Datta, S. and Williamson, J.M. (1998), 'Inference based on imputed failure times for the proportional hazards model with interval-censored data', *Journal of the*

American Statistical Association **93**(441), 318-327.

Silverman, B.W. (1984), 'A fast and efficient cross-validation method for smoothing parameter choice in spline regression', *Journal of the American Statistical Association* **79**(387), 584-589.

Sun, J. (2006), *The statistical analysis of interval-censored failure time data*, Springer.

Sun, W. and Yuan, Y. (2006), *Optimization theory and methods: nonlinear programming*, Springer.

Tanner, M.A. and Wong, W.H. (1987), 'An application of imputation to an estimation problem in grouped lifetime analysis', *Technometrics* **29**(1), 23-32.

Tsiatis, A.A. (1990), 'Estimating regression parameters using linear rank tests for censored data', *The Annals of Statistics* **18**(1), 354-372.

Turnbull, B.W. (1976), 'The empirical distribution function with arbitrarily grouped, censored and truncated data', *Journal of the Royal Statistical Society. Series B (Methodological)* **38**(3), 290-295.

Wang, L., Sun, J. and Tong, X. (2010), 'Regression analysis of case II interval-censored failure time data with the additive hazards model', *Statistica Sinica* **20**(4), 1709-1723.

Wei, G.C. and Tanner, M. A. (1991), 'Applications of multiple imputation to the analysis of censored regression data', *Biometrics* **47**(4), 1297-1309.

Wellner, J. A. and Zhan, Y. (1997), 'A hybrid algorithm for computation of the nonparametric maximum likelihood estimator from censored data', *Journal of the American Statistical Association* **92**(439), 945-959.

Wright, S. J. (1997), *Primal-dual interior-point methods*, Vol. 54, Siam.

Yu, Q., Li, L. and Wong, G. Y. (2000), 'On consistency of the self-consistent estimator of survival functions with interval-censored data', *Scandinavian Journal of Statistics* **27**(1), 35-44.

Yu, Y. and Ruppert, D. (2002), 'Penalized spline estimation for partially linear single-index models', *Journal of the American Statistical Association* **97**(460), 1042-1054.

Zeng, D., Cai, J. and Shen, Y. (2006), 'Semiparametric additive risks model for interval-censored data', *Statistica Sinica* **16**(1), 287.

Zeng, D. and Lin, D. (2007), 'Efficient estimation for the accelerated failure time model', *Journal of the American Statistical Association* **102**(480), 1387-1396.

Appendix A R codes for PH model

Note that the following codes are the selected R codes for this dissertation. The electronic version of the R codes will be provided under the request of the examiners.

A.1 Data generation

```
# n: sample size p: censoring proportion categorial variable
Ca = matrix(rep(0, n))
ur = runif(n)
Ca[ur <= 0.5] = 1
Ca[ur > 0.5] = 0
# Covariate matrix
X = cbind(Ca, runif(n, min = 0, max = 3), runif(n, min = 0, max = 5))
# beta values
beta = matrix(c(1, -0.3, 0.5))
# Generate data
SimData <- function(beta, X, p, n) {
    u = runif(n)
    t = (-log(1 - u)/exp(X %*% beta))^(1/3)

    Cl = runif(n)
    Cr = Cl + runif(n)

    indicator = matrix(rep(0, n))
    a = runif(n)
    indicator[a <= p] = 1
```

```
    indicator[a > p] = 0

    L = matrix(rep(0, n))
    R = matrix(rep(0, n))
    status = matrix(rep(0, n))

    for (i in 1:n) {
        if (indicator[i] == 0) {
            L[i] = t[i]
            R[i] = t[i]
            status[i] = 0
        } else {
            if (t[i] <= Cl[i]) {
                L[i] = 0
                R[i] = Cl[i]
                status[i] = 1
            } else if (Cl[i] < t[i] && t[i] < Cr[i]) {
                L[i] = Cl[i]
                R[i] = Cr[i]
                status[i] = 2
            } else if (t[i] >= Cr[i]) {
                L[i] = Cr[i]
                R[i] = Inf
                status[i] = 3
            }
        }
    }
    return(cbind(L, R, status))
}
data = SimData(beta, X, p, n)
```

A.2 Classify data

```
#count: equal count in each bin
classification=function(data,count,n){
#summarize observations
fobs=matrix(data[,1][data[,3]==0])
nf=dim(fobs)[1]
lc=matrix(data[,2][data[,3]==1])
nl=dim(lc)[1]
ilc=matrix(data[,1][data[,3]==2])
irc=matrix(data[,2][data[,3]==2])
ni=dim(ilc)[1]
rc=matrix(data[,1][data[,3]==3])
nr=dim(rc)[1]
#combine all observations
obs=rbind(fobs,lc,ilc,irc,rc)
#order and distinct them
odobs=unique(sort(obs))
#equal number for each bin,
dn=length(odobs)
nbins=ceiling(dn/count)
#bin edg
binedg=matrix(0,nbins+1,1)
for (j in 2:nbins){
binedg[j]=odobs[(j-1)*count]
binedg[nbins+1]=max(odobs)
}
#bin width
binw=diff(binedg)
#matrix: number in each bin for the observations
nob=matrix(0,nbins,5)
idf=matrix(0,nf,1)
idl=matrix(0,nl,1)
idil=matrix(0,ni,1)
idir=matrix(0,ni,1)
```

```
idr=matrix(0,nr,1)
{if (nf==0){
nob[,1]=matrix(0,nbins,1)}
else{
for (i in 1:nf){
for (j in 1:nbins){
if (binedg[j]<fobs[i] & fobs[i]<=binedg[j+1]){
nob[j,1]=nob[j,1]+1
idf[i]=j}
}}
}
if (nl==0){
nob[,2]=matrix(0,nbins,1)}
else{
for (i in 1:nl){
for (j in 1:nbins){
if (binedg[j]<lc[i] & lc[i]<=binedg[j+1]){
nob[j,2]=nob[j,2]+1
idl[i]=j}
}}
}
if (ni==0){
nob[,3]=nob[,4]=matrix(0,nbins,1)}
else{
for (i in 1:ni){
for (j in 1:nbins){
if (binedg[j]<ilc[i] & ilc[i]<=binedg[j+1]){
nob[j,3]=nob[j,3]+1
idil[i]=j}
if (binedg[j]<irc[i] & irc[i]<=binedg[j+1])
{nob[j,4]=nob[j,4]+1
idir[i]=j}
}}
```

```
}
if (nr==0){
nob[,5]=matrix(0,nbins,1)}
else{
for (i in 1:nr){
for (j in 1:nbins){
if (binedg[j]<rc[i] & rc[i]<=binedg[j+1]){
nob[j,5]=nob[j,5]+1
idr[i]=j}
}}
}
}
#matrix: firts column contains observations,
#second column contains bin ID
classifyf=cbind(fobs,idf)
classifyl=cbind(lc,idl)
classifyil=cbind(ilc,idil)
classifyir=cbind(irc,idir)
classifyr=cbind(rc,idr)

return(list(nbins=nbins,nf=nf,nl=nl,ni=ni,nr=nr,
binw=binw,binedg=binedg,classifyf=classifyf,
classifyl=classifyl,classifyil=classifyil,

classifyir=classifyir,classifyr=classifyr,
nob=nob))
}
info=classification(data,count,n)
```

A.3 The MPL method

```
#penalty R matrix
#nbins: number of bins
pen=function(nbins){
M=matrix(0,nbins,nbins)
for (u in 1:nbins){
M[u,u]=6
}
for (u in 1:(nbins-1)){
M[u+1,u]=-4
M[u,u+1]=-4
}
for (u in 1:(nbins-2)){
M[u,u+2]=1
M[u+2,u]=1
}
M[1,1]=M[nbins,nbins]=1
M[2,2]=M[nbins-1,nbins-1]=5
M[2,1]=M[1,2]=M[nbins,nbins-1]=M[nbins-1,nbins]=-2
return(M)
}

#MPL estimation
#smooth: smoothing value
maxiter=5000
mplPH=function(data,X,count,n,maxiter,smooth){
info=classification(data,count,n)
nbins=info$nbins
#Penalty R matrix
R=pen(nbins)
#number of observations for each data type
nf=info$nf
nl=info$nl
ni=info$ni
```

```
nr=info$nr
nob=info$nob
#exactly observed failure time data
classifyf=info$classifyf
yf=classifyf[,1]
idf=classifyf[,2]
#left censored
classifyl=info$classifyl
yl=classifyl[,1]
idl=classifyl[,2]
#left endpoint for interval-censored data
classifyil=info$classifyil
yil=classifyil[,1]
idil=classifyil[,2]
#right endpoint for interval-censored data
classifyir=info$classifyir
yir=classifyir[,1]
idir=classifyir[,2]
#right-censored data
classifyr=info$classifyr
yr=classifyr[,1]
idr=classifyr[,2]
#bin width and points
binw=info$binw
binedg=info$binedg
p=dim(X)[2]
id=data[,3]
#covariate matrix for each data type
tX=X
X=X-matrix(rep(colMeans(X),each=n),ncol=p)
Xf=matrix(X[id==0],nf,p)
Xl=matrix(X[id==1],nl,p)
Xi=matrix(X[id==2],ni,p)
```

```
Xr=matrix(X[id==3],nr,p)
#initial values
eta=1e-1
beta0=matrix(0,p,1)
bh0=matrix(1,nbins,1)
oldbeta=beta0
oldbh=bh0
#cumulative baseline
oldcumbh=cumsum(binw*oldbh)
oldcumbh=c(0,oldcumbh)
#cumulative baseline hazard function
{if (nf==0){
oldcbhf=0}
else{
oldcbhf=oldcumbh[idf]+oldbh[idf]*(yf-binedg[idf])
}
if (nl==0){
oldcbhl=0}
else{
oldcbhl=oldcumbh[idl]+oldbh[idl]*(yl-binedg[idl])
}
if (ni==0){
oldcbhil=0
oldcbhir=0}
else{
oldcbhil=oldcumbh[idil]+oldbh[idil]*(yil-binedg[idil])
oldcbhir=oldcumbh[idir]+oldbh[idir]*(yir-binedg[idir])
}
if (nr==0){
oldcbhr=0}
else{
oldcbhr=oldcumbh[idr]+oldbh[idr]*(yr-binedg[idr])
}
```

```
}
#cumulative hazard function
oldvf=exp(Xf%*%oldbeta)
oldvl=exp(Xl%*%oldbeta)
oldvi=exp(Xi%*%oldbeta)
oldvr=exp(Xr%*%oldbeta)
oldchf=oldcbhf*oldvf
oldchl=oldcbhl*oldvl
oldchil=oldcbhil*oldvi
oldchir=oldcbhir*oldvi
oldchr=oldcbhr*oldvr
#survival function
oldsl=exp(-oldchl)
oldsil=exp(-oldchil)
oldsir=exp(-oldchir)
oldgl=1-oldsl
oldgi=oldsil-oldsir
cvg=matrix(0,maxiter,2)
#begin algorithm
for (iter in 1:maxiter){
#algorithm for updating beta
llik0=sum(log(oldgl))+sum(log(oldgi))-sum(oldchr)
+sum(Xf%*%oldbeta-oldchf)
#gradient for beta
oldeta=oldchir*oldsir-oldchil*oldsil
oldxi=(oldchir^2)*oldsir-(oldchil^2)*oldsil
A=(oldchl*oldsl)/oldgl
B=oldeta/oldgi
D=matrix(1,nf,1)-oldchf
{if (nl==1){
AA=t(Xl)%*%A%*%matrix(1,nl,1)}
else{
AA=t(Xl)%*%diag(c(A))%*%matrix(1,nl,1)
```

```
}
if (ni==1){
BB=t(Xi)%*%B%*%matrix(1,ni,1)}
else{
BB=t(Xi)%*%diag(c(B))%*%matrix(1,ni,1)
}
if (nr==1){
CC=t(Xr)%*%oldchr%*%matrix(1,nr,1)}
else{
CC=t(Xr)%*%diag(c(oldchr))%*%matrix(1,nr,1)
}
if (nf==1){
DD=t(Xf)%*%D%*%matrix(1,nf,1)}
else{
DD=t(Xf)%*%diag(c(D))%*%matrix(1,nf,1)
}
}
bgrad=AA+BB-CC+DD
#Hessian matrix for beta
E=(oldchl*oldsl*(oldsl-1+oldchl))/(oldgl)^2
F=(oldxi-oldeta)/oldgi+(oldeta/oldgi)^2
{if (nl==1){
EE=t(Xl)%*%E%*%Xl}
else{
EE=t(Xl)%*%diag(c(E))%*%Xl
}
if (ni==1){
FF=t(Xi)%*%F%*%Xi}
else{
FF=t(Xi)%*%diag(c(F))%*%Xi
}
if (nr==1){
CCC=t(Xr)%*%oldchr%*%Xr}
```

```
else{
CCC=t(Xr)%*%diag(c(oldchr))%*%Xr
}
if (nf==1){
GG=t(Xf)%*%oldchf%*%Xf}
else{
GG=t(Xf)%*%diag(c(oldchf))%*%Xf
}
}
Hesb=-EE-FF-CCC-GG
#new estimates for beta
incb=solve(Hesb)%*%bgrad
newbeta=oldbeta-incb
#half new cumulative hazard and survival
newvf=exp(Xf%*%newbeta)
newvl=exp(Xl%*%newbeta)
newvi=exp(Xi%*%newbeta)
newvr=exp(Xr%*%newbeta)
hnewchf=oldcbhf*newvf
hnewchl=oldcbhl*newvl
hnewchil=oldcbhil*newvi
hnewchir=oldcbhir*newvi
hnewchr=oldcbhr*newvr
hnewsl=exp(-hnewchl)
hnewsil=exp(-hnewchil)
hnewsir=exp(-hnewchir)
hnewgl=1-hnewsl
hnewgi=hnewsil-hnewsir
llik1=sum(log(hnewgl))+sum(log(hnewgi))-sum(hnewchr)
+sum(Xf%*%newbeta-hnewchf)
ome=0.6
while(llik1<=llik0){
#newbeta
```

```
newbeta=oldbeta-ome*incb
#half new cumulative hazard and survival functions
newvf=exp(Xf%*%newbeta)
newvl=exp(Xl%*%newbeta)
newvi=exp(Xi%*%newbeta)
newvr=exp(Xr%*%newbeta)
hnewchf=oldcbhf*newvf
hnewchl=oldcbhl*newvl
hnewchil=oldcbhil*newvi
hnewchir=oldcbhir*newvi
hnewchr=oldcbhr*newvr
hnewsl=exp(-hnewchl)
hnewsil=exp(-hnewchil)
hnewsir=exp(-hnewchir)
hnewgl=1-hnewsl
hnewgi=hnewsil-hnewsir
llik1=sum(log(hnewgl))+sum(log(hnewgi))-sum(hnewchr)
+sum(Xf%*%newbeta-hnewchf)
if (ome>=1e-2)
ome=ome*0.6
else if (ome < 1e-2 & ome >= 1e-5)
ome = ome*5e-2
else if (ome<1e-5 & ome>1e-20)
ome = ome*1e-5
else
break
}
llik1=sum(log(hnewgl))+sum(log(hnewgi))-sum(hnewchr)
+sum(nob[,1]*log(oldbh))-sum(hnewchf)-
(smooth/(1-smooth))*t(oldbh)%*%R%*%oldbh
dpen=2*(smooth/(1-smooth))*R%*%oldbh
#new estimates for baseline hazrd
a1=matrix(0,(nbins-1),1)
```

```
c1=matrix(0,(nbins-1),1)
d1=matrix(0,(nbins-1),1)
a=matrix(0,nbins,1)
b1=matrix(0,nbins,1)
b2=matrix(0,nbins,1)
c=matrix(0,nbins,1)
d=matrix(0,nbins,1)
for (i in 1:(nbins-1)){
if (nl==0){
a1[i]=0}
else{
a1[i]=sum(newvl[idl==i+1]*hnewsl[idl==i+1]
/hnewgl[idl==i+1])
}
if (nr==0){
c1[i]=0}
else{
c1[i]=sum(newvr[idr==i+1])
}
if (nf==0){
d1[i]=0}
else{
d1[i]=sum(newvf[idf==i+1])
}
}
cuma1=c(rev(cumsum(rev(a1))),0)
cumc1=c(rev(cumsum(rev(c1))),0)
cumd1=c(rev(cumsum(rev(d1))),0)
for (i in 1:nbins){
if (nl==0){
a[i]=0}
else{
a[i]=sum(newvl[idl==i]*(yl[idl==i]-binedg[i])
```

```
*hnewsl[idl==i]/hnewgl[idl==i])+binw[i]*cuma1[i]
}
if (ni==0){
b1[i]=0
b2[i]=0}
else{
for (j in 1:ni){
if (idil[j]<i & idir[j]==i)
b2[i]=b2[i]+newvi[j]*(yir[j]-binedg[i])*hnewsir[j]
/hnewgi[j]
else if (idil[j]<i & idir[j]>i)
b2[i]=b2[i]+binw[i]*newvi[j]*hnewsir[j]/hnewgi[j]
else if (idil[j]==i & idir[j]==i){
b1[i]=b1[i]+newvi[j]*(yil[j]-binedg[i])*hnewsil[j]
/hnewgi[j]
b2[i]=b2[i]+newvi[j]*(yir[j]-binedg[i])*hnewsir[j]
/hnewgi[j]}
else if (idil[j]==i & idir[j]>i){
b1[i]=b1[i]+newvi[j]*(yil[j]-binedg[i])*hnewsil[j]
/hnewgi[j]
b2[i]=b2[i]+binw[i]*newvi[j]*hnewsir[j]/hnewgi[j]}
else if (idil[j]>i & idir[j]>i){
b1[i]=b1[i]+binw[i]*newvi[j]}
else{
b1[i]=b1[i]+0
b2[i]=b2[i]+0}
}
}
if (nr==0){
c[i]=0}
else{
c[i]=sum(newvr[idr==i]*(yr[idr==i]-binedg[i]))
+binw[i]*cumc1[i]
```

```
}
if (nf==0){
d[i]=0}
else{
d[i]=sum(newvf[idf==i]*(yf[idf==i]-binedg[i]))
+binw[i]*cumd1[i]
}
}
nume=a+b2+nob[,1]/oldbh-matrix(apply(dpen,1,
function(x) min(x,0)))
deno=b1+c+d+matrix(apply(dpen,1,function(x) max(x,0)))
newbh=oldbh*((nume+eta)/(deno+eta))
incbh=newbh-oldbh
cumbh=cumsum(binw*newbh)
cumbh=c(0,cumbh)
#new cumulative baseline hazard function
{if (nf==0){
newcbhf=0}
else{
newcbhf=cumbh[idf]+newbh[idf]*(yf-binedg[idf])
}
if (nl==0){
newcbhl=0}
else{
newcbhl=cumbh[idl]+newbh[idl]*(yl-binedg[idl])
}
if (ni==0){
newcbhil=0
newcbhir=0}
else{
newcbhil=cumbh[idil]+newbh[idil]*(yil-binedg[idil])
newcbhir=cumbh[idir]+newbh[idir]*(yir-binedg[idir])
}
```

```
if (nr==0){
newcbhr=0}
else{
newcbhr=cumbh[idr]+newbh[idr]*(yr-binedg[idr])
}
}
#new cumulative hazard, sruvival functions
newchf=newcbhf*newvf
newchl=newcbhl*newvl
newchil=newcbhil*newvi
newchir=newcbhir*newvi
newchr=newcbhr*newvr
newsl=exp(-newchl)
newsil=exp(-newchil)
newsir=exp(-newchir)
#new g function
newgl=1-newsl
newgi=newsil-newsir
llik2=sum(log(newgl))+sum(log(newgi))-sum(newchr)
+sum(nob[,1]*log(newbh))-sum(newchf)-
(smooth/(1-smooth))*t(newbh)%*%R%*%newbh
ome=0.6
while(llik2<=llik1){
newbh=oldbh+ome*incbh
cumbh=cumsum(binw*newbh)
cumbh=c(0,cumbh)
#new cumulative baseline hazard function
{if (nf==0){
newcbhf=0}
else{
newcbhf=cumbh[idf]+newbh[idf]*(yf-binedg[idf])
}
if (nl==0){
```

```
newcbhl=0}
else{
newcbhl=cumbh[idl]+newbh[idl]*(yl-binedg[idl])
}
if (ni==0){
newcbhil=0
newcbhir=0}
else{
newcbhil=cumbh[idil]+newbh[idil]*(yil-binedg[idil])
newcbhir=cumbh[idir]+newbh[idir]*(yir-binedg[idir])
}
if (nr==0){
newcbhr=0}
else{
newcbhr=cumbh[idr]+newbh[idr]*(yr-binedg[idr])
}
}
```

```
#new cumulative hazard, sruvival functions
newchf=newcbhf*newvf
newchl=newcbhl*newvl
newchil=newcbhil*newvi
newchir=newcbhir*newvi
newchr=newcbhr*newvr
newsl=exp(-newchl)
newsil=exp(-newchil)
newsir=exp(-newchir)
#new g function
newgl=1-newsl
```

```
newgi=newsil-newsir
llik2=sum(log(newgl))+sum(log(newgi))-sum(newchr)
+sum(nob[,1]*log(newbh))-sum(newchf)-
(smooth/(1-smooth))*t(newbh)%*%R%*%newbh
if (ome>=1e-2)
ome=ome*0.6
else if (ome < 1e-2 & ome >= 1e-5)
ome = ome*5e-2
else if (ome<1e-5 & ome>1e-20)
ome = ome*1e-5
else
break
}
plik=sum(log(newgl))+sum(log(newgi))-sum(newchr)
+sum(nob[,1]*log(newbh))+sum(Xf%*%newbeta-newchf)-
(smooth/(1-smooth))*t(newbh)%*%R%*%newbh
cvg[iter,1]=iter
cvg[iter,2]=plik
if (max(abs(newbh-oldbh))<1e-5 &
max(abs(newbeta-oldbeta))<1e-5)
{
cvg = cvg[1:iter,]
break
}
else{
oldbh=newbh
oldbeta=newbeta
oldcbhf=newcbhf
oldcbhl=newcbhl
oldcbhil=newcbhil
oldcbhir=newcbhir
oldcbhr=newcbhr
oldchf=newchf
```

```
oldchl=newchl
oldchil=newchil
oldchir=newchir
oldchr=newchr
oldsl=newsl
oldsil=newsil
oldsir=newsir
oldgl=newgl
oldgi=newgi
}
}
bh= newbh*exp(-colMeans(tX)%*%newbeta)[1]
{if (nf==0){
newcbhf=0}
else{
newcbhf=cumbh[idf]+bh[idf]*(yf-binedg[idf])
}
if (nl==0){
newcbhl=0}
else{
newcbhl=cumbh[idl]+bh[idl]*(yl-binedg[idl])
}
if (ni==0){
newcbhil=0
newcbhir=0}
else{
newcbhil=cumbh[idil]+bh[idil]*(yil-binedg[idil])
newcbhir=cumbh[idir]+bh[idir]*(yir-binedg[idir])
}
if (nr==0){
newcbhr=0}
else{
newcbhr=cumbh[idr]+bh[idr]*(yr-binedg[idr])
```

```
}
}
#new cumulative hazard, sruvival functions
newchf=newcbhf*newvf
newchl=newcbhl*newvl
newchil=newcbhil*newvi
newchir=newcbhir*newvi
newchr=newcbhr*newvr
newsl=exp(-newchl)
newsil=exp(-newchil)
newsir=exp(-newchir)
#new g function
newgl=1-newsl
newgi=newsil-newsir
llik=sum(log(newgl))+sum(log(newgi))-sum(newchr)
+sum(nob[,1]*log(bh))+sum(Xf%*%newbeta-newchf)
#Hessian matrix for beta
neweta=newchir*newsir-newchil*newsil
newxi=(newchir^2)*newsir-(newchil^2)*newsil
E=(newchl*newsl*(newsl-1+newchl))/(newgl)^2
F=(newxi-neweta)/newgi+(neweta/newgi)^2
{if (nl==1){
EE=t(Xl)%*%E%*%Xl}
else{
EE=t(Xl)%*%diag(c(E))%*%Xl
}
if (ni==1){
FF=t(Xi)%*%F%*%Xi}
else{
FF=t(Xi)%*%diag(c(F))%*%Xi
}
if (nr==1){
CCC=t(Xr)%*%newchr%*%Xr}
```

```
else{
CCC=t(Xr)%*%diag(c(newchr))%*%Xr
}
if (nf==1){
GG=t(Xf)%*%newchf%*%Xf}
else{
GG=t(Xf)%*%diag(c(newchf))%*%Xf
}
}
Hesb=-EE-FF-CCC-GG
#Hessian matrix for theta
{if (nl==0){
tHesl=matrix(0,nbins,nbins)
}
else{
suml=matrix(0,(nbins-1),1)
for (u in 1:(nbins-1)){
suml[u]=-sum((newsl[idl==u+1]*
(newvl[idl==u+1])^2)/(newgl[idl==u+1])^2)
}
cuml=c(rev(cumsum(rev(suml))),0)
Al=matrix(0,nbins,1)
for (u in 1:nbins){
Al[u]=-sum((newsl[idl==u]*((yl[idl==u]-binedg[u])
*newvl[idl==u])^2)/(newgl[idl==u])^2)+
cuml[u]*(binw[u])^2
}
tHesl=diag(c(Al))
Bl=matrix(0,(nbins-1),1)
for (u in 1:(nbins-1)){
Bl[u]=-sum((newsl[idl==u+1]*(yl[idl==u+1]-
binedg[u+1])*(newvl[idl==u+1])^2)
/(newgl[idl==u+1])^2)
```

```
}
for (i in 2:nbins){
for (j in 1:(i-1)){
tHesl[i,j]=binw[j]*Bl[i-1]+binw[j]*cuml[i]*binw[i]
}
}
for (i in 1:(nbins-1)){
for (j in (i+1):nbins){
tHesl[i,j]=tHesl[j,i]
}
}
}
}

{if (ni==0){
tHesi=matrix(0,nbins,nbins)
}
else{
Ai=matrix(0,nbins,1)
for (u in 1:nbins){
for (i in 1:ni){
if (idil[i]<u & idir[i]==u)
Ai[u]=Ai[u]-(newsil[i]*newsir[i]*
((yir[i]-binedg[u])*newvi[i])^2)/(newgi[i])^2
else if (idil[i]<u & idir[i]>u)
Ai[u]=Ai[u]-(newsil[i]*newsir[i]*
(binw[u]*newvi[i])^2)/(newgi[i])^2
else if (idil[i]==u & idir[i]==u)
Ai[u]=Ai[u]-(newsil[i]*newsir[i]*
((yil[i]-yir[i])*newvi[i])^2)/(newgi[i])^2
else if (idil[i]==u & idir[i]>u)
Ai[u]=Ai[u]-(newsil[i]*newsir[i]*
((yil[i]-binedg[u]-binw[u])*newvi[i])^2)
```

```
/(newgi[i])^2
else
Ai[u]=Ai[u]+0
}
}
tHesi=diag(c(Ai))
for (i in 2:nbins){
for (j in 1:(i-1)){
for (k in 1:ni){
if (idil[k]<j & idir[k]==i)
tHesi[i,j]=tHesi[i,j]-newsil[k]*newsir[k]*binw[j]*
(yir[k]-binedg[i])*(newvi[k])^2/(newgi[k])^2
else if (idil[k]==j & idir[k]==i)
tHesi[i,j]=tHesi[i,j]-newsil[k]*newsir[k]*
(yir[k]-binedg[i])*(binw[j]-yil[k]+binedg[j])*
(newvi[k])^2/(newgi[k])^2
else if (idil[k]<j & idir[k]>i)
tHesi[i,j]=tHesi[i,j]-newsil[k]*newsir[k]*
binw[i]*binw[j]*(newvi[k])^2/(newgi[k])^2
else if (idil[k]==j & idir[k]>i)
tHesi[i,j]=tHesi[i,j]-binw[i]*(binw[j]-yil[k]+binedg[j])
*newsil[k]*newsir[k]*(newvi[k])^2/(newgi[k])^2
else
tHesi[i,j]=tHesi[i,j]+0
}
}
}
for (i in 1:(nbins-1)){
for (j in (1+i):nbins){
tHesi[i,j]=tHesi[j,i]
}
}
}
```

```
}

{if (nf==0){
Af=matrix(0,nbins,1)
}
else{
Af=matrix(0,nbins,1)
for (u in 1:nbins){
Af[u]=-nob[u,1]/(bh[u])^2
}
}
}
tHesf=diag(c(Af))
tHes=tHesl+tHesi+tHesf

#second derivative with repective to beta and theta
summl=matrix(0,p,(nbins-1))
summr=matrix(0,p,(nbins-1))
summf=matrix(0,p,(nbins-1))
for (u in 1:(nbins-1)){
if (nl==0){
summl[,u]=matrix(0,p,1)
}
else{
summl[,u]=-t(matrix(Xl[idl==u+1],nob[u+1,2],p))%*%
((newchl[idl==u+1]+newsl[idl==u+1]-1)*newvl[idl==u+1]
*newsl[idl==u+1]/(newgl[idl==u+1])^2)
}
if (nr==0){
summr[,u]=matrix(0,p,1)
}
else{
summr[,u]=-t(matrix(Xr[idr==u+1],nob[u+1,5],p))
```

```
%*%newvr[idr==u+1]
}
if (nf==0){
summf[,u]=matrix(0,p,1)
}
else{
summf[,u]=-t(matrix(Xf[idf==u+1],nob[u+1,1],p))
%*%newvf[idf==u+1]
}
}
cumml=matrix(0,p,(nbins-1))
cummr=matrix(0,p,(nbins-1))
cummf=matrix(0,p,(nbins-1))
for (u in 1:(nbins-1)){
cumml[,u]=summl[,u:(nbins-1)]%*%matrix(1,(nbins-u),1)
cummr[,u]=summr[,u:(nbins-1)]%*%matrix(1,(nbins-u),1)
cummf[,u]=summf[,u:(nbins-1)]%*%matrix(1,(nbins-u),1)
}
cumml=cbind(cumml,matrix(0,p,1))
cummr=cbind(cummr,matrix(0,p,1))
cummf=cbind(cummf,matrix(0,p,1))
dbtl=matrix(0,p,nbins)
dbti=matrix(0,p,nbins)
dbtr=matrix(0,p,nbins)
dbtf=matrix(0,p,nbins)
for (u in 1:nbins){
if (nl==0){
dbtl[,u]=matrix(0,p,1)
}
else{
dbtl[,u]=-t(matrix(Xl[idl==u],nob[u,2],p))%*%
((newchl[idl==u]+newsl[idl==u]-1)*(yl[idl==u]-binedg[u])
*newvl[idl==u]*newsl[idl==u]/(newgl[idl==u])^2)
```

```
+binw[u]*cumml[,u]
}
if (ni==0){
dbti[,u]=matrix(0,p,1)
}
else{
for (i in 1:ni){
if (idil[i]<u & idir[i]==u)
dbti[,u]=dbti[,u]+t(Xi[i,])*((yir[i]-binedg[u])*newvi[i]*
newsir[i]*(newgi[i]+newsil[i]*(newchil[i]-newchir[i]))
/(newgi[i])^2)
else if (idil[i]<u & idir[i]>u)
dbti[,u]=dbti[,u]+t(Xi[i,])*(binw[u]*newvi[i]*newsir[i]*
(newgi[i]+newsil[i]*(newchil[i]-newchir[i]))
/(newgi[i])^2)
else if (idil[i]==u & idir[i]==u)
dbti[,u]=dbti[,u]+t(Xi[i,])*((newgi[i]*((yir[i]-binedg[u])
*newvi[i]*newsir[i]-(yil[i]-binedg[u])*newvi[i]*newsil[i])
+(newchil[i]-newchir[i])*newsil[i]*newsir[i]*(yir[i]-yil[i])
*newvi[i])/(newgi[i])^2)
else if (idil[i]==u & idir[i]>u)
dbti[,u]=dbti[,u]+t(Xi[i,])*((newgi[i]*(binw[u]*newvi[i]
*newsir[i]-(yil[i]-binedg[u])*newvi[i]*newsil[i])+
(newchil[i]-newchir[i])*newsil[i]*newsir[i]*
(binw[u]-yil[i]+binedg[u])*newvi[i])/(newgi[i])^2)
else if (idil[i]>u & idir[i]>u)
dbti[,u]=dbti[,u]-t(Xi[i,])*(binw[u]*newvi[i])
else
dbti[,u]=dbti[,u]+matrix(0,p,1)
}
}
if (nr==0){
dbtr[,u]=matrix(0,p,1)
```

```
}
else{
dbtr[,u]=-t(matrix(Xr[idr==u],nob[u,5],p))
%*%((yr[idr==u]-binedg[u])*newvr[idr==u])
+binw[u]*cummr[,u]
}
if (nf==0){
dbtf[,u]=matrix(0,p,1)
}
else{
dbtf[,u]=-t(matrix(Xf[idf==u],nob[u,1],p))
%*%((yf[idf==u]-binedg[u])*newvf[idf==u])
+binw[u]*cummf[,u]
}
}
dbt=dbtl+dbti+dbtr+dbtf
#Hessian matrix
H=rbind(cbind(Hesb,dbt),cbind(t(dbt),tHes))
HesbinvD=matrix(diag(solve(-Hesb)))
tHesp=-tHes+2*smooth*R/(1-smooth)
tHespinvD=matrix(diag(solve(tHesp)))
Hessp=rbind(cbind(-Hesb,-dbt),cbind(-t(dbt),tHesp))
#dHessp=diag(Hessp); idHessp=diag(1/sqrt(dHessp))
#sHessp=idHessp%*%Hessp%*%idHessp
#sHessp is better scaled than Hessp
#Hesspinv=idHessp%*%(solve(sHessp)%*%idHessp)
#varcov=Hesspinv%*%(-H)%*%Hesspinv
varcov=solve(Hessp)%*%(-H)%*%solve(Hessp)
{if (p==1){
betavar=varcov[1:p,1:p]
}
else{
betavar=matrix(diag(varcov[1:p,1:p]))
```

```
}
}
betavar[betavar<0]=HesbinvD[betavar<0]
bhvar=matrix(diag(varcov[(p+1):(p+nbins),(p+1):(p+nbins)]))
bhvar[bhvar<0]=tHespinvD[bhvar<0]

#selection criterion for smoothing value
df=sum(diag(solve(-Hessp)%*%H))
AIC=-llik+df

#plot of baseline hazard
time.point=seq(0.05,1,length=50)
tbh=3*time.point^2
ebh=matrix(0,50,1)
ebhvar=matrix(0,50,1)
for (i in 1:50){
for (j in 1:nbins){
if (binedg[j]<time.point[i] & time.point[i]<=binedg[j+1]){
ebh[i]=bh[j]
ebhvar[i]=bhvar[j]
}
}
}
nd=500
minv=min(time.point)
maxv=max(time.point)
delta=(maxv-minv)/nd
bin.point=matrix(seq(minv,maxv,delta))
mp=matrix(0,nd,1)
for (i in 1:nd){
mp[i]=(bin.point[i]+bin.point[i+1])/2
}
ISEs=matrix(0,nd,1)
```

```
for (i in 1:nd){
for (j in 1:nbins){
if (binedg[j]<mp[i] & mp[i]<=binedg[j+1]){
ISEs[i]=(3*mp[i]^2-bh[j])^2
}
}
}
ISE=delta*sum(ISEs)
par(mfrow=c(2,1))
plot(cvg[,1],cvg[,2],type='l',xlab='iteration',
ylab='cvg',main='convergence view')
plot(time.point,tbh,type='l',col='red',
ylab='baseline hazard function',
main='ture hazard v.s estimated hazard')
lines(time.point,ebh,type='l',col='blue')
return(list(binedg=binedg,newbeta=newbeta,ebh=ebh,
betavar=betavar,ebhvar=ebhvar,ISE=ISE,AIC=AIC))
}
phfit=mplPH(data,X,count,n,maxiter,smooth)
```

A.4 Monte Carlo simulation

```
#Select the smoothing value
#smv: vector of smoothing values
selSm<-function(data,X,count,n,maxiter,smv){
time.point=seq(0.05,1,length=50)
tbh=3*time.point^2
ns=dim(smv)[1]
id=matrix(seq(1,ns,1))
AICs=matrix(0,ns,1)
newbetas=matrix(0,ns,3)
betavars=matrix(0,ns,3)
```

```
ISEs=matrix(0,ns,1)
ebhs=matrix(0,50,ns)
for (i in 1:ns){
phfit=mplPH(data,X,count,n,maxiter,smv[i])
AICs[i]=phfit$AIC
newbetas[i,]=matrix(phfit$newbeta,nrow=1)
betavars[i,]=matrix(phfit$betavar,nrow=1)
ebhs[,i]=matrix(phfit$ebh)
ISEs[i]=phfit$ISE
print(i)
}
AICm=min(AICs)
id=id[AICs==AICm]
smop=smv[id]
newbetaop=newbetas[id,]
betavarop=betavars[id,]
ISEop=ISEs[id]
ebhop=ebhs[,id]
return(list(smop=smop,AICm=AICm,AICs=AICs,
newbetaop=newbetaop,
betavarop=betavarop,ISEop=ISEop,

newbetas=newbetas,betavars=betavars,ISEs=ISEs))
}
SmSel=selSm(data,X,count,n,maxiter,smv)

#Monte Carlo Simulation.
#n: sample size.
#p: censoring proportion.
#count: equal count of observations in each bin.
#beta: beta values.
#N: N Monte Carlo samples.
#maxiter: number of iterations in the Newton-MI algorithm.
```

```
#smop: smoothing value selected by SmSel.
MCsimu<-function(n,p,count,beta,N,maxiter,smop){
#initials
np=dim(beta)[1]
betas=matrix(0,N,np)
betavars=matrix(0,N,np)
time.point=seq(0.05,1,length=50)
ebhs=matrix(0,50,N)
ebhvars=matrix(0,50,N)
ISEs=matrix(0,N,1)
#Monte Carlo simulation
for(i in 1:N){
#categorial variable
Ca=matrix(rep(0,n))
ur=runif(n)
Ca[ur<=0.5]=1
Ca[0.5<ur & ur<=1]=0
#Covariate matrix
X=cbind(Ca,runif(n,min=0,max=3),runif(n,min=0,max=5))
# data generation
data=SimData(beta,X,p,n)
phfit=mplPH(data,X,count,n,maxiter,smop)
betas[i,]=matrix(phfit$newbeta,nrow=1)
ebhs[,i]=matrix(phfit$ebh)
betavars[i,]=matrix(phfit$betavar,nrow=1)
ebhvars[,i]=matrix(phfit$ebhvar)
ISEs[i]=phfit$ISE
print(i)
}
#average of integrated squared error
mISE=mean(ISEs)
#mean, bias, std, astd, mse of beta
meanb=matrix(colMeans(betas),1,np)
```

```
biasbeta=t(beta)-meanb
varb=matrix(0,1,np)
for (j in 1:np){
varb[1,j]=var(betas[,j])
}
stdb=sqrt(varb)
astdb=matrix(colMeans(sqrt(betavars)),1,np)
mseb=biasbeta^2+varb
#summary for beta
summaryb=rbind(meanb,biasbeta,stdb,astdb,mseb)
#sample mean, std and astd of baseline hazard estimate
meanbh=matrix(colMeans(t(ebhs)))
varbh=matrix(0,50,1)
for (j in 1:50){
varbh[j]=var(ebhs[j,])
}
#MC standard deviation
stdbh=sqrt(varbh)
#Average asymptotic standard deviation
aastdbh=matrix(colMeans(t(sqrt(ebhvars))))
#summary for the baseline hazard
# 95% MC PWCI
mclbh=meanbh-1.96*stdbh
mcubh=meanbh+1.96*stdbh
# average of 95% asymptotic PWCI
aasylbh=meanbh-1.96*aastdbh
aasyubh=meanbh+1.96*aastdbh
mclbh[mclbh<0]=0
aasylbh[aasylbh<0]=0
summarybh=cbind(meanbh,mclbh,mcubh,aasylbh,aasyubh)
return(list(summaryb=summaryb,summarybh=summarybh,
mISE=mISE))
}
```

```
sim=MCsimu(n,p,count,beta,N,maxiter,smop)
```

A.5 Method comparison

```
#Datasets used for method comparisons
beta=matrix(c(1,-0.3,0.5))
GenerateData=function(beta,n,p,N){
obsets=matrix(0,n,3*N)
Xs=matrix(0,n,3*N)
for (i in 1:N){
Ca=matrix(rep(0,n))
ur=runif(n)
Ca[ur<=0.5]=1
Ca[ur>0.5]=0
#Covariate matrix
Xs[,(3*i-2):(3*i)]=cbind(Ca,runif(n,min=0,max=3),
runif(n,min=0,max=5))
data=SimData(beta,Xs[,(3*i-2):(3*i)],p,n)
obsets[,(3*i-2):(3*i)]=data
}
return(list(obsets=obsets,Xs=Xs))
}
dataset=GenerateData(beta,n,p,N)
datas=dataset$obsets
Xs=dataset$Xs

datasets=matrix(0,n,5*N)
int.datas=matrix(0,n,2*N)
for (i in 1:N){
int.datas[,(2*i-1):(2*i)]=datas[,(3*i-2):(3*i-1)]
int.datas[,2*i][int.datas[,2*i]==Inf]=NA
datasets[,(5*i-4):(5*i-3)]=int.datas[,(2*i-1):(2*i)]
```

```
datasets[,(5*i-2):(5*i)]=Xs[,(3*i-2):(3*i)]
}
colnames(datasets)=rep(c('L','R','x1','x2','x3'),N)

#MC simulations for the method from Pan (1999)
pan.sim=function(datasets,n,N){
times=seq(0.05,1,length=50)
betas=matrix(0,N,3)
ebhs=matrix(0,50,N)
ISEs=matrix(0,N,1)
for (i in 1:N){
dataset=data.frame(datasets[,(5*i-4):(5*i)])
pan.fit=intcox(Surv(L, R, type = "interval2")
~x1 + x2+x3, data = dataset)
betas[i,]=matrix(pan.fit$coef,nrow=1)
time.value=matrix(pan.fit$time.point)
cumhaz=matrix(pan.fit$lambda0)
bh=rbind(0,diff(cumhaz))
ntp=dim(time.value)[1]
ebh=matrix(0,50,1)
for (k in 1:50){
for (j in 1:(ntp-1)){
if (time.value[j]<times[k] & times[k]<=time.value[j+1]){
ebh[k]=((times[k]-time.value[j])/(time.value[j+1]-time.value[j]))
*bh[j]+((time.value[j+1]-times[k])/(time.value[j+1]-time.value[j]))
*bh[j+1]
}
}
}
ebhs[,i]=ebh
nd=500
minv=min(times)
maxv=max(times)
```

```
delta=(maxv-minv)/nd
bin.point=matrix(seq(minv,maxv,delta))
mp=matrix(0,nd,1)
for (k in 1:nd){
mp[k]=(bin.point[k]+bin.point[k+1])/2
}
ISE=matrix(0,nd,1)
embh=matrix(0,nd,1)
for (k in 1:nd){
for (j in 1:(ntp-1)){
if (time.value[j]<mp[k] & mp[k]<=time.value[j+1]){
embh[k]=((mp[k]-time.value[j])/(time.value[j+1]-time.value[j]))
*bh[j]+((time.value[j+1]-mp[k])/(time.value[j+1]-time.value[j]))
*bh[j+1]
ISE[k]=(3*mp[k]^2-embh[k])^2
}
}
}
ISEs[i]=delta*sum(ISE)
print(i)
}
mISE=mean(ISEs)
meanb=matrix(colMeans(betas),1,3)
biasbeta=t(beta)-meanb
varb=matrix(0,1,3)
for (j in 1:3){
varb[1,j]=var(betas[,j])
}
stdb=sqrt(varb)
mseb=biasbeta^2+varb
summaryb=rbind(meanb,biasbeta,stdb,mseb)
meanbh=matrix(colMeans(t(ebhs)))
varbh=matrix(0,50,1)
```

```
for (j in 1:50){
varbh[j]=var(ebhs[j,])
}
stdbh=sqrt(varbh)
mclbh=meanbh-1.96*stdbh
mcubh=meanbh+1.96*stdbh
mclbh[mclbh<0]=0
summarybh=cbind(meanbh,mclbh,mcubh)
return(list(summaryb=summaryb,betas=betas,
summarybh=summarybh,mISE=mISE))
}
pansim=pan.sim(datasets,n,N)


#Monte Carlo simulations for our MPL method
MCsimu<-function(datas,Xs,n,count,N,maxiter,smop){
#initials
np=dim(beta)[1]
betas=matrix(0,N,np)
betavars=matrix(0,N,np)
time.point=seq(0.05,1,length=50)
ebhs=matrix(0,50,N)
ebhvars=matrix(0,50,N)
ISEs=matrix(0,N,1)
#MC simulation
for(i in 1:N){
phfit=mplPH(datas[,(3*i-2):(3*i)],Xs[,(3*i-2):(3*i)],
count,n,maxiter,smop)
betas[i,]=matrix(phfit$newbeta,nrow=1)
ebhs[,i]=matrix(phfit$ebh)
betavars[i,]=matrix(phfit$betavar,nrow=1)
ebhvars[,i]=matrix(phfit$ebhvar)
ISEs[i]=phfit$ISE
```

```
print(i)
}
mISE=mean(ISEs)
#mean, bias, std, astd and mse of beta
meanb=matrix(colMeans(betas),1,np)
biasbeta=t(beta)-meanb
varb=matrix(0,1,np)
for (j in 1:np){
varb[1,j]=var(betas[,j])
}
stdb=sqrt(varb)
astdb=matrix(colMeans(sqrt(betavars)),1,np)
mseb=biasbeta^2+varb
#summary for beta
summaryb=rbind(meanb,biasbeta,stdb,astdb,mseb)
#mean, std and astd of baseline hazard estimate
meanbh=matrix(colMeans(t(ebhs)))
varbh=matrix(0,50,1)
for (j in 1:50){
varbh[j]=var(ebhs[j,])
}
#MC standard deviation
stdbh=sqrt(varbh)
#Average asymptotic standard deviation
aastdbh=matrix(colMeans(t(sqrt(ebhvars))))
#summary for the baseline hazard estimate
# 95% MC PWCI
mclbh=meanbh-1.96*stdbh
mcubh=meanbh+1.96*stdbh
# average of 95% asymptotic PWCI
aasylbh=meanbh-1.96*aastdbh
aasyubh=meanbh+1.96*aastdbh
mclbh[mclbh<0]=0
```

```
aasylbh[aasylbh<0]=0
summarybh=cbind(meanbh,mclbh,mcubh,aasylbh,aasyubh)
return(list(summaryb=summaryb,summarybh=summarybh,
mISE=mISE))
}
sim=MCsimu(datas,Xs,n,count,N,maxiter,smop)
```

Appendix B R codes for AH model

B.1 Data generation

```
# Generate data
#beta values
beta=matrix(c(1,-0.3,0.5))
#n: sample size.
#p: censoring proportion.
event <- function(beta,p,n){
t=matrix(0,n,1)
X=matrix(0,n,3)
for (i in 1:n){
repeat{
Ca=0
ur=runif(1)
Ca[ur<=0.5]=1
Ca[ur>0.5]=0
X[i,]=c(Ca,runif(1,min=0,max=3),

        runif(1,min=0,max=5))

t1=log(1-runif(1))
t2=X[i,]%*%beta
f=function(x)(x^3)+x*t2[1]+t1[1]
t[i]=uniroot.all(f,c(0,10))
if ((t[i]!=0) & (length(t[i])!=0))
```

```
break
}
}

Cl=runif(n)
Cr=Cl+runif(n)

indicator=matrix(rep(0,n))
sw=runif(n)
indicator[sw<=p]=1
indicator[sw>p]=0

L=matrix(rep(0,n))
R=matrix(rep(0,n))
status=matrix(rep(0,n))

for (i in 1:n){
if (indicator[i]==0) {L[i]=t[i]
R[i]=t[i]
status[i]=0}
else{
if (t[i]<=Cl[i]) {L[i]=0
R[i]=Cl[i]
status[i]=1}
else if (Cl[i]<t[i] && t[i]<=Cr[i]) {L[i]=Cl[i]
R[i]=Cr[i]
status[i]=2}
else if (t[i]>Cr[i]) {L[i]=Cr[i]
R[i]=Inf
status[i]=3}
}
}
return(cbind(L,R,status,X))
```

```
}
dataset=event(beta,p,n)

data=dataset[,1:3]
X=dataset[,4:6]
```

B.2 Data classification

```
#count: equal count in each bin
classification=function(data,count,n){
#summarize observations
fobs=matrix(data[,1][data[,3]==0])
nf=dim(fobs)[1]
lc=matrix(data[,2][data[,3]==1])
nl=dim(lc)[1]
ilc=matrix(data[,1][data[,3]==2])
irc=matrix(data[,2][data[,3]==2])
ni=dim(ilc)[1]
rc=matrix(data[,1][data[,3]==3])
nr=dim(rc)[1]
#combine all observations
obs=rbind(fobs,lc,ilc,irc,rc)
#order and distinct them
odobs=unique(sort(obs))
#equal number for each bin
dn=length(odobs)
nbins=ceiling(dn/count)
#bin edg
binedg=matrix(0,nbins+1,1)
for (j in 2:nbins){
binedg[j]=odobs[(j-1)*count]
binedg[nbins+1]=max(odobs)
```

```
}
#bin length
binw=diff(binedg)
#number in each bin for the observations
nob=matrix(0,nbins,5)
idf=matrix(0,nf,1)
idl=matrix(0,nl,1)
idil=matrix(0,ni,1)
idir=matrix(0,ni,1)
idr=matrix(0,nr,1)
{if (nf==0){
nob[,1]=matrix(0,nbins,1)}
else{
for (i in 1:nf){
for (j in 1:nbins){
if (binedg[j]<fobs[i] & fobs[i]<=binedg[j+1]){
nob[j,1]=nob[j,1]+1
idf[i]=j}
}}
}
if (nl==0){
nob[,2]=matrix(0,nbins,1)}
else{
for (i in 1:nl){
for (j in 1:nbins){
if (binedg[j]<lc[i] & lc[i]<=binedg[j+1]){
nob[j,2]=nob[j,2]+1
idl[i]=j}
}}
}
if (ni==0){
nob[,3]=nob[,4]=matrix(0,nbins,1)}
else{
```

```
for (i in 1:ni){
for (j in 1:nbins){
if (binedg[j]<ilc[i] & ilc[i]<=binedg[j+1]){
nob[j,3]=nob[j,3]+1
idil[i]=j}
if (binedg[j]<irc[i] & irc[i]<=binedg[j+1])
{nob[j,4]=nob[j,4]+1
idir[i]=j}
}}
}
if (nr==0){
nob[,5]=matrix(0,nbins,1)}
else{
for (i in 1:nr){
for (j in 1:nbins){
if (binedg[j]<rc[i] & rc[i]<=binedg[j+1]){
nob[j,5]=nob[j,5]+1
idr[i]=j}
}}
}
}
classifyf=cbind(fobs,idf)
classifyl=cbind(lc,idl)
classifyil=cbind(ilc,idil)
classifyir=cbind(irc,idir)
classifyr=cbind(rc,idr)

return(list(nbins=nbins,nf=nf,nl=nl,ni=ni,nr=nr,
binw=binw,binedg=binedg,classifyf=classifyf,

classifyl=classifyl,classifyil=classifyil,
classifyir=classifyir,classifyr=classifyr,nob=nob))
}
```

```
info=classification(data,count,n)
```

B.3 The MPL estimation

```
#R matrix for smoothing
pen=function(nbins){
M=matrix(0,nbins,nbins)
for (u in 1:nbins){
M[u,u]=6
}
for (u in 1:(nbins-1)){
M[u+1,u]=-4
M[u,u+1]=-4
}
for (u in 1:(nbins-2)){
M[u,u+2]=1
M[u+2,u]=1
}
M[1,1]=M[nbins,nbins]=1
M[2,2]=M[nbins-1,nbins-1]=5
M[2,1]=M[1,2]=M[nbins,nbins-1]=M[nbins-1,nbins]=-2
return(M)
}

#MPL estimation
sigma=0.5
rho=5000
maxiter=300
#smooth: smoothing value
mplAHM=function(data,X,count,n,sigma,rho,maxiter,smooth){
info=classification(data,count,n)
nbins=info$nbins
```

```
#Penalty R matrix
R=pen(nbins)
#number of obser in each data type
nf=info$nf
nl=info$nl
ni=info$ni
nr=info$nr
nob=info$nob
#exactly observed failure time
classifyf=info$classifyf
yf=classifyf[,1]
idf=classifyf[,2]
#left censored
classifyl=info$classifyl
yl=classifyl[,1]
idl=classifyl[,2]
#left endpoint for interval-censored
classifyil=info$classifyil
yil=classifyil[,1]
idil=classifyil[,2]
#right endpoint for interval-censored
classifyir=info$classifyir
yir=classifyir[,1]
idir=classifyir[,2]
#right-censored data
classifyr=info$classifyr
yr=classifyr[,1]
idr=classifyr[,2]
#bin width and points
binw=info$binw
binedg=info$binedg
p=dim(X)[2]
id=data[,3]
```

```
#covariate matrix
Xf=matrix(X[id==0],nf,p)
Xl=matrix(X[id==1],nl,p)
Xi=matrix(X[id==2],ni,p)
Xr=matrix(X[id==3],nr,p)
#initial values
beta0=matrix(0,p,1)
lamda0=matrix(1,nbins+ni+n,1)
s0=matrix(1,nbins+ni+n,1)
bh0=matrix(1,nbins,1)
oldlamda=lamda0
olds=s0
oldbeta=beta0
oldbh=bh0
oldeta=rbind(oldbh,oldbeta)
mu=sum(oldlamda*olds)/(nbins+n+ni)
#cumulative baseline
oldcumbh=cumsum(binw*oldbh)
oldcumbh=c(0,oldcumbh)
#cumulative baseline hazard
{if (nf==0){
oldcbhf=0}
else{
oldcbhf=oldcumbh[idf]+oldbh[idf]*(yf-binedg[idf])
}
if (nl==0){
oldcbhl=0}
else{
oldcbhl=oldcumbh[idl]+oldbh[idl]*(yl-binedg[idl])
}
if (ni==0){
oldcbhil=oldcbhir=0}
else{
```

```
oldcbhil=oldcumbh[idil]+oldbh[idil]*(yil-binedg[idil])
oldcbhir=oldcumbh[idir]+oldbh[idir]*(yir-binedg[idir])
}
if (nr==0){
oldcbhr=0}
else{
oldcbhr=oldcumbh[idr]+oldbh[idr]*(yr-binedg[idr])
}
}
#cumulative hazard
oldchf=oldcbhf+(Xf%*%oldbeta)*yf
oldchl=oldcbhl+(Xl%*%oldbeta)*yl
oldchil=oldcbhil+(Xi%*%oldbeta)*yil
oldchir=oldcbhir+(Xi%*%oldbeta)*yir
oldchr=oldcbhr+(Xr%*%oldbeta)*yr
#hazard function for exactly failure data
oldhzf=oldbh[idf]+Xf%*%oldbeta
#survival function
oldsvl=exp(-oldchl)
oldsvil=exp(-oldchil)
oldsvir=exp(-oldchir)
# f function
oldf=rbind(-oldbh,-oldhzf,
-(oldbh[idl]+Xl%*%oldbeta),-(oldbh[idil]+Xi%*%oldbeta),
-(oldbh[idir]+Xi%*%oldbeta),-(oldbh[idr]+Xr%*%oldbeta)
)
#M matrix
M0=cbind(-diag(1,nbins),matrix(0,nbins,p))
{if (nf==0){
Mf=cbind(matrix(0,nf,nbins),-Xf)}
else{
Mf=cbind(matrix(0,nf,nbins),-Xf)
for (j in 1:nf){
```

```
Mf[j,][idf[j]]=-1
}
}
if (nl==0){
Ml=cbind(matrix(0,nl,nbins),-Xl)}
else{
Ml=cbind(matrix(0,nl,nbins),-Xl)
for (j in 1:nl){
Ml[j,][idl[j]]=-1
}
}
if (ni==0){
Mil=Mir=cbind(matrix(0,ni,nbins),-Xi)}
else{
Mil=Mir=cbind(matrix(0,ni,nbins),-Xi)
for (j in 1:ni){
Mil[j,][idil[j]]=-1
Mir[j,][idir[j]]=-1
}
}
if (nr==0){
Mr=cbind(matrix(0,nr,nbins),-Xr)}
else{
Mr=cbind(matrix(0,nr,nbins),-Xr)
for (j in 1:nr){
Mr[j,][idr[j]]=-1
}
}
}
M=rbind(M0,Mf,Ml,Mil,Mir,Mr)
#gradient for beta
{if (nl==1){
A=(yl*oldsvl)/(1-oldsvl)}
```

```
else{
A=diag(c((yl*oldsvl)/(1-oldsvl)))
}
if (ni==1){
B=(yir*oldsvir-yil*oldsvil)/(oldsvil-oldsvir)}
else{
B=diag(c((yir*oldsvir-yil*oldsvil)/(oldsvil-oldsvir)))
}
if (nr==1){
C=yr}
else{
C=diag(c(yr))
}
if (nf==1){
D=1/oldhzf-yf
}
else{
D=diag(c(1/oldhzf-yf))
}
}
oldbgrad=(t(Xl)%*%A%*%matrix(1,nl,1)
+t(Xi)%*%B%*%matrix(1,ni,1)-t(Xr)%*%C%*%matrix(1,nr,1)
+t(Xf)%*%D%*%matrix(1,nf,1))
#gradient for theta
tgradl=matrix(0,nbins,1)
tgradi=matrix(0,nbins,1)
tgradr=matrix(0,nbins,1)
tgradf=matrix(0,nbins,1)
suml=sumr=sumf=matrix(0,nbins-1,1)
for (k in (1:nbins-1)){
if (nl==0){
suml[k]=0}
else{
```

```
suml[k]=sum(oldsvl[idl==k+1]/(1-oldsvl[idl==k+1]))
}
sumr[k]=nob[k+1,5]
sumf[k]=nob[k+1,1]
}
cuml=c(rev(cumsum(rev(suml))),0)
cumr=c(rev(cumsum(rev(sumr))),0)
cumf=c(rev(cumsum(rev(sumf))),0)
for (u in 1:nbins){
if (nl==0){
tgradl[u]=0}
else{
tgradl[u]=sum(((oldsvl[idl==u])*(yl[idl==u]-binedg[u]))
/(1-oldsvl[idl==u]))+binw[u]*cuml[u]
}
if (ni==0){
tgradi[u]=0}
else{
for (i in 1:ni){
if (idil[i]<u & idir[i]==u)
tgradi[u]=tgradi[u]+((yir[i]-binedg[u])*oldsvir[i])
/(oldsvil[i]-oldsvir[i])
else if (idil[i]<u & idir[i]>u)
tgradi[u]=tgradi[u]+(binw[u]*oldsvir[i])
/(oldsvil[i]-oldsvir[i])
else if (idil[i]==u & idir[i]==u)
tgradi[u]=tgradi[u]+((yir[i]-binedg[u])*oldsvir[i]-
(yil[i]-binedg[u])*oldsvil[i])/(oldsvil[i]-oldsvir[i])
else if (idil[i]==u & idir[i]>u)
tgradi[u]=tgradi[u]+(binw[u]*oldsvir[i]-
(yil[i]-binedg[u])*oldsvil[i])/(oldsvil[i]-oldsvir[i])
else if (idil[i]>u & idir[i]>u)
tgradi[u]=tgradi[u]-binw[u]
```

```
else
tgradi[u]=tgradi[u]+0
}
}
if (nr==0){
tgradr[u]=0}
else{
tgradr[u]=sum(yr[idr==u]-binedg[u])+cumr[u]*binw[u]
}
if (nf==0){
tgradf[u]=0}
else{
tgradf[u]=sum(1/oldhzf[idf==u]
-(yf[idf==u]-binedg[u]))-cumf[u]*binw[u]
}
}
oldtgrad=tgradl+tgradi-tgradr+tgradf
#first derivative for objective function
olddPhi=-(rbind(oldtgrad,oldbgrad)
-(smooth/(1-smooth))*rbind(2*R%*%oldbh,matrix(0,p)))
cvg=matrix(0,maxiter,2)
#begin algorithm
for (iter in 1:maxiter){
#Hessian matrix for beta
if (nl==1){
E=-((yl^2)*oldsvl)/(1-oldsvl)^2}
else{
E=diag(c(-((yl^2)*oldsvl)/(1-oldsvl)^2))
}
if (ni==1){
G=-(((yir-yil)^2)*oldsvil*oldsvir)/(oldsvil-oldsvir)^2}
else{
G=diag(c(-(((yir-yil)^2)*oldsvil*oldsvir)
```

```
/(oldsvil-oldsvir)^2))
}
if (nf==1){
L=-1/oldhzf^2}
else{
L=diag(c(-1/oldhzf^2))
}
bHes=t(Xl)%*%E%*%Xl+t(Xi)%*%G%*%Xi+t(Xf)%*%L%*%Xf
#second derivative with respective to beta and theta
{if (nl==0){
cumml=matrix(0,p,nbins-1)}
else{
summl=matrix(0,p,nbins-1)
cumml=matrix(0,p,nbins-1)
for (k in 1:(nbins-1)){
summl[,k]=matrix(0,p,1)
summl[,k]=t(matrix(Xl[idl==k+1],nob[k+1,2],p))
%*%((-yl[idl==k+1]*(oldsvl[idl==k+1]))
/(1-oldsvl[idl==k+1])^2)
}
for (k in 1:(nbins-1)){
cumml[,k]=summl[,k:(nbins-1)]%*%matrix(1,(nbins-k),1)
}
}
}
cumml=cbind(cumml,matrix(0,p,1))
dbtl=matrix(0,p,nbins)
dbti=matrix(0,p,nbins)
dbtf=matrix(0,p,nbins)
for (u in 1:nbins){
if (nl==0){
dbtl[,u]=matrix(0,p,1)}
else{
```

```
dbtl[,u]=t(matrix(Xl[idl==u],nob[u,2],p))%*%
((-yl[idl==u]*(oldsvl[idl==u])*(yl[idl==u]-binedg[u]))
/(1-oldsvl[idl==u])^2)+binw[u]*cumml[,u]
}
if (ni==0){
dbti[,u]=matrix(0,p,1)}
else{
for (i in 1:ni){
if (idil[i]<u & idir[i]==u)
dbti[,u]=dbti[,u]+t(Xi[i,])*(((yir[i]-binedg[u])
*(yil[i]-yir[i])*oldsvil[i]*oldsvir[i])
/(oldsvil[i]-oldsvir[i])^2)
else if (idil[i]<u & idir[i]>u)
dbti[,u]=dbti[,u]+t(Xi[i,])*((-binw[u]*oldsvil[i]
*oldsvir[i]*(yir[i]-yil[i]))
/(oldsvil[i]-oldsvir[i])^2)
else if (idil[i]==u & idir[i]==u)
dbti[,u]=dbti[,u]+t(Xi[i,])*(((-(yil[i]-yir[i])^2)
*oldsvil[i]*oldsvir[i])/(oldsvil[i]-oldsvir[i])^2)
else if (idil[i]==u & idir[i]>u)
dbti[,u]=dbti[,u]+t(Xi[i,])*(((binw[u]-yil[i]+binedg[u])
*(yil[i]-yir[i])*oldsvil[i]*oldsvir[i])
/(oldsvil[i]-oldsvir[i])^2)
else
dbti[,u]=dbti[,u]+matrix(0,p,1)
}
}
if (nf==0){
dbtf[,u]=matrix(0,p,1)}
else{
dbtf[,u]= t(matrix(Xf[idf==u],nob[u,1],p))
%*%(1/(oldhzf[idf==u])^2)
}
```

```
}
dbt=dbtl+dbti-dbtf
#Hessian matrix for theta
{if (nl==0){
tHesl=matrix(0,nbins,nbins)}
else{
Al=matrix(0,nbins,1)
suml=matrix(0,nbins-1,1)
for (u in 1:(nbins-1)){
suml[u]=-sum(oldsvl[idl==u+1]/(1-oldsvl[idl==u+1])^2)
}
cuml=c(rev(cumsum(rev(suml))),0)
for (u in 1:nbins){
Al[u]=-sum(((oldsvl[idl==u])*(yl[idl==u]-binedg[u])^2)
/(1-oldsvl[idl==u])^2)+((binw[u])^2)*cuml[u]
}
tHesl=diag(c(Al))
Bl=matrix(0,nbins-1,1)
for (u in 1:(nbins-1)){
Bl[u]=-sum((oldsvl[idl==u+1]*(yl[idl==u+1]-binedg[u+1]))
/(1-oldsvl[idl==u+1])^2)
}
for (i in 2:nbins){
for (j in 1:(i-1)){
tHesl[i,j]=Bl[i-1]*binw[j]+binw[i]*binw[j]*cuml[i]
}}
for (i in 1:(nbins-1)){
for (j in (i+1):nbins){
tHesl[i,j]=tHesl[j,i]
}}
}
}
```

```
{if (ni==0){
tHesi=matrix(0,nbins,nbins)}
else{
Ai=matrix(0,nbins,1)
for (u in 1:nbins){
for (i in 1:ni){
if (idil[i]<u & idir[i]==u)
Ai[u]=Ai[u]+(-oldsvil[i]*oldsvir[i]*(yir[i]-binedg[u])^2)
/(oldsvil[i]-oldsvir[i])^2
else if (idil[i]<u & idir[i]>u)
Ai[u]=Ai[u]+(-oldsvil[i]*oldsvir[i]*(binw[u])^2)
/(oldsvil[i]-oldsvir[i])^2
else if (idil[i]==u & idir[i]==u)
Ai[u]=Ai[u]+(-oldsvil[i]*oldsvir[i]*(yil[i]-yir[i])^2)
/(oldsvil[i]-oldsvir[i])^2
else if (idil[i]==u & idir[i]>u)
Ai[u]=Ai[u]+(-oldsvil[i]*oldsvir[i]*(binw[u]-yil[i]
+binedg[u])^2)/(oldsvil[i]-oldsvir[i])^2
else
Ai[u]=Ai[u]+0
}}
tHesi=diag(c(Ai))
for (i in 2:nbins){
for (j in 1:(i-1)){
for (k in 1:ni){
if (idil[k]==j & idir[k]==i)
tHesi[i,j]=tHesi[i,j]+((yir[k]-binedg[i])*oldsvil[k]
*oldsvir[k]*(yil[k]-binedg[j]-binw[j]))
/(oldsvil[k]-oldsvir[k])^2
else if (idil[k]<j & idir[k]==i)
tHesi[i,j]=tHesi[i,j]+(-binw[j]*(yir[k]-binedg[i])
*oldsvil[k]*oldsvir[k])/(oldsvil[k]-oldsvir[k])^2
else if (idil[k]==j & idir[k]>i)
```

```
tHesi[i,j]=tHesi[i,j]+(-binw[i]*oldsvil[k]*oldsvir[k]
*(binw[j]+binedg[j]-yil[k]))/(oldsvil[k]-oldsvir[k])^2
else if (idil[k]<j & idir[k]>i)
tHesi[i,j]=tHesi[i,j]+(-oldsvil[k]*oldsvir[k]*binw[i]
*binw[j])/(oldsvil[k]-oldsvir[k])^2
else
tHesi[i,j]=tHesi[i,j]+0
}}
}
for (i in 1:(nbins-1)){
for (j in (i+1):nbins){
tHesi[i,j]=tHesi[j,i]
}}
}
}

{if (nf==0){
tHesf=matrix(0,nbins,nbins)}
else{
Af=matrix(0,nbins,1)
for (u in 1:nbins){
Af[u]=sum(-1/(oldhzf[idf==u])^2)
}
tHesf=diag(c(Af))
}
}
tHes=tHesl+tHesi+tHesf
#Hessian matrix
H1=cbind(tHes,t(dbt))
H2=cbind(dbt,bHes)
H=rbind(H1,H2)
#second derivative for objective function
sdPhi=-(H-(smooth/(1-smooth))
```

```
*rbind(cbind(2*R,matrix(0,nbins,p)),matrix(0,p,nbins+p)))
#Jacobian matrix
W1=cbind(sdPhi,t(M),matrix(0,nbins+p,nbins+n+ni))
W2=cbind(M,matrix(0,nbins+n+ni,nbins+n+ni)
,diag(nbins+n+ni))
W3=cbind(matrix(0,nbins+n+ni,nbins+p)
,diag(c(olds)),diag(c(oldlamda)))
W=rbind(W1,W2,W3)
# Newton disrection
V=rbind(olddPhi+t(M)%*%oldlamda,oldf+olds,diag(c(olds))
%*%diag(c(oldlamda))%*%matrix(1,nbins+n+ni,1)
-sigma*mu*matrix(1,nbins+n+ni,1))
direc=solve(W,-V)
deta=matrix(direc[1:(nbins+p)])
dlamda=matrix(direc[(nbins+p+1):((nbins+n+ni)+(nbins+p))])
ds=matrix(direc[((nbins+n+ni)+(nbins+p)+1)
:(2*(nbins+n+ni)+(nbins+p))])
#mu bound
mc=(1-0.01)*mu
#new estimates and new mu
ome=1
repeat{
neweta=oldeta+ome*deta
newbh=matrix(neweta[1:nbins])
newbeta=matrix(neweta[(nbins+1):(nbins+p)])
newlamda=oldlamda+ome*dlamda
news=olds+ome*ds
newmu=sum(newlamda*news)/(nbins+n+ni)
#new survival function
cumbh=cumsum(binw*newbh)
cumbh=c(0,cumbh)
{if (nf==0){
cbhf=0}
```

```
else{
cbhf=cumbh[idf]+newbh[idf]*(yf-binedg[idf])
}
if (nl==0){
cbhl=0}
else{
cbhl=cumbh[idl]+newbh[idl]*(yl-binedg[idl])
}
if (ni==0){
cbhil=cbhir=0}
else{
cbhil=cumbh[idil]+newbh[idil]*(yil-binedg[idil])
cbhir=cumbh[idir]+newbh[idir]*(yir-binedg[idir])
}
if (nr==0){
cbhr=0}
else{
cbhr=cumbh[idr]+newbh[idr]*(yr-binedg[idr])
}
}
chf=cbhf+(Xf%*%newbeta)*yf
chl=cbhl+(Xl%*%newbeta)*yl
chil=cbhil+(Xi%*%newbeta)*yil
chir=cbhir+(Xi%*%newbeta)*yir
chr=cbhr+(Xr%*%newbeta)*yr
hzf=newbh[idf]+Xf%*%newbeta
svl=exp(-chl)
svil=exp(-chil)
svir=exp(-chir)
#new f
f=rbind(-newbh,-hzf,
-(newbh[idl]+Xl%*%newbeta),-(newbh[idil]+Xi%*%newbeta),
-(newbh[idir]+Xi%*%newbeta),-(newbh[idr]+Xr%*%newbeta)
```

```
)


#new gradient for beta
if (nl==1){
A=(yl*svl)/(1-svl)}
else{
A=diag(c((yl*svl)/(1-svl)))
}
if (ni==1){
B=(yir*svir-yil*svil)/(svil-svir)}
else{
B=diag(c((yir*svir-yil*svil)/(svil-svir)))
}
if (nr==1){
C=yr}
else{
C=diag(c(yr))
}
if (nf==1){
D=1/hzf-yf}
else{
D=diag(c(1/hzf-yf))
}
bgrad=(t(Xl)%*%A%*%matrix(1,nl,1)+t(Xi)%*%B%*%matrix(1,ni,1)
-t(Xr)%*%C%*%matrix(1,nr,1)+t(Xf)%*%D%*%matrix(1,nf,1))
#new gradient for theta
tgradl=matrix(0,nbins,1)
tgradi=matrix(0,nbins,1)
tgradr=matrix(0,nbins,1)
tgradf=matrix(0,nbins,1)
suml=sumr=sumf=matrix(0,nbins-1,1)
for (k in (1:nbins-1)){
if (nl==0){
```

```
suml[k]=0}
else{
suml[k]=sum(svl[idl==k+1]/(1-svl[idl==k+1]))
}
sumr[k]=nob[k+1,5]
sumf[k]=nob[k+1,1]
}
cuml=c(rev(cumsum(rev(suml))),0)
cumr=c(rev(cumsum(rev(sumr))),0)
cumf=c(rev(cumsum(rev(sumf))),0)
for (u in 1:nbins){
if (nl==0){
tgradl[u]=0}
else{
tgradl[u]=sum(((svl[idl==u])*(yl[idl==u]-binedg[u]))
/(1-svl[idl==u]))+binw[u]*cuml[u]
}
if (ni==0){
tgradi[u]=0}
else{
for (i in 1:ni){
if (idil[i]<u & idir[i]==u)
tgradi[u]=tgradi[u]+((yir[i]-binedg[u])*svir[i])
/(svil[i]-svir[i])
else if (idil[i]<u & idir[i]>u)
tgradi[u]=tgradi[u]+(binw[u]*svir[i])
/(svil[i]-svir[i])
else if (idil[i]==u & idir[i]==u)
tgradi[u]=tgradi[u]+((yir[i]-binedg[u])*svir[i]
-(yil[i]-binedg[u])
*svil[i])/(svil[i]-svir[i])
else if (idil[i]==u & idir[i]>u)
tgradi[u]=tgradi[u]+(binw[u]*svir[i]-(yil[i]-binedg[u])
```

```
*svil[i])/(svil[i]-svir[i])
else if (idil[i]>u & idir[i]>u)
tgradi[u]=tgradi[u]-binw[u]
else
tgradi[u]=tgradi[u]+0
}
}
if (nr==0){
tgradr[u]=0}
else{
tgradr[u]=sum(yr[idr==u]-binedg[u])+binw[u]*cumr[u]
}
if (nf==0){
tgradf[u]=0}
else{
tgradf[u]=sum(1/hzf[idf==u]-(yf[idf==u]-binedg[u]))
-binw[u]*cumf[u]
}
}
tgrad=tgradl+tgradi-tgradr+tgradf
# new first derivative of objective function
dPhi=-(rbind(tgrad,bgrad)-(smooth/(1-smooth))
*rbind(2*R%*%newbh,matrix(0,p)))
#neighborhood
indicator=0
if (sqrt(t(dPhi+t(M)%*%newlamda)%*%
(dPhi+t(M)%*%newlamda))<=rho*mu
& sqrt(t(f+news)%*%(f+news))<=rho*mu
& min(newlamda*news)>=mu/2
& min(newlamda)>=0 & min(news)>=0)
indicator=1
if ((newmu<=mc) & (indicator==1) )
break
```

```
#select the step length
ome=ome*0.6
# mu bound
mc=(1-0.01*ome)*mu
}
pllf=-((sum(log(1-svl))+sum(log(svil-svir))-sum(chr)
+sum(log(hzf)-chf))
-(smooth/(1-smooth))*t(newbh)%*%R%*%newbh)
llik=sum(log(1-svl))+sum(log(svil-svir))-sum(chr)
+sum(log(hzf)-chf)
cvg[iter,1]=iter
cvg[iter,2]=newmu
if (cvg[iter,2]<1e-5)
{
cvg = cvg[1:iter,]
break
}
else{
oldeta=neweta
olds=news
oldlamda=newlamda
mu=newmu
oldsvl=svl
oldsvil=svil
oldsvir=svir
oldhzf=hzf
oldf=f
olddPhi=dPhi}
}
bh=newbh
cumbh=cumsum(binw*bh)
cumbh=c(0,cumbh)
{if (nf==0){
```

```
cbhf=0}
else{
cbhf=cumbh[idf]+bh[idf]*(yf-binedg[idf])
}
if (nl==0){
cbhl=0}
else{
cbhl=cumbh[idl]+bh[idl]*(yl-binedg[idl])
}
if (ni==0){
cbhil=cbhir=0}
else{
cbhil=cumbh[idil]+bh[idil]*(yil-binedg[idil])
cbhir=cumbh[idir]+bh[idir]*(yir-binedg[idir])
}
if (nr==0){
cbhr=0}
else{
cbhr=cumbh[idr]+bh[idr]*(yr-binedg[idr])
}
}
chf=cbhf+(Xf%*%newbeta)*yf
chl=cbhl+(Xl%*%newbeta)*yl
chil=cbhil+(Xi%*%newbeta)*yil
chir=cbhir+(Xi%*%newbeta)*yir
chr=cbhr+(Xr%*%newbeta)*yr
hzf=bh[idf]+Xf%*%newbeta
svl=exp(-chl)
svil=exp(-chil)
svir=exp(-chir)
#calculate information matrix
#Hessian matrix for beta
if (nl==1){
```

```
E=-((yl^2)*svl)/(1-svl)^2}
else{
E=diag(c(-((yl^2)*svl)/(1-svl)^2))
}
if (ni==1){
G=-(((yir-yil)^2)*svil*svir)/(svil-svir)^2}
else{
G=diag(c(-(((yir-yil)^2)*svil*svir)/(svil-svir)^2))
}
if (nf==1){
L=-1/hzf^2}
else{
L=diag(c(-1/hzf^2))
}
bHes=t(Xl)%*%E%*%Xl+t(Xi)%*%G%*%Xi+t(Xf)%*%L%*%Xf
#second derivative with respective to beta and theta
{if (nl==0){
cumml=matrix(0,p,nbins-1)}
else{
summl=matrix(0,p,nbins-1)
cumml=matrix(0,p,nbins-1)
for (k in 1:(nbins-1)){
summl[,k]=t(matrix(Xl[idl==k+1],nob[k+1,2],p))
%*%((-yl[idl==k+1]*(svl[idl==k+1]))
/(1-svl[idl==k+1])^2)
}
for (k in 1:(nbins-1)){
cumml[,k]=summl[,k:(nbins-1)]
%*%matrix(1,(nbins-k),1)
}
}
}
cumml=cbind(cumml,matrix(0,p,1))
```

```
dbtl=matrix(0,p,nbins)
dbti=matrix(0,p,nbins)
dbtf=matrix(0,p,nbins)
for (u in 1:nbins){
if (nl==0){
dbtl[,u]=matrix(0,p,1)}
else{
dbtl[,u]=t(matrix(Xl[idl==u],nob[u,2],p))%*%
((-yl[idl==u]*(svl[idl==u])*(yl[idl==u]-binedg[u]))
/(1-svl[idl==u])^2)+binw[u]*cumml[,u]
}
if (ni==0){
dbti[,u]=matrix(0,p,1)}
else{
for (i in 1:ni){
if (idil[i]<u & idir[i]==u)
dbti[,u]=dbti[,u]+t(Xi[i,])*(((yir[i]-binedg[u])
*(yil[i]-yir[i])*svil[i]*svir[i])/(svil[i]-svir[i])^2)
else if (idil[i]<u & idir[i]>u)
dbti[,u]=dbti[,u]+t(Xi[i,])*((-binw[u]*svil[i]
*svir[i]*(yir[i]-yil[i]))/(svil[i]-svir[i])^2)
else if (idil[i]==u & idir[i]==u)
dbti[,u]=dbti[,u]+t(Xi[i,])*(((-(yil[i]-yir[i])^2)
*svil[i]*svir[i])/(svil[i]-svir[i])^2)
else if (idil[i]==u & idir[i]>u)
dbti[,u]=dbti[,u]+t(Xi[i,])*(((binw[u]-yil[i]
+binedg[u])*(yil[i]-yir[i])*svil[i]*svir[i])
/(svil[i]-svir[i])^2)
else
dbti[,u]=dbti[,u]+matrix(0,p,1)
}
}
if (nf==0){
```

```
dbtf[,u]=matrix(0,p,1)}
else{
dbtf[,u]= t(matrix(Xf[idf==u],nob[u,1],p))
%*%(1/(hzf[idf==u])^2)
}
}
dbt=dbtl+dbti-dbtf
#Hessian matrix for theta
{if (nl==0){
tHesl=matrix(0,nbins,nbins)}
else{
Al=matrix(0,nbins,1)
suml=matrix(0,nbins-1,1)
for (u in 1:(nbins-1)){
suml[u]=-sum(svl[idl==u+1]/(1-svl[idl==u+1])^2)
}
cuml=c(rev(cumsum(rev(suml))),0)
for (u in 1:nbins){
Al[u]=-sum(((svl[idl==u])*(yl[idl==u]
-binedg[u])^2)/(1-svl[idl==u])^2)
+((binw[u])^2)*cuml[u]
}
tHesl=diag(c(Al))
Bl=matrix(0,nbins-1,1)
for (u in 1:(nbins-1)){
Bl[u]=-sum((svl[idl==u+1]*(yl[idl==u+1]-binedg[u+1]))
/(1-svl[idl==u+1])^2)
}
for (i in 2:nbins){
for (j in 1:(i-1)){
tHesl[i,j]=Bl[i-1]*binw[j]+binw[i]*binw[j]*cuml[i]
}}
for (i in 1:(nbins-1)){
```

```
for (j in (i+1):nbins){
tHesl[i,j]=tHesl[j,i]
}}
}
}

{if (ni==0){
tHesi=matrix(0,nbins,nbins)}
else{
Ai=matrix(0,nbins,1)
for (u in 1:nbins){
for (i in 1:ni){
if (idil[i]<u & idir[i]==u)
Ai[u]=Ai[u]+(-svil[i]*svir[i]*(yir[i]-binedg[u])^2)
/(svil[i]-svir[i])^2
else if (idil[i]<u & idir[i]>u)
Ai[u]=Ai[u]+(-svil[i]*svir[i]*(binw[u])^2)
/(svil[i]-svir[i])^2
else if (idil[i]==u & idir[i]==u)
Ai[u]=Ai[u]+(-svil[i]*svir[i]*(yil[i]-yir[i])^2)
/(svil[i]-svir[i])^2
else if (idil[i]==u & idir[i]>u)
Ai[u]=Ai[u]+(-svil[i]*svir[i]*(binw[u]-yil[i]
+binedg[u])^2)/(svil[i]-svir[i])^2
else
Ai[u]=Ai[u]+0
}}
tHesi=diag(c(Ai))
for (i in 2:nbins){
for (j in 1:(i-1)){
for (k in 1:ni){
if (idil[k]==j & idir[k]==i)
tHesi[i,j]=tHesi[i,j]+((yir[k]-binedg[i])*svil[k]*svir[k]
```

```
*(yil[k]-binedg[j]-binw[j]))/(svil[k]-svir[k])^2
else if (idil[k]<j & idir[k]==i)
tHesi[i,j]=tHesi[i,j]+(-binw[j]*(yir[k]-binedg[i])
*svil[k]*svir[k])/(svil[k]-svir[k])^2
else if (idil[k]==j & idir[k]>i)
tHesi[i,j]=tHesi[i,j]+(-binw[i]*svil[k]*svir[k]
*(binw[j]+binedg[j]-yil[k]))/(svil[k]-svir[k])^2
else if (idil[k]<j & idir[k]>i)
tHesi[i,j]=tHesi[i,j]+(-svil[k]*svir[k]*binw[i]
*binw[j])/(svil[k]-svir[k])^2
else
tHesi[i,j]=tHesi[i,j]+0
}}
}
for (i in 1:(nbins-1)){
for (j in (i+1):nbins){
tHesi[i,j]=tHesi[j,i]
}}
}
}

{if (nf==0){
tHesf=matrix(0,nbins,nbins)}
else{
Af=matrix(0,nbins,1)
for (u in 1:nbins){
Af[u]=sum(-1/(hzf[idf==u])^2)
}
tHesf=diag(c(Af))
}
}
tHes=tHesl+tHesi+tHesf
#Hessian matrix
```

```
H=rbind(cbind(bHes,dbt),cbind(t(dbt),tHes))
HesbinvD=matrix(diag(solve(-bHes)))
tHesp=-tHes+2*smooth*R/(1-smooth)
tHespinvD=matrix(diag(solve(tHesp)))
Hessp=rbind(cbind(-bHes,-dbt),cbind(-t(dbt),tHesp))
varcov=solve(Hessp)%*%(-H)%*%solve(Hessp)
{if (p==1){
betavar=varcov[1:p,1:p]
}
else{
betavar=matrix(diag(varcov[1:p,1:p]))
}
}
betavar[betavar<0]=HesbinvD[betavar<0]
bhvar=matrix(diag(varcov[(p+1):(p+nbins), (p+1):(p+nbins)]))
bhvar[bhvar<0]=tHespinvD[bhvar<0]
#selection criterion for smoothing value
df=sum(diag(solve(-Hessp)%*%H))
AIC=-llik+df
#plot of the estimated baseline hazard and true baseline hazard
time.point=seq(0.05,1,length=50)
tbh=3*time.point^2
ebh=matrix(0,50,1)
ebhvar=matrix(0,50,1)
nd=500
ISEs=matrix(0,nd,1)
for (i in 1:50){
for (j in 1:nbins){
if (binedg[j]<time.point[i] & time.point[i]<=binedg[j+1]){
ebh[i]=bh[j]
ebhvar[i]=bhvar[j]
}
}
```

```
}
minv=min(time.point)
maxv=max(time.point)
delta=(maxv-minv)/nd
bin.point=matrix(seq(minv,maxv,delta))
mp=matrix(0,nd,1)
for (i in 1:nd){
mp[i]=(bin.point[i]+bin.point[i+1])/2
}
for (i in 1:nd){
for (j in 1:nbins){
if (binedg[j]<mp[i] & mp[i]<=binedg[j+1]){
ISEs[i]=(3*mp[i]^2-bh[j])^2
}
}
}
ISE=delta*sum(ISEs)
par(mfrow=c(2,1))
plot(cvg[,1],cvg[,2],type='l',xlab='iteration',
ylab='duality measure',main='convergence view')
plot(time.point,tbh,type='l',col='red',
ylab='baseline hazard function'
,main='ture hazard v.s estimated hazard')
lines(time.point,ebh,type='l',col='blue')
return(list(newbeta=newbeta,ebh=ebh,cvg=cvg,
betavar=betavar,ebhvar=ebhvar,ISE=ISE,AIC=AIC))
}
ahfit=mplAHM(data,X,count,n,sigma,rho,maxiter,smooth)
```

B.4 Monte Carlo simulation

```
#Select smoothing value.
#smv: vector of smoothing values.
selSm<-function(data,X,count,n,sigma,rho,maxiter,smv){
ns=dim(smv)[1]
id=matrix(seq(1,ns,1))
AICs=matrix(0,ns,1)
newbetas=matrix(0,ns,3)
betavars=matrix(0,ns,3)
ISEs=matrix(0,ns,1)
ebhs=matrix(0,50,ns)
for (i in 1:ns){
ahfit=mplAHM(data,X,count,n,sigma,rho,maxiter,smv[i])
AICs[i]=ahfit$AIC
newbetas[i,]=matrix(ahfit$newbeta,nrow=1)
betavars[i,]=matrix(ahfit$betavar,nrow=1)
ebhs[,i]=matrix(ahfit$ebh)
ISEs[i]=ahfit$ISE
print(i)
}
AICm=min(AICs)
id=id[AICs==AICm]
smop=smv[id]
newbetaop=newbetas[id,]
betavarop=betavars[id,]
ISEop=ISEs[id]
ebhop=ebhs[,id]
return(list(smop=smop,AICm=AICm,AICs=AICs
,newbetaop=newbetaop

,betavarop=betavarop,ISEop=ISEop,newbetas=newbetas
,betavars=betavars,ISEs=ISEs))
}
SmSel=selSm(data,X,count,n,sigma,rho,maxiter,smv)
```

```
#generate datasets for simulations
simudata=function(beta,n,p,N){
obsets=matrix(0,n,3*N)
Xs=matrix(0,n,3*N)
for (i in 1:N){
dataset=event(beta,p,n)
obsets[,(3*i-2):(3*i)]=dataset[,1:3]
Xs[,(3*i-2):(3*i)]=dataset[,4:6]
}
return(list(obsets=obsets,Xs=Xs))
}
dataset=simudata(beta,n,p,N)

#MC simulations.
#N: N MC samples.
MCsimu<-function(dataset,sigma,rho,count,beta,N,maxiter,smop){
#initials
np=dim(beta)[1]
betas=matrix(0,N,np)
betavars=matrix(0,N,np)
time.point=seq(0.05,1,length=50)
ebhs=matrix(0,50,N)
ebhvars=matrix(0,50,N)
ISEs=matrix(0,N,1)
datas=dataset$obsets
Xs=dataset$Xs
#MC simulation
for(i in 1:N){
ahfit=mplAHM(datas[,(3*i-2):(3*i)],Xs[,(3*i-2):(3*i)]
,count,n,sigma,rho,maxiter,smop)
betas[i,]=matrix(ahfit$newbeta,nrow=1)
ebhs[,i]=matrix(ahfit$ebh)
betavars[i,]=matrix(ahfit$betavar,nrow=1)
```

```
ebhvars[,i]=matrix(ahfit$ebhvar)
ISEs[i]=ahfit$ISE
print(i)
}
mISE=mean(ISEs)
#mean, bias, std, astd and mse of beta
meanb=matrix(colMeans(betas),1,np)
biasbeta=t(beta)-meanb
varb=matrix(0,1,np)
for (j in 1:np){
varb[1,j]=var(betas[,j])
}
stdb=sqrt(varb)
astdb=matrix(colMeans(sqrt(betavars)),1,np)
#astdb=matrix(sqrt(colMeans(betavars)),1,2)
mseb=biasbeta^2+varb
#summary for beta
summaryb=rbind(meanb,biasbeta,stdb,astdb,mseb)
#mean, std and astd of theta
meanbh=matrix(colMeans(t(ebhs)))
varbh=matrix(0,50,1)
for (j in 1:50){
varbh[j]=var(ebhs[j,])
}
#MC standard deviation
stdbh=sqrt(varbh)
#Average asymptotic standard deviation
aastdbh=matrix(colMeans(t(sqrt(ebhvars))))
# 95% MC PWCI
mclbh=meanbh-1.96*stdbh
mcubh=meanbh+1.96*stdbh
# average of 95% PWCI
aasylbh=meanbh-1.96*aastdbh
```

```
aasyubh=meanbh+1.96*aastdbh
mclbh[mclbh<0]=0
aasylbh[aasylbh<0]=0
summarybh=cbind(meanbh,mclbh,mcubh,aasylbh,aasyubh)
return(list(summaryb=summaryb,summarybh=summarybh
,mISE=mISE))
}
sim=MCsimu(dataset,sigma,rho,count,beta,N,maxiter,smop)
```

B.5 Method comparisons

```
#generate datasets for comparisons
#generate right censored data
beta=matrix(c(1,-0.3,0.5))
event <- function(beta,n){
t=matrix(0,n,1)
X=matrix(0,n,3)
for (i in 1:n){
repeat{
Ca=0
ur=runif(1)
Ca[ur<=0.5]=1
Ca[ur>0.5]=0
X[i,]=c(Ca,runif(1,min=0,max=3),runif(1,min=0,max=5))
t1=log(1-runif(1))
t2=X[i,]%*%beta
f=function(x) (x^3)+x*t2[1]+t1[1]
t[i]=uniroot.all(f,c(0,10))
if ((t[i]!=0) & (length(t[i])!=0))
break
}
}
```

```
C=2*runif(n)

L=matrix(rep(0,n))
R=matrix(rep(0,n))
status=matrix(rep(0,n))

for (i in 1:n){
if (t[i]<=C[i]){
L[i]=t[i]
R[i]=t[i]
status[i]=0
}
else{
L[i]=C[i]
R[i]=Inf
status[i]=3
}
}
return(cbind(L,R,status,X))
}
#N: N MC samples
simudata=function(beta,n,N){
obsets=matrix(0,n,3*N)
Xs=matrix(0,n,3*N)
for (i in 1:N){
dataset=event(beta,n)
obsets[,(3*i-2):(3*i)]=dataset[,1:3]
Xs[,(3*i-2):(3*i)]=dataset[,4:6]
}
return(list(obsets=obsets,Xs=Xs))
}
dataset=simudata(beta,n,N)
```

```
#MC simulations for the method by Lin and Ying (1994)
datas=dataset$obsets
Xs=dataset$Xs
times=matrix(0,n,2*N)
for (i in 1:N){
for (j in 1:n){
if (datas[j,3*i]==0){
times[j,(2*i-1)]=datas[j,(3*i-2)]
times[j,2*i]=1}
else{
times[j,(2*i-1)]=datas[j,(3*i-2)]
times[j,2*i]=0}
}
}

L.sim=function(times,Xs,n,N){
betas=matrix(0,N,3)
betavars=matrix(0,N,3)
time.point=seq(0.05,1,length=50)
ebhs=matrix(0,50,N)
ISEs=matrix(0,N,1)
for (i in 1:N){
set.seed(1010)
times[,(2*i-1)]=times[,(2*i-1)]+runif(n)*1e-2
surv=Surv(times[,(2*i-1)],times[,(2*i)])
L.phfit=ahaz(surv,Xs[,(3*i-2):(3*i)])
betas[i,]=matrix(coef(L.phfit),nrow=1)
betavars[i,]=matrix(diag(summary(L.phfit)$cov),nrow=1)
cumahaz=predict(L.phfit,type='cumhaz')
cumhaz=matrix(cumahaz$cumhaz)
bh=diff(cumhaz)
time.value=matrix(cumahaz$time)
ntp=dim(time.value)[1]
```

```
ebh=matrix(0,50,1)
for (k in 1:50){
for (j in 1:(ntp-1)){
if (time.value[j]<time.point[k]
& time.point[k]<=time.value[j+1]){
ebh[k]=((time.point[k]-time.value[j])
/(time.value[j+1]-time.value[j]))*bh[j]
+((time.value[j+1]-time.point[k])
/(time.value[j+1]-time.value[j]))*bh[j+1]
}
}
}
ebhs[,i]=ebh
nd=500
minv=min(time.point)
maxv=max(time.point)
delta=(maxv-minv)/nd
bin.point=matrix(seq(minv,maxv,delta))
mp=matrix(0,nd,1)
for (k in 1:nd){
mp[k]=(bin.point[k]+bin.point[k+1])/2
}
ISE=matrix(0,nd,1)
embh=matrix(0,nd,1)
for (k in 1:nd){
for (j in 1:(ntp-1)){
if (time.value[j]<mp[k]
& mp[k]<=time.value[j+1]){
embh[k]=((mp[k]-time.value[j])
/(time.value[j+1]-

    time.value[j]))*bh[j]
+((time.value[j+1]-mp[k])
```

```
/(time.value[j+1]-time.value[j]))*bh[j+1]
ISE[k]=(3*mp[k]^2-embh[k])^2
}
}
}
ISEs[i]=delta*sum(ISE)
print(i)
}
mISE=mean(ISEs)
meanb=matrix(colMeans(betas),1,3)
biasbeta=t(beta)-meanb
varb=matrix(0,1,3)
for (j in 1:3){
varb[1,j]=var(betas[,j])
}
stdb=sqrt(varb)
astdb=matrix(colMeans(sqrt(betavars)),1,3)
mseb=biasbeta^2+varb
summaryb=rbind(meanb,biasbeta,stdb,astdb,mseb)
meanbh=matrix(colMeans(t(ebhs)))
varbh=matrix(0,50,1)
for (j in 1:50){
varbh[j]=var(ebhs[j,])
}
stdbh=sqrt(varbh)
mclbh=meanbh-1.96*stdbh
mcubh=meanbh+1.96*stdbh
mclbh[mclbh<0]=0
summarybh=cbind(meanbh,mclbh,mcubh)
return(list(summaryb=summaryb,summarybh=summarybh,mISE=mISE))
}
Lsim=L.sim(times,Xs,n,N)
```

Appendix C R codes for AFT model

C.1 Data generation

```
# Generate data. n: sample size. p: censoring proportion. categorial
# variable.
Ca = matrix(rep(0, n))
ur = runif(n)
Ca[ur <= 0.5] = 1
Ca[0.5 < ur & ur <= 1] = 0
# Covariate matrix
X = cbind(Ca, runif(n, min = 0, max = 3), runif(n, min = 0, max = 5))
beta = matrix(c(1, -0.3, 0.5))

event <- function(beta, X, p, n) {
    epsilond = rweibull(n, shape = 3, scale = 1)
    t = exp(X %*% beta) * epsilond
    Cl = runif(n)
    Cr = Cl + runif(n)
    indicator = matrix(rep(0, n))
    a = runif(n)
    indicator[a <= p] = 1
    indicator[a > p] = 0

    L = matrix(rep(0, n))
```

```
    R = matrix(rep(0, n))
    status = matrix(rep(0, n))

    for (i in 1:n) {
        if (indicator[i] == 0) {
            L[i] = t[i]
            R[i] = t[i]
            status[i] = 0
        } else {
            if (t[i] <= Cl[i]) {
                L[i] = 0
                R[i] = Cl[i]
                status[i] = 1
            } else if (Cl[i] < t[i] && t[i] < Cr[i]) {
                L[i] = Cl[i]
                R[i] = Cr[i]
                status[i] = 2
            } else if (t[i] >= Cr[i]) {
                L[i] = Cr[i]
                R[i] = Inf
                status[i] = 3
            }
        }
    }
    return(cbind(L, R, status))
}
data = event(beta, X, p, n)
```

C.2 The MPL estimation

```
#MPL estimation.
#smooth: smoothing value.
#nknots: number of knots.
#maxiter: number of iteration in the algorithm.
maxiter=5000
mplAFT=function(data,X,nknots,smooth,maxiter){
#classify each cesoring type
id=data[,3]
yf=matrix(data[,1][id==0])
nf=dim(yf)[1]
yl=matrix(data[,2][id==1])
nl=dim(yl)[1]
yil=matrix(data[,1][id==2])
yir=matrix(data[,2][id==2])
ni=dim(yil)[1]
yr=matrix(data[,1][id==3])
nr=dim(yr)[1]
#sample size
n=dim(X)[1]
#number of beta
p=dim(X)[2]
#classify covariate matrix
Xf=matrix(X[id==0],nf,p)
Xl=matrix(X[id==1],nl,p)
Xi=matrix(X[id==2],ni,p)
Xr=matrix(X[id==3],nr,p)
#initial value
beta0=matrix(0,p,1)
theta0=matrix(1,(nknots),1)
oldbeta=beta0
oldtheta=theta0
e=10^(-3)
#time scaled
```

```
oldyfs=yf*exp(-Xf%*%oldbeta)
oldyls=yl*exp(-Xl%*%oldbeta)
oldyils=yil*exp(-Xi%*%oldbeta)
oldyirs=yir*exp(-Xi%*%oldbeta)
oldyrs=yr*exp(-Xr%*%oldbeta)
#select knots mu and variance
oldmaxv=max(rbind(oldyfs,oldyls,oldyils,oldyirs,oldyrs))
oldminv=min(rbind(oldyfs,oldyls,oldyils,oldyirs,oldyrs))
oldbinw=(oldmaxv-oldminv)/(nknots-1)
oldknots=matrix(seq(oldminv,oldmaxv,oldbinw))
oldsig=(2/3)*oldbinw
oldgaus0=pnorm(0,oldknots,oldsig)*sqrt(2*pi*oldsig^2)
#baseline hazard and cumulative hazard when time t is scaled
{if (nf==0){
oldbhf=1
oldchf=0}
else{
oldbhf=matrix(0,nf,1)
oldchf=matrix(0,nf,1)
for (i in 1:nf){
oldbhf[i]=t(dnorm(oldyfs[i],oldknots,oldsig)
*sqrt(2*pi*oldsig^2))%*%oldtheta
oldchf[i]=t(pnorm(oldyfs[i],oldknots,oldsig)
*sqrt(2*pi*oldsig^2)-oldgaus0)%*%oldtheta
}
}

if (nl==0){
oldbhl=0
oldchl=0}
else{
oldbhl=matrix(0,nl,1)
oldchl=matrix(0,nl,1)
```

```
for (i in 1:nl){
oldbhl[i]=t(dnorm(oldyls[i],oldknots,oldsig)
*sqrt(2*pi*oldsig^2))%*%oldtheta
oldchl[i]=t(pnorm(oldyls[i],oldknots,oldsig)
*sqrt(2*pi*oldsig^2)-oldgaus0)%*%oldtheta
}
}

if (ni==0){
oldbhil=0
oldbhir=0
oldchil=0
oldchir=0}
else{
oldbhil=matrix(0,ni,1)
oldbhir=matrix(0,ni,1)
oldchil=matrix(0,ni,1)
oldchir=matrix(0,ni,1)
for (i in 1:ni){
oldbhil[i]=t(dnorm(oldyils[i],oldknots,oldsig)
*sqrt(2*pi*oldsig^2))%*%oldtheta
oldbhir[i]=t(dnorm(oldyirs[i],oldknots,oldsig)
*sqrt(2*pi*oldsig^2))%*%oldtheta
oldchil[i]=t(pnorm(oldyils[i],oldknots,oldsig)
*sqrt(2*pi*oldsig^2)-oldgaus0)%*%oldtheta
oldchir[i]=t(pnorm(oldyirs[i],oldknots,oldsig)
*sqrt(2*pi*oldsig^2)-oldgaus0)%*%oldtheta
}
}

if (nr==0){
oldbhr=0
oldchr=0}
```

```
else{
oldbhr=matrix(0,nr,1)
oldchr=matrix(0,nr,1)
for (i in 1:nr){
oldbhr[i]=t(dnorm(oldyrs[i],oldknots,oldsig)
*sqrt(2*pi*oldsig^2))%*%oldtheta
oldchr[i]=t(pnorm(oldyrs[i],oldknots,oldsig)
*sqrt(2*pi*oldsig^2)-oldgaus0)%*%oldtheta
}
}
}
#survival function
oldsl=exp(-oldchl)
oldsil=exp(-oldchil)
oldsir=exp(-oldchir)
{if (nl==0){
oldgl=1}
else{
oldgl=1-oldsl
}
if (ni==0){
oldgi=1}
else{
oldgi=oldsil-oldsir
}
}
cvg=matrix(0,maxiter,2)
#begin algorithm
for (iter in 1:maxiter){
#algorithm for updating beta
llik0=sum(log(oldgl))+sum(log(oldgi))-sum(oldchr)
-sum(Xf%*%oldbeta+oldchf)+sum(log(oldbhf))
#eta function for scaled time
```

```
etaf=oldbhf*oldyfs
etal=oldbhl*oldyls
etail=oldbhil*oldyils
etair=oldbhir*oldyirs
etar=oldbhr*oldyrs
ksi=etail*oldsil-etair*oldsir
#A function for scaled time
{if (nf==0){
Af=0}
else{
Af=matrix(0,nf,1)
for (i in 1:nf){
Af[i]=t(dnorm(oldyfs[i],oldknots,oldsig)
*sqrt(2*pi*oldsig^2))
%*%(oldtheta*(oldyfs[i]-oldknots)/oldsig^2)
}
}

if (nl==0){
Al=0}
else{
Al=matrix(0,nl,1)
for (i in 1:nl){
Al[i]=t(dnorm(oldyls[i],oldknots,oldsig)
*sqrt(2*pi*oldsig^2))
%*%(oldtheta*(oldyls[i]-oldknots)/oldsig^2)
}
}

if (ni==0){
Ail=0
Air=0}
else{
```

```
Ail=matrix(0,ni,1)
Air=matrix(0,ni,1)
for (i in 1:ni){
Ail[i]=t(dnorm(oldyils[i],oldknots,oldsig)
*sqrt(2*pi*oldsig^2))
%*%(oldtheta*(oldyils[i]-oldknots)/oldsig^2)
Air[i]=t(dnorm(oldyirs[i],oldknots,oldsig)
*sqrt(2*pi*oldsig^2))
%*%(oldtheta*(oldyirs[i]-oldknots)/oldsig^2)
}
}

if (nr==0){
Ar=0}
else{
Ar=matrix(0,nr,1)
for (i in 1:nr){
Ar[i]=t(dnorm(oldyrs[i],oldknots,oldsig)
*sqrt(2*pi*oldsig^2))
%*%(oldtheta*(oldyrs[i]-oldknots)/oldsig^2)
}
}
}
#gradient vector for beta
E=-etal*oldsl/oldgl
F=ksi/oldgi
G=-1+oldyfs*Af/oldbhf+etaf
{if (nl==1){
EE=t(Xl)%*%E%*%matrix(1,nl,1)}
else{
EE=t(Xl)%*%diag(c(E))%*%matrix(1,nl,1)
}
if (ni==1){
```

```
FF=t(Xi)%*%F%*%matrix(1,ni,1)}
else{
FF=t(Xi)%*%diag(c(F))%*%matrix(1,ni,1)
}
if (nr==1){
GG=t(Xr)%*%etar%*%matrix(1,nr,1)}
else{
GG=t(Xr)%*%diag(c(etar))%*%matrix(1,nr,1)
}
if (nf==1){
JJ=t(Xf)%*%G%*%matrix(1,nf,1)}
else{
JJ=t(Xf)%*%diag(c(G))%*%matrix(1,nf,1)
}
}
bgrad=EE+FF+GG+JJ
#H function for scaled time
Hf=oldbhf*oldyfs-Af*oldyfs^2
Hl=oldbhl*oldyls-Al*oldyls^2
Hil=oldbhil*oldyils-Ail*oldyils^2
Hir=oldbhir*oldyirs-Air*oldyirs^2
Hr=oldbhr*oldyrs-Ar*oldyrs^2
#Hessian matrix for beta
K=(Hl-etal^2)*oldsl/oldgl-(etal*oldsl/oldgl)^2
#h function for scaled interval-censored
h=oldsil*Hil-oldsir*Hir
qi=(etail^2)*oldsil-(etair^2)*oldsir
L=(-h+qi)/oldgi-(ksi/oldgi)^2
#B and C function for scaled fully observed
{if (nf==0){
B=0
C=0}
else{
```

```
B=matrix(0,nf,1)
C=matrix(0,nf,1)
for (i in 1:nf){
B[i]=t(dnorm(oldyfs[i],oldknots,oldsig)
*sqrt(2*pi*oldsig^2))
%*%(oldtheta*((oldyfs[i]-oldknots)/oldsig^2)^2)
C[i]=t(dnorm(oldyfs[i],oldknots,oldsig)
*sqrt(2*pi*oldsig^2))%*%(oldtheta/oldsig^2)
}
}
}
M=(-oldyfs*Af+(B-C)*oldyfs^2)/oldbhf
-(oldyfs*Af/oldbhf)^2-Hf
{if (nl==1){
KK=t(Xl)%*%K%*%Xl}
else{
KK=t(Xl)%*%diag(c(K))%*%Xl
}
if (ni==1){
LL=t(Xi)%*%L%*%Xi}
else{
LL=t(Xi)%*%diag(c(L))%*%Xi
}
if (nr==1){
NN=t(Xr)%*%Hr%*%Xr}
else{
NN=t(Xr)%*%diag(c(Hr))%*%Xr
}
if (nf==1){
MM=t(Xf)%*%M%*%Xf}
else{
MM=t(Xf)%*%diag(c(M))%*%Xf
}
```

```
}
Hesb=KK+LL-NN+MM
#new estimates for beta
incb=solve(Hesb)%*%bgrad
newbeta=oldbeta-incb
#new time scaled
yfs=yf*exp(-Xf%*%newbeta)
yls=yl*exp(-Xl%*%newbeta)
yils=yil*exp(-Xi%*%newbeta)
yirs=yir*exp(-Xi%*%newbeta)
yrs=yr*exp(-Xr%*%newbeta)
#new knots mu and variance
maxv=max(rbind(yfs,yls,yils,yirs,yrs))
minv=min(rbind(yfs,yls,yils,yirs,yrs))
binw=(maxv-minv)/(nknots-1)
knots=matrix(seq(minv,maxv,binw))
sig=(2/3)*binw
gaus0=pnorm(0,knots,sig)*sqrt(2*pi*sig^2)

{if (nf==0){
hnewbhf=1
hnewchf=0}
else{
hnewbhf=matrix(0,nf,1)
hnewchf=matrix(0,nf,1)
for (i in 1:nf){
hnewbhf[i]=t(dnorm(yfs[i],knots,sig)
*sqrt(2*pi*sig^2))%*%oldtheta
hnewchf[i]=t(pnorm(yfs[i],knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%oldtheta
}
}
```

```
if (nl==0){
hnewbhl=0
hnewchl=0}
else{
hnewbhl=matrix(0,nl,1)
hnewchl=matrix(0,nl,1)
for (i in 1:nl){
hnewbhl[i]=t(dnorm(yls[i],knots,sig)
*sqrt(2*pi*sig^2))%*%oldtheta
hnewchl[i]=t(pnorm(yls[i],knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%oldtheta
}
}

if (ni==0){
hnewbhil=0
hnewbhir=0
hnewchil=0
hnewchir=0}
else{
hnewbhil=matrix(0,ni,1)
hnewbhir=matrix(0,ni,1)
hnewchil=matrix(0,ni,1)
hnewchir=matrix(0,ni,1)
for (i in 1:ni){
hnewbhil[i]=t(dnorm(yils[i],knots,sig)
*sqrt(2*pi*sig^2))%*%oldtheta
hnewbhir[i]=t(dnorm(yirs[i],knots,sig)
*sqrt(2*pi*sig^2))%*%oldtheta
hnewchil[i]=t(pnorm(yils[i],knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%oldtheta
hnewchir[i]=t(pnorm(yirs[i],knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%oldtheta
```

```
}
}

if (nr==0){
hnewbhr=0
hnewchr=0}
else{
hnewbhr=matrix(0,nr,1)
hnewchr=matrix(0,nr,1)
for (i in 1:nr){
hnewbhr[i]=t(dnorm(yrs[i],knots,sig)
*sqrt(2*pi*sig^2))%*%oldtheta
hnewchr[i]=t(pnorm(yrs[i],knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%oldtheta
}
}

#half new survival function
hnewsl=exp(-hnewchl)
hnewsil=exp(-hnewchil)
hnewsir=exp(-hnewchir)
{if (nl==0){
hnewgl=1}
else{
hnewgl=1-hnewsl
}
if (ni==0){
hnewgi=1}
else{
hnewgi=hnewsil-hnewsir
}
}
llik1=sum(log(hnewgl))+sum(log(hnewgi))-sum(hnewchr)
-sum(Xf%*%newbeta+hnewchf)+sum(log(hnewbhf))
```

```
ome=0.6
while(llik1<=llik0){
#newbeta
newbeta=oldbeta-ome*incb
#new time scaled
yfs=yf*exp(-Xf%*%newbeta)
yls=yl*exp(-Xl%*%newbeta)
yils=yil*exp(-Xi%*%newbeta)
yirs=yir*exp(-Xi%*%newbeta)
yrs=yr*exp(-Xr%*%newbeta)
#new knots mu and variance
maxv=max(rbind(yfs,yls,yils,yirs,yrs))
minv=min(rbind(yfs,yls,yils,yirs,yrs))
binw=(maxv-minv)/(nknots-1)
knots=matrix(seq(minv,maxv,binw))
sig=(2/3)*binw
gaus0=pnorm(0,knots,sig)*sqrt(2*pi*sig^2)

if (nf==0){
hnewbhf=1
hnewchf=0}
else{
hnewbhf=matrix(0,nf,1)
hnewchf=matrix(0,nf,1)
for (i in 1:nf){
hnewbhf[i]=t(dnorm(yfs[i],knots,sig)
*sqrt(2*pi*sig^2))%*%oldtheta
hnewchf[i]=t(pnorm(yfs[i],knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%oldtheta
}
}

if (nl==0){
```

```
hnewbhl=0
hnewchl=0}
else{
hnewbhl=matrix(0,nl,1)
hnewchl=matrix(0,nl,1)
for (i in 1:nl){
hnewbhl[i]=t(dnorm(yls[i],knots,sig)
*sqrt(2*pi*sig^2))%*%oldtheta
hnewchl[i]=t(pnorm(yls[i],knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%oldtheta
}
}

if (ni==0){
hnewbhil=0
hnewbhir=0
hnewchil=0
hnewchir=0}
else{
hnewbhil=matrix(0,ni,1)
hnewbhir=matrix(0,ni,1)
hnewchil=matrix(0,ni,1)
hnewchir=matrix(0,ni,1)
for (i in 1:ni){
hnewbhil[i]=t(dnorm(yils[i],knots,sig)
*sqrt(2*pi*sig^2))%*%oldtheta
hnewbhir[i]=t(dnorm(yirs[i],knots,sig)
*sqrt(2*pi*sig^2))%*%oldtheta
hnewchil[i]=t(pnorm(yils[i],knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%oldtheta
hnewchir[i]=t(pnorm(yirs[i],knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%oldtheta
}
```

```
}

if (nr==0){
hnewbhr=0
hnewchr=0}
else{
hnewbhr=matrix(0,nr,1)
hnewchr=matrix(0,nr,1)
for (i in 1:nr){
hnewbhr[i]=t(dnorm(yrs[i],knots,sig)
*sqrt(2*pi*sig^2))%*%oldtheta
hnewchr[i]=t(pnorm(yrs[i],knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%oldtheta
}
}
#half new survival function
hnewsl=exp(-hnewchl)
hnewsil=exp(-hnewchil)
hnewsir=exp(-hnewchir)
if (nl==0){
hnewgl=1}
else{
hnewgl=1-hnewsl
}
if (ni==0){
hnewgi=1}
else{
hnewgi=hnewsil-hnewsir
}
llik1=sum(log(hnewgl))+sum(log(hnewgi))
-sum(hnewchr)-sum(Xf%*%newbeta+hnewchf)
+sum(log(hnewbhf))
if (ome>=1e-2)
```

```
ome=ome*0.6
else if (ome < 1e-2 & ome >= 1e-5)
ome = ome*5e-2
else if (ome<1e-5 & ome>1e-20)
ome = ome*1e-5
else
break
}
#R matrix
R=matrix(0,nknots,nknots)
nd=500
delta=(maxv-minv)/nd
binedg=matrix(seq(minv,maxv,delta))
mp=matrix(0,nd,1)
for (i in 1:nd){
mp[i]=(binedg[i]+binedg[i+1])/2
}
for (u in 1:nknots){
for (r in 1:nknots){
R[u,r]=delta*sum((-1/(sig)^2+((mp-knots[u])/(sig)^2)^2)
*(-1/(sig)^2+((mp-knots[r])/(sig)^2)^2)
*dnorm(mp,knots[u],sig)*sqrt(2*pi*(sig)^2)
*dnorm(mp,knots[r],sig)*sqrt(2*pi*(sig)^2))
}
}
#algorithm for updating theta
llik2=sum(log(hnewgl))+sum(log(hnewgi))-sum(hnewchr)
-sum(hnewchf)+sum(log(hnewbhf))
-(smooth/(1-smooth))*t(oldtheta)%*%R%*%oldtheta
dpen=2*(smooth/(1-smooth))*R%*%oldtheta
#new estimates for theta
a=matrix(0,nknots,1)
b1=matrix(0,nknots,1)
```

```
b2=matrix(0,nknots,1)
c=matrix(0,nknots,1)
d1=matrix(0,nknots,1)
d2=matrix(0,nknots,1)
for (j in 1:(nknots)){
a[j]=sum(hnewsl*(pnorm(yls,knots[j],sig)
*sqrt(2*pi*(sig)^2)-gaus0[j])/hnewgl)
b1[j]=sum(hnewsir*(pnorm(yirs,knots[j],sig)
*sqrt(2*pi*(sig)^2)-gaus0[j])/hnewgi)
b2[j]=sum(hnewsil*(pnorm(yils,knots[j],sig)
*sqrt(2*pi*(sig)^2)-gaus0[j])/hnewgi)
c[j]=sum(pnorm(yrs,knots[j],sig)
*sqrt(2*pi*(sig)^2)-gaus0[j])
d1[j]=sum(dnorm(yfs,knots[j],sig)
*sqrt(2*pi*(sig)^2)/hnewbhf)
d2[j]=sum(pnorm(yfs,knots[j],sig)
*sqrt(2*pi*(sig)^2)-gaus0[j])
}
#nume=a+b1+d1-min(0,dpen)
nume=a+b1+d1-matrix(apply(dpen,1
,function(x) min(x,0)))
#deno=b2+c+d2+max(0,dpen)
deno=b2+c+d2+matrix(apply(dpen,1
,function(x) max(x,0)))
newtheta=oldtheta*((nume+e)/(deno+e))
incbh=newtheta-oldtheta

{if (nf==0){
bhf=1
chf=0}
else{
bhf=matrix(0,nf,1)
chf=matrix(0,nf,1)
```

```
for (i in 1:nf){
bhf[i]=t(dnorm(yfs[i],knots,sig)
*sqrt(2*pi*sig^2))%*%newtheta
chf[i]=t(pnorm(yfs[i],knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%newtheta
}
}

if (nl==0){
bhl=0
chl=0}
else{
bhl=matrix(0,nl,1)
chl=matrix(0,nl,1)
for (i in 1:nl){
bhl[i]=t(dnorm(yls[i],knots,sig)
*sqrt(2*pi*sig^2))%*%newtheta
chl[i]=t(pnorm(yls[i],knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%newtheta
}
}

if (ni==0){
bhil=0
bhir=0
chil=0
chir=0}
else{
bhil=matrix(0,ni,1)
bhir=matrix(0,ni,1)
chil=matrix(0,ni,1)
chir=matrix(0,ni,1)
for (i in 1:ni){
```

```
bhil[i]=t(dnorm(yils[i],knots,sig)
*sqrt(2*pi*sig^2))%*%newtheta
bhir[i]=t(dnorm(yirs[i],knots,sig)
*sqrt(2*pi*sig^2))%*%newtheta
chil[i]=t(pnorm(yils[i],knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%newtheta
chir[i]=t(pnorm(yirs[i],knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%newtheta
}
}

if (nr==0){
bhr=0
chr=0}
else{
bhr=matrix(0,nr,1)
chr=matrix(0,nr,1)
for (i in 1:nr){
bhr[i]=t(dnorm(yrs[i],knots,sig)
*sqrt(2*pi*sig^2))%*%newtheta
chr[i]=t(pnorm(yrs[i],knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%newtheta
}
}
}


#half new survival function
sl=exp(-chl)
sil=exp(-chil)
sir=exp(-chir)
{if (nl==0){
gl=1}
else{
```

```
gl=1-sl
}
if (ni==0){
gi=1}
else{
gi=sil-sir
}
}
llik3=sum(log(gl))+sum(log(gi))-sum(chr)
-sum(chf)+sum(log(bhf))
-(smooth/(1-smooth))*t(newtheta)%*%R%*%newtheta
ome=0.6
while (llik3<=llik2){
#new estimates for baseline
newtheta=oldtheta+ome*incbh
if (nf==0){
bhf=1
chf=0}
else{
bhf=matrix(0,nf,1)
chf=matrix(0,nf,1)
for (i in 1:nf){
bhf[i]=t(dnorm(yfs[i],knots,sig)
*sqrt(2*pi*sig^2))%*%newtheta
chf[i]=t(pnorm(yfs[i],knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%newtheta
}
}

if (nl==0){
bhl=0
chl=0}
else{
```

```
bhl=matrix(0,nl,1)
chl=matrix(0,nl,1)
for (i in 1:nl){
bhl[i]=t(dnorm(yls[i],knots,sig)
*sqrt(2*pi*sig^2))%*%newtheta
chl[i]=t(pnorm(yls[i],knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%newtheta
}
}

if (ni==0){
bhil=0
bhir=0
chil=0
chir=0}
else{
bhil=matrix(0,ni,1)
bhir=matrix(0,ni,1)
chil=matrix(0,ni,1)
chir=matrix(0,ni,1)
for (i in 1:ni){
bhil[i]=t(dnorm(yils[i],knots,sig)
*sqrt(2*pi*sig^2))%*%newtheta
bhir[i]=t(dnorm(yirs[i],knots,sig)
*sqrt(2*pi*sig^2))%*%newtheta
chil[i]=t(pnorm(yils[i],knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%newtheta
chir[i]=t(pnorm(yirs[i],knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%newtheta
}
}

if (nr==0){
```

```
bhr=0
chr=0}
else{
bhr=matrix(0,nr,1)
chr=matrix(0,nr,1)
for (i in 1:nr){
bhr[i]=t(dnorm(yrs[i],knots,sig)
*sqrt(2*pi*sig^2))%*%newtheta
chr[i]=t(pnorm(yrs[i],knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%newtheta
}
}
#new survival function
sl=exp(-chl)
sil=exp(-chil)
sir=exp(-chir)
if (nl==0){
gl=1}
else{
gl=1-sl
}
if (ni==0){
gi=1}
else{
gi=sil-sir
}
llik3=sum(log(gl))+sum(log(gi))-sum(chr)
-sum(chf)+sum(log(bhf))
-(smooth/(1-smooth))*t(newtheta)%*%R%*%newtheta
if (ome>=1e-2)
ome=ome*0.6
else if (ome < 1e-2 & ome >= 1e-5)
ome = ome*5e-2
```

```
else if (ome<1e-5 & ome>1e-20)
ome = ome*1e-5
else
break
}
plik=sum(log(gl))+sum(log(gi))-sum(chr)
-sum(Xf%*%newbeta+chf)+sum(log(bhf))
-(smooth/(1-smooth))*t(newtheta)%*%R%*%newtheta
cvg[iter,1]=iter
cvg[iter,2]=plik
if (max(abs(newtheta-oldtheta))<1e-5
& max(abs(newbeta-oldbeta))<1e-5)
{
cvg = cvg[1:iter,]
break
}
else{
oldbeta=newbeta
oldtheta=newtheta
oldyfs=yfs
oldyls=yls
oldyils=yils
oldyirs=yirs
oldyrs=yrs
oldknots=knots
oldsig=sig
oldbhf=bhf
oldbhl=bhl
oldbhil=bhil
oldbhir=bhir
oldbhr=bhr
oldchr=chr
oldchf=chf
```

```
oldsl=sl
oldsil=sil
oldsir=sir
oldgl=gl
oldgi=gi
}

#second derivative with respective to beta
#new A function for scaled time
if (nf==0){
Af=0}
else{
Af=matrix(0,nf,1)
for (i in 1:nf){
Af[i]=t(dnorm(yfs[i],knots,sig)
*sqrt(2*pi*sig^2))%*%(newtheta*(yfs[i]-knots)/sig^2)
}
}
if (nl==0){
Al=0}
else{
Al=matrix(0,nl,1)
for (i in 1:nl){
Al[i]=t(dnorm(yls[i],knots,sig)*sqrt(2*pi*sig^2))
%*%(newtheta*(yls[i]-knots)/sig^2)
}
}
if (ni==0){
Ail=0
Air=0}
else{
Ail=matrix(0,ni,1)
Air=matrix(0,ni,1)
for (i in 1:ni){
```

```
Ail[i]=t(dnorm(yils[i],knots,sig)*sqrt(2*pi*sig^2))
%*%(newtheta*(yils[i]-knots)/sig^2)
Air[i]=t(dnorm(yirs[i],knots,sig)*sqrt(2*pi*sig^2))
%*%(newtheta*(yirs[i]-knots)/sig^2)
}
}
if (nr==0){
Ar=0}
else{
Ar=matrix(0,nr,1)
for (i in 1:nr){
Ar[i]=t(dnorm(yrs[i],knots,sig)*sqrt(2*pi*sig^2))
%*%(newtheta*(yrs[i]-knots)/sig^2)
}
}
#new H function for scaled time
Hf=yfs*bhf-Af*yfs^2
Hl=yls*bhl-Al*yls^2
Hil=yils*bhil-Ail*yils^2
Hir=yirs*bhir-Air*yirs^2
Hr=yrs*bhr-Ar*yrs^2
#new Hessian matrix for beta
etal=bhl*yls
etail=bhil*yils
etair=bhir*yirs
ksi=etail*sil-etair*sir
K=(Hl-etal^2)*sl/gl-(etal*sl/gl)^2
#new h function for scaled interval-censored
h=sil*Hil-sir*Hir
qi=(etail^2)*sil-(etair^2)*sir
L=(-h+qi)/gi-(ksi/gi)^2
#new B and C function for scaled fully observed
if (nf==0){
```

```
B=0
C=0}
else{
B=matrix(0,nf,1)
C=matrix(0,nf,1)
for (i in 1:nf){
B[i]=t(dnorm(yfs[i],knots,sig)*sqrt(2*pi*sig^2))
%*%(newtheta*((yfs[i]-knots)/sig^2)^2)
C[i]=t(dnorm(yfs[i],knots,sig)*sqrt(2*pi*sig^2))
%*%(newtheta/sig^2)
}
}
M=(-yfs*Af+(B-C)*yfs^2)/bhf-(yfs*Af/bhf)^2-Hf
if (nl==1){
KK=t(Xl)%*%K%*%Xl}
else{
KK=t(Xl)%*%diag(c(K))%*%Xl
}
if (ni==1){
LL=t(Xi)%*%L%*%Xi}
else{
LL=t(Xi)%*%diag(c(L))%*%Xi
}
if (nr==1){
NN=t(Xr)%*%Hr%*%Xr}
else{
NN=t(Xr)%*%diag(c(Hr))%*%Xr
}
if (nf==1){
MM=t(Xf)%*%M%*%Xf}
else{
MM=t(Xf)%*%diag(c(M))%*%Xf
}
```

```
Hesb=KK+LL-NN+MM
#second derivative with respective to theta
sig=matrix(sig,nknots,1)
diagl=matrix(0,nknots,1)
diagi=matrix(0,nknots,1)
diagf=matrix(0,nknots,1)
for (j in 1:(nknots)){
diagl[j]=-(sum(sl*((pnorm(yls,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j])^2)/gl)+
sum((sl*(pnorm(yls,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j])/gl)^2))
diagi[j]=-(sum((sir*(pnorm(yirs,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j])^2
-sil*(pnorm(yils,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j])^2)/gi)
+sum(((sir*(pnorm(yirs,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j])
-sil*(pnorm(yils,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j]))/gi)^2))
diagf[j]=-(sum((dnorm(yfs,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)/bhf)^2))
}
tHesl=diag(c(diagl))
tHesi=diag(c(diagi))
tHesf=diag(c(diagf))
for (j in (2:(nknots))){
for (k in (1:(j-1))){
tHesl[k,j]=-(sum(sl*(pnorm(yls,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j])
*(pnorm(yls,knots[k],sig[k])
*sqrt(2*pi*(sig[k])^2)-gaus0[k])/gl)
+sum((pnorm(yls,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j])
```

```
*(pnorm(yls,knots[k],sig[k])
*sqrt(2*pi*(sig[k])^2)-gaus0[k])*(sl^2)/(gl^2)))

tHesi[k,j]=-(sum((sir*(pnorm(yirs,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j])
*(pnorm(yirs,knots[k],sig[k])
*sqrt(2*pi*(sig[k])^2)-gaus0[k])
-sil*(pnorm(yils,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j])
*(pnorm(yils,knots[k],sig[k])
*sqrt(2*pi*(sig[k])^2)-gaus0[k]))/gi)
+sum((sir*(pnorm(yirs,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j])
-sil*(pnorm(yils,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j]))
*(sir*(pnorm(yirs,knots[k],sig[k])
*sqrt(2*pi*(sig[k])^2)-gaus0[k])
-sil*(pnorm(yils,knots[k],sig[k])
*sqrt(2*pi*(sig[k])^2)-gaus0[k]))/gi^2))

tHesf[k,j]=-sum(dnorm(yfs,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)*dnorm(yfs,knots[k],sig[k])
*sqrt(2*pi*(sig[k])^2)/bhf^2)
}
}
tHes=tHesl+tHesi+tHesf
for (j in (2:(nknots))){
for (k in (1:(j-1))){
tHes[j,k]=tHes[k,j]
}
}
#second derivative with respective to beta and theta
dbtl=matrix(0,p,(nknots))
```

```
dbti=matrix(0,p,(nknots))
dbtr=matrix(0,p,(nknots))
dbtf=matrix(0,p,(nknots))
for (j in 1:(nknots)){
if (nl==1){
dbtl[,j]=(t(Xl)%*%((-dnorm(yls,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)*yls*sl
+sl*(pnorm(yls,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j])*bhl*yls)/gl
+(sl^2)*(pnorm(yls,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j])*bhl*yls/gl^2)
%*%matrix(1,nl,1))}
else{
dbtl[,j]=(t(Xl)%*%diag(c((-dnorm(yls,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)*yls*sl
+sl*(pnorm(yls,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j])*bhl*yls)/gl
+(sl^2)*(pnorm(yls,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j])
*bhl*yls/gl^2))%*%matrix(1,nl,1))
}

if (ni==1){
dbti[,j]=(t(Xi)%*%(((-dnorm(yirs,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)*yirs*sir
+sir*(pnorm(yirs,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j])*bhir*yirs)
-(-dnorm(yils,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)*yils*sil
+sil*(pnorm(yils,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j])*bhil*yils))/gi
-(sir*(pnorm(yirs,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j])
```

```
-sil*(pnorm(yils,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j]))
*(bhil*yils*sil-bhir*yirs*sir)/gi^2)
%*%matrix(1,ni,1))}
else{
dbti[,j]=(t(Xi)%*%diag(c(((-dnorm(yirs,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)*yirs*sir
+sir*(pnorm(yirs,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j])*bhir*yirs)
-(-dnorm(yils,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)*yils*sil
+sil*(pnorm(yils,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j])*bhil*yils))/gi
-(sir*(pnorm(yirs,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j])
-sil*(pnorm(yils,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)-gaus0[j]))
*(bhil*yils*sil-bhir*yirs*sir)/gi^2))
%*%matrix(1,ni,1))
}

if (nr==1){
dbtr[,j]=t(Xr)%*%(dnorm(yrs,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)*yrs)%*%matrix(1,nr,1)}
else{
dbtr[,j]=t(Xr)%*%diag(c(dnorm(yrs,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)*yrs))%*%matrix(1,nr,1)
}

if (nf==1){
dbtf[,j]=(t(Xf)%*%(((yfs-knots[j])
*dnorm(yfs,knots[j],sig[j])*sqrt(2*pi*(sig[j])^2)
/(sig[j])^2)*(yfs/bhf)
```

```
-dnorm(yfs,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)*yfs*Af/bhf^2
+dnorm(yfs,knots[j],sig[j])*sqrt(2*pi*(sig[j])^2)*yfs)
%*%matrix(1,nf,1))}
else{
dbtf[,j]=(t(Xf)%*%diag(c(((yfs-knots[j])
*dnorm(yfs,knots[j],sig[j])*sqrt(2*pi*(sig[j])^2)
/(sig[j])^2)*(yfs/bhf)
-dnorm(yfs,knots[j],sig[j])*sqrt(2*pi*(sig[j])^2)
*yfs*Af/bhf^2+dnorm(yfs,knots[j],sig[j])
*sqrt(2*pi*(sig[j])^2)*yfs))%*%matrix(1,nf,1))
}
}
dbt=dbtl+dbti+dbtr+dbtf
#Hessian matrix
H=rbind(cbind(Hesb,dbt),cbind(t(dbt),tHes))
HesbinvD=matrix(diag(solve(-Hesb)))
tHesp=-tHes+2*smooth*R/(1-smooth)
tHespinvD=matrix(diag(solve(tHesp)))
Hessp=rbind(cbind(-Hesb,-dbt),cbind(-t(dbt),tHesp))
varcov=solve(Hessp)%*%(-H)%*%solve(Hessp)
{if (p==1){
betavar=varcov[1:p,1:p]
}
else{
betavar=matrix(diag(varcov[1:p,1:p]))
}
}
betavar[betavar<0]=HesbinvD[betavar<0]
thetavar=matrix(diag(varcov[(p+1):(p+nknots)
, (p+1):(p+nknots)]))
varcov.theta=matrix(varcov[(p+1):(p+nknots)
, (p+1):(p+nknots)],nknots,nknots)
```

```
thetavar[thetavar<0]=tHespinvD[thetavar<0]
diag(varcov.theta)=thetavar
#selection criterion
df=sum(diag(solve(Hessp)%*%(-H)))
#df=sum(diag(Hesspinv%*%(-H)))
llik=sum(log(gl))+sum(log(gi))
-sum(chr)-sum(Xf%*%newbeta+chf)+sum(log(bhf))
AIC=-llik+df
#plots of estimated hazard and true hazard
time.point=matrix(seq(0,1,length=50))
num.p=dim(time.point)[1]
minv=min(time.point)
maxv=max(time.point)
nd=500
delta=(maxv-minv)/nd
binedg=matrix(seq(minv,maxv,delta))
mp=matrix(0,nd,1)
for (i in 1:nd){
mp[i]=(binedg[i]+binedg[i+1])/2
}
ISEbhs=matrix(0,nd,1)
ISEdfs=matrix(0,nd,1)
ebh=matrix(0,num.p,1)
edf=matrix(0,num.p,1)
ebhvar=matrix(0,num.p,1)
#if (datatype=='E'){
for (i in 1:num.p){
ebh[i]=t(dnorm(time.point[i],knots,sig)
*sqrt(2*pi*sig^2))%*%newtheta
edf[i]=exp(time.point[i])
*t(dnorm(exp(time.point[i]),knots,sig)
*sqrt(2*pi*sig^2))%*%newtheta*
exp(-t(pnorm(exp(time.point[i]),knots,sig)
```

```
*sqrt(2*pi*sig^2)-gaus0)%*%newtheta)
ebhvar[i]=t(dnorm(time.point[i],knots,sig)
*sqrt(2*pi*sig^2))%*%(varcov.theta)
%*%(dnorm(time.point[i],knots,sig)*sqrt(2*pi*sig^2))
}
tbh=3*time.point^2
tdf=3*exp(3*time.point-exp(3*time.point))
medf=matrix(0,nd,1)
for (i in 1:nd){
ISEbhs[i]=(3*mp[i]^2-t(dnorm(mp[i],knots,sig)
*sqrt(2*pi*sig^2))%*%newtheta)^2
medf[i]=exp(mp[i])*(t(dnorm(exp(mp[i]),knots,sig)
*sqrt(2*pi*sig^2))%*%newtheta)*
exp(-t(pnorm(exp(mp[i]),knots,sig)
*sqrt(2*pi*sig^2)-gaus0)%*%newtheta)
ISEdfs[i]=(3*exp(3*mp[i]-exp(3*mp[i]))-medf[i])^2
}
ISEbh=delta*sum(ISEbhs)
ISEdf=delta*sum(ISEdfs)
par(mfrow=c(2,1))
plot(time.point,tdf,type='l',col='red')
lines(time.point,edf,type='l',col='blue')
plot(time.point,tbh,type='l',col='red')
lines(time.point,ebh,type='l',col='blue')
return(list(newbeta=newbeta,newtheta=newtheta
,cvg=cvg,betavar=betavar
,thetavar=thetavar,ebhvar=ebhvar
,ebh=ebh,edf=edf,AIC=AIC,
ISEbh=ISEbh,ISEdf=ISEdf))
}
```

C.3 MC simulations for method comparisons

```
#Select the smoothing value
#smv: vector of smoothing values
selSm<-function(data,X,nknots,smv,maxiter){
  ns=dim(smv)[1]
  id=matrix(seq(1,ns,1))
  AICs=matrix(0,ns,1)
  np=dim(X)[2]
  newbetas=matrix(0,ns,np)
  betavars=matrix(0,ns,np)
  time.point=matrix(seq(0,1,0.05))
  num.p=dim(time.point)[1]
  ISEs=matrix(0,ns,1)
  ebhs=matrix(0,num.p,ns)
  for (i in 1:ns){
    aftfit=mplAFT(data,X,nknots,smv[i],maxiter)
    AICs[i]=aftfit$AIC
    newbetas[i,]=matrix(aftfit$newbeta,nrow=1)
    betavars[i,]=matrix(aftfit$betavar,nrow=1)
    ebhs[,i]=matrix(aftfit$ebh)
    ISEs[i]=aftfit$ISE
    print(i)
  }
  AICm=min(AICs)
  id=id[AICs==AICm]
  smop=smv[id]
  newbetaop=newbetas[id,]
  betavarop=betavars[id,]
  ISEop=ISEs[id]
  ebhop=ebhs[,id]
  return(list(smop=smop,AICm=AICm,newbetaop=newbetaop
              ,betavarop=betavarop,ISEop=ISEop,ebhop=ebhop
              ,AICs=AICs,newbetas=newbetas,
```

```
                betavars=betavars,ISEs=ISEs))
}
SmSel=selSm(data,X,nknots,smv,maxiter)

#Datasets used for method comparisons.
#N: N Monte Carlo samples.
GenerateData=function(beta,n,p,N){
obsets=matrix(0,n,3*N)
Xs=matrix(0,n,3*N)
for (i in 1:N){
Ca=matrix(rep(0,n))
ur=runif(n)
Ca[ur<=0.5]=1
Ca[ur>0.5]=0
#Covariate matrix
Xs[,(3*i-2):(3*i)]=cbind(Ca,runif(n,min=0,max=3)
,runif(n,min=0,max=5))
obsets[,(3*i-2):(3*i)]=event(beta
,Xs[,(3*i-2):(3*i)],p,n)
}
return(list(obsets=obsets,Xs=Xs))
}
dataset=GenerateData(beta,n,p,N)
datas=dataset$obsets
Xs=dataset$Xs

#MC simulations for the method by Komarek et al. (2005).
#nknots: number of knots
Ksim=function(datas,Xs,nknots,N){
betas=matrix(0,N,3)
astds=matrix(0,N,3)
time.point=matrix(seq(0,1,length=50))
num.p=dim(time.point)[1]
```

```
ebhs=matrix(0,num.p,N)
edfs=matrix(0,num.p,N)
ISEebhs=matrix(0,N,1)
ISEedfs=matrix(0,N,1)
for (i in 1:N){
t1=datas[,(3*i-2)]
t2=datas[,(3*i-1)]
t1[t1==0]=-NA
t2[t2==Inf]=NA
x1=Xs[,(3*i-2)]
x2=Xs[,(3*i-1)]
x3=Xs[,(3*i)]
surv <- Surv(t1, t2, type = "interval2")
Kfit<- smoothSurvReg(surv~x1+
x2+x3,info=FALSE, knots=seq(-6,6,length=nknots)
, difforder=3,init.logscale=0)
regre=Kfit$regre
betas[i,]=matrix(regre$Value[2:4],nrow=1)
astds[i,]=matrix(regre$Std.Error2[2:4],nrow=1)
scale=regre$Value[6]
intercept=regre$Value[1]
sig=(2/3)*diff(seq(-6,6,length=nknots))[1]
knots=matrix(seq(-6,6,length=nknots))
theta=Kfit$spline
theta=as.matrix(as.matrix(theta)[,3])
ebh=matrix(0,num.p,1)
edf=matrix(0,num.p,1)
for (k in 1:num.p){
ebh[k]=((1/time.point[k])*(1/scale)*(t(dnorm((log(time.point[k])
-intercept)/scale,knots,sig))%*%theta))/
(1-t(pnorm((log(time.point[k])-intercept)
/scale,knots,sig))%*%theta)
edf[k]=(1/scale)*(t(dnorm((time.point[k]-intercept)
```

```
/scale,knots,sig))%*%theta)
}
ebhs[,i]=ebh
edfs[,i]=edf
minv=min(time.point)
maxv=max(time.point)
nd=500
delta=(maxv-minv)/nd
binedg=matrix(seq(minv,maxv,delta))
mp=matrix(0,nd,1)
for (k in 1:nd){
mp[k]=(binedg[k]+binedg[k+1])/2
}
embh=matrix(0,nd,1)
emdf=matrix(0,nd,1)
ISEebh=matrix(0,nd,1)
ISEedf=matrix(0,nd,1)
for (k in 1:nd){
embh[k]=((1/mp[k])*(1/scale)*(t(dnorm((log(mp[k])-intercept)
/scale,knots,sig))%*%theta))/
(1-t(pnorm((log(mp[k])-intercept)/scale,knots,sig))%*%theta)
ISEebh[k]=(3*mp[k]^2-embh[k])^2
emdf[k]=(1/scale)*(t(dnorm((mp[k]-intercept)
/scale,knots,sig))%*%theta)
ISEedf[k]=(3*exp(3*mp[k]-exp(3*mp[k]))-emdf[k])^2
}
ISEebhs[i]=delta*sum(ISEebh)
ISEedfs[i]=delta*sum(ISEedf)
print(i)
}
mISEebh=mean(ISEebhs)
mISEedf=mean(ISEedfs)
meanb=matrix(colMeans(betas),1,3)
```

```
biasbeta=t(beta)-meanb
varb=matrix(0,1,3)
for (j in 1:3){
varb[1,j]=var(betas[,j])
}
stdb=sqrt(varb)
mseb=biasbeta^2+varb
astdb=matrix(colMeans(astds),1,3)
summaryb=rbind(meanb,biasbeta,stdb,astdb,mseb)
meanbh=matrix(colMeans(t(ebhs)))
meandf=matrix(colMeans(t(edfs)))
varbh=matrix(0,num.p,1)
vardf=matrix(0,num.p,1)
for (j in 1:num.p){
varbh[j]=var(ebhs[j,])
vardf[j]=var(edfs[j,])
}
stdbh=sqrt(varbh)
stddf=sqrt(vardf)
mclbh=meanbh-1.96*stdbh
mcubh=meanbh+1.96*stdbh
mcldf=meandf-1.96*stddf
mcudf=meandf+1.96*stddf
mclbh[mclbh<0]=0
mcldf[mcldf<0]=0
summarybh=cbind(meanbh,mclbh,mcubh,meandf,mcldf,mcudf)
return(list(summaryb=summaryb,summarybh=summarybh
,mISEebh=mISEebh,mISEedf=mISEedf))
}
K.sim=Ksim(datas,Xs,nknots,N)

#Monte Carlo simulations for our MPL method.
#n: sample size.
```

```
#p: censoring proportion.
#nknots: number of knots.
#beta: beta values.
#N: N Monte Carlo samples.
#maxiter: number of iterations in the Newton-MI algorithm.
#smop: smoothing value selected by SmSel.
MCsimu<-function(datas,Xs,nknots,smop,beta,N,maxiter){
#initials
np=dim(beta)[1]
betas=matrix(0,N,np)
betavars=matrix(0,N,np)
time.point=matrix(seq(0,1,length=50))
num.p=dim(time.point)[1]
ebhs=matrix(0,num.p,N)
edfs=matrix(0,num.p,N)
ebhvars=matrix(0,num.p,N)
ISEbhs=matrix(0,N,1)
ISEdfs=matrix(0,N,1)
#MC simulation
for(i in 1:N){
aftfit=mplAFT(datas[,(3*i-2):(3*i)]
,Xs[,(3*i-2):(3*i)],nknots,smop,maxiter)
betas[i,]=matrix(aftfit$newbeta,nrow=1)
betavars[i,]=matrix(aftfit$betavar,nrow=1)
ebhs[,i]=matrix(aftfit$ebh)
edfs[,i]=matrix(aftfit$edf)
ebhvars[,i]=matrix(aftfit$ebhvar)
ISEbhs[i]=aftfit$ISEbh
ISEdfs[i]=aftfit$ISEdf
print(i)
}
mISEbh=mean(ISEbhs)
mISEdf=mean(ISEdfs)
```

```
#mean, bias, std, astd and mse of beta
meanb=matrix(colMeans(betas),1,np)
biasbeta=meanb-t(beta)
varb=matrix(0,1,np)
for (j in 1:np){
varb[1,j]=var(betas[,j])
}
stdb=sqrt(varb)
astdb=matrix(colMeans(sqrt(betavars)),1,np)
mseb=biasbeta^2+varb
#summary for beta
summaryb=rbind(meanb,biasbeta,stdb,astdb,mseb)
#sample mean, std and astd of the hazard
meanbh=matrix(colMeans(t(ebhs)))
meandf=matrix(colMeans(t(edfs)))
varbh=matrix(0,num.p,1)
vardf=matrix(0,num.p,1)
for (j in 1:num.p){
varbh[j]=var(ebhs[j,])
vardf[j]=var(edfs[j,])
}
# MC standard deviation
stdbh=sqrt(varbh)
stddf=sqrt(vardf)
#Average asymptotic standard deviation
aastdbh=matrix(colMeans(t(sqrt(ebhvars))))
# 95% MC PWCI
mclbh=meanbh-1.96*stdbh
mcubh=meanbh+1.96*stdbh
mcldf=meandf-1.96*stddf
mcudf=meandf+1.96*stddf
# average of asymptotic 95% PWCI
aasylbh=meanbh-1.96*aastdbh
```

```
aasyubh=meanbh+1.96*aastdbh
mclbh[mclbh<0]=0
mcldf[mcldf<0]=0
aasylbh[aasylbh<0]=0
summarybh=cbind(meanbh,mclbh,mcubh,aasylbh,aasyubh
,meandf,mcldf,mcudf)
return(list(summaryb=summaryb,summarybh=summarybh
,mISEbh=mISEbh,mISEdf=mISEdf))
}
sim=MCsimu(datas,Xs,nknots,smop,beta,N,maxiter)
```