HW#6 Survival Analysis I

NAME: JIA-HAN SHIH

Problem 1.

Perform penalized maximum likelihood estimation with optimal smoothing for the lung cancer data with $x_1 = x_2 = ZNF264$ under the Clayton copula ($\theta = 18$).

Solution.

To find the optimal smoothing parameters κ_1 and κ_2, we first apply the "*splineCox.reg*" function in R "*joint.Cox*" package to the lung cancer data which is available in the R "*compound.Cox*" package. The data consists (t_i, δ_i), $i = 1, 2, \ldots, n$, where $t_i = \min(T_i, U_i)$ is the observed event time, T_i is time-to-death, U_i is the censoring time, $\delta_i = \mathbf{I}(t_i = T_i)$ is the censoring indicator, and the sample size is $n = 125$.

The smoothing parameter for the marginal distribution of death is $\kappa_1 = \mathbf{50909094}$. The outputs of the "*splineCox.reg*" function is given in Figure 1. On the other hand, the smoothing parameter for the marginal distribution of censoring time is $\kappa_2 = \mathbf{1818190}$. The outputs of the "*splineCox.reg*" function is given in Figure 2.

Some additional investigations have been done here. We slightly modify the R code in the "*splineCox.reg*" function. The modified function is given in Appendix 1. The new outputs from the modified function are very similar to the original results, they are $\kappa_1^* = \mathbf{51616164}$ and $\kappa_2^* = \mathbf{2121222}$. However, we obtain smoother figures (Figure 3 and Figure 4).

All figures reveal that

$$df_j = \mathrm{tr}\{\hat{H}_{PLj}^{-1}\hat{H}_j\} = 6(1+5) \to 3(1+2) \quad \text{as } \kappa_j \to \infty, \quad j = 1, 2.$$

NOTE: The grid points are set as "seq(10,**7e+7**,length.out = 100)" except for Figure 2 due to some errors. The grid points for Figure 2 is "seq(10,**1e+7**,length.out = 100)".

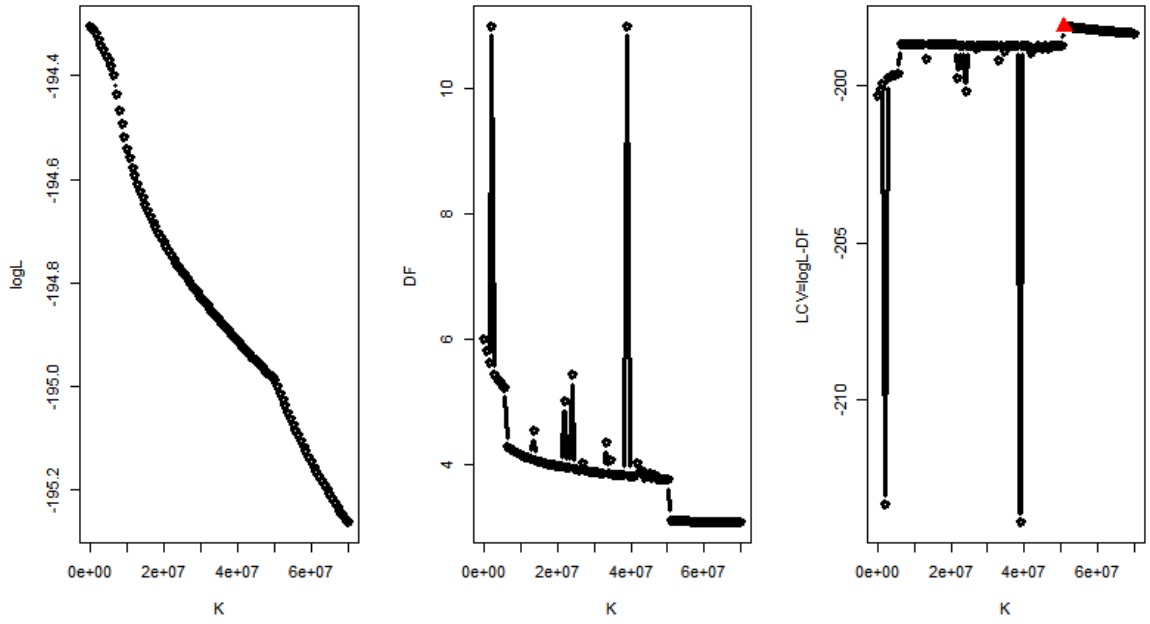

Figure 1. The output of the "*splineCox.reg*" function for death (κ_1).

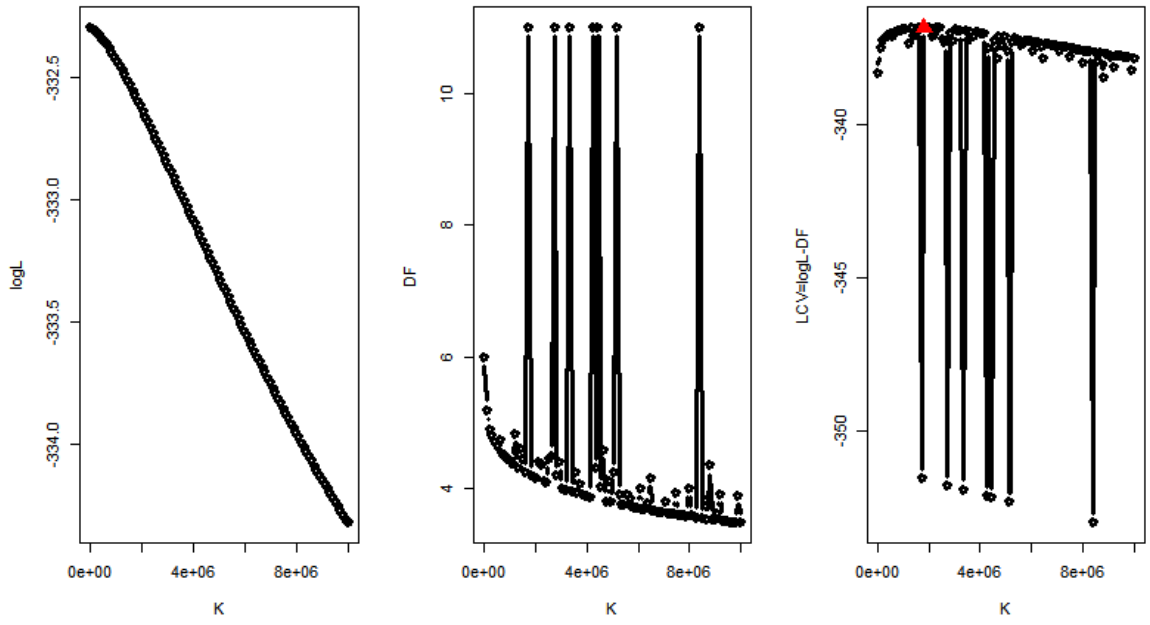

Figure 2. The output of the "*splineCox.reg*" function for censoring time (κ_2).

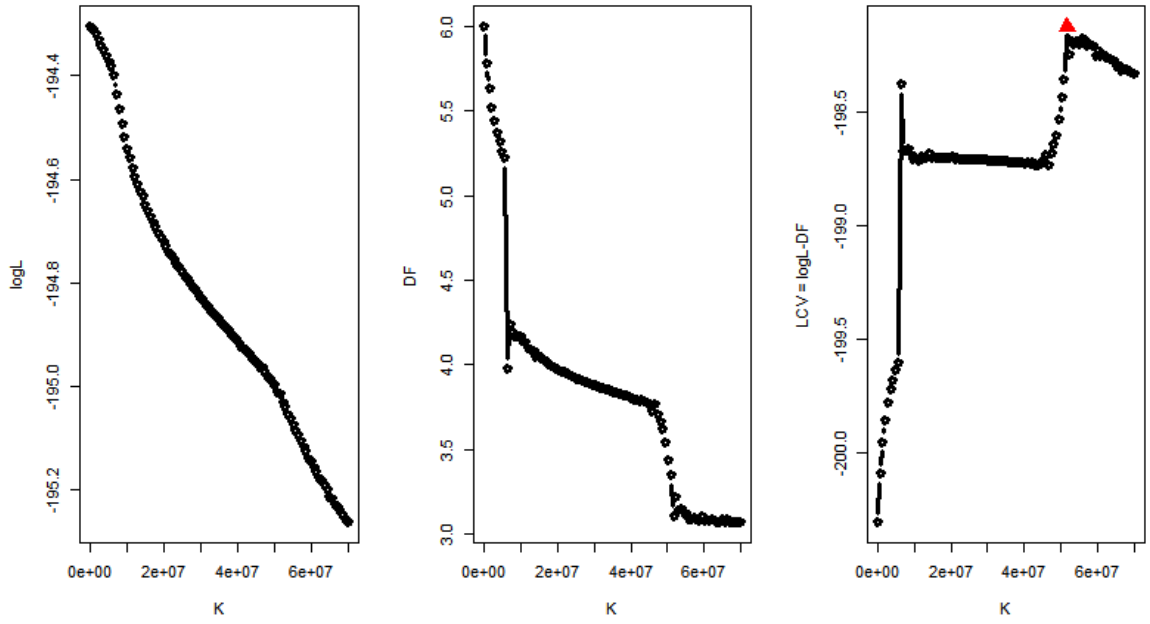

Figure 3. The output of the modified function for death (κ_1^*).

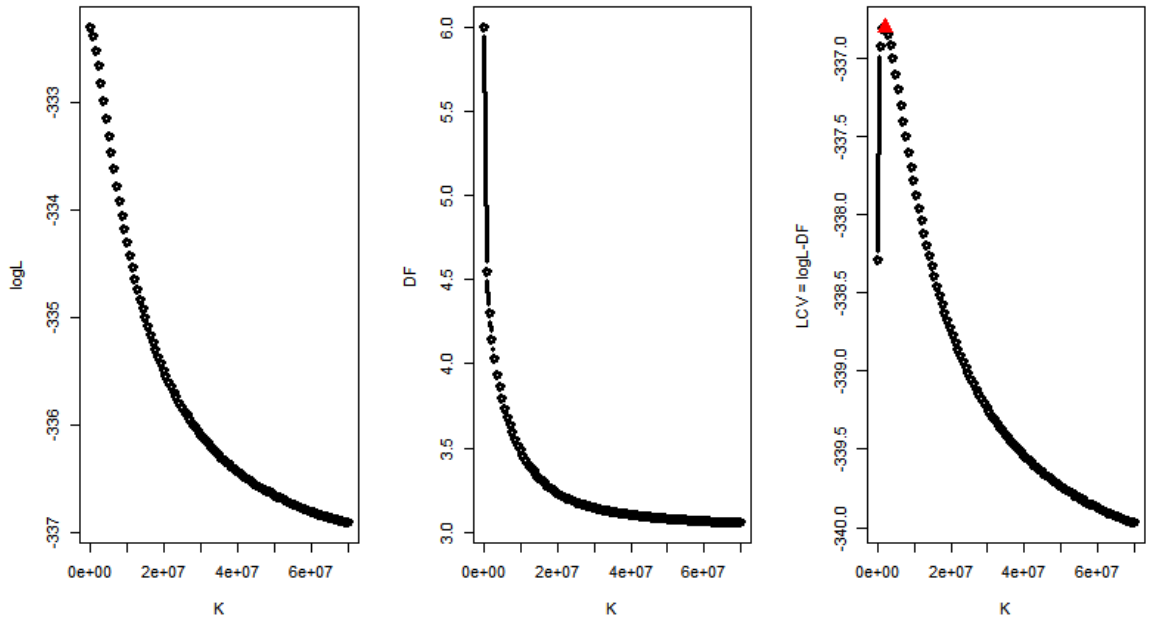

Figure 4. The output of the modified function for censoring time (κ_2^*).

We use $\kappa_1^* = 51616164$ and $\kappa_2^* = 1818190$ for the following analysis.

With the optimal smoothing parameters in hand, we can perform the penalized maximum likelihood estimation. Under the Clayton copula ($\theta = 18$), the penalized log-likelihood function for the spline model is

$$\begin{aligned}\ell_{PL}(\beta_1, \beta_2, \Lambda_0, \Gamma_0 \mid \theta) = &\sum_{i=1}^{n} \delta_i [\beta_1 x_{1i} - \theta \log\{ S_T(t_i \mid x_{1i}) \} + \log \lambda_0(t_i)] - \kappa_1^* \mathbf{g}^{\mathrm{T}} \boldsymbol{\Omega} \mathbf{g} \\ &+ \sum_{i=1}^{n} (1-\delta_i) [\beta_2 x_{2i} - \theta \log\{ S_U(t_i \mid x_{2i}) \} + \log \gamma_0(t_i)] - \kappa_2^* \mathbf{h}^{\mathrm{T}} \boldsymbol{\Omega} \mathbf{h} \\ &- (1+1/\theta) \sum_{i=1}^{n} \log\{ S_T(t_i \mid x_{1i})^{-\theta} + S_U(t_i \mid x_{2i})^{-\theta} - 1 \},\end{aligned}$$

where

$$\lambda_0(t) = \sum_{\ell=1}^{5} g_\ell M_\ell(t), \qquad \gamma_0(t) = \sum_{\ell=1}^{5} h_\ell M_\ell(t),$$

$$\Lambda_0(t) = \sum_{\ell=1}^{5} g_\ell I_\ell(t), \qquad \Gamma_0(t) = \sum_{\ell=1}^{5} g_\ell I_\ell(t),$$

$$S_T(t \mid x_1) = \exp\{ -\Lambda_0(t) e^{\beta_1 x_1} \}, \qquad S_U(t \mid x_2) = \exp\{ -\Gamma_0(t) e^{\beta_2 x_2} \},$$

$$\mathbf{g} = (g_1, g_2, g_3, g_4, g_5)^{\mathrm{T}}, \quad g_\ell > 0, \quad \mathbf{h} = (h_1, h_2, h_3, h_4, h_5)^{\mathrm{T}}, \quad h_\ell > 0, \quad \ell = 1, \ldots, 5,$$

$$\boldsymbol{\Omega} = \frac{1}{\Delta^5} \begin{bmatrix} 192 & -132 & 24 & 12 & 0 \\ -132 & 96 & -24 & -12 & 12 \\ 24 & -24 & 24 & -24 & 24 \\ 12 & -12 & -24 & 96 & -132 \\ 0 & 12 & 24 & -132 & 192 \end{bmatrix}, \quad \text{and} \quad \Delta = \frac{\max_i(t_i) - \min_i(t_i)}{2}.$$

The formulas for the cubic M- and I-spline basis are available in Emura and Chen (2018).

We first try to obtain the penalized maximum likelihood estimator (MLE) by the R "*optim*" and "*nlm*" functions. To make our program stable, we consider transformations for the positive parameters as $g_\ell \equiv \exp(\tilde{g}_\ell)$ and $h_\ell \equiv \exp(\tilde{h}_\ell)$, $\ell = 1, \ldots, 5$. We estimate the transformed parameters $\tilde{g}_\ell$ and $\tilde{h}_\ell$ which have unrestricted ranges. The initial values are all set as 0. The estimation results are given in Table 1.

Table 1. Estimation results by using R functions based on the lung cancer data

Parameter	*optim* MLE (95% CI)	*nlm* MLE (95% CI)
β_1	0.1918 (-0.0028, 0.3864)	0.1921 (0.0098, 0.3743)
β_2	0.1714 (-0.0146, 0.3573)	0.1716 (-0.0042, 0.3474)
g_1	0.0227 (0.0051, 0.1006)	0.0226 (0.0051, 0.0998)
g_2	0.3567 (0.2343, 0.5430)	0.3565 (0.2369, 0.5366)
g_3	1.0307 (0.7145, 1.4869)	1.0302 (0.7208, 14724)
g_4	1.7088 (1.1618, 2.5132)	1.7080 (1.1715, 2.4900)
g_5	1.0236 (0.6846, 1.5304)	1.0231 (0.6901, 1.5168)
h_1	0.0000 (-, -)	0.0000 (-, -)
h_2	0.4005 (0.2403, 0.6676)	0.4004 (0.2411, 0.6650)
h_3	1.2283 (0.7909, 1.9077)	1.2277 (0.7967, 1.8920)
h_4	1.8894 (1.1973, 2.9815)	1.8889 (1.2127, 2.9421)
h_5	1.1254 (0.6447, 1.9644)	1.1251 (0.6518, 1.9422)

Table 1 reveals that both "*optim*" and "*nlm*" produce similar results on the MLE. Somehow the estimates of h_1 are very small, "*optim*" returns "1.7×10^{-6}" and "*nlm*" returns 7.5×10^{-9}. The 95% confidence intervals (CI) are computed as

$$\hat{\beta}_j \pm 1.96\mathrm{SE}(\hat{\beta}_j), \qquad j=1,2,$$

$$\exp\{\hat{\tilde{g}}_\ell \pm 1.96\mathrm{SE}(\hat{\tilde{g}}_\ell)\}, \qquad \exp\{\hat{\tilde{h}}_\ell \pm 1.96\mathrm{SE}(\hat{\tilde{h}}_\ell)\}, \qquad \ell=1,\ldots,5.$$

The 95% CI of h_1 is unreliable since $\mathrm{SE}(\hat{\tilde{h}}_1)$ is too large. Also, in order to obtain a positive definite covariance matrix of the MLE when we use "*nlm*", we add 10^{-4} to the diagonal elements of the numerical hessian matrix. The 95% CI produced by "*nlm*" are slightly narrower than the 95% CI produced by "*optim*" may relate to this adjustment. Therefore, we suggest using "*optim*" since it does not require adjustments. The estimated baseline hazard functions (by using "*optim*") and their 95% CI are given in Figure 5.

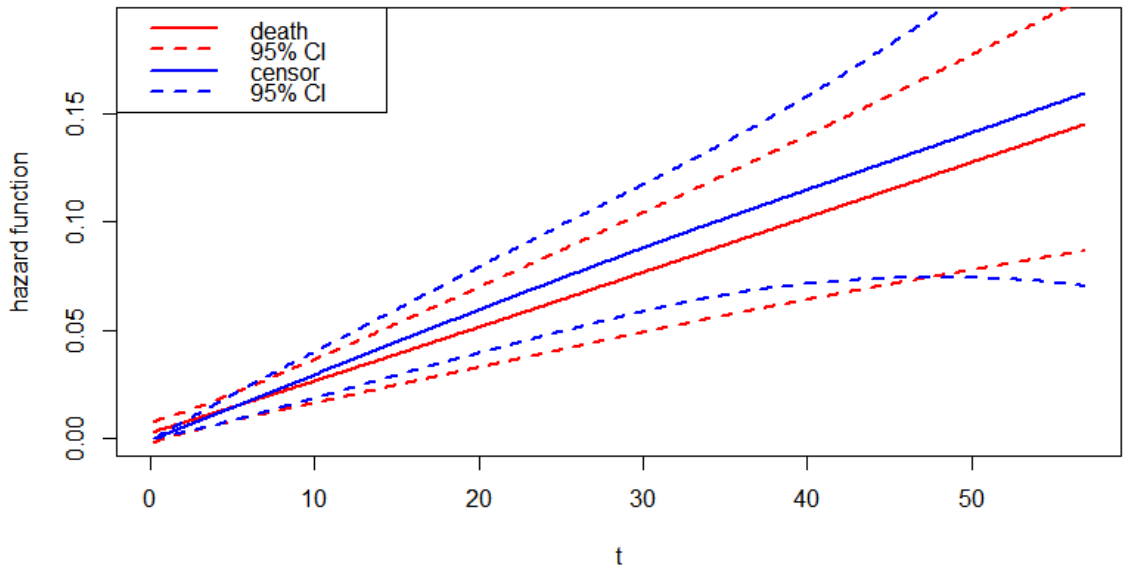

Figure 5. Estimated hazard functions based on the lung cancer data.

Figure 5 reveals that the estimated hazard functions are very similar to linear functions. This is due to the large values of smoothing parameters, we have $df_1 = 3.11$ and $df_2 = 4.14$ which are both very close to 3. For fixed t, the 95% CI for the baseline hazard functions are computed as

$$\hat{\lambda}_0(t) \pm 1.96 \mathrm{SE}\{\hat{\lambda}_0(t)\},$$

where

$$\mathrm{SE}\{\hat{\lambda}_0(t)\} = \sqrt{\mathrm{var}\{\hat{\lambda}_0(t)\}} = \sqrt{\mathrm{var}\{\mathbf{M}(t)\hat{\mathbf{g}}\}} = \sqrt{\mathbf{M}(t)\,\mathrm{var}(\hat{\mathbf{g}})\mathbf{M}^{\mathrm{T}}(t)}\,.$$

The value of $\mathrm{var}(\hat{\mathbf{g}})$ are available from the output of the "*optim*". The 95% CI for $\hat{\gamma}_0(t)$ is obtained in a similar fashion. The R codes are given in Appendix 2.

Next, we aim to apply the Newton-Raphson algorithm to obtain the penalized MLE. We first define the target parameter as $\boldsymbol{\varphi} = (\beta_1, \beta_2, \mathbf{g}, \mathbf{h})$. Then, we state the Newton-Raphson algorithm below.

Algorithm 1 Newton-Raphson algorithm

Step 1 Choose initial value $\boldsymbol{\varphi}^{(0)}$

Step 2 Repeat the Newton-Raphson iteration for $k = 0, 1, 2, \ldots$

$$\boldsymbol{\varphi}^{(k+1)} = \boldsymbol{\varphi}^{(k)} - \left\{ \frac{\partial^2 \ell_{PL}(\boldsymbol{\varphi} \mid \theta)}{\partial \boldsymbol{\varphi}^{\mathrm{T}} \partial \boldsymbol{\varphi}} \bigg|_{\boldsymbol{\varphi}=\boldsymbol{\varphi}^{(k)}} \right\}^{-1} \left\{ \frac{\partial \ell_{PL}(\boldsymbol{\varphi} \mid \theta)}{\partial \boldsymbol{\varphi}} \bigg|_{\boldsymbol{\varphi}=\boldsymbol{\varphi}^{(k)}} \right\}.$$

- If $\max\{ |\boldsymbol{\varphi}^{(k+1)} - \boldsymbol{\varphi}^{(k)}| \} < \varepsilon$, where ε is a small number, then stop and $\boldsymbol{\varphi}^{(k)}$ is the penalized MLE.

In order to use Algorithm 1, we need all the first- and second-order derivatives of the penalized log-likelihood function. However, instead of putting all $12 + 12 \times 12 = 156$ formulas, we give some comparisons between the exact values and the numerical derivatives below. Our formulas agree with the numerical derivative produced by "*grad*" and "*hessian*" functions in the R "*numDeriv*" package.

Score function

```
> grad(llh.func,try.para)
 [1] -1182.574270  1130.035668  -822.208430   -50.181552  -291.727853   197.985348  -441.840586
 [8]   314.694914   319.243537   190.895834    88.773206    -3.178413
> score.func(try.para)
 [1] -1182.574270  1130.035668  -822.208430   -50.181552  -291.727853   197.985348  -441.840586
 [8]   314.694914   319.243537   190.895834    88.773206    -3.178413
```

Hessian function (partly reported)

```
> hessian(llh.func,try.para)[1,]
 [1] -53802.8856  51935.2695 -14641.9386 -10589.9209  -5733.8359  -2211.7212   -448.7316  14135.1520
 [9]  10185.8201   5502.5731   2123.1329    434.6111
> hessian.func(try.para)[1,]
 [1] -53802.8861  51935.2699 -14641.9388 -10589.9212  -5733.8361  -2211.7213   -448.7317  14135.1521
 [9]  10185.8202   5502.5732   2123.1330    434.6111
>
> hessian(llh.func,try.para)[2,]
 [1]  51935.2695 -50501.5398  14135.1520  10185.8201   5502.5731   2123.1329    434.6111 -13745.6028
 [9]  -9856.1351  -5305.5009  -2045.3223   -421.9771
> hessian.func(try.para)[2,]
 [1]  51935.2699 -50501.5402  14135.1521  10185.8202   5502.5732   2123.1330    434.6111 -13745.6031
 [9]  -9856.1354  -5305.5011  -2045.3224   -421.9771
>
> hessian(llh.func,try.para)[3,]
 [1] -14641.93856  14135.15195  -6785.32417  -2896.91589  -1831.79369   -616.15301    -89.39827
 [8]   5096.74344   3482.96130   1723.48657    562.39589     89.39827
> hessian.func(try.para)[3,]
 [1] -14641.93882  14135.15208  -6785.32417  -2896.91589  -1831.79369   -616.15301    -89.39827
 [8]   5096.74344   3482.96130   1723.48656    562.39589     89.39827
>
> hessian(llh.func,try.para)[8,]
 [1]  14135.15195 -13745.60281   5096.74344   3482.96130   1723.48656    562.39589     89.39827
 [8]  -4821.00532  -3462.73386  -1729.41153   -564.81554    -89.39827
> hessian.func(try.para)[8,]
 [1]  14135.15208 -13745.60307   5096.74344   3482.96130   1723.48656    562.39589     89.39827
 [8]  -4821.00532  -3462.73386  -1729.41153   -564.81554    -89.39827
```

After checking our first- and second-order derivatives of the penalized log-likelihood function are correct. We then use Algorithm 1 to perform the Newton-Raphson algorithm to obtain the penalized MLE. However, Algorithm 1 constantly diverge. It even diverges when we set the initial value as the estimates produced by "*optim*" or "*nlm*". This may relate to that we are maximizing the penalized log-likelihood function with respect to total 12 parameters.

NOTE: The R codes for the first- and second-order derivatives of penalized log-likelihood function are not given here since they are too long.

Appendix 1

```
Mod.splineCox.reg = function (t.event,event,Z,xi1 = min(t.event),xi3 = max(t.event),
                                        kappa_grid = c(seq(10,1e+17,length = 30)),LCV_plot = TRUE) {

  d = event
  p = ncol(Z)
  Omega = c(192,-132,24,12,0,-132,96,-24,-12,12,24,
            -24,24,-24,24,12,-12,-24,96,-132,0,12,24,-132,192)
  Omega = matrix(Omega,5,5)/((xi3-xi1)/2)^5
  l.func = function(phi) {
    b = phi[(5+1):(5+p)]
    g = exp(phi[1:5])
    l = -K*t(g)%*%Omega%*%g
    r = as.vector(M.spline(t.event,xi1 = xi1,xi3 = xi3)%*%g)
    R = as.vector(I.spline(t.event,xi1 = xi1,xi3 = xi3)%*%g)
    bZ = as.numeric(Z%*%b)
    l = l+sum(d*(log(r)+bZ))
    l = l-sum(exp(bZ)*R)
    return(-l)
  }

  DF_upper = 18+p
  L = DF = NULL

  for (k in 1:length(kappa_grid)) {
    K = kappa_grid[k]
    initial.par = rep(0,5+p)
    try.par = initial.par
    repeat {
      res = try(optim(try.par,l.func,control = list(maxit = 5000,reltol = 1e-9),
                      method = "BFGS",hessian = TRUE),silent = TRUE)
      if (class(res) == "try-error") {
        try.par = initial.par+runif(5+p,-1,1)
        next
      }
      if (res$convergence == 0) {break}
```

```
        try.par = initial.par+runif(5+p,-1,1)
      }
      D_PL = diag(c(1/exp(res$par[1:5]),rep(1,p)))
      H_PL = -D_PL%*%res$hessian%*%D_PL
      H = H_PL
      H[1:5,1:5] = H[1:5,1:5]+2*K*Omega
      K = 0
      L[k] = -l.func(res$par)
      if (is.na(det(H_PL)) | det(H_PL) == 0) {DF[k] = DF_upper}
      else {DF[k] = min(max(sum(diag(solve(H_PL,tol = 10^(-50))%*%H)),p+2),DF_upper)}
    }

    K_est = kappa_grid[L-DF == max(L-DF)][1]
    DF_est = DF[L-DF == max(L-DF)][1]

    if (LCV_plot == TRUE) {
      par(mfrow = c(1,3))
      plot(kappa_grid,L,xlab = "K",ylab = "logL",type = "b",lwd = 3)
      plot(kappa_grid,pmin(DF,10+p),xlab = "K",ylab = "DF",type = "b",lwd = 3)
      plot(kappa_grid,L-DF,xlab = "K",ylab = "LCV = logL-DF",type = "b",lwd = 3)
      points(K_est,max(L-DF),xlab = "K",col = "red",pch = 17,cex = 2)
    }

    list(kappa = K_est,DF = DF_est,LCV = max(L-DF),kappa_grid = kappa_grid,
         L_grid = L,DF_grid = DF,LCV_grid = L-DF)

}
```

Appendix 2

```
minllh.func = function(para) {

  beta1.para = para[1]
  beta2.para = para[2]
  g1 = exp(para[3])
  g2 = exp(para[4])
  g3 = exp(para[5])
  g4 = exp(para[6])
  g5 = exp(para[7])
  h1 = exp(para[8])
  h2 = exp(para[9])
  h3 = exp(para[10])
  h4 = exp(para[11])
  h5 = exp(para[12])

  g.para = c(g1,g2,g3,g4,g5)
  h.para = c(h1,h2,h3,h4,h5)

  Pen1 = Mod.kappa1*as.numeric(t(g.para)%*%Omega%*%g.para)
  Pen2 = Mod.kappa2*as.numeric(t(h.para)%*%Omega%*%h.para)

  M.sp = M.spline(t.event,tmin,tmax)
  I.sp = I.spline(t.event,tmin,tmax)
  l.f = M.sp%*%g.para
  g.f = M.sp%*%h.para
  L.f = I.sp%*%g.para
  G.f = I.sp%*%h.para

  coef1 = exp(beta1.para*Z1)
  coef2 = exp(beta2.para*Z2)
  ST = exp(-coef1*L.f)
  SU = exp(-coef2*G.f)
  Ct = ST^(-theta)+SU^(-theta)-1

  A1 = event*(beta1.para*Z1-theta*log(ST)+log(l.f))
```

```
    A2 = (1-event)*(beta2.para*Z2-theta*log(SU)+log(g.f))
    A3 = (1+1/theta)*log(Ct)

    -(sum(A1+A2-A3)-Pen1-Pen2)

}

library(joint.Cox)
library(compound.Cox)

data("Lung")
t.event = Lung$t.vec
event = Lung$d.vec
Z1 = as.matrix(Lung$ZNF264)
Z2 = Z1

tmin = min(t.event)
tmax = max(t.event)
Delta = (tmax-tmin)/2
Omega = matrix(c(192,-132,24,12,0,-132,96,-24,-12,12,24,-24,24,-24,24,
                 12,-12,-24,96,-132,0,12,24,-132,192),5,5)/Delta^5
theta = 18

res1 = splineCox.reg(t.event,event,Z1,kappa_grid = seq(10,7e+7,length.out = 100))
kappa1 = res1$kappa; kappa1

res2 = splineCox.reg(t.event,1-event,Z2,kappa_grid = seq(10,1e+7,length.out = 100))
kappa2 = res2$kappa; kappa2

Mod.res1 = Mod.splineCox.reg(t.event,event,Z1,kappa_grid = seq(10,7e+7,length.out = 100))
Mod.kappa1 = Mod.res1$kappa; Mod.kappa1

Mod.res2 = Mod.splineCox.reg(t.event,1-event,Z2,kappa_grid = seq(10,7e+7,length.out = 100))
Mod.kappa2 = Mod.res2$kappa; Mod.kappa2

res.opt = optim(rep(0,12),minllh.func,control = list(reltol = 1e-9),method = "BFGS",hessian = TRUE); res.opt
res.nlm = nlm(minllh.func,rep(0,12),iterlim = 1000,steptol = 1e-9,hessian = TRUE); res.nlm
```

```
round(c(res.opt$par[1:2],exp(res.opt$par[3:12])),6)
round(c(res.nlm$estimate[1:2],exp(res.nlm$estimate[3:12])),6)

g.est = exp(res.opt$par[3:7])
h.est = exp(res.opt$par[8:12])
var.matrix = solve(res.opt$hessian,tol = 1e-50)[3:12,3:12]
g.var = diag(g.est)%*%var.matrix[1:5,1:5]%*%diag(g.est)
h.var = diag(h.est)%*%var.matrix[6:10,6:10]%*%diag(h.est)

round(res.opt$par[1:2]-1.96*sqrt(diag(solve(res.opt$hessian,tol = 1e-50))[1:2]),4)
round(res.opt$par[1:2]+1.96*sqrt(diag(solve(res.opt$hessian,tol = 1e-50))[1:2]),4)
round(exp(res.opt$par[3:7]-1.96*sqrt(diag(solve(res.opt$hessian,tol = 1e-50))[3:7])),4)
round(exp(res.opt$par[3:7]+1.96*sqrt(diag(solve(res.opt$hessian,tol = 1e-50))[3:7])),4)
round(exp(res.opt$par[8:12]-1.96*sqrt(diag(solve(res.opt$hessian,tol = 1e-50))[8:12])),4)[-1]
round(exp(res.opt$par[8:12]+1.96*sqrt(diag(solve(res.opt$hessian,tol = 1e-50))[8:12])),4)[-1]

#det(solve(res.nlm$hessian,tol = 1e-50))
#det(solve(res.nlm$hessian+diag(1e-4,12),tol = 1e-50))
round(res.nlm$estimate[1:2]-1.96*sqrt(diag(solve(res.nlm$hessian+diag(1e-4,12),tol = 1e-50))[1:2]),4)
round(res.nlm$estimate[1:2]+1.96*sqrt(diag(solve(res.nlm$hessian+diag(1e-4,12),tol = 1e-50))[1:2]),4)
round(exp(res.nlm$estimate[3:7]-1.96*sqrt(diag(solve(res.nlm$hessian+diag(1e-4,12),tol = 1e-50))[3:7])),4)
round(exp(res.nlm$estimate[3:7]+1.96*sqrt(diag(solve(res.nlm$hessian+diag(1e-4,12),tol = 1e-50))[3:7])),4)
round(exp(res.nlm$estimate[8:12]-1.96*sqrt(diag(solve(res.nlm$hessian+diag(1e-4,12),tol =
1e-50))[8:12])),4)[-1]
round(exp(res.nlm$estimate[8:12]+1.96*sqrt(diag(solve(res.nlm$hessian+diag(1e-4,12),tol =
1e-50))[8:12])),4)[-1]

t.grid = seq(tmin,tmax,length.out = 500)
ht.hat = M.spline(t.grid,tmin,tmax)%*%g.est
hu.hat = M.spline(t.grid,tmin,tmax)%*%h.est

ht.se = sqrt(diag(M.spline(t.grid,tmin,tmax)%*%g.var%*%t(M.spline(t.grid,tmin,tmax))))
hu.se = sqrt(diag(M.spline(t.grid,tmin,tmax)%*%h.var%*%t(M.spline(t.grid,tmin,tmax))))

par(mfrow = c(1,1))
plot(t.grid,ht.hat,type = "l",lwd = 2,ylab = "hazard function",xlab = "t",col = "red",
     ylim = c(0,max(c(ht.hat,hu.hat))*1.2))
```

```
lines(t.grid,hu.hat,lwd = 2,col = "blue")
lines(t.grid,ht.hat+1.96*ht.se,lty = 2,lwd = 2,col = "red")
lines(t.grid,ht.hat-1.96*ht.se,lty = 2,lwd = 2,col = "red")
lines(t.grid,hu.hat+1.96*hu.se,lty = 2,lwd = 2,col = "blue")
lines(t.grid,hu.hat-1.96*hu.se,lty = 2,lwd = 2,col = "blue")
legend("topleft",c("death","95% CI","censor","95% CI"),lwd = 2,
       lty = c(1,2,1,2),col = c("red","red","blue","blue"))
```