AN ALGORITHM FOR FITTING MIXTURES OF GOMPERTZ DISTRIBUTIONS TO CENSORED SURVIVAL DATA

G.J. MCLACHLAN[1]

Centre for Statistics

Department of Mathematics

The University of Queensland

S.K. NG

Centre for Statistics

Department of Mathematics,

The University of Queensland

P. ADAMS

Department of Mathematics

The University of Queensland

D.C. MCGIFFIN

Department of Surgery

The University of Alabama at Birmingham

A.J. GAILBRAITH,

Department of Surgery

The Prince Charles Hospital

[1]*Address for correspondence*: Department of Mathematics, The University of Queensland, Queensland 4072, Australia.

PURPOSE AND DESCRIPTION

We consider the fitting of a mixture of two Gompertz distributions to censored survival data. This model is therefore applicable where there are two distinct causes for failure that act in a mutually exclusive manner, and the baseline failure time for each cause follows a Gompertz distribution. For example, in a study of a disease such as breast cancer, suppose that failure corresponds to death, whose cause is attributed either to breast cancer or some other cause. In this example, the mixing proportion for the component of the mixture representing time to death from a cause other than breast cancer may be interpreted to be the cure rate for breast cancer (Gordon, 1990a and 1990b). This Gompertz mixture model whose components are adjusted multiplicatively to reflect the age of the patient at the origin of the survival time, is fitted by maximum likelihood via the EM algorithm (Dempster, Laird, and Rubin, 1977). There is the provision to handle the case where the mixing proportions are formulated in terms of a logistic model to depend on a vector of covariates associated with each survival time. The algorithm can also handle the case where there is only one cause of failure, but which may happen at infinity for some patients with a nonzero probability (Farewell, 1982).

METHOD

Let T be a random variable denoting the failure time for a patient with an associated p-dimensional covariate vector $\boldsymbol{x}$ which contains at least the single covariate w. The density of T is postulated to be the two-component mixture

$$f(t; \boldsymbol{x}, \boldsymbol{\Psi}) = \sum_{i=1}^{2} \pi_i(\boldsymbol{x}; \boldsymbol{\alpha}) f_i(t; w, \boldsymbol{\theta}_i), \tag{1}$$

where

$$\begin{aligned} f_i(t; w, \boldsymbol{\theta}_i) &= h_i(t; w, \boldsymbol{\theta}_i) \exp\{-\Lambda_i(t; w, \boldsymbol{\theta}_i)\}, & (2)\\ h_i(t; w, \boldsymbol{\theta}_i) &= \exp(\gamma_i w + \lambda_i + \beta_i t), & (3)\\ \Lambda_i(t; w, \boldsymbol{\theta}_i) &= e^{\gamma_i w} e^{\lambda_i} (e^{\beta_i t} - 1)/\beta_i, & (4) \end{aligned}$$

and

$$\boldsymbol{\theta}_i = (\lambda_i, \beta_i, \gamma_i)'$$

for $i = 1, 2$, and where

$$\boldsymbol{\Psi} = (\boldsymbol{\alpha}', \boldsymbol{\theta}_1', \boldsymbol{\theta}_2')'.$$

Here a prime denotes vector transpose.

The mixing proportion $\pi_i(\boldsymbol{x}; \boldsymbol{\alpha})$ is modelled to depend on $\boldsymbol{x}$ through the logistic model for which

$$\begin{aligned} \pi_1(\boldsymbol{x}; \boldsymbol{\alpha}) &= 1 - \pi_2(\boldsymbol{x}; \boldsymbol{\alpha}) \\ &= e^{a + \boldsymbol{b}'\boldsymbol{x}} / (1 + e^{a + \boldsymbol{b}'\boldsymbol{x}}), \end{aligned}$$

where $\boldsymbol{\alpha} = (a, \boldsymbol{b}')'$ is the parameter vector.

It can be seen from (3) that the hazard function corresponding to the ith component of the mixture has the form

$$h_i(t; w, \boldsymbol{\theta}_i) = e^{\gamma_i w} h_{i,0}(t; \boldsymbol{\theta}_i),$$

where the baseline hazard function $h_{i,0}(t; \boldsymbol{\theta}_i)$ is represented by the Gompertz model, for which

$$h_{i,0}(t; \boldsymbol{\theta}_i) = \exp(\lambda_i + \beta_i t) \qquad (i = 1, 2).$$

Hence the covariate w is modelled to have an additive effect on the ith component hazard function on the log scale. Often in practice, w will be the age of the patient at the instance from which the time to failure is being measured.

For the jth entity ($j = 1, ..., n$), we observe

$$\boldsymbol{y}_j = (t_j, \boldsymbol{x}_j', \delta_{1j}, \delta_{2j}, \delta_{3j})',$$

where δ_{1j}, δ_{2j} and δ_{3j} are zero-one indicator variables with $\delta_{ij} = 1$ if entity j failed due to cause i and zero otherwise ($i = 1, 2$), and $\delta_{3j} = 1$ if entity j was censored at time t_j; $\boldsymbol{x}_j$ is the covariate vector recorded on the jth entity ($j = 1, \ldots, n$). Thus

$$\delta_{1j} + \delta_{2j} + \delta_{3j} = 1$$

for $j = 1, ..., n$. As above, we let $\boldsymbol{\Psi} = (\boldsymbol{\alpha}', \boldsymbol{\theta}_1', \boldsymbol{\theta}_2')'$ be the vector containing all the unknown parameters. Then the log likelihood for $\boldsymbol{\Psi}$ formed on the basis of $\boldsymbol{y}_1, ..., \boldsymbol{y}_n$ is given by

$$\begin{aligned} \log L(\boldsymbol{\Psi}) \;=\; & \sum_{j=1}^{n} \Bigg[\sum_{i=1}^{2} \delta_{ij} \log\{\pi_i(\boldsymbol{x}_j; \boldsymbol{\alpha}) f_i(t_j; w_j, \boldsymbol{\theta}_i)\} \\ & + \delta_{3j} \log \sum_{i=1}^{2} \{\pi_i(\boldsymbol{x}_j; \boldsymbol{\alpha}) S_i(t_j; w_j, \boldsymbol{\theta}_i)\} \Bigg], \end{aligned}$$

where

$$S_i(t; w, \boldsymbol{\theta}_i) = \exp\{-\Lambda_i(t; w, \boldsymbol{\theta}_i)\}$$

is the ith component survival function.

The parameter vector $\boldsymbol{\Psi}$ is estimated by maximum likelihood. The likelihood equation

$$\partial \log L(\boldsymbol{\Psi}) / \partial \boldsymbol{\Psi} = \mathbf{0}$$

is solved via the EM algorithm of Dempster et al. (1977). For those entities with censored observations (that is, $\delta_{3j} = 1$), we introduce the zero-one indicator vector, $\boldsymbol{z}_j = (z_{1j}, z_{2j})'$, where $z_{ij} = 1$ or 0 according as entity j would have failed from cause i or not ($i = 1, 2$). The EM algorithm is applied with the $\boldsymbol{y}_j$ and the $\boldsymbol{z}_j$ viewed as the complete-data. The actual time to failure for those entities with $\delta_{3j} = 1$ was not

introduced as an incomplete variable in the complete-data framework, as it did not simplify the calculations.

It follows on application of the EM algorithm in the aforementioned framework that on the $(k+1)$th iteration, the estimate of $\boldsymbol{\Psi}$, $\boldsymbol{\Psi}^{(k+1)}$, is the maximizer of the function

$$Q(\boldsymbol{\Psi};\, \boldsymbol{\Psi}^{(k)}) = \sum_{j=1}^{n}\sum_{i=1}^{2}[\delta_{ij}\log\{\pi_i(\boldsymbol{x}_j;\boldsymbol{\alpha})f_i(t_j;w_j,\boldsymbol{\theta}_i)\} + \delta_{3j}\tau_i(t_j;\boldsymbol{x}_j,\, \boldsymbol{\Psi}^{(k)})\log\{\pi_i(\boldsymbol{x}_j;\, \boldsymbol{\alpha})S_i(t_j;w_j,\boldsymbol{\theta}_i)\}\Big], \quad (5)$$

where $\boldsymbol{\Psi}^{(k)}$ is the estimate of $\boldsymbol{\Psi}$ on the previous kth iteration and

$$\tau_i(t_j;\boldsymbol{x}_j,\, \boldsymbol{\Psi}) = \pi_i(\boldsymbol{x}_j;\boldsymbol{\alpha})S_i(t_j;w_j,\boldsymbol{\theta}_i)/\sum_{h=1}^{2}\pi_h(\boldsymbol{x}_j;\boldsymbol{\alpha})S_h(t_j;w_j,\boldsymbol{\theta}_h)$$

is the posterior probability that the jth entity with censored survival time t_j would have failed due to cause $i (i = 1, 2)$.

Unfortunately, for Gompertz component distributions, $\boldsymbol{\Psi}^{(k+1)}$ does not exist in closed form, as discussed in McLachlan and Krishnan (1997). In our algorithm, it is computed iteratively, commencing with $\boldsymbol{\alpha}^{(k+1)}$, since its calculation does not involve $\boldsymbol{\theta}_1^{(k+1)}$ and $\boldsymbol{\theta}_2^{(k+1)}$. The estimate $\boldsymbol{\alpha}^{(k+1)}$ is computed iteratively by Newton-Raphson. The computation of $\boldsymbol{\theta}_1^{(k+1)}$ and $\boldsymbol{\theta}_2^{(k+1)}$ is then undertaken iteratively using a quasi-Newton method. As noted by Dempster et al. (1977), it is not essential that $\boldsymbol{\Psi}^{(k+1)}$ globally maximizes $Q(\boldsymbol{\Psi};\, \boldsymbol{\Psi}^{(k)})$ for the monotonicity property of the sequence of iterates $\{\boldsymbol{\Psi}^{(k)}\}$ to hold. A sufficient condition is

$$Q(\boldsymbol{\Psi}^{(k+1)},\, \boldsymbol{\Psi}^{(k)}) \geq Q(\boldsymbol{\Psi}^{(k)},\, \boldsymbol{\Psi}^{(k)}). \quad (6)$$

The use of (6) as a means of choosing $\boldsymbol{\Psi}^{(k+1)}$ corresponds to using the generalized EM algorithm (Dempster et al. 1977).

In order to improve the speed of convergence, an E-step is performed in our program after the computation of $\boldsymbol{\alpha}^{(k+1)}$ and before the computation of the other subvectors $\boldsymbol{\theta}_1^{(k+1)}$ and $\boldsymbol{\theta}_2^{(k+1)}$ in $\boldsymbol{\Psi}^{(k+1)}$. In the terminology of Meng and Rubin (1993), it can be viewed as applying a multicycle ECM (expectation conditional-maximization) algorithm, where a cycle is defined to be an E-step followed by a CM-step. This multicycle E-step is effected here by updating $\boldsymbol{\alpha}^{(k)}$ by $\boldsymbol{\alpha}^{(k+1)}$ in $\boldsymbol{\Psi}^{(k)}$ in the right-hand side of (5). A multicycle ECM algorithm is not necessarily a GEM algorithm. However, it does increase the likelihood function at each cycle, and hence at each iteration (Meng and Rubin, 1993). To further enhance convergence in our program, another E-step is performed after the computation of $\boldsymbol{\alpha}^{(k+1,m+1)}$ for each m. This may affect the monotone convergence of the consequent sequence of likelihood values. In our particular applications the monotone convergence was preserved.

The standard errors of the maximum likelihood estimator $\hat{\boldsymbol{\Psi}}$ of $\boldsymbol{\Psi}$ is assessed using the bootstrap methodology of Efron (1979, 1982). A number K (as specified by

the user) of independent bootstrap samples are obtained with each being randomly drawn with replacement from the observed data $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$.

The program can also handle the case where there is only one cause of failure, but which may not happen for some entities; that is, there is a nonzero probability that the time to failure is infinity. In this case, $f_1(t_j; w_j, \boldsymbol{\theta}_1)$ and $S_1(t_j; w_j, \boldsymbol{\theta}_1)$ are each set identically equal to 1. The mixing proportion $\pi_1(\boldsymbol{x}_j; \boldsymbol{\alpha})$ now represents the prior probability that an entity with covariate vector $\boldsymbol{x}_j$ will not fail (that is, will have an infinite failure time).

ABOUT THE SOFTWARE

The software is written in FORTRAN77. To compile the algorithms, several steps are required to be carried out which are listed in the file "makefile.txt". The main algorithm is named MGOMP in which the subroutine CHOL (algorithm AS6) and SYMINV (algorithm AS7) are called to invert a positive definite symmetric matrix. To generate random numbers for the bootstrap application, the subroutine RANDOM (algorithm AS183) is called to return a pseudo-random integer taken from a uniform distribution between 0 and 1, excluding the end points. In addition, the MINPACK routine HYBRD1 (Moré et al.,1980) is called to find a solution of a system of two non-linear equations. Thus, there are five FORTRAN files named MGOMP.f, CHOL.f, SYMINV.f, RANDOM.f, and HYBRID.f in the package.

To run the software, two input files are required. The first file "para.dat" provides the initial estimates for the parameters $\lambda_i, \beta_i, \gamma_i\ (i = 1, 2), a$ and the coefficient b_1 of w. The other elements in the coefficient $\boldsymbol{b}$ of $\boldsymbol{x}$ are set initially to zero. The second file "surv.dat" contains the data $\boldsymbol{y}_j = (t_j, w_j, \delta_{1j}, \delta_{2j}, \boldsymbol{x}_j')'$ of each observation, where t_j, w_j, $\boldsymbol{x}_j$, and $\delta_{ij}\ (i = 1, 2)$ are defined as in the previous section.

For the sake of achieving satisfactory accuracy on the parameter estimates, double-precision is used in the algorithm. The stopping criterion for convergence is based on the absolute relative changes in the parameters, and a value of 10^{-4} is used for the cumulative (absolute) relative change. The final estimates of the parameters are given in the output file "fort.25".

The structure of the subroutine GETEST is given below. It defines the input and output variables in the algorithms.

SUBROUTINE GETEST(NBOOT,MODEL,PI2EQ1,SIZE,FTIME,AGE,DELTA,
NUMCOV,COV,ESTIN,ESTOUT,LLOUT,COEOUT,ITER,IFAIL,CODE)

NBOOT	Integer	input:	bootstrap iterations indicator: $= 1$ indicate the complete data is used; $\neq 1$ indicate the number of bootstrap iterations needed.
MODEL	Integer	input:	model indicator: $= 1$ if model with two distinct causes of failure; $= 2$ if only one cause of failure is considered.
PI2EQ1	Integer	input:	mixing proportion indicator: $= 0$ if fitting two components of Gompertz distributions; $= 1$ if fitting single Gompertz distribution ($\pi_{2j} = 1\ \forall j$).
SIZE	Integer	input:	the number of entities in the data set.
FTIME	Double precision array (SIZE)	input:	the failure (or censoring) time of each of the entities.
AGE	Double precision array (SIZE)	input:	the age of each of the entities at the time. of operation, that is w_j
DELTA	Integer array (3,SIZE)	input:	zero-one indicator: $d(1,j) = 1$ if entity j failed due to Cause 1; $d(2,j) = 1$ if entity j failed due to Cause 2; $d(3,j) = 1$ if entity j was censored.
NUMCOV	Integer	input:	the number of additional covariates besides AGE to be included in the logistic model.
COV	Double precision array (MAXCOV,SIZE)	input:	the entries $(l,j)(l = 1,\ldots,NUMCOV)$ give the values for the lth covariate of the jth entity, MAXCOV=20 is the maximum number of covariates.
ESTIN	Double precision array (8)	input:	the initial estimates for the parameters: λ_1, β_1, γ_1, λ_2, β_2, γ_2, a and b_1.
ESTOUT	Double precision array (MB,8)	output:	the final estimates for the parameters for each bootstrap iteration k $(k = 1,\ldots,NBOOT)$, MB=100 is the maximum number of bootstraps.
LLOUT	Double precision array (MB)	output:	the final estimates for each of calculated log-likelihoods for each bootstrap iteration.
COEOUT	Double precision array (MB,MAXCOV)	output:	the entries $(k,l)(k = 1,\ldots,NBOOT)$ give the final estimate of the coefficient for the lth extra covariate in the kth bootstrap.
ITER	Integer	output:	the number of iterations made in estimation.
IFAIL	Integer	output:	fault indicator: $= 0$ if no error is detected; $= 1$ if input or model-specification error is detected; $= 2$ if error is detected in estimation of a & b_1; $= 3$ if error is detected in estimation of $\boldsymbol{\theta}_1$ & $\boldsymbol{\theta}_2$.
CODE	Integer	output:	failure code for fault indicator IFAIL=2 or 3: $= 0$ if no error is detected; $= 1$ if error is detected in matrix inversion (for IFAIL=2); $= 2$ if suspected non-termination of a & b_1 (for IFAIL=2); = IFAULT (for IFAIL=3); see the description in the subroutine HYBRD1 of file "HYBRID.F".

EXAMPLE

As an illustration, the Stanford heart transplant data (Crowley and Hu, 1977) are analysed, using this algorithm to fit a Gompertz mixture model. These data have been analysed also by Larson and Dinse (1985) and Kuk (1992). We consider only the subset of 65 patients who received a transplant and who had complete data on the covariates of interest: mismatch score and age at transplant. Deaths were attributed to transplant rejection (Group 1) or other causes (Group 2). Among the 65 transplant recipients, there were 29 rejection deaths, 12 deaths from other causes, and 24 censored observations. Here the covariates are transformed to have zero means and unit variances. We include both covariates in the logistic model and the age at transplant in the component Gompertz distributions. The standard errors of the maximum likelihood estimates are obtained by bootstrap resampling with $K = 100$ replications. The results are presented in Table 1.

Table 1: Maximum likelihood estimates (standard errors) for two-component Gompertz mixture model

	Logistic model	Component parameters	
		Rejection death	Other causes
Covariate	MLE (S.E.)	MLE (S.E.)	MLE (S.E.)
Constant	1.396 (0.499)		
Mismatch score	0.358 (0.746)		
Age at transplant	0.303 (0.556)	1.023 (0.346)	0.275 (0.688)
λ		-6.335 (0.428)	-3.927 (0.742)
β		-0.0015 (0.0007)	-0.0055 (0.0124)

```
      PROGRAM MGOMP
c     ****************************************************************
c     **    Purpose: To fit a mixture of two Gompertz distributions    **
c     **               to censored survival data                       **
c     **Input files: (1) data file "surv.dat";                         **
c     **             (2) initial estimates file "para.dat"             **
c     **Output file: final estimates file "fort.25"                    **
c     ****************************************************************
      integer MB,MAX,MAXCOV
      parameter(MB=100)
      parameter(MAX=5000)
      parameter(MAXCOV=20)
      integer nboot,model,pi2eq1,size,delta(3,MAX),numcov,ifail
      integer iter,code
      double precision ftime(MAX),age(MAX),cov(MAXCOV,MAX),estin(8)
      double precision estout(MB,8),llout(MB),coeout(MB,MAXCOV)
c
      CALL init(nboot,model,pi2eq1,size,ftime,age,delta,numcov,
     +  cov,estin,ifail)
      if (ifail.ne.0) goto 200
      CALL getest(nboot,model,pi2eq1,size,ftime,age,delta,numcov,
     +  cov,estin,estout,llout,coeout,iter,ifail,code)
200   CALL output(nboot,size,numcov,estin,estout,llout,coeout,
     +  iter,ifail,code)
      stop
      end

      SUBROUTINE init(nboot,model,pi2eq1,size,ftime,age,delta,numcov,
     +  cov,estin,ifail)
c     ****************************************************************
c     ** To get data, initial estimates and model specification       **
c     ****************************************************************
      integer MB,MAX,MAXCOV
      parameter(MB=100)
      parameter(MAX=5000)
      parameter(MAXCOV=20)
      common/randc/ix,iy,iz
      integer ix,iy,iz
      integer nboot,model,pi2eq1,size,delta(3,MAX),totcov,ifail
      double precision ftime(MAX),age(MAX),cov(MAXCOV,MAX),estin(8)
      double precision tempc(MAXCOV,MAX)
      character*1 userin
c
      open(21,file='surv.dat')
      open(22,file='para.dat')
      ifail=1
      print *,'Please select an option: [c] for continue '
      print *,'                         [q] for quit  '
      read *,userin
      if (userin(1:1).eq.'q') goto 900
      print *,' '
```

```
      print *,'The program fits the model either to the complete ',
     +  'data, or with optional'
      print *,'bootstrapping.  Select 1 to use the complete data, ',
     +  'or type the number of'
      print *,'bootstrap repetitions which you require.  Your ',
     +  'choice?'
      read *, nboot
      if (nboot.gt.MB) then
        print *,'The maximum number of bootstraps allowed is ',MB
        goto 900
      endif
c     input random seeds for SUBROUTINE random()
      if (nboot.gt.1) then
        print *,' '
        print *,'Input the seeds for random number generation: ',
     +    'input 3 integers between 1 and 30,000, e.g. 1 328 70'
        read *, ix, iy, iz
      endif
      print *,' '
      print *,'Which model is chosen?  Type 1 or 2 '
      read *, model
      print *,' '
      print *,'Type 1 for PI_2=1.0, 0 otherwise. Your choice? '
      read *, pi2eq1
      print *,' '
      print *,'How many observations are there? '
      read *, size
      if (size.gt.MAX) then
        print *,'The maximum number of observations allowed is ',MAX
        goto 900
      endif
      print *,' '
      print *,'Please input the total number of additional covariates',
     +  ' (other than w) in'
      print *,'the survival data file.  If you do not have any',
     +  ' additional covariates'
      print *,'in the data file, input 0. (At the next step, you may',
     +  ' choose some (or all)'
      print *,'of them to be included in the logistic model)'
      read *, totcov
      if (totcov.gt.MAXCOV) then
        print *,'The maximum number of covariates allowed is ',MAXCOV
        goto 900
      endif
      do 40 j=1,size
        read (21,*) ftime(j),age(j),delta(1,j),delta(2,j),
     +    (tempc(k,j),k=1,totcov)
        delta(3,j)=1-delta(1,j)-delta(2,j)
40    continue
      if (totcov.ne.0) CALL getcov(size,totcov,tempc,numcov,cov)
      print *,' '
      print *,'RUNNING PROGRAM.  PLEASE WAIT!'
      read (22,*) (estin(i),i=1,8)
      if (pi2eq1.eq.1) then
```

```
        estin(7)=0.0
        estin(8)=0.0
      endif
      ifail=0
900   return
      end

      SUBROUTINE getcov(size,totcov,tempc,numcov,cov)
c     ****************************************************************
c     ** To include additional covariates in the logistic model      **
c     ****************************************************************
      parameter(MAX=5000)
      parameter(MAXCOV=20)
      integer size,totcov,numcov,addcov(MAXCOV)
      double precision tempc(MAXCOV,MAX),cov(MAXCOV,MAX)
c
      numcov=0
      do 10 l=1,MAXCOV
        addcov(l)=0
10    continue
      print *,' '
      print *,'Your data set has ',totcov, ' additional covariates.',
     +  '  Type the covariate'
      print *,'number to be included in the logistic model.  Type 0',
     +  ' to stop, or a number'
      print *,'in the range of 1 to ', totcov
      read *, k
      if (k.eq.0) then
        print *,'No additional covariate is included!'
        goto 900
      endif
      l=1
c     WHILE (k.ne.0) DO
100   if (k.ne.0) then
        numcov=numcov+1
        addcov(l)=k
        l=l+1
        print *,' '
        print *,'What is the next covariate?  Type 0 to stop, or a '
        print *,'number in the range of 1 to ', totcov
        read *, k
        goto 100
      endif
c     ENDWHILE
      print *,' '
      print *,'Include covariate numbers: '
      print *,'       ',(addcov(l),l=1,numcov)
c     user specifies additional covariates
      do 300 j=1,size
        do 200 l=1,numcov
          cov(l,j)=tempc(addcov(l),j)
200     continue
300   continue
900   return
```

```
      end

      SUBROUTINE getest(nboot,model,pi2eq1,size,ftime,age,delta,
     +  numcov,cov,estin,estout,llout,coeout,iter,ifail,code)
c     ****************************************************************
c     ** To obtain the bootstrap samples and get the final estimates   **
c     ****************************************************************
      integer MB,MAX,MAXCOV
      parameter(MB=100)
      parameter(MAX=5000)
      parameter(MAXCOV=20)
      integer nboot,model,pi2eq1,size,delta(3,MAX),bdelta(3,MAX)
      integer numcov,iter,ifail,code
      double precision ftime(MAX),age(MAX),cov(MAXCOV,MAX),estin(8)
      double precision bftime(MAX),bage(MAX),bcov(MAXCOV,MAX)
      double precision estout(MB,8),llout(MB),coeout(MB,MAXCOV)
c
      if (nboot.eq.1) then
c       for complete data (nboot.eq.1)
        ifail=2
        CALL mle(nboot,model,pi2eq1,size,ftime,age,delta,numcov,cov,
     +  estin,1,estout,llout,coeout,iter,ifail,code)
      else
c       for (nboot.gt.1)
        do 500 k=1,nboot
          ifail=2
c         WHILE (ifail.ne.0) DO
300       if (ifail.ne.0) then
            CALL boot(size,ftime,age,delta,numcov,cov,bftime,
     +      bage,bdelta,bcov)
            CALL mle(nboot,model,pi2eq1,size,bftime,bage,bdelta,
     +      numcov,bcov,estin,k,estout,llout,coeout,iter,ifail,code)
            goto 300
          endif
c         ENDWHILE
500     continue
      endif
      return
      end

      SUBROUTINE mle(nboot,model,pi2eq1,size,tj,xj,dj,numcov,cj,
     +  estin,bnum,estout,llout,coeout,iter,ifail,code)
c     ****************************************************************
c     ** To obtain the maximum likelihood estimates via EM algorithm   **
c     ****************************************************************
      integer MB,MAX,MAXCOV
      parameter(MB=100)
      parameter(MAX=5000)
      parameter(MAXCOV=20)
      integer nboot,model,pi2eq1,size,dj(3,MAX),numcov,bnum
      integer iter,ifail,code,done
      double precision tj(MAX),xj(MAX),cj(MAXCOV,MAX),estin(8)
      double precision estout(MB,8),llout(MB),coeout(MB,MAXCOV)
      double precision est(8),newest(8),pi(2,MAX),s(2,MAX),tau(2,MAX)
```

```
      double precision coe(MAXCOV),ll,ddiff
      intrinsic abs
c
      done=0
      code=0
      iter=0
      do 10 i=1,8
        est(i)=estin(i)
        newest(i)=est(i)
10    continue
      do 12 l=1,numcov
        coe(l)=0.0
12    continue
c     WHILE (done.ne.1) AND (code.eq.0) DO
15    if (done.ne.1.and.code.eq.0) then
        CALL setpi(size,pi2eq1,numcov,xj,est(7),est(8),coe,cj,pi)
        CALL calcs(model,size,tj,xj,est,s)
c       for mixture model of two components
        if (pi2eq1.eq.0) then
          CALL calcab(size,numcov,est,pi,xj,dj,cj,coe,s,
     +      newest(7),newest(8),tau,ifail,code)
        else
          CALL caltau(size,pi,s,tau)
          ifail=0
        endif
c       ifail.ne.0 implies code.ne.0
        if (ifail.ne.0) goto 30
        ifail=3
        CALL callhy(model,size,tj,xj,dj,est,newest,tau,ifail,code)
c       ifail.ne.0 implies code.ne.0
        if (ifail.ne.0) goto 30
        CALL calcl(model,size,tj,xj,dj,tau,newest(2),newest(3),1,
     +  newest(1))
        CALL calcl(model,size,tj,xj,dj,tau,newest(5),newest(6),2,
     +  newest(4))
        ddiff=0.0
        do 20 i=1,8
          if (newest(i).ne.0.0) then
            ddiff=ddiff+abs((est(i)-newest(i))/newest(i))
          endif
          est(i)=newest(i)
20      continue
        if (nboot.eq.1) CALL findll(model,pi2eq1,size,tj,xj,
     +    dj,est,pi,ll)
        if (ddiff.lt.0.0001) done=1
        iter=iter+1
30      continue
        goto 15
      endif
c     ENDWHILE
      if (code.ne.0) goto 900
      llout(bnum)=ll
      do 100 i=1,8
        estout(bnum,i)=est(i)
```

```
100   continue
      do 200 l=1,numcov
        coeout(bnum,l)=coe(l)
200   continue
900   return
      end

      SUBROUTINE boot(size,ftime,age,delta,numcov,cov,bftime,
     +  bage,bdelta,bcov)
c     ****************************************************************
c     ** To obtain a nonparametric bootstrap sample                  **
c     ****************************************************************
      integer MAX,MAXCOV
      parameter(MAX=5000)
      parameter(MAXCOV=20)
      integer size,delta(3,MAX),numcov,bdelta(3,MAX),num
      real rand,random
      double precision ftime(MAX),age(MAX),cov(MAXCOV,MAX)
      double precision bftime(MAX),bage(MAX),bcov(MAXCOV,MAX)
      intrinsic int
c
      do 150 j=1,size
        rand=random()
        rand=1.+rand*float(size)
        num=int(rand)
        bftime(j)=ftime(num)
        bage(j)=age(num)
        bdelta(1,j)=delta(1,num)
        bdelta(2,j)=delta(2,num)
        bdelta(3,j)=delta(3,num)
        do 120 l=1,numcov
          bcov(l,j)=cov(l,num)
120     continue
150   continue
      return
      end

      SUBROUTINE setpi(size,pi2eq1,numcov,xj,a,b,coe,cj,pi)
c     ****************************************************************
c     ** To evaluate the mixing proportion \pi_i (i=1,2)             **
c     ****************************************************************
      integer MAX,MAXCOV
      parameter(MAX=5000)
      parameter(MAXCOV=20)
      integer size,pi2eq1,numcov
      double precision xj(MAX),a,b,coe(MAXCOV),cj(MAXCOV,MAX)
      double precision pi(2,MAX),tmp
      intrinsic exp
c
      do 100 j=1,size
        if (pi2eq1.eq.1) then
          pi(1,j)=0.0
        else
          tmp=a+b*xj(j)
```

```
          do 50 l=1,numcov
            tmp=tmp+coe(l)*cj(l,j)
50        continue
          pi(1,j)=exp(tmp)/(1.0+exp(tmp))
        endif
        pi(2,j)=1.0-pi(1,j)
100   continue
      return
      end

      SUBROUTINE calcs(model,size,tj,xj,est,s)
c     ****************************************************************
c     ** To evaluate the component survival functions s_i (i=1,2)      **
c     ****************************************************************
      integer MAX
      parameter(MAX=5000)
      integer model,size,group,tmp
      double precision tj(MAX),xj(MAX),est(8),s(2,MAX)
      double precision lambda,beta,gamma,t1,t2,udrfl
      intrinsic exp
c
      udrfl=-75.
      do 100 group=1,2
        tmp=(group-1)*3+1
        lambda=est(tmp)
        beta=est(tmp+1)
        gamma=est(tmp+2)
        do 50 j=1,size
          if (model.eq.2.and.group.eq.1) then
            s(group,j)=1.0
          else
            t1=exp(lambda+gamma*xj(j))
            t2=exp(beta*tj(j))-1.0
            if ((-t1*t2/beta).lt.udrfl) then
              s(group,j)=0.
              goto 50
            endif
            s(group,j)=exp(-t1*t2/beta)
          endif
50      continue
100   continue
      return
      end

      SUBROUTINE calcab(size,numcov,est,pi,xj,dj,cj,coe,s,anew,bnew,
     +  tau,ifail,code)
c     ****************************************************************
c     ** To perform iterative steps for the computation of the         **
c     ** coefficients \alpha of the covariates                         **
c     ****************************************************************
      integer MAX,MAXCOV
      parameter(MAX=5000)
      parameter(MAXCOV=20)
      integer size,dj(3,MAX),numcov,ifail,code,ctr,order,done
```

```
      double precision est(8),pi(2,MAX),xj(MAX),cj(MAXCOV,MAX)
      double precision coe(MAXCOV),s(2,MAX),anew,bnew,tau(2,MAX)
      double precision a,b,newcoe(MAXCOV),ddiff
      double precision arr(MAXCOV+2,MAXCOV+2),u(MAXCOV+2)
      intrinsic abs
c
      ctr=0
      code=0
      order=numcov+2
      done=0
      a=est(7)
      b=est(8)
c     WHILE (done.ne.1) AND (code.eq.0) DO
10    if (done.ne.1.and.code.eq.0) then
        ddiff=0.
        CALL caltau(size,pi,s,tau)
        CALL setarr(size,a,b,pi,xj,dj,cj,coe,numcov,tau,order,arr,u)
        CALL invert(arr,order,code)
        if (code.ne.0) goto 100
        anew=a
        bnew=b
        do 20 i=1,order
          anew=anew+arr(1,i)*u(i)
          bnew=bnew+arr(2,i)*u(i)
20      continue
        do 30 l=1,numcov
          newcoe(l)=coe(l)
30      continue
        do 50 i=1,order
          do 40 l=1,numcov
            newcoe(l)=newcoe(l)+arr(l+2,i)*u(i)
40        continue
50      continue
        ddiff=abs(a-anew)+abs(b-bnew)
        do 60 l=1,numcov
          ddiff=ddiff+abs(newcoe(l)-coe(l))
          coe(l)=newcoe(l)
60      continue
        CALL setpi(size,pi2eq1,numcov,xj,anew,bnew,newcoe,cj,pi)
        a=anew
        b=bnew
        if (ddiff.lt.0.0001) done=1
        ctr=ctr+1
        if (ctr.gt.800 .or. ddiff.gt.15.0) code=2
100     continue
        goto 10
      endif
c     ENDWHILE
      if (code.eq.0) then
        ifail=0
        CALL caltau(size,pi,s,tau)
      endif
      return
      end
```

```
      SUBROUTINE caltau(size,pi,s,tau)
c     ****************************************************************
c     ** To evaluate the posterior probability \tau, using           **
c     ** pre-calculated values for \pi_i and s_i (i=1,2)             **
c     ****************************************************************
      integer MAX
      parameter(MAX=5000)
      integer size
      double precision pi(2,MAX),s(2,MAX),tau(2,MAX),temp
c
      do 100 j=1,size
        temp=pi(1,j)*s(1,j)
        tau(1,j)=temp/(temp+pi(2,j)*s(2,j))
        tau(2,j)=1.0-tau(1,j)
100   continue
      return
      end

      SUBROUTINE setarr(size,a,b,pi,xj,dj,cj,coe,numcov,tau,order,arr,u)
c     ****************************************************************
c     ** To evaluate the matrix arr and the colume vector u          **
c     ****************************************************************
      integer MAX,MAXCOV
      parameter(MAX=5000)
      parameter(MAXCOV=20)
      integer size,dj(3,MAX),numcov,order
      double precision a,b,pi(2,MAX),xj(MAX),cj(MAXCOV,MAX)
      double precision coe(MAXCOV),tau(2,MAX)
      double precision arr(MAXCOV+2,MAXCOV+2),u(MAXCOV+2)
      double precision varr(MAX),xarr(MAXCOV+2,MAX),tmp,yj,t
c
      do 20 i=1,order
        u(i)=0.0
        do 10 k=1,order
          arr(i,k)=0.0
10      continue
20    continue
      do 40 j=1,size
        varr(j)=pi(1,j)*pi(2,j)
        xarr(1,j)=1.0
        xarr(2,j)=xj(j)
        do 30 l=1,numcov
          xarr(l+2,j)=cj(l,j)
30      continue
40    continue
      do 70 i=1,order
        do 60 k=1,order
          tmp=0.0
          do 50 j=1,size
            tmp=tmp+xarr(i,j)*xarr(k,j)*varr(j)
50        continue
          arr(i,k)=tmp
60      continue
```

```
70    continue
      do 100 j=1,size
        yj=float(dj(1,j))+dj(3,j)*tau(1,j)
        tmp=a+b*xarr(2,j)
        do 80 l=1,numcov
          tmp=tmp+coe(l)*cj(l,j)
80      continue
        t=exp(tmp)
        u(1)=u(1)+yj-t/(1.0+t)
        do 90 k=2,order
          u(k)=u(k)+yj*xarr(k,j)-xarr(k,j)*t/(1.0+t)
90      continue
100   continue
      return
      end

      SUBROUTINE invert(arr,order,code)
c     ******************************************************************
c     ** To return the inverse of the input matrix arr in the same     **
c     ** matrix (so arr is over-written)                               **
c     ******************************************************************
      integer MAXCOV
      parameter(MAXCOV=20)
      integer order,code,ctr,nullty,ifault
      real matrix((MAXCOV+2)*(MAXCOV+3))
      real inv((MAXCOV+2)*(MAXCOV+3)),work(MAXCOV+2)
      double precision arr(MAXCOV+2,MAXCOV+2)
c
      ctr=0
      do 20 i=1,order
        do 10 k=1,i
          ctr=ctr+1
          matrix(ctr)=arr(i,k)
10      continue
20    continue
      nullty=0
      ifault=0
      CALL syminv(matrix,order,ctr,inv,work,nullty,ifault)
      if (ifault.ne.0) then
        code=1
        goto 900
      endif
      ctr=0
      do 40 i=1,order
        do 30 k=1,i
          ctr=ctr+1
          arr(i,k)=inv(ctr)
          arr(k,i)=inv(ctr)
30      continue
40    continue
900   return
      end

      SUBROUTINE callhy(model,size,tj,xj,dj,est,newest,tau,ifail,code)
```

```
c     ****************************************************************
c     ** To call HYBRD1 for solving system of nonlinear equations      **
c     ****************************************************************
      integer MAX
      parameter(MAX=5000)
      integer model,size,dj(3,MAX),ifail,code,group
      double precision tj(MAX),xj(MAX),est(8),newest(8),tau(2,MAX)
      external evalfn, HYBRD1
      integer sizehy,n,indlo,tmp
      double precision beta,gamma,x(2),xtol,fvec(2),work(100)
      common/hybr1/hmodel,hsize,hdj,hgroup
      integer hmodel,hsize,hdj(3,MAX),hgroup
      common/hybr2/htj,hxj,htau
      double precision htj(MAX),hxj(MAX),htau(2,MAX)
c
      code=0
      indlo=1
      if (model.eq.2) indlo=2
c     set values for the common statement
      hmodel=model
      hsize=size
      do 10 j=1,hsize
        htj(j)=tj(j)
        hxj(j)=xj(j)
        htau(1,j)=tau(1,j)
        htau(2,j)=tau(2,j)
        hdj(1,j)=dj(1,j)
        hdj(2,j)=dj(2,j)
        hdj(3,j)=dj(3,j)
10    continue
      do 20 group=indlo,2
        if (code.ne.0) goto 20
        hgroup=group
        tmp=(group-1)*3+1
        beta=est(tmp+1)
        gamma=est(tmp+2)
        ifault=0
        n=2
        x(1)=beta
        x(2)=gamma
        xtol=0.000000001
        sizehy=100
        CALL HYBRD1(evalfn,n,x,fvec,xtol,ifault,work,sizehy)
c       ifault=1 implies no error is detected (see subroutine HYBRD1 in
c       file "hybrid.f"); change code=1 for ifault=0
        if (ifault.ne.1) then
          code=1
          if (ifault.ne.0) code=ifault
          goto 20
        endif
        newest(tmp+1)=x(1)
        newest(tmp+2)=x(2)
20    continue
      if (code.eq.0) ifail=0
```

```
      return
      end

      SUBROUTINE evalfn(n,x,fvec,iflag)
c     ******************************************************************
c     ** To evaluate the system of functions                           **
c     ******************************************************************
      integer MAX
      parameter(MAX=5000)
      integer n,iflag
      double precision x(*),fvec(*)
      double precision t1,t2,t3,dc,tc,hval
      double precision tot1,tot2,beta,gamma,lambda
      common/hybr1/hmodel,hsize,hdj,hgroup
      integer hmodel,hsize,hdj(3,MAX),hgroup
      common/hybr2/htj,hxj,htau
      double precision htj(MAX),hxj(MAX),htau(2,MAX)
      intrinsic exp
c
      beta=x(1)
      gamma=x(2)
      CALL calcl(hmodel,hsize,htj,hxj,hdj,htau,beta,gamma,
     +  hgroup,lambda)
      tot1=0.0
      tot2=0.0
      do 100 j=1,hsize
        dc=hdj(hgroup,j)
        tc=htau(hgroup,j)
        t1=exp(lambda+gamma*hxj(j))
        t2=exp(beta*htj(j))-1.0
        hval=(t1*t2)
        t1=(dc+hdj(3,j)*tc)
        t2= -htj(j)/beta*exp(lambda+gamma*hxj(j)+beta*htj(j))
        t3=hval/(beta*beta)
        tot1=tot1+dc*htj(j)+t1*(t2+t3)
        t1=dc*hxj(j)
        t2= -(dc+hdj(3,j)*tc)
        t3=hxj(j)*hval/beta
        tot2=tot2+t1+t2*t3
100   continue
      fvec(1)=tot1
      fvec(2)=tot2
      return
      end

      SUBROUTINE calcl(model,size,tj,xj,dj,tau,beta,gamma,ind,lambda)
c     ******************************************************************
c     ** To solve for lambda given current estimates of beta & gamma   **
c     ******************************************************************
      integer MAX
      parameter(MAX=5000)
      integer model,size,dj(3,MAX),ind
      double precision tj(MAX),xj(MAX),tau(2,MAX),beta,gamma,lambda
      double precision t,t1,t2
```

```
      intrinsic log
c
      lambda=0.0
      if (ind.eq.1.and.model.eq.2) then
        beta=0.0
        gamma=0.0
        goto 900
      endif
      t1=0.0
      t2=0.0
      do 10 j=1,size
        t1=t1+float(dj(ind,j))
        t=exp(gamma*xj(j))*(exp(beta*tj(j))-1.0)
        t2=t2+(float(dj(ind,j))+dj(3,j)*tau(ind,j))*t
10    continue
      lambda=log(beta*t1/t2)
900   return
      end

      SUBROUTINE findll(model,pi2eq1,size,tj,xj,dj,est,pi,ll)
c     ********************************************************************
c     ** To evaluate the maximum log likelihood                         **
c     ********************************************************************
      integer MAX
      parameter(MAX=5000)
      integer model,pi2eq1,size,dj(3,MAX)
      double precision tj(MAX),xj(MAX),est(8),pi(2,MAX),ll
      double precision t1,t2,t3,te,tf,mul,udrfl
      double precision l1,g1,b1,l2,g2,b2,p1,p2
      intrinsic log
c
      udrfl=-75.
      t1=0.0
      t2=0.0
      t3=0.0
      l1=est(1)
      b1=est(2)
      g1=est(3)
      l2=est(4)
      b2=est(5)
      g2=est(6)
      do 20 j=1,size
        p1=pi(1,j)
        p2=pi(2,j)
        te=exp(l2+g2*xj(j))*(exp(b2*tj(j))-1)/b2
        if (pi2eq1.eq.1) then
          t1=t1+dj(2,j)*log(p2)
        else
          t1=t1+dj(1,j)*log(p1)+dj(2,j)*log(p2)
        endif
        t2=t2+dj(2,j)*(l2+g2*xj(j)+b2*tj(j)-te)
        if (model.eq.2) then
          mul=1.0
        else
```

```
          tf=exp(l1+g1*xj(j))*(exp(b1*tj(j))-1)/b1
          t2=t2+dj(1,j)*(l1+g1*xj(j)+b1*tj(j)-tf)
          if (-tf.lt.udrfl) then
            mul=0.
          else
            mul=exp(-tf)
          endif
        endif
        if (-te.lt.udrfl) then
          t3=t3+dj(3,j)*log(p1*mul)
        else
          t3=t3+dj(3,j)*log(p1*mul+p2*exp(-te))
        endif
20    continue
      ll=t1+t2+t3
      return
      end

      SUBROUTINE output(nboot,size,numcov,estin,estout,llout,
     +  coeout,iter,ifail,code)
c     ****************************************************************
c     ** To write the result into output file "fort.25"              **
c     ****************************************************************
      integer MB,MAXCOV
      parameter(MB=100)
      parameter(MAXCOV=20)
      integer nboot,size,numcov,iter,ifail,code
      double precision estin(8),estout(MB,8),llout(MB)
      double precision coeout(MB,MAXCOV),sum(8+MAXCOV),bar(8+MAXCOV)
      double precision se(8+MAXCOV),sumsq(8+MAXCOV)
c
      write (25,900) (estin(i),i=1,3)
      write (25,905) (estin(i),i=4,6)
      write (25,910) (estin(i),i=7,8)
      write (25,915) size
      if (ifail.ne.0) then
        print *,'The program is terminated: ifail= ',ifail
        write (25,920) ifail
        if (ifail.eq.2.and.code.eq.1) write (25,922)
        if (ifail.eq.2.and.code.eq.2) write (25,925)
        if (ifail.eq.3) write (25,927) code
      else
        print *,' '
        print *,'FINISH.  THE RESULT IS AT THE OUTPUT FILE "fort.25".'
        do 100 l=1,8+numcov
          sum(l)=0.
          sumsq(l)=0.
100     continue
        write (25,930)
        do 500 k=1,nboot
          if (nboot.eq.1) then
            write (25,935) k,iter,llout(k)
            write (25,940) (estout(k,l),l=1,3)
            write (25,945) (estout(k,l),l=4,6)
```

```
            write (25,950) (estout(k,l),l=7,8)
            if (numcov.ne.0) write (25,955) (coeout(k,l),l=1,numcov)
            goto 500
          endif
c          if (nboot.eq.1) goto 500
          do 200 l=1,8
            sum(l)=sum(l)+estout(k,l)
            sumsq(l)=sumsq(l)+estout(k,l)*estout(k,l)
200       continue
          do 300 l=1,numcov
            sum(8+l)=sum(8+l)+coeout(k,l)
            sumsq(8+l)=sumsq(8+l)+coeout(k,l)*coeout(k,l)
300       continue
500     continue
        if (nboot.eq.1) goto 650
        do 550 l=1,8+numcov
          bar(l)=sum(l)/float(nboot)
          se(l)=sqrt(sumsq(l)/float(nboot)-bar(l)*bar(l))
550     continue
        do 600 l=1,8+numcov
          if (l.le.8) then
            write (25,960) l,bar(l),se(l)*float(nboot)/float(nboot-1)
          else
            write (25,965) l-8,bar(l),
     +      se(l)*float(nboot)/float(nboot-1)
          endif
600     continue
650    endif
900    format('Initial Estimates:',/2x,'lambda1=',F9.5,'   beta1=',
     +  F9.5,'   gamma1=',F9.5)
905    format(2x,'lambda2=',F9.5,'   beta2=',
     +  F9.5,'   gamma2=',F9.5)
910    format(8x,'a=',F9.5,'     b_1=',F9.5)
915    format(/,'There are ',I6,' values in the data set',/)
920    format(/,'The program is terminated: ifail= ',I3)
922    format(/,'The inverion of matrix does not make good!')
925    format(/,'Suspected non-termination of a and b_1!')
927    format(/,'The failure code in HYBRD1 routine is ',I3)
930    format(/,'-------------------------------------------',
     +  /,'Summary of results:')
935    format(/,'Bootstrap ',I3,': after iteration ',I3,
     +  ', Log Likelihood= ',F15.8)
940    format('  Estimates:',/2x,'lambda1=',F11.7,'    beta1=',
     +  F11.7,'    gamma1=',F11.7)
945    format(2x,'lambda2=',F11.7,'    beta2=',
     +  F11.7,'    gamma2=',F11.7)
950    format(8x,'a=',F11.7,'      b_1=',F11.7)
955    format(2x,'Coefficients for additional covariates are: ',
     +  /,20F9.5)
960    format('mean (s.e) of estimate ',I2,': ',F11.7,' (',F9.7,')')
965    format('mean (s.e) of additional covariate ',I2,': ',F11.7,
     +  ' (',F9.7,')')
       return
       end
```

BIBLIOGRAPHY

Crowley, J. and Hu, M. (1977). Covariance analysis of heart transplant survival data. *Journal of the American Statistical Associations* **72**, 27–36.

Dempster, A.P., Laird, N.M. and Rubin, D.B. (1977). Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society B* **39**, 1–38.

Efron, B. (1979). Bootstrap methods: another look at the jackknife. *Annals of Statistics* **7**, 1–26.

Efron, B. (1982). *The Jackknife, the Bootstrap and Other Resampling Plans*. Philadelphia: SIAM.

Farewell, V.T. (1982). The use of mixture models for the analysis of survival data with long-term survivors. *Biometrics* **38**, 1041–1046.

Gordon, N.H. (1990). Maximum likelihood estimation for mixtures of two Gompertz distributions when censoring occurs. *Communications in Statistics - Simulation and Computation* **19**, 733–747.

Gordon, N.H. (1990). Application of the theory of finite mixtures for the estimation of 'cure' rates of treated cancer patients. *Statistics in Medicine* **9**, 397–407.

Kuk, Y.C. (1992). A semiparametric mixture model for the analysis of competing risks data. *Australian Journal of Statistics* **34**, 169–180.

Larson, M.G. and Dinse, G.E. (1985). A mixture model for the regression analysis of competing risks data. *Applied Statistics* **34**, 201–211.

McLachlan, G.J. and Krishnan, T. (1997). *The EM algorithm and Extensions*. New York: Wiley.

Meng, X.L and Rubin, D.R. (1993). Maximum likelihood estimation via the ECM algorithm: a general framework. *Biometrika* **80**, 267–278.

Moré, J.J., Garbow, B.S. and Hillstrom, K.E. (1980). *User Guide for MINPACK-1*, ANL-80-74. Argonne National Laboratory, Chicago.