Example A2 (Klein & Moeschberger, 2003; p.447)

Consider a two-parameter Weibull model. The log-likelihood function is given by

$$L(\lambda, \alpha) = n\log\lambda + n\log\alpha + (\alpha - 1)\sum_i \log t_i - \lambda\sum_i t_i^{\alpha} .$$

Based on ten data, we want to find the maximum likelihood estimator (MLE). We present the method of steepest ascent to find the MLE of (λ, α). The following is the algorithm.

Steepest ascent algorithm:

Step1.initial value (λ^0, α^0)

Step2.Updated: $\begin{bmatrix} \lambda^{k+1} \\ \alpha^{k+1} \end{bmatrix} = \begin{bmatrix} \lambda^{k} \\ \alpha^{k} \end{bmatrix} + d^k \begin{bmatrix} u_\lambda(\lambda, \alpha) \\ u_\alpha(\lambda, \alpha) \end{bmatrix}\Bigg|_{(\lambda,\alpha)=(\lambda^k,\alpha^k)}$,

where $u_\lambda(\lambda, \alpha) = \dfrac{\partial L(\lambda, \alpha)}{\partial\lambda}, u_\alpha(\lambda, \alpha) = \dfrac{\partial L(\lambda, \alpha)}{\partial\alpha}$ and

$$d^k = \arg\max_d L\{\lambda^k + du_\lambda(\lambda^k, \alpha^k), \alpha^k + du_\alpha(\lambda^k, \alpha^k)\}$$

Step3. If $|u_\lambda(\lambda, \alpha)| < \varepsilon$ and $|u_\alpha(\lambda, \alpha)| < \varepsilon$ then stop.

In the algorithm, I want to compute d^k by one-dimensional Newton-Raphson. First, we should calculate the first-order and second-order of the function

$L\{\lambda^k + du_\lambda(\lambda^k, \alpha^k), \alpha^k + du_\alpha(\lambda^k, \alpha^k)\}$ respect to d. Let

$$\begin{aligned} g(d) &= L\{\lambda^k + du_\lambda(\lambda^k, \alpha^k), \alpha^k + du_\alpha(\lambda^k, \alpha^k)\} \\ &= n\log\{\lambda^k + du_\lambda(\lambda^k, \alpha^k)\} + n\log\{\alpha^k + du_\alpha(\lambda^k, \alpha^k)\} \\ &\quad + \{\alpha^k + du_\alpha(\lambda^k, \alpha^k) - 1\}\sum_i \log t_i - \{\lambda^k + du_\lambda(\lambda^k, \alpha^k)\}\sum_i t_i^{\alpha^k + du_\alpha(\lambda^k, \alpha^k)} \end{aligned}$$

First-order is

$$\begin{aligned} g'(d) = \frac{\partial g(d)}{\partial d} &= \frac{nu_\lambda(\lambda^k, \alpha^k)}{\lambda^k + du_\lambda(\lambda^k, \alpha^k)} + \frac{nu_\alpha(\lambda^k, \alpha^k)}{\alpha^k + du_\alpha(\lambda^k, \alpha^k)} + u_\alpha(\lambda^k, \alpha^k)\sum_i \log t_i \\ &\quad - u_\lambda(\lambda^k, \alpha^k)\sum_i t_i^{\alpha^k + du_\alpha(\lambda^k, \alpha^k)} - \{\lambda^k + du_\lambda(\lambda^k, \alpha^k)\}\sum_i \{t_i^{\alpha^k + du_\alpha(\lambda^k, \alpha^k)} u_\alpha(\lambda^k, \alpha^k)\log t_i\} \end{aligned}$$

Second-order is

$$
\begin{aligned}
g''(d)\frac{\partial^2 g(d)}{\partial d^2} = &-\frac{n u_\lambda(\lambda^k,\alpha^k)^2}{\{\lambda^k + d u_\lambda(\lambda^k,\alpha^k)\}^2} - \frac{n u_\alpha(\lambda^k,\alpha^k)^2}{\{\alpha^k + d u_\alpha(\lambda^k,\alpha^k)\}^2} + \\
&-2u_\lambda(\lambda^k,\alpha^k)\sum_i \{ t_i^{\alpha^k + d u_\alpha(\lambda^k,\alpha^k)} u_\alpha(\lambda^k,\alpha^k)\log t_i \} \\
&-\{\lambda^k + d u_\lambda(\lambda^k,\alpha^k)\}\sum_i \{ t_i^{\alpha^k + d u_\alpha(\lambda^k,\alpha^k)} u_\alpha(\lambda^k,\alpha^k)^2(\log t_i)^2 \}
\end{aligned}
$$

Then use the one-dimensional Newton-Raphson to find d^k. The algorithm:

Step1.initial value d_0

Step2.Updated: $d_{i+1} = d_i - \dfrac{g'(d_i)}{g''(d_i)}$

Step3. If $|g'(d_i)| < \varepsilon$ then stop.

Because I don't know how to select initial of d, the following rule is no special reason. For k =0, I select the initial $d_0 = 1$.For k >0, I select the initial $d_0 = d^k$.

Result is given in table1.

Table 1 Set $\varepsilon = 10^{-1}$

Step k	λ^k	α^k	$L(\lambda^k,\alpha^k)$	$u_\lambda(\lambda^k,\alpha^k)$	$u_\alpha(\lambda^k,\alpha^k)$	d^k
0	1.024590	1.00000	-9.757073	1.776357×10^{-15}	7.0338193273	0.0985955
1	1.024590	1.69350	-7.491127	-1.802714	0.0005888786	0.0888365
2	0.864443	1.69355	-7.338982	0.005208932	0.6627223309	0.1063588
3	0.864997	1.76404	-7.312165	-0.3069025	0.1008135244	0.1063588
4	0.832355	1.77476	-7.307237	0.09777561	0.1649825523	0.1063588
5	0.842754	1.79231	-7.306136	-0.1315206	-0.0195284819	0.1063588
6	0.828766	1.79023	-7.305800	0.07843998	0.0618140718	0.1063588

The following is the result from textbook.

Step k	λ_k	α_k	$L(\lambda,\alpha)$	u_λ	u_α	d_k
0	1.024	1.000	−9.757	0.001	7.035	0.098
1	1.025	1.693	−7.491	0.001	−1.80	0.089
2	0.865	1.694	−7.339	0.661	0.001	0.126
3	0.865	1.777	−7.311	0.000	−0.363	0.073
4	0.839	1.777	−7.307	0.121	0.000	0.128
5	0.839	1.792	−7.306	0.000	−0.072	0.007

Some places are different. Also I check the MLE by the following figure1. By the figure1 we can sure that the MLE are correct. I change the stopping criteria:

$$
|x^{k+1} - x^k| < \varepsilon
$$

And do the same thing again. Result is in the table2 and figure2.

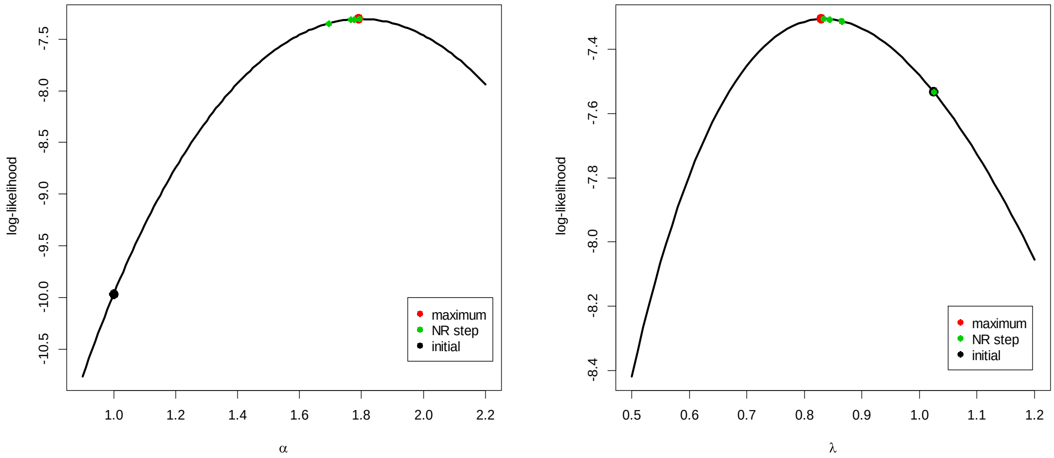

Figure1

Table 2 Set $\varepsilon = 10^{-3}$

Step k	λ^k	α^k	$L(\lambda^k, \alpha^k)$	$u_\lambda(\lambda^k, \alpha^k)$	$u_\alpha(\lambda^k, \alpha^k)$	d^k
0	1.024590	1.00000	-9.757073	1.776357×10^{-15}	7.0338193273	0.0994456
1	1.024590	1.69948	-7.491281	- 1.827545	-0.051930054	0.0872348
2	0.865164	1.69495	-7.338068	0.01022059	0.648619430	0.1267853
3	0.863868	1.77718	-7.311120	-0.3515822	0.001756603	0.0726625
4	0.838321	1.77731	-7.306584	0.0005921650	0.117860428	0.1256964
5	0.838396	1.79213	-7.305697	-0.06898850	0.002226248	0.0717058
6	0.833449	1.79229	-7.305526	0.001061548	0.024072191	0.1203709
7	0.833577	1.79518	-7.305491	-0.01432489	0.001001794	0.0730871
8	0.832530	1.79526	-7.305483	0.0004186131	0.005337692	0.1158475

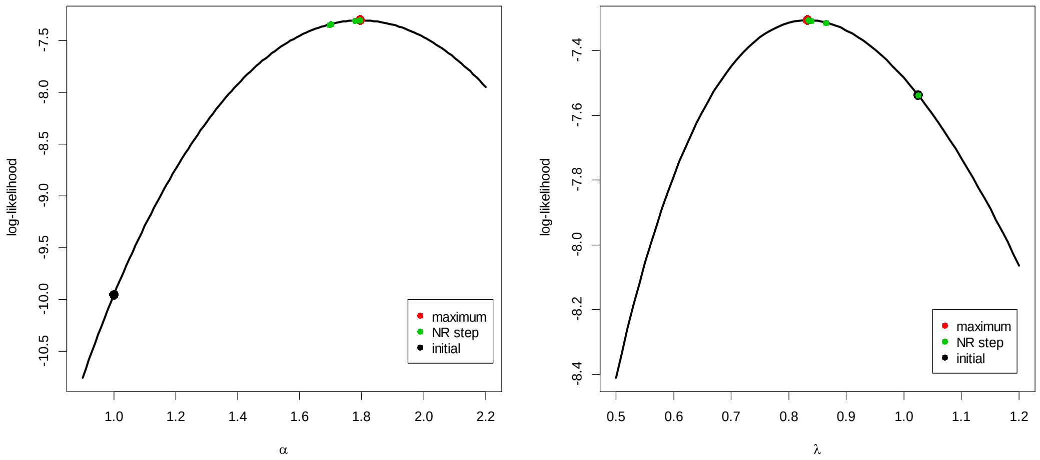
log-likelihood
-7.5
-8.0
-8.5
-9.0
-9.5
-10.0
-10.5
1.0
1.2
1.4
1.6
1.8
2.0
2.2
α
maximum
NR step
initial
log-likelihood
-7.4
-7.6
-7.8
-8.0
-8.2
-8.4
0.5
0.6
0.7
0.8
0.9
1.0
1.1
1.2
λ
maximum
NR step
initial

Figure2

Code

```
score_func = function(lamda,alpha){
matrix(
    c(n/lamda-sum(t^alpha),n/alpha+sum(log(t))-lamda*sum(log(t)*t^alpha)),
    2,1)
}

l_d=function(d){
score_vector = score_func(lamda,alpha)
(n*log(lamda+d*score_vector[1,1])+
n*log(alpha+d*score_vector[2,1])+
(alpha+d*score_vector[2,1]-1)*sum(log(t))-
(lamda+d*score_vector[1,1])*sum(t^(alpha+d*score_vector[2,1])))
}


score_d_func = function(d){
score_vector = score_func(lamda,alpha)

n*score_vector[1,1]/(lamda+d*score_vector[1,1])+
n*score_vector[2,1]/(alpha+d*score_vector[2,1])+
score_vector[2,1]*sum(log(t))-score_vector[1,1]*
sum(t^(alpha+d*score_vector[2,1]))-(lamda+d*score_vector[1,1])*
sum(t^(alpha+d*score_vector[2,1])*score_vector[2,1]*log(t))
}

hessian_d_func = function(d){
score_vector = score_func(lamda,alpha)

-n*score_vector[1,1]^2/(lamda+d*score_vector[1,1])^2-
n*score_vector[2,1]^2/(lamda+d*score_vector[2,1])^2-
2*score_vector[1,1]*sum(t^(alpha+d*score_vector[2,1])*score_vector[2,1]*log(t))-
(lamda+d*score_vector[1,1])*
sum(t^(alpha+d*score_vector[2,1])*(score_vector[2,1]*log(t))^2)
}
t = c(2.57,0.58,0.82,1.02,0.78,0.46,1.04,0.43,0.69,1.37)
n=length(t)
lamda = n/sum(t)
```

```
alpha = 1
lamda_step=c(lamda)
alpha_step=c(alpha)
par_old = matrix(c(lamda_step,alpha_step),2,1)
par_new = matrix(NA,2,1)


d=c(1)
score_step_lamda = c(score_func(n/sum(t),1)[1,1])
score_step_alpha = c(score_func(n/sum(t),1)[2,1])
AI = 1

repeat{
D = d[AI]
lamda = par_old[1,1]
alpha = par_old[2,1]
AI_d = 1
repeat{
D[AI_d+1] = D[AI_d]-score_d_func(D[AI_d])/hessian_d_func(D[AI_d])
error = abs(score_d_func(D[AI_d])
if( error < 10^-1 ){break}
AI_d = AI_d+1
}
d[AI+1] = D[AI_d]

par_new = par_old+d[AI+1]*score_func(par_old[1,1],par_old[2,1])
error1 = abs(score_func(par_old[1,1],par_old[2,1])[1,1])
error2 = abs(score_func(par_old[1,1],par_old[2,1])[2,1])
if( (error1 < 10^-1) && (error2 < 10^-1) ){break}
par_old = par_new
lamda = par_old[1,1]
alpha = par_old[2,1]
AI = AI+1
lamda_step[AI] = par_old[1,1]
alpha_step[AI] = par_old[2,1]
score_step_lamda[AI] = score_func(par_old[1,1],par_old[2,1])[1,1]
score_step_alpha[AI] = score_func(par_old[1,1],par_old[2,1])[2,1]
}
```

```
ll = c()
for( i in 1:Al ){
lamda = lamda_step[i]
alpha = alpha_step[i]
ll[i] = l_d(0)
}
ll


ll2 = c()
lamda = lamda_step[Al]
a = seq(0.9,2.2,by = 0.01)
for( i in 1:length(a) ){
alpha = a[i]
ll2[i] = l_d(0)
}

plot(a,ll2,type = "l",xlab = expression(alpha),ylab = "log-likelihood",lwd = 3)
alpha = alpha_step[1]
points(alpha,l_d(0),cex=1.5,col=1,pch=16)
alpha = alpha_step[Al]
points(alpha,l_d(0),cex=1.5,col=2,pch=16)
for( j in 2:(Al-1)){
alpha = alpha_step[j]
points(alpha,l_d(0),cex=1,col=3,pch=16)
}
legend(1.95,-10,legend=c("maximum","NR
step","initial"),pch=c(16,16),col=c(2,3,1))

ll2 = c()
alpha = alpha_step[Al]
a = seq(0.5,1.2,by = 0.01)
for( i in 1:length(a) ){
lamda = a[i]
ll2[i] = l_d(0)
}
```

```
plot(a,ll2,type = "l",xlab = expression(lambda),ylab = "log-likelihood",lwd = 3)
lamda = lamda_step[1]
points(lamda,l_d(0),cex=1.5,col=1,pch=16)
lamda = lamda_step[AI]
points(lamda,l_d(0),cex=1.5,col=2,pch=16)
for( j in 2:(AI-1)){
lamda = lamda_step[j]
points(lamda,l_d(0),cex=1,col=3,pch=16)
}
legend(1.05,-8.2,legend=c("maximum","NR
step","initial"),pch=c(16,16),col=c(2,3,1))


score_func = function(lamda,alpha){
matrix(
    c(n/lamda-sum(t^alpha),n/alpha+sum(log(t))-lamda*sum(log(t)*t^alpha)),
    2,1)
}

l_d=function(d){
score_vector = score_func(lamda,alpha)
(n*log(lamda+d*score_vector[1,1])+
n*log(alpha+d*score_vector[2,1])+
(alpha+d*score_vector[2,1]-1)*sum(log(t))-
(lamda+d*score_vector[1,1])*sum(t^(alpha+d*score_vector[2,1])))
}

###############change criteria

score_d_func = function(d){
score_vector = score_func(lamda,alpha)

n*score_vector[1,1]/(lamda+d*score_vector[1,1])+
n*score_vector[2,1]/(alpha+d*score_vector[2,1])+
score_vector[2,1]*sum(log(t))-score_vector[1,1]*
sum(t^(alpha+d*score_vector[2,1]))-(lamda+d*score_vector[1,1])*
sum(t^(alpha+d*score_vector[2,1])*score_vector[2,1]*log(t))
}
```

```
hessian_d_func = function(d){
score_vector = score_func(lamda,alpha)

-n*score_vector[1,1]^2/(lamda+d*score_vector[1,1])^2-
n*score_vector[2,1]^2/(lamda+d*score_vector[2,1])^2-
2*score_vector[1,1]*sum(t^(alpha+d*score_vector[2,1])*score_vector[2,1]*log(t))-
(lamda+d*score_vector[1,1])*
sum(t^(alpha+d*score_vector[2,1])*(score_vector[2,1]*log(t))^2)
}
t = c(2.57,0.58,0.82,1.02,0.78,0.46,1.04,0.43,0.69,1.37)
n=length(t)
lamda = n/sum(t)
alpha = 1
lamda_step=c(lamda)
alpha_step=c(alpha)
par_old = matrix(c(lamda_step,alpha_step),2,1)
par_new = matrix(NA,2,1)


d=c(1)
score_step_lamda = c(score_func(n/sum(t),1)[1,1])
score_step_alpha = c(score_func(n/sum(t),1)[2,1])
AI = 1

repeat{
D = d[AI]
lamda = par_old[1,1]
alpha = par_old[2,1]
AI_d = 1
repeat{
D[AI_d+1] = D[AI_d]-score_d_func(D[AI_d])/hessian_d_func(D[AI_d])
error = abs(D[AI_d+1]-D[AI_d])
if( error < 10^-3 ){break}
AI_d = AI_d+1
}
d[AI+1] = D[AI_d]
```

```
par_new = par_old+d[AI+1]*score_func(par_old[1,1],par_old[2,1])
error1 = abs(par_old[1,1]-par_new[1,1])
error2 = abs(par_old[2,1]-par_new[2,1])
if( (error1 < 10^-3) && (error2 < 10^-3) ){break}
par_old = par_new
lamda = par_old[1,1]
alpha = par_old[2,1]
AI = AI+1
lamda_step[AI] = par_old[1,1]
alpha_step[AI] = par_old[2,1]
score_step_lamda[AI] = score_func(par_old[1,1],par_old[2,1])[1,1]
score_step_alpha[AI] = score_func(par_old[1,1],par_old[2,1])[2,1]
}
ll = c()
for( i in 1:AI ){
lamda = lamda_step[i]
alpha = alpha_step[i]
ll[i] = l_d(0)
}
ll

ll2 = c()
lamda = lamda_step[AI]
a = seq(0.9,2.2,by = 0.01)
for( i in 1:length(a) ){
alpha = a[i]
ll2[i] = l_d(0)
}

plot(a,ll2,type = "l",xlab = expression(alpha),ylab = "log-likelihood",lwd = 3)
alpha = alpha_step[1]
points(alpha,l_d(0),cex=1.5,col=1,pch=16)
alpha = alpha_step[AI]
points(alpha,l_d(0),cex=1.5,col=2,pch=16)
for( j in 2:(AI-1)){
alpha = alpha_step[j]
points(alpha,l_d(0),cex=1,col=3,pch=16)
}
```

```
legend(1.95,-10,legend=c("maximum","NR
step","initial"),pch=c(16,16),col=c(2,3,1))

ll2 = c()
alpha = alpha_step[AI]
a = seq(0.5,1.2,by = 0.01)
for( i in 1:length(a) ){
lamda = a[i]
ll2[i] = l_d(0)
}
plot(a,ll2,type = "l",xlab = expression(lambda),ylab = "log-likelihood",lwd = 3)
lamda = lamda_step[1]
points(lamda,l_d(0),cex=1.5,col=1,pch=16)
lamda = lamda_step[AI]
points(lamda,l_d(0),cex=1.5,col=2,pch=16)
for( j in 2:(AI-1)){
lamda = lamda_step[j]
points(lamda,l_d(0),cex=1,col=3,pch=16)
}
legend(1.05,-8.2,legend=c("maximum","NR
step","initial"),pch=c(16,16),col=c(2,3,1))
```