CSSE-37 December 9, 2010

Robust algorithm associated with a parameterization for the three-parameter lognormal distribution

Yoshio Komori

Department of Systems Design and Informatics
Kyushu Institute of Technology

680-4 Kawazu
Iizuka, 820-8502, Japan
komori@ces.kyutech.ac.jp

Abstract

Associated with a parameterization for the three-parameter lognormal distribution, algorithm was proposed by Komori and Hirose (2004), which can find a local maximum likelihood (ML) estimate surely if it exists. Nevertheless, by Vera and Diaz-Garcia (2008) it was shown that performance in finding a local ML estimate deteriorated by adopting the parameterization only and using other algorithm. In this short paper, it will be shown that Komori and Hirose's algorithm and the parameterization recover performance under the same setting for simulated data as that in [Vera and Diaz-Garcia, 2008].

1 Introduction

Parameter estimation methods for the three-parameter lognormal distribution have been studied by many researchers. Many of such studies, for example, are introduced in [9].

The probability density function is given by

$$f(x;\alpha,\beta,\gamma) \stackrel{\text{def}}{=} \frac{1}{\sqrt{2\pi}(x-\alpha)\beta}\exp\left[-\frac{\{\ln((x-\alpha)/\gamma)\}^2}{2\beta^2}\right] \qquad (x>\alpha,\ \beta>0,\ \gamma>0),$$

where x is a variable and α, β and γ are parameters. When x_i $(1 \le i \le n)$ are independent observations, we have the likelihood function: $L(\alpha,\beta,\gamma) \stackrel{\text{def}}{=} \prod_{i=1}^n f(x_i;\alpha,\beta,\gamma)$. For the observations, $x_1 > x_2 \ge \cdots \ge x_{n-1} > x_n$ will be assumed in the sequel without loss of generality.

Noting that a random variable $\ln(X-\alpha)$ is normally distributed, we can see that $L(\alpha,\beta,\gamma)$ achieves its maximum at a point $(\alpha,\hat\beta(\alpha),\hat\gamma(\alpha))$ for a given $\alpha < x_n$, where

$$\hat\beta(\alpha) \stackrel{\text{def}}{=} \sqrt{\frac{1}{n}\sum_{i=1}^n\{\ln(x_i-\alpha)-\ln\hat\gamma(\alpha)\}^2}, \qquad \hat\gamma(\alpha) \stackrel{\text{def}}{=} \exp\left[\frac{1}{n}\sum_{i=1}^n\ln(x_i-\alpha)\right].$$

It is, however, known that $L(\alpha,\hat\beta(\alpha),\hat\gamma(\alpha)) \to \infty$ as $\alpha \to x_n - 0$. Instead of a maximum likelihood (ML) estimate in the usual meaning, thus, a local ML estimate is dealt with, which makes $L(\alpha,\hat\beta(\alpha),\hat\gamma(\alpha))$ relatively and absolutely maximum under the condition $x_n - \alpha > \delta$ for a small $\delta > 0$ [3].

In order to find the local ML estimate of α, one possible search is to display $L(\alpha,\hat\beta(\alpha),\hat\gamma(\alpha))$, but it may have difficulties in some cases in which the shape of $L(\alpha,\hat\beta(\alpha),\hat\gamma(\alpha))$ is complicated or the range of α to search is too wide [1, 3, 5]. Also when an iterative solver like Newton's method is used, similar or other difficulties can happen [7]. That is why many researchers tackled this estimation problem.

On the other hand, in order to avoid such difficulties Munro and Wixley [8] have proposed a parameterization for the three-parameter estimation, where local ML estimates for α, β and γ are sought independently and simultaneously. In the sequel we will simply call a triplet of them a local ML estimate. Their parameterization is given by $\alpha \stackrel{\text{def}}{=} \mu - \sigma/\lambda$, $\beta \stackrel{\text{def}}{=} \lambda$ and $\gamma \stackrel{\text{def}}{=} \sigma/\lambda$ and leads to

$$\begin{aligned} &f(x;\mu-\sigma/\lambda,\lambda,\sigma/\lambda) \\ &= \frac{1}{\sqrt{2\pi}\{\sigma+\lambda(x-\mu)\}}\exp\left[-\frac{\{\ln(\sigma+\lambda(x-\mu))-\ln\sigma\}^2}{2\lambda^2}\right]. \end{aligned} \tag{1. 1}$$

This can permit λ to be negative. We call it the extended lognormal distribution permitting that $\lambda \ne 0$ and $\sigma > 0$. The parameterization is much helpful to improve the convergency of many iterative methods [2, 4]. It is, however, probable that methods fail to find a local ML estimate. For example, see Subsection 3.3 in [6]. In addition, it is unclear whether they fail although a local ML estimate exists or they cannot find because no local ML estimate exists.

These two problems have been overcome with algorithm and another parameterization proposed by Komori and Hirose [7]. That is, the combination of them makes it possible to judge whether a local ML estimate exists or not, and to find it surely if it exists.

Vera and Díaz-García [9] have proposed a global Simulated Annealing (SA) optimization heuristic in the parameterizations mentioned above as well as Wingo's parameterization [10]. In their simulation, however, successful rates in finding a local ML estimate have deteriorated even in Komori and Hirose's parameterization.

In the present paper we will show that not the SA algorithm but Komori and Hirose's algorithm should be used in the parameterization to find a local ML estimate surely if it exits. The paper is organized as follows. In Section 2, we will briefly introduce our parameterization and algorithm. In Section 3, we will give simulation studies and discussion.

2 Komori and Hirose's parameterization and algorithm

The substitution of $\tau \stackrel{\text{def}}{=} \sigma - \lambda\mu$ and $s \stackrel{\text{def}}{=} \ln\sigma$ into (1. 1) yields

$$f(x; -\tau/\lambda, \lambda, \mathrm{e}^s/\lambda) = \frac{1}{\sqrt{2\pi}(\lambda x+\tau)} \exp\left[-\frac{\{\ln(\lambda x+\tau)-s\}^2}{2\lambda^2}\right] \qquad (\lambda \neq 0).$$

By arranging $\ln \bar{L}(\lambda, \tau, s) \stackrel{\text{def}}{=} \sum_{i=1}^n \ln f(x_i; -\tau/\lambda, \lambda, \mathrm{e}^s/\lambda)$, we obtain

$$\begin{aligned}\ln \bar{L}(\lambda, \tau, s) = &-\frac{n}{2\lambda^2}\left\{s - \frac{1}{n}\sum_{i=1}^n \ln(\lambda x_i+\tau)\right\}^2 - n\ln\sqrt{2\pi} \\ &+\frac{1}{2n\lambda^2}\left\{\sum_{i=1}^n \ln(\lambda x_i+\tau)\right\}^2 - \frac{1}{2\lambda^2}\sum_{i=1}^n\{\ln(\lambda x_i+\tau)\}^2 - \sum_{i=1}^n \ln(\lambda x_i+\tau).\end{aligned}$$

The first term has the maximum value 0 when $s = (1/n)\sum_{i=1}^n \ln(\lambda x_i+\tau)$. Hence, all we need to do is to maximize the following function $F(\lambda, \tau)$:

$$F(\lambda,\tau) \stackrel{\text{def}}{=} \frac{1}{2n\lambda^2}\left\{\sum_{i=1}^n \ln(\lambda x_i+\tau)\right\}^2 - \frac{1}{2\lambda^2}\sum_{i=1}^n\{\ln(\lambda x_i+\tau)\}^2 - \sum_{i=1}^n \ln(\lambda x_i+\tau).$$

In order to achieve this, the algorithm is given as follows [7]. It provides the profile of $F(\lambda, \tau)$ concerning $\lambda > 0$ or $\lambda < 0$, respectively.

In the interval $\lambda > 0$:

1) $\tau \leftarrow \tau^*$, $\lambda \leftarrow \varepsilon_0 > 0$.

2) If $\lambda > \lambda^+_{max}$, end. Otherwise, $\tau_{min} \leftarrow -\lambda x_n$, $\tau_{max} \leftarrow \tau^+_U(\lambda)$.

3) If $\tau_{min} < \tau < \tau_{max}$ and $\dfrac{\partial F}{\partial \tau}(\lambda, \tau) > 0$, $\tau_{min} \leftarrow \tau$. If $\tau_{min} < \tau < \tau_{max}$ and $\dfrac{\partial F}{\partial \tau}(\lambda, \tau) \leq 0$, $\tau_{max} \leftarrow \tau$.

4) If $\dfrac{\partial F}{\partial \tau}(\lambda, (\tau_{min}+\tau_{max})/2) > 0$, then $\tau_{min} \leftarrow (\tau_{min}+\tau_{max})/2$. Otherwise, $\tau_{max} \leftarrow (\tau_{min}+\tau_{max})/2$.

5) If $(\tau_{max}-\tau_{min})/|\tau_{max}| > \varepsilon_1$, then go to 4). Otherwise, $\tau \leftarrow \tau_{max}$.

6) If $\left|\dfrac{\partial F}{\partial \tau}(\lambda, \tau)\right| < \varepsilon_2$, then record $(\lambda, \tau, F(\lambda, \tau))$, $\lambda \leftarrow \lambda + \triangle\lambda$ and go to 2). Otherwise, end.

In the interval $\lambda < 0$, replace 1), 2) and 6) with 1′), 2′) and 6′), respectively:

1′) $\tau \leftarrow \tau^*$, $\lambda \leftarrow -\varepsilon_0$.

2′) If $\lambda < \lambda_{min}^-$, end. Otherwise, $\tau_{min} \leftarrow -\lambda x_1$, $\tau_{max} \leftarrow \tau_U^-(\lambda)$.

6′) If $\left|\dfrac{\partial F}{\partial \tau}(\lambda, \tau)\right| < \varepsilon_2$, then record $(\lambda, \tau, F(\lambda, \tau))$, $\lambda \leftarrow \lambda - \triangle\lambda$ and go to 2′). Otherwise, end.

Here,

$$\tau^* \stackrel{\text{def}}{=} \frac{1}{n}\sqrt{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(x_i - x_j)^2}, \qquad \tau_U^+(\lambda) \stackrel{\text{def}}{=} -\lambda x_n\left(\frac{1 - \bar{x}/x_n \mathrm{e}^{-\lambda^2}}{1 - \mathrm{e}^{-\lambda^2}}\right) \quad (\lambda > 0),$$
$$\tau_U^-(\lambda) \stackrel{\text{def}}{=} -\lambda x_1\left(\frac{1 - \bar{x}/x_1 \mathrm{e}^{-\lambda^2}}{1 - \mathrm{e}^{-\lambda^2}}\right) \quad (\lambda < 0).$$

In addition, ε_0, ε_1, ε_2, λ_{max}^+, $\triangle\lambda$ and λ_{min}^- are preassigned constants. In MATLAB codes mentioned in Section 3, for example, some of them are given as follows [7]:

$$\varepsilon_0 = 0.05, \quad \varepsilon_1 = 10^{-14}, \quad \varepsilon_2 = 0.01, \quad \lambda_{max}^+ = 6, \quad \lambda_{min}^- = -6.$$

Note that in 5) an approximate to the solution, say $\tau_0(\lambda)$, of $\frac{\partial F}{\partial \tau}(\lambda, \tau) = 0$ is obtained. Using $\{(\lambda, F(\lambda, \tau))\}$ in the records in 6) and 6′), we can plot $F(\lambda, \tau_0(\lambda))$. If a local ML estimate exists and we set $\triangle\lambda$ at a sufficiently small positive value, from the plot data we can immediately get the extreme point of $F(\lambda, \tau)$ with high accuracy. Because the algorithm is based on the bisection method and $\tau_0(\lambda)$ lies in $(-\lambda x_n, \tau_U^+(\lambda))$ or $(-\lambda x_1, \tau_U^-(\lambda))$, the algorithm is significantly robust.

3 Simulation studies and discussion

In this section we give numerical results for data simulated by using the same function as one used by Vera and Díaz-García [9].

Table 1 shows the results given by them concerning the combination of the SA algorithm and our parameterization. In their simulation, μ and σ have been fixed at 0 and 1, respectively. In addition, by communicating with one of them, we have known that in MATLAB the function 'randn' with a method 'state' and a different initial state each time was used to generate 1000 sets of pseudo-random data and the existence of a non-degenerate solution was manually checked for each data set. This means that it is almost impossible to reproduce the same data sets and results.

Thus, whereas we reconstruct a similar setting for simulation by using the same function, differently from their way we use a constant initial state for the function and seek for existence rates automatically by utilizing $F(\lambda, \tau_0(\lambda))$. For example, Fig. 1 shows profiles of F. One of them is the case of no ML estimate, whose data set was generated by setting $(\lambda, n) = (2, 10)$ and the initial state at 0 for the function 'randn'. The other is the case of

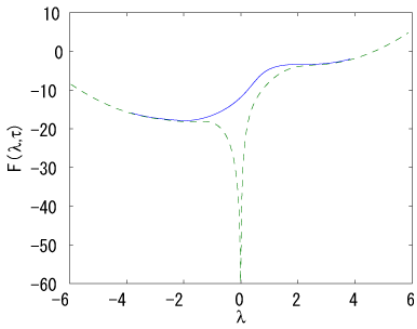
Table 1: Successful rate in finding a local ML estimate and its existence rate in [9]

		λ							
		.25	.50	.75	1.0	1.25	1.5	1.75	2.0
	10	.989	.966	.942	.880	.736	.651	.604	.623
		(.962)	(.951)	(.903)	(.791)	(.655)	(.375)	(.169)	(.077)
n	15	1.00	.997	.995	.991	.974	.926	.822	.686
		(.990)	(.994)	(.990)	(.975)	(.944)	(.891)	(.697)	(.379)
	20	1.00	1.00	1.00	1.00	.999	.996	.991	.973
		(.987)	(1.00)	(.999)	(.999)	(.996)	(.983)	(.941)	(.825)

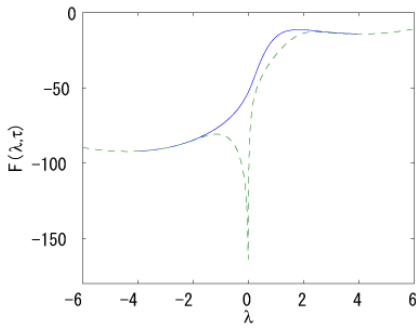
A value in parentheses indicates an existence rate of a local ML estimate.

Table 2: Existence rate of a local ML estimate by our algorithm

		λ							
		.25	.50	.75	1.0	1.25	1.5	1.75	2.0
	10	.978	.961	.909	.826	.707	.543	.393	.259
n	15	1.00	1.00	.997	.980	.944	.857	.710	.519
	20	1.00	1.00	1.00	1.00	.996	.983	.922	.800



Case of no ML estimate



Case of existence a local ML estimate

Figure 1: Profiles of $F(\lambda, \tau)$

existence of a local ML estimate, whose data set was generated by setting $(\lambda, n) = (2, 20)$ and the same initial state. In the figure, the solid curves indicate $F(\lambda, \tau_0(\lambda))$, whereas the dotted curves indicate $F(\lambda, \tau_U^+(\lambda))$ or $F(\lambda, \tau_U^-(\lambda))$, which we may regard as $F(\lambda, \tau_0(\lambda)))$ for large $|\lambda|$ [7]. Here, note that the interval $[-6, 6]$ of λ is large enough for the global search.

MATLAB codes for the simulation are given in the appendix. Other codes for examples are also obtainable from Komori's homepage:

http://galois.ces.kyutech.ac.jp/˜komori/

Thus, it is possible for readers to check these results if they want.

From the tables, we can see that the existence rates are improved in all cases except the three cases $(\lambda, n) = (1.5, 15)$, $(20, 1.75)$ and $(20, 2.0)$. At the same time because in all cases our parameterization and algorithm always successfully find a local ML estimate

if it exists, we can conclude that they much improve performance in finding estimates, compared with the combination of the SA algorithm and our parameterization.

Acknowledgements

The author is grateful to Professor José Fernando Vera for helpful communications.

Appendix

Three MATLAB codes for our simulation are given as follows, whose names are mainbisection_2p_for_simulation.m, dF_dTau.m and funcF.m.

—— **mainbisection_2p_for_simulation.m** ——

```
% Filename: mainbisection_2p_for_simulation.m
% This is the main code for Monte-Carlo simulation.
%%%
% Initialization for a data set
lambda=2; % 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75 or 2.
initstate=0; % from 0 to 2^32-1
ndata=10; % 10, 15 or 20
al=-1/lambda;
lnGam=-log(lambda);
randn('state',initstate);
%%%
eps0=0.05;
eps1=1e-14;
eps2=1e-2;
dLam=0.05;
i_lam_max=1000;
%%%
num_sim=1000; % 1000
%%%
record_plus=zeros(i_lam_max,3);
record_minus=record_plus;
record_all=zeros(i_lam_max*2,3);
max_triplet=zeros(10,3);
%%%
positive_cnt=0;
sim_exist_cnt=0;
for sim_i=1:num_sim
    % Data set part --- begin ---
    cur_state=randn('state');
    Z=randn(ndata,1);
    tmp=lnGam+lambda*Z;
    datax=exp(tmp)+al;
    % Data set part --- end ---
```

```
%
datax=sort(datax,'descend');
xmax=datax(1);
xmin=datax(length(datax));
xmean=mean(datax);
%
tau_ast=0;
for ii=1:ndata-1
    tmp=datax(ii+1:ndata)-datax(ii);
    tau_ast=tau_ast+sum(tmp.*tmp);
end
tau_ast=sqrt(tau_ast)/ndata;
%
break_flag_plus=0;
% Search in the region lambda > 0
tau_tmp=tau_ast;
for i_lam=1:i_lam_max
    lam_tmp=eps0+dLam*(i_lam-1);
    tau_min=-lam_tmp*xmin;
    tmp=exp(-lam_tmp*lam_tmp);
    tau_max=tau_min*(1-xmean/xmin*tmp)/(1-tmp);
    if (tau_min<tau_tmp) && (tau_tmp<tau_max)
        [val,errflag]=dF_dTau(datax, lam_tmp, tau_tmp);
        if 0>errflag
            return;
        elseif 1==errflag
            break_flag_plus=1;
            break;
        end
        if 0<val
            tau_min=tau_tmp;
        else
            tau_max=tau_tmp;
        end
    end
    %
    while (eps1 < (tau_max-tau_min)/abs(tau_max))
        [val,errflag]=dF_dTau(datax, lam_tmp, (tau_min+tau_max)/2);
        if 0>errflag
            return;
        elseif 1==errflag
            break_flag_plus=1;
            break;
        end
        if 0<val
            tau_min=(tau_min+tau_max)/2;
        else
```

```
            tau_max=(tau_min+tau_max)/2;
        end
    end
    %
    tau_tmp=tau_max;
    [val,errflag]=dF_dTau(datax, lam_tmp, tau_tmp);
    if 0>errflag
        return;
    elseif 1==errflag
        break_flag_plus=1;
        break;
    end
    if eps2>abs(val)
        record_plus(i_lam,:)=...
            [lam_tmp tau_tmp funcF(datax,lam_tmp,tau_tmp)];
    else
        break_flag_plus=2;
        break;
    end
end
if i_lam_max==i_lam
    break_flag_plus=3;
end
i_lam_max_plus1=i_lam-1;
%
break_flag_minus=0;
% Search in the region lambda < 0
tau_tmp=tau_ast;
for i_lam=1:i_lam_max
    lam_tmp=-eps0-dLam*(i_lam-1);
    tau_min=-lam_tmp*xmax;
    tmp=exp(-lam_tmp*lam_tmp);
    tau_max=tau_min*(1-xmean/xmax*tmp)/(1-tmp);
    if (tau_min<tau_tmp) && (tau_tmp<tau_max)
        [val,errflag]=dF_dTau(datax, lam_tmp, tau_tmp);
        if 0>errflag
            return;
        elseif 1==errflag
            break_flag_plus=1;
            break;
        end
        if 0<val
            tau_min=tau_tmp;
        else
            tau_max=tau_tmp;
        end
    end
```

```
    %
    while (eps1 < (tau_max-tau_min)/abs(tau_max))
        [val,errflag]=dF_dTau(datax, lam_tmp, (tau_min+tau_max)/2);
        if 0>errflag
            return;
        elseif 1==errflag
            break_flag_plus=1;
            break;
        end
        if 0<val
            tau_min=(tau_min+tau_max)/2;
        else
            tau_max=(tau_min+tau_max)/2;
        end
    end
    %
    tau_tmp=tau_max;
    [val,errflag]=dF_dTau(datax, lam_tmp, tau_tmp);
    if 0>errflag
        return;
    elseif 1==errflag
        break_flag_plus=1;
        break;
    end
    if eps2>abs(val)
        record_minus(i_lam,:)=...
            [lam_tmp tau_tmp funcF(datax,lam_tmp,tau_tmp)];
    else
        break_flag_minus=2;
        break;
    end
end
if i_lam_max==i_lam
    break_flag_minus=3;
end
i_lam_max_minus1=i_lam-1;
%
% Summarizing the record
for ii=1:i_lam_max_minus1
    record_all(ii,:)=record_minus(i_lam_max_minus1+1-ii,:);
end
for ii=1:i_lam_max_plus1
    record_all(i_lam_max_minus1+ii,:)=record_plus(ii,:);
end
i_lam_max_all=i_lam_max_minus1+i_lam_max_plus1;
%
% Search for the local maximum
```

```
    i_max_triplet=0;
    up_down_flag=0;
    f_tmp=record_all(1,3);
    for ii=2:i_lam_max_all
        if f_tmp>record_all(ii,3)
            if 1==up_down_flag
                i_max_triplet=i_max_triplet+1;
                max_triplet(i_max_triplet,:)=...
                    [record_all(ii-1,1:2) f_tmp];
                up_down_flag=-1;
            end
            up_down_flag=-1;
        else
            up_down_flag=1;
        end
        f_tmp=record_all(ii,3);
    end
    %
    mle_triplet=max_triplet(1,:);
    for ii=2:i_max_triplet
        if mle_triplet(3)<max_triplet(ii,3)
            mle_triple=max_triplet(ii,:);
        end
    end
    %
    if 0~=i_max_triplet
        sim_exist_cnt=sim_exist_cnt+1;
        if 0<mle_triplet(1)
            positive_cnt=positive_cnt+1;
        end
    else
        fstate=cur_state;
    end
    %
end
%
if 0~=sim_exist_cnt
    fprintf(1, ...
     '(lambda, ndata, nsim, ex_rate, p_rate) =(%f, %d, %d, %f, %f)\n',...
     lambda, ndata, num_sim, sim_exist_cnt/num_sim,...
     positive_cnt/sim_exist_cnt);
else
    fprintf(1, '(lambda, ndata, nsim, ex_rate) = (%f, %d, %d, %f)\n',...
        lambda, ndata, num_sim, sim_exist_cnt/num_sim);
end
```

—— dF_dTau.m ——

```
% Filename: dF_dTau.m
% This function gives a value of dF/dTau.
function [val,errflag] = dF_dTau(xdata, lambda, tau)
  if 1e-15>=abs(lambda)
      fprintf(1, 'lambda is invalid!\n');
      val=0;
      errflag=-1;
      return;
  end
  %
  lam2=lambda*lambda;
  tmp=lambda*xdata+tau;
  if 1~=isempty(find(tmp<=0))
      val=0;
      errflag=1;
      return;
  end
  lnTmp=log(tmp);
  tmp1=sum(1./tmp);
  val=sum(lnTmp)*tmp1/(length(xdata)*lam2)-sum(lnTmp./tmp)/lam2-tmp1;
  errflag=0;
end
```

—— **funcF.m** ——

```
% Filename: funcF.m
% This function gives a value of F.
function val = funcF(xdata, lambda, tau)
  lam2=lambda*lambda;
  tmp=lambda*xdata+tau;
  lnTmp=log(tmp);
  tmp1=sum(lnTmp);
  val=tmp1*tmp1/(2*length(xdata)*lam2)-sum(lnTmp.*lnTmp)/(2*lam2)-tmp1;
end
```

References

[1] R.C.H. Cheng and T.C. Iles. Embedded models in three-parameter distributions and their estimation. *J. Royal Statist. Soc.* B, **52**(**1**):135–149, 1990.

[2] J.F. Eastham, V.N. LaRiccia, and J.H. Schuenemeyer. Small sample properties of the maximum likelihood estimators for an alternative parameterization of the three-parameter lognormal distribution. *Comm. Statist.—Simulation Comput.*, **16**(**3**):871–884, 1987.

[3] B.M. Hill. The three-parameter lognormal distribution and Bayesian analysis of a point-source epidemic. *J. Amer. Statist. Assoc.*, **58**:72–84, 1963.

[4] H. Hirose. Maximun likelihood parameter estimation in the three-parameter lognormal distribution using the continuation method. *Comput. Statist. Data Anal.*, **24**:139–152, 1997.

[5] N.L. Johnson, S. Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*. John Wiley & Sons, New York, 1994.

[6] Y. Komori and H. Hirose. Easy estimation of by a new parameterization for the three-parameter lognormal distribution. Technical Report CSSE-16, Kyushu Institute of Technology, 2001. Preprint of a paper.

[7] Y. Komori and H. Hirose. Easy estimation of by a new parameterization for the three-parameter lognormal distribution. *J. Statist. Comput. Simulation*, **74**(**1**):63–74, 2004.

[8] A.H. Munro and R.A.J. Wixley. Estimation on order statistics of small samples from a three-parameter lognormal distribution. *J. Amer. Statist. Assoc.*, **65**(**329**):212–225, 1970.

[9] J.F. Vera and J.A. Díaz-García. A global simulated annealing heuristic for the three-parameter lognormal maximum likelihood estimation. *Comput. Statist. Data Anal.*, **52**:5055–5065, 2008.

[10] D.R. Wingo. Fitting three-parameter lognormal models by numerical global optimization - an improved algorithm. *Comput. Statist. Data Anal.*, **2**:13–25, 1984.