Chapter 8
MONTE CARLO SIMULATIONS OF DELOCALIZED TRANSITIONS

Abstract In this chapter we introduce Monte Carlo (MC) simulations of models based on delocalized transitions, and compare the MC results with the deterministic solutions of the corresponding differential equations. We present the R codes for fixed time interval MC methods to simulate CW-OSL, LM-OSL, TL and ITL processes and discuss how luminescence processes can be described within the general framework of birth and death processes. Vectorized R codes are discussed, and we show how the speed of the R codes can be improved significantly by using vectorized commands. We provide the R codes for estimating the stochastic uncertainties ($CV\%$) in a luminescence model, and present examples of luminescence phenomena as birth and death processes. We show an example of luminescence signals from a system of small clusters, as one may encounter in nanodosimetric materials. The chapter concludes with a Monte Carlo simulation of irradiation processes within the GOT model.

Code 8.1: Simple MC implementation of CW-OSL process

```
# Simulate CW-OSL process using the simplest MC code
# Original Mathematica program by Vasilis Pagonis
# R version written by Johannes Friedrich, 2018
rm(list = ls(all=T))
options(warn=-1)
library(matrixStats)
# Define Parameters
mu <- 0.03 # probability of optical excitation per second
deltat <- 1
times <- seq(1, 100, deltat) # time sequence
```

```
n0  <- 500 # initial number of electrons at t=0
# Number of iterations of the Monte carlo process
mcruns <- 100
nMatrix  <- matrix(NA, nrow = length(times), ncol = mcruns)
# The 3 main Monte Carlo loops follow
system.time(invisible(
  for (k in 1:mcruns)
  {  n <- n0
  for (t in 1:length(times)){
    for (j in 1:n){
      r <- runif(1)  # random number in (0,1)
      P <- mu*deltat
      if (r < P) n <- n - 1  # the electron has recombined
    }
    nMatrix[t,k] <- n  }  }  ))
# Take average of iterations
avgn <- rowMeans(nMatrix)
## plot MC and analytical solution n(t)
par(mfrow=c(1,3))
pch<-c(NA,NA,1,NA)
lty<-c(NA,NA,NA,"solid")
col<-c(NA,NA,"black","red")
matplot(x=times,nMatrix,xlab = "Time [s]",
        ylab = "Remaining electrons",ylim=c(0,700))
legend("topright",bty="n",legend=c("(a)"," ",
 "n(t) ","MC","n0=500 M=100"))
plot(x = times,      y = avgn,type = "p",pch = 1,,ylim=c(0,700),
     xlab = "Time [s]", ylab = "Average of Remaining electrons")
curve(n0*exp(-mu*x),0,max(t),add=TRUE,col="red",lwd=2)
legend("topright",bty="n",c("(b)   "," "," ","MC (M=100)",
  "Analytical"),pch=pch,lty=lty,col=col)
plot(x = times,   y = mu*avgn,type = "p",pch = 1,ylim=c(0,20),
     xlab = "Time [s]",  ylab = "Average of CW-OSL signal")
legend("topright",bty="n",c("(c)      CW-OSL"," ","MC (M=100)",
 "Analytical"),pch=pch,lty=lty,col=col)
curve(n0*mu*exp(-mu*x),from=0,max(t),add=TRUE,col="red",lwd=2)

  ##    user  system elapsed
  ##    3.22    0.00    3.23
```

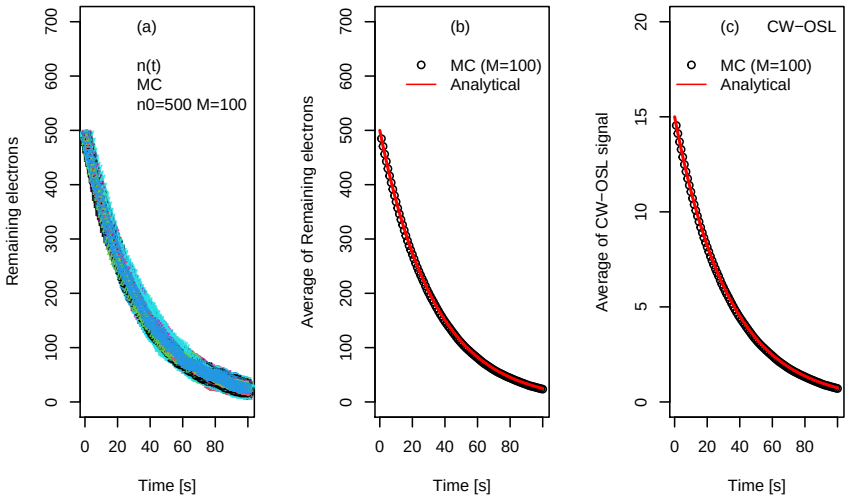

Fig. 8.1: Simplest MC implementation of CW-OSL luminescence process. (a) Plot of M =100 MC runs with the same initial number of electrons n_0=500, simulating a total of 50,000 electrons. (b) Plot of the average of the M =100 MC iterations in (a). The solid lines in (b) and (c) represents the analytical solution of the differential equation.

Stochastic process	Birth rate	Death rate	Luminescence process
Simple linear pure death	$\lambda_n = 0$	$\mu_n = \mu n$	CW-OSL ITL
Generalized simple linear pure death	$\lambda_n = 0$	$\mu_n = \mu(t)\, n$	TL LM-OSL
Simple nonlinear pure birth	$\lambda_n = \lambda_n(n)$	$\mu_n = 0$	Dose response GOT Eq.(**??**)
Generalized simple nonlinear pure death	$\lambda_n = 0$	$\mu_n = \mu(n)$	TL (MOK model) OSL (MOK model)

Table 8.1: Examples of various luminescence processes and their corresponding stochastic birth-death processes.

Code 8.2: Populations P(j) of stochastic simple death process

```
# Populations P(j) of stochastic simple death process
rm(list = ls(all=T))
n0<-100
mu<-.03
x<-seq(1,100)
f<-function(u) {choose(n0,u)*exp(-mu*u*t)*
    ((1-exp(-mu*u*t))**(n0-u))}
times<-c(.01,1,20,40,110)
TF=c(FALSE,rep(TRUE,4))
for (i in 1:5){
  t<-times[i]
  area<-sum(unlist(lapply(x,f)))
  curve(choose(n0,round(x))*exp(-mu*x*t)*
((1-exp(-mu*x*t))**(n0-x))/area,
  1,100,ylim=c(0,.4),lwd=3,add=TF[i],col=i,lty=i,
  xlab="# of particles j",ylab="probability Pn(j,t)")}
legend("topleft",bty="n",c("t=.01 s","1 s ","20 s ","40 s ",
 "110 s"), col=1:5,lty=1:5,lwd=3)
legend("top",bty="n",c("Populations P(j,t)"," ",
"Stochastic death process"))
```

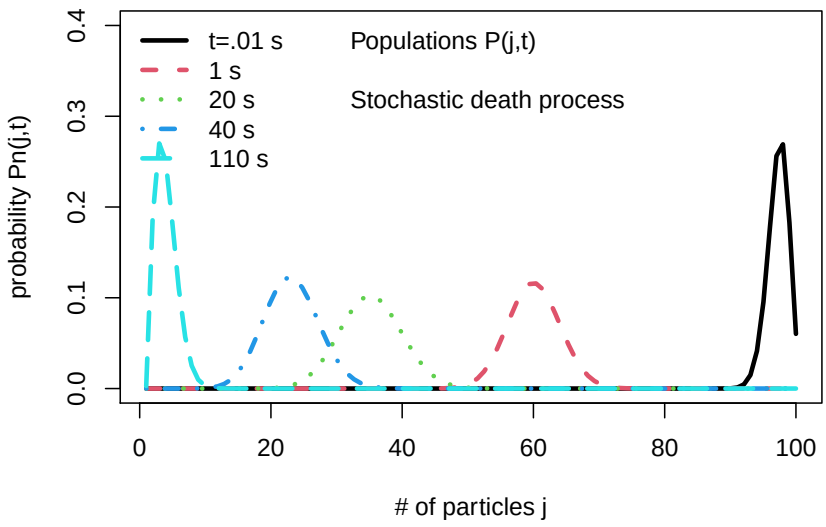

Fig. 8.2: Plots of $P_j(t)$ from Eq.(**??**), for a simple death stochastic process. As the time t increases from right to left in this plot, the width initially increases and then decreases with t. For more details see Lawless et al. [21].

Code 8.3: Plots of stochastic simple death process

```
rm(list = ls(all=T))
n0<-100
mu<-.03
par(mfrow=c(1,3))
curve(n0*exp(-mu*x),0,100,lwd=3,xlab="Time t, s",ylab="<n(t)>",
      ylim=c(0,140))
legend("top",bty="n",c("(a)"," ","Stochastic", "<n>"))
curve(sqrt(n0)*sqrt(exp(-mu*x)-exp(-2*mu*x)),0,100,lwd=3,
      xlab="Time t, s",ylab=expression(sigma),ylim=c(0,8))
legend("top",bty="n",c(expression("(b)"," ","Stochastic",sigma)))
 curve(100*sqrt(exp(mu*x)-1)/n0,1,100,xlab="Time t, s",lwd=3,
             ylab="CV[%]",ylim=c(0,7))
 legend("top",bty="n",c("(c)"," ","Stochastic","CV[%]"))
```

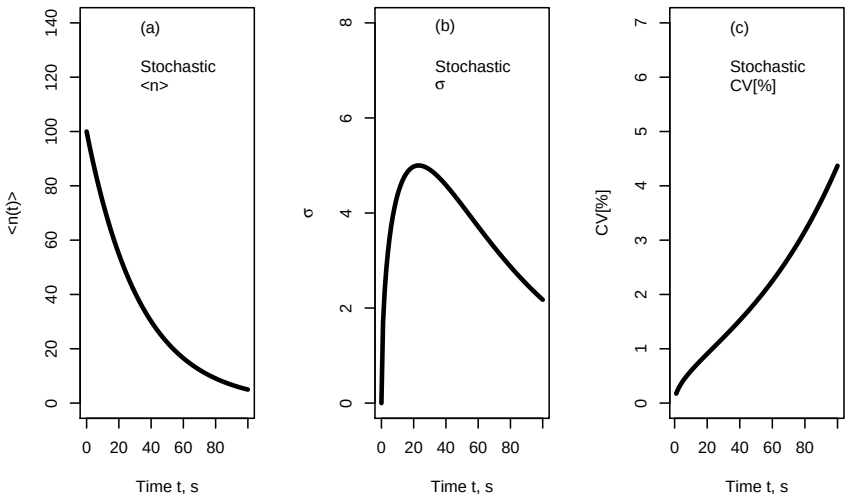

Fig. 8.3: (a) Plot of stochastic mean number of particles $< n(t >$. (b) Plot of stochastic standard deviation σ_n . (c) Plot of *CV[%]* from Eq.(**??**),(**??**) and (**??**).

Code 8.4: Vectorized MC implementation of CW-OSL

```
# Vectorized MC code for first-order CW-OSL process
rm(list = ls(all=T))
options(warn=-1)
library(matrixStats)
mcruns<-300
n0<-500
mu<-.03
deltat<-1
tmax<-100
times<-seq(1,tmax,deltat)
nMatrix <-  matrix(NA, nrow = length(times), ncol = mcruns)
nMC<-rep(NA,length(times))
system.time(
    for (k in 1:mcruns){
        n<-n0              #initialize each of the M=100 MC runs
        for (t in 1:length(times)){
        vec<-rep(runif(n))    #create a vector vec,
        #containing n random numbers between 0 and 1
        P<-mu*deltat
        n<-length(vec[vec>P]) #if the random number in vec is >P,
        #then the corresponding electron survives
     nMC[t]<-n } # store number of electrons n in the vector nMC
     nMatrix[,k]<-nMC # store single run in column k of nMatrix
    })
#Find average of n(t), CW-OSL signal, and CV[%]
avgn<-rowMeans(nMatrix)
avgCWOSL<-mu*rowMeans(nMatrix)
sd<-rowSds(nMatrix)
cv<-100*sd/avgn
par(mfrow=c(1,3))
pch<-c(NA,NA,1,NA)
lty<-c(NA,NA,NA,"solid")
col<-c(NA,NA,2,1)
```

```
plot(times,avgn,ylab="Remaining electrons n(t)",
     xlab="Time [s]",ylim=c(0,700),col=2)
curve(n0*exp(-mu*x),0,tmax,add=TRUE,col=1,lwd=2)
legend("topright",bty="n",c("(a)    n(t)"," ","MC (M=100)",
 "Analytical"),pch=pch,lty=lty,col=col)
plot(times,avgCWOSL,ylab="CWOSL",xlab="Time [s]",ylim=c(0,20),
col=2)
legend("topright",bty="n",c("(b)    CW-OSL"," ","MC (M=100)",
  "Analytical"),pch=pch,lty=lty,col=col)
curve(n0*mu*exp(-mu*x),0,tmax,add=TRUE,col=1,lwd=2)
plot(times,cv,ylab="CV[%]",xlab="Time [s]",ylim=c(0,27),col=2)
curve(100*sqrt((exp(mu*x)-1)/n0),0,max(times),add=TRUE,
      col=1,lwd=2)
legend("topleft",bty="n",c("(c)    CV[%]"," ","MC (M=100)",
    "Analytical"),pch=pch,lty=lty,col=col)

  ##    user  system elapsed
  ##    0.34    0.00    0.35
```

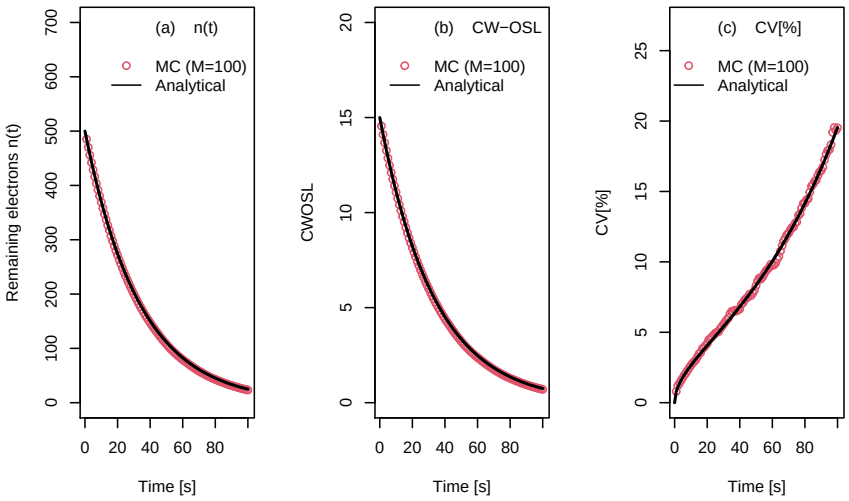

Fig. 8.4: Vectorized MC implementation of the first order CW-OSL luminescence process. (a) Plot of M =100 MC runs with the same initial number of electrons n_0=500, simulating a total of 50,000 electrons. (b) Average of the M =100 MC iterations in (a). (c) The corresponding *CV[%]*. The solid lines represent the analytical solution of the differential equation. For more details ad examples, see Pagonis et al. [40].

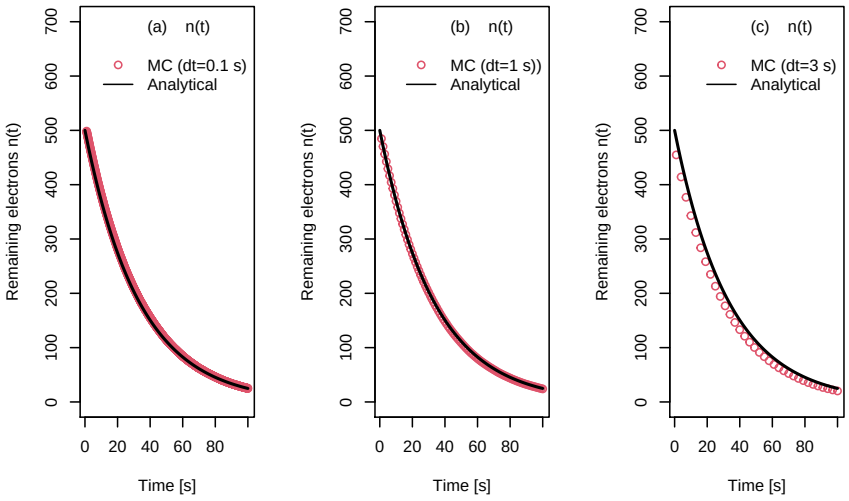

Fig. 8.5: The effect of the parameter *deltat* in the previous MC implementation of the first order CW-OSL luminescence process, with (a) *deltat*=0.1 s, (b) *deltat*=1 s, (c) *deltat*=3 s. All three runs are carried out with the same parameters M =100 MC runs, initial number of electrons n_0=500, $\mu = 0.03$ s^{-1}.

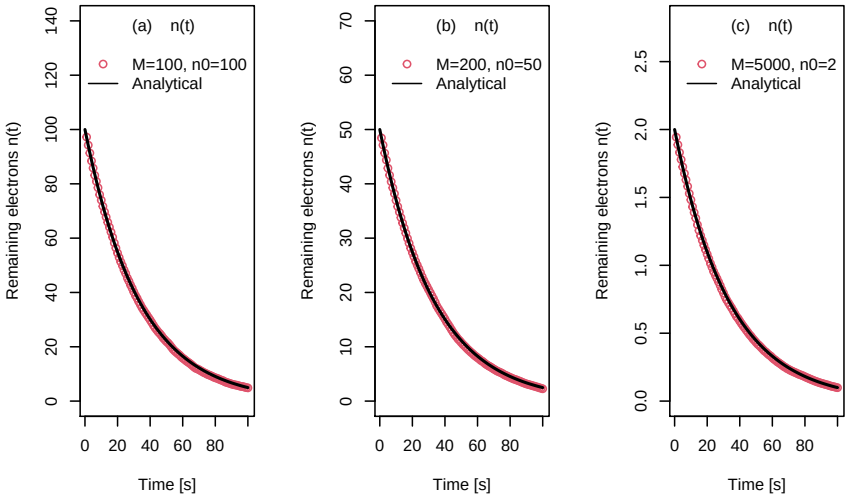

Fig. 8.6: The effect of the parameter n_0 in the previous MC implementation of the first order CW-OSL luminescence process, with (a) $n_0 = 100$ and $M = 100$ MC runs, (b) $n_0 = 50$ and $M = 200$, (c) $n_0 = 2$ and $M = 5000$. All three runs are carried out with the same parameters $\mu = 0.03$ s^{-1} and the same total number of electrons $n_0 \times M = 10^4$.

Code 8.5: Vectorized MC implementation of TL

```
rm(list = ls(all=T))
options(warn=-1)
library(matrixStats)
mcruns<-100
n0<-500
s<-1e12
E<-1
kb<-8.617e-5
tmax<-150
deltat<-1
times<-seq(0,tmax,deltat)
nMatrix<-TLMatrix<-matrix(NA,nrow=length(times),ncol=mcruns)
nMC<-TL<-rep(NA,length(times))
system.time(
for (j in 1:mcruns){
  n<-n0
  for (t in 1:length(times)){
    vec<-rep(runif(n))
    P<-s*exp(-E/(kb*(t+273)))*deltat
    n<-length(vec[vec>P])
    nMC[t]<-n
    TL[t]<-n*P}
nMatrix[,j]<-nMC
TLMatrix[,j]<-TL
 })
#Find average n(t),average CW-OSL signal and CV[%]
avgn<-rowMeans(nMatrix)
avgTL<-rowMeans(TLMatrix)
sd<-rowSds(TLMatrix)
cv<-100*sd/avgTL
## Calculate the analytical error of TL in first order peak
```

```
x1<-times+273
k<-function(u) {integrate(function(p){s*exp(-E/(kb*p))},
 273,u)[[1]]}
y1<-lapply(x1,k)
x<-unlist(x1)
y<-unlist(y1)
errn<-sqrt(n0*(exp(-y)-exp(-2*y)))
nanalyt<-n0*exp(-y)
TLanalyt<-n0*s*exp(-E/(kb*x))*exp(-y)
# plots
par(mfrow=c(1,3))
pch<-c(NA,NA,1,NA)
lty<-c(NA,NA,NA,"solid")
col<-c(NA,NA,2,1)
plot(times,avgn,ylab="Remaining electrons n(t)",
col=2,xlab=expression("Temperature ["^"o"*"C]"),ylim=c(0,700))
lines(x-273,nanalyt,col=1)
legend("topleft",bty="n",c("(a)    n(t)"," ","MC",
 "Lambert Eq."),pch=pch,lty=lty,col=col)
plot(times,avgTL,ylab="TL",
 col=2,xlab=expression("Temperature ["^"o"*"C]"),ylim=c(0,23))
lines(x-273,TLanalyt,col=1)
legend("topleft",bty="n",c("(b)    TL"," ","MC",
 "Lambert Eq."),pch=pch,lty=lty,col=col)
plot(times,cv,ylab="CV[%]",ylim=c(0,150),
col=2,xlab=expression("Temperature ["^"o"*"C]"))
lines(x-273,100*errn*s*exp(-E/(kb*x))/TLanalyt,col=1)
legend("topleft",bty="n",c("(c)    CV[%]"," ","MC",
  "Analytical"), pch=pch,lty=lty,col=col)
```

```
  ##    user  system elapsed
  ##    0.31    0.00    0.31
```

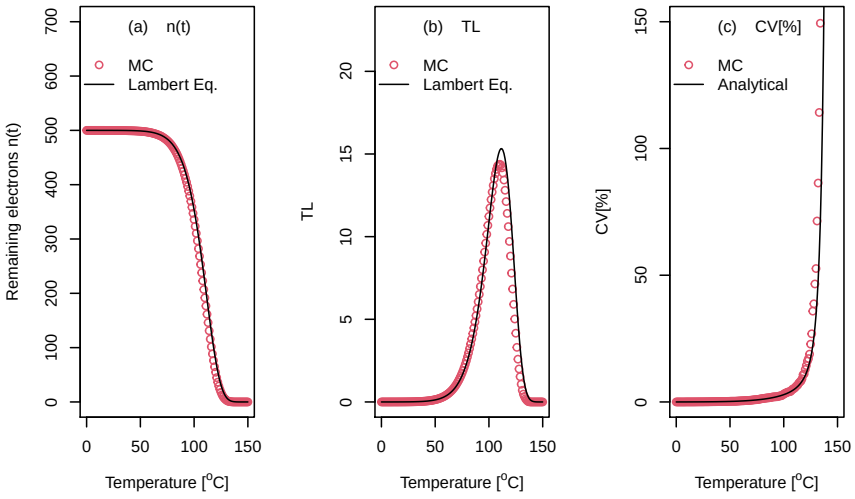

Fig. 8.7: Vectorized MC implementation of the first order TL luminescence process. (a) Plot of $< n(t) >$ for M =100 MC runs with the same initial number of electrons n_0=500, simulating a total of 50,000 electrons; (b) Average of the corresponding TL signal. (c) The corresponding *CV[%]*. The solid lines represent the analytical equations. For details, see Pagonis et al. [40].

Code 8.6: Vectorized MC implementation of LM-OSL

```
rm(list = ls(all=T))
options(warn=-1)
library(matrixStats)
mcruns<-100
n0<-500
tmax<-60
A<-0.2
deltat<-1
times<-seq(1,tmax,deltat)
nMatrix<-LMMatrix<-matrix(NA,nrow=length(times),ncol=mcruns)
```

```
nMC<-LM<-rep(NA,length(times))
system.time(
for (j in 1:mcruns){
  n<-n0
  for (t in 1:length(times)){
    vec<-rep(runif(n))
    P<-deltat*t*A/tmax
    n<-length(vec[vec>P])
    nMC[t]<-n
    LM[t]<-n*P}
nMatrix[,j]<-nMC
LMMatrix[,j]<-LM
 })
#Find average of n(t),LM-OSL signal and CV[%]
avgn<-rowMeans(nMatrix)
avgLM<-rowMeans(LMMatrix)
sd<-rowSds(LMMatrix)
cv<-100*sd/avgLM
par(mfrow=c(1,3))
pch<-c(NA,NA,1,NA)
lty<-c(NA,NA,NA,"solid")
col<-c(NA,NA,2,1)
plot(times,avgn,ylab="Remaining electrons n(t)",
xlab="Time [s]",ylim=c(0,700),col=2)
curve(n0*exp(-A*x^2/(2*tmax)),0,tmax,add=TRUE,
col=1,lwd=2)
legend("topright",bty="n",c("(a)    ",
" ","MC","Analytical"),
pch=pch,lty=lty,col=col)
plot(times,avgLM,ylab="LMOSL",xlab="Time [s]",
col=2,ylim=c(0,24))
legend("topright",bty="n",c("(b)    LM-OSL"," ","MC",
"Analytical"), pch=pch,lty=lty,col=col)
curve(A*n0*exp(-A*x^2/(2*tmax))*x/tmax,0,tmax,add=TRUE,
col=1,lwd=2)
plot(times,cv,ylab="CV[%]",xlab="Time [s]",ylim=c(0,150),
col=2)
curve(100*sqrt((exp(A*x^2/(2*tmax))-1)/n0),0,tmax,add=TRUE,
col=1,lwd=2)
legend("topleft",bty="n",c("(c)    CV[%]"," ","MC (M=100)",
 "Analytical"), pch=pch,lty=lty,col=col)
```

```
##    user  system elapsed
##    0.08    0.00    0.07
```

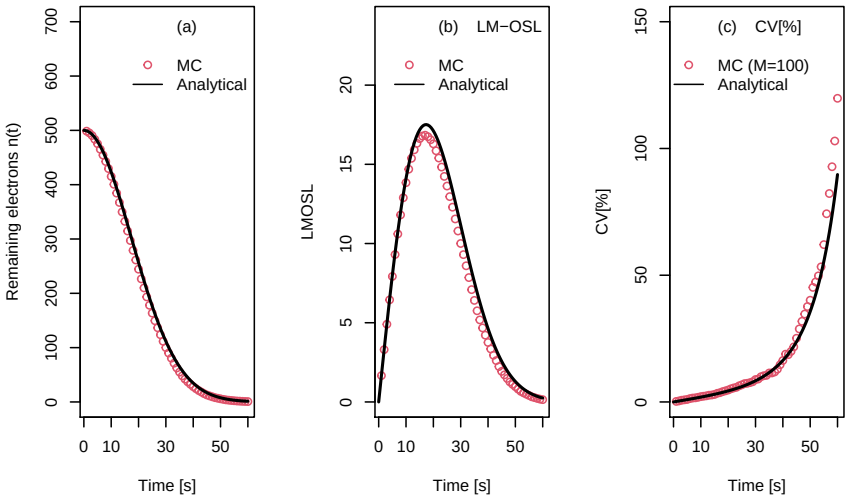

Fig. 8.8: Vectorized MC implementation of the first order LM-OSL luminescence process. (a) Plot of the mean $< n(t) >$ for M =100 MC runs with the same initial number of electrons n_0=500, simulating a total of 50,000 electrons. (b) Average of the corresponding LM-OSL signal. (c) The corresponding *CV[%]*. The solid lines represent the analytical equations from Chapter 3.

Code 8.7: Vectorized MC code for TL in GOT model

```
# GOT MODEL- Monte Carlo code for TL
rm(list = ls(all=T))
options(warn=-1)
library(matrixStats)
library(lamW)
mcruns<-100
n0<-500
N<-1000
s<-1e12
E<-1
R<-0.6
```

```
kb<-8.617e-5
tmax<-200
deltat<-1
times<-seq(1,tmax,deltat)
nMatrix<-TLMatrix<-matrix(NA,nrow=length(times),ncol=mcruns)
nMC<-TL<-rep(NA,length(times))
system.time(
for (j in 1:mcruns){
  n<-n0
  for (t in 1:length(times)){
    vec<-rep(runif(n))
    P<-s*exp(-E/(kb*(t+273)))*n/((N-n)*R+n)
    n<-length(vec[vec>P])
    nMC[t]<-n
    TL[t]<-n*P}
nMatrix[,j]<-nMC
TLMatrix[,j]<-TL
 })
#Find average of n(t), average TL signal and CV[%]
avgn<-rowMeans(nMatrix)
avgTL<-rowMeans(TLMatrix)
sd<-rowSds(TLMatrix)
cv<-100*sd/avgTL
# plots
par(mfrow=c(1,3))
pch<-c(NA,NA,1,NA)
lty<-c(NA,NA,NA,"solid")
col<-c(NA,NA,2,1)
k<-function(u) {integrate(function(p){exp(-E/(kb*p))},
300,u)[[1]]}
x1<-300:450
y1<-lapply(x1,k)
x<-unlist(x1)
y<-unlist(y1)
c<-(n0/N)*(1-R)/R
zTL<-(1/c)-log(c)+(s*n0/(c*N*R))*y
plot(times,avgn,ylab="Remaining electrons n(t)",
col=2,xlab=expression("Temperature ["^"o"*"C]"),ylim=c(0,700))
lines(x-273,(N*R/(1-R))/(lambertW0(exp(zTL))),col=1)
legend("topright",bty="n",c("(a)    n(t)"," ","MC",
 "Lambert Eq."),pch=pch,lty=lty,col=col)
plot(times,avgTL,ylim=c(0,14),ylab="TL",
col=2,xlab=expression("Temperature ["^"o"*"C]"))
# plots
```

```
lines(x-273,(N*R/((1-R)^2))*s*exp(-E/(kb*x))/
(lambertW0(exp(zTL))+lambertW0(exp(zTL))^2),col=1)
legend("topleft",bty="n",c("(b)    TL"," ","MC",
  "Lambert Eq."), pch=pch,lty=lty,col=col)
plot(times,cv,ylab="CV[%]",ylim=c(0,120),
col=2,xlab=expression("Temperature ["^"o"*"C]"))
legend("topleft",bty="n",c("(c)    CV[%]"," ","MC"))

  ##    user  system elapsed
  ##    0.36    0.00    0.37
```

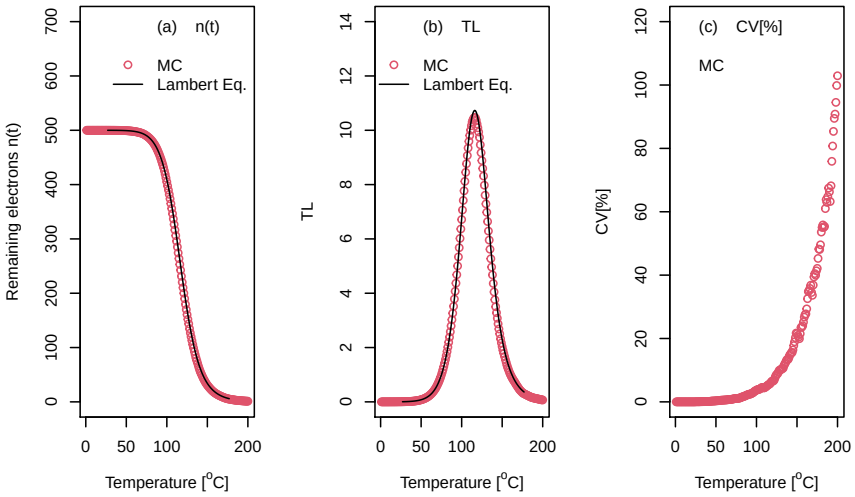

Fig. 8.9: Vectorized MC simulation of TL in a system of $M = 100$ large clusters of defects, based on the GOT equation. The total number of traps in each cluster is $N = 1000$, and initially $n_0 = 500$ of these traps are filled. (a) The average $n(t)$ (b) The average TL signal; (c) The corresponding *CV[%]*. The solid lines in (a) and (b) represent the analytical equations which are based on the Lambert function. For details, see Pagonis et al. [40].

F

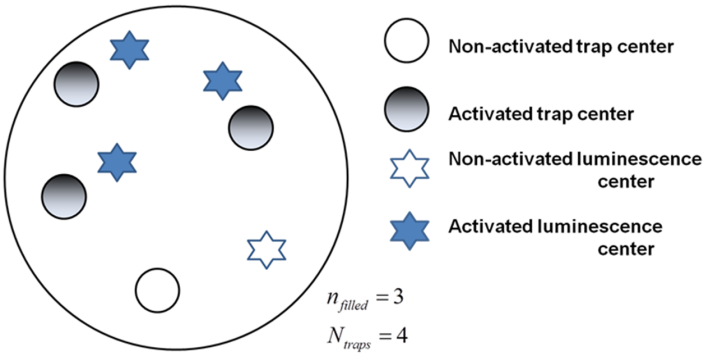

Fig. 8.10: Schematic representation of a small trap cluster, consisting of a total of four traps in each cluster ($N_{traps} = 4$ shown as both open and solid circles). Only three of these traps are initially filled ($n_{filled} = 3$ shown as solid circles). Charge balance in the system is ensured by assuming the existence of an equal number of four luminescence centers (shown as both open and solid stars), three of which have been activated (shown as solid stars). The solid is assumed to consist of a large number of clusters (e.g. $N_{clusters} = 10^5$). For a detailed description, see Pagonis et al. [34].

F

Table 8.2: Listing of local and global variables used in the simulations of luminescence from small clusters, and their typical values (see also Fig.8.10 for a pictorial presentation of the local variables). From Pagonis et al. [34].

Variable type	**Description**	**Typical Value**
Local		
$n_{local}(t)$	The number of remaining filled traps in the cluster.	3
n_{filled}	The number of initially filled traps per cluster. ($n_{filled} \leq N_{traps}$)	3
N_{traps}	The total number of traps per cluster	4
Global		
$n(t)$	The total number of remaining filled traps in the system. This is calculated by summing over all clusters	3×10^5
n_0	The total number of initially filled traps in the system. n_o is found from $n_o = N_{clusters} \times n_{filled}$ (with $n_o \leq N$)	3×10^5
$N_{clusters}$	The number of trap clusters in the system.	10^5

F

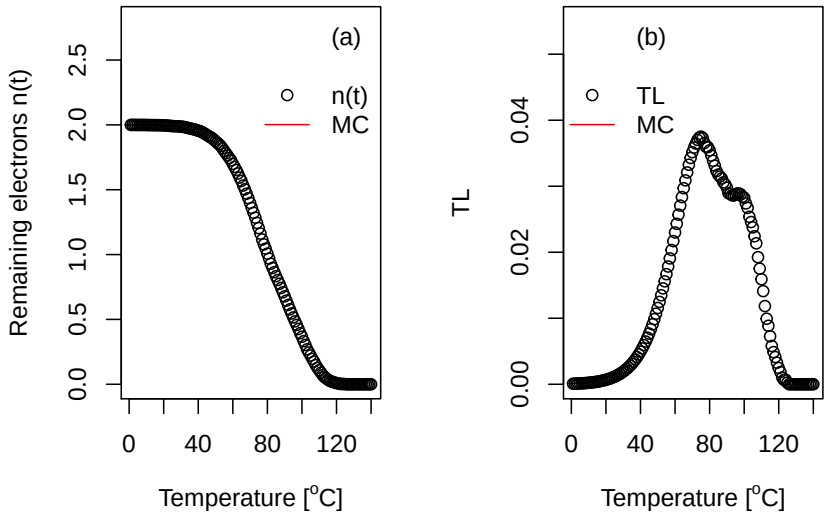

Fig. 8.11: Simulation of TL signal from a system of $M = 3000$ small clusters of defects, based on the GOT equation. The total number of traps in each cluster is $N = 3$, and initially $n_0 = 2$ of these traps are filled. The double peak structure is caused by the large retrapping ratio $R = 100$. For a more detailed description, see Pagonis et al. [34].

We now consider a MC simulation of the irradiation process within the OTOR/GOT model.

8.1 Irradiation process as a nonlinear pure birth problem

In this section we provide an example of MC simulation for the irradiation of a dosimetric material, within the OTOR model.

The irradiation process can be described by the following differential equation, which is derived by applying the QE conditions to the irradiation stage of the OTOR model (Lawless et al. [20], their Eq.7):

$$\frac{dn}{dt} = \frac{(N-n)R}{(N-n)R+n}X \tag{8.1}$$

The symbols in this equation are the same as in the OTOR system discussed so far in this book, with the additional symbol of X ($cm^{-3}s^{-1}$) representing the rate of production of electron-hole pairs in the system, per unit volume and per unit of time. As usual, R is the dimensionless retrapping ratio in the OTOR model, such that $R = A_n/A_m$, and $n(t), N$ are the instantaneous occupancy and total concentration of traps in the sample. The quantity $\alpha = \frac{(N-n)R}{(N-n)R+n}$ on the right hand side of Eq.(8.1) is dimensionless, and is also < 1. This quantity represents the ratio of the concentration of trapped electrons $(N-n)R$, over the total concentration $(N-n)R+n$ that they will either be retrapped or lost in the recombination centers during the irradiation process.

As we saw in Chapter 4, the analytical solution of this equation is found in terms of the Lambert function W:

$$n/N = 1 + W[(R-1)\exp(R-1-RXt/N)]/(1-R) \tag{8.2}$$

The MC simulations simplify by dividing both sides of Eq.(8.1) by X, to obtain:

$$\frac{dn}{d(Xt)} = \frac{(N-n)R}{(N-n)R+n} \tag{8.3}$$

We now change our time parameter from t to the new time parameter $D = Xt$, which has units of concentration (cm^{-3}) and which is proportional to the dose received by the sample. The previous equation in stochastic form becomes:

$$\Delta(n) = \frac{(N-n)R}{(N-n)R+n}\Delta D \tag{8.4}$$

The following R code solves this difference equation with the parameters $N = 10^{10}$, $X = 10^5\ s^{-1}$, $R = A_n/A_m = 1.2$, $n_0 = 10^5$ and the results are shown in Fig.8.12.

The plot of $n(t)$ in Fig.8.12a shows good agreement between the MC code and the solution of the differential equation for the irradiation process (see the detailed discussion in Chapter 4). Fig.8.12b shows the uncertainty σ_n, and Fig.8.12c shows the corresponding of *CV[%]* as a function of irradiation dose D. As the irradiation proceeds, the value of the stochastic coefficient CV[%] decreases continuously with irradiation time from about 30% to a value of about 1%.

There are no analytical solutions for the σ_n and the *CV[%]* results in this example, since the corresponding differential equation is non-linear.

Code 8.8: Vectorized Irradiation MC code in GOT model

```
#Vectorized Irradiation MC code in GOT model
rm(list = ls(all=T))
options(warn=-1)
library(matrixStats)
library(lamW)
mcruns<-100
deltat<-1
tmax<-1000
Dvalues<-seq(1,tmax,deltat)
nMatrix <-  matrix(NA, nrow = length(Dvalues), ncol = mcruns)
nMC<-rep(NA,length(Dvalues))
N<-100
n0<-1
R<-1.2
system.time(
  for (k in 1:mcruns){
    n<-n0                #initialize each of the M=100 MC runs
    for (t in Dvalues){
      vec<-rep(runif(n))    #create a vector vec,
      #containing n random numbers between 0 and 1
      P<-R*(N-n)*(1/n)/(R*(N-n)+n)*deltat
      dn<-length(vec[vec<P]) # if the random # in vec is <P,
      n<-n+dn #then increase the number of filled traps
      nMC[t]<-n+dn } #store number of electrons n in vector nMC
  nMatrix[,k]<-nMC # store single MC run in column k of nMatrix
  })
#Find average of n(t), CW-OSL signal, and CV[%]
avgn<-rowMeans(nMatrix)
sd<-rowSds(nMatrix)
cv<-100*sd/avgn
par(mfrow=c(1,3))
pch<-c(NA,NA,1,NA)
lty<-c(NA,NA,NA,"solid")
xlabs=expression("D [cm"^-3*"]")
plot(Dvalues,avgn,ylab="Remaining electrons n(t)",
     xlab=xlabs,ylim=c(0,140))
legend("topright",bty="n",c("(a)    n(t)"," ","MC",
                            "Lambert Eq."),pch=pch,lty=lty)
lines(Dvalues,N*(1+lambertW0((R-1)*exp(R-1-R*Dvalues/N))/(1-R)),
      col="red",lwd=3)
plot(Dvalues,sd,ylab=c(expression(sigma[n]*" "),"  ","MC"),
xlab=xlabs, ylim=c(0,8),col="blue")
legend("topleft",bty="n",legend=c(expression("(b)","  ",
sigma[n]*" ")),pch=pch,lty=lty,col="blue")
```

```
plot(Dvalues,cv,ylab="CV[%]", xlab=xlabs,ylim=c(0,60),col="red")
legend("topleft",bty="n",c("(c)  CV[%]"," ","MC"),pch=pch)

  ##    user  system elapsed
  ##    0.76    0.00    0.78
```

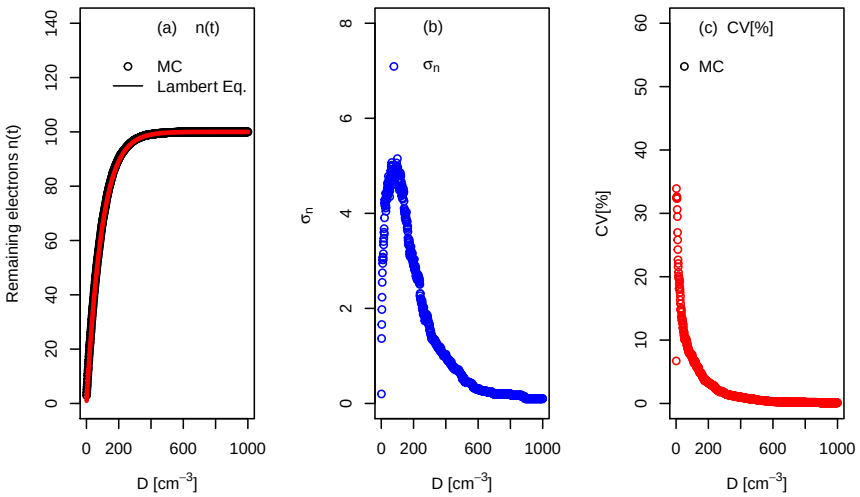

Fig. 8.12: MC simulation of irradiation process in the GOT model, as a nonlinear pure birth problem. The parameters are $n_0 = 1$, $M = 100$ MC runs, $R = 1.2$, $N = 100$. (a) Plot of the trapped electrons $n(t)$. The solid line is the Lambert W solution discussed in Chapter 4. (b) Plot of the uncertainty σ_n for the population $n(t)$. (c) Plot of the corresponding *CV[%]*. There are no analytical expressions for (b) and (c).