Chapter 1
INTRODUCTION TO LUMINESCENCE SIGNALS AND MODELS

Abstract In this introductory chapter we introduce two types of thermally and optically stimulated luminescence signals, which are commonly used for luminescence dosimetry and luminescence dating. Phenomena like thermoluminescence (TL), optically stimulated luminescence (OSL) and radioluminescence (RL), usually take place in a time scales of seconds, while time-resolved (TR) luminescence phenomena usually take place in a ms or μs scale. We provide an overview of commonly used luminescence models, based on delocalized and localized transitions, and discuss optical absorption (OA) and Electron Spin Resonance experiments (ESR), and their connection and importance in luminescence dosimetry. This chapter concludes with a brief discussion of what types of information researchers typically extract from the experimental data described in this chapter.

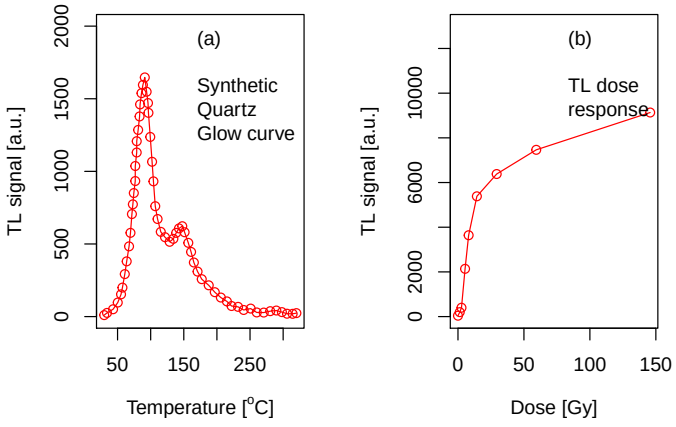


Fig. 1.1: (a) Example of a TL glow curve for synthetic quartz, after irradiation with a beta dose of 2 Gy. (b) The TL dose response of this quartz sample, obtained by plotting the maximum intensity of the TL signal as a function of the irradiation dose. For more details see Kitis et al. [15].

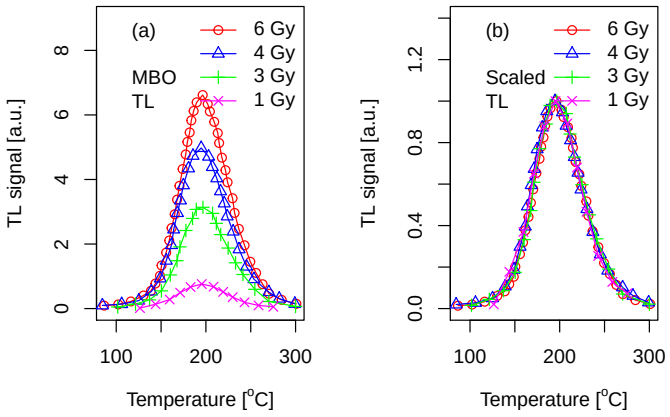


Fig. 1.2: (a) Example of a series of TL glow curves for sample MBO, at different beta doses. (b) The data in (a) is normalized to the maximum TL height. For more details see Pagonis et al. [32].

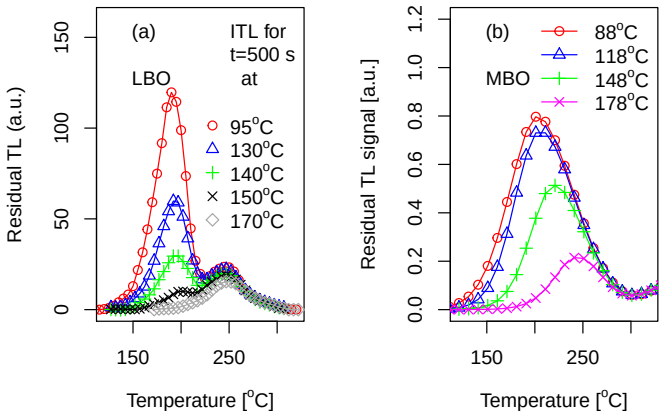


Fig. 1.3: (a) Example of a series of TL glow curves for sample LBO, after irradiation and heating up to the temperatures indicated in the legends, and (b) A very similar series of TL glow curves for sample MBO. The different behaviors of these two samples indicate that different luminescence mechanisms are involved in each case. For more details see Kitis et al. [18].

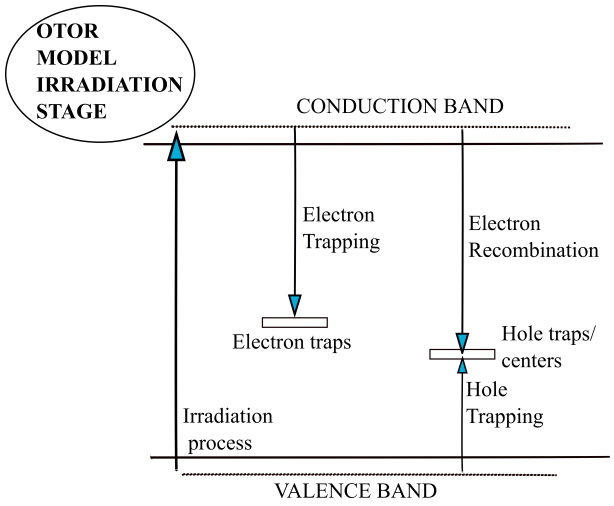


Fig. 1.4: The simplest OTOR model, showing the various electronic transitions during the irradiation stage. Irradiation creates electrons and holes in the conduction and valence bands, respectively. Electrons in the conduction band can subsequently either be trapped in electron traps, or they can recombine with holes at the recombination centers/hole traps. Holes in the valence band can be trapped in the same recombination centers/hole traps.

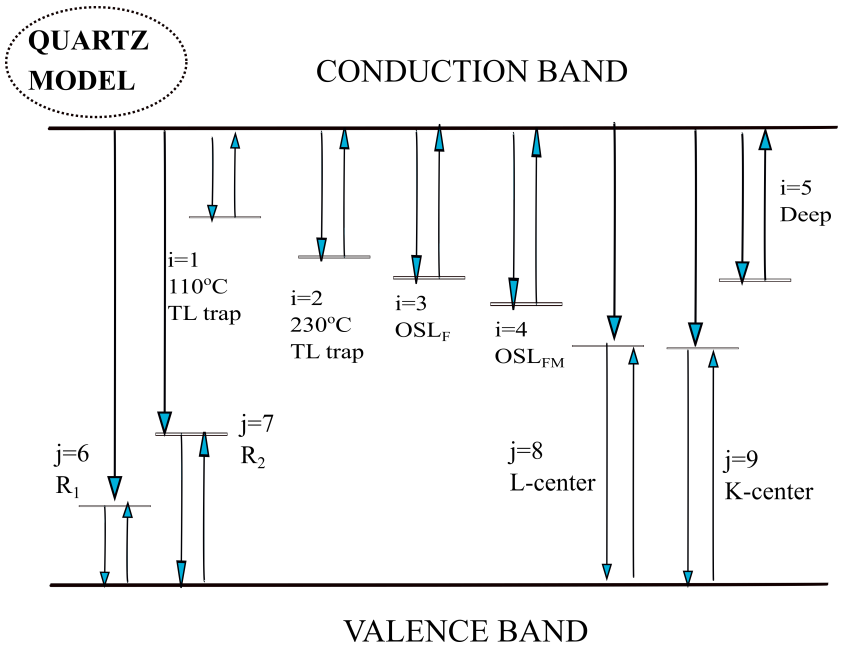


Fig. 1.5: Example of a complex *delocalized model* for quartz, involving multiple traps and centers (after Bailey [1]). Each arrow represents a transition between energy levels and the conduction and valence bands. These arrows also represent a mathematical term in a system of differential equations, as discussed in Chapter 11.

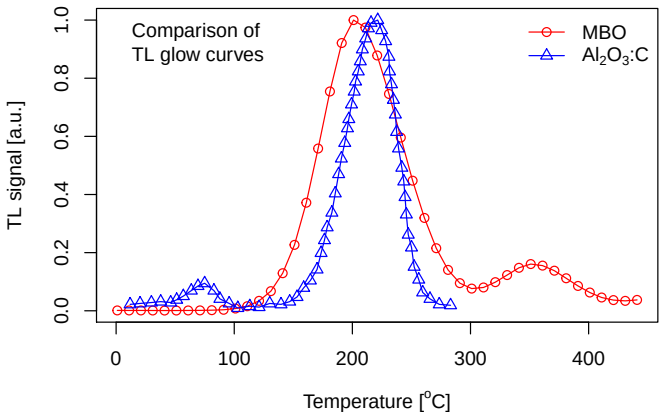


Fig. 1.6: Comparison of the TL glow curves MBO and Al_2O_3:C. The main dosimetric peak around 200°C for MBO is very nearly symmetric, while for Al_2O_3:C the main peak is asymmetric. Notice also the difference between the widths of the main dosimetric peaks in the two materials. For more details see Pagonis et al. [28] and from Kitis et al. [18].

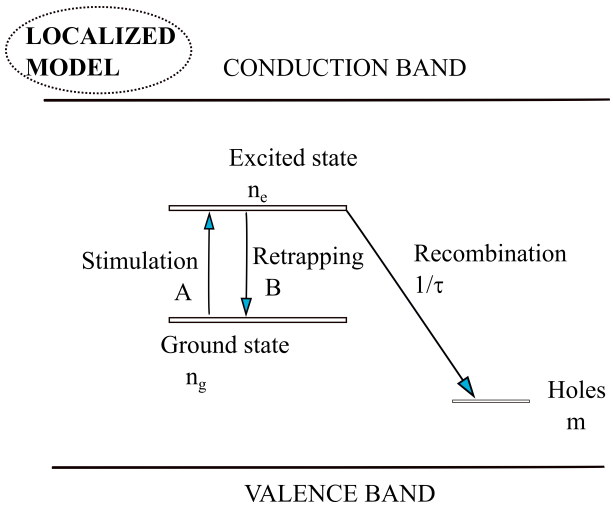


Fig. 1.7: A typical *localized transition model*, consisting of the ground state and the excited state of a trapped electron, and an additional energy level associated with a recombination center. Electrons are excited from the ground state (n_g) into the excited state (n_e) of the trap at a rate A, and can also relax back into the ground state of the trap at the rate B. The transition indicated by the rate $1/\tau$ occurs from the excited state of the trap to an energy level of the recombination center (after Pagonis et al. [36]).

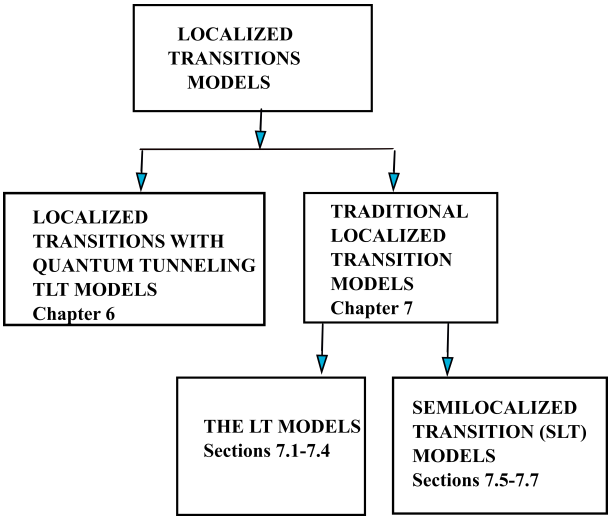


Fig. 1.8: The organization of localized transition models in chapters 6 and 7 of this book.

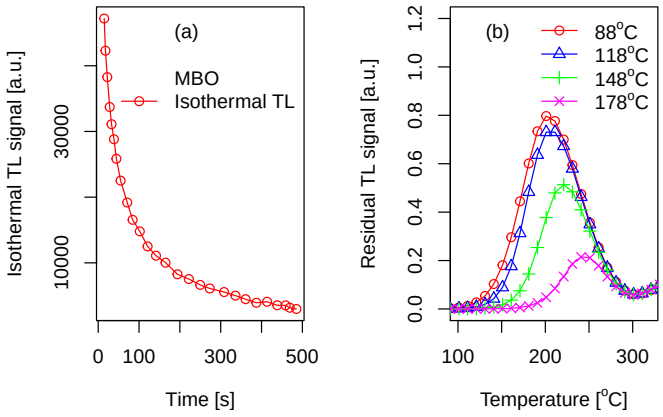


Fig. 1.9: (a) Isothermal TL signal for sample MBO and (b) a series of RTL glow curves, measured after an isothermal TL experiment for 500 s at the indicated temperatures. For more details see Kitis et al. ([18]).

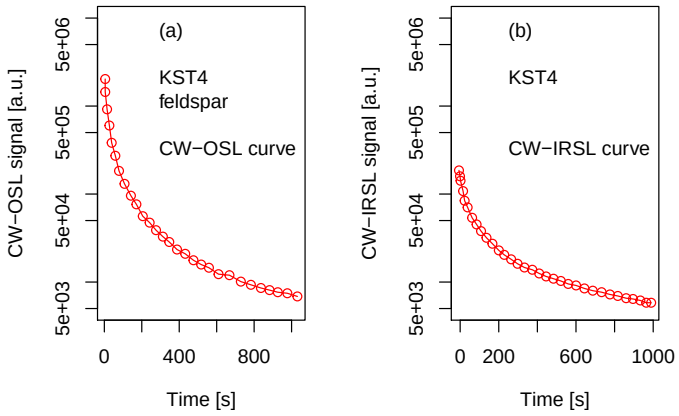


Fig. 1.10: Examples of (a) A CW-OSL curve and (b) CW-IRSL curve, from the same geological feldspar sample KST4. For more details see Kitis et al. [17].

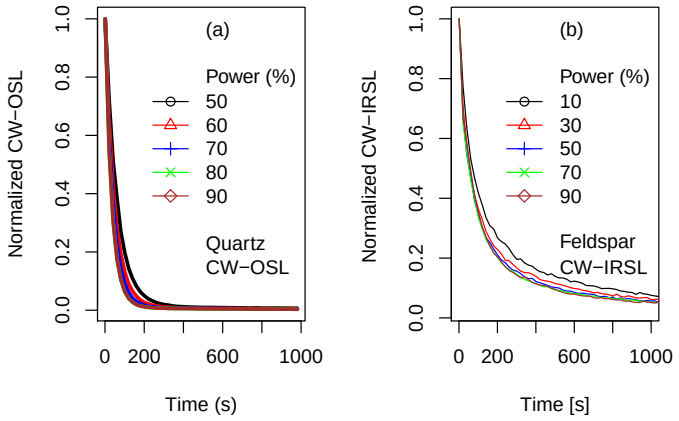


Fig. 1.11: (a) CW-OSL for quartz at different illumination powers in the range 50-90% of maximum power. (b) CW-IRSL signal at different illumination powers in the range 50-90% of maximum power, for a feldspar sample (laboratory code KST4). Both sets of data have been normalized to the maximum intensity. The shape of both signals changes with the illumination power. For more details see Pagonis et al. [40] and Polymeris [48].

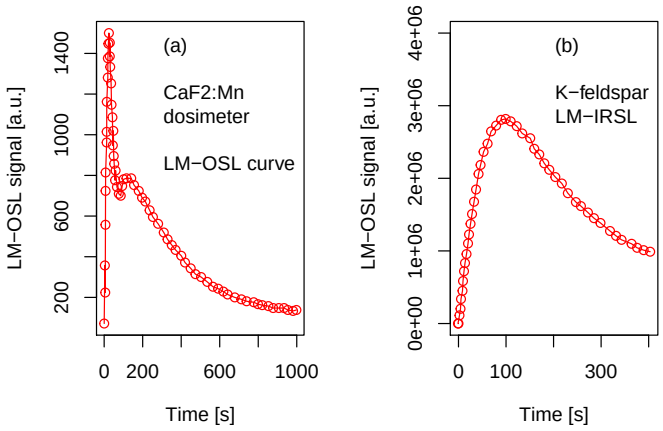


Fig. 1.12: Examples of (a) LM-OSL curve for the dosimetric material CaF_2:N (Kitis et al. [16]). (b) LM-IRSL for a K-feldspar. For more details see Bulur and Göksu [6].

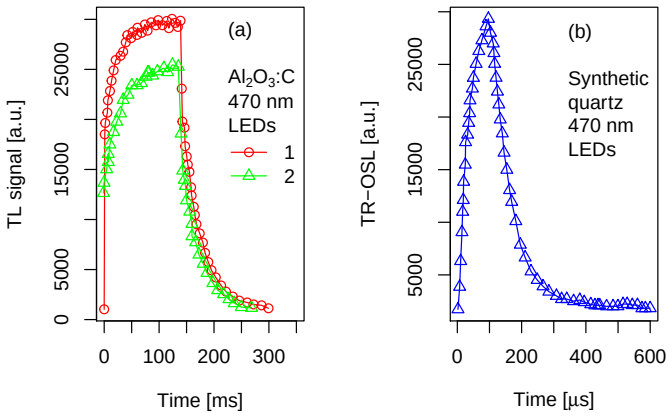


Fig. 1.13: Examples of TR-OSL curves for (a) Al_2O_3:C and (b) High purity synthetic quartz. Notice the very different time scales on the horizontal axis. The curves labeled 1 and 2 in (a) represent repeated measurements on the same sample, after the trapped electrons have been partially depleted. For more details see Pagonis et al. [43].

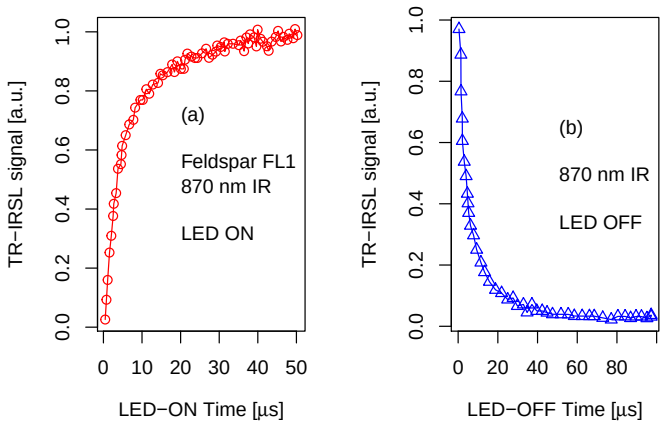


Fig. 1.14: Examples of TR-IRSL data for a feldspar sample FL1. (a) The TR-IRSL intensity as a function of the stimulation time during a 50 μs LED ON pulse. (b) The TR-IRSL intensity as a function of the stimulation time during the 100 μs LED OFF period. For more details see Pagonis et al. [29].

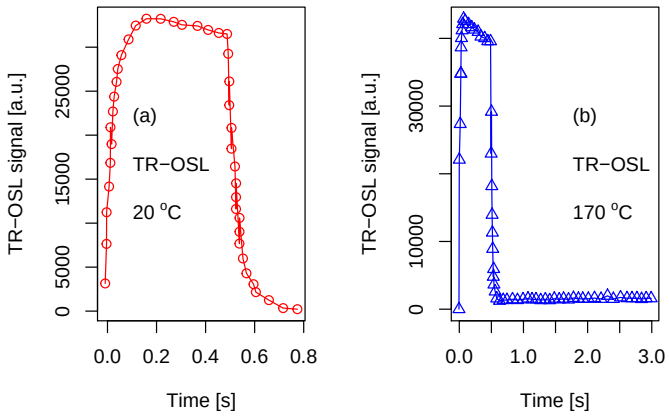


Fig. 1.15: Examples of TR-OSL data for $Al_2O_3 : C$ (a) At room temperature 20°C (b) at a stimulation temperature of 170°C. For more details see Pagonis et al. [28].

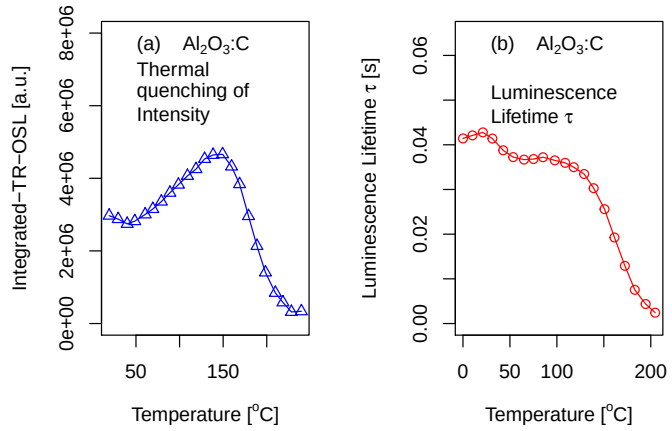


Fig. 1.16: Examples of TR-OSL data for Al_2O_3:C, showing the phenomenon of thermal quenching. (a) The integrated TR-OSL intensity as a function of the stimulation temperature. (b) The luminescence lifetime parameter τ as a function of the stimulation temperature. For more details see Pagonis et al. [28].

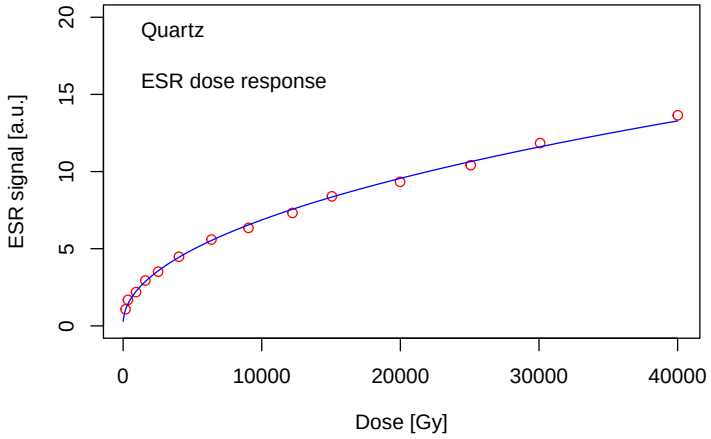


Fig. 1.17: Example of the dose response of ESR data from quartz. Redrawn from Pagonis et al. [38], original data from Duval [9].

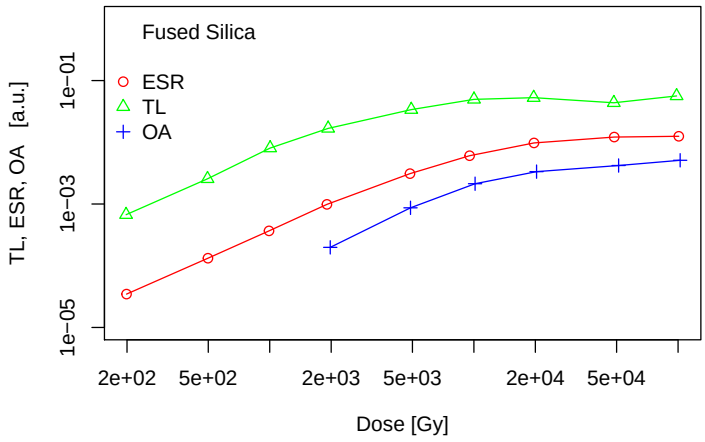


Fig. 1.18: Measurements of TL, ESR and OA signals as a function of the irradiation dose, from a single sample of fused silica. Note the log scale in both axes. For more details see Pagonis et al. [39], original data from Wieser et al. [53].

Part I
LUMINESCENCE SIGNALS FROM DELOCALIZED TRANSITIONS

Chapter 2
ANALYSIS AND MODELING OF TL DATA

Abstract In this chapter we provide detailed R codes which show how researchers can analyze and model their experimental TL data. We provide R codes for the initial rise method and the method of various heating rates, which allow evaluation of both the activation energy E and the frequency factor s. We present R codes for numerically integrating the simple one trap one recombination model (OTOR), as well as for numerically integrating the equations for first, second and general order kinetics using R. We discuss the general one trap (GOT) differential equation and its analytical solution, which is based on the Lambert W function. Several examples are given for using computerized glow curve deconvolution analysis (CGCD) for single-peak and multiple-peak TL glow curves, based on the R-packages *tgcd* and the Lambert W function. Specific examples are given of using the new R package *RLumCarlo*, to simulate TL glow curves with different kinetic parameters. The chapter concludes with a list of recommended experimental protocols, which experimentalists can apply when studying TL signals.

Code 2.1: System of differential equations for OTOR

```
# Solution of the system of ODE's for the OTOR model
rm(list=ls())
library(deSolve)
TLOTOR <- function(t, x, parms) {
  with(as.list(c(parms, x)), {
    dn1 <- - n1*s*exp(-E1/(kb*(273+hr*t)))+ nc*An*(N1-n1)
```

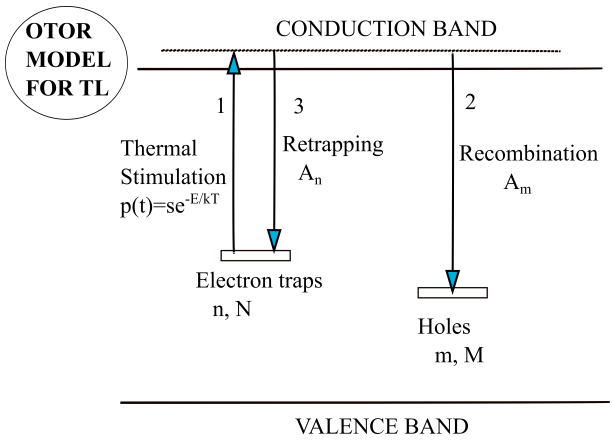


Fig. 2.1: Schematic diagram of the OTOR model for a TL process.

```
    # n1=concentration of trapped electrons
    dnc <-  n1*s*exp(-E1/(kb*(273+hr*t)))-nc*An*(N1-n1)-m*Am*nc
    # nc=concentration of conduction band electrons
    dm <- -m*Am*nc
    # m=concentration of recombination centers
    res <- c(dn1, dnc, dm)
    list(res)
  })}
## Parameters
hr<-1   # heating rate in K/s
parms  <- c(E1 =1, s=10^12, kb=8.617*10^-5, hr=hr,
            An = 10^-7,  N1 = 10^10,Am = 10^-7)
## vector of timesteps
times  <- seq(0, 250)
temps<-times*hr
## Initial conditions for the system
y <- xstart <- c(n1 = 10^9, nc = 0, m = 10^9)
## Solve system of differential equations
out <-  lsoda(xstart, times, TLOTOR, parms)
## Plotting
par(mfrow=c(1,2))
plot(temps,out[,"n1"],xlab=expression("Temperature ["^"o"*"C]"),
ylab =expression("Filled traps n(T),cm"^-3*" "),ylim=c(0,1.2e9))
legend("left",bty="n",expression("OTOR","n(T)"))
legend("topleft",bty="n", expression("(a)"))
plot(temps,out[,"m"]*parms["Am"]*out[,"nc"],pch=1,col="red",
xlab=expression("Temperature ["^"o"*"C]"),
```

```
ylim=c(0,2.8e7),ylab="TL [a.u.] and nc (t)",xlim=c(0,250))
lines(temps,10*out[,"nc"],xlab=expression("Temperature [
"^"o"*"C]"),ylab="nc(t)",typ="o",pch=2,col="blue")
legend("topright",bty="n", expression("(b)"))
legend("topleft",bty="n", pch=c(1,2),expression("TL","ncx10"),
col=c("red","blue"),lwd=1)
```

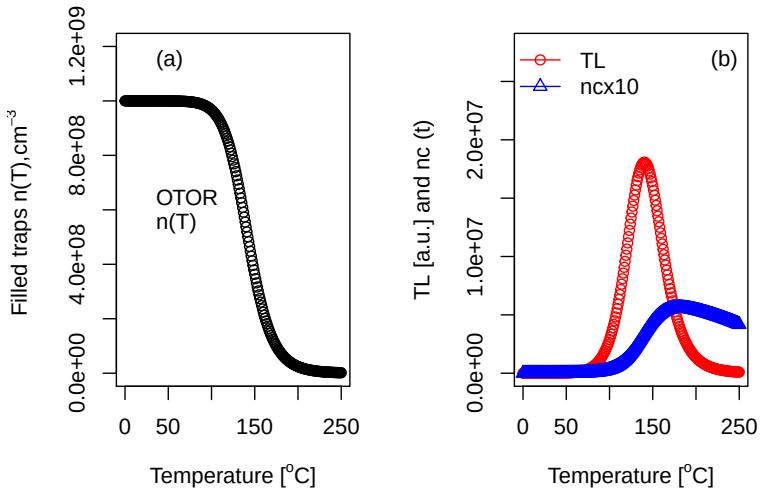


Fig. 2.2: Numerical solution of the system of differential equations for the OTOR model. (a) The concentrations of trapped electrons $n(T)$ and (b) of the conduction band electrons $n_c(T)$ (triangles), and luminescence intensity $I(T)$ (circles), during a typical TL experiment. Note that $n_c(T)$ has been multiplied by a factor of 10, for display purposes.

Code 2.2: ODE for TL: First order kinetics

```
# Numerically solve the ODE for TL: first order kinetics
rm(list=ls())
library(package ="deSolve")
```

```
# Define Parameters
k_B <- 8.617e-5  # Boltzmann constant
E <- 1 # electron trap depth [eV]
s <- 1e12 # frequency factor [1/s]
delta.t <- 1
t <- seq(0, 200, delta.t)
n.0 <- 1e10
N.traps <- 1e11
ODE <- function(t, state, parmameters){
        with(as.list(c(state, parameters)),{
        dn <-  -s*exp(-E/(k_B*(273+t))) * n
    list(c(dn)) })}
parameters <- c(N.traps = N.traps, s = s, E = E, k_B = k_B)
state <- c(n = n.0)
num_ODE <- ode(y = state, times = t, func = ODE,
parms = parameters)
# Plot remaining electrons and TL as a function of temperature
par(mfrow=c(1,2))
plot(x =num_ODE[,1],
y = num_ODE[,2], xlab=expression("Temperature ["^"o"*"C]"),
 ylab = expression("Filled traps n(T), cm"^-3*" "),
col = "red")
legend("topright",bty="n",c("(a)","n(T)"))
plot(num_ODE[-1,1], y = abs(diff(num_ODE[,2])),
xlab=expression("Temperature ["^"o"*"C]"),
ylab = "TL signal [a.u.]",xlim=c(0,170))
legend("topleft",bty="n",c("(b)","TL"))
```

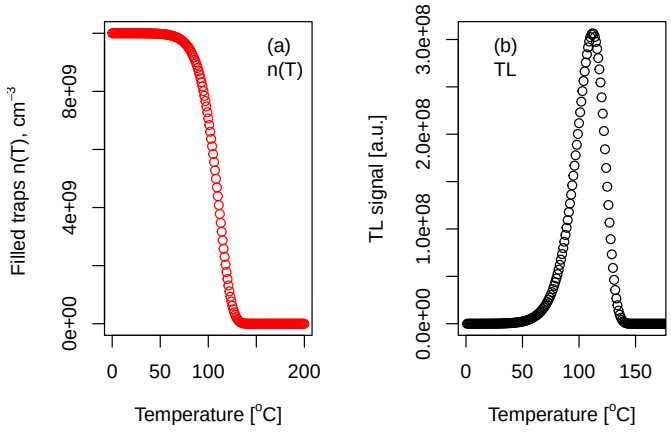


Fig. 2.3: Numerical solution of the Randall and Wilkins Eq.(**??**), for first order kinetics TL. (a) Remaining trapped electrons $n(T)$, and (b) The corresponding TL signal.

Code 2.3: First-order TL by varying the initial trap concentrations (tgcd)

```
# Simulate first-order glow peaks with various
# initial electron trap concentrations (n0).
rm(list=ls())
# library(tgcd)
library(package ="tgcd")
temps <- seq(300, 440, by=1)
peak1 <- simPeak(temps,n0=0.2e10,Nn=1e10,ff=1e12, ae=1.0, hr=1,
                 typ="f",plot=FALSE)
peak2 <- simPeak(temps,n0=0.4e10,Nn=1e10,ff=1e12, ae=1.0, hr=1,
                 typ="f",plot=FALSE)
peak3 <- simPeak(temps,n0=0.6e10,Nn=1e10,ff=1e12, ae=1.0, hr=1,
                 typ="f",plot=FALSE)
peaks<-cbind(peak1$tl, peak2$tl, peak3$tl)
matplot(temps, peaks, type="o",  pch=c(0,1,2),
        col=c("black","red","blue"),lwd=2,
        xlab="Temperature (K)", ylab="TL intensity [a.u.]")
legend("topleft",bty="n", pch=c(NA,NA,NA,0,1,2),
       c(expression('First order peaks',' ','n'[o]*'/N'),
"0.2","0.4","0.6"),col=c(NA,NA,NA,"black","red","blue"))
```

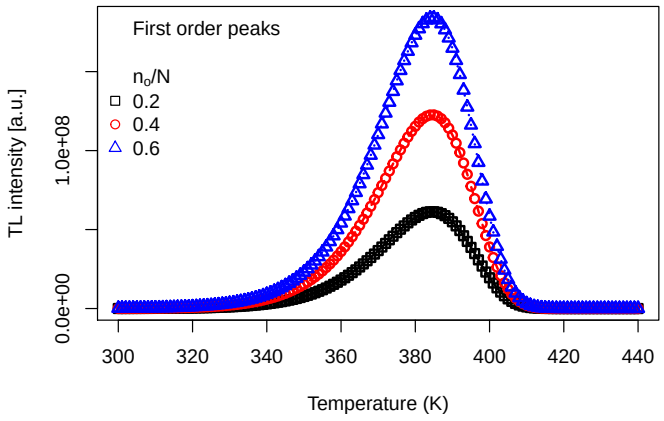


Fig. 2.4: Simulation of first-order TL glow peaks with three different initial electron trap concentrations $n_0 = 0.2\,, 0.4,\ 0.6 \times 10^{10}\ cm^{-3}$. The location of the maximum TL intensity does not shift significantly as n_0 changes, and the peak height is proportional to n_0.

Code 2.4: Second-order TL by varying the initial trap concentrations (tgcd)

```
# Simulate second-order glow peaks with various
# initial electron trap concentrations (n0).
rm(list=ls())
library(tgcd)
temps <- seq(300, 500, by=1)
peak1 <- simPeak(temps,n0=0.2e10,Nn=1e10,ff=1e12, ae=1.0, hr=1,
typ="s",plot=FALSE)
peak2 <- simPeak(temps,n0=0.4e10,Nn=1e10,ff=1e12, ae=1.0, hr=1,
typ="s",plot=FALSE)
peak3 <- simPeak(temps,n0=0.6e10,Nn=1e10,ff=1e12, ae=1.0, hr=1,
typ="s",plot=FALSE)
peaks<-cbind(peak1$tl, peak2$tl, peak3$tl)
matplot(temps, peaks, type="o",  pch=c(0,1,2),
        col=c("black","red","blue"),lwd=2,
        xlab="Temperature (K)", ylab="TL intensity [a.u.]")
```

```
legend("topright",bty="n", pch=c(NA,NA,NA,0,1,2),
       c(expression('Second order peaks',' ','n'[0]*'/N'),
"0.2","0.4","0.6"),col=c(NA,NA,NA,"black","red","blue"))
```

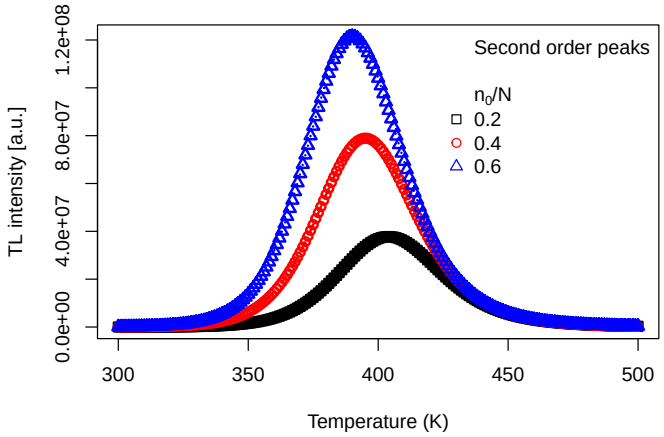


Fig. 2.5: Simulation of second order TL glow peaks with three different initial electron trap concentrations $n_0 = 0.2\,, 0.4,\ 0.6 \times 10^{10}\ cm^{-3}$. The location of the maximum TL intensity shifts with n_0, and the *area* under the peak is proportional to n_0.

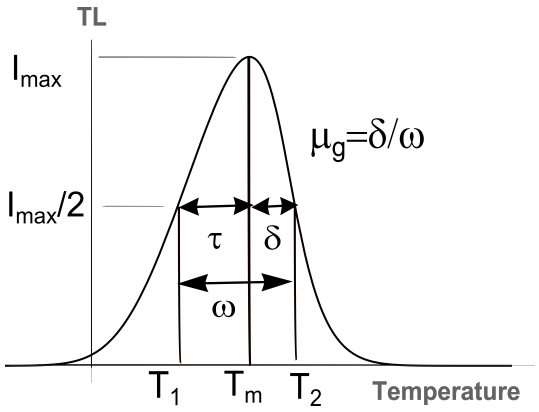


Fig. 2.6: Schematic diagram of TL peak. The half-widths τ, δ, ω define the geometrical shape factor $\mu_g = \delta/\omega$. First-order peaks have an asymmetric form, while second-order peaks are very nearly symmetric.

Code 2.5: 1st and 2nd order TL with the same parameters

```
# Simulate a first and a second  order TL glow peak
# with the same kinetic parameters
rm(list=ls())
library(package ="tgcd")
temps <- seq(300, 500, by=.2)
peak1 <- simPeak(temps, n0=1e10, Nn=1e10, ff=1e12, ae=1, hr=1,
                 typ="f",plot=FALSE)
peak2 <- simPeak(temps, n0=1e10, Nn=1e10,  ff=1e12, ae=1, hr=1,
                 typ="s",plot=FALSE)
n<-cbind(peak1$n, peak2$n)
par(mfrow=c(1,2))
matplot(temps, n, type="l", lwd=3,lty=c(1,2),
xlab="Temperature (K)",
ylab=expression("Filled traps (cm"^"-3"*")"))
legend("topright",bty="n", expression("(a)"))
legend("right",bty="n", lty=c(1,2),expression("b=1","b=2"),
       col=c("black","red"),lwd=2)
peaks<-cbind(peak1$tl, peak2$tl)
matplot(temps, peaks, type="l", lwd=3, lty=c(1,2),
xlab="Temperature (K)",ylab="TL intensity [a.u.])")
legend("right",bty="n", lty=c(1,2),expression("b=1","b=2"),
       col=c("black","red"),lwd=2)
legend("topright",bty="n", expression("(b)"))

print.noquote("Parameters for first order TL peak")
print(c(peak1$sp[1],peak1$sp[2],peak1$sp[3]))
print(c(peak1$sp[4],peak1$sp[5],peak1$sp[6]))
cat("\nShape factor=",round(peak1$sp[7],3))

print.noquote("Parameters for second order TL peak")
print(c(peak2$sp[1],peak2$sp[2],peak2$sp[3]))
print(c(peak2$sp[4],peak2$sp[5],peak2$sp[6]))
cat("\nShape factor=",round(peak2$sp[7],3))
```

```
## [1] Parameters for first order TL peak
##       T1       T2       Tm
## 367.2004 397.0405 384.6000
##       d1       d2      thw
```

```
## 17.39956 12.44053 29.84009
##
## Shape factor= 0.417[1] Parameters for second order TL peak
##       T1       T2       Tm
## 363.5701 405.6734 383.8000
##       d1       d2      thw
## 20.22992 21.87338 42.10330
##
## Shape factor= 0.52
```

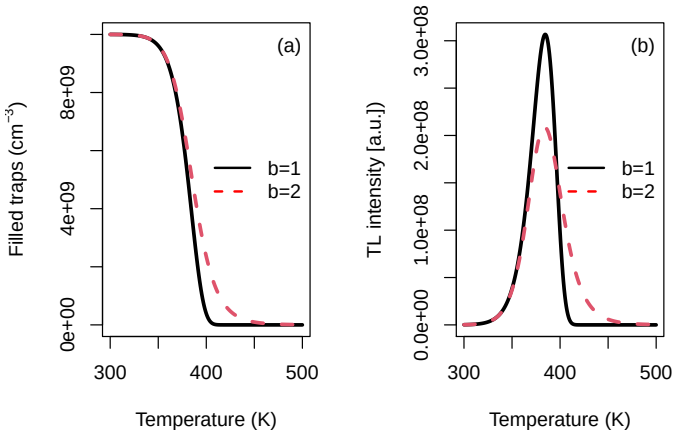


Fig. 2.7: Comparison of simulated first and second order TL peaks (solid and dashed lines respectively), evaluated with the same kinetic parameters. The second order process is slower, and the peak shape for second order is almost symmetric.(a) Filled traps $n(T)$ (b) The corresponding TL signal.

Code 2.6: The initial rise method: find energy E from TL data

```
# Apply the initial rise method to find the activation energy E
# Load the data from txt file, which  contains pairs of
# data  in the form: (Temperature_in_C,TL_Intensity (any units)
rm(list=ls())
library(tgcd)
data("Kitis")
```

```
x<-Kitis$x001[,1]
y<-Kitis$x001[,2]
mydata<-data.frame(x,y)
kB<-8.617*1e-5 # Boltzmann constant in eV/K
initialPos<-270  #analyze data points from #270 to #320
finalPos<-320
x<- mydata[,1][initialPos:finalPos]
y<- mydata[,2][initialPos:finalPos]
rangeData<-cbind(x,y)
y<-log(y)
x<-1/(kB*x)
bestfit<-lm(y~x)
summary(bestfit)
coefficients(bestfit)
par(mfrow=c(1,2))
plot(mydata,col="blue",
xlab=expression("Temperature [K]"),ylab = "TL (a.u.)")
lines(rangeData,col="red",lwd = 3)
legend("topright",bty="n","(a)")
plot(x, y, xlab = "1/(kT)  [1/eV]",ylab = "ln(TL)")
abline(lm(y~x))
legend("topright",bty="n",c("(b)"," ","Initial","Rise"))
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.158165 -0.030093  0.001636  0.027663  0.172387
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 39.97106    0.44678   89.46   <2e-16 ***
## x           -1.23111    0.01616  -76.17   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05592 on 49 degrees of freedom
## Multiple R-squared:  0.9916,Adjusted R-squared:  0.9915
## F-statistic:  5802 on 1 and 49 DF,  p-value: < 2.2e-16
##
## (Intercept)           x
##   39.971063   -1.231108
```

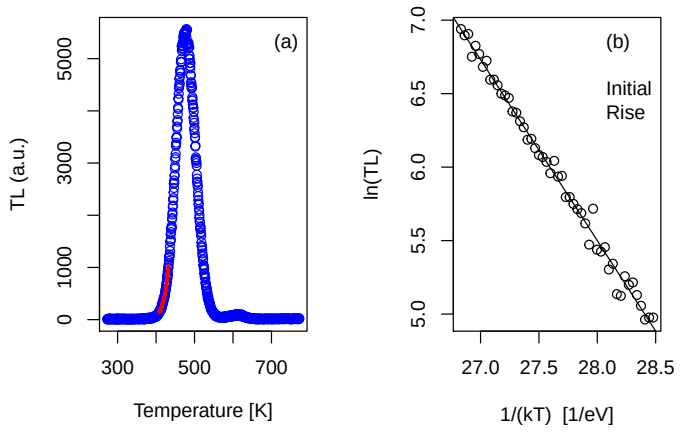


Fig. 2.8: Example of applying the initial rise method to find the activation energy E. (a) The red line indicates the initial rise area being analyzed; (b) The slope of the best line line gives the activation energy ($-E$).

Code 2.7: TL glow curve for four different heating rates

```
# Plot the same TL glow curve for four different heating rates
rm(list=ls())
library(tgcd)
temps <- seq(300, 440, by=1)  # temperature in K
hRates<-c(1,2,3,4)
## function to calculate TL for different hehating rates####
findTL<-function(hRate)
{peak<-simPeak(temps,n0=0.2e10,Nn=1e10,ff=1e12,ae=1.0, hr=hRate,
typ="f",plot=FALSE)
peak$tl}
### Calculate TL with different heating rates
TLs<-sapply(hRates,findTL)
matplot(temps,TLs,type="o", lty=c(1,1,1,1),lwd=1,  pch=1:4,
col=1:4,xlab="Temperature (K)", ylim=c(0,8e7),
ylab=expression(paste("TL/ ",beta," (counts/K)")))
legend("topright",bty="n","Heating rate method")
legend("topleft",bty="n", pch=c(1:4,NA),expression(
```

```
  "1 K/s","2","3","4"," "),col=c(1:4,NA),lwd=1)
```

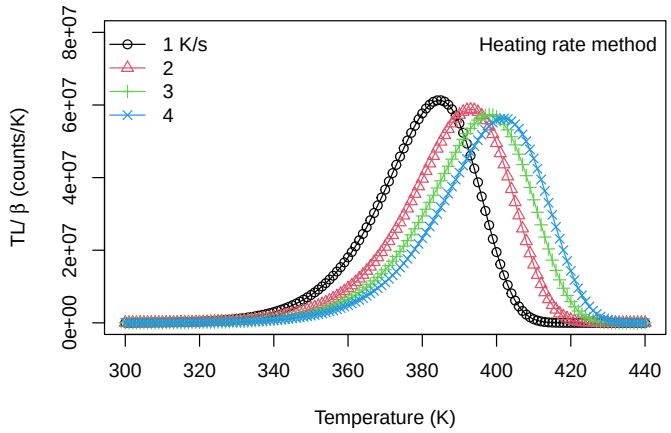


Fig. 2.9: Simulation of the same TL glow curve measured with four different heating rates $\beta = 1, 2, 3, 4$ K/s. As the heating rate increases, the TL peak shifts to the right towards higher temperatures, and the intensity decreases. The area under the curves stays the same. Notice that the y-scale is TL/β, i.e. in counts/K, and not in counts/s.

Code 2.8: Apply heating rate method to TL data, to find E,s

```
# Apply the heating rate method to find E,s
rm(list=ls())
library(tgcd)
library(scales)
kB<-8.617*1e-5 # Boltzmann constant in eV/K
temps <- seq(340, 420, by=2)
#### function to calculate TL for different heating rates####
findTL<-function(hRate)
{peak<-simPeak(temps,n0=0.2e10,Nn=1e10,ff=1e12,ae=1.0, hr=hRate,
typ="f",plot=FALSE)
peak$tl}
######## function to find Tmax ######
```

```
findTmax<-function(hRate)
{peak<-simPeak(temps,n0=0.2e10,Nn=1e10,ff=1e12,ae=1.0, hr=hRate,
typ="f",plot=FALSE)
temps[[match(max(peak$tl),peak$tl)]]}
########## calculate 1/kTmax and log(Tmax^2/beta)
hRates<-c(1,2,3,4)
maxTL<-sapply(hRates,findTmax)
TLs<-sapply(hRates,findTL)
####### plots
par(mfrow=c(1,2))
matplot(temps,TLs,type="o",lty="solid",lwd=1,pch=1:4,
 col=c(1:4),xlab="Temperature (K)",  ylim=c(0,8e7),
ylab=expression(paste("TL/ ",beta," (counts/K)")))
y<-log(maxTL^2/hRates)
x<-1/(kB*maxTL)
coefficients(lm(y~x))
energy<-coefficients(lm(y~x))[[2]]
intercept<-coefficients(lm(y~x))[[1]]
cat('\nFrequency factor s=',
    scientific(exp(-intercept)*energy/kB),' s^-1')
legend("topright",bty="n","(a)")
legend("topleft",bty="n", pch=c(1:4,NA),expression(
  "1 K/s","2","3","4"," "),col=c(1:4,NA),lwd=1)
plot(x,y,xlab="1/(kT)   (1/eV)",
ylab=expression(paste('ln(Tmax'^'2'*')/',beta)),ylim=c(10.4,12))
abline(lm(y ~ x))
legend("topleft",bty="n",legend=c("(b)","Heating rate",
"method"))
```

```
## (Intercept)           x
## -17.1459974   0.9619712
##
## Frequency factor s= 3.12e+11  s^-1
```

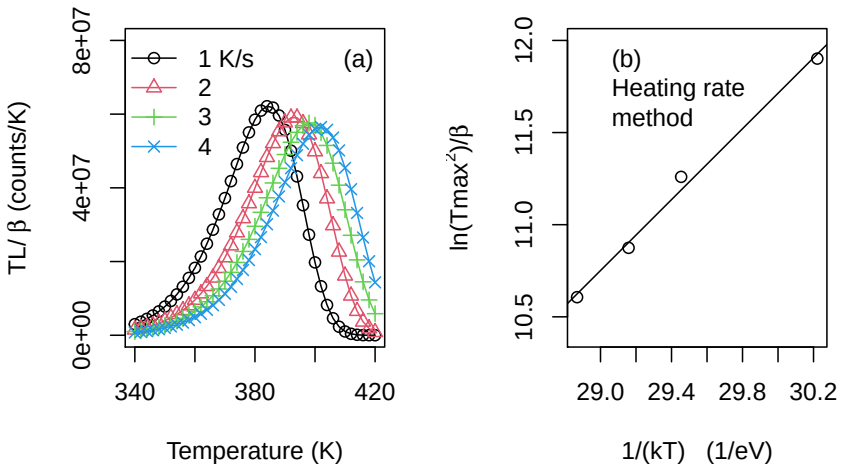


Fig. 2.10: Applying the heating rate method to obtain both the activation energy E and the frequency factor s. (a) The simulated TL glow curves (b) The slope and intercept of the best fit line yield both parameters (E, s).

Code 2.9: The GOT equation for TL in OTOR (deSolve)

```
# Numerical solution of the GOT equation for TL
rm(list=ls())
library("deSolve")
# Define Parameters
A_n <- 1e-10 # coefficient of retrapping
A_m <- 1e-8 # coefficient of recombination
k_B <- 8.617e-5  # Boltzmann constant
E <- 1 # electron trap depth [eV]
s <- 1e12 # frequenc factor [1/s]
delta.t <- 1
t <- seq(0, 300, delta.t)
# time = temperature, i.e. heating rate= 1 K/s
n.0 <- 1e10  # initial concentration of filled traps
N.traps <- 1e11   # total concentrations of available traps
# Calculate numerical ODE solution
```

```
ODE <- function(t, state, parmameters){
with(as.list(c(state, parameters)),{
dn <- -s*exp(-E/(k_B*(273+t))) * n^2 * A_m / ((N.traps - n) *
A_n + n * A_m)
list(c(dn))  })}
parameters <- c(N.traps = N.traps, s = s, E = E, k_B = k_B,
A_m = A_m, A_n = A_n)
state <- c(n = n.0)
num_ODE <- ode(y = state, times = t, func = ODE,
parms = parameters)
# Plot filled traps n(T) and TL as a function of temperature
par(mfrow=c(1,2))
plot(x = num_ODE[,1],
    y = num_ODE[,2],xlab=expression("Temperature ["^"o"*"C]"),
    ylab =expression("Filled traps (cm"^"-3"*")"),col = "red")
legend("topright",bty="n",expression('(a)',' ','GOT','n(T)'))
plot(x = num_ODE[-1,1],  y = abs(diff(num_ODE[,2])),
xlab=expression("Temperature ["^"o"*"C]"),
     ylab = "TL signal [a.u.]")
legend("topright",bty="n",c("(b)"," ","TL"))
```

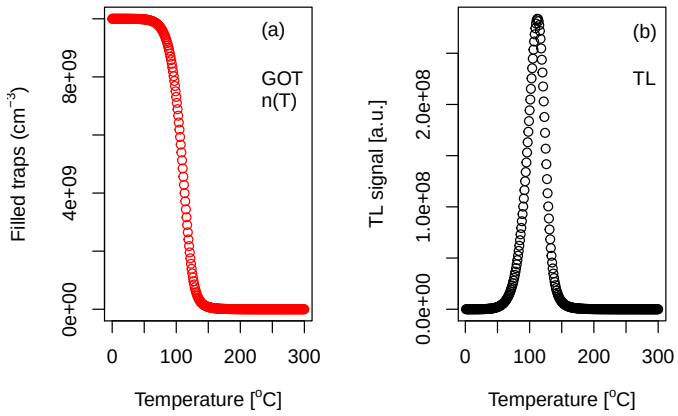


Fig. 2.11: Numerical solution of the GOT equation for TL. (a) Filled traps $n(T)$ (b) The TL glow curve.

Code 2.10: Plot the W0-Lambert solution of GOT equation

```
rm(list=ls())
library(lamW)
##  Example of plot for Lambert W-function from x=0 to x=100
xs <- seq(0, 100, by=1)
ys <- lambertW0(xs)
## Plot the analytical solution of GOT, using Lambert W-function
x1<-300:450   # temperatures in K
kB<-8.617E-5
no<-1E8
N<-1E10
R<-.01
c<-(no/N)*(1-R)/R
En<-1
s<-1E12
beta<-1
k<-function(u) {integrate(function(p){exp(-En/(kB*p))},
                          300,u)[[1]]}
y1<-lapply(x1,k)
x<-unlist(x1)
y<-unlist(y1)
zTL<-(1/c)-log(c)+(s*no/(c*N*R))*y
# plots
par(mfrow=c(1,2))
plot(xs, ys, type="l", col="red", lwd=3, ylim=c(0,5),
     xlab="x", ylab="Lambert W0(x)")
legend("topleft",bty="n",c("(a)"," ","Lambert",
"function W0(x)"))
plot(x,(N*R/((1-R)^2))*s*exp(-En/(kB*x))/(lambertW0(exp(zTL))
+lambertW0(exp(zTL))^2),xlab="Temperature, K",col="blue",
     ylab="TL with W0(T)")
legend("topleft",bty="n",c("(b)"," ","TL", "using", "W0(x)"))
```

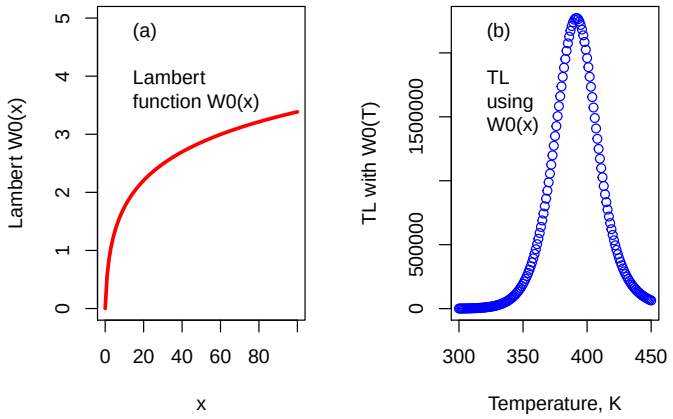


Fig. 2.12: (a) Plot of the Lambert function $W(x)$ between $x = 0$ and $x = 100$. (b) Plot of the analytical solution of the GOT Eq.(**??**), using the Lambert W function.

Code 2.11: Deconvolution of Glocanin glow curve (tgcd)

```
# Deconvolution of Reference glow curve #1 (project GLOCANIN)
rm(list=ls())
library("tgcd")
data(Refglow)   # Load the data
# Deconvolve data with 1 peak using the LAMBERT W function
startingPars <-
  cbind(c(15.0),c(1.0),c(520), c(0.1))    # Im, E, Tm, R
invisible(capture.output(TL1 <- tgcd(Refglow$x001, npeak=1,
model="lw",inisPAR=startingPars,nstart=10,edit.inis=FALSE)))
print.noquote("Best fit parameters")
TL1$pars
cat("\nGeometrical shape factor=",
round(TL1$sp[,7],2))
cat("\nFigure Of Merit FOM=",round(TL1$FOM,4))

  ## [1] Best fit parameters
  ##          INTENS(Im) ENERGY(E) TEMPER(Tm) rValue(r)
```

```
## 1th-Peak   10968.39  1.182267   490.4689     1e-16
##
## Geometrical shape factor= 0.43
## Figure Of Merit FOM= 0.0097
```

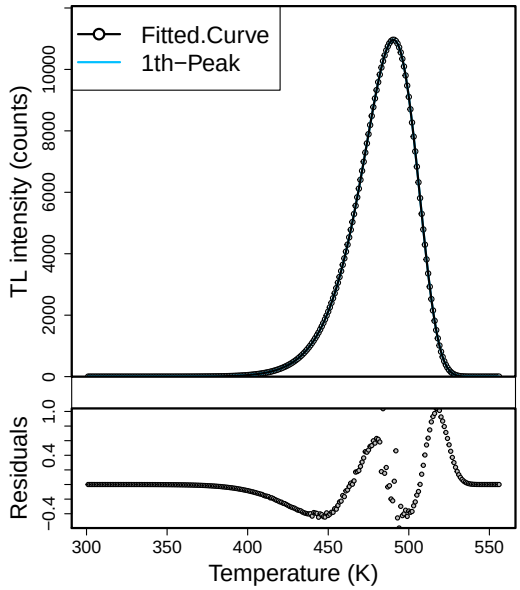


Fig. 2.13: Deconvolution of TL data containing a single peak, using the Lambert W function. The data is from Reference glow curve #1 in the intercomparison project GLOCANIN (Bos et al. [3]).

Code 2.12: Deconvolution of TL user data (.txt file, tgcd)

```
# Deconvolve data with 2 peaks using the LAMBERT W function
rm(list=ls())
library("tgcd")
# Load the data
mydata = read.table("lbodata.txt")
startingPars <-
```

```
cbind(c(105.0,5.0),c(1.1,1.4),c(460,550),c(0.01,.01)) #Im,E,Tm,R
invisible(capture.output(TL1 <- tgcd(mydata, npeak=2,
model="lw",inisPAR=startingPars, nstart=10, edit.inis=FALSE)))
print.noquote("Best fit parameters")
TL1$pars
cat("\nGeometrical shape factors"," ")
round(TL1$sp[,7],2)
```

```
## [1] Best fit parameters
##          INTENS(Im) ENERGY(E) TEMPER(Tm)    rValue(r)
## 1th-Peak  115.97882  1.209594   464.2374 5.133850e-11
## 2th-Peak   22.96337  1.158213   516.7809 2.233305e-01
##
## Geometrical shape factors  1th-Peak 2th-Peak
##     0.45     0.43
```

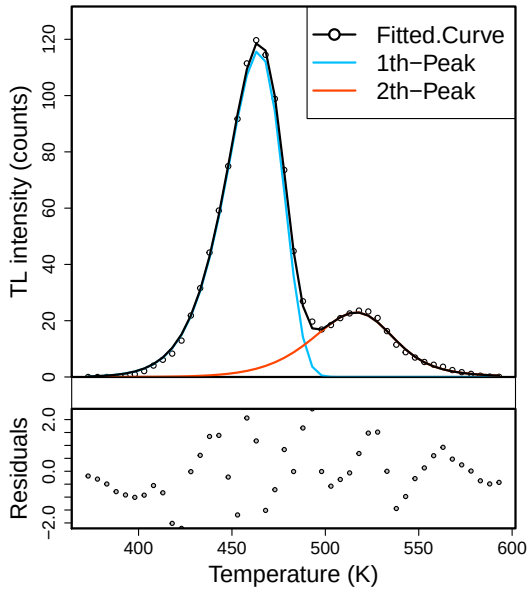


Fig. 2.14: Deconvolution of experimental data for dosimetric material LBO with 2 peaks (Kitis et al. [18]), by using the Lambert W function.

Code 2.13: Deconvolution of 9-peak Glocanin TL data (tgcd)

```
# Deconvolve TL signal using 9 peaks (no background subtraction)
# a GOK model using user-supplied initial kinetic parameters.
rm(list=ls())
library("tgcd")
data(Refglow)
knPars <-
cbind(c(9824,21009,27792,50520,7153, 5496,6080,1641,2316), # Im
c(1.24, 1.36, 2.10, 2.65, 1.43, 1.16, 2.48, 2.98, 2.25), # E
c(387, 428, 462, 488, 493, 528, 559, 585, 602), # Tm
c(1.02, 1.15, 1.99, 1.20, 1.28, 1.19, 1.40, 1.01, 1.18)) # b
invisible(capture.output(TL1 <- tgcd(Refglow$x009, npeak=9,
model="g1",inisPAR=knPars, nstart=10, edit.inis=FALSE)))
print.noquote("Best fit parameters")
TL1$pars
```

```
## [1] Best fit parameters
##           INTENS(Im) ENERGY(E) TEMPER(Tm) bValue(b)
## 1th-Peak   9820.004  1.237294   387.3132  1.025077
## 2th-Peak  20959.877  1.361432   428.1756  1.149763
## 3th-Peak  27324.883  2.058686   462.1433  1.825707
## 4th-Peak  52194.441  2.498238   488.1280  1.119436
## 5th-Peak   7519.826  1.411680   495.8060  1.045013
## 6th-Peak   5351.507  1.474229   524.2461  1.388807
## 7th-Peak   7077.083  2.177145   557.5652  1.334783
## 8th-Peak   1547.388  3.343393   586.0483  1.057578
## 9th-Peak   2311.923  2.249152   602.6770  2.000000
```

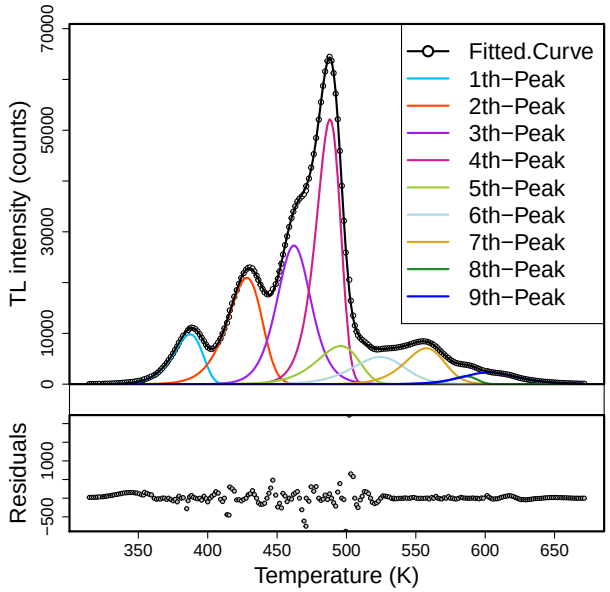


Fig. 2.15: Deconvolution of a glow curve from the GLOCANIN project using 9 peaks with a GOK model, and with user-supplied initial kinetic parameters (Bos et al. [3]).

```
# This is a continuation of R code above)
hist(TL1$residuals,lwd=3,xlab="Residuals",
ylab="Frequency",main=NULL)
legend("topright",bty="n",c("Fitting","Residuals"))
```

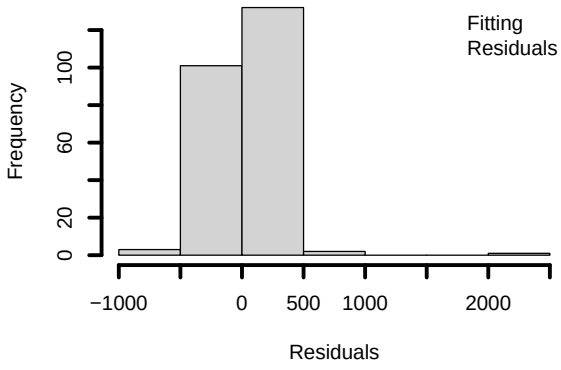


Fig. 2.16: Histogram of residuals $y_i^{expt} - y_i^{fit}$ from the best fit shown in Fig.2.15, for the nine-peak TL glow curve of the GLOCANIN project, with a GOK model.

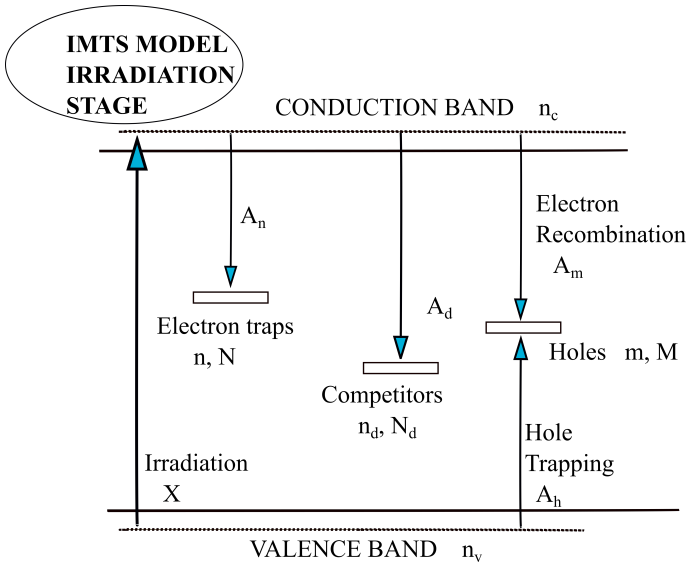


Fig. 2.17: The general phenomenological interactive multitrap system (IMTS) model, describing stimulated luminescence effects, during the irradiation stage.

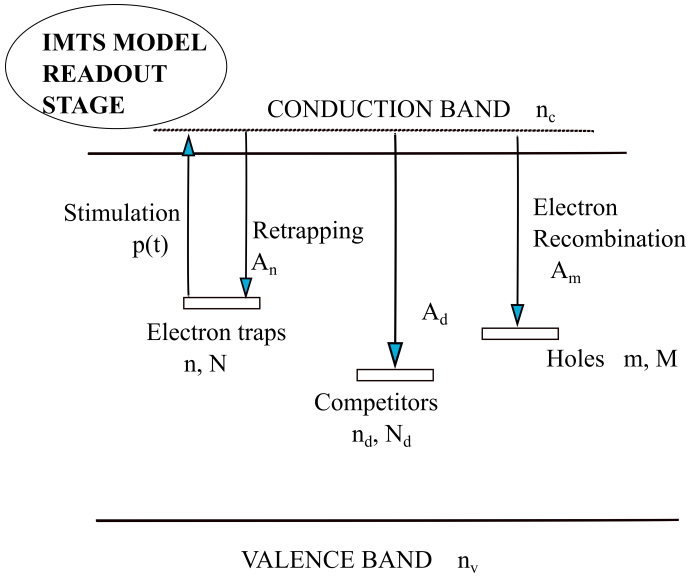


Fig. 2.18: Schematic diagram of the (IMTS) model, describing stimulated luminescence effects during the thermal or optical stimulation stage.

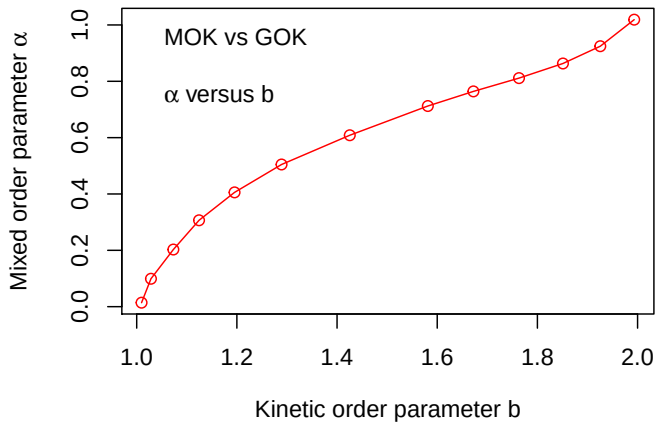


Fig. 2.19: The mixed order kinetics parameter α as a function of the general order kinetics parameter b.

Code 2.14: MOK deconvolution of Glocanin TL (tgcd)

```
# Deconvolution of Reference GLOCANIN glow curve #1 with MOK
rm(list=ls())
library("tgcd")
data(Refglow)
# Load the data.
# Deconvolve data with 1 peak using the MOK expression
startingPars <-
  cbind(c(15.0),  c(1.0), c(520), c(0.1)) # Im, E, Tm, R
invisible(capture.output(TL1 <- tgcd(Refglow$x001, npeak=1,
model="m1",inisPAR=startingPars, nstart=10, edit.inis=FALSE)))
print.noquote("Best fit parameters")
TL1$pars

  ## [1] Best fit parameters
  ##          INTENS(Im) ENERGY(E) TEMPER(Tm)   aValue(a)
  ## 1th-Peak   10967.93  1.182694    490.341 0.000499707
```

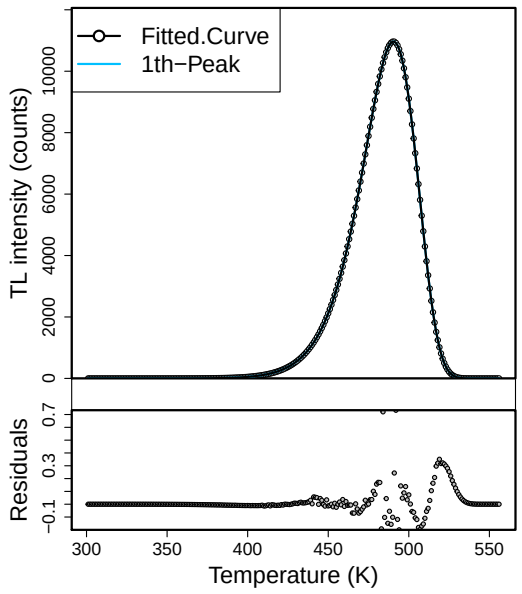


Fig. 2.20: Deconvolution of Reference glow curve #1 in the GLOCANIN project, using Mixed Order Kinetics (MOK).

Function Name	Description
plot_RLumCarlo	Plots 'RLumCarlo' modeling results (the averaged signal or the number of remaining electrons), with modeling uncertainties.
run_MC_ISO_DELOC	Simulation of ITL signals using the one trap one recombination center (OTOR) model.
run_MC_LM_OSL_DELOC	Simulation of LM-OSL signals using the delocalized OTOR model.
run_MC_CW_OSL_DELOC	Simulation of CW-OSL signals using the OTOR model.
run_MC_TL_DELOC	Simulation of TL signals using the OTOR model.

Table 2.1: Table of DELOCalized functions available in the package *RLumCarlo*.

Code 2.15: Combine 3 plots for isothermal experiment

```
##=========================================================##
## COMBINE 3 PLOTS  FOR DELOCALIZED ITL
##=========================================================##
rm(list = ls(all=T))
suppressMessages(library(RLumCarlo))
## set time vector
times <- seq(0, 400)
run_MC_ISO_DELOC(T=220, E=1.4, s=1e12,R=1e-3, times = times) %>%
plot_RLumCarlo(norm = TRUE, col="red",legend = F)
run_MC_ISO_DELOC(T=230, E=1.4, s=1e12,R=1e-3, times = times) %>%
plot_RLumCarlo(norm = TRUE, col="green",add = TRUE)
run_MC_ISO_DELOC(T=240, E=1.4, s=1e12,R=1e-3, times = times) %>%
plot_RLumCarlo(norm = TRUE, col="blue",add = TRUE, times= times)
legend("top",bty="n",legend=c("Isothermal TL signal",
```

Process	Symbol	Parameter in RLumCarlo function	Units	Typical values
Delocalized TL	E	Thermal activation energy of the trap	eV	0.5-3
	s	Frequency factor of the trap	1/s	1E8-1E16
	times	Sequence of time steps for simulation (heating rate is 1 K/s)	s	0-700
	clusters	Number of MC runs	1	1E1-1E4
	N_e	Total number of electron traps available	1	2-1E5
	n_filled	Number of filled electron traps at the beginning of the simulation	1	1-1E5
	R	Delocalized retrapping ratio	1	0-1
Delocalized CW-IRSL	A	Optical excitation rate from trap to conduction band	1/s	1E-3-1
	times	Sequence of time steps for simulation	s	0-500
	clusters	Number of MC runs	1	1E1-1E4
	N_e	Total number of electron traps available	1	2-1E5
	n_filled	Number of filled electron traps at the beginning of the simulation	1	1-1E5
	R	Delocalized retrapping ratio	1	0-1
Delocalized ISO	E	Thermal activation energy of the trap	eV	0.5-3
	s	Frequency factor of the trap	1/s	1E8-1E16
	T	Temperature of the isothermal process	°C	20-300
	times	Sequence of time steps for simulation	s	0-1000
	clusters	Number of MC runs	1	1E1-1E4
	N_e	Number of electrons	1	2-1E5
	n_filled	Number of filled electron traps at the beginning of the simulation	1	1-1E5
	R	Delocalized retrapping ratio	1	0-1
Delocalized LM-OSL	A	Optical excitation rate from trap to conduction band	1/s	1E-3-1
	times	Sequence of time steps for simulation	s	0-3000
	clusters	Number of MC runs	1	1E1-1E4
	N_e	Total number of electron traps available	1	2-1E5
	n_filled	Number of filled electron traps at the beginning of the simulation	1	1-1E5
	R	Delocalized retrapping ratio	1	0-1

Table 2.2: Table of input parameters for DELOC functions in *RLumCarlo*

```
expression("at 220, 230, 240"^"o"*"C")))
```

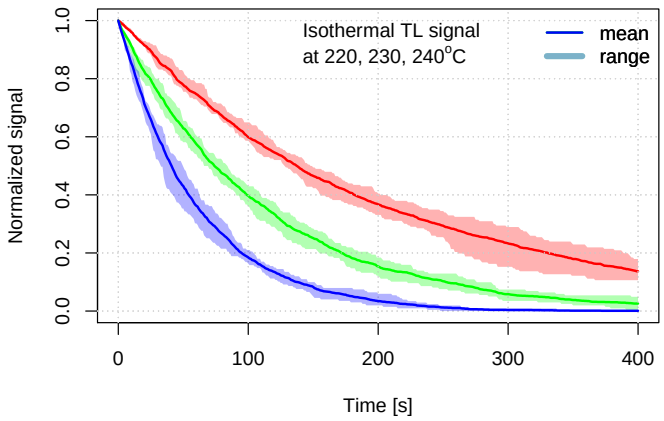


Fig. 2.21: Example of combining 3 plots for ITL signals from DELOCalized transitions, within the OTOR model.

Code 2.16: Single MC plot for delocalized TL

```
##================================================##
## Example 1: Single MC Plot  for delocalized TL
##================================================##
rm(list = ls(all=T))
library(RLumCarlo)
run_MC_TL_DELOC(
s = 3.5e12,
E = 1.45,
R = 0.1,
times = 100:350
) %>%
#Plot results of the MC simulation
plot_RLumCarlo
legend("topleft",bty="n",c("TL signal", "using
DELOC","function"))
```

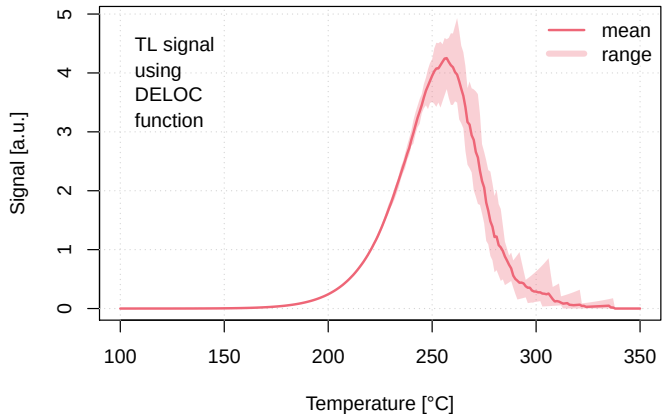


Fig. 2.22: MC simulation of TL glow curve from the OTOR delocalized transition model.

Code 2.17: MC for delocalized TL: multiple parameters

```
##================================================##
##================================================##
## Plot multiple TL  curves with varying params
##================================================##
# define your parameters
rm(list = ls(all=T))
library(RLumCarlo)
times=seq(150,350,1)
s=rep(3.5e12,4)
E=rep(1.45,4)
R<-c(0.7e-6,1e-6,0.01,0.1)
clusters=100
N_e =c(400, 500, 700, 400)
n_filled =c(400, 500, 300, 70)
method="par"
output ="signal"
col=c(1,2,3,4) # different colors for the individual curves
plot_uncertainty <- c(TRUE,TRUE,TRUE,TRUE) # plot  uncertainty?
```

```
add_TF <- c(FALSE,rep(TRUE, (length(R)-1)))
for (u in 1:length(R)){
   results <-run_MC_TL_DELOC(times=times, s=s[u],E=E[u],
clusters =clusters, N_e = N_e[u],n_filled = n_filled[u],
R=R[u], method = method, output = output)
   plot_RLumCarlo(results,add=add_TF[u],legend = FALSE,
col=col[u], ylim=c(0,20))
}
legend("left",bty="n",c("TL","DELOC","many variables"))
legend("topright",bty="n",ncol=5,cex=0.55,title = "parameters" ,
       legend=c(paste0("E = ", E),paste0("s = ", s),
                paste0("n_filled = ", n_filled),
                paste0("N_e = ", N_e),
                paste0("R = ", R)), text.col=col)
```

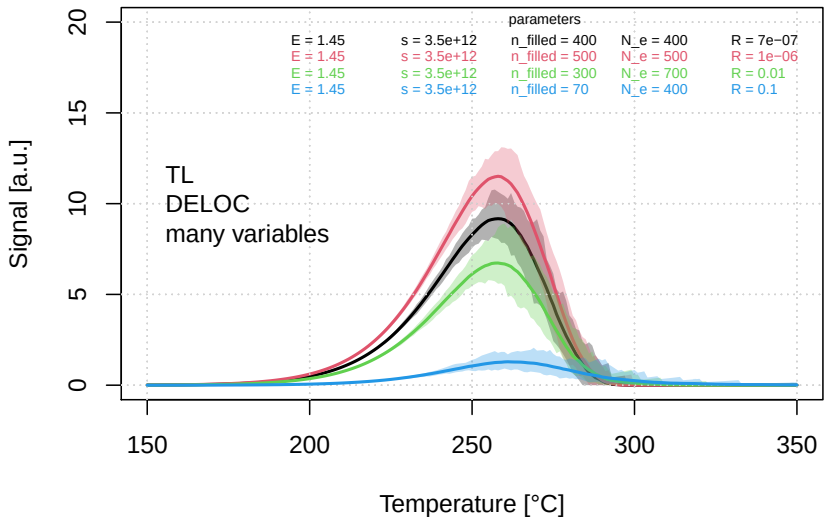


Fig. 2.23: Simulations of DELOCalized TL transitions, by varying many parameters in the model.

Chapter 3
ANALYSIS OF EXPERIMENTAL OSL DATA

Abstract In this chapter we provide detailed R codes which show how researchers can analyze and model their experimental continuous wave OSL signals (CW-OSL), and linearly modulated OSL signals (LM-OSL). We present detailed examples for computerized analysis of OSL experimental data, using the R packages *Luminescence* and *numOSL*, and how to extract the relevant optical cross sections. On the modeling side, we present the GOT equation derived from the OTOR model, show how this equation can be integrated numerically using R, and compare with the analytical solutions for OSL, which are based on the Lambert W function. We show how to transform the shapeless experimental CW-OSL signals into peak-shaped LM-OSL signals, and present several examples of using the R package *RLumCarlo* to simulate OSL signals. The chapter concludes with a list of recommended protocols for analyzing OSL data in the laboratory.

Code 3.1: Fitting 2-component CW-OSL signal (numosl)

```
### Fitting CW-OSL signal with package numOSL (2 components)
rm(list=ls())
library("numOSL")
data <- read.table("KST4ph300IR.txt")
data<-data.frame(data[2:500,1],data[2:500,2]) #data t=1-500 s
a<-decomp(data, ncomp=2)
print.noquote("Best fit parameters")
a$LMpars
cat("\nFOM=",a$FOM)
```

```
## [1] Best fit parameters
##            Ithn   seIthn      Lamda      seLamda
## Comp.1 58290.49 657.5423 0.17560611 0.0016782065
## Comp.2 83985.26 697.0981 0.02776158 0.0004630479
##
## FOM= 7.719681
```

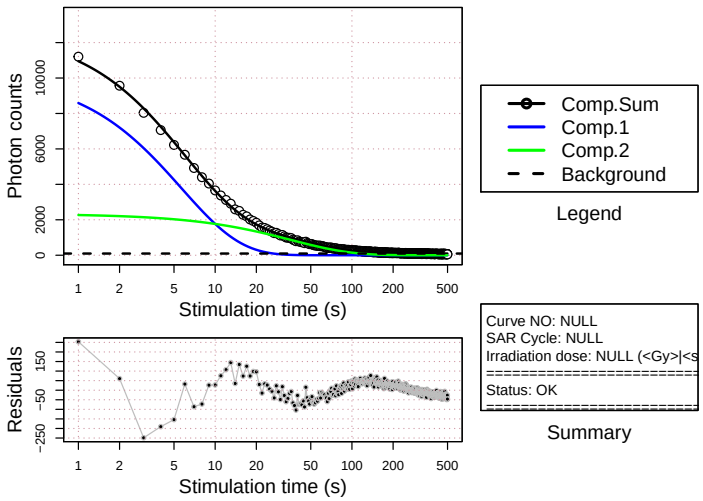


Fig. 3.1: Example of fitting the first 500 s of a CW-OSL signal with two exponential components, using the package *numOSL*. The CW-OSL data are from a freshly irradiated aliquot of feldspar sample KST4 (Pagonis et al. [44]).

Code 3.2: Fitting 3-component CW-OSL signal (Luminescence)

```
# Fitting CW-OSL  with package Luminescence (3 components)
rm(list=ls())
suppressMessages(library("Luminescence",warn.conflicts= FALSE))
##load and fit the data
data <- read.table("KST4ph300IR.txt")
data<-data.frame(data[1:800,1],data[1:800,2]) #use t=1-800 s
```

```
invisible(capture.output(fit <- fit_CWCurve(
  values = data,
  main = " ",
  n.components.max = 3)))
cat("\nBest fit parameters"," ")
cat("\nI01=",fit$data$I01,"  I02=",fit$data$I02,
    "  I03=",fit$data$I03)
cat("\nlambda1=",fit$data$lambda1,"  lambda2=",fit$data$lambda2)
cat("\nlambda3=",fit$data$lambda3)

  ##
  ## Best fit parameters
  ## I01= 40696.02   I02= 82084.83   I03= 91582.97
  ## lambda1= 0.2342178   lambda2= 0.04617059
  ## lambda3= 0.002939544
```

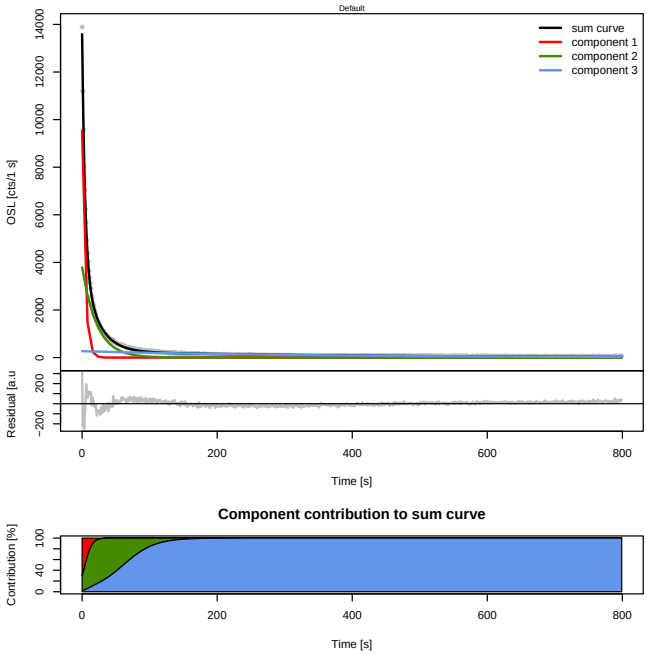


Fig. 3.2: Example of fitting the first 800 s of a CW-OSL signal from feldspar sample KST4, with three exponential components, using the package *Luminescence*. Experimental data from Pagonis et al. [44].

Code 3.3: Solve GOT equation for CWOSL (deSolve)

```
# Numerically Solve GOT equation for CWOSL using package deSolve
rm(list=ls())
library("deSolve")
# Define Parameters
A_n <- 1e-10 # coefficient of retrapping
A_m <- 1e-8 # coefficient of recombination
Aopt <- 0.1 # Stimulation coefficient [1/s]
delta.t <- 1
t <- seq(0, 300, delta.t)
n.0 <- 1e10  # initial concentration of filled traps
N.traps <- 1e11   # total concentrations of available traps
# Calculate numerical ODE solution
ODE <- function(t, state, parmameters){
  with(as.list(c(state, parameters)),{
    dn <-  -Aopt * n^2 * A_m / ((N.traps - n) * A_n + n * A_m)
    list(c(dn)) })}
parameters <- c(N.traps=N.traps,Aopt= Aopt,A_m = A_m,A_n = A_n)
state <- c(n = n.0)
num_ODE <- ode(y = state, times = t, func = ODE,
parms = parameters)
# Plot remaining filled traps  and TL signal
par(mfrow=c(1,2))
plot(num_ODE[,1],num_ODE[,2],xlab = "Time[s]",
ylab = "Filled traps",col = "red")
legend("topright",bty="n",c("(a)"," ","n(t)","GOT model"))
plot(num_ODE[-1,1],y = abs(diff(num_ODE[,2])),
xlab = "Time[s]", ylab = "CW-OSL signal [a.u.]")
legend("topright",bty="n",c("(b)"," ","CW-OSL"))
```

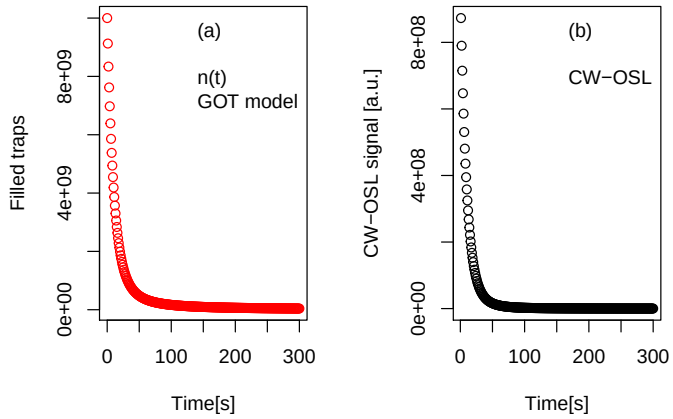


Fig. 3.3: Numerical solution of the GOT differential equation (**??**) for CW-OSL, using the R package *deSolve*. (a) Filled traps $n(t)$. (b) The corresponding CW-OSL signal.

Code 3.4: Plot of the Lambert W-function solution for CW-OS in the GOT model

```
## Plot the CW-OSL solution of GOT, using Lambert W-function
rm(list=ls())
library(lamW)
t<-0:300
kB<-8.617E-5
no<-1E10
N<-1E10
R<-.01
c<-(no/N)*(1-R)/R
lambda<-.01
x<-unlist(t)
zCWOSL<-unlist((1/c)-log(c)+(lambda*no/(c*N*R))*t)
# plots
par(mfrow=c(1,2))
plot(x,(N*R/(1-R))/(lambertW0(exp(zCWOSL))),xlab="Time, s",
```

```
ylab="Filled traps n(T)",ylim=c(0,N),col="red")
legend("topright",bty="n",c("(a)  n(t)"," ","using","W0(x)"))
plot(x,(N*R/((1-R)^2))*lambda/(lambertW0(exp(zCWOSL))
          +lambertW0(exp(zCWOSL))^2),
     xlab="Time, s",ylab="CW-OSL signal")
legend("topright",bty="n",c("(b)",  "CW-OSL"))
```

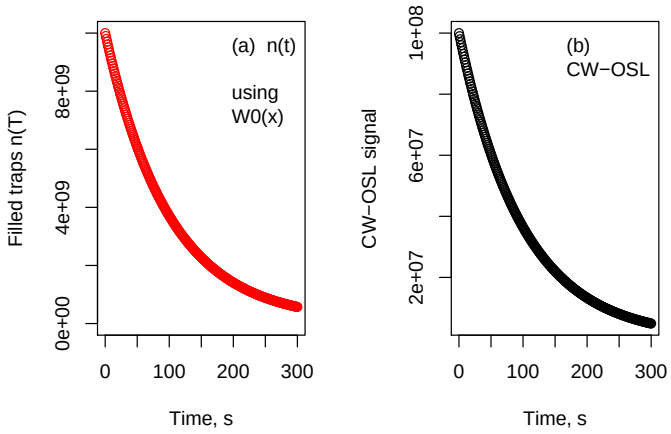


Fig. 3.4: Plot of the analytical solution of GOT differential equation for CW-OSL, using the Lambert W function solution from Eq.(**??**). (a) Filled traps $n(T)$ and (b) The corresponding CW-OSL signal.

Code 3.5: Single plot MC for delocalized CW-OSL

```
##================================================##
## MC simulations for delocalized CW-OSL
##================================================##
rm(list = ls(all=T))
library(RLumCarlo)
run_MC_CW_OSL_DELOC(
   A = 0.12,
   R = 0.1,
   times = 0:100
) %>%
   #Plot results of the MC simulation
```

```
   plot_RLumCarlo(legend = F,ylab="CW-OSL [a.u.]")
legend("top",bty="n", legend=c("CW-OSL", "with function DELOC",
"Package RLumCarlo"))
```

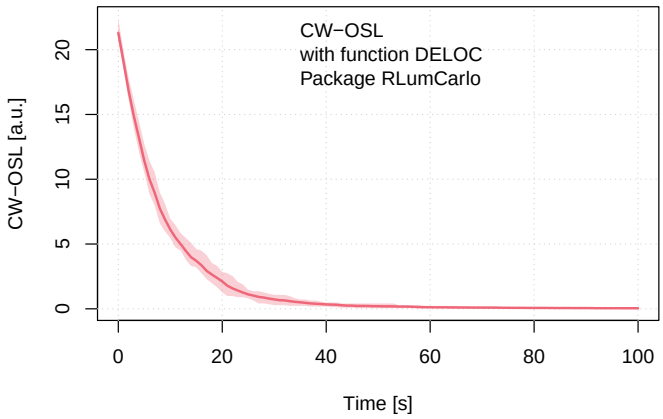


Fig. 3.5: Simplest example of MC simulation for CW-OSL DELOCalized transition process, within the OTOR model and based on the GOT Eq.(**??**). The parameters here are the excitation rate $\lambda = 0.12$ s^{-1} and retrapping ratio $R = 0.1$.

Code 3.6: Single plot MC for delocalized CW-OSL

```
##================================================##
## MC simulations for delocalized CW-OSL
##================================================##
rm(list = ls(all=T))
library(RLumCarlo)
run_MC_CW_OSL_DELOC(
n_filled=100,N_e=100,A = 0.12,
R = 0.1,
times = 0:100,
output="remaining_e"
) %>%
#Plot results of the MC simulation
plot_RLumCarlo(legend = F,xlab="Time (s)",
```

```
ylab="Filled Traps n(t)")
legend("topright",bty="n",legend=c("Remaining trapped electrons",
"during CW-OSL experiment"))
```

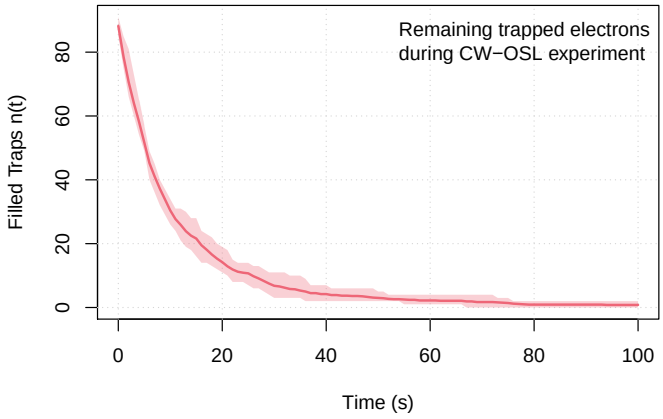


Fig. 3.6: Plotting the remaining electrons $n(t)$ during a DELOCalized CW-OSL simulation.

Code 3.7: Combine two plots from RlumCarlo

```
##=====================================================##
## Example: COMBINE TWO PLOTS  for delocalized CW-OSL
##=====================================================##
rm(list = ls(all=T))
library(RLumCarlo)
## set time vector
times <- seq(0, 50)
run_MC_CW_OSL_DELOC(A = 0.1, R = 0.01,n_filled=100,N_e=100,
                    times = times) %>%
plot_RLumCarlo(norm = TRUE, col="blue",
                   legend = TRUE)
legend("top",bty="n",legend=c("CW-OSL signal",
expression("with A=0.1, 0.2 s"^"-1"*"")))
run_MC_CW_OSL_DELOC(A = 0.2, R = 0.01,n_filled=100,N_e=100,
```

```
                        times = times) %>%
plot_RLumCarlo(norm = TRUE, col="red", add = TRUE)
```

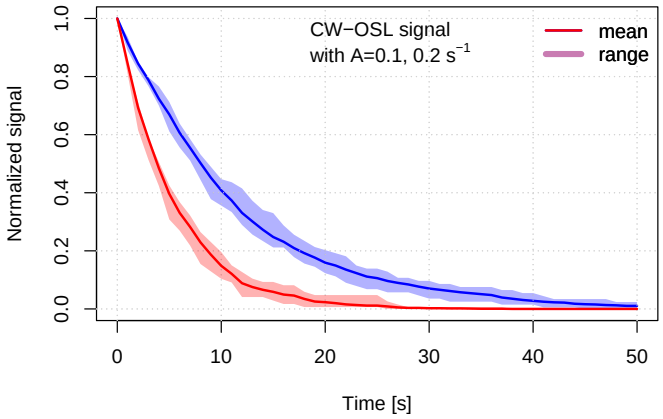


Fig. 3.7: Combine two plots for delocalized CW-OSL signals, with two different excitation rates $A = 0.1,\ 0.2\ \mathrm{s}^{-1}$.

Code 3.8: MC for delocalized CW-OSL: multiple parameters

```
## Simulate CW-OSL DELOC with several parameter changes
##======================================================##
rm(list = ls(all=T))
library(RLumCarlo)
# define your parameters
A <- c(1,.3,.5,.1)
times <- seq(0,60,1)
R<-c(1e-7,1e-6,0.01,0.1) # sequence of different R values
clusters <- 200 # number of Monte Carlo simulations
N_e <- c(200, 500, 700, 400) # number of free electrons
n_filled <- c(200, 500, 100, 70) # number of filled traps
method <-"par"
output <- "signal"
col <- c(1,2,3,4) # different colors for the individual curves
plot_uncertainty <- c(T,F,T,F) # plot the uncertainty?
```

```
add_TF <- c(F,rep(T, (length(R)-1)))
for (u in 1:length(R)){
 results <-run_MC_CW_OSL_DELOC(A=A[u],times,clusters =clusters,
   N_e = N_e[u],   n_filled = n_filled[u],
R=R[u], method = method, output = output)
  plot_RLumCarlo(results,add=add_TF[u],legend = F, col=col[u])
}
legend("topright",ncol=4,bty="n",cex=.65,title = "CW-OSL
function DELOC" ,
       legend=c(paste0("A = ", A),
                paste0("n_filled = ", n_filled),
                paste0("N_e = ", N_e),
                paste0("R = ", R)), text.col=col)
```

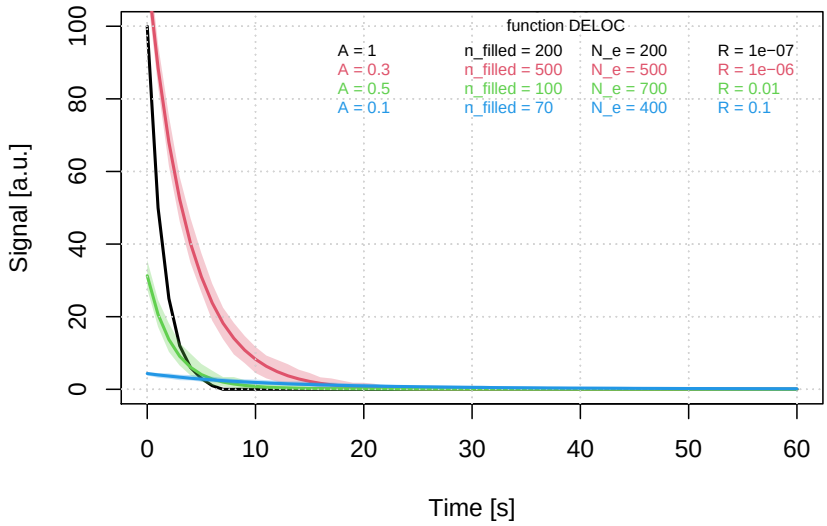


Fig. 3.8: Simulation of several CW-OSL curves using the DELOC functions of the *RLumCarlo* package, with several parameter changes.

Code 3.9: Solve the GOT equation for LM-OSL (deSolve)

```
# Numerically solve the GOT equation for LM-OSL
rm(list=ls())
library("deSolve")
# Define Parameters
A_n <- 1e-10 # coefficient of retrapping
A_m <- 1e-8 # coefficient of recombination
Aopt <- 0.1 # Stimulation coefficient [1/s]
delta.t <- .2
t <- seq(0, 50, delta.t)
n.0 <- 1e10  # initial concentration of filled traps
N.traps <- 1e11   # total concentrations of available traps
# Calculate numerical ODE solution
ODE <- function(t, state, parmameters){
  with(as.list(c(state, parameters)),{
    dn <-  -Aopt *t*  n^2 * A_m / ((N.traps - n)* A_n+n* A_m)
    list(c(dn))
  })}
parameters <- c(N.traps =N.traps,Aopt = Aopt,A_m=A_m,A_n = A_n)
state <- c(n = n.0)
num_ODE <- ode(y = state, times = t, func = ODE,
parms = parameters)
# Plot remaining electrons and LM-OSL as a function of time
par(mfrow=c(1,2))
plot(x = num_ODE[,1],
    y = num_ODE[,2],xlab = "Time[s]", ylab = "Filled traps",
col = "red")
legend("topright",bty="n",c("(a)"," ","n(t)"))
plot(x = num_ODE[-1,1], y = abs(diff(num_ODE[,2])),
xlab = "Time[s]", ylab="LM-OSL signal[a.u.]")
legend("topright",bty="n",c("(b)"," ","LM-OSL"))
```

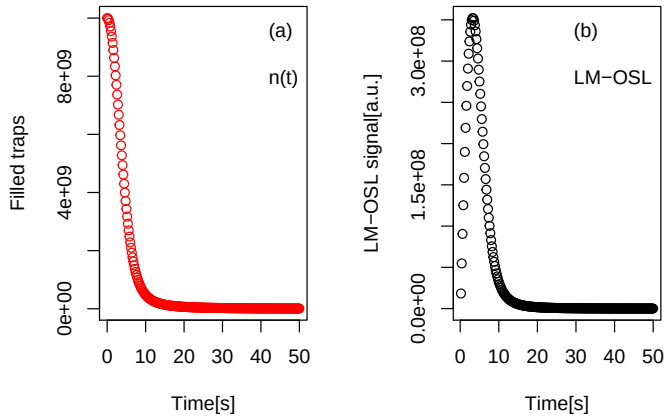


Fig. 3.9: Numerical solution of the GOT differential equation for LM-OSL. (a) Filled traps $n(t)$ and (b) The respective LM-OSL signal.

Code 3.10: Plot W0(x) solution of GOT equation, for LM-OSL

```
# plot analytical solution of GOT equation for LM-OSL with W(x)
rm(list=ls())
library(lamW)
## Plot the analytical solution of GOT, using Lambert W-function
t<-0:120
kB<-8.617E-5
no<-1E10
N<-1E10
R<-.46
c<-(no/N)*(1-R)/R
lambda<-.003
x<-unlist(t)
zLM<-unlist((1/c)-log(c)+(lambda*no/(c*N*R))*t^2/2)
# plots
par(mfrow=c(1,2))
plot(x,(N*R/(1-R))/(lambertW0(exp(zLM))),xlab="Time, s",
     ylab=expression("n(t)  [cm"^-3*"]"),ylim=c(0,N))
```

```
legend("topright",bty="n",c("(a)"," ","n(t)"))
plot(x,(N*R/((1-R)^2))*lambda*x/(lambertW0(exp(zLM))
 +lambertW0(exp(zLM))^2), xlab="Time, s",ylab="LM-OSL [a.u.]")
legend("topright",bty="n",c("(b)","LM-OSL","with W0"))
```

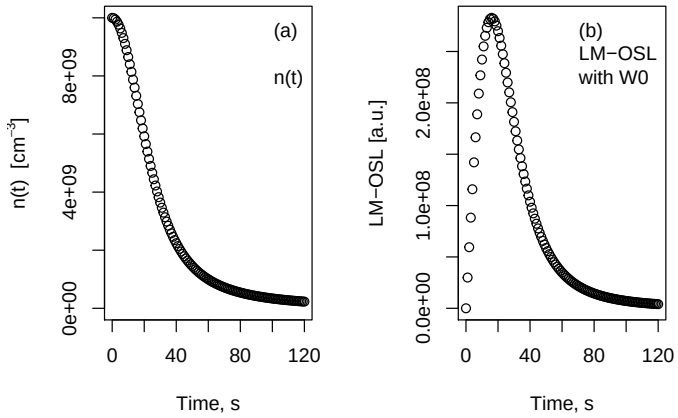


Fig. 3.10: Plots of the analytical solution of GOT equation Eq.(**??**) for LM-OSL, using the Lambert function *W[z]*. (a) Filled traps $n(t)$ and (b) The LM-OSL signal.

Code 3.11: Analysis of 3-component LM-OSL signal (numOSL)

```
### Example: Analyze LM-OSL signal using package numOSL
rm(list=ls())
library("numOSL")
CaF2LMx <- read.table("CaF2LMx.txt")
CaF2LMy <- read.table("CaF2LMy.txt")
d<-data.frame(CaF2LMx,CaF2LMy)
a<-decomp(d,ncomp=3,typ="lm",log="",
       control.args=list(maxiter=10))
print.noquote("Best fit parameters")
a$LMpars
cat("\nFOM=",a$FOM)
```

```
## [1] Best fit parameters
##             Ithn    seIthn      Lamda     seLamda
## Comp.1  44763.78  1218.961 1.83464448 0.059250636
## Comp.2  86822.86 11297.470 0.09116074 0.011548705
## Comp.3 191062.56 11639.885 0.01385274 0.001722475
##
## FOM= 5.037459
```

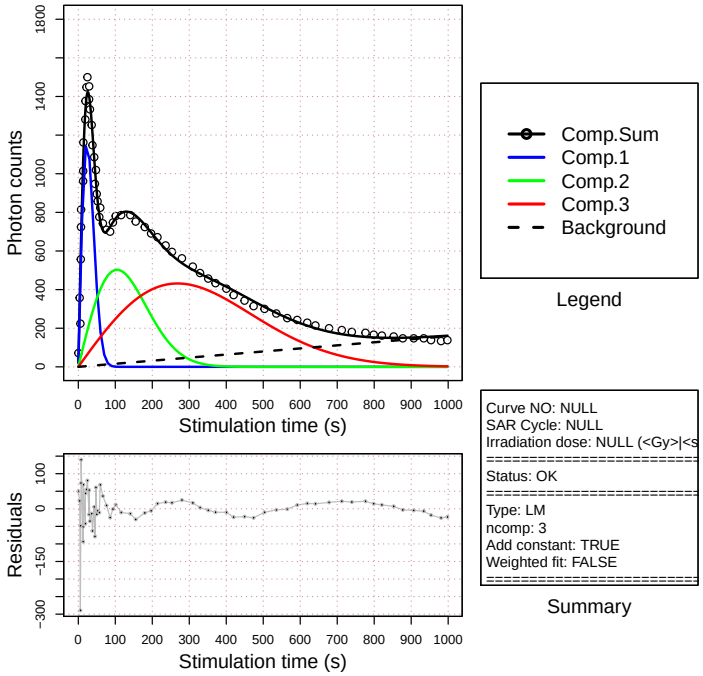


Fig. 3.11: Example of analyzing an LM-OSL signal from the dosimetric material CaF_2:N with three first order components, using the package *numOSL* (Kitis et al. [16]).

Code 3.12: Fit LM-OSL data with 3 first order components (Luminescence)

```
# fit  LM-OSL  with 3 exponentials using Luminescence package
rm(list=ls())
suppressMessages(library("Luminescence",warn.conflicts = FALSE))
## fit LM data with background subtraction
CaF2LMx <- read.table("CaF2LMx.txt")
CaF2LMy <- read.table("CaF2LMy.txt")
d<-data.frame(CaF2LMx,CaF2LMy)
bgdx<-seq(from=15,to=1000,by=15)
bgdy<-seq(from=3,to=200,by=3)
bgd<-data.frame(bgdx,bgdy)
invisible(capture.output(a<-fit_LMCurve(d,bgd,
n.components = 3, main=" ",
start_values = data.frame(Im = c(1500,800,500),
xm = c(30,130,200)))))
cat("\nImax values: ",a$data$Im1,a$data$Im2,a$data$Im3)
cat("\ntmax values: ",a$data$xm1,a$data$xm2,a$data$xm3)
cat("\nn0 values: ",a$data$n01,a$data$n02,a$data$n03)
cat("\nCross-section values (cm^2)")
cat("\n",a$data$cs1,a$data$cs2,a$data$cs3)
cat("\nR^2=",a$data$`pseudo-R^2`)
```

```
##
## Imax values:  1159.203 467.8042 456.2355
## tmax values:  23.24563 99.72249 248.6371
## n0 values:  44427.09 76913.83 187026.1
## Cross-section values (cm^2)
##  2.168303e-17 1.178193e-18 1.895269e-19
## R^2= 0.9864
```

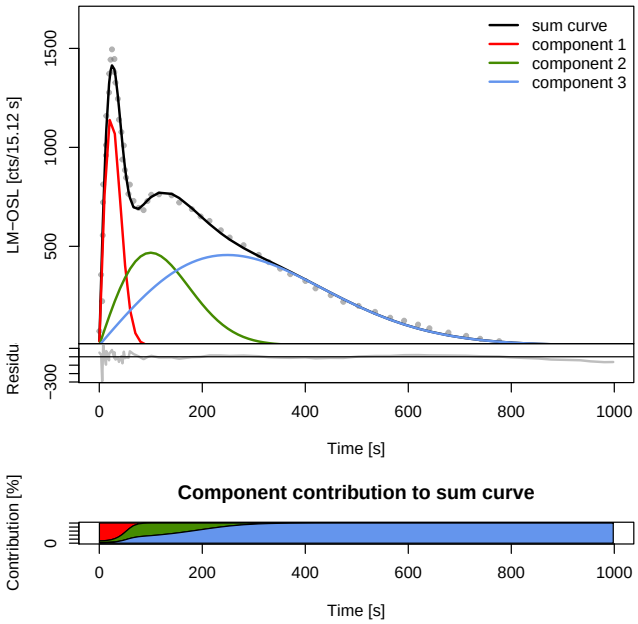


Fig. 3.12: Example of fitting LM-OSL data from the dosimetric material CaF_2:N with 3 first order components, using the package *Luminescence* (Kitis et al. [16]).

Code 3.13: Transform CW into pseudo-LM data (.txt data file)

```
##Read CWOSL data from .txt file and transform to pseudo-LMIRSL
rm(list=ls())
##produce x and y (time and count data for the data set)
par(mfrow=c(1,2))
CWdata <- read.table("KST4ph300IR.txt")
t<-CWdata[,1][1:100]
CW<-CWdata[,2][1:100]
u<-sqrt(2*t*max(t))
Iu<-u*CW/max(t)
plot(data.frame(t,CW),ylim=c(0,20000))
legend("topright",bty="n",legend=c("(a)"," ","CW-IRSL data",
  "KST4 feldspar"))
```

```
plot(u,Iu,xlab="u-parameter",typ="o",ylab="Pseudo-LM-IRSL",
     col="red",ylim=c(0,3200))
legend("topright",bty="n",legend=c("(b)"," ","Pseudo-LM-IRSL",
                                   "data"))
```

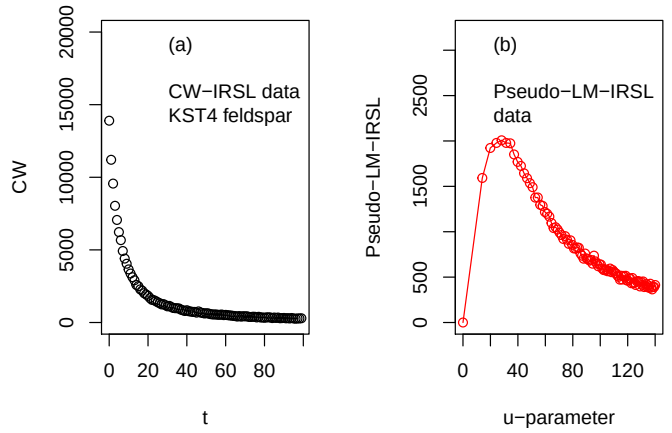


Fig. 3.13: Example of transforming a CW-IRSL signal into pseudo-LM-IRSL data, by reading a *.txt* file containing two columns, containing the time and the corresponding CW-intensities. The experimental data is discussed in Pagonis et al. [44].

Code 3.14: CW-IRSL into pseudo-LM-IRSL data (Luminescence package)

```
##Read CW-IRSL data from .txt file and transform to pseudo-LMOSL
rm(list=ls())
suppressMessages(library("Luminescence",warn.conflicts =FALSE))
##produce x and y (time and count data for the data set)
par(mfrow=c(1,2))
values <- read.table("KST4ph300IR.txt")
t<-values[,1][1:100]
CW<-values[,2][1:100]
data<-data.frame(t,CW)
plot(data,xlab="Time [s]",ylab="Original CW-IRSL [cts/s]",
```

```
     ylim=c(0,20000))
legend("topright",bty="n",legend=c("(a)"," ","CW-IRSL data",
          "KST4 feldspar"))
##transform values
data.transformed <- CW2pLM(data)
plot(data.transformed,xlab="u-parameter",typ="o",
     ylab="Pseudo-LMIRSL",col="red",ylim=c(0,3000))
legend("topright",bty="n",legend=c("(b)"," ","Pseudo-LM-OSL",
                                   "data"))
```

Chapter 4
DOSE RESPONSE OF DOSIMETRIC MATERIALS

Abstract In this chapter we discuss theoretical and experimental aspects of the dose response of dosimetric materials, and use analytical equations to fit experimental data. We define the superlinearity index $g(D)$, and the supralinearity index $f(D)$, and discuss various functions commonly used to describe shape of dose response curves in TL, OSL, OA and ESR signals: the saturating exponential, double saturating exponential, single exponential plus linear and the recently derived equation using the Lambert W function. We show how to numerically integrate the equations for the irradiation stage in the OTOR model, and compare with the analytical solution based on the Lambert W function. Several detailed R codes are given of fitting experimental data ESR, TL, OSL data, including situations with superlinear dose response. We simulate experiments in which the sample temperature is variable during the irradiation process, and which may affect the dose response of the material. We discuss superlinear dose response as the result of competition between two electron traps during the irradiation, and present experimental data analysis using the new analytical Pagonis-Kitis-Chen (PKC) equation which describes superlinearity effects. This chapter will conclude with an overview of the analytical dose response equations based on the Lambert function, and with a discussion of the importance of the Lambert function in the description of luminescence phenomena.

Code 4.1: Fit dose response data with saturating exponential

```
#Fit dose response data with Saturating Exponential
rm(list = ls(all=T))
options(warn=-1)
library("minpack.lm")
library("lamW")
```

```
## fit to saturation exponential ----
t = c( 0,50.7117,100.534,152.135,204.626,272.242)
y = c(0,33.144,42.205,43.1055,44.4157,43.7098)
fit_data <-data.frame(  t ,y)
plot(fit_data,ylim=c(0,max(y)),xlab="Dose [Gy]",
ylab="OSL (L/T)")
fit <- minpack.lm::nlsLM(
  formula = y ~ N * (1-exp(-b* t)),
  data = fit_data,
  start = list(N= max(y),b = .01))
N_fit <- coef(fit)[1]
b_fit <- coef(fit)[2]
## plot analytical solution
t1<-0:300
lines(x = t1,  y = N_fit  * (1-exp(-b_fit* t1)),col = "blue")
legend("right",bty="n",legend=c("Quartz OSL"," ",
"Libyan quartz"))
## print results
cat("\nfitted N: ", N_fit)
cat("\nfitted Do: ", round(1/b_fit,2), "Gy")
```

```
  ##
  ## fitted N:  44.16116
  ## fitted Do:  35.83 Gy
```

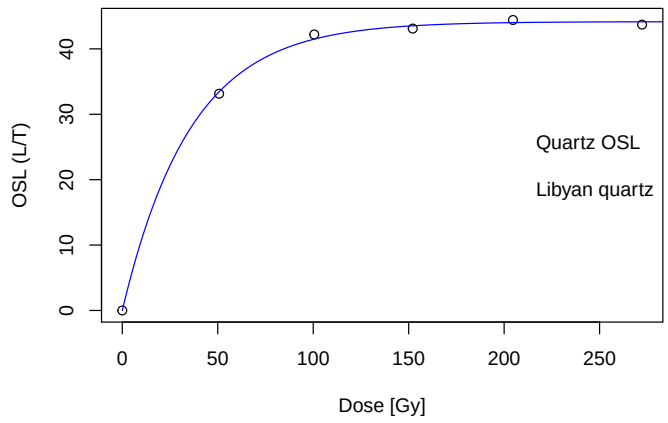


Fig. 4.1: Fitting dose response data with a saturating exponential. For more details see Pagonis et al. [38], original data from Li et al. [22].

Code 4.2: Irradiation: OTOR, Lambert analytical solution

```
# OTOR MODEL- IRRADIATION Lambert solution code
rm(list = ls(all=T))
options(warn=-1)
library(lamW)
library("deSolve")
## Plot  analytical solution of OTOR irradiation stage, using W
t<-0:100
N<-1E10
R<-.1
X<-1e10
k<-function(u){1+(1/(1-R))*lambertW0((R-1)*exp(-(R*X*u/N)+R-1))}
y1<-lapply(t,k)
x<-unlist(t)
y<-unlist(y1)
plot(x,y,xlab="Irradiation time t, s",ylab="Filled Traps n/N",
pch=1)
# Numerically Solve GOT equation for IRRADIATION  using  deSolve
n.0 <- 0  # initial concentration of filled traps
# Calculate numerical ODE solution
ODE <- function(t, state, parmameters){
  with(as.list(c(state, parameters)),{
    dn <-   (N-n) * X*R / ((N - n) * R + n )
    list(c(dn)) })}
parameters <- c(N=N, X = X, R =R)
state <- c(n = n.0)
num_ODE <- ode(y = state, times =t,func=ODE,parms=parameters)
lines(x = num_ODE[,1],y = num_ODE[,2]/N,xlab ="Time[s]",
type="l",col = "red")
legend("right",bty="n",pch=c(NA,1),legend =c("W(t)", "ODE"),
       col = c("red","black"), lwd = 1)
```

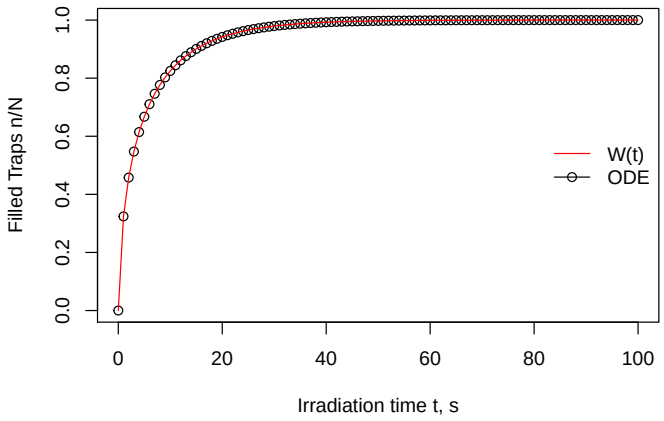


Fig. 4.2: Dose response in the OTOR model, using the Lambert analytical solution Eq.(**??**). The analytical solution (solid line) is compared with the numerical solution of the differential Eq.(**??**) shown as circles.

Code 4.3: Fit of experimental TL dose response data using W(x)

```
rm(list = ls(all=T))
options(warn=-1)
library("minpack.lm")
library("lamW")
## fit to Lambert equation  ----
t = c(0.00394056,0.00451523,0.39225,0.412035,0.703062,0.741318,
 0.742221,1.49553,1.49758,1.5158,2.98473,3.00304,3.02282,
 5.99852, 6.05755)
y = c(2.45103,2.80847,3.97964,4.28586,5.30465,5.10007,5.66177,
6.21706,7.49364,6.82965,8.50226,7.88934,8.19555,11.0809,11.7953)
fit_data <-data.frame(  t ,y)
plot(fit_data,ylim=c(0,max(y)),xlab="Dose [Gy]",
     ylab="OSL (L/T)",xlim=c(-.5,6.5))
fit <- minpack.lm::nlsLM(
  formula =y~N*(1+lambertW0((abs(R)-1)*exp(abs(R)-1-abs(b)*
   (t+abs(f))))/(1-abs(R))),
  data = fit_data,
```

```
  start = list(N= max(y),R=.9, b = .1,f=0.3))
N_fit <- coef(fit)[1]
R_fit <- abs(coef(fit)[2])
b_fit <- abs(coef(fit)[3])
f_fit <- abs(coef(fit)[4])
## plot analytical solution
t<- seq(from=-0.5,to=7,by=.02)
lines(
  x = t,
  y=N_fit*(1+lambertW0((R_fit-1)*exp(R_fit-1-b_fit*
                                       (t+f_fit)))/(1-R_fit)),
  col = "blue")
legend("right",bty="n",legend=c("Berger (1990)",
                                "Volcanic glass"," "))
## print results
cat("\nfitted N: ", N_fit)
cat("\nfitted R: ", R_fit)
cat("\nfitted Dc: ", round(1/b_fit,2), "Gy")
cat("\nfitted f: ", f_fit)
```

```
  ##
  ## fitted N:  19.14201
  ## fitted R:  5.276899e-06
  ## fitted Dc:  21.6 Gy
  ## fitted f:  0.2279443
```

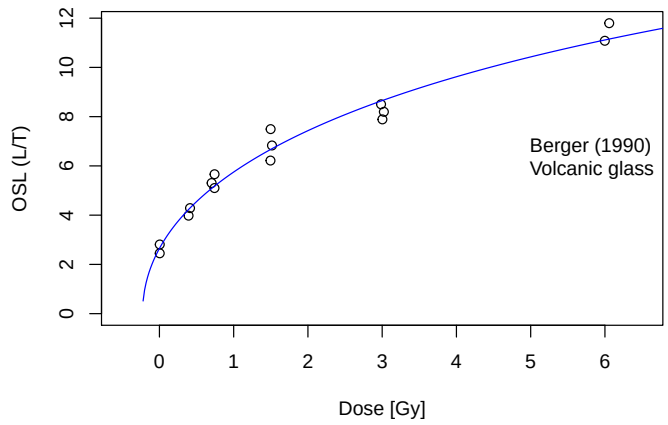


Fig. 4.3: Fit of experimental TL dose response data using the Lambert Eq.(**??**), with a non-zero intercept on the dose axis. For more details see Pagonis et al. [38], original data from Berger [2].

Code 4.4: Fit of experimental ESR dose response data using Lambert equation

```
rm(list=ls())
options(warn=-1)
library("minpack.lm")
library("lamW")
# Load the data
TLqzx<-c(174.13, 345.027, 931.847, 1603.74, 2524.39, 4031.12,
6372.18,9044.09,12217.5,15058.3,19981.5,25072.3,30082.3,40011.4)
TLqzy<-c(1.07478, 1.68389, 2.18591, 2.93875, 3.51271, 4.48122,
5.59377,6.3484,7.3184,8.39557,9.33133,10.4105,11.8478,13.6478)
plot(TLqzx,TLqzy,type="p",pch=1,col="red",
xlab=expression("Dose [Gy]"),ylab ="ESR signal [a.u.]",
ylim=c(0,20))
legend("topleft",bty = "n",
legend = c("Quartz"," ","ESR dose response"))
## fit to Lambert equation  ----
t <-TLqzx
y<-TLqzy
fit_data <-data.frame(  t ,y)
#plot(fit_data,ylim=c(0,max(y)))
fit <- minpack.lm::nlsLM(
formula=y~N*(1+lambertW0((abs(R)-1)*exp(abs(R)-1-b*t))/
(1-abs(R))),
  data = fit_data,
  start = list(N= max(y),R=.9, b = .01)
)
N_fit <- coef(fit)[1]
R_fit <- abs(coef(fit)[2])
b_fit <- coef(fit)[3]
## plot analytical solution
t<- seq(from=0,to=40000,by=100)
lines(  x = t,
y=N_fit*(1+lambertW0((R_fit-1)*exp(R_fit-1-b_fit*t))/(1-R_fit)),
  col = "blue")
```

```
## print results
cat("\nfitted N: ", N_fit)
cat("\nfitted R: ", R_fit)
cat("\nfitted Dc: ", 1/b_fit, " Gy")

  ##
  ## fitted N:  51.24355
  ## fitted R:  1.432573e-08
  ## fitted Dc:  995428.2  Gy
```

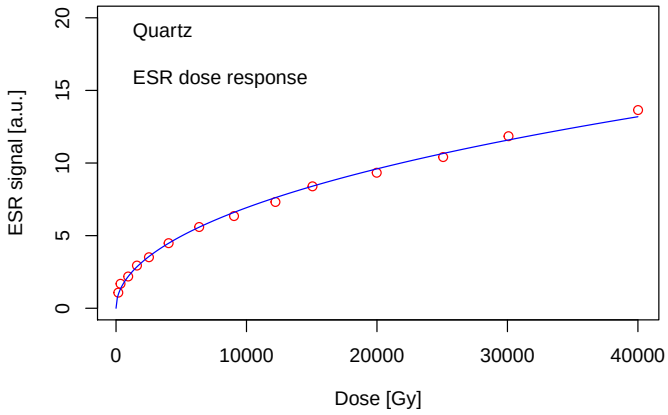


Fig. 4.4: Fit of experimental ESR dose response data using the Lambert Eq.(**??**). For more details see Pagonis et al. [38], original data from Duval [9].

Code 4.5: Fit of experimental OSL dose response data using W(x)

```
rm(list = ls(all=T))
options(warn=-1)
library("minpack.lm")
library("lamW")
par(mfrow=c(1,2))
## fit to saturation exponential ----
t = c(-34.2466, 34.2466, 68.4932, 273.973, 1027.4,1986.3,3013.7,
      5000, 7979.45, 10000)
```

```
y = c(1.04664, 0.000978474, 2.76386, 7.24592,12.6008,14.5329,
      15.8956,  17.1905, 17.847, 18.0952)
fit_data <-data.frame(  t ,y)
plot(fit_data,ylim=c(0,max(y)),xlab="Dose [Gy]",
ylab="OSL (L/T)")
fit <- minpack.lm::nlsLM(
formula=y~N*(1+lambertW0((abs(R)-1)*
exp(abs(R)-1-b*t))/(1-abs(R))),
  data = fit_data,
  start = list(N= max(y),R=.9, b = .01))
N_fit <- coef(fit)[1]
R_fit <- abs(coef(fit)[2])
b_fit <- coef(fit)[3]
## plot analytical solution
t<- seq(from=0,to=10000,by=100)
lines(  x = t,
y=N_fit*(1+lambertW0((R_fit-1)*exp(R_fit-1-b_fit* t))/
(1-R_fit)),  col = "blue")
legend("right",bty="n",legend=c("(a)"," ","Fine grain",
"Quartz"," "))
## print results
cat("\nFine grain"," ")
cat("\nfitted N: ", N_fit)
cat("\nfitted R: ", R_fit)
cat("\nfitted Dc: ",round( 1/b_fit,2), "Gy")
## fit to Lambert equation ----
t = c(0, 3.5583, 44.1822, 258.718, 1051.62, 2044.98, 3003.94,
      5024.61, 7046.32, 9992.29)
y = c(0, 0.93512, 2.61108, 4.99104, 6.36704, 6.42148,
      6.43643, 6.46792, 6.77215, 6.97391)
fit_data <-data.frame(  t ,y)
plot(fit_data,ylim=c(0,max(y)),xlab="Dose[Gy]",ylab="OSL (L/T)")
fit <- minpack.lm::nlsLM(
formula=y~N*(1+lambertW0((abs(R)-1)*
exp(abs(R)-1-b*t))/(1-abs(R))),
  data = fit_data,
  start = list(N= max(y),R=10, b = 1e-4))
N_fit <- coef(fit)[1]
R_fit <- abs(coef(fit)[2])
b_fit <- coef(fit)[3]
## plot analytical solution
t<- seq(from=0,to=10000,by=100)
lines(  x = t,
y=N_fit*(1+lambertW0((R_fit-1)*exp(R_fit-1-b_fit*t))/(1-R_fit)),
```

```
  col = "blue")
legend("right",bty="n",legend=c("(b)"," ","Coarse grain",
"Quartz"," "))
## print results
cat("\nCoarse grain"," ")
cat("\nfitted N: ", N_fit)
cat("\nfitted R: ", R_fit)
cat("\nfitted Dc: ",round( 1/b_fit,2)," Gy")

  ##
  ## Fine grain
  ## fitted N:  17.57116
  ## fitted R:  0.3971654
  ## fitted Dc:  1325.83 Gy
  ## Coarse grain
  ## fitted N:  6.610193
  ## fitted R:  1.938374e-06
  ## fitted Dc:  403.02  Gy
```

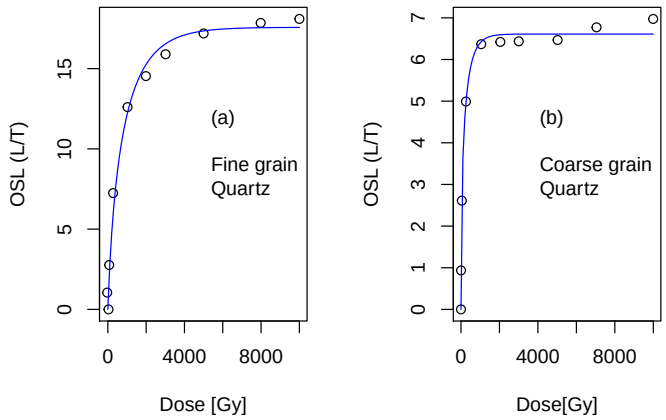


Fig. 4.5: Fit of experimental SAR-OSL experimental dose response data, for (a) fine grain and (b) coarse grain quartz samples, using the Lambert equation. For more details see Pagonis et al. [38], original data by Timar-Gabor et al. [51] (their Figure 3).

Code 4.6: Irradiation of thermally unstable trap (OTOR)

```
# OTOR MODEL- IRRADIATION of thermally unstable trap (OTOR)
rm(list = ls(all=T))
options(warn=-1)
library(lamW)
library("deSolve")
## Plot the numerical solution of OTOR irradiation stage,
## at different irradiation temperatures
t<-0:100
N<-1E10
R<-.1
X<-1e10
E<-1
s<-1e12
kb<-8.617e-5
TISO<-20
p<-s*exp(-E/(kb*(273+TISO)))
# Numerically Solve GOT equation for IRRADIATION  using  deSolve
n.0 <- 0  # initial concentration of filled traps
# Calculate numerical ODE solution
ODE <- function(t, state, parmameters){
  with(as.list(c(state, parameters)),{
    dn <-  ( (N-n) * X*R-(p*(n**2))) / ((N - n) * R + n )
      list(c(dn))  })}
parameters <- c(N=N, X = X, R =R, p=p)
state <- c(n = n.0)
num_ODE <- ode(y = state,times = t,func = ODE,parms=parameters)
plot(x = num_ODE[,1], y = num_ODE[,2]/N,pch=2,ylim=c(0,1.1),
xlab = "Time[s]", ylab = "Trap filling ratio, n/N",type="b",
      col = "red")
# Change the irradiation temperature TISO
TISO<-100
p<-s*exp(-E/(kb*(273+TISO)))
parameters <- c(N=N, X = X, R =R, p=p)
num_ODE2 <- ode(y = state, times = t, func=ODE,parms=parameters)
lines(x = num_ODE2[,1],    y = num_ODE2[,2]/N, pch=3,
col = "Black",type="b")
legend("right",bty="n",pch=c(2,3),expression("T=" ~ 20^o ~C ~"",
"T=" ~ 100^o ~C ~""),col=c("red","black"),lwd=2)
```

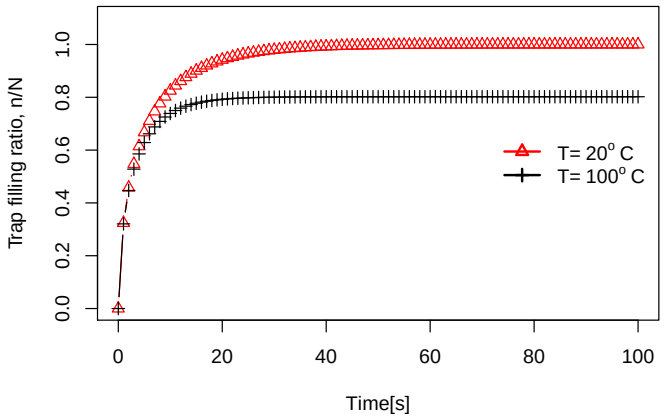


Fig. 4.6: Simulation of the effect of sample temperature during the irradiation process, within the OTOR model. For a more detailed study of this effect, see Chen et al. [7].

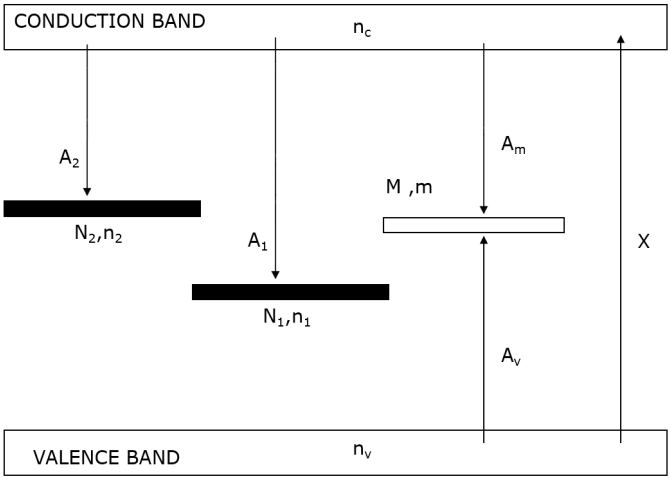


Fig. 4.7: The 2T1C model of Bowman and Chen [4], describing competition effects during the irradiation stage. Various electronic transitions are shown during the irradiation process. Transition X: Creation of electron-hole pairs by radiation. Transitions A_v and A_m: Trapping of holes and recombination of electrons at the recombination centers (RC). Transitions A_1,A_2: Trapping of electrons in the dosimetric and competitor trap. For more details, see Pagonis et al. [39] .

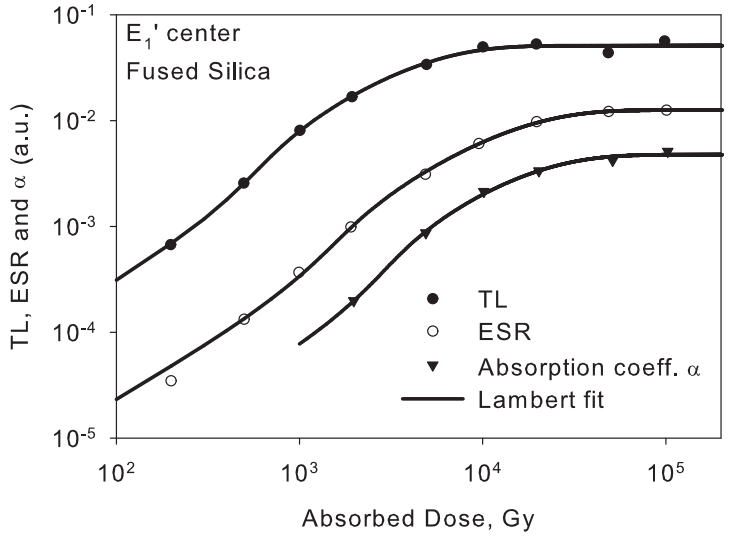


Fig. 4.8: Superlinear dose dependence of the E_1' center concentration (ESR), TL and OA signals, from a single sample of fused silica. Note the log scale in both axes. The solid lines are fitted using the PKC Eq.(**??**). For more details see Pagonis et al. [39], original data from Wieser [53].

Code 4.7: TL dose response of anion deficient aluminum oxide

```
rm(list = ls(all=T))
options(warn=-1)
library("minpack.lm")
library("lamW")
## fit to Pagonis-Kitis-Chen PKC superlinearity equation ----
t=c(0.0537568,0.103385,0.156481,0.211929,0.260776,
0.321015,0.36483,0.414625,0.478926,0.535569,0.589476,
0.638899,0.824344,0.951985,1.18991,1.63688,3.19332)
y=c(6694.89,15592.6,24767.6,39360.5,52176.2,78075.6,
101463,131855,189548,227223,272406,376197,469268,
634929,792121,1.16105e6,1.6992e6)
fit_data <-data.frame(  t ,y)
fit <- minpack.lm::nlsLM(
formula=y~N*(1-(lambertW0(abs(B)*exp(abs(B)-lamda*t))/
abs(B))**beta),
  data = fit_data,
```

```
  start = list(N= max(y),B=5, lamda = 10,beta=0.1))
N_fit <- coef(fit)[1]
B_fit <- coef(fit)[2]
lamda_fit <- coef(fit)[3]
beta_fit <- coef(fit)[4]
## print results
cat("\nfitted N: ", N_fit)
cat("\nfitted B: ", B_fit)
cat("\nfitted Dc: ", round(1/lamda_fit,2)," Gy")
cat("\nfitted beta: ", beta_fit)
## plot analytical solution
t<- seq(from=0,to=4,by=.01)
plot(fit_data,log="xy",xlab="Dose [Gy]",ylab="TL")
legend("topleft",bty="n",legend=c("Nikiforov et al. (2014)",
"Anion-deficient","Aluminum Oxide"))
lines( x = t,
y=N_fit*(1-(lambertW0(abs(B_fit)*exp(abs(B_fit)-lamda_fit*
t))/abs(B_fit))**beta_fit),
       col = "blue",log="xy")

  ##
  ## fitted N:  2095289
  ## fitted B:  6.870824
  ## fitted Dc:  0.05  Gy
  ## fitted beta:  0.02999771
```

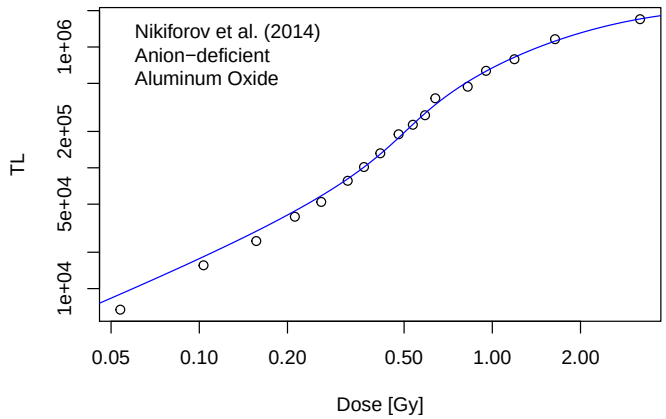


Fig. 4.9: TL dose response of an anion deficient aluminum oxide single crystal, which had low initial sensitivity to irradiation. The superlinear dose response is fitted with the PKC equation. For more details see Nikiforov et al. [26], see also the discussion in Pagonis et al. [39].

Code 4.8: Fit to Supralinearity index f(D) using Lambert W

```
rm(list = ls(all=T))
options(warn=-1)
library("minpack.lm")
library("lamW")
## fit g(D) to Lambert equation ----
t = c(0.0811131, 0.171804,0.450923,0.857988,1.88341,
4.44069,8.44947,17.2683,40.7152,83.2104,176.246,
415.552,819.456,1674.74,3948.68,8983.32)
y = c(1.03326,0.983911,1.08074,1.10465,1.12844,1.28023,
1.37731,1.50481,1.76636,1.79021,1.45428, 0.813382,
0.428722,0.208669,0.0799713,0.0305689)
fit_data <-data.frame(  t ,y)
fit <- minpack.lm::nlsLM(
  formula = y ~ N * (1-(lambertW0(abs(B)*exp(abs(B)- t/Dc))/
  abs(B))**beta)/t,
  data = fit_data,
  start = list(N= max(y),B=1.2, Dc = 5,beta=0.1)
)
N_fit <- coef(fit)[1]
B_fit <- abs(coef(fit)[2])
Dc_fit <- coef(fit)[3]
beta_fit <- coef(fit)[4]
## print results
cat("\nfitted N: ", N_fit)
cat("\nfitted B: ", B_fit)
cat("\nfitted Dc: ", round(Dc_fit,2)," Gy")
cat("\nfitted beta: ", beta_fit)
## plot analytical solution
t<- seq(from=0.01,to=10000,by=.2)
plot(fit_data,log="x",ylab="Supralinearity Index f(D)",
xlab="Irradiation Dose, Gy")
lines( x = t,
y=N_fit*(1-(lambertW0(abs(B_fit)*exp(abs(B_fit)-t/Dc_fit))/
abs(B_fit))**beta_fit)/t,       col = "blue",log="x")
legend("topright",bty="n",legend=c("Edmund (2007)",
```

```
"figure 5.11","Probe A"))

  ##
  ## fitted N:  375.0032
  ## fitted B:  1.237203
  ## fitted Dc:  5.07  Gy
  ## fitted beta:  0.03240109
```

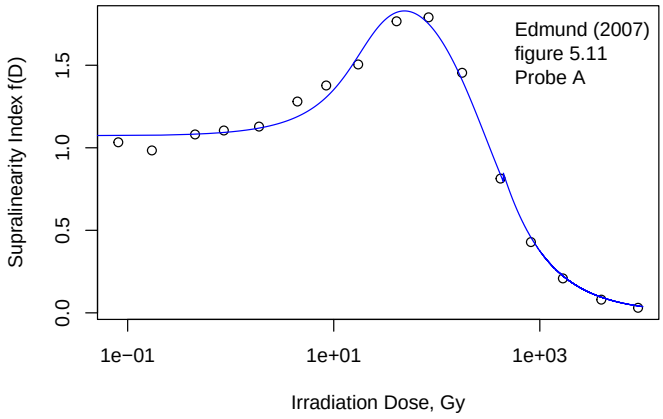


Fig. 4.10: Fit to the superlinear behavior of an Al_2O_3:C probe using the PKC superlinearity equations. For more details of the analysis see Pagonis et al. [39], original experimental data from Edmund [10].

Chapter 5
TIME-RESOLVED OSL EXPERIMENTS

Abstract In this chapter we discuss experimental data and models for time-resolved optically and infrared stimulated luminescence (TR-OSL, TR-IRSL).These techniques can help researchers understand the luminescence mechanisms involved in dosimetric materials. We show how to use R in order to extract the luminescence lifetimes from TR experimental data, and show how to analyze the temperature dependence of luminescence lifetimes and luminescence intensity. Specific R codes are provided for TR experiments in quartz and how they can be analyzed with R to obtain the thermal quenching parameters W and C, based on the Mott-Seitz competition mechanism. We show how to analyze TR-OSL signals from *delocalized* transitions and also TR-IRSL signals involving *localized* transitions in feldspars involving quantum tunneling. This chapter concludes with the presentation of a TR-photoluminescence model (TR-PL) for the important dosimetric material Al_2O_3:C.

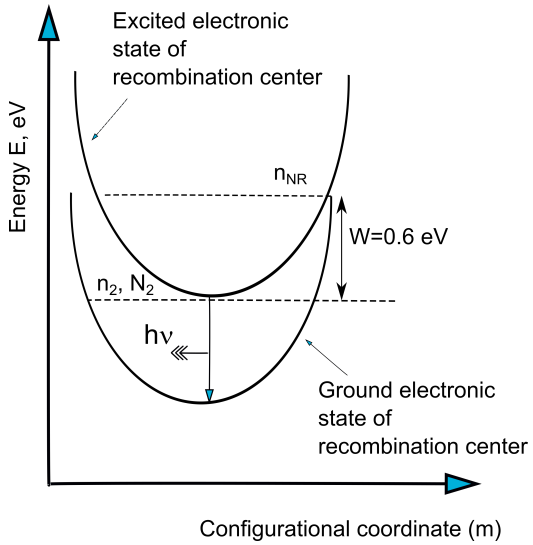


Fig. 5.1: The configurational diagram for quartz, based on the Mott-Seitz mechanism of thermal quenching. From Pagonis et al. [30].

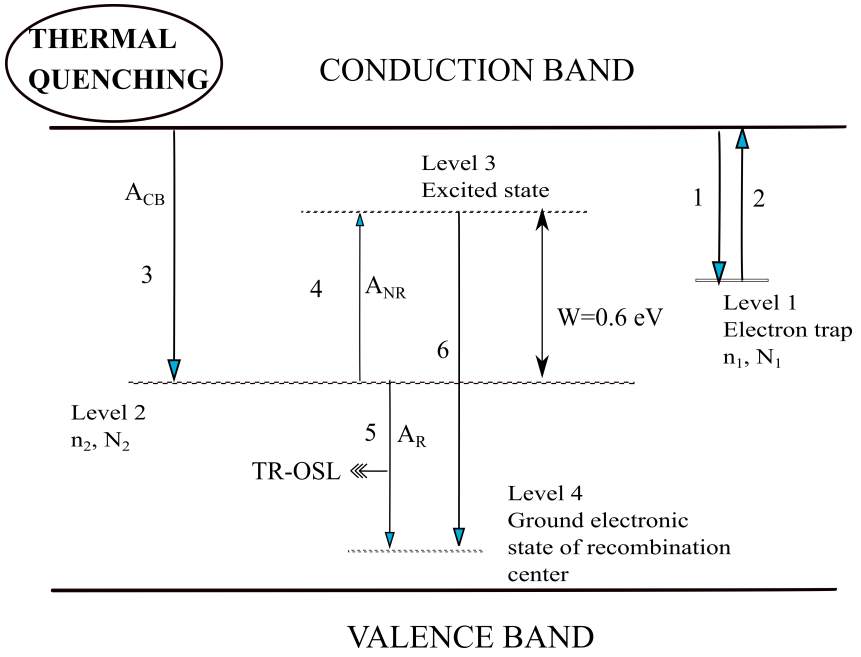


Fig. 5.2: The kinetic model of Pagonis et al. [30] for thermal quenching in quartz, based on the Mott-Seitz mechanism.

Code 5.1: Analysis of TR-OSL experimental data in quartz

```
rm(list=ls())
library("minpack.lm")
mydata <- read.table("chithamboTROSLqzdata.txt")
plot(mydata,pch=2,col="red",  ylab="TR-OSL [a.u.]",
     xlab=expression(paste("Time [",mu,"s]")))
legend("topright",bty="n",legend=c("Sedimentary","quartz"," ",
"470 nm","LEDs"))
# fit ON data
t<-mydata[,1][1:20]
y<-mydata[,2][1:20]
fit_data <-data.frame(t ,y)
fit <- minpack.lm::nlsLM(
```

```
formula = y ~ N * (1-exp(- t/b))+abs(bgd), data = fit_data,
  start = list(N= max(y),b = 40,bgd=10))
t1<-0:55
lines(x = t1,  y = coef(fit)[1]  *
(1-exp(- t1/coef(fit)[2]))+abs(coef(fit)[3]),col = "blue")
coef(fit)
# fit OFF data
t<-mydata[,1][21:length(mydata[,1])]-51.27
y<-mydata[,2][21:length(mydata[,1])]
fit_data <-data.frame(t ,y)
fit <- minpack.lm::nlsLM(
  formula = y ~ N * (exp(- t/b))+abs(bgd),data = fit_data,
  start = list(N= max(y),b = 40,bgd=10))
t1<-5:300
lines(x=t1+51.2,y=coef(fit)[1]*(exp(-(t1)/coef(fit)[2]))+
abs(coef(fit)[3]),col = "blue")
coef(fit)
```

```
##           N           b         bgd
## 254.2423690  38.0707228   0.8669149
##          N          b        bgd
## 221.49526  38.47338   2.67049
```

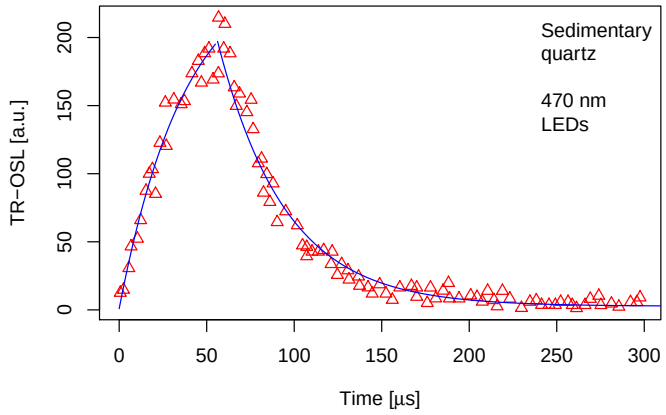


Fig. 5.3: Examples of TR-OSL curves for sedimentary quarts with 60 μs pulse. For more details see Chithambo et al. [8].

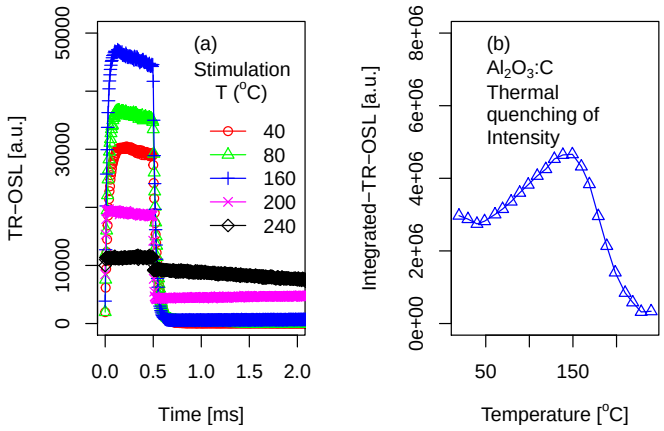


Fig. 5.4: Experimental dependence of (a) the TR-OSL luminescence signals, and (b) of the integrated TR-OSL intensity, on the stimulation temperature for Al_2O_3:C sample. For more details see Pagonis et al. [28].

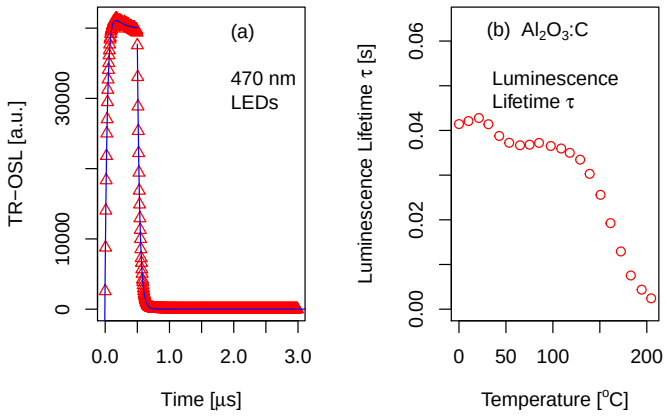


Fig. 5.5: (a) Analysis of the TR-OSL signal measured at a stimulation temperature of 100°C for an Al_2O_3:C sample, using exponential functions shown as solid lines. The signal contains a slower temperature-dependent phosphorescence signal, sometimes referred to as the "delayed-OSL" signal. (b) Experimental dependence of the luminescence lifetime τ on the stimulation temperature for the same sample. For more details see Pagonis et al. [28].

Code 5.2: Analysis of TR-OSL experimental data in alumina

```
rm(list=ls())
library("minpack.lm")
par(mfrow=c(1,2))
aluminatau470nmx<-unlist(read.table("aluminax1.txt"))
aluminatau470nmy<-unlist(read.table("aluminay1.txt"))
x<-aluminatau470nmx[8:20]+273
y<-aluminatau470nmy[8:20]
kB<-8.617*1e-5 # Boltzmann constant in eV/K
fit_data <-data.frame(x ,y)
plot(aluminatau470nmx+273,aluminatau470nmy,
xlab=expression("Temperature [K]"),
ylab="Integrated-TR-OSL [a.u.]")
legend("topright",bty="n","(a)")
legend("bottomleft",bty = "n", legend =
c(expression('Al'[2]*'O'[3]*':C',' ',
 'Thermal','quenching',' ','Fit to find C,W')))
fit <- minpack.lm::nlsLM(
  formula = y ~ N /(1+c*exp(-W/(kB*x))),data = fit_data,
  start = list(N= max(y),c=1e6, W =1))
x1<-seq(from=350,to=500,by=1)
lines(x=x1,y=coef(fit)[1]/(1+coef(fit)[2]*
exp(-coef(fit)[3]/(kB*x1))),       col = "blue")
coef(fit)
al2o3risoOSLvsTempx<-unlist(read.table("aluminax2.txt"))
al2o3risoOSLvsTempy<-unlist(read.table("aluminay2.txt"))
x<-al2o3risoOSLvsTempx[4:10]
y<-al2o3risoOSLvsTempy[4:10]
y<-log(y)
x<-1/(kB*(x+273.15))
bestfit<-lm(y~x)
coefficients(bestfit)
plot(x, y, xlab = "1/(kT)   [1/eV]",ylab = "ln(TR-OSL)")
legend("topright",bty = "n", legend =c('(b)',expression(' ',
```

```
'Fit to find E'[th]*' ')))
abline(lm(y~x))

  ##             N             c             W
  ## 3.698208e-02 5.659581e+11 1.014901e+00
  ## (Intercept)           x
  ## 17.19471165 -0.06542585
```

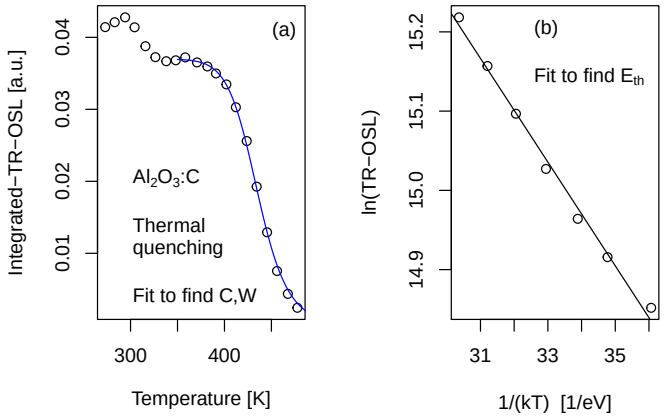


Fig. 5.6: Experimental determination of the thermal quenching parameters C, W, E_{th}. (a) The values of C, W are obtained by fitting the *decreasing* part of the data. The best fit is shown by the solid line. (b) The thermal activation energy E_{th} is obtained with an Arrhenius analysis of the *increasing* part of the data in (a). For more details see Pagonis et al. [28].

Code 5.3: Fit microcline TR-IRSL data with analytical equation

```
#Fit FL1 TR-IRSL data with analytical equation
rm(list = ls(all=T))
options(warn=-1)
library("minpack.lm")
```

```
par(mfrow=c(1,2))
## fit ON data to analytical TR-IRSL equation ----
mydata <- read.table("FL1ONdata.txt")
t<-1e-6*mydata[,1]
y<-mydata[,2]
mydata<-data.frame(t,y)
plot(t*1e6,y,xlab="Time [s]",ylab="TR-IRSL [Normalized]",
col="black",pch=1)
fit_data <-mydata
fit <- minpack.lm::nlsLM(
  formula=y~ imax*(1-exp (-rho*(log(1 + A*t)) ** 3.0))+bgd,
  data = fit_data,
  start = list(imax=3,A=1e6,rho=0.001,bgd=min(y)))
imax_fit <- coef(fit)[1]
A_fit <- coef(fit)[2]
rho_fit <- coef(fit)[3]
bgd_fit <- coef(fit)[4]
## plot analytical solution
t1<-seq(from=1e-7,to=5e-5,by=1e-7)
lines(
  x = t1*1e6,
  y =imax_fit*(1-exp (-rho_fit*(log(1 + A_fit*
t1)) ** 3.0))+bgd_fit,   col = "red",lwd=2)
legend("topleft",bty="n","(a)")
legend("right",bty="n", pch=c(NA,NA,NA,1,NA),lwd=2,
       lty=c(NA,NA,NA,NA,"solid"),
       c(expression('TR-IRSL','Microcline',' ',
                    'Experiment','Analytical')),
col=c(NA,NA,NA,"black","red"))
## print results
cat("\nParameters from Least squares fit")
cat("\nImax=",formatC(imax_fit,format="e",digits=2)," cts/s",
       sep="    ","A=",round(A_fit,digits=2)," (s^-1)")
cat("\nrho=",round(rho_fit,digits=4),sep=" ",
       "  bgd=",round(bgd_fit,digits=2)," cts/s")
## Use same parameters to fit the OFF data
mydata <- read.table("FL1OFFdata.txt")
t<-1e-6*mydata[,1]
y<-mydata[,2]
mydata<-data.frame(t,y)
plot(t*1e6,y,xlab="Time [s]",ylab="TR-IRSL [Normalized]",
col="black",pch=1)
## plot analytical solution
t1<-seq(from=5e-7,to=1e-4,by=1e-7)
```

```
lines(
  x = t1*1e6,
  y =.06+ imax_fit*(exp (-rho_fit*(log(1 + A_fit*t1)) ** 3.0)-
exp (-rho_fit*(log(1 + A_fit*(t1+5e-5))) ** 3.0))+bgd_fit,
  col = "red",lwd=2)
legend("topleft",bty="n","(b)")
legend("right",bty="n", pch=c(NA,NA,NA,1,NA),lwd=2,
       lty=c(NA,NA,NA,NA,"solid"),
       c(expression('TR-IRSL','OFF data',' ',
'Experiment','Analytical')),col=c(NA,NA,NA,"black","red"))

  ##
  ## Parameters from Least squares fit
  ## Imax=    1.03e+00     cts/s    A=    6592795     (s^-1)
  ## rho= 0.0212   bgd= -0.04  cts/s
```

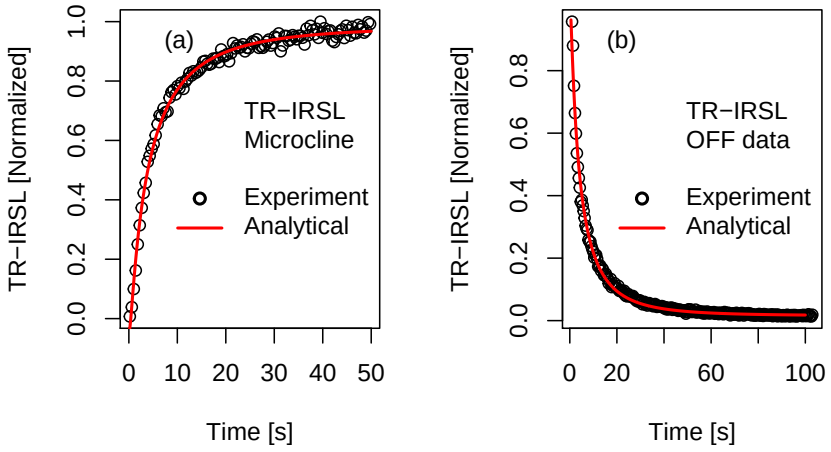


Fig. 5.7: Fit TR-IRSL data for microcline sample FL1, with the analytical Eqs.(**??**) and (**??**). (a) The IR excitation is ON (b) The excitation is OFF. For more details see Pagonis et al. [29].

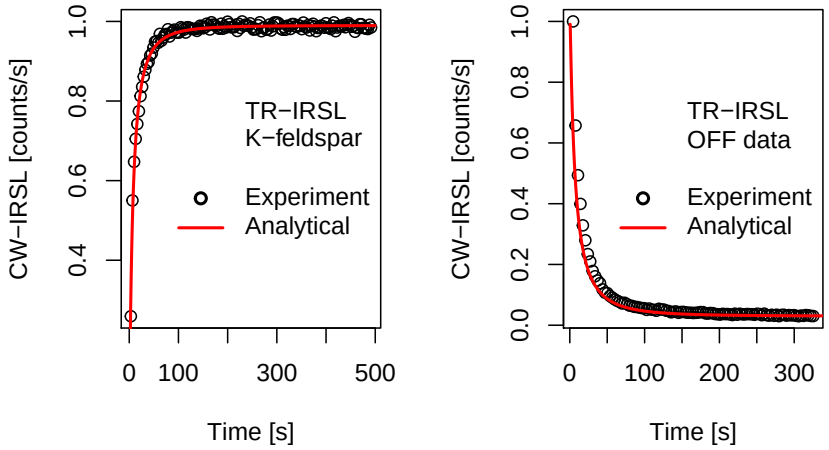


Fig. 5.8: Fit of K-rich feldspar TR-IRSL data with analytical Eqs.(**??**) and (**??**). (a) The IR excitation is ON (b) The excitation is OFF. For more details see Pagonis et al. [29].

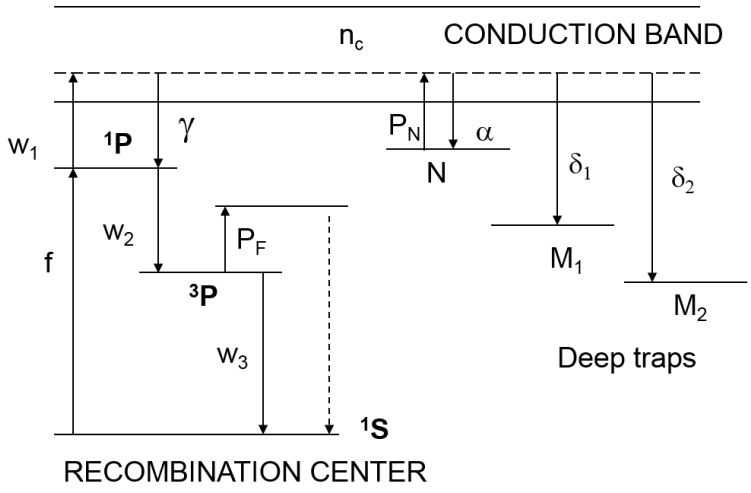


Fig. 5.9: The TR-PL model for time-resolved photoluminescence (TR-PL) experiments in α-Al_2O_3:C by Pagonis et al. [34], based on an earlier version of the model by Nikiforov et al. [27].

Code 5.4: Simulation of TR-PL experiments in Al2O3:C

```
#Simulate TR-PL experiments with Nikiforov/Pagonis model
rm(list = ls(all=T))
library("deSolve")
PagonisAlumina <- function(t, x, parms) {
  with(as.list(c(parms, x)), {
    dn<- -s*exp(-E/(kb*(273+T)))*n+alpha*(N-n)*nc
    dm1<- delta1*(M1-m1)*nc
    dm2<- delta2*(M2-m2)*nc
    dnc<- s*n*exp(-E/(kb*(273+T)))-delta1*(M1-m1)*nc-
    delta2*(M2-m2)*nc-Gamma*nF*nc-alpha*(N-n)*nc+w1*n1P
    dnF<- -Gamma*nF*nc+w1*n1P
    dn3P<-w2*n1P-C*exp(-W/(kb*(273+T)))*n3P-w3*n3P
    dn1P<-f+Gamma*nF*nc-w1*n1P-w2*n1P
    res <- c(dn,dm1,dm2,dnc,dnF,dn3P,dn1P)
    list(res)  })}

TempVar<-function(T){
parms<- c(E=1.3, s=1e13,alpha=1e-14, delta1=1e-12, delta2=1e-14,
  Gamma=1e-11,N=1e13,M1=1e15,T=T,M2=1e14,C=1e13,W=1,w1=1,w2=3e3,
  w3=w3,f=1e10,kb=8.617e-5)
  y <- xstart <- c(n = 0, m1=1e14,m2=0,nc=0,nF=1e14,n3P=0,n1P=0)
  out <-  ode(xstart, times, PagonisAlumina, parms)  }
w3<-29
times  <- seq(0, .2,by=.005)
Temps<-seq(70,220,10)
Lc<-temps<-matrix(NA,nrow=length(times),ncol=length(Temps))
areaLc<-vector(length=length(Temps))
for (i in 1:length(Temps)){
  T<-Temps[i]
  a<-TempVar(T)
  Lc[,i]<- w3*a[,"n3P"]
  areaLc[i]<-sum(Lc[,i],rm.NA=TRUE)}
par(mfrow=c(1,2))
plot(times,Lc[,6],typ="o",col="black",xlim=c(0,.2),
ylim=c(0,1.7e10),pch=1, xlab="Time [s]",
ylab="TR-OSL Intensity [a.u.]")
lines(times,Lc[,12],typ="o",col="red",pch=2)
```

```
lines(times,Lc[,15],typ="o",col="blue",pch=3)
legend("topleft",bty="n",pch=c(NA,1,2,3),
lty=c(NA,rep("solid",3)),col=c(NA,"black",
"red","blue"),legend=c(expression("(a)  Al"[2]*"O"[3]*":C",
"120"^o*"C","180"^o*"C","210"^o*"C")))
plot(Temps,areaLc,typ="o",col="blue",lwd=2,pch=1,ylim=c(0,6e11),
xlab=expression("Stimulation Temperature"^o*"C"),
ylab="TR-OSL Areas [a.u.]")
legend("topleft",bty="n",legend=c("(b)"," ","Thermal quenching",
  "TR-OSL Areas"))
```

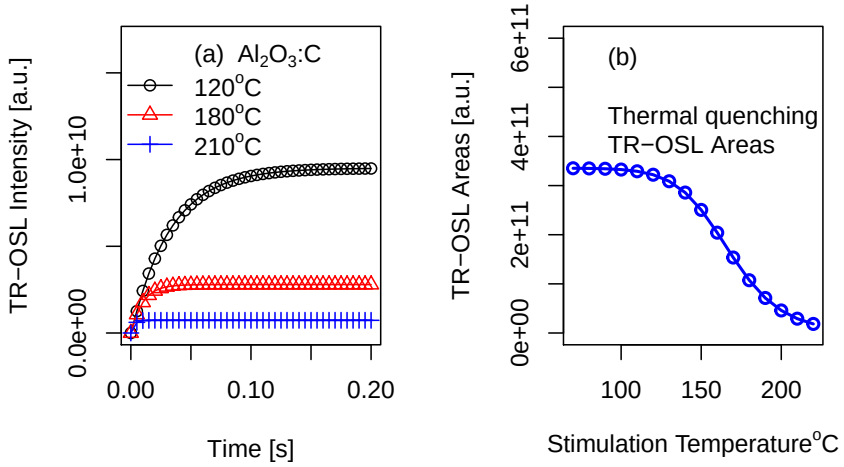


Fig. 5.10: Simulation of TR-PL experiments for Al_2O_3:C while the light excitation is ON. (a) The TR-OSL intensity at three different stimulation temperatures T =120, 180, 210°C; (b) The area under the TR-OSL curves in (a), is plotted as a function of the stimulation temperature and shows the effect of thermal quenching. For more details see Pagonis et al. [34].

Part II
LUMINESCENCE SIGNALS FROM LOCALIZED TRANSITIONS

Chapter 6
LOCALIZED TRANSITIONS AND QUANTUM TUNNELING

Abstract In this chapter we consider the different types of quantum tunneling localized transition models (TLT) models. We describe four different types of TLT models: ground state tunneling models (GST), irradiation ground state tunneling models (IGST), excited state tunneling models (EST) and thermally-assisted excited state tunneling models (TA-EST). We provide R codes for exploring the properties of each model, discuss their physical principles, and code approximate analytical solutions to the differential equations describing each model. R codes are provided for simulating the nearest neighbor distribution in a random distribution of defects in a solid, and also provide an example of analyzing experimental data to obtain the g-factor for the anomalous fading phenomenon (AF). Additional R codes simulate simultaneous irradiation and tunneling, and excited state tunneling phenomena. We show how to analyze experimental TL and OSL data for freshly irradiated samples, using the analytical Kitis-Pagonis equations KP-TL and KP-CW. Finally we present the thermally-assisted excited state model used in low temperature thermochronometry studies, and show how quantum tunneling phenomena can be simulated using the TUN functions in the package *RLumCarlo*.

Code 6.1: The nearest neighbors distribution

```
# Fig 1 in Pagonis and Kulp paper
# Original Mathematica Program written by V Pagonis
# R  version written by Johannes Friedrich
rm(list = ls(all = TRUE)) # empties the environment
library("plot3D")
library("FNN")
## Define Parameters ----
sideX <- 100e-9 # lenght of quader in m
```

```
sideX_nm <- sideX*1e9 # length of quader in nm
N_pts <- 50
alpha <- 9e9
N_centers <- 300
rho <- N_centers/sideX^3
r_prime <- function(r) (4*pi*rho/3)^(1/3) * r * 1e-9
xyz_traps <- data.frame(
  x = sample(1:sideX_nm, N_pts, replace = TRUE),
  y = sample(1:sideX_nm, N_pts, replace = TRUE),
  z = sample(1:sideX_nm, N_pts, replace = TRUE))
xyz_centers <- data.frame(
  x = sample(1:sideX_nm, N_centers, replace = TRUE),
  y = sample(1:sideX_nm, N_centers, replace = TRUE),
  z = sample(1:sideX_nm, N_centers, replace = TRUE))
par(mfrow=c(1,2))
  plot3D::scatter3D(xyz_centers$x, #plot centers (blue)
                xyz_centers$y,cex=1,
                xyz_centers$z, bty = "g", pch = 1, theta = 30,
                phi = 30, col = "blue")
  plot3D::scatter3D(xyz_traps$x, # add traps (red)
                xyz_traps$y, cex=1,
                xyz_traps$z, bty = "g", pch = 17,theta = 30,
                 phi = 30, col = "red",add = TRUE)
legend("topright",bty="n","(a)")
    ## find nearest neighbour
    dist <- FNN::get.knnx(data = as.matrix(xyz_centers),
                          query = as.matrix(xyz_traps),k = 1)
   ## plot histogram
    distance_nm<-as.vector(dist$nn.dist)
    hist(distance_nm,xlim=c(0,22),main=" ")
    ## calc analytical solution
    r <- seq(0, 20, 0.1)
    distr_ana <- 3 * 8 * r_prime(r)^2 * exp(-(r_prime(r)^3))
    ## plot analytical solution
    lines(x = r,  y = distr_ana)
legend("topright",bty="n","(b)")
```

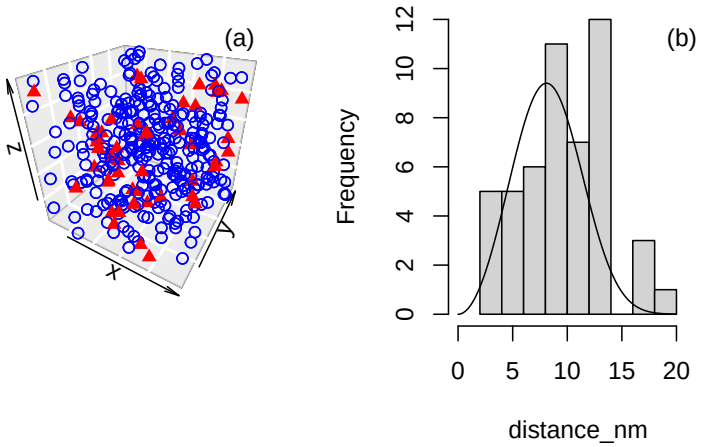


Fig. 6.1: (a) A cube with side $d = 100$ nm contains 50 electrons (triangles) and 300 recombination centers (circles). (b) Histogram of the nearest neighbor distances of electron-acceptor pairs from the cube in (a). The solid line in (b) represents the analytical equation for the distribution of nearest neighbors Eq.(**??**). For more information see Pagonis and Kulp [42].

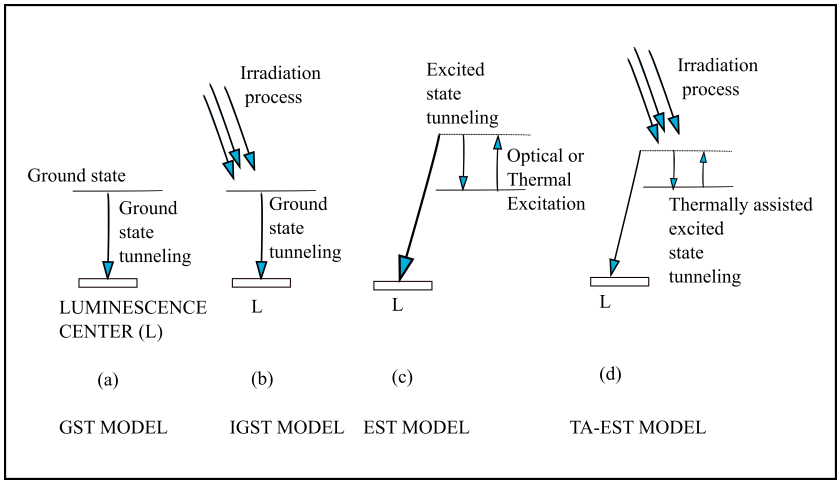


Fig. 6.2: Schematic depiction of several TLT models: (a) The ground state tunneling (GST) model (Tachiya and Mozumder [50], Huntley [11]). (b) The more general irradiation and ground state tunneling (IGST) model studied by Li and Li [23], in which anomalous fading and natural irradiation are taking place simultaneously. (c) The excited state tunneling (EST) model (Jain et al. [12]). (d) Simultaneous irradiation and thermally assisted excited state tunneling (TA-EST) model by Brown et al. [5].

Code 6.2: Time evolution of the nearest neighbors distribution

```
rm(list=ls())
s<-3e15                    # frequency factor
rho<-1e-6                  # rho-prime values 0.005-0.02
rc<-0.0                    # for freshly irradiated samples, rc=0
times<-3.154e7*c(0,1e2,1e4,1e6)          # times in seconds
rprimes<-seq(from=rc,to=2.2,by=0.002)    # rprime=0-2.2

##### function to find distribution of distances ###
fingDistr<-function(tim){3*(rprimes**2.0)*exp(-(rprimes**3.0))*
    exp(-exp(-(rho**(-1/3))*rprimes)*s*tim)}
######

distribs<-sapply(times,fingDistr)
# Plots
cols=c(NA,NA,NA,1:4)
matplot(rprimes,distribs,xlab="Dimensionless distance r'",
ylab="Nearest neighbor Distribution g(r')",type="l",lwd=4)
```

```
legend("topright",bty="n", lty=c(NA,NA,NA,1:4), lwd=4,
col=cols,legend = c("Elapsed", "time"  ," ","t=0 s",
 expression("10"^"2"*" years"),
expression("10"^"4"*" years"),expression("10"^"6"*" years")))
```

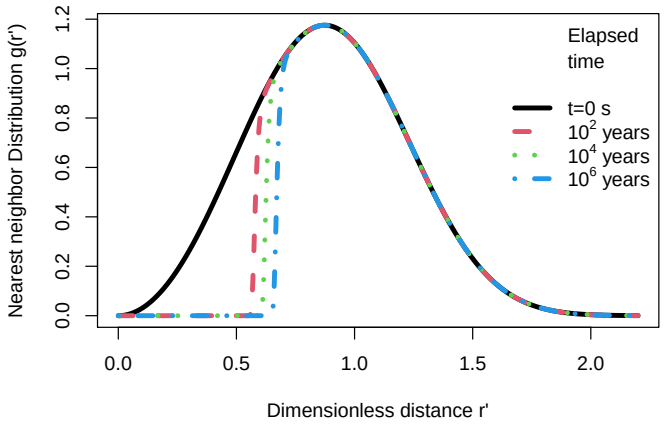


Fig. 6.3: Examples of the nearest neighbor distribution at different times $t = 0, 10^2, 10^4, 10^6$ years, based on Eq.(**??**). The solid black line represents the nearly symmetric distribution $g(r')$ at time $t = 0$. As time increases, the "tunneling front" is the almost vertical line which moves to the right, as more and more electrons are recombining at larger distances r'.

Code 6.3: Ground state tunneling: Remaining electrons n(t)

```
# Simulate Loss of charge due to ground state tunneling (GST)
rm(list=ls())
z<-1.8
s<-3e15                    # frequency factor
rho1<-1e-6                 # rho-prime values
rho2<-5e-6
rho3<-1e-5
years<-1000
elapsedt<-3.154e7*years
```

```
t<-seq(from=1,to=elapsedt,by=1e6)
gr1<-100*exp(-rho1*(log(z*s*t)**3.0))
gr2<-100*exp(-rho2*(log(z*s*t)**3.0))
gr3<-100*exp(-rho3*(log(z*s*t)**3.0))
plot(unlist(t)/(3.154e7),unlist(gr1),type="l",lty=1,lwd=2,
ylim=c(0,130),pch=1,ylab="Remaining electrons [%]",
xlab="Elapsed time t [years]")
lines(unlist(t)/(3.154e7),unlist(gr2),lwd=2,col="red",lty=2)
lines(unlist(t)/(3.154e7),unlist(gr3),lwd=3,col="green",lty=2)
legend("topright",bty="n",lwd=2, lty=c(NA,1,2,3),
legend = c("Acceptor Density"  ,expression("10"^"-6"*""),
expression("5x10"^"-6"*""),expression("10"^"-5"*"")),
 col=c(NA,"black","red","green"))
```

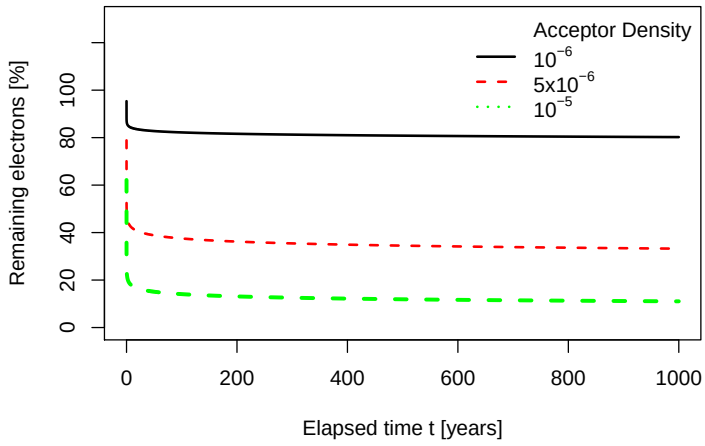


Fig. 6.4: Examples of the loss of charge due to ground state tunneling, for different dimensionless acceptor densities $\rho' = 10^{-6}, 5\times10^{-6}, 10^{-5}$, based on Eq.(**??**). As ρ' increases, the rate of charge loss increases. As the elapsed time t increases, the initial fast loss of charge is followed by a "long tailed" decay, characteristic of quantum tunneling phenomena.

Code 6.4: Anomalous fading (AF) and the g-factor

```
rm(list=ls())
# Load the data of Anomalous fading (AF) and calculate the g-factor
par(mfrow=c(1,2))
mydata <- read.table("durnago0sok.txt")
T<-mydata[,1]
TL<-mydata[,2]/1e5
plot(T,TL,type="o",pch=1,col="red",xlim=c(50,450),ylim=c(0,1.6),
  xlab=expression("Temperature ["^"o"*"C]"),ylab ="TL (a.u.)")
mydata2 <- read.table("durango10daysok.txt")
T2<-mydata2[,1]
TL2<-mydata2[,2]/1e5
lines(T2,TL2,type="o",pch=2,col="blue")
legend("left",bty="n","(a)")
legend("topleft",bty="n",lwd=2, lty=c(NA,NA,1,2,3),
pch=c(NA,NA,1,2),
legend=c(expression('Anomalous', 'fading ', 't=0 s',
't=10 days')),col=c(NA,NA,"red","blue"))
mydata3 <- read.table("DurangoAFdataok.txt")
t<-mydata3[,1]
RTL<-mydata3[,2]
#plot(t,RTL,xlab="ln(t/to)",ylab="Remnat TL [%]")
y<-RTL
x<-t
bestfit<-lm(y~x)
coefficients(bestfit)
plot(x, y, xlab=expression("ln(t/to)"),ylab = "RTL [%]")
abline(lm(y~x))
slope<-abs(coefficients(bestfit)[[2]])
g<-230.2*slope
paste0("g-factor=",round(g,digits=2)," % per decade")
legend("topright",bty="n",
legend=c(expression('Durango apatite',
'g-factor=20.5% ',)))
legend("left",bty="n","(b)")
```

```
  ## (Intercept)           x
  ##  0.99615757 -0.08918921
  ## [1] "g-factor=20.53 % per decade"
```

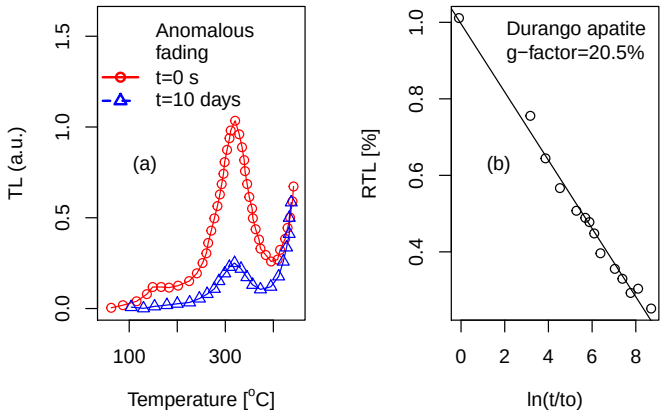


Fig. 6.5: Anomalous fading effect in Durango apatite. (a) The TL signal is measured immediately after irradiation, and after 10 days have elapsed at room temperatures. (b) Analysis of the remnant TL signal from (a), to obtain the g-factor for this materials. For more details, see Polymeris et al. [49].

Code 6.5: Simultaneous irradiation and anomalous fading in nature

```
# Simultaneous irradiation and anomalous fading in nature
rm(list=ls())
s<-3e15                    # frequency factor
rho1<-1e-6                 # rho-prime values 0.005-0.02
rho2<-2e-6
rho3<-3e-6
Do<-538
X<-3/(1000*365*3600*24)       #natural dose rate X=3 Gy/Ka
curve((1-exp(-x/Do))*exp(-rho1*(log(Do*s/X)**3.0)),1,3500, 200,
lty=1,lwd=3,
ylab=expression('L'[FADED]*'  [a.u.]'),xlab="Natural Dose [Gy]",
col=1,ylim=c(0,1.2))
curve((1-exp(-x/Do))*exp(-rho2*(log(Do*s/X)**3.0)),1,3500,200,
col=2,lty=2, lwd=3,   add=TRUE)
curve((1-exp(-x/Do))*exp(-rho3*(log(Do*s/X)**3.0)),1,3500,200,
col=3,   lwd=3,   lty=3,add=TRUE)
```

```
legend("topright",bty="n",lwd=3, lty=c(NA,1,2,3),legend =
c("Acceptor Density"  ,expression("10"^"-6"*""),
expression("2x10"^"-6"*""),expression("3x10"^"-6"*"")),
col=c(NA,1:3))
legend('topleft',bty="n",c("Simultaneous",
"Irradiation and fading") )
```

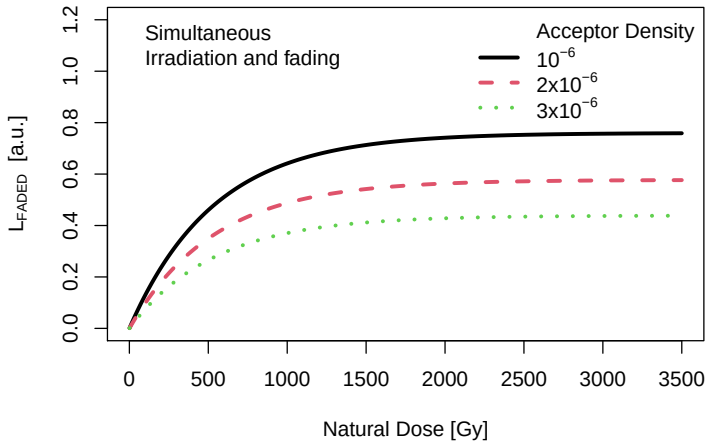


Fig. 6.6: Examples of the accumulated charge $n(t)$ during simultaneous irradiation and fading in nature, due to ground state tunneling, for three different dimensionless acceptor densities $\rho' = 10^{-6}, 2\times10^{-6}, 3\times10^{-6}$. As ρ' increases, the rate of charge accumulation and the corresponding saturation signal *decrease*. For more details, see Pagonis and Kitis [37].

Code 6.6: CW-IRSL data fitted with KP-CW equation

```
#Fit CW-IRSL data with KP-CW equation
rm(list = ls(all=T))
options(warn=-1)
library("minpack.lm")
## fit to analytical KP-CW equation for CW-IRSL (TLT model)-
mydata <- read.table("ph300s0IR.asc")
```

```
t<-as.numeric(gsub(",", ".", gsub("\\.", "", mydata[,1])))
y<-as.numeric(gsub(",", ".", gsub("\\.", "", mydata[,3])))
mydata<-data.frame(t,y)
plot(t,y,log="xy",
xlab="Time [s]",ylab="CW-IRSL [counts/s]",col="black",pch=1)
fit_data <-mydata
fit <- minpack.lm::nlsLM(
 formula=y~ imax*exp (-rho*(log(1 + A*t)) ** 3.0)*
    (log(1 + A*t) ** 2.0)/(1 +  t*A)+bgd,
  data = fit_data,
  start = list(imax=3,A=1.1,rho=0.03,bgd=min(y)))
imax_fit <- coef(fit)[1]
A_fit <- coef(fit)[2]
rho_fit <- coef(fit)[3]
bgd_fit <- coef(fit)[4]
## plot analytical solution
lines(
  x = t,
  y =imax_fit*exp (-rho_fit*(log(1 + A_fit*t)) ** 3.0)*
    (log(1 + A_fit*t) ** 2.0)/(1 +  t*A_fit)+bgd_fit,
  col = "red",lwd=2,log="xy")
legend("topright",bty="n", pch=c(NA,NA,1,NA),lwd=2,
lty=c(NA,NA,NA,"solid"),
c(expression('KST4 feldspar',' ',
'Experiment','KP-CW equation')),col=c(NA,NA,"black","red"))
## print results
cat("Parameters from Least squares fit"," ")
cat("\nImax=",formatC(imax_fit,format="e",digits=2)," cts/s",
sep="    ","A=",round(A_fit,digits=2)," (s^-1)")
cat("\nrho=",round(rho_fit,digits=4),sep=" ",
"  bgd=",round(bgd_fit,digits=2)," cts/s")
```

```
  ## Parameters from Least squares fit
  ## Imax=    2.72e+04     cts/s    A=    7.07     (s^-1)
  ## rho= 0.0073   bgd= 47.05  cts/s
```

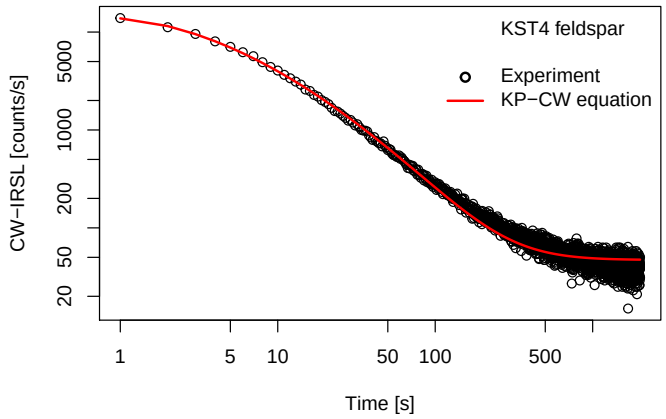


Fig. 6.7: Experimental CW-IRSL glow curves from freshly irradiated KST4 feldspar sample, fitted using the KP-CW analytical Eq.(**??**). For more details and examples, see Pagonis et al. [36].

Code 6.7: Kitis-Pagonis analytical equation for TL (KP-TL)

```
rm(list = ls(all=T))
options(warn=-1)
library("minpack.lm")
library(expint)
## Least squares fit to TL using the KP-TL eqt (TLT model)-
mydata <- read.table("ph300s0.asc")
t<-as.numeric(gsub(",", ".", gsub("\\.", "", mydata[,1])))
y<-as.numeric(gsub(",", ".", gsub("\\.", "", mydata[,3])))
y<-y/max(y)
mydata<-data.frame(t,y)
plot(t,y,xlab="Temperature [\u00B0C]",ylab="Normalized TL",
col="black",pch=1,xlim=c(200,450))
kb<-8.617e-5
z<-1.8
fit_data <-mydata
kB<-8.617E-5
En<-1.45
T<-t+273
```

```
fit <- minpack.lm::nlsLM(
 formula=y~imax* exp(-rho*( (log(1+z*s*kB*((T**2.0)/
abs(En))*exp(-En/(kB*T))*(1-2*kB*T/En)))**3.0))*
(En**2.0-6*(kB**2.0)*(T**2.0))*( (log(1+z*s*kB*((T**2.0)/
abs(En))*exp(-En/(kB*T))*(1-2*kB*T/En)))**2.0)/
(En*kB*s*(T**2)*z-2*(kB**2.0)*s*z*(T**3.0)+exp(En/(kB*T))*En),
  data = fit_data,
start = list(imax=1e12,s=1e11,rho=.009),upper=c(1e20,1e13,.02),
 lower=c(1e11,1e11,.008))
# Obtain parameters from best fit
imax_fit <- coef(fit)[1]
s_fit <- coef(fit)[2]
rho_fit <- coef(fit)[3]
En_fit <- En
## plot analytical solution
lines(
x = t,imax_fit* exp(-rho_fit*( (log(1+z*s_fit*kB*((T**2.0)/
abs(En_fit))*exp(-En_fit/(kB*T))*(1-2*kB*T/En_fit)))**3.0))*
(En_fit**2.0-6*(kB**2.0)*(T**2.0))*((log(1+z*s_fit*kB*((T**2.0)/
abs(En_fit))*exp(-En_fit/(kB*T))*(1-2*kB*T/En_fit)))**2.0)/
(En_fit*kB*s_fit*(T**2)*z-2*(kB**2.0)*s_fit*z*(T**3.0)+
exp(En_fit/(kB*T))*En_fit),col="red",lwd=2)
legend("topleft",bty="n", pch=c(NA,NA,1,NA),lwd=2,
lty=c(NA,NA,NA,"solid"),
c(expression('KST4 feldspar',' ',
'Experiment','KP-TL Eq.')),col=c(NA,NA,"black","red"))
## print results
cat("\nBest fit parameters"," ")
cat("\nImax=", formatC(imax_fit, format = "e", digits = 2),
" s=",formatC(s_fit, format = "e", digits = 2)," (s^-1) ")
cat("\nrho=",round(rho_fit,digits=4),sep=" ","E=",En_fit," eV")
```

```
  ##
  ## Best fit parameters
  ## Imax= 1.40e+13  s= 3.50e+12  (s^-1)
  ## rho= 0.0096 E= 1.45  eV
```

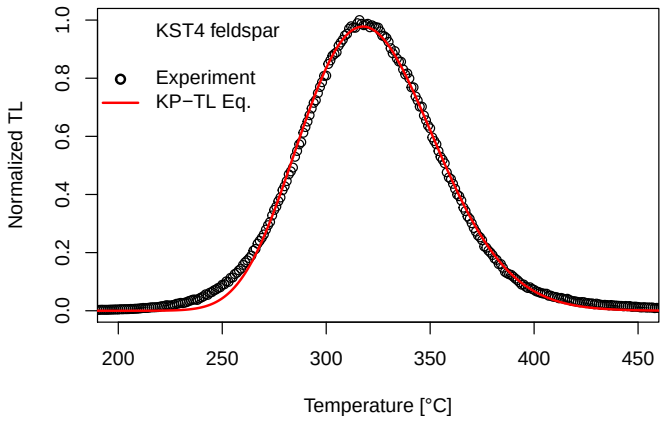


Fig. 6.8: Experimental TL glow curves from freshly irradiated KST4 feldspar sample, fitted using the KP-TL analytical Eq.(**??**). Note that the solid line does not describe the experimental data very accurately at low temperatures, due to the approximations involved in the KP equations. For more details and MC examples, see Pagonis et al. [36].

Function Name	**Description**
plot_RLumCarlo	Plots 'RLumCarlo' modeling results (the averaged signal or the number of remaining electrons), with modeling uncertainties.
run_MC_CW_IRSL_TUN	Simulation of CW-IRSL signals due to tunneling transitions from the excited state of the trap, into a recombination center (RC) for the EST model.
run_MC_ISO_TUN	Simulation of ITL signals due to tunneling from the excited state of the trapped charge, into the RC (EST).
run_MC_LM_OSL_TUN	Simulation of LM-IRSL signals due to tunneling from the excited state of the trapped charge, into the RC (EST).
run_MC_TL_TUN	Simulation of TL signals due to tunneling from the excited state of the trapped charge, into the RC (EST).

Table 6.1: Table of TUNneling functions available in the package *RLumCarlo*.

Process	Parameter	Description	Units	Typical values
TL with tunneling	E	Thermal activation energy	eV	0.5-3
	s	Effective frequency factor	1/s	1E8-1E16
	rho	Dimensionless density of recombination centers ρ'	1	1E-6-1E-2
	r_c	Critical distance (>0) for thermally/optically pretreated samples	1	0.1-0.8
	times	Sequence of time steps (heating rate 1 K/s)	s	0-700
	clusters	Number of MC runs	1	1E1-1E4
	N_e	Total number of electron traps available	1	2-1E5
	delta.r	Increments of the distance r'	1	0.01-0.1
CW-IRSL with tunneling	A	Effective optical excitation rate	1/s	1E-3-1
	rho	Dimensionless density of recombination centers ρ'	1	1E-6-1E-2
	times	Sequence of time steps	s	0-500
	clusters	Number of MC runs	1	1E1-1E4
	N_e	Total number of electron traps available	1	2-1E5
	r_c	Critical distance (>0) for thermally/optically pretreated sample	1	0.1-0.8
	delta.r	Increments of the distance r'	1	0.01-0.1
ISO with tunneling	E	Thermal activation energy	eV	0.5-3
	s	Effective frequency factor	1/s	1E8-1E16
	T	Temperature of the isothermal process	°C	20-300
	rho	Dimensionless density of recombination centers ρ'	1	1E-6-1E-2
	times	Sequence of time steps for simulation	s	0-1000
	clusters	Number of MC runs	1	1E1-1E4
	N_e	Total number of electron traps available	1	2-1E5
	r_c	Critical distance (>0) for thermally/optically pretreated sample	1	0.1-0.8
	delta.r	increments of the distance r'	1	0.01-0.1
LM-OSL with tunneling	A	Effective optical excitation rate	1/s	1E-3-1
	rho	Dimensionless density of recombination centers ρ'	1	1E-6-1E-2
	times	Sequence of time steps	s	0-3000
	clusters	Number of MC runs	1	1E1-1E4
	N_e	Total number of electron traps available	1	2-1E5
	r_c	Critical distance (>0) for thermally/optically pretreated sample	1	0.1-0.8
	delta.r	Increments of the distance r'	1	0.01-0.1

Table 6.2: Table of input parameters for TUN functions in *RLumCarlo*.

Code 6.8: Single plot MC simulations for tunneling CW-IRSL

```
##================================================##
## Example:Single Plot for  tunneling CW-IRSL
##================================================##
rm(list = ls(all=T))
suppressMessages(library("RLumCarlo"))
run_MC_CW_IRSL_TUN(
A = 5,
rho = 5e-3,
times = 0:500,
r_c = 0.5,
delta.r = 1e-2,
method = "par",
output = "signal"
) %>%
#Plot results of the MC simulation
plot_RLumCarlo(norm = F, legend = F)
legend("top",bty="n",legend=c("CW-IRSL","TUN function"))
```

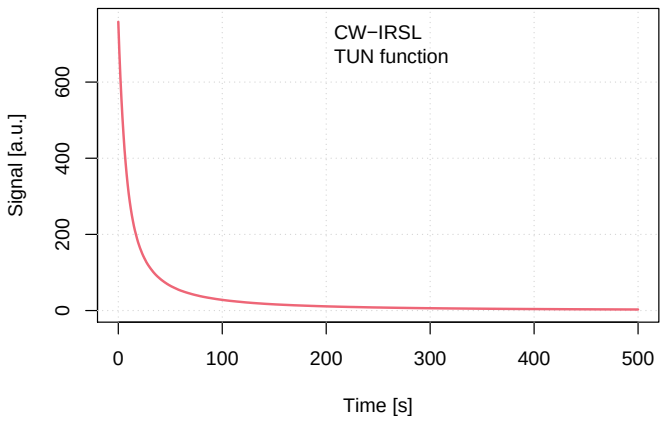


Fig. 6.9: Simulation of CW-IRSL signal using the function *run_MC_CW_IRSL_TUN* in the package *RLumCarlo*.

Code 6.9: Combining two plots in CW-IRSL experiment

```
rm(list = ls(all=T))
library(RLumCarlo)
times <- seq(0, 200)
run_MC_CW_IRSL_TUN(A = 3, rho = 0.003, times = times) %>%
  plot_RLumCarlo(norm = TRUE, lty=1,legend = TRUE)
run_MC_CW_IRSL_TUN(A = 6, rho = 0.003, times = times) %>%
  plot_RLumCarlo(norm = TRUE, lty=2,col="blue",add = TRUE)
legend("top",bty="n",legend=c(expression("CW-IRSL","TUN function",
 "A=3.0 s"^-1*" ","A=6.0 s"^-1*" ")),lty=c(NA,NA,1:2))
```

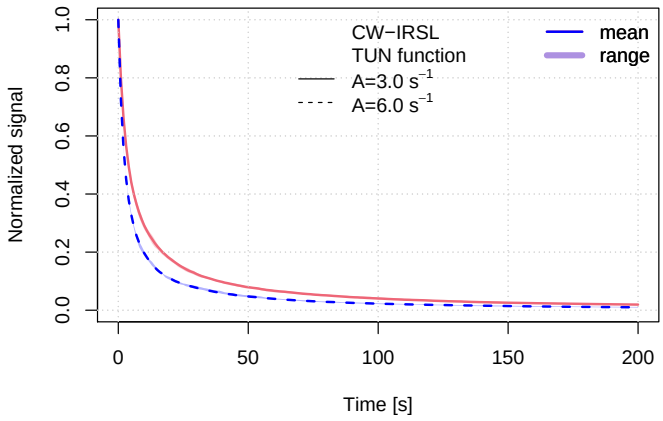


Fig. 6.10: Simulation of two CW-IRSL signals using the function *run_MC_CW_IRSL_TUN* in the package *RLumCarlo*, for two different excitation rates A.

Code 6.10: MC for tunneling ITL: remaining electrons

```
# MC Model for remaining charges and ITL signal
rm(list = ls(all=T))
library(RLumCarlo)
par(mfrow=c(1,2))
results <- run_MC_ISO_TUN(
  E = 1.0,
  s = 1e12,
  T = 250,
  rho = 0.01,
  clusters=100,
  times = seq(0, 200),
  output = "remaining_e"
) %T>%
  plot_RLumCarlo(
    legend = FALSE,
    ylab = "Remaining electrons"  )
legend("topright",bty="n",legend=c("(a)"," ","n(t)",
"TUN function"))

results <- run_MC_ISO_TUN(
  E = 1.2,
  s = 1e12,
  T = 250,
  rho = 0.01,
  times = seq(0, 200)
) %T>%
  plot_RLumCarlo(norm = FALSE, legend = FALSE)
legend("topright",bty="n",legend=c("(b)"," ","ITL signal"))
```

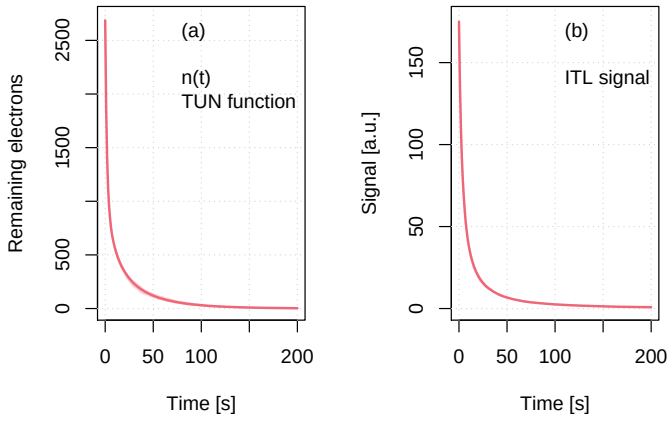


Fig. 6.11: Simulation of (a) remaining electrons $n(t)$ during an ITL experiment, and (b) of the resulting ITL signal, using the function *run_MC_ISO_TUN* in the package *RLumCarlo*.

Code 6.11: Single plot MC simulations for tunneling LM-OSL

```
##=================================================##
## Example 1: MC simulations of tunneling LM_IRSL
##=================================================##
rm(list = ls(all=T))
library(RLumCarlo)
run_MC_LM_OSL_TUN(
A =3.0,
rho = 1e-2,
times = 0:200,
clusters = 100,
N_e = 20,
r_c = 0.001,
delta.r = 1e-1,
method = "par",
output = "signal"
) %>%
```

```
# Plot results of the MC simulation
plot_RLumCarlo(norm = F,legend=F)
legend("topright",bty="n",legend=c("LM-IRSL signal",
"TUN function",""))
```

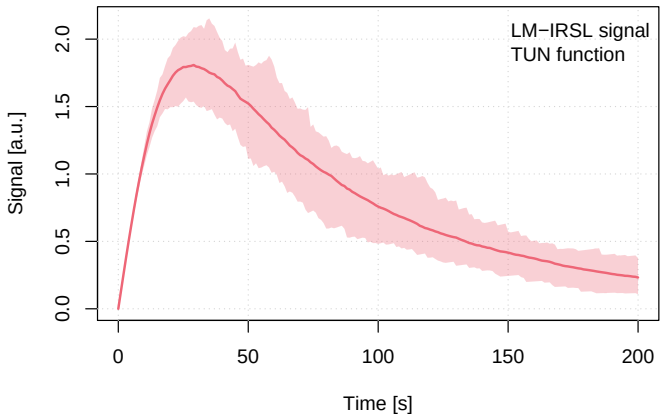


Fig. 6.12: Simulation of LM-IRSL signal using the function *run_MC_LM_OSL_TUN* in the package *RLumCarlo*.

Code 6.12: Simulation of TL from pretreated sample

```
##============================================
## Simulate TL from pretreated sample
##============================================
rm(list = ls(all=T))
library(RLumCarlo)
s <- 3.5e12
rho <- 0.015
E <- 1.45
r_c <- c(0,0.8,1.0)
times <- seq(100, 450) # time = temperature
results <- lapply(r_c, function(x) {
  run_MC_TL_TUN(
```

```
    s = s,
    E = E,
    rho = rho,
    r_c = x,
    times = times
  )})%>%
plot_RLumCarlo(norm = FALSE, legend = TRUE)
legend("topleft",bty="n",legend=c(expression('Critical Radius',
' ','r'[c]*'=0, 0.8, 1.0')) )
```

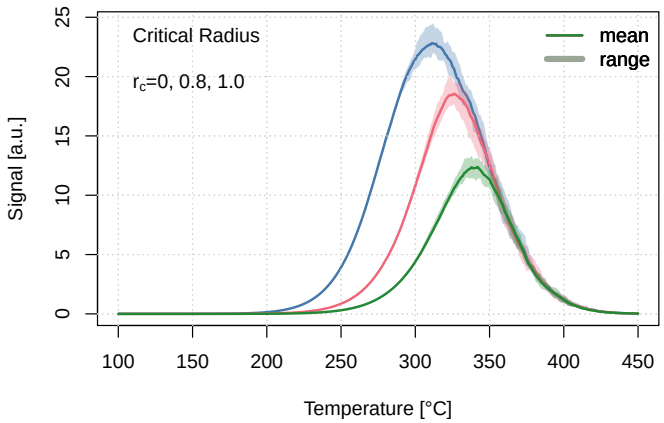


Fig. 6.13: Simulation of TL signal using the function *run_MC_TL_TUN* in the package *RLumCarlo*. The sample has been thermally treated after irradiation, and the effect of the thermal treatment is described by the variable critical radius parameter $r_c' = 0,\ 0.8,\ 1.0$.

Chapter 7
LOCALIZED TRANSITIONS: THE LT AND SLT MODEL

Abstract In this chapter we study two localized models found in the luminescence literature, the localized transition model (LT), and the semilocalized transitions model (SLT). We provide R codes for numerically solving the differential equations for the LT model, and compare the solution with the analytical equations involving the Lambert W function. Several examples are presented using the R package *RLumCarlo* to simulate these types of localized transition phenomena. We present R codes for the important hybrid SLT model developed by Mandowski, which can be used to explain the anomalous heating rate effect observed in several dosimetric materials. Finally, the chapter concludes with the presentation of a simplified form of the SLT model, developed by Pagonis et al.

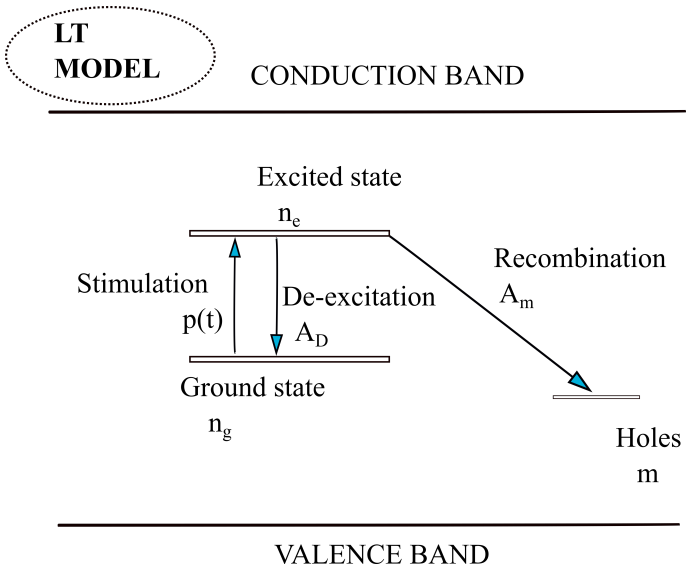


Fig. 7.1: Energy band diagram for the localized transition LT model.

Code 7.1: Plot W(x) solution of LT model

```
rm(list=ls())
library(lamW)
## Plot the analytical solution of LT, using Lambert W-function
x1<-300:450
kB<-8.617E-5
no<-1E8
r<-1e4
En<-1
s<-1E13
beta<-1
k<-function(u) {integrate(function(p){exp(-En/(kB*p))},
300,u)[[1]]}
y1<-lapply(x1,k)
x<-unlist(x1)
y<-unlist(y1)
zTL<-(r/no)-log(no/r)+(s*y)
plot(x,r*s*exp(-En/(kB*x))/(lambertW0(exp(zTL))
      +lambertW0(exp(zTL))^2),type="l",col="red",
```

```
lwd=3,lty=1,xlab="Temperature, K",ylab="TL in LT model [a.u.]")
zTL<-(r/no)-log(no/r)+(s*y)
r<-1e8
zTL<-(r/no)-log(no/r)+(s*y)
lines(x,r*s*exp(-En/(kB*x))/(lambertW0(exp(zTL))
      +lambertW0(exp(zTL))^2),col="green",
      lwd=3,  lty=2,   xlab="Temperature, K",ylab="TL")
r<-1e9
zTL<-(r/no)-log(no/r)+(s*y)
lines(x,r*s*exp(-En/(kB*x))/(lambertW0(exp(zTL))
      +lambertW0(exp(zTL))^2),col="blue",
      lwd=3,  lty=3,    xlab="Temperature, K",ylab="TL")
legend("topright",bty="n",lwd=2, lty=c(NA,NA,1,2,3),
legend = c("Retrapping ratio r", expression("(cm"^"-3"*")"),
expression("10"^"4"*""),expression("10"^"8"*""),
expression("10"^"9"*"")),
       col=c(NA,NA,"red","green","blue"))
```

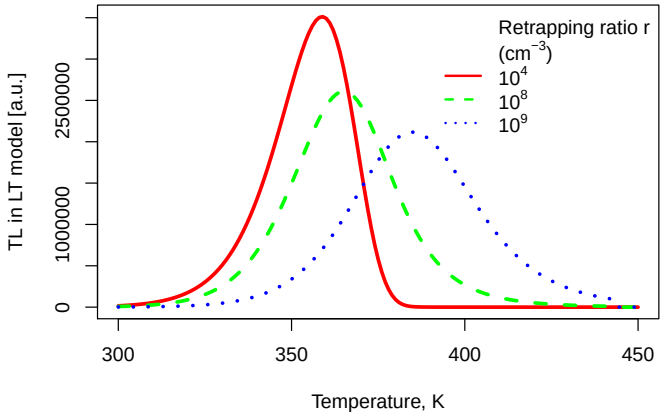


Fig. 7.2: Plots of the analytical solution of the LT model, using the Lambert W function, for three values of the retrapping ratio $r = 10^4$, 10^8, 10^9 cm^{-3} and $n_0 = 10^8$ cm^{-3}. As r increases, the shape of the TL glow curve changes from an asymmetric peak for first order kinetics, to a symmetric shape for second order kinetics. For details, see Kitis and Pagonis [14].

Process	Symbol	Parameter in RLumCarlo function	Unit	Typical values
Localized TL	E	Thermal activation energy of the trap	eV	0.5-3
	s	Frequency factor of the trap	1/s	1E8-1E16
	times	Sequence of time steps for simulation (heating rate 1 K/s)	s	0-700
	clusters	Number of MC runs	1	1E1-1E4
	n_filled	Number of filled electron traps at the beginning of the simulation	1	1-1E5
	r	Localized retrapping ratio	1	0-1E5
Localized CW-IRSL	A	Optical excitation rate from ground state of the trap to the excited state	1/s	1E-3-1
	times	Sequence of time steps for simulation	s	0-500
	clusters	Number of MC runs	1	1E1-1E4
	n_filled	Number of filled electron traps at the beginning of the simulation	1	1-1E5
	r	Localized retrapping ratio	1	0-1E5
Localized ISO	E	Thermal activation energy of the trap	eV	0.5-3
	s	Frequency factor of the trap	1/s	1E8-1E16
	T	Temperature of the isothermal process	°C	20-300
	times	Sequence of time steps for simulation	s	0-1000
	clusters	Number of MC runs	1	1E1-1E4
	n_filled	Number of filled electron traps at the beginning of the simulation	1	1-1E5
	r	Localized retrapping ratio	1	0-1E5
Localized LM-OSL	A	Optical excitation rate from ground state of the trap to the excited state	1/s	1E-3-1
	times	Sequence of time steps for simulation	s	0-3000
	clusters	Number of MC runs	1	1E1-1E4
	n_filled	Number of filled electron traps at the beginning of the simulation	1	1-1E5
	r	Localized retrapping ratio	1	0-1E5

Table 7.1: Table of input parameters for LOC functions in *RLumCarlo*.

Function Name	Description
plot_RLumCarlo	Plots 'RLumCarlo' modeling results (the averaged signal or the number of remaining electrons), with modeling uncertainties.
run_MC_CW_IRSL_LOC	Simulation of CW-IRSL signals in the LT model, due to tunneling transitions from the excited state of the trap, into a recombination center (RC).
run_MC_ISO_LOC	Simulation of ITL signals in the LT model, due to tunneling from the excited state into the RC.
run_MC_LM_OSL_LOC	Simulation of LM-IRSL signals in the LT model, due to tunneling from the excited state into the RC.
run_MC_TL_LOC	Simulation of TL signals in the LT model, due to tunneling from the excited state into the RC.

Table 7.2: Table of LOCalized transition functions available in the package *RLumCarlo*.

Code 7.2: Single plot MC simulations for localized CW-IRSL (LT model)

```
##============================================================##
## Example: MC simulations for localized CW_IRSL (LT model)
##============================================================##
rm(list = ls(all=T))
library(RLumCarlo)
run_MC_CW_IRSL_LOC(
A = 0.12,
times = 0:100,
clusters = 10,
n_filled = 100,
r = 1e-7,
method = "seq",
output = "signal"
) %>%
#Plot results of the MC simulation
plot_RLumCarlo(legend = T)
 legend("top",bty="n",c("CW-IRSL signal","LOC function",
"LT model"))
```

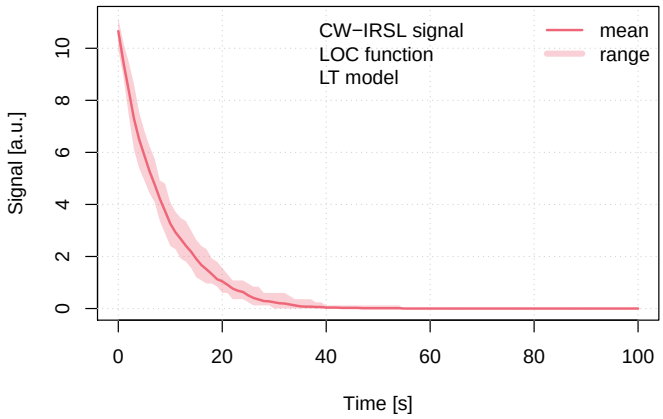


Fig. 7.3: Simulation of CW-IRSL signal in the LT model, using the function *run_MC_LM_IRSL_LOC* in the package *RLumCarlo*.

Code 7.3: Single plot MC simulations for localized LM-OSL

```
##============================================================##
##============================================================##
## MC simulations for localized LM-OSL
##============================================================##
rm(list = ls(all=T))
library(RLumCarlo)
run_MC_LM_OSL_LOC(
A = 0.1,
times = 0:250,
clusters = 100,
n_filled = 50,
r = 1e-7,
method = "seq",
output = "signal"
) %>%
#Plot results of the MC simulation
plot_RLumCarlo(legend = T)
legend("top",bty="n",c("LM-IRSL signal","LOC function",
"LT model"))
```

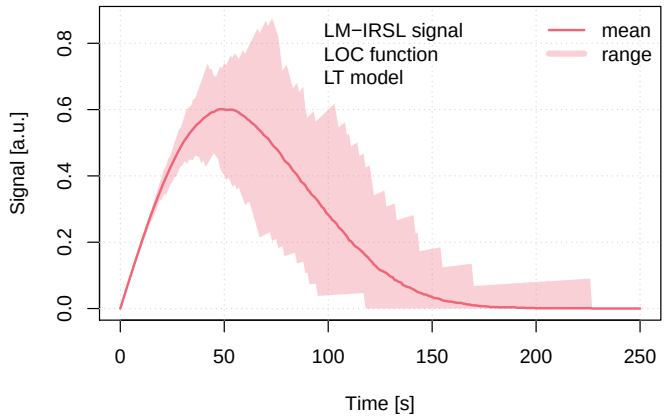


Fig. 7.4: Simulation of LM-IRSL signal in the LT model, using the function *run_MC_LM_IRSL_LOC* in the package *RLumCarlo*.

Code 7.4: Localized TL with variable retrapping ratio r

```
##=====================================================##
## Localized TL with variable retrapping ratio r (LT model)
##=====================================================##
rm(list = ls(all=T))
library(RLumCarlo)
f<-function(rvar,vars,addTF){
  run_MC_TL_LOC(
    s = 1e12,
    E = 1,
    times = 0:300,
    r = rvar
  ) %>%
    #Plot results of the MC simulation
    plot_RLumCarlo(legend = F,plot_uncertainty=NULL,type="o",
  pch=vars,col=vars,add=addTF, xlim=c(50,275))}
f(1,1,FALSE)
f(1e3,2,TRUE)
f(1e4,3,TRUE)
```

```
legend("topright",bty="n",pch=c(NA,NA,NA,1:3),
col=c(NA,NA,NA,1:3),legend = c("Retrapping"  ,"Ratio r ",
 " ","1",  expression("10"^"3"*""),expression("10"^"4"*"")) )
```

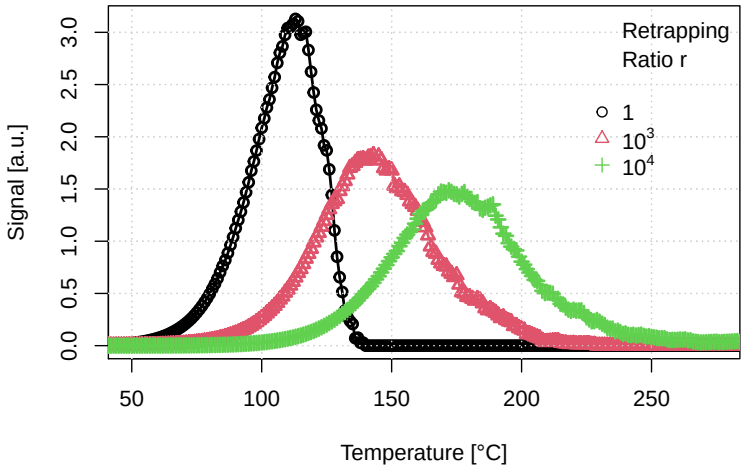


Fig. 7.5: Simulation of TL signal in the LT model, by using the function *run_MC_TL_LOC* in the package *RLumCarlo,* for three different values of the retrapping ratio r.

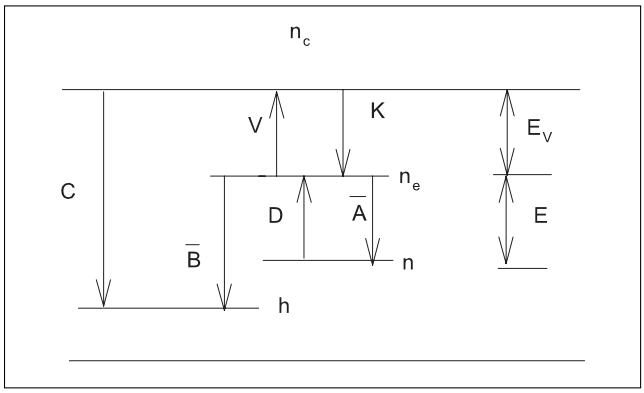


Fig. 7.6: The SLT model proposed by Mandowski [24].

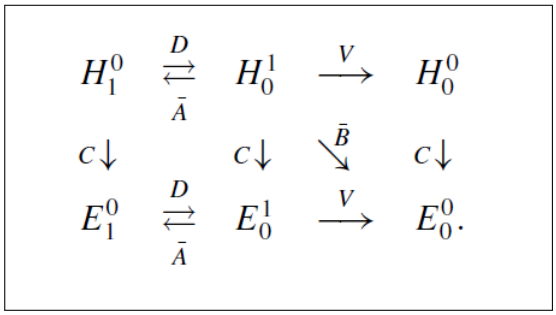


Fig. 7.7: Schematic representation of the possible transitions in the SLT model by Mandowski [24].

Code 7.5: Mandowski SLT model: simulation of TL experiment

```
# Mandowski SLT model: simulation of TL experiment
rm(list=ls())
library("deSolve")
TLMandowski <- function(t, x, parms) {
  with(as.list(c(parms, x)), {
 dH01<- -(s*exp(-E/(kb*(273+hr*t)))+C*(H00-E01-E10))*H01+A*H10
 dH10 <- s*exp(-E/(kb*(273+hr*t)))*H01-(A+B+sv*exp(-Ev/(kb*
    (273+hr*t)))+C*(H00-E01-E10))*H10
 dH00<-sv*exp(-Ev/(kb*(273+hr*t)))*H10-C*(H00-E01-E10)*H00
 dE01<-C*(H00-E01-E10)*H01-s*exp(-E/(kb*(273+hr*t)))*E01+A*E10
 dE10<-C*(H00-E01-E10)*H10+s*exp(-E/(kb*(273+hr*t)))*E01-
      (A+sv*exp(-Ev/(kb*(273+hr*t))))*E10
 dE00<-B*H10+C*(H00-E01-E10)*H00+sv*exp(-Ev/(kb*(273+hr*t)))*E10
     res <- c(dH01,dH10,dH00,dE01,dE10,dE00)
    list(res)  })}
enVar<-function(en){
parms  <- c(E =en, s=1e10, sv=1e10, kb=8.617*10^-5,
hr=1, C=C,B=B,Ev=0.7,A=2*10^4)
y <- xstart <- c(H01 = 10^17,   H10=0,H00=0,E10=0,E01=0,E00=0)
out <-  ode(xstart, times, TLMandowski, parms, method = "lsoda",
 atol = 1,rtol=1) }
C<-1e-10
B<-1e4
temps<-times  <- seq(0, 360,by=.5)
```

```
Lc<-Lb<-matrix(NA,nrow=length(times),ncol=4)
areaLb<-areaLc<-vector(length=4)
  en<-.95
  a<-enVar(en)
  Lc<-(C*(a[,"H00"]-a[,"E01"]-a[,"E10"])*
          (a[,"H10"]+a[,"H01"]+a[,"H00"]))
  Lb<-B*a[,"H10"]
  areaLc<-sum(Lc)*0.5
  areaLb<-sum(Lb)*0.5
par(mfrow=c(2,3))
xlabs=expression("Temperature ["^"o"*"C]")
plot(temps,Lb,typ="l",xlim=c(100,270),ylim=c(0,4e15),col="blue",
pch=1,xlab=expression("Temperature ["^"o"*"C]"),lwd=5,
ylab="LB [a.u.])")
legend("topleft",bty="n",col="red",
legend=c("(a)","Mandowski","SLT model","LB TL peak"))
plot(temps,Lc,typ="o",xlim=c(100,270),ylim=c(0,.7e14),col="red",
xlab=xlabs,ylab="LC [a.u.])")
legend("topleft",bty="n",col="red",
legend=c("(b)"," ","LC TL peak"))
plot(temps,a[,"H00"],typ="o",xlim=c(100,270),ylim=c(0,2e15),
xlab=xlabs,ylab=expression("H00 [cm"^-3*"]"),col="red",pch=1)
legend("top",bty="n",c("(c)"," ","H00"))
plot(temps,a[,"H01"],typ="o",xlim=c(100,270),ylim=c(0,2e17),
xlab=xlabs,ylab=expression("H01 [cm"^-3*"]"),col="green")
lines(temps,a[,"E00"],typ="o",col="gray",pch=3)
legend("topleft",bty="n",c("(d)"," ","H01"))
text(200,1.2e17,"E00")
plot(temps,a[,"E01"],typ="o",xlim=c(100,270),ylim=c(0,2e15),
xlab=xlabs,ylab=expression("E01 [cm"^-3*"]"),col="magenta")
legend("top",bty="n",c("(e)"," ","E01"))
plot(temps,a[,"H10"],typ="o",xlim=c(100,270),ylim=c(0,4e11),
col="black",xlab=xlabs,ylab=expression("H10,  E10 [cm"^-3*"]"))
lines(temps,a[,"E10"],typ="o",col="green",)
legend("top",bty="n",c("(f)"," ","H10"))
text(220,1e11, "E10")
```

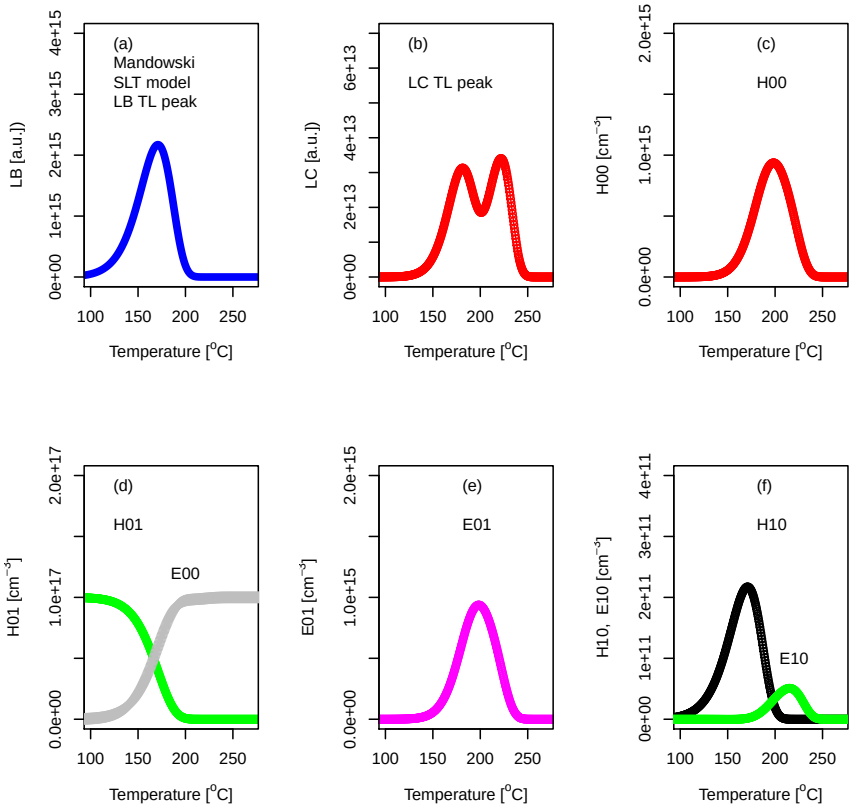


Fig. 7.8: Simulations of a TL process in the Mandowski SLT model, using the parameters given in the text, showing plots of the various concentrations H_m^n and E_m^n. (a) The L_B peak due to localized transitions in the model; (b) The double L_C peak from delocalized transitions; (c)-(f) The concentrations H_m^n and E_m^n as a function of temperature. *Caution:* Note that the numerical integration used in the *deSolve* package can often become numerically unstable for this type of model, see the discussion in the text.

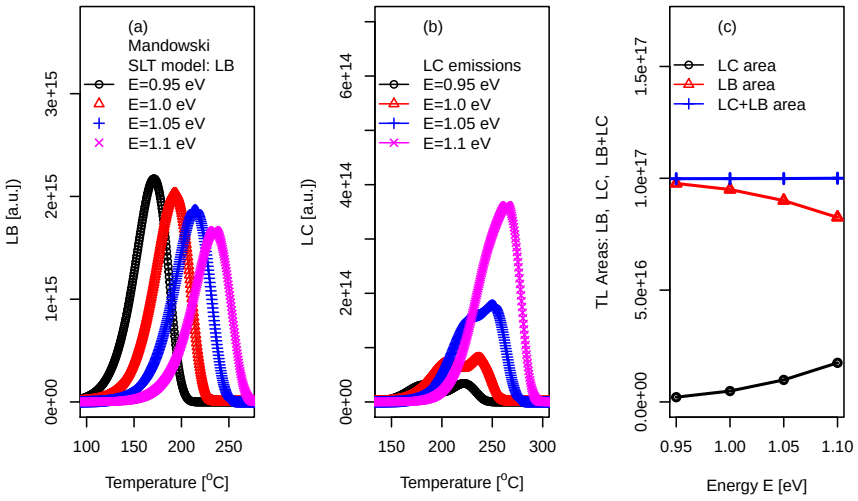


Fig. 7.9: Simulations of the Mandowski SLT model, using the parameters given in the text, and four different values of the parameter $E = 0.95, 1.0, 1.05, 1.1$ eV. (a) The L_B peaks due to localized transitions in the model; (b) The double L_C peaks from delocalized transitions; (c) The areas under the L_C and L_B peaks, and their sum $L_C + L_B$ as a function of the activation energy E.

Code 7.6: Mandowski model: the anomalous heating rate effect

```
#Mandowski model: the anomalous heating rate effect
rm(list=ls())
library("deSolve")
TLMandowski <- function(t, x, parms) {
  with(as.list(c(parms, x)), {
  dH01<- -(s*exp(-E/(kb*(273+hr*t)))+C*(H00-E01-E10))*H01+A*H10
    dH10 <- s*exp(-E/(kb*(273+hr*t)))*H01-(A+B+sv*exp(-Ev/(kb*
  (273+hr*t)))+C*(H00-E01-E10))*H10
    dH00<-sv*exp(-Ev/(kb*(273+hr*t)))*H10-C*(H00-E01-E10)*H00
  dE01<-C*(H00-E01-E10)*H01-s*exp(-E/(kb*(273+hr*t)))*E01+A*E10
    dE10<-C*(H00-E01-E10)*H10+s*exp(-E/(kb*(273+hr*t)))*E01-
```

```
      (A+sv*exp(-Ev/(kb*(273+hr*t))))*E10
dE00<-B*H10+C*(H00-E01-E10)*H00+sv*exp(-Ev/(kb*(273+hr*t)))*E10
    res <- c(dH01,dH10,dH00,dE01,dE10,dE00)
    list(res)  })}
hrVar<-function(hr){
  parms  <- c(E =1.05, s=1e10, sv=1e10, kb=8.617*10^-5,
              hr=hr, C=C,B=B,Ev=0.7,A=2*10^4)
  y <- xstart <- c(H01 = 10^17,   H10=0,H00=0,E10=0,E01=0,E00=0)
  out <-ode(xstart,times, TLMandowski, parms, method = "lsoda",
              atol = 1,rtol=1) }
C<-1e-10
B<-1e4
times  <- seq(0, 300,by=.5)
Lc<-Lb<-temps<-matrix(NA,nrow=length(times),ncol=4)
areaLb<-areaLc<-vector(length=4)
for (i in 1:4){
  hr<-i
  a<-hrVar(hr)
  Lc[,i]<-(C*(a[,"H00"]-a[,"E01"]-a[,"E10"])*
             (a[,"H10"]+a[,"H01"]+a[,"H00"]))/hr
  Lb[,i]<-B*a[,"H10"]/hr
  temps[,i]<-hr*times
  areaLc[i]<-sum(Lc[,i])*hr*0.5
  areaLb[i]<-sum(Lb[,i])*hr*0.5
}
par(mfrow=c(1,3))
var<-c(NA,NA,NA,1:4)
matplot(cbind(temps),cbind(Lb),lty="solid",typ="o",pch=1:4,
xlim=c(150,270),ylim=c(0,3e15),lwd=2, col=1:4 ,
xlab=expression("Temperature ["^"o"*"C]"), ylab="LB [a.u.])")
legend("topleft",bty="n",pch=var,lty="solid",col=var,
legend=c("(a)",
"Mandowski","SLT model: LB","1 K/s","2 K/s","3 K/s","4 K/s"))
matplot(cbind(temps),cbind(Lc),lty="solid",typ="o",pch=1:4,
xlim=c(170,300),ylim=c(0,7e14),lwd=2,col=1:4,
xlab=expression("Temperature ["^"o"*"C]"), ylab="LC [a.u.])")
legend("topleft",bty="n",pch=var,lty="solid",col=var,
legend=c("(b)"," ","LC peaks","1 K/s","2 K/s","3 K/s","4 K/s"))
matplot(1:4,cbind(areaLc,areaLb,areaLb+areaLc),lty="solid",
typ="o",lwd=3, pch=1:3,col=1:3,ylim=c(0,1.7e17),
xlab="Heating rate [K/s]",ylab="TL Areas: LB,  LC,  LB+LC")
legend("topleft",bty="n",pch=c(NA,NA,1:3),lty="solid",col=c(NA,
NA,1:3),legend= c("(c)"," ","LC area","LB area","LC+LB area"))
```

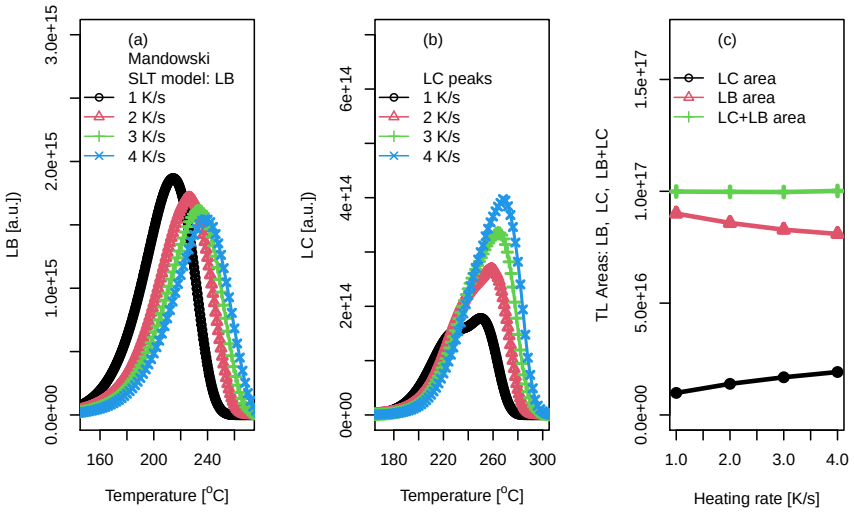


Fig. 7.10: Simulations of the anomalous heating rate effect using the Mandowski model, with the parameters given in the text, and for four different values of the heating rate $\beta = 1{-}4$ K/s. (a) The L_B peaks due to localized transitions in the model; (b) The L_C peaks from delocalized transitions. (c) The areas under the TL glow curves in (a),(b) and their sum $L_C + L_B$, as a function of the heating rate.

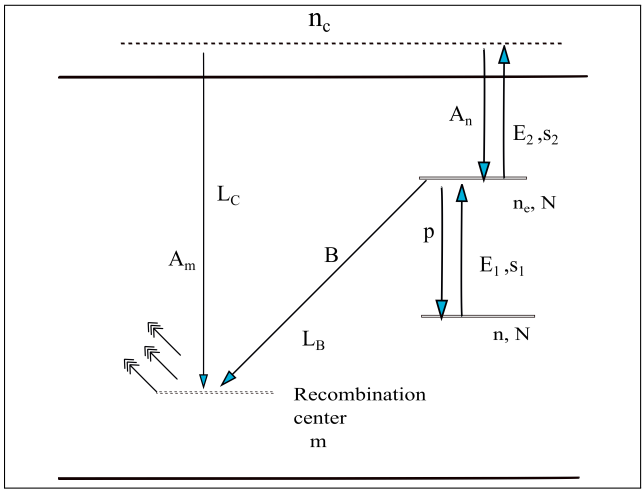


Fig. 7.11: The simplified SLT model proposed by Pagonis et al. [31].

Code 7.7: Pagonis SLT model: the anomalous heating rate effect

```
#Pagonis SLT model: the anomalous heating rate effect
rm(list=ls())
library("deSolve")
PagonisHR <- function(t, x, parms) {
  with(as.list(c(parms, x)), {
    dn<- -s1*n*exp(-E1/(kb*(273+hr*t)))+p*
      (An*(N-n)*(m-n)+s1*n*exp(-E1/(kb*(273+hr*t))))/
( s2*exp(-E2/(kb*(273+hr*t)))+p+B)
      dm<- -Am*m*(m-n)-B*
        (An*(N-n)*(m-n)+s1*n*exp(-E1/(kb*(273+hr*t))))/(
          s2*exp(-E2/(kb*(273+hr*t)))+p+B)
      res <- c(dn,dm)
    list(res)  })}
hrVar<-function(hr){
parms  <- c(p=p,s1=s1,s2=s2,An=An,Am=Am,
            B=B,N=N,E1=E1,E2=E2,kb=kb)
y <- xstart <- c(n = 1e11, m=1e11)
out <-  ode(xstart, times, PagonisHR, parms) }
```

```
p<-1e9
s1<-1e9
s2<-1e12
An<-1e-8
Am<-1e-6
B<-1e7
N<-1e13
E1<-0.8
E2<-0.5
kb<-8.617e-5
times  <- seq(0, 350,by=1)
Lc<-Lb<-temps<-matrix(NA,nrow=length(times),ncol=4)
areaLb<-areaLc<-vector(length=4)
for (i in 1:4){
  hr<-i
  a<-hrVar(hr)
  Lc[,i]<- Am*a[,"m"]*(a[,"m"]-a[,"n"])/hr
  Lb[,i]<-B*  ((An*(N-a[,"n"])*(a[,"m"]-a[,"n"])+s1*a[,"n"]*
 exp(-E1/(kb*(273+hr*times))))/( s2*exp(-E2/(kb*
(273+hr*times)))+p+B))/hr
   temps[,i]<-273+hr*times
  areaLc[i]<-sum(Lc[,i],rm.NA=TRUE)*hr
  areaLb[i]<-sum(Lb[,i],rm.NA=TRUE)*hr
}
par(mfrow=c(1,3))
var<-c(NA,NA,NA,1:4)
matplot(cbind(temps),cbind(Lb),lty="solid",typ="o",pch=1:4,
xlim=c(370,550),ylim=c(0,22e8),lwd=2, col=1:4 ,
xlab=expression("Temperature [K]"), ylab="LB [a.u.])")
legend("topleft",bty="n",pch=var,lty="solid",col=var,
legend=c("(a)",
"Pagonis","SLT model: LB","1 K/s","2 K/s","3 K/s","4 K/s"))
matplot(cbind(temps),cbind(Lc),lty="solid",typ="o",pch=1:4,
xlim=c(370,550),ylim=c(0,5e8),lwd=2,col=1:4,
xlab=expression("Temperature [K]"), ylab="LC [a.u.])")
legend("topleft",bty="n",pch=var,lty="solid",col=var,
legend=c("(b)"," ","LC emissions","1 K/s","2 K/s","3 K/s",
"4 K/s"))
matplot(1:4,cbind(areaLc,areaLb,areaLb+areaLc),lty="solid",
typ="o",lwd=3, pch=1:3,col=1:3,ylim=c(0,1.7e11),
xlab="Heating rate [K/s]",ylab="TL Areas: LB,  LC,  LB+LC")
legend("topleft",bty="n",pch=c(NA,NA,1:3),lty="solid",
col=c(NA,NA,
1:3),legend= c("(c)"," ","LC area","LB area","LC+LB area"))
```

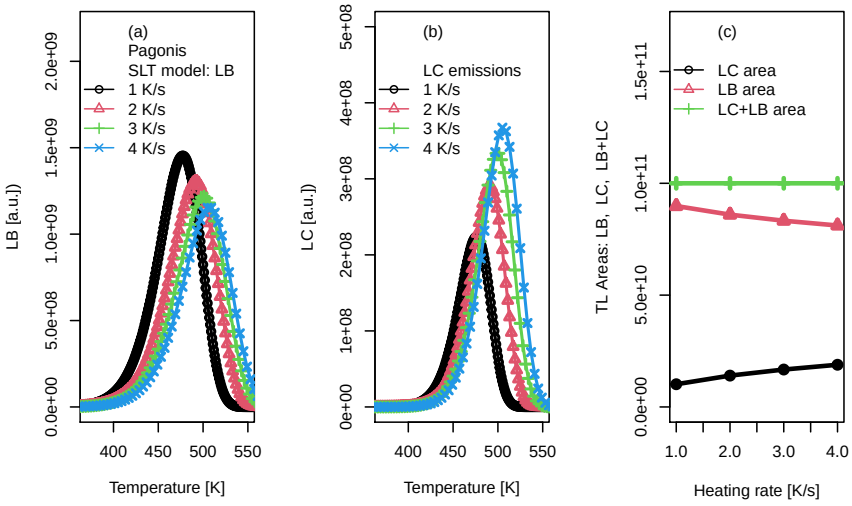


Fig. 7.12: Simulations of the anomalous heating rate effect using the simplified SLT model by Pagonis et al. [31], with the parameters given in the text, and for four different values of the heating rate $\beta = 1 - 4$ K/s. (a) The L_B peaks due to localized transitions in the model; (b) The L_C peaks from delocalized transitions; (c) The areas under the TL glow curves in (a),(b) and their sum $L_C + L_B$, as a function of the heating rate.

Part III
MONTE CARLO SIMULATIONS OF LUMINESCENCE SIGNALS

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
  ##    2.85    0.01    2.90
```

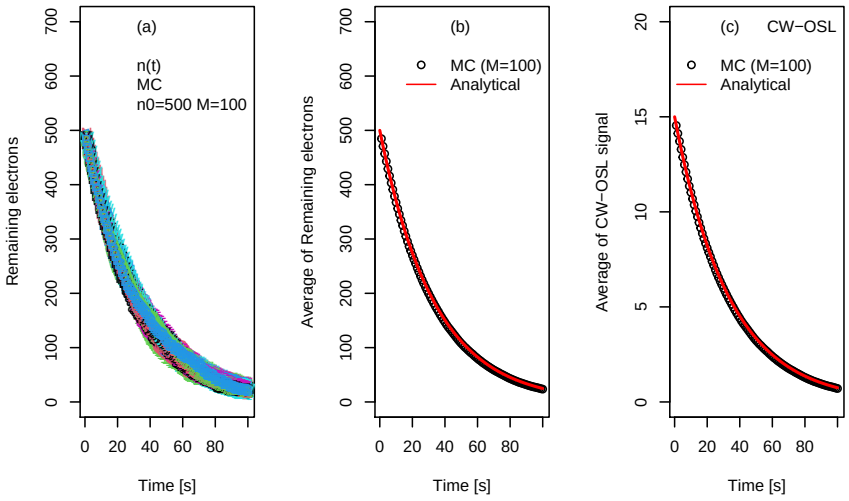


Fig. 8.1: Simplest MC implementation of CW-OSL luminescence process. (a) Plot of M =100 MC runs with the same initial number of electrons n_0=500, simulating a total of 50,000 electrons. (b) Plot of the average of the M =100 MC iterations in (a). The solid lines in (b) and (c) represents the analytical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

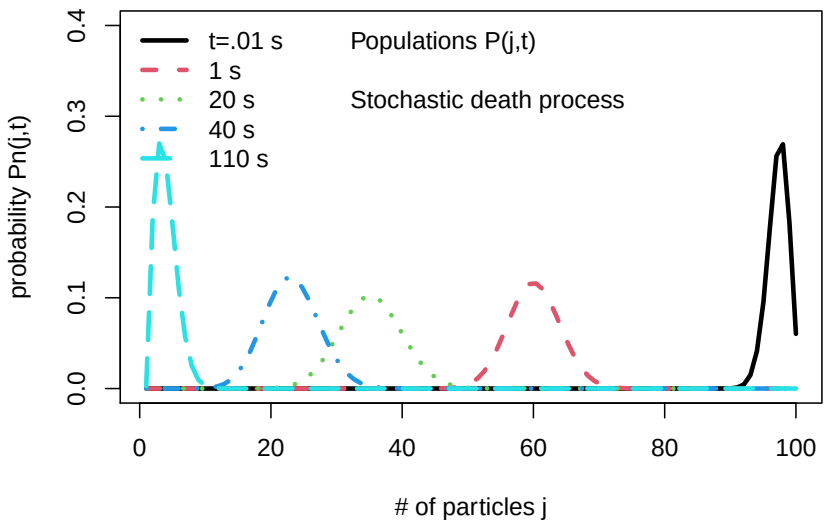


Fig. 8.2: Plots of $P_j(t)$ from Eq.(**??**), for a simple death stochastic process. As the time t increases from right to left in this plot, the width initially increases and then decreases with t. For more details see Lawless et al. [21].

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[%]"))
```

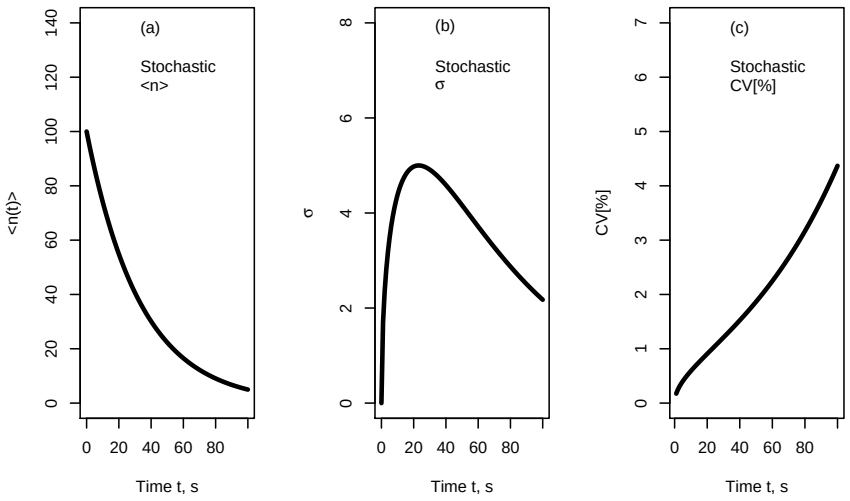


Fig. 8.3: (a) Plot of stochastic mean number of particles $< n(t >$. (b) Plot of stochastic standard deviation σ_n . (c) Plot of *CV[%]* from Eq.(**??**),(**??**) and (**??**).

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
  ##    0.33    0.00    0.33
```

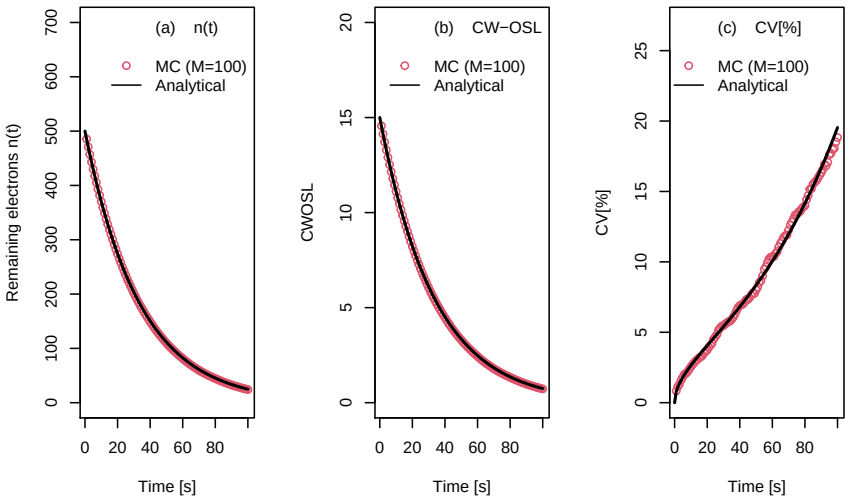


Fig. 8.4: Vectorized MC implementation of the first order CW-OSL luminescence process. (a) Plot of M =100 MC runs with the same initial number of electrons n_0=500, simulating a total of 50,000 electrons. (b) Average of the M =100 MC iterations in (a). (c) The corresponding *CV[%]*. The solid lines represent the analytical solution of the differential equation. For more details ad examples, see Pagonis et al. [41].

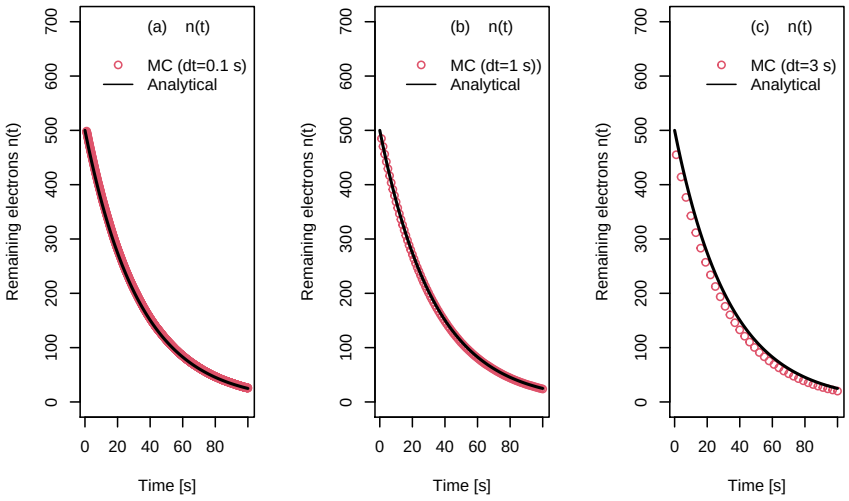


Fig. 8.5: The effect of the parameter *deltat* in the previous MC implementation of the first order CW-OSL luminescence process, with (a) *deltat*=0.1 s, (b) *deltat*=1 s, (c) *deltat*=3 s. All three runs are carried out with the same parameters M =100 MC runs, initial number of electrons n_0=500, $\mu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

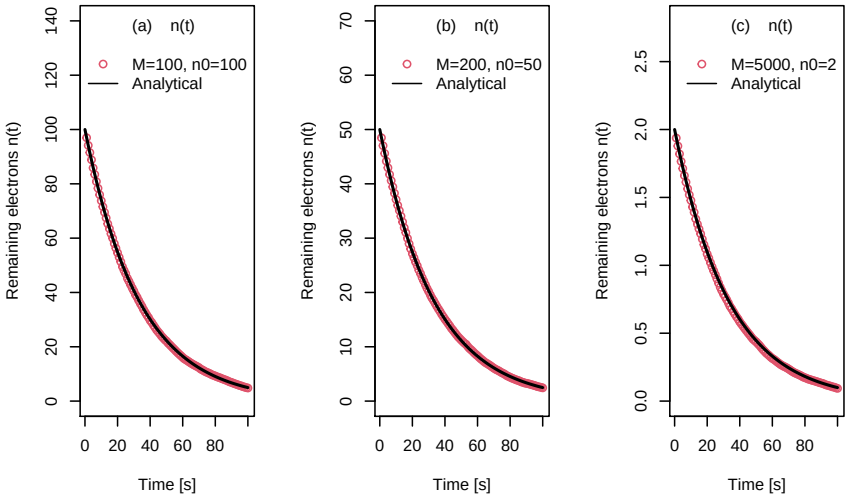


Fig. 8.6: The effect of the parameter n_0 in the previous MC implementation of the first order CW-OSL luminescence process, with (a) $n_0 = 100$ and $M = 100$ MC runs, (b) $n_0 = 50$ and $M = 200$, (c) $n_0 = 2$ and $M = 5000$. All three runs are carried out with the same parameters $\mu = 0.03$ s^{-1} 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
  ##    0.30    0.00    0.29
```

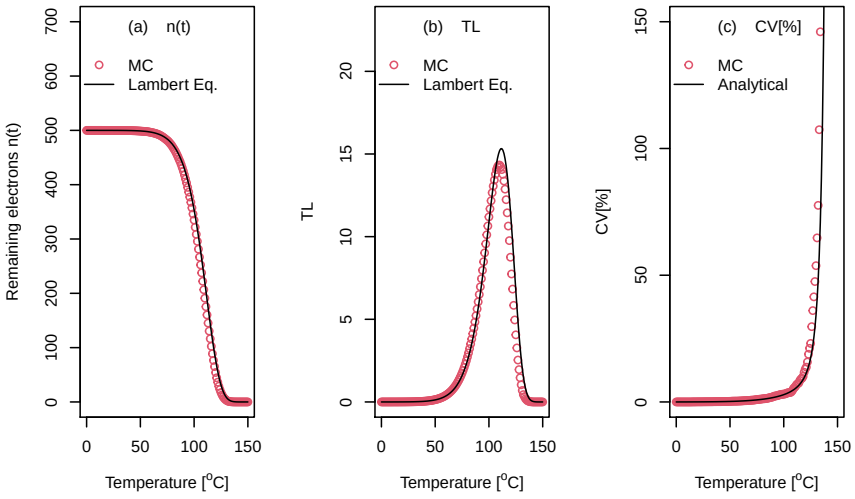


Fig. 8.7: Vectorized MC implementation of the first order TL luminescence process. (a) Plot of $< n(t) >$ for M =100 MC runs with the same initial number of electrons n_0=500, simulating a total of 50,000 electrons; (b) Average of the corresponding TL signal. (c) The corresponding *CV[%]*. The solid lines represent the analytical equations. For details, see Pagonis et al. [41].

8.1 Vectorized MC simulation of first order LM-OSL process

The following vectorized code shows an example of a first order kinetics LM-OSL curve, obtained with the same method as in the previous section, with the results shown in Fig.8.8.

The LM-OSL simulation contains $M = 100$ MC runs for a system of $n_0 = 500$ initially trapped electrons, with an excitation rate $p(t) = \sigma I t/P = 0.2$ s^{-1} and a total excitation period of $P = 60$ s. Figure 8.8a shows the results of the average $< n(t) >$ of the MC runs, while Fig.8.8b shows the corresponding LM-OSL signal.

Fig.8.8c is a plot of the stochastic coefficient of variation CV[%] of the LM-OSL intensity, as a function of excitation time t. As the time increases, CV[%] also *increases* monotonically. The solid lines are the analytical solutions of the corresponding differential equation for the deterministic process (see the extensive discussion in Chapter 3).

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
```

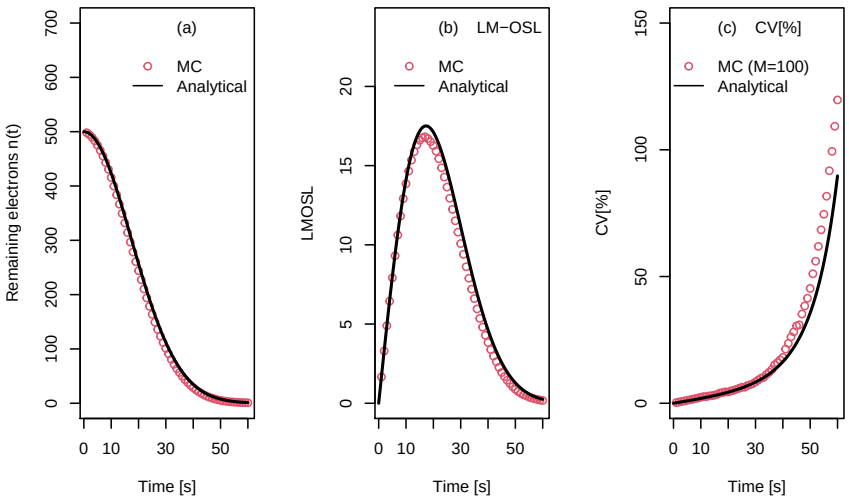


Fig. 8.8: Vectorized MC implementation of the first order LM-OSL luminescence process. (a) Plot of the mean $< n(t) >$ for M =100 MC runs with the same initial number of electrons n_0=500, simulating a total of 50,000 electrons. (b) Average of the corresponding LM-OSL signal. (c) The corresponding *CV[%]*.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

8.2 Vectorized MC simulation of TL in the GOT model

Pagonis et al. [35] used the simple MC method described in the previous section, to solve the differential equation for the GOT model. An example of their method is presented in this section; this example also uses a fixed time interval.

As we saw in Chapter 2, the GOT differential equation for TL in the OTOR model is:

$$I(t) = -\frac{dn}{dt} = s\, e^{-\frac{E}{kT(t)}}\, \frac{n^2}{(N-n)R+n} \tag{8.1}$$

where $R = A_n/A_m$ is the retrapping ratio in the GOT model, $E\,(eV)$ is the thermal activation energy of the trap, and $s\,(s^{-1})$ is the associated frequency factor. This equation becomes a difference equation:

$$\Delta n = -s\, e^{-\frac{E}{kT(t)}}\, \frac{n^2}{(N-n)R+n} \Delta t \tag{8.2}$$

The code below solves Eq.(8.2), by using the same MC method as in the previous sections. A fixed time interval Δt =1 s is used in the simulation. The results are shown in Fig.8.9.

The corresponding TL intensity $I_{TL}(T)$ is calculated from the values of Δn in Eq.(8.2) using the expression:

$$I_{TL}(T) = -\frac{1}{\beta}\frac{\Delta n}{\Delta t} = -\frac{1}{\beta} n\, s\, \exp\left[-\frac{E}{k_B T}\right] \tag{8.3}$$

Figure 8.9a shows the results of the average $< n(t) >$ for M =100 MC runs, while Fig.8.9b shows the corresponding mean TL signal.

Fig.8.9c is a plot of the stochastic coefficient of variation CV[%] of the TL intensity, as a function of temperature. As the temperature increases, CV[%] also increases monotonically. The solid line in Figs.8.9ab is the analytical solution of the corresponding differential equation for the deterministic process, in terms of the Lambert W function (see the extensive discussion in Chapter 2). There are *no* analytical solutions for the stochastic coefficient of variation CV[%] shown in Fig.8.9c. This is because of the non-linear nature of the deterministic differential equation in the GOT model.

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
##    0.35    0.00    0.34
```

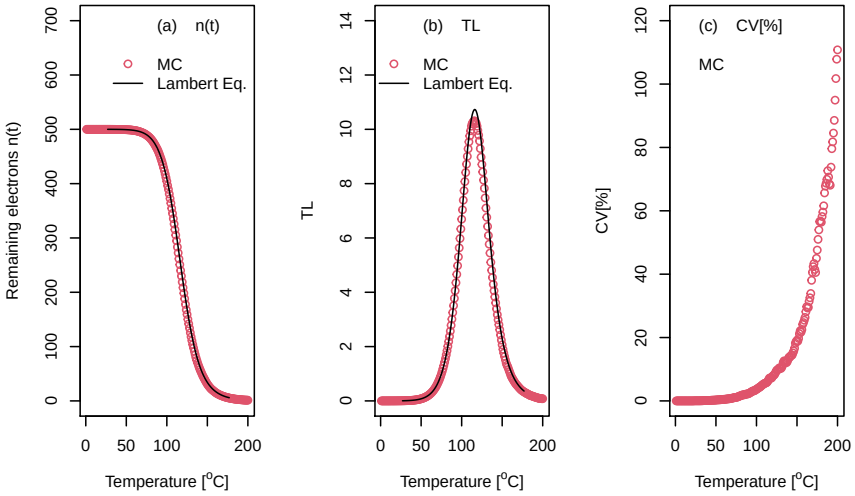


Fig. 8.9: Vectorized MC simulation of TL in a system of $M = 100$ large clusters of defects, based on the GOT equation. The total number of traps in each cluster is $N = 1000$, and initially $n_0 = 500$ of these traps are filled. (a) The average $n(t)$ (b) The average TL signal; (c) The corresponding $CV[\%]$. The solid lines in (a) and (b) represent the analytical equations which are based on the Lambert function. For details, see Pagonis et al. [41].

8.3 Monte Carlo simulation of TL/OSL from a system of small trap clusters

The MC method becomes most useful when one is dealing with a small number of electrons and traps, as in the case of nanodosimetric materials. The example below simulates such a system, with the details given in the paper by Pagonis et al. [35].

For a small system of trap clusters, one can make a clear distinction between *local variables* describing the internal structure of each cluster, and *global variable*s which describe the whole group of clusters. The system simulated by these authors has some similarities to the one used by Mandowski and Świątek [25], and consists of a large number of small clusters of traps and recombination centers. The local physical parameters characterizing each cluster are: the total number of traps per cluster N_{traps}, the number of initially filled traps per cluster n_{filled}, (in general $n_{filled} \leq N_{traps}$), and the instantaneous number of remaining filled traps in the cluster denoted by n_{local}. As the system develops in time during the optical or thermal stimulation process, the value of the local variable n_{local} will decrease from its initial value down to zero, as more recombinations take place within the cluster.

In terms of global variables, the system is described by the number of trap clusters in the system $N_{clusters}$. The total number of available traps in the system is given by the product $N = N_{clusters}N_{traps}$, while the total number of initially filled traps n_0 is given by $n_0 = N_{clusters}n_{filled}$ (with $n_0 \leq N$). The equalities $n_{filled} = N_{traps}$ and $n_0 = N$, denote a system with all traps initially filled. The total instantaneous number of filled tarps is denoted by the global variable n, which is calculated as the sum of remaining filled traps n_{filled} over all clusters in the system, i.e.:

$$n = \sum_{all\ clusters} n_{local} \tag{8.4}$$

Clearly the variables n_{local}, n_{filled} and N_{traps} represent *local* variables characterizing each cluster in the system, while the *global* variables $N_{clusters}$, n_0, n and N characterize the whole system of trap clusters.

Figure 8.10 shows schematically an example of such a system of small trap clusters, in which there are 4 traps in each cluster (shown as both open and solid circles), with only 3 of them being initially filled (shown as solid circles). One ensures the charge balance in the system, by assuming the existence of an equal number of 4 luminescence centers (shown as both open and solid stars), 3 of which have been activated (shown as solid stars).

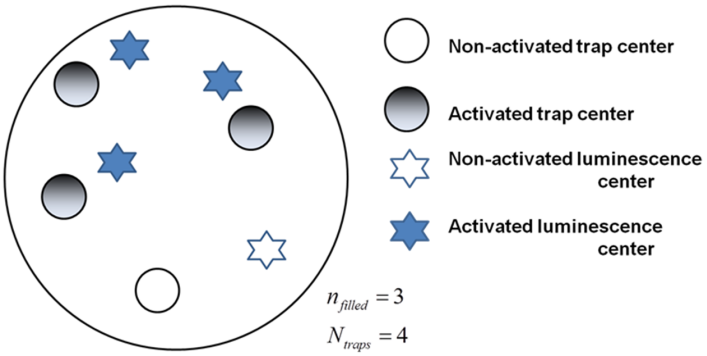


Fig. 8.10: Schematic representation of a small trap cluster, consisting of a total of four traps in each cluster ($N_{traps} = 4$ shown as both open and solid circles). Only three of these traps are initially filled ($n_{filled} = 3$ shown as solid circles). Charge balance in the system is ensured by 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. [35].

In this example one could simulate, for instance, a large number of clusters in the system, resulting in a total number of initially filled traps. From a physical point of view, the activated luminescence centers may exist in physical proximity to the filled traps, since they both could have been created simultaneously during the irradiation process. As the system of trap clusters in Fig.8.10 develops in time, the local variable n_{local} will vary from an initial value of n_{filled} to zero at the end of the thermal/optical excitation process. Similarly the global variable n will vary from an initial value of n_0 to its final value of zero at the end of the process.

The luminescence intensity from the overall system of trap clusters will consist of the sum of signals from all clusters in the system.

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. [35].

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

The implicit physical assumption in this description is that each cluster acts as an independent entity as far as the luminescence process is concerned, since each electron participates only in local processes within the cluster. The result of the following R code shows that the TL intensity in this system of small clusters contains two peaks, while the solution of the corresponding stochastic differential equation for the corresponding system of large clusters contains a single TL peak. It is noted that there are no analytical equations to describe these graphs in Fig.8.11, and one must use MC to simulate such a system.

The parameters in this simulation are: $n_0 = 2$, $N = 3$, $s = 10^{12}$ s^{-1},$E =$ 0.8 eV, and a large retrapping ratio R =100. The large value of R is the cause of the double peak structure shown in Fig.8.11.

This result in Fig.8.11 has certain similarity to the double peak structure obtained in the SLT model by Mandowski [24], although the physical descriptions of the two models are very different.

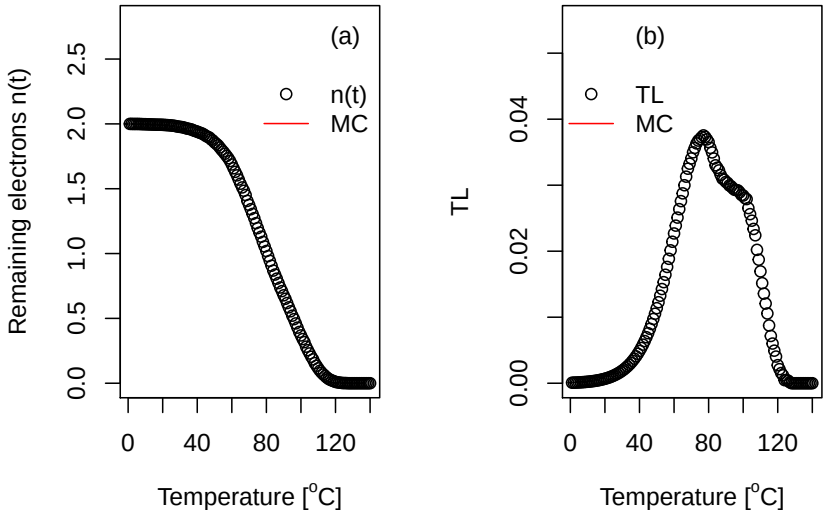


Fig. 8.11: Simulation of TL signal from a system of $M = 3000$ small clusters of defects, based on the GOT equation. The total number of traps in each cluster is $N = 3$, and initially $n_0 = 2$ of these traps are filled. The double peak structure is caused by the large retrapping ratio $R = 100$. For a more detailed description, see Pagonis et al. [35].

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

8.4 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):

$$\frac{dn}{dt} = \frac{(N-n)R}{(N-n)R+n} X \tag{8.5}$$

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.(8.5) 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:

$$n/N = 1 + W[(R-1)\exp(R-1-RXt/N)]/(1-R) \tag{8.6}$$

The MC simulations simplify by dividing both sides of Eq.(8.5) by X, to obtain:

$$\frac{dn}{d(Xt)} = \frac{(N-n)R}{(N-n)R+n} \tag{8.7}$$

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:

$$\Delta(n) = \frac{(N-n)R}{(N-n)R+n}\Delta D \tag{8.8}$$

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
##    0.73    0.00    0.74
```

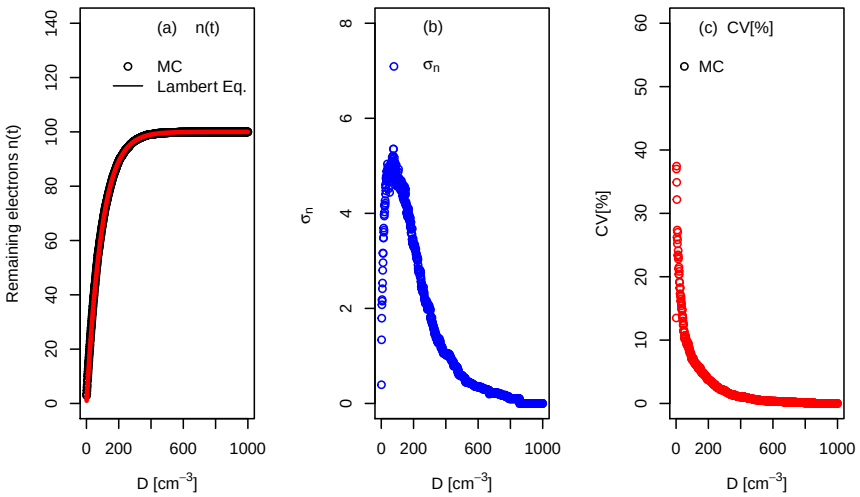


Fig. 8.12: MC simulation of irradiation process in the GOT model, as a nonlinear pure birth problem. The parameters are $n_0 = 1$, $M = 100$ MC runs, $R = 1.2$, $N = 100$. (a) Plot of the trapped electrons $n(t)$. The solid line is the Lambert W solution discussed in Chapter 4. (b) Plot of the uncertainty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

Chapter 9
MONTE CARLO SIMULATIONS OF LOCALIZED TRANSITIONS

Abstract In this chapter we discuss fixed time interval MC simulations of luminescence signals produced by localized transitions in the TLT and LT models. Examples of R codes are provided for TL, CW-IRSL and LM-IRSL signals in the excited state tunneling (EST) model, and the MC results are compared with the analytical Kitis-Pagonis equations (KP-CW, KP-TL). The R codes also provide estimates for the stochastic coefficients of variation CV% for a variety of processes.This chapter concludes with a MC simulation for the LT model.

Code 9.1: Vectorized MC code for tunneling TL transitions (TLT model)

```
# Vectorized MC code for tunneling transitions (TLT model)
# Original Mathematica program by Vasilis Pagonis
# R version written by Johannes Friedrich, 2018
rm(list = ls(all=T))
rho <- 5e-3
En<-1.43
s<-3.5e12
kB<-8.617e-5
deltat <- 1
times <- seq(0, 500, deltat)
# In this example time =temperature, i.e. a heating rate=1 K/s
r <- seq(0, 2, 0.1)
clusters <- 20
n0<-100
signal<-array(0,dim=c(length(times),
  ncol = length(r), clusters))
# Run MC simulation
```

```
system.time(invisible(for(c in 1:clusters)
{
  for(k in 1:length(r)){
    n <- n0
    for (t in 1:length(times)){
      P <- s*exp(-En/(kB*(t+273)))*exp(-rho^(-1/3) * r[k])
      vec<-rep(runif(n))
      n<-length(vec[vec>P*deltat])
      signal[t,k,c] <- n * P * 3 * r[k]^2 * exp(-r[k]^3) }}})
)
par(mfrow=c(1,2))
# plot an example : the result from the first cluster
matplot(signal[,,1],type = "l",lty="solid",
ylab = "Partial TL glow curves for constant(r')",
ylim=c(0,3.2),xlab=expression("Temperature ["^"o"*"C]"),lwd = 2)
legend("topleft",bty="n",legend=c("(a)"," ","Partial TL",
"glow curves"))
#add the signals from all clusters
sum_signal <- sapply(1:clusters, function(y){
  vapply(1:length(times), function(x){
    sum(signal[x,,y])
  }, FUN.VALUE = 1)  })
# add the signals from all r values
TL <- rowMeans(sum_signal)
# plot and normalize the TL signal
plot(  x = times, y = TL/max(TL),type = "l", lwd = 3,
ylim=c(0,1.6), xlab=expression("Temperature ["^"o"*"C]"),
ylab="Average TL signal")
legend("topleft",bty="n",legend=c("(b)"," ",
"Sum of partial TL","glow curves"))
## plot analytical solution Kitis-Pagonis
z<-1.8
T<-times+273
TLanalyt<-exp(-rho*( (log(1+z*s*kB*((T**2.0)/
 abs(En))*exp(-En/(kB*T))*(1-2*kB*T/En)))**3.0))*
(En**2.0-6*(kB**2.0)*(T**2.0))*( (log(1+z*s*kB*((T**2.0)/
   abs(En))*exp(-En/(kB*T))*(1-2*kB*T/En)))**2.0)/
   (En*kB*s*(T**2)*z-2*(kB**2.0)*s*z*(T**3.0)+
exp(En/(kB*T))*En)
lines(times,TLanalyt/max(TLanalyt),lty="dashed",col="red",
lwd=3)
```

```
##    user  system elapsed
##    1.51    0.00    1.53
```

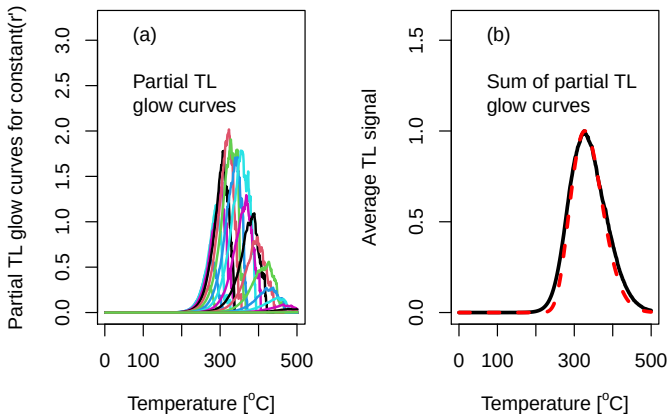


Fig. 9.1: MC simulation of TL signals in the TLT model, for the parameters $\rho' = 5 \times 10^{-3}$, M =20 MC runs, $n_0 = 100$ initially trapped electrons, $E = 1.43$ eV, $s = 3.5 \times 10^{13}$ s^{-1}. (a) Example of partial TL glow curves evaluated for each distance r' . (b) The sum of the partial TL glow curves from (a), normalized to its maximum. The dashed line in (b) represents the approximate analytical KP-TL equation from Chapter 6, also normalized to its maximum value.

Code 9.2: Vectorized MC code for tunneling CW-IRSL transitions (TLT model)

```
# Vectorized MC code for tunneling transitions (TLT model)
# Original Mathematica program by Vasilis Pagonis
# R version written by Johannes Friedrich, 2018
rm(list = ls(all=T))
rho <- 5e-3
A<-2
deltat <- 1
times <- seq(1, 400, deltat)
# In this example time =temperature, i.e. a heating rate=1 K/s
r <- seq(0, 2.2, 0.1)
clusters <- 10
n0<-500
signal<-array(0,dim=c(length(times),
```

```
                      ncol = length(r), clusters))
# Run MC simulation
system.time(invisible(for(c in 1:clusters)
{
  for(k in 1:length(r)){
    n <- n0
    for (t in 1:length(times)){
      P <- A*exp(-rho^(-1/3) * r[k])
      vec<-rep(runif(n))
      n<-length(vec[vec>P*deltat])
      signal[t,k,c] <- n * P * 3 * r[k]^2 * exp(-r[k]^3) }}})
)
par(mfrow=c(1,2))
# plot an example : the result from the first cluster
matplot(signal[,,1],type = "l",lty="solid",
ylab = "Partial CW-IRSL curves",
ylim=c(0,14),xlab=expression("Time [s]"),lwd = 2)
legend("topleft",bty="n",legend=c("(a)"," ","Partial CW-IRSL",
" curves"))
#add the signals from all clusters
sum_signal <- sapply(1:clusters, function(y){
  vapply(1:length(times), function(x){
    sum(signal[x,,y])
  }, FUN.VALUE = 1)  })
# add the signals from all r values
TL <- rowMeans(sum_signal)
# plot and normalize the TL signal
plot(  x = times, y = TL/max(TL),type = "l", lwd = 3,
ylim=c(0,1.4), xlab=expression("Time [s]"),
       ylab="Average signal")
legend("topleft",bty="n",legend=c("(b)"," ",
"Sum of partial","CW-IRSL curves"))
## plot analytical solution Kitis-Pagonis
z<-1.8
CWanalyt<-exp(-rho*( (log(1+z*A*times))**3.0))*
  ( (log(1+z*A*times))**2.0)/(1+z*A*times)
lines(times,CWanalyt/max(CWanalyt),lty="dashed",col="red",lwd=3)

  ##    user  system elapsed
  ##    1.35    0.00    1.36
```

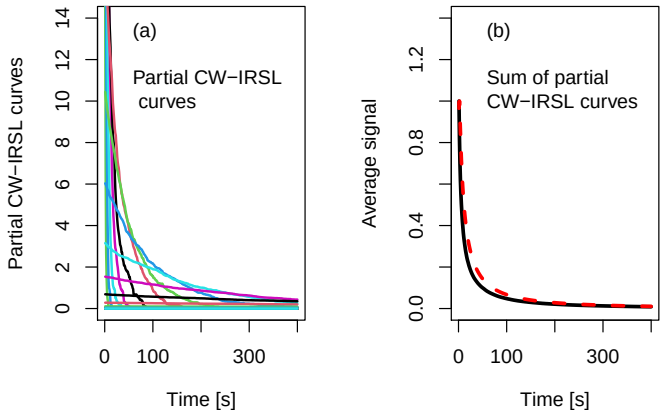


Fig. 9.2: MC simulation of CW-IRSL signals in the TLT model, for the parameters $\rho' = 5 \times 10^{-3}$, M =10 MC runs, $n_0 = 500$ initially trapped electrons, and IR excitation rate $A = 2$ s^{-1}. (a) Example of partial CW-IRSL curves evaluated for each distance r' . (b) The sum of the partial CW-IRSL curves from (a), normalized to its maximum. The dashed line in (b) represents the approximate analytical KP-CW equation from Chapter 6, also normalized to its maximum value (kitis and Pagonis [13]).

Code 9.3: Vectorized MC code for tunneling LM-IRSL transitions (TLT model)

```
# Vectorized MC code for tunneling transitions (TLT model)
# Original Mathematica program by Vasilis Pagonis
# R version written by Johannes Friedrich, 2018
rm(list = ls(all=T))
rho <- 5e-3
A<-2                # IR exciation rate in s^-1
deltat <- 1
times <- seq(0, 400, deltat)
r <- seq(0, 2.2, 0.1)
clusters <- 10
n0<-500
signal<-array(0,dim=c(length(times),
ncol = length(r), clusters))
```

```
# Run MC simulation
system.time(invisible(for(c in 1:clusters)
{  for(k in 1:length(r)){
    n <- n0
    for (t in 1:length(times)){
      P <- A*t/max(times)*exp(-rho^(-1/3) * r[k])
      vec<-rep(runif(n))
      n<-length(vec[vec>P*deltat])
 signal[t,k,c] <- n * P * 3 * r[k]^2 * exp(-r[k]^3) }}}))
par(mfrow=c(1,2))
# plot an example : the result from the first cluster
matplot(signal[,,1],type = "l",lty="solid",
ylab = "Partial LM-IRSL curves",
ylim=c(0,7),xlab=expression("Time [s]"),lwd = 2)
legend("topleft",bty="n",legend=c("(a)"," ","Partial LM-IRSL",
" curves"))
#add the signals from all clusters
sum_signal <- sapply(1:clusters, function(y){
  vapply(1:length(times), function(x){
    sum(signal[x,,y])
  }, FUN.VALUE = 1)  })
# add the signals from all r values
TL <- rowMeans(sum_signal)
# plot and normalize the TL signal
plot(  x = times, y = TL/max(TL),type = "l", lwd = 3,
ylim=c(0,1.6), xlab=expression("Time [s]"),
ylab="Average LM-IRSL signal")
legend("topleft",bty="n",legend=c("(b)"," ",
"Sum of partial","LM-IRSL curves"))
## plot analytical solution Kitis-Pagonis
z<-1.8
LManalyt<-exp(-rho*( (log(1+z*A*times^2/(2*max(times))))**3.0))*
times*((log(1+z*A*times^2/(2*max(times))))**2.0)/(1+z*A*times^2/
(2*max(times)))
lines(times,LManalyt/max(LManalyt),lty="dashed",col="red",lwd=3)
```

```
##    user  system elapsed
##    1.75    0.00    1.74
```

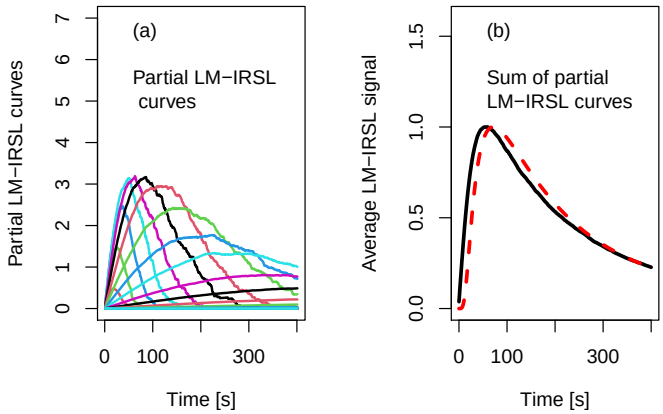


Fig. 9.3: MC simulation of LM-IRSL signals in the TLT model, for the parameters $\rho' = 5 \times 10^{-3}$, M =10 MC runs, $n_0 = 100$ initially trapped electrons, $A = 2\ s^{-1}$. (a) Example of partial LM-IRSL curves evaluated for each distance r' ; (b) The sum of the partial LM-IRSL curves from (a), normalized to its maximum. The dashed line in (b) represents the approximate analytical equation by Kitis and Pagonis [13], also normalized to its maximum value.

Code 9.4: Vectorized MC code for TL in localized TL transitions (LT model)

```
# Vectorized MC code for localized TL transitions (LT model)
rm(list = ls(all=T))
options(warn=-1)
library(matrixStats)
library(lamW)
mcruns<-100
n0<-500
s<-1e13
E<-1
kb<-8.617e-5
r<-1e2
tmax<-120
deltat<-1
times<-seq(1,tmax,deltat)  #heating rate=1 K/s
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/(r+n)*deltat
      n<-length(vec[vec>P])
      nMC[t]<-n
      TL[t]<-nMC[t]*P}
    nMatrix[,j]<-nMC
    TLMatrix[,j]<-TL })
#Find average avgn, average TL signal,CV[%]
avgn<-rowMeans(nMatrix)
avgTL<-rowMeans(TLMatrix)
sd<-rowSds(TLMatrix)
cv<-100*sd/avgTL
# plots
par(mfrow=c(1,2))
pch<-c(NA,NA,1,NA)
lty<-c(NA,NA,NA,"solid")
col<-c(NA,NA,"black","red")
plot(times,avgTL,ylab="TL",
     xlab=expression("Temperature ["^"o"*"C]"))
# plot analytical solution
k<-function(u) {integrate(function(p){exp(-E/(kb*p))},
300,u)[[1]]}
y1<-lapply(times+273,k)
x<-unlist(273+times)
y<-unlist(y1)
zTL<-(r/n0)-log(n0/r)+(s*y)
lines(x-273,r*s*exp(-E/(kb*x))/(lambertW0(exp(zTL))
+lambertW0(exp(zTL))^2),type="l",col="red")
legend("topleft",bty="n",c("(a)  TL"," ","MC",
                           "Analyt."),pch=pch,lty=lty,col=col)
plot(times,cv,ylab="(b)  CV[%]",ylim=c(0,120),
     xlab=expression("Temperature ["^"o"*"C]"))
legend("topleft",bty="n",c("CV[%]"," ","MC"),
pch=pch,lty=lty,col=col)
```

```
##    user  system elapsed
##    0.24    0.00    0.23
```

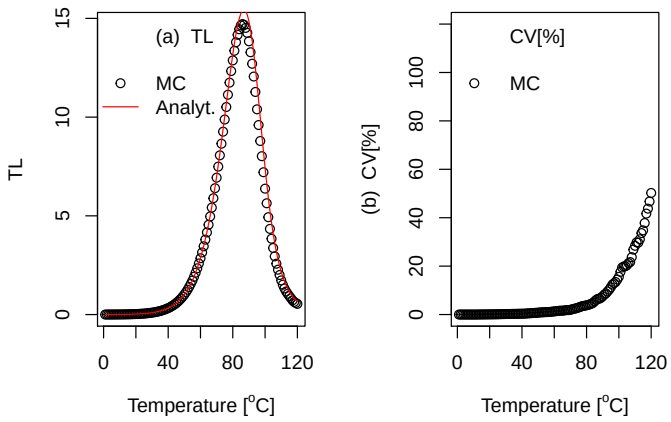


Fig. 9.4: (a) MC simulation of TL signals in the LT model, for the parameters $r = 10^2$, M =100 MC runs, $n_0 = 500$, $E = 1$ eV, $s = 10^{13}$ s^{-1}. The solid line is the analytical solution by Kitis and Pagonis [13]. (b) The corresponding coefficient of variation *CV[%]*.

retrapping ratio $r = 10^2$, 10^3, 10^4.

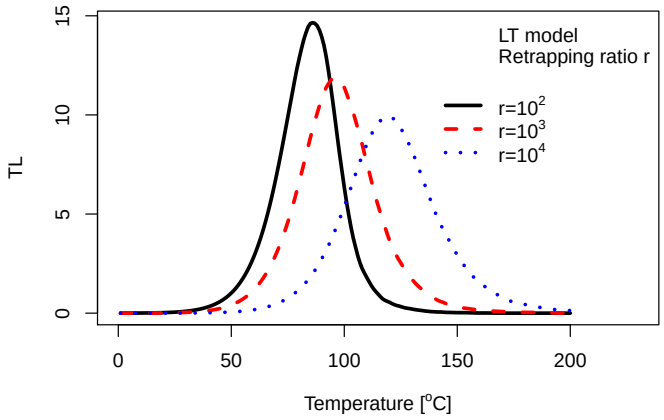


Fig. 9.5: MC simulation of TL signals in the LT model, for three different values of the retrapping ratio $r = 10^2$, 10^3, 10^4 cm^{-3}.

Chapter 10
KINETIC MONTE CARLO SIMULATIONS

Abstract While previous chapters looked at examples of MC methods with a fixed time interval, in this chapter we present several examples of Kinetic Monte Carlo methods (KMC). KMC methods are based on the concept that the variable time intervals between random events follow an exponential distribution. R codes are provided here for KMC methods applied to luminescence signals as stochastic birth and death processes. The chapter concludes with a different example of the KMC method, which describes quantum tunneling processes and the anomalous fading effect from a microscopic point of view. We compare the stochastic KMC results and the corresponding CV% uncertainties with the solution of the corresponding deterministic differential equation.

Code 10.1: Stochastic CW-OSL process using KMC

```
#  Stochastic CW-OSL process using KMC
rm(list = ls(all=T))
library(matrixStats)
mcruns<-100
n0<-500
mu<-.03
tmax<-100
times<-(1:tmax)
nMatrix<-matrix(NA,nrow=tmax-1,ncol=mcruns)
system.time(
for (k in 1:mcruns)
{n<-n0
 allt<-t<-0.5
    for(i in 1:n){
    P<-mu
```

```
    t<-t+rexp(1)/(P*n)
    n<-n-1
    allt<- rbind(allt,t)
    }
 depth.class <- cut(allt,times, include.lowest = TRUE)
 singlerun <- tapply(seq(from=n0,to=0,by=-1), depth.class,
mean,na.rm = FALSE)
 singlerun<-as.vector(singlerun)
 nMatrix[,k]<-singlerun
})
par(mfrow=c(1,3))
matplot(nMatrix,typ="l",col="green",ylim=c(0,700),
xlab="Time [s]",ylab="Remaining electrons")
timesMC<-seq(from=1,to=tmax-1,by=1)
avgn<-rowMeans(nMatrix,na.rm=TRUE)
avgCWOSL<-mu*avgn
sd<-rowSds(nMatrix,na.rm=TRUE)
cv<-100*sd/avgn
pch<-c(NA,NA,1,NA)
lty<-c(NA,NA,NA,"solid")
col<-c(NA,NA,"black","red")
lines(timesMC,avgn,ylab="Remaining electrons n(t)",
     xlab="Time [s]")
curve(n0*exp(-P*x),from=0,to=150,add=TRUE,col="red",lwd=2)
legend("topright",bty="n",c("(a)  n(t)"," ","MC (M=100)",
  "Analytical"),pch=pch,lty=lty,col=col)
plot(timesMC,avgCWOSL,ylab="CWOSL",xlab="Time [s]",ylim=c(0,21))
legend("topright",bty="n",c("(b)  CW-OSL"," ","MC (M=100)",
   "Analytical"),pch=pch,lty=lty,col=col)
curve(n0*mu*exp(-mu*x),from=0,max(times),add=TRUE,
col="red",lwd=2)
plot(timesMC,cv,ylab="CV[%]",xlab="Time [s]",ylim=c(0,40))
curve(100*sqrt((exp(mu*x)-1)/n0),0,max(times),add=TRUE,
      col="red",lwd=2)
legend("topleft",bty="n",c("(c)  CV[%]"," ","MC (M=100)",
   "Analytical"),pch=pch,lty=lty,col=col)

  ##    user  system elapsed
  ##    0.37    0.00    0.38
```

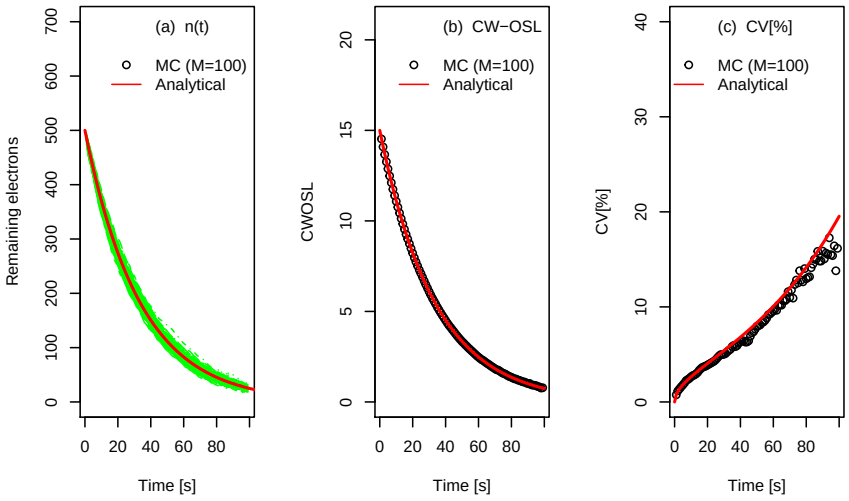


Fig. 10.1: Simulation of CW-OSL as a stochastic death process, using KMC.

Code 10.2: Stochastic LM-OSL process using KMC

```
# Stochastic LM-OSL process using KMC
rm(list = ls(all=T))
library(matrixStats)
mcruns<-100
n0<-500
tmax<-100
A<-.1
nMatrix<-matrix(NA,nrow=tmax-1,ncol=mcruns)

times<-seq(1,tmax-1,1)
system.time(
for (k in 1:mcruns)
{n<-n0
 allt<-t<-1
    for(i in 1:n-1){
    P<-A*t/tmax
    t<-t+rexp(1)/(P*n)
    n<-n-1
```

```
    allt<- rbind(allt,t)
    }
 depth.class <- cut(allt,seq(1:tmax), include.lowest = TRUE)
 singlerun <- tapply(seq(from=n0,to=0,by=-1), depth.class,
mean,na.rm = FALSE)
 singlerun<-as.vector(singlerun)
 nMatrix[,k]<-singlerun
})
par(mfrow=c(1,3))
matplot(nMatrix,typ="l",col="green",ylim=c(0,700),
xlab="Time [s]",ylab="Remaining electrons")
avgn<-rowMeans(nMatrix,na.rm = TRUE)
cv<-100*rowSds(nMatrix,na.rm=TRUE)/avgn
# plots
pch<-c(NA,NA,1,NA)
lty<-c(NA,NA,NA,"solid")
col<-c(NA,NA,"black","red")
lines(seq(from=1,to=tmax-1,by=1),avgn,lwd=2,col="black")
curve(n0*exp(-A*x^2/(2*(tmax-1))),1,tmax,lwd=3,
col="red",add=TRUE)
legend("topright",bty="n",c("(a)  n(t) for LM-OSL",
" ","MC","Analytical"),pch=pch,lty=lty,col=col)
avgLM<-A*(times/(tmax-1))*as.numeric(avgn)
plot(times,avgLM,ylab="LMOSL",xlab="Time [s]",ylim=c(0,15))
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="red",lwd=2)

plot(times,cv,ylab="CV[%]",xlab="Time [s]",ylim=c(0,120))
curve(100*sqrt((exp(A*x^2/(2*tmax))-1)/n0),0,tmax,add=TRUE,
col="red",lwd=2)
legend("topleft",bty="n",c("(c)  CV[%]"," ","MC (M=100)",
 "Analytical"), pch=pch,lty=lty,col=col)
```

```
##    user  system elapsed
##    0.39    0.00    0.39
```

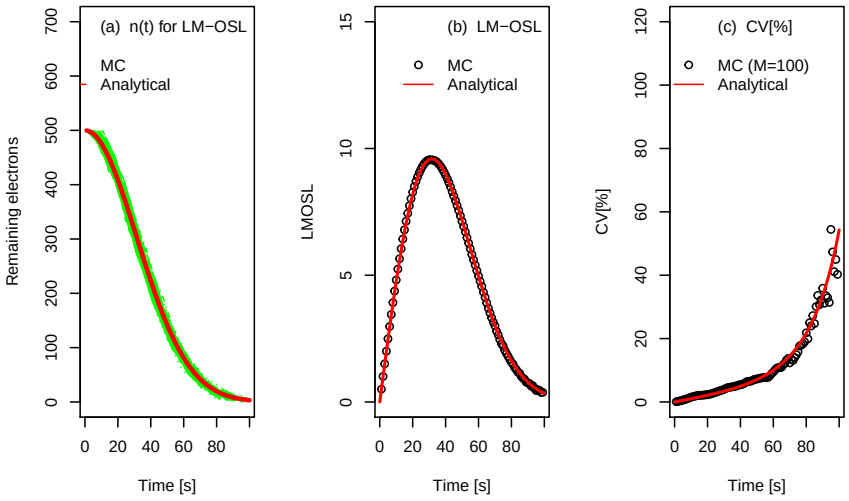


Fig. 10.2: Simulation of stochastic LM-OSL process using KMC method. The parameters are n_0 =500 trapped electrons at time $t = 0$, maximum stimulation time $P = 100$ s, LM-OSL stimulation rate $A = 0.1$ s^{-1}, and $M = 100$ MC runs. (a) Plot of the $M = 100$ MC runs, (b) Plot of the average MC $< n(t) >$, (c) Plot of the coefficient of variation *CV[%]*. The solid lines in these graphs represent the analytical solutions for the LM-OSL deterministic process.

Code 10.3: Stochastic first order TL process using KMC

```
# Stochastic first order TL process using KMC
rm(list = ls(all=T))
library(matrixStats)
mcruns<-100
n0<-500
tmax<-110
s<-1e13
E<-1
kb<-8.617e-5

times<-seq(1,tmax-1,1)
nMatrix<-matrix(NA,nrow=tmax-1,ncol=mcruns)
```

```
system.time(
for (k in 1:mcruns)
{n<-n0
 allt<-t<-30
    for(i in 1:n-1){
    P<-s*exp(-E/(8.617e-5*(t+273)))
    t<-t+rexp(1)/(P*n)
    n<-n-1
    allt<- rbind(allt,t)
    }
 depth.class <- cut(allt,seq(1:tmax), include.lowest = TRUE)
 singlerun <- tapply(seq(from=n0,to=0,by=-1), depth.class,
mean,na.rm = FALSE)
 singlerun<-as.vector(singlerun)
 nMatrix[,k]<-singlerun
})
# Calculate analytical solution
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)
# plot MC and analytical
par(mfrow=c(1,3))
matplot(nMatrix,typ="l",col="green",xlim=c(30,100),
ylab="Remaining electrons",
ylim=c(0,700),xlab=expression("Temperature ["^"o"*"C]"))
avgn<-rowMeans(nMatrix,na.rm = TRUE)
cv<-100*rowSds(nMatrix,na.rm=TRUE)/avgn
pch<-c(NA,NA,1,NA)
lty<-c(NA,NA,NA,"solid")
col<-c(NA,NA,"black","red")
lines(seq(from=1,to=tmax-1,by=1),avgn,lwd=2,col="black",
typ="p",pch=1)
legend("topright",bty="n",c("(a)  n(t)"," ","MC",
 "Analyt."),pch=pch,lty=lty,col=col)
lines(x-273,nanalyt,col="red",lwd=2)
k<-function(u){s*exp(-E/(kb*(u+273)))}
y<-lapply(times,k)
TLMC<-as.numeric(y)*as.numeric(avgn)
```

```
plot(times,TLMC,col="black",ylab="TL",
ylim=c(0,23),xlab=expression("Temperature ["^"o"*"C]"))
lines(x-273,TLanalyt,col="red",lwd=2)
legend("topleft",bty="n",c("(b)  TL"," ","MC",
  "Analyt."), pch=pch,lty=lty,col=col)
plot(times,cv,ylab="CV[%]",ylim=c(0,150),col="blue",
     xlab=expression("Temperature ["^"o"*"C]"))
legend("topleft",bty="n",c("(c)  CV[%]"," ","MC"))

  ##    user  system elapsed
  ##    0.41    0.00    0.40
```

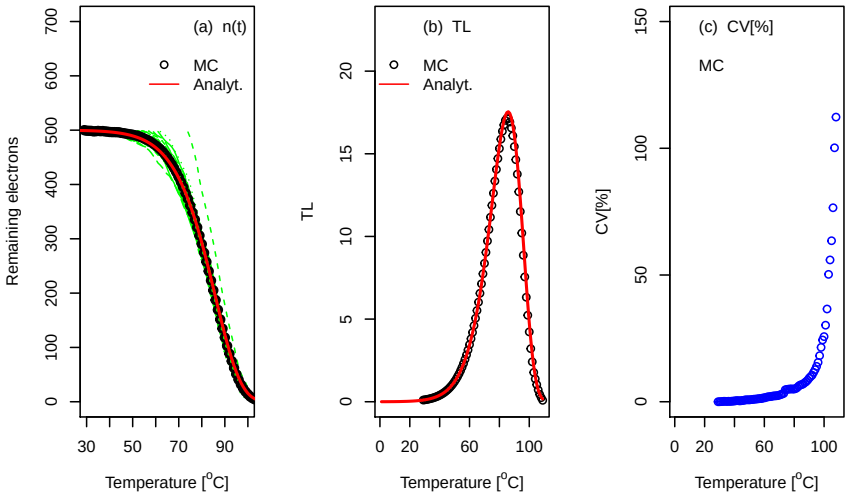


Fig. 10.3: Simulation of stochastic first order TL process using a KMC method. The parameters are n_0 =500 trapped electrons at time $t = 0$, maximum temperature $T = 110$°C, frequency factor $s = 10^{13}$ s^{-1}, energy $E = 1$ eV, heating rate $\beta = 1$ K/s. (a) Plot of the $M = 100$ MC runs, (b) Plot of the average MC $< n(t) >$, (c) Plot of the coefficient of variation CV[%]. The solid lines represent the analytical solutions for the TL deterministic process.

Code 10.4: Microscopic description of quantum tunneling

```
# Original Mathematica program by Vasilis Pagonis
# R version written by Johannes Friedrich, 2018
# The code reproduces Fig 2 of Pagonis and Kulp (2010)
rm(list = ls(all = TRUE)) # empties the environment
library("plot3D")
library("FNN")
## Define Parameters ----
sideX <- 200e-9 # lenght of quader in m
sideX_nm <- sideX*1e9 # length of quader in nm
s_tun <- 3e15
alpha <- 4e9
N_pts <- 100
clusters <- 50
rho_prime <- 1e-5

N_centers <- as.integer(rho_prime * sideX^3 *3 *alpha^3/(4*pi))
rho <- N_centers/ sideX^3
all_times_matrix <- matrix(NA,nrow = N_pts,ncol = clusters)
## Run MC -----
for(j in 1:clusters) {
  if(j %% 100 == 0) print(j)
  n_pts <- N_pts
  n_centers <- N_centers
  xyz_traps <- data.frame(
    x = sample(1:sideX_nm, N_pts, replace = TRUE),
    y = sample(1:sideX_nm, N_pts, replace = TRUE),
    z = sample(1:sideX_nm, N_pts, replace = TRUE)
  )
  xyz_centers <- data.frame(
    x = sample(1:sideX_nm, n_centers, replace = TRUE),
    y = sample(1:sideX_nm, n_centers, replace = TRUE),
    z = sample(1:sideX_nm, n_centers, replace = TRUE)
  )
    ### r calc distances ------
  for(i in 1:(n_pts)){
    ## find next neighbours with package FNN
    dist <- FNN::get.knnx(data = as.matrix(xyz_centers),
                          query = as.matrix(xyz_traps),
                          k = 1)
    all_dist <- as.data.frame(dist)
    P <- runif(n = length(all_dist$nn.dist), min = 0, max = 1)
    # P <- runif(n = 1, min = 0, max = 1)
    recomb_time <- - s_tun^(-1) * exp(alpha * all_dist$nn.dist *
    1e-9) * log(1-P)
```

```
    e_remove <- which.min(recomb_time)
    h_remove <- all_dist$nn.index[e_remove]
    ##remove index from data.frame
    xyz_centers <- xyz_centers[-h_remove,]
    xyz_traps <- xyz_traps[-e_remove,]
    all_times_matrix[i,j] <- recomb_time[e_remove]
  } # end n_pts loop
  all_times_matrix[,j] <- cumsum(all_times_matrix[,j])
} ## end cluster-loop
all_times_matrix <- log10(all_times_matrix)
### plot results ------
times_avg <- rowMeans(all_times_matrix)
matplot(x = all_times_matrix,
        y = (N_pts-1):0,xlim = c(-10,20), col = "grey",
        ylab = "remaining electrons",
        xlab = "log10(times)",pch = 1)
points(
  x = times_avg,  y = (N_pts-1):0, col = "blue")
sd <- apply(all_times_matrix, 1, sd)
sd_error <- sd/sqrt(N_pts)
## plot error bars
arrows(times_avg-sd_error,
       (N_pts-1):0,
       times_avg+sd_error,
       length=0.05,angle=90, code=3)
t <- 10^seq(-15,20,1)
lines(
  x = log10(t),
  y = N_pts * exp(-rho_prime * log(1.8 * s_tun * t)^3),
col = "red",lwd=3)
legend("topright",bty="n",
        legend = c("500 MC runs", "MC average",
        "analytical solution"),
        col = c("grey", "blue", "red"),
        pch = c(1,1,NA),lwd = c(NA,NA,2))
```

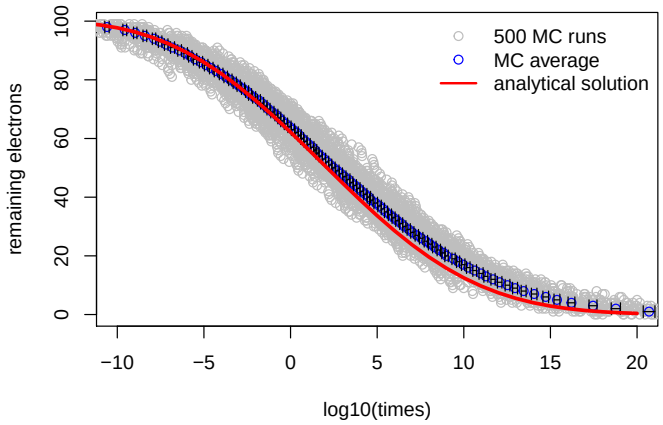


Fig. 10.4: Simulation of microscopic description of ground state quantum tunneling in luminescent materials with the parameters given in the text. The gray area indicates the results from M =500 simulations of the same system, and the solid circles indicate the average of the 500 runs. The standard error of the 500 runs is about equal to the drawing size of individual circles. The solid line represents the analytical Eq.(**??**). For more details see Pagonis and Kulp [42].

Part IV
COMPREHENSIVE LUMINESCENCE MODELS

Chapter 11
COMPREHENSIVE LUMINESCENCE MODELS FOR QUARTZ

Abstract This chapter presents several empirical comprehensive models which were developed in order to explain complex luminescence mechanisms and phenomena in quartz. We show detailed R codes for the *Baiiley2001* and the *Pagonis2008* quartz models, by using the R programs *KMS* developed by Peng and Pagonis, and also using the R package *RlumModel* by Friedrich et al. We show how to simulate the history of natural quartz samples, and how to modify the models by changing one or more of the original model parameters. R codes are provided for studying the phenomena of thermal quenching, for simulating the dose response of TL and OSL signals and also for the superlinear dose response in quartz. We show how to simulate the important phenomena of phototransfer, predose effect and thermal transfer of holes, pulse annealing and the SAR protocol in quartz. This chapter concludes with several examples from the extensive R package *RLumModel.*

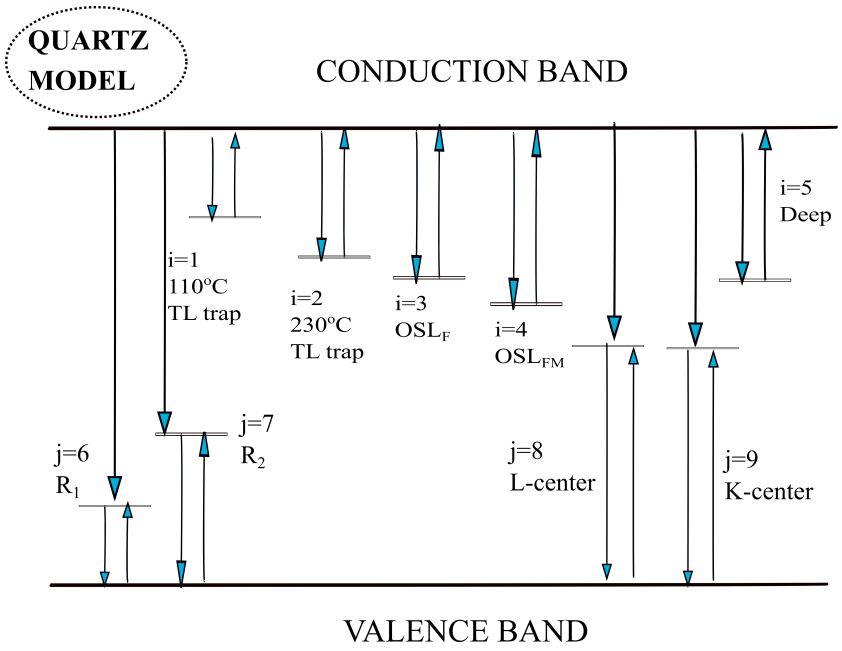


Fig. 11.1: Schematic diagram of the Bailey model, consisting of a total of nine energy levels. The arrows indicate possible transitions. (After Bailey [1]).

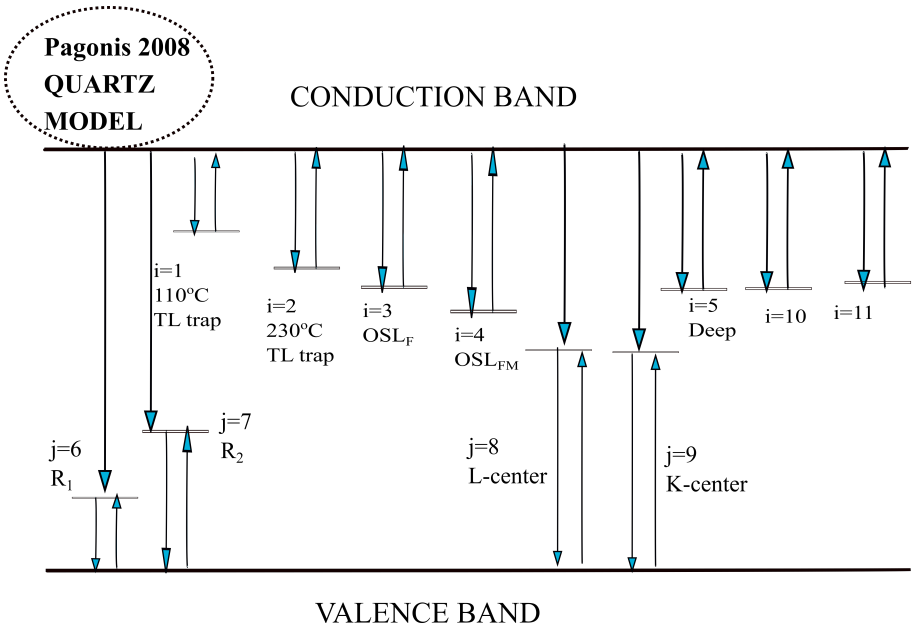


Fig. 11.2: The quartz model of Pagonis et al. [46]. Levels 10 and 11 are the additional levels introduced to the original model by Bailey [1].

Table 11.1: The various functions available in the R programs *KMS*, described in Peng and Pagonis [47].

FUNCTIONS CALLED IN KMS PROGRAMS
setInis() Set all center populations equal to zero for quartz crystallization. **setPars()** Initialize the model using appropriate kinetic parameters. **irradiate(*temp*, *tim*, *doseRate*)** Irradiate at *temp* ^{o}C for *tim* s with a dose rate of *doseRate* Gy/s. **heatAt(*temp*, *tim*)** Heat at *temp* ^{o}C for *tim* s. **heatTo(*temp1*, *temp2*, *hRate*)** Heat from *temp1* ^{o}C to *temp2* ^{o}C with a heating rate of *hRate* ^{o}C/s. **stimOSL(*temp*, *tim*, *pValue*, *nChannel*)** OSL stimulation at temp ^{o}C for *tim* s with a photon stimulation flux of $pValue\ s^{-1}cm^{-2}$ The OSL signal is evaluated at the equally spaced number of channels *nChannel*. **stimTL(*lowTemp*, *upTemp*, *hRate*, *nChannel*)** TL stimulation from *lowTemp* ^{o}C to *upTemp* ^{o}C with a heating rate of *hRate* ^{o}C/s, the number of equally spaced channels is *nChannel*

Table 11.2: The steps in simulating a *natural* sedimentary quartz sample, in the *Bailey2001* model in *KMS* [1]. Step 5a is used in the original *Bailey2001* model, step 5b is used in the *Pagonis2008* model in KMS.

	Steps in simulation of *natural* sedimentary quartz sample in the Bailey2001 model [1]
1	All electron and hole concentrations set to zero during the crystallization process.
2	Geological dose of $1000Gy$ with a dose rate of $1Gy/s$ at 20°C.
3	Heat to $350°C$ (simulation of geological time).
4	Illumination at 200°C for $100s$, repeated exposures to sunlight over a long period.
5a	Burial dose of 20 Gy at 0.01 Gy/s at 220°C.
5b	Burial dose of 20 Gy at a very low natural dose rate of 3×10^{-11} Gy/s at 20°C.

Code 11.1: Natural history of quartz sample Pagonis2008 model

```
# Use Pagonis model to simulate quartz sample history and
# TL measured in lab after irradiating with 10 Gy at 1 Gy/s
rm(list=ls())
library("deSolve")
source("Pagonis_Model.R")
setPars()
setInis()
irradiate(temp=20, tim=1000, doseRate=1) #1000 Gy at 1 Gy/s
heatAt(temp=20, tim=60)                  #Relaxation
heatTo(temp1=20, temp2=350, hRate=5)     # Heat to 350 degC
heatTo(temp1=350, temp2=200, hRate=-5)   # Cool down to RT
stimOSL(temp=200, tim=100, pValue=2.0, nChannel=1000) #Bleach
irradiate(temp=20, tim=20/1e-11, doseRate=1e-11)# Burial dose
heatAt(20,60)
storeNat<-inis #Store concentrations for natural sample
TL<-stimTL(lowTemp=20,upTemp=500,hRate=5,nChannel=480) #TL
heatTo(temp1=500,temp2=20,hRate=-5) # Cool down
tlx<-TL[,"tlx"]
tly<-TL[,"tly"]
par(mfrow=c(1,2))
plot(tlx,tly,type="l",ylim=c(0,16000),lwd=2,
xlab=expression("Temperature "^"o"*"C]"), ylab = "TL [a.u.]")
legend("topleft",bty="n",legend=c("(a)","Pagonis model",
"Natural TL"))
# repeat irradiation and TL for lab irradiation
inis<-storeNat   # restore concentrations for natural sample
```

```
irradiate(temp=20, tim=10, doseRate=1)
heatAt(temp=20, tim=60)
TL<-stimTL(lowTemp=20,upTemp=500,hRate=5,nChannel=480)
heatTo(temp1=500,temp2=20,hRate=-5)
tlx<-TL[,"tlx"]
tly<-TL[,"tly"]
plot(tlx,tly,type="l",xlab=expression("Temperature ["^"o"*"C]"),
ylab = "TL [a.u.]",ylim=c(0,400000),lwd=2)
legend("topright",bty="n",legend=c("(b)", " ","Regenerated TL"))
```

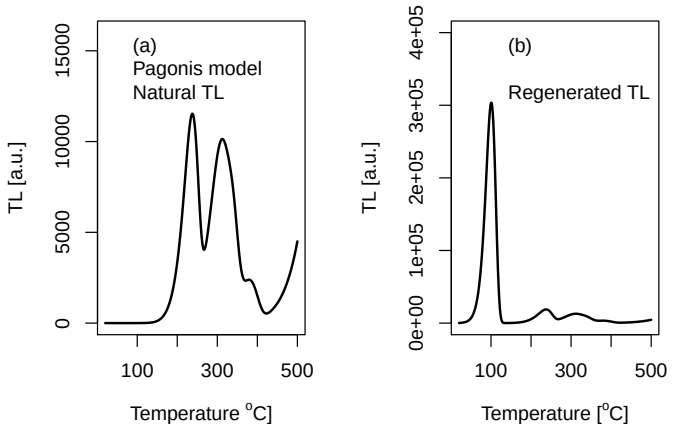


Fig. 11.3: Simulation of the natural history of a quartz sample, using the *Pagonis2008* model in *KMS* programs. (a) The TL glow curve, after the natural sample is heated in the laboratory with a heating rate of 5 K/s. Note the absence of the 110°C TL peak. (b) The TL glow curve, after the natural sample is irradiated with 10 Gy and then heated in the laboratory with a heating rate of 5 K/s. Note the restored 110°C TL peak in (b), which is missing in (a).

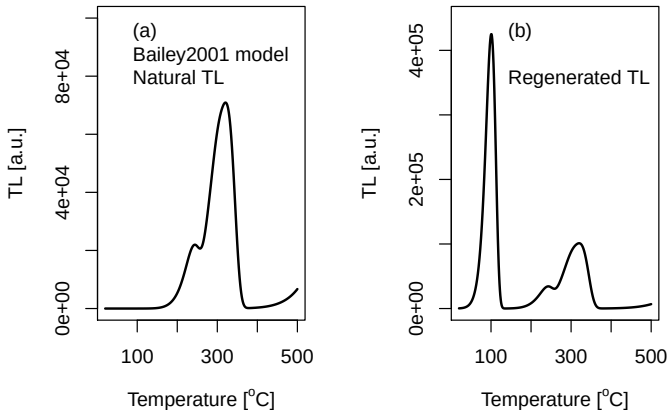


Fig. 11.4: Simulation of the natural history of a quartz sample, using the *Bailey2001* model in *KMS*. Compare these results with the similar results in Fig.11.3, which were obtained using the *Pagonis2008* model.

Code 11.2: Thermal quenching of TL signal in quartz (KMS)

```
# Use Pagonis model to simulate thermal quenching in quartz
# TL measured in laboratory with different heating rates
rm(list=ls())
library("deSolve")
source("Pagonis_Model.R")
beta<-2*seq(1:6)
tlylow<-tlyhigh<-vector(length = length(beta))
tlx<-tly<-matrix(nrow=480,ncol=length(beta))
for (i in 1:6){
  setPars()
  setInis()
  irradiate(temp=20, tim=1000, doseRate=1) #1000 Gy at 1 Gy/s
  heatAt(temp=20, tim=60)                  #Relaxation
  heatTo(temp1=20, temp2=350, hRate=5)     # Heat to 350 degC
  heatTo(temp1=350, temp2=200, hRate=-5)   # Cool down to RT
  stimOSL(temp=200, tim=100, pValue=2.0, nChannel=1000) #Bleach
  irradiate(temp=20, tim=20/1e-11, doseRate=1e-11)# Burial dose
  heatAt(20,60)
```

```
 irradiate(temp=20, tim=100, doseRate=1)
heatAt(temp=20, tim=60)
TL<-stimTL(lowTemp=20,upTemp=500,hRate=beta[i],nChannel=480)
tlx[,i]<-TL[,"tlx"]
tly[,i]<-TL[,"tly"]/i
tlyhigh[i]<-sum(tly[,i][130:400])
tlylow[i]<-sum(tly[,i][1:130])}
par(mfrow=c(1,2))
matplot(tlx,tly,typ="l",lty="solid",lwd=2,
xlab=expression("Temperature ["^"o"*"C]"), ylab = "TL [a.u.]",
ylim=c(0,350000),xlim=c(30,380))
legend("topleft",bty="n",legend=c("(a)","Heating 2-12 K/s",
"Thermal quenching"))
plot(beta,tlylow/max(tlylow),pch=1,typ="o",ylim=c(0,1.2),
xlab="Heating rate [K/s]", ylab = "Normalized TL Area")
lines(beta,colSums(tly)/max(colSums(tly)),pch=2,typ="o")
lines(beta,tlyhigh/max(tlyhigh),pch=3,typ="o")
legend("bottomleft",bty="n",legend=expression("(b)",
"Area 0-130"^"o"*"C","Area 0-500"^"o"*"C","Area
130-500"^"o"*"C"),pch=c(NA,1,2,3))
```

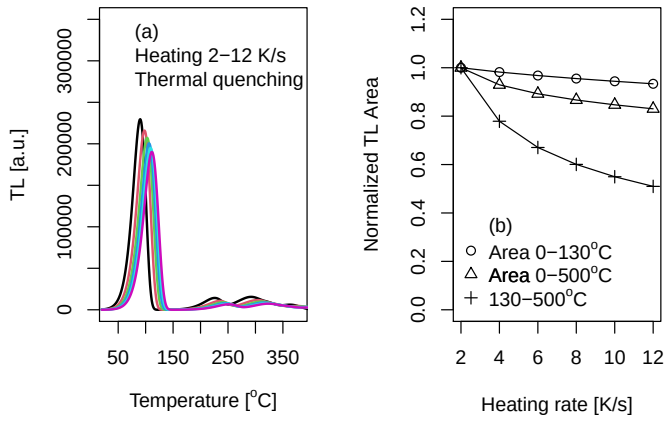


Fig. 11.5: Simulation of the thermal quenching of TL signal in quartz, demonstrated by using various heating rates in the *Pagonis2008* model of *KMS*. (a) The TL glow curves for heating rates 2-12 K/s, and (b) The areas under different parts of the TL glow curves in (a), showing the thermal quenching effects.

Code 11.3: TL dose response of quartz sample (KMS)

```
# Dose Response of 110degC TL peak in quartz
rm(list=ls())
library("deSolve")
library("minpack.lm")
library("lamW")
source("Bailey01_Model.R")
setPars()
setInis()
irradiate(temp=20, tim=1000, doseRate=1) #1000 Gy at 1 Gy/s
heatAt(temp=20, tim=60)                  #Relaxation
heatTo(temp1=20, temp2=350, hRate=5)     # Heat to 350 degC
heatTo(temp1=350, temp2=200, hRate=-5)   # Cool down to RT
stimOSL(temp=200, tim=100, pValue=2.0, nChannel=1000) #Bleach
irradiate(temp=20, tim=20/1e-11, doseRate=1e-11)# Burial dose
heatAt(20,60)    #Store concentrations for natural sample
storeNat<-inis  # in variable storeNat
reDose<-c(1,3,5,10,20,60,100,200,300,400,500,700)
tlm<-vector(length=length(reDose))
tlx<-tly<-matrix(nrow=480,ncol=length(reDose))
for (i in seq(length(reDose))) {
inis<-storeNat
irradiate(temp=20,tim=reDose[i],doseRate=1)
heatAt(temp=20,tim=60)
res<-stimTL(lowTemp=20,upTemp=500,hRate=5,nChannel=480)
heatTo(temp1=500,temp2=20,hRate=-5)
tlx[,i]<-res[,"tlx"]
tly[,i]<-res[,"tly"]
tlm[i]<-approx(x=res[,"tlx"],y=res[,"tly"],xout=330)$y }
par(mfrow=c(1,2))
matplot(tlx,tly,type="l",lty="solid",ylab = "TL [a.u.]",
xlab=expression("Temperature "^"o"*"C]"),ylim=c(0,1.4e6),
xlim=c(30,380))
legend("topright",bty="n",legend=c("(a)"," ","TL glow curves"))
plot(reDose,tlm,type="p",xlab="Dose [Gy]",ylab="TL [a.u]",
ylim=c(0,.95e6))
legend("topright",bty="n",legend=c("(b)","Dose response",
"Lambert fit"),pch=c(NA,1),lty=c(NA,NA,"solid"))
t <-reDose
y <-tlm
fit_data <-data.frame(  t ,y)
#plot(fit_data,ylim=c(0,max(y)))
```

```
fit <- minpack.lm::nlsLM(
formula=y~N*(1+lambertW0((abs(R)-1)*exp(abs(R)-1-b*t))/
(1-abs(R))),
data=fit_data,start = list(N=5* max(y),R=0.1, b = .002))
N_fit <- coef(fit)[1]
R_fit <- abs(coef(fit)[2])
b_fit <- coef(fit)[3]
## plot analytical solution
curve(
N_fit*(1+lambertW0((R_fit-1)*exp(R_fit-1-b_fit*x))/(1-R_fit)),
0,700,col = "blue",add=TRUE,lwd=2)
cat("\nfitted N: ", N_fit)
cat("\nfitted R: ", R_fit)
cat("\nfitted D0: ", 1/b_fit, " Gy")

  ##
  ## fitted N:  768919.4
  ## fitted R:  4.819477e-08
  ## fitted D0:  1045.857  Gy
```

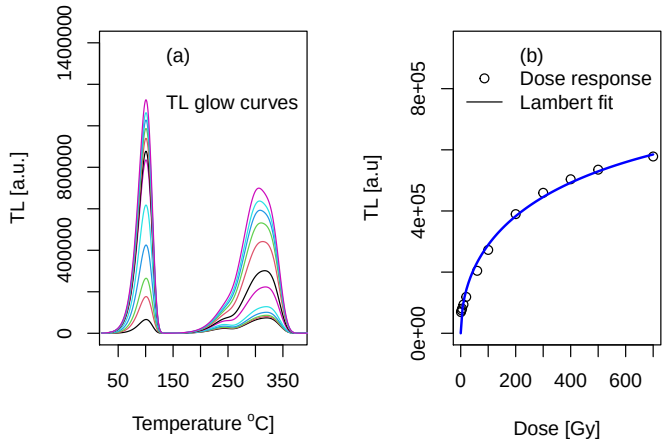


Fig. 11.6: Simulation of the dose response curve of the 330°C TL peak in a sedimentary quartz sample. (a) The TL glow curves (b) The dose response curve, fitted with the Lambert dose response equation we saw in Chapter 4.

Code 11.4: Dose Response of quartz OSL signal, Bailey2001 model

```
# Dose Response of quartz OSL signal measured at 125degC
rm(list=ls())
library("deSolve")
library("minpack.lm")
library("lamW")
source("Bailey01_Model.R")
setPars()
setInis()
irradiate(temp=20, tim=1000, doseRate=1) #1000 Gy at 1 Gy/s
heatAt(temp=20, tim=60)                  #Relaxation
heatTo(temp1=20, temp2=350, hRate=5)     # Heat to 350 degC
heatTo(temp1=350, temp2=200, hRate=-5)   # Cool down to RT
stimOSL(temp=200, tim=100, pValue=2.0, nChannel=1000) #Bleach
irradiate(temp=20, tim=20/1e-11, doseRate=1e-11)# Burial dose
heatAt(20,60)    #Store concentrations for natural sample
storeNat<-inis  # in variable storeNat
reDose<-c(1,50,100,200,300,400,500,700)
oslm<-vector(length=length(reDose))
oslx<-osly<-matrix(nrow=100,ncol=length(reDose))
for (i in seq(length(reDose))) {
  inis<-storeNat #Restore natural concentrations
  irradiate(temp=20,tim=reDose[i],doseRate=1)
  heatAt(temp=20,tim=60)
  heatTo(20,125,hRate=5)
  res<-stimOSL(temp=125, tim=100, pValue=2.0, nChannel=100)
  oslx[,i]<-res[,"oslx"]
  osly[,i]<-res[,"osly"]
  oslm[i]<-res[,"osly"][1]}
par(mfrow=c(1,2))
matplot(oslx,osly,type="l",lty="solid",xlim=c(0,20),
xlab="Time [s]",ylab = "OSL [a.u.]",ylim=c(0,1.6e8))
legend("topright",bty="n",legend=c("(a)"," ","OSL curves"))
plot(reDose,oslm,type="p",xlab="Dose [Gy]",ylab="OSL [a.u]",
ylim=c(0,2.6e8))
legend("topright",bty="n",legend=c("(b)","OSL response",
"Lambert fit"),pch=c(NA,1),lty=c(NA,NA,"solid"))
t <-reDose
y <-oslm
fit_data <-data.frame(  t ,y)
fit <- minpack.lm::nlsLM(
  formula=y~N*(1+lambertW0((abs(R)-1)*exp(abs(R)-1-b*t))/
                 (1-abs(R))),  data = fit_data,
  start = list(N= max(y),R=0.1, b = .002))
N_fit <- coef(fit)[1]
```

```
R_fit <- abs(coef(fit)[2])
b_fit <- coef(fit)[3]
curve(
  N_fit*(1+lambertW0((R_fit-1)*exp(R_fit-1-b_fit*x))/(1-R_fit)),
  1,700,add=TRUE )
cat("\nfitted N: ", N_fit)
cat("\nfitted R: ", R_fit)
cat("\nfitted Dc: ", 1/b_fit, " Gy")

  ##
  ## fitted N:  200028153
  ## fitted R:  0.2111977
  ## fitted Dc:  504.679  Gy
```

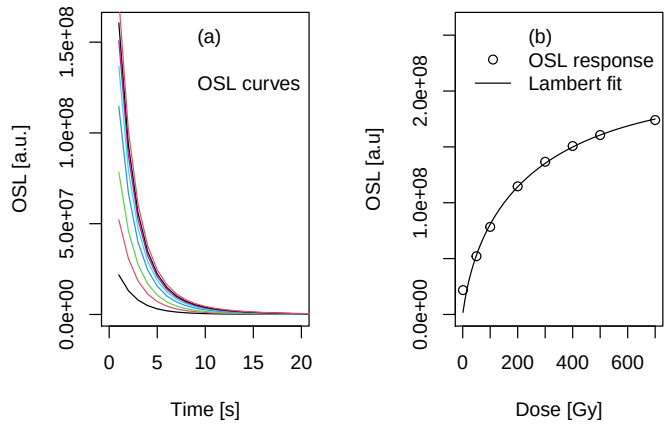


Fig. 11.7: Simulation of the dose response curve of the OSL signal measured at 125°C, in the *Bailey2001* model. (a) The CW-OSL curves and (b) The dose response curve, fitted with the Lambert function we saw in Chapter 4.

Code 11.5: Superlinearity in annealed quartz samples

```
# Superlinear TL dose response in annealed quartz
rm(list = ls(all=T))
library("deSolve")
```

```
source("Bailey01_Model.R")
reDose<-seq(0.1,2,by=1.9/7)
setInis()
setPars()
irradiate(temp=20, tim=1000, doseRate=1) #1000 Gy at 1 Gy/s
heatAt(temp=20, tim=60)                  #Relaxation
heatTo(temp1=20, temp2=350, hRate=5)     # Heat to 350 degC
heatTo(temp1=350, temp2=200, hRate=-5)   # Cool down to RT
stimOSL(temp=200, tim=100, pValue=2.0, nChannel=1000) #Bleach
irradiate(temp=20, tim=20/1e-11, doseRate=1e-11)#Burial 100 Gy
heatTo(20,700,hRate=5)
heatAt(700,3600)
heatTo(700,20,hRate=-5)
storeNat<-inis
tlm<-vector(length=8)
tlx<-tly<-matrix(nrow=480,ncol=8)
# natural dose rate =0.1 Gy
for (i in seq(8)) {
  inis<-storeNat
  irradiate(temp=20,tim=reDose[i],doseRate=1)
  heatAt(temp=20,tim=60)
  res<-stimTL(lowTemp=20,upTemp=500,hRate=5,nChannel=480)
  tlx[,i]<-res[,"tlx"]
  tly[,i]<-res[,"tly"]
  tlm[i]<-approx(x=res[,"tlx"],y=res[,"tly"],xout=330)$y }
par(mfrow=c(1,3))
matplot(tlx,tly,type="l",ylim=c(0,470),xlim=c(20,380),
lty="solid",lwd=2,xlab=expression("Temperature ["^"o"*"C]"),
ylab = "TL [a.u.]")
legend("topright",bty="n",legend=c("(a)","TL glow curves",
"Annealed quartz"))
plot(reDose,tlm,type="o",xlab = "Dose [Gy]",ylab = "TL [a.u.]",
ylim=c(0,1.4*max(tlm)),xlim=c(0,2.5))
legend("topleft",bty="n",legend=c("(b)"," ","Superlinear",
"Dose response"))
x<- log(reDose)
y<- log(tlm)
rangeData<-cbind(x,y)
lm(y~x)$coefficients
plot(x, y, xlab = "ln(Dose)",ylab = "ln(TL)",ylim=c(0,7),
xlim=c(-2,1.5))
legend("topleft",bty="n",legend=c("(c)"," ","Log-Log",
"slope=1.39"))
abline(lm(y~x))
```

```
## (Intercept)           x
##    2.481778    1.397661
```

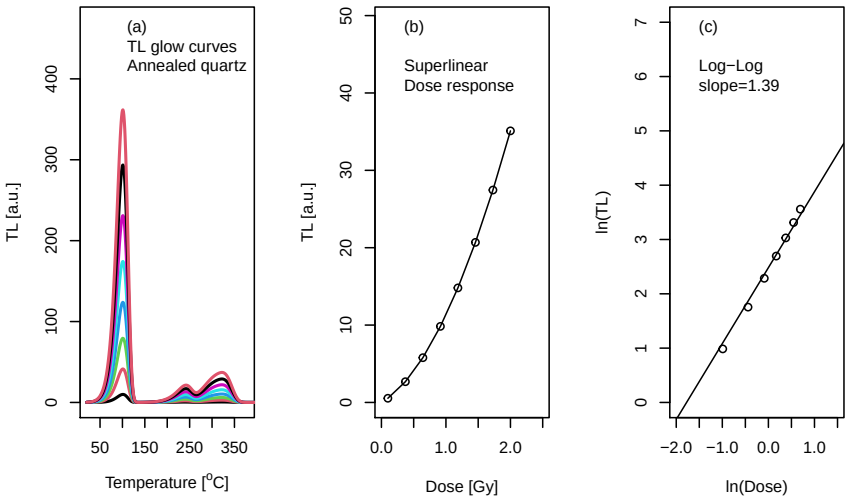


Fig. 11.8: Simulated superlinear dose response of 330°C TL peak in annealed quartz using the *Bailey2001* model. The sample was annealed for 1 h at 700°C, before measuring its TL dose response between 0.1 and 2 Gy. (a) The TL glow curves at different doses; (b) The superlinear dose dependence of the TL signal at 330°C; (c) The data in (b) is plotted on a log-log scale and fitted with a linear function, yielding a superlinearity index of $g(D)$=1.39.

Code 11.6: Phototransfer phenomenon using Bailey2001 model

```
#Phototransfer simulation using Bailey2001 model
rm(list = ls(all=T))
library("deSolve")
source("Bailey01_Model.R")
photo<-function(x,colr,ln,pchs){
  setInis()
  setPars()
  pars["N6"] <- 3e10 # change the parameter N6 in Bailey model
  # N6=total concentration of holes in hole reservoir R1
  irradiate(temp=20, tim=1000, doseRate=1) #1000 Gy at 1 Gy/s
```

```
  heatAt(temp=20, tim=60)                    #Relaxation
  heatTo(temp1=20, temp2=350, hRate=5)       # Heat to 350 degC
  heatTo(temp1=350, temp2=200, hRate=-5)     # Cool down to RT
  stimOSL(temp=200, tim=100, pValue=2.0, nChannel=1000) #Bleach
  # Large Burial dose 100 Gy
  irradiate(temp=20, tim=200/1e-11, doseRate=1e-11)
  # end natural sample simulation, store concentrations
# Next is the optical Bleach
stimOSL(temp=20, tim=x, pValue=2.0, nChannel=1000)
heatAt(temp=20, tim=60)
TL<-stimTL(lowTemp=20,upTemp=500,hRate=5,nChannel=150) #TL
heatTo(temp1=500,temp2=20,hRate=-5) # Cool down
tlx<-TL[,"tlx"]
tly<-TL[,"tly"]
par(new=TRUE)
plot(tlx,tly,type="o",xlab=expression("Temperature ["^"o"*"C]"),
 ylab = "TL [a.u.]",xlim=c(0,400),ylim=c(0,7e5),col=colr,lty=ln,
lwd=1,pch=pchs)
}   #end function photo
for (i in 1:3)
  {bleachTime<-c(1,3,6)
   colr<-c("red","blue","black")
   ln<-rep("solid",3)
   pchs<-c(1,2,4)
   photo(bleachTime[i],colr[i],ln[i],pchs[i])}
legend("topleft",bty="n","Phototransfer")
legend("topright",bty="n",lty=c(NA,ln),col=c(NA,colr),lwd=1,
 pch=c(NA,pchs), c("Bleaching times","1 s","3 s","6 s"))
```

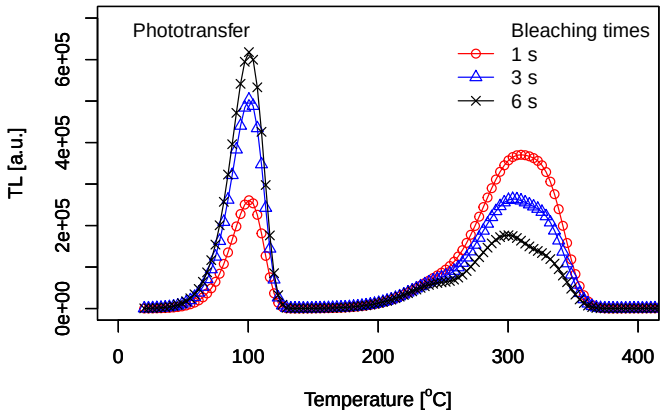
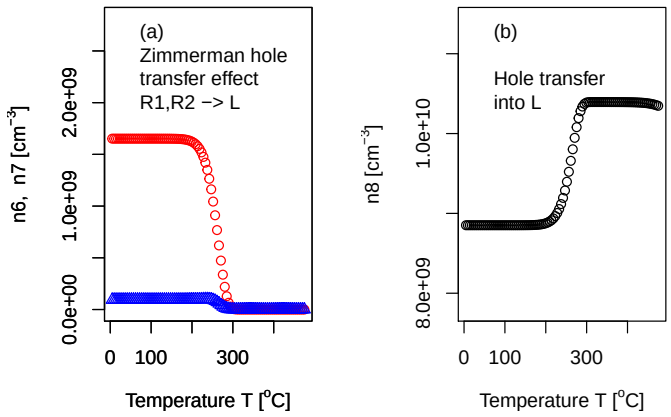


Fig. 11.9: Simulation of phototransfer process in the *Bailey2001* model. As the optical stimulation time increases, the height of the TL peak at 110°C increases, while simultaneously the 330°C TL peak decreases. For a more detailed study of this, and other quartz phenomena, see Pagonis et al. [33].

Code 11.7: The Zimmerman model: thermal transfer of holes in quartz

```
# The Zimmerman thermal transfer of holes in quartz
rm(list=ls())
library("deSolve")
source("Bailey01_Model.R")
setInis()
setPars()
pars["N6"] <- 3e10   # change the concentration of holes in R1
irradiate(temp=20, tim=1000, doseRate=1) #1000 Gy at 1 Gy/s
heatAt(temp=20, tim=60)                  #Relaxation
heatTo(temp1=20, temp2=350, hRate=5)     # Heat to 350 degC
heatTo(temp1=350, temp2=200, hRate=-5)   # Cool down to RT
stimOSL(temp=200, tim=100, pValue=2.0, nChannel=1000) #Bleach
irradiate(temp=20, tim=100/1e-11, doseRate=1e-11)#Burial 100 Gy
storeNat<-inis
# Set up parameters and vectors
testDose<-.1
```

```
nTemps<-15
initTemp<-100
finalTemp<-500
actTemp<-seq(initTemp,finalTemp,by=(finalTemp-initTemp)/
               (nTemps-1))
# Loop for activation temperatures
inis<-storeNat   # Reset concentrations for natural sample
res2<-heatTo(25,500,hRate=1)    # heat to activation temperature
par(mfrow=c(1,2))
xlabel<-expression("Temperature T ["^"o"*"C]")
ylabel<-expression("n6,  n7 [cm"^"-3"*"]")
plot(res2[,"time"],res2[,"n6"],xlab = xlabel
,ylab =ylabel,ylim=c(0,1.7*max(res2[,"n6"])),pch=1,col="red")
legend("topleft",bty="n",legend=c("(a)","Zimmerman hole",
"transfer effect","R1,R2 -> L"))
par(new = TRUE)
plot(res2[,"time"],res2[,"n7"],xlab = xlabel
,ylab =ylabel,ylim=c(0,1.7*max(res2[,"n6"])),pch=2,col="blue")
plot(res2[,"time"],res2[,"n8"],xlab = xlabel
,ylab =expression("n8 [cm"^"-3"*"]"),
ylim=c(.75*max(res2[,"n8"]),1.1*max(res2[,"n8"])))
legend("topleft",bty="n",legend=c("(b)"," ","Hole transfer",
"into L"))
```



Code 11.8: Simulation of pulse annealing experiment with Bailey2001 model

Fig. 11.10: The Zimmerman thermal transfer process for holes in quartz. (a) The concentrations of holes n_6 and n_7 in hole reservoirs R_1 and R_2, decrease as the temperature T is increased. (b) The concentration of holes n_8 in recombination center L also increases, indicating a thermal transfer of holes.

```
#Simulation of pulse annealing experiment with
# Bailey2001 model
rm(list=ls())
library("deSolve")
source("Bailey01_Model.R")
setInis()
setPars ()
pars["N6"] <- 3e9
### Start natural sample simulations
irradiate(temp=20, tim=1000, doseRate=1)
heatAt(temp=20, tim=60)
heatTo(temp1=20, temp2=350, hRate=5)
heatTo(temp1=350, temp2=200, hRate=-5)
stimOSL(temp=200, tim=100, pValue=2.0, nChannel=10)
irradiate(temp=20, tim=51/1e-11, doseRate=1e-11)  #Burial 51 Gy
## End of natural history with natural irradiation
res<- heatAt(temp=20, tim=60)
storeNat<-inis
nTemps<-20
initTemp<-100
finalTemp<-350
actTemp<-seq(initTemp,finalTemp,by=(finalTemp-initTemp)/
(nTemps-1))
tlm<-vector(length=nTemps)
tlx<-tly<-matrix(nrow=1000,ncol=nTemps)
tl<-osl<-rep(0,nTemps)
par(mfrow=c(1,2))
for (i in seq(nTemps)) {
  heatTo(20,actTemp[i],hRate=1)
  heatAt(actTemp[i],tim=10)
  heatTo(actTemp[i],20,hRate=-1)
  heatTo(20,125,hRate=1)
  resOSL<-stimOSL(temp=125, tim=0.1, pValue=2.0, nChannel=10)
  osl[i]<-sum(resOSL[,"osly"])
  heatTo(125,20,hRate=-1)
  irradiate(temp=20,tim=0.1,doseRate=1)
  heatAt(20,tim=60)
```

```
  resTL<-stimTL(20,160,hRate=1,nChannel=110)
  tl[i]<-max(resTL[,"tly"])}
xlabel<-expression("Activation T ["^"o"*"C]")
plot(actTemp,osl/tl,xlab=xlabel,
ylim=c(0,200000),ylab="Sensitivity corrected OSL/TL")
legend("topright",bty="n",legend=c("(a)"," ","Pulse annealing",
"Natural sample"))
#
inis<-storeNat  # Restore concentrations of natural sample
heatTo(20,125,hRate=1)
stimOSL(temp=125, tim=200, pValue=2.0, nChannel=10)
heatTo(125,20,hRate=-1)
irradiate(temp=20,tim=56,doseRate=1)  #56 Gy in lab
for (i in seq(nTemps)) {
  heatTo(20,actTemp[i],hRate=1)
  heatAt(actTemp[i],tim=10)
  heatTo(actTemp[i],20,hRate=-1)
  heatTo(20,125,hRate=1)
  resOSL<-stimOSL(temp=125, tim=0.1, pValue=2.0, nChannel=10)
  osl[i]<-sum(resOSL[,"osly"])
  heatTo(125,20,hRate=-1)
  irradiate(temp=20,tim=0.1,doseRate=1)
  heatAt(20,tim=60)
  resTL<-stimTL(20,160,hRate=1,nChannel=110)
  tl[i]<-max(resTL[,"tly"])}
plot(actTemp,osl/tl,xlab=xlabel,ylim=c(0,320000),
ylab="Sensitivity corrected OSL/TL")
legend("topright",bty="n",legend=c("(b)"," ",
"Irradiated sample"))
```

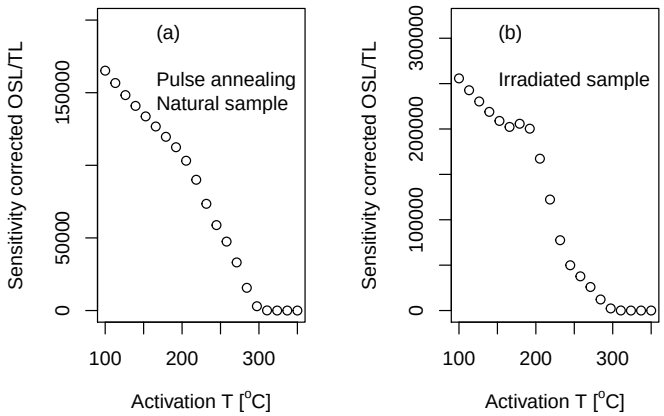


Fig. 11.11: Simulation of the experimental pulse annealing protocol of Wintle and Murray [54] for (a) A natural quartz sample, and (b) An optically bleached and irradiated sample. For details of the simulation, see Pagonis et al. [45].

Table 11.3: Steps in the simulation of the TT-OSL protocol of Wang et al. [52], based on the simulations of Pagonis et al. [46].

Step	Description
1	Geological dose - irradiation of $1000Gy$ at $1Gy/s$.
2	Geological time - heat to $350^{\circ}C$.
3	Illuminate for $100s$ at $200^{\circ}C$.
4	Burial dose - $200Gy$ at $220^{\circ}C$ at $0.01Gy/s$.
5	Regenerative dose D_i at $20^{\circ}C$ and at $1Gy/s$.
6	Preheat to 260°C for $10s$.
7	Blue stimulation at $125^{\circ}C$ for $270s$.
8	Preheat to $260^{\circ}C$ for $10s$.
9	Blue stimulation at $125^{\circ}C$ for $90s$ (L_{TT-OSL}).
10	Test Dose = $7.8Gy$
11	Preheat to $220^{\circ}C$ for $20s$
12	Blue stimulation at 125°C for $90s$ (T_{TT-OSL}).
13	Anneal to 300°C for $10s$
14	Blue stimulation at 125°C for $90s$.
15	Preheating at $260^{\circ}C$ for $10s$.
16	Blue stimulation at $125^{\circ}C$ for $90s$ (L_{BT-OSL}).
17	Test Dose = $7.8Gy$.
18	Preheat to 220^{o}C for $20s$.
19	Blue stimulation at $125^{0}C$ for $90s$ (T_{BT-OSL}).
20	Repeat 1-19 for different regenerative doses D_i=0-$4000Gy$ in step 5.

Code 11.9: SAR protocol using the Pagonis model in KMS

```
### TEMPLATE1 in Peng and Pagonis (2016)
# SAR protocol using the Pagonis model in KMS
library(deSolve)
source("Pagonis_Model.R")
setPars()
setInis()
irradiate(temp=20,tim=50000/3.17e-11,doseRate=3.17e-11)
heatAt(temp=20,tim=60)
stimOSL(temp=20,tim=6000,pValue=4.73e16,nChannel=6000)
nCycle<-2
for (i in seq(nCycle)) {
irradiate(temp=20,tim=10/3.17e-11,doseRate=3.17e-11)
heatAt(temp=20,tim=60)
stimOSL(temp=20,tim=6000,pValue=4.73e16,nChannel=6000)
} #end for.
```

```
irradiate(temp=20,tim=20/3.17e-11,doseRate=3.17e-11)
heatAt(temp=20,tim=60)
### TEMPLATE4 in Peng and Pagonis (2016)
reDose<-c(1e-13,8,16,24,32,1e-13,8)
LxTx<-sLxTx<-vector(length=7)
for (i in seq(7)) {
irradiate(temp=20,tim=reDose[i]/0.1,doseRate=0.1)
heatAt(temp=20,tim=60)
heatTo(temp1=20,temp2=260,hRate=5)
heatAt(temp=260,tim=10)
heatTo(temp1=260,temp2=125,hRate=-5)
Lxdat<-stimOSL(temp=125,tim=100,pValue=4.73e16,nChannel=1000)
heatTo(temp1=125,temp2=20,hRate=-5)
irradiate(temp=20,tim=1/0.1,doseRate=0.1)
heatAt(temp=20,tim=60)
heatTo(temp1=20,temp2=220,hRate=5)
heatAt(temp=220,tim=10)
heatTo(temp1=220,temp2=125,hRate=-5)
Txdat<-stimOSL(temp=125,tim=100,pValue=4.73e16,nChannel=1000)
heatTo(temp1=125,temp2=20,hRate=-5)
LxTx[i]<-sum(Lxdat[1:5,"osly"])/sum(Txdat[1:5,"osly"])
 } #end for.
plot(reDose,LxTx,pch=c(3,2,2,2,2,1,4),col=c(1,rep(2,4),3,4),
lwd=2)
```

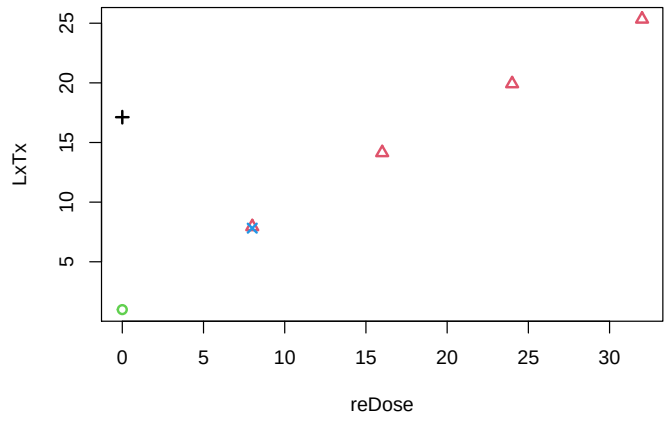


Fig. 11.12: Simulation of a sequence of thermal/optical events using the *Pagonis_Model* code in the KMS programs (Peng and Pagonis [47]).

Code 11.10: Sequence of thermal/optical events

```
# Sequence of thermal/optical events (RLumModel)
rm(list=ls())
suppressMessages(library(package = "RLumModel"))
sequence <- list(
 IRR = c(temp = 20, dose = 10, dose_rate = 1),
 TL = c(temp_begin = 20, temp_end = 400, heating_rate = 1))
model.output <- model_LuminescenceSignals(
model = "Pagonis2008",   sequence = sequence, main=" ",
verbose = FALSE)
```

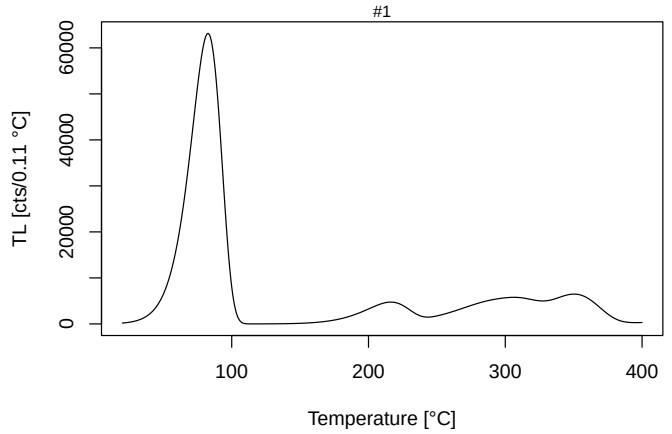


Fig. 11.13: Simulation of a sequence of thermal/optical events using the package *RLumModel*.

Code 11.11: Dose response of TL (RlumModel package)

```
#Dose response of TL (RlumModel package
rm(list=ls())
suppressMessages(library(package = "RLumModel"))
##set list with laboratory doses
```

```
Lab.dose <- seq(from = 2, to = 8, by = 2)
model.output <- lapply(Lab.dose, function(x){
  sequence <- list(
    IRR = c(20, x, 0.1),
    TL = c(20, 400, 1))
  TL_data <- model_LuminescenceSignals(
    sequence = sequence,
    model = "Bailey2001",
    plot = FALSE,
    verbose = FALSE)
  return(Luminescence::get_RLum(TL_data, recordType = "TL$",
drop = FALSE))
})
model.output.merged <- merge_RLum(model.output)
plot_RLum(
  object = model.output.merged,
  xlab = "Temperature [\u00B0C]",
  ylab = "TL signal [a.u.]",main=" ",lwd=3,lty=1:4,
  legend.text = paste(Lab.dose, " Gy"),
  combine = TRUE)
legend("top",bty="n",legend=c("TL dose response","RLumModel"))
```

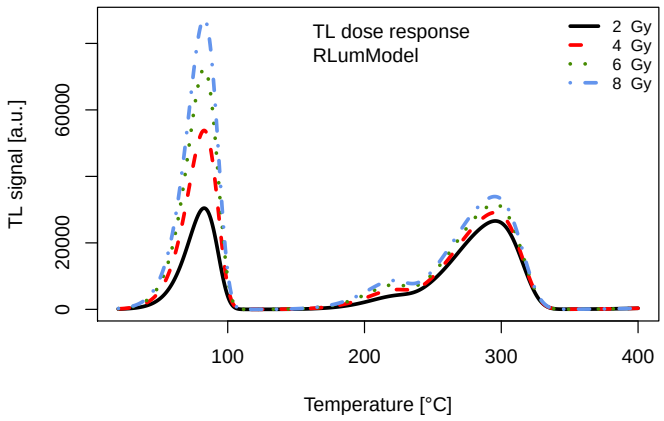


Fig. 11.14: Simulation of the dose response of the TL signal in the laboratory for doses in the range 2-8 Gy for a quartz sample, using the package *RLumModel*.

Code 11.12: Effect of burial temperature on TL of quartz (RlumModel)

```
#Effect of burial temperature on TL of quartz (RlumModel)
rm(list=ls())
suppressMessages(library(package = "RLumModel"))
##set list with burial temperatures
burial.temp <- seq(from = 0, to = 30, by = 10)
model.output <- lapply(burial.temp, function(x){
  sequence <- list(
    IRR = c(x, 20, 1e-11),
    TL = c(20, 400, 1))
  TL_data <- model_LuminescenceSignals(
    sequence = sequence,
    model = "Bailey2001",
    plot = FALSE,
    verbose = FALSE)
  return(Luminescence::get_RLum(TL_data, recordType = "TL$",
drop = FALSE))
})
model.output.merged <- merge_RLum(model.output)
plot_RLum(
  object = model.output.merged,
  xlab = "Temperature [\u00B0C]",
  ylab = "TL signal [a.u.]",main=" ",lwd=3,lty=1:4,
  legend.text = paste(burial.temp, "\u00B0C"),ylim=c(0,70000),
  combine = TRUE)
legend("top",bty="n",legend=c("Burial Temperature","RLumModel"))
```

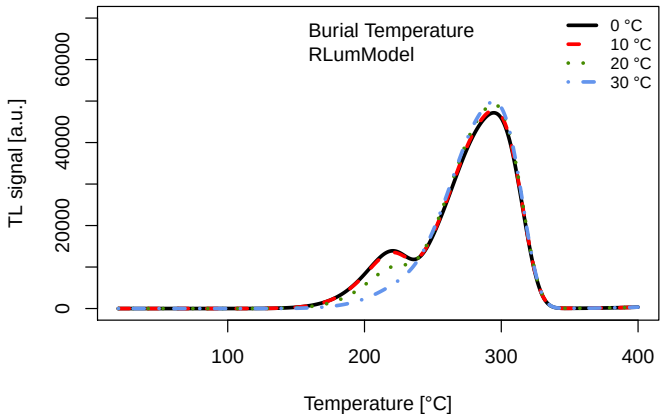


Fig. 11.15: Simulation of the effect of the burial temperature $T = 0 - 30$°C on the TL of a natural quartz sample. For a detailed study, see Koul et al. [19].

Chapter 12
COMPREHENSIVE MODELS FOR FELDSPARS

Abstract In this chapter we provide R codes for four different models previously developed for feldspars, apatites and other materials exhibiting quantum tunneling phenomena. These are the ground state tunneling (GST), irradiation-GST model (IGST), excited state tunneling model (EST) and thermally-assisted tunneling model (TA-EST). We demonstrate appropriate R functions which can simulate a wide variety of processes in feldspars, for both natural and laboratory irradiated samples. We present specialized codes for analyzing CW-IRSL and TL signals from freshly irradiated samples, as well as for simulating a variety of multiple stage experiments, involving thermal and optical pretreatments of samples in the laboratory. The chapter concludes with several R examples for the TA-EST model, which can be used for low temperature thermochronology studies.

Table 12.1: The various FSF functions used in this chapter. The first column indicates the model for which each function can be used.

Model	THE FSF FUNCTIONS
GST	***AFfortimeT(tim)***
	Sets parameter *distr* at the end of the anomalous fading period *tim* (in s)
IGST	***irradfortimeT(tirr)***
	Sets parameter *distr* at the end of the irradiation time *timCW* (in s)
EST	***CWfortimeT(timCW)***
	Sets parameter *distr* at the end of the IR stimulation time *timCW* (in s)
EST	***CWsignal(timCW)***
	Evaluates and and returns the CW-IRSL signal
EST	***stimIRSL()***
	Calls *CWfortimeT* and *CWsignal*; sets *distr* and returns the CW-IRSL signal
EST	***heatTo(Tph)***
	Sets parameter *distr* at the end of preheating to temperature Tph (in ˚C)
EST	***heatAt(Tph,tph)***
	Sets parameter *distr* at the end of preheating for time *tph* (in s)
	at a temperature *Tph* (in ˚C)
EST	***TLsignal(temp)***
	Evaluates and and returns the TL signal at temperatures *temp* (in ˚C)
EST	***stimTL()***
	Calls functions *heatTo* and *TLsignal,* sets *distr* and returns the TL signal
TA-EST	***irradandThermalfortimeT(tirr)***
	Sets parameter *distr* at the end of irradiation time *tirr* (in s), for a fixed
	sample temperature (*Tirr*)
TA-EST	***irradattemp(Tirr)***
	Sets parameter *distr* at the end of irradiation temperature *Tirr* (in ˚C), for
	a fixed irradiation time (*tirr*)

Code 12.1: The nearest neighbor distribution at geological times

```
# The nearest neighbor distribution at geological times
rm(list=ls())
s<-3e15                  # frequency factor
rho<-1e-6                # rho-prime values 0.005-0.02
rc<-0.0                  # for freshly irradiated samples, rc=0
timesAF<-3.154e7*c(0,1e2,1e4,1e6)          # times in seconds
rprimes<-seq(from=rc,to=2.2,by=0.002)    # rprime=0-2.2

##### function to find distribution of distances ###
AFfortimeT<-function(tim){3*(rprimes**2.0)*exp(-(rprimes**3.0))*
    exp(-exp(-(rho**(-1/3))*rprimes)*s*tim)}
######
```

```
distribs<-sapply(timesAF,AFfortimeT)
# Plots
cols=c(NA,NA,NA,"black","red","green","blue")
matplot(rprimes,distribs,xlab="Dimensionless distance r'",
ylab="Nearest neighbor Distribution g(r')",type="l",lwd=4)
legend("topright",bty="n", lty=c(NA,NA,NA,1,2,3,4), lwd=4,
col=cols,legend = c("Elapsed", "time"  ," ","t=0 s",
 expression("10"^"2"*" years"),
expression("10"^"4"*" years"),expression("10"^"6"*" years")))
```

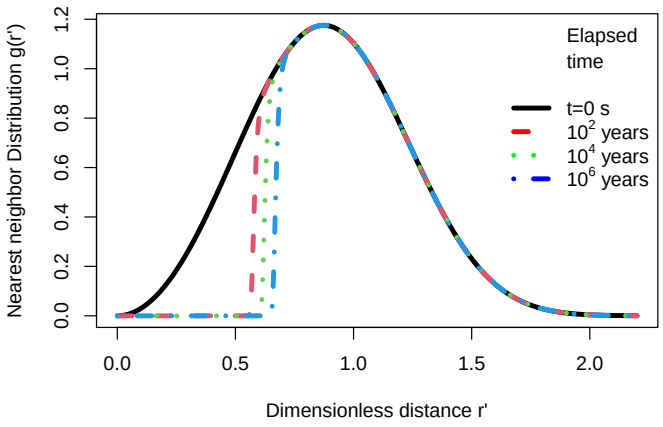


Fig. 12.1: Examples of the nearest neighbor distribution at different times $t = 0, 10^2, 10^4, 10^6$ years. The solid black line represents the *unfaded* nearly symmetric distribution at time $t = 0$. As time increases, the "tunneling front" is the almost vertical line which moves to the right, as more and more electrons are recombining at different distances r'. See also the discussion in Chapter 6.

Code 12.2: Anomalous fading at geological and laboratory times

```
# Anomalous fading  over geological and laboratory times
rm(list = ls(all=T))
rho<-1e-6                  # Dimensionless acceptor density
dr<-.01                    #Step in dimensionless distance r'
rprimes<-seq(0,2.2,dr)     #Values of r'=0-2.2 in steps of dr
```

```
s<-3e15
seff<-s*exp(-rprimes*(rho**(-1/3.0)))  # Effective A

######################## Functions  ################

######################## Anomalous fading Functions
#### AFfortimeT
AFfortimeT<-function(timAF){distr<<-distr*exp(-(seff*timAF))}
######################## End of Functions

########################## Simulations ###############
par(mfrow=c(1,2))
#### Example : Anomalous fading
# Long term fading  0-10^4 years in nature
distr<<-3*rprimes^2*exp(-rprimes^3)
timesAF<-3.154e7*c(.1,.2,.5,1,2,5,10,20,50,100,200,500,1000,
2000,5000,1e4)
n<-dr*colSums(sapply(timesAF,AFfortimeT))
plot(timesAF/(3.154e7),100*n,typ="p",lwd=2,pch=1,col="red",
     ylim=c(70,110),xlab=expression("Time [years]"),
     ylab="Remaining charge [%]")
legend("topleft",bty="n",legend=c("(a)"," ","AF in nature",
"Analyt. Eq."),lwd=2,lty=c(NA,NA,NA,1), col=c(NA,NA,NA,"blue"))
lines(timesAF/3.154e7,y=100*exp(-rho*(log(1.8*s*timesAF)**3.0)),
   lwd=2,col="blue")
### Repeat for short term fading 0-10 days in lab
distr<<-3*rprimes^2*exp(-rprimes^3)  # unfaded distribution
timesAF<-3600*24*c(1e-4,1e-3,1e-2,.1,.2,.5,seq(1,10,.5))
n<-dr*colSums(sapply(timesAF,AFfortimeT))
plot(timesAF/(3600*24),100*n,typ="p",lwd=2,pch=1,col="red",
     ylim=c(70,110),xlab=expression("Time [days]"),
     ylab="Remaining charge [%]")
legend("topleft",bty="n",legend=c("(b)"," ","AF in Lab",
 "Remaining charge","0-10 days"))
lines(timesAF/(3600*24),
y=100*exp(-rho*(log(1.8*s*timesAF)**3.0)),lwd=2,col="blue")
```

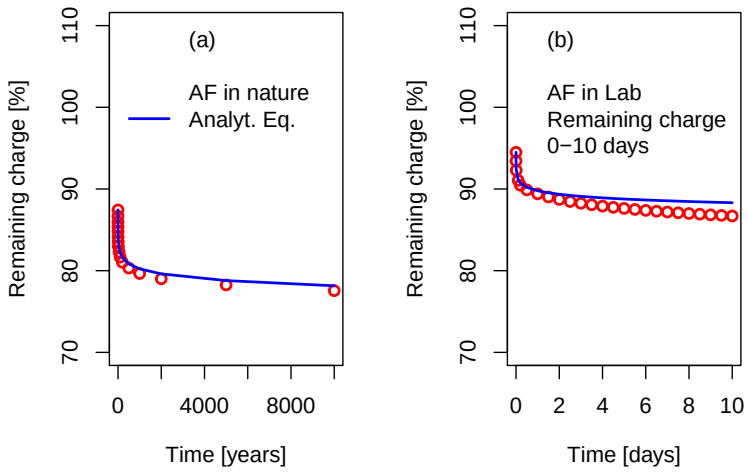


Fig. 12.2: (a) Simulation of long term anomalous fading in nature over a time period of 10^4 years, starting with an unfaded sample. The solid line indicates the approximate analytical Eq.(**??**). (b) Short term AF in the laboratory, over a period of 10 days after the end of irradiation. The parameters in the model are typical for feldspars.

Code 12.3: Feldspar irradiation in nature

```
## Distribution of distances for feldspar irradiation in nature
rm(list = ls(all=T))
rho<-2e-6                  # Dimensionless acceptor density
Do<-538                    #Do in Gy
yr<-365*24*3600            #year is seconds
Ddot<-3/(1e3*yr)           #Low natural dose rate = 3 Gy/kA
dr<-.04                    #Step in dimensionless distance r'
rprimes<-seq(0.01,2.2,dr) #Values of r'=0-2.2 in steps of dr
s<-2e15                    # Frequency factors in s^-1
seff<-s*exp(-rprimes*(rho**(-1/3.0)))  # Effective s
tau<-1/seff
######################## Irradiation Function
#### irradfortimeT
irradfortimeT<-function(tirr){distr<<-unFaded*(Ddot*tau/
(Do+Ddot*tau))*  (1-exp(-(Do+Ddot*tau)/(Do*tau)*tirr))}
# function calculates new distribution at end of irradiation
```

```
###################### End of Functions
unFaded<-3*rprimes^2*exp(-rprimes^3)  # Unfaded sample
irrTimes<-c(6.67e4,1.67e5,1e6)*yr
distribs<-sapply(irrTimes,irradfortimeT)
########## plot distibutions
matplot(rprimes,distribs,xlab="Dimensionless Distance r'",
typ="o",lty="solid", ylab="Distribution of r'",pch=c(2,3,4),
lwd=2,col=c("blue","green","red"))
lines(rprimes,unFaded,col="black",pch=1,typ="p",lwd=2)
legend("topright",bty="n",legend=c(
  expression("Natural Irradiation",
"Unfaded","tirr=6.7x10"^4*" y", "tirr=1.7x10"^5*" y",
"Field saturation")),  pch=c(NA,1,2,3,4),
col=c(NA,"black","blue","green","red"),lwd=2)
```

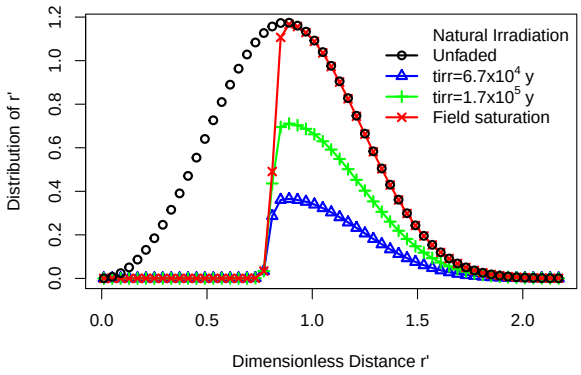


Fig. 12.3: Simulation of irradiation process in nature. As the irradiation time increases, the asymmetric distribution of distances r' approaches the field saturation distribution ($\times$ symbols). The symmetric curve indicates the initial distribution of distances, for the unfaded sample (o symbols). Compare with the results of Li and Li [23].

Code 12.4: Feldspar irradiation in nature- Dose response

```
## Feldspar irradiation in nature  - Dose response
rm(list = ls(all=T))
rho<-2e-6                  # Dimensionless acceptor density
```

```
s<-3e15                  # Frequency factors in s^-1
Do<-538                    #Do in Gy
yr<-365*24*3600            #year is seconds
Ddot<-2.85/(1e3*yr)        #Low natural dose rate = 2.85 Gy/Ka
dr<-.05                    #Step in dimensionless distance r'
rprimes<-seq(0.01,2.2,dr) #Values of r'=0-2.2 in steps of dr
seff<-s*exp(-rprimes*(rho**(-1/3.0)))  # Effective s
tau<-1/seff
######################## Irradiation Function
#### irradfortimeT
irradfortimeT<-function(tirr){distr<<-unFaded*(Ddot*tau/
  (Do+Ddot*tau))*  (1-exp(-(Do+Ddot*tau)/(Do*tau)*tirr))}
# function calculates new distribution at end of irradiation
######################## End of Function ##########
par(mfrow=c(1,2))
unFaded<-3*rprimes^2*exp(-rprimes^3)  # Unfaded sample
irrTimes<-10^seq(2,6,by=.2)*yr
distribs<-sapply(irrTimes,irradfortimeT)
########## plot distibutions
matplot(rprimes,distribs,typ="l",ylim=c(0,1.8),lty="solid",
xlab="Dimensionless Distance r'",  ylab="Distribution of r'",
lwd=2)
legend("topleft",bty="n",legend=c(expression("(a)",
"Irradiation in Nature","Distributions of r'",
"tirr=100-10"^"6"*"y")))
plot(irrTimes/yr,colSums(distribs)*dr,typ="p",lwd=2,
xlab="Time [y]",ylab="Trap filling ratio n(t)/N",ylim=c(0,1.1))
legend("topleft",bty="n",legend=c("(b)"," ",
"Trap filling n(t)/N",  "Analyt. Eq."),lwd=2,
lty=c(NA,NA,NA,1),pch=c(NA,NA,1), col=c(NA,NA,NA,"red"))
lines(irrTimes/yr,(1-exp(-Ddot*irrTimes/Do))*exp(-rho*
(log(Do*s/Ddot)**3.0)),lwd=2,col="red")
```

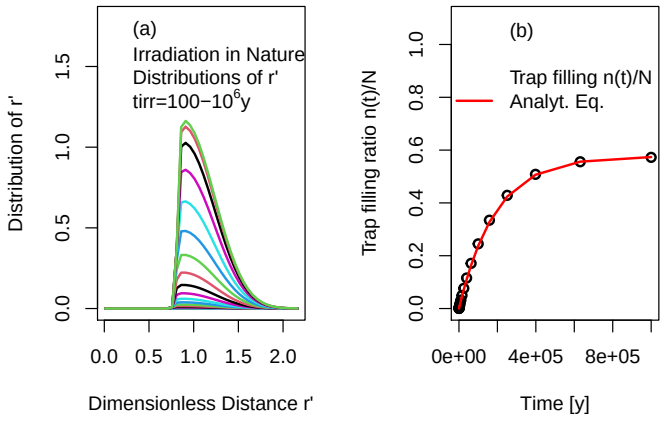


Fig. 12.4: Simulation of irradiation process in nature with a slow dose rate of 2.85 Gy/ka. As the irradiation time increases, the asymmetric distribution of distances r' approaches the field saturation distribution. The symmetric curve indicates the initial distribution of distances, for the unfaded sample (o symbols).

F

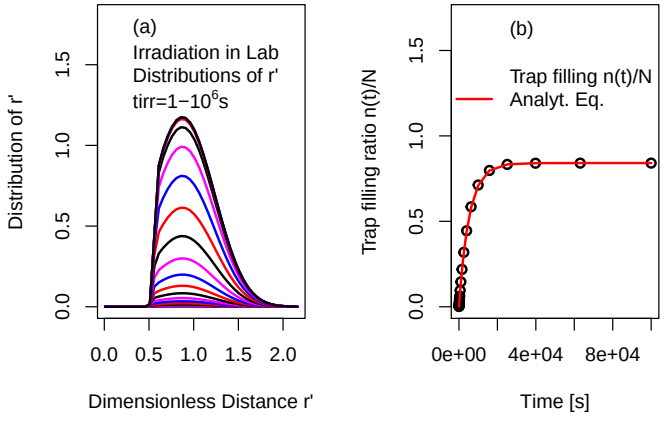


Fig. 12.5: Simulation of irradiation process in the *laboratory* with a dose rate of 0.1 Gy/s and for irradiation times $tirr = 1 - 10^6$ s. As the irradiation time increases, both the asymmetric distribution of distances r' in (a), and the trap filling ratio $n(t)/N$ in (b) approach saturation.

Code 12.5: Simulation of CW-IRSL signal from freshly irradiated feldspars

```
## CW-IRSL function for freshly irradiated feldspars (EST)
rm(list = ls(all=T))
source("Pagonis2020FSF.R")# Loads the functions
rho<-.013                 # Dimensionless acceptor density
dr<-.05                   # Step in dimensionless distance r'
rprimes<-seq(0,2.2,dr)    # Values of r'=0-2.2 in steps of dr'
A<-3                      # A=stun*sigma*I/B
Aeff<-A*exp(-rprimes*(rho**(-1/3.0)))  # Effective A
########################## Simulations ################
par(mfrow=c(1,3))
#### Example : IRSL for unfaded sample
distr<<-3*rprimes^2*exp(-rprimes^3)  # unfaded distribution
distr1<-distr
timesCW<-seq(1,100)
IRsignal1<-stimIRSL()
distr2<-distr
########## End of Simulations, next Plot the results #######

plot(rprimes,distr1,ylim=c(0,1.9),typ="o",pch=1,col="black",
     ylab="Distribution of r'",xlab="Dimensionless distance r'")
lines(rprimes,distr2,ylim=c(0,1.9),typ="o",pch=2,col="blue",
     ylab="Distribution of r'",xlab="Dimensionless distance r'")
legend("topleft",bty="n",legend=c("(a)"," ",
" Distribution of r'", " before and after IR"))
matplot(timesCW,t(sapply(timesCW,CWsignal)),typ="l",lty="solid",
ylim=c(0,.004), lwd=2, xlab=expression("Times [s]"),
ylab="CW-IRSL [a.u.]")
legend("topleft",bty="n",legend=c("(b)","    ","CW-IRSL curves",
   "for each r'"))
plot(timesCW,IRsignal1,typ="p",lwd=2,pch=1,col="red",
ylim=c(0,.17),xlab=expression("Times [s]"),
ylab="CW-IRSL [a.u.]")
legend("topleft",bty="n",legend=c("(c)"," ","CW-IRSL signal",
  "Unfaded sample"," ","Sum of CW-IRSL","curves"))
lines(timesCW,3*rho*A*1.8*exp(-rho*(log(1+1.8*A*timesCW))**3.0)*
```

```
(log(1 +1.8* A*timesCW) ** 2.0)/(1 + 1.8*A* timesCW),lwd=2,
col="blue")
```

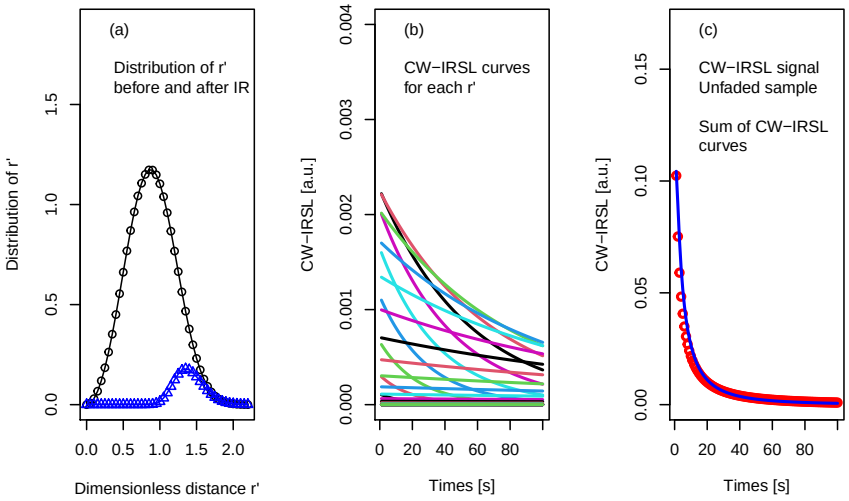


Fig. 12.6: Simulation of CW-IRSL experiment in the EST model, for freshly irradiated samples. (a) The distributions of trapped charge $n(r', t)$ at the beginning and at the end of the CW-IRSL experiment ; (b) The CW-IRSL curves evaluated for each distance r'; (c) The sum of the curves shown in (b), yields the total CW-IRSL signal. The solid line in (c) is the analytical KP-CW equation developed by Kitis and Pagonis [13]. The parameters in the model are typical for feldspars.

Code 12.6: Simulation of TL signal from freshly irradiated feldspars

```
## Simple TL function for freshly irradiated feldspars
rm(list = ls(all=T))
source("Pagonis2020FSF.R")# Loads the FSF functions
rho<-.013                  # Dimensionless acceptor density
s<-3.5e12                  # Frequency factors in s^-1
E<-1.45                    # Energy in eV
Tph<-300                   #Preheat temperature (C)
tph<-10                    #Preheat time (s)
dr<-.1                    #Step in dimensionless distance r'
rprimes<-seq(0.01,2.2,dr)  #Values of r'=0-2.2 in steps of dr
kb<-8.617e-5               #Boltzmann constant
```

```
beta<-1
Tpreheat<-273+320          # Preheat Temperature
seff<-s*exp(-rprimes*(rho**(-1/3.0)))  # Effective s
########################## Simulations ###############
par(mfrow=c(1,3))
# #### Example #1: TL for unfaded sample
distr<<-3*rprimes^2*exp(-rprimes^3)
distr1<-distr
temps<-273+seq(1:400)      # Temperatures for TL
manyTL<-t(sapply(temps,TLsignal))
TL1<-stimTL()              # Evaluate TL signal
distr2<-distr

########## End of Simulations, next Plot the results #######
plot(rprimes,distr1,ylim=c(0,1.9),typ="o",pch=1,col="black",
     ylab="Distribution of r'",xlab="Dimensionless distance r'")
lines(rprimes,distr2,typ="o",pch=2,col="red",
     ylab="Distribution of r'",xlab="Dimensionless distance r'")
legend("topleft",bty="n",legend=c("(a)"," ",
"   Distribution of r'",  "   before and after", "TL"))
matplot(temps-273,manyTL,typ="l",lty="solid",
xlim=c(175,450),ylim=c(0,.03),lwd=2,
xlab=expression("Temperature ["^"o"*"C]"), ylab="TL [a.u.]")
legend("topleft",bty="n",legend=c("(b)","    ","TL curves ",
  "for each r'"))
plot(temps-273,TL1,typ="l", lwd=2,pch=1,col="black",
xlim=c(175,450),ylim=c(0,.018),
xlab=expression("Temperature ["^"o"*"C]"),ylab="TL [a.u.]")
legend("topleft",bty="n",legend=c("(c)"," ","Sum of all",
   "TL curves "))
```

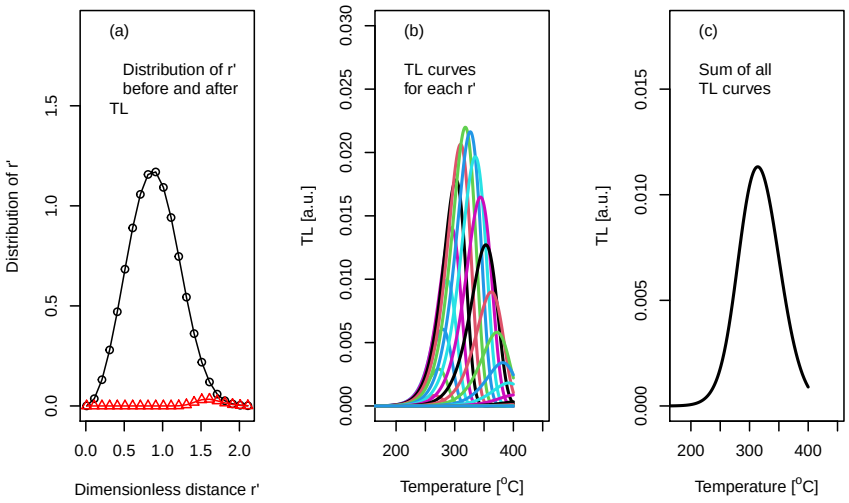


Fig. 12.7: Simulation of TL glow curve for freshly irradiated samples by heating up to 380°C, just below the high temperature end of the TL glow curve. (a) The distributions of distances r' before and after the heating the sample are shown as circles and triangles, respectively; (b) The partial first order TL glow curves; (c) The sum of the glow curves from (b).

Code 12.7: TL from thermally/optically pretreated feldspar samples

```
## Examples of TL for thermally and optically treated samples
rm(list = ls(all=T))
source("Pagonis2020FSF.R")# Loads the FSF functions
rho<-.013                 # Dimensionless acceptor density
Po<-s<-3.5e12             # Frequency factors in s^-1
E<-1.45                   # Energy in eV
Tph<-300                  #Preheat temperature (C)
tph<-10                   #Preheat time (s)
dr<-.05                   #Step in dimensionless distance r'
rprimes<-seq(0.01,2.2,dr)  #Values of r'=0-2.2 in steps of dr
kb<-8.617e-5              #Boltzmann constant
beta<-1
seff<-s*exp(-rprimes*(rho**(-1/3.0)))  # Effective s
```

```
A<-5
Aeff<-A*exp(-rprimes*(rho**(-1/3.0)))  # Effective A
########################## Simulations ###############
par(mfrow=c(1,2))
   ## Example #1: TL of unfaded sample
distr<<-3*rprimes^2*exp(-rprimes^3)  # unfaded distribution
distr1<-distr              # Store distr for plotting later
temps<-273+seq(1:500)      # Temperatures for TL
TL1<-stimTL()              # Evaluate TL signal
   ## Example #2: Heat to temperature Tpreheat, then measure TL
distr<<-3*rprimes^2*exp(-rprimes^3)  # unfaded distribution
Tpreheat<-273+320          # Preheat Temperature
heatTo(Tpreheat)           # Heat to 320 degC
distr2<-distr              # Store distr for plotting
TL2<-stimTL()              # Store TL
## Example #3: Heat for 30 s at Tpreheat=320C, then measure TL
distr<<-3*rprimes^2*exp(-rprimes^3)
heatTo(Tpreheat)           # Heat to 320 degC
heatAt(Tpreheat,30)        # Heat for 30 s at 320C
distr3<-distr
TL3<-stimTL()
   ## Example #4: CW-IRSL excitation for 10 s, then measure TL
distr<<-3*rprimes^2*exp(-rprimes^3)  # unfaded distribution
timesCW<-seq(1,10)
invisible(sapply(timesCW,CWfortimeT) ) # IR for 10 s
distr4<-distr
TL4<-stimTL()
########## End of Simulations, next Plot the results #######
plot(rprimes,distr1,ylim=c(0,1.9),typ="l",pch=1,col="black",
lwd=2,ylab="Distribution of r'",
xlab="Dimensionless distance r'")
legend("topleft",bty="n",legend=c("(a)","Various Examples",
 "of r' Distributions"))
text1 <- data.frame(x=c(.5,.8,1.,1.25),
 y=c(1,.7,.55,.3), labels=c("1","2","3","4"))
text(text1)
lines(rprimes,distr2,typ="l",col="red",lwd=2)
lines(rprimes,distr3,typ="l",col="blue",lwd=2)
lines(rprimes,distr4,typ="l",col="magenta",lwd=2)
plot(temps-273,TL1,xlim=c(200,450),ylim=c(0,.017),typ="l",
     col="black",lwd=2,
xlab=expression("Temperature ["^"l"*"C]"), ylab="TL [a.u.]")
lines(temps-273,TL2,typ="l",col="red",lwd=2)
lines(temps-273,TL3,typ="l",col="blue",lwd=2)
```

```
lines(temps-273,TL4,typ="l",col="magenta",lwd=2)
legend("topleft",bty="n",legend=c("(b)"," ",
 "Corresponding TL"))
text1 <- data.frame(x=c(270,310,330,350),
y=c(0.009,.006,.004,.002), labels=c("1","2","3","4"))
text(text1)
```

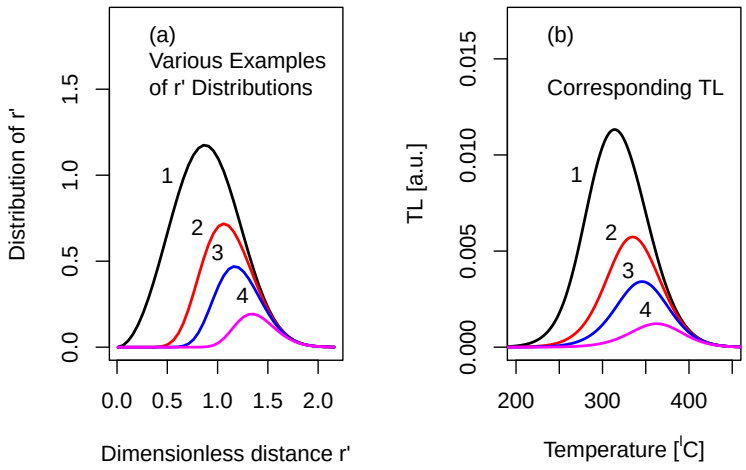


Fig. 12.8: Several examples of the simulation functions for thermally and optically treated samples. The parameters in the model are typical for feldspars. (a) The distribution of distances and (b) The corresponding TL glow curves, for the following processes: (1) TL for unfaded sample; (2) Heat to a preheat temperature T=320°C, then measure TL; (3) Heat for 30 s at T=320°C, then measure TL; (4) CW-IRSL excitation for 10 s, then measure TL.

Code 12.8: Dose response of feldspar in the TA-EST model of Brown et al.

```
#Dose response of feldspars, irradiation in nature (TA-EST)
rm(list = ls(all=T))
rho<-3e-3                  # Dimensionless acceptor density
Po<-s<-2e15                # Frequency factors in s^-1
E<-1.3                     # Energy in eV
```

```
Do<-1600                    #Do in Gy
yr<-365*24*3600             #year is seconds
Ddot<-2.85/(1e3*yr)         #Low natural dose rate = 1 Gy/kA
dr<-.05                     #Step in dimensionless distance r'
rprimes<-seq(0.01,2.2,dr) #Values of r'=0-2.2 in steps of dr
kb<-8.617e-5                #Boltzmann constant
seff<-s*exp(-rprimes*(rho**(-1/3.0)))  # Effective s
Tirr<-273-4
Peff<-(1/(1/s+1/seff))*exp(-E/(kb*Tirr))

########## Irradiation functions  ##########
#### irradfortimeT
irradandThermalfortimeT<-function(tirr){distr<<-distrUnfaded*
  (Ddot*rprimes/(Do*Peff+Ddot))*(1-exp(-(Ddot/Do+Peff)*tirr))}
###################### End of Functions ##########
par(mfrow=c(1,2))
distrUnfaded<-3*rprimes^2*exp(-rprimes^3)  # Unfaded sample
irrTimes<-10^seq(2,6,by=.2)*yr
distribs<-sapply(irrTimes,irradandThermalfortimeT)

########## plot distibutions
matplot(rprimes,distribs,typ="l",ylim=c(0,.45),lty="solid",
xlab="Dimensionless Distance r'",
ylab="Distribution of r'",lwd=2)
legend("topleft",bty="n",legend=c(expression("(a)",
"Irradiation in nature",
"Distributions of r'","tirr=10"^3*"-10"^6*"y")))
plot(irrTimes/yr,colSums(distribs)*dr,typ="o",lwd=2,
 xlab="Time [y]",ylab="Trap filling ratio n(t)/N",
 ylim=c(0,.27))
legend("topleft",bty="n",legend=c(" ","(b)",
  "Trap filling n(t)/N"))
```

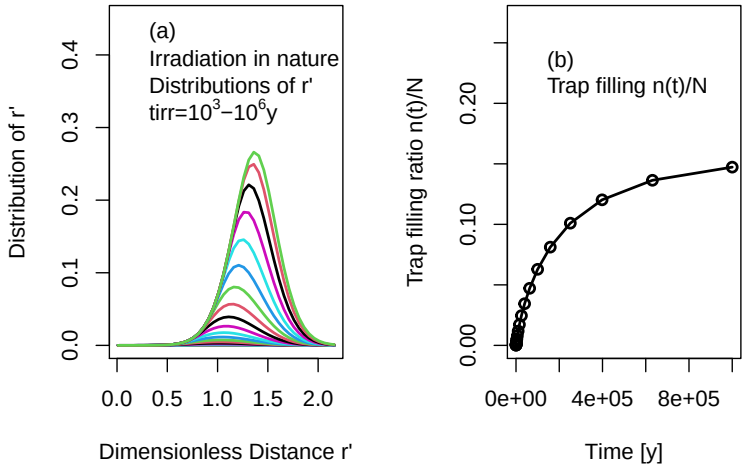


Fig. 12.9: Simulation of irradiations in nature in the TA-EST model, for a fixed burial temperature -4°C. (a) The distributions of the distance parameter r' at various irradiation times. Compare the shape of these distributions with Fig.12.4. (b) The corresponding dose response, shown as the trap filling ratio $n(t)/N$.

Code 12.9: Irradiations at various steady-state temperatures (TA-EST model)

```
## Multiple feldspar irradiations at steady-state temperatures

rm(list = ls(all=T))
rho<-1e-2                   # Dimensionless acceptor density
Po<-s<-2e15                 # Frequency factors in s^-1
E<-1.3                      # Energy in eV
Do<-1600                    #Do in Gy
yr<-365*24*3600             #year is seconds
Ddot<-2.85/(1e3*yr)         #Low natural dose rate = 1 Gy/kA
dr<-.05                     #Step in dimensionless distance r'
rprimes<-seq(0.01,2.2,dr) #Values of r'=0-2.2 in steps of dr
kb<-8.617e-5                #Boltzmann constant
seff<-s*exp(-rprimes*(rho**(-1/3.0)))  # Effective s
```

```
# Peff<-(1/(1/s+1/seff))*exp(-E/(kb*Tirr))   Effective P

########## Irradiation functions  ##########
#### irradatsometemp
irradatsometemp<-function(Tirr){
  Peff<-(1/(1/s+1/seff))*exp(-E/(kb*Tirr))
  distr<<-distrUnfaded*
  (Ddot*rprimes/(Do*Peff+Ddot))*(1-exp(-(Ddot/Do+Peff)*tirr))}
###################### End of Functions ##########
tirr<-1e3*yr
Tirrs<-273+c(-4,0,4,8)      # Burial temperatures
distrUnfaded<-3*rprimes^2*exp(-rprimes^3)  # Unfaded sample
distribs<-sapply(Tirrs,irradatsometemp)

########## plot distibutions
cols=c("black","red","blue","magenta")
pchs=c(1,2,3,4)
matplot(rprimes,distribs,typ="o",pch=pchs,col=cols,
ylim=c(0,.003),lty="solid",
xlab="Dimensionless Distance r'", ylab="Distribution of r'",
lwd=2)
legend("topleft",bty="n",legend=c("Irradiation in nature",
"Various Burial Temperatures"," ","Distributions of distances"))
legend("topright",bty="n",legend=c(expression("-4 "^o*"C",
  "  0  "^o*"C"," +4  "^o*"C"," +8  "^o*"C")),pch=pchs,col=cols)
```

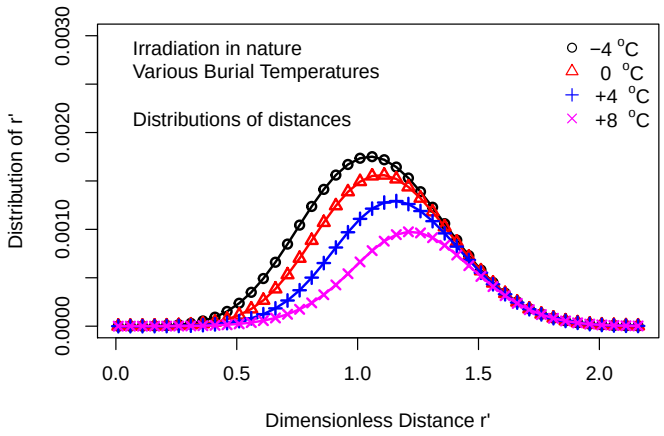


Fig. 12.10: Multiple irradiations in nature using the TA-EST model, for a variable burial temperature $Tirr$= -4, 0, 4, 8 °C, and for a fixed irradiation time $tirr = 10^3$ y.