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
```

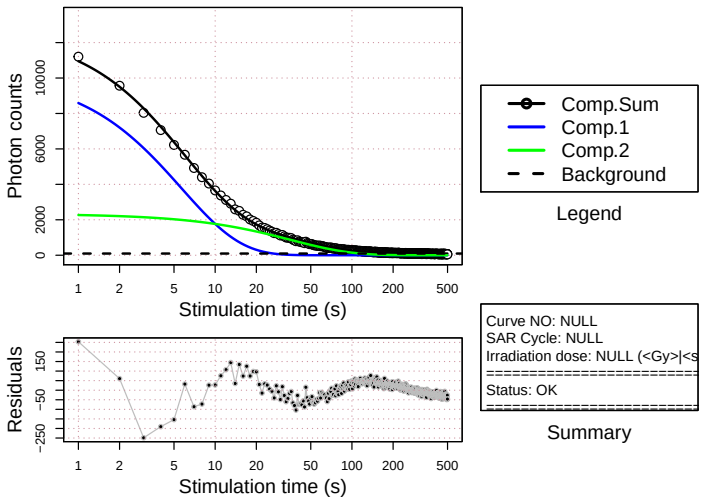

Fig. 3.1: Example of fitting the first 500 s of a CW-OSL signal with two exponential components, using the package *numOSL*. The CW-OSL data are from a freshly irradiated aliquot of feldspar sample KST4 (Pagonis et al. [44]).

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
```

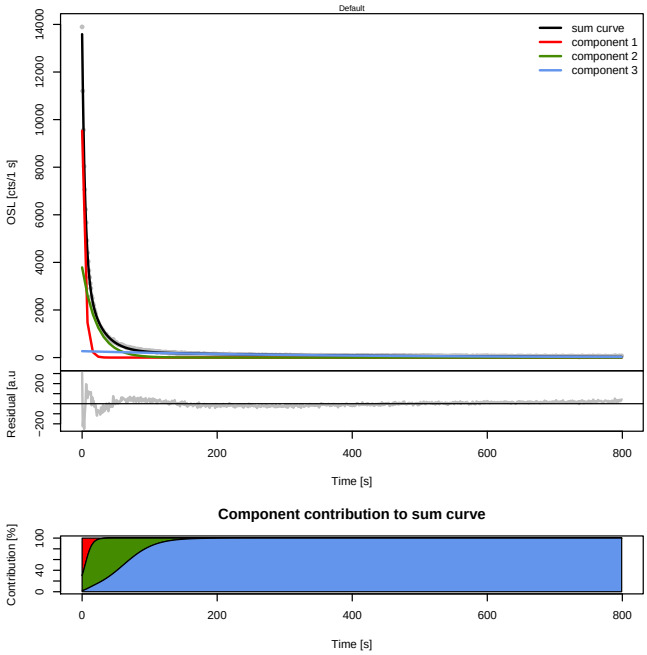

Fig. 3.2: Example of fitting the first 800 s of a CW-OSL signal from feldspar sample KST4, with three exponential components, using the package *Luminescence*. Experimental data from Pagonis et al. [44].

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"))
```

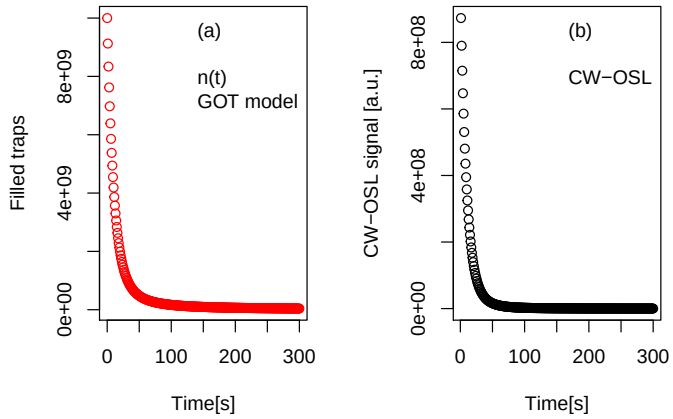

Fig. 3.3: Numerical solution of the GOT differential equation (**??**) for CW-OSL, using the R package *deSolve*. (a) Filled traps $n(t)$. (b) The corresponding CW-OSL signal.

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"))
```

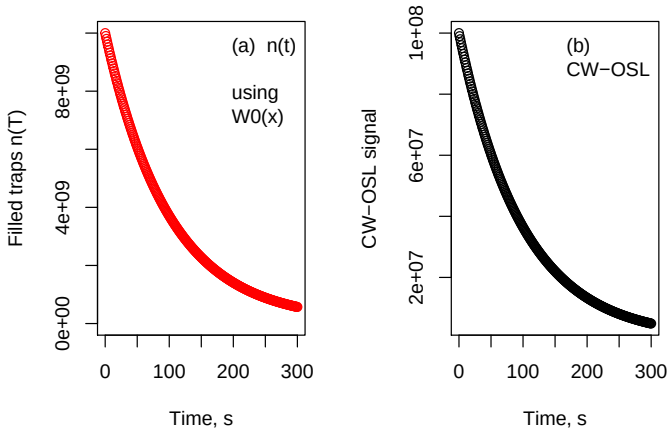

Fig. 3.4: Plot of the analytical solution of GOT differential equation for CW-OSL, using the Lambert W function solution from Eq.(**??**). (a) Filled traps $n(T)$ and (b) The corresponding CW-OSL signal.

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"))
```

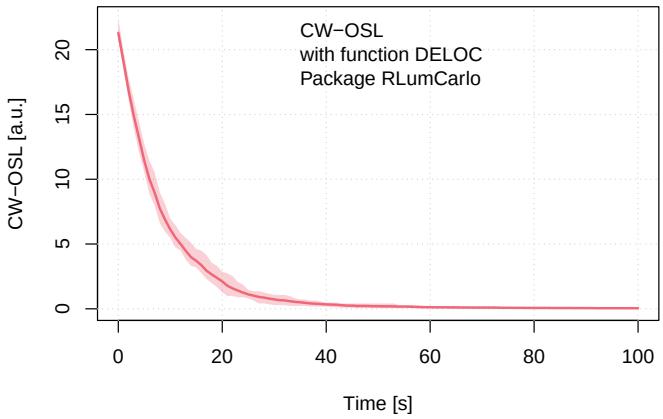

Fig. 3.5: Simplest example of MC simulation for CW-OSL DELOCalized transition process, within the OTOR model and based on the GOT Eq.(**??**). The parameters here are the excitation rate $\lambda = 0.12$ s^{-1} and retrapping ratio $R = 0.1$.

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"))
```

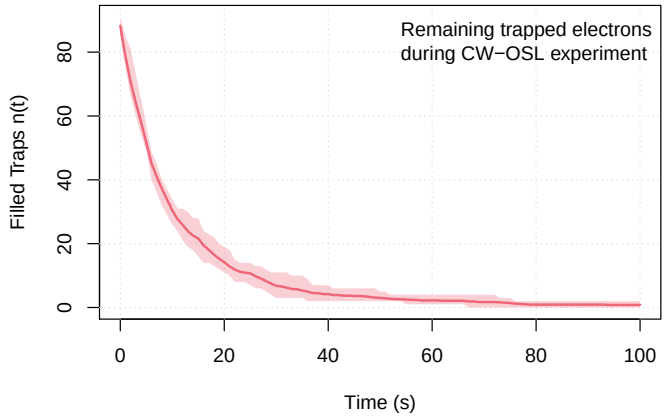

Fig. 3.6: Plotting the remaining electrons $n(t)$ during a DELOCalized CW-OSL simulation.

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)
```

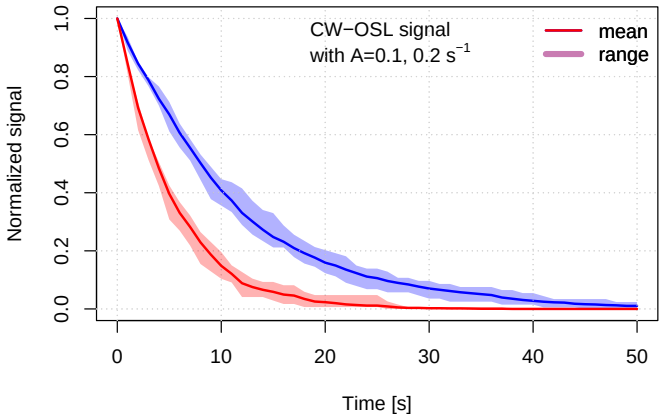

Fig. 3.7: Combine two plots for delocalized CW-OSL signals, with two different excitation rates $A = 0.1,\ 0.2\ \mathrm{s}^{-1}$.

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)
```

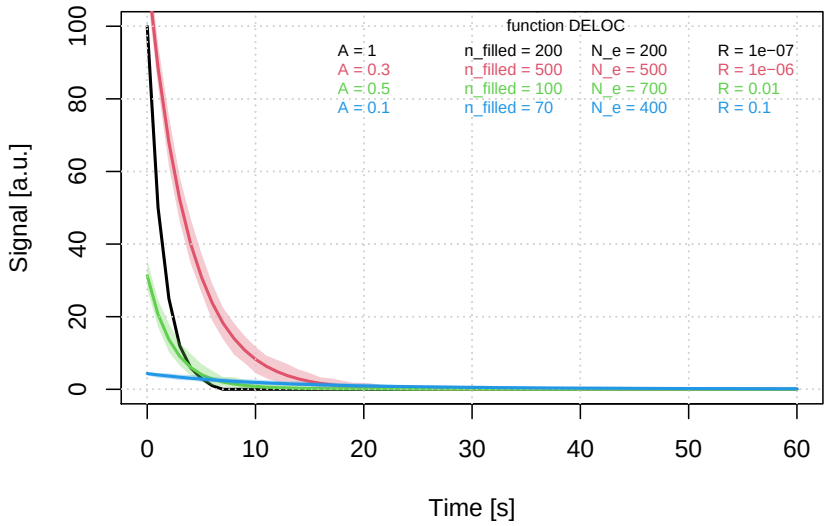

Fig. 3.8: Simulation of several CW-OSL curves using the DELOC functions of the *RLumCarlo* package, with several parameter changes.

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"))
```

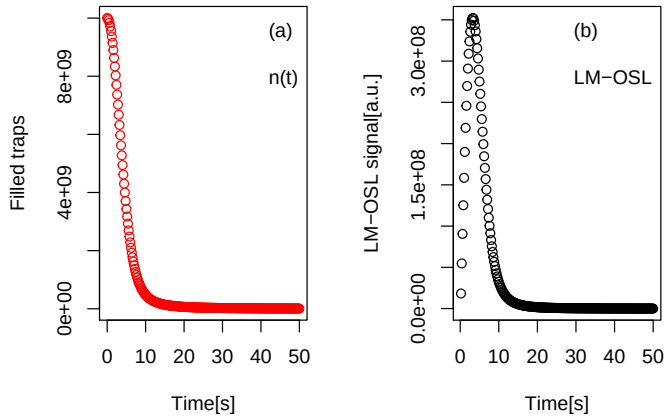

Fig. 3.9: Numerical solution of the GOT differential equation for LM-OSL. (a) Filled traps $n(t)$ and (b) The respective LM-OSL signal.

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"))
```

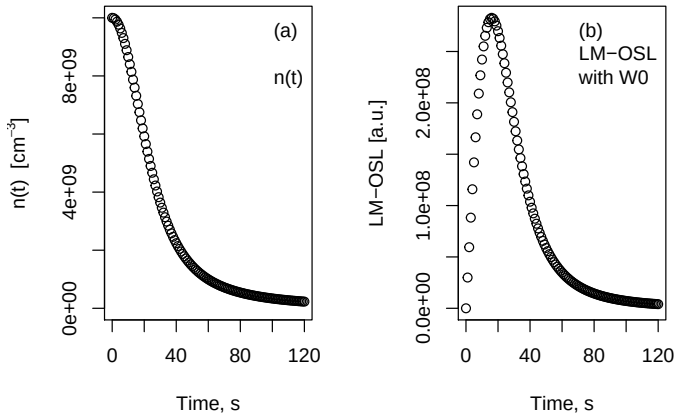

Fig. 3.10: Plots of the analytical solution of GOT equation Eq.(**??**) for LM-OSL, using the Lambert function *W[z]*. (a) Filled traps $n(t)$ and (b) The LM-OSL signal.

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
```

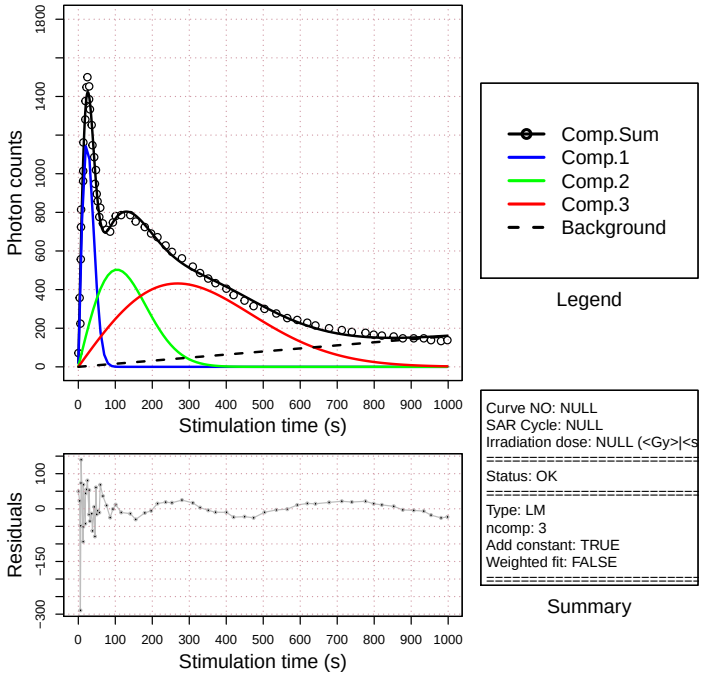

Fig. 3.11: Example of analyzing an LM-OSL signal from the dosimetric material CaF_2:N with three first order components, using the package *numOSL* (Kitis et al. [16]).

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
```

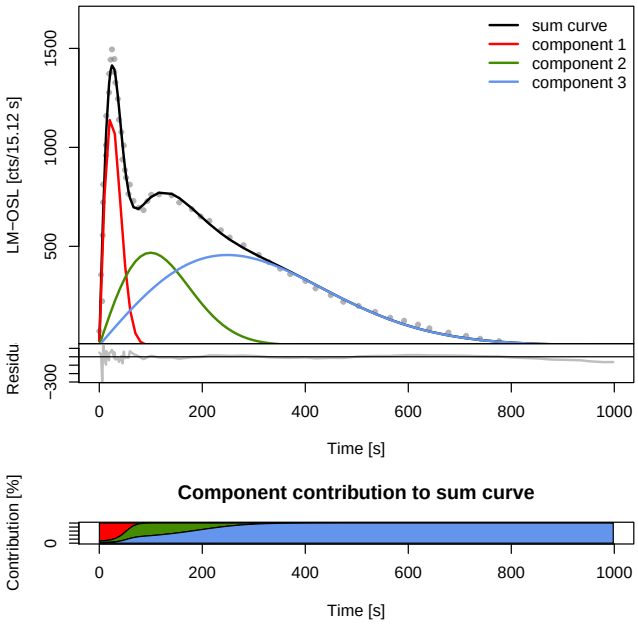

Fig. 3.12: Example of fitting LM-OSL data from the dosimetric material CaF_2:N with 3 first order components, using the package *Luminescence* (Kitis et al. [16]).

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"))
```

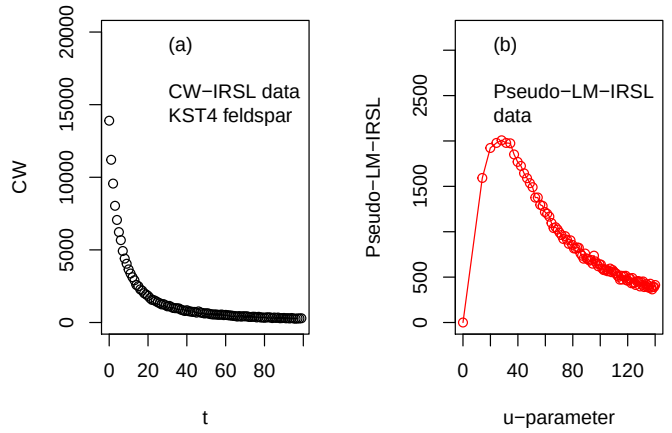

Fig. 3.13: Example of transforming a CW-IRSL signal into pseudo-LM-IRSL data, by reading a *.txt* file containing two columns, containing the time and the corresponding CW-intensities. The experimental data is discussed in Pagonis et al. [44].

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"))
```