Chapter 11
INFRARED STIMULATED LUMINESCENCE SIGNALS: DATA ANALYSIS

Code 11.1: CW-IRSL data fitted with KP-CW equation

```
# J1000 CW-IRSL data fitted with KP-CW equation
from scipy import optimize
from sympy import *
import numpy as np
import matplotlib.pyplot as plt
from prettytable import PrettyTable
import warnings
warnings.filterwarnings("ignore")
data = np.loadtxt('J1000cwirsl.txt')
x_data,y_data=data[:,0][1:800], data[:,1][1:800]
y_data=y_data/max(y_data)
plt.subplot(2,1, 1);
def test_func(x, imax_fit,rho_fit, A_fit,bgd_fit):
    return imax_fit*np.exp (-rho_fit*(np.log(1 + A_fit*x))\
** 3.0)*(np.log(1+A_fit*x)**2.0)/(1+x*A_fit)+bgd_fit
params, cov = optimize.curve_fit(test_func,\
x_data, y_data)
drho= round(np.sqrt(cov[1][1]),5)
dA = round(np.sqrt(cov[2][2]),2)
dimax = round(np.sqrt(cov[0][2]),2)
plt.scatter(x_data, y_data, label='J1000 feldspar');
plt.plot(x_data, test_func(x_data, *params[0:4]),
label='KP-CW equation');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.ylabel('CW-IRSL signal [a.u.]');
```

```
plt.xlabel('Time [s]');
plt.subplot(2,1, 2);
plt.plot(x_data,test_func(x_data, *params[0:4])-\
y_data,"o",label='Residuals');
leg = plt.legend();
leg.get_frame().set_linewidth(0.0)
plt.ylabel('Residuals');
plt.xlabel('Time [s]');
plt.ylim(-.1,.1);
plt.tight_layout();
imax,rho, A, bgd=int(params[0]),round(params[1],5),\
round(params[2],2),round(params[-1],3)
res=test_func(x_data, *params)-y_data
FOM=round(100*np.sum(abs(res))/np.sum(y_data),2)
myTable=PrettyTable(["A",'dA', "rho",  "d(rho)",\
"s'(s^-1)","ds'",'bgd']);
myTable.add_row([imax,dimax,rho,drho, A, dA,bgd]);
print('FOM=',FOM,' %')
```

```
  FOM= 3.64  %
```

```
print(myTable)
```

```
  +---+------+---------+--------+----------+------+-------+
  | A |  dA  |   rho   | d(rho) | s'(s^-1) | ds'  |  bgd  |
  +---+------+---------+--------+----------+------+-------+
  | 2 | 0.08 | 0.01157 | 0.0003 |  12.07   | 0.33 | 0.033 |
  +---+------+---------+--------+----------+------+-------+
```

```
plt.show();
```

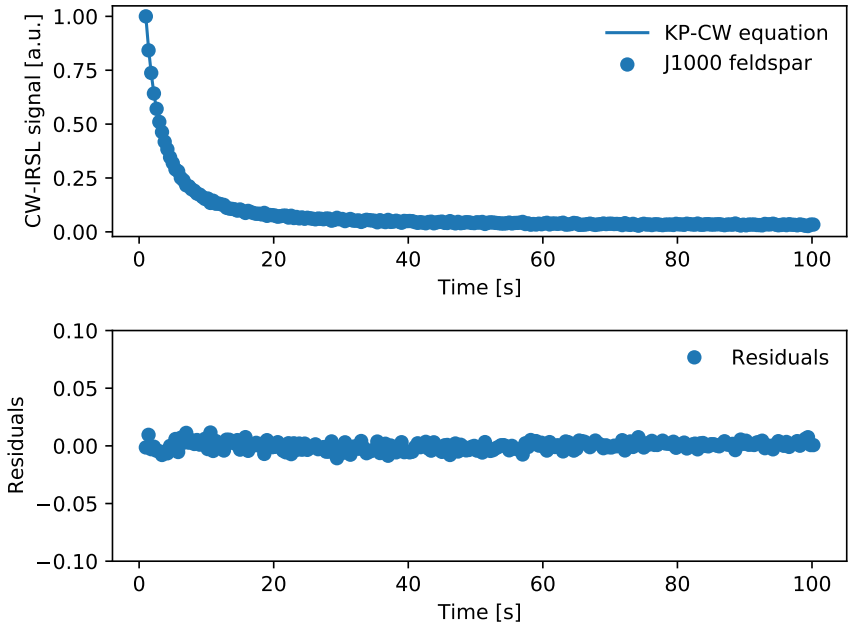

Fig. 11.1: Experimental CW-IRSL curve from a freshly irradiated J1000 feldspar sample, fitted using the KP-CW analytical Eq.(**??**). For more details and examples from several other types of feldspars, see Pagonis et al. [1].

Code 11.2: LM-IRSL data fitted with KP-LM equation

```
# Deconvolution with the KP-LM equation, fixed bgd
from scipy import optimize
import numpy as np
import matplotlib.pyplot as plt
from prettytable import PrettyTable
import warnings
warnings.filterwarnings("ignore")
data = np.loadtxt('J1000 50degC LMIRSL.txt')
x_data,y_data=data[:,0], data[:,1]
y_data=y_data/max(y_data)
def KVLMOSL(x, A,sprime,rho):
    F=np.log(1+abs(sprime)*x**2/(2*P))
```

```
    LMOSL= abs(A)*np.exp (-abs(rho)*\
F** 3.0)*(F**2.0)/(1+abs(sprime)*x**2/(2*P))
    return LMOSL
def total_LMOSL(x, *inis):
    u=np.array([0 for i in range(len(x_data))])
    As, sprimes=inis[0:nPks], inis[nPks:2*nPks]
    rho=inis[-1]
    for i in range(nPks):
        u=u+KVLMOSL(x,As[i],sprimes[i],rho)
    u=u+bgd*x/P
    return u
nPks=2
P=int(max(x_data))
bgd=y_data[-1]
inis=[1,1,3,100,.01]
params, cov = optimize.curve_fit(total_LMOSL,\
x_data, y_data,p0=inis)
plt.scatter(x_data, y_data, label='J1000 feldspar');
plt.plot(x_data, total_LMOSL(x_data, *params),'r-',
         label='KP-LM equation',linewidth=3);
for i in range(0,nPks):
    FOKLMi=KVLMOSL(x_data, params[i], params[i+nPks],
    params[-1]);
    plt.plot(x_data,FOKLMi);
plt.plot(x_data,x_data*bgd/P);
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.ylabel('LM-IRSL signal [a.u.]');
plt.xlabel('Time [s]');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
A1,sprime1,A2,sprime2=[round(params[x],2) for x in range(4)]
rho, drho=f'{params[-1]:.4f}',f'{np.sqrt(cov[4][4]):.4f}'
dA1,dsprime1,dA2,dsprime2=[round(np.sqrt(cov[x][x]),2)\
for x in range(4)]
res=total_LMOSL(x_data, *params)-y_data
FOM=round(100*np.sum(abs(res))/np.sum(y_data),2)
myTable=PrettyTable(["A",'dA', "rho",  "d(rho)",\
"s'(s^-1)","ds","FOM"]);
myTable.add_row([A1,dA1,rho,drho, sprime1, dsprime1,FOM]);
myTable.add_row([A2,dA2,' ',' ', sprime2, dsprime2,' ']);
print(myTable)
```

```
  +------+------+--------+--------+----------+------+------+
  |  A   |  dA  |  rho   | d(rho) | s'(s^-1) |  ds  | FOM  |
```

```
  +------+------+--------+--------+----------+------+------+
  | 1.67 | 0.02 | 0.0127 | 0.0006 |   0.65   | 0.02 | 4.17 |
  | 3.48 | 0.11 |        |        |  59.75   | 5.61 |      |
  +------+------+--------+--------+----------+------+------+

plt.show()
```

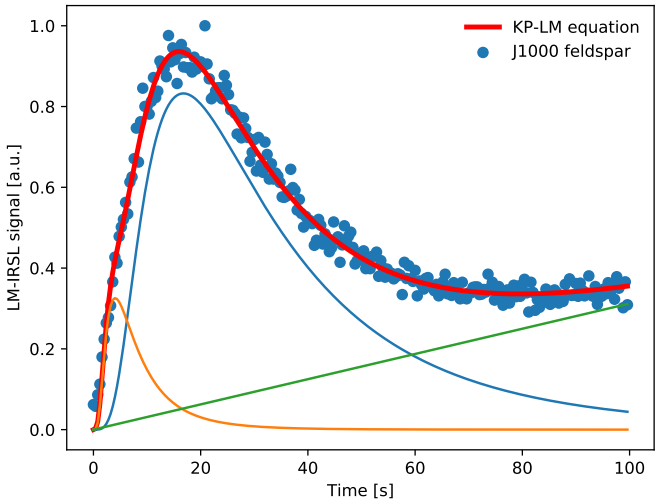

Fig. 11.2: Deconvolution of LM-IRSL signal for sample J1000 based on the KP-LM equation, with two components plus a linearly increasing background. For more details see Pagonis et al. [1]

Code 11.3: J1000 deconvolution with optimal number of peaks N=7

```
# J1000 deconvolution with optimal number of peaks N=7
from scipy import optimize
import numpy as np
import matplotlib.pyplot as plt
from prettytable import PrettyTable
import warnings
warnings.filterwarnings("ignore")
data = np.loadtxt('J1000prompt.txt')
x_data,y_data = data[:, 0], data[:, 1]
z,   kB =1.8, 8.617E-5
```

```
def TL(T, B,En ,rho, s):
    return abs(B)* np.exp(-rho*( (np.log(1+z*s*kB*(((T+\
    273)**2.0)/np.abs(En))*np.exp(-En/(kB*(T+273)))*\
    (1-2*kB*(T+273)/En)))**3.0))*(En**2.0-6*(kB**2.0)*\
     ((T+273)**2.0))*((np.log(1+z*s*kB*(((T+273)**2.0)/\
    abs(En))*np.exp(-En/(kB*(T+273)))*(1-2*kB*(T+273)/\
    En)))**2.0)/(En*kB*s*((T+273)**2)*z-2*(kB**2.0)*\
    s*z*((T+273)**3.0)+np.exp(En/(kB*(T+273)))*En)
def total_TL(T, *inis):
    u=np.array([0 for i in range(len(x_data))])
    Bs, Ens,rho,s=    inis[0:nPks], inis[nPks:2*nPks],inis[-2]\
    ,inis[-1]
    for i in range(nPks):
        u=u+TL(T,Bs[i],Ens[i],rho,s)
    return u
nPks=7
B=[2e17]*7
lowB, highB=[0.01*x for x in B], [50*x for x in B]
lowrho, highrho, rho= [0.003,.015,.008]
lows, highs, s =[1e11, 1e14, 1e12]
En=[.8, .9,1.06,1.19,1.4,1.6,1.7]
lowEn,highEn =[0.9*x for x in En],[1.1*x for x in En]
inis=B+En+[rho]+[s]
lowbnds=lowB+lowEn+[lowrho]+[lows]
highbnds=highB+highEn+[highrho]+[highs]
params, params_covariance = optimize.curve_fit(total_TL,\
x_data,y_data,p0=inis,bounds=(lowbnds,highbnds),maxfev=10000)
plt.scatter(x_data, y_data,c='r',label='Sample J1000');
plt.plot(x_data, total_TL(x_data,
 *params),c='black',label='KP-TL  N='+str(nPks),linewidth=1);
for i in range(0,nPks):
    plt.plot(x_data, TL(x_data, params[i],params[nPks+i],\
    params[-2],params[-1]));
leg = plt.legend();
leg.get_frame().set_linewidth(0.0);
plt.ylabel('TL [a.u.]');
plt.xlabel(r'Temperature T [$^{o}$C]');
res=total_TL(x_data, *params)-y_data
FOM=100*np.sum(abs(res))/np.sum(y_data)
plt.text(90,.2e6,'1');
plt.text(110,1e6,'2');
plt.text(150,1e6,'3');
plt.text(210,.4e6,'4');
plt.text(280,.3e6,'5');
```

```
plt.text(330,.2e6,'6');
plt.text(390,.2e6,'7');
plt.show();
```

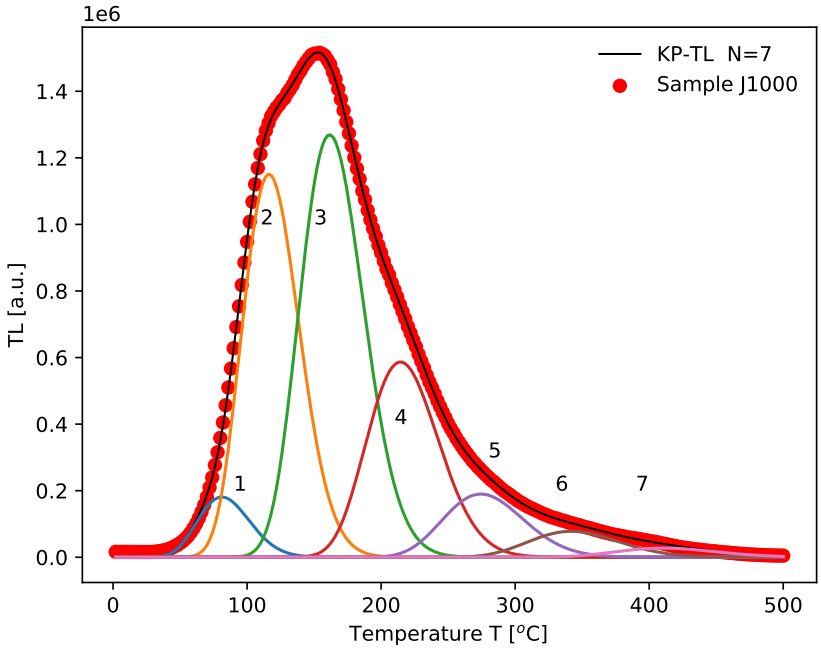

Fig. 11.3: Deconvolution of prompt TL signal from sample J1000 using the original KP-TL equation, with the optimal number of peaks N=7. For more details see Pagonis et al. [1] and Chaps.4-5 of this book.

Code 11.4: Fitting CW-IRSL signal with stretched exponential

```
#Fitting CW-IRSL signal with stretched exponential
#  deconvolution of feldspar CW-IRSL with stretched exponential
from scipy import optimize
import numpy as np
import matplotlib.pyplot as plt
from prettytable import PrettyTable
```

```
import warnings
data = np.loadtxt('KST4ph300IR.txt')
x_data,y_data=data[:,0][0:100], data[:,1][0:100]
y_data=y_data/max(y_data)
def STRETCH(t, A,tau,beta):
    CW=A*np.exp(-(t/tau)**beta)
    return CW
def total_CW(t, *inis):
    u=np.array([0 for i in range(len(x_data))])
    As, taus=    inis[0:nPks], inis[nPks:2*nPks]
    beta=inis[-1]
    for i in range(nPks):
        u=u+STRETCH(t,As[i],taus[i],beta)
    return u
P=int(max(x_data))
t = np.linspace(0, P, P)
nPks=1
inis=[1,5,.1]
params, cov = optimize.curve_fit(total_CW,\
x_data,y_data,p0=inis,maxfev=10000)
plt.scatter(x_data, y_data,c='r',label='KST4 feldsparCW-IRSL ');
plt.plot(x_data, total_CW(x_data,
 *params),c='black',label='Stretched Exponential equation');
for i in range(0,nPks):
    CWi=STRETCH(t, params[i],params[nPks+i],params[-1]);
    plt.plot(t,CWi);
leg = plt.legend();
leg.get_frame().set_linewidth(0.0);
plt.ylabel('CW-IRSL [a.u.]');
plt.xlabel(r'Stimulation time [s]');
res=total_CW(x_data, *params)-y_data
FOM=100*np.sum(abs(res))/np.sum(y_data)
As=[round(x,2) for x in params[0:nPks]]
taus=[round(x,1) for x in params[nPks:2*nPks]]
dAs=[round(np.sqrt(cov[x][x]),2) for x in range(0,nPks)]
dtaus=[round(np.sqrt(cov[x][x]),2) for x in\
range(nPks,2*nPks)]
dbeta=round(np.sqrt(cov[2][2]),2)
beta=round(params[-1],2)
myTable = PrettyTable([ "A (a.u.)","dA",\
'tau (s)','dtau (s)','beta','dbeta'])
myTable.add_row([As[0],dAs[0],taus[0],dtaus[0],beta,dbeta]);
print('FOM=',round(FOM,1),' %')
print(myTable)
```

```
plt.show();
```

```
  FOM= 12.2  %
  +----------+------+---------+----------+------+-------+
  | A (a.u.) |  dA  | tau (s) | dtau (s) | beta | dbeta |
  +----------+------+---------+----------+------+-------+
  |   1.04   | 0.01 |   6.5   |   0.21   | 0.61 |  0.01 |
  +----------+------+---------+----------+------+-------+
```

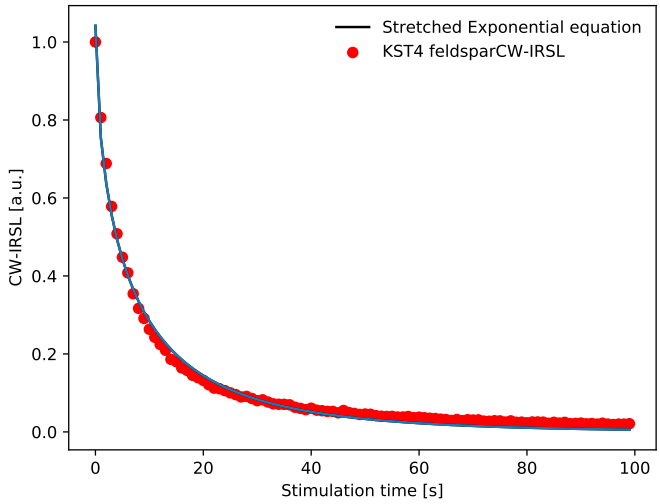

Fig. 11.4: Fitting CW-IRSL signal with the stretched exponential Eq.(**??**). The experimental data is from feldspar sample KST4 (Polymeris et al. [2])

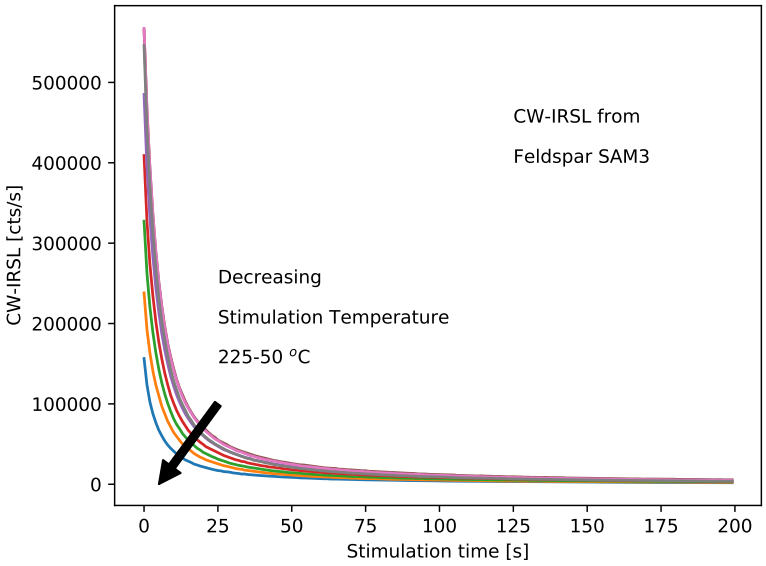

Fig. 11.5: The experimental CW-IRSL curves from feldspar sample SAM3, measured at elevated stimulation temperatures in the range 50-225°C. For experimental details see Şahiner et al. [3]

Code 11.5: Analysis of SAM3 CW-IRSL signals using stretched exponentials

```
#  Analysis of CW-IRSL measured at high stimulation temeprature
from scipy import optimize
import numpy as np
import matplotlib.pyplot as plt
[betas, dbetas, As,dAs,taus,dtaus,maxys,bgds,dbgds]= [[] \
for _ in range(9)]
fils=['irrthenst50.txt','irrthenst75.txt','irrthenst100.txt',
'irrthenst125.txt','irrthenst150.txt','irrthenst175.txt',
'irrthenst200.txt','irrthenst225.txt']
def total_CW(t, *inis):
    [As, taus, betas,bgd]=  inis
    CW=As*np.exp(-(t/abs(taus))**np.abs(betas))+np.abs(bgd)
    return CW
```

```
def findbeta(fil):
    data = np.loadtxt(fil)
    y_data=data
    x_data=np.arange(0,len(y_data))
    maxys.append(max(y_data))
    y_data=y_data/max(y_data)
    inis=[1,5,.1,min(y_data)]
    params, cov = optimize.curve_fit(total_CW,\
    x_data,y_data,p0=inis,maxfev=10000)
    res=total_CW(x_data, *params)-y_data
    FOM=100*np.sum(abs(res))/np.sum(y_data)
    As.append(params[0]);
    taus.append(params[1])
    betas.append(params[2])
    bgds.append(params[3])
    dAs.append(np.sqrt(cov[0][0]))
    dtaus.append(np.sqrt(cov[1][1]))
    dbetas.append(np.sqrt(cov[2][2]))
    dbgds.append(np.sqrt(cov[3][3]))
    return
for j in range(len(fils)):
    findbeta(fils[j])
temps=[50,75,100,125,150,175,200,225]
plt.subplot(2,2,1);
plt.errorbar(temps, maxys, marker='s',yerr=np.sqrt(maxys),\
label=r'Amplitude A$');
plt.ylim(0,6e5);
plt.ylabel(r'Amplitude A$');
plt.xlabel(r'Stimulation T [$^o$C]');
plt.subplot(2,2,2);
plt.errorbar(temps, betas, marker='s',yerr=dbetas,\
label=r'Coefficient $\beta$');
plt.ylim(0.6,.7);
plt.ylabel(r'Coefficient $\beta$');
plt.xlabel(r'Stimulation T [$^o$C]');
leg = plt.legend();
leg.get_frame().set_linewidth(0.0);
plt.subplot(2,2,3);
plt.errorbar(temps, np.abs(taus), marker='s',yerr=dtaus);
plt.ylim(0,8);
plt.ylabel(r'$\tau$ [s]');
plt.xlabel(r'Stimulation T [$^o$C]');
plt.subplot(2,2,4);
plt.errorbar(temps,np.abs(bgds),marker='o',yerr=dbgds);
```

```
plt.ylabel(r'bgd [a.u.]');
plt.xlabel(r'Stimulation T [$^o$C]');
plt.ylim(0,.022);
plt.tight_layout()
plt.show()
```

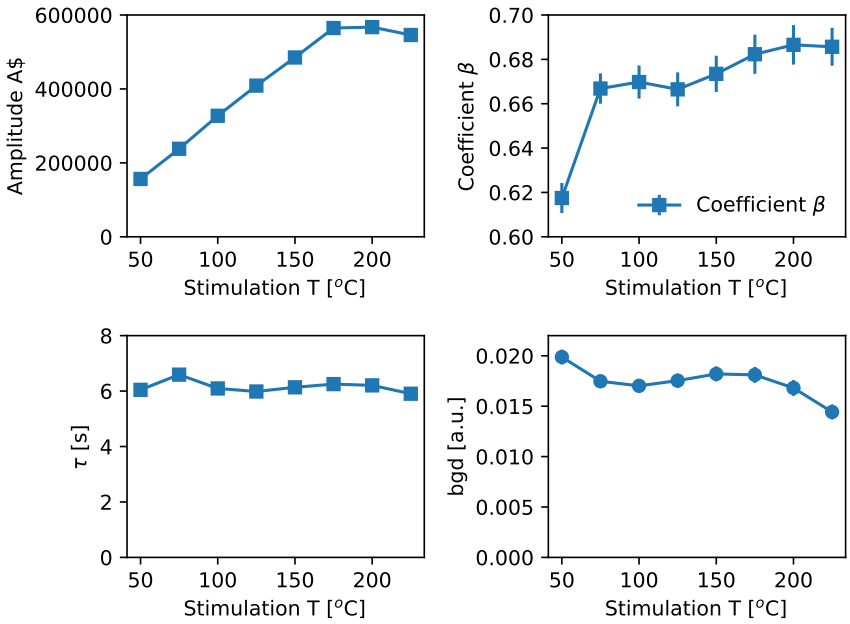

Fig. 11.6: The experimental CW-IRSL curves from Fig.11.5 for feldspar sample SAM3 are analyzed using the stretched exponential function. The data is measured at elevated stimulation temperatures in the range 50-225°C, and is plotted as a function of the stimulation temperature. (a) Amplitude A; (b) coefficient β; (c) decay time τ; (d) constant background component. For experimental details see Şahiner et al. [3].

References

1. V. Pagonis, N. D. Brown, J. Peng, G. Kitis, G. S. Polymeris, On the deconvolution of promptly measured luminescence signals in feldspars, Journal of Luminescence 239 (2021) 118334. doi:https://doi.org/10.1016/j.jlumin.2021.118334.
   URL https://www.sciencedirect.com/science/article/pii/S0022231321004506
2. G. Polymeris, E. Theodosoglou, G. Kitis, N. Tsirliganis, A. Koroneos, K. Paraskevopoulos, Preliminary results on structural state characterization of K-feldspars by using thermoluminescence, Mediterranean Archaeology and Archaeometry 13 (3) (2013) 155–161.
3. E. Şahiner, G. Kitis, V. Pagonis, N. Meriç, G. S. Polymeris, Tunnelling recombination in conventional, post-infrared and post-infrared multi-elevated temperature IRSL signals in microcline K-feldspar, Journal of Luminescence 188 (2017) 514–523.