Chapter 13
TIME-RESOLVED LUMINESCENCE: DATA ANALYSIS

Code 13.1: Analysis of TR-OSL in sedimentary quartz

```
# Analysis of TR-OSL in sedimentary quartz
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from scipy import optimize
from prettytable import PrettyTable
data = np.loadtxt('chithamboTROSLqzdata.txt')
x_data,y_data = data[:, 0][0:27], data[:, 1][0:27]
plt.plot(x_data,y_data,"o");
def SEfit(x_data,N,Dc):
    u=N*(1-np.exp(-x_data/np.abs(Dc)))
    return u
init_vals=[200,40]
params, cov = optimize.curve_fit(SEfit,\
x_data, y_data,p0=init_vals)
dN = round(np.sqrt(cov[0][0]),1)
dDo = round(np.sqrt(cov[1][1]),1)
x_vals=np.arange(0,60,1)
plt.plot(x_vals, SEfit(x_vals, *params[0:2]),c="b",\
label='SE fit');
plt.scatter(x_data, y_data, label='Experiment');
leg = plt.legend()
```

```
leg.get_frame().set_linewidth(0.0)
plt.ylim(0,260);
plt.ylabel('TR-OSL [a.u.]');
plt.xlabel('Time [$\mu$s]');
plt.title('TR-OSL sedimentary quartz');
plt.text(200,150,'470 nm LEDs');
plt.tight_layout()
myTable = PrettyTable(["N", "dN","tau (microsec.)",\
"d(tau) (microsec.)"])
myTable.add_row([round(params[0],1),dN,round(params[1],1),dDo])
x_data,y_data = data[:, 0][28:99], data[:, 1][28:99]
plt.plot(x_data,y_data,"o");
def decayfit(x_data,N,Dc):
    u=N*(np.exp(-x_data/np.abs(Dc)))
    return u
init_vals=[200,40]
params, cov = optimize.curve_fit(decayfit,\
x_data, y_data,p0=init_vals)
dN = round(np.sqrt(cov[0][0]),1)
dDo = round(np.sqrt(cov[1][1]),1)
x_vals=np.arange(60,300,1)
plt.plot(x_vals, decayfit(x_vals, *params[0:2]),c="b");
plt.scatter(x_data, y_data);
plt.tight_layout()
myTable.add_row([round(params[0],1),dN,round(params[1],1),dDo])
print(myTable)
plt.show()
```

```
  +--------+------+-----------------+--------------------+
  |   N    |  dN  | tau (microsec.) | d(tau) (microsec.) |
  +--------+------+-----------------+--------------------+
  | 235.9  | 14.2 |       33.6      |        4.0         |
  | 1099.0 | 98.2 |       34.9      |        1.3         |
  +--------+------+-----------------+--------------------+
```

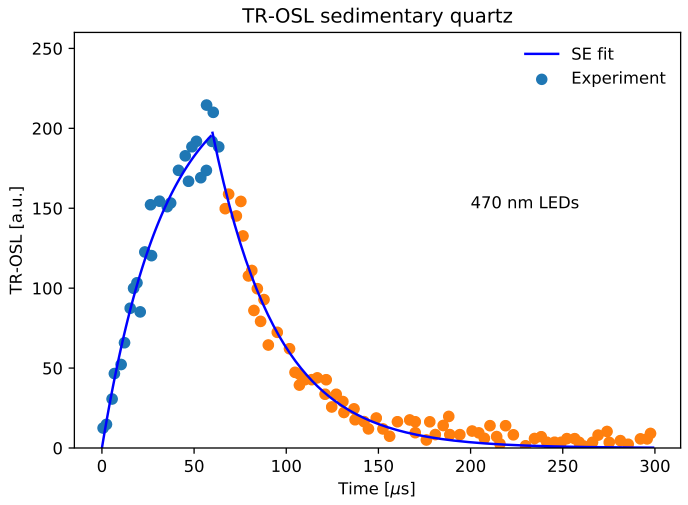

Fig. 13.1: Examples of TR-OSL curves for sedimentary quarts with 60 μs pulse, fitted with the FOK-TR equations. For more details see Chithambo et al. [1].

Code 13.2: The effect of thermal quenching on TL glow curve for quartz

```
#The effect of thermal quenching on TL glow curve for quartz
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from scipy import stats
def fn1():
    for i in range(len(files)):
        data = np.loadtxt(files[i])
        x_data,y_data = data[:, 0], data[:, 1]
        plt.plot(x_data,y_data,'o-');
        area[i]=np.sum([x_data[u]*y_data[u] for u in \
range(len(x_data))])
        Tmax[i]=x_data[np.argmax(y_data)]
def fn2():
    for i in range(len(files)):
        data = np.loadtxt(files[i])
        x_data,y_data = data[:, 0][lowpts[i]:hipts[i]],\
data[:, 1][lowpts[i]:hipts[i]]
        x_data=[1/(8.617e-5*u) for u in np.array(x_data)]
        y_data=np.log(y_data)
        plt.plot(x_data,y_data);
        slope, intercept,r_value,p_value, std_err=\
stats.linregress(x_data,y_data)
        fit=[x*slope+intercept for x in x_data]
        plt.plot(x_data,fit,'o-');
        slope,std_err= np.round(slope,2), np.round(std_err,2)
        Evalues[i]=round(slope,3)
        dEvalues[i]=round(std_err,3)
files=('B2qz025.txt','B2qz05.txt','B2qz2.txt','B2qz5.txt',\
'B2qz10.txt')
plt.subplot(221);
plt.title('(a)');
plt.ylabel('TL signal [a.u.]');
plt.xlabel(r'Temperature [K]');
plt.ylim(0,22);
plt.text(350,17,'0.25-10 K/s');
area=[0 for x in range(len(files))]
Tmax=[0 for x in range(len(files))]
fn1()
plt.subplot(222);
plt.title('(b)  Initial rise');
plt.ylabel('TL [a.u.]');
plt.xlabel(r'Temperature [K]');
Evalues=[0 for x in range(len(files))]
Rvalues=[0 for x in range(len(files))]
dEvalues=[0 for x in range(len(files))]
lowpts=[3 for x in range(len(files))]
hipts=[9 for x in range(len(files))]
fn2()
plt.subplot(223);
plt.plot(Tmax,np.array(area)/1e5,'o');
plt.xlabel(r'T$_{max}$ [K]');
plt.ylabel(r'Area of TL peak [a.u.]');
plt.text(330,.2,"Thermal quenching");
plt.text(330,.08,"of TL area");
plt.title('(c)');
plt.ylim(0,1);
plt.subplot(224);
plt.ylim(0,1.2);
plt.title('(d)');
plt.text(330,.3,"Thermal quenching");
plt.text(330,.15,"of Energy E");
plt.plot(Tmax,-np.array(Evalues),'r^');
plt.xlabel(r'T$_{max}$ [K]');
plt.ylabel(r'Energy E [eV]');
plt.tight_layout()
plt.show()
```

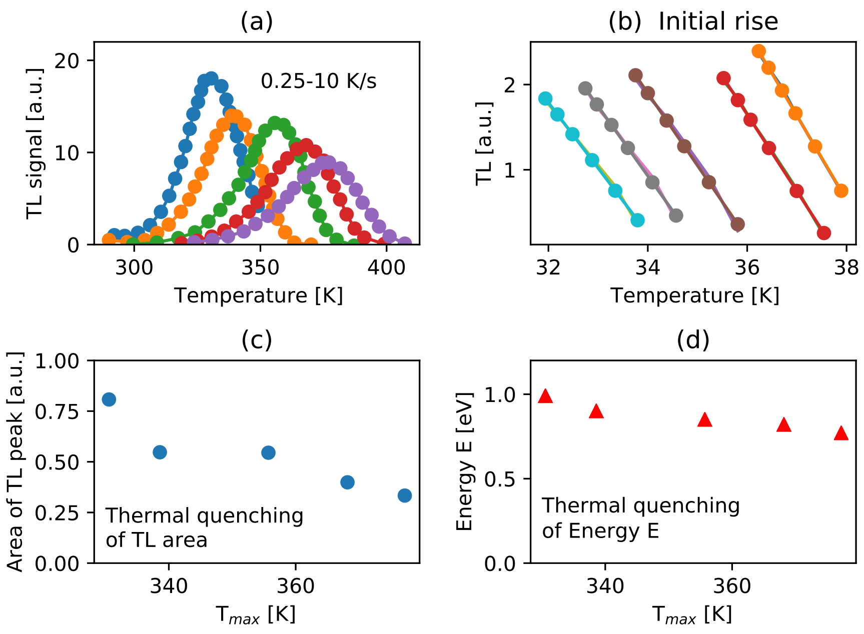

Fig. 13.2: The effect of thermal quenching on the TL glow curve of quartz, measured with several heating rates in the range $\beta = 0.25 - 10$ K/s. (a) As the heating rate increases, the TL peak shifts to the right towards higher temperatures, and the intensity decreases. (b) The Arrhenius plots for the initial rise analysis of the data in (a). (c) The area under the curves also decreases as the heating rate increases, due to thermal quenching. (d) The E value obtained from the initial rise method in (b), also decreases as β increases, due to thermal quenching. For more details see Subedi et al. [2].

Code 13.3: Analysis of TR-OSL experimental data in alumina

```
# Analysis of TR-OSL experimental data in alumina
from scipy import optimize
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from prettytable import PrettyTable
data = np.loadtxt('aluminaxy1.txt')
x_data, y_data = np.array(data[:, 0][8:20]), \
np.array(data[:, 1][8:20])
kB=8.617e-5
def tq(x_data, N,c, W):
    return N /(1+c*np.exp(-W/(kB*(273+x_data))))
params, cov = optimize.curve_fit(tq,\
x_data, y_data,bounds=(0,[.04,1e12,1]))
plt.subplot(1,2, 1);
plt.plot(x_data,y_data,"o",c='black',label='Experimental');
plt.plot(x_data, tq(x_data, *params),
label='Best fit',c='r',linewidth=2);
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.ylabel(r'Luminescence Lifetime $\tau$ [s]');
plt.xlabel(r'Stimulation Temperature [$^{o}$C]');
plt.title('(a)   Finding C,W');
plt.text(90,.02,"Luminescence");
plt.text(90,.017,r"lifetime $\tau$");
C,W=format(params[1],"10.1E"),round(params[2],4)
dC = round(np.sqrt(cov[1][1]),4)
dW = round(np.sqrt(cov[2][2]),3)
data = np.loadtxt('aluminaxy2.txt')
x0, y0 = np.array(data[:, 0]), np.array(data[:, 1])
x_data=1/(kB*(273+np.array(x0)[4:10]))
y_data=np.log(np.array(y0)[4:10])
def arrh(x_data,slope,inter):
    return slope*x_data+inter
params, cov = optimize.curve_fit(arrh,x_data, y_data)
plt.subplot(1,2, 2);
plt.plot(x_data,y_data,"o",c="black",label='Experimental');
x_vals=np.arange(30,35,.1)
plt.plot(x_vals, arrh(x_vals, *params),
label='Best fit',c='r',linewidth=2);
```

```
Eth=-round(params[0],3)
dEth = round(np.sqrt(cov[0][0]),3)
plt.title(r'(b)    Finding E$_{th}$');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.ylabel(r'ln($\tau$)');
plt.xlabel(r'1/(kT) [eV$^{-1}$]');
plt.text(31, 14.95,'Arrhenius plot');
plt.tight_layout()
myTable=PrettyTable(["C","dC","W (eV)","dW", "Eth (eV)","dEth"]);
myTable.add_row([C,dC,W,dW,Eth,dEth]);
print(myTable)
plt.show()

   +------------+-----+--------+-------+----------+-------+
   |     C      |  dC | W (eV) |   dW  | Eth (eV) |  dEth |
   +------------+-----+--------+-------+----------+-------+
   |    3.9E+11 | 0.0 |  1.0   | 0.001 |   0.07   | 0.002 |
   +------------+-----+--------+-------+----------+-------+
```

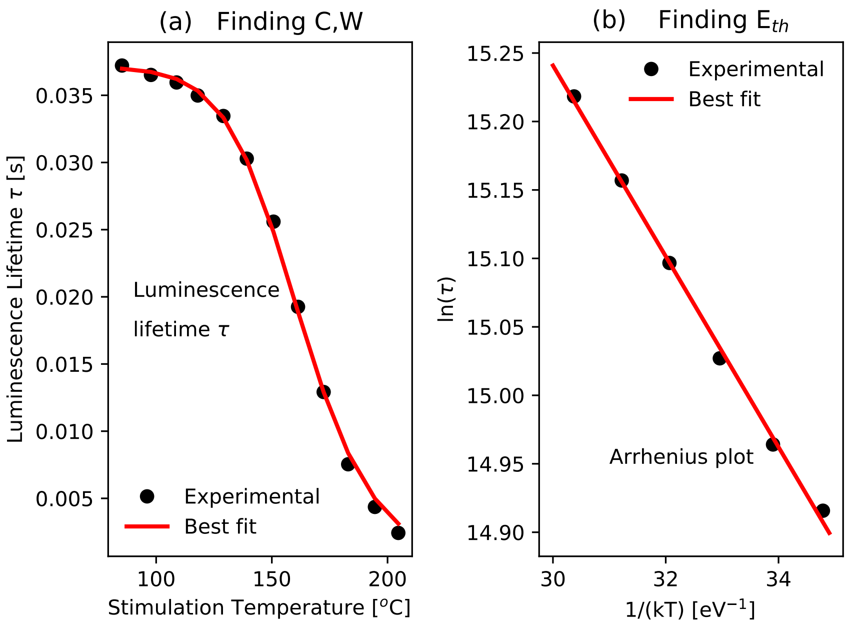

Fig. 13.3: Experimental determination of the thermal quenching parameters C, W, E_{th}. (a) The values of C, W are obtained by fitting the *decreasing* part of the data in Fig.**??**b using Eq.(**??**). The best fit is shown by the solid line. (b) The thermal activation energy E_{th} is obtained with an Arrhenius analysis of the *increasing* part of the data in Fig.**??**a using Eq.(**??**). For more details see Pagonis et al. [3].

Code 13.4: Fit microcline TR-IRSL data with analytical equation

```
# Analysis of TR-OSL in microcline feldspar quartz
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from scipy import optimize
from prettytable import PrettyTable
data = np.loadtxt('FL1ONdata.txt')
z=1.8
x_data, y_data = np.array(data[:, 0]), np.array(data[:, 1])
def TRfit(x_data, imax, rho, s):
    u=imax*(1-np.exp (-rho*(np.log(1 +z* s* x_data)) **3.0))
    return u
init_vals=[1,.007,10]
params, cov = optimize.curve_fit(TRfit, x_data, y_data, \
p0=init_vals)
plt.subplot(1,2, 1);
x_vals=np.arange(0,50,1)
plt.plot(x_vals, TRfit(x_vals, *params),c="black",\
label='Analytical fit');
plt.scatter(x_data, y_data, c="g",label='Experiment');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.ylim(0,1.2);
plt.ylabel('TR-IRSL [a.u.]');
plt.xlabel('Time [$\mu$s]');
plt.title('(a) LED ON period');
imax=round(params[0],1)
rho=round(params[1],3)
```

```
s=params[2]
x_vals=np.arange(0,105,1)
data = np.loadtxt('FL1OFFdata.txt')
x_data, y_data = np.array(data[:, 0]), np.array(data[:, 1])
u=imax*((np.exp (-rho*(np.log(1 +z* s* x_vals)) **3.0))-\
(np.exp (-rho*(np.log(1 +z* s* (x_vals+50)) **3.0))))+.02
plt.subplot(1,2, 2);
plt.plot(x_vals, u,c="green",\
label='Analytical fit');
plt.scatter(x_data, y_data, c='r',label='Experiment');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.ylim(0,1.2);
plt.ylabel('TR-IRSL [a.u.]');
plt.xlabel('Time [$\mu$s]');
plt.title('(b) LED OFF period');
plt.text(20,0.5,'TR-IRSL microcline');
plt.tight_layout()
s=format(params[2]*1e6,'10.1E')
ds = format(np.sqrt(cov[2][2])*1e6,'10.1E')
drho = round(np.sqrt(cov[1][1]),3)
myTable=PrettyTable(["LED","imax", "rho",  "d(rho)",\
"s (s^-1)","ds (s^-1)"]);
myTable.add_row(["ON",imax,rho,drho, s, ds]);
print(myTable)
plt.show()
```

```
+-----+------+-------+--------+------------+------------+
| LED | imax |  rho  | d(rho) |  s (s^-1)  | ds (s^-1)  |
+-----+------+-------+--------+------------+------------+
|  ON | 1.0  | 0.025 | 0.002  |    2.8E+06 |    2.2E+05 |
+-----+------+-------+--------+------------+------------+
```

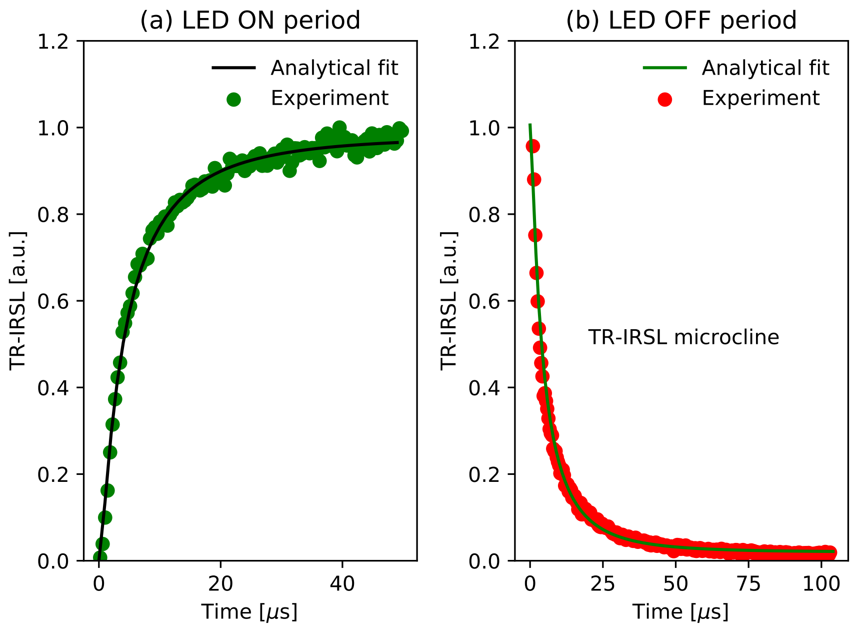

Fig. 13.4: Fit TR-IRSL data for microcline sample FL1, with the analytical Eqs.(**??**) and (**??**). (a) The IR excitation is ON (b) The excitation is OFF. For more details see Pagonis et al. [4].

Code 13.5: Fit TR-IRSL data with exponential plus stretched exponential

```
#Fitting CW-IRSL signal with stretched exponential plus exponential
from scipy import optimize
import numpy as np
import matplotlib.pyplot as plt
from prettytable import PrettyTable
from scipy.special import lambertw
import warnings
data = np.loadtxt('t100.txt')
```

```
x_data,y_data=data[:,0], data[:,1]
y_data=y_data/max(y_data)
def STRETCH(t, A,tau,beta,tau2):
    CW=A*np.exp(-(t/tau)**beta)+ (1-A)*np.exp(-t/tau2)
    return CW
def total_CW(t, *inis):
    u=np.array([0 for i in range(len(x_data))])
    As, taus,betas,taus2=  inis
    for i in range(nPks):
        u=u+STRETCH(t,As,taus,betas,taus2)
    return u
P=int(max(x_data))
t = np.linspace(0, P, P)
nPks=1
inis=[.5,10,.5,100]
params, cov = optimize.curve_fit(total_CW,\
x_data,y_data,p0=inis,maxfev=10000)
plt.scatter(x_data, y_data,c='r',label='FL1 feldspar TR-IRSL ');
plt.plot(x_data, total_CW(x_data,
 *params),c='black',label='Analytical equation');
for i in range(0,nPks):
    CWi=STRETCH(t, *params);
    plt.plot(t,CWi);
leg = plt.legend();
leg.get_frame().set_linewidth(0.0);
plt.ylabel('CW-IRSL [a.u.]');
plt.xlabel('Time [$\mu$s]');
res=total_CW(x_data, *params)-y_data
FOM=100*np.sum(abs(res))/np.sum(y_data)
print('FOM=',round(FOM,1),' %')
As=round(params[0],2)
As2=round(params[3],4)
taus=round(params[1],1)
taus2=int(params[3])
dAs=round(np.sqrt(cov[0][0]),4)
dtaus=round(np.sqrt(cov[1][1]),2)
dbeta=round(np.sqrt(cov[2][2]),3)
beta=round(params[2],3)
dtaus2=int(np.sqrt(cov[3][3]))
myTable = PrettyTable([ "A (a.u.)",\
'tau (s)','dtau','beta','dbeta','tau2','dtau2'])
myTable.add_row([As,taus,dtaus,beta,dbeta,taus2,dtaus2]);
print(myTable)
plt.show();
```

```
FOM= 17.9  %
+----------+---------+------+-------+-------+------+-------+
| A (a.u.) | tau (s) | dtau |  beta | dbeta | tau2 | dtau2 |
+----------+---------+------+-------+-------+------+-------+
|   0.98   |   2.5   | 0.01 | 0.573 | 0.002 | 376  |   7   |
+----------+---------+------+-------+-------+------+-------+
```

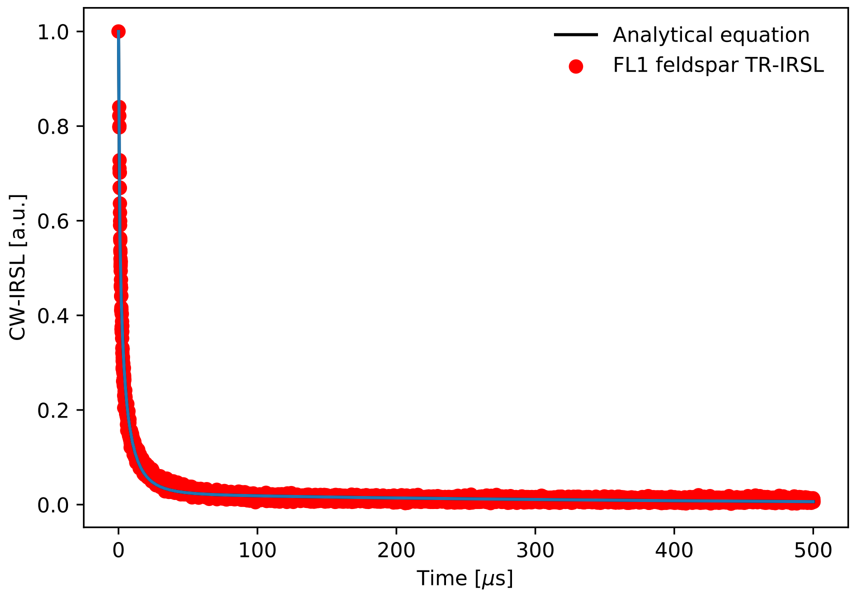

Fig. 13.5: Fit of the TR-IRSL signal during the relaxation period of a pulse, for sample FL1 studied by Pagonis et al. [5]. The data is fitted using the linear combination of a decaying exponential plus a stretched exponential function.

References

1. M. L. Chithambo, C. Ankjærgaard, V. Pagonis, Time-resolved luminescence from quartz: An overview of contemporary developments and applications, Physica B: Condensed Matter 481 (2016) 8–18.
2. B. Subedi, E. Oniya, G. S. Polymeris, D. Afouxenidis, N. C. Tsirliganis, G. Kitis, Thermal quenching of thermoluminescence in quartz samples of various origin, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 269 (6) (2011) 572–581.
3. V. Pagonis, C. Ankjærgaard, M. Jain, R. Chen, Thermal dependence of time-resolved blue light stimulated luminescence in Al_2O_3:C, Journal of Luminescence 136 (2013) 270–277.
4. V. Pagonis, C. Ankjærgaard, M. Jain, M. L. Chithambo, Quantitative analysis of time-resolved infrared stimulated luminescence in feldspars, Physica B: Condensed Matter 497 (2016) 78–85.
5. V. Pagonis, P. Morthekai, A. K. Singhvi, J. Thomas, V. Balaram, G. Kitis, R. Chen, Time-resolved infrared stimulated luminescence signals in feldspars: Analysis based on exponential and stretched exponential functions, Journal of Luminescence 132 (9) (2012) 2330–2340.