Chapter 5
TL FROM QUANTUM TUNNELING PROCESSES: DATA ANALYSIS

Code 5.1: Anomalous fading (AF) and the g-factor

```
# Anomalous fading (AF) and calculating  the g-factor
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from scipy.optimize import curve_fit
from prettytable import PrettyTable
def linearFunc(x2,intercept,slope):
    y2 = intercept + slope * x2
    return y2
data = np.loadtxt('durnago0sok.txt')
x0, y0 = np.array(data[:, 0]), np.array(data[:, 1])
data = np.loadtxt('durango10daysok.txt')
x1, y1 = np.array(data[:, 0]), np.array(data[:, 1])
data = np.loadtxt('DurangoAFdataok.txt')
x2, y2 = np.array(data[:, 0]), np.array(data[:, 1])
plt.subplot(1,2, 1);
plt.plot(x0,y0,'b^-',label='t=0s');
plt.plot(x1,y1,'ro-',label='t=10 days');
plt.title('(a)');
plt.ylim(0,1.1e5);
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.ylabel('Remnant TL signal [a.u.]');
plt.xlabel(r'Temperature [$^{o}$C]');
plt.text(100, 6e4,'Durango');
```

```
plt.text(100,5e4,'apatite');
plt.subplot(1,2, 2);
plt.plot(x2,y2,"o",c="r");
plt.ylabel('Remnant TL signal [a.u.]');
plt.xlabel(r'ln(time)');
plt.title('(b)');
plt.text(5,.9,"g-factor");
plt.text(5,.85,r"analysis");
plt.text(5,.76,r"g=21.0%");
params,cov=curve_fit(linearFunc,x2,y2)
plt.plot(x2,linearFunc(x2,*params),label='Fit');
inter,slope = np.round(params,4)
d_inter,d_slope = np.round(np.sqrt(np.diag(cov)),3)
residuals = y2- linearFunc(x2, *params)
ss1=np.sum(residuals**2.0)
ss2 = np.sum((y2-np.mean(y2))**2.0)
rsqr=np.round(1-ss1/ss2,3)
g=np.round(-230.2*slope)
myTable=PrettyTable(['g (%)','slope (a.u.)','dslope (a.u.)',\
'R^2'])
myTable.add_row([g,-slope, d_slope, rsqr]);
print(myTable)
plt.tight_layout()
plt.show()
```

```
  +-------+--------------+---------------+-------+
  | g (%) | slope (a.u.) | dslope (a.u.) |  R^2  |
  +-------+--------------+---------------+-------+
  |  21.0 |    0.0892    |     0.003     | 0.989 |
  +-------+--------------+---------------+-------+
```

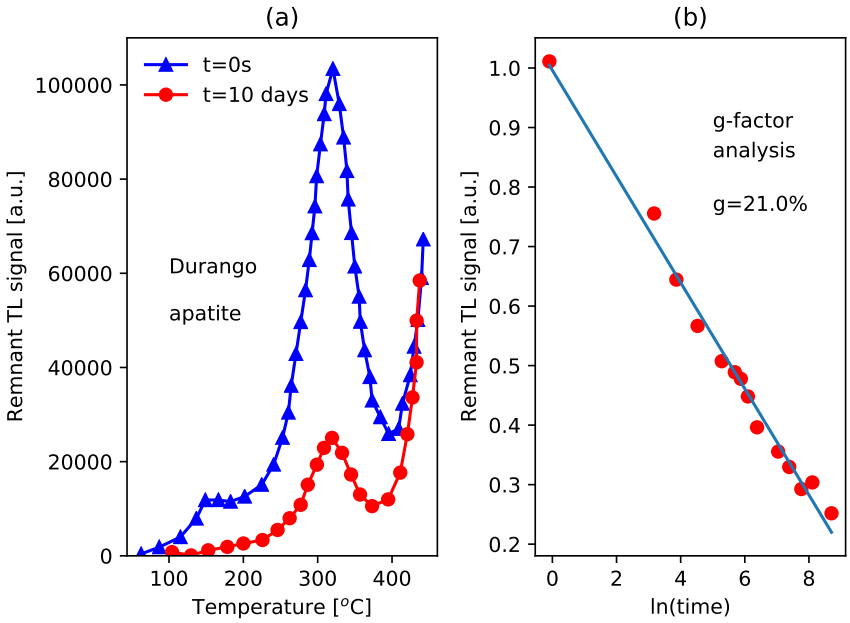

Fig. 5.1: Anomalous fading effect in Durango apatite. (a) The TL signal is measured immediately after irradiation, and also after 10 days have elapsed with the irradiated sample stored at room temperature. (b) Analysis of the remnant TL signal obtained as in (a), to obtain the g-factor for this materials. For more details, see Polymeris et al. [**?**].

Code 5.2: Fit MBO data with KP-TL equation

```
# Fit TL with KP-TL (Kitis-Pagonis) analytical equation
from scipy import optimize
from sympy import *
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from prettytable import PrettyTable
data = np.loadtxt('MBO6gynew.txt')
x_data,y_data = data[:, 0], data[:, 1]
z,   kB,      En=\
```

```
1.8, 8.617E-5,1.0
def test_func(x, B,rho, s):
    return B* np.exp(-rho*( (np.log(1+z*s*kB*(((x+\
    273)**2.0)/np.abs(En))*np.exp(-En/(kB*(x+273)))*\
    (1-2*kB*(x+273)/En)))**3.0))*(En**2.0-6*(kB**2.0)*\
    ((x+273)**2.0))*((np.log(1+z*s*kB*(((x+273)**2.0)/\
    abs(En))*np.exp(-En/(kB*(x+273)))*(1-2*kB*(x+273)/\
    En)))**2.0)/(En*kB*s*((x+273)**2)*z-2*(kB**2.0)*\
    s*z*((x+273)**3.0)+np.exp(En/(kB*(x+273)))*En)
params, cov = optimize.curve_fit(test_func,\
x_data, y_data,bounds=(0,[1e20,.02,1e14]))
drho= round(np.sqrt(cov[1][1]),5)
plt.subplot(2,1, 1);
plt.plot(x_data, y_data,'o', c='lightgreen',label='Experiment');
plt.plot(x_data, test_func(x_data, *params),
label='KP-TL equation',   c='black',linewidth=2);
plt.title('(a)');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.ylabel('TL signal [a.u.]');
plt.xlabel(r'Temperature T [$^{o}$C]');
#plt.text(400, 1.8e4,'KST4 feldspar');
plt.subplot(2,1, 2);
plt.plot(x_data,test_func(x_data, *params)-y_data,'o',\
label='Residuals');
plt.title('(b)');
plt.ylim(-1,1);
plt.ylabel('Residuals');
plt.xlabel(r'Temperature T [$^{o}$C]');
plt.tight_layout()
B,rho, s=format(params[0],"10.2E"),round(params[1],5),\
format(round(params[2],2),"10.2E")
res=test_func(x_data, *params)-y_data
FOM=round(100*np.sum(abs(res))/np.sum(y_data),2)
myTable=PrettyTable(["B", "rho",  "d(rho)",\
"s(s^-1)","FOM"]);
myTable.add_row([B,rho,drho, s,FOM]);
print(myTable)
```

```
  +------------+--------+---------+------------+------+
  |     B      |  rho   |  d(rho) |  s(s^-1)   | FOM  |
  +------------+--------+---------+------------+------+
  |   1.99E+12 | 0.0111 | 0.00028 |   7.53E+10 | 4.09 |
  +------------+--------+---------+------------+------+
```

```
plt.show()
```

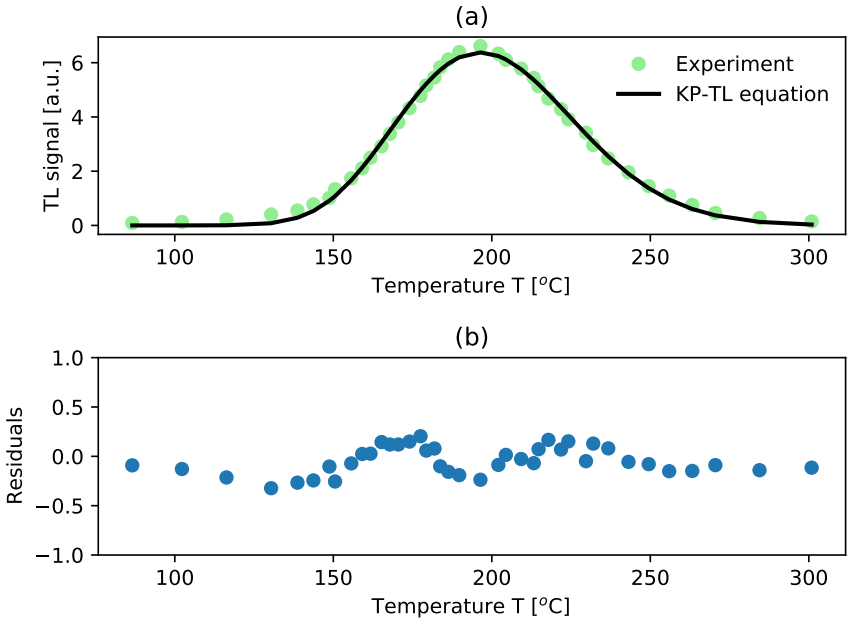

Fig. 5.2: Experimental TL glow curves from freshly irradiated Mg_4BO_7:Dy,Na sample, fitted using the KP-TL analytical Eq.(**??**). For more detailed modeling of experiments on this material, see Pagonis et al. [**?**].

Code 5.3: Deconvolution of 5-peak glow curve for BAL21 sample

```
#Deconvolution of 5-peak glow curve for SAM3 sample
# using the  KP-TL equation
from scipy import optimize
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from prettytable import PrettyTable
import warnings
warnings.filterwarnings("ignore")
data = np.loadtxt('initialsensSAM3.txt')
x_data,y_data = data[:, 0], data[:, 1]
```

```
kB=8.617E-5
nPks=5
z,   kB,     s, =1.8, 8.617E-5,1e12
def TL(T, B,En ,rho):
    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, rho=    inis[0:nPks],inis[-1]
    for i in range(nPks):
        u=u+TL(T,Bs[i],Ens[i],rho)
    return u
inis=(5e16,9e16,10e16,1e16,.1e16,.004)
Ens=[.82, .95,1.06,1.19,1.355]
params, cov = optimize.curve_fit(total_TL,\
x_data,y_data,p0=inis)
plt.subplot(2,1, 1);
plt.scatter(x_data, y_data, label='SAM3 feldspar data');
plt.plot(x_data, total_TL(x_data,
*params),c='r',linewidth=3, label='Analytical KV-TL');
plt.plot(x_data, TL(x_data, params[0],Ens[0],params[5]));
for i in range(0,nPks):
     plt.plot(x_data, TL(x_data, params[i],Ens[i],params[5]));
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.ylabel('TL signal [a.u.]');
plt.xlabel(r'Temperature T [K]');
plt.subplot(2,1, 2);
plt.scatter(x_data,total_TL(x_data, *params)
-y_data,c='r',linewidth=2,label='Residuals');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.ylabel('Residuals');
plt.xlabel(r'Temperature T [K]');
plt.ylim(-5e4,5e4);
plt.tight_layout()
res=total_TL(x_data, *params)-y_data
rho=round(params[5],4)
drho=round(np.sqrt(cov[5][5]),4)
```

```
FOM=round(100*np.sum(abs(res))/np.sum(y_data),2)
myTable = PrettyTable(["rho","drho","FOM (%)"])
myTable.add_row([rho,drho,FOM])
print(myTable)
plt.show()
```

```
+--------+--------+---------+
|  rho   |  drho  | FOM (%) |
+--------+--------+---------+
| 0.0064 | 0.0002 |   3.26  |
+--------+--------+---------+
```

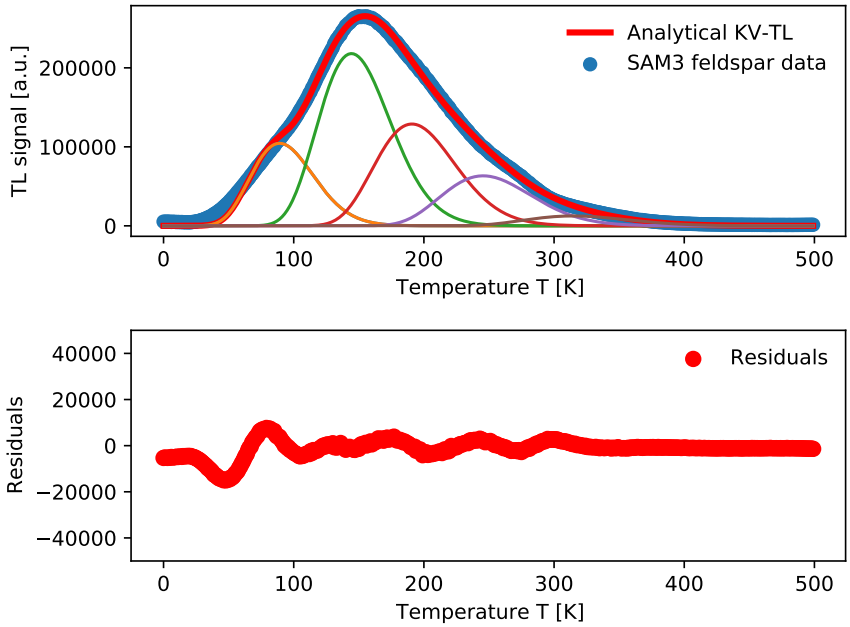

Fig. 5.3: Experimental TL glow curve from freshly irradiated feldspar SAM3 sample, fitted using 5 peaks and the KP-TL analytical Eq.(**??**). For additional modeling of TL experiments on this and other feldspars, see Pagonis et al. [**?**, **?**].

Code 5.4: Deconvolution of MBO data with transformed KP-TL equation

```
# Deconvolution of MBO data with transformed KP-TL equation
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('MBO6gynew.txt')
x_data,y_data = data[:, 0]+273.15, data[:, 1]
z,   kB,      En=\
1.8, 8.617E-5,1.0
Tmax=x_data[np.argmax(y_data)]
imax=max(y_data)
def test_func(T, rho):
    fm=4.90537*rho**1.21038
    Fm=np.log(1+(1/fm)*(1-2*kB*T/En))
    F=np.log(1+(1/fm)*((T/Tmax)**2.0)*np.exp(-En*(Tmax-T)/\
    (kB*T*Tmax))*(1-2*kB*T/En))
    return imax*np.exp(-En*(Tmax-T)/(kB*T*Tmax))*((F/Fm)**2.0)*\
    np.exp(-rho*(F**3.0)-F)/np.exp(-rho*(Fm**3.0)-Fm)
params, cov = optimize.curve_fit(test_func,\
x_data, y_data,bounds=(0.002,.02))
drho= round(np.sqrt(cov[0][0]),5)
plt.subplot(2,1, 1);
plt.plot(x_data, y_data,'o', c='lightgreen',label='Experiment');
plt.plot(x_data, test_func(x_data, *params),label=\
'KP-TL equation',c='black',linewidth=2);
plt.title('(a)');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.ylabel('TL signal [a.u.]');
plt.xlabel(r'Temperature T [$^{o}$C]');
#plt.text(400, 1.8e4,'KST4 feldspar');
plt.subplot(2,1, 2);
plt.plot(x_data,test_func(x_data, *params)-y_data,'o',\
label='Residuals');
plt.title('(b)');
plt.ylim(-1,1);
plt.ylabel('Residuals');
plt.xlabel(r'Temperature T [$^{o}$C]');
plt.tight_layout()
rho=round(params[0],4)
res=test_func(x_data, *params)-y_data
FOM=round(100*np.sum(abs(res))/np.sum(y_data),2)
```

```
myTable=PrettyTable([ "rho",  "d(rho)","FOM"]);
myTable.add_row([rho,drho, FOM]);
#print(myTable)
plt.show()
```

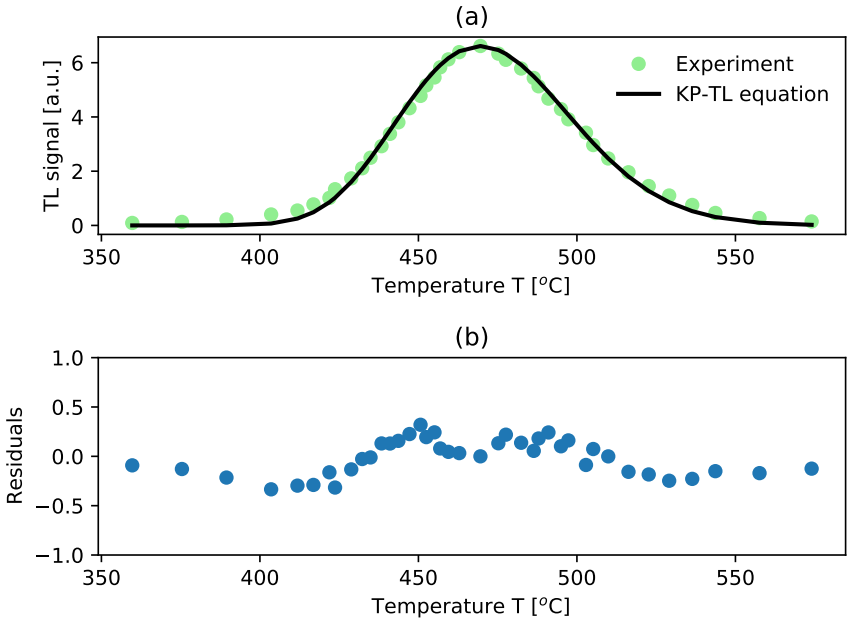

Fig. 5.4: Experimental TL glow curve from freshly irradiated Mg_4BO_7:Dy,Na (MBO) sample, fitted using a single peak with the transformed KP-TL analytical Eqs.(**??**)-(**??**). For additional modeling of experiments on this material, see Pagonis et al. [**?**].

Code 5.5: Finding the optimal number of components to fit the glow curves

```
# Finding the optimal number of components to fit
# the  glow curve for sample J1000
from scipy import optimize
import numpy as np
import matplotlib.pyplot as plt
from prettytable import PrettyTable
```

```
import warnings
warnings.filterwarnings("ignore")
#Deconvolution of prompt TL with N=5-6-7 components
#using original KP-TL
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
def main(nPks):
    Bs=[2e17]*7
    B=Bs[0:nPks]
    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]
    Ens=[.8, .9,1.06,1.19,1.4,1.6,1.7,1.8]
    En=Ens[0:nPks]
    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.subplot(2,2, nPks-4);
    plt.scatter(x_data, y_data,c='r');
    plt.plot(x_data, total_TL(x_data,
     *params),c='black',label='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)
    FOMS[nPks-5]=FOM
FOMS=[0]*3
for nPks in range(7,4,-1):
    main(nPks)
plt.subplot(2,2, 4);
plt.plot(range(5,8),FOMS,'o-');
plt.ylabel('FOM [%]');
plt.xlabel(r'# of components N');
plt.ylim([0, 5]);
my_xticks = [5,6,7]
plt.xticks(range(5,8), my_xticks);
plt.tight_layout()
plt.show()
```

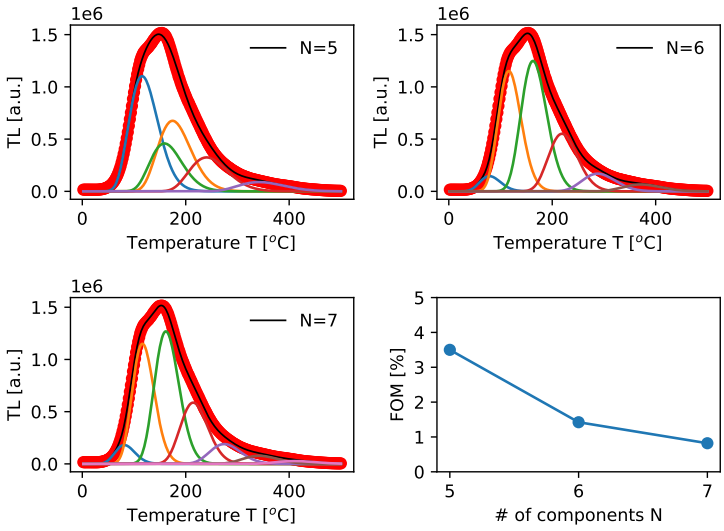

Fig. 5.5: (a) Deconvolution of prompt TL signal from sample J1000, using N=5-6-7 components with the original KP-TL equation; (d) The FOM values from (a)-(c) are plotted as a function of the number of components N, showing a minimum FOM at N=7 [**?**].

Code 5.6: 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='Granitic gneiss');
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();
```

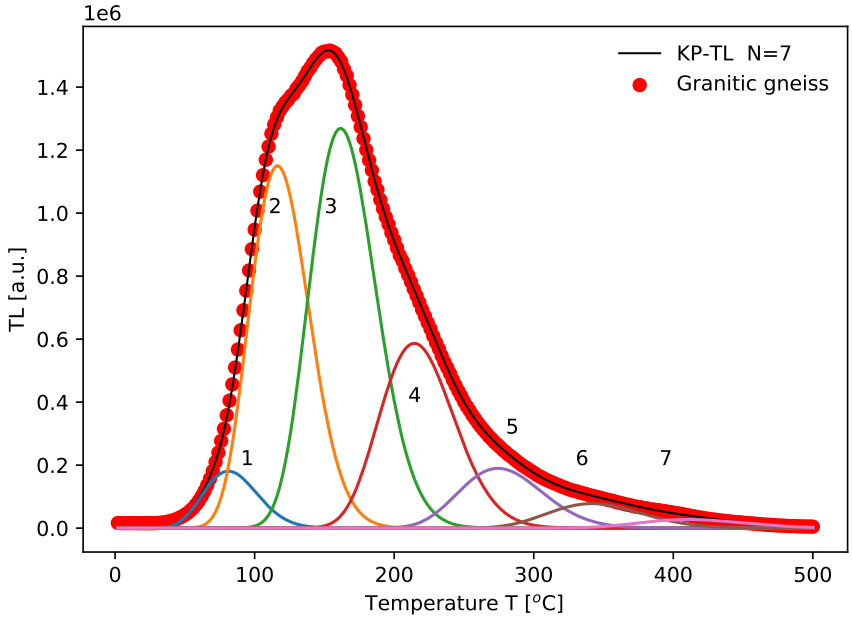

Fig. 5.6: Deconvolution of the prompt TL signal from sample J1000, using the original KP-TL equation, and with the optimal number of peaks N=7.