Chapter 3

TL SIGNALS FROM DELOCALIZED TRANSITIONS: DATA ANALYSIS

Code 3.1: Deconvolution of Glocanin TL with FOK-TL eqt

```
# Deconvolution of Glocanin TL with the original FOK-TL equation
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('glocanin1.txt')
x_data,y_data = data[:, 0], data[:, 1]/max(data[:, 1])
kB,  beta= 8.617e-5,  1
# function for evaluating the FOK-TL (R-W) equation
def TLFOK(T,A,s,E):
    expint=kB*(T**2.0)/(beta*E)*\
    np.exp(-E/(kB*T))*(1-2*kB*T)/E
    return A*np.exp(-E/(kB*T))*np.exp(-(s/beta)*expint)
inis=([1e15,1e10,1])  # starting values (A, s, E) for the fit
# find optimal parameters
params, cov=optimize.curve_fit(TLFOK,x_data,y_data,inis)
# params are the best fit values for the parameters
# cov is the covariance of the best fit parameters
plt.subplot(2,1, 1);
plt.scatter(x_data, y_data, label='Glocanin TL #1');
plt.plot(x_data, TLFOK(x_data, *params),
c='r',linewidth=3, label='FOK-TL equation');
leg = plt.legend()
```

```
leg.get_frame().set_linewidth(0.0)
plt.ylabel('TL signal [a.u.]');
plt.xlabel(r'Temperature T [K]');
plt.xlim(375,550);
plt.subplot(2,1, 2);
plt.plot(x_data,TLFOK(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.xlim(375,550);
plt.ylim(-0.0001,.0001);
plt.tight_layout()
A=format(params[0],"10.1E")
dA = format(np.sqrt(cov[0][0]),"10.1E")
s=format(params[1],"10.1E")
ds = format(np.sqrt(cov[1][1]),"10.1E")
E=round(params[2],3)
dE = round(np.sqrt(cov[2][2]),7)
res=TLFOK(x_data, *params)-y_data
FOM=round(100*np.sum(abs(res))/np.sum(y_data),3)
myTable = PrettyTable([ "A","s (s^-1)","ds","E(eV)","dE"])
myTable.add_row([A,s,ds,E,dE]);
print('FOM=',FOM, ' %')
print(myTable)
plt.show()
```

```
FOM= 0.003  %
+------------+------------+------------+-------+-------+
|     A      |  s (s^-1)  |     ds     | E(eV) |   dE  |
+------------+------------+------------+-------+-------+
|    3.6E+12 |    9.8E+10 |    7.6E+06 | 1.183 | 3e-06 |
+------------+------------+------------+-------+-------+
```

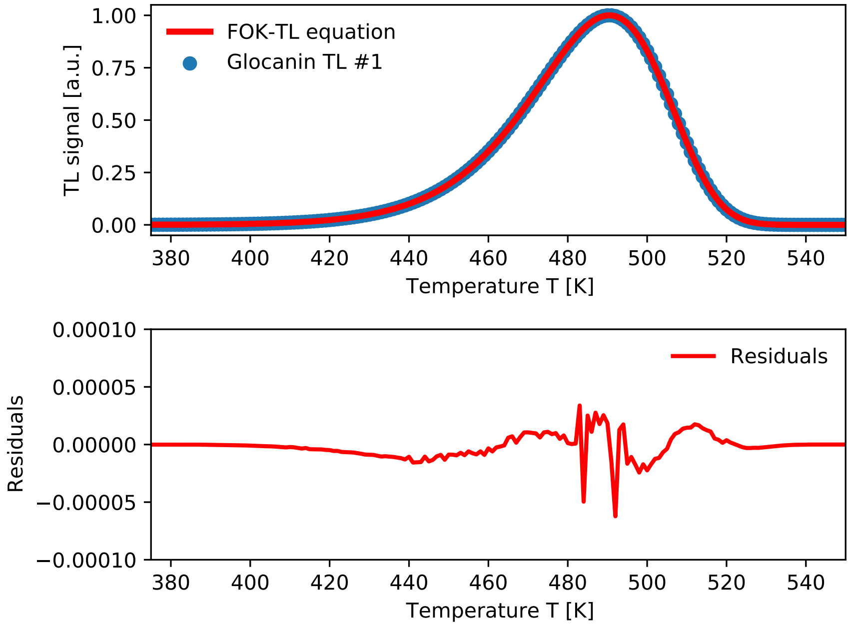

Fig. 3.1: Deconvolution of simulated TL data containing a single peak, using the FOK-TL deconvolution Eq.(**??**) (R-W equation). The data is from Reference glow curve #1 in the intercomparison project GLOCANIN (Bos et al. [1]).

Code 3.2: Deconvolution of Glocanin TL with transformed FOK-TL equation

```
# Deconvolution of Glocanin TL with the transformed FOK-TL eqt
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('glocanin1.txt')
```

```
x_data,y_data = data[:, 0], data[:, 1]/max(data[:, 1])
kB,  beta= 8.617e-5,  1
imax=max(y_data)
Tmax=x_data[np.argmax(y_data)]
def TLFOK(T,E):
    return imax*np.exp(1+(E/(kB*T))*((T-Tmax)/Tmax)-\
    (T**2/Tmax**2)*(1-2*kB*T/E)*np.exp((E/(kB*T))*\
    ((T-Tmax)/Tmax))-2*kB*Tmax/E)
params, cov=optimize.curve_fit(TLFOK,x_data,y_data,(1))
E=round(params[0],3)
dE = round(np.sqrt(cov[0][0]),4)
res=TLFOK(x_data, *params)-y_data
FOM=round(100*np.sum(abs(res))/np.sum(y_data),2)
myTable = PrettyTable([ "E(eV)","dE","FOM(%)"])
myTable.add_row([E,dE,FOM]);
print(myTable)
```

```
+-------+--------+--------+
| E(eV) |   dE   | FOM(%) |
+-------+--------+--------+
| 1.176 | 0.0012 |  1.57  |
+-------+--------+--------+
```

Code 3.3: Deconvolution of Glocanin TL using the GOK-TL equation

```
# Deconvolution of Glocanin TL with original GOK-TL
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('glocanin1.txt')
x_data,y_data = data[:, 0], data[:, 1]/max(data[:, 1])
kB,  beta= 8.617e-5,  1
def TLGOK(T,A,sprime,E,b):
    expint=kB*(T**2.0)/(beta*E)*\
    np.exp(-E/(kB*T))*(1-2*kB*T)/E
    return A*np.exp(-E/(kB*T))*((1+sprime*(b-1)*expint)\
    **(-b/(b-1)) )
```

```
params, cov=optimize.curve_fit(TLGOK,x_data,y_data,([1,2e10,1.,\
1.5]),bounds=((1e-10,1e8,1,1),(1e20,1e14,1.3,1.999)))
A=format(params[0],"10.1E")
dA = format(np.sqrt(cov[0][0]),"10.1E")
sprime=format(params[1],"10.1E")
E=round(params[2],4)
dE = round(np.sqrt(cov[2][2]),7)
b=round(params[3],5)
db = round(np.sqrt(cov[3][3]),7)
res=TLGOK(x_data, *params)-y_data
FOM=round(100*np.sum(abs(res))/np.sum(y_data),2)
myTable = PrettyTable([ "A","s' (s^-1)","E(eV)",\
"dE","b","db"])
myTable.add_row([A,sprime,E,dE,b,db]);
print("FOM=",FOM," %")
print(myTable)
```

```
FOM= 0.21  %
+------------+------------+--------+---------+-----+---------+
|     A      | s' (s^-1)  | E(eV)  |    dE   |  b  |    db   |
+------------+------------+--------+---------+-----+---------+
|    3.4E+12 |    9.1E+10 | 1.1796 | 1.9e-06 | 1.0 | 1.5e-06 |
+------------+------------+--------+---------+-----+---------+
```

Code 3.4: Deconvolution of Glocanin TL with transformed GOK-TL

```
# Deconvolution of Glocanin TL #1 with transformed GOK
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('glocanin1.txt')
x_data,y_data = data[:, 0], data[:, 1]
y_data=y_data/max(y_data)
kB=8.617E-5
imax=max(y_data)
def GOK_func(T, Tmax,b, En):
    return imax* np.exp(En/(kB*T)*(T-Tmax)/Tmax)*(b**\
```

```
    ((b/(b-1))))*((1+(b-1)*2*kB*Tmax/En+(b-1)*(1-2*kB*T/\
    En)*np.exp(En/(kB*T)*(T-Tmax)/Tmax)*(T**2.0)/(Tmax**\
    2.0))**(b/(1-b)))
params, cov=optimize.curve_fit(GOK_func,x_data,\
y_data,bounds=((460,1.001,.7),(520,2.0,1.3)))
Tmax=params[0]
s=np.exp(params[2]/(kB*Tmax))*(params[2]/(kB*(Tmax**2.0)))
sf=format(s, "10.1E")
Tmax=format(params[0],"10.1E")
b=round(params[1],3)
E=round(params[2],4)
db = format(np.sqrt(cov[1][1]),"10.1E")
dE = format(np.sqrt(cov[2][2]),"10.1E")
res=GOK_func(x_data, *params)-y_data
FOM=round(100*np.sum(np.abs(res))/np.sum(y_data),2)
myTable = PrettyTable([ "b","db", "E(eV)","dE(eV)",\
"s(s^-1)"])
myTable.add_row([b,db,E,dE,sf]);
print("FOM=",FOM," %")
print(myTable)
```

```
FOM= 0.03  %
+-------+------------+--------+------------+------------+
|   b   |     db     | E(eV)  |   dE(eV)   |  s(s^-1)   |
+-------+------------+--------+------------+------------+
| 1.001 |    1.1E-04 | 1.1828 |    7.8E-05 |    8.2E+10 |
+-------+------------+--------+------------+------------+
```

Code 3.5: Deconvolution of Glocanin TL with KV-TL equation

```
# Deconvolution of single TL peak with Lambert-OTOR equation
from scipy import optimize
import numpy as np
from prettytable import PrettyTable
import warnings
warnings.filterwarnings("ignore")
from scipy.special import wrightomega
data = np.loadtxt('glocanin1.txt')
x_data,y_data = data[:, 0], data[:, 1]/max( data[:, 1])
kB, beta=  8.617e-5,  1
def W_func(T,A,sprime, E, c):
```

```
   expint=kB*(T**2.0)/(beta*E)*np.exp(-E/(kB*T))*(1-2*kB*T/E)
   zTL=(1/c)-np.log(c)+(sprime*expint)
   lam=wrightomega(zTL)
   return A*np.exp(-E/(kB*T))/(lam+lam**2)
params, cov=optimize.curve_fit(W_func,x_data,y_data,([1e10,2e9,\
1.,10]),bounds=((1e8,1e8,.9,1e-4),(1e15,1e14,1.3,1e4)))
A=format(params[0],"10.1E")
sprime=format(params[1],"10.1E")
E=round(params[2],3)
c=round(params[3],1)
dA = format(np.sqrt(cov[0][0]),"10.1E")
dsprime = format(np.sqrt(cov[1][1]),"10.1E")
dE = round(np.sqrt(cov[2][2]),5)
dc = round(np.sqrt(cov[3][3]),1)
res=W_func(x_data, *params)-y_data
FOM=round(100*np.sum(abs(res))/np.sum(y_data),1)
myTable = PrettyTable()
myTable.add_row([ "A","dA", "s' (s^-1)","ds' (s^-1)"]);
myTable.add_row([A,dA,sprime,dsprime]);
myTable.add_row([" "]*4);
myTable.add_row(["E (eV)","dE(eV)","c","dc"]);
myTable.add_row([E,dE,c,dc]);
print("FOM=",FOM," %")
print(myTable)
```

```
FOM= 0.5  %
+------------+------------+------------+------------+
|  Field 1   |  Field 2   |  Field 3   |  Field 4   |
+------------+------------+------------+------------+
|     A      |     dA     | s' (s^-1)  | ds' (s^-1) |
|    3.1E+08 |    7.4E-05 |    6.9E+10 |    3.2E-07 |
|            |            |            |            |
|   E (eV)   |   dE(eV)   |     c      |     dc     |
|   1.176    |   1e-05    |   9998.2   |    2.3     |
+------------+------------+------------+------------+
```

Code 3.6: Deconvolution of alumina TL with transformed KV-TL eqt

```
# Deconvolution of alumina TL using the transformed KV-TL eqt
from scipy import optimize
import numpy as np
import matplotlib.pyplot as plt
from prettytable import PrettyTable
import warnings
warnings.filterwarnings("ignore")
from scipy.special import wrightomega
data = np.loadtxt('aluminaTLshort.txt')
x_data,y_data =273+ data[:, 0], data[:, 1]/max(\
data[:, 1])
# x_data=[273+u for u in np.array(x_data)]
x_data=np.array(x_data)
kB=8.617E-5
Imax=max(y_data)
Tmx=x_data[np.argmax(y_data)]
def W_func(T,R, E, c,Tmax):
    F=kB*(T**2.0)*np.exp(-E/(kB*T))*(1-2*kB*T/E)/E
    Fm=kB*(Tmax**2.0)*np.exp(-E/(kB*Tmax))*(1-2*kB*Tmax/E)/E
    a=kB*Tmax**2.0*(1-1.05*R**1.26)
    Z=1/c-np.log(c)+(F*E*np.exp(E/(kB*Tmax)))/a
    Zm=1/c-np.log(c)+(Fm*E*np.exp(E/(kB*Tmax)))/a
    argW=wrightomega(Z)
    argWm=wrightomega(Zm)
    return Imax*np.exp(-E/(kB*T)*(Tmax-T)/Tmax)*\
         (argWm+argWm**2.0)/(argW+argW**2.0)
params,cov=optimize.curve_fit(W_func,x_data,y_data,\
p0=(.5,1,10,Tmx),bounds=((.1,.9,1e-4,Tmx-5),(.9,1.3,1e4,Tmx+5)))
plt.subplot(2,1, 1);
plt.plot(x_data, W_func(x_data, *params),'-',linewidth=4);
plt.scatter(x_data, y_data, label='Experiment');
plt.plot(x_data, W_func(x_data, *params),
c='r',linewidth=3, label='Trasformed KV-TL eq.');
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.plot(x_data,W_func(x_data, *params)-\
y_data,'o',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(-.2,.2);
plt.tight_layout()
R, E, c, Tmax=[round(params[x],2) for x in range(4)]
dR, dE, dc, dTmax=[round(np.sqrt(cov[x][x]),2) for x in range(4)]
res=W_func(x_data, *params)-y_data
FOM=round(100*np.sum(abs(res))/np.sum(y_data),1)
myTable = PrettyTable([ "R", "dR","E(eV)","dE(eV)","c","dc"])
myTable.add_row([R,dR,E,dE,c,dc]);
print("FOM=",FOM," %")
print(myTable)
plt.show()
```

```
FOM= 2.7  %
+------+------+-------+--------+-------+-------+
|  R   |  dR  | E(eV) | dE(eV) |   c   |   dc  |
+------+------+-------+--------+-------+-------+
| 0.25 | 0.03 |  0.98 |  0.01  | 38.57 | 11.86 |
+------+------+-------+--------+-------+-------+
```

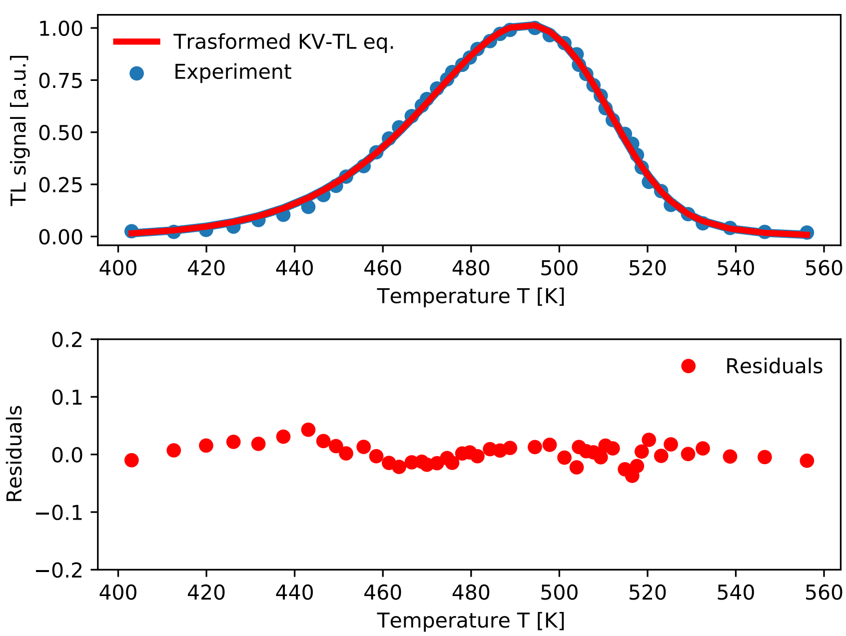

Fig. 3.2: Deconvolution of TL peak for the dosimetric material Al_2O_3:C, with the transformed KV-TL Eq.(**??**). Experimental data from Pagonis et al. [2]

Code 3.7: Deconvolution of alumina TL with MOK-TL equation

```
# Deconvolution of Glocanin #1 with original MOK equation
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('aluminaTLshort.txt')
x_data= 273+np.array(data[:, 0])
y_data=np.array(data[:, 1])/max(data[:, 1])
kB,  beta=  8.617e-5, 1
def TLMOK(T,A,sprime,E,alpha):
    expint=kB*(T**2.0)/(beta*E)*\
    np.exp(-E/(kB*T))*(1-2*kB*T/E)
    Ft=np.exp(sprime*expint)
    return A*np.exp(-E/(kB*T))*Ft/((Ft-alpha)**2.0)
params, cov=optimize.curve_fit(TLMOK,x_data,y_data,([2e10,2e9,\
1.,.01]),bounds=((1e9,1e8,.9,1e-4),(1e17,1e14,1.3,1)))
Tmax=x_data[np.argmax(y_data)]
A=format(params[0],"10.1E")
dA = format(np.sqrt(cov[0][0]),"10.1E")
sprime=format(params[1],"10.1E")
dsprime = format(np.sqrt(cov[1][1]),"10.1E")
E=round(params[2],3)
dE = round(np.sqrt(cov[2][2]),3)
alpha=round(params[3],5)
dalpha = round(np.sqrt(cov[3][3]),3)
res=TLMOK(x_data, *params)-y_data
FOM=round(100*np.sum(abs(res))/np.sum(y_data),2)
myTable = PrettyTable()
myTable.add_row([ "A","dA", "s' (s^-1)","ds' (s^-1)"]);
myTable.add_row([A,dA,sprime,dsprime]);
myTable.add_row([" "]*4);
myTable.add_row(["E (eV)","dE(eV)","alpha","dalpha"]);
myTable.add_row([E,dE,alpha,dalpha]);
print(myTable)

   +------------+------------+------------+------------+
```

```
  |  Field 1   |  Field 2   |  Field 3   |  Field 4   |
  +------------+------------+------------+------------+
  |     A      |     dA     | s' (s^-1)  | ds' (s^-1) |
  |    1.4E+11 |    3.6E+06 |    3.8E+09 |    1.1E+08 |
  |            |            |            |            |
  |   E (eV)   |   dE(eV)   |   alpha    |   dalpha   |
  |   1.075    |   0.002    |  0.33103   |    0.02    |
  +------------+------------+------------+------------+
```

Code 3.8: Deconvolution of alumina TL peak with transformed MOK-TL

```
# Deconvolution of Glocanin TL with transformed  MOK
from scipy import optimize
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from prettytable import PrettyTable
kB=8.617E-5
def MOK(T,alpha,E,Tm):
     fmok=(2.6-0.9203*alpha+0.324*(alpha**3.338))/(2.6-\
     2.9203*alpha+0.324*(alpha**3.338))
     FT=np.exp((1/fmok)*(T**2.00)/(Tm**2.0)*np.exp(E*(T-Tm)/\
     (kB*T*Tm))*(1-2.0*kB*T/E))
     FTm=np.exp((1-2*kB*Tm/E)/fmok)
     return imax*np.exp(E*(T-Tm)/(kB*T*Tm))*((FTm-alpha)**2.0)\
      *FT/(FTm*((FT-alpha)**2.0))
data = np.loadtxt('aluminaTLshort.txt')
x_data= 273+np.array(data[:, 0])
y_data=np.array(data[:, 1])
y_data=y_data/max(y_data)
imax=max(y_data)     #find imax from given data
Tmax=x_data[np.argmax(y_data)]
T=np.arange(390,570,1);
params,cov =optimize.curve_fit(MOK,x_data,y_data,p0=(.9,1.07,\
Tmax),bounds=((1e-5,.9,Tmax-5),(0.99999,1.3,Tmax+5)))
alphafit,Efit,Taxfit = np.round(params,2)
```

```
dalpha, dE,dTm = np.round(np.sqrt(np.diag(cov)),2)
res=MOK(x_data, *params)-y_data
FOM=round(100*np.sum(abs(res))/np.sum(y_data),2)
myTable=PrettyTable(["alpha","daplha","E (eV)","dE","FOM(%)"])
myTable.add_row([alphafit,dalpha, Efit, dE,FOM]);
print(myTable)
```

```
+-------+--------+--------+------+--------+
| alpha | daplha | E (eV) |  dE  | FOM(%) |
+-------+--------+--------+------+--------+
|  0.33 |  0.05  |  1.07  | 0.02 |  2.48  |
+-------+--------+--------+------+--------+
```

Code 3.9: Deconvolution of LBO data using transformed KV-TL equation

```
# Deconvolution using transformed Lambert equation
# ======================================================
from scipy import optimize
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from prettytable import PrettyTable
from scipy.special import wrightomega
data = np.loadtxt('lbodata.txt')
x_data,y_data = data[:, 0], data[:, 1]
kB=8.617E-5
def W_func(T,Imax,Tmax,R, E):
    F=kB*(T**2.0)*np.exp(-E/(kB*T))*(1-2*kB*T/E)/E
    Fm=kB*(Tmax**2.0)*np.exp(-E/(kB*Tmax))*(1-2*kB*Tmax/E)/E
    a=kB*Tmax**2.0*(1-1.05*R**1.26)
    Z=R/(1-R)-np.log((1-R)/R)+(F*E*np.exp(E/(kB*Tmax)))/a
    Zm=R/(1-R)-np.log((1-R)/R)+(Fm*E*np.exp(E/(kB*Tmax)))/a
    argW=wrightomega(Z)
    argWm=wrightomega(Zm)
    return Imax*np.exp(-E/(kB*T)*(Tmax-T)/Tmax)*\
    (argWm+argWm**2.0)/(argW+argW**2.0)
def total_TL(T, Imax1,Tmax1,R1, E1,Imax2,Tmax2,R2, E2):
    return W_func(T,Imax1,Tmax1,R1, E1)+W_func(T,Imax2,Tmax2,\
    R2, E2)
```

```
params, cov=optimize.curve_fit(total_TL,x_data,y_data,
p0=(119,463,1e-4,1.1,20,524,1e-4,1.2))
plt.subplot(2,1, 1);
plt.plot(x_data,y_data,'o',label='LBO data');
plt.plot(x_data, total_TL(x_data, *params),'+-',\
label='KV-TL transf.');
plt.plot(x_data, W_func(x_data, *params[0:4]),'-',\
label='peak 1');
plt.plot(x_data, W_func(x_data, *params[4:8]),'-',\
label='peak 2');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.ylabel('TL signal [a.u.]');
plt.xlabel(r'Temperature T [K]');
plt.ylim(0,140);
plt.text(400, 100,'Sample');
plt.text(400, 80,'LBO');
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(-10,10);
plt.tight_layout()
imax1,Tmax1, R1,E1=int(params[0]),round(params[1],2),\
round(params[2],2),round(params[3],3)
imax2,Tmax2,R2,E2=int(params[4]),round(params[5],2),\
round(params[6],2),round(params[7],3)
beta= 1
dR1 = round(np.sqrt(cov[2][2]),2)
dE1 = round(np.sqrt(cov[3][3]),2)
dR2 = round(np.sqrt(cov[6][6]),2)
dE2 = round(np.sqrt(cov[7][7]),2)
res=total_TL(x_data, *params)-y_data
FOM=round(100*np.sum(abs(res))/np.sum(y_data),2)
myTable = PrettyTable(["Imax", "Tmax","R", "dR","E(eV)",\
"dE(eV)","FOM(%)"])
myTable.add_row([imax1,Tmax1,R1,dR1,E1,dE1,FOM]);
myTable.add_row([imax2,Tmax2,R2,dR2,E2,dE2,"-"]);
print(myTable)
plt.show()

  +------+--------+------+------+-------+--------+--------+
```

```
| Imax |  Tmax  |  R   |  dR  | E(eV) | dE(eV) | FOM(%) |
+------+--------+------+------+-------+--------+--------+
| 118  | 464.35 | 0.04 | 0.02 | 1.221 |  0.01  |  2.7   |
|  23  | 518.23 | 0.63 | 0.69 | 1.668 |  0.46  |   -    |
+------+--------+------+------+-------+--------+--------+
```

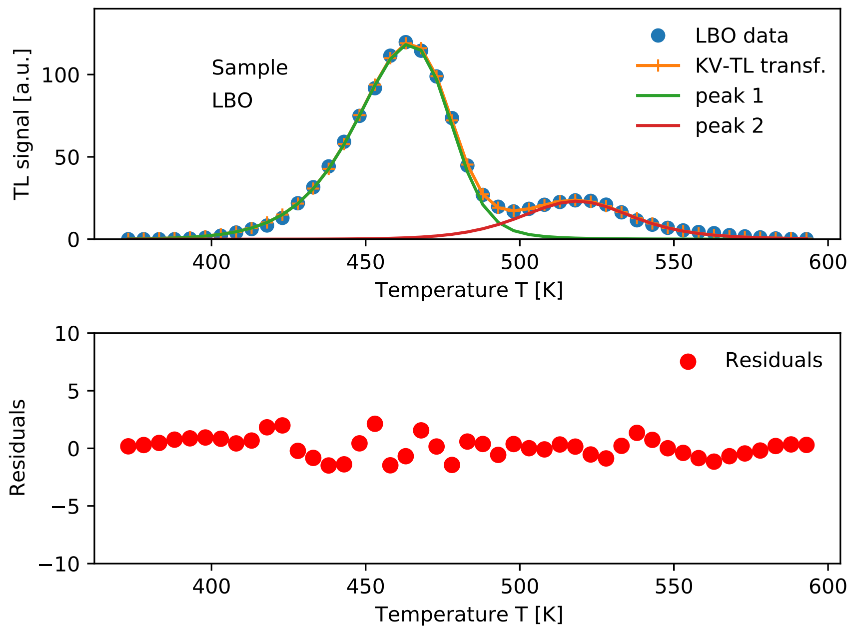

Fig. 3.3: Deconvolution of TL data containing two peaks from the dosimetric material LiB_4O_7:Cu,In (LBO), using the transformed KV-TL equation based on the Lambert W function. For more details of the experiment, see Kitis et al. [3].

Code 3.10: Deconvolution of LBO TL data using GOK-TL equation

```
# Deconvolution of TL user data (.txt file, GOK)
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('lbodata.txt')
x_data,y_data = data[:, 0], data[:, 1]
kB=8.617E-5
def TL(T, imax,b, En,Tmax):
    return imax* np.exp(En/(kB*T)*(T-Tmax)/Tmax)*(b**\
    ((b/(b-1))))*((1+(b-1)*2*kB*Tmax/En+(b-1)*(1-2*kB*T/En)*\
    np.exp(En/(kB*T)*(T-Tmax)/Tmax)*(T**2.0)/(Tmax**2.0))**\
    (b/(1-b)))
def total_TL(T, imax1,b1, En1,Tmax1, imax2,b2, En2,Tmax2):
    return TL(T, imax1,b1, En1,Tmax1)+TL(T,imax2,b2, En2,Tmax2)
params, cov = optimize.curve_fit(total_TL, x_data,
y_data,bounds=((80,1.001,.8,430, 20,1.001,.8,480),
(140,2.0,1.3,480, 40,2.0,1.3,540)))
imax1,b1,E1,Tmax1=int(params[0]),round(params[1],2),\
round(params[2],2),int(params[3]),
imax2,b2,E2,Tmax2=int(params[4]),round(params[5],2),\
round(params[6],2),int(params[7]),
beta= 1
s1=np.exp(E1/(kB*Tmax1))*(beta*E1/(kB*(Tmax1**2.0)))\
/(1+(b1-1)*2*kB*Tmax1/E1)
s1=format(s1,"10.2E")
s2=np.exp(E2/(kB*Tmax2))*(beta*E2/(kB*(Tmax2**2.0)))\
/(1+(b2-1)*2*kB*Tmax2/E2)
s2=format(s2,"10.2E")
db1 = round(np.sqrt(cov[1][1]),2)
dE1 = round(np.sqrt(cov[2][2]),2)
db2 = round(np.sqrt(cov[5][5]),2)
dE2 = round(np.sqrt(cov[6][6]),2)
res=total_TL(x_data, *params)-y_data
FOM=round(100*np.sum(abs(res))/np.sum(y_data),2)
myTable = PrettyTable(["Imax", "b","db", "E(eV)",\
"dE(eV)", "s(s^-1)","FOM(%)"])
myTable.add_row([imax1,b1,db1,E1,dE1,s1,FOM]);
myTable.add_row([imax2,b2,db2,E2,dE2,s2,"-"]);
print(myTable)
```

```
    +------+------+------+-------+--------+------------+--------+
    | Imax |  b   |  db  | E(eV) | dE(eV) |  s(s^-1)   | FOM(%) |
    +------+------+------+-------+--------+------------+--------+
    | 116  | 1.06 | 0.03 |  1.24 |  0.02  |   2.10E+12 |  3.05  |
```

```
|  23  | 1.5  | 0.22 |  1.3  |  0.18  |   2.73E+11 |   -    |
+------+------+------+-------+--------+-----------+--------+
```

Code 3.11: Deconvolution of 9-peak LiF TL using transformed KV-TL eqt

```
#Deconvolution of 9-peak TL using the transformed KV-TL eqt
from scipy import optimize
import numpy as np
import matplotlib.pyplot as plt
from prettytable import PrettyTable
import warnings
warnings.filterwarnings("ignore")
from scipy.special import wrightomega
data = np.loadtxt('Refglow009.txt')
x_data,y_data = data[:, 0], data[:, 1]
plt.scatter(x_data, y_data, label='Experiment');
kB=8.617E-5
def TL(T,imax,R,E,Tmax):
    F=kB*(T**2.0)*np.exp(-E/(kB*T))*(1-2*kB*T/E)/E
    Fm=kB*(Tmax**2.0)*np.exp(-E/(kB*Tmax))*(1-2*kB*Tmax/E)/E
    a=kB*Tmax**2.0*(1-1.05*R**1.26)
    Z=R/(1-R)-np.log((1-R)/R)+(F*E*np.exp(E/(kB*Tmax)))/a
    Zm=R/(1-R)-np.log((1-R)/R)+(Fm*E*np.exp(E/(kB*Tmax)))/a
    argW=wrightomega(Z)
    argWm=wrightomega(Zm)
    return imax*np.exp(-E/(kB*T)*(Tmax-T)/Tmax)*\
    (argWm+argWm**2.0)/(argW+argW**2.0)
def total_TL(T, *inis):
    u=np.array([0 for i in range(len(x_data))])
    imaxs,     Rs,        Es,         Tmaxs=\
    inis[0:9],inis[9:18],inis[18:27], inis[27:36]
    for i in range(9):
        u=u+TL(T,imaxs[i],Rs[i], Es[i],Tmaxs[i])
    return u
inis = (9824,21009,27792,50520,7153,5496,6080,1641,2316,
0.01,0.01,.01,.01,.01,.01,.01,.01,.01,
1.24,1.36,2.10, 2.65,1.43, 1.16,2.48,2.98,2.25,
 387,428,462,488,493,528,559,585, 602)
```

```
params, params_covariance = optimize.curve_fit(total_TL,\
x_data,y_data,p0=inis)
plt.subplot(2,1, 1);
plt.scatter(x_data, y_data, label='Experiment');
plt.plot(x_data, total_TL(x_data,
*params),c='r',linewidth=3, label='KV-TL transf.');
for i in range(9):
    plt.plot(x_data, TL(x_data, *params[i:36:9]));
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.ylabel('TL signal [a.u.]');
plt.xlabel(r'Temperature T [K]');
plt.text(350, 58000,'TLD-700');
plt.text(350, 50000,'GLOCANIN');
plt.text(350, 42000,'Refglow#9');
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(-20000,20000);
plt.tight_layout()
res=total_TL(x_data, *params)-y_data
FOM=round(100*np.sum(abs(res))/np.sum(y_data),2)
myTable = PrettyTable(["Imax", "R", "E (eV)", "Tmax (K)"])
for i in range(9):
    myTable.add_row(np.round(params[i:36:9],2));
print('FOM=',FOM,' %')
print(myTable)
plt.show()
```

```
FOM= 3.04  %
+----------+------+--------+----------+
|   Imax   |  R   | E (eV) | Tmax (K) |
+----------+------+--------+----------+
| 9784.91  | 0.01 |  1.23  |  387.2   |
| 21072.11 | 0.03 |  1.32  |  428.46  |
| 27527.44 | 0.1  |  1.91  |  462.46  |
| 52836.24 | 0.04 |  2.47  |  488.03  |
|  8957.9  | 0.01 |  1.27  |  491.91  |
| 5920.54  | 0.04 |  0.92  |  525.15  |
| 6067.67  | 0.05 |  2.36  |  559.38  |
| 1840.33  | 0.0  |  2.87  |  583.57  |
```

```
| 2419.11   | 0.15 |  1.85   |  603.13   |
+----------+------+--------+----------+
```

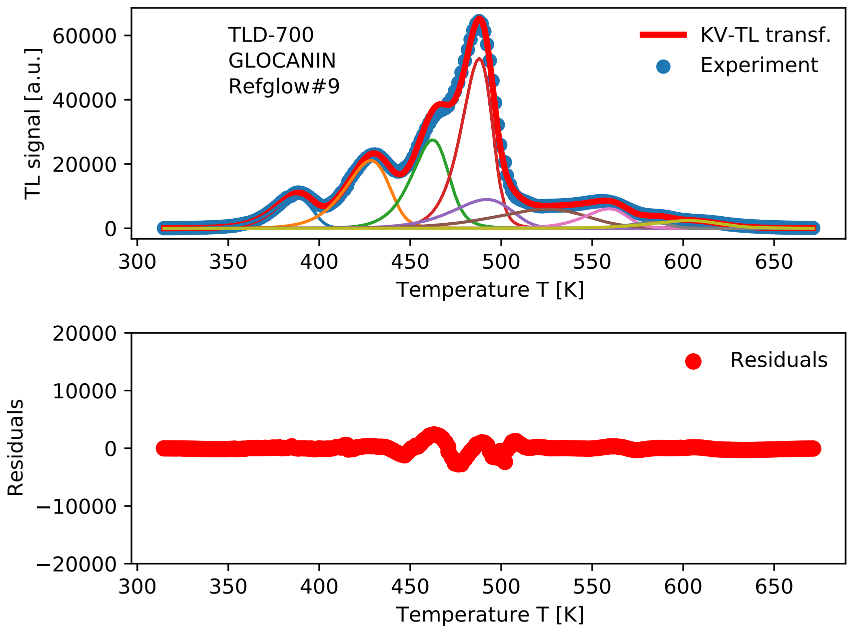

Fig. 3.4: Deconvolution of a glow curve from the GLOCANIN project using 9 peaks with the transformed KV-TL Eq.(**??**), and with user-supplied initial kinetic parameters (Bos et al. [1]). Bottom: Histogram plot of the residuals $y_i^{expt} - y_i^{fit}$.

Code 3.12: Deconvolution of 9-peak data using transformed GOK-TL eqt

```
#Deconvolution of 9-peak data with transformed GOK-TL eqt
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('Refglow009.txt')
x_data,y_data = data[:, 0], data[:, 1]
plt.scatter(x_data, y_data, label='Experiment');
kB=8.617E-5
def TL(T, imax,b, En,Tmax):
    return imax* np.exp(En/(kB*T)*(T-Tmax)/Tmax)*(b**\
    ((b/(b-1))))*((1+(b-1)*2*kB*Tmax/En+(b-1)*(1-2*kB*T/\
    En)*np.exp(En/(kB*T)*(T-Tmax)/Tmax)*(T**2.0)/(Tmax**\
    2.0))**(b/(1-b)))
def total_TL(T, *inis):
    u=np.array([0 for i in range(len(x_data))])
    imaxs,     bs,        Es,         Tmaxs=\
    inis[0:9],inis[9:18],inis[18:27], inis[27:36]
    for i in range(9):
        u=u+TL(T,imaxs[i],bs[i], Es[i],Tmaxs[i])
    return u
inis = (9824,21009,27792,50520,7153,5496,6080,1641,2316,
1.02, 1.15, 1.99,1.20, 1.28,1.19,1.40,1.01,1.18,
1.24,1.36,2.10, 2.65,1.43, 1.16,2.48,2.98,2.25,
 387,428,462,488,493,528,559,585, 602)
params, params_covariance = optimize.curve_fit(total_TL,\
x_data,y_data,p0=inis)
plt.subplot(2,1, 1);
plt.scatter(x_data, y_data, label='Experiment');
plt.plot(x_data, total_TL(x_data,
*params),c='r',linewidth=3, label='CGCD');
for i in range(9):
    plt.plot(x_data, TL(x_data, *params[i:36:9]));
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.ylabel('TL signal [a.u.]');
plt.xlabel(r'Temperature T [K]');
plt.text(350, 58000,'TLD-700');
plt.text(350, 50000,'GLOCANIN');
plt.text(350, 42000,'Refglow#9');
plt.subplot(2,1, 2);
res=total_TL(x_data, *params)-y_data;
plt.hist(res,22,label='Residuals');
plt.ylabel('Distribution of residuals');
plt.xlabel(r'Residuals');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
```

```
plt.tight_layout()
FOM=round(100*np.sum(abs(res))/np.sum(y_data),2)
myTable = PrettyTable(["Imax", "b", "E (eV)", "Tmax (K)"])
for i in range(9):
    myTable.add_row(np.round(params[i:36:9],2));
print('FOM=',FOM,' %')
print(myTable)
plt.show()
```

```
FOM= 0.84  %
+----------+------+--------+----------+
|   Imax   |  b   | E (eV) | Tmax (K) |
+----------+------+--------+----------+
| 9789.41  | 1.02 |  1.22  |  387.37  |
| 21123.73 | 1.18 |  1.38  |  428.24  |
| 27550.15 | 2.01 |  2.14  |  462.33  |
| 50556.3  | 1.19 |  2.62  |  488.18  |
| 7393.51  | 1.23 |  1.45  |  493.41  |
|  5590.2  | 1.18 |  1.15  |  527.78  |
| 6082.15  | 1.38 |  2.47  |  558.97  |
| 1537.97  | 1.0  |  3.03  |  585.25  |
| 2355.67  | 1.87 |  2.07  |  601.83  |
+----------+------+--------+----------+
```

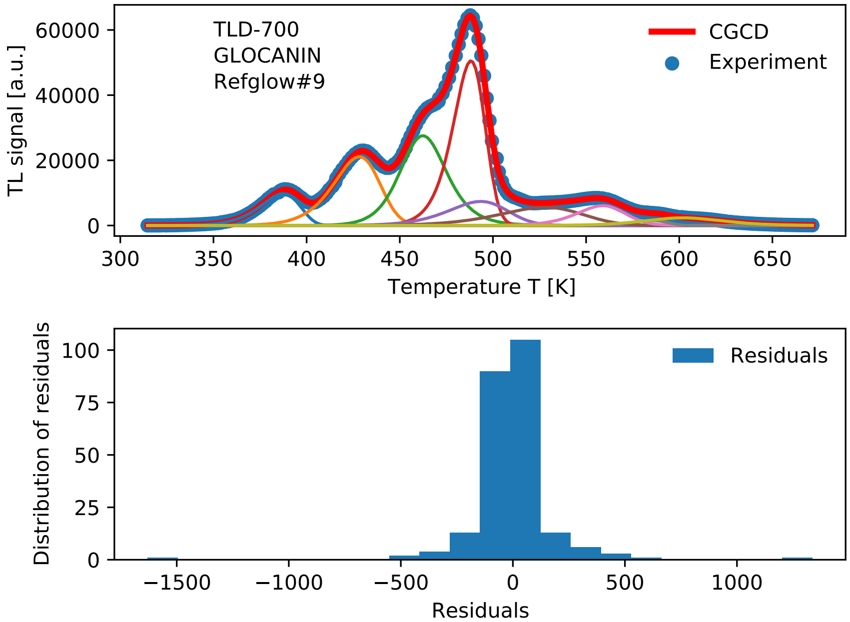

Fig. 3.5: Deconvolution of a glow curve from the GLOCANIN project using 9 peaks with the transformed GOK-TL equation, and with user-supplied initial kinetic parameters (Bos et al. [1]). Bottom: Histogram of residuals $y_i^{expt} - y_i^{fit}$ from the best fit for the nine-peak TL glow curve of the GLOCANIN project, with a GOK-TL model.

References

1. A. J. J. Bos, T. M. Piters, J. M. Gómez-Ros, A. Delgado, An Intercomparison of Glow Curve Analysis Computer Programs: I. Synthetic Glow Curves, Radiation protection dosimetry 47 (1993) 473–477. doi:10.1093/oxfordjournals.rpd.a081789.
2. 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.
3. G. Kitis, G. S. Polymeris, I. K. Sfampa, M. Prokic, N. Meriç, V. Pagonis, Prompt isothermal decay of thermoluminescence in Mg_4BO_7: Dy, Na and Li_4BO_7:Cu,In dosimeters, Radiation Measurements 84 (2016) 15–25. doi:https://doi.org/10.1016/j.radmeas.2015.11.002.
URL http://www.sciencedirect.com/science/article/pii/S1350448715300731