Chapter 7
ITL SIGNALS: DATA ANALYSIS

Code 7.1: Three-exponential isothermal analysis for Durango apatite

```
# Isothermal analysis for Durango apatite
#  deconvolution of 220degC ITL data with sum of 3 exponentials
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('DurangoITL220.txt')
x_data,y_data = data[:, 0][3:1000], data[:, 1][3:1000]
y_data=y_data/max(y_data)
def oneexpon(x_data,N,tau):
    u=np.abs(N)*(np.exp(-x_data/np.abs(tau)))
    return u
def total_ITL(t, *inis):
    u=np.array([0 for i in range(len(x_data))])
    Ns, taus =    inis[0:nPks], inis[nPks:2*nPks]
    for i in range(nPks):
        u=u+oneexpon(t,Ns[i],taus[i])
    return u
t = np.linspace(0, len(x_data), len(x_data))
nPks=3
N=[1,.5,.1]
tau=[10,100,1000]
inis=N+tau
params, cov = optimize.curve_fit(total_ITL,\
x_data,y_data,p0=inis,maxfev=10000)
plt.scatter(x_data, y_data,c='r',label='Durango ITL signal');
plt.plot(x_data, total_ITL(x_data,
 *params),c='black',label='Sum of exponentials',linewidth=1);
sums=[0]*nPks
for i in range(0,nPks):
    ITLi=oneexpon(t, params[i],params[nPks+i]);
    sums[i]=np.sum(ITLi)
    plt.plot(t,ITLi);
pc1=round(100*sums[0]/(sums[0]+sums[1]+sums[2]),1)
pc2=round(100*sums[1]/(sums[0]+sums[1]+sums[2]),1)
pc3=round(100*sums[2]/(sums[0]+sums[1]+sums[2]),1)
leg = plt.legend();
leg.get_frame().set_linewidth(0.0);
plt.ylabel('ITL [a.u.]');
plt.xlabel(r'Stimulation time [s]');
res=total_ITL(x_data, *params)-y_data
FOM=100*np.sum(abs(res))/np.sum(y_data)
Ns=[round(x,3) for x in params[0:nPks]]
taus=[round(x,1) for x in params[nPks:2*nPks]]
dN=[round(np.sqrt(cov[x][x]),3) for x in range(3)]
dtaus=[round(np.sqrt(cov[x][x]),1) for x in range(3,6)]
myTable = PrettyTable([ "Curve","N (a.u.)","dN (a.u)",\
'tau (s)',"dtau (s)","%"])
myTable.add_row(["1",Ns[0],dN[0],taus[0],dtaus[0],pc1]);
myTable.add_row(["2",Ns[1],dN[1],taus[1],dtaus[1],pc2]);
myTable.add_row(["3",Ns[2],dN[2],taus[2],dtaus[2],pc3]);
print('FOM=',round(FOM,2),' %')
print(myTable)
plt.show();
```

```
FOM= 1.6  %
+-------+----------+----------+---------+----------+------+
| Curve | N (a.u.) | dN (a.u) | tau (s) | dtau (s) |  %   |
+-------+----------+----------+---------+----------+------+
|   1   |  0.572   |  0.006   |   22.4  |   0.3    | 10.7 |
|   2   |  0.412   |  0.005   |   92.5  |   1.4    | 31.2 |
|   3   |  0.141   |  0.002   |  642.0  |   7.9    | 58.1 |
+-------+----------+----------+---------+----------+------+
```

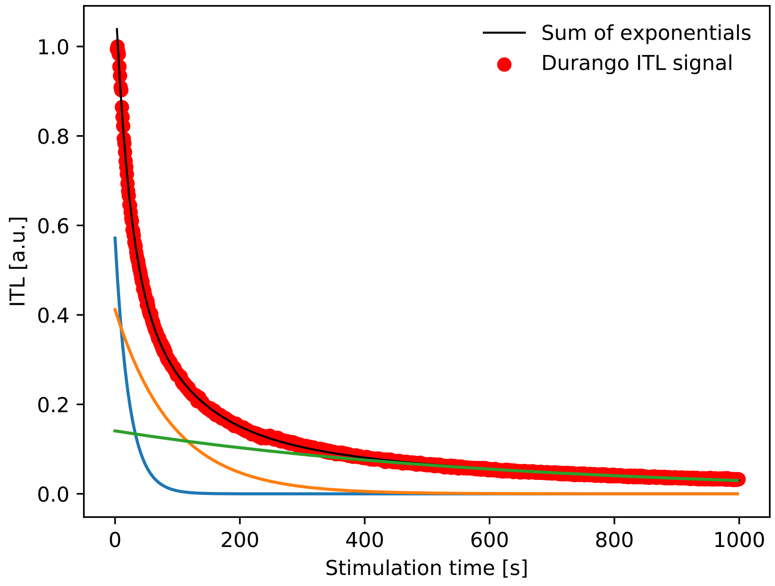

Fig. 7.1: ITL decay curve for Durango apatite measured at a constant temperature of 220°C. The solid black line is the best fit with three exponential decay curves. For more details see Sfampa et al. [**?**].

Code 7.2: Deconvolution of ITL using the KV-ITL equation

```
#  deconvolution with KV-ITL equation Durango 220deg ITL Data
from scipy import optimize
import numpy as np
import matplotlib.pyplot as plt
from prettytable import PrettyTable
from scipy.special import wrightomega
import warnings
data = np.loadtxt('DurangoITL220.txt')
x_data,y_data = data[:, 0][3:1000], data[:, 1][3:1000]
y_data=y_data/max(y_data)
plt.plot(x_data,y_data);
def KVITL(t, A,c,sprime):
    zITL=(1/c)-np.log(c)+sprime*t
    lam=wrightomega(zITL)
    ITL=A/(lam+lam**2)
    return ITL
def total_ITL(t, *inis):
    u=np.array([0 for i in range(len(x_data))])
    As, cs ,sprimes=    inis[0:nPks], inis[nPks:2*nPks],\
    inis[2*nPks:3*nPks]
    bgd=inis[-1]
    for i in range(nPks):
        u=u+KVITL(t,As[i],cs[i],sprimes[i])
    u=u+bgd
    return u
t = np.linspace(1, 1000, 1000)
nPks=1
A=[max(y_data)]*nPks
lowA, highA=[0.01*x for x in A], [200 for x in A]
c=[1]*nPks
lowc, highc= [0.001*x for x in c], [1e4*x for x in c]
sprime=[.01]*nPks
lowsprime, highsprime=  [0.001*x for x in sprime],\
[10*x for x in sprime]
bgd, lowbgd, highbgd=[.1,0,.15]
inis=A+c+sprime+[bgd]
lowbnds=lowA+lowc+lowsprime+[lowbgd]
highbnds=highA+highc+highsprime+[highbgd]
params, cov = optimize.curve_fit(total_ITL,\
x_data,y_data,p0=inis,bounds=(lowbnds,highbnds),maxfev=10000)
plt.scatter(x_data, y_data,c='r',label='Durango ITL signal');
plt.plot(x_data, total_ITL(x_data,
 *params),c='black',label='KV-ITL',linewidth=1);
sums=[0]*nPks
for i in range(0,nPks):
    ITLi=KVITL(t, params[i],params[nPks+i], params[2*nPks+i]);
    sums[i]=np.sum(ITLi)
    plt.plot(t,ITLi);
bgdarea=max(t)*params[-1]
plt.plot(t,[params[-1]]*len(t));
leg = plt.legend();
leg.get_frame().set_linewidth(0.0);
plt.ylabel('ITL-OSL [a.u.]');
plt.xlabel(r'Stimulation time [s]');
res=total_ITL(x_data, *params)-y_data
FOM=100*np.sum(abs(res))/np.sum(y_data)
print('FOM=',round(FOM,1),' %')
```

```
As=[round(x,1) for x in params[0:nPks]]
cs=[round(x,2) for x in params[nPks:2*nPks]]
sprimes=[round(x,3) for x in params[2*nPks:3*nPks]]
dAs=[round(np.sqrt(cov[x][x]),1) for x in range(nPks)]
dcs=[round(np.sqrt(cov[x][x]),2) for x in range(nPks,2*nPks)]
dsprimes=[round(np.sqrt(cov[x][x]),3) for x in\
range(2*nPks,3*nPks)]
bgd=round(params[-1],3)
myTable = PrettyTable([ "A (a.u.)","dA",\
'c','dc',"s' (s^-1)","ds'",'bgd'])
myTable.add_row([As[0],dAs[0],cs[0],dcs[0],\
sprimes[0],dsprimes[0],bgd]);
print(myTable)
plt.show();
```

```
FOM= 8.0  %
+----------+------+------+------+-----------+------+-------+
| A (a.u.) |  dA  |  c   |  dc  | s' (s^-1) | ds'  |  bgd  |
+----------+------+------+------+-----------+------+-------+
|   73.3   | 63.0 | 0.12 | 0.06 |    0.1    | 0.04 | 0.039 |
+----------+------+------+------+-----------+------+-------+
```

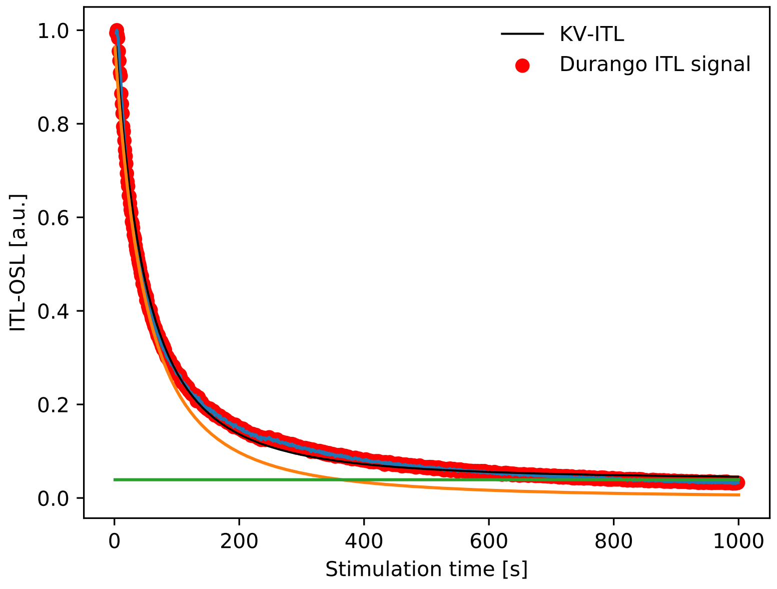

Fig. 7.2: (a) ITL decay curve for Durango apatite at a constant temperature of 220°C. The solid black line is the best fit with a KV-ITL component plus a constant background, using Eq.(**??**). For a more details analysis see Sfampa et al. [**?**].

Code 7.3: Deconvolution of ITL using the MOK-ITL equation

```
#  deconvolution with MOK-ITL equation Durango 220deg ITL Data
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('DurangoITL220.txt')
x_data,y_data = data[:, 0][3:1000], data[:, 1][3:1000]
y_data=y_data/max(y_data)
# b in this code represents the MOK parameter alpha
def MOKITL(t, A, b, sprime):
    F=np.exp(sprime*t)
    ITL=A*F/((F-b)**2)
    return ITL
def total_MOKITL(t, *inis):
    u=np.array([0 for i in range(len(x_data))])
    As, bs ,sprimes=inis[0:nPks], inis[nPks:2*nPks],\
    inis[2*nPks:3*nPks]
    bgd=inis[-1]
    for i in range(nPks):
        u=u+MOKITL(t,As[i],bs[i],sprimes[i])
    ubgd=bgd
    u=u+ubgd
    return u
t = np.linspace(0, 1000, 1000)
nPks=1
A=[100]*nPks
lowA, highA=[0.1 for x in A], [1e4 for x in A]
b=[0.1]*nPks
lowb, highb= [0.001 for x in b], [1 for x in b]
sprime=[.01]*nPks
lowsprime, highsprime=  [1e-4 for x in sprime],\
[1e4 for x in sprime]
```

```
bgd=.01
lowbgd,highbgd=0,.3
inis=A+b+sprime+[bgd]
lowbnds=lowA+lowb+lowsprime+[lowbgd]
highbnds=highA+highb+highsprime+[highbgd]
params, cov = optimize.curve_fit(total_MOKITL,\
x_data,y_data,p0=inis,bounds=(lowbnds,highbnds),maxfev=10000)
res=total_MOKITL(x_data, *params)-y_data
FOM=100*np.sum(abs(res))/np.sum(y_data)
print('FOM=',round(FOM,1),' %')
As=[round(x,3) for x in params[0:nPks]]
alpha=[round(x,2) for x in params[nPks:2*nPks]]
sprimes=[round(x,4) for x in params[2*nPks:3*nPks]]
dAs=[round(np.sqrt(cov[x][x]),3) for x in range(nPks)]
dalpha=[round(np.sqrt(cov[x][x]),2) for x in range(nPks,2*nPks)]
dsprimes=[round(np.sqrt(cov[x][x]),4) for x in\
range(2*nPks,3*nPks)]
bgd=round(params[-1],2)
myTable = PrettyTable(["A (a.u.)","dA",\
'alpha','dalpha',"s' (s^-1)","ds' (s^-1)"])
myTable.add_row([As[0],dAs[0],alpha[0],dalpha[0],\
sprimes[0],dsprimes[0]]);
print(myTable)
```

```
FOM= 8.2  %
+----------+------+-------+--------+-----------+------------+
| A (a.u.) |  dA  | alpha | dalpha | s' (s^-1) | ds' (s^-1) |
+----------+------+-------+--------+-----------+------------+
|   0.1    | 0.02 |  0.68 |  0.03  |   0.004   |   0.0005   |
+----------+------+-------+--------+-----------+------------+
```

Code 7.4: Deconvolution of ITL using the GOK-ITL equation

```
#  deconvolution with GOK-ITL equation Durango 220deg ITL Data
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('DurangoITL220.txt')
x_data,y_data = data[:, 0][3:1000], data[:, 1][3:1000]
```

```
y_data=y_data/max(y_data)
def GOKITL(t, A, b, sprime):
    ITL=A*(1+(b-1)*sprime*t)**(-b/(b-1))
    return ITL
def total_GOKITL(t, *inis):
    u=np.array([0 for i in range(len(x_data))])
    As, bs ,sprimes=inis[0:nPks], inis[nPks:2*nPks],\
    inis[2*nPks:3*nPks]
    bgd=inis[-1]
    for i in range(nPks):
        u=u+GOKITL(t,As[i],bs[i],sprimes[i])
    ubgd=bgd
    u=u+ubgd
    return u
t = np.linspace(0, 1000, 1000)
nPks=1
A=[max(y_data)]*nPks
lowA, highA=[0.01*x for x in A], [1 for x in A]
b=[1.5]*nPks
lowb, highb= [1.001 for x in b], [2 for x in b]
sprime=[10]*nPks
lowsprime, highsprime=  [1e-4 for x in sprime],\
[1e4 for x in sprime]
bgd=.01
lowbgd,highbgd=0,.3
inis=A+b+sprime+[bgd]
lowbnds=lowA+lowb+lowsprime+[lowbgd]
highbnds=highA+highb+highsprime+[highbgd]
params, cov = optimize.curve_fit(total_GOKITL,\
x_data,y_data,p0=inis,bounds=(lowbnds,highbnds),maxfev=10000)
res=total_GOKITL(x_data, *params)-y_data
FOM=100*np.sum(abs(res))/np.sum(y_data)
As=[round(x,3) for x in params[0:nPks]]
bs=[round(x,1) for x in params[nPks:2*nPks]]
sprimes=[round(x,4) for x in params[2*nPks:3*nPks]]
dAs=[round(np.sqrt(cov[x][x]),3) for x in range(nPks)]
dbs=[round(np.sqrt(cov[x][x]),1) for x in range(nPks,2*nPks)]
dsprimes=[round(np.sqrt(cov[x][x]),4) for x in\
range(2*nPks,3*nPks)]
bgd=round(params[-1],3)
print('FOM=',round(FOM,1)," %");
myTable = PrettyTable(["A (a.u.)","dA",\
'b','db',"s' (s^-1)","ds' (s^-1)"])
myTable.add_row([As[0],dAs[0],bs[0],dbs[0],\
sprimes[0],dsprimes[0]]);
print(myTable)
```

```
  FOM= 7.5  %
  +----------+-------+-----+-----+-----------+------------+
  | A (a.u.) |   dA  |  b  |  db | s' (s^-1) | ds' (s^-1) |
  +----------+-------+-----+-----+-----------+------------+
  |  0.998   | 0.006 | 2.0 | 0.1 |   0.0108  |   0.0002   |
  +----------+-------+-----+-----+-----------+------------+
```

Code 7.5: Deconvolution of ITL using the KP-ITL equation

```
# deconvolution with KP-ITL equation Durango 220deg ITL Data
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('DurangoITL220.txt')
x_data,y_data = data[:, 0][3:1000], data[:, 1][3:1000]
y_data=y_data/max(y_data)
plt.subplot(2,1, 1);
def test_func(x, imax_fit,rho_fit, A_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)
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]),3)
plt.plot(np.log(x_data), y_data,'+',c='r', label='Experiment');
plt.plot(np.log(x_data), test_func(x_data, *params[0:4]),
label='KP-ITL equation');
leg = plt.legend();
leg.get_frame().set_linewidth(0.0);
plt.ylabel('ITL signal [a.u.]');
plt.xlabel('ln(time)');
plt.text(1,.5,'Durango ITL signal');
plt.subplot(2,1, 2);
plt.plot(np.log(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('ln(time)');
plt.ylim(-.1,.1);
plt.tight_layout();
imax,rho, A=int(params[0]),round(params[1],5),\
round(params[2],2)
res=test_func(x_data, *params)-y_data
FOM=round(100*np.sum(abs(res))/np.sum(y_data),2)
myTable=PrettyTable(["imax", "rho",  "d(rho)",\
"sprime(s^-1)","dsprime","FOM"]);
myTable.add_row([imax,rho,drho, A, dA,FOM]);
print(myTable)
plt.show()
```

```
  +------+---------+--------+--------------+---------+-----+
  | imax |   rho   | d(rho) | sprime(s^-1) | dsprime | FOM |
  +------+---------+--------+--------------+---------+-----+
  |  1   | 0.00235 | 1e-05  |     1.26     |  0.004  | 1.4 |
  +------+---------+--------+--------------+---------+-----+
```

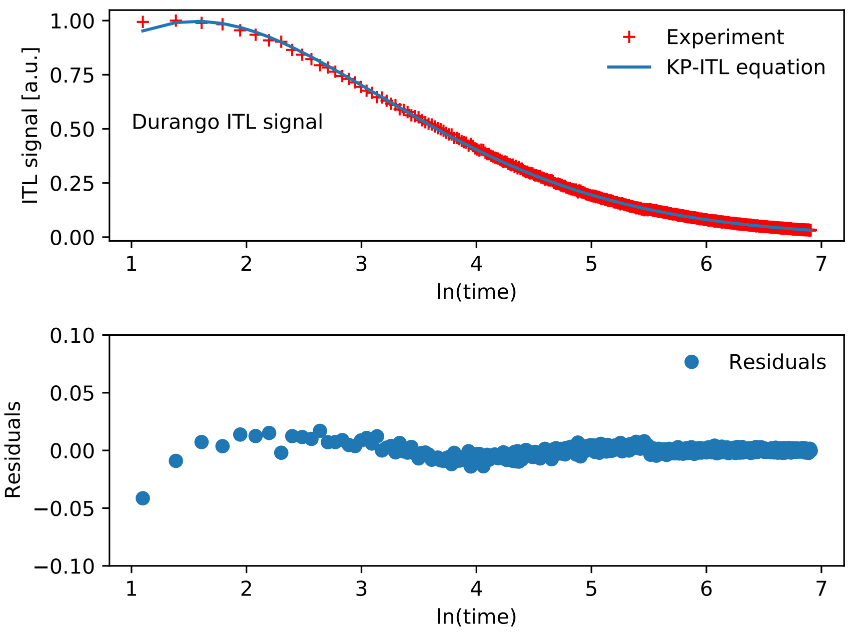

Fig. 7.3: (a) ITL decay curve for Durango apatite at a temperature of 220°C. The solid line is the best fit with a single KP-ITL component with Eq.(**??**). (b) The residuals of the best fitting process. For more details see Sfampa et al. [**?**].

Code 7.6: Peak shape transformation of ITL signals from LBO

```
# Peak shape transformation of ITL signals
import numpy as np
import matplotlib.pyplot as plt
from scipy import optimize
from prettytable import PrettyTable
fils=['LBOITL1.txt','LBOITL3.txt','LBOITL6.txt',
'LBOITL7.txt','LBOITL8.txt']
plt.subplot(1,2,1);
marks=['+','o','^','.','x']
labls=[r'95$^o$C',r'120$^o$C',r'145$^o$C',r'170$^o$C',\
r'195$^o$C']
for j in range(2,5):
    data = np.loadtxt(fils[j],delimiter=',')
    x_data,y_data =1+np.array(data[:,0]), np.array(data[:,1])
    plt.plot(x_data,y_data,marks[j]+'-',label=labls[j]);
    plt.xlabel('Stimulation time t [s]');
    plt.ylabel(r'ITL signal  [a.u.]');
plt.title('(a)');
plt.text(300,.7,'ITL signal');
leg = plt.legend();
leg.get_frame().set_linewidth(0.0);
plt.subplot(1,2,2);
for j in range(2,5):
    data = np.loadtxt(fils[j],delimiter=',')
    x_data, y_data = 1+np.array(data[:, 0]), np.array(data[:, 1])
    Lt=np.array(x_data)*np.array(y_data)
    plt.plot(np.log(x_data),Lt,marks[j],label=labls[j]);
    plt.xlabel('lnt');
    plt.ylabel(r't . I(t)');
plt.ylim(0,60);
plt.title('(b)');
plt.text(2.2,42,'t.I(t) vs lnt');
leg = plt.legend();
leg.get_frame().set_linewidth(0.0);
```

```
plt.tight_layout()
plt.show()
```

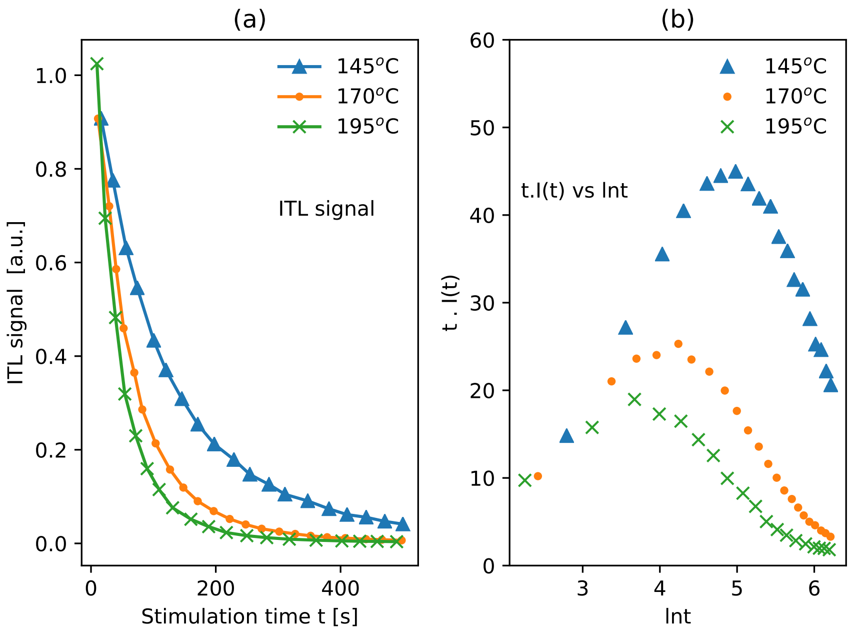

Fig. 7.4: (a) ITL decay curves for sample Li_4 BO_7:Cu,In (LBO), measured at three different temperatures. (b) The featureless data in (a) is transformed into peak-shaped signals. For more details see Kitis et al. [**?**].

Code 7.7: Peak shape transformation of CW-IRSL signals from feldspar

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import optimize
from prettytable import PrettyTable
fils=['OSL0s.txt','OSL100s.txt','OSL1000s.txt']
plt.subplot(1,2,1);
marks=['+','o','^']
labls=['0 s','100 s','1000 s']
for j in range(0,3):
    data = np.loadtxt(fils[j])
    x_data, y_data = 1+np.array(data[:, 0]), \
    np.array(data[:, 1])
    plt.plot(x_data,y_data,marks[j]+'-',label=labls[j]);
plt.xlabel('Stimulation time t [s]');
plt.ylabel(r'I(t) CW-IRSL intensity  [cts/s]');
plt.title('(a)');
plt.text(500,2.5e6,'CW-IRSL signal');
leg = plt.legend();
leg.get_frame().set_linewidth(0.0);
plt.subplot(1,2,2);
for j in range(0,3):
    data = np.loadtxt(fils[j])
    x_data,y_data=1+np.array(data[:,0]),\
    np.array(data[:,1])
    Lt=np.array(x_data)*np.array(y_data)
    plt.plot(np.log(x_data),Lt,marks[j],label=labls[j]);
plt.ylim(0,1.7e7);
plt.xlabel('lnt');
plt.ylabel(r't . I(t)');
plt.title('(b)');
plt.text(4,.45e7,'t.I(t) vs lnt');
leg = plt.legend();
leg.get_frame().set_linewidth(0.0);
plt.tight_layout()
plt.show()
```

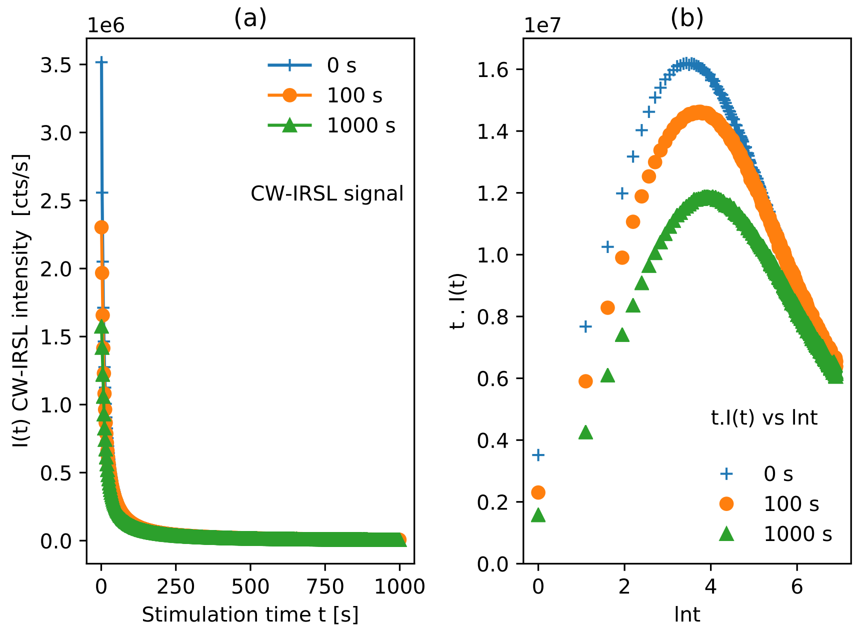

Fig. 7.5: (a) CW-IRSL decay curves for feldspar KST4, measured after exposure of an irradiated aliquot to blue LEDs for 0, 100, 1000 s. (b) The featureless data in (a) is transformed into peak-shaped signals. For more details see Sfampa et al. [**?**].

References