Chapter 15
DOSE RESPONSE OF LUMINESCENCE SIGNALS: DATA ANALYSIS

Code 15.1: Fit dose response data with saturating exponential and PKC

```
#Fit dose response data with Saturating Exponential
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import lambertw
import warnings
warnings.filterwarnings("ignore")
from scipy import optimize
from prettytable import PrettyTable
## fit to SE equation  ----
x_data = ([-34.2, 34.2466,68.4932,273.973, 1027.4,1986.3,3013.7,
      5000, 7979.45, 10000])
y_data = ([1.04, 0.000978474, 2.76386, 7.24592,12.6008,14.5329,
      15.8956,  17.1905, 17.847, 18.0952])
# x_data =np.array([0,50.7117,100.534,152.135,204.626,272.242])
# y_data = np.array([0,33.144,42.205,43.1055,44.4157,43.7098])
plt.plot(x_data,y_data,"o");
def SE(x_data,A,Do):
    u=A*(1-np.exp(-x_data/np.abs(Do)))
    return u
def lambertfit(x_data,N,R,Dc):
    u=np.real(N*(1+lambertw((np.abs(R)-1)*np.exp(np.abs(R)-1-\
    x_data/np.abs(Dc)))/(1-np.abs(R))))
    u.astype(float)
    return u
init_vals=[20,20]
```

```
params, cov = optimize.curve_fit(SE,\
x_data, y_data,p0=init_vals)
[A,Do]=[round(params[x],1) for x in range(2)]
dA = round(np.sqrt(cov[0][0]),2)
dDo = int(np.sqrt(cov[1][1]))
x_vals=np.arange(0,1e4,1e2)
plt.plot(x_vals, SE(x_vals, *params[0:2]),linestyle='--',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(0,30);
plt.ylabel('OSL (L/T)');
plt.xlabel('Dose [Gy]');
plt.title('Fine grain dose response');
dA = round(np.sqrt(cov[0][0]),1)
dDo = round(np.sqrt(cov[1][1]),1)
init_vals=[100.0,.1,100]
params, cov = optimize.curve_fit(lambertfit,\
x_data, y_data,p0=init_vals)
dN = round(np.sqrt(cov[0][0]),0)
dR = round(abs(np.sqrt(cov[1][1])),1)
dDc = int(np.sqrt(cov[2][2]))
plt.plot(x_vals, lambertfit(x_vals, *params[0:4]),\
label='PKC fit');
plt.scatter(x_data, y_data);
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.tight_layout()
myTable = PrettyTable(["-","N", "dN", "R","dR","Do/Dc (Gy)","dDc"])
myTable.add_row(["SE",A,dA,' ','',Do,\
dDo])
myTable.add_row(["PKC",round(params[0],0),dN,\
round(abs(params[1]),1),dR,abs(int(params[2])),dDc])
print(myTable)
plt.show()
```

```
+-----+------+-----+-----+-----+------------+-------+
|  -  |  N   |  dN |  R  |  dR | Do/Dc (Gy) |  dDc  |
+-----+------+-----+-----+-----+------------+-------+
|  SE | 17.0 | 0.7 |     |     |   712.3    | 133.5 |
| PKC | 18.0 | 1.0 | 0.4 | 0.3 |    1326    |  529  |
+-----+------+-----+-----+-----+------------+-------+
```

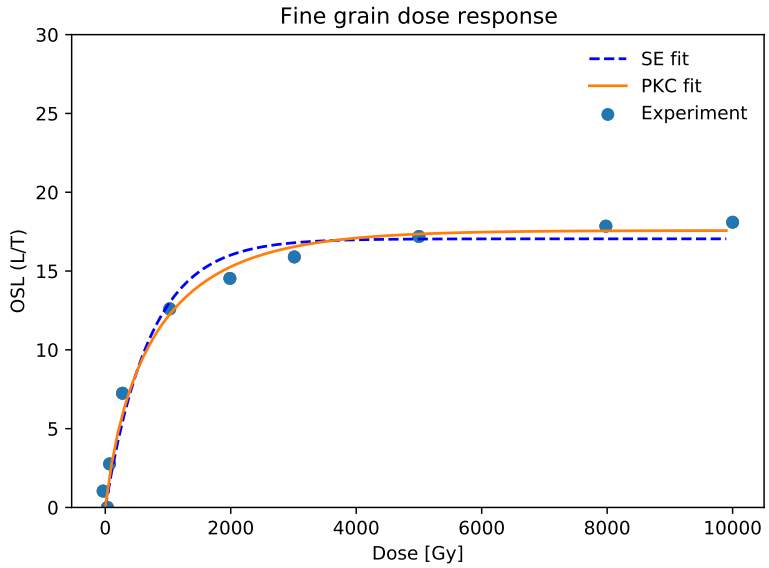

Fig. 15.1: Fitting dose response data from fine grain quartz, with a saturating exponential (SE) and the PKC equation. For more details see Pagonis et al. [1], original data from Timar-Gabor et al. [2]

Code 15.2: Fit of OSL coarse grain quartz data using PKC, SE and DSE

```
#Fit dose response data with Saturating Exponential
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import lambertw
import warnings
warnings.filterwarnings("ignore")
from scipy import optimize
from prettytable import PrettyTable
## fit to SE equation  ----
x_data= ([0, 3.55, 44.18, 258.718, 1051.62, 2044.98, 3003.94,
      5024.61, 7046.32, 9992.29])
y_data = ([0, 0.93512, 2.61108, 4.99104, 6.36704, 6.42148,
      6.43643, 6.46792, 6.77215, 6.97391])
# x_data =np.array([0,50.7117,100.534,152.135,204.626,272.242])
```

```
# y_data = np.array([0,33.144,42.205,43.1055,44.4157,43.7098])
plt.plot(x_data,y_data,"o");
def SE(x_data,A,Do):
    u=abs(A)*(1-np.exp(-x_data/np.abs(Do)))
    return u
def lambertfit(x_data,N,R,Dc):
    u=np.real(N*(1+lambertw((np.abs(R)-1)*np.exp(np.abs(R)-1-\
    x_data/np.abs(Dc)))/(1-np.abs(R))))
    u.astype(float)
    return u
def DSE(x_data,A,Do1,B,Do2):
    u=A*(1-np.exp(-x_data/np.abs(Do1)))+B*(1-\
    np.exp(-x_data/np.abs(Do2)))
    return u
init_vals=[1,1000,1,10000]                #DSE
params, cov = optimize.curve_fit(DSE,\
x_data, y_data,p0=init_vals)
[A,Do]=[round(params[x],1) for x in range(2)]
dA = round(np.sqrt(cov[0][0]),2)
dDo = int(np.sqrt(cov[1][1]))
x_vals=np.arange(0,1e4,1e1)
plt.plot(x_vals, DSE(x_vals, *params[0:4]),c="b",\
linestyle="dashed",label='DSE fit');
plt.xlim(0,3200);
[A1,Do1,A2,Do2]=[abs(round(params[x],1)) for x in range(4)]
dA1 = round(np.sqrt(cov[0][0]),2)
dDo1 = int(np.sqrt(cov[1][1]))
dA2 = round(np.sqrt(cov[2][2]),2)
dDo2 = int(np.sqrt(cov[3][3]))
myTable = PrettyTable([" ","A1", "dA1", "Do1","dDo1","A2",\
"dA2","D2","dD2"])
myTable.add_row(["DSE",A1,dA1,Do1,\
dDo1,A2,dA2,Do2,dDo2])
print(myTable)
init_vals=[20,20]                         #SE
params, cov = optimize.curve_fit(SE,\
x_data, y_data,p0=init_vals)
[A,Do]=[round(params[x],1) for x in range(2)]
dA = round(np.sqrt(cov[0][0]),2)
dDo = int(np.sqrt(cov[1][1]))
plt.plot(x_vals, SE(x_vals, *params[0:2]),c="r",\
linestyle="dotted",label='SE fit');
plt.scatter(x_data, y_data, label='Experiment');
leg = plt.legend()
```

```
leg.get_frame().set_linewidth(0.0)
plt.ylabel('OSL (L/T)');
plt.xlabel('Dose [Gy]');
plt.title('Coarse grain quartz');
dA = round(np.sqrt(cov[0][0]),1)
dDo = round(np.sqrt(cov[1][1]),1)
init_vals=[100.0,.1,100]
params, cov = optimize.curve_fit(lambertfit,\
x_data, y_data,p0=init_vals)
dN = round(np.sqrt(cov[0][0]),0)
dR = format(abs(np.sqrt(cov[1][1])),"10.1E")
dDc = int(np.sqrt(cov[2][2]))
plt.plot(x_vals, lambertfit(x_vals, *params[0:4]),\
label='PKC fit');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.tight_layout()
myTable2 = PrettyTable([" ","N", "dN", "Do (Gy)","dDc"])
myTable2.add_row(["SE",A,dA,Do,\
dDo])
print(myTable2)
myTable3=PrettyTable([" ","N","dN", "R","dR","Dc (Gy)","dDc"])
myTable3.add_row(["PKC",round(params[0],0),dN,\
format(abs(params[1]),"10.1E"),dR,abs(int(params[2])),dDc])
print(myTable3)
plt.show()
```

```
+-----+-----+------+-----+------+-----+------+-------+-----+
|     |  A1 | dA1  | Do1 | dDo1 |  A2 | dA2  |   D2  | dD2 |
+-----+-----+------+-----+------+-----+------+-------+-----+
| DSE | 1.8 | 0.34 | 5.5 |  2   | 4.8 | 0.34 | 240.3 |  39 |
+-----+-----+------+-----+------+-----+------+-------+-----+
+----+-----+-----+---------+------+
|    |  N  |  dN | Do (Gy) | dDc  |
+----+-----+-----+---------+------+
| SE | 6.5 | 0.2 |  123.4  | 26.5 |
+----+-----+-----+---------+------+
+-----+-----+-----+------------+------------+---------+-----+
|     |  N  |  dN |     R      |     dR     | Dc (Gy) | dDc |
+-----+-----+-----+------------+------------+---------+-----+
| PKC | 7.0 | 0.0 |    1.2E-06 |    2.5E-06 |   402   |  44 |
+-----+-----+-----+------------+------------+---------+-----+
```

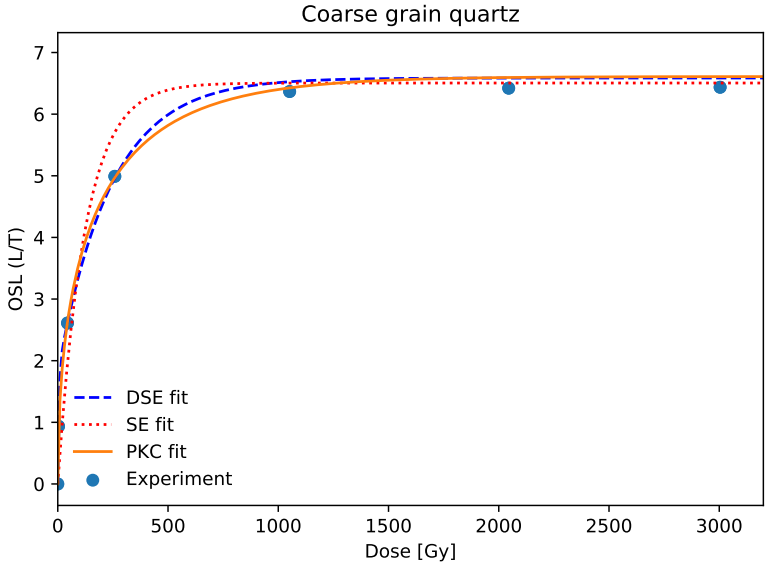

Fig. 15.2: Fit of experimental ESR dose response data using the Lambert Eq.(**??**). For more details see Pagonis et al. [1], original data from Timar-Gabor et al. [2]

Code 15.3: Using the PKC-S anion to fit superlinear data for aluminum oxide

```
# superlinear dose response  of anion deficient aluminum oxide
from scipy.special import lambertw
import numpy as np
import matplotlib.pyplot as plt
from scipy import optimize
import warnings
warnings.filterwarnings("ignore")
from prettytable import PrettyTable
## fit to PKC-S equation  ----
t = ([0.0537568,0.103385,0.156481,0.211929,0.260776,
0.321015,0.36483,0.414625,0.478926,0.535569,0.589476,
0.638899,0.824344,0.951985,1.18991,1.63688,3.19332])
y = ([6694.89,15592.6,24767.6,39360.5,52176.2,78075.6,
101463,131855,189548,227223,272406,376197,469268,
634929,792121,1.16105e6,1.6992e6])
```

```
x_data=np.array(t)
y_data=np.array(y)
def lambertfit(x_data,N,B,Dc,beta):
    u=np.real(lambertw((np.abs(B))*np.exp(np.abs(B)-(x_data/\
    np.abs(Dc))))/(np.abs(B)))
    u=N*(1-(u**beta))
    u.astype(float)
    return u
init_vals=[2e6,10,.01,.01]
params, cov = optimize.curve_fit(lambertfit,\
x_data, y_data,p0=init_vals)
dN = int(np.sqrt(cov[0][0]))
dB = round(np.sqrt(cov[1][1]),2)
dDc = round(np.sqrt(cov[2][2]),3)
dbeta = round(np.sqrt(cov[2][2]),2)
x_vals=np.arange(0.03,4,.01)
plt.subplot(1,2, 1);
plt.plot(x_vals, lambertfit(x_vals, *params[0:5]),c="b",\
label='Lambert fit');
plt.title('(a)');
plt.scatter(x_data, y_data, label='Experiment');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.ylabel('OSL (L/T)');
plt.xlabel('Dose [Gy]');
plt.text(1.8,.7e6,'Anion-deficient');
plt.text(1.8,.6e6,'Aluminum Oxide');
plt.text(1.8,.5e6,'Linear scale');
plt.subplot(1,2, 2);
plt.title('(b)');
plt.plot(np.log(x_vals), np.log(lambertfit(x_vals, \
*params[0:5])),c="b",label='Lambert fit');
plt.scatter(np.log(x_data), np.log(y_data), label='Experiment');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.ylabel('OSL (L/T)');
plt.xlabel('Dose [Gy]');
plt.text(-1,10,'Log-log scale');
plt.title('(b)');
plt.tight_layout()
myTable = PrettyTable(["N", "B","dB", "Dc (Gy)",\
"d(Dc","beta","dbeta"])
myTable.add_row([format(params[0],"10.1E"),\
round(np.abs(params[1]),2),dB,round(np.abs(params[2])\
```

```
, 2),dDc,round(np.abs(params[3]),5),dbeta])
print(myTable)
plt.show()
```

```
+------------+------+------+---------+-------+------+-------+
|     N      |  B   |  dB  | Dc (Gy) |  d(Dc | beta | dbeta |
+------------+------+------+---------+-------+------+-------+
|    2.1E+06 | 6.87 | 1.44 |   0.05  | 0.012 | 0.03 |  0.01 |
+------------+------+------+---------+-------+------+-------+
```

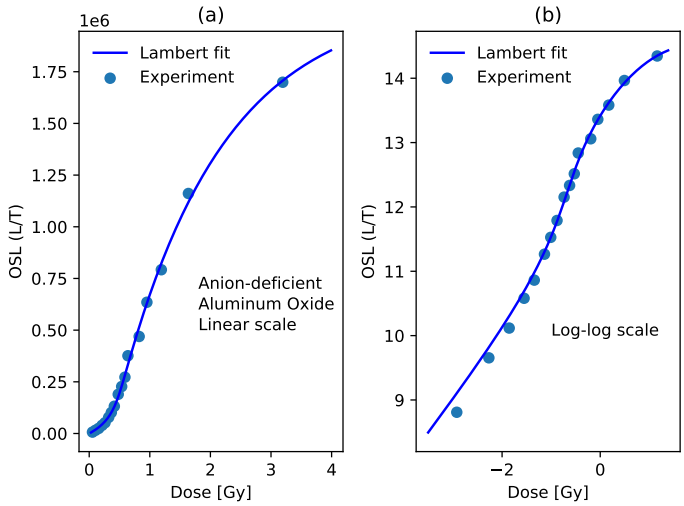

Fig. 15.3: TL dose response of an anion deficient aluminum oxide single crystal, which had low initial sensitivity to irradiation. The superlinear dose response is fitted with the PKC equation. For more details see Nikiforov et al. [3], see also the discussion in Pagonis et al. [4].

Code 15.4: Fit to Supralinearity index f(D) using PKC equation

```
# Fit to PKC-S equation for Supralinearity index f(D)
from scipy.special import lambertw
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from scipy import optimize
```

```
from prettytable import PrettyTable
## fit to PKC equation  ----
t = ([0.0811131, 0.171804,0.450923,0.857988,1.88341,
4.44069,8.44947,17.2683,40.7152,83.2104,176.246,
415.552,819.456,1674.74,3948.68,8983.32])
y = ([1.03326,0.983911,1.08074,1.10465,1.12844,1.28023,
1.37731,1.50481,1.76636,1.79021,1.45428, 0.813382,
0.428722,0.208669,0.0799713,0.0305689])
x_data=np.array(t)
y_data=np.array(y)
def lambertfit(x_data,N,B,Dc,beta):
    u=np.real(lambertw((np.abs(B))*np.exp(np.abs(B)-(x_data/\
np.abs(Dc))))/(np.abs(B)))
    u=N*(1-(u**beta))/x_data
    u.astype(float)
    return u
init_vals=[100,10,1,.01]
params, params_covariance = optimize.curve_fit(lambertfit,\
x_data, y_data,p0=init_vals)
x_vals=np.arange(0.1,1e4,.1)
plt.subplot(1,2, 1);
plt.plot(x_vals,lambertfit(x_vals,*params[0:5]),c="b",\
label='Lambert fit');
plt.scatter(x_data, y_data, label='Experiment');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.ylabel('Supralinearity index f(D)');
plt.title('(a)');
plt.xlabel('Dose [Gy]');
plt.text(4000,.65,'Probe A');
plt.text(4000,.5,'Linear scale');
plt.subplot(1,2, 2);
plt.title('(b)');
plt.plot(np.log(x_vals),lambertfit(x_vals,*params[0:5]),\
c="b",label='Lambert fit');
plt.scatter(np.log(x_data), y_data, label='Experiment');
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.ylabel('Supralinearity index f(D)');
plt.xlabel('ln(Dose) [KGy]');
plt.text(-2,.5,'Log-Linear scale');
plt.title('(a)');
plt.tight_layout()
res=lambertfit(x_data,*params[0:5])-y_data
```

```
FOM=round(100*np.sum(abs(res))/np.sum(y_data),2)
myTable = PrettyTable(["N", "B", "Dc (KGy)","beta","FOM (%)"])
myTable.add_row([round(params[0],2),format(np.abs(params[1]),\
"10.2E"),format(np.abs(params[2]), "10.2E"),round(np.abs(\
params[3]),4),FOM])
print(myTable)
plt.show()
```

```
+--------+------------+------------+--------+---------+
|   N    |     B      |  Dc (KGy)  |  beta  | FOM (%) |
+--------+------------+------------+--------+---------+
| 374.39 |   1.24E+00 |   5.17E+00 | 0.0332 |   3.68  |
+--------+------------+------------+--------+---------+
```

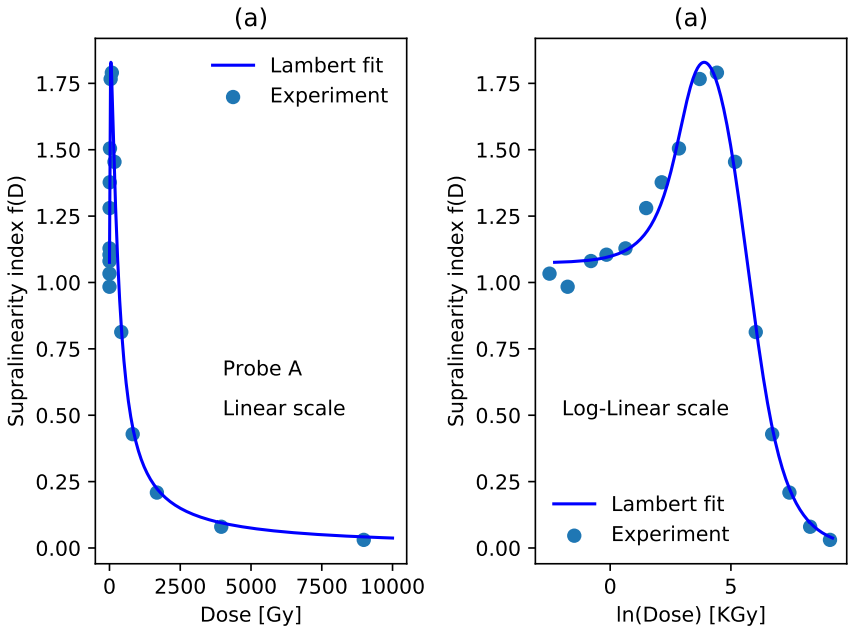

Fig. 15.4: Fit to the superlinear behavior of an Al_2O_3:C probe using the PKC-S equations. For more details of the analysis see Pagonis et al. [4], original experimental data from Edmund [5].

References

1. V. Pagonis, G. Kitis, R. Chen, A new analytical equation for the dose response of dosimetric materials, based on the Lambert W function, Journal of Luminescence 225 (2020) 117333. doi:https://doi.org/10.1016/j.jlumin.2020.117333.
   URL http://www.sciencedirect.com/science/article/pii/S0022231320305639
2. A. Timar-Gabor, A. Vasiliniuc, D. A. G. Vandenberghe, C. Cosma, A. G. Wintle, Investigations into the reliability of SAR-OSL equivalent doses obtained for quartz samples displaying dose response curves with more than one component, Radiation Measurements 47 (9) (2012) 740 – 745. doi:http://dx.doi.org/10.1016/j.radmeas.2011.12.001.
   URL http://www.sciencedirect.com/science/article/pii/S1350448711005671
3. S. V. Nikiforov, V. S. Kortov, M. G. Kazantseva, Simulation of the superlinearity of dose characteristics of thermoluminescence of anion-defective aluminum oxide, Physics of the Solid State 56 (3) (2014) 554–560.
   URL https://doi.org/10.1134/S1063783414030214
4. V. Pagonis, G. Kitis, R. Chen, Superlinearity revisited: A new analytical equation for the dose response of defects in solids, using the Lambert W function, Journal of Luminescence 227 (2020) 117553. doi:https://doi.org/10.1016/j.jlumin.2020.117553.
   URL http://www.sciencedirect.com/science/article/pii/S0022231320310449
5. J. Edmund, Effects of temperature and ionization density in medical luminescence dosimetry using al2o3:c (phd thesis, riso, denmark), Ph.D. thesis, Risø National Laboratory, riso-PhD-38(EN) (2007).