Chapter 10

INFRARED STIMULATED LUMINESCENCE SIGNALS: MODELS

Code 10.1: Example of summing partial TL curves in the EST model

```
#Example of summing partial TL curves in EST model
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
##### function to find distribution of distances ###
def partialTL(tmps,rprime):
    seff=s*np.exp(-(rho**(-1/3))*rprime)
    TLpartial=3*(rprime**2.0)*np.exp(-(rprime**3.0))*\
    (seff/hr)*np.exp(-E/(kB*(273+tmps)))*\
    np.exp(-seff*rprime*kB*((273+tmps)**2.0)/(hr*E)*\
    np.exp(-E/(kB*(273+tmps)))*(1-2*kB*(273+tmps)/E))
    return TLpartial*dr
kB,         s,        E ,    hr=\
8.617e-5,  2e15,  1.3 ,   1
t = np.linspace(0, 500, 500)
temps=hr*t
rho=1e-3                # rho-prime value
dr=0.1
rprimes=np.arange(0,2.2,dr)    # rprime=0-2.2
plt.subplot(1,2, 1);
for i in range(len(rprimes)):
    plt.plot(temps,[partialTL(x,rprimes[i]) for x in temps]);
plt.text(250,.0035,"Partial TL for ");
plt.text(250,.0032,"r'=0-2.2");
plt.ylabel('Partial TL signal [a.u.]');
plt.xlabel(r'Temperature [$^{o}$C]');
plt.title('(a)');
plt.subplot(1,2,2);
u=np.array([[partialTL(x,rprimes[i]) for x in temps]\
for i in range(len(rprimes))])
plt.plot(temps,sum(u),c='b');
plt.text(300,.01, 'Sum of');
plt.text(300,.009, 'partial');
plt.text(300,.008, 'TL curves');
plt.ylabel('Remnant TL signal [a.u.]');
plt.xlabel(r'Temperature [$^{o}$C]');
plt.title('(b)');
plt.tight_layout()
plt.show()
```

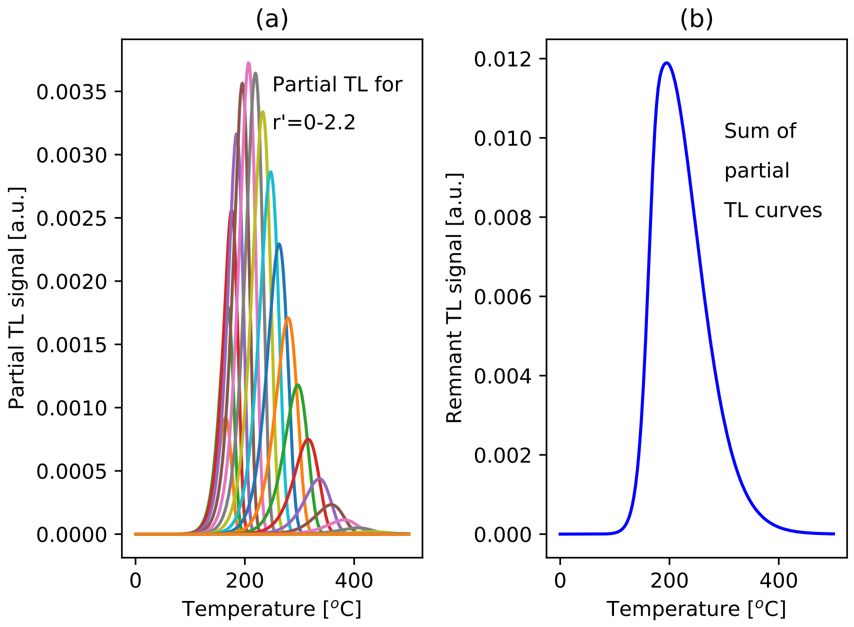

Fig. 10.1: (a) Simulation of the TL signal from an unfaded sample. (a) The 22 partial first order TL glow curves which correspond to different distances r' in the EST model; (b) The total TL signal from the sample is calculated as the sum of the partial TL glow curves in (a). For more details on this type of simulation, see Pagonis et al. [1] and Chap.4 of this book.

Code 10.2: Example of summing partial CW curves in the EST model

```
#Example of summing partial CW-IRSL curves in the EST model
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
##### function to find distribution of distances ###
def partialCWIRSL(tims,rprime):
    seff=A*np.exp(-(rho**(-1/3))*rprime)
    CWIRSLpartial=3*(rprime**2.0)*np.exp(-(rprime**3.0))*seff*\
    np.exp(-seff*rprime*tims)
    return CWIRSLpartial*dr
A, P=5, 400
t = np.linspace(0, P, P)
rho=1e-3                # rho-prime value
dr=0.1
rprimes=np.arange(0,2.2,dr)    # rprime=0-2.2
plt.subplot(1,2, 1);
for i in range(len(rprimes)):
    plt.plot(t,[partialCWIRSL(x,rprimes[i]) for x in t]);
plt.text(120,.0035,"Partial CW-IRSL");
plt.text(120,.003,"for r'=0-2.2");
plt.ylabel('Partial CW-IRSL signal [a.u.]');
plt.xlabel(r'Time [s]');
plt.title('(a)');
plt.text(120,.005,'EST model');
plt.subplot(1,2,2);
u=np.array([[partialCWIRSL(x,rprimes[i]) for x in t]\
for i in range(len(rprimes))])
plt.plot(t,sum(u),c='b');
plt.text(100,.015, 'Sum of');
plt.text(100,.013, 'partial');
plt.text(100,.011, 'CW-IRSL curves');
plt.ylabel('CW-IRSL signal [a.u.]');
plt.xlabel(r'Time [s]');
plt.title('(b)');
plt.tight_layout()
plt.show()
```

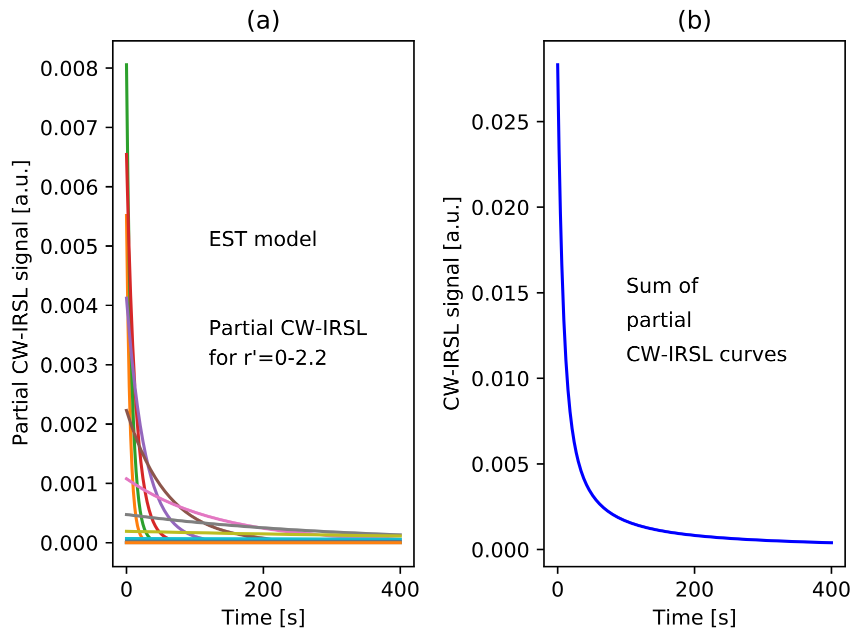

Fig. 10.2: (a) Simulation of the CW-IRSL signal from an unfaded sample. (a) The 22 partial first order CW-IRSL curves which correspond to different distances r'; (b) The total CW-IRSL signal from the sample is calculated as the sum of the partial curves in (a). For more details on this type of simulation, see Pagonis et al. [1]

Code 10.3: Example of summing partial LM curves in the EST model

```
#Example of summing partial LM-IRSL curves in the EST model
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
##### function to partial CW curves ###
def partialLMIRSL(tims,rprime):
    seff=A*np.exp(-(rho**(-1/3))*rprime)
    LMIRSLpartial=3*(rprime**2.0)*np.exp(-(rprime**3.0))*seff*\
    tims*np.exp(-seff*rprime*tims**2/P)
    return LMIRSLpartial*dr
A, P=5, 200
t = np.linspace(0, P, P)
rho=0.013                # rho-prime value
dr=0.1
rprimes=np.arange(0,2.2,dr)    # rprime=0-2.2
plt.subplot(1,2, 1);
for i in range(len(rprimes)):
    plt.plot(t,[partialLMIRSL(x,rprimes[i]) for x in t]);
plt.text(80,.3,"Partial LM-IRSL");
plt.text(80,.25,"for r'=0-2.2");
plt.ylabel('Partial LM-IRSL signal [a.u.]');
plt.xlabel(r'Time [s]');
plt.title('(a)');
plt.text(80,.2,'EST model');
plt.subplot(1,2,2);
u=np.array([[partialLMIRSL(x,rprimes[i]) for x in t]\
for i in range(len(rprimes))])
plt.plot(t,sum(u),c='b');
plt.text(80,2, 'Sum of');
plt.text(80,1.75, 'partial');
plt.text(80,1.5, 'LM-IRSL curves');
plt.ylabel('LM-IRSL signal [a.u.]');
plt.xlabel(r'Time [s]');
plt.title('(b)');
plt.tight_layout()
plt.show()
```

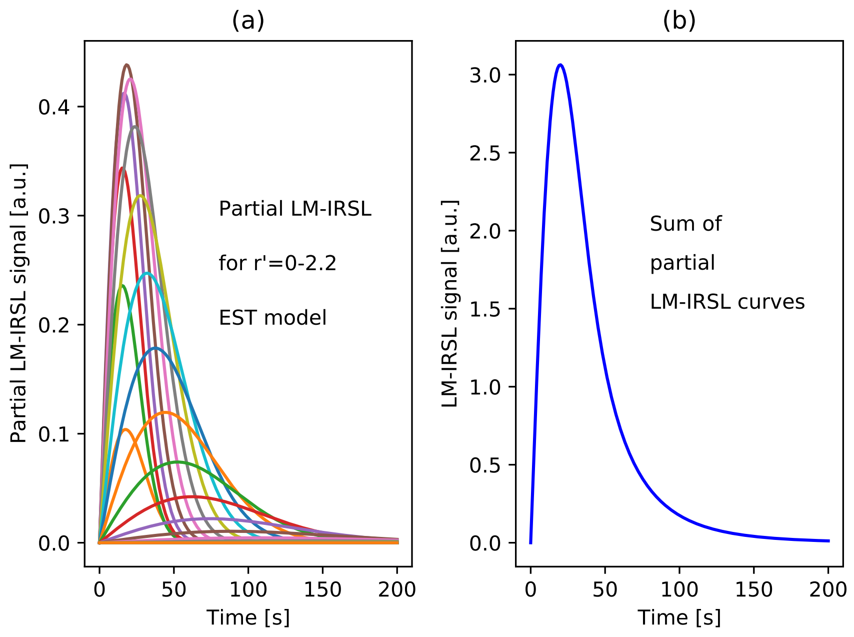

Fig. 10.3: Simulation of the LM-IRSL signal from an unfaded sample. (a) The 22 partial first order LM-IRSL curves which correspond to different distances r'; (b) The total LM-IRSL signal from the sample is calculated as the sum of the partial LM-IRSL curves in (a). For more details on this type of simulation, see Pagonis et al. [1]

Code 10.4: Plots of the KP-CW equation for CW-IRSL signals

```
#plot KP-CW equation for localized processes
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
# pCW in this code represents the effective frequency s'
def KPCW(rho,pCW):
    CW=np.exp (-rho*(np.log(1 + pCW*t))\
    ** 3.0)*(np.log(1+pCW*t)**2.0)/(1+t*pCW)
    plt.plot(t,CW/max(CW),symbs[j-1], linewidth=2,
    label=labls[j-1]);
t = np.linspace(1, 100, 100)
pCW=3
rhos=[1e-3,5e-3,1e-2]
labls=[r'$\rho$'+"'="+str(x) for x in rhos]
symbs=['+-','^-','o-']
plt.subplot(1,2,1);
for j in range(1,4):
    KPCW(rhos[j-1],pCW)
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.xlabel('Time [s]');
plt.text(45,.6,'KP-CW equation');
plt.ylabel('CW [a.u]');
plt.title('(a)');
plt.subplot(1,2,2);
pCW=[3,5,7]
labls=["s'="+str(x)+r' s$^-1$' for x in pCW]
for j in range(1,4):
    KPCW(0.01,pCW[j-1])
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.xlabel('Time [s]');
plt.ylabel('CW [a.u]');
plt.title('(b)');
plt.tight_layout()
plt.show()
```

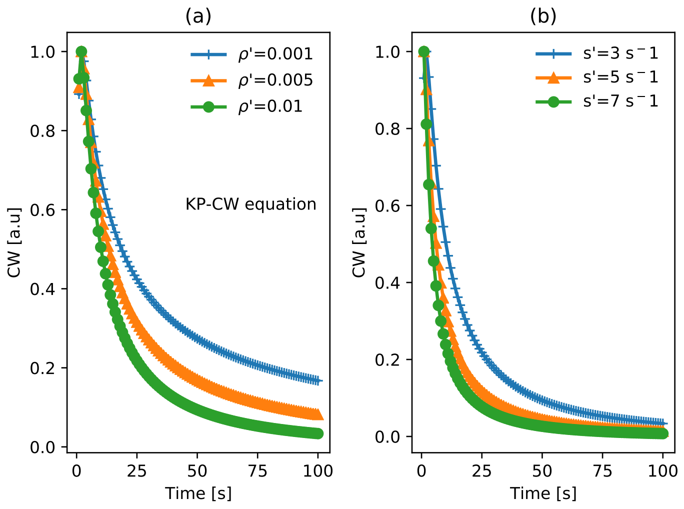

Fig. 10.4: (a) Plot of the KP-CW Eq.(**??**) for 3 different values of the dimensionless density ρ' =0.001, 0.005, 0.01. As the value of ρ' increases, the CW-IRSL curve decreases faster; (b) Plot of the KP-CW equation for different effective frequency factors s'=3, 5, 7 s^{-1} during the CW-IRSL experiment.

Code 10.5: Plots of the KP-LM equation for LM-IRSL signals

```
#plot KP-LM equation for localized processes
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
# pLM in this code represents the effective frequency s'
def KPLM(rho,pLM):
    LM=np.exp (-rho*(np.log(1 + pLM*(t**2/(2*P))))** 3.0)*\
    (np.log(1+pLM*(t**2/(2*P)))**2.0)/(1+(t**2/(2*P))*pLM)
    plt.plot(t,LM/max(LM),symbs[j-1], linewidth=2,
    label=labls[j-1]);
t = np.linspace(1, 100, 100)
P=max(t)
pLM=3
rhos=[1e-3,5e-3,1e-2]
labls=[r'$\rho$'+"'="+str(x) for x in rhos]
```

```
symbs=['+-','^-','o-']
plt.subplot(1,2,1);
for j in range(1,4):
    KPLM(rhos[j-1],pLM)
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.xlabel('Time [s]');
plt.text(50,.75,'KP-LM equation');
plt.ylabel('LM-IRSL [a.u]');
plt.title('(a)');
plt.subplot(1,2,2);
pLM=[3,5,7]
labls=["s'="+str(x)+r' s$^-1$' for x in pLM]
for j in range(1,4):
    KPLM(0.01,pLM[j-1])
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.xlabel('Time [s]');
plt.ylabel('LM-IRSL [a.u]');
plt.title('(b)');
plt.tight_layout()
plt.show()
```

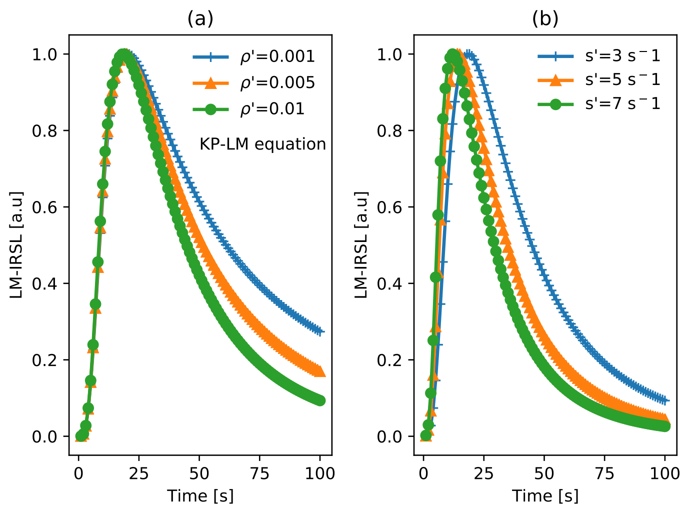

Fig. 10.5: (a) Plot of the KP-LM Eq.(**??**) for different $\rho' = 0.001, 0.005, 0.01$ and constant $s'=3$ s^{-1} during the CW-IRSL experiment. (b) Plots of the KP-LM equation for different values of the effective frequency factor $s' = 3$, 5, 7 s^{-1}.

Code 10.6: Dependence of the stretched exponential on model parameters

```
#Fitting CW-IRSL signal with stretched exponential
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
# pLM in this code represents the effective frequency s'
def KPLM(tau,beta):
    SE=A*np.exp(-(t/tau)**beta)
    plt.plot(t,SE/max(SE),symbs[j-1], linewidth=2,
    label=labls[j-1]);
t = np.linspace(1, 100, 30)
A=1
taus=[5,10,15]
beta=0.5
labls=[r'$\tau$'+"="+str(x)+" s" for x in taus]
symbs=['+-','^-','o-']
plt.subplot(1,2,1);
for j in range(1,4):
    KPLM(taus[j-1],beta)
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.xlabel('Time [s]');
plt.text(50,.6,'Stretched');
plt.text(50,.52,'Exponential');
plt.ylabel('LM-IRSL [a.u]');
plt.title('(a)');
plt.subplot(1,2,2);
betas=[.3,.5,.7]
tau=15
labls=[r"$\beta$="+str(x) for x in betas]
for j in range(1,4):
    KPLM(tau,betas[j-1])
leg = plt.legend()
leg.get_frame().set_linewidth(0.0)
plt.xlabel('Time [s]');
plt.ylabel('LM-IRSL [a.u]');
plt.title('(b)');
plt.tight_layout()
plt.show()
```

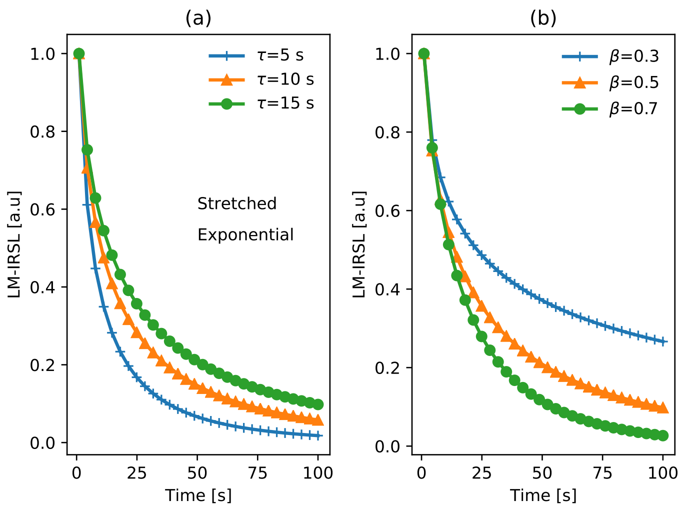

Fig. 10.6: (a) Plot of the stretched exponential function Eq.(**??**) for (a) different decay times τ =5, 10, 15 s and (b) different stretched exponential coefficients β =0.3, 0.5, 0.7.

References

1. V. Pagonis, J. Friedrich, M. Discher, A. Müller-Kirschbaum, V. Schlosser, S. Kreutzer, R. Chen, C. Schmidt, Excited state luminescence signals from a random distribution of defects: A new Monte Carlo simulation approach for feldspar, Journal of Luminescence 207 (2019) 266–272. `doi:https://doi.org/10.1016/j.jlumin.2018.11.024`. URL `http://www.sciencedirect.com/science/article/pii/S0022231318317368`